STOCHASTIC OPTIMIZATION FOR MULTI-AGENT STATISTICAL LEARNING AND CONTROL

Alec Koppel

A DISSERTATION

in

Electrical and Systems Engineering

Presented to the Faculties of the University of Pennsylvania
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
2017

Supervisor of Dissertation

Alejandro Ribeiro, Rosenbluth Associate Professor of Electrical and Systems Engineering

Graduate Group Chairperson

Alejandro Ribeiro, Rosenbluth Associate Professor of Electrical and Systems Engineering

Dissertation Committee
Vijay Kumar, Nemirovsky Family Dean of School of Engineering and Applied Science at the University of Pennsylvania
Brian M. Sadler, Principal Scientific Lead and Fellow of the U.S. Army Research Laboratory
Jonathan Fink, Research Scientist, U.S. Army Research Laboratory

STOCHASTIC OPTIMIZATION FOR MULTI-AGENT
STATISTICAL LEARNING AND CONTROL

To my mother Leslie and grandfather Seymour.

Acknowledgments

My father's hero Richard Feyman once said "I would rather have questions that can't be answered than answers that can't be questioned." As I think back over the course of my doctorate, one of the key characteristics that has defined my evolution from apprentice to principle investigator has been the honing of my ability to ask good questions. The development of this skill has transpired alongside changes to my personality and perspective. This is a tribute to the people who played a crucial role in this evolution.

I am deeply grateful for the opportunity I've had to work with my doctoral advisor Alejandro Ribeiro over the years. When I began graduate school, $80-90\%$ of what I would say is wrong, and despite familiarity with mathematics, I did not speak the language, nor did I have any clue how to ask questions. Through his tireless efforts, dedication, and meticulous mentorship style, I truly feel transformed into a literate member of our research community. I hope I can pay forward his efforts on my behalf to future generations of researchers.

I would like to further thank Drs. Jonathan Fink and Brian M. Sadler of the U.S. Army Research Laboratory as well as Dean Vijay Kumar for agreeing to serve on my doctoral committee. I am grateful to have had countless constructive conversations with Dr. Sadler over the years deepening the theoretical foundations of my work. On the other hand, discussions with Dean Kumar and Dr. Fink have helped me ensure the questions I ask are of practical importance to intelligent systems and robots, rather than wandering too far down the rabbit hole of learning theory.

Over the course of my Phd, I've been fortunate to make some meaningful friendships that have strengthened my research purpose. Specifically, I benefited greatly from working closely with Drs. Garrett Warnell and Ethan Stump of the U.S. Army Research Laboratory who shifted my focus from pure optimization problems to those in the intersection of optimization and statistical learning. Parts II and III of this thesis would not have been possible without their contributions. I'd also like to acknowledge Professor Daniel D. Lee of UPenn for taking the time out over the past year to listen and exchange ideas regarding learning in robotics. These discussions were invaluable for embarking upon the next great chapter of my research career, which begins with Chapter 8 of this thesis.

I would also like to extend a deep sense of gratitude to the friends in the laboratory to which I belonged over the past five years. I would especially like to thank Aryan Mokhtari, with whom I've collaborated on several side projects that were crucial to my understanding of optimization, and have had many insightful discussions on personal and professional goals. I will really miss being labmates but I know we'll continue being friends for a lifetime. I'd also like to mention other friends in graduate school with me, whose presence has made the whole process much more enjoyable, and includes but is not limited to: Felicia Jakubiec, Ceyhun Eksin, Santiago Segarra, Santiago Paternain, Weiyu Huang, Mahyar Fazlyab, Luiz Chamon, Behnaz Arzani, Ekaterina Tolstaya, Konstantinos Gatsis, Tarik Tosun, and Erdem Varol. Your friendships during graduate school really made the experience whole.

Lastly, I'd like to thank my family for the role they've had in continually pushing me forward over the years. Their role in the story of my graduate school arc has little to do with my evolving professional perspective, but everything to do with shaping my heart and mind, and I would not have achieved this feat without their love and support. In particular, my mother and grandfather, whose continual belief in me and talking me through tough times made completing graduate school possible.

Alec Koppel, Philadelphia, June 2017

ABSTRACT

STOCHASTIC OPTIMIZATION FOR MULTI-AGENT
STATISTICAL LEARNING AND CONTROL

Alec Koppel
Alejandro Ribeiro

The goal of this thesis is to develop a mathematical framework for optimal, accurate, and affordable complexity statistical learning among networks of autonomous agents. We begin by noting the connection between statistical inference and stochastic programming, and consider extensions of this setup to settings in which a network of agents each observes a local data stream and would like to make decisions that are good with respect to information aggregated across the entire network. There is an open-ended degree of freedom in this problem formulation, however: the selection of the estimator function class which defines the feasible set of the stochastic program. Our central contribution is the design of stochastic optimization tools in reproducing kernel Hilbert spaces that yield optimal, accurate, and affordable complexity statistical learning for a multi-agent network. To obtain this result, we first explore the relative merits and drawbacks of different function class selections.

In **Part I**, we consider multi-agent expected risk minimization this problem setting for the case that each agent seems to learn a common globally optimal generalized linear models (GLMs) by developing a stochastic variant of Arrow-Hurwicz primal-dual method. We establish convergence to the primal-dual optimal pair when either consensus or "proximity" constraints encode the fact that we want all agents' to agree, or nearby agents to make decisions that are close to one another. Empirically, we observe that these convergence results are substantiated but that convergence may not translate into statistical accuracy. More broadly, optimality within a given estimator function class is *not* the same as one that makes minimal inference errors.

The optimality-accuracy tradeoff of GLMs motivates subsequent efforts to learn more sophisticated estimators based upon learned feature encodings of the data that is fed into the statistical model. The specific tool we turn to in **Part II** is dictionary learning, where we optimize both over regression weights and an encoding of the data, which yields a non-convex problem. We investigate the use of stochastic methods for online task-driven dictionary learning, and obtain promising performance for the task of a ground robot learning to anticipate control uncertainty based on its past experience. Heartened by this implementation, we then consider extensions of this framework for a multi-agent network to each learn globally optimal task-driven dictionaries based on stochastic primal-dual methods. However, it is here the non-convexity of the optimization problem causes problems: stringent

conditions on stochastic errors and the duality gap limit the applicability of the convergence guarantees, and impractically small learning rates are required for convergence in practice.

Thus, we seek to learn nonlinear statistical models while preserving convexity, which is possible through kernel methods (**Part III**). However, the increased descriptive power of nonparametric estimation comes at the cost of infinite complexity. Thus, we develop a stochastic approximation algorithm in reproducing kernel Hilbert spaces (RKHS) that ameliorates this complexity issue while preserving optimality: we combine the functional generalization of stochastic gradient method (FSGD) with greedily constructed low-dimensional subspace projections based on matching pursuit. We establish that the proposed method yields a controllable trade-off between optimality and memory, and yields highly accurate parsimonious statistical models in practice. Then, we develop a multi-agent extension of this method by proposing a new node-separable penalty function and applying FSGD together with low-dimensional subspace projections. This extension allows a network of autonomous agents to learn a memory-efficient approximation to the globally optimal regression function based only on their local data stream and message passing with neighbors. In practice, we observe agents are able to stably learn highly accurate and memory-efficient nonlinear statistical models from streaming data.

From here, we shift focus to a more challenging class of problems, motivated by the fact that true learning is not just revising predictions based upon data but augmenting behavior over time based on temporal incentives. This goal may be described by Markov Decision Processes (MDPs): at each point, an agent is in some state of the world, takes an action and then receives a reward while randomly transitioning to a new state. The goal of the agent is to select the action sequence to maximize its long-term sum of rewards, but determining how to select this action sequence when both the state and action spaces are infinite has eluded researchers for decades. As a precursor to this feat, we consider the problem of policy evaluation in infinite MDPs, in which we seek to determine the long-term sum of rewards when starting in a given state when actions are chosen according to a fixed distribution called a policy. We reformulate this problem as a RKHS-valued compositional stochastic program and we develop a functional extension of stochastic quasi-gradient algorithm operating in tandem with the greedy subspace projections mentioned above. We prove convergence with probability 1 to the Bellman fixed point restricted to this function class, and we observe a state of the art trade off in memory versus Bellman error for the proposed method on the Mountain Car driving task, which bodes well for incorporating policy evaluation into more sophisticated, provably stable reinforcement learning techniques, and in time, developing optimal collaborative multi-agent learning-based control systems.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, decreasing computing costs alongside pervasive data availability has fueled the transition of learning systems from imagination to just another feature in technologies such as smart devices [74], autonomous automobiles [25], medical imaging [210], and even robot assistants [124]. However, there is a large gap between the adoption of off-the-shelf learning methods which require all observations to be available at once [1] and the design learning systems that can may *continually adapt* to new evidence. Our focus is on developing a principled framework for convergent, accurate, and lightweight supervised learning in collaborative decentralized systems such as robotic [11] or computer networks [3].

From a scientific perspective, supervised learning is statistical inference [66]. We are given a batch of data N data points $\{\mathbf{x}_n, y_n\}_{n=1}^N \subset \mathbb{R}^p \times \mathbb{R}$ which are samples of a pair of random variables $(\mathbf{x}, y)$, and want to devise a mathematical model (estimator) $\hat{y}_n = f(\mathbf{x}_n)$ of an input-output relationship among it. For example, consider a batch of U.S. census data that contains demographic and salary information. Given the demographic profile of a new individual, we may try to predict her income. Alternatively, given a video feed, some of whose frames contain an object of interest such as a person's face, we may try to identify whether that person's face appears in a future video. The aforementioned examples correspond to regression or classification, respectively, and more generally are referred to as estimation [204].

To make inferences that adhere closely to the ground-truth, however, in addition to designing an estimator, one must define a merit of its quality, which, practically speaking, is whether the estimator predicts correctly $\mathbb{1}\{\hat{y}_n = y_n\}$ [18], also called its statistical accuracy. Thus, in principle, we'd like to find an estimator which minimizes the number of mistakes made averaged over all possible data:

$$f^\star = \operatorname*{argmin}_{f \in \mathscr{F}} \mathbb{E}_{\mathbf{x},y}[\mathbb{1}\{f(\mathbf{x}) \neq y\}] \tag{1.1}$$

Here $\mathbb{1}\{E\}$ denotes the indicator function of an event E, which in (1.1) is 1 if the estimator makes a mistake, and null otherwise. Optimizing inference accuracy (1.1) (Baye's risk) directly is an intractable integer program, and thus we replace the indicator by a convex surrogate loss function $\ell(f(\mathbf{x}), y)$ which is small when $f(\mathbf{x})$ and y are close and large when they're far. One example is a probabilistic model of the odds ratio of whether or not an outcome will happen [133] in the case of the logistic model of classification. Then, we seek to optimize the estimator $f(\mathbf{x}_n)$ that defines the statistical model with respect to this surrogate loss $\ell(f(\mathbf{x}_n), y_n)$ averaged over all data points:

$$\tilde{f}^* = \underset{f \in \mathscr{F}}{\operatorname{argmin}} \, \mathbb{E}_{\mathbf{x},y}[\ell(f(\mathbf{x}), y)] \tag{1.2}$$

Moreover, $\mathscr{F}$ denotes some function class to which the estimator f belongs – we defer details of the estimator function class for the moment, but clearly a good selection is one which makes the optimization problem solvable, admits a moderate complexity solution, and closes the gap between optimizing statistical (1.1) and optimizing our surrogate loss (1.2). The formulation (1.2), of which maximum likelihood estimation (MLE) is a special case [163], is referred to as the *general learning setting* or expected risk minimization [202], and establishes a mathematical foundation linking optimization and statistics.

Our goal is to develop optimal, moderate complexity solutions to (1.2) that are also statistically accurate, i.e., to obtain solutions to (1.2) which are not far from the one that minimizes the error rate (1.1). We focus on the case where the number of training examples N is infinite (which necessitates the expectation in (1.1)-(1.2)), or independent samples $(\mathbf{x}_n, y_n)$ are arriving in a streaming fashion, and further when data is scattered across an interconnected network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of autonomous agents. We note that currently, there are little to no methods which meet these specifications, even for the centralized setting. Thus, we propose investigating this problem space through different selections of $\mathscr{F}$ in order to find an appropriate choice. An overview of our preliminary findings is given in Figure 1.1, and the first half of the following subsection. These results then motivate the main result of this thesis developed in Part III.

Motivation for Decentralized Methods There are two technological settings in which decentralized information processing is important. The first is industrial-scale machine learning [32], in which billions of training examples are available at a centralized location, and one would like to find an optimal statistical model in terms of all data. It is beneficial in terms of the number of samples one may process per iterative optimization step to decentralize/parallelize data processing in order to obtain computational speedup [158].

The later setting in which decentralized learning may be beneficial is networked autonomous systems such as sensor or robotic networks. In this setting, decentralized processing is only justified when the communication and computational cost of centralized

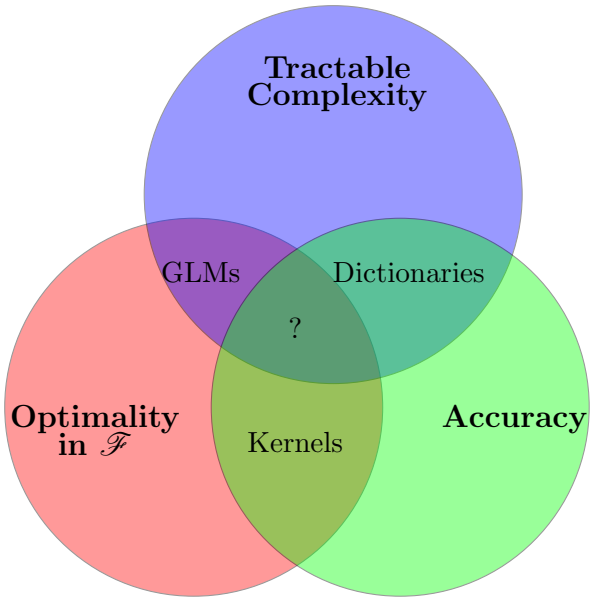


Figure 1.1: A summary of existing statistical learning tools in terms of different choices of estimator function class $\mathscr{F}$: generalized linear models (GLMs) are low complexity and yield solutions whose convergence follows from classical stochastic approximation methods. **Part I** develops a convergent framework for multi-agent statistical learning with GLMs but their lack of statistical accuracy on empirically important problems motivates the pursuit of dictionary methods in **Part II**. Here we observe that in centralized settings dictionaries can obtain state of the art statistical accuracy, but their convergence is limited by the non-convexity defined by their training, which precludes the ability to find the optimal dictionary representation and causes numerical instabilities that must be surmounted by heuristics. These drawbacks motivate the central question of this thesis regarding how to learn nonlinear statistical models while preserving convexity. We address this question by noting that kernel methods yield a framework that meet these criteria, but at the cost of defining a convex optimization problem over an infinite dimensional function space. Thus we are faced with the question of how to ameliorate this complexity issue while preserving the optimality and accuracy properties of nonparametric methods.

aggregation of data ("flooding") exceeds the cost of local-only computations and communications with neighboring nodes. We note that we are in this regime when the data available to each node is consistently in flux, i.e., *streaming data* settings: in this situation, it would be necessary to do centralized aggregation of data at each time slot, whose computational and communication cost exceeds a decentralized online processing strategy. This setup is the focus of the work pursued in this thesis.

1.1 Summary of Initial Findings

Part I of this thesis develops distributed stochastic optimization tools for (1.2) for the case that the estimators of each agent $i \in \mathcal{V}$ in a multi-agent network are generalized

linear models (GLMs): $f_i(\mathbf{x}_i) = \mathbf{w}_i^T \mathbf{x}_i$ for some $\mathbf{w}_i \in \mathbb{R}^p$ and $\mathscr{F} = \mathbb{R}^p$. In **Chapter 2**, we first address the case that the hypothesis that each agent's data is sampled from a common distribution, which appeared as [93]. When the random pairs $(\mathbf{x}_i, y_i)$ have a common distribution for each $i \in \mathcal{V}$ then convexity implies that at optimality, each agents' statistical models coincide: $\mathbf{w}_i = \mathbf{w}_j$ for each (i, j). Under this hypothesis, we are in the province of online consensus optimization [136]. We develop a decentralized stochastic optimization tool to solve multi-agent extensions of (1.2) exactly based on primal-dual method [9]. Then, in **Chapter 3**, we draw upon the lessons learned for the case that each agent observes samples from a common distribution to develop tools which are applicable to more general settings where agents seek to collaborate but not coincide, so as to retain the distinct perspective induced by their possibly unique local data stream [97]. Part I primarily focuses on addressing the issues of how to learn globally optimal statistical model parameters which are of low complexity, while ignoring whether the space of GLMs is sufficiently rich to attain practically useful inference performance, i.e., to come close to (1.1).

John Tukey once said, "far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise" [201]. In Part I, we solve the problem (1.2) when $(\mathbf{x}_i, y_i)$ have a common distribution for each $i \in \mathcal{V}$ and each agent learns a GLM exactly, but observe that this class of estimators is not rich enough to solve inference tasks that arise in practice, such as computer network security (Section 2.5). This suggests that solving an optimization problem *does not* necessarily translate to satisfactory statistical accuracy, unless the optimization problem hews closely to the actual inference problem (1.1). The result of this research thread is that multi-agent online learning with linear statistical models belongs firmly to the intersection of methods which yield optimality within their function class, are moderate complexity, but are not statistically accurate (see Figure 1.1).

An active research question is how to specify the function class $\mathscr{F}$ to which estimator the $f(\mathbf{x}_n)$ belongs so as to preserve computational tractability of (1.2) but come within striking distance of $f^\star$ that solves (1.1), the one which directly optimizes statistical accuracy [18]. Typically, increasing the complexity of function space $\mathscr{F}$ yields more accurate estimates at the price of additional computational cost [28]. Numerous approaches exist to bridge the gap between $\tilde{f}^*$ and $f^\star$: neural networks [72], nonparametric methods [192], and those which exploit specialized knowledge of the data domain, called feature extraction/signal representation (unsupervised learning) techniques. This later approach addresses the discrepancy between (1.1) and (1.2) by transforming the data vectors $\mathbf{x}_n$ into a form which may be more amenable to statistical inference through dimensionality reduction [84], transformation into a Fourier domain [145], or its multi-resolution extensions [122]. Alternatively, one may seek to learn a signal representation that is specifically tailored to the inference

task one is attempting to solve [12], which is referred to as dictionary learning.

In **Part II** of this thesis, we investigate the use of dictionary learning methods for solving (1.2) when the estimators are a generalization of GLMs $\hat{y} = \alpha(\mathbf{x}, \mathbf{D})^T\mathbf{w}$ that allow for a feature encoding $\alpha = \alpha(\mathbf{x}, \mathbf{D}) \in \mathbb{R}^k$ of the data $\mathbf{x}$ to be learned alongside the statistical model parameters $\mathbf{w} \in \mathbb{R}^k$ (**Chapter 4**). The feature encoding is obtained by defining a separate "representation" loss $s(\boldsymbol{\alpha}_t, \mathbf{D}; \mathbf{x}_t)$ that depends on the proximity between $\mathbf{D}\boldsymbol{\alpha}_t$ and the data point $\mathbf{x}_t$ and may incentivize, for instance, sparsity. Here $\mathbf{D}$ is a matrix of size $p \times k$, where k is the number of dictionary elements (or atoms) which is fixed, but chosen by us. The feature encoding is computed as the following projection [2]

$$\boldsymbol{\alpha}(\mathbf{D}; \mathbf{x}_t) := \underset{\boldsymbol{\alpha}_t \in \mathbb{R}^k}{\operatorname{argmin}}\, s(\boldsymbol{\alpha}_t, \mathbf{D}; \mathbf{x}_t)\ . \tag{1.3}$$

In standard dictionary learning [58], we seek to optimize (1.3) with respect to $\mathbf{D}$, but in task-driven dictionary learning, we instead use the feature encoding directly for statistical inference. Therefore, our statistical loss[1] for this setting becomes

$$(\mathbf{D}^*, \mathbf{w}^*) := \underset{\mathbf{D} \in \mathcal{D}, \mathbf{w} \in \mathcal{W}}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{x},\mathbf{y}}\Big[\ell\big(\boldsymbol{\alpha}(\mathbf{D}, \mathbf{x})^T\mathbf{w}, \mathbf{y}\big)\Big]. \tag{1.4}$$

The estimator $\boldsymbol{\alpha}(\mathbf{D}, \mathbf{x})^T\mathbf{w}$ has more descriptive power than a GLM since we are allowed to search over the space of signal representations in the form of dictionary matrices while simultaneously searching for regression weights. However, this increased descriptive power comes at the cost of making the optimization problem (1.4) non-convex, a difficulty shared with neural networks. Nonetheless, for the centralized setting, by optimizing over both the data representation and the statistical model parameters using stochastic gradient descent, we obtain superior performance on a truly challenging estimation problem: that of an autonomous robot attempting to forecast where it will experience more steering mistakes based on its past experience (Chapter 4.2), which appeared as [92]. This is in contrast to the lackluster performance of GLMs on the real problem of attacker detection in computer networks in Section 2.5.

Heartened by this successful application, in **Chapter 5**, we develop a mathematical framework for online task-driven dictionary learning in multi-agent systems by building upon the computational tools developed in Chapter 2. In particular, we again associate each agent i in a digraph $\mathcal{G}$ with a unique sequence of training examples $(\mathbf{x}_{i,n}, y_{i,n})$ and

[1] In Chapter 4, we use the notation $\ell\big(\mathbf{D}, \mathbf{w}; \mathbf{x}, \mathbf{y}\big) = \ell\big(\boldsymbol{\alpha}(\mathbf{D}, \mathbf{x})^T\mathbf{w}, \mathbf{y}\big)$ to emphasize that the dictionary $\mathbf{D}$ and model parameters $\mathbf{w}$ are our choice, whereas the data, mathematically represented as a pair of random variables $(\mathbf{x}, \mathbf{y})$, are not.

consider the decentralized stochastic program

$$\{\mathbf{D}_i^*, \mathbf{w}_i^*\}_{i=1}^V := \underset{\mathbf{D}_i \in \mathcal{D}, \mathbf{w}_i \in \mathcal{W}}{\operatorname{argmin}} \quad \sum_{i=1}^V \mathbb{E}_{\mathbf{x}_i, y_i} \Big[\ell_i(\boldsymbol{\alpha}(\mathbf{D}_i, \mathbf{x}_i)^T \mathbf{w}_i, y_i) \Big]$$
$$\text{s. t.} \quad \mathbf{D}_i = \mathbf{D}_j, \ \mathbf{w}_i = \mathbf{w}_j, \ j \in n_i. \tag{1.5}$$

where estimators take the form $f_i(\mathbf{x}) = \boldsymbol{\alpha}(\mathbf{D}_i, \mathbf{x}_i)^T \mathbf{w}_i$. We solve (1.5) using similar primal-dual optimization methods, but it is here that we run into the limitations of dictionary methods. In the centralized setting (1.4), the non-convexity of the optimization problem does not empirically cause problems, since there is no duality gap (the difference between the solution to the primal problem and dual problem [139]). But to obtain exact solutions for consensus-constrained problems, the duality gap becomes a problem. Moreover, we theoretically and empirically observe the stochastic approximation error, i.e., the directional error caused by the fact that we use stochastic gradients instead of true gradients of the Lagrangian, contributes more to instability in non-convex settings than for convex multi-agent settings [95]. Despite these drawbacks, we obtain satisfactory performance in practice for control uncertainty prediction problems in robotic teams. In this application, we have a network of interconnected robots traversing distinct paths and recording sensory observations, and we successfully apply solutions to (1.5) such that the steering mistakes of one robot may be anticipated by another (Section 5.5).

A major drawback of substituting an alternative signal representation into the general learning problem and optimizing jointly over the representation and regression weights, as is done in supervised dictionary learning, is that we lose convexity, which is fundamental to computational efficiency. Theoretically and empirically, we observe instability due to non-convexity is more prominent in multi-agent settings than centralized settings due to complications arising from Lagrange duality. This observation disincentivizes investigating more sophisticated statistical techniques for multi-agent learning that also yield non-convex problems, namely, neural networks [103]. Nascent optimization tools have been developed for non-convex optimization methods in decentralized settings [52], but their applicability to streaming data (online) settings has not yet materialized, and thus it is unclear whether task-driven dictionaries or neural networks can be stably trained in online multi-agent settings. This remark motivates a broader question: is there a way to achieve or surpass the increased expressive power of dictionary learning methods relative to GLMs while preserving convexity, and ensure the resulting estimator is of moderate complexity? This question is addressed in **Part III** of this thesis, whose contributions are summarized in the following subsection.

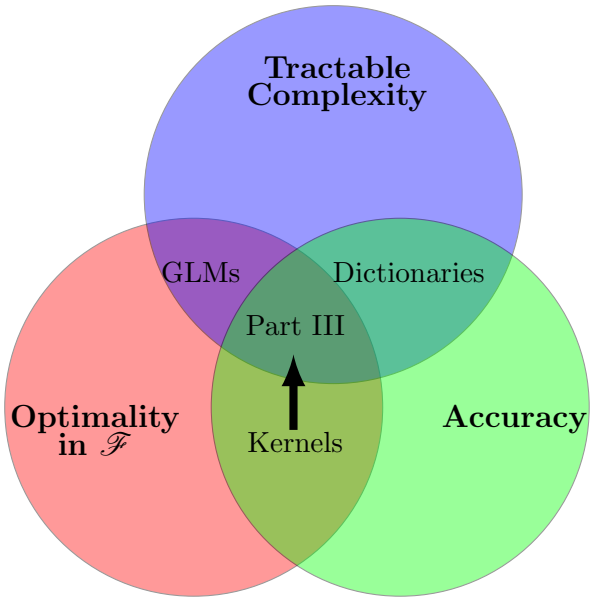


Figure 1.2: Part I allowed multi-agent networks to learn low complexity optimal linear statistical models, but their statistical accuracy in practice is limited. Part II developed decentralized dictionary methods for statistical learning based on the fact that in centralized settings which obtain state of the art statistical accuracy on a real robotics problem. However, the non-convexity defined by their training precludes our ability to find optimal dictionary-based statistical models. Thus Part III proposes using nonparametric estimation methods using reproducing kernels to learn accurate statistical models defined by infinite-dimensional stochastic convex programs. We develop methods such that we obtain moderate complexity kernel representations whose optimality properties are preserved (Chapter 6) and show how it may be used for multi-agent settings in Chapter 7. Then, we leverage tools of Chapter 6 to address a long-standing open problem in reinforcement learning: how to solve Bellman's evaluation equation (Chapter 8).

1.2 Main Contribution

Elements of Statistical Accuracy When designing estimators that are optimal within a function class that are also as accurate as possible, it is important to first understand from whence accuracy comes. The accuracy of an estimator $f \in \mathscr{F}$ hinges on a few contributing factors: the number of data samples, the representation complexity of f for a fixed function class $\mathscr{F}$, and whether the optimal Baye's risk estimator (1.1) belongs to $\mathscr{F}$ to begin with. The best achievable accuracy for N fixed samples is proportional to $1/N$ [202,204]. The third source of statistical error regarding whether the estimator that truly makes the minimal number of mistakes, i.e., how close it is to the one which achieves the minimal Baye's risk regardless of function class $\mathscr{F}$ is not addressed in this thesis, and is an active research area.

Our focus is on the second source: how to obtain an as-accurate-as-possible estimator for a fixed function class $\mathscr{F}$ whose complexity is at-worst moderate. By selecting arbitrarily complicated functional representations such as infinite-dimensional nonparametric methods

[109] or neural networks [103], it is possible to drive down the statistical error close to zero, but at the cost of high complexity estimates that are not applicable to light-weight autonomous systems and rapid adaptation. Thus, instead, in pursuit of methods well-suited to adaptive systems, we seek accurate and optimal estimators of at-worst moderate complexity. We are able to establish that the statistical error vanishes with an estimator of *fixed complexity*. The mathematical formalism for this development is *reproducing kernel Hilbert spaces* and nonparametric estimation, the focus of **Part III** of this dissertation. In **Chapter 6**, in particular, we consider the centralized statistical optimization problem

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}} \mathbb{E}_{\mathbf{x},y}[\ell(f(\mathbf{x}), y)] + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 \,, \tag{1.6}$$

which is a special case of (1.2) when the function class $\mathscr{F} = \mathcal{H}$ is a Hilbert space equipped with a "feature map" called reproducing kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Here $\mathcal{X} \subset \mathbb{R}^p$ denotes the feature space to which training examples $\mathbf{x}_n$ belong. We defer an exact discussion of the technicalities of κ to Chapter 6, but note that common examples include polynomial kernel and the radial basis kernel, i.e., $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + q\right)^b$ and $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2\tilde{\sigma}^2}\right\}$, respectively, where $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. The benefit of this kernel function is that Riesz Representation Theorems from functional analysis [212] allow us to transform an optimization problem over a function space into one over a weighted combination of kernel evaluations over training examples [169]:

$$f(\mathbf{x}) = \sum_{n=1}^{N} w_n \kappa(\mathbf{x}_n, \mathbf{x}) \,. \tag{1.7}$$

where N in (1.7) coincides with the training sample size. The Hilbert-norm regularizer $\frac{\lambda}{2}\|f\|_{\mathcal{H}}^2$ in (1.7) is needed to guarantee the applicability of (1.7). Observe that the expectation in (1.6) means that the data sample size $N = \infty$ is infinite, which implies that f has infinite complexity. Worse yet, when applying functional generalizations of stochastic gradient method (SGD) to (1.6)

$$f_{t+1} = f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t) \tag{1.8}$$

and inductively applying the Representer Theorem (1.7), we obtain a parametric representation of f_t that has complexity comparable to t, the iteration index [89].

In Chapter 6, we address the complexity explosion of f^* as well as f_t [cf. (1.8)], the functional generalization of SGD (also referred to as kernel adaptive filtering [193]), through the use of greedily constructed subspace projections so that we may guarantee exact convergence of f_t to f^* when the projection-based error attenuates over time [98] – see Figure 1.2. Then, we augment this result so that we obtain a low-memory representation $f^\infty = \lim f_t$

which is guaranteed to be close to f^*. The resulting empirical statistical accuracy of this approach far surpasses methods developed in Parts I and II, and yields *online training* performance that is comparable to nonparametric statistical methods which are *trained with all data available in advance*. Thus, carefully sparsified nonparametric methods suggest a path forward for intelligent systems to accurately learn from streaming data while maintaining provable global stability due to convexity.

Chapter 7 builds upon the lessons learnt from Chapter 6 to address the problem of kernelized stochastic optimization in multi-agent systems. This setting is one in which each agent among a network of interconnected agents seeks to learn a memory-efficient *nonlinear statistical model* which is approximately optimal with respect to information aggregated across the entire network based upon only its local data stream $\{\mathbf{x}_{i,n}, y_{i,n}\}$. We consider the case that each agent's data stream $\{\mathbf{x}_{i,n}, y_{i,n}\}$ is drawn from random pairs $(\mathbf{x}_i, y_i)$ which have a common distribution. Therefore at optimality we would like to satisfy the consensus constraint $f_i = f_j \, , (i,j) \in \mathcal{E}$, where f_i denotes the statistical mode of node i. This setting is captured by the nonparametric decentralized stochastic program

$$\begin{aligned} f^* = \; & \underset{\{f_i\} \in \mathcal{H}}{\operatorname{argmin}} \quad \sum_{i \in \mathcal{V}} \left(\mathbb{E}_{\mathbf{x}_i, y_i} \Big[\ell_i(f_i(\mathbf{x}), y_i) \Big] + \frac{\lambda}{2} \|f_i\|_{\mathcal{H}}^2 \right) \\ & \text{such that} \quad f_i = f_j \, , (i,j) \in \mathcal{E} \end{aligned} \tag{1.9}$$

whose (even approximate) solution has eluded the research community for a decade, although some prior attempts have been made [67, 141]. In this chapter, we develop a decentralized stochastic approximation-based method to nearly exactly solve (1.9). However, due to the subtleties of the RKHS $\mathcal{H}$, i.e., the fact that the Representer Theorem has not been established for RKHS-valued stochastic saddle point problems, we cannot make use of Lagrange duality to directly address the constraint in (1.9), and therefore cannot make use of the primal-dual stochastic methods wielded to address consensus and proximity constraints in Parts I - II. Instead, we adopt an approximate primal-only approach based on penalty methods, which in the context of multi-agent optimization is called distributed gradient descent [136]. In particular, we apply functional SGD to the proposed node-separable penalty function operating with greedily constructed subspace projections akin to the techniques in Chapter 6 to obtain a true method for decentralized statistical learning. We obtain a provably stable and memory-efficient method for each individual in the network to find, based on its local data stream and message passing with its neighbors, a close approximation to the globally optimal regression function [96]. We further observe that the resulting protocol translates well into practice: we obtain state of the art performance for decentralized online multi-class classification in Section 7.4 that attains comparable accuracy to centralized batch solutions to kernelized function estimation. Thus, the RKHS provides a foundation

for stable collaborative statistical learning from streaming data in multi-agent systems, in contrast to our attempts to extend dictionary methods to decentralized settings.

Path Forward for Collaborative Statistical Learning Up until now, we have addressed the problem of statistical inference from streaming data in centralized and decentralized settings. That is, from a training examples $\mathbf{x}_n$, we seek to predict y_n as $\hat{y}_n = f(\mathbf{x}_n)$. Based on the function class $\mathscr{F}$ to which f belongs, the difficulty of the optimization problem that defines finding f varies, as does its statistical performance. By exploiting structural properties of the choice of $\mathscr{F}$ in a principled way, while also making use of convex optimization techniques such as Lagrange duality and penalty methods, we have developed tools that successfully allow a network of interconnected agents to collaboratively learn accurate statistical models from their local data streams and message passing with their neighbors. While we have not solved nonparametric multi-agent stochastic programs exactly, we have solved them approximately in a memory-efficient way that is *provably stable* and yields *good performance in practice*. It is left to future directions, discussed in more detail in Chapter 9, to solve multi-agent nonparametric stochastic programs exactly using Lagrange duality, as well as extend this framework to settings such as, e.g., RKHSs with compositional multi-layer kernels (and possibly come within striking distance of the off-line accuracy benchmarks set forth by deep learning) and different hypotheses regarding agents' data which may motivate use of proximity constraints as in Chapter 3.

From Statistical Learning to Stochastic Control From here, we shift focus to a more challenging class of problems, motivated by the fact that for true learning, making good predictions is not enough. We would also like an autonomous agent to augment its behavior over time based on rewards. This notion of tailoring actions to incentives may be formulated by assuming the agent, starting at state $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^p$ at time t, selects an action vector $\mathbf{a}_t \in \mathcal{A} \subset \mathbb{R}^q$ at time t whose choice influences the state of the world $\mathbf{x}_{t+1} \in \mathcal{X} \subset \mathbb{R}^p$ at time $t+1$, where we denote $\mathbf{x}_{t+1}$ as $\mathbf{y}_t$ for disambiguation. Moreover, when the agent transitions to state $\mathbf{y}_t$, an instantaneous reward $r(\mathbf{x}_t, \mathbf{a}_t, \mathbf{y}_t)$ is assigned which quantifies, for instance, proximity to a goal location, platform stability, or portfolio revenue. We consider the case where the action selection $\mathbf{a}_t \in \mathcal{A}$ causes a transition to next state $\mathbf{y}_t \in \mathcal{X}$ that follows some state and action dependent conditional probability density $\mathbf{y}_t \sim \mathbb{P}(\cdot \,|\, \mathbf{x}_t, \mathbf{a}_t)$. and the reward function is a map $r : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$. This situation may be modeled by a Markov decision process (MDP) [185] with a continuous state and action space which is a quintuple $(\mathcal{X}, \mathcal{A}, \mathbb{P}, r, \gamma)$, where $\mathbb{P}$ is the action-dependent transition probability of the process and r is the reward map, as noted above. The general goal in an MDP is for the agent to choose actions $\{\mathbf{a}_t\}_{t=1}^{\infty}$ which maximize the reward accumulation starting from an initial state $\mathbf{x}$, also called the *value function*:

$$V(\mathbf{x}, \{\mathbf{a}_t\}_{t=0}^{\infty}) = \mathbb{E}_{\mathbf{y}}\Big[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{x}_t, \mathbf{a}_t, \mathbf{y}_t) \,\Big|\, \mathbf{x}_0 = \mathbf{x}, \{\mathbf{a}_t\}_{t=0}^{\infty}\Big]. \tag{1.10}$$

The determination of the optimal action sequence $\{\mathbf{a}_t\}$ in (1.11) is a mathematically challenging problem when $\mathcal{X}$ and $\mathcal{A}$ are continuous whose efficient solution has eluded the research community for decades [20, 23]. Rather than attempt to harpoon the "white whale" of reinforcement learning directly, i.e., the determination of how to choose the action sequence, in **Chapter 8**, we first answer a long-standing question of how to determine how good an action sequence is with respect to the long term accumulation of rewards. This task is known as *policy evaluation*.

In policy evaluation, control decisions $\mathbf{a}_t$ are chosen according to a fixed stationary stochastic policy $\pi : \mathcal{X} \to \rho(\mathcal{A})$, where $\rho(\mathcal{A})$ denotes the set of probability distributions over $\mathcal{A}$. Policy evaluation underlies methods that seek optimal policies through repeated evaluation and improvement [104]. In policy evaluation, we seek to compute the *value* of a policy when starting in state $\mathbf{x}$, quantified by the discounted expected sum of rewards, or value function $V^{\pi}(\mathbf{x})$:

$$V^{\pi}(\mathbf{x}) = \mathbb{E}_{\mathbf{y}}\Big[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{x}_t, \mathbf{a}_t, \mathbf{y}_t) \,\Big|\, \mathbf{x}_0 = \mathbf{x}, \mathbf{a}_t = \pi(\mathbf{x}_t)\}_{t=0}^{\infty}\Big]. \tag{1.11}$$

The value function (1.11) is parameterized by a discount factor $\gamma \in (0, 1)$, which determines the agent's farsightedness, and guarantees the series in (1.11) is always finite. In this chapter, we build upon the lessons learned regarding how to use greedily projected stochastic approximation tools in reproducing kernel Hilbert spaces for statistical inference in Chapter 6 to develop a method for *policy evaluation* in infinite spaces. To do so, we consider Bellman's evaluation equation which we reformulate as a compositional stochastic program, and then derive a stochastic quasi-gradient algorithm in a RKHS which operates together with the subspaces projections designed in Chapter 6. The benefit of the proposed nonparametric approach is that we are able to learn a value function estimate for a fixed policy that is guaranteed to converge to the true value function almost surely and is guaranteed to be of moderate memory. We observe a state of the art trade off of Bellman error (a measure of the quality of the value function estimate) and memory efficiency of our proposed approach on the Mountain Car navigation task, which bodes well for more challenging applications in robotics and econometrics. Moreover, the result of this chapter lays a foundation for finding optimal policies through alternating policy evaluation and improvement steps, and possibly finding optimal action-value functions directly through Bellman optimality equations. In **Chapter 9** we summarize the major lessons learned from Chapters 2 - 8 in terms of implications for future research directions.

Part I

Generalized Linear Models

Chapter 2

Online learning in homogeneous networks

In this chapter, we address supervised learning from streaming data in multi-agent systems, where each agent seeks to learn, based on its local data stream and message passing with its neighbors, a common parameter vector that defines a linear statistical model that is as good as a centralized agent which has access to all data in advance. The specific setting considered here consists of convex cost functions that are sequentially revealed to individual agents. In offline centralized learning the functions of all agents and all times are known beforehand and a *constant* and *common* action is selected for agents to play. In online centralized learning, functions are still available at a central location but are revealed sequentially. The *common* action to be played by agents is selected ex ante using past observations and incurs a cost ex post after the current functions become available. In distributed online learning the agents select actions based on previous cost functions observed locally and messages received from neighbors in past communication exchanges. This chapter proposes the use of a saddle point algorithm so that distributed online strategies achieve comparable performance to centralized offline strategies.

Centralized online learning problems can be formulated in the language of regret minimization [176, 202]. In this setting, a learner makes a sequence of plays to which Nature provides the answer in the form of a loss function. Regret accumulates over time of the loss difference between the online learner and a clairvoyant offline learner to which cost functions have been revealed beforehand, which we interpret as a measure of the price for causal prediction. For convex losses, several algorithms are known to achieve regret whose growth with the accumulated operating time T is sublinear – which entails vanishing cost differences between online and offline plays at specific times. Germane to this chapter is online gradient descent in which plays are updated by descending on the gradient of observed costs. Despite the mismatch of descending on the prior function while incurring a cost

in the current function, online gradient descent achieves regret that grows not faster than a function of order $O(\sqrt{T})$ in general and not faster than $O(\log T)$ under more stringent conditions [228]. Other methods to control regret growth are proximal maps [54], mirror descent [157, 177], and dual averaging [187]. All of these strategies may be understood as special cases of a strategy known as "follow the regularized leader" [176].

We develop methods for distributed online learning by building upon methods for deterministic settings. In deterministic settings, optimal actions of separable convex costs are computed using distributed optimization algorithms which can be categorized into primal methods, dual methods, and primal-dual methods. In primal methods agents descent along their local gradients while averaging their signals with those of their neighbors, [77, 134, 157, 223]. In dual methods agents reformulate distributed optimization as an agreement constrained optimization problem and ascend in the dual domain using the fact that dual function gradients can be computed while cooperating with neighboring nodes only [78, 154]. Variations of dual methods include the alternating direction method of multipliers [110, 167] and second order methods that rely on approximate Newton steps [224]. Primal-dual methods combine primal descent with dual ascent [9,134,140]. Primal methods have been generalized to distributed online learning and have proved effective for particular cases where averaging is advantageous [197, 220].

We develop a variant of the saddle point method [9, 135] to solve distributed online learning problems which achieves a regret that grows at a rate not faster than $O(\sqrt{T})$. We introduce of the concept of regret and extend it to networked settings, which we illustrate with applications to decentralized recursive least squares (Section 2.1.1) and decentralized online support vector machines (Section 2.1.2). The saddle point algorithm is developed in Section 2.2 by drawing parallels with deterministic and stochastic optimization. The method relies on the addition of equality constraints and the definition of an online Lagrangian associated with the instantaneous cost function. Primal descent steps on the Lagrangian are used to update actions and dual ascent steps are used to update the multipliers associated with the equality constraints. As in the case of online gradient descent, there is a mismatch between descending on past online Lagrangians to find an action to be played at the current time and incur a cost associated with a function that can be arbitrarily different. Despite this mismatch, and again, analogous to online gradient descent, the saddle point method achieves regret of order $O(\sqrt{T})$ (Section 2.3). This result is first established in terms of the global networked regret (Theorem 1) and then shown to hold for the regrets of individual agents as well (Theorem 2). Implementations of decentralized recursive least squares and decentralized online support vector machines demonstrate that the theoretical findings translate into practice.

2.1 Regret Minimization for Distributed Learning

We consider formulations of learning problems with instantaneous actions $\tilde{\mathbf{w}}_t \in X \subseteq \mathbb{R}^p$ chosen by a player, instantaneous functions $\ell_t : \mathbb{R}^p \to \mathbb{R}^q$ chosen by nature, and associated losses $l_t(\tilde{\mathbf{w}}_t, \ell_t)$ that indicate how good the choice of playing $\tilde{\mathbf{w}}_t$ is when nature selects the function ℓ_t. In *offline* learning the functions ℓ_t for times $t = 1, \dots T$ are known beforehand at time $t = 0$ and used to select a fixed strategy $\tilde{\mathbf{w}}_t = \tilde{\mathbf{w}}$ for all times. The total loss associated with the selection of $\tilde{\mathbf{w}}$ is $\sum_{t=1}^{T} l_t(\tilde{\mathbf{w}}, \ell_t)$. In *online* learning the function ℓ_t is revealed at time t and we are required to choose $\tilde{\mathbf{w}}_t$ without knowing ℓ_t but rather the functions $\mathbf{f}_u$ that nature played at earlier times $u < t$. The total loss associated with the variables $\tilde{\mathbf{w}}_t$ played for times $1 \leq t \leq T$ is the sum $\sum_{t=1}^{T} l_t(\tilde{\mathbf{w}}_t, \ell_t)$. The regret associated with these plays is defined as the difference between their aggregate cost and the minimum aggregate cost achievable by offline policies

$$\mathbf{Reg}_T^{\mathrm{C}} := \sum_{t=1}^{T} l_t(\tilde{\mathbf{w}}_t, \ell_t) - \inf_{\tilde{\mathbf{w}} \in X} \sum_{t=1}^{T} l_t(\tilde{\mathbf{w}}, \ell_t). \tag{2.1}$$

In regret formulations of online learning the goal is to design strategies that observe past functions $\mathbf{f}_u$ played by nature at times $u < t$ to select and action $\tilde{\mathbf{w}}_t$ that makes the regret $\mathbf{Reg}_T^C$ in (2.1) small. In particular, we say that a strategy learns optimal plays if $\mathbf{Reg}_T^C/T$ vanishes with growing T. We emphasize that the functions ℓ_t are arbitrary and that while the offline strategy has the advantage of knowing all functions beforehand, online strategies have the advantage of changing their plays at each time.

In this chapter we subsume the functions ℓ_t and the loss l_t into the function $\ell_t : \mathbb{R}^p \to \mathbb{R}$ such that $\ell_t(\tilde{\mathbf{w}}) = l_t(\tilde{\mathbf{w}}, \ell_t)$ and consider cases in which functions ℓ_t are written as a sum of components available at different nodes of a network. To be specific start by defining the optimal offline strategy as $\tilde{\mathbf{w}}^* = \operatorname{argmin}_{\tilde{\mathbf{w}}} \sum_{t=1}^{T} \ell_t(\tilde{\mathbf{w}})$ and rewrite the regret in (2.1) as

$$\mathbf{Reg}_T^{\mathrm{C}} = \sum_{t=1}^{T} \ell_t(\tilde{\mathbf{w}}_t) - \sum_{t=1}^{T} \ell_t(\tilde{\mathbf{w}}^*). \tag{2.2}$$

Further consider a symmetric and connected network $\mathcal{G} = (V, \mathcal{E})$ with V nodes forming the vertex set $\mathcal{V} = \{1, \dots, V\}$ and $E = |\mathcal{E}|$ directed edges of the form $e = (j, k)$. That the network is symmetric means that if $e = (j, k) \in \mathcal{E}$ it must also be that $e' = (k, j) \in \mathcal{E}$. That the network is connected means that all pairs of nodes are connected by a chain of edges. We also define the neighborhood of j as the set of nodes $n_j := \{k : (j, k) \in \mathcal{E}\}$ that share an edge with j. Each node in the network is associated with a sequence of cost functions $\ell_{i,t} : \mathbb{R}^p \to \mathbb{R}$ for all times $t \geq 1$. If a common variable $\tilde{\mathbf{w}}$ is played for all these functions

the global network cost at time t is then given by

$$\ell_t(\tilde{\mathbf{w}}) = \sum_{i=1}^{V} \ell_{i,t}(\tilde{\mathbf{w}}). \tag{2.3}$$

The functions $\ell_{i,t}$ in (2.4), and as a consequence the functions ℓ_t are assumed convex for all times t but are otherwise arbitrary.

Combining the definitions in (2.2) and (2.3) we can consider a coordinated game where all agents play a common variable $\tilde{\mathbf{w}}_t$ at time t. The accumulated regret associated with playing the coordinated sequence $\{\tilde{\mathbf{w}}_t\}_{t=1}^T$, as opposed to playing the optimal $\tilde{\mathbf{w}}^* = \operatorname{argmin}_{\tilde{\mathbf{w}}} \sum_{t=1}^T \ell_t(\tilde{\mathbf{w}})$ for all times t, can then be expressed as

$$\mathbf{Reg}_T^{\mathrm{C}} = \sum_{t=1}^{T} \sum_{i=1}^{V} \ell_{i,t}(\tilde{\mathbf{w}}_t) - \sum_{t=1}^{T} \sum_{i=1}^{V} \ell_{i,t}(\tilde{\mathbf{w}}^*). \tag{2.4}$$

An alternative formulation is to consider that agents play their own variables $\mathbf{w}_{i,t}$ to incur their own local cost $\ell_{i,t}(\mathbf{w}_{i,t})$. In this case we have the aggregate cost $\sum_{i=1}^V \ell_{i,t}(\mathbf{w}_{i,t})$ which leads to the definition of the uncoordinated regret by time T as

$$\mathbf{Reg}_T^{\mathrm{U}} = \sum_{t=1}^{T} \sum_{i=1}^{V} \ell_{i,t}(\mathbf{w}_{i,t}) - \sum_{t=1}^{T} \sum_{i=1}^{V} \ell_{i,t}(\tilde{\mathbf{w}}^*). \tag{2.5}$$

This formulation is of little interest because agents are effectively independent of each other. Indeed, to reduce the regret in (2.5) it suffices to let agents learn strategies that are good with respect to their local costs $\sum_{t=1}^T \ell_{i,t}(\mathbf{w}_{i,t})$. A simple local gradient descent policy can achieve small regret with respect to the optimal *local* action $\mathbf{w}_i^* = \operatorname{argmin}_{\mathbf{w}_i} \sum_{t=1}^T \ell_{i,t}(\mathbf{w}_i)$ [113]. This uncoordinated strategy is likely to result in negative regret in (2.5) since the variable $\tilde{\mathbf{w}}^*$ is chosen as common across all agents.

A more appropriate formulation is to consider games where agents have an incentive to learn the cost functions of their peers. Suppose then that each agent in the network plays his own variables $\mathbf{w}_{i,t}$ which are not necessarily identical to the variables $\mathbf{w}_{j,t}$ played by other agents $j \neq i$ in the same time slot. However, we still want each agent to learn a play that is optimal with respect to the global cost in (2.3). Thus, we formulate a problem in which the *local regret* of agent j is defined as

$$\mathbf{Reg}_T^{j} = \sum_{t=1}^{T} \sum_{i=1}^{V} \ell_{i,t}(\mathbf{w}_{j,t}) - \sum_{t=1}^{T} \sum_{i=1}^{V} \ell_{i,t}(\tilde{\mathbf{w}}^*). \tag{2.6}$$

The regret formulations in (2.4) and (2.6) are identical. This means that (2.6) corresponds to a problem in which agent j aspires to learn a play that is as good as the play that can

be learned by a centralized agent that has access to the cost functions $\ell_{i,t}$ of all agents i. However, the assumption here is that only the local functions $\ell_{j,t}$ are known to agent j.

By further considering the sum of all local regrets in (2.6) we define *global networked regret* as

$$\mathbf{Reg}_T := \frac{1}{V}\sum_{j=1}^{V}\mathbf{Reg}_T^j = \frac{1}{V}\sum_{t=1}^{T}\sum_{i,j=1}^{V}\ell_{i,t}(\mathbf{w}_{j,t}) - \sum_{t=1}^{T}\sum_{i=1}^{V}\ell_{i,t}(\tilde{\mathbf{w}}^*), \tag{2.7}$$

where we used (2.6) and simplified terms to write the second equality. In this chapter we develop a variation of the saddle point algorithm of Arrow and Hurwicz [9] to find a strategy whose local and global network regrets [cf. (2.6) and (2.7)] are of order not larger than $O(\sqrt{T})$. We also show that the proposed algorithm can be implemented by agents that have access to their local cost functions only and perform causal variable exchanges with peers in their network neighborhood. This saddle point algorithm is presented in Section 2.2 presenting two examples.

2.1.1 Distributed recursive least squares

As an example problem that admits the formulation in (2.7) consider a distributed version of recursive least squares (RLS). Suppose we want to estimate a signal $\tilde{\mathbf{w}} \in \mathbb{R}^p$ when agents collect observations $\mathbf{y}_{it} \in \mathbb{R}^q$ that relate to $\tilde{\mathbf{w}}$ according to the model $\mathbf{y}_{it} = \mathbf{H}_{i,t}\tilde{\mathbf{w}} + \mathbf{w}_{i,t}$, where the noise $\mathbf{w}_{i,t}$ is Gaussian and independent and identically distributed across nodes and time. The optimal estimator $\tilde{\mathbf{w}}^*$ given the observations $\mathbf{y}_{i,t}$ for all i and t is the least mean squared error estimator $\tilde{\mathbf{w}}^* = \text{argmin}_x \sum_{t=1}^{T}\sum_{i=1}^{V}\|\mathbf{H}_{i,t}\tilde{\mathbf{w}} - \mathbf{y}_{i,t}\|^2$. If the signals $\mathbf{y}_{i,t}$ are known for all nodes i and times t the optimal estimator $\tilde{\mathbf{w}}^*$ can be easily computed. In this chapter we are interested in cases where the signal $\mathbf{y}_{j,t-1}$ is revealed at time $t-1$ to sensor j which then proceeds to determine the causal signal estimate $\mathbf{w}_{j,t} \in \mathbb{R}^p$ as a function of past observations $y_{j,u}$ for $u = 1, \ldots, t-1$ and information received from neighboring nodes in previous time slots. This is tantamount to a distributed RLS problem because signals are revealed sequentially to agents of a distributed network. Setting aside for the moment the issue of how to select $\mathbf{w}_{j,t}$ the regret in (2.6) is a measure of goodness for $\mathbf{w}_{j,t}$ with respect to a clairvoyant centralized estimator. The particular form of (2.6) becomes

$$\mathbf{Reg}_T^j = \sum_{t=1}^{T}\sum_{i=1}^{V}\|\mathbf{H}_{i,t}\mathbf{w}_{j,t} - \mathbf{y}_{i,t}\|^2 - \sum_{t=1}^{T}\sum_{i=1}^{V}\|\mathbf{H}_{i,t}\tilde{\mathbf{w}}^* - \mathbf{y}_{i,t}\|^2. \tag{2.8}$$

The regret $\mathbf{Reg}_T^j$ in (2.8) is measuring the mean squared error penalty that agent j is incurring by estimating the signal $\tilde{\mathbf{w}}$ as $\mathbf{w}_{j,t}$ instead of the optimal estimator $\tilde{\mathbf{w}}^*$. In that sense it can be interpreted as the penalty for distributed causal operation with respect to centralized clairvoyant operation – the estimate $\tilde{\mathbf{w}}^*$ is centralized because it has access to

the observations of all nodes and clairvoyant because it has access to the current observation $\mathbf{y}_{i,t}$. The algorithms developed in this chapter are such that the regret penalty $\mathbf{Reg}_T^j$ in (2.8) grows at a sub-linear rate not larger than $O(\sqrt{T})$ – see Sections 2.2 and 2.3.

2.1.2 Decentralized Online Support Vector Machines

As a second example consider the problem of training a support vector machine (SVM) for binary classification [39]. Suppose that each agent i is given a training data set $\mathcal{S}_i$ with T elements that are revealed sequentially. The elements of this set are pairs $(\mathbf{x}_{i,t}, y_{i,t})$ where $\mathbf{x}_{i,t} \in \mathbb{R}^p$ is a feature vector having a known binary label $y_{i,t} \in \{-1, 1\}$. Given the aggregate training set $\mathcal{S} = \cup_{i=1}^V \mathcal{S}_i$ we seek a decision hyperplane which best separates data points with distinct labels. That is, we seek a vector $\tilde{\mathbf{w}} \in \mathbb{R}^p$ such that $\tilde{\mathbf{w}}^T \mathbf{x}_{i,t} > 0$ whenever $y_{i,t} = 1$ and $\tilde{\mathbf{w}}^T \mathbf{x}_{i,t} > 0$ for $y_{i,t} = -1$. Since data sets are not separable in general, we consider a soft margin formulation which penalizes misclassifications through the hinge loss $l((\mathbf{x}_{i,t}, y_{i,t}); \tilde{\mathbf{w}}) := \max(0, 1 - y_{i,t} \tilde{\mathbf{w}}^T \mathbf{x}_{i,t})$. The hinge loss $l((\mathbf{x}_{i,t}, y_{i,t}); \tilde{\mathbf{w}})$ is null if the label $y_{i,t}$ is correctly classified by the hyperplane defined by $\tilde{\mathbf{w}}$ – which happens when $\tilde{\mathbf{w}}^T \mathbf{x}_{i,t} > 0$ for $y_{i,t} = 1$ and $\tilde{\mathbf{w}}^T \mathbf{x}_{i,t} < 0$ for $y_{i,t} = -1$ – and grows linearly with the distance between the point $\mathbf{x}_{i,t}$ and the classifying hyperplane otherwise. To balance model complexity with training error we further add a quadratic regularization term so that the optimal classifier $\tilde{\mathbf{w}}^*$ is the one that minimizes the cost

$$\tilde{\mathbf{w}}^* = \underset{\tilde{\mathbf{w}} \in X}{\operatorname{argmin}} \frac{\zeta}{2} \|\tilde{\mathbf{w}}\|_2^2 + \frac{1}{VT} \sum_{t=1}^{T} \sum_{i=1}^{V} \max\Big(0, 1 - y_{i,t} \cdot \tilde{\mathbf{w}}^T \mathbf{x}_{i,t}\Big), \tag{2.9}$$

where ζ is the regularization constant tuning classifier bias and variance. The classifier $\tilde{\mathbf{w}}^*$ that results from solving (2.9) is the centralized batch classifier.

To consider distributed online versions of SVM training define functions $\ell_{i,t} : \mathbb{R}^p \to \mathbb{R}$ with values

$$\ell_{i,t}(\tilde{\mathbf{w}}) = \frac{\zeta}{2} \|\tilde{\mathbf{w}}\|_2^2 \; + \; \max\Big(0, 1 - y_{i,t} \tilde{\mathbf{w}}^T \mathbf{x}_{i,t}\Big), \tag{2.10}$$

so that the minimization argument in (2.9) can be written as $\sum_{i,t} \ell_{i,t}(\tilde{\mathbf{w}})$. This modification yields the case where each agent in the network has access only to a distinct local labeled data set.

The total penalty $\sum_{i,t} \ell_{i,t}(\tilde{\mathbf{w}}^*)$ of the optimal batch or offline classifier $\tilde{\mathbf{w}}^*$ quantifies the number of misclassifications incurred when the observations of all nodes and all times are known beforehand. The various online classifiers whose performances are described by (2.4), (2.5), and (2.6) quantify the number of misclassifications incurred when the class corresponding to feature vectors $\mathbf{x}_{i,t}$ is predicted causally using features $\mathbf{x}_{i,u}$ and associated

classes $y_{i,u}$ observed at past times $u < t$. The centralized regret $\mathbf{Reg}_T^{\mathrm{C}}$ in (2.4) corresponds to the case when all observations are causally available at a central location. The uncoordinated regret $\mathbf{Reg}_T^{\mathrm{U}}$ in (2.5) corresponds to the case where classification is based on past local observations only. The local regrets $\mathbf{Reg}_T^j$ in (2.6) corresponds to cases when each of the agents is trying to accumulate past global network knowledge through communication with local agents.

2.2 Arrow-Hurwicz Saddle Point Algorithm

We turn to developing a saddle point algorithm to control the growth of the local and global network regrets [cf. (2.6) and (2.7)]. Since the regret functions $\mathbf{Reg}_T^j$ defined in (2.6) are the same for all agents j, plays $\mathbf{w}_{j,t}$ that are good for one agent are also good for another. Thus, a suitable strategy is to select actions $\mathbf{w}_{j,t}$ which are the same for every agent. Since the network $\mathcal{G}$ is assumed to be connected, this relationship can be attained by imposing the constraint $\mathbf{w}_{j,t} = \mathbf{w}_{k,t}$ for all pairs of neighboring nodes $(j,k) \in \mathcal{E}$. To write more compactly define the column vector $\mathbf{w}_t := [\mathbf{w}_{1,t}; \ldots; \mathbf{w}_{V,t}] \in \mathbb{R}^{Vp}$ and the augmented graph edge incidence matrix $\mathbf{C} \in \mathbb{R}^{Ep \times Vp}$. The matrix $\mathbf{C}$ is formed by $E \times V$ square blocks of dimension p. If the edge $e = (j,k)$ links node j to node k the block (e,j) is $[\mathbf{C}]_{ej} = \mathbf{I}_p$ and the block $[\mathbf{C}]_{ek} = -\mathbf{I}_p$, where $\mathbf{I}_p$ denotes the identity matrix of dimension p. All other blocks are identically null, i.e., $[\mathbf{C}]_{ek} = \mathbf{0}_p$ for all edges $e \neq (j,k)$. With this definitions the constraint $\mathbf{w}_{j,t} = \mathbf{w}_{k,t}$ for all pairs of neighboring nodes can be written as

$$\mathbf{C}\mathbf{w}_t = \mathbf{0}, \qquad \text{for all } t = 1, \ldots, T. \tag{2.11}$$

The edge incidence matrix $\mathbf{C}$ has exactly p null singular values. We denote as $0 < \gamma$ the smallest nonzero singular value of $\mathbf{C}$ and as Γ the largest singular value of $\mathbf{C}$. The singular values γ and Γ are measures of network connectedness.

Imposing the constraint in (2.11) for all times t requires global coordination – indeed, the formulation would be equivalent to the centralized regret problem in (2.4). Instead, we consider a modification of (2.3) in which we add a linear penalty term to incentivize coordination. Introduce then dual variables $\boldsymbol{\lambda}_{e,t} = \boldsymbol{\lambda}_{jk,t} \in \mathbb{R}^p$ associated with the constraint $\mathbf{w}_{j,t} - \mathbf{w}_{k,t} = \mathbf{0}$ and consider the addition of penalty terms of the form $\boldsymbol{\lambda}_{jk,t}^T(\mathbf{w}_{j,t} - \mathbf{w}_{k,t})$. For an edge that starts at node j, the multiplier $\boldsymbol{\lambda}_{jk,t}$ is assumed to be kept at node j. Define the stacked vector $\boldsymbol{\lambda}_t := [\boldsymbol{\lambda}_{1,t}; \ldots; \boldsymbol{\lambda}_{E,t}] \in \mathbb{R}^{Ep}$ and the online Lagrangian at time t as

$$\mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t) \;=\; \sum_{i=1}^{V} \ell_{i,t}(\mathbf{w}_{i,t}) + \boldsymbol{\lambda}_t^T \mathbf{C}\mathbf{w}_t \;= f_t(\mathbf{w}) + \boldsymbol{\lambda}_t^T \mathbf{C}\mathbf{w}_t. \tag{2.12}$$

The definition in (2.12) corresponds to the Lagrangian associated with the minimization

of the instantaneous function $\sum_{i=1}^{V} \ell_{i,t}(\mathbf{w}_{i,t})$ subject to the agreement constraint $\mathbf{C}\mathbf{w}_t = \mathbf{0}$. Using this interpretation of the online Lagragian we consider the use of the Arrow-Hurwicz saddle point method. This method exploits the fact that primal-dual optimal pairs are saddle points of the Lagrangian to work through successive primal gradient descent steps and dual gradient ascent steps. Particularized to the online Lagrangian in (2.12) the saddle point algorithm takes the form

$$\mathbf{w}_{t+1} = \mathcal{P}_W[\mathbf{w}_t - \eta \nabla_{\mathbf{w}} \mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)], \tag{2.13}$$

$$\boldsymbol{\lambda}_{t+1} = \mathcal{P}_\Lambda[\boldsymbol{\lambda}_t + \eta \nabla_{\boldsymbol{\lambda}} \mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)], \tag{2.14}$$

where η is a given stepsize. The notation $\mathcal{P}_\Lambda(\boldsymbol{\lambda})$ denotes projection of dual variables on a given convex compact set Λ. We assume that the set of multipliers Λ can be written as a Cartesian product of sets Λ_{jk} so that the projection of $\boldsymbol{\lambda}$ into Λ is equivalent to the separate projection of the components $\boldsymbol{\lambda}_{jk}$ into the sets Λ_{jk}. The notation $\mathcal{P}_W(\mathbf{w})$ denotes projection onto the set of feasible primal variables so that we have $\mathbf{w}_j \in W$ for all the V components of the vector $\mathbf{w} := [\mathbf{w}_1; \dots; \mathbf{w}_V]$.

The pair of iterations in (2.13)-(2.14) can be implemented in a distributed manner such that the variables kept at node j, namely, $\mathbf{w}_{j,t}$ and $\boldsymbol{\lambda}_{jk,t}$, are updated using the values of other local variables and variables of neighboring nodes, namely, $\mathbf{w}_{k,t}$ and $\boldsymbol{\lambda}_{kj,t}$ for $k \in n_j$. In particular, take the gradient with respect to $\mathbf{w}_j$ in (2.13) and observe that only the term $\nabla_{\mathbf{w}_j} \ell_{j,t}(\mathbf{w}_{j,t})$ is not null in the sum in (2.12). Further observe that when taking the gradient of the linear penalty term $\boldsymbol{\lambda}_t^T \mathbf{C}\mathbf{w}_t$ the variable $\mathbf{w}_j$ appears only in the terms associated with edges of the form $e = (j,k)$ or $e = (k,j)$. Thus, the gradient of this penalty term with respect to $\mathbf{w}_j$ can be written as $\nabla_{\mathbf{w}_j}(\boldsymbol{\lambda}_t^T \mathbf{C}\mathbf{w}_t) = \sum_{k \in n_j} \boldsymbol{\lambda}_{jk,t} - \boldsymbol{\lambda}_{kj,t}$. These two observations imply that the gradient of the online Lagrangian with respect to the primal variable $\mathbf{w}_{j,t}$ of node j can be written as

$$\nabla_{\mathbf{w}_j} \mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t) = \nabla_{\mathbf{w}_j} \ell_{j,t}(\mathbf{w}_{j,t}) + \sum_{k \in n_j} (\boldsymbol{\lambda}_{jk,t} - \boldsymbol{\lambda}_{kj,t}). \tag{2.15}$$

The computation of this gradient only depends on the local gradient of the local loss function $\ell_{j,t}$, the local primal variable $\mathbf{w}_{j,t}$, the local dual variables $\boldsymbol{\lambda}_{jk,t}$ and the dual variables $\boldsymbol{\lambda}_{kj,t}$ of neighboring nodes $k \in n_j$. Similarly, to determine the gradient of the online Lagrangian with respect to the dual variable $\boldsymbol{\lambda}_{jk}$ observe that the only term in (2.12) that involves this variable is the one associated with the constraint $\mathbf{w}_{j,t} - \mathbf{w}_{k,t}$. Therefore, the gradient with respect to $\boldsymbol{\lambda}_{jk}$ can be written as

$$\nabla_{\boldsymbol{\lambda}_{jk}} \mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t) = \mathbf{w}_{j,t} - \mathbf{w}_{k,t}. \tag{2.16}$$

Algorithm 1 DSPA: Distributed Saddle Point Algorithm

Require: initial actions $\mathbf{w}_0$, dual variables $\boldsymbol{\lambda}_0 = \mathbf{0}$, step-size $\eta = 1/\sqrt{T}$

for $t = 1, 2, \ldots, T$ **do**
  for agent $j \in V$ **do**
    Send primal and dual variables $\mathbf{w}_{j,t}, \boldsymbol{\lambda}_{jk,t}$ to neighbors $k \in n_j$
    Receive variables $\mathbf{w}_{k,t}, \boldsymbol{\lambda}_{jk,t}$ from neighboring agents $k \in n_j$
    Update local action $\mathbf{w}_{j,t}$ with (2.17)

$$\mathbf{w}_{j,t+1} = \mathcal{P}_W \Big[\mathbf{w}_{j,t} - \nabla_{\mathbf{w}_j} \ell_{j,t}(\mathbf{w}_{j,t}) + \sum_{k \in n_j} (\boldsymbol{\lambda}_{jk,t} - \boldsymbol{\lambda}_{kj,t}) \Big]$$

  end for
  for communication link $(j,k) \in \mathcal{E}$ **do**
    Update Lagrange Multipliers at network link (j,k) [cf. (2.18)]

$$\boldsymbol{\lambda}_{jk,t+1} = \mathcal{P}_{\boldsymbol{\Lambda}} \Big[\boldsymbol{\lambda}_{ij,t} + \eta_t \left(\mathbf{w}_{j,t} - \mathbf{w}_{k,t} \right) \Big]$$

  end for
end for

To compute this gradient at node j we use the local primal variable $\mathbf{w}_{j,t}$ and the neighboring play $\mathbf{w}_{k,t}$. Separating (2.13) along the components $\mathbf{w}_{j,t}$ associated with node j it follows that the primal iteration is equivalent to the V parallel updates

$$\mathbf{w}_{j,t+1} = \mathcal{P}_X \Big[\mathbf{w}_{j,t} - \eta \Big(\nabla_{\mathbf{w}_j} \ell_{j,t}(\mathbf{w}_{j,t}) + \sum_{k \in n_j} (\boldsymbol{\lambda}_{jk,t} - \boldsymbol{\lambda}_{kj,t}) \Big) \Big], \tag{2.17}$$

where $\mathcal{P}_X(\mathbf{w}_{j,t})$ denotes projection of $\mathbf{w}_{j,t}$ into the feasible primal set X. Likewise, separating (2.14) into the subcomponents along the $\boldsymbol{\lambda}_{jk}$ direction yields the E parallel updates

$$\boldsymbol{\lambda}_{jk,t+1} = \mathcal{P}_{\Lambda_{jk}} \Big[\boldsymbol{\lambda}_{jk,t} + \eta \left(\mathbf{w}_{j,t} - \mathbf{w}_{k,t} \right) \Big], \tag{2.18}$$

where $\mathcal{P}_{\Lambda_{jk}}$ denotes projection of $\boldsymbol{\lambda}_{jk}$ into the dual set Λ_{jk}. Node j can implement (2.17)-(2.18) by using local variables and receiving variables $\boldsymbol{\lambda}_{kj,t}$ and $\mathbf{w}_{k,t}$ maintained at neighboring nodes $k \in n_j$.

As an example application consider the distributed RLS problem in Section 2.1.1. From (2.8), we glean that local functions are $\ell_{i,t}(\mathbf{w}_{i,t}) = \|\mathbf{H}_{i,t}\mathbf{w}_{i,t} - \mathbf{y}_{i,t}\|^2$ to conclude that the primal update at agent j shown in (2.17) takes the specific form

$$\mathbf{w}_{j,t+1} = \mathcal{P}_W \Big[\mathbf{w}_{j,t} - \eta \Big(2\mathbf{H}_{j,t}^T \big(\mathbf{H}_{j,t}\mathbf{w}_{j,t} - \mathbf{y}_{j,t} \big) + \sum_{k \in n_j} (\boldsymbol{\lambda}_{jk,t} - \boldsymbol{\lambda}_{kj,t}) \Big) \Big]. \tag{2.19}$$

As a second application consider the SVM classification problem of Section 2.1.2. In this case the functions $\ell_{i,t}$ are given in (2.10) and the specific form of (2.17) is case the functions $\ell_{i,t}$ are given in (2.10) and the specific form of (2.17) is

$$\mathbf{w}_{j,t+1} = \mathcal{P}_W\Big[\mathbf{w}_{j,t} - \eta\Big(\zeta\mathbf{w}_{j,t} - y_{i,t}\mathbf{x}_{i,t}\mathbb{I}\big(y_{i,t}\tilde{\mathbf{w}}^T\mathbf{x}_{i,t} < 1\big) + \sum_{k\in n_j}(\boldsymbol{\lambda}_{jk,t} - \boldsymbol{\lambda}_{kj,t})\Big)\Big], \qquad (2.20)$$

where $\mathbb{I}\big(y_{i,t}\tilde{\mathbf{w}}^T\mathbf{x}_{i,t} < 1\big) = 1$ when $y_{i,t}\tilde{\mathbf{w}}^T\mathbf{x}_{i,t} < 1$ and $\mathbb{I}\big(y_{i,t}\tilde{\mathbf{w}}^T\mathbf{x}_{i,t} < 1\big) = 0$ otherwise. The conditional subtraction in the third term on the right hand side of (2.20) comes from the computation of the subgradient of the hinge loss, and moves the current classifier in the direction of mistaken feature vectors weighted by the label. This update may be interpreted as a projected version of the Perceptron algorithm [27,68] with a dual correction term that incorporates side information about neighbors' classifiers. For both, RLS and SVM, the dual iteration is as given in (2.18) because the form of this update is independent of the specific form of the cost functions $\ell_{i,t}$.

Remark 1 Recursive application of the primal and dual iterations in (2.13)-(2.14), or, equivalently, (2.17)-(2.18), would result in the minimization of the instantaneous global cost $\sum_{i=1}^{V}\ell_{i,t}(\mathbf{w}_{i,t})$ subject to the agreement constraint $\mathbf{C}\mathbf{w}_t = \mathbf{0}$. However, (2.13) and (2.14) are applied only once for each online Lagrangian and, moreover, this instantantaneous minimization is *not* the optimization problem that specifies the optimal action $\tilde{\mathbf{w}}^*$ which we defined as the minimizer of the accumulated cost $\sum_{t=1}^{T}\sum_{i=1}^{V}\ell_{i,t}(\tilde{\mathbf{w}})$. In fact, the variables $\mathbf{w}_{t+1}$ are obtained upon descending on the online Lagrangian $\mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)$ associated with the functions $\ell_{i,t}$ – that are observed at time t – but their contribution to the regrets in (2.6) and (2.7) is determined by the functions $\ell_{i,t+1}$ – which are to be observed after playing $\mathbf{w}_{t+1}$ at time $t+1$. It is thus not obvious that (2.13)-(2.14) is a viable strategy to control regret, even though it will turn out to be so under mild assumptions; see Section 2.3. The justification for the use of these iterations comes from modeling the functions $\ell_{i,t}$ as drawn from a stationary distribution. This renders the problem of regret minimization equivalent to the solution of a stochastic optimization problem and (2.13)-(2.14) equivalent to a stochastic saddle point algorithm. In general, methods that work in stochastic optimization tend to work for regret minimization. Do observe, however, that no stochastic model is assumed in this chapter. The functions $\ell_{i,t}$ are arbitrary.

2.3 Regret Bounds

We turn to establishing that the local and global network regrets in (2.6) and (2.7) associated with the saddle point algorithm in (2.13)-(2.14) grow not faster than $O(\sqrt{T})$. In order to obtain these results, some technical conditions are required which we state below.

AS1 *The network $\mathcal{G}$ is connected. The smallest nonzero singular value of the incidence matrix $\mathbf{C}$ is γ, the largest singular value is Γ, and the network diameter is D.*

AS2 *The gradients of the loss functions for any $\mathbf{w}$ is bounded by a constant L, i.e.*

$$\|\nabla \ell_t(\mathbf{w})\| \leq L. \tag{2.21}$$

AS3 *The loss functions $\ell_{i,t}(\mathbf{w})$ are Lipschitz continuous with modulus $K_{i,t} \leq K$,*

$$\|\ell_{i,t}(\mathbf{w}) - \ell_{i,t}(\mathbf{y})\| \;\leq\; K_{i,t}\|\mathbf{w}-\mathbf{y}\| \;\leq\; K\|\mathbf{w}-\mathbf{y}\|. \tag{2.22}$$

AS4 *The set W of feasible plays is included in the 2-norm ball of radius $C_{\mathbf{w}}/V$.*

$$W \;\subseteq\; \big\{\tilde{\mathbf{w}} \in \mathbb{R}^p : \|\tilde{\mathbf{w}}\| \leq C_{\mathbf{w}}/V\big\}. \tag{2.23}$$

AS5 *The convex set Λ_{jk} onto which the dual variables $\boldsymbol{\lambda}_{jk,t}$ are projected is included in a 1-norm ball of radius $C_{\boldsymbol{\lambda}}$,*

$$\Lambda_{jk} \;\subseteq\; \big\{\boldsymbol{\lambda} \in \mathbb{R}^p : \|\boldsymbol{\lambda}\|_1 \;\leq C_{\boldsymbol{\lambda}}\big\}, \tag{2.24}$$

for some constant $C_{\boldsymbol{\lambda}} \geq DVK + 1$.

Assumption 1 is standard in distributed algorithms. Assumptions 2 and 3 are typical in the analysis of saddle point algorithms. The bounds on the sets X and Λ_{jk} in assumptions 4 and 5 are constructed so that the iterates $\mathbf{w}_{j,t}$ and $\boldsymbol{\lambda}_{jk,t}$ are bounded by the respective constants in (2.23) and (2.24). The constant $C_{\mathbf{w}}/N$ in Assumption 4 is chosen so that the 2-norm of the stacked primal iterates $\mathbf{w}_t := [\mathbf{w}_{1,t}; \ldots; \mathbf{w}_{V,t}]$ are bounded as $\|\mathbf{w}_t\| \leq C_{\mathbf{w}}$.

The various bounds in Assumptions 1 - 5 permit bounding the norm of the gradients of the online Lagrangians in (2.12). For the gradient with respect to the primal variable $\mathbf{w}$, use of the triangle and Cauchy-Schwarz inequalities yields

$$\|\nabla_{\mathbf{w}} \mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\| = \|\nabla \ell_t(\mathbf{w}_t) + \mathbf{C}^T\boldsymbol{\lambda}_t\| \leq \|\nabla \ell_t(\mathbf{w}_t)\| + \|\mathbf{C}^T\|\|\boldsymbol{\lambda}_t\|. \tag{2.25}$$

Use now the bounds in (2.21) and (2.24) and the definition of Γ as the largest singular value of $\mathbf{C}$ to simplify (2.25) to

$$\|\nabla_{\mathbf{w}} \mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\| \leq L + \Gamma\sqrt{E}C_{\boldsymbol{\lambda}} := L_{\mathbf{w}}, \tag{2.26}$$

where we defined $L_{\mathbf{w}}$ for future reference. For the gradient with respect to the dual variable

$\boldsymbol{\lambda}$, we can similarly write

$$\|\nabla_{\boldsymbol{\lambda}} \mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\| \;=\; \|\mathbf{C}\mathbf{w}_t\| \leq \|\mathbf{C}\|\|\mathbf{w}_t\| \;\leq\; \Gamma C_{\mathbf{w}} \;:=\; L_{\boldsymbol{\lambda}}. \tag{2.27}$$

Our results concerning local and global networked regret are both derived from the following lemma that simultaneously bounds the uncoordinated regret in (2.5) and the weighted penalty disagreement $\sum_{t=1}^{T} \boldsymbol{\lambda}^T \mathbf{C}\mathbf{w}_t$ as we formally state next.

Lemma 1 *Consider the sequence* $\mathbf{w}_t := [\mathbf{w}_{1,t}; \dots; \mathbf{w}_{V,t}]$ *generated by the saddle point algorithm in* (2.17)-(2.18). *Let* $\tilde{\mathbf{w}}^*$ *be the optimal offline action in* (2.6), *assume* $\boldsymbol{\lambda}_1 = \mathbf{0}$ *and further assume that assumptions 1 - 5 hold. If we select* $\eta = 1/\sqrt{T}$ *we have that for all* $\boldsymbol{\lambda} \in \Lambda$ *it holds*

$$\sum_{t=1}^{T} \sum_{i=1}^{V} \Big[\ell_{i,t}(\mathbf{w}_{i,t}) - \ell_{i,t}(\tilde{\mathbf{w}}^*) \Big] + \sum_{t=1}^{T} \boldsymbol{\lambda}^T \mathbf{C}\mathbf{w}_t \leq \frac{\sqrt{T}}{2} \left(\|\mathbf{w}_1 - \tilde{\mathbf{w}}^*\|^2 + \|\boldsymbol{\lambda}\|^2 + L_{\mathbf{w}}^2 + L_{\boldsymbol{\lambda}}^2 \right). \tag{2.28}$$

Proof: The proof is broken up into three parts. In the first part, we use the definition of the saddle point primal iterate and the first order characterization of convexity to bound the difference between the current algorithmic choice $\mathbf{w}_t$ and an arbitrary $\mathbf{w} \in X$. In the second, we mirror the first step in the dual variable $\boldsymbol{\lambda}$. We wrap up by combining the bounds obtained in the previous two steps, summing over time and using feasibility and boundedness properties to simplify expressions.

Begin then by considering the squared 2-norm of the difference between the iterate $\mathbf{w}_{t+1}$ at time $t+1$ and an arbitrary point $\mathbf{w} \in X$ and use (2.13) to express $\mathbf{w}_{t+1}$ in terms of $\mathbf{w}_t$,

$$\|\mathbf{w}_{t+1} - \mathbf{w}\|^2 = \|\mathcal{P}_W[\mathbf{w}_t - \eta \nabla_{\mathbf{w}} \mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)] - \mathbf{w}\|^2. \tag{2.29}$$

Since $\mathbf{w} \in W$ the distance between the projected vector $\mathcal{P}_W[\mathbf{w}_t - \eta \nabla_{\mathbf{w}} \mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)]$ and $\mathbf{w}$ is smaller than the distance before projection. Use this fact in (2.29) and expand the square to write

$$\begin{aligned} \|\mathbf{w}_{t+1} - \mathbf{w}\|^2 \;&\leq \|\mathbf{w}_t - \eta \nabla_{\mathbf{w}} \mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \mathbf{w}\|^2 \\ &= \|\mathbf{w}_t - \mathbf{w}\|^2 - 2\eta \nabla_{\mathbf{w}} \mathcal{O}_t(\mathbf{w}_t, \lambda_t)^T (\mathbf{w}_t - \mathbf{w}) + \eta^2 \|\nabla_{\mathbf{w}} \mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\|^2. \end{aligned} \tag{2.30}$$

Further note that as stated in (2.26) the norm of the primal gradient of the online Lagrangian is bounded by $L_{\mathbf{w}}$. Substitute this bound for the corresponding term in (2.30) and reorder

terms to write

$$\nabla_{\mathbf{w}}\mathcal{O}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)^T(\mathbf{w}_t-\mathbf{w}) \leq \frac{1}{2\eta}\left(\|\mathbf{w}_t-\mathbf{w}\|^2-\|\mathbf{w}_{t+1}-\mathbf{w}\|^2\right)+\frac{\eta L_{\mathbf{w}}^2}{2}. \tag{2.31}$$

Observe now that since the functions $\ell_{i,t}(\mathbf{w}_i)$ are convex, the online Lagrangian is a convex function of $\mathbf{w}$ [cf. (2.12)]. Thus, it follows from the first order convexity condition that

$$\mathcal{O}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)-\mathcal{O}_t(\mathbf{w},\boldsymbol{\lambda}_t) \leq \nabla_{\mathbf{w}}\mathcal{O}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)^T(\mathbf{w}_t-\mathbf{w}). \tag{2.32}$$

Substituting the upper bound in (2.31) for the right hand side of the inequality in (2.32) yields

$$\mathcal{O}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)-\mathcal{O}_t(\mathbf{w},\boldsymbol{\lambda}_t) \leq \frac{1}{2\eta}\left(\|\mathbf{w}_t-\mathbf{w}\|^2-\|\mathbf{w}_{t+1}-\mathbf{w}\|^2\right)+\frac{\eta L_{\mathbf{w}}^2}{2}. \tag{2.33}$$

We set this analysis aside and proceed to repeat the steps in (2.29)-(2.33) for the distance between the iterate $\boldsymbol{\lambda}_{t+1}$ at time $t+1$ and an arbitrary multiplier $\boldsymbol{\lambda}$.

$$\|\boldsymbol{\lambda}_{t+1}-\boldsymbol{\lambda}\|^2 = \|\mathcal{P}_{\Lambda}[\boldsymbol{\lambda}_t+\eta\nabla_{\boldsymbol{\lambda}}\mathcal{O}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)]-\boldsymbol{\lambda}\|^2, \tag{2.34}$$

where we have substituted (2.14) to express $\boldsymbol{\lambda}_{t+1}$ in terms of $\boldsymbol{\lambda}_t$. Using the non-expansive property of the projection operator in (2.34) and expanding the square, we obtain

$$\begin{aligned}\|\boldsymbol{\lambda}_{t+1}-\boldsymbol{\lambda}\|^2 &\leq \|\boldsymbol{\lambda}_t+\eta\nabla_{\boldsymbol{\lambda}}\mathcal{O}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)-\boldsymbol{\lambda}\|^2. \\ &= \|\boldsymbol{\lambda}_t-\boldsymbol{\lambda}\|^2+2\eta\nabla_{\boldsymbol{\lambda}}\mathcal{O}_t(\mathbf{w}_t,\lambda_t)^T(\boldsymbol{\lambda}_t-\boldsymbol{\lambda})+\eta^2\|\nabla_{\boldsymbol{\lambda}}\mathcal{O}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)\|^2.\end{aligned} \tag{2.35}$$

Reorder terms and substitute the bound $L_{\boldsymbol{\lambda}}$ for the norm of the dual subgradient of the online Lagrangian given in (2.27) to write

$$\nabla_{\boldsymbol{\lambda}}\mathcal{O}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)^T(\boldsymbol{\lambda}_t-\boldsymbol{\lambda}) \geq \frac{1}{2\eta}\left(\|\boldsymbol{\lambda}_{t+1}-\boldsymbol{\lambda}\|^2-\|\boldsymbol{\lambda}_t-\boldsymbol{\lambda}\|^2\right)-\frac{\eta}{2}L_{\boldsymbol{\lambda}}^2. \tag{2.36}$$

Note that the online Lagrangian [cf. (2.12)] is a linear function of its Lagrange multipliers, which implies that online Lagrangian differences for fixed $\mathbf{w}_t$ satisfy

$$\mathcal{O}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)-\mathcal{O}_t(\mathbf{w}_t,\boldsymbol{\lambda}) \geq \nabla_{\boldsymbol{\lambda}}\mathcal{O}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)^T(\boldsymbol{\lambda}_t-\boldsymbol{\lambda}). \tag{2.37}$$

Substitute the lower bound (2.36) into the right hand side of (2.37) to obtain

$$\mathcal{O}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)-\mathcal{O}_t(\mathbf{w}_t,\boldsymbol{\lambda}) \geq \frac{1}{2\eta}\left(\|\boldsymbol{\lambda}_{t+1}-\boldsymbol{\lambda}\|^2-\|\boldsymbol{\lambda}_t-\boldsymbol{\lambda}\|^2\right)-\frac{\eta}{2}\,L_{\boldsymbol{\lambda}}^2. \tag{2.38}$$

We now turn to combining the bounds in (2.33) and (2.38). To do so observe that the term $\mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)$ appears in both inequalities. Thus, subtraction of the terms in inequality (2.38) from those in (2.33) followed by reordering terms yields

$$\begin{aligned}\mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}) - \mathcal{O}_t(\mathbf{w}, \boldsymbol{\lambda}_t) \\ \leq \frac{1}{2\eta}\left(\|\mathbf{w}_t - \mathbf{w}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}\|^2 + \|\boldsymbol{\lambda}_t - \boldsymbol{\lambda}\|^2 - \|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}\|^2\right) + \frac{\eta}{2}\left(L_{\mathbf{w}}^2 + L_{\boldsymbol{\lambda}}^2\right).\end{aligned} \tag{2.39}$$

Now sum (2.39) over time to write

$$\sum_{t=1}^{T} \mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}) - \mathcal{O}_t(\mathbf{w}, \boldsymbol{\lambda}_t) \leq \frac{1}{2\eta}\left(\|\mathbf{w}_1 - \mathbf{w}\|^2 + \|\boldsymbol{\lambda}\|^2\right) + \frac{\eta}{2} T(L_{\mathbf{w}}^2 + L_{\boldsymbol{\lambda}}^2). \tag{2.40}$$

Here we have used the telescopic property of the summand on the right hand side of (2.40) and omitted the subtraction of the nonnegative quantity $\|\boldsymbol{\lambda}_T - \boldsymbol{\lambda}\|^2$. Using the explicit expression for the online Lagrangian in (2.12) we can write the online Lagrangian difference on the left side of (2.39) as

$$\mathcal{O}_t(\mathbf{w}_t, \boldsymbol{\lambda}) - \mathcal{O}_t(\mathbf{w}, \boldsymbol{\lambda}_t) = \sum_{i=1}^{V} \ell_{i,t}(\mathbf{w}_{i,t}) + \boldsymbol{\lambda}^T \mathbf{C}\mathbf{w}_t - \sum_{i=1}^{V} \ell_{i,t}(\mathbf{w}) - \boldsymbol{\lambda}_t^T \mathbf{C}\mathbf{w}. \tag{2.41}$$

Let now $\mathbf{w}$ be an arbitrary feasible point for the coordinated regret game, i.e., one for which $\mathbf{w}_i = \mathbf{w}_j$ for all i and j, or, equivalently, one for which $\mathbf{C}\mathbf{w} = \mathbf{0}$. For these feasible points the last term in (2.41) vanishes. Substituting the resulting expression for the left hand side of (2.40) yields, after reordering terms,

$$\sum_{t=1}^{T}\sum_{i=1}^{V} \left(\ell_{i,t}(\mathbf{w}_{i,t}) - \ell_{i,t}(\mathbf{w})\right) + \sum_{t=1}^{T} \boldsymbol{\lambda}^T \mathbf{C}\mathbf{w}_t \leq \frac{1}{2\eta}\left(\|\mathbf{w}_1 - \mathbf{w}\|^2 + \|\boldsymbol{\lambda}\|^2\right) + \frac{\eta}{2} T(L_{\mathbf{w}}^2 + L_{\boldsymbol{\lambda}}^2), \tag{2.42}$$

for arbitrary feasible point $\mathbf{w}$ satisfying $\mathbf{C}\mathbf{w} = \mathbf{0}$. The bound in (2.42) holds for $\tilde{\mathbf{w}}^*$ because $\tilde{\mathbf{w}}^*$ is optimal for coordinated regret – thus feasible, in particular. The result in (2.28) follows by making $\mathbf{w} = \tilde{\mathbf{w}}^*$ and $\eta = 1/\sqrt{T}$ in (2.42).

■

From Lemma 1 we obtain a bound for the uncoordinated regret $\mathbf{Reg}_T^{\mathrm{U}}$ defined in (2.5). To do so simply note that $\boldsymbol{\lambda} = \mathbf{0}$ belongs to the set Λ. Using this particular value of $\boldsymbol{\lambda}$ in (2.28) yields

$$\mathbf{Reg}_T^{\mathrm{U}} = \sum_{t=1}^{T}\sum_{j=1}^{V} \ell_{j,t}(\mathbf{w}_{j,t}) - \sum_{t=1}^{T}\sum_{i=1}^{V} \ell_{j,t}(\tilde{\mathbf{w}}^*) \leq \frac{\sqrt{T}}{2}\left(\|\mathbf{w}_1 - \mathbf{w}\|^2 + L_{\mathbf{w}}^2 + L_{\boldsymbol{\lambda}}^2\right). \tag{2.43}$$

This bound is of little use because, as we mentioned in Section 2.1, agents can reduce uncoordinated regret by just operating independently of each other. Observe, however, that the relationship in (2.28) also includes the weighted penalty disagreement $\sum_{t=1}^{T} \boldsymbol{\lambda}^T \mathbf{C}\mathbf{w}_t$. The presence of this term indicates that the actions of different users can't be too different and that it should be possible to relate global networked regret to uncoordinated regret. This is indeed possible and leads to the regret bound that we introduce in the following theorem.

Theorem 1 *Let* $\mathbf{w}_t := [\mathbf{w}_{1,t}; \ldots; \mathbf{w}_{V,t}]$ *denote the sequence generated by the saddle point algorithm in* (2.17)-(2.18) *and let* $\tilde{\mathbf{w}}^*$ *be the optimal offline action in* (2.6)*. If Assumptions 1-5 hold, with the initialization* $\boldsymbol{\lambda}_1 = \mathbf{0}$ *and step size* $\eta = 1/\sqrt{T}$*, the global network regret [cf.* (2.7)*] is bounded by*

$$\mathbf{Reg}_T \leq \frac{\sqrt{T}}{2}\big(\|\mathbf{w}_1 - \tilde{\mathbf{w}}^*\|^2 + MC_{\boldsymbol{\lambda}}^2 + L_{\mathbf{w}}^2 + L_{\boldsymbol{\lambda}}^2\big) \;=\; O(\sqrt{T}). \tag{2.44}$$

Proof: We begin by writing the expression for $\mathbf{Reg}_T$, and add and subtract the left hand side of (2.28), the uncoordinated regret plus a constraint slack penalizing node disagreement to write

$$\begin{aligned}
\mathbf{Reg}_T \;&=\; \sum_{t=1}^{T} \frac{1}{V} \sum_{j,k=1}^{V} \ell_{k,t}(\mathbf{w}_{j,t}) - \sum_{t=1}^{T}\sum_{k=1}^{V} \ell_{k,t}(\tilde{\mathbf{w}}^*) \\
&= \sum_{t=1}^{T} \left(\frac{1}{V} \sum_{j,k=1}^{V} \ell_{k,t}(\mathbf{w}_{j,t}) - \sum_{k=1}^{V} \ell_{k,t}(\mathbf{w}_{k,t}) - \boldsymbol{\lambda}^T \mathbf{C}\mathbf{w}_t \right) \\
&\qquad + \sum_{t=1}^{T} \left(\sum_{k=1}^{V} \ell_{k,t}(\mathbf{w}_{k,t}) - \sum_{k=1}^{V} \ell_{k,t}(\tilde{\mathbf{w}}^*) + \boldsymbol{\lambda}^T \mathbf{C}\mathbf{w}_t \right).
\end{aligned} \tag{2.45}$$

The second time summation on the right side of (2.45) may be bounded with Lemma 1. Thus we turn to providing an upper estimate of the first sum. Assumption 3 regarding the Lipschitz continuity of the loss functions implies

$$\sum_{j,k=1}^{V} \Big[\ell_{k,t}(\mathbf{w}_{j,t}) - \ell_{k,t}(\mathbf{w}_{k,t}) \Big] \leq \sum_{j,k=1}^{V} K_{k,t} \|\mathbf{w}_{j,t} - \mathbf{w}_{k,t}\|. \tag{2.46}$$

Maximize over the right hand side of (2.46) to obtain an expression for the magnitude of the worst case node discrepancy

$$\sum_{j,k=1}^{V} K_{k,t} \|\mathbf{w}_{j,t} - \mathbf{w}_{k,t}\| \leq V^2 K \max_{j,k} \|\mathbf{w}_{j,t} - \mathbf{w}_{k,t}\|. \tag{2.47}$$

Using Assumption 1 regarding the diameter of the network, the worst case node discrepancy on the right hand side of (2.47) may be bounded above by the magnitude of the constraint slack as $\max_{j,k} \|\mathbf{w}_{j,t} - \mathbf{w}_{k,t}\| \leq D\|\mathbf{C}\mathbf{w}_t\|$. Substituting this bound into (2.47) yields

$$\sum_{j,k=1}^{V} \Big[\ell_{k,t}(\mathbf{w}_{j,t}) - \ell_{k,t}(\mathbf{w}_{k,t})\Big] \leq DV^2K\|\mathbf{C}\mathbf{w}_t\|. \tag{2.48}$$

We return to bounding the first sum in the right hand side of (2.45). To do so write $\sum_{k=1}^{V} \ell_{k,t}(\mathbf{w}_{k,t}) = (1/N)\sum_{j,k=1}^{V} \ell_{k,t}(\mathbf{w}_{k,t})$, which we can do because $\ell_{k,t}(\mathbf{w}_{k,t})$ is independent of j. Use this to substitute $(1/N)\sum_{j,k=1}^{V} \ell_{k,t}(\mathbf{w}_{j,t}) - \sum_{k=1}^{V} \ell_{k,t}(\mathbf{w}_{k,t})$ for the bound in (2.48) to write

$$\begin{aligned}\sum_{t=1}^{T}\left(\frac{1}{V}\sum_{j,k=1}^{V}\Big[\ell_{k,t}(\mathbf{w}_{j,t}) - \ell_{k,t}(\mathbf{w}_{k,t})\Big] - \boldsymbol{\lambda}^T\mathbf{C}\mathbf{w}_t\right) \qquad (2.49)\\ \leq \sum_{t=1}^{T}\left(DVK\|\mathbf{C}\mathbf{w}_t\| - \boldsymbol{\lambda}^T\mathbf{C}\mathbf{w}_t\right) = \sum_{t=1}^{T}\left(DVK\frac{\mathbf{C}\mathbf{w}_t}{\|\mathbf{C}\mathbf{w}_t\|} - \boldsymbol{\lambda}\right)^T\mathbf{C}\mathbf{w}_t.\end{aligned}$$

where the last equality follows from grouping terms. The difference between node losses evaluated at other nodes' predictions and their own is bounded by the magnitude of the constraint violation and a Lagrangian penalty term. We annihilate the right hand side of (2.49) by constructing a dual feasible $\tilde{\boldsymbol{\lambda}}$ as follows. Partition the edge set $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ with $\mathcal{E}^+ = \{e : [\sum_{t=1}^{T} \mathbf{C}[\mathbf{w}_t]_e \geq \mathbf{0}\}$. and $\mathcal{E}^- = \{e : [\sum_{t=1}^{T}[\mathbf{C}\mathbf{w}_t]_e < \mathbf{0}\}$. Define $\tilde{\boldsymbol{\lambda}}$ as

$$\tilde{\boldsymbol{\lambda}}_e \leq \begin{cases} DVK[\mathbf{C}\mathbf{w}_t]_e/\|\mathbf{C}\mathbf{w}_t\| + 1 & \text{for } e \in \mathcal{E}^+ \text{ and all } t \\ DVK[\mathbf{C}\mathbf{w}_t]_e/\|\mathbf{C}\mathbf{w}_t\| - 1 & \text{for } e \in \mathcal{E}^- \text{ and all } t \end{cases} \tag{2.50}$$

We may construct a fixed finite $\tilde{\boldsymbol{\lambda}}$ follows from the compactness of X and hence the boundedness of $\|\mathbf{C}\mathbf{w}_t\|$. Note that

$$\|\tilde{\boldsymbol{\lambda}}\| \leq \sqrt{|\mathcal{E}^+|(DVK+1)^2 + |\mathcal{E}^-|(DVK-1)^2} \leq \sqrt{E}(DVK+1) \leq \sqrt{E}C_{\boldsymbol{\lambda}}. \tag{2.51}$$

so $\tilde{\boldsymbol{\lambda}}$ is dual feasible in the sense of (2.24). The first inequality in (2.51) follows from the fact that computing $\|\tilde{\boldsymbol{\lambda}}\|$ is a sum over the entries of a unit vector, while the second inequality

uses the relationship $|\mathcal{E}^+| - |\mathcal{E}^-| \leq M$. Now plug $\boldsymbol{\lambda} = \tilde{\boldsymbol{\lambda}}$ into (2.49) to write

$$\begin{aligned}
&\sum_{t=1}^{T} \left(DVK \frac{\mathbf{C}\mathbf{w}_t}{\|\mathbf{C}\mathbf{w}_t\|} - \tilde{\boldsymbol{\lambda}} \right)^T \mathbf{C}\mathbf{w}_t \qquad (2.52) \\
&\quad \leq \sum_{t=1}^{T} \sum_{e \in \mathcal{E}^+} \left(DVK \frac{[\mathbf{C}\mathbf{w}_t]_e}{\|\mathbf{C}\mathbf{w}_t\|} - DVK \frac{[\mathbf{C}\mathbf{w}_t]_e}{\|\mathbf{C}\mathbf{w}_t\|} - 1 \right) [\mathbf{C}\mathbf{w}_t]_e \\
&\qquad + \sum_{t=1}^{T} \sum_{e \in \mathcal{E}^-} \left(DVK \frac{[\mathbf{C}\mathbf{w}_t]_e}{\|\mathbf{C}\mathbf{w}_t\|} - DVK \frac{[\mathbf{C}\mathbf{w}_t]_e}{\|\mathbf{C}\mathbf{w}_t\|} + 1 \right) [\mathbf{C}\mathbf{w}_t]_e \\
&\quad = \sum_{t=1}^{T} \left(\sum_{e \in \mathcal{E}^+} -[\mathbf{C}\mathbf{w}_t]_e + \sum_{e \in \mathcal{E}^-} [\mathbf{C}\mathbf{w}_t]_e \right) = 0.
\end{aligned}$$

With the dual variable selection given by (2.50), we have made first three terms on the right hand side of (2.45) null. Now apply Lemma 1 to the last three terms on the right hand side of (2.45) and substitute in the bound for the magnitude of $\tilde{\boldsymbol{\lambda}}$ in (2.51), which allows us to conclude (2.44). ■

Theorem 1 provides a guarantee that the saddle point iterates achieve a global networked regret that grows not faster than $O(\sqrt{T})$. This rate is the same that can be guaranteed in centralized problems when functions are not strongly convex. The learning rate depends on primal initialization, network size and topology, as well as smoothness properties of the loss functions. The learning rate result established in Theorem 1 is a bound on the global networked regret which is the average the local regrets incurred by each agent. By relating the uncoordinated regret bound in (2.43) with the local regret defined in (2.6) we obtain a similar bound on the regret of each individual agent as we formally state next.

Theorem 2 *Let* $\mathbf{w}_t := [\mathbf{w}_{1,t}; \ldots; \mathbf{w}_{V,t}]$ *be the sequence generated by the saddle point algorithm in* (2.17)-(2.18) *and let* $\tilde{\mathbf{w}}^*$ *be the global batch learner in* (2.6)*. If Assumptions 1-5 hold, with the initialization* $\boldsymbol{\lambda}_1 = \mathbf{0}$ *and step size* $\eta = 1/\sqrt{T}$*, the local regret of node* j *[cf.* (2.6)*] is bounded by*

$$\mathbf{Reg}_T^j \leq \frac{\sqrt{T}}{2} \big(\|\mathbf{w}_1 - \tilde{\mathbf{w}}^*\|^2 + M C_{\boldsymbol{\lambda}}^2 + L_{\mathbf{w}}^2 + L_{\boldsymbol{\lambda}}^2 \big) \;=\; O(\sqrt{T}). \qquad (2.53)$$

Proof: Begin writing the expression for local regret of node j and add and subtract the

left hand side of (2.43).

$$\begin{aligned}\mathbf{Reg}_T^j &= \sum_{t=1}^{T}\sum_{k=1}^{V}\ell_{k,t}(\mathbf{w}_{j,t}) - \sum_{t=1}^{T}\sum_{k=1}^{V}\ell_{k,t}(\tilde{\mathbf{w}}^*) \\ &= \sum_{t=1}^{T}\left(\sum_{k=1}^{V}\left[\ell_{k,t}(\mathbf{w}_{j,t}) - \ell_{k,t}(\mathbf{w}_{k,t})\right] - \boldsymbol{\lambda}^T\mathbf{C}\mathbf{w}_t\right) \\ &\quad + \sum_{t=1}^{T}\left(\sum_{k=1}^{V}\left[\ell_{k,t}(\mathbf{w}_{k,t}) - \ell_{k,t}(\tilde{\mathbf{w}}^*)\right] + \boldsymbol{\lambda}^T\mathbf{C}\mathbf{w}_t\right)\end{aligned} \tag{2.54}$$

The last three terms of (2.54) were bounded in Lemma 1, so we turn our focus to the first tree terms in an analogous manner to the proof of Theorem 1. The Lipschitz continuity of the losses in Assumption 3 yields

$$\sum_{k=1}^{V}\left(\ell_{k,t}(\mathbf{w}_{j,t}) - \ell_{k,t}(\mathbf{w}_{k,t})\right) \leq \sum_{k=1}^{V} K_{k,t}\|\mathbf{w}_{j,t} - \mathbf{w}_{k,t}\|. \tag{2.55}$$

Now, maximize over the right hand side of (2.55) to write an expression for the maximum difference between node predictions

$$\sum_{k=1}^{V} K_{k,t}\|\mathbf{w}_{j,t} - \mathbf{w}_{k,t}\| \leq VK \max_k \|\mathbf{w}_{j,t} - \mathbf{w}_{k,t}\|. \tag{2.56}$$

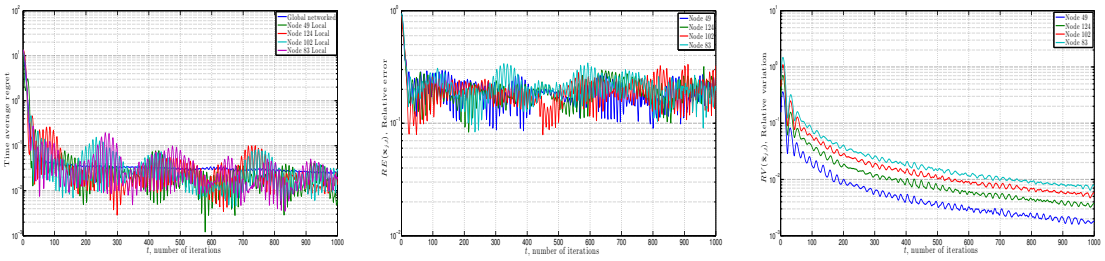
The quantity on the right hand side of (2.56) can be expressed in terms of the magnitude of the constraint violation. In particular, the definition of the diameter as the maximum of shortest paths between nodes combined with the triangle inequality allows us to write

$$VK \max_k \|\mathbf{w}_{j,t} - \mathbf{w}_{k,t}\| \leq DVK\|\mathbf{C}\mathbf{w}_t\|. \tag{2.57}$$

We substitute in the right hand side of (2.57) into (2.55), and apply the resulting inequality to the first three terms on the right hand side of (2.54) to obtain

$$\begin{aligned}&\sum_{t=1}^{T}\left(\sum_{k=1}^{V}\left[\ell_{k,t}(\mathbf{w}_{j,t}) - \ell_{k,t}(\mathbf{w}_{k,t})\right] - \boldsymbol{\lambda}^T\mathbf{C}\mathbf{w}_t\right) \\ &\qquad \leq \sum_{t=1}^{T}\left(DVK\|\mathbf{C}\mathbf{w}_t\| - \boldsymbol{\lambda}^T\mathbf{C}\mathbf{w}_t\right) = \sum_{t=1}^{T}\left(DVK\frac{\mathbf{C}\mathbf{w}_t}{\|\mathbf{C}\mathbf{w}_t\|} - \boldsymbol{\lambda}\right)^T\mathbf{C}\mathbf{w}_t.\end{aligned} \tag{2.58}$$

Using $\boldsymbol{\lambda} = \tilde{\boldsymbol{\lambda}}$ defined in (2.50), we make the right hand side of (2.58) null in precisely the same manner as (2.52). Returning to the first three terms of (2.54), we apply Lemma 1 and substitute in the expression for the magnitude of $\tilde{\boldsymbol{\lambda}}$ in (2.51) to yield (2.53). ■



(a) Average regret vs. iter. t (b) Rel. error vs. iter. t (c) Relative var. vs. iter. t

Figure 2.1: In a $V = 200$ node random network with connection probability $\rho = 0.2$, figures 2.1(a)-2.1(b) show average global networked regret $\mathbf{Reg}_T/T$ and local regrets $\mathbf{Reg}_t^j/t$ for a representative sample of agents, and $\mathrm{RE}(\mathbf{w}_{j,t})$, respectively, versus iteration t. Average regrets sharply decline and then stabilize, consistent with the regret bounds dependence on a fixed step size $\epsilon = 1/\sqrt{T}$. Decentralized online learning is corroborated by both distance to the global batch learner, measured by $\mathrm{RE}(\mathbf{w}_{j,t})$, decreasing, as shown in 2.1(b), and consensus in the primal variable, which is shown in Figure 2.2(c). In the later, we plot $\mathrm{RV}(\bar{\mathbf{w}}_{j,t})$ versus iteration t, which goes to null as agents learn all information available throughout the network.

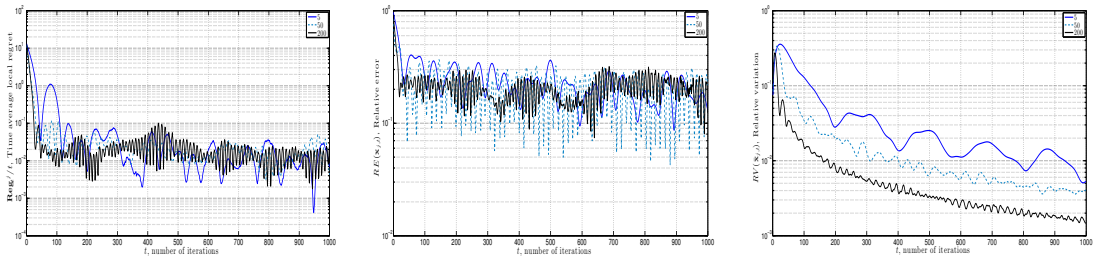
Theorem 2 establishes that the local regret of each individual agent in the network grows at a rate not larger than $O(\sqrt{T})$, which is equivalent to saying that its time average vanishes as $O(1/\sqrt{T})$. It follows that individuals learn global information while only having access to local observations and the strategies of neighboring agents. The constants that bound the regret growth depend on the initial condition, network connectivity, and properties of the loss functions.

2.4 Empirical Regret Performance

We study the numerical behavior of the saddle point algorithm in (2.17)-(2.18) when used to solve the distributed recursive least squares problem in Section 2.1.1 for a variety of network sizes, topologies, and levels of connectivity (Sections 2.4.1 - 2.4.3). We also investigate how saddle point iterates compare against other networked online learning methods (Section 2.4.4). The primal iteration for recursive least squares is given by the explicit expression in (2.19). Besides the local and global network regrets in (2.6) and (2.7) that we know grow not faster than $O(\sqrt{T})$ [cf. theorems 1 and 2] we also study the relative error of the estimates $\mathbf{w}_{j,t}$ relative to the optimal batch estimator $\tilde{\mathbf{w}}^*$ and the relative agreement between estimates $\mathbf{w}_{j,t}$ and $\mathbf{w}_{k,t}$ of different agents. The relative error associated with the estimate $\mathbf{w}_{j,t}$ of agent j at time t is defined as

$$\mathrm{RE}(\mathbf{w}_{j,t}) := \frac{\|\mathbf{w}_{j,t} - \tilde{\mathbf{w}}^*\|}{\|\tilde{\mathbf{w}}^*\|}. \tag{2.59}$$

The agreement between estimates of different nodes is defined in terms of the variable time averages $\bar{\mathbf{w}}_{j,t} := (1/t)\sum_{u=1}^{t} \mathbf{w}_{j,u}$. For the average estimate $\bar{\mathbf{w}}_{j,t}$ of agent j at time t we



(a) Average regret vs. iter. t (b) Relative error vs. iter. t (c) Relative variation vs. iter. t

Figure 2.2: Learning achieved by an arbitrary agent in networks of size $V = 5$, $V = 50$, and $V = 200$ with nodes randomly connected with prob. $\rho = 0.2$. 2.2(a)-2.2(b) show $\mathbf{Reg}_t^j/t$, the time average local regret, and $\mathrm{RE}(\mathbf{w}_{j,t})$, the relative error, respectively, as compared with iteration t. Both $\mathbf{Reg}_t^j/t$ and $\mathrm{RE}(\mathbf{w}_{j,t})$ decline sharply, but with more instability in smaller networks, and stabilize near 10^{-2} and 10^{-1}, respectively. Figure 2.2(c) shows $RV(\bar{\mathbf{w}}_{j,t})$ versus iteration t, and illustrate that node j's average prediction remain close to that of other nodes. Network disagreement becomes more stable and declines faster with increasing V, as information contained per individual required for learning the global batch strategy declines.

define the average relative variation as

$$\mathrm{RV}(\bar{\mathbf{w}}_{j,t}) := \frac{1}{V}\sum_{k=1}^{V}\frac{\|\bar{\mathbf{w}}_{j,t} - \bar{\mathbf{w}}_{k,t}\|}{\|\tilde{\mathbf{w}}^*\|}. \tag{2.60}$$

The average relative variation $\mathrm{RV}(\bar{\mathbf{w}}_{j,t})$ denotes the average Euclidean error between $\bar{\mathbf{w}}_{j,t}$ and all others, relative to the magnitude of the offline strategy $\tilde{\mathbf{w}}^*$. The reason to focus on time averages $\bar{\mathbf{w}}_{j,t}$ instead of the plain estimates $\mathbf{w}_{j,t}$ is that the latter tend to oscillate around the batch estimate $\tilde{\mathbf{w}}^*$ and agreement between estimates of different agents is difficult to visualize.

For all of the subsequent numerical experiments, we consider $q = 1$ and $p = 10$ – i.e., observations $\mathbf{y}_{it} = \mathbf{H}_{i,t}\tilde{\mathbf{w}} + \mathbf{w}_{i,t}$ are scalar and the signal $\tilde{\mathbf{w}}$ has dimension $p = 10$. The matrices $\mathbf{H}_{i,t} = \mathbf{H}_i \in \mathbb{R}^{1\times p}$ are constant across time but vary across agents. The components of the vector $\mathbf{H}_i$ are chosen with equal probability from $\{1/p, 2/p, \ldots, 1\}$. The random noise terms $\mathbf{w}_{i,t} \in \mathbb{R}$ are Gaussian distributed with zero mean and variance $\sigma^2 = 0.1$ and the true signal is $\tilde{\mathbf{w}} = \mathbf{1}$. Further observe that since $q < p$ it is impossible to estimate $\tilde{\mathbf{w}}$ without cooperation between members of the network because the individual signals of each agent are not sufficient to determine $\tilde{\mathbf{w}}$. In all cases we run (2.19) - (2.18) for a total of $T = 10^3$ iterations with step size $\epsilon = 1/\sqrt{T} = 0.03$. Agents initialize as $\mathbf{w}_{j,1} = \mathbf{0}$ for all j and $\boldsymbol{\lambda}_{jk,1} = \mathbf{0}$ for all (j, k).

The trajectories of a sample run for a random network with $V = 200$ nodes in which the probability of connecting two nodes is $\rho = 0.2$ are shown in Figure 2.1. The time average of the global and local regrets, $\mathbf{Reg}_t/t$ and $\mathbf{Reg}_t^j/t$, respectively, for representative nodes are shown in Figure 2.1(a). Observe that $\mathbf{Reg}_t/t$ decreases until $t \approx 200$ iterations and

then stabilizes at $\mathbf{Reg}_t/t \approx 5 \times 10^{-2}$. This is consistent with the result in Theorem 1 in which regret of order $O(\sqrt{T})$ is attained by selecting a stepsize of order $O(T)$. To obtain smaller regret values the algorithm has to be run with smaller stepsize. The same decline is observed for the average local regrets $\mathbf{Reg}_t^j/t$. The only difference is that the $\mathbf{Reg}_t^j/t$ exhibit oscillating variations as iterations progress. These are not present in $\mathbf{Reg}_t/t$ which averages values across the whole network.

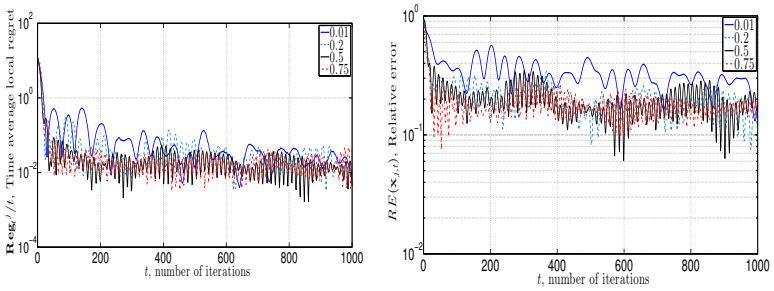
Learning of the global batch strategy can be corroborated by reduction of the Euclidean distance to $\tilde{\mathbf{w}}^*$ at each node and the achievement of primal variable consensus. Figure 2.1(b) shows that the relative error $\mathrm{RE}(\mathbf{w}_{j,t})$ declines with the iteration index t and stabilizes below 0.4 for $t \geq 100$, demonstrating that the former goal is achieved, though the noise in the observations yields persistent oscillations. Figure 2.1(c) plots $\mathrm{RV}(\bar{\mathbf{w}}_{j,t})$ versus iteration t. Observe that agents also converge towards a common value, i.e. $\mathrm{RV}(\bar{\mathbf{w}}_{j,t}) \leq 10^{-2}$ for $t \geq 700$, as exchange of local information successfully allows agents to learn a globally optimal strategy.

2.4.1 Network size

To investigate the dependence of the learning rates in theorems 1 and 2 with the network size V we run (2.19) - (2.18) for problem instances with $V = 5$, $V = 50$, and $V = 200$ nodes. Connections between nodes are random, with the probability of two nodes being connected set to $\rho = 0.2$. Figure 2.2 shows the results of this numerical experiment for an arbitrary agent in the network. In Figure 2.2(a), we show $\mathbf{Reg}_t^j/t$ over iteration t. Observe that as V increases, $\mathbf{Reg}_t^j/t$ declines at comparable rates for the different network sizes, and this rate similarity is also reflected in the trajectory of $\mathrm{RE}(\mathbf{w}_{j,t})$ over time t, as shown in Figure 2.2(b). To reach the benchmark $\mathrm{RE}(\mathbf{w}_{j,t}) \leq 10^{-3}$ we require $t = 354$, $t = 279$, and $t = 232$ for $V = 5$, $V = 50$, and $V = 200$, respectively.

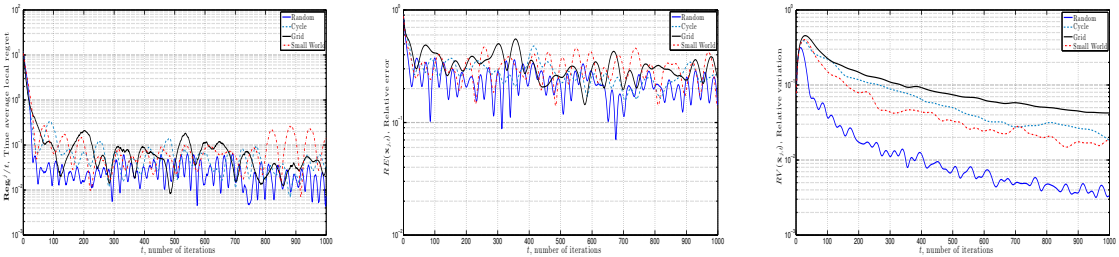
While learning occurs at comparable rates in the different networks, the trajectories are more oscillatory in smaller networks. To be specific, in Figure 2.2(b). we note that to achieve $\mathrm{RE}(\mathbf{w}_{j,t}) \leq 0.2$, the algorithm requires $t = 106$, $t = 30$, and $t = 18$ iterations for $V = 5$, $V = 50$, and $V = 200$, respectively. Moreover, average wavelength of oscillations of $\mathrm{RE}(\mathbf{w}_{j,t})$ is $\tau = 50$, $\tau = 20$, and $\tau = 10$ for $V = 5$, $V = 50$, and $V = 200$, respectively. This stability difference reflects the fact that as V increases, the fraction of information per node contained in each agent's prediction decreases. Hence each dual variable must compensate for a larger relative level of discrepancy per communication link in smaller networks. Equivalently, for agents in larger networks to achieve comparable learning rates, information must diffuse faster.

Figure 2.2(c) shows that the network reaches consensus, as measured with $\mathrm{RV}(\bar{\mathbf{w}}_{j,t})$, faster with larger V. In particular, for the benchmark $\mathrm{RV}(\bar{\mathbf{w}}_{j,t}) \leq 10^{-2}$, the algorithm



(a) Average regret vs. t (b) Relative error vs. iteration t (c) Relative var. $\text{RV}(\mathbf{w}_{j,t})$ vs. t

Figure 2.3: Saddle point algorithm learning rates and discrepancy on a random $V = 50$ node network with connection probability $\rho \in \{0.01, 0.2, 0.5, 0.75\}$. Figure 2.3(a)-2.3(b) show $\mathbf{Reg}_T^j/T$, the time average local regret of an arbitrary node in the network, and $\text{RE}(\mathbf{w}_{j,t})$, the relative error, respectively, as compared with iteration t. Both $\mathbf{Reg}_T^j/T$ and $\text{RE}(\mathbf{w}_{j,t})$ are more oscillatory in less connected networks. Figure 2.3(c) shows $RV(\bar{\mathbf{w}}_{j,t})$ versus iteration t. Primal variable consensus is more difficult to achieve in networks with fewer communication links.



(a) Average regret vs. iter. t (b) Relative err. vs. iter. t (c) Relative var. vs. iter t

Figure 2.4: Saddle point algorithm run on $V = 50$ node cycle, grid, random and small world networks, where edges are generated randomly between agents with probability $\rho = 0.2$ in the later two. Model noise is sampled from $\mathbf{w}_{i,t} \sim \mathcal{N}(0, 0.1)$. Figure 2.4(a)-2.4(b) show $\mathbf{Reg}_T^j/T$ and $\text{RE}(\mathbf{w}_{j,t})$, respectively, over iteration t. Learning slows and numerical oscillations become more prevalent with increasing network diameter. Grid and cycle networks have larger diameter than small world and random networks, resulting in slower information propagation. Figure 2.4(c) shows that the agents reach consensus slower in terms of $\text{RV}(\bar{\mathbf{w}}_{j,t})$ with increasing network diameter.

requires $t = 719$, $t = 317$, and $t = 179$ iterations for $V = 5$, $V = 50$, and $V = 200$ node networks, respectively. This suggests that the agreement constraint plays a larger role in maintaining a comparable learning rate in larger networks, as the relative variation must be smaller for individuals to learn global information in larger networks.

2.4.2 Node connectivity

To understand the impact of network connectivity on algorithm performance we fix the network size to $V = 50$ and run (2.19) - (2.18) on random networks where the probability of connecting two nodes takes values $\rho \in \{0.01, 0.2, 0.5, 0.75\}$. Figure 2.3 shows the results of this experimental setup. Figure 2.3(a) depicts $\mathbf{Reg}_t^j/t$ versus iteration t, and illustrates that the difference in connectivity levels leads to a negligible difference in the learning rate. However, we see that numerical stability varies substantially. The sparsely connected networks experience more oscillatory behavior, as may be observed in the plot of of relative error versus iteration t in Figure 2.3(b). This stability difference follows from the slower rate of information diffusion, and also coincides with slowing convergence to the batch strategy. Figure 2.3(c) shows the evolution of $\text{RV}(\bar{\mathbf{w}}_{j,t})$ over time. The achievement of primal variable consensus is more challenging in sparsely connected networks. That is, for the benchmark $\text{RV}(\bar{\mathbf{w}}_{j,t}) \leq 2 \times 10^{-2}$, the algorithm requires $t = 875$, $t = 183$, $t = 120$, and $t = 43$ iterations for the cases $\rho = 0.01$, $\rho = 0.2$, $\rho = 0.5$, and $\rho = 0.75$, respectively. Intuitively, the discrepancy in agents' predictions is smaller when more communication links are present in the network.

2.4.3 Topology and Diameter

To study the interplay of network topology and diameter on the learning rates established in Theorems 1 and 2, we fix the network size to $V = 50$ and run (2.19) - (2.18) over random graphs, small world graphs, cycles, and grids. In the first two, the probability that node pairs are randomly connected is fixed at $\rho = 0.2$. The latter two are deterministically generated. A cycle is a closed directed chain of nodes. Grids are formed by taking the two-dimensional integer lattice of size $\sqrt{N} \times \sqrt{N}$, with $\sqrt{N}$ rounded to the nearest integer. Connections are drawn between adjacent nodes in the lattice as well as between remainder nodes at the boundary. Cycles, grids and random networks have progressively larger number of connections per node and smaller diameter. Random networks have small degree and small diameter; see [208, 211].

We present the results of this experiment in Figure 2.4. In Figure 2.4(a), we plot $\mathbf{Reg}_t^j/t$ compared with iteration t. Observe that the rate at which $\mathbf{Reg}_t^j/t$ decreases is comparable across the different networks, yet we can differentiate the learning achieved in the different settings by the benchmark $\mathbf{Reg}_t^j/t \leq 10^{-2}$. To surpass this bound, the algorithm requires

$t = 293$, $t = 221$ iterations for random and small world networks, respectively, whereas for grids and cycles it requires $t = 483$, $t = 865$ iterations. This indicates that structured deterministic networks are a more difficult setting for networked online learning, and the randomness present in random and small world networks allows more effective information flow.

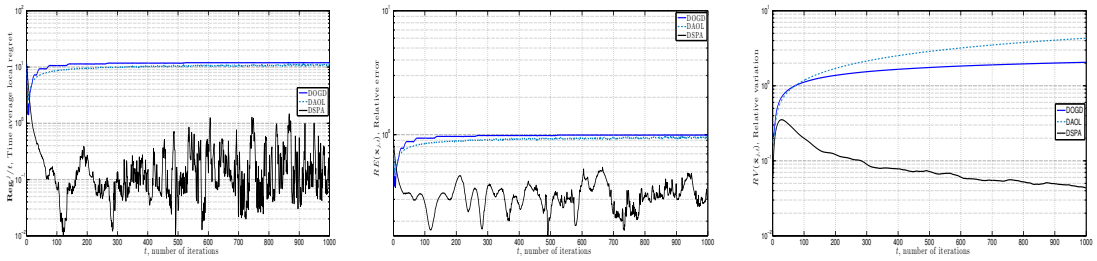
In the plot of $\mathrm{RE}(\mathbf{w}_{j,t})$ over time t shown in Fig 2.4(b), we see a slower rate of convergence towards the batch learner in the structured deterministic networks: $\mathrm{RE}(\mathbf{w}_{j,t}) \leq 0.2$ requires $t = 81$, $t = 176$, $t = 556$, and $t = 578$ iterations for random, small world, grid, and cycle networks, respectively, which validates the relationship observed in Figure 2.4(a). We observe this rate difference more readily in Fig 2.4(c), which plots $\mathrm{RV}(\bar{\mathbf{w}}_{j,t})$ over time t. To obtain $\mathrm{RV}(\bar{\mathbf{w}}_{j,t}) \leq 5 \times 10^{-2}$, the algorithm requires $t = 49$, $t = 301$, $t = 809$, and $t = 525$ iterations respectively for random, small world, grid, and cycle networks. These experiments indicate that information propagation slows in large diameter networks, causing more numerical oscillations and decreasing the learning rate. Put another way, networks in which agents may communicate more effectively reach consensus.

2.4.4 Algorithm Comparison

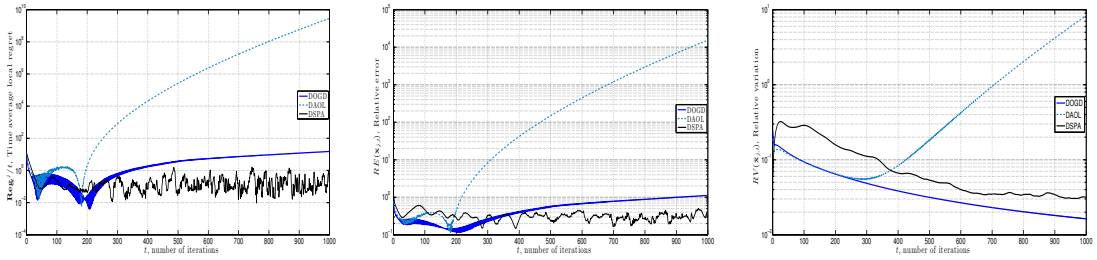
We turn to comparing the saddle point method against other recent works in networked online convex optimization. To that end, we consider grid and cycle topologies with $V = 50$ agents. We implement Distributed Online Gradient Descent (DOGD) [197], and Distributed Autonomous Online Learning (DAOL) [219]. Both of these are consensus protocols based on an iterative weighted averaging process. The difference between these two methods is that DOGD performs many gradient averaging steps per node update, whereas DAOL executes only one.

Figure 2.5 shows the results of this comparison. In figures 2.5(a) and 2.5(d), we plot $\mathbf{Reg}_t^j/t$ versus the time t on a grid and cycle network, respectively. Both DOGD and DAOL fail to achieve learning: for all $t \geq 100$, $\mathbf{Reg}_t^j/t \approx 10$ in the grid network. Moreover, in the cycle case $\mathbf{Reg}_t^j/t \approx 10$ for all $t \geq 500$ for DOGD, while DAOL suffers unbounded regret as t increases. On the other hand, $\mathbf{Reg}_t^j/t \leq 5 \times 10^{-2}$ for $t \geq 580$ for the saddle point algorithm (DSPA), and experiences a superior edge in learning performance in cycle networks, relative to consensus methods.

The dynamics apparent in the regret plots appear in the relative error performance metric as well, as may be observed in Figures 2.5(b) and 2.5(e), which plot $\mathrm{RE}(\mathbf{w}_{j,t})$ versus time t for grid and cycle networks, respectively. In the grid network, for all $t \geq 100$, $\mathrm{RE}(\mathbf{w}_{j,t}) \approx 10$ for DOGD, DAOL. In cycle networks DOGD achieves a near constant error after $t = 300$ iterations, while DAOL incurs an unbounded $\mathrm{RE}(\mathbf{w}_{j,t})$ with increasing t. Averaging neighbors' predictions is an ineffective strategy for this setting.



(a) Average regret vs. iter. t (b) Relative err vs. iter. t (c) Relative var. vs. iter. t



(d) Average regret vs. iter. t (e) Relative error vs. iter. t (f) Relative variation vs. iter. t

Figure 2.5: Comparison of saddle point method (DSPA) against other decentralized online learning methods as measured by $\mathbf{Reg}_T^j/T$, $\mathrm{RE}(\mathbf{w}_{j,t})$, and $\mathrm{RV}(\bar{\mathbf{w}}_{j,t})$ versus iteration t on grid (top) and cycle (bottom) networks of size $V = 50$ with $\mathbf{w}_{i,t} \sim \mathcal{N}(0, 0.1)$ signal noise. DSPA yields a more effective learning strategy in both network settings, measured in terms of $\mathbf{Reg}_T^j/T$, $\mathrm{RE}(\mathbf{w}_{j,t})$, when compared with consensus methods. DSPA and DOGD achieve comparable levels of consensus in cycle networks, though DOGD fails to learn the batch strategy, as seen in Figure 2.5(d) and Figure 2.5(f). Gradient averaging methods (DOGD, DAOL) fail to learn in the grid network, and DAOL diverges in the cycle setting. Such methods may not be an appropriate tool for networked online learning problems since they seek a consensus which may diverge from the global batch strategy.

The gradient averaging methods fail to achieve consensus in the primal variable in the grid network, as may be seen in Figure 2.5(c), which plots $\mathrm{RV}(\bar{\mathbf{w}}_{j,t})$ versus t. Moreover, while DOGD reaches a comparable level of agent discrepancy to DSPA in the cycle network, as may be seen in Figure 2.5(f), DAOL experiences an unbounded growth in the average relative variation. Thus, in the later setting, DOGD yields a strategy which achieves consensus but diverges from the strategy of the batch learner. If the dimension of the signal to be estimated is less that of the observations, the averaging process of the consensus algorithms fails to move towards the optimum since averaging node predictions does not yield the average of individual loss functions' optima.

2.5 Computer Network Security

We test the use of the saddle point algorithm in (2.17)-(2.18) to train a SVM for detecting security breaches in computer networks. The setting is one in which a number of service providers track user connectivity information in order to predict which users may be potentially harmful. This scenario is naturally cast as an online learning problem since users connect sequentially. If we further consider a network of interconnected service providers we see that each of them would benefit from additional information from other hosts, yet direct information sharing is problematic in terms of communication cost, delay, and the possibility of revealing sensitive proprietary information. This is casted naturally as a *networked* online learning problem where the service providers train their classifiers based on their local information and communication with neighboring peers. Instead of sharing the values of their feature vectors the different service providers exchange multipliers and classification vectors.

In the language of sections 2.1.2 and 2.2 we consider service providers that collect feature vectors $\mathbf{x}_{i,t}$ that they tag as friendly or malicious by setting $y_{i,t} = 1$ or $y_{i,t} = -1$, respectively. Starting with the local feature $\mathbf{x}_{i,t}$ and class $y_{i,t}$ given, as well as with the current local classifier $\mathbf{w}_{i,t}$ and multipliers $\boldsymbol{\lambda}_{ij,t}$ and $\boldsymbol{\lambda}_{ji,t}$ also given, we use the primal iteration in (2.17), which for the particular case of SVM classification takes the specific form in (2.20), to update the local classifier. The vector $\mathbf{w}_{i,t+1}$ is then used to predict the label $y_{i,t+1}$ corresponding to feature $\mathbf{x}_{i,t+1}$. The correct label is observed and recorded for use in the subsequent iteration. The updated classifier $\mathbf{w}_{i,t+1}$ is also shared with neighboring providers that use it to update their Lagrange multipliers using (2.18). The updated multipliers are then shared with neighbors as well. This permits updating of the classifier $\mathbf{w}_{i,t+1}$ through the use of (2.20). The feature vectors $\mathbf{x}_{i,t}$ in our experiments are described next.

2.5.1 Feature Vectors

We use the feature vectors in the data set in [15] which is constructed from approximately seven weeks of tcpdump data of network traffic that is processed into connection records. The training set on which we test the saddle point algorithm consists of $d = 4.94 \times 10^5$ single sample points of size $\tilde{p} = 41$ which contain client connectivity information, whose features fall into three categories: basic, content, and traffic features; see tables I - IV in the supplementary material and [189]. The basic features in Table I consist of information contained in a TCP/IP connection, such as protocol type and user and host information. The content features in Table II consist of those that are most useful for detecting attacks related to user to root and remote to local attacks, examples of which include number of failed login attempts and root access attempts. The traffic features in Table III are computed with respect to a window interval around the connection, and consist of two groups: same host features and same service features. The former tracks connections in the prior two seconds that have the same host destination as the current connection and compute relevant statistics. The latter examines connections in the past two seconds that have the same service type as the current connection. We also record this same information averaged from the perspective of hosts over the last 100 user connections. These metrics are the traffic features shown in Table IV.

Basic, content, and traffic information are recorded for each user connection to construct a set of feature vectors $\{\mathbf{v}_k\}_{k=1}^d$, with labels $y_k \in \{-1, 1\}$ denoting whether a user is harmless or an attacker, respectively. The labels are formed by modifying the data in [15] to merge all the attacker types into one group, and adjusting the number of positive and negative training examples to be approximately equal. Feature statistics reported in Tables I - IV in the supporting document reflect these adjustments. Many features in data set in [15] are categorical (nominal), which we modify to obtain binary features. In particular, for each possible value the categorical variable may take, we construct a binary indicator variable denoting whether the variable takes on a particular value. For example, the feature Protocol Type in [15] takes integer values $1, 2, 3$ corresponding to TCP, UDP, or ICMP, from which we construct three separate indicator variables for protocol type of the individual connection. With this modification, the feature vectors $\mathbf{x}_{i,t}$ are extended to dimension $p = 62$.

2.5.2 Empirical Results

We implement (2.20)-(2.18) for this intrusion detection problem in a cycle network with $N = 50$ nodes. We randomly partition the adjusted data from [15] into N blocks such that each service provider (node) trains a classifier online on its own data subset. We run the simulation over the entire one-percent adjusted training set, i.e. using $NT = 5 \times 10^5$ data points, with a constant step size $\epsilon = 1/\sqrt{T} \approx 0.01$. The regularization parameter

Table 2.1: Components of feature vector for detecting computer network attacks: Standard user connection features.

No.	Feature Type	Type	Range	Description
1.	Duration	Integer	$[0, 5.84 \times 10^4]$	Connection duration
2-4.	Protocol Type	Binary	$\{0, 1\}$	Indicators for protocols TCP, UDP, or ICMP
5-9.	Service	Binary	$\{0, 1\}$	Indicators for http, ftp, smtp, telnet, otherwise "other"
10-25.	Flag	Binary	$\{0, 1\}$	Indicators for connection statuses: SF, S0, S1, S2, S3, OTH, REJ, RSTO, RSTOS0, SH, RSTRH, SHR, RSTOS0, SH, RSTRH, SHR
26.	Source Bytes	Integer	$[0, 6.93 \times 10^8]$	Bytes sent from user
27.	Destination Bytes	Integer	$[0, 5.16 \times 10^6]$	Bytes received by host
28.	Land	Binary	$\{0, 1\}$	Indicator: 1 if source/destination IP addresses and port Number equal, 0 else
29.	Wrong Fragment	Integer	$[0, 3]$	Number of bad checksum packets
30.	Urgent Packets	Integer	$[0, 3]$	Number of packets with urgent bit activated

Table 2.2: Components of feature vector for detecting computer network attacks: Content features tracking suspicious user to host behavior.

No.	Feature Type	Type	Range	Description
31.	Hot	Integer	$[0, 30]$	No. of "hot" actions: enter system dir., or create/execute programs
32.	Number of Failed Logins	Integer	$[0, 5]$	Number of failed logins per connection
33.	Login	Binary	$\{0, 1\}$	1 if login is correct, 0 otherwise
34.	Number Compromised	Integer	$[0, 884]$	Number of "not found" connection errors
35.	Root Shell	Binary	$\{0, 1\}$	1 if root gets the shell, 0 otherwise
36.	su Attempted	Binary	$\{0, 1\}$	1 if su command used, 0 otherwise
37.	Number Root Commands	Integer	$[0, 993]$	Number of user operations done as root
38.	Number File Creations	Integer	$[0, 28]$	Number files user created during session
39.	Number Shell Accesses	Integer	$[0, 2]$	Number of logins of normal users
40.	Number access files	Integer	$[0, 8]$	Number of operations on control files
41.	Number Outbound Commands	Integer	0	Number of outbound ftp commands
42.	Hot Login	Binary	$\{0, 1\}$	1 if admin/root accessed, 0 else
43.	Guest Login	Binary	$\{0, 1\}$	1 if guest login used, 0 else

Table 2.3: Components of feature vector for detecting computer network attacks: Time traffic features derived from user behavior in the last two seconds.

No.	Feature Type	Type	Range	Description
44.	Count	Integer	$[0, 511]$	Number requests for same dest. IP
45.	Server Count	Integer	$[0, 511]$	Number requests for same dest. port
46.	Server Rate	Real	$[0, 1]$	Prop. of connections flagged (4) s0 - s3, among those in Count (23)
47.	Server S. Error Rate	Real	$[0, 1]$	Prop. of users flagged in (4) as s0 - s3, per Server Count (24)
48.	REJ Error Rate	Real	$[0, 1]$	Prop. of users flagged in (4) as REJ, compared with Count (23)
49.	Server Error Rate	Real	$[0, 1]$	Prop. of users flagged (4) as REJ, compared with Server Count (24)
50.	Same Server Rate	Real	$[0, 1]$	Prop. of connections to same service, compared to Count (23)
51.	Different Server Rate	Real	$[0, 1]$	Proportion of connections to different services, per Count (23)
52.	Server Different from Host Rate	Real	$[0, 1]$	Prop. of connections to diff. dest. compared to Server Count (24)

Table 2.4: Components of feature vector for detecting computer network attacks: Machine traffic features derived from the past 100 connections to host. These features are computed with respect same host/client indicators as time traffic features.

No.	Feature Type	Type	Range	Description
53.	Dst. Host Count	Integer	$[0, 255]$	Number requests for same dest. IP
54.	Dst. Host Srv. Count	Integer	$[0, 255]$	Number requests for same dest. port
55.	Dst. Host Same Srv. Rate	Real	$[0, 1]$	Prop. users to same service, compared to Dst. Host Count (32)
56.	Dst. Host Diff. Srv. Rate	Real	$[0, 1]$	Prop. users to diff. services, compared to Dst. Host Count (32)
57.	Dst. Host. Same Src. Port Rate	Real	$[0, 1]$	Prop. users to same source port, compared to Dst. Host Srv. Count (33)
58.	Dst. Host Srv. Diff Host Rate	Real	$[0, 1]$	Prop. users to diff. dest. machine, compared to Dst. Host Srv. Count (32)
59.	Dst. Host Serror Rate	Real	$[0, 1]$	Prop. users flagged (4) as s0 - s3, compared to Dst. Host Count (32)
60.	Dst. Host Srv. Serror Rate	Real	$[0, 1]$	Prop. users flagged (4) as s0 - s3, compared to Dst. Host Srv. Count (33)
61.	Dst. Host R. Error Rate	Real	$[0, 1]$	Proportion of users flagged (4) as REJ, as compared to Dst. Host Count (32)
62.	Dst. Host Srv. Error Rate	Real	$[0, 1]$	Prop. users flagged (4) as REJ, compared to Dst. Host Srv. Count (33)

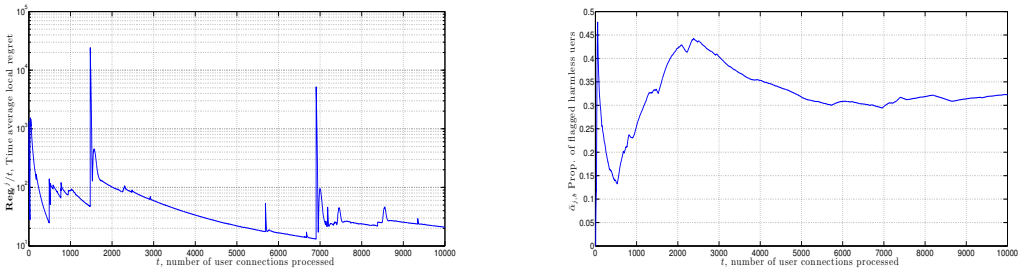


Figure 2.6: Left: Average local regret $\mathbf{Reg}_t^j/t$ vs. number of user connections processed t on a $N = 50$ node cycle network for the network security application of the distributed online SVM saddle point algorithm (2.20)-(2.18). Local regret of node $j = 29$ vanishes with t as node j's classifier converges to the global batch classifier computed with Liblinear [63]. Spikes in $\mathbf{Reg}_t^j/t$ correspond to misclassifications, and follow from nondifferentiability of the hinge loss. Large spikes at $t = 1476, 6905$ correspond to attacker examples not previously seen by the service provider that compromise its security, from which it quickly recovers. Right: Time average error rate of incorrectly flagging a benign user $\bar{\alpha}_{j,T} = \sum_{t=1}^{T} P(\hat{y}_{j,t} = 1 \mid y_{j,t} = -1)$ on a test set of $T = 1 \times 10^4$ user connections. The error rate stabilizes between $[0.30, 0.33]$ as the server learns to not flag benign users unnecessarily, despite widely varying connectivity information

$\zeta = \log(62)$ is chosen after 10-fold cross-validation and the Residual Information Criterion (RIC) as in [180]. The primal and dual variables are initialized at time $t = 1$ as zero vectors $\mathbf{w}_{j,1} = \mathbf{0}$ for all j and $\boldsymbol{\lambda}_{jk,1} = \mathbf{0}$ for all (j, k). We compute the global batch classifier $\tilde{\mathbf{w}}^*$ with Liblinear [63].

Fig. 2.6 shows $\mathbf{Reg}_t^j/t$ the time average local regret for the (arbitrarily chosen) node $j = 29$. The iteration index t corresponds to the number of user connections processed. Observe that $\mathbf{Reg}_t^j/t$ decays with the number of processed user connections at the rate guaranteed by Theorem 1. The large instantaneous magnitude of $\mathbf{Reg}_t^j/t$ is a result of the large ranges of features such as Source and Destination Bytes. Large spikes at $t = 1,476$ and $t = 6,905$ correspond to attacker examples not previously observed. Observe that by $t = 10^5$, $\mathbf{Reg}_t^j/t \leq 21$, indicating that the service provider effectively learns an intrusion classifier as good as the one with user information aggregated at a central location for all times in advance. Each time an attacker compromises the host, which correspond to a spike in the local regret trajectory, the intrusion detection protocol recovers quickly.

We turn to studying the classifier error rates. Denote the vector of predictions $\hat{y}_{j,t}$, which is of length t and whose uth entry is given by $[\hat{y}_{j,t}]_u = \text{sgn}(\mathbf{w}_{j,t}^T \mathbf{x}_{j,u})$ for users $u \leq t$. We break misclassifications into two categories: (i) the false alarm rate $\alpha_{j,t} := P(\hat{y}_{j,u} = 1 \mid y_{j,u} = -1)$ which tracks the proportion of friendly users predicted as attackers; (ii) the error rate $\beta_{j,t} := P(\hat{y}_{j,u} = -1 \mid y_{j,u} = 1)$ which accounts for the attackers that were not detected. These quantities are computed as the number of entries of $\hat{y}_{j,t}$ that equal 1 over the number of associated users $u \leq t$ with label -1, and vice versa. We consider the average false alarm rate $\bar{\alpha}_{j,t} = \sum_{u=1}^{t} P(\hat{y}_{j,u} = 1 \mid y_{j,u} = -1)/t$, and the average error rate

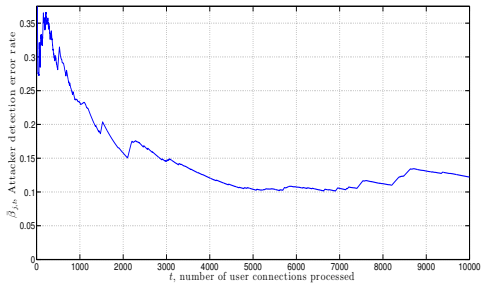


Figure 2.7: Time average empirical probability of failing to detect an attacker $\bar{\beta}_{j,T} = \sum_{t=1}^{T} P(\hat{y}_{j,t} = -1 \mid y_{j,t} = 1)$ on a test set of $T = 1 \times 10^4$ user connections. The error rate stabilizes between $[0.10, 0.15]$ as the host learns to deny service to a variety of attacker profiles.

$\bar{\beta}_{j,t} = \sum_{u=1}^{t} P(\hat{y}_{j,u} = -1 \mid y_{j,u} = 1)/t$ on a test data set of size $T = 1 \times 10^4$.

Fig. 2.6 shows the evolution of $\bar{\alpha}_{j,t}$, the average false alarm rate, versus the number of connections processed t. The expected classifier accuracy $(1 - \bar{\alpha}_{j,t})$ stabilizes between $[0.67, 0.70]$ after a burn-in period $t \geq 5 \times 10^3$ (false alarm rate $[0.30, 0.33]$), indicating that an inordinate proportion of friendly users are not denied service in this intrusion detection protocol. A node's ability to flag harmful users is the essential performance metric. We also investigate the error rate of service provider $j = 29$ on a test set of fixed size $T = 1 \times 10^4$, with the number of connections processed t. The average rate of correctly detecting an attacker, or power, $(1 - \bar{\beta}_{j,t})$ begins near null and stabilizes between $[0.86, 0.90]$ for $t \geq 3 \times 10^3$, which is competitive given the difficulty predicting attacks in commercial settings. The price for this accuracy level is its conservative treatment of normal users.

2.6 Takeaways for Decentralized Consensus Learning of GLMs

We extended the idea of online convex optimization to networked settings, where nodes are allowed to make autonomous learning decisions while incorporating neighbors' information. We developed regret formulations illustrating the distributed learning goal and proposed the use of a saddle point algorithm to solve such problems. Theorem 1 showed that the saddle point iterates achieve the networked online learning goal, which is the sub linear growth rate of global networked regret: the time average regret goes to null at a rate of $O(1/\sqrt{T})$. Theorem 2 guaranteed that individual agents also achieve this learning rate as well.

Numerical analysis demonstrated the algorithm performance dependency on network size, connectivity, and topology: learning rates are comparable across different network sizes but more prone to numerical oscillations in smaller networks. Similarly, network topologies with smaller diameter yield more stable predictions. We applied this algorithm

to the problem of training a SVM classifier online over a network, and consider an attacker detection problem in a computer network security application. Empirically this method yields reasonable, but not state of the art, prediction accuracy and is able to maintain the privacy of distinct nodes' users connectivity data.

Overall, this approach requires each agent to learn a linear statistical model $f_i(\mathbf{x}_i) = \mathbf{w}_i^T \mathbf{x}_i$ and operates on the hypothesis that each agent seeks to learn decision model parameters. While this later hypothesis may be reasonable, the former limits the empirical statistical performance of this approach. However, the mathematical tools developed in this chapter are useful for addressing a more general hypothesis regarding the relationship between agents' data in the following chapter. We defer addressing more general selections for $\mathscr{F}$ than $\mathbb{R}^p$ until Parts II and III.

Chapter 3

Online learning in heterogeneous networks

In this chapter, we extend the ideas of Chapter 2 to the case where agents aim to keep their decision variables *close* to one another but *not coincide* in order to minimize this global objective while giving preference to possibly distinct local signals. The motivation for this problem comes from the fact that consensus optimization methods implicitly operate on the hypothesis that the distribution of observations at each node is identical, which does not hold for a variety of problems in signal processing [166] and robotics [92, 95].

More specifically, in distributed optimization problems, agent agreement may not always be the primary goal. In large-scale settings where one aims to leverage parallel processing architectures to alleviate computational bottlenecks, agreement constraints are suitable. In contrast, if there are different priors on information received at distinct subsets of agents, then requiring the network to reach a common decision may degrade local accuracy. Specifically, if the observations at each node are independent but *not* identically distributed, consensus may yield a sub-optimal solution. Moreover, there are tradeoffs in complexity and communications, and it may be that only a subset of nodes requires a solution.

Various attempts to extend multi-agent optimization techniques to exploit heterogeneous correlation structures among observations received by each agent have been proposed, motivated by multi-task learning [41]. For instance, attempts to extend primal averaging techniques to generic inequality constraints in the online decentralized setting via penalty methods were developed in [195], but require the use of diminishing step-size rules and growing penalty parameters, which are outperformed by constant learning rates in dynamic estimation settings. An alternative primal averaging approach for multi-agent systems with multiple distinct but correlated optima was developed in [45], but only for the square loss. In the later work, Euclidean penalties are added to agents' local objectives to incentivize tracking of multiple interrelated optima, which may or may not capture a generic correlation

structure among agents' data streams.

In this chapter, we seek to solve problems in which each agent aims to minimize a global cost $\sum_i \ell_i$ subject to a network proximity constraint, which allows agents the leeway to select actions which are good with respect to a global cost while not ignoring the structure of locally observed information. This setting may correspond to a multi-target tracking problem in a sensor network or a collaborative learning task in a robotic network where each robot is operating in a distinct domain, i.e. instances of multi-task learning [41]. However, we allow for constraints to be generically chosen convex inequalities, rather than a Euclidean penalty, as in [45]. We design multi-agent optimization strategies where agents reach a common understanding of global information, while still retaining their local perspectives.

We propose a modification of the saddle point method [9,134] introduced in Chapter 2 to solve online multi-agent optimization problems with network proximity constraints, which we prove converges in expectation to a primal-dual optimal pair of this problem when a constant algorithm step-size is chosen. We demonstrate the proposed method's utility on a spatially correlated random field estimation problem in a sensor network in Section 3.4, and apply this tool to a source localization problem in Section 3.5.

3.1 Multi-Agent Optimization with Proximity Constraints

We consider agents i of a symmetric, connected, and directed network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ (Assumption 6) with $|\mathcal{V}| = V$ nodes and $|\mathcal{E}| = E$ edges and denote as $n_i := \{j : (i,j) \in \mathcal{E}\}$ the neighborhood of agent i. For simplicity we assume that the number of edges E is even. Each of the agents is associated with a (non-strongly) convex loss function $\ell_i : \mathcal{W} \times \Theta_i \to \mathbb{R}$ that is parameterized by a decision variable $\mathbf{w}_i \in \mathcal{W} \subset \mathbb{R}^p$ and a random vector $\boldsymbol{\theta}_i \in \Theta_i \subset \mathbb{R}^q$ with a proper distribution. The functions $\ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i)$ for different $\boldsymbol{\theta}_i$ are interpreted as the merit of a particular statistical model $\mathbf{w}_i$, and the random vector $\boldsymbol{\theta}$ may be particularized, for instance, to a random pair $\boldsymbol{\theta} = (\mathbf{z}, \mathbf{y})$. For this case, the random pair correspond to, e.g., feature vectors $\mathbf{z}$ together with their binary labels $\mathbf{y} \in \{-1, 1\}$ or real values $\mathbf{y} \in \mathbb{R}$, respectively, for classification or regression.

In this chapter, we focus on the case where $\boldsymbol{\theta}_i$ represents data which revealed to node i *sequentially* through realizations $\boldsymbol{\theta}_{i,t}$ at time t, and agents would like to process this information incrementally. Mathematically this is equivalent to the case where the total number of samples T revealed to agent i is not necessarily finite. In the online setting considered here the functions $\ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i)$ are termed instantaneous because they are observed at particular points in time associated with realizations $\boldsymbol{\theta}_{i,t}$ of the random vector $\boldsymbol{\theta}_i$; see

Section 3.2. A possible goal for agent i is the computation of the optimal local estimate,

$$\mathbf{w}_i^{\mathrm{L}} := \operatorname*{argmin}_{\mathbf{w}_i \in \mathcal{W}} L_i(\mathbf{w}_i) := \operatorname*{argmin}_{\mathbf{w}_i \in \mathcal{W}} \mathbb{E}_{\boldsymbol{\theta}_i}[\ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i)] \ . \tag{3.1}$$

We refer to $L_i(\mathbf{w}_i) := \mathbb{E}_{\boldsymbol{\theta}_i}[\ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i)]$ as the local average function at node i. We further assume $\mathcal{W}$ to be a compact convex subset of $\mathbb{R}^p$ associated with the p-dimensional parameter vector of agent i.

When we consider the network as a whole we can define the stacked vector $\mathbf{w} = [\mathbf{w}_1, \ldots, \mathbf{w}_V]$, which is an element of the product set $\mathcal{W}^V \subset \mathbb{R}^{Vp}$, and the aggregate function $L(\mathbf{w}) := \sum_{i=1}^{V} \mathbb{E}_{\boldsymbol{\theta}_i}[\ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i)]$. It then follows that the set of problems in (3.1) is equivalent to the aggregate problem

$$\mathbf{w}^{\mathrm{L}} = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}^V} L(\mathbf{w}) := \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}^V} \sum_{i=1}^{V} \mathbb{E}_{\boldsymbol{\theta}_i}[\ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i)] \ . \tag{3.2}$$

For convenience, we further define the stacked instantaneous function as $\ell(\mathbf{w}, \boldsymbol{\theta}) = \sum_i \ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i)$. That (3.1) and (3.2) describe the same problem is true because there is no coupling between the variables $\mathbf{w}_i$ at different agents. In many situations, however, the parameters $\mathbf{w}_i^{\mathrm{L}}$ that different agents want to estimate are related. It then makes sense to couple decisions of different agents as a means of letting agents exploit each others' observations. Consensus optimization problems work on the hypothesis that all agents are interested in learning the same decision parameters $\mathbf{w}_i$ for all $i \in V$. In this case, we modify (3.2) by introducing consensus constraints of the form

$$\mathbf{w}_i = \mathbf{w}_j, \ \text{for all } j \in n_i \ . \tag{3.3}$$

For a connected network this constraint makes all variables $\mathbf{w}_i$ equal – hence the definition as a consensus problem. This hypothesis implicitly only makes sense in cases where agents observe information drawn from a common distribution, which may be overly restrictive. In general, parameters of nearby nodes are expected to be close but are not necessarily all equal, as is the situation in, e.g., the estimation of a smooth field that is albeit not uniform. To model this situation we introduce a convex local proximity function with real-valued range of the form $h_{ij}(\mathbf{w}_i, \mathbf{w}_j)$ and a tolerance $\gamma_{ij} \geq 0$ [97]. These are used to couple the decisions of agent i to those of its neighbors $j \in n_i$ through the definition of the optimal

estimates as the solution of the constrained optimization problem

$$\begin{aligned} \mathbf{w}^* \in \underset{\mathbf{w}\in\mathcal{W}^V}{\operatorname{argmin}} \; & \sum_{i=1}^{V} \mathbb{E}_{\boldsymbol{\theta}_i}[\ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i)] \\ \text{s.t.} \quad & h_{ij}(\mathbf{w}_i, \mathbf{w}_j) \leq \gamma_{ij}, \quad \text{for all } j \in n_i. \end{aligned} \tag{3.4}$$

In the formulation in (3.4), $\mathbf{w}^*$ belongs to a set of constrained optimizers $\mathcal{W}^*$, i.e., $\mathbf{w}^*$ is not unique, due to the weak convexity of the local objectives $L_i(\mathbf{w})$. Moreover, for this set to be non-empty, we assume that the set of optimizers $\mathcal{W}^*$ has non-empty intersection with the primal feasible set $\mathcal{W}$ – a condition satisfied under Slater's condition (Assumption 9).

We assume that the proximity function $h_{ij}(\mathbf{w}_i, \mathbf{w}_j)$ that couples node i to node j is equivalent to the proximity function $h_{ji}(\mathbf{w}_j, \mathbf{w}_i)$ that couples node j to node i, i.e., that for all $\mathbf{w}_i$ and $\mathbf{w}_j$ we have $h_{ij}(\mathbf{w}_i, \mathbf{w}_j) = h_{ji}(\mathbf{w}_j, \mathbf{w}_i)$ and $\gamma_{ij} = \gamma_{ji}$. This implies that the constraints $h_{ij}(\mathbf{w}_i, \mathbf{w}_j) \leq \gamma_{ij}$ and $h_{ji}(\mathbf{w}_j, \mathbf{w}_i) \leq \gamma_{ji}$ are redundant. We also define the stacked constraint function $h : \mathcal{W}^V \to \mathbb{R}^E$. We keep them separate to maintain symmetry of the algorithm derived in Section 3.2.

The consensus constraints in (3.3) are a particular example of a proximity function $h_{ij}(\mathbf{w}_i, \mathbf{w}_j)$ but so is the norm constraint $\|\mathbf{w}_i - \mathbf{w}_j\|^2 \leq \gamma_{ij}$. This latter choice makes the estimates $\mathbf{w}_i^*$ and $\mathbf{w}_j^*$ of neighboring nodes close to each other but not necessarily equal. Implicitly, this allows i to incorporate the (relevant) information of neighboring nodes without detrimentally incorporating the information of far away nodes that are only weakly correlated with the estimator of node i.

The goal of this chapter is to develop an algorithm to solve (3.4) in distributed online settings where nodes don't know the distribution of the random vector $\boldsymbol{\theta}_i$ but observe local instantaneous functions $\ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i)$ sequentially. An important observation here is that the workhorse distributed gradient descent (DGD) [77, 157, 194, 223] and dual methods [78, 154, 209] can't be used to solve (3.4) because they work only when the constraints $h_{ij}(\mathbf{w}_i, \mathbf{w}_j)$ are linear. Extensions of DGD to inequality constraints have been considered in [195], but constraints are assumed to be local only, and thus may not capture cross-agent correlations. While penalty-based variants of [195] may be developed for (3.4), their performance guarantees would hinge on use of attenuating learning rates, which have been found to be empirically inferior to methods based on constant step-sizes. These observations motivate an alternative approach based on Lagrange duality. In particular, we will see that a stochastic saddle point method can be distributed when the functions $h_{ij}(\mathbf{w}_i, \mathbf{w}_j)$ are not necessarily linear and converges to the solution of (3.4) when local instantaneous functions $\ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i)$ are independently sampled over time. Before developing this algorithm, we discuss a representative example to clarify ideas.

Example (LMMSE Estimation of a Random Field). A Gauss-Markov random field is one in which the value of the field at the location of sensor i, denoted by $\mathbf{w}_i$, is of interest. Consider a sequential estimation problem in which the nodes of the sensor network acquire noisy linear transformations of the field's value at their respective positions. Formally, let $\boldsymbol{\theta}_{i,t} \in \mathbb{R}^q$ be the observation collected by sensor i at time t. Observations $\boldsymbol{\theta}_{i,t}$ are noisy linear transformations $\boldsymbol{\theta}_{i,t} = \mathbf{H}_i \mathbf{w}_i + \mathbf{w}_{i,t}$ of a signal $\mathbf{w}_i \in \mathbb{R}^p$ contaminated with Gaussian noise $\mathbf{w}_{i,t} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ independently distributed across nodes and time. Ignoring neighboring observations, the minimum mean square error local estimation problem at node i can then be written in the form of (3.1) with $\ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i) = \|\mathbf{H}_i \mathbf{w}_i - \boldsymbol{\theta}_i\|^2$. The quality of these estimates can be improved using the correlated information of adjacent nodes but would be hurt by trying to make estimates uniformly equal across the network. This problem specification can be captured by the mathematical formulation

$$\begin{aligned} \mathbf{w}^* := \ & \underset{\mathbf{w} \in \mathcal{W}^V}{\operatorname{argmin}} \ \sum_{i=1}^{V} \mathbb{E}_{\boldsymbol{\theta}_i} \Big[\|\mathbf{H}_i \mathbf{w}_i - \boldsymbol{\theta}_i\|^2 \Big] \\ & \text{s.t.} \qquad (1/2)\|\mathbf{w}_i - \mathbf{w}_j\|^2 \leq \gamma_{ij}, \quad \text{for all } j \in n_i. \end{aligned} \tag{3.5}$$

The constraint $(1/2)\|\mathbf{w}_i - \mathbf{w}_j\|^2 \leq \gamma_{ij}$ makes the estimate $\mathbf{w}_i^*$ of node i close to the estimates $\mathbf{w}_j^*$ of neighboring nodes $j \in n_i$ but not so close to the estimates $\mathbf{w}_k^*$ of nonadjacent nodes $k \notin n_i$. The problem formulation in (3.5) is a particular case of (3.4) with $\ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i) = \|\mathbf{H}_i \mathbf{w}_i - \boldsymbol{\theta}_i\|^2$ and $h_{ij}(\mathbf{w}_i, \mathbf{w}_j) = (1/2)\|\mathbf{w}_i - \mathbf{w}_j\|^2$.

3.2 Primal-Dual Method

Recall that a decentralized algorithm is one in which node i has access to local functions $\ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i)$ and local constraints $h_{ij}(\mathbf{w}_i, \mathbf{w}_j) \leq \gamma_{ij}$ and exchanges information with neighbors $j \in n_i$ only. Recall also that the algorithm is further said to be online if the distribution of θ_i is unknown and agent i has access to independent observations $\boldsymbol{\theta}_{i,t}$ that are acquired sequentially. Our goal is to develop an online decentralized algorithm to solve (3.4). To achieve this we consider the approximate Lagrangian relaxation of (3.4) which we state as

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = \sum_{i=1}^{V} \left[\mathbb{E}_{\boldsymbol{\theta}_i}[\ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i)] + \frac{1}{2} \sum_{j \in n_i} \left(\lambda_{ij} \left(h_{ij}(\mathbf{w}_i, \mathbf{w}_j) - \gamma_{ij} \right) - \frac{\delta \eta_t}{2} \lambda_{ij}^2 \right) \right], \tag{3.6}$$

where $\lambda_{ij} \in \mathbb{R}^+$ is a nonnegative Lagrange multiplier associated with the proximity constraint between node i and node j, and the factor of $1/2$ comes from the redundancy of the constraints $h_{ij}(\mathbf{w}_i, \mathbf{w}_j) \leq \gamma_{ij}$ and $h_{ji}(\mathbf{w}_j, \mathbf{w}_i) \leq \gamma_{ij}$, and helps scale the contribution of each when computing gradients (see Proposition 1). Observe that (3.6) *does not* define the

Lagrangian of the optimization problem (3.4), but instead defines an *augmented Lagrangian* due to the presence of the last term on the right-hand side. This last term $-(\delta\eta_t/2)\lambda_{ij}^2$, with scalar parameters δ and η_t, is a regularizer on the dual variable, whose utility arises in controlling the accumulation of constraint violation of the algorithm over time. See Section 3.3 for details.

To solve (3.4), stochastic approximation is necessary. In particular, the necessity for operating with stochastic gradients rather than true gradients comes from the fact that computing gradients of the statistical average objective in (3.4) has complexity that is at least on the order of the sample size T, which in setting considered here may be infinite. Furthermore, due to the online nature of the problem, at time t, each individual agent in the network only has access to random variables $\{\boldsymbol{\theta}_{i,u}\}_{u\leq t}$. Thus, computations involving the average objective involve data $\{\boldsymbol{\theta}_{i,u}\}_{u>t}$ which is not yet observed, and thus is unavailable.

Therefore, we propose applying a stochastic saddle point algorithm to (3.6) which operates by alternating primal and dual stochastic gradient descent and ascent steps respectively. Consider the stochastic approximation of the augmented Lagrangian evaluated at observed realizations $\boldsymbol{\theta}_{i,t}$ of the random vectors $\boldsymbol{\theta}_i$, which we define as

$$\hat{\mathcal{L}}_t(\mathbf{w},\boldsymbol{\lambda}) = \sum_{i=1}^{V}\Big[\ell_i(\mathbf{w}_i,\boldsymbol{\theta}_{i,t}) + \frac{1}{2}\sum_{j\in n_i}\lambda_{ij}\left(h_{ij}(\mathbf{w}_i,\mathbf{w}_j)-\gamma_{ij}\right) - \frac{\delta\eta_t}{2}\lambda_{ij}^2\Big]. \tag{3.7}$$

Define the stacked dual variable as $\boldsymbol{\lambda} := [\lambda_1;\cdots;\lambda_E] \in \mathbb{R}^E$. Moreover, denote the network aggregate random vector as $\boldsymbol{\theta} = [\boldsymbol{\theta}_1;\cdots;\boldsymbol{\theta}_V]$. The stochastic saddle point method applied to the stochastic Lagrangian stated in (3.7) takes the form

$$\mathbf{w}_{t+1} = \mathcal{P}_{\mathcal{W}^V}\Big[\mathbf{w}_t - \eta_t\nabla_{\mathbf{w}}\hat{\mathcal{L}}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)\Big], \tag{3.8}$$

$$\boldsymbol{\lambda}_{t+1} = \Big[\boldsymbol{\lambda}_t + \eta_t\nabla_{\boldsymbol{\lambda}}\hat{\mathcal{L}}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)\Big]_+, \tag{3.9}$$

where $\nabla_{\mathbf{w}}\hat{\mathcal{L}}(\mathbf{w}_t,\boldsymbol{\lambda}_t)$ and $\nabla_{\boldsymbol{\lambda}}\hat{\mathcal{L}}(\mathbf{w}_t,\boldsymbol{\lambda}_t)$, are the primal and dual stochastic gradients of the augmented Lagrangian with respect to $\mathbf{w}$ and $\boldsymbol{\lambda}$, respectively. These stochastic subgradients are approximations of the gradients of (3.6) evaluated at the current realization of the random vector $\boldsymbol{\theta}$. The notation $\mathcal{P}_{\mathcal{W}}^V(\mathbf{w})$ denotes component-wise orthogonal projection of the individual primal variables $\mathbf{w}_i$ onto the given convex compact set $\mathcal{W}$, and $[\cdot]_+$ denotes the projection onto the E-dimensional nonnegative orthant $\mathbb{R}_+^E$. As an abuse of notation, we also use $[\cdot]_+$ to denote scalar positive projection where appropriate.

The method stated in (3.8) - (3.9) can be implemented with decentralized computations across the network, as we state in the following proposition.

Proposition 1 *Let $\mathbf{w}_{i,t}$ be the ith component of the primal iterate $\mathbf{w}_t$ and $\lambda_{ij,t}$ the i,jth*

component the dual iterate $\boldsymbol{\lambda}_t$*. The primal variable update is equivalent to the set of* V *parallel local variable updates*

$$\mathbf{w}_{i,t+1} = \mathcal{P}_{\mathcal{W}}\Big[\mathbf{w}_{i,t} - \eta_t\Big(\nabla_{\mathbf{w}_i}\ell_i(\mathbf{w}_{i,t};\boldsymbol{\theta}_{i,t}) + \frac{1}{2}\sum_{j\in n_i}(\lambda_{ij,t}+\lambda_{ji,t})\nabla_{\mathbf{w}_i}h_{ij}(\mathbf{w}_{i,t},\mathbf{w}_{j,t})\Big)\Big]\ . \quad (3.10)$$

Likewise, the dual variable updates in (3.9) *are equivalent to the* E *parallel updates*

$$\lambda_{ij,t+1} = \Big[(1-\eta_t^2\delta)\lambda_{ij,t} + \eta_t\left(h_{ij}(\mathbf{w}_{i,t},\mathbf{w}_{j,t}) - \gamma_{ij}\right)\Big]_+. \quad (3.11)$$

Proof: To compute the primal stochastic gradient of the Lagrangian in (3.6), observe that in the instantaneous Lagrangian in (3.7) only a few summands depend on $\mathbf{w}_i$. In the first sum only the one associated with the local objective $\ell_i(\mathbf{w}_i,\boldsymbol{\theta}_{i,t})$ depends on $\mathbf{w}_i$. In the second sum the terms that depend on $\mathbf{w}_i$ include the local constraints $h_i(\mathbf{w}_i,\mathbf{w}_j)-\gamma_{ij}$ and the neighboring constraints $h_j(\mathbf{w}_j,\mathbf{w}_i)-\gamma_{ji}$. Taking gradients of these terms yields,

$$\nabla_{\mathbf{w}_i}\hat{\mathcal{L}}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t) = \ \nabla_{\mathbf{w}_i}\ell_i(\mathbf{w}_{i,t};\boldsymbol{\theta}_{i,t}) + \sum_{j\in n_i}(\boldsymbol{\lambda}_{ij,t}+\boldsymbol{\lambda}_{ji,t})^T\nabla_{\mathbf{w}_i}h_{ij}(\mathbf{w}_{i,t},\mathbf{w}_{j,t}). \quad (3.12)$$

Writing (3.8) componentwise and substituting $\nabla_{\mathbf{w}_i}\hat{\mathcal{L}}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)$ for its expression in (3.12), the result in (3.10) follows.

To prove (3.11) we just need to compute the gradient $\hat{\mathcal{L}}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t)$ of the stochastic Lagrangian with respect to the Lagrange multipliers associated with edge (i,j). By noting that only one summand in (3.7) depends on this multiplier we conclude that

$$\nabla_{\lambda_{ij}}\hat{\mathcal{L}}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t) = h_{ij}(\mathbf{w}_{i,t},\mathbf{w}_{j,t}) - \gamma_{ij} - \eta_t\delta\lambda_{ij,t}\ . \quad (3.13)$$

After gathering terms in (3.13) and substituting the result into (3.9), we obtain (3.11). ■

With primal variables $\mathbf{w}_{i,t}$ and Lagrange multipliers $\lambda_{ij,t}$ maintained and updated by node i, Proposition 1 implies that the saddle point method in (3.8)-(3.9) can be translated into a decentralized protocol in which: (i) The primal and dual variables variables of distinct agents across the network are decoupled from one another. (ii) The updates require exchanges of information among neighboring nodes only. This protocol is summarized in Algorithm 2.

Indeed, in the primal update in (3.11) agent i can compute the stochastic gradient $\nabla_{\mathbf{w}_i}\ell_i(\mathbf{w}_{i,t};\boldsymbol{\theta}_{i,t})$ of its objective function by making use of its local observations $\boldsymbol{\theta}_{i,t}$ and its decision variable $\mathbf{w}_{i,t}$ at the previous time slot t. To compute the gradients of the constraint functions $\nabla_{\mathbf{w}_i}h_{ij}(\mathbf{w}_{i,t},\mathbf{w}_{j,t})$ the primal variables $\mathbf{w}_{j,t}$ of neighboring nodes $j\in n_i$ are needed on top of the local variables $\mathbf{w}_{i,t}$, but these can be communicated from neighbors. To implement (3.10) agent i also needs access to the Lagrange multipliers $\lambda_{ij,t}$ associated

Algorithm 2 SSPM: Stochastic Saddle Point Method

Require: initialization $\mathbf{w}_0$ and $\boldsymbol{\lambda}_0 = \mathbf{0}$, step-size η_t, regularizer δ

for $t = 1, 2, \ldots, T$ **do**

  loop in parallel agent $i \in V$

    Send primal and dual variables $\mathbf{w}_{i,t}, \boldsymbol{\lambda}_{ij,t}$ to nbhd. $j \in n_i$

    Receive variables $\mathbf{w}_{j,t}, \boldsymbol{\lambda}_{ij,t}$ from neighbors $j \in n_i$

    Update local parameter $\mathbf{w}_{i,t}$ with (3.10)

$$\mathbf{w}_{i,t+1} = \mathcal{P}_{\mathcal{W}}\Big[\mathbf{w}_{i,t} - \eta_t\Big(\nabla_{\mathbf{w}_i}\ell_i(\mathbf{w}_{i,t};\boldsymbol{\theta}_{i,t}) + \sum_{j\in n_i}(\lambda_{ij,t} + \lambda_{ji,t})\nabla_{\mathbf{w}_i}h_{ij}(\mathbf{w}_{i,t},\mathbf{w}_{j,t})\Big)\Big]\,.$$

  end loop

  loop in parallel communication link $(i,j) \in \mathcal{E}$

    Update dual variables at network link (i,j) [cf. (3.11)]

$$\lambda_{ij,t+1} = \Big[(1-\eta_t^2\delta)\lambda_{ij,t} + \eta_t\left(h_{ij}(\mathbf{w}_{i,t},\mathbf{w}_{j,t}) - \gamma_{ij}\right)\Big]_+$$

  end loop

end for

with the network proximity constraints $h_{ij}(\mathbf{w}_i, \mathbf{w}_j)$ and the multipliers $\lambda_{ji,t}$ associated with the network proximity constraints $h_{ji}(\mathbf{w}_j, \mathbf{w}_i)$. The multipliers $\lambda_{ij,t}$ are locally available at node i and the multipliers $\lambda_{ji,t}$ can be communicated from neighbors.

To implement the dual update in (3.11) agent i needs access to its own dual variable $\lambda_{ij,t}$ as well as the local decision variables $\mathbf{w}_{i,t}$. It also needs access to the primal variables $\mathbf{w}_{j,t}$ of neighbors $j \in n_i$ to compute the local dual gradient which is given as the constraint slack $h_{ij}(\mathbf{w}_{i,t}, \mathbf{w}_{j,t}) - \gamma_{ij}$. As in the primal, these neighboring variables can be communicated from neighbors. We can then implement (3.10) after nodes exchange primal and dual variables $\mathbf{w}_{i,t}$ and $\lambda_{ij,t}$, proceed to implement (3.11) after they exchange updated primal variables $\mathbf{w}_{i,t}$, and conclude with the exchange of primal and dual variables $\mathbf{w}_{i,t}$ and $\lambda_{ij,t}$ that are needed to implement the primal iteration at time t. These local operations repeated in synchrony by all nodes is equivalent to the centralized operations in (3.8)-(3.9).

In the following section, we analyze the iterations in (3.8)-(3.9), which implies convergence of the equivalent iterations in (3.10) - (3.11). We close here with an example and a remark.

Example (LMMSE Estimation of a Random Field). Revisit the random filed estimation problem of Section 3.1 that we summarize in the problem formulation in (3.5). Recalling the identifications $\ell_i(\mathbf{w}_i, \boldsymbol{\theta}_i) = \|\mathbf{H}_i\mathbf{w}_i - \boldsymbol{\theta}_i\|^2$ and $h_{ij}(\mathbf{w}_i, \mathbf{w}_j) = (1/2)\|\mathbf{w}_i - \mathbf{w}_j\|^2$ it follows that the local primal update in (3.10) takes the form

$$\mathbf{w}_{i,t+1} = \mathcal{P}_{\mathcal{W}}\Big[\mathbf{w}_{i,t} - \eta_t\Big[2\mathbf{H}_i^T\left(\mathbf{H}_i\mathbf{w}_{i,t} - \boldsymbol{\theta}_{i,t}\right) + \frac{1}{2}\sum_{j\in n_i}\Big(\lambda_{ij,t} + \lambda_{ji,t}\Big)\Big(\mathbf{w}_{i,t} - \mathbf{w}_{j,t}\Big)\Big]\Big]. \quad (3.14)$$

Likewise, the specific form of the dual update in (3.11) is

$$\lambda_{ij,t+1} = \left[(1-\eta_t^2\delta)\lambda_{ij,t} + (\eta_t/2)\left(\|\mathbf{w}_{i,t}-\mathbf{w}_{j,t}\|^2 - \gamma_{ij}\right)\right]_+ . \tag{3.15}$$

The empirical utility of the decentralized estimation scheme in (3.14) - (3.15) is studied in Section 3.4. Alternative functional forms for the network proximity constraints are studied for a source localization problem in Section 3.5 . ■

Remark 2 If the proximity constants are $\gamma_{ij} = \gamma_{ji}$ and the initial Lagrange multipliers satisfy $\lambda_{ij,0} = \lambda_{ji,0}$ it follows from (3.11) that $\lambda_{ij,t} = \lambda_{ji,t}$ for all subsequent times t. This is as it should be because the constraints $h_{ij}(\mathbf{w}_i, \mathbf{w}_j) \leq \gamma_{ij}$ and $h_{ji}(\mathbf{w}_j, \mathbf{w}_i) \leq \gamma_{ji}$ are redundant. If these multipliers are equal for all times, the primal update in (3.10) does not necessitate exchange of dual variables. This does not save communication cost as it is still necessary to exchange primal variables $\mathbf{w}_{i,t}$.

3.3 Convergence in Expectation

We turn to establishing that the saddle point algorithm defined by (3.8)-(3.9) converges to the primal-dual optimal point of the problem stated in (3.4) when a constant algorithm step-size is used. In particular, we establish bounds on the objective function error sequence $L(\mathbf{w}_t) - L(\mathbf{w}^*)$ and the network-aggregate constraint violation, both in expectation, where $\mathbf{w}^*$ is defined by (3.4). As a consequence, the time-average primal vector converges to the optimal objective function $L(\mathbf{w}^*)$ at a rate of $\mathcal{O}(1/\sqrt{T})$, while incurring constraint violation on the order of $\mathcal{O}(T^{-1/4})$, both on average, where T is the total number of iterations. To establish these results, we note some facts of the problem setting, and then introduce a few standard assumptions.

First, observe that the dual stochastic gradient is independent of random vectors $\boldsymbol{\theta}_{i,t}$ [cf. (3.11)], and hence for all t,

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{w}_t, \boldsymbol{\lambda}_t) = \nabla_{\boldsymbol{\lambda}} \hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t). \tag{3.16}$$

Also pertinent to analyzing the performance of the stochastic saddle point method is the fact that the primal stochastic gradient of the Lagrangian is an unbiased estimator of the true primal gradient. Let $\mathcal{F}_t$ be a sigma algebra that measures the history of the algorithm up until time t, i.e., a collection that contains at least the variables $\{\mathbf{w}_u, \boldsymbol{\lambda}_u, \boldsymbol{\theta}_u\}_{u=1}^t \subseteq \mathcal{F}_t$. That the primal stochastic gradient is an unbiased estimate of the true primal gradients

means that,

$$\mathbb{E}\left[\nabla_{\mathbf{w}}\hat{\mathcal{L}}(\mathbf{w}_t, \boldsymbol{\lambda}_t) \,\middle|\, \mathcal{F}_t\right] = \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}_t, \boldsymbol{\lambda}_t) \,. \tag{3.17}$$

Furthermore, the compactness of the sets $\mathcal{W}$ permits the bounding of the magnitude of the iterates $\mathbf{w}_{i,t}$ by a constant R/V, which in turn implies that the network-wide iterates may be bounded in magnitude as

$$\|\mathbf{w}_t\| \leq R \text{ for all } t \,. \tag{3.18}$$

To prove convergence of the stochastic saddle point method, some conditions are required of the network, loss functions, and constraints, which we state below.

AS6 *(Network connectivity) The network $\mathcal{G}$ is symmetric and connected with diameter D.*

AS7 *(Smoothness) The stacked instantaneous objective is Lipschitz continuous in expectation with constant G_ℓ, i.e., for distinct primal variables $\mathbf{w}, \tilde{\mathbf{w}} \in \mathcal{W}$ and all $\boldsymbol{\theta}$,*

$$\mathbb{E}\left[\|\ell(\mathbf{w}, \boldsymbol{\theta}) - L(\tilde{\mathbf{w}}, \boldsymbol{\theta})\|\right] \leq G_\ell \|\mathbf{w} - \tilde{\mathbf{w}}\| \,. \tag{3.19}$$

Moreover, the stacked constraint function $h(\mathbf{w})$ is Lipschitz continuous with modulus L_h. That is, for distinct primal variables $\mathbf{w}, \tilde{\mathbf{w}} \in \mathcal{W}$, we may write

$$\|h(\mathbf{w}) - h(\tilde{\mathbf{w}})\| \leq G_h \|\mathbf{w} - \tilde{\mathbf{w}}\|. \tag{3.20}$$

AS8 *(Stochastic Gradient Variance) The expected square-magnitudes of the primal gradients of the local objectives are upper bounded, i.e.*

$$\max_{i \in V} \mathbb{E}\left[\|\nabla_{\mathbf{w}_i} \ell(\mathbf{w}_i, \boldsymbol{\theta}_i)\|^2\right] \leq \sigma_{\mathbf{w}}^2 \tag{3.21}$$

AS9 *(Existence of Optima) The set of primal-dual optimal pairs $\mathcal{W}^* \times \boldsymbol{\Lambda}^*$ of the constrained problem* (3.4) *has non-empty intersection with the feasible domain $\mathcal{W}^V \times \mathbb{R}_+^E$.*

Assumption 6 ensures that the graph is connected and the rate at which information diffuses across the network is finite. This condition is standard in distributed algorithms [154, 157]. Assumption 7 states that the stacked objective and constraints are sufficiently smooth, and have bounded gradients, a stipulation that frequently is required in the analysis of convex optimization methods [21, 42]. Assumption 8 is standard in the analysis of stochastic approximation methods [198]. Moreover, Assumption 9 establishes that the restriction to a finite primal domain $\mathcal{W}$ does not preclude our ability to find a primal-dual optimal pair of (3.4), and has been used to establish existence of solutions to constrained convex

programs [22]. It easily may be guaranteed by the existence of a strictly feasible $\mathbf{w}$, i.e., Slater's condition holds [134, Assumption 2].

Assumption 7 taken with the bound on the primal iterates [cf. (3.18)] permits the bounding of the expected primal and dual gradients of the Lagrangian by constant terms and terms that depend on the magnitude of the dual variable. In particular, we compute the mean-square-magnitude of the primal gradient of the stochastic augmented Lagrangian as

$$\begin{aligned}\mathbb{E}[\|\nabla_{\mathbf{w}}\hat{\mathcal{L}}_t(\mathbf{w},\boldsymbol{\lambda})\|^2] &\leq V\max_i \mathbb{E}\left[\|\nabla_{\mathbf{w}_i}\ell_i(\mathbf{w}_i,\boldsymbol{\theta}_t)\|^2\right] + E\|\boldsymbol{\lambda}\|^2\max_{(i,j)\in\mathcal{E}}\|\nabla_{\mathbf{w}_i}h_{ij}(\mathbf{w}_i,\mathbf{w}_j)\|^2 \\ &\leq V\sigma_{\mathbf{w}}^2 + EG_h^2\|\boldsymbol{\lambda}\|^2 \leq (V+E)G^2(1+\|\boldsymbol{\lambda}\|^2) \end{aligned} \tag{3.22}$$

where we have applied the triangle inequality in the first expression and considered the worst-case bounds. The second inequality makes use of the smoothness properties defined in (3.19) and the fact that the constraint $h_{ij}(\mathbf{w}_i,\mathbf{w}_j)$ is independent of $\boldsymbol{\theta}$. On the right-hand side of (3.22) we have defined $G := \max(\sigma_{\mathbf{w}}, G_h)$ to simplify the expression. We further may derive a bound on the expected magnitude of the dual stochastic gradient of the augmented Lagrangian by making use of Assumption 7. That is,

$$\begin{aligned}\mathbb{E}\left[\|\nabla_{\boldsymbol{\lambda}}\hat{\mathcal{L}}_t(\mathbf{w},\boldsymbol{\lambda})\|^2\right] &\leq E\max_{(i,j)\in\mathcal{E}}(h_{ij}(\mathbf{w}_i,\mathbf{w}_j)-\gamma_{ij})^2 + \delta^2\eta_t^2\|\boldsymbol{\lambda}\|^2 \\ &\leq EG_h^2\|\mathbf{w}\|^2 + \delta^2\eta_t^2\|\boldsymbol{\lambda}\|^2 \leq EG_h^2R^2 + \delta^2\eta_t^2\|\boldsymbol{\lambda}\|^2. \end{aligned} \tag{3.23}$$

The first inequality makes use of the triangle inequality and a worst-case bound on the constraint slack, whereas the second uses the Lipschitz continuity of the constraint (Assumption 7), and the last is an application of the compactness of the primal domain $\mathcal{W}^V$. We proceed with a remark.

Remark 3 Rather than bound the primal and dual gradients of the Lagrangian by constants, as is conventionally done in the analysis of primal-dual algorithms, we instead consider upper estimates in terms of the magnitude of the dual variable $\boldsymbol{\lambda}$. In doing so, we alleviate the need for the dual variable to be restricted to a compact subset of the nonnegative real numbers $\mathbb{R}_+^E$. The use of unbounded Lagrange multipliers allow us to mitigate the growth of constraint violation over time using the dual regularization term $(\delta\eta_t/2)\|\boldsymbol{\lambda}\|^2$ in (3.6).

The following lemma is used in the proof of the main theorem, and bounds the Lagrangian difference $\hat{\mathcal{L}}_t(\mathbf{w}_t,\boldsymbol{\lambda}_t) - \hat{\mathcal{L}}_t(\mathbf{w},\boldsymbol{\lambda}_t)$ by a telescopic quantity involving the primal and dual iterates, as well as the magnitude of the primal and dual gradients.

Lemma 2 *Denote as* $(\mathbf{w}_t, \boldsymbol{\lambda}_t)$ *the sequence generated by the saddle point algorithm in* (3.8) *and* (3.9) *with stepsize* η_t*. If Assumptions 6 - 9 hold, the instantaneous Lagrangian difference sequence* $\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \hat{\mathcal{L}}_t(\mathbf{w}, \boldsymbol{\lambda}_t)$ *satisfies the decrement property*

$$\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}) - \hat{\mathcal{L}}_t(\mathbf{w}, \boldsymbol{\lambda}_t) \leq \frac{1}{2\eta_t}\Big(\|\mathbf{w}_t - \mathbf{w}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}\|^2 + \|\boldsymbol{\lambda}_t - \boldsymbol{\lambda}\|^2 - \|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}\|^2\Big) + \frac{\eta_t}{2}\Big(\|\nabla_{\mathbf{w}}\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\|^2 + \|\nabla_{\boldsymbol{\lambda}}\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\|^2\Big)\,. \tag{3.24}$$

Proof: Consider the squared 2-norm of the difference between the iterate $\mathbf{w}_{t+1}$ at time $t+1$ and an arbitrary feasible point $\mathbf{w} \in \mathcal{W}^V$ and use (3.8) to express $\mathbf{w}_{t+1}$ in terms of $\mathbf{w}_t$,

$$\|\mathbf{w}_{t+1} - \mathbf{w}\|^2 = \|\mathcal{P}_{\mathcal{W}^V}[\mathbf{w}_t - \eta_t \nabla_{\mathbf{w}}\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)] - \mathbf{w}\|^2. \tag{3.25}$$

Since $\mathbf{w} \in \mathcal{W}^V$, the distance between the projected vector $\mathcal{P}_{\mathcal{W}^V}[\mathbf{w}_t - \eta_t \nabla_{\mathbf{w}}\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)]$ and $\mathbf{w}$ is smaller than the distance before projection. Use this fact in (3.25) and expand the square

$$\begin{aligned}\|\mathbf{w}_{t+1} - \mathbf{w}\|^2 &\leq \|\mathbf{w}_t - \eta_t \nabla_{\mathbf{w}}\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \mathbf{w}\|^2 \\ &= \|\mathbf{w}_t - \mathbf{w}\|^2 - 2\eta_t \nabla_{\mathbf{w}}\hat{\mathcal{L}}_t(\mathbf{w}_t, \lambda_t)^T(\mathbf{w}_t - \mathbf{w}) + \eta_t^2\|\nabla_{\mathbf{w}}\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\|^2. \end{aligned} \tag{3.26}$$

We reorder terms of the above expression such that the gradient inner product is on the left-hand side, yielding

$$\begin{aligned}&\nabla_{\mathbf{w}}\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)^T(\mathbf{w}_t - \mathbf{w}) \\ &\quad\leq \frac{1}{2\eta_t}\left(\|\mathbf{w}_t - \mathbf{w}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}\|^2\right) + \frac{\eta_t}{2}\|\nabla_{\mathbf{w}}\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\|^2.\end{aligned} \tag{3.27}$$

Observe now that since the functions $\ell_{i,t}(\mathbf{w}_i, \boldsymbol{\theta})$ and $h_{ij}(\mathbf{w}_i, \mathbf{w}_j)$ are convex, the online Lagrangian is a convex function of $\mathbf{w}$ [cf. (3.6)]. Thus, it follows from the first order convexity condition that

$$\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \hat{\mathcal{L}}_t(\mathbf{w}, \boldsymbol{\lambda}_t) \leq \nabla_{\mathbf{w}}\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)^T(\mathbf{w}_t - \mathbf{w}). \tag{3.28}$$

Substituting the upper bound in (3.27) for the right hand side of the inequality in (3.28) yields

$$\begin{aligned}&\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \hat{\mathcal{L}}_t(\mathbf{w}, \boldsymbol{\lambda}_t) \\ &\quad\leq \frac{1}{2\eta_t}\left(\|\mathbf{w}_t - \mathbf{w}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}\|^2\right) + \frac{\eta_t}{2}\|\nabla_{\mathbf{w}}\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\|^2.\end{aligned} \tag{3.29}$$

We set this analysis aside and proceed to repeat the steps in (3.25)-(3.29) for the distance between the iterate $\boldsymbol{\lambda}_{t+1}$ at time $t+1$ and an arbitrary multiplier $\boldsymbol{\lambda}$.

$$\|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}\|^2 = \|[\boldsymbol{\lambda}_t + \eta_t \nabla_{\boldsymbol{\lambda}} \hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)]_+ - \boldsymbol{\lambda}\|^2, \tag{3.30}$$

where we have substituted (3.9) to express $\boldsymbol{\lambda}_{t+1}$ in terms of $\boldsymbol{\lambda}_t$. Using the non-expansive property of the projection operator in (3.30) and expanding the square, we obtain

$$\begin{aligned} \|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}\|^2 &\leq \|\boldsymbol{\lambda}_t + \eta_t \nabla_{\boldsymbol{\lambda}} \hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \boldsymbol{\lambda}\|^2. \\ &= \|\boldsymbol{\lambda}_t - \boldsymbol{\lambda}\|^2 + 2\eta_t \nabla_{\boldsymbol{\lambda}} \hat{\mathcal{L}}_t(\mathbf{w}_t, \lambda_t)^T (\boldsymbol{\lambda}_t - \boldsymbol{\lambda}) + \eta_t^2 \|\nabla_{\boldsymbol{\lambda}} \hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\|^2. \end{aligned} \tag{3.31}$$

Reorder terms in the above expression such that the gradient-iterate inner product term is on the left-hand side as

$$\nabla_{\boldsymbol{\lambda}} \hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)^T (\boldsymbol{\lambda}_t - \boldsymbol{\lambda}) \geq \frac{1}{2\eta_t} \left(\|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}\|^2 - \|\boldsymbol{\lambda}_t - \boldsymbol{\lambda}\|^2 \right) - \frac{\eta_t}{2} \|\nabla_{\boldsymbol{\lambda}} \hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\|^2. \tag{3.32}$$

Note that the online Lagrangian [cf. (3.6)] is a concave function of its Lagrange multipliers, which implies that instantaneous Lagrangian differences for fixed $\mathbf{w}_t$ satisfy

$$\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}) \geq \nabla_{\boldsymbol{\lambda}} \hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)^T (\boldsymbol{\lambda}_t - \boldsymbol{\lambda}). \tag{3.33}$$

By using the lower bound stated in (3.32) for the right hand side of (3.33), we may write

$$\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}) \geq \frac{1}{2\eta_t} \left(\|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}\|^2 - \|\boldsymbol{\lambda}_t - \boldsymbol{\lambda}\|^2 \right) - \frac{\eta_t}{2} \|\nabla_{\boldsymbol{\lambda}} \hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\|^2. \tag{3.34}$$

We now turn to establishing a telescopic property of the instantaneous Lagrangian by combining the expressions in (3.29) and (3.34). To do so observe that the term $\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)$ appears in both inequalities. Thus, subtraction in inequality (3.34) from those in (3.29) followed by reordering terms yields

$$\begin{aligned} \hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}) - \hat{\mathcal{L}}_t(\mathbf{w}, \boldsymbol{\lambda}_t) \leq \frac{1}{2\eta_t} &\left(\|\mathbf{w}_t - \mathbf{w}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}\|^2 + \|\boldsymbol{\lambda}_t - \boldsymbol{\lambda}\|^2 - \|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}\|^2 \right) \\ &+ \frac{\eta_t}{2} \left(\|\nabla_{\mathbf{w}} \hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\|^2 + \|\nabla_{\boldsymbol{\lambda}} \hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\|^2 \right), \end{aligned} \tag{3.35}$$

which is as stated in (3.24). ■

Lemma 7 exploits the fact that the stochastic augmented Lagrangian is convex-concave with respect to its primal and dual variables to obtain an upper bound for the difference $\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \hat{\mathcal{L}}_t(\mathbf{w}, \boldsymbol{\lambda}_t)$ in terms of the difference between the primal and dual iterates to a fixed primal-dual pair $(\mathbf{w}, \boldsymbol{\lambda})$ at the next and current time, as well as the square magnitudes

of the primal and dual gradients. This property is the basis for establishing the convergence of the primal iterates to their constrained optimum given by (3.4) in terms of objective function evaluation and constraint violation, when a specific constant step-size is chosen, as we state next.

Theorem 3 *Denote* $(\mathbf{w}_t, \boldsymbol{\lambda}_t)$ *as the sequence generated by the saddle point algorithm in* (3.8)-(3.9) *and suppose Assumptions 6 - 9 hold. Suppose the algorithm is run for* T *iterations with a constant step-size selected as* $\eta_t = \eta = 1/\sqrt{T}$*, then the average time aggregation of the objective function error sequence* $\mathbb{E}L(\mathbf{w}_t) - L(\mathbf{w}^*)$*, with* $\mathbf{w}^*$ *defined as in* (3.4)*, grows sublinearly with the final iteration index* T *as*

$$\sum_{t=1}^{T} \mathbb{E}[L(\mathbf{w}_t) - L(\mathbf{w}^*)] \leq \mathcal{O}(\sqrt{T}). \tag{3.36}$$

Moreover, the time-aggregation of the average constraint violation of the algorithm grows sublinearly in final time T *as*

$$\sum_{(i,j)\in\mathcal{E}} \mathbb{E}\Big[\sum_{t=1}^{T} \Big(h_{ij}(\mathbf{w}_{i,t}, \mathbf{w}_{j,t}) - \gamma_{ij}\Big)\Big]_+ \leq \mathcal{O}(T^{3/4}). \tag{3.37}$$

Proof: We first consider the expression in (3.24), and expand the left-hand side using the definition of the augmented Lagrangian in (3.7). Doing so yields the following expression,

$$\begin{aligned}
&\sum_{i=1}^{V}[\ell_i(\mathbf{w}_{i,t}, \boldsymbol{\theta}_{i,t}) - \ell_i(\mathbf{w}_i, \boldsymbol{\theta}_{i,t})] + \frac{\delta\eta_t}{2}(\|\boldsymbol{\lambda}_t\|^2 - \|\boldsymbol{\lambda}\|^2) \qquad (3.38)\\
&\quad + \sum_{(i,j)\in\mathcal{E}} \left[\lambda_{ij}\left(h_{ij}(\mathbf{w}_{i,t}, \mathbf{w}_{j,t}) - \gamma_{ij}\right) - \lambda_{ij,t}\left(h_{ij}(\mathbf{w}_i, \mathbf{w}_j) - \gamma_{ij}\right)\right]\\
&\quad \leq \frac{1}{2\eta_t}\Big(\|\mathbf{w}_t - \mathbf{w}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}\|^2 + \|\boldsymbol{\lambda}_t - \boldsymbol{\lambda}\|^2 - \|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}\|^2\Big)\\
&\quad\quad + \frac{\eta_t}{2}\Big(\|\nabla_{\mathbf{w}}\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\|^2 + \|\nabla_{\boldsymbol{\lambda}}\hat{\mathcal{L}}_t(\mathbf{w}_t, \boldsymbol{\lambda}_t)\|^2\Big),
\end{aligned}$$

after gathering like terms. Compute the expectation of (3.38) conditional on $\mathcal{F}_0$, the sigma algebra that measures the *entire* algorithm history, and substitute in the bounds for the mean-square-magnitude of the primal and dual gradients of the stochastic augmented La-

grangian given in (3.22) and (3.23), respectively, into the right-hand side to obtain

$$\begin{aligned}
&\mathbb{E}\Big[L(\mathbf{w}_t) - L(\mathbf{w}) + \frac{\delta\eta_t}{2}(\|\boldsymbol{\lambda}_t\|^2 - \|\boldsymbol{\lambda}\|^2) \\
&\quad + \sum_{(i,j)\in\mathcal{E}} \left(\lambda_{ij}\left(h_{ij}(\mathbf{w}_{i,t},\mathbf{w}_{j,t}) - \gamma_{ij}\right) - \lambda_{ij,t}\left(h_{ij}(\mathbf{w}_i,\mathbf{w}_j) - \gamma_{ij}\right)\right)\Big] \\
&\quad \leq \mathbb{E}\Big[\frac{1}{2\eta_t}\Big(\|\mathbf{w}_t - \mathbf{w}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}\|^2 + \|\boldsymbol{\lambda}_t - \boldsymbol{\lambda}\|^2 - \|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}\|^2\Big) \\
&\qquad + \frac{\eta_t}{2}\Big((V+E)G^2(1 + \|\boldsymbol{\lambda}_t\|^2) + EG_h^2R^2 + \delta^2\eta_t^2\|\boldsymbol{\lambda}_t\|^2\Big)\Big],
\end{aligned} \tag{3.39}$$

where we have also used the fact that the constraint functions $h_{ij}(\mathbf{w}_i, \mathbf{w}_j)$ appearing as the third term on the left-hand side are independent of $\boldsymbol{\theta}$, and noting that the right-hand side of (3.39) is equal to its expectation. Observe that $\mathbf{w} \in \mathcal{W}$ is an arbitrary feasible point, which implies that $h_{ij}(\mathbf{w}_i, \mathbf{w}_j) \leq \gamma_{ij}$ for all $(i,j) \in \mathcal{E}$. Making use of this property to annihilate the last term on the left-hand side of (3.39) and subtracting $(\delta\eta_t/2)\|\boldsymbol{\lambda}_t\|^2$ from both sides yields

$$\begin{aligned}
&\mathbb{E}\Big[L(\mathbf{w}_t) - L(\mathbf{w}) + \sum_{(i,j)\in\mathcal{E}} \left(\lambda_{ij}\left(h_{ij}(\mathbf{w}_{i,t},\mathbf{w}_{j,t}) - \gamma_{ij}\right) - \frac{\delta\eta_t}{2}\lambda_{ij}^2\right)\Big] \\
&\quad \leq \mathbb{E}\Big[\frac{1}{2\eta_t}\Big(\|\mathbf{w}_t - \mathbf{w}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}\|^2 + \|\boldsymbol{\lambda}_t - \boldsymbol{\lambda}\|^2 - \|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}\|^2\Big) \\
&\qquad + \frac{\eta_t}{2}\Big(K + ((V+E)G^2 + \delta^2\eta_t^2 - \delta)\|\boldsymbol{\lambda}_t\|^2\Big)\Big].
\end{aligned} \tag{3.40}$$

after reordering terms, and defining the constant $K := (V+E)G^2 + EG_h^2R^2$. Now sum the expression (3.40) over times $t = 1, \ldots, T$ for a fixed T, and select the constant δ to satisfy $(V+E)G^2 + \delta^2\eta_t^2 \leq \delta$ for a constant step-size $\eta_t = \eta$ to drop the term involving $\|\boldsymbol{\lambda}_t\|^2$ from the right-hand side as

$$\begin{aligned}
&\mathbb{E}\Big[\sum_{t=1}^{T}[L(\mathbf{w}_t) - L(\mathbf{w})] + \sum_{(i,j)\in\mathcal{E}} \lambda_{ij}\Big(\sum_{t=1}^{T}\left(h_{ij}(\mathbf{w}_{i,t},\mathbf{w}_{j,t}) - \gamma_{ij}\right)\Big) - \frac{\delta\eta T}{2}\|\boldsymbol{\lambda}\|^2\Big] \\
&\qquad\qquad \leq \frac{1}{2\eta}\Big(\|\mathbf{w}_1 - \mathbf{w}\|^2 + \|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}\|^2\Big) + \frac{\eta TK}{2}.
\end{aligned} \tag{3.41}$$

In (3.41), we exploit the telescopic property of the summand over differences in the magnitude of primal and dual iterates to a fixed primal-dual pair $(\mathbf{w}, \boldsymbol{\lambda})$ which appears as the first term on right-hand side of (3.40), and the fact that the resulting expression is deterministic. By assuming the dual variable is initialized as $\boldsymbol{\lambda}_1 = \mathbf{0}$ and subtracting the resulting

$(1/2\eta)\|\boldsymbol{\lambda}\|^2$ term to the other side, the expression in (3.41) becomes

$$\mathbb{E}\Big[\sum_{t=1}^{T}[L(\mathbf{w}_t) - L(\mathbf{w})] + \sum_{(i,j)\in\mathcal{E}} \lambda_{ij}\Big(\sum_{t=1}^{T}\left(h_{ij}(\mathbf{w}_{i,t}, \mathbf{w}_{j,t}) - \gamma_{ij}\right)\Big) - \Big(\frac{\delta\eta T}{2} + \frac{1}{2\eta}\Big)\|\boldsymbol{\lambda}\|^2\Big] \\ \leq \frac{1}{2\eta}\|\mathbf{w}_1 - \mathbf{w}\|^2 + \frac{\eta T K}{2}. \tag{3.42}$$

At this point, we note that the left-hand side of the expression in (3.42) consists of two terms. The first is the accumulation over time of the global loss, which is a sum of all local losses at each node as defined in (3.2). The second term is the inner product of the an arbitrary Lagrange multiplier $\boldsymbol{\lambda}$ with the time-aggregation of constraint violation, and the last is a term which depends on the magnitude of this multiplier. We may use these later terms to define an "optimal" Lagrange multiplier to control the growth of the long-term constraint violation of the algorithm. This technique is inspired by the approach in [80,114]. To do so, define the *augmented* dual function $\tilde{g}(\boldsymbol{\lambda})$ using the later two terms on the left-hand side of (3.42)

$$\tilde{g}(\boldsymbol{\lambda}) = \mathbb{E}\Big[\sum_{(i,j)\in\mathcal{E}} \lambda_{ij}\Big(\sum_{t=1}^{T}\Big(\left(h_{ij}(\mathbf{w}_{i,t}, \mathbf{w}_{j,t}) - \gamma_{ij}\right)\Big)\Big) - \Big(\frac{\delta\eta T}{2} + \frac{1}{2\eta}\Big)\|\boldsymbol{\lambda}\|^2\Big]. \tag{3.43}$$

Computing the gradient of (3.43) and solving the resulting stationary equation over the range $\mathbb{R}_+^E$ yields

$$\tilde{\lambda}_{ij} = \mathbb{E}\Big[\Big(\frac{1}{2(T\delta\eta + 1/\eta)}\Big)\sum_{t=1}^{T}\Big[h_{ij}(\mathbf{w}_{i,t}, \mathbf{w}_{j,t}) - \gamma_{ij}\Big]_+\Big] \tag{3.44}$$

for all $(i,j) \in \mathcal{E}$. Substituting the selection $\boldsymbol{\lambda} = \tilde{\boldsymbol{\lambda}}$ defined by (3.44) into (3.42) results in the following expression

$$\mathbb{E}\left[\sum_{t=1}^{T}[L(\mathbf{w}_t) - L(\mathbf{w})] + \sum_{(i,j)\in\mathcal{E}} \frac{\Big[\sum_{t=1}^{T}\Big(h_{ij}(\mathbf{w}_{i,t}, \mathbf{w}_{j,t}) - \gamma_{ij}\Big)\Big]_+^2}{2(T\delta\eta + 1/\eta)}\right] \leq \frac{1}{2\eta}\|\mathbf{w}_1 - \mathbf{w}\|^2 + \frac{\eta T K}{2}. \tag{3.45}$$

Now select the constant step-size $\eta = 1/\sqrt{T}$, and substitute the result into (3.45), using the

formula for K defined following expression (3.40), to obtain

$$\mathbb{E}\Bigg[\sum_{t=1}^{T}[L(\mathbf{w}_t) - L(\mathbf{w})] + \sum_{(i,j)\in\mathcal{E}} \frac{\Big[\sum_{t=1}^{T}\Big(h_{ij}(\mathbf{w}_{i,t}, \mathbf{w}_{j,t}) - \gamma_{ij}\Big)\Big]_+^2}{2\sqrt{T}(\delta+1)}\Bigg]$$
$$\leq \frac{\sqrt{T}}{2}\left(\|\mathbf{w}_1 - \mathbf{w}\|^2 + (V+E)G^2 + EG_h^2R^2\right). \tag{3.46}$$

The expression in (3.46) allows us to derive both the convergence of the global objective and the feasibility of the stochastic saddle point iterates.

We first consider the average objective error sequence $\mathbb{E}[L(\mathbf{w}_t) - L(\mathbf{w}^*)]$. To do so, subtract the last term on the left-hand side of (3.46) from both sides, and note that the resulting term is non-positive. This observation allows us to omit the constraint slack term in (3.46), which taken with the selection $\mathbf{w} = \mathbf{w}^*$ [cf. (3.4)] and pulling the expectation inside the summand, yields

$$\sum_{t=1}^{T}\mathbb{E}[(L(\mathbf{w}_t) - L(\mathbf{w}^*))] \leq \frac{\sqrt{T}}{2}\left(\|\mathbf{w}_1 - \mathbf{w}^*\|^2 + K\right) = \mathcal{O}(\sqrt{T}), \tag{3.47}$$

which is as stated in (3.36).

Now we turn to establishing a sublinear growth of the constraint violation in T, using the expression in (3.46). First, observe that the objective function error sequence is bounded above as

$$L(\mathbf{w}_t) - L(\mathbf{w}^*) \leq G_\ell\|\mathbf{w}_t - \mathbf{w}^*\| \leq 2G_\ell R \tag{3.48}$$

An immediate implication of (3.48) is the relation $L(\mathbf{w}_t) - L(\mathbf{w}^*) \geq -2G_\ell R$, which may be obtained by switching the order, and again applying the Lipschitz continuity of F with the compactness of $\mathcal{W}^V$. Substituting this lower bound for the objective function error sequence into the first term on the left-hand side of (3.46) and adding the result to both sides yields

$$\mathbb{E}\Bigg[\sum_{(i,j)\in\mathcal{E}} \frac{\Big[\sum_{t=1}^{T}\Big(h_{ij}(\mathbf{w}_{i,t}, \mathbf{w}_{j,t}) - \gamma_{ij}\Big)\Big]_+^2}{2\sqrt{T}(\delta+1)}\Bigg] \leq \frac{\sqrt{T}}{2}\left(\|\mathbf{w}_1 - \mathbf{w}^*\|^2 + K\right) + 2TG_\ell R. \tag{3.49}$$

which, after multiplying both sides by $2\sqrt{T}(\delta+1)$ yields

$$\mathbb{E}\Bigg[\sum_{(i,j)\in\mathcal{E}}\Big[\sum_{t=1}^{T}\Big(h_{ij}(\mathbf{w}_{i,t}, \mathbf{w}_{j,t}) - \gamma_{ij}\Big)\Big]_+^2\Bigg] \leq \Big(2\sqrt{T}(\delta+1)\Big)\Big(\frac{\sqrt{T}}{2}\left(\|\mathbf{w}_1 - \mathbf{w}^*\|^2 + K\right) + 2TG_\ell R\Big). \tag{3.50}$$

We complete the proof by noting that the square of the network-in-aggregate constraint violation $\sum_{(i,j)}[\sum_{t=1}^{T}(h_{ij}(\mathbf{w}_{i,t},\mathbf{w}_{j,t})-\gamma_{ij})]_+^2$ upper bounds the square of individual proximity constraint violations since it is a sum of positive squared terms, i.e.,

$$\mathbb{E}\left[\sum_{(i,j)\mathcal{E}}\Big[\sum_{t=1}^{T}(h_{ij}(\mathbf{w}_{i,t},\mathbf{w}_{j,t})-\gamma_{ij})\Big]_+^2\right] \geq \left[\Big[\sum_{t=1}^{T}(h_{ij}(\mathbf{w}_{i,t},\mathbf{w}_{j,t})-\gamma_{ij})\Big]_+^2\right]. \tag{3.51}$$

Thus the right-hand side of (3.51) may be used in place of the left-hand side of (3.50), implying that

$$\mathbb{E}\left[\Big[\sum_{t=1}^{T}(h_{ij}(\mathbf{w}_{i,t},\mathbf{w}_{j,t})-\gamma_{ij})\Big]_+^2\right] \leq \Big(2\sqrt{T}(\delta+1)\Big)\Big(\frac{\sqrt{T}}{2}\left(\|\mathbf{w}_1-\mathbf{w}^*\|^2+K\right)+2TG_\ell R\Big)\,. \tag{3.52}$$

Compute the square root of both sides of (3.52), and sum the resulting expression over all $(i,j)\in\mathcal{E}$ to conclude (3.37). ■

Theorem 3 establishes that the stochastic saddle point method, when run with a fixed algorithm step-size, yields an objective function error sequence whose difference is bounded by a constant strictly less times than T, the final iteration index. Moreover, the time-accumulation of the constraint violation incurred by the algorithm is strictly smaller than T, the final iteration index. Thus, for larger T, the iterate average difference between $L(\mathbf{w}_t)$ and $L(\mathbf{w}^*)$ goes to null in expectation, as does the average constraint violation in expectation.

This result is comparable to results for stochastic gradient method for unconstrained weakly convex problems with constant step-sizes with no smoothness assumptions, provided the data domain and feasible set are compact. In this setting, convergence to a neighborhood on the order of $T\eta$ is standard – see [138], Section 2.2, eqn. 2.19, or [14], Section 4, for instance. In such cases, convergence to a neighborhood of size $\mathcal{O}(\eta T)$ is attained in terms of primal sub-optimality for the time-average vector, and the step-size is chosen as $\eta=\mathcal{O}(1/\sqrt{T})$ to balance the growth of constant terms with the minimizing of neighborhood size. It must be noted, however, that the neighborhood for accumulation of constraint violation $\mathcal{O}(\eta T^{5/4})$ is larger than the primal sub-optimality, yielding the larger accumulation of constraint violation over T as $\mathcal{O}(T^{3/4})$ for this step-size choice. The reason we present results in this way is to draw the connection with regret analysis in online learning [228]. .

Theorem 3 also allows us to establish convergence of the average iterates to a specific level of accuracy dependent on the total number of iterations T, as we subsequently state.

Corollary 1 *Let $\bar{\mathbf{w}}_T=(1/T)\sum_{t=1}^{T}\mathbf{w}_t$ be the vector formed by averaging the primal iterates $\mathbf{w}_t$ over times $t=1,\ldots,T$. Under Assumptions 6 - 9, with constant algorithm step-size*

$\eta_t = 1/\sqrt{T}$, *the objective function evaluated at* $\bar{\mathbf{w}}_T$ *satisfies*

$$\mathbb{E}\big[L(\bar{\mathbf{w}}_T) - L(\mathbf{w}^*)\big] \leq \mathcal{O}(1/\sqrt{T}) \tag{3.53}$$

Moreover, the constraint violation evaluated at the average vector $\bar{\mathbf{w}}_T$ *satisfies*

$$\mathbb{E}\big[\sum_{(i,j)\in\mathcal{E}} \big[h_{ij}(\bar{\mathbf{w}}_{i,T}, \bar{\mathbf{w}}_{j,T}) - \gamma_{ij}\big]_+\big] = \mathcal{O}(T^{-\frac{1}{4}}). \tag{3.54}$$

Proof: Consider the expressions in Theorem 3. In particular, to prove (3.53), we consider the expression in (3.36), divide the expression by T, and use the definition of convexity of the expected objective $\mathbb{E}[L(\mathbf{w})]$ which says that the average of average function values upper bounds the average function evaluated at the average vector, i.e.

$$\mathbb{E}[L(\bar{\mathbf{w}}_T)] \leq \mathbb{E}\Big[\frac{1}{T}\sum_{t=1}^{T} L(\mathbf{w}_t)\Big] \tag{3.55}$$

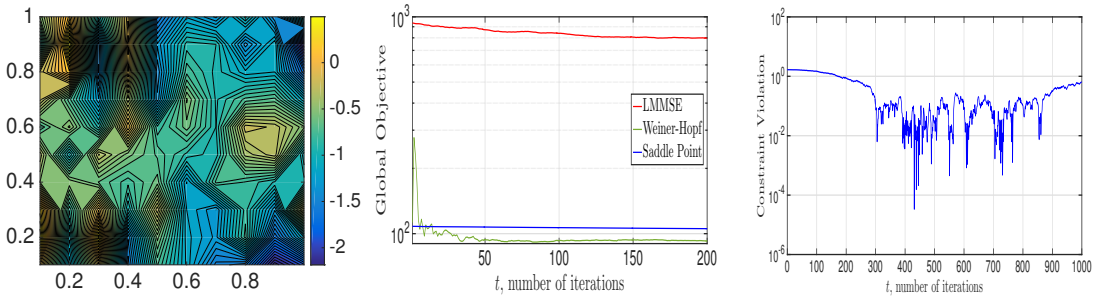
and similarly for the average of the expected constraint functions $\mathbb{E}[h_{ij}(\mathbf{w}_i, \mathbf{w}_i)]$,

$$\mathbb{E}[h_{ij}(\bar{\mathbf{w}}_{i,T}, \bar{\mathbf{w}}_{j,T})] \leq \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} h_{ij}(\bar{\mathbf{w}}_{i,t}, \bar{\mathbf{w}}_{j,t})]\right] \tag{3.56}$$

Apply the relation (3.55) to the expressions in (3.36) divided by T to obtain (3.53). To conclude, (3.54) we apply (3.56) to each term in the summand (3.37) divided by T. ■

Corollary 1 shows that the average saddle point primal iterates $\bar{\mathbf{w}}_T$ converge to within a margin $\mathcal{O}(1/\sqrt{T})$ in terms of objective function evaluation to the optimal objective $L(\mathbf{w}^*)$ on average, where T is the number of iterations. Moreover, the primal average vector also yields the bound in expectation on the network proximity constraint violation as $\mathcal{O}(T^{-1/4})$. We note that for a fixed T, this result amounts to convergence to a neighborhood on average. The radius of this neighborhood crucially depends on using the expressions in (3.22) and (3.23), which are the variances of the primal and dual gradients of the stochastic augmented Lagrangian (3.7), respectively.

After cancelling out key terms in the proof, the remaining constant $(\|\mathbf{w}_1 - \mathbf{w}^*\|^2 + (V + E)G^2 + EG_h^2R^2)$ on the right-hand side of (3.47) determines the radius of convergence, for a fixed T, where we have substituted in the definition of $K = (V + E)G^2 + EG_h^2R^2$. This expression depends on initialization $\mathbf{w}_1$, the size of the network, the Lipschitz modulus of continuity of the objective and constraints (Assumption 7), and the diameter of set $\mathcal{W}$ (3.18). Similarly, for the constraint violation, the constant in front terms involving T on the right-hand side of (3.52) depend on the initialization, the dual regularization constant



(a) Snapshot of field across $\mathcal{A}$ (b) Ob. $L(\mathbf{w}_t)$ vs. iter. t (c) Cons. Violation vs. iter. t

Figure 3.1: Saddle point algorithm applied to the problem of estimating a correlated random field. Nodes are deployed uniformly in a square region of size 200×200 meters in a grid formation (at the integer lattice within the Cartesian plane), and node estimators are correlated according to the distance-based model $\rho(\mathbf{w}_i, \mathbf{w}_j) = e^{-\|l_i - l_j\|}$, where $\mathbf{w}_i$ and $\mathbf{w}_j$ are the decisions of nodes i and j, and l_j are their respective locations. A normalized snapshot of the field at time $t = 45$ is given in Fig. 3.1(a) – observe that nearby values are similar. The saddle point method achieves comparable accuracy to the Weiner-Hopf filter, and far outperforms a simple LMMSE estimator which ignores observation correlation. The saddle point method achieves this performance by satisfying proximity constraints that encode sensor correlations (Fig. 3.1(c)).

δ, $K = (V + E)G^2 + EG_h^2R^2$, as discussed above, the objective Lipschitz constant G_ℓ, and the diameter R of set $\mathcal{W}$.

The convergence of recursively averaged saddle point iterates with constant step-size has appeared in the deterministic setting in [134] and in the context of regret for online learning in [114]. Theorem 3 and Corollary 1 are the first attempts at translating this type of result into the constrained stochastic programming case with weakly convex objectives. In doing so, we attain comparable rates to stochastic gradient method for the weakly convex unconstrained stochastic case [14, 138] for the primal sub-optimality, but slower rates for the reduction of constraint violation.

3.4 Random Field Estimation

Consider the task of estimating a planar spatially correlated Gaussian random field in a specified region $\mathcal{A} \subset \mathbb{R}^2$. A planar random field is a random function of spatial components u and v, which index the value of the field across region $\mathcal{A}$ (u for x-axis, v for y-axis). The random field is further parameterized by the correlation matrix $\mathbf{R}_w$, which is assumed to follow a spatial correlation structure of the form $\rho(\mathbf{w}_i, \mathbf{w}_j) = e^{-\|l_i - l_j\|}$, where $l_i \in \mathcal{A}$ and $l_j \in \mathcal{A}$ are the respective locations of sensor i and sensor j in the deployed region, see, e.g., [55]. Observe that now each node has a unique signal-to-noise ratio based upon its location and that more distant nodes are less important; however, their contribution to the aggregate objective $L(\mathbf{w})$ still incentivizes global coordination.

We consider making use of a sensor network (the example in Section 3.1 and 3.2). Sensors collect observations $\boldsymbol{\theta}_{i,t}$ which are noisy linear transformations of the value of the field $\mathbf{w}_{u,v}(t) \in \mathbb{R}^p$ they would like to estimate at time t. That is, we consider the observation model $\boldsymbol{\theta}_{it} = \mathbf{H}_i \mathbf{w}_{u,v}(t) + \boldsymbol{\omega}_{i,t}$ with Gaussian noise $\boldsymbol{\omega}_{i,t} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_q)$ that is i.i.d across time and node, with $\sigma^2 = 2$. The goal is for each sensor to sequentially minimize its local estimation error, which amounts to online maximum likelihood estimation where the estimators of distinct sensors depend on one another.

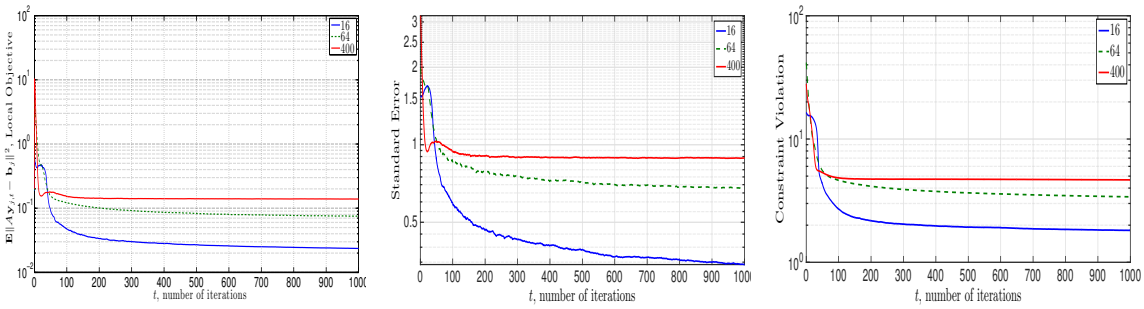
To solve this problem, we deploy $V = 100$ sensors in a grid formation along the scaled positive integer lattice, where neighboring nodes have a constant distance from one another in a 200×200 meter square region $\mathcal{A} = \{(x, y) : 200 \geq x \geq 0, 200 \geq y \geq 0\}$. At each instantaneous time, then, the observations across the network are given by $\mathbf{w}_t = \boldsymbol{\mu} + \mathbf{C}^T \mathbf{z}_t$, where $\boldsymbol{\mu}$ is a fixed mean vector of length V chosen uniformly at random from the fractions $\{1/V, 2/V, \ldots, 1\}$, $\mathbf{C}$ denotes the Cholesky factorization of the correlation matrix $\mathbf{R}_w$, and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ is a Gaussian random vector – see, for instance, [151]. An example instance of the field values observed by the deployed grid network (rescaled within the unit box) are displayed in Figure 3.1(a). Observe that nearby values are similar to each other.

We make use of the saddle point algorithm [cf. (3.10) - (3.11)], whose updates for the random field estimation problem are given by the explicit expressions in (3.14) and (3.15), respectively. We select $\gamma_{ij} = \rho(\mathbf{w}_i, \mathbf{w}_j)$. Besides the local and global losses which on average converge to a neighborhood of the constrained optima depending on the final iteration index T when a constant step-size is used (Theorem 3), we also study the amount of constraint violation over time, stated as $\sum_{j \in n_i} (\|\mathbf{w}_{i,t} - \mathbf{w}_{j,t}\|^2 - \gamma_{ij})$.

To compute $\mathbf{w}^*$ for a single time slot, stack observations $\boldsymbol{\theta} = [\boldsymbol{\theta}_1; \cdots; \boldsymbol{\theta}_V]$ and observation models $\mathbf{H} = [\mathbf{H}_1; \cdots; \mathbf{H}_V]$. Then the optimal estimator is the one that solves the weighted least-squares estimate derived from the Weiner-Hopf equations $\mathbf{w}^* = (\mathbf{H}\mathbf{R}_w\mathbf{H}^T + \frac{1}{\sigma^2}\mathbf{I})^{-1}(\mathbf{H} + \frac{1}{\sigma^2}\mathbf{I})\mathbf{R}_w\boldsymbol{\theta}$. The optimal estimator $\mathbf{w}^*$ is the one that would dictate stacking signals $\boldsymbol{\theta}_{i,t}$ for all nodes i and times t at a centralized location into one large linear system and substituting the sample variance $\hat{\sigma}^2$ in the prior computation. We consider an incremental variant of such a strategy, similar to the Levinson-Durbin recursion [86].

We consider problem instances where observations and signal estimates are scalar ($p = q = 1$), the scalar $\mathbf{H} = 1$, and the field is set a vector of ones, and run the algorithm for for $T = 1000$ iterations with a constant step-size strategy $\eta_t = 10^{-2.75}$. We further select the dual regularization parameter $\delta = 10^{-5}$. The noise level is set to $\sigma^2 = 10$. We compare the performance of the algorithm with that of a simple LMMSE estimator strategy which does not take advantage of the correlation structure of the sensor network, as well as the sequential implementation of a Weiner-Hopf estimator which optimally exploits correlation.

In Figure 3.1, we plot the results of this numerical experiment. Figure 3.1(b) shows the



(a) Local obj. vs. iter. t (b) Error $\|\mathbf{w}_{i,t} - \mathbf{w}^*\|$ vs. iter. t (c) Cons. violation vs. t

Figure 3.2: Comparison of the saddle point method with proximity constraints [cf. (3.62) - (3.63)] with dual regularization $\delta = 10^{-7}$ and hybrid step-size strategy $\eta_t = \min(\eta, \eta t_0/t)$ with $t_0 = 100$ and $\eta = 10^{-1.5}$ on the source localization problem stated in (3.57) using the convex approximation (3.58). We fix the network topology as a grid and vary the number of sensors as $V = 16$, $V = 64$, and $V = 400$ which are deployed in a square region of size 1000×1000 meters. The noise perturbing observations at sensor i is zero-mean Gaussian, with a variance proportional its distance to the source as $\sigma^2 = 0.5\|\mathbf{l}_i - \mathbf{w}^*\|$, where $\mathbf{l}_i$ is the location of node i. Observe that in larger networks, the rate at which nodes are able to localize the source is slower in terms of objective convergence and standard error to the true source. Moreover, we see that the level of constraint violation is larger with increasing V.

global objective $(1/V)\sum_i \mathbb{E}_{\boldsymbol{\theta}_i}[\ell_i(\mathbf{w}_{i,t}, \boldsymbol{\theta}_i)]$ versus iteration t. We note that the numerical behavior of the local objective is similar to the global objective, and is thus omitted. We see that when nodes incorporate the correlation structure of the random field into their estimation strategy via the quadratic proximity constraint with γ_{ij} chosen according to the correlation of node i and its neighbors $j \in n_i$, the estimation performance improves. We observe that for small t, the saddle point method outperforms the Weiner-Hopf estimator in terms of estimation accuracy, but after a burn in period the later performs more favorably. In contrast, the LMMSE estimator which ignores correlation does not appear to yield an effective tool for this context.

In Figure 3.1(c), we plot the local constraint violation of an arbitrarily chosen sensor $i \in V$, and observe that the algorithm successsfully keeps the estimates of node i close to those of its neighbors, where the closeness constraint is given by the correlation structure of the random field. Thus by using proximity constraints, individual sensors are successfully able to incorporate spatial information about the random field into their estimation.

3.5 Source Localization

We now consider the use of the stochastic saddle point method given in (3.8) - (3.9) to solve an online source localization problem. In particular, we consider an array of V sensors, where $\mathbf{l}_i \in \mathbb{R}^p$ denotes the position of the sensor i in some deployed environment $\mathcal{A} \subset \mathbb{R}^p$. Each

node seeks to learn the location of a source signal $\mathbf{w} \in \mathbb{R}^p$ through its access to noisy range observations of the form $r_{i,t} = \|\mathbf{w} - \mathbf{l}_i\| + \varepsilon_{i,t}$ where $\varepsilon_t = [\varepsilon_{1,t}; \cdots; \varepsilon_{V,t}]$ is some unknown noise vector. The goal of each sensor i in the network is, given access to sequentially observed range measurements $r_{i,t}$, to learn the position of the source $\mathbf{w}$, assuming it is aware of its location $\mathbf{l}_i$ in the deployed region. Range-based source localization has been studied in a variety of fields, from wireless communications to geophysics [102, 164].

Rather than considering a range-based least squares problem, which is nonconvex and may be solved approximately using semidefinite relaxations [46], we consider the squared range-based least squares (SR-LS) problem, stated as

$$\mathbf{w}^* := \underset{\mathbf{w} \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{V} \mathbb{E}_{\mathbf{r}_i} \Big(\|\mathbf{l}_i - \mathbf{w}\|^2 - r_i^2 \Big)^2 . \tag{3.57}$$

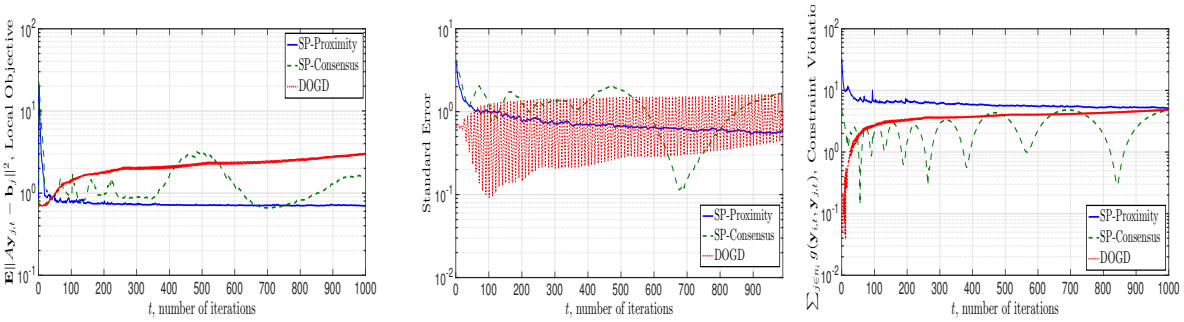
Although this problem is also nonconvex (due to, for instance, the fact that when the outer square is expanded, a quartic term appears), it may be solved approximately in a lower-complexity manner as a quadratic program – see [19], Section II-B and references therein. To do so, expand the square in the first term in the objective stated in (3.57) and consider the modified argument inside the expectation $(\alpha - 2\mathbf{l}_i^T \mathbf{w} + \|\mathbf{l}_i\|^2 - r_i^2)^2$ with the constraint $\|\mathbf{w}\|^2 = \alpha$. Proceeding as in [19], Section II-B, approximate this transformation by a convex unconstrained problem by defining matrix $\mathbf{A} \in \mathbb{R}^{V \times (p+1)}$ whose ith row associated with sensor i is given as $\mathbf{A}_i = [-2\mathbf{l}_i^T; 1]$, and vector $\mathbf{b} \in \mathbb{R}^V$ with ith entry as $\mathbf{b}_i = r_i^2 - \|\mathbf{l}_i\|^2$ and relaxing the constraint $\|\mathbf{w}\|^2 = \alpha$. Further define $\mathbf{y} = [\mathbf{w}; \alpha] \in \mathbb{R}^{p+1}$. Then, by dropping a quadratic equality constraint induced by this change of variables, (3.57) may be approximated as

$$\mathbf{y}^* := \underset{\mathbf{y} \in \mathbb{R}^{p+1}}{\operatorname{argmin}} \sum_{i=1}^{V} \mathbb{E}_{\mathbf{b}_i} \Big(\|\mathbf{A}_i \mathbf{y} - \mathbf{b}_i\|^2 \Big) ; \tag{3.58}$$

which is a least mean-square error problem. We note that the techniques in [19] to solve this problem exactly do not apply to the online setting [130].

We propose solving (3.58) in decentralized settings which more effectively allow for each sensor to operate based on real-time observations. To do so, each sensor keeps a local copy $\mathbf{y}_i$ of the global source estimate $\mathbf{y}$ based on information that is available with local information only and via message exchange with neighboring sensors. However, each sensor would still like to attain the greater estimation accuracy associated with aggregating range observations over the entire network. We proceed to illustrate how this may be achieved by using the proximity constrained optimization in Section 3.1.

In application domains such as wireless communications or acoustics [166], the quality of



(a) Local ob. vs. iter. t (b) Error $\|\mathbf{w}_{i,t} - \mathbf{w}^*\|$ vs. iter. t (c) Cons. violation vs. iter. t

Figure 3.3: Comparison of proximity and consensus algorithms on the source localization problem stated in (3.57) using the convex approximation (3.58) for an $V = 64$ node grid network deployed as an 8×8 square in a 1000×1000 meter region for the case that the noise perturbing observations at node i is zero-mean Gaussian, with a variance proportional its distance to the source as $\sigma^2 = 2\|\mathbf{l}_i - \mathbf{w}^*\|$, where $\mathbf{l}_i$ is the location of node i. We run the saddle point method with proximity constraints [cf (3.62) - (3.63)] given in (3.61) using dual regularization $\delta = 10^{-7}$, as compared with the saddle point method which executes a consensus constraint (3.3), as well as Distributed Online Gradient Descent (DOGD) [197], which is a weighted averaging gradient method. For the former two, we use hybrid step-size strategy $\eta_t = \min(\eta, \eta t_0/t)$ with $t_0 = 100$ and $\eta = 10^{-1.5}$, and for DOGD we use constant step-size $10^{-1.5}$. We observe that the proximity-constrained saddle point method yields the best performance in terms of objective convergence and standard error, although it incurs higher levels of constraint violation.

the observed range measurements is better for sensors which are in closer proximity to the source. Motivated by this fact, we consider the case where sensor i weights the importance of neighboring sensors $j \in n_i$ by aiming to keep its estimate $\mathbf{w}_i$ within an ℓ_2 ball centered at its neighbors estimate $\mathbf{w}_j$, whose radius is given by the pairwise minimum of the estimated distance to the source. This goal may be achieved via the quadratic inequality constraint

$$\|\mathbf{w}_i - \mathbf{w}_j\|^2 \leq \min\{\|\mathbf{w}_i - \mathbf{l}_i\|^2, \|\mathbf{w}_j - \mathbf{l}_j\|^2\} \text{ for all } j \in n_i \ . \tag{3.59}$$

Observe that (3.58) with the constraint (3.59) is a non-convex variant of a QCQP due to the minimum on the right-hand side (see, for instance, [6]). We may convexify the constraint by rearranging the right-hand side of (3.59) and replacing the resulting maximum by the log-sum-exp function – see [33], Chapter 2. Thus we obtain

$$(1/2)\Big(\|\mathbf{w}_i - \mathbf{w}_j\|^2 + \log\left(e^{\|\mathbf{w}_i - \mathbf{l}_i\|^2} + e^{\|\mathbf{w}_j - \mathbf{l}_j\|^2}\right)\Big) \leq 0 \ , \tag{3.60}$$

which is a convex constraint, since the later term is a composition of a monotone function with a convex function. Taking (3.58) together with the constraint (3.60), and noting that a constraint on $\mathbf{w}_i$ is equivalent to a constraint on the first p entries of $\mathbf{y}_i$ after appending

a 0 to the $p+1$-th entry of $\mathbf{l}_i$, we may write

$$\begin{aligned} \min_{\mathbf{y}\in\mathbb{R}^{N(p+1)}} \quad & \sum_{i=1}^{V} \mathbb{E}_{\mathbf{b}_i}\Big(\|\mathbf{A}_i\mathbf{y}_i - \mathbf{b}_i\|^2\Big), \\ \text{s.t.} \quad & (1/2)\Big(\|\mathbf{y}_i - \mathbf{y}_j\|^2 + \log\Big(e^{\|\mathbf{y}_i-\mathbf{l}_i\|^2} + e^{\|\mathbf{y}_j-\mathbf{l}_j\|^2}\Big)\Big) \leq 0, \end{aligned} \tag{3.61}$$

where the constraint for node i is with respect to all of its neighbors $j \in n_i$. Observe that the problem in (3.61) is of the form (3.4). Define $g(\mathbf{y}_i, \mathbf{y}_j)$ as the constraint function the left-hand side of (3.60). Then primal update of the saddle point method stated in (3.8) specialized to this problem setting for node i is stated as

$$\mathbf{y}_{i,t+1} = \mathbf{y}_{i,t} - \eta_t\Big(2\mathbf{A}_{i,t}^T\big(\mathbf{A}_{i,t}\mathbf{y}_{i,t} - \mathbf{b}_{i,t}\big) + \sum_{j\in n_i}\boldsymbol{\lambda}_{ij,t}\Big(\frac{e^{\|\mathbf{y}_{i,t}-\mathbf{l}_i\|^2}(\mathbf{y}_{i,t}-\mathbf{l}_i)}{e^{\|\mathbf{y}_{i,t}-\mathbf{l}_i\|^2} + e^{\|\mathbf{y}_{j,t}-\mathbf{l}_j\|^2}} + (\mathbf{y}_{i,t} - \mathbf{y}_{j,t})\Big), \tag{3.62}$$

where we omit the use of set projections for simplicity, while the dual update [cf. (3.9)] executed at the link layer of the sensor network is

$$\boldsymbol{\lambda}_{ij,t+1} = \big[(1-\delta\eta_t^2)\boldsymbol{\lambda}_{ij,t} + \eta_t g(\mathbf{y}_{i,t}, \mathbf{y}_{j,t})\big]_+. \tag{3.63}$$

We turn to analyzing the empirical the performance of the saddle point updates (3.62) - (3.63) to solve localization problems in a decentralized manner, such that nodes more strongly weight the importance of sensors in closer proximity to the source in the sense of (3.60). Besides the local objective $\mathbb{E}_{\mathbf{b}_i}\|\mathbf{A}_i\mathbf{y}_i - \mathbf{b}_i\|^2$, which we know converges to its contained optimal value, we also study the standard error to the source signal $\mathbf{w}^*$, denoted as $\|\mathbf{w}_{i,t} - \mathbf{w}^*\|$. Recall that we recover $\mathbf{w}_{i,t}$ from $\mathbf{y}_{i,t}$ by taking its first p elements. We further consider the magnitude of the constraint violation for this problem, which when considering the proximity constrained problem in (3.61), is given by

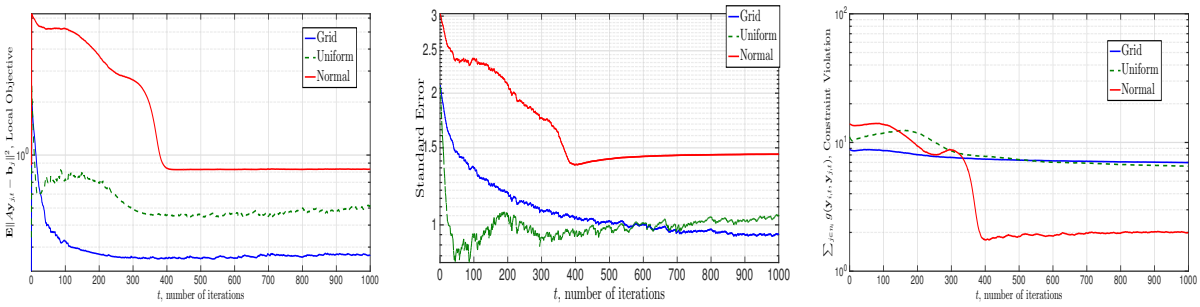
$$\begin{aligned} \sum_{j\in n_i}(1/2)g(\mathbf{y}_{i,t}, \mathbf{y}_{j,t}) = \sum_{j\in n_i}(1/2)\Big(&\|\mathbf{y}_{i,t} - \mathbf{y}_{j,t}\|^2 \\ &+ \log\Big(e^{\|\mathbf{y}_{i,t}-\mathbf{l}_{i,t}\|^2} + e^{\|\mathbf{y}_{j,t}-\mathbf{l}_j\|^2}\Big)\Big), \end{aligned} \tag{3.64}$$

and when implementing consensus methods, is given by

$$\sum_{j\in n_i} h(\mathbf{y}_{i,t}, \mathbf{y}_{j,t}) = \sum_{j\in n_i}\|\mathbf{y}_{i,t} - \mathbf{y}_{j,t}\| \tag{3.65}$$

for a randomly chosen sensor in the network.

Throughout the rest of this section, we fix the dual regularization parameter $\delta = 10^{-7}$,



(a) Local obj. vs. iter. t (b) Error $\|\mathbf{w}_{i,t} - \mathbf{w}^*\|$ vs. iter. t (c) Cons. violation vs. iter. t

Figure 3.4: Numerical results on the localization problem stated in (3.57) using the convex approximation (3.58) for the saddle point method with proximity constraints [cf. (3.62) - (3.63)] with hybrid step-size strategy $\eta_t = \min(\eta, \eta t_0/t)$ with $t_0 = 100$ and $\eta = 10^{-1.5}$. We run the algorithm for a variety of spatial deployment strategies, which induce different network topologies. We consider a square region of size 1000×1000 meters, and deploy nodes in grid formations, uniformly at random, and according to a two-dimensional Normal distribution. In the later two cases, sensors which are a distance of 50 meters or less are connected by an edge. The noise perturbing observations at sensor i is zero-mean Gaussian, with a variance proportional to its distance to the source as $\sigma^2 = 0.5\|\mathbf{l}_i - \mathbf{w}^*\|$, where $\mathbf{l}_i$ is the location of node i. We see that while the Normal configuration yields the worst localization performance, it achieves the lowest levels of constraint violation. In contrast, Uniform and Grid configurations both are effective spatial deployment strategies to localize the source in terms of local objective convergence and standard error.

and study the performance of the saddle point method with proximity constraints as compared with two methods which attempt to satisfy consensus constraints. We further analyze the saddle point method in (3.62) - (3.63) for a variety of network sizes to understand the practical effect of the learning rate on the number of sensors, and for different spatial deployment strategies which induce different network topologies.

3.5.1 Consensus Comparison

In this subsection, we compare the saddle point method on a proximity constrained problem as compared with methods which implement variations of the consensus protocol. In particular, we run the saddle point method for the localization problem given in (3.62) - (3.63) with proximity constraints, as compared with the same primal-dual scheme when the consensus constraint in (3.3) is used. We further compare these instantiations of the saddle point method with distributed online gradient descent (DOGD) [197], which is a scheme that operates by having each node selecting its next iterate by taking a weighted average of its neighbors and descending through the negative of the local stochastic gradient. For each of these methods, we run the localization procedure for a total of 1000 iterations for $\tilde{T} = 100$ different runs when each node initializes its local variable $\mathbf{y}_{i,0}$ uniformly at random from the unit interval, and plot the sample mean of the results.

We consider problem instances of (3.57) when the number of sensors is fixed at $V = 64$, and are spatially deployed in a grid formation as an 8×8 square in a planar ($p = 2$) region of size 1000×1000. Moreover, the noise perturbing the observations at node i is zero-mean Gaussianan, with a variance proportional its distance to the source as $\sigma^2 = 2\|\mathbf{l}_i - \mathbf{w}^*\|$, where $\mathbf{l}_i$ is the location of node i. The true source signal $\mathbf{w}^*$ is located at the average location of the sensors. For the saddle point methods, we find a hybrid step-size strategy to be most effective, and hence set $\eta_t = \min(\eta, \eta t_0/t)$ with $t_0 = 100$ and $\eta = 10^{-1.5}$. For DOGD, we find best performance to correspond to using a constant outer step-size $\eta = 10^{-1.5}$, along with a halving scheme step-size in the inner recursive averaging loop [197].

We plot the results Figure 3.3 for an arbitrarily chosen node $i \in V$. Observe that the saddle point method which implements the network proximity constraints method yields the best performance in terms of objective convergence. In particular, by $t = 500$ iterations, in Figure 3.3(a) we observe that the saddle point algorithm implemented with proximity constraints (SP-Proximity) achieves objective convergence to a neighborhood, i.e., $\mathbb{E}_{\mathbf{b}_i}\|\mathbf{A}_i\mathbf{y}_{i,t} - \mathbf{b}_i\|^2 \leq 1$. In contrast, the saddle point with consensus constraints (SP-Consensus) and DOGD respectively experience numerical oscillations and divergent behavior after a burn-in period of $t = 100$.

This trend is confirmed in the plot of the standard error to the optimizer $\|\mathbf{w}_{i,t} - \mathbf{w}^*\|$ of the original problem (3.57) in Figure 3.3(b). We see that SP-Proximity yields convergence to a neighborhood between 10^{-1} and 1 by $t = 200$ iterations, whereas SP-Consensus and DOGD experience numerical oscillations and do not appear to localize the source signal $\mathbf{w}^*$. While SP-Proximity exhibits superior behavior in terms of objective and standard error convergence, it incurs larger levels of constraint violation than its consensus counterpoints, as may be observed in Figure 3.3(c). To be specific, SP-Proximity on average experiences constraint violation [cf. (3.64)] on average an order of magnitude larger than SP-Consensus and DOGD [cf. (3.65)] for the first $t = 400$ iterations. After this benchmark, the magnitude of the constraint of the different methods converges to around 5. Thus, we see that achieving smaller constraint violation and implementing consensus constraints may lead to inferior source localization accuracy.

3.5.2 Impact of Network Size

In this subsection, we study the effect of the size of the deployed sensor network on the ability of the proximity-constrained saddle point method to effectively localize the source signal. We fix the topology of the deployed sensors as a grid network, and again set the source signal $\mathbf{w}^*$ to be the average of node positions in a planar ($p = 2$) spatial region $\mathcal{A}$ of size 1000×1000 meters. We set the noise distribution which perturbs the range measurements of node i to be zero-mean Gaussian with variance $.5\|\mathbf{l}_i - \mathbf{w}^*\|$. We run the

algorithm stated in (3.62) - (3.63) with hybrid step-size strategy $\eta_t = \min(\eta, \eta t_0/t)$ with $t_0 = 100$ and $\eta = 10^{-1.5}$ for a total of $T = 1000$ iterations for $\tilde{T} = 100$ total runs where each node initializes its local variable $\mathbf{y}_{i,0}$ uniformly at random from the unit interval, and plot the sample mean results for problem instances of (3.61) when the network size is varied as $V = 16$, $V = 64$, $V = 400$, which correspond respectively to 4×4, 8×8, and 20×20 grid sensor formations.

We plot the results of this numerical setup in Figure 3.2 for a randomly chosen sensor in the network. Observe that in Figure 3.2(a), which shows the convergence behavior in terms of the local objective $\mathbb{E}_{\mathbf{b}_i}\|\mathbf{A}_i\mathbf{y}_{i,t} - \mathbf{b}_i\|^2$ versus iteration t, that the rate at which sensors are able to localize the source is comparable across the different network sizes; however, the convergence accuracy is higher in smaller networks. In particular, by $t = 1000$, the objective converges to respective values 0.03, 0.08, and 0.14 for the $V = 16$, $V = 64$, $V = 400$ node networks. This relationship between convergence accuracy and number of sensors in the network is corroborated in the plot of the standard error $\|\mathbf{w}_{i,t} - \mathbf{w}^*\|$ to the true source location $\mathbf{w}^*$ in Figure 3.2(b). We see that the standard error across the different networks converges to within a radius of 1 to the optimum, but the rate at which convergence is exhibited is comparable across the different network sizes. In particular, by $t = 400$, we observe the standard error benchmarks 0.41, 0.74, and 0.9 for the $V = 16$, $V = 64$, and $V = 400$ node networks.

A similar pattern may be gleaned from Figure 3.2(c), in which we plot the magnitude of the constraint violation $\sum_{j \in n_i} g(\mathbf{y}_{i,t}, \mathbf{y}_{j,t})$ as given in (3.64) with iteration t. Observe that for the networks with $V = 16$, $V = 64$, and $V = 400$ sensors, respectively, we have the constraint violation benchmarks 2.1, 4, and 4.74 by $t = 300$. Moreover, the rate at which benchmarks are achieved is comparable across the different network sizes, such that the primary difference in the dual domain is the asymptotic magnitude of constraint violation, but not dual variable convergence rate.

3.5.3 Effect of Spatial Deployment

We turn to studying the impact of the way in which sensors are spatially deployed on their ability to localize the source signal, which implicitly is an analysis of the impact of the network topology on the empirical convergence behavior. To do so, we consider a problem instance in which the source signal $\mathbf{w}^*$ is located at the average of sensor positions in the network in a planar ($p = 2$) spatial region $\mathcal{A}$ of size 1000×1000 meters. The noise distribution which perturbs the range measurements received at node i is fixed as zero-mean Gaussian with variance $.5\|\mathbf{l}_i - \mathbf{w}^*\|$, implying that nodes which are closer to the source receive observations with higher SNR. Each node initializes its local variable $\mathbf{y}_{i,0}$ uniformly at random from the unit interval, and then executes the saddle point method

stated in (3.62) - (3.63) for a total of $T = 1000$ iterations for $\tilde{T} = 100$ total runs. We consider the sample mean results of the $\tilde{T} = 100$ for problem instances of (3.61) when the sensor deployment strategy is either a grid formation, uniformly at random, or according to a two-dimensional Gaussian distribution. In the later two cases, sensors which are closer than a distance of 50 meters are connected. Since in general random networks of these types will not be connected, we repeatedly generate such networks until we obtain the first one which has the a comparable Fiedler number (second-smallest eigenvalue of the graph Laplacian matrix) as the grid network, which is a standard measure of network connectivity – see [47], Ch. 1.

We display the results of this localization experiment in Figure 3.4. In Figure 3.4(a), we plot the local objective as compared with iteration t across these different sensor deployment strategies. We see that sensor localization performance is best in terms of objective convergence in the grid network, followed by network topologies generated from uniform and Normal spatial deployment strategies. In particular, by $t = 400$, the grid, Uniform, and Normal sensor networks achieve the objective ($\mathbb{E}_{\mathbf{b}_i}\|\mathbf{A}_i\mathbf{y}_{i,t} - \mathbf{b}_i\|^2$) benchmarks 0.26, 0.45, and 0.83. This trend is not corroborated by our analysis of these sensor networks' ability to learn the true source $\mathbf{w}^*$ as measured by the standard error $\|\mathbf{w}_{i,t} - \mathbf{w}^*\|$ (Figure 3.4(b)). In particular, to achieve the benchmark $\|\mathbf{w}_{i,t} - \mathbf{w}^*\| \leq 1$ we see that the Uniform topology requires $t = 26$ iterations, whereas the grid network requires $t = 557$ iterations, and the Normal network does not achieve the error bound by $t = 1000$. However, we observe that the grid network experiences more stable convergence behavior in terms of its error sequence, as compared with the other two networks.

In Figure 3.4(c), we display the constraint violation [cf. (3.64)] incurred by the proximity-constrained saddle point method when we vary the sensor deployment strategy. Observe that the grid and Uniform network topologies incur comparable levels of constraint violation, whereas the sensor network induced by choosing spatial locations according to a two-dimensional Gaussian distribution is able to maintain closer levels of network proximity by nearly an order of magnitude.

3.6 Perspective on Collaborative Adaptive GLM Learning

We considered multi-agent stochastic optimization problems where the hypothesis that all agents are trying to learn common parameters may be violated. In doing so, agents to make decisions which give preference to locally observed information while incorporating the relevant information of others. This problem class incorporates sequential estimation problems in multi-agent settings where observations are independent but *not identically* distributed. We formulated this task as a decentralized stochastic program with convex proximity constraints which incentivize distinct nodes to make decisions which are close to

one another. We considered an augmented Lagrangian relaxation of the problem, to which we apply a stochastic variant of the saddle point method of Arrow and Hurwicz to solve it. We established that under a constant step-size regime the time-average suboptimality and constraint violation are contained in a neighborhood whose radius vanishes with increasing number of iterations (Theorem 3). As a consequence, we obtain in Corollary 1 that the average primal vectors converge to the optimum while satisfying the network proximity constraints.

Numerical analysis on a random field estimation problem in a sensor network illustrated the benefits of using the saddle point method with proximity constraints as compared with a simple LMMSE estimator scheme. We find that these benefits are more pronounced in problem instances with lower SNR and larger spatial regions in which sensors are deployed, e.g., instances where the correlation structure of the information across the network plays a larger role. We further considered a source localization problem in a sensor network, where sensors collect noisy range estimates whose SNR is proportional to their distance to the true source signal. In this problem setting, the proximity-constrained saddle point method outperforms methods which attempt to execute consensus constraints.

The proximity constrained multi-agent optimization problem formulated in this chapter yielded statistical learning tools for generalized linear models that are able to address more general hypotheses regarding the relationship between each agents' data, and thus outperform consensus methods when the hypothesis that each agent's data is identically distributed is violated. However, the applications considered in this chapter are relatively simple ones in which linear statistical models are sufficiently descriptive so as to attain good regression performance. In general, this will not be true, especially for challenging robotic inference tasks we seek to address in the next part of this dissertation. Thus the next part of this thesis attempts to build upon the statistical optimization tools developed in Part I for the restrictive estimator class $\mathscr{F} = \mathbb{R}^p$ to develop methods for more general selections of $\mathscr{F}$.

Part II

Task-Driven Dictionary Learning

Chapter 4

Dictionary Learning

Chapter 4 extends the general learning problem with generalized linear models to one in which we seek to learn both a parameter vector and an encoding of the feature space which are tuned to the inference task we seek to solve. The reason for seeking feature encoding/extraction techniques is to reveal latent insight into the data which may improve inference performance. To learn such a representation, a variety of objectives may be considered. If the vector's dimension is very large, dimensionality reduction is of interest, which has classically been approached with principal component analysis [84] where basis elements are required to form a mutually orthogonal set. If instead specialized domain knowledge is available, finding representations based on particularized functions, i.e. wavelets for natural imagery [122], is more appropriate. Alternatively, one may seek to learn signal representations of a feature space directly from data, as in dictionary learning. Dictionary learning has been applied to unsupervised learning problems such as inpainting or denoising [58,119], and supervised tasks like classification [155, 213].

We propose tailoring the dictionary to a supervised learning task, as in [12]. This idea is referred to as *discriminative* dictionary learning, and has shown promise as compared to their unsupervised counterparts [34, 118, 156]. The problem of developing a dictionary representation of a signal specifically suited to a supervised learning task is a difficult optimization problem. In the offline setting, one may use block coordinate descent [196], or alternating gradient methods [216], either of which are only effective for small-scale batch settings. In the centralized online setting, prior approaches have made use of stochastic approximation [58,117]. Subsequently, we develop a mathematical definition of task-driven dictionary learning.

4.1 Data-Driven Signal Representations

Consider a set of N signals $\{\mathbf{x}_t\}_{t=1}^N$ each of which lies in an p-dimensional feature space as $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^p$. We aim to represent the signals $\mathbf{x}_t$ as combinations of a common set of k linear basis elements $\{\mathbf{d}_l\}_{l=1}^k$, which are unknown and must be learned from data. We group these k basis elements into a dictionary matrix $\tilde{\mathbf{D}} = [\mathbf{d}_1, \ldots, \mathbf{d}_k] \in \mathbb{R}^{p\times k}$ and denote the coding of $\mathbf{x}_t$ as $\boldsymbol{\alpha}_t \in \mathbb{R}^k$. One may think of the coding $\boldsymbol{\alpha}_t$ as the coefficients of $\mathbf{x}_t$ with respect to dictionary $\tilde{\mathbf{D}}$. For a given dictionary, the coding problem calls for finding a representation $\boldsymbol{\alpha}_t$ such that the signal $\mathbf{x}_t$ is close to its dictionary representation $\tilde{\mathbf{D}}\boldsymbol{\alpha}_t$. This goal can be mathematically formulated by introducing a loss function $s(\boldsymbol{\alpha}_t, \tilde{\mathbf{D}}; \mathbf{x}_t)$ that depends on the proximity between $\tilde{\mathbf{D}}\boldsymbol{\alpha}_t$ and the data point $\mathbf{x}_t$ and formulating the coding problem as [2]

$$\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t) := \underset{\boldsymbol{\alpha}_t \in \mathbb{R}^k}{\operatorname{argmin}}\, s(\boldsymbol{\alpha}_t, \tilde{\mathbf{D}}; \mathbf{x}_t)\ . \tag{4.1}$$

Hereafter, we assume that basis elements are normalized to have norms $\|\mathbf{d}_l\| \leq 1$ so that the dictionary is restricted to the convex compact set $\mathcal{D} := \{\tilde{\mathbf{D}} \in \mathbb{R}^{m\times k} : \|\mathbf{d}_l\| \leq 1, \text{ for all } l\}$.

The dictionary learning problem associated with the loss function $s(\boldsymbol{\alpha}_t, \tilde{\mathbf{D}}; \mathbf{x}_t)$ entails finding a dictionary $\tilde{\mathbf{D}}$ such that the signals $\mathbf{x}_t$ are close to their representations $\tilde{\mathbf{D}}\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t)$ for all possible t, which is approached by minimizing $s(\boldsymbol{\alpha}_t, \tilde{\mathbf{D}}; \mathbf{x}_t)$ with respect to $\tilde{\mathbf{D}}$ as well. Instead, however, we focus on discriminative problems where the goal is to find a dictionary that is well adapted to a specific classification or regression task [12]. Thus, we associate with each $\mathbf{x}_t$ a variable $\mathbf{y}_t \in \mathcal{Y}$ that represents a discrete label – in the case of classification problems – or a set of associated vectors $\mathcal{Y} \subset \mathbb{R}^q$ – in the case of regression. We then use the coding $\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x})$ in (4.10) as a feature representation of the signal $\mathbf{x}_t$ and introduce the classifier $\mathbf{w}$ that is used to predict the label or vector $\mathbf{y}_t$ when given the signal $\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x})$. The merit of the classifier $\mathbf{w} \in \mathcal{W} \subset \mathbb{R}^k$ is measured by the smooth loss function $\ell\big(\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t), \mathbf{w}; (\mathbf{x}_t, \mathbf{y}_t)\big)$ that captures how well the classifier $\mathbf{w}$ may predict $\mathbf{y}_t$ when given the sparse coding $\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t)$ that we compute using the dictionary $\tilde{\mathbf{D}}$. The discriminative dictionary learning problem is formulated as the joint determination of the dictionary $\tilde{\mathbf{D}} \in \mathcal{D}$ and classifier $\mathbf{w} \in \mathcal{W} \subset \mathbb{R}^k$ that minimize the cost $\ell\big(\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t), \mathbf{w}; (\mathbf{x}_t, \mathbf{y}_t)\big)$ averaged over the training set,

$$(\tilde{\mathbf{D}}^*, \mathbf{w}^*) := \underset{\tilde{\mathbf{D}}\in\mathcal{D}, \mathbf{w}\in\mathcal{W}}{\operatorname{argmin}} \frac{1}{N}\sum_{t=1}^{N} \ell\big(\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t), \mathbf{w}; (\mathbf{x}_t, \mathbf{y}_t)\big). \tag{4.2}$$

For given dictionary $\tilde{\mathbf{D}}$ and signal sample $\mathbf{x}_t$ we compute the code $\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t)$ as per (4.10), predict $\mathbf{y}_t$ using $\mathbf{w}$, and measure the prediction error with the loss function $\ell\big(\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t), \mathbf{w}; (\mathbf{x}_t, \mathbf{y}_t)\big)$. The optimal pair $(\tilde{\mathbf{D}}^*, \mathbf{w}^*)$ in (4.2) is the one that minimizes the

cost averaged over the given sample pairs $(\mathbf{x}_t, \mathbf{y}_t)$. Observe that $\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t)$ is not a variable in the optimization in (4.2) but a mapping for a implicit dependence of the loss on the dictionary $\tilde{\mathbf{D}}$. To simplify notation we henceforth write (4.2) as

$$(\tilde{\mathbf{D}}^*, \mathbf{w}^*) := \underset{\tilde{\mathbf{D}} \in \mathcal{D}, \mathbf{w} \in \mathcal{W}}{\operatorname{argmin}} \frac{1}{N} \sum_{t=1}^{N} \ell\big(\tilde{\mathbf{D}}, \mathbf{w}; (\mathbf{x}_t, \mathbf{y}_t)\big) \,. \tag{4.3}$$

The optimization problem in (4.3) is not assumed to be convex – this would be restrictive because the dependence of h on $\tilde{\mathbf{D}}$ is, partly, through the mapping $\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t)$ defined by (4.10). In general, only local minima of (4.3) can be found.

Our goal in this chapter is to study online algorithms that solve (4.3) as training pairs $(\mathbf{x}_t, \mathbf{y}_t)$ become available. To do so we assume that training pairs $(\mathbf{x}_t, \mathbf{y}_t)$ are independently sampled from a common probability distribution and replace (4.3) by

$$(\tilde{\mathbf{D}}^*, \mathbf{w}^*) := \underset{\tilde{\mathbf{D}} \in \mathcal{D}, \mathbf{w} \in \mathcal{W}}{\operatorname{argmin}} \mathbb{E}_{\mathbf{x},\mathbf{y}} \Big[\ell\big(\tilde{\mathbf{D}}, \mathbf{w}; (\mathbf{x}, \mathbf{y})\big) \Big]. \tag{4.4}$$

The problems in (4.12) and (4.3) are equivalent in the limit of $N \to \infty$ if $(\mathbf{x}_t, \mathbf{y}_t)$ are independently drawn from the joint distribution of the random pair $(\mathbf{x}, \mathbf{y})$. The problem in (4.12), as the one in (4.3), is not convex. We clarify the formulation in (4.12) with two examples.

Example 1 (Sparse unsupervised learning) When we have $k < p$, the formulation in (4.10) aims at finding a dictionary that reduces data dimensionality from p to k. In this chapter we are more interested in the overdetermined case in which $k > p$ but we want the codes $\boldsymbol{\alpha}_t$ to be sparse. These sparsity constraints can be written as upper limits on the zero norm of $\boldsymbol{\alpha}_t$ but that would yield computationally intractable formulations. To circumvent this issue, sparsity can be incentivized by adding, e.g., elastic net regularization terms [13, 79], in which case we can write the loss function $s(\boldsymbol{\alpha}_t, \tilde{\mathbf{D}}; \mathbf{x}_t)$ in (4.10) as

$$s(\boldsymbol{\alpha}_t, \tilde{\mathbf{D}}; \mathbf{x}_t) = \tilde{s}(\boldsymbol{\alpha}_t, \tilde{\mathbf{D}}; \mathbf{x}_t) + \zeta_1 \|\boldsymbol{\alpha}_t\|_1 + \frac{\zeta_2}{2} \|\boldsymbol{\alpha}_t\|_2^2 \,. \tag{4.5}$$

In (4.5), $\tilde{s}(\boldsymbol{\alpha}_t, \tilde{\mathbf{D}}; \mathbf{x}_t)$ measures proximity between $\mathbf{x}_t$ and $\tilde{\mathbf{D}}\boldsymbol{\alpha}_t$, the ℓ_1 term $\zeta_1 \|\boldsymbol{\alpha}_t\|_1$ encourages sparsity, and the ℓ_2 term $(\zeta_2/2)\|\boldsymbol{\alpha}_t\|_2^2$ is a smooth regularizer. Common choices for the proximity functions are the Euclidean distance $\tilde{s}(\boldsymbol{\alpha}_t, \tilde{\mathbf{D}}; \mathbf{x}_t) = \|\mathbf{x}_t - \tilde{\mathbf{D}}\boldsymbol{\alpha}_t\|^2/2$ and the l_∞ norm $\tilde{s}(\boldsymbol{\alpha}_t, \tilde{\mathbf{D}}; \mathbf{x}_t) = \|\mathbf{x}_t - \tilde{\mathbf{D}}\boldsymbol{\alpha}_t\|_\infty = \max_i |\mathbf{x}_{i,t} - \tilde{\mathbf{D}}_i \boldsymbol{\alpha}_{i,t}|$. In an unsupervised problem we simply want to make $\mathbf{x}_t$ and $\tilde{\mathbf{D}}\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t)$ close to each other across elements of the training set. We achieve that by simply making $\ell\big(\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t), \tilde{\mathbf{D}}, \mathbf{w}; (\mathbf{x}_t, \mathbf{y}_t)\big) = s\big(\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t), \tilde{\mathbf{D}}; (\mathbf{x}_t, \mathbf{y}_t)\big)$.

Example 2 (Sparse logistic regression) Given a training set of pairs $(\mathbf{x}_t, y_t)$ where

$\mathbf{x}_t \in \mathbb{R}^p$ is a feature vector with associated binary label $y_t \in \{-1, 1\}$, we seek a decision hyperplane $\mathbf{w} \in \mathbb{R}^k$ which best separates data points with distinct labels. However, instead of looking for linear separation in the original space, we seek for linear separation in a sparse coded space. Thus, let $\boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t)$ be the sparse coding of $\mathbf{x}_t$ computed through (4.10) when using the loss function in (4.5). We want to find a classifier $\mathbf{w}$ such that $\mathbf{w}^T \boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t) > 0$ when $y_t = 1$ and $\mathbf{w}^T \boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t) < 0$ when $y_t = -1$. This hyperplane need not exist but we can always model the probability of observing $y_t = 1$ through its odds ratio relative to $y_t = -1$. This yields the optimal classifier $\mathbf{w}^*$ as the one that minimizes the logistic loss

$$\ell(\tilde{\mathbf{D}}, \mathbf{w}; (\mathbf{x}_t, \mathbf{y}_t)) = -\log \left\{ \mathrm{P}\left(y_t = \pm 1 \,\middle|\, \boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t), \mathbf{w} \right) \right\} = -\log \left\{ \frac{1}{1 + \exp(-y_t \mathbf{w}^T \boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t))} \right\} . \tag{4.6}$$

For a feature vector $\mathbf{x}_t$, (4.6) models the probability of the label y_t being 1 or -1 as determined by the inner product $\mathbf{w}^T \boldsymbol{\alpha}^*(\tilde{\mathbf{D}}; \mathbf{x}_t)$. Substituting (4.6) into (4.12) yields the discriminative dictionary learning problem for logistic regression with sparse features.

4.2 Predicting Control Uncertainty in Ground Robots

We investigate the use of task-driven dictionaries for uncertainty quantification of control decisions based on sensory information received on-board by a ground robot. The motivation for this work is as follows: maneuvering along a planned path is the most basic control task in ground robotics, but a robot's ability to accomplish this task is governed by a plethora of physical phenomena, such as ground interactions and shear deformations, that are either difficult to model from first principles, or yield dynamical systems for which obtaining control inputs is intractable. Though driving slowly enough may hide the difference between the kinematic models that underlie most path planning schemes and the ground truth, this precludes any mission with an operational tempo or scale that necessitates operation at high speeds.

By abstracting away the complicated physical phenomena using statistical models of the difference between expected and true state behavior, which we call the *model disturbance*, we may obtain the basic predictive feed-forward power we need to drive a modern control system, such as a classically-inspired two-degree-of-freedom controller [10] or a sampling-based receding-horizon controller [76]. Here we consider the model disturbance to be a function of both model mismatch and unknown environmental effects. In the past, statistical models of ground robotic velocity disturbances have been built by batch-fitting statistics to an entire dataset of one operational environment [65, 162]. The descriptive capability of the statistical model which captures unexpected maneuvers experienced by the platform is

inherently contingent on the variation of the terrain in which the vehicle operates, as well as the sophistication of the parametric form of the chosen model.

In practice, we might not have access to a dataset for a particular operational environment, or the environment may have such dramatically different surfaces that a generic model is too imprecise. Moreover, such techniques would not allow real-time adaptation to environmental effects as they are experienced by the platform. In this case, an adaptive approach, where one sequentially revises estimates of the model disturbance based on new information, is advantageous. Classical approaches such as the Extended Kalman Filter have been successfully applied to handle this adaptive state estimation problem in ground robotics [162, 174]. However, the memory tuning of this estimator plays a large role in its predictive capability: with long memory, the resulting model will eventually end up trying to poorly fit the entire environment; with short memory, the resulting model will be unable to make use of data collected on earlier traversals of the same location or surface type.

One solution to this memory window issue is to develop multiple models in parallel, one for each distinct type of surface, and use a separate estimation procedure, such as visual classification, to decide on which surface the robot is driving and learn a model associated with only that particular surface [106]. In the adaptive setting [142], this classifier must also be learned online, but the number of classes needed to properly model the environment is seldom known in advance, and online multi-class classification is a computationally expensive procedure. With this motivation in mind, we tailor statistical learning techniques to address these shortcomings. In particular, in this chapter we

1. develop an approach to perform online regression over the disturbance statistical model, jointly with a representational basis, or dictionary, of the feature space which consists of control and perception information, by making use of supervised dictionary learning techniques [12, 93] (Sections 4.2 and 4.3);

2. quantify the advantages of using this learning technique which incorporates robotic perception as compared to approaches based on batch statistics or simple adaptive approaches on a ground robot which collects visual and odometric data while driving continuously over multiple surfaces (Section 4.4).

The resulting approach, by incorporating the robot's real-time sensing and perception capabilities into an adaptive disturbance predictor, effectively bypasses the need for an explicit surface classification step by parameterizing the statistical disturbance model over the visual features and control signals that are observed while the platform experiences the disturbances. Related approaches to predicting steering mistakes based on statistical learning have been considered such as, e.g., Gaussian process based models [147], or treating model mismatch and environmental effects separately [4, 5]. Our approach considers these effects

jointly, and makes use of discriminative matrix-factorization-based methods. Furthermore, we empirically demonstrate the proposed framework's ability to quantify uncertainty that comes from unmodelled system dynamics and unknown environmental effects along a given path in real-time on a ground robot.

4.2.1 Control Uncertainty Forecasting

We consider the problem of learning unmodelled system dynamics and exogenous environmental effects, which we call the model disturbance, on the platform's path planning scheme. To do so, we adopt an approach similar to [147] by considering the following discrete-time nonlinear state-space system of equations

$$\mathbf{x}_{k+1} = s(\mathbf{x}_k, \mathbf{u}_k) + g(\mathbf{a}_k) \ , \tag{4.7}$$

where $\mathbf{x}_k \in \mathcal{X}$ is the system state, which consists of pose and possibly velocity information, $\mathbf{u}_k$ is the control input at time index k, and the map $s : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is a simple kinematic model that is chosen a priori, such as an effective wheel-base [222] or general kinematic slip model [175]. We concatenate control inputs and signals $\mathbf{z}_k$ observed by the platform such as visual, acoustic, or LIDAR information into feature vectors $\mathbf{a}_k := (\mathbf{u}_k; \mathbf{z}_k) \in \mathcal{A} \subset \mathbb{R}^p$ upon which unexpected maneuvers depend. To be specific, we define the model disturbance $g : \mathcal{A} \to \mathcal{X}$ in (4.7) as a general nonlinear map that captures unmodelled system dynamics due to unmodelled system dynamics that may be both structural and random, which we assume is a function of only control and sensory information. A key point of departure between this formulation of model disturbance and that of prior works is its dependence on the environment, which is parameterized by the perceptive and sensing capabilities of the platform.

In this work, control decisions are provided externally to the system by a user as an open-loop system, and their impacts are then empirically measured. However, the predictive technique developed in the next section may be fed into a model predictive control framework to adjust its cost functional according to learned disturbances.

We consider the model disturbance as a stochastic process depending on feature vectors $\mathbf{a} \in \mathcal{A}$ which aggregate past control and sensory information. Consider $\mathbf{x}_{k-1}$ as the system state which follows a purely kinematic model that is chosen a priori, the *estimated* state $\hat{\mathbf{x}}_k$ obtained via on-board sensor measurements, and control inputs $\mathbf{u}_{k-1}$ and signals $\mathbf{z}_{k-1}$ which been observed at the previous time slot $k-1$. Then we may rearrange (4.7) to obtain an estimate for the model disturbance $\hat{g}(\mathbf{a}_{k-1})$, i.e.

$$\hat{g}(\mathbf{a}_{k-1}) = \hat{\mathbf{x}}_k - s(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \ . \tag{4.8}$$

Our goal is to characterize the unknown true disturbance mapping $g : \mathbf{a}_k \mapsto g(\mathbf{a}_k)$ based upon realizations of the pairs $(\mathbf{a}_k, \hat{g}(\mathbf{a}_k))$ of feature vectors and physical measurements which estimate the model disturbance. For simplicity, henceforth we will consider $\hat{g}(\mathbf{a})$ to be scalar-valued.

Observe that if a generic path planner's kinematic model perfectly captures the ground truth state, the expression in (4.8) will be null. Since this is not the case, the model disturbance is a quantifiable phenomenon, especially across varying operating terrains for the platform. Thus, we seek to characterize the map g by learning a Gaussian approximation of the exogenous environmental and dynamical effects. In particular, we consider the random pair $(\mathbf{a}, \hat{g}(\mathbf{a}))$ to be related through a conditional Gaussian distribution of the form $\hat{g}(\mathbf{a}) \mid \mathbf{a} \sim \mathcal{N}(\mu(\mathbf{a}), \sigma^2(\mathbf{a}))$ with unknown mean $\mu(\mathbf{a})$ and scalar variance $\sigma^2(\mathbf{a})$ that depend on the system state and observed sensory information $\mathbf{a}$, i.e.

$$\mathrm{P}\left[\hat{g}(\mathbf{a}) \mid \mathbf{a}\right] = \frac{1}{\sqrt{2\pi\sigma^2(\mathbf{a})}} \exp\left[-\frac{(\hat{g}(\mathbf{a}) - \mu(\mathbf{a}))^2}{2\sigma^2(\mathbf{a})}\right] . \tag{4.9}$$

Information about the map $g(\cdot)$ in the form of realizations the random pair $(\mathbf{a}, \hat{g}(\mathbf{a}))$ are sequentially revealed as the robot explores the feature space associated with its operating environment. In order for information about the disturbance to be leveraged for path planning, disturbance predictions must be made on an incremental basis, which motivates the formulation of learning the Gaussian approximation of the model disturbance as an online learning problem, whereby we seek to repeatedly revise predictions of the Gaussian disturbance approximation based on newly available information.

Since $g(\cdot)$ in the state space model given in (4.7) represents a complicated relationship between robotic sensory perception and unexpected effects of control decisions, we expect the relationship between $(\mathbf{a}, \hat{g}(\mathbf{a}))$ to be highly nonlinear, in which case the performance of a simple regressor on the likelihood given in (4.9) may be boosted by learning an alternative feature encoding of signals $\mathbf{a}$. Motivated by this observation, we seek to represent realizations $\mathbf{a}_k$ as a combination of m common basis elements (or atoms) $\mathbf{d}_l$ which are unknown and must be learned from data. We stack $\mathbf{d}_k$ into a matrix which we call the dictionary matrix $\mathbf{D} \in \mathbb{R}^{|\mathcal{A}| \times m}$ and denote the coding of $\mathbf{a}_k$ as $\boldsymbol{\alpha}_k \in \mathbb{R}^m$. For a given dictionary, the coding problem calls for finding a representation $\boldsymbol{\alpha}_k$ such that the signal $\mathbf{a}_k$ is close to its dictionary representation $\mathbf{D}\boldsymbol{\alpha}_k$, which may be mathematically formulated by introducing a loss function that depends on the proximity between $\mathbf{D}\boldsymbol{\alpha}_k$ and the data point $\mathbf{a}_k$, specifically, we consider an elastic-net minimization [2]

$$\boldsymbol{\alpha}^*(\mathbf{D}; \mathbf{a}_k) := \underset{\boldsymbol{\alpha}_k \in \mathbb{R}^m}{\operatorname{argmin}} (1/2)\|\mathbf{a}_k - \mathbf{D}\boldsymbol{\alpha}_k\|_2^2 + \lambda\|\boldsymbol{\alpha}_k\|_1 + \nu\|\boldsymbol{\alpha}_k\|_2, \tag{4.10}$$

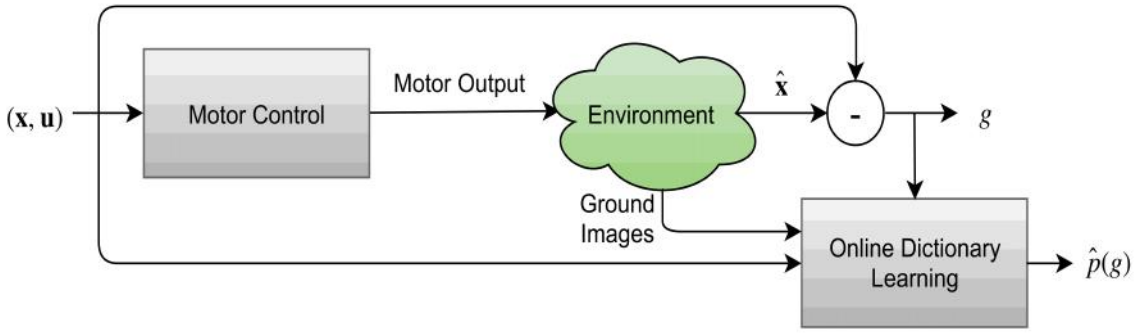


Figure 4.1: Overview of our system. The platform's state $\mathbf{x}$ and control $\mathbf{u}$ intended by a kinematic planner differ from the ground truth $\hat{\mathbf{x}}$ measured by an inertial measurement unit by the model disturbance g due to factors such as modelling errors in the motor control and environmental effects. Our goal is to develop a learning procedure to sequentially estimate the probability distribution of g based upon of just $\mathbf{u}$ and images $\mathbf{z}$ of the terrain.

which may be efficiently solved [57]. Hereafter, we assume that basis elements are normalized to have norms $\|\mathbf{d}_l\| \leq 1$ so that the dictionary is restricted to the convex compact set $\mathcal{D} := \{\mathbf{D} \in \mathbb{R}^{|\mathcal{A}|\times m} : \|\mathbf{d}_l\| \leq 1, \text{ for all } l\}$.

The dictionary learning problem associated with the elastic net entails finding a dictionary $\mathbf{D}$ such that the signals $\mathbf{a}_k$ are close to their representations $\mathbf{D}\boldsymbol{\alpha}^*(\mathbf{D};\mathbf{a}_k)$ for all possible k. Here, however, we focus on discriminative problems where the goal is to find a dictionary that is well adapted to a specific classification or regression task, as in [12]. To do so, we use the coding $\boldsymbol{\alpha}^*(\mathbf{D};\mathbf{a})$ in (4.10) as a feature representation of the signal $\mathbf{a}$ and introduce regressors $\mathbf{w}_1$ and $\mathbf{w}_2$ that are used to predict the first and second-order statistics $\mu(\mathbf{a})$ and $\sigma^2(\mathbf{a})$ when given the signal $\boldsymbol{\alpha}^*(\mathbf{D};\mathbf{a})$ through general maps of the form

$$\hat{\mu}(\mathbf{a}) = \ell(\mathbf{w}_1, \boldsymbol{\alpha}^*(\mathbf{D};\mathbf{a})), \quad \hat{\sigma}^2(\mathbf{a}) = l(\mathbf{w}_2, \boldsymbol{\alpha}^*(\mathbf{D};\mathbf{a})), \tag{4.11}$$

where $\hat{\mu}(\mathbf{a})$ and $\hat{\sigma}^2(\mathbf{a})$ denote estimators for the true moments of the model disturbance, and h and l map regressors and sparse codes to their estimators. Specific forms for (4.11) are given in Section 4.3.2, (4.17) - (4.18). In the next section, we develop our feed-forward scheme to predict the model disturbance.

4.3 Online Task-Driven Dictionary Learning

4.3.1 Formal Development

We formulate the task of learning regressors and signal representations tuned for predicting a Gaussian approximation of the model disturbance as an expected risk minimization problem associated with the density function in (4.9). To do so, observe that by substituting estimators $\hat{\mu}(\mathbf{a})$ and $\hat{\sigma}^2(\mathbf{a})$ as defined in (4.11) into the density (4.9), the likelihood for a

given data sample and model disturbance $(\mathbf{a}, \hat{g}(\mathbf{a}))$ becomes a function of the regressors $\mathbf{w}_1$ and $\mathbf{w}_2$, as well as the choice of feature representation via the dictionary matrix $\mathbf{D}$, that is $\mathrm{P}\left[\hat{g}(\mathbf{a}_k) \mid \mathbf{a}_k\right] = \mathrm{P}\left[\hat{g}(\mathbf{a}_k) \mid \mathbf{a}_k, \mathbf{D}, \mathbf{w}_1, \mathbf{w}_2\right]$. The merit of the regressors $\mathbf{w}_1$ and $\mathbf{w}_2$ and dictionary are then measured in terms of the maximum likelihood estimation (MLE) problem associated with predicting the Gaussian approximation of the model disturbance as in (4.9), i.e.

$$(\mathbf{D}^*, \mathbf{w}_1^*, \mathbf{w}_2^*) := \underset{\mathbf{D}\in\mathcal{D}, \mathbf{w}_1, \mathbf{w}_2}{\operatorname{argmin}} \mathbb{E}_{\mathbf{a}, \hat{g}(\mathbf{a})}\Big(- \log \mathrm{P}\left[\hat{g}(\mathbf{a}) \mid \mathbf{a}, \mathbf{D}, \mathbf{w}_1, \mathbf{w}_2\right] \Big). \tag{4.12}$$

Subsequently we define loss function ℓ as the negative log-likelihood of the expression (4.9) when the estimators $\hat{\mu}(\mathbf{a})$ and $\hat{\sigma}^2$ are substituted, that is $\ell(\mathbf{w}_1, \mathbf{w}_2, \mathbf{D}; (\mathbf{a}, \hat{g}(\mathbf{a}))) := - \log \mathrm{P}\left[\hat{g}(\mathbf{a}) \mid \mathbf{a}, \mathbf{D}, \mathbf{w}_1, \mathbf{w}_2\right]$. We may develop algorithms which are capable of sequentially adapting to new information when training pairs $(\mathbf{a}_k, \hat{g}(\mathbf{a}_k))$ are independently sampled from a common probability distribution as in (4.9). Observe that $\boldsymbol{\alpha}^*(\mathbf{D}; \mathbf{a})$ is not a variable in the optimization in (4.12) but a mapping for a implicit dependence of the loss on the dictionary $\mathbf{D}$. The optimization problem in (4.12) is not to convex – this would be too restrictive because the dependence of ℓ on $\mathbf{D}$ is, partly, through the mapping $\boldsymbol{\alpha}^*(\mathbf{D}; \mathbf{a}_k)$ defined by (4.10). In general, only local minima of (4.12) can be found. The relationship between the nonlinear state space control problem in (4.7) and the dictionary learning problem associated with the Gaussian approximation of the model disturbance (4.12) on the robotic platform is summarized in Figure 4.1.

We derive an algorithmic solution to the problem stated in (4.12) such that we may predict in real-time the parameters which define the Gaussian approximation of the unknown model disturbance. To do so, we make use of stochastic gradient descent [161] which has been shown to asymptotically converge in expectation for the task-driven dictionary problem [cf. (4.12)] in [94,95]. Specialized to this problem setting, the update rule for the regression vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ for predicting the first and second-order statistics in (4.9) are given as

$$\begin{aligned} \mathbf{w}_{1,k+1} &= \mathbf{w}_{1,k} + \eta_k \left(\nabla_{\mathbf{w}_1} \log \mathrm{P}\left[\hat{g}(\mathbf{a}_k) \mid \mathbf{a}_k, \mathbf{D}_k, \mathbf{w}_{1,k}, \mathbf{w}_{2,k}\right]\right), \\ \mathbf{w}_{2,k+1} &= \mathbf{w}_{2,k} + \eta_k \left(\nabla_{\mathbf{w}_2} \log \mathrm{P}\left[\hat{g}(\mathbf{a}_k) \mid \mathbf{a}_k, \mathbf{D}_k, \mathbf{w}_{1,k}, \mathbf{w}_{2,k}\right]\right), \end{aligned} \tag{4.13}$$

which amounts to an online MLE gradient step based on the last sample $(\mathbf{a}_k, \hat{g}(\mathbf{a}_k))$. Note that the negative associated with a stochastic descent step with respect to (4.12) cancels with negative in front of the log-likelihood, yielding (4.13). Here η_k is an algorithm step-size which may either be a small constant or chosen as $O(1/k)$, where k indexes the iteration number induced by the discretization of the state space dynamics in (4.7).

We turn to developing the stochastic gradient update with respect to the dictionary $\mathbf{D}$. Following Appendix of [12], define $Z \subset \{1, \ldots, m\}$ as the set of nonzero entries of

Algorithm 3 LOGD: Learning Online Gaussian Disturbance

Require: $\mathbf{D}_0, \mathbf{w}_{1,0}, \mathbf{w}_{2,0}$, initial dictionary and regressors, control policy $\{\mathbf{u}_u\}_{u=1,\dots}$, regularizers $\lambda, \nu \in \mathbb{R}$, step-size η_k.

1: **for** $k = 1, 2, \dots$ **do**

2: Compute disturbance $\hat{g}(\mathbf{a}_k)$ via estimated state $\hat{\mathbf{x}}_k$

3: Observe signals $\mathbf{z}_k$, use past control $\mathbf{u}_k$ to compute coding:

$$\boldsymbol{\alpha}_k^* := \underset{\boldsymbol{\alpha}_k \in \mathbb{R}^s}{\operatorname{argmin}} (1/2)\|\mathbf{a}_k - \mathbf{D}\boldsymbol{\alpha}_k\|_2^2 + \lambda\|\boldsymbol{\alpha}_k\|_1 + \nu\|\boldsymbol{\alpha}_k\|_2, \text{ [cf. (4.10)]}$$

4: Update dictionary [cf. (4.15)] and regressors [cf. (4.13)]

$$\begin{aligned}
\mathbf{D}_{k+1} &= \mathbf{D}_k - \eta_k \left[-\mathbf{D}\boldsymbol{\beta}\boldsymbol{\alpha}_k^* + (\mathbf{a}_k - \mathbf{D}_t\boldsymbol{\alpha}_k^*)\boldsymbol{\beta}_k{}^T \right] , \\
\mathbf{w}_{1,k+1} &= \mathbf{w}_{1,k} + \eta_k \left(\nabla_{\mathbf{w}_1} \log \mathrm{P}\left[\hat{g}(\mathbf{a}_k) \mid \mathbf{a}_k, \mathbf{D}_k, \mathbf{w}_{1,k}, \mathbf{w}_{2,k} \right] \right) , \\
\mathbf{w}_{2,k+1} &= \mathbf{w}_{2,k} + \eta_k \left(\nabla_{\mathbf{w}_2} \log \mathrm{P}\left[\hat{g}(\mathbf{a}_k) \mid \mathbf{a}_k, \mathbf{D}_k, \mathbf{w}_{1,k}, \mathbf{w}_{2,k} \right] \right) ,
\end{aligned}$$

5: **end for**

$\boldsymbol{\alpha}^* = \boldsymbol{\alpha}^*(\tilde{\mathbf{a}}, \mathbf{D})$ [cf. (4.10)] for a given $\tilde{\mathbf{a}} \in \mathcal{A}$ and let $\mathbf{D}_Z$ denote the truncated submatrix $\mathbf{D}$ consisting of its nonzero columns. Then the nonzero components of $\boldsymbol{\alpha}^*$ are given by $\boldsymbol{\alpha}_Z^* = (\mathbf{D}_Z^T\mathbf{D}_Z + \nu I)^{-1}(\mathbf{D}_Z\tilde{\mathbf{a}} - \lambda \operatorname{sgn}(\boldsymbol{\alpha}^*))$, which allows us to compute the stochastic gradient of (4.12) respect to the dictionary given the data pair $(\tilde{\mathbf{a}}, \hat{g}(\tilde{\mathbf{a}}))$ by applying Proposition 1 of [12] which yields the expression

$$\nabla_{\mathbf{D}} \ell(\mathbf{w}_1, \mathbf{w}_2, \mathbf{D}; (\tilde{\mathbf{a}}, \hat{g}(\tilde{\mathbf{a}}))) = -\mathbf{D}\boldsymbol{\beta}\boldsymbol{\alpha}^* + (\tilde{\mathbf{a}} - \mathbf{D}_k\boldsymbol{\alpha}^*)\boldsymbol{\beta}^T. \tag{4.14}$$

where $\boldsymbol{\beta} \in \mathbb{R}^m$ is defined as $\boldsymbol{\beta}_Z = ([\mathbf{D}]_Z^T[\mathbf{D}]_Z + \nu I)^{-1}\nabla_{\boldsymbol{\alpha}_Z} \ell(\mathbf{w}_1, \mathbf{w}_2, \mathbf{D}; (\tilde{\mathbf{a}}, \hat{g}(\tilde{\mathbf{a}})))$ and $\boldsymbol{\beta}_{Z^c} = 0$, as in [12], Proposition 1. The dictionary update is then given by descending along then negative of the stochastic gradient in (4.14) evaluated at the training pair $(\mathbf{a}_k, \hat{g}(\mathbf{a}_k))$ at time k, i.e.

$$\mathbf{D}_{k+1} = \mathbf{D}_k - \eta_k \left[-\mathbf{D}\boldsymbol{\beta}\boldsymbol{\alpha}_k^* + (\mathbf{a}_k - \mathbf{D}_t\boldsymbol{\alpha}_k^*)\boldsymbol{\beta}_k{}^T \right] , \tag{4.15}$$

where η_k is the algorithm step-size.

This dictionary update in (4.15) revises the learned signal representation based on realizations of the random pair $(\mathbf{a}, \hat{g}(\mathbf{a}))$ to be good with respect to predicting the Gaussian approximation of the model disturbance, as stated in (4.9). The proposed method for learning online the regressors on the first and second-order statistics of the model disturbance, as well as the dictionary supervised to this prediction task is summarized in Algorithm 3.

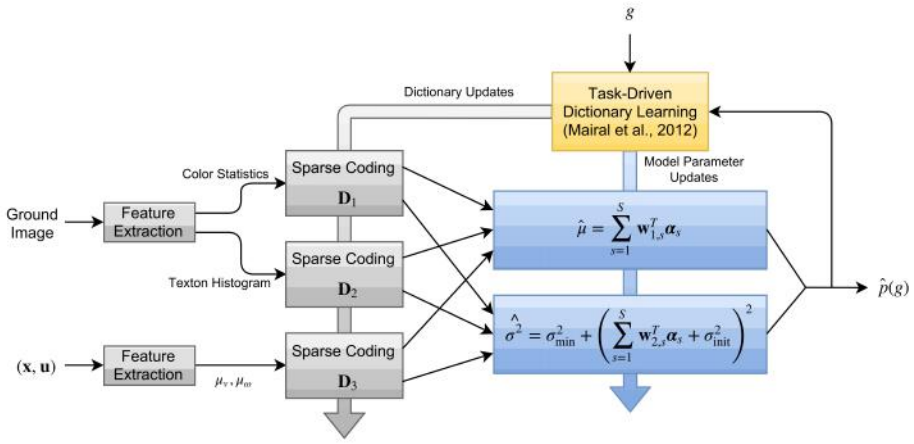


Figure 4.2: Graphical depiction of the online learning technique used in this chapter. First, we compute and scale the linear and angular velocity means and also color and texture features from the ground image. Using the available dictionaries, we generate sparse codes for each of these features and then, using our model parameters, we compute the estimated mean and variance of the assumed distribution. If we are given the actual disturbance observations, g, we use task-driven dictionary learning [116] to compute updates to the dictionaries, $\mathbf{D}_s$, and model parameters, $\mathbf{W}_s$.

4.3.2 Implementation Details

We now detail the specific implementation of Algorithm 3 used on the robot. In particular, we describe the features used, and we also address several practical issues that must be considered in order to achieve a working implementation, including accounting for multiple data modalities and a strategy for selecting an appropriate learning rate. For the purpose of characterizing the model disturbance, we define the quantities of interest as follows. We divide the ground plane on which the robot sits into several spatial *patches*, which we index here by k. Given a planned trajectory, we associate with each involved patch three pieces of information. The first is a descriptor of the time series that describes the control signal that is to be applied during the physical traversal of the patch. In our case, we extract the two scalar quantities: the mean of the linear velocity, μ_v, and the mean of the angular velocity, μ_ω, which we denote jointly as $\boldsymbol{\mu}_k^v$. The second and third pieces of information we associate with a given patch are visual features that we compute using a camera-collected image of that patch. The first such feature is given by computing the mean, variance, skewness, and kurtosis of each of the RGB color channels and also the same values for each of the channels in the CIELab color space. We jointly denote these statistics as $\mathbf{c}_k$. The second feature computed from the image of the ground is a texture descriptor. We use a texton histogram [107] computed using a 512-element codebook derived from the Brodatz texture dataset [37]. We denote this histogram as $\mathbf{h}_k$.

We jointly denote the three pieces of information described above as $\mathbf{a}_k = (\boldsymbol{\mu}_k^v, \mathbf{c}_k, \mathbf{h}_k)$, which serves as our descriptor of the trajectory for a particular segment of time. For each

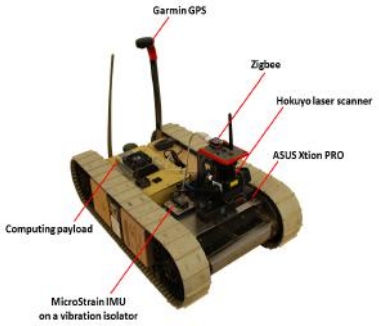
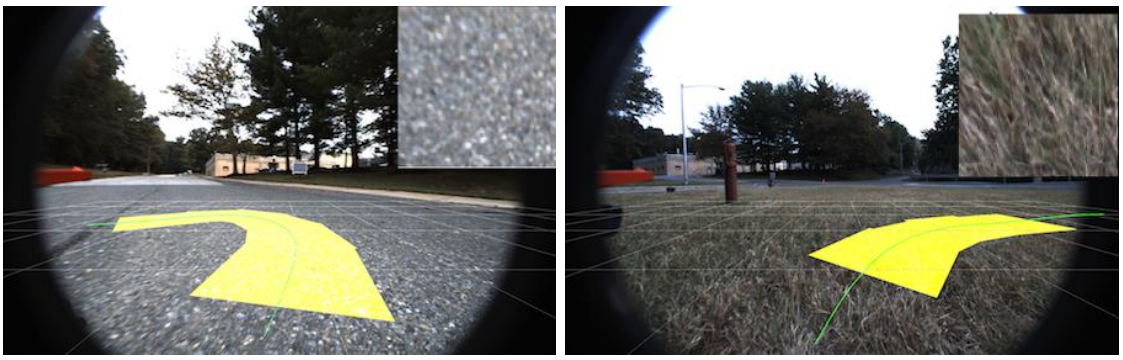


Figure 4.3: An iRobot *Packbot* was used in our experiments. It was additionally configured with a high-resolution camera.



(a) (b)

Figure 4.4: Sample images from the ((a)) *pavement* and ((b)) *grass* data sets. Note that each image is annotated with the path that was driven (green line) to collect disturbance measurements and an example of an extracted ground footprint used for learning (yellow highlight and inset upper right).

such segment, we assume the experienced disturbances are independent and identically distributed Gaussian random variables, where the parameters of the Gaussian distribution depend only on $\mathbf{a}_k$ and our model parameters $(\mathbf{D}, \mathbf{W})$. Between the time window $[k, k+1]$, the robot traverses a particular ground patch associated with visual features $\mathbf{c}_k, \mathbf{h}_k$ dictated by the control selection $\mathbf{u}_k = (\boldsymbol{\nu}_k, \boldsymbol{\omega}_k)$; however, there may be multiple disturbance measurements $\hat{g}(\mathbf{a}_k)$ for each time window k. Denote T_k as the number of disturbance measurements T_k in interval $[k, k+1]$, which depends on the traveling speed of the robot over this ground patch. This issue is handled by considering a Gaussian process model for disturbance measurements $\hat{g}(\mathbf{a}_k)_t$ for $t = 1, \ldots, T_k$ in the time window $[k, k+1]$, and assuming that $\hat{g}(\mathbf{a}_k)_u$ and $\hat{g}(\mathbf{a}_k)_t$ are independent and identically distributed for $u \neq t$ when conditioned on a fixed $\mathbf{a}_k$. Then the stochastic approximation of the expected loss in (4.12) is given by the negative log joint-likelihood over the disturbance data in time slot $[k, k+1]$, and yields the instantaneous loss

$$\tilde{\ell}(\mathbf{w}_1, \mathbf{w}_2, \mathbf{D}; (\mathbf{a}_k, \hat{g}(\mathbf{a}_k))) = \sum_{t=1}^{T_k} -\log \mathrm{P}\left[\hat{g}(\mathbf{a}_k)_t \mid \mathbf{a}_k, \mathbf{D}, \mathbf{w}_1, \mathbf{w}_2\right] \tag{4.16}$$

As described above, the feature vectors $\mathbf{a}_k$ arise from several different modalities. On our platform, they consist of velocity command information and both color and texture features computed using an image of the terrain, all of which have different meanings. Because of this, we impose a dictionary model on each modality *separately*, an approach inspired by [16], and then use the resulting *set* of sparse codes to jointly compute our disturbance distribution estimate. See Figure 4.2 for a depiction of this framework.

To begin, we first ensure that the feature vectors arising from each modality have a similar order of magnitude by applying a scaling factor $1/r_s$, where r_s is, e.g., the average vector norm of features from the s^{th} modality. Next, we compute the set of sparse codes, $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_S\}$, using a set of dictionaries $\mathbf{D} = \{\mathbf{D}_1, \ldots, \mathbf{D}_S\}$, each of which has $m = 90$ atoms. The total number of modalities for our setting is $S = 3$. We then use these codes and a corresponding set of model parameters, $\mathbf{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_S\}$, to compute the estimators of disturbance mean and variance (a special case of the expressions in (4.11)) according to

$$\hat{\mu}(\mathbf{a}) = \sum_{s=1}^{S} \mathbf{w}_{1,s}^T \boldsymbol{\alpha}_s \tag{4.17}$$

$$\hat{\sigma}^2(\mathbf{a}) = \sigma_{\min}^2 + \left(\sum_{s=1}^{S} \mathbf{w}_{2,s}^T \boldsymbol{\alpha}_s + \sigma_{\text{init}}^2 \right)^2 \tag{4.18}$$

where $\mathbf{w}_{1,s}^T$ and $\mathbf{w}_{2,s}^T$ represent the first and second rows of $\mathbf{W}_s$, respectively. Note that the

form os (4.18) ensures a positive variance prediction. For the data generated by platform, we select $\sigma^2_{\text{min}} = 0.01$ and $\sigma^2_{\text{init}} = 0.1$.

The loss function we use, which results from substituting (4.17) and (4.18) into the joint negative-log-likelihood function in (4.16) with a variable sized mini-batch of disturbance observations, is extremely nonlinear. Further, because it is difficult to determine its Lipschitz constant, the use of a backtracking line search [8] to find an appropriate step size is essential to successfully implementing Algorithm 3.

4.4 Experiments on Robotic Platform

In this section we focus on quantifying the advantages of our learning technique in a real-world experimental setting. To do so, we collect data on a the iRobot *Packbot* [218] as depicted in Fig. 4.3. The *Packbot* is a ground platform equipped with a skid-steer tracked drive system with on-board computation. The base platform weighs 18 kg and is capable of 2 m/s speeds. The robot used in our experiments is additionally equipped with a quad-core Intel i7 computing payload, a Microstrain 3DM-GX3 inertial measurement unit (IMU), and an Allied Vision Manta G-235 1/1.2" Color CMOS Camera with a 4.5 mm lens. Since the *packbot* is a skid-steer platform, we model it with state $\mathbf{x}_k = (x_x, y_x, \theta_x)$, linear and angular velocity controls given by $\mathbf{u}_k = (\boldsymbol{\nu}_k, \boldsymbol{\omega}_k)$, and an ideal motion-model given by

$$s(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} \dot{x_k} \\ \dot{y_k} \\ \dot{\theta_k} \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta_k) & -\sin(\theta_k) & 0 \\ \sin(\theta_k) & \cos(\theta_k) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{A(\theta)} \begin{bmatrix} \boldsymbol{\nu}_k \\ \boldsymbol{\omega}_k \end{bmatrix} \tag{4.19}$$

Track or wheel slip is essential to the operation of a skid-steer platform, and kinematic models of the disturbance $g(\mathbf{a}_k)$ are traditionally a function of control-state, e.g., $\boldsymbol{\nu}_k$, $\boldsymbol{\omega}_k$, $\boldsymbol{\nu}_k\boldsymbol{\omega}_k$ [65]. Learning a parameterization of the full disturbance function $g(\mathbf{a}_k)$ requires measurements of $(\dot{x_k}, \dot{y_k}, \dot{\theta_k})$ which are often not available directly. While an integrated perturbative approach can infer this disturbance function from coarse position sensors such as GPS [175], it requires a long sample path to do so. On the other hand, a consumer grade IMU is able to make accurate, direct measurements of $\dot{\theta}_k$. In our previous work [65], we have demonstrated that angular rate disturbance accounts for most of the error. Thus, we seek to use the approach in Section 4.3 to model the angular rate disturbance as measured by the difference between commanded angular velocity and the one observed by the IMU.

In our experiments, images of the terrain are collected at 5 Hz and a resolution of 1936-by-1216 pixels. Operator-generated sequences of control input $\mathbf{u}_k$ are used to sufficiently

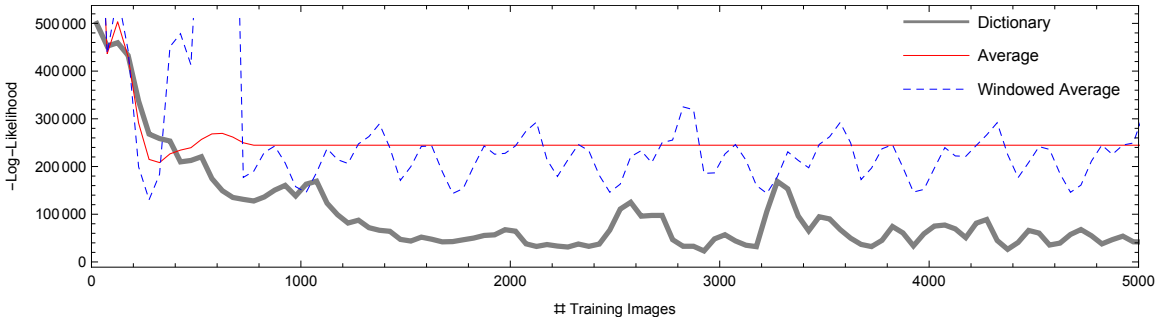


Figure 4.5: To validate the performance of our online learning algorithm, we periodically compute the loss function for a test set of images and disturbance data. Here we depict the likelihood-based loss function for our learned model compared to the performance of lower-order models. Observe that the dictionary-based method which incorporates the robot's perception performs best.

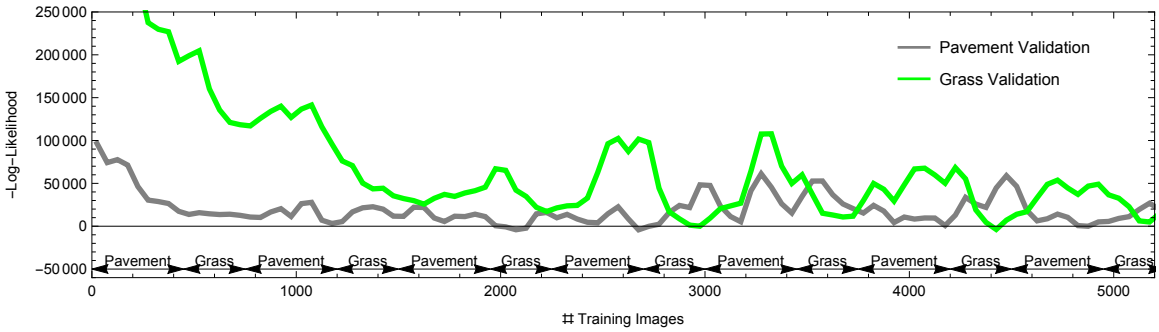
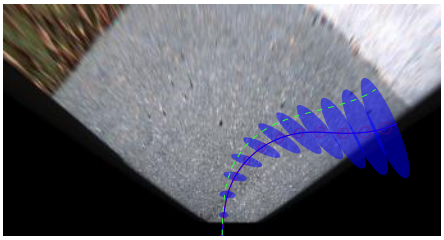


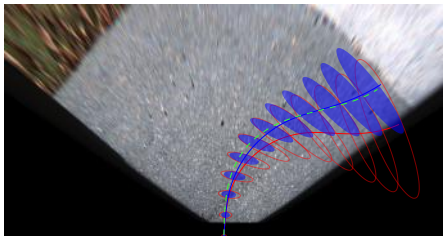
Figure 4.6: We examine the evolution of terrain-specific test set performance as training data encounters the two terrain types. Note that early in the learning process, for example, *pavement* training data corresponds with a decrease in performance of the model when predicting disturbance on *grass*. However, after sufficient training data has been encountered, the learning algorithm has adapted and is able to do a good job of predicting disturbances on either terrain type.

sample the space of maneuvers available to the *Packbot*. The path driven by the robot is sampled in the time following each image such that approximately six ground patches with disturbance measurements are completely visible in each image [1]. Hypothesizing about the nature of the disturbance's dependence on terrain type, we collected two independent data sets of the robot operating on *pavement* and *grass*. The *pavement* and *grass* data sets have approximately 550 and 330 images respectively. By providing our machine learning algorithm with epochs of time-series data from each of these data sets in turn, we are able to simulate the online learning performance of a system that sequentially encounters a variety of terrain. Sample visual observations of these terrains are depicted in Fig. 4.4.

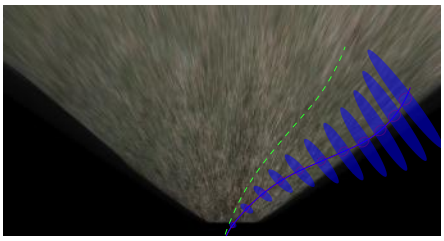
[1]Even with a wide angle camera lens, the most aggressive maneuvers result in paths that quickly leave the field of view; these footprints are excluded from our data.



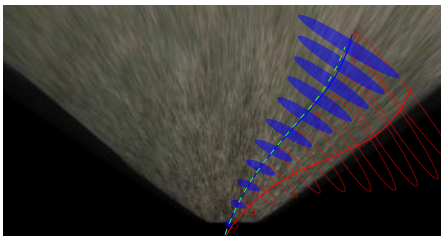
(a) Initial prediction



(b) Final prediction



(c) Initial prediction



(d) Final prediction

Figure 4.7: State uncertainty propagated according to model prediction and control-input time series for an example drawn from the terrain and grass test sets before training (Figs. 4.7(c) and 4.7(a)) and after training (Figs. 4.7(b) and 4.7(d)). The green dashed line is the actual driven path, the blue filled ellipses show the prediction based on our dictionary learning algorithm, and the red path/ellipses depict the *average* model Observe that the prediction generated by our method almost exactly matches the *actual* disturbance experienced by the platform, meaning that we successfully may predict where steering mistakes will be likely along the future reference trajectory.

4.4.1 Empirical Stability

We begin our analysis of results by comparing the performance of our online machine learning algorithm with two low-order models. Each of these models is based on windowed sample-estimates of the disturbance statistics, i.e., mean and variance estimates for each ground footprint computed using 300 samples over 1 s of time. These estimates of disturbance statistics are then used to generate an *average* model which returns the average sample statistics across the entire training set and an *windowed average* model which does the same operation but only over training images in the last 10 s. These models have been selected for comparison to highlight two competing goals when learning disturbance statistics: (1) to learn a model that is valid across the entire operational domain, e.g., the *average* model, and (2) to learn a model that tracks local disturbance statistics, e.g., a filtering approach [175] or the *windowed average* model. As stated above, our online dictionary learning algorithm seeks to dynamically balance these goals.

In order to fairly test and validate the performance of learning across these models, we set aside a uniformly-sampled test set of 51 images from the *pavement* domain and 35 images from the *grass* domain. We provide a continuous stream of training data by providing each learning algorithm with the remaining 800 images from our datasets in repetition. Learning updates are computed by averaging gradients across five images at a time (i.e., a mini-

batch). Performance on the test set is computed by querying each model for the predicted disturbance statistics in each image based on the logged control signals and visual features. The predicted disturbance statistics are used along with the actual measured disturbance data to compute the joint negative-log-likelihood that arises from (4.16).

The negative-log-likelihood is depicted in Fig. 4.5 as a function of the number of training images. There are two points to be taken from this figure. First, note that the *windowed average* approach performs as expected, oscillating as training images encounter different subsets of each terrain. Second, note that our dictionary learning algorithm is able to converge to a solution that outperforms either of these lower-order models in log-likelihood.

If we split our test set into subsets consisting only of *pavement* and *grass* terrain data, we can examine the performance of our algorithm as it encounters training epochs of each terrain type in turn. Figure 4.6 depicts the negative-log-likelihood function for each test sub-set. At several particular instances, the performance on one test sub-set decreases when the algorithm is provided training examples of the opposing terrain, e.g., the *grass* test set when *pavement* training images are presented at $k = 1500$ training images. While at first glance this result may seem to indicate a negative characteristic of our algorithm, we point out that over the long-term this interplay between terrain types is attenuated - the dictionary model is adapting to our problem!

4.4.2 Predictive Performance

We now analyze the predictive performance of our model. That is, how well is it able to predict the disturbance statistics of the test set? Given the control signal and visual features at each point in the test set, we are able to compute a predicted disturbance mean and variance. In Fig. 4.8, we depict the mean and 2σ envelope for our algorithm's prediction overlaid on the raw disturbance signal from our test set. Note that our algorithm is able to faithfully predict the statistics of the disturbance signal over a test set that has not been used for training. While this picture does hint at the advantages of pursing statistical estimation of model disturbance, e.g., the 2σ bound does a good job of capturing the rapid variation present in the raw disturbance data, a more compelling result is played out when one examines the full-state uncertainty propagation.

If we integrate a time-series of control $\mathbf{u}_k$ according to (4.7) with $s(\mathbf{x}_k, \mathbf{u}_k)$ given by (4.19) and $\hat{g}(\mathbf{a}_{k-1})$ given by our online dictionary model, we can predict not only the expected vehicle trajectory but also it's full-state covariance. Figure 4.7 depicts example predicted trajectories from our test set. To illustrate the systematic errors that a control algorithm is subject to when no consideration of model disturbance is made, we show the comparison between (1) our initial prediction before any training data has been ingested and (2) the final prediction after our trained model has converged.

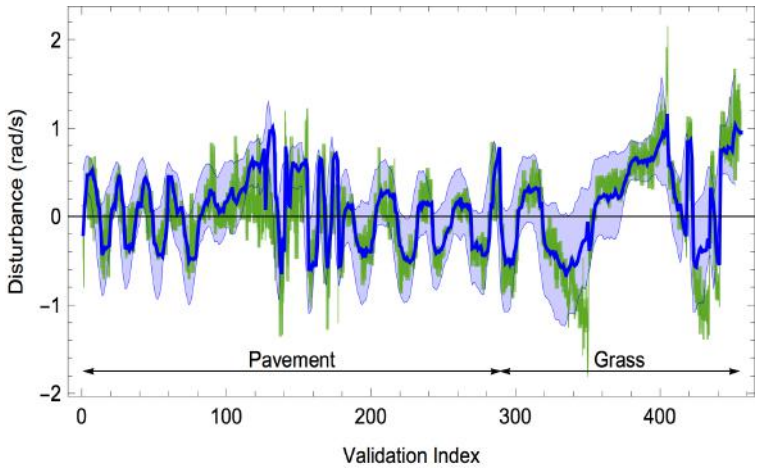


Figure 4.8: The statistics of the disturbance prediction across the test set is visualized in blue as a solid line for the mean predicted disturbance with a shaded envelope depicting $\pm 2\sigma$, and the true disturbance is given by the green line.

In particular, the green dashed line denotes the actual driven path, the blue filled ellipses show the prediction based on our dictionary learning algorithm, and the red path/ellipses depict the *average* model. Examining the final prediction in these figures, it is clear that our learned model using visual perception is successfully predicting significant terrain-dependent disturbance in both the grass and pavement settings. We do note that while the *average* model does a poor job of predicting the mean disturbance statistics, it does adopt a variance that admits the actually driven path.

In this chapter we used an online dictionary learning technique to generate predictions of the model disturbance distribution given real-world experiential data. We are able to generate these predictions from motion control signals and low-level visual features, thus bypassing the need for explicit terrain classification. Further, these predictions are an important input to modern planning and control systems because they enable robust control decisions. Given the promising results shown here, the flexibility of the model with respect to input features, and the applicability of this technique to real-time operation, we are confident our approach can be adapted to field robotic systems. The success of this approach on a truly challenging learning problem suggests that dictionary methods may hold promise for collaborative multi-agent learning, which we investigate in the next chapter.

Chapter 5

Dictionary Learning in Multi-Agent Systems

In Chapters 2 and 3, we observe that GLM-based methods while provably stable, lack competitive accuracy performance. On the other hand, dictionary learning methods define non-convex optimization problems which elude stability, but tend to perform well in practice (Chapter 4). Motivated by these observations, in this chapter we attempt to extend supervised dictionary methods to multi-agent settings with streaming data. We exploit the Lagrange duality tools that work well for the GLM setting, and the techniques developed in the previous chapter based on dictionary learning. Ultimately, we would like to develop tools for multi-agent systems to achieve stability as well as achieve good inference performance from streaming data.

One salient motivation for this problem formulation is designing robotic teams which exhibit real-time adaptivity. More specifically, consider a team of mobile robots deployed throughout an unknown environment and charged with some high-level task such as exploration or navigation. Critical to the success of this team is the ability of each platform to learn from and adapt to its new surroundings. While individual platforms can make and learn from their own local observations, it would be far more efficient if the team could perform these tasks jointly using its communication network. In self-supervised learning-for-control tasks, for example, the team as a whole can more quickly cover the entire observation space, thereby providing individual platforms access to observations that they may not be able to experience locally. Additionally, the quality of inferences based on information aggregated over the entire network are likely to be superior to those based on local observations, based on classical theory of consistency of statistical estimators.

Thus, we extend online discriminative dictionary learning [12] to networked settings, where a team of agents seeks to learn a common dictionary and model parameters based upon streaming data. We consider tools from stochastic approximation [161] and its de-

centralized extensions which have incorporated ideas from distributed optimization such as weighted averaging [77, 134, 157], dual reformulations where each agent ascends in the dual domain [78, 154], and primal-dual methods [9, 93, 134].

Our main technical contribution is the formulation of the online multi-agent discriminative dictionary learning problem as a distributed stochastic program and the development of a block variant of the primal-dual algorithm proposed in [9, 93] (Section 5.2). Moreover, we establish that under an attenuating step-size regime, an asymptotic stationary solution is attained in expectation in Section 5.3. In Section 5.4, we analyze the proposed framework's empirical performance on a texture classification problem based upon image data for a variety of network settings and demonstrate its capacity to solve a new class of collaborative multi-class classification problems in decentralized settings. In Section 5.5 we consider the algorithm's use in a mobile robotic team for navigability assessment of unknown environments.

5.1 Task-Driven Dictionaries for Multi-Agent Systems

We want to solve (4.12) in distributed settings where signal and observation pairs are independently observed by agents of a network. Agents aim to learn a dictionary and model parameters that are common with all others while having access to local information only. In particular, associated with each agent i is a local random variable and associated output variable $(\mathbf{x}_i, \mathbf{y}_i)$ and each agent's goal is to learn over the aggregate training domain $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{V}$. As in Chapters 2 and 3, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a symmetric and connected network with node set $\mathcal{V} = \{1, \ldots, V\}$ and $E = |\mathcal{E}|$ directed edges of the form $e = (i, j)$ and further define the neighborhood of i as the set of nodes $n_i := \{j : (i, j) \in \mathcal{E}\}$ that share an edge with i. When each of the V agents observes a pair $(\mathbf{x}_i, \mathbf{y}_i)$, the function h in (4.12) can be written as a sum of local losses,

$$\ell(\mathbf{D}, \mathbf{w}; (\mathbf{x}, \mathbf{y})) = \sum_{i=1}^{V} \ell_i(\mathbf{D}_i, \mathbf{w}_i; (\mathbf{x}_i, \mathbf{y}_i)), \tag{5.1}$$

where we have defined the vertically concatenated dictionary $\mathbf{D} := [\mathbf{D}_1; \ldots; \mathbf{D}_V] \in \mathbb{R}^{Vm \times k}$, with each $\mathbf{D}_i \in \mathbb{R}^{m \times k}$ and model parameter $\mathbf{w} := [\mathbf{w}_1; \ldots; \mathbf{w}_V] \in \mathbb{R}^{Vk}$.

Substituting (5.1) into the objective in (4.12) yields a problem in which the agents learn dictionaries and classifiers that depend on their local observations only. The problem to be formulated here is one in which the agents learn common dictionaries and models. Since the network $\mathcal{G}$ is assumed to be connected, this relationship can be attained by imposing the constraints $\mathbf{D}_i = \mathbf{D}_j$ and $\mathbf{w}_i = \mathbf{w}_j$ for all pairs of neighboring nodes $(i, j) \in \mathcal{E}$. With

these constraints, we obtain the distributed stochastic program

$$\{\mathbf{D}_i^*, \mathbf{w}_i^*\}_{i=1}^V := \underset{\mathbf{D}_i \in \mathcal{D}, \mathbf{w}_i \in \mathcal{W}}{\operatorname{argmin}} \quad \sum_{i=1}^{V} \mathbb{E}_{\mathbf{y}_i, \mathbf{x}_i} \Big[\ell_i(\mathbf{D}_i, \mathbf{w}_i; (\mathbf{x}_i, \mathbf{y}_i) \Big]$$
$$\text{s. t.} \qquad \mathbf{D}_i = \mathbf{D}_j, \ \mathbf{w}_i = \mathbf{w}_j, \ j \in n_i. \tag{5.2}$$

When the agreement constraints in (5.2) are satisfied, the objective is equivalent to one in which all the observations are made at a central location and a single dictionary and model are learnt. Thus, (5.2) corresponds to a problem in which each agent i, having only observed the local pairs $(\mathbf{x}_i, \mathbf{y}_i)$, aims to learn a dictionary representation and model parameters that are optimal when information is aggregated globally over the network. The decentralized discriminative learning problem is to develop an iterative algorithm that relies on communication with neighbors only so that agent i learns the optimal (common) dictionary $\mathbf{D}_i^* = \mathbf{D}_j^*$ and discriminative model $\mathbf{w}_i^* = \mathbf{w}_j^*$. We present in the following section an algorithm that is shown in Section 5.3 to converge to a local optimum of (5.2).

Remark 4 Decentralized learning techniques may be applied to solving pattern recognition tasks in networks of autonomous robots operating in real-time, provided that realizations of the output variables are generated by a process which is internal to the individual platforms. In particular, consider the formulation in (5.2), and let $\mathbf{y}_i$ represent the difference between state information associated with a commanded trajectory and that which is observed by on-board sensors of robot i. Most robots are equipped with sensors such as gyroscopes, accelerometers, and inertial measurement units, which make this self-supervisory information available. In this case, the interconnected network of robots does not need external supervision or human in the loop in order to perform discriminative learning. In Section 5.5 we propose solving problems of the form (5.2) in a network of interconnected robots operating in a field setting by tracking the difference between measurements made via inertial measurement units (IMU) and movements which are controlled by a joystick.

5.2 Block Saddle Point Method

To write the constraints in (5.2) more compactly, define the augmented graph edge incidence matrix $\mathbf{C}_D \in \mathbb{R}^{(E\times V)(m\times k)}$ associated with the dictionary constraint. The matrix $\mathbf{C}_D$ is formed by $E\times V$ square blocks of dimension $m\times k$. If the edge $e = (i, j)$ links node i to node j the block (e, i) is $[\mathbf{C}_D]_{ei} = \mathbf{I}_{mk}$ and the block $[\mathbf{C}_D]_{ej} = -\mathbf{I}_{mk}$, where $\mathbf{I}_m$ denotes the identity matrix of dimension mk. All other blocks are identically null, i.e., $[\mathbf{C}]_{ei} = [\mathbf{C}]_{ej} = \mathbf{0}_{m\times k}$ when $e \neq (i, j)$. Likewise, the matrix $\mathbf{C}_w \in \mathbb{R}^{(E\times V)k}$ is defined by $E\times V$ blocks of dimension k with $[\mathbf{C}_w]_{ei} = \mathbf{I}_k$ and $[\mathbf{C}_w]_{ej} = -\mathbf{I}_k$ when $e = (i, j)$ and $[\mathbf{C}]_{ei} = [\mathbf{C}]_{ej} = \mathbf{0}_k$ otherwise.

Then the constraints $\mathbf{D}_i = \mathbf{D}_j$ and $\mathbf{w}_i = \mathbf{w}_j$ for all pairs of neighboring nodes can be written as

$$\mathbf{C}_D\mathbf{D} = \mathbf{0}, \qquad \mathbf{C}_w\mathbf{w} = \mathbf{0}. \tag{5.3}$$

The edge incidence matrices $\mathbf{C}_D$ and $\mathbf{C}_w$ have exactly mk and k null singular values, respectively. We denote as $0 < \gamma$ the smallest nonzero singular value of $\mathbf{C} := [\mathbf{C}_D; \mathbf{C}_w]$ and as Γ the largest singular value of $\mathbf{C}$, which both measure network connectedness.

Imposing the constraints in (5.3) for all realizations of the local random variables requires global coordination – indeed, the formulation would be equivalent to the centralized problem in (4.12). Instead, we consider a modification of (5.1) in which we add linear penalty terms to incentivize the selection of coordinated actions. Introduce then dual variables $\mathbf{\Lambda}_e = \mathbf{\Lambda}_{ij} \in \mathbb{R}^{m\times k}$ associated with the constraint $\mathbf{D}_i - \mathbf{D}_j = \mathbf{0}$ and consider the addition of penalty terms of the form $\mathrm{tr}[\mathbf{\Lambda}_{ij}^T(\mathbf{D}_i - \mathbf{D}_j)]$. For an edge that starts at node i, the multiplier $\mathbf{\Lambda}_{ij}$ is assumed to be kept at node i. Similarly, introduce dual variables $\boldsymbol{\nu}_{ij}$ associated with the constraint $\mathbf{w}_i - \mathbf{w}_j = \mathbf{0}$ for all neighboring node pairs and penalty terms $\boldsymbol{\nu}_{ij}^T(\mathbf{w}_i - \mathbf{w}_j)$. By introducing the stacked matrices $\mathbf{\Lambda} := [\mathbf{\Lambda}_1; \ldots; \mathbf{\Lambda}_E] \in \mathbb{R}^{Em\times k}$ and $\boldsymbol{\nu} := [\boldsymbol{\nu}_1; \ldots; \boldsymbol{\nu}_E] \in \mathbb{R}^{Ek}$ which are restricted to compact convex sets $\mathfrak{L}$ and $\mathcal{N}$, we can write the Lagrangian of this problem as

$$\mathcal{L}(\mathbf{D}, \mathbf{w}, \mathbf{\Lambda}, \boldsymbol{\nu}) \;=\; \sum_{i=1}^{V} \mathbb{E}_{\mathbf{y}_i,\mathbf{x}_i}\left[\ell_i(\mathbf{D}_i, \mathbf{w}_i; (\mathbf{x}_i, \mathbf{y}_i))\right] + \mathrm{tr}\big(\mathbf{\Lambda}^T\mathbf{C}_D\mathbf{D}\big) + \boldsymbol{\nu}^T\mathbf{C}_w\mathbf{w}\;. \tag{5.4}$$

The problem in (5.2) is nonconvex. Thus, we use the dual formulation in (5.4) to develop a distributed algorithm that converges to a KKT point of (5.2).

To do so, suppose agent i receives local observation pairs $(\mathbf{x}_{i,t}, \mathbf{y}_{i,t})$ at time t and define the instantaneous Lagrangian as the stochastic approximation of (5.4) evaluated with the observations $\{(\mathbf{x}_{i,t}, \mathbf{y}_{i,t})\}_{i=1}^V$ aggregated across the network as

$$\hat{\mathcal{L}}_t(\mathbf{D}, \mathbf{w}, \mathbf{\Lambda}, \boldsymbol{\nu}) \;=\; \sum_{i=1}^{V} \ell_i(\mathbf{D}_i, \mathbf{w}_i; (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})) + \mathrm{tr}\big(\mathbf{\Lambda}^T\mathbf{C}_D\mathbf{D}\big) + \boldsymbol{\nu}^T\mathbf{C}_w\mathbf{w}\;. \tag{5.5}$$

We consider the use of the Arrow-Hurwicz saddle point method to solve (5.2) by alternating block variable updates, in order to exploit the fact that primal-dual stationary pairs are saddle points of the Lagrangian to work through successive primal alternating gradient descent steps and dual gradient ascent steps. We first define orthogonal projection of a vector $\mathbf{u} \in \mathbb{R}^p$ onto a convex set $\mathcal{C} \subset \mathbb{R}^p$ as $\mathcal{P}_{\mathcal{C}}[\mathbf{u}] = \mathrm{argmin}_{\mathbf{v}} \|\mathbf{v} - \mathbf{u}\|_2^2$. Applied to the stochastic approximate Lagrangian in (5.5), the primal step of the projected stochastic

saddle point method takes the form

$$\mathbf{D}_{t+1} = \mathcal{P}_{\mathcal{D}^V}\left[\mathbf{D}_t - \epsilon_t \nabla_{\mathbf{D}} \hat{\mathcal{L}}_t(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)\right] , \tag{5.6}$$

$$\mathbf{w}_{t+1} = \mathcal{P}_{\mathcal{W}^V}\left[\mathbf{w}_t - \epsilon_t \nabla_{\mathbf{w}} \hat{\mathcal{L}}_t(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)\right] , \tag{5.7}$$

where $\nabla_{\mathbf{D}} \hat{\mathcal{L}}_t(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)$ and $\nabla_{\mathbf{w}} \hat{\mathcal{L}}_t(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)$, are stochastic subgradients of the Lagrangian with respect to $\mathbf{D}$ and $\mathbf{w}$, respectively. $\mathcal{P}_{\mathcal{D}^V}[\cdot]$ and $\mathcal{P}_{\mathcal{W}^V}[\cdot]$ denote orthogonal projections onto sets $\mathcal{D}^V$ and $\mathcal{W}^V$, which are V-fold Cartesian products of respective sets $\mathcal{D}$ and $\mathcal{W}$. The dual iteration is defined as

$$\mathbf{\Lambda}_{t+1} = \mathcal{P}_{\mathfrak{L}^E}\left[\mathbf{\Lambda}_t + \eta_t \nabla_{\mathbf{\Lambda}} \hat{\mathcal{L}}_t(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)\right] , \tag{5.8}$$

$$\boldsymbol{\nu}_{t+1} = \mathcal{P}_{\mathcal{N}^E}\left[\boldsymbol{\nu}_t + \eta_t \nabla_{\boldsymbol{\nu}} \hat{\mathcal{L}}_t(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)\right] , \tag{5.9}$$

where $\nabla_{\mathbf{\Lambda}} \hat{\mathcal{L}}_t(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)$ and $\nabla_{\boldsymbol{\nu}} \hat{\mathcal{L}}(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)$ are the stochastic subgradients of the Lagrangian with respect to $\mathbf{\Lambda}$ and $\boldsymbol{\nu}$, respectively. Moreover, $\mathcal{P}_{\mathfrak{L}^E}[\cdot]$ and $\mathcal{P}_{\mathcal{N}^E}[\cdot]$ denote orthogonal projections onto dual feasible sets $\mathfrak{L}^E$ and $\mathcal{N}^E$ which are compact subsets of Euclidean space of respective dimension Ek and $Em \times k$. Additionally, η_t is a step size chosen as $\mathcal{O}(1/t)$ – see Section 5.3.

We now show that the algorithm specified by (5.6)-(5.9) yields an effective tool for discriminative learning in multi-agent settings.

Proposition 2 *The gradient computations in* (5.6)-(5.7) *may be separated along the local primal variables* $\mathbf{D}_{i,t}$ *and* $\mathbf{w}_{i,t}$ *associated with node* i*, yielding* $2V$ *parallel updates*

$$\mathbf{D}_{i,t+1} = \mathcal{P}_{\mathcal{D}}\Big[\mathbf{D}_{i,t} - \eta_t\Big(\nabla_{\mathbf{D}_i} \ell_i(\mathbf{D}_{i,t}, \mathbf{w}_{i,t}; (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})) + \sum_{j \in n_i} (\mathbf{\Lambda}_{ij,t} - \mathbf{\Lambda}_{ji,t})\Big)\Big] , \tag{5.10}$$

$$\mathbf{w}_{i,t+1} = \mathcal{P}_{\mathcal{W}}\Big[\mathbf{w}_{i,t} - \eta_t\Big(\nabla_{\mathbf{w}_i} \ell_i(\mathbf{D}_{i,t}, \mathbf{w}_{i,t}; (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})) + \sum_{j \in n_i} (\boldsymbol{\nu}_{ij,t} - \boldsymbol{\nu}_{ji,t})\Big)\Big] , \tag{5.11}$$

where $\mathcal{P}_{\mathcal{D}}[\cdot]$ *denotes the orthogonal projection operator onto set* $\mathcal{D}$*, and likewise for* $\mathcal{P}_{\mathcal{W}}[\cdot]$*. Moreover, the dual gradients in the updates of* $\mathbf{\Lambda}_{ij,t}$ *and* $\boldsymbol{\nu}_{ij,t}$ *respectively in* (5.8)-(5.9) *may separated into* $2E$ *parallel updates associated with edge* (i, j)

$$\mathbf{\Lambda}_{ij,t+1} = \mathcal{P}_{\mathfrak{L}}\Big[\mathbf{\Lambda}_{ij,t} + \eta_t \left(\mathbf{D}_{i,t+1} - \mathbf{D}_{j,t+1}\right)\Big] , \tag{5.12}$$

$$\boldsymbol{\nu}_{ij,t+1} = \mathcal{P}_{\mathcal{N}}\Big[\boldsymbol{\nu}_{ij,t} + \eta_t \left(\mathbf{w}_{i,t+1} - \mathbf{w}_{j,t+1}\right)\Big] , \tag{5.13}$$

which allows for distributed computation across the network. Again, $\mathcal{P}_{\mathfrak{L}}[\cdot]$ *and* $\mathcal{P}_{\mathcal{N}}[\cdot]$ *denote projections onto sets* $\mathfrak{L}$ *and* $\mathcal{N}$*.*

Proof:

The set $\mathcal{D}^V$ may be written as a Cartesian product of sets $\mathcal{D}$. We assume that projection $\mathcal{P}_{\mathcal{D}^V}[\cdot]$ of the stacked iterates $\mathbf{D}$ into $\mathcal{D}^V$ is equivalent to the separate projection $\mathcal{P}_{\mathcal{D}}[\cdot]$ of the components node-wise components $\mathbf{D}_i$ into the sets $\mathcal{D}$. The other primal domain, as well as the dual domains, are defined as Cartesian products of lower dimensional sets for each node and edge. We assume a similar condition holds for the set projections with onto the stacked primal and dual sets $\mathcal{W}^V$, $\mathfrak{L}^M$, and $\mathcal{N}^M$, allowing the stacked iterates to be separated into their local node and edge-wise components via projections onto local sets $\mathcal{W}$, $\mathfrak{L}$, and $\mathcal{N}$.

To compute the primal stochastic gradient of the Lagrangian [cf. (5.4)] with respect to a local dictionary $\mathbf{D}_{i,t}$ for a signal-output pair $(\mathbf{x}_{i,t}, \mathbf{y}_{i,t})$, apply the node-separability of the global cost in (5.1) to the first term in (5.4), and note that all terms of the derivative of the second term with respect to $\mathbf{D}_i$ in (5.4) are null except those associated with node i and neighbors j, to obtain

$$\nabla_{\mathbf{D}_i} \hat{\mathcal{L}}_t(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) = \nabla_{\mathbf{D}_i} \ell_i(\mathbf{D}_{i,t}, \mathbf{w}_{i,t}; (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})) + \sum_{j \in n_i} (\mathbf{\Lambda}_{ij,t} - \mathbf{\Lambda}_{ji,t}) \tag{5.14}$$

Substitute this into the update (5.6) and separate update along direction associated with agent i to obtain

$$\mathbf{D}_{i,t+1} = \mathcal{P}_{\mathcal{D}}\Big[\mathbf{D}_{i,t} - \eta_t\Big(\nabla_{\mathbf{D}_i} \ell_i(\mathbf{D}_{i,t}, \mathbf{w}_{i,t}; (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})) + \sum_{j \in n_i} (\mathbf{\Lambda}_{ij,t} - \mathbf{\Lambda}_{ji,t})\Big)\Big] \,, \tag{5.15}$$

Analogous logic applies to the update for $\mathbf{w}_i$ at node i, and is thus omitted. To develop the dual variable updates, compute the stochastic subgradient of (5.4) with respect to the Lagrange multipliers associated with edge (i, j) and the dictionary agreement constraint. Note that all terms in the sum $\mathrm{tr}(\mathbf{\Lambda}^T \mathbf{C}\mathbf{D})$ are null except those associated with edge (i, j) to obtain

$$\nabla_{\mathbf{\Lambda}_{ij}} \hat{\mathcal{L}}_t(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) = \mathbf{D}_{i,t+1} - \mathbf{D}_{j,t+1} \,. \tag{5.16}$$

This local subgradient corresponds to the communication link between agent i and agent j. Separating the update in (5.9) along variables associated with edge (i, j), we obtain the local update

$$\mathbf{\Lambda}_{ij,t+1} = \mathcal{P}_{\mathfrak{L}}\Big[\mathbf{\Lambda}_{ij,t} + \eta_t \left(\mathbf{D}_{i,t+1} - \mathbf{D}_{j,t+1}\right)\Big] \,. \tag{5.17}$$

Again, analogous reasoning regarding the agreement constraint slack term for $\mathbf{w}$ yields the update for Lagrange multiplier $\boldsymbol{\nu}_{ij}$. Thus we obtain the statement in Proposition 2. ■

The D4L algorithm follows by letting node i implement (5.10)-(5.11) as we summarize

Algorithm 4 D4L: Decentralized Dynamic Discriminative Dictionary Learning

Require: Initialization $(\mathbf{D}_0, \mathbf{w}_0, \mathbf{\Lambda}_0, \boldsymbol{\nu}_0)$, regularizers ζ_1, ζ_2 , step-size η_t, network $\mathcal{G}$.

1: **for** $t = 0, 1, 2, \ldots$ **do**

2: Acquire local signal and observation pair $(\mathbf{x}_{i,t}, \mathbf{y}_{i,t})$.

3: Coding [cf. 4.10], $\boldsymbol{\alpha}^*_{i,t} := \operatorname{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^k} f(\boldsymbol{\alpha}, \mathbf{D}_{i,t}; \mathbf{x}_{i,t})$.

4: Send $(\mathbf{\Lambda}_{ij,t}, \boldsymbol{\nu}_{ij,t})$ and receive $(\mathbf{\Lambda}_{ji,t}, \boldsymbol{\nu}_{ji,t})$ for all $j \in n_i$.

5: Update dictionary and model parameters [cf. (5.10) and (5.11)]

$$\mathbf{D}_{i,t+1} = \mathcal{P}_{\mathcal{D}}\Big[\mathbf{D}_{i,t} - \eta_t\Big(\nabla_{\mathbf{D}_i}\ell_i(\mathbf{D}_{i,t}, \mathbf{w}_{i,t}; (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})) + \sum_{j\in n_i}(\mathbf{\Lambda}_{ij,t} - \mathbf{\Lambda}_{ji,t})\Big)\Big] ,$$

$$\mathbf{w}_{i,t+1} = \mathcal{P}_{\mathcal{W}}\Big[\mathbf{w}_{i,t} - \eta_t\Big(\nabla_{\mathbf{w}_i}\ell_i(\mathbf{D}_i, \mathbf{w}_{i,t}; (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})) + \sum_{j\in n_i}(\boldsymbol{\nu}_{ij,t} - \boldsymbol{\nu}_{ji,t})\Big)\Big] .$$

6: Send $(\mathbf{D}_{i,t}, \mathbf{w}_{i,t})$ and receive $(\mathbf{D}_{j,t}, \boldsymbol{\nu}_{j,t})$ for all $j \in n_i$.

7: Update Lagrange multipliers [cf. (5.12) and (5.13)]

$$\mathbf{\Lambda}_{ij,t+1} = \mathcal{P}_{\mathfrak{L}}\Big[\mathbf{\Lambda}_{ij,t} + \eta_t\left(\mathbf{D}_{i,t+1} - \mathbf{D}_{j,t+1}\right)\Big] ,$$

$$\boldsymbol{\nu}_{ij,t+1} = \mathcal{P}_{\mathcal{N}}\Big[\boldsymbol{\nu}_{ij,t} + \eta_t\left(\mathbf{w}_{i,t+1} - \mathbf{w}_{j,t+1}\right)\Big] .$$

8: **end for**

in Algorithm 4. To do so, node i utilizes its local primal iterates $\mathbf{D}_{i,t}$ and $\mathbf{w}_{i,t}$, its local dual iterates $\mathbf{\Lambda}_{ij,t}$ and $\boldsymbol{\nu}_{ij,t}$, and its local instantaneous observed pair $(\mathbf{x}_{i,t}, \mathbf{y}_{i,t})$. The variable $\mathbf{D}_{i,t} \in \mathbb{R}^{m\times k}$ is the local dictionary matrix associated with agent i, and $\mathbf{w}_{i,t} \in \mathbb{R}^k$ is its associated parameter vector (regressor or classifier). Node i also needs access to the neighboring multipliers $\mathbf{\Lambda}_{ji}$ and $\boldsymbol{\nu}_{ji}$ to implement (5.10) and (5.11) as well as to the neighboring primal iterates $\mathbf{D}_{j,t}$ and $\mathbf{w}_{j,t}$ to implement (5.12) and (5.13). The core steps of D4L in Algorithm 4 are the primal iteration in Step 5 and the dual iteration in Step 7. Steps 4 and 6 refer to the exchange of dual and primal variables that are necessary to implement steps 5 and 7, respectively. Step 1 refers to the acquisition of the signal and observation pair and Step 2 to the computation of the code in 4.10 using the local current dictionary iterate $\mathbf{D}_{i,t}$. We discuss the specific use of Algorithm 4 to learning discriminative sparse signal representations in a distributed setting to clarify ideas.

Example 3 (Distributed sparse dictionary learning) Consider a multi-agent system in which signals are independently observed at each agent, and the data domain has latent structure which may be revealed via learning discriminative representations that are sparse. In this case, we select the particular form of f in (4.10) as the elastic net [cf. (4.5)] with the Euclidean distance $\tilde{\ell}(\boldsymbol{\alpha}_t, \tilde{\mathbf{D}}; \mathbf{x}_t) = \|\mathbf{x}_t - \tilde{\mathbf{D}}\boldsymbol{\alpha}_t\|^2/2$. Then the dictionary update in (5.10)

may be derived from the subgradient optimality conditions of the elastic-net (see [13]):

$$\begin{aligned} &\mathbf{d}_l(\mathbf{x}_{i,t} - \tilde{\mathbf{D}}\boldsymbol{\alpha}^*) - \zeta_2\boldsymbol{\alpha}_l^* = \zeta_1 \text{sgn}(\boldsymbol{\alpha}_l^*) && \text{if } \boldsymbol{\alpha}_l^* \neq 0 \ , \\ &\mathbf{d}_l(\mathbf{x}_{i,t} - \tilde{\mathbf{D}}\boldsymbol{\alpha}^*) - \zeta_2\boldsymbol{\alpha}_l^* \leq \zeta_1 && \text{otherwise} \ , \end{aligned} \tag{5.18}$$

where $\text{sgn}(\boldsymbol{\alpha}^*)$ is a vector of signs of $\boldsymbol{\alpha}^*$. Proceeding as in the Appendix of [12], define $Z \subset \{1, \ldots, k\}$ as the set of nonzero entries of $\boldsymbol{\alpha}^* = \boldsymbol{\alpha}^*(\mathbf{x}, \tilde{\mathbf{D}})$. Then $\boldsymbol{\alpha}^*$ is the solution to the system of linear inequalities in (5.18), i.e.

$$\boldsymbol{\alpha}_Z^* = (\tilde{\mathbf{D}}_Z^T \tilde{\mathbf{D}}_Z + \zeta_2 I)^{-1}(\tilde{\mathbf{D}}_Z \mathbf{x} - \zeta_1 \text{sgn}(\boldsymbol{\alpha}^*)_Z) \ . \tag{5.19}$$

At time t, to compute the stochastic gradient with of (5.4) respect to a local dictionary, apply Proposition 1 of [12] which yields

$$\nabla_{\mathbf{D}_i} \hat{\mathcal{L}}_t(\mathbf{D}_t, \mathbf{w}_t, \boldsymbol{\Lambda}_t) = -\mathbf{D}_{i,t}\boldsymbol{\beta}_{i,t}\boldsymbol{\alpha}_{i,t}^* + (\mathbf{x}_{i,t} - \mathbf{D}_{i,t}\boldsymbol{\alpha}_{i,t}^*)\boldsymbol{\beta}_{i,t}{}^T + \sum_{j \in n_i} (\boldsymbol{\Lambda}_{ij,t} - \boldsymbol{\Lambda}_{ji,t}) \ . \tag{5.20}$$

$\boldsymbol{\alpha}_{i,t}^* = \boldsymbol{\alpha}_{i,t}^*(\mathbf{D}_{i,t}; \mathbf{x}_{i,t})$ is shorthand for (4.5) and $Z_{i,t}$ is defined as the set of indices associated with nonzero entries of $\boldsymbol{\alpha}_{i,t}^*$. Moreover, we define $\boldsymbol{\beta}^{i,t} \in \mathbb{R}^k$ as

$$\begin{aligned} \boldsymbol{\beta}_l^{i,t} &= ([\mathbf{D}_{i,t}]_l^T [\mathbf{D}_{i,t}]_l + \zeta_2 I)^{-1} \times \\ &\qquad \nabla_{\boldsymbol{\alpha}_l} \ell_i(\mathbf{D}_{i,t}, \mathbf{w}_{i,t}; (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})) \text{ if } l \in Z_{i,t} \ , \\ \boldsymbol{\beta}_l^{i,t} &= 0 \text{ if } l \notin Z_{i,t} \ , \end{aligned} \tag{5.21}$$

as in [12], Proposition 1. This result is established via a perturbation analysis of the elastic-net optimality conditions, substituting the solution of (4.5) into ℓ_i, and applying the chain rule.

5.3 Convergence Analysis

We turn to establishing that the saddle point algorithm in (5.6)-(5.9) asymptotically converges to a stationary point of the problem (5.2). Before proceeding with our analysis, we define the primal descent direction with respect to $\mathbf{D}$ associated with the projected block stochastic saddle point method as

$$\tilde{\nabla}_{\mathbf{D}} \hat{\mathcal{L}}_t(\mathbf{D}_t, \mathbf{w}_t, \boldsymbol{\Lambda}_t, \boldsymbol{\nu}_t) = \Big(\mathbf{D}_t - \mathcal{P}_{\mathcal{D}^V}\Big[\mathbf{D}_t - \eta_t \nabla_{\mathbf{D}} \hat{\mathcal{L}}_t(\mathbf{D}_t, \mathbf{w}_t, \boldsymbol{\Lambda}_t, \boldsymbol{\nu}_t)\Big]\Big)/\eta_t \ , \tag{5.22}$$

and the dual ascent direction with respect to $\mathbf{\Lambda}$ as

$$\tilde{\nabla}_{\mathbf{\Lambda}}\hat{\mathcal{L}}_t(\mathbf{D}_{t+1},\mathbf{w}_{t+1},\mathbf{\Lambda}_t,\boldsymbol{\nu}_t)=\Big(\mathbf{\Lambda}_t-\mathcal{P}_{\mathfrak{L}^E}\Big[\mathbf{\Lambda}_t+\eta_t\nabla_{\mathbf{\Lambda}}\hat{\mathcal{L}}_t(\mathbf{D}_{t+1},\mathbf{w}_{t+1},\mathbf{\Lambda}_t,\boldsymbol{\nu}_t)\Big]\Big)/\eta_t\ . \quad (5.23)$$

The projected stochastic gradients $\tilde{\nabla}_{\mathbf{w}}\hat{\mathcal{L}}_t(\mathbf{D}_t,\mathbf{w}_t,\mathbf{\Lambda}_t,\boldsymbol{\nu}_t)$ and $\tilde{\nabla}_{\boldsymbol{\nu}}\hat{\mathcal{L}}_t(\mathbf{D}_{t+1},\mathbf{w}_{t+1},\mathbf{\Lambda}_t,\boldsymbol{\nu}_t)$ associated with variables $\mathbf{w}$ and $\boldsymbol{\nu}$ are analogously defined to (5.22) and (5.23), respectively. Note descent (respectively, ascent) using projected stochastic gradients in [cf. (5.22) - (5.23)] is equivalent to using the projected stochastic saddle point method [cf. (5.10) - (5.13)].

To establish convergence of D4L, some technical conditions are required which we state below.

AS10 *(Connected Network) The network $\mathcal{G}$ is connected with diameter D. The singular values of the incidence matrix $\mathbf{C}$ are respectively upper and lower bounded by Γ and $\gamma>0$.*

AS11 *(Smoothness) The Lagrangian has Lipschitz continuous gradients in the primal and dual variables with constants $L_{\mathbf{D}}$, $L_{\mathbf{w}}$, $L_{\mathbf{\Lambda}}$, and $L_{\boldsymbol{\nu}}$. This implies that, e.g.,*

$$\|\nabla_{\mathbf{D}}\mathcal{L}(\mathbf{D},\mathbf{w},\mathbf{\Lambda},\boldsymbol{\nu})-\nabla_{\mathbf{D}}\mathcal{L}(\tilde{\mathbf{D}},\mathbf{w},\mathbf{\Lambda},\boldsymbol{\nu})\|\leq L_{\mathbf{D}}\|\mathbf{D}-\tilde{\mathbf{D}}\|_F\ . \quad (5.24)$$

Moreover, the projected gradients of the Lagrangian in the primal and dual variables are bounded with block constants $G_{\mathbf{D}}$, $G_{\mathbf{w}}$, $G_{\mathbf{\Lambda}}$, and $G_{\boldsymbol{\nu}}$, which implies that, e.g.,

$$\|\tilde{\nabla}_{\mathbf{D}}\mathcal{L}(\mathbf{D},\mathbf{w},\mathbf{\Lambda},\boldsymbol{\nu})\|\leq G_{\mathbf{D}}\ . \quad (5.25)$$

AS12 *(Diminishing step-size) The step-size η_t satisfies*
(i) $\sum_{t=1}^{\infty}\eta_t=\infty$, (non-summability)
(ii) $\sum_{t=0}^{\infty}\eta_t^2<\infty$, (square-summability).

AS13 *(Stochastic Approximation Error) The bias of the stochastic gradients of the Lagrangian with respect to each block variable asymptotically converges to null at a rate on the order of the algorithm step-size, which allows us to write, e.g.*

$$\|\mathbb{E}\left[\boldsymbol{\delta}_{\mathbf{D},t}\,\middle|\,\mathcal{F}_t\right]\|\leq A\eta_t\ , \quad (5.26)$$

where $\boldsymbol{\delta}_{\mathbf{D},t}:=\tilde{\nabla}_{\mathbf{D}}\mathcal{L}(\mathbf{D}_t,\mathbf{w}_t,\mathbf{\Lambda}_t,\boldsymbol{\nu}_t)-\tilde{\nabla}_{\mathbf{D}}\hat{\mathcal{L}}_t(\mathbf{D}_t,\mathbf{w}_t,\mathbf{\Lambda}_t,\boldsymbol{\nu}_t)$ denotes the stochastic errors of the Lagrangian with respect to the dictionary $\mathbf{D}$. $\boldsymbol{\delta}_{\mathbf{w},t}$, $\boldsymbol{\delta}_{\mathbf{\Lambda},t}$, and $\boldsymbol{\delta}_{\boldsymbol{\nu},t}$ are similarly defined for the other block variables.

Moreover, let $\mathcal{F}_t$ be a sigma algebra that measures the history of the system up until time t. Then, the conditional second moments of the projected stochastic gradients are bounded

by σ^2 for all times t, which for example allows us to write

$$\mathbb{E}\left[\|\tilde{\nabla}_{\mathbf{D}}\hat{\mathcal{L}}_t(\mathbf{D}_t, \mathbf{w}_t, \boldsymbol{\Lambda}_t, \boldsymbol{\nu}_t)\|^2 \,\middle|\, \mathcal{F}_t\right] \leq \sigma^2. \tag{5.27}$$

We define σ^2 in (5.27) *as a worst-case bound on the projected stochastic gradient variance with respect to $\mathbf{D}$, $\mathbf{w}$, $\boldsymbol{\Lambda}$, and $\boldsymbol{\nu}$.*

Assumption 10 is standard in distributed algorithms (see, for instance, [157]). Moreover, Assumption 11 is common in analysis of descent methods dating back to [21], and is guaranteed to hold by making use of iterates which are projected into compact sets $\mathcal{D}^V$, $\mathcal{W}^V$, $\mathfrak{L}^E$, and $\mathcal{N}^E$. For instance, in [134], orthogonal projections of the primal and dual iterates are used to guarantee boundedness of gradients. Assumption 12 specifies that a diminishing step-size condition for the algorithm must be used, which frequently appears as a condition for attaining almost sure convergence of stochastic methods [138, 198]. Additionally, Assumption 13 provides conditions on the stochastic approximation errors, both of which have been considered in stochastic optimization with non-convex objectives [215].

Remark 5 Assumption 12 stipulates that Algorithm 4 is run with a diminishing step-size. Therefore, the magnitude of the difference between subsequent algorithm iterates is attenuating as the step-size

$$\mathbf{D}_{t+1} - \mathbf{D}_t = -\eta_t \tilde{\nabla}_{\mathbf{D}}\hat{\mathcal{L}}_t(\mathbf{D}_t, \mathbf{w}_t, \boldsymbol{\Lambda}_t, \boldsymbol{\nu}_t) \propto \eta_t$$

It's well known that diminishing step-sizes in stochastic methods asymptotically make the stochastic error of the algorithm go to null at a rate comparable to the step-size [138]. The stochastic error induced by set projections is proportional to the overall stochastic error of the algorithm, and therefore is also proportional to the step-size. The statement (5.26) in Assumption 13 makes a statement quantifying this error. Additionally, note that the right-hand side of this expression is null in the deterministic setting.

Observe that the projected stochastic gradients in the updates in (5.6) - (5.7) imply that the primal variables themselves are contained in compact sets $\mathcal{D}^V$ and $\mathcal{W}^V$, which allows us to write

$$\|\mathbf{D}\|_F \leq \sqrt{Vk}\,, \qquad \|\mathbf{w}\| \leq K_{\mathbf{w}}\,, \tag{5.28}$$

for all dictionaries $\mathbf{D} \in \mathcal{D}^V$ and model parameters $\mathbf{w} \in \mathcal{W}^V$. The compactness of dual sets $\mathfrak{L}$ and $\mathcal{N}$ ensure the primal gradients are bounded [cf.(5.25)], and the respective dual gradients in $\boldsymbol{\Lambda}$ and $\boldsymbol{\nu}$ are bounded by constants $G_{\boldsymbol{\Lambda}} = \Gamma\sqrt{Vk}$ and $G_{\boldsymbol{\nu}} = \Gamma K_{\mathbf{w}}$.

With the technical setting clarified, we may state our main result, which says that the proposed algorithm on average asymptotically achieves a first-order stationarity condition

of the Lagrangian associated with the optimization problem stated in (5.2).

Theorem 4 *Denote* $(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)$ *as the sequence generated by the block saddle point algorithm in* (5.6)-(5.9)*. If Assumptions 10 - 13 hold true, then the first-order stationary condition with respect to the primal variables*

$$\lim_{t\to\infty} \mathbb{E}[\|\tilde{\nabla}_{\mathbf{D}} \mathcal{L}(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)\|] = 0 \,, \tag{5.29}$$

$$\lim_{t\to\infty} \mathbb{E}[\|\tilde{\nabla}_{\mathbf{w}} \mathcal{L}(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)\|] = 0 \tag{5.30}$$

is asymptotically achieved in expectation. Moreover, the asymptotic feasibility condition

$$\lim_{t\to\infty} \mathbb{E}[\|\tilde{\nabla}_{\mathbf{\Lambda}} \mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)\|] = 0 \tag{5.31}$$

$$\lim_{t\to\infty} \mathbb{E}[\|\tilde{\nabla}_{\boldsymbol{\nu}} \mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)\|] = 0 \tag{5.32}$$

is attained in an expected sense.

Proof: The analysis is broken up into distinct components for the primal and dual variables. In the primal, we consider the Lagrangian difference of iterates at the next and current time. We expand terms, use properties of the stochastic gradients and function smoothness, and take conditional expectations on past information to establish a decrement property. We then mirror this analysis in the dual. At this point we leverage the step-size rules and apply (5.44). Then we consider the magnitude of block gradient differences which we bound by a term that diminishes with the step-size, which implies (5.45) holds, yielding an the expected asymptotic convergence to a stationary solution. We use the shorthand $\nabla_{\mathbf{D}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1} := \nabla_{\mathbf{D}} \mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)$ and $\nabla_{\mathbf{D}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t} := \nabla_{\mathbf{D}} \mathcal{L}(\mathbf{D}_{t}, \mathbf{w}_{t}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)$, and analogous notation for the other variables.

Begin by considering the difference of Lagrangians evaluated at the primal variables at the next and current time, and apply Taylor's Theorem to quadratically approximate the former term

$$\begin{aligned} \mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) - \mathcal{L}(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) \leq & \begin{bmatrix} \nabla_{\mathbf{D}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_t \\ \nabla_{\mathbf{w}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_t \end{bmatrix}^T \begin{bmatrix} \mathbf{D}_{t+1} - \mathbf{D}_t \\ \mathbf{w}_{t+1} - \mathbf{w}_t \end{bmatrix} \\ & + \frac{L_{\mathbf{D}}^2}{2} \|\mathbf{D}_{t+1} - \mathbf{D}_t\|_F^2 + \frac{L_{\mathbf{w}}^2}{2} \|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2, \end{aligned} \tag{5.33}$$

where we have applied the Lipschitz gradient property the Lagrangian to the final two terms as stated in (5.24) to the last term of (5.33). The difference of the current and next iterates may be written as $\mathbf{D}_{t+1} - \mathbf{D}_t = -\eta_t \tilde{\nabla}_{\mathbf{D}} \hat{\mathcal{L}}_t(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)$ and $\mathbf{w}_{t+1} - \mathbf{w}_t =$

$-\eta_t \tilde{\nabla}_{\mathbf{w}} \hat{\mathcal{L}}_t(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)$, which we substitute into the right hand side of (5.33), yielding

$$\mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) - \mathcal{L}(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) \leq -\eta_t \begin{bmatrix} \nabla_{\mathbf{D}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_t \\ \nabla_{\mathbf{w}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_t \end{bmatrix}^T \begin{bmatrix} \tilde{\nabla}_{\mathbf{D}} \hat{\mathcal{L}}_t(\cdot,\cdot,\cdot,\cdot)_t \\ \tilde{\nabla}_{\mathbf{w}} \hat{\mathcal{L}}_t(\cdot,\cdot,\cdot,\cdot)_t \end{bmatrix} + \frac{\eta_t^2}{2} \Big(L_{\mathbf{D}}^2 \|\tilde{\nabla}_{\mathbf{D}} \hat{\mathcal{L}}_t(\cdot,\cdot,\cdot,\cdot)_t\|_F^2 + L_{\mathbf{w}}^2 \|\tilde{\nabla}_{\mathbf{w}} \hat{\mathcal{L}}_t(\cdot,\cdot,\cdot,\cdot)_t\|^2 \Big), \quad (5.34)$$

Take the expectation of (5.34) conditional on the filtration $\mathcal{F}_t$, apply the finite conditional variance condition [cf. (5.27)] stated in Assumption 13 and use the definition of the projected stochastic gradient error of the Lagrangian with respect to $\mathbf{D}$ to write

$$\begin{aligned} &\mathbb{E}[\mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) - \mathcal{L}(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) \mid \mathcal{F}_t] \quad (5.35) \\ &\leq -\eta_t \begin{bmatrix} \nabla_{\mathbf{D}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_t \\ \nabla_{\mathbf{w}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_t \end{bmatrix}^T \begin{bmatrix} \tilde{\nabla}_{\mathbf{D}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_t - \boldsymbol{\delta}_{\mathbf{D},t} \\ \tilde{\nabla}_{\mathbf{w}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_t - \boldsymbol{\delta}_{\mathbf{w},t} \end{bmatrix} + \frac{\eta_t^2 \sigma^2}{2} \Big(L_{\mathbf{D}}^2 + L_{\mathbf{w}}^2 \Big), \end{aligned}$$

Now use the fact that the projected gradient method defines a descent direction, which appears, for instance, as [26], Lemma 2.1(i), to the first term of (5.35). We state a reformulation of this lemma here so that it is more amenable to our analysis as follows:

Lemma 3 *Let $\mathbf{w} \in \mathcal{W}$ where $\mathcal{W}$ is a convex set, and $\mathbf{d} = (\mathbf{w} - \mathcal{P}_{\mathcal{W}}[\mathbf{w} - \eta_t \nabla_{\mathbf{w}} \ell(\mathbf{w})])/\eta_t$ be the descent direction defined by projected gradient method, with ℓ a convex function of $\mathbf{w}$. Then the following holds*

$$\nabla_{\mathbf{w}} \ell(\mathbf{w})^T \mathbf{d} \geq \|(\mathbf{w} - \mathcal{P}_{\mathcal{W}}[\mathbf{w} - \eta_t \nabla_{\mathbf{w}} \ell(\mathbf{w})])/\eta_t\|^2 = \|\mathbf{d}\|^2 \quad (5.36)$$

Applying (5.36) to the first term on the right-hand side of (5.35) yields

$$\begin{aligned} &\mathbb{E}[\mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) - \mathcal{L}(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) \mid \mathcal{F}_t] \quad (5.37) \\ &\leq -\eta_t \left\| \begin{bmatrix} \tilde{\nabla}_{\mathbf{D}} \mathcal{L}_t(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) \\ \tilde{\nabla}_{\mathbf{w}} \mathcal{L}_t(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) \end{bmatrix} \right\|^2 + \eta_t^2 \Big(A(G_{\mathbf{D}} + G_{\mathbf{W}}) + \frac{\sigma^2}{2} \big(L_{\mathbf{D}}^2 + L_{\mathbf{w}}^2 \big) \Big), \end{aligned}$$

where we used the Cauchy-Schwartz inequality, the bias condition in (5.26), and the bound on the partial gradients of the Lagrangian with respect to $\mathbf{D}$ and $\mathbf{w}$ as stated in (5.25) of Assumption 11 to the second term inside the brackets on the right hand side of (5.35).

Set this analysis aside for now and consider Taylor expansion around Lagrangian evaluated at the dual iterates at the next and current time, which since the Lagrangian is a

linear function of its multipliers, allows us to write

$$\mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_{t+1}, \boldsymbol{\nu}_{t+1}) - \mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) = \begin{bmatrix} \nabla_{\mathbf{\Lambda}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1} \\ \nabla_{\boldsymbol{\nu}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1} \end{bmatrix}^T \begin{bmatrix} \mathbf{\Lambda}_{t+1} - \mathbf{\Lambda}_t \\ \boldsymbol{\nu}_{t+1} - \boldsymbol{\nu}_t \end{bmatrix} \tag{5.38}$$

The difference of the current and next iterates may be written as $\mathbf{\Lambda}_{t+1} - \mathbf{\Lambda}_t = \eta_t \tilde{\nabla}_{\mathbf{\Lambda}} \hat{\mathcal{L}}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)$ and $\boldsymbol{\nu}_{t+1} - \boldsymbol{\nu}_t = \eta_t \tilde{\nabla}_{\boldsymbol{\nu}} \hat{\mathcal{L}}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)$, which we substitute into the right hand side of (5.38), yielding

$$\mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_{t+1}, \boldsymbol{\nu}_{t+1}) - \mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) = \eta_t \begin{bmatrix} \nabla_{\mathbf{\Lambda}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1} \\ \nabla_{\boldsymbol{\nu}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1} \end{bmatrix}^T \begin{bmatrix} \tilde{\nabla}_{\mathbf{\Lambda}} \hat{\mathcal{L}}_t(\cdot,\cdot,\cdot,\cdot)_{t+1} \\ \tilde{\nabla}_{\boldsymbol{\nu}} \hat{\mathcal{L}}_t(\cdot,\cdot,\cdot,\cdot)_{t+1} \end{bmatrix} \tag{5.39}$$

Substitute the definition of the dual stochastic gradient errors $\boldsymbol{\delta}_{\mathbf{\Lambda},t}$, $\boldsymbol{\delta}_{\boldsymbol{\nu},t}$ in Assumption 13 into the right hand side of (5.39) to obtain

$$\begin{aligned} &\mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_{t+1}, \boldsymbol{\nu}_{t+1}) - \mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) \\ &= \eta_t \begin{bmatrix} \nabla_{\mathbf{\Lambda}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1} \\ \nabla_{\boldsymbol{\nu}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1} \end{bmatrix}^T \begin{bmatrix} \tilde{\nabla}_{\mathbf{\Lambda}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1} - \boldsymbol{\delta}_{\mathbf{\Lambda},t} \\ \tilde{\nabla}_{\boldsymbol{\nu}} \mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1} - \boldsymbol{\delta}_{\boldsymbol{\nu},t} \end{bmatrix} \end{aligned} \tag{5.40}$$

Applying the fact that the direction defined by the projected gradient method is an ascent direction, [26, Lemma 2.1(i)] (see Lemma 3), and computing the expectation conditional on the algorithm history $\mathcal{F}_t$ up to time t, we may write

$$\begin{aligned} &\mathbb{E}[\mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_{t+1}, \boldsymbol{\nu}_{t+1}) - \mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t) \,|\, \mathcal{F}_t] \\ &\geq \eta_t \left\| \begin{bmatrix} \tilde{\nabla}_{\mathbf{\Lambda}} \mathcal{L}_t(\cdot,\cdot,\cdot,\cdot)_{t+1} \\ \tilde{\nabla}_{\boldsymbol{\nu}} \mathcal{L}_t(\cdot,\cdot,\cdot,\cdot)_{t+1} \end{bmatrix} \right\|^2 - \eta_t^2 A(G_{\mathbf{\Lambda}} + G_{\boldsymbol{\nu}}) \,, \end{aligned} \tag{5.41}$$

where we have also applied the bias condition of the dual stochastic errors of the Lagrangian [cf. (5.26)] in Assumption 13 to the last term. We establish a martingale relationship of the projected primal and dual stochastic gradients by summing the relation in (5.35) and

with the negative of (5.41), which yields

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\mathbf{D}_{t+1},\mathbf{w}_{t+1},\boldsymbol{\Lambda}_{t+1},\boldsymbol{\nu}_{t+1})-\mathcal{L}(\mathbf{D}_t,\mathbf{w}_t,\boldsymbol{\Lambda}_t,\boldsymbol{\nu}_t)\,|\,\mathcal{F}_t] \\ \leq -\eta_t\Big(\left\|\begin{matrix}\tilde{\nabla}_{\mathbf{D}}\mathcal{L}(\mathbf{D}_t,\mathbf{w}_t,\boldsymbol{\Lambda}_t,\boldsymbol{\nu}_t)\\ \tilde{\nabla}_{\mathbf{w}}\mathcal{L}(\mathbf{D}_t,\mathbf{w}_t,\boldsymbol{\Lambda}_t,\boldsymbol{\nu}_t)\end{matrix}\right\|^2+\left\|\begin{matrix}\tilde{\nabla}_{\boldsymbol{\Lambda}}\mathcal{L}(\mathbf{D}_{t+1},\mathbf{w}_{t+1},\boldsymbol{\Lambda}_t,\boldsymbol{\nu}_t)\\ \tilde{\nabla}_{\boldsymbol{\nu}}\mathcal{L}(\mathbf{D}_{t+1},\mathbf{w}_{t+1},\boldsymbol{\Lambda}_t,\boldsymbol{\nu}_t)\end{matrix}\right\|^2\Big) \\ +\eta_t^2\Big(A\big(G_{\mathbf{D}}+G_{\mathbf{w}}+G_{\boldsymbol{\Lambda}}+G_{\boldsymbol{\nu}}\big)+\frac{\sigma^2}{2}\big(L_{\mathbf{D}}^2+L_{\mathbf{w}}^2\big)\Big)\,.\end{aligned} \tag{5.42}$$

Observe that the left hand side of (5.42) is telescopic, and hence if we sum this relation over all t we obtain a finite quantity in expectation. By applying the step-size rules stated in Assumption 12 with the fact that $\mathcal{L}$ is lower-bounded since the primal and dual domains are compact, the following holds in expectation

$$\sum_{t=1}^{\infty}\eta_t\mathbb{E}\Big[\left\|\begin{matrix}\tilde{\nabla}_{\mathbf{D}}\mathcal{L}_t(\mathbf{D}_t,\mathbf{w}_t,\boldsymbol{\Lambda}_t,\boldsymbol{\nu}_t)\\ \tilde{\nabla}_{\mathbf{w}}\mathcal{L}_t(\mathbf{D}_t,\mathbf{w}_t,\boldsymbol{\Lambda}_t,\boldsymbol{\nu}_t)\end{matrix}\right\|^2+\left\|\begin{matrix}\tilde{\nabla}_{\boldsymbol{\Lambda}}\mathcal{L}_t(\mathbf{D}_{t+1},\mathbf{w}_{t+1},\boldsymbol{\Lambda}_t,\boldsymbol{\nu}_t)\\ \tilde{\nabla}_{\boldsymbol{\nu}}\mathcal{L}_t(\mathbf{D}_{t+1},\mathbf{w}_{t+1},\boldsymbol{\Lambda}_t,\boldsymbol{\nu}_t)\end{matrix}\right\|^2\Big]<\infty\,, \tag{5.43}$$

We continue by stating a Lemma which appears as Proposition 1.2.4 in [21], which allows us to draw conclusions regarding the asymptotic properties of (5.43).

Lemma 4 *Let $\{a_t\}$ and $\{b_t\}$ be two nonnegative scalar sequences such that $\sum_{t=1}^{\infty}a_t=\infty$ and $\sum_{t=1}^{\infty}a_tb_t<\infty$. Then*

$$\liminf_{t\to\infty}b_t=0. \tag{5.44}$$

Furthermore, if $|b_{t+1}-b_t|\leq Ba_t$ for some constant $B>0$, then

$$\lim_{t\to\infty}b_t=0. \tag{5.45}$$

Observe that (5.43) satisfies the conditions of (5.44), and thus the expected limit infimum of the sequence converges to null,

$$\liminf_{t\to\infty}\mathbb{E}\Big[\left\|\begin{matrix}\tilde{\nabla}_{\mathbf{D}}\mathcal{L}_t(\mathbf{D}_t,\mathbf{w}_t,\boldsymbol{\Lambda}_t,\boldsymbol{\nu}_t)\\ \tilde{\nabla}_{\mathbf{w}}\mathcal{L}_t(\mathbf{D}_t,\mathbf{w}_t,\boldsymbol{\Lambda}_t,\boldsymbol{\nu}_t)\end{matrix}\right\|^2+\left\|\begin{matrix}\tilde{\nabla}_{\boldsymbol{\Lambda}}\mathcal{L}_t(\mathbf{D}_{t+1},\mathbf{w}_{t+1},\boldsymbol{\Lambda}_t,\boldsymbol{\nu}_t)\\ \tilde{\nabla}_{\boldsymbol{\nu}}\mathcal{L}_t(\mathbf{D}_{t+1},\mathbf{w}_{t+1},\boldsymbol{\Lambda}_t,\boldsymbol{\nu}_t)\end{matrix}\right\|^2\Big]=0 \tag{5.46}$$

Using the convergence in (5.46), we establish the whole sequence of partial Lagrangian gradients converge. Since the logic is equivalent in each block variable, it is enough to consider just the primal-dual pair $(\mathbf{D},\boldsymbol{\Lambda})$. Consider the expected absolute difference of the Lagrangian gradients evaluated at the next and current iterate [eqn. (48) of [215]], and

apply Jensen's inequality to write

$$\begin{aligned}&\left|\|\mathbb{E}[\tilde{\nabla}_{\mathbf{D}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1}]\|_F^2-\|\mathbb{E}[\tilde{\nabla}_{\mathbf{D}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t}]\|_F^2\right| \\ &\quad\leq \mathbb{E}[\|\tilde{\nabla}_{\mathbf{D}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1}+\tilde{\nabla}_{\mathbf{D}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t}\|_F\times\|\tilde{\nabla}_{\mathbf{D}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1}-\tilde{\nabla}_{\mathbf{D}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t}\|_F]\ .\end{aligned} \tag{5.47}$$

Apply the non-expansive property of the projection operator, the bound on the primal gradient in (5.25), Lipschitz continuity to right hand side of (5.47) to express this gradient difference in terms of the difference between the next and current iterate as

$$\begin{aligned}&\mathbb{E}[\|\tilde{\nabla}_{\mathbf{D}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1}+\tilde{\nabla}_{\mathbf{D}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t}\|_F\times\|\tilde{\nabla}_{\mathbf{D}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1}-\tilde{\nabla}_{\mathbf{D}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t}\|_F] \\ &\qquad\leq 2G_{\mathbf{D}}\|\nabla_{\mathbf{D}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1}-\nabla_{\mathbf{D}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t}\|_F] \\ &\qquad\leq 2G_{\mathbf{D}}L_{\mathbf{D}}\mathbb{E}[\|\mathbf{D}_{t+1}-\mathbf{D}_t\|]\ .\end{aligned} \tag{5.48}$$

Substitute $\mathbf{D}_{t+1}-\mathbf{D}_t=-\eta_t\tilde{\nabla}_{\mathbf{D}}\hat{\mathcal{L}}_t(\mathbf{D}_t,\mathbf{w}_t,\mathbf{\Lambda}_t,\boldsymbol{\nu}_t)$ into the right hand side of (5.49) and apply with the bound on the second conditional moment of the stochastic gradient stated in (5.27) of Assumption 13 to write

$$2G_{\mathbf{D}}L_{\mathbf{D}}\eta_t\mathbb{E}[\|\tilde{\nabla}_{\mathbf{D}}\hat{\mathcal{L}}_t(\mathbf{D}_t,\mathbf{w}_t,\mathbf{\Lambda}_t,\boldsymbol{\nu}_t)\|]\leq 2G_{\mathbf{D}}L_{\mathbf{D}}\sigma\eta_t\ . \tag{5.49}$$

With (5.49), the second condition of Lemma 4 is satisfied, whereby we may conclude the gradient sequence converges in expectation

$$\lim_{t\to\infty}\mathbb{E}[\|\tilde{\nabla}_{\mathbf{D}}\mathcal{L}(\mathbf{D}_t,\mathbf{w}_t,\mathbf{\Lambda}_t,\boldsymbol{\nu}_t)\|]=0\ . \tag{5.50}$$

Since the other primal block sequence $\mathbf{w}_t$ is updated in an analogous manner to that of $\mathbf{D}$, the analysis with the same logic, implying that a first order stationary condition of the Lagrangian is achieved asymptotically in expectation, i.e. $\mathbb{E}[\|\tilde{\nabla}_{\mathbf{w}}\mathcal{L}(\mathbf{D}_t,\mathbf{w}_t,\mathbf{\Lambda}_t,\boldsymbol{\nu}_t)\|]\to 0$.

We next establish that the whole dual gradient sequence with respect to $\mathbf{\Lambda}$ is converging in magnitude to null. We use the shorthand notation $\nabla_{\mathbf{\Lambda}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1}:=\nabla_{\mathbf{\Lambda}}\mathcal{L}(\mathbf{D}_{t+1},\mathbf{w}_{t+1},\mathbf{\Lambda}_{t+1},\boldsymbol{\nu}_{t+1})$ and $\nabla_{\mathbf{\Lambda}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t}:=\nabla_{\mathbf{\Lambda}}\mathcal{L}(\mathbf{D}_{t+1},\mathbf{w}_{t+1},\mathbf{\Lambda}_{t},\boldsymbol{\nu}_{t})$. Continue by considering the expected absolute difference of the Lagrangian gradients evaluated at the next and current dual iterate, and applying Jensen's inequality to write

$$\begin{aligned}&\left|\|\mathbb{E}[\tilde{\nabla}_{\mathbf{\Lambda}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1}]\|_F^2-\|\mathbb{E}[\tilde{\nabla}_{\mathbf{\Lambda}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t}]\|_F^2\right| \\ &\quad\leq \mathbb{E}[\|\tilde{\nabla}_{\mathbf{\Lambda}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1}+\tilde{\nabla}_{\mathbf{\Lambda}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t}\|_F \\ &\qquad\times\|\tilde{\nabla}_{\mathbf{\Lambda}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t+1}-\tilde{\nabla}_{\mathbf{\Lambda}}\mathcal{L}(\cdot,\cdot,\cdot,\cdot)_{t}\|_F] \\ &\quad=\mathbb{E}[\|\mathbf{C}_{\mathbf{D}}\mathbf{D}_{t+1}+\mathbf{C}_{\mathbf{D}}\mathbf{D}_t\|_F\|\mathbf{C}_{\mathbf{D}}\mathbf{D}_{t+1}-\mathbf{C}_{\mathbf{D}}\mathbf{D}_t\|_F]\end{aligned} \tag{5.51}$$

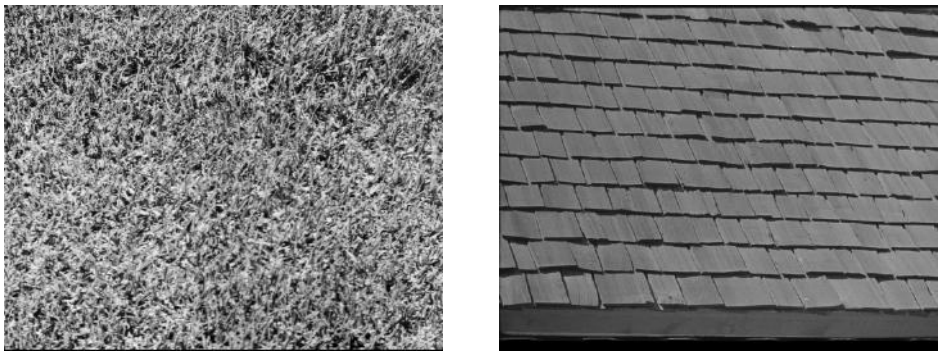


Figure 5.1: Sample images from the Brodatz texture database.

where the last equality in (5.51) follows from the computation of the dual gradient of the Lagrangian in (5.16). Now apply the triangle inequality and the compactness of the set $\mathcal{D}$ to express the right hand side of (5.51) in terms of the iterate difference, yielding

$$\begin{aligned}\mathbb{E}[\|\mathbf{C_D}\mathbf{D}_{t+1} + \mathbf{C_D}\mathbf{D}_t\|_F\|\mathbf{C_D}\mathbf{D}_{t+1} - \mathbf{C_D}\mathbf{D}_t\|_F] &\leq 2\Gamma\sqrt{Vk}\mathbb{E}[\|\mathbf{D}_{t+1} - \mathbf{D}_t\|] \,, \\ = 2\Gamma\sqrt{Vk}\eta_t\mathbb{E}[\|\tilde{\nabla}_{\mathbf{D}}\hat{\mathcal{L}}_t(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)\|] &\leq 2\Gamma\sqrt{Vk}\sigma\eta_t \,,\end{aligned} \tag{5.52}$$

where the second equality comes from the substitution $\mathbf{D}_{t+1} - \mathbf{D}_t = -\eta_t \nabla_{\mathbf{D}}\hat{\mathcal{L}}(\mathbf{D}_t, \mathbf{w}_t, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)$, and the last inequality comes from applying the bound in (5.27). As in the analysis of the primal gradient sequence, we may now apply (5.45) in Lemma 4, which implies that the expected projected dual gradient sequence converges to null in magnitude, i.e. $\lim_{t\to\infty}\mathbb{E}[\|\tilde{\nabla}_{\mathbf{\Lambda}}\mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)\|] = 0$. By noting that the analysis for $\mathbf{\Lambda}$ is analogous to that of the other dual variable $\boldsymbol{\nu}$, we may also conclude $\mathbb{E}[\|\tilde{\nabla}_{\boldsymbol{\nu}}\mathcal{L}(\mathbf{D}_{t+1}, \mathbf{w}_{t+1}, \mathbf{\Lambda}_t, \boldsymbol{\nu}_t)\|] \to 0$. ■

Theorem 4 guarantees that the block saddle point method as stated in (5.6) - (5.9) solves the problem of learning a dictionary and discriminative model over that dictionary representation of the feature space in a decentralized online manner. In particular, the algorithm asymptotically converges to a KKT point of the problem (5.2) in expectation. This implies that the primal variables converge to a local minimum of the objective, and the dual variables asymptotically enforce feasibility, i.e. the network agreement constraints are satisfied in expectation. We next turn to the practical consequences of this theorem by studying the algorithm performance on a canonical computer vision task.

5.4 Empirical Evaluation of Multi-Agent Dictionaries

Consider the task of visual pattern recognition in large scale image databases. Because the sample size is assumed to be very large, learning over the data all at once is impractical.

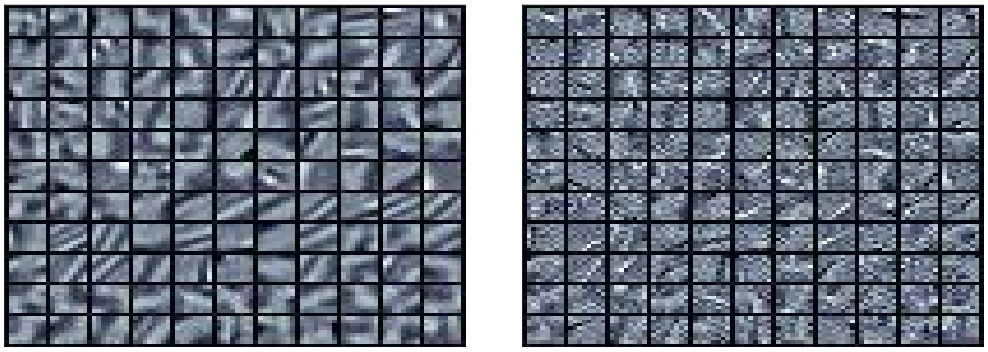


Figure 5.2: Initialized (left) and final (right) dictionary for 8-by-8 grayscale patches. These dictionaries were computed using the centralized ($V = 1$) algorithm with step-size $\eta = 0.25$.

Instead, we consider processing images a few at a time. Moreover, image processing is computationally demanding domain in which learning at a centralized location may be too slow. By leveraging a network of interconnected computing nodes and using Algorithm 4, we may effectively accelerate the rate at which such large-scale pattern recognition tasks may be solved.

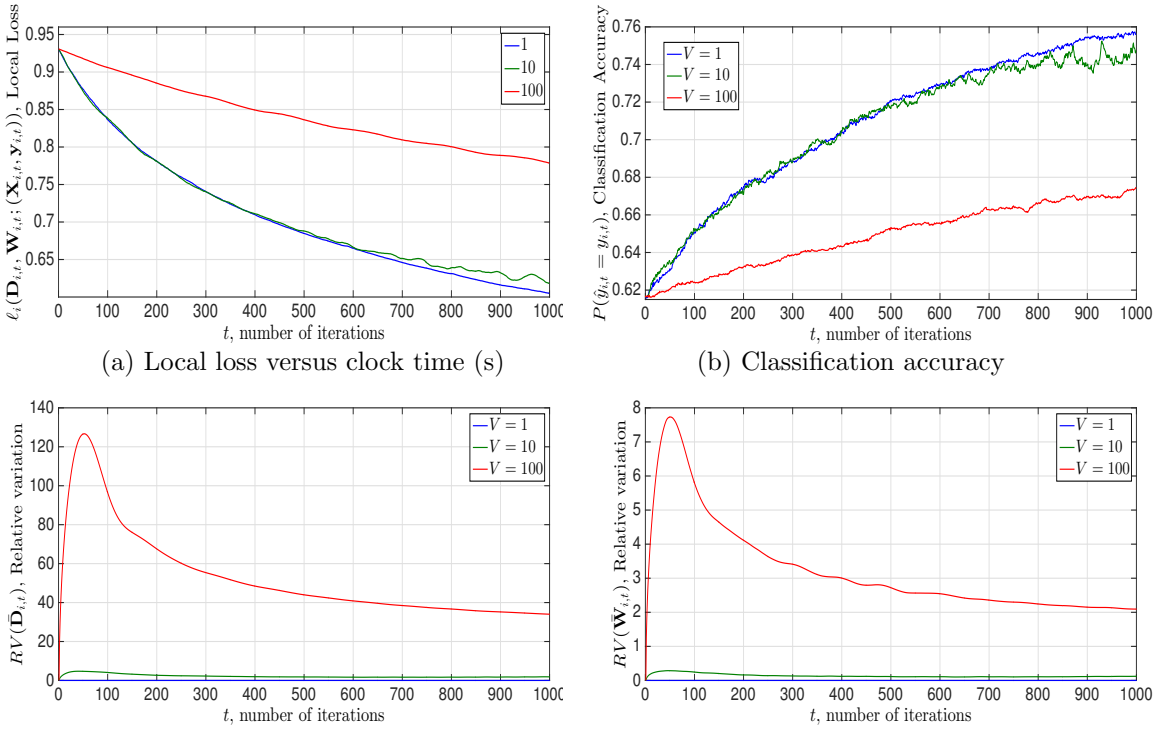
To do so, we make use of recent work on sparse representations [213] in which f in 4.10 is an elastic-net (4.5) problem using the the Euclidean distance $\tilde{\ell}(\boldsymbol{\alpha}_t, \tilde{\mathbf{D}}; \mathbf{x}_t) = \|\mathbf{x}_t - \tilde{\mathbf{D}}\boldsymbol{\alpha}_t\|^2/2$. For this case, 4.10 may be efficiently computed via least angle regression [57]. We study the performance of Algorithm 4 for a multi-class classification.

We conduct numerical experiments on the Brodatz dataset [37] for a variety of network sizes and topologies. In the case of studying the impact of network size, we also compare the algorithm performance to the centralized case, i.e. $V = 1$. Moreover, we consider the case where each agent observes training examples which are incomplete random subsets of the total class labels, yet aims to learn a classifier over all possible classes.

We restrict ourselves to $D = 4$ class labels {*grass, bark, straw, herringbone_weave*} in the Brodatz texture database, sample images of which are shown in Figure 5.1. This data subset consists of one grayscale image per texture, yielding four 512×512 images in total consisting of $1,820$ overlapping patches of size 24×24.

5.4.1 Feature Generation

Inspired by the two-dimensional texton features discussed in [107], we generate texture features $\tilde{\boldsymbol{\alpha}}^*$ as the sum of the sparse dictionary representations of sub-patches. That is, we classify patch of size 24×24 by first extracting nine non-overlapping 8-by-8 sub-patches $\mathbf{x}^{(i)}$ of each image. We vectorize (column-major order) each sub-patch, subtract off the sample mean, and divide by its norm such that $\mathbf{x}^{(i)}$ is zero-mean and has unit ℓ_2 norm. Stacking these sub-patches column wise in a matrix $\mathbf{X} = \left[\mathbf{x}^{(1)}; \cdots; \mathbf{x}^{(9)}\right]$, we compute the aggregate sparse coding $\tilde{\boldsymbol{\alpha}}^*_{i,t}$ at agent i at time t according to $\tilde{\boldsymbol{\alpha}}^*(\mathbf{X}_{i,t}, \mathbf{D}_{i,t}) = \sum_{l=1}^{9} \boldsymbol{\alpha}^*(\mathbf{D}_{i,t}; \mathbf{x}^{(l)}_{i,t})$, which implies that the local stochastic gradient of the dictionary



(a) Local loss versus clock time (s)

(b) Classification accuracy

(c) Relative variation vs. time (s)

(d) Relative variation vs. time (s)

Figure 5.3: Learning achieved by an arbitrary agent in networks of size $V = 1$ (centralized), $V = 10$, and $V = 100$ with nodes randomly connected with prob. $\rho = 0.2$. 5.3(a)-5.3(b) show $\ell_i(\mathbf{D}_{i,t},\mathbf{W}_{i,t};(\mathbf{X}_{i,t},\mathbf{y}_{i,t}))$ and $\sum_{i=1}^{V} P(\hat{\mathbf{y}}_{i,t} = \mathbf{y}_{i,t})/V$ versus clock time in seconds, both of which decline faster in smaller networks. Figures 5.3(c)-5.3(d) show that network disagreement in terms of $RV(\bar{\mathbf{D}}_{i,t})$ and $RV(\bar{\mathbf{W}}_{i,t})$ becomes more stable and declines faster with smaller V. Algorithm performance in moderate sized networks is comparable to the centralized case.

$\nabla_{\mathbf{D}_i}\tilde{h}_{i,t}(\mathbf{D}_{i,t}, \mathbf{w}_{i,t}; (\mathbf{X}_{i,t}, \mathbf{y}_{i,t}))$ is the sum of contributions $\nabla_{\mathbf{D}_i} h_{i,t}(\mathbf{D}_{i,t}, \mathbf{w}_{i,t}; (\mathbf{x}_{i,t}, \mathbf{y}_{i,t}))$ from each sub-patch.

5.4.2 Loss Function and Performance Metrics

We cast texture classification as a multi-class logistic regression problem in which agent i receives example images $\mathbf{x}_i$ and is charged with outputting a binary sequence $\mathbf{y}_i \in \{0,1\}^D$ where D is the number of classes. Each component $y_{i,c}$ of the vector $\mathbf{y}_i \in \{0,1\}^D$ is a binary indicator of whether the signal falls into class d. The local instantaneous loss ℓ_i is the λ-regularized negative log-likelihood of the probabilistic model [133], stated as

$$\ell_i(\mathbf{D}_i,\mathbf{W}_i; (\mathbf{X}_i, \mathbf{y}_i)) = \log\left(\sum_{d=1}^{D} e^{\mathbf{w}_{i,c}^T \tilde{\boldsymbol{\alpha}}_i^* + w_{i,d}^0}\right) - \sum_{d=1}^{D}\left(y_{i,c}\mathbf{w}_{i,c}^T\tilde{\boldsymbol{\alpha}}_i^* + w_{i,d}^0\right) + \xi\|\mathbf{W}_i\|_F^2 \ ,$$

where $g_d(\tilde{\boldsymbol{\alpha}}_i^*) = e^{\mathbf{w}_{i,d}^T\tilde{\boldsymbol{\alpha}}_i^* + w_{i,d}^0}$, are computed using column d, $\mathbf{w}_c$, of the weight matrix $\mathbf{W}_i \in \mathbb{R}^{(k+1)\times D}$. $w_{i,d}^0$ is a bias term.

To ensure identifiability, the last column of $\mathbf{W}_i$ is set to zero. With $\mathbf{W}_i$, the probability that $\tilde{\boldsymbol{\alpha}}_i^*$ belongs to class d is given by $g_d(\tilde{\boldsymbol{\alpha}}_i^*)/\sum_{d'} g_{d'}(\tilde{\boldsymbol{\alpha}}_i^*)$. Further, the classification is made via maximum-likelihood label assignment, i.e. $\tilde{d} = \text{argmax}_d\, g_d(\tilde{\boldsymbol{\alpha}}_i^*)/\sum_{d'} g_{d'}(\tilde{\boldsymbol{\alpha}}_i^*)$ is the only nonzero entry of $\mathbf{y}_{i,t}$.

Besides the local loss ℓ_i which we know converges to a KKT point as a consequence of Theorem 4, we also study the network average classification accuracy $\sum_{i=1}^{V} P(\hat{\mathbf{y}}_{i,t} = \mathbf{y}_{i,t})/V$ at each iteration. Here $\mathbf{y}_{i,t}$ denotes the true texture label, $\hat{\mathbf{y}}_{i,t}$ denotes the predicted label, and $P(\hat{\mathbf{y}}_{i,t} = \mathbf{y}_{i,t})$ represents the empirical classification rate on a fixed test set of size $\tilde{T} = 4096$. We also consider the relative variation of the average classifiers, stated as

$$\text{RV}(\bar{\mathbf{W}}_{i,t}) = \frac{1}{V}\sum_{j=1}^{V}\|\bar{\mathbf{W}}_{i,t} - \bar{\mathbf{W}}_{j,t}\|_F, \tag{5.53}$$

where $\bar{\mathbf{W}}_{i,t} = \sum_{s=1}^{t}\mathbf{W}_{i,s}/t$ which quantifies how far individual agents' classifiers are from consensus. We consider time averages $\bar{\mathbf{W}}_{i,t}$ instead of the plain estimates $\mathbf{W}_{i,t}$ because the latter tend to oscillate around the stationary point $\mathbf{W}^*$ and agreement between estimates of different agents is difficult to visualize.

5.4.3 Implementation Details

(i) Dictionary Size As in [12], we find that increasing the size of the dictionary led to better classifier performance on the Brodatz textures, but the relative gains with increasing k diminish beyond a threshold, motivating the selection $k = 128$. One could make k arbitrarily large, but the computational complexity of the algorithm is proportional to k. We show the initialized and final 128-element, 8-by-8 patch dictionaries in Figure 5.2.

(ii) Mini-Batching We adopt a mini-batching procedure: at each iteration, we replace the single labeled patch with a small batch of $\hat{T} = 200$ randomly-drawn labeled patches: for each patch, a label is first drawn uniformly at random from the set of all possible labels; then, the patch is selected uniformly at random from the set of all patches with that label. We then compute the dictionary and classifier gradient values for the iteration by averaging the gradient values generated by each individual patch within the mini-batch. This process reduces the variance of the local stochastic gradients, which empirically accelerates convergence.

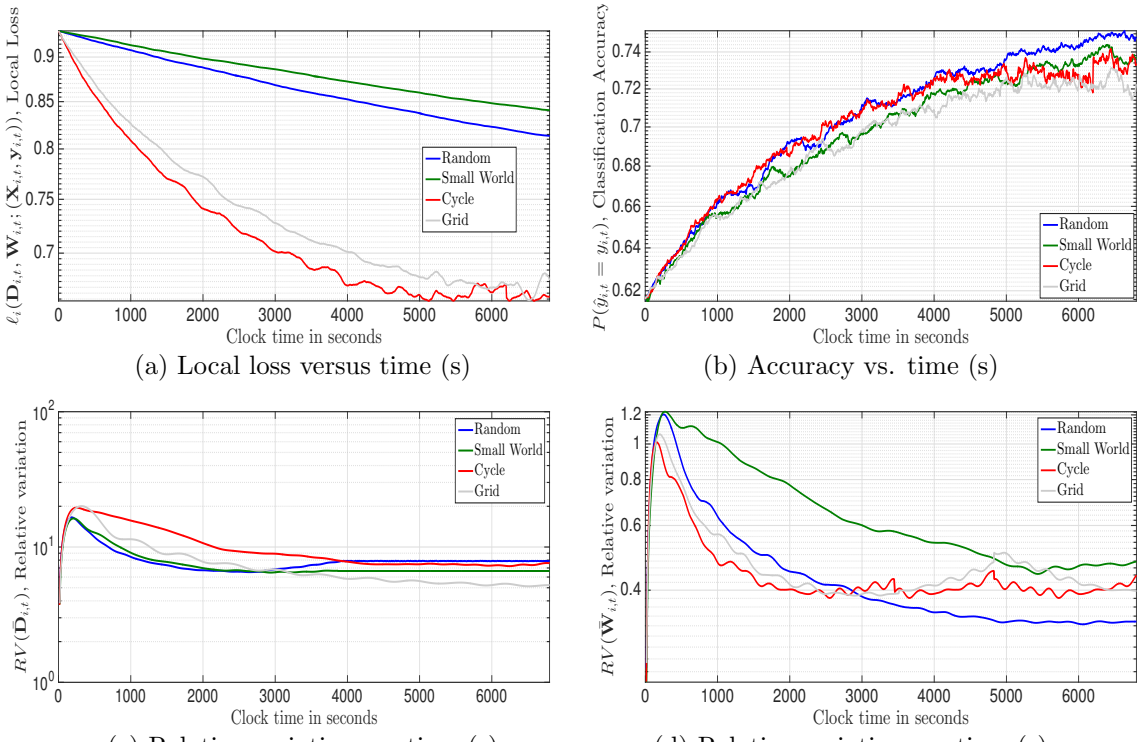
(iii) Initialization We initialize $\mathbf{D}$ using unsupervised dictionary learning for a small set of randomly-drawn initialization data [117]. We then used the labels and the dictionary representations of the data to initialize the classifier parameters $\mathbf{W}$.

(iv) Regularization and Step-size Selection Following [12], we select regularizers $\zeta_1 = 0.125$, $\zeta_2 = 0$, and $\xi = 10^{-9}$. We adopt the learning-rate selection strategy discussed in [12], i.e. a hybrid scheme $\eta_t = \min(\eta, \eta t_0/t)$ which is a constant η for the first $t_0 = T/2$ iterations, after which it attenuates. $t_0 = T/2$ has been selected via cross-validation over a small grid search.

Convergence guarantees for stochastic gradient algorithms in non-convex settings only occur in cases where a diminishing step-size mitigates the stochastic approximation error, which may not occur if the initial constant step-size is too large. We experimentally observed that values of η which avoid this behavior are smaller than effective values for the centralized version by order(s) of magnitude. Thus, when comparing D4L to its centralized counterpart, we select η that yield convergence for both settings, i.e., the smaller values appropriate for D4L. For the Brodatz dataset, we found that $\epsilon = 0.05$ led to convergence in all cases. Subsequently, we run the algorithm for a total of $\bar{T} = 20$ sample paths, and report the results in terms of the empirical mean.

5.4.4 Results on Texture Database

(i) Network Size To investigate the dependence of the convergence rate in Theorem 4 on the network size V we run Algorithm 4 for problem instances with $V = 1$ (centralized), $V = 10$, and $V = 100$ nodes. For the latter two cases, connections between nodes are random, with the probability of two nodes being connected set to $\rho = 0.2$. Because such randomly generated networks are not guaranteed to be connected, we repeatedly generate networks according to this rule and take the first which is connected. We repeat this process until we obtain a network which has a fixed connectivity ratio $\varrho = 0.2$ (the average node degree divided by the maximum node degree), which implies that the average degree for individual nodes is fixed as the network size grows. In this experiment, each agent observes training examples from all label classes. The centralized case $V = 1$ is comparable to existing state of the art supervised learning methods. For this numerical experiment, we display a given



(a) Local loss versus time (s)

(b) Accuracy vs. time (s)

(c) Relative variation vs. time (s)

(d) Relative variation vs. time (s)

Figure 5.4: D4L run on $V = 20$ node cycle, grid, random and small world networks, where edges are generated randomly between agents with probability $\rho = 0.2$ in the latter two. Figure 5.4(a)-5.4(b) show $\ell_i(\mathbf{D}_{i,t},\mathbf{W}_{i,t};(\mathbf{X}_{i,t},\mathbf{y}_{i,t}))$ and $\sum_{i=1}^{V} P(\hat{\mathbf{y}}_{i,t} = \mathbf{y}_{i,t})/V$, respectively, over clock time (s) for an arbitrarily chosen agent $i \in V$. Learning slows and numerical oscillations become more prevalent in networks with random connectivity patterns. Structured deterministic networks such as grids and cycles have larger diameter than small world and random networks, yet achieve superior performance. Figures 5.4(c)-5.4(d) shows that the agents reach consensus in terms of $\text{RV}(\bar{\mathbf{D}}_{i,t})$ and $\text{RV}(\bar{\mathbf{W}}_{i,t})$ at comparable rates across the different network topologies.

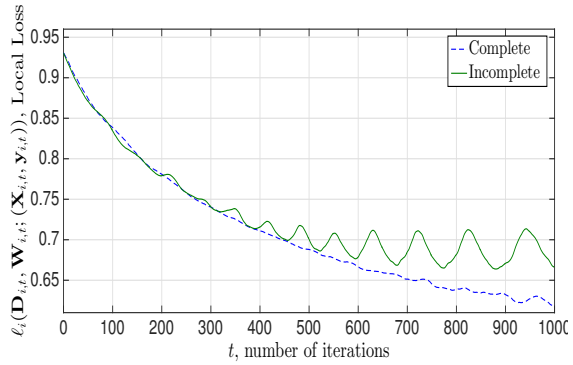
performance metric as compared with clock time in seconds per node.

Figure 5.3 shows the empirical result for a randomly selected agent in the network. In Figure 5.3(a), we show $\ell_i(\mathbf{D}_{i,t},\mathbf{W}_{i,t};(\mathbf{X}_{i,t},\mathbf{y}_{i,t}))$ over clock time in seconds. Observe that as V increases, the log-likelihood $\ell_i(\mathbf{D}_{i,t},\mathbf{W}_{i,t};(\mathbf{X}_{i,t},\mathbf{y}_{i,t}))$ declines at comparable rates for networks of moderate size, yet it is significantly slower for the $V = 100$ node network. To be specific, both the centralized and $V = 10$ node network achieve $\ell_i(\mathbf{D}_{i,t},\mathbf{W}_{i,t};(\mathbf{X}_{i,t},\mathbf{y}_{i,t})) \leq 0.65$ after 7000 seconds, while the $V = 100$ node network remains at 0.77 over its run. This performance discrepancy is corroborated in Figure 5.3(b), which shows the classification accuracy on a fixed test set over clock time (s): for $V = 1$, $V = 10$, and $V = 100$, we respectively achieve accuracy near 76%, 75%, and 67% by 7000 seconds.

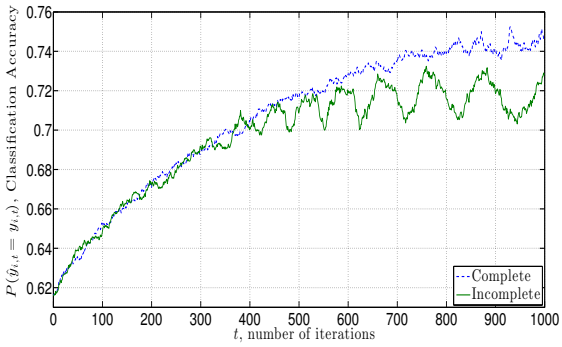
In Figure 5.3(c) we investigate how far the agents are from consensus as measured by $\text{RV}(\bar{\mathbf{D}}_{i,t})$ over clock time (s). Trivially $\text{RV}(\bar{\mathbf{D}}_{i,t}) = 0$ for the centralized case, but for the $V = 10$ and $V = 100$ node networks the algorithm achieves $\text{RV}(\bar{\mathbf{D}}_{i,t}) \leq 1.9$ by 3000 seconds and $\text{RV}(\bar{\mathbf{D}}_{i,t}) \leq 35$ by 7000 seconds, respectively. Thus in larger networks information diffuses more slowly. Moreover, the agreement constraints are more difficult to satisfy and delay the convergence to stationarity. This difference in consensus may also be observed in Figure 5.3(d), which shows $\text{RV}(\bar{\mathbf{W}}_{i,t})$ over clock time (s). We observe an order of magnitude difference in the relative variation between the $V = 10$ and $V = 100$ node networks for both the dictionary and model parameters.

(ii) Network Topology and Diameter We study the dependence of the convergence rate of Algorithm 4 in Theorem 4 on the network topology by fixing the network size to $V = 20$ and running (5.6) - (5.9) over random graphs, small world graphs, cycles, and grids. In the first two, the probability that node pairs are randomly connected is fixed at $\rho = 0.2$. Again, we repeatedly generate these random networks and select the first realization which is connected for our simulation. The latter two are deterministically generated. A cycle is a closed directed chain of nodes. Grids are formed by taking the two-dimensional integer lattice of size $\sqrt{V} \times \sqrt{V}$, with $\sqrt{V}$ rounded to the nearest integer. Connections are drawn between adjacent nodes in the lattice as well as between remainder nodes at the boundary. Cycles, grids and random networks have progressively larger number of connections per node and smaller diameter. Random networks have small degree and small diameter; see [208, 211].

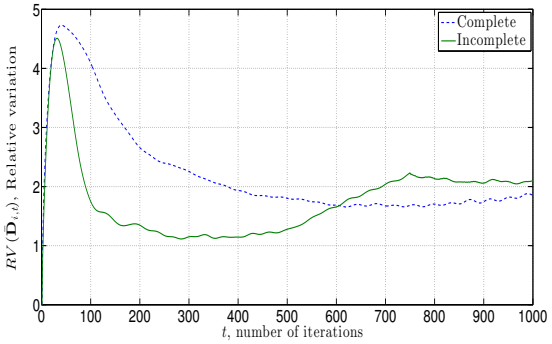
We present the results of this experiment in Figure 5.4 relative to the clock time in seconds per node. In Figure 5.4(a), we plot $\ell_i(\mathbf{D}_{i,t},\mathbf{W}_{i,t};(\mathbf{X}_{i,t},\mathbf{y}_{i,t}))$ over clock time. Observe that the rate at which $\ell_i(\mathbf{D}_{i,t},\mathbf{W}_{i,t};(\mathbf{X}_{i,t},\mathbf{y}_{i,t}))$ decreases is faster in the grid and cycle networks as compared with the random and small world networks, indicating that structured deterministic networks are an easier setting for finding good signal representations in a decentralized manner. This point is supported by the classification results in Figure 5.4(b). Observe that the algorithm achieves an accuracy near 75% for cycle and grid networks as



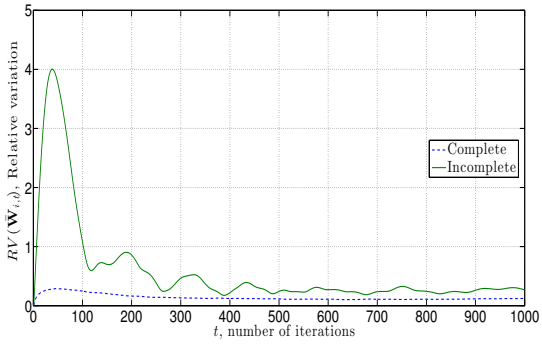
(a) Local loss vs. iteration t



(b) Classification accuracy vs. iter. t



(c) Relative variation vs. iter. t



(d) Relative variation vs. iter. t

Figure 5.5: Algorithm 4 run on a $V = 10$ node random networks, where edges are generated randomly between agents with probability $\rho = 0.2$. "Incomplete" refers to the case where each agent observes training examples that comprise a random incomplete subset of the total data labels. Figure 5.4(a)-5.4(b) show $\ell_i(\mathbf{D}_{i,t},\mathbf{W}_{i,t};(\mathbf{X}_{i,t},\mathbf{y}_{i,t}))$ and $\sum_{i=1}^{V} P(\hat{\mathbf{y}}_{i,t} = \mathbf{y}_{i,t})/V$, respectively, over iteration t for an arbitrarily chosen agent $j \in V$, as compared with the case where this agent observes training examples from all classes. Observe that learning is still achieved for this more challenging context, yet the algorithm exhibits increased oscillatory behavior and decreased accuracy for the incomplete case. Figures 5.4(c)-5.4(d) shows that the agents reach consensus in terms of $\mathrm{RV}(\bar{\mathbf{D}}_{i,t})$ and $\mathrm{RV}(\bar{\mathbf{W}}_{i,t})$ at comparable rates for complete cases and incomplete cases, albeit with more oscillations in the latter. Moreover, the algorithm still converges despite the instability in $\mathrm{RV}(\bar{\mathbf{D}}_{i,t})$.

compared to 72% and 73% for random and small world networks, respectively.

We study effect of network topology on the algorithm's convergence to consensus in 5.4(c), where we plot RV($\bar{\mathbf{D}}_{i,t}$) over clock time. Observe that the initial burn-in period is comparable across the different networks except for the cycle. Moreover, this difference in convergence to consensus as measured by the relative variation is corroborated in 5.4(d), where we plot RV($\bar{\mathbf{W}}_{i,t}$) versus clock time. Observe that the cycle yields the slowest convergence rate, yet is more stable than the small world and random networks. Surprisingly, the grid network has superior convergence to consensus both in terms of dictionary and model parameters.

(iii) Complete vs. Incomplete Sampling We study the performance of D4L in the setting of *incomplete sampling*, which refers to the case that each agent in the network observes only training examples from a fixed random subset of the total number of class labels, yet is charged with the task of classifying all classes. Each node receives training examples from only a subset of classes, which is chosen using sampling with replacement from the set of classes. We run the algorithm on a $V = 10$ node random network with connection probability $\rho = 0.2$.

We display these results in Figure 5.5 juxtaposed with the complete sampling setting, both for a randomly selected agent in the network. In Figure 5.5(a), we plot $\ell_i(\mathbf{D}_{i,t},\mathbf{W}_{i,t};(\mathbf{X}_{i,t},\mathbf{y}_{i,t}))$ over iteration t. During an initial burn-in period of $t \leq 300$ the local losses decline at comparable rates, after which the algorithm experiences greater numerical oscillations and its convergence rate slows for the incomplete case. These oscillations and slower convergence are also present in the plot of classification performance versus iteration t in Figure 5.5(b). By $t = 350$ both cases achieve an accuracy of 70%; however, the incomplete sampling oscillates around this benchmark whereas the complete case continues to improve. Increased oscillations occur when agents observe only training examples from a subset of the total number of classes.

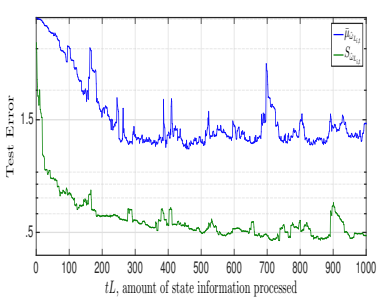
We study how the incomplete sampling, or the implicit partition of the feature space across the network, impacts the network disagreement in Figure 5.5(c), where we plot RV($\bar{\mathbf{D}}_{i,t}$) over time t. Observe during an initial burn-in period of $t \leq 100$ that the relative variation is smaller in the case of incomplete sampling for the dictionary, yet by $t \geq 600$ the complete sampling case more closes enforces consensus. Moreover, RV($\bar{\mathbf{D}}_{i,t}$) slowly climbs despite the convergence of the algorithm to a neighborhood of a stationary point. We observe a improved constraint slack convergence in the plot of RV($\bar{\mathbf{W}}_{i,t}$) over time t in Figure 5.5(d), i.e. for $t \geq$, RV($\bar{\mathbf{W}}_{i,t}$) $\leq 5 \times 10^{-1}$ for the incomplete sampling case. This suggests that the empirical effect of non-convexity is predominantly limited to the dictionary learning procedure.

(a) An iRobot Packbot comparable to our platform.

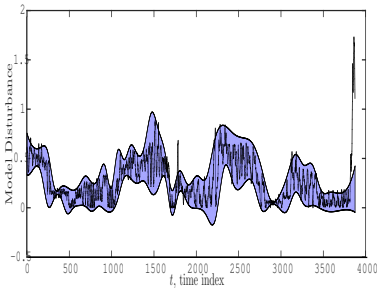
(b) Trajectory overlay of visual observations.

(c) Extracted image from grass trajectory.

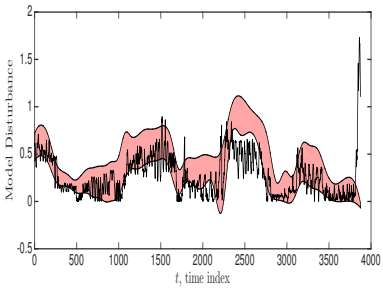
Figure 5.6: An iRobot Packbot comparable to our experimental platform in Figure 5.6(a). In Figure 5.6(b), we display the hand-generated flat-ground model used to determine the ground patch (patch example in Figure 5.6(c)) the vehicle crosses associated with its perceptive capability, and append the associated image to the state information to build feature vectors.



(a) Test error vs. time tL



(b) Actual model disturbance.



(c) Predicted Gaussian envelope.

Figure 5.7: Performance of Algorithm 4 on an $V = 10$ network of robots over a trajectory containing grass and pavement for a randomly chosen robot. The test set estimation error (Fig. 5.7(a)) demonstrates that the algorithm may successfully predict the model disturbance over a time window of size $L = 49$. First and second-order statistics of model disturbance empirical distribution across pavement and grass are given for an actual sample trajectory in Fig. 5.7(b) within a Gaussian envelope for visualization purposes, where the standard deviation is computed using the actual disturbance experienced by the platform over this run. Figure 5.7(c) shows the predicted disturbance envelope using the estimates given by the regressors learned using Algorithm 4. Observe that D4L yields an effective tool for online prediction of unexpected maneuvers experienced by the network.

5.5 Collaborative Robotic Network Experiments

Consider a team of mobile robots deployed in an unknown environment that is charged with a simple task such as path-planning or exploration. In order to execute this task successfully, each robot must augment its pre-specified controller to account for unexpected environmental effects. These unexpected effects, or *model disturbance* [92, 147], may be adaptively learned via recursive averaging or Kalman filtering in order to drive a robust control block [162,174]. Such approaches suffer from sensitivity to observed data and memory windows. An alternative approach, based on terrain classification or adaptive statistical estimation, tailors the robots' planning to a particular setting it autonomously identifies online through supervision provided by sensory feedback [4, 106].

Hence, in this section, our goal is to develop a real-time prediction scheme on a robotic network such that the steering mistakes of one robot may be avoided by another. The purpose of learning these unexpected maneuvers is to incorporate this information into a robust closed-loop control framework. As a first step towards this objective, we turn to demonstrating that the method in Algorithm 4 in Section 5.2 allows a robotic network to successfully predict model disturbance, a statistical measure of steering mistakes, such that the steering errors of one robot may be avoided by another, when operating in a variety of terrains.

We collected data on an iRobot Packbot [218], depicted in Fig. 5.6(a), a ground platform equipped with a skid-steer tracked drive system with on-board computation. The platform weighs 18 kg, is capable of 2 m/s speeds, and is equipped with a quad-core Intel i7 computing payload, a Microstrain 3DM-GX3 inertial measurement unit (IMU), and an Allied Vision Manta G-235 1/1.2" Color CMOS Camera with a 4.5 mm lens. Images of size 1936-by-1216 pixels were collected at 10 Hz. We use a joystick to drive sample robot trajectories on both grass and pavement, during which we record color images and also the commanded linear and angular velocities. Further, we use the IMU to measure the *actual* angular velocity experienced by our platform. This allows us to simulate an $V = 10$ robotic cycle network.

Feature Space We construct feature vectors $\mathbf{x}_{i,t}$ consisting of these collected images, denoted as $\mathbf{z}_{i,t}$, as well as commanded angular velocities $\boldsymbol{\omega}_{i,t}$ over the interval of time in which robot i drove over the selected ground patch.

Ground patches $\mathbf{z}_{i,t}$ are obtained by dividing the traversed path into a fixed number of rectangular regions of size comparable to the Packbot platform. Using a flat-ground model, we determine the polygonal location of these patches in the recorded images (see Fig. 5.6(b)), and use this information to compute a single, rectified, 64-by-64 color image corresponding to each patch, an example of which is shown in Figure 5.6(c). In addition to the computed patch features, we also determined the specific time interval during which

the robot drove over each of the above-mentioned ground patches. To enforce consistency of dimension among the $\mathbf{x}_{i,t}$, we truncated these intervals to the first $\hat{T} = 49$ samples.

For each patch, we compute the following features: the mean, variance, skewness and kurtosis of the color values in each of the red, green, and blue, and Lab color channels of the on-board camera; the texton histogram of the image, obtained using the method described in [107]. We initialize each $\mathbf{D}_{i,t}$ using a 512-element texton dictionary via the Brodatz texture database [37].

Target variable Over these same 49-sample, truncated time intervals we obtain the *model disturbance*, which tracks the difference between true measured platform behavior and that which is generated from our commanded control inputs. We compute estimates of the disturbance mean and standard deviation with mini-batch size $\hat{T} = 49$ as

$$\bar{\mu}_{\hat{\boldsymbol{\omega}}_{L_{i,t}}} = \frac{1}{\hat{T}} \sum_{u \in [\hat{T}(t-1)+1, t\hat{T}]} \tilde{\boldsymbol{\omega}}_{i,u} - \boldsymbol{\omega}_{i,u} \ , \tag{5.54}$$

$$S^2_{\hat{\boldsymbol{\omega}}_{L_{i,t}}} = \frac{1}{\hat{T}-1} \sum_{k \in [\hat{T}(t-1)+1, t\hat{T}]} \left(\tilde{\boldsymbol{\omega}}_{i,u} - \boldsymbol{\omega}_{i,u} - \bar{\mu}_{\hat{\boldsymbol{\omega}}_{L_{i,t}}} \right)^2 \ , \tag{5.55}$$

where $\tilde{\boldsymbol{\omega}}_{i,t}$ denotes the true measured angular velocity experienced by the platform as measured by the IMU. For future reference, we denote the feature concatenation over the time window $u \in [\hat{T}(t-1)+1, t\hat{T}]$ as $\mathbf{x}_{\hat{T}_{i,t}}$ for index t, robot i.

The problem is supervised since the feature vectors $\mathbf{z}_{i,k}$ and control information $\boldsymbol{\omega}_{i,k}$ are provided sequentially to the platform. The target variables which are obtained using on-board sensory information is the first and second order statistics of the model disturbance $\boldsymbol{\omega}_{i,k} - \tilde{\boldsymbol{\omega}}_{i,k}$. Observe that there are random pairs of this form for each platform, and yet individuals in the network would like to predict their disturbance based on the experience of all agents in the network, thus incentivizing decentralized collaboration.

To estimate the disturbance statistics (5.54) and (5.55), each robot computes linear regression coefficients, $\mathbf{w}_{i,t}$ and $\mathbf{v}_{i,t}$, that map sparse codes $\alpha^*(\mathbf{D}_{i,t}; \mathbf{x}_{L_{i,t}})$ of concatenated $\mathbf{x}_u$ for $u \in [\hat{T}(t-1)+1, t\hat{T}]$ to the estimates of the disturbance mean and variance given in (5.54) and (5.55), respectively. Thus the local instantaneous losses defined by regressors $\mathbf{w}_{i,t}$ and $\mathbf{v}_{i,t}$ are $\|\mathbf{w}_{i,t}^T \boldsymbol{\alpha}^*(\mathbf{D}_{i,t}; \mathbf{x}_{\hat{T}_{i,t}}) - \bar{\mu}_{\hat{\boldsymbol{\omega}}_{L_{i,t}}}\|^2$ and $\|\mathbf{v}_{i,t}^T \boldsymbol{\alpha}^*(\mathbf{D}_{i,t}; \mathbf{x}_{\hat{T}i,t}) - S_{\hat{\boldsymbol{\omega}}_{L_{i,t}}}\|^2$ – the actual estimators are given as $\mathbf{w}_{i,t}^T \boldsymbol{\alpha}^*(\mathbf{D}_{i,t}; \mathbf{x}_{\hat{T}i,t})$ and $\mathbf{v}_{i,t}^T \boldsymbol{\alpha}^*(\mathbf{D}_{i,t}; \mathbf{x}_{\hat{T}i,t})$. Due to computational considerations, we use a common dictionary for both mean and variance prediction.

We implement Algorithm 4 on an $V = 10$ cycle network of robots to compute linear regression coefficients, $\mathbf{w}_{i,t}$ and $\mathbf{v}_{i,t}$ in order to estimate (5.54) and (5.55) for the next time-slot of size $\hat{T}$. We select the following parameters: sparsity dimension $k = 64$, constant step-size $\eta_k = \eta = 0.05$, and 4.10 as the elastic-net with regularization parameters $\zeta_1 = 0.125$, and $\zeta_2 = 10^{-3}$.

In Fig. 5.7 we display the results of the algorithm performance for a randomly chosen robot $i \in [V]$. The estimates $\mathbf{w}_{i,t}^T \alpha^*(\mathbf{D}_{i,t}; \mathbf{x}_{L_{i,t}})$ and $\mathbf{v}_{i,t}^T \alpha^*(\mathbf{D}_{i,t}; \mathbf{x}_{L_{i,t}})$, respectively, are used to estimate the first and second-order statistics of the angular velocity disturbance in (5.54) and (5.55). In Fig. 5.7(a) we plot the Euclidean error of the predicted statistical estimators as compared with the actual sample mean and standard deviations on a hold-out test set. We observe that the both the sample mean and sample standard deviation predictors exhibit convergent behavior.

We further evaluate the performance of Algorithm 4 for predicting the disturbance statistics around a specific collection of robotic trajectories which traverse both pavement and grass. Fig. 5.7(b) shows the actual disturbance data experienced by the platform for the trajectory of one robot driving over a specific trajectory of grass and pavement, overlaid with a range of two sample standard deviations $S_{\hat{\boldsymbol{\omega}}_{L_{i,t}}}$ [cf. (5.55)] of the sample means $\bar{\mu}_{\hat{\boldsymbol{\omega}}_L}$ [cf. (5.54)] . In Fig. 5.7(c), we show the same disturbance data overlaid with the decentralized estimation scheme given by D4L for an arbitrarily chosen node $i \in [V]$. The prediction using D4L (the red envelope in Figure 5.7(c)) closely matches the actual disturbance data which is given by the bold black line.

5.6 Distributed Dictionaries Limited by Non-convexity

This work extends the discriminative dictionary learning of [12] (Chapter 4) to networked settings. To do so, we consider cases where losses are node-separable and introduced agreement constraints, yielding a decentralized stochastic non-convex program. By considering the Lagrangian relaxation of an agreement-constrained system, we develop a block variant of the Arrow-Hurwicz saddle point method to solve it. Moreover, we establish the convergence of the algorithm to a KKT point of the problem in expectation.

Experiments on a texture classification problem demonstrated comparable classifier performance between the centralized and decentralized settings, and illustrated the convergence rate dependence on the network. Moreover, the proposed method allows multi-agent systems to learn over a new class of pattern recognition problems. In doing so, networks of interconnected computing servers may collaboratively solve such problems at an accelerated rate as compared with centralized methods. We additionally applied this method to a mobile robotic network deployed in an unknown domain charged with the task of collaboratively analyzing the navigability of distinct paths traversed by each robot, such that the unexpected maneuvers made by one robot may be predicted by another.

In summary, dictionary methods achieve superior performance relative to generalized linear models on practical problems. However, the non-convexity that defines their training is a serious drawback, and requires unusual technical conditions in order to establish stability (Assumption 13) which may well fail to translate into practice. We also note that the

duality gap in non-convex settings is not null. Furthermore, our empirical experiments on challenging texture classification tasks highlight the instabilities caused by non-convexity – in particular, impractically small learning rates are required for convergence (Section 5.4.4). This leads us to shift focus in subsequent chapters to an alternative way to go beyond nonlinear statistical models that preserves convexity, by conducting optimization directly in a function space rather than over specially constructed Euclidean spaces.

Part III

Reproducing Kernels and Nonparametric Estimation

Chapter 6

Memory-Efficient Kernel Methods

In this chapter, we address a more general selection of the space of estimators $\mathscr{F}$ than the approaches in Parts I and II of this thesis. In particular, we now shift focus to the case where $\mathscr{F} = \mathcal{H}$ is a reproducing kernel Hilbert space. Reproducing kernel Hilbert spaces (RKHS) provide the ability to approximate functions using nonparametric functional representations. Although the structure of the space is determined by the choice of kernel, the set of functions that can be represented is still sufficiently rich so as to permit close approximation of a large class of functions. This resulting expressive power makes RKHS an appealing choice in many learning problems where we want to estimate an unknown function that is specified as optimal with respect to some empirical risk. When learning these optimal function representations in a RKHS, the representer theorem is used to transform the search over functions into a search over parameters, where the number of parameters grows with each new observation that is processed [144, 212]. This growth is what endows the representation with expressive power. However, this growth also results in function descriptions that are as complex as the number of processed observations, and, more importantly, in training algorithms that exhibit a cost per iteration that grows with each new iterate [89, 150]. The resulting unmanageable training cost renders RKHS learning approaches inefficient for large data sets and outright inapplicable in streaming applications. This is a well-known limitation which has motivated the development of several heuristics to reduce the growth in complexity. These heuristics typically result in suboptimal functional approximations [160].

This paper proposes a new technique for learning nonparametric function approximations in a RKHS that respects optimality and ameliorates the complexity issues described above. We accomplish this by: (i) shifting the goal from that of finding an approximation that is optimal to that of finding an approximation that is optimal within a class of parsimonious (sparse) kernel representations; (ii) designing a training method that follows a trajectory of intermediate representations that are also parsimonious. The proposed technique, *parsimonious online learning with kernels* (POLK), provides a controllable tradeoff

between complexity and optimality and we provide theoretical guarantees that neither factor becomes untenable.

Formally, we propose solving expected risk minimization problems, where the goal is to learn a regressor that minimizes a loss function quantifying the merit of a statistical model averaged over a data set. We focus on the case when the number of training examples, N, is either very large, or the training examples arrive sequentially. Further, we assume that these input-output examples, $(\mathbf{x}_n, \mathbf{y}_n)$, are i.i.d. realizations drawn from a stationary joint distribution over the random pair $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$. This problem class is popular in many fields and particularly ubiquitous in text [165], image [119], and genomic [188] processing. Here, we consider finding regressors that are not vector valued parameters, but rather functions $f \in \mathcal{H}$ in a hypothesized function class $\mathcal{H}$. This function estimation task allows one to learn nonlinear statistical models and is known to yield better results in applications where linearity of a given statistical model is overly restrictive such as computer vision and object recognition [109, 131]. The adequacy of the regressor function f is evaluated by the convex loss function $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that quantifies the merit of the estimator $f(\mathbf{x})$ evaluated at feature vector $\mathbf{x}$. This loss is averaged over all possible training examples to define the statistical loss $L(f) := \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), y)]$, which we combine with a Tikhonov regularizer to construct the regularized loss $R(f) := \operatorname{argmin}_{f \in \mathcal{H}} L(f) + (\lambda/2)\|f\|^2_{\mathcal{H}}$ [62, 178]. We then define the optimal function as

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}} R(f) := \operatorname*{argmin}_{f \in \mathcal{H}} \mathbb{E}_{\mathbf{x},\mathbf{y}} \Big[\ell(f(\mathbf{x}), y) \Big] + \frac{\lambda}{2} \|f\|^2_{\mathcal{H}} \tag{6.1}$$

The optimization problem in (6.1) is intractable in general. However, when $\mathcal{H}$ is equipped with a *reproducing kernel* $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, a nonparametric function estimation problem of the form (6.1) may be reduced to a parametric form via the representer theorem [144, 212]. This theorem states that the optimal argument of (6.1) is in the span of kernel functions that are centered at points in the given training data set, and it reduces the problem to that of determining the N coefficients of the resulting linear combination of kernels (Section 6.1). This results in a function description that is data driven and flexible, alas very complex. As we consider problems with larger training sets, the representation of f requires a growing number of kernels [144]. In the case of streaming applications this number would grow unbounded and the kernel matrix as well as the coefficient vector would grow to infinite dimension and an infinite amount of memory would be required to represent f. It is therefore customary to reduce this complexity by forgetting training points or otherwise requiring that f^* admit a parsimonious representation in terms of a sparse subset of kernels. This overcomes the difficulties associated with a representation of unmanageable complexity but a steeper difficulty is the *determination* of this optimal parsimonious representation as we

explain in the following section

To understand the challenge in determining optimal parsimonious representations, recall that kernel optimization methods borrow techniques from vector valued (i.e., without the use of kernels) stochastic optimization in the sense that they seek to optimize (6.1) by replacing the descent direction of the objective with that of a stochastic estimate [31, 161]. Stochastic optimization is well understood in vector valued problems to the extent that recent efforts are concerned with improving convergence properties through the use of variance reduction [49, 83, 168], or stochastic approximations of Newton steps [29, 128, 129, 170]. Stochastic optimization in kernel spaces, however, exhibits two peculiarities that make it more challenging:

1. The implementation of stochastic methods for expected risk minimization in a RKHS requires storage of kernel matrices and weight vectors that together are proportional to the iteration index. This is true even if we require that the solution f^* admit a sparse representation because, while it may be true that the asymptotic solution admits a sparse representation, the intermediate iterates are not necessarily sparse; see, e.g., [89].

2. The problem in (1) makes it necessary to use sparse approximations of descent directions. However, these sparse approximates are not guaranteed to be valid stochastic descent directions. Consequently, there are no guarantees that a path of sparse approximation learns the optimal sparse approximation.

Issue (1) is a key point of departure between kernel stochastic optimization and its vector valued counterpart. It implies that redefining f^* to encourage sparsity may make it easier to work with the RKHS representation after it has been learnt. However, the stochastic gradients that need to be computed to find such representation have a complexity that grows with the order of the iteration index [89]. Works on stochastic optimization in a RKHS have variously ignored the intractable growth of the parametric representation of $f \in \mathcal{H}$ [53, 112, 150, 221], or have augmented the learned function to limit the memory issues associated with kernelization using online sparsification procedures. These approaches focus on limiting the growth of the kernel dictionary through the use of forgetting factors [89], random dropping [225], and compressive sensing techniques [59, 73, 160]. These approaches overcome Issue (1) but they do so at the cost of dropping optimality [cf. Issue (2)]. This is because these sparsification techniques introduce a bias in the stochastic gradient which nullifies convergence guarantees.

Past works that have considered *supervised* sparsification (addressing issues (1)-(2)) have only been developed for special cases such as online support vector machines (SVM) [207], off-line logistic regression [227], and off-line SVM [81]. The works perhaps most

similar to ours, but developed only for SVM [207]) fixes the number of kernel dictionary elements, or the model order, in advance rather tuning the kernel dictionary to guarantee stochastic descent, i.e. determining which kernel dictionary elements are most important for representing f^*. Further, the analysis of the resulting bias induced by sparsification requires overly restrictive assumptions and is conducted in terms of time-average objective sub-optimality, a looser criterion than almost sure convergence. For specialized classes of loss functions, the bias of the descent direction induced by unsupervised sparsification techniques using random sub-sampling does not prevent the derivation of bounds on the time-average sub-optimality (regret) [225]; however, this analysis omits important cases such as support vector machines and kernel ridge regression.

In this work, we build upon past works which combine functional generalizations of first-order stochastic optimization methods operating in tandem with supervised sparsification. In particular, descending along the gradient of the objective in (6.1) is intractable when the sample size N is not necessarily finite, and thus stochastic methods are necessary. In Section 6.2, we build upon [89] in deriving the generalization of stochastic gradient method called functional SGD (Section 6.2.1). The complexity of this online functional iterative optimization is proportional to the iteration index, a complicating factor of kernel methods which is untenable for streaming settings.

Thus, we project the FSGD iterates onto sparse subspaces which are constructed from the span of a small number of kernel dictionary elements (Section 7.2.2). To find these sparse subspaces of the RKHS, we make use of greedy sampling methods based on matching pursuit [148]. The use of this technique is motivated by: (i) The fact that kernel matrices induced by arbitrary data streams will not, in general, satisfy requisite conditions for methods that enforce sparsity through convex relaxation [40]; (ii) That having function iterates that exhibit small model order is of greater importance than exact recovery since SGD iterates are not the goal signal but just a noisy stepping stone to the optimal f^*. Therefore, we construct these instantaneous sparse subspaces by making use of kernel orthogonal matching pursuit [203], a greedy search routine which, given a function and an approximation budget ϵ, returns its a sparse approximation and guarantees its output to be in a specific Hilbert-norm neighborhood of its function input.

To guarantee stochastic descent, we tie the size of the error neighborhood induced by sparse projections to the magnitude of the functional stochastic gradient and other problem parameters, thereby keeping only those kernel dictionary elements necessary for convergence (Section 6.3). The result is that we are able to conduct stochastic gradient descent using only sparse projections of the stochastic gradient, maintaining a convergence path of moderate complexity towards the optimal f^* (6.1). When the data and target domains ($\mathcal{X}$ and $\mathcal{Y}$, respectively) are compact, for a certain approximation budget depending on the

stochastic gradient algorithm step-size, we show that the sparse stochastically projected FSGD sequence still converges almost surely to the optimum of (6.5) under both attenuating and constant learning rate schemes. Moreover, the model order of this sequence remains finite for a given choice of constant step-size and approximation budget, and is, in the worst-case, comparable to the covering number of the data domain [149, 226].

In Section 6.4 we present numerical results on synthetic and empirical data for large-scale kernelized supervised learning tasks. We observe stable convergence behavior of POLK comparable to vector-valued first-order stochastic methods in terms of objective function evaluation, punctuated by a state of the art trade-off between test set error and number of samples processed. Further, the proposed method reduces the complexity of training kernel regressors by orders of magnitude. In Section 6.5 we discuss our main findings. In particular, we suggest that there is a path forward for kernel methods as an alternative to neural networks that provides a more interpretable mechanism for inference with nonlinear statistical models and that one may achieve high generalization capability without losing convexity, an essential component for efficient training.

6.1 Statistical Optimization in Reproducing Kernel Hilbert Spaces

Supervised learning is often formulated as an optimization problem that computes a set of parameters $\theta \in \Theta$ to minimize the average of a loss function $l : \Theta \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ for training examples $(\mathbf{x}_n, \mathbf{y}_n) \in \mathcal{X} \times \mathcal{Y}$. When the number of training examples N is finite, this goal is referred to as *empirical risk minimization* [191], and may be solved using batch optimization techniques. The optimal $\boldsymbol{\theta}$ is the one that minimizes the regularized average loss, $\tilde{R}(\theta; \{\mathbf{x}_n, y_n\}_{n=1}^N) = \frac{1}{N}\sum_{n=1}^N l(\boldsymbol{\theta}; (\mathbf{x}_n, y_n))$, over the set of training data $\mathcal{S} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, i.e.,

$$\theta^* = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}}\, \tilde{R}(\boldsymbol{\theta}; \mathcal{S}) = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}}\, \frac{1}{N}\sum_{n=1}^{N} l(\boldsymbol{\theta}; \mathbf{x}_n, y_n) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2 \,. \tag{6.2}$$

We focus on the case when the inputs are vectors $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^p$ and the target domain $\mathcal{Y} \subseteq \{0, 1\}$ in the case of classification or $\mathcal{Y} \subseteq \mathbb{R}$ in the case of regression.

6.1.1 Supervised Kernel Learning

In the case of supervised kernel learning [109, 132], Θ is taken to be a Hilbert space, denoted here as $\mathcal{H}$. Elements of $\mathcal{H}$ are *functions*, $f : \mathcal{X} \to \mathcal{Y}$, that admit a representation in terms of elements of $\mathcal{X}$ when $\mathcal{H}$ has a special structure. In particular, equip $\mathcal{H}$ with a unique *kernel*

function, $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that:

$$(i)\ \langle f, \kappa(\mathbf{x}, \cdot)) \rangle_{\mathcal{H}} = f(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathcal{X}\ , \qquad (ii)\ \mathcal{H} = \overline{\text{span}\{\kappa(\mathbf{x}, \cdot)\}} \quad \text{for all } \mathbf{x} \in \mathcal{X}\ . \tag{6.3}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the Hilbert inner product for $\mathcal{H}$. We further assume that the kernel is positive semidefinite, i.e. $\kappa(\mathbf{x}, \mathbf{x}') \geq 0$ for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. Function spaces with this structure are called reproducing kernel Hilbert spaces (RKHS).

In (6.3), property (i) is called the reproducing property of the kernel, and is a consequence of the Riesz Representation Theorem [212]. Replacing f by $\kappa(\mathbf{x}', \cdot)$ in (6.3) (i) yields the expression $\langle \kappa(\mathbf{x}', \cdot), \kappa(\mathbf{x}, \cdot) \rangle_{\mathcal{H}} = \kappa(\mathbf{x}, \mathbf{x}')$, which is the origin of the term "reproducing kernel." This property provides a practical means by which to access a nonlinear transformation of the input space $\mathcal{X}$. Specifically, denote by $\phi(\cdot)$ a nonlinear map of the feature space that assigns to each $\mathbf{x}$ the kernel function $\kappa(\cdot, \mathbf{x})$. Then the reproducing property of the kernel allows us to write the inner product of the image of distinct feature vectors $\mathbf{x}$ and $\mathbf{x}'$ under the map ϕ in terms of kernel evaluations only: $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}} = \kappa(\mathbf{x}, \mathbf{x}')$. This is commonly referred to as the *kernel trick*, and it provides a computationally efficient tool for learning nonlinear functions.

Moreover, property (6.3) (ii) states that any function $f \in \mathcal{H}$ may be written as a linear combination of kernel evaluations. For kernelized and regularized empirical risk minimization, the Representer Theorem [88, 169] establishes that the optimal f in the hypothesis function class $\mathcal{H}$ may be written as an expansion of kernel evaluations *only* at elements of the training set as

$$f(\mathbf{x}) = \sum_{n=1}^{N} w_n \kappa(\mathbf{x}_n, \mathbf{x})\ . \tag{6.4}$$

where $\mathbf{w} = [w_1, \cdots, w_N]^T \in \mathbb{R}^N$ denotes a set of weights. The upper summand index N in (6.4) is henceforth referred to as the model order. Common choices κ include the polynomial kernel and the radial basis kernel, i.e., $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T \mathbf{x}' + q\right)^b$ and $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2\tilde{\sigma}^2}\right\}$, respectively, where $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$.

We may now formulate the kernel variant of the empirical risk minimization problem as the one that minimizes the loss functional $L : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ plus a complexity-reducing penalty. The loss functional L may be written as an average over instantaneous losses $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, each of which penalizes the average deviation between $f(\mathbf{x}_n)$ and the associated output y_n over the training set $\mathcal{S}$. We denote the data loss and complexity loss as $R : \mathcal{H} \to \mathbb{R}$, and consider the problem

$$f^* = \underset{f \in \mathcal{H}}{\text{argmin}}\, R(f; \mathcal{S}) = \underset{f \in \mathcal{H}}{\text{argmin}}\, \frac{1}{N} \sum_{n=1}^{N} \ell(f(\mathbf{x}_n), y_n) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2\ . \tag{6.5}$$

The above problem, referred to as Tikhonov regularization [62], is one in which we aim to learn a general nonlinear relationship between $\mathbf{x}_n$ and y_n through a function f. Throughout, we assume ℓ is convex with respect to its first argument $f(\mathbf{x})$. By substituting the Representer Theorem expansion in (6.4) into (6.5), the optimization problem amounts to finding an optimal set of coefficients $\mathbf{w}$ as

$$\begin{aligned} f^* &= \underset{\mathbf{w}\in\mathbb{R}^N}{\operatorname{argmin}} \frac{1}{N}\sum_{n=1}^{N} \ell(\sum_{m=1}^{N} w_m \kappa(\mathbf{x}_m, \mathbf{x}_n), y_n) + \frac{\lambda}{2}\| \sum_{n,m=1}^{N} w_n w_m \kappa(\mathbf{x}_m, \mathbf{x}_n)\|_{\mathcal{H}}^2 \\ &= \underset{\mathbf{w}\in\mathbb{R}^N}{\operatorname{argmin}} \frac{1}{N}\sum_{n=1}^{N} \ell(\mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{X}}(\mathbf{x}_n), y_n) + \frac{\lambda}{2}\mathbf{w}^T \mathbf{K}_{\mathbf{X},\mathbf{X}} \mathbf{w} \end{aligned} \tag{6.6}$$

where we have defined the Gram matrix (variously referred to as the *kernel matrix*) $\mathbf{K}_{\mathbf{X},\mathbf{X}} \in \mathbb{R}^{N\times N}$, with entries given by the kernel evaluations between $\mathbf{x}_m$ and $\mathbf{x}_n$ as $[\mathbf{K}_{\mathbf{X},\mathbf{X}}]_{m,n} = \kappa(\mathbf{x}_m, \mathbf{x}_n)$. We further define the vector of kernel evaluations $\boldsymbol{\kappa}_{\mathbf{X}}(\cdot) = [\kappa(\mathbf{x}_1, \cdot) \dots \kappa(\mathbf{x}_N, \cdot)]^T$, which are related to the kernel matrix as $\mathbf{K}_{\mathbf{X},\mathbf{X}} = [\boldsymbol{\kappa}_{\mathbf{X}}(\mathbf{x}_1) \dots \boldsymbol{\kappa}_{\mathbf{X}}(\mathbf{x}_N)]$. The dictionary of training points associated with the kernel matrix is defined as $\mathbf{X} = [\mathbf{x}_1, \ \dots \ , \mathbf{x}_N]$.

Observe that by exploiting the Representer Theorem, we transform a nonparametric infinite dimensional optimization problem in $\mathcal{H}$ (6.5) into a finite N-dimensional parametric problem (6.6). Thus, for empirical risk minimization, the RKHS provides a principled framework to solve nonparametric regression problems as via search over $\mathbb{R}^N$ for an optimal set of coefficients. A motivating example is presented next to clarify the setting of supervised kernel learning.

Example 4 *(Kernel Logistic Regression)* Consider the case of *kernel logistic regression* (KLR), with feature vectors $\mathbf{x}_n \in \mathcal{X} \subseteq \mathbb{R}^p$ and binary class labels $y_n \in \{0, 1\}$. We seek to learn a function $f \in \mathcal{H}$ that allows us to best approximate the distribution of an unknown class label given a training example $\mathbf{x}$ under the assumed model

$$\mathbb{P}\left(y = 0 \mid \mathbf{x}\right) = \frac{\exp\left\{f(\mathbf{x})\right\}}{1 + \exp\left\{f(\mathbf{x})\right\}} \ . \tag{6.7}$$

In classical logistic regression, we assume that f is linear, i.e., $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$. In KLR, on the other hand, we instead seek a nonlinear function of the form given in (6.4). By making use of (7.4) and (6.4), we may formulate a maximum-likelihood estimation (MLE) problem to find the optimal function f on the basis of $\mathcal{S}$ by solving for the $\mathbf{w}$ that maximizes the

λ-regularized average negative log likelihood over $\mathcal{S}$, i.e.,

$$
\begin{aligned}
f^* &= \operatorname*{argmin}_{f\in\mathcal{H}} \frac{1}{N}\sum_{n=1}^{N}\left[-\log \mathbb{P}(y=y_n \mid \mathbf{x}=\mathbf{x}_n) + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2\right] \qquad (6.8)\\
&= \operatorname*{argmin}_{f\in\mathcal{H}} \frac{1}{N}\sum_{n=1}^{N}\left[\log\left(1+\exp\{f(\mathbf{x}_n)\}\right) - \mathbb{1}(y_n=1) - f(\mathbf{x}_n)\mathbb{1}(y_n=0) + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2\right]\\
&= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^N} \frac{1}{N}\sum_{n=1}^{N}\left[\log\left(1+\exp\{\mathbf{w}^T\boldsymbol{\kappa}_{\mathbf{X}}(\mathbf{x}_n)\}\right) - \mathbb{1}(y_n=1) - \mathbf{w}^T\boldsymbol{\kappa}_{\mathbf{X}}(\mathbf{x}_n)\mathbb{1}(y_n=0) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{K}_{\mathbf{X},\mathbf{X}}\mathbf{w}\right],
\end{aligned}
$$

where $\mathbb{1}(\cdot)$ represents the indicator function. Solving (6.8) amounts to finding a function f that, given a feature vector $\mathbf{x}$ and the model outlined by (7.4), best represents the class-conditional probabilities that the corresponding label y is either 0 or 1.

6.1.2 Online Kernel Learning

The goal of this paper is to solve problems of the form (6.5) when training examples $(\mathbf{x}_n, \mathbf{y}_n)$ either become sequentially available or their total number is not necessarily finite. To do so, we consider the case where $(\mathbf{x}_n, \mathbf{y}_n)$ are independent realizations from a stationary joint distribution of the random pair $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ [181]. In this case, the objective in (6.5) may be written as an expectation over this random pair as

$$
\begin{aligned}
f^* = \operatorname*{argmin}_{f\in\mathcal{H}} R(f) :&= \operatorname*{argmin}_{f\in\mathcal{H}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}),y)] + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 \qquad (6.9)\\
&= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{\mathcal{I}},\{\mathbf{x}_n\}_{n\in\mathcal{I}}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(\sum_{n\in\mathcal{I}} w_n\kappa(\mathbf{x}_n,\mathbf{x}),y)] + \frac{\lambda}{2}\|\sum_{n,m\in\mathcal{I}} w_n w_m \kappa(\mathbf{x}_m,\mathbf{x}_n)\|_{\mathcal{H}}^2 \quad .
\end{aligned}
$$

where we define the average loss as $L(f) := \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}),y)]$. In the second equality in (6.9), we substitute in the expansion of f given by the Representer Theorem generalized to the infinite sample-size case established in [144], with $\mathcal{I}$ as some countably infinite indexing set.

6.2 Parsimonious Online Learning with Kernels

We turn to deriving an algorithmic solution to the kernelized expected risk minimization problem stated in (6.1). To do so, two complexity bottlenecks must be overcome. The first is that in order to develop a numerical optimization scheme such as gradient descent, we must compute the functional gradient (Frechét derivative) of the expected risk $L(f)$ with respect to f, which requires infinitely many realizations of the random pair $(\mathbf{x}, y)$. This bottleneck is handled via stochastic approximation, as detailed in Section 6.2.1. The second issue is that when making use of the stochastic gradient method in the RKHS setting,

the resulting parametric updates require memory storage whose complexity is proportional to the iteration index (the curse of kernelization), which rapidly becomes unaffordable. To alleviate this memory explosion, we introduce our sparse stochastic projection scheme based upon kernel orthogonal matching pursuit in Section 6.2.2.

6.2.1 Functional Stochastic Gradient Descent

Following [89], we derive the generalization of the stochastic gradient method for the RKHS setting. The resulting procedure is referred to as *functional stochastic gradient descent* (FSGD). First, given an independent realization $(\mathbf{x}_t, y_t)$ of the random pair $(\mathbf{x}, y)$, we compute the stochastic functional gradient (Frechét derivative) of $L(f)$, stated as

$$\nabla_f \ell(f(\mathbf{x}_t), y_t)(\cdot) = \frac{\partial \ell(f(\mathbf{x}_t), y_t)}{\partial f(\mathbf{x}_t)} \frac{\partial f(\mathbf{x}_t)}{\partial f}(\cdot) \tag{6.10}$$

where we have applied the chain rule. Now, define the short-hand notation $\ell'(f(\mathbf{x}_t), y_t) := \partial \ell(f(\mathbf{x}_t), y_t)/\partial f(\mathbf{x}_t)$ for the derivative of $\ell(\mathbf{f}(\mathbf{x}_t), y_t)$ with respect to its first scalar argument $f(\mathbf{x}_t)$ evaluated at $\mathbf{x}_t$. To evaluate the second term on the right-hand side of (6.10), differentiate both sides of the expression defining the reproducing property of the kernel [cf. (6.3)(i)] with respect to f to obtain

$$\frac{\partial f(\mathbf{x}_t)}{\partial f} = \frac{\partial \langle f, \kappa(\mathbf{x}_t, \cdot)) \rangle_{\mathcal{H}}}{\partial f} = \kappa(\mathbf{x}_t, \cdot) \tag{6.11}$$

With this computation in hand, we present the stochastic gradient method for the kernelized λ-regularized expected risk minimization problem in (6.1) as

$$f_{t+1} = (1 - \eta_t \lambda) f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t) = (1 - \eta_t \lambda) f_t - \eta_t \ell'(f_t(\mathbf{x}_t), y_t) \kappa(\mathbf{x}_t, \cdot) \, , \tag{6.12}$$

where $\eta_t > 0$ is an algorithm step-size either chosen as diminishing with $\mathcal{O}(1/t)$ or a small constant – see Section 6.3. We further require that, given $\lambda > 0$, the step-size satisfies $\eta_t < 1/\lambda$ and the sequence is initialized as $f_0 = 0 \in \mathcal{H}$. Given this initialization, by making use of the Representer Theorem (6.4), at time t, the function f_t may be expressed as an expansion in terms of feature vectors $\mathbf{x}_t$ observed thus far as

$$f_t(\mathbf{x}) = \sum_{n=1}^{t-1} w_n \kappa(\mathbf{x}_n, \mathbf{x}) = \mathbf{w}_t^T \boldsymbol{\kappa}_{\mathbf{X}_t}(\mathbf{x}) \, . \tag{6.13}$$

On the right-hand side of (6.13) we have introduced the notation $\mathbf{X}_t = [\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}] \in \mathbb{R}^{p \times (t-1)}$ and $\boldsymbol{\kappa}_{\mathbf{X}_t}(\cdot) = [\kappa(\mathbf{x}_1, \cdot), \ \ldots \ , \kappa(\mathbf{x}_{t-1}, \cdot)]^T$. Moreover, observe that the kernel expansion in (6.13), taken together with the functional update (6.12), yields the fact that

performing the stochastic gradient method in $\mathcal{H}$ amounts to the following parametric updates on the kernel dictionary $\mathbf{X}$ and coefficient vector $\mathbf{w}$:

$$\mathbf{X}_{t+1} = [\mathbf{X}_t,\ \ \mathbf{x}_t], \quad \mathbf{w}_{t+1} = [(1-\eta_t\lambda)\mathbf{w}_t,\ \ -\eta_t \ell'(f_t(\mathbf{x}_t), y_t)]\ , \tag{6.14}$$

Observe that this update causes $\mathbf{X}_{t+1}$ to have one more column than $\mathbf{X}_t$. We define the *model order* as number of data points M_t in the dictionary at time t (the number of columns of $\mathbf{X}_t$). FSGD is such that $M_t = t-1$, and hence grows unbounded with iteration index t.

6.2.2 Model Order Control via Stochastic Projection

To mitigate the model order issue described above, we shall generate an approximate sequence of functions by orthogonally projecting functional stochastic gradient updates onto subspaces $\mathcal{H}_{\mathbf{D}} \subseteq \mathcal{H}$ that consist only of functions that can be represented using some dictionary $\mathbf{D} = [\mathbf{d}_1,\ \ldots,\ \mathbf{d}_M] \in \mathbb{R}^{p\times M}$, i.e., $\mathcal{H}_{\mathbf{D}} = \{f\ :\ f(\cdot) = \sum_{n=1}^{M} w_n \kappa(\mathbf{d}_n, \cdot) = \mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\} = \text{span}\{\kappa(\mathbf{d}_n, \cdot)\}_{n=1}^{M}$. For convenience we have defined $[\boldsymbol{\kappa}_{\mathbf{D}}(\cdot) = \kappa(\mathbf{d}_1, \cdot)\ldots\kappa(\mathbf{d}_M, \cdot)]$, and $\mathbf{K}_{\mathbf{D},\mathbf{D}}$ as the resulting kernel matrix from this dictionary. We will enforce parsimony in function representation by selecting dictionaries $\mathbf{D}$ that $M_t << t$.

We first show that, by selecting $\mathbf{D} = \mathbf{X}_{t+1}$ at each iteration, the sequence (6.12) derived in Section 6.2.1 may be interpreted as carrying out a sequence of orthogonal projections. To see this, rewrite (6.12) as the quadratic minimization

$$\begin{aligned} f_{t+1} &= \operatorname*{argmin}_{f\in\mathcal{H}} \left\| f - \Big((1-\eta_t\lambda) f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t)\Big)\right\|_{\mathcal{H}}^2 \\ &= \operatorname*{argmin}_{f\in\mathcal{H}_{\mathbf{X}_{t+1}}} \left\| f - \Big((1-\eta_t\lambda) f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t)\Big)\right\|_{\mathcal{H}}^2, \end{aligned} \tag{6.15}$$

where the first equality in (6.15) comes from ignoring constant terms which vanish upon differentiation with respect to f, and the second comes from observing that f_{t+1} can be represented using only the points $\mathbf{X}_{t+1}$, using (6.14). Notice now that (6.15) expresses f_{t+1} as the orthogonal projection of the update $(1-\eta_t\lambda)f_t - \eta_t\nabla_f \ell(f_t(\mathbf{x}_t), y_t)$ onto the subspace defined by dictionary $\mathbf{X}_{t+1}$.

Rather than select dictionary $\mathbf{D} = \mathbf{X}_{t+1}$, we propose instead to select a different dictionary, $\mathbf{D} = \mathbf{D}_{t+1}$, which is extracted from the data points observed thus far, at each iteration. The process by which we select $\mathbf{D}_{t+1}$ will be discussed shortly, but is of of dimension $p \times M_{t+1}$, with $M_{t+1} << t$. As a result, we shall generate a function sequence f_t that differs from the functional stochastic gradient method presented in Section 6.2.1. The function f_{t+1} is parameterized dictionary $\mathbf{D}_{t+1}$ and weight vector $\mathbf{w}_{t+1}$. We denote columns of $\mathbf{D}_{t+1}$ as $\mathbf{d}_n$ for $n = 1, \ldots, M_{t+1}$, where the time index is dropped for notational clarity

but may be inferred from the context.

To be specific, we propose replacing the update (6.15) in which the dictionary grows at each iteration by the stochastic projection of the functional stochastic gradient sequence onto the subspace $\mathcal{H}_{\mathbf{D}_{t+1}} = \text{span}\{\kappa(\mathbf{d}_n, \cdot)\}_{n=1}^{M_{t+1}}$ as

$$\begin{aligned} f_{t+1} &= \underset{f \in \mathcal{H}_{\mathbf{D}_{t+1}}}{\text{argmin}} \left\| f - \Big((1-\eta_t\lambda) f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t) \Big) \right\|_{\mathcal{H}}^2 \\ &:= \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \Big[(1-\eta_t\lambda) f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t) \Big]. \end{aligned} \tag{6.16}$$

where we define the projection operator $\mathcal{P}$ onto subspace $\mathcal{H}_{\mathbf{D}_{t+1}} \subset \mathcal{H}$ by the update (6.16).

Coefficient update The update (6.16), for a fixed dictionary $\mathbf{D}_{t+1} \in \mathbb{R}^{p \times M_{t+1}}$, may be expressed in terms of the parameter space of coefficients only. In order to do so, we first define the stochastic gradient update without projection, given function f_t parameterized by dictionary $\mathbf{D}_t$ and coefficients $\mathbf{w}_t$, as

$$\tilde{f}_{t+1} = (1-\eta_t\lambda) f_t - \eta_t \nabla_f \ell(f_t; \mathbf{x}_t, \mathbf{y}_t). \tag{6.17}$$

This update may be represented using dictionary and weight vector

$$\tilde{\mathbf{D}}_{t+1} = [\mathbf{D}_t, \ \ \mathbf{x}_t], \quad \tilde{\mathbf{w}}_{t+1} = [(1-\eta_t\lambda)\mathbf{w}_t, \ \ -\eta_t \ell'(f_t(\mathbf{x}_t), y_t)] \ . \tag{6.18}$$

Observe that $\tilde{\mathbf{D}}_{t+1}$ has $\tilde{M} = M_t + 1$ columns, which is also the length of $\tilde{\mathbf{w}}_{t+1}$. For a fixed dictionary $\mathbf{D}_{t+1}$, the stochastic projection in (6.16) amounts to a least-squares problem on the coefficient vector. To see this, make use of the Representer Theorem to rewrite (6.16) in terms of kernel expansions, and that the coefficient vector is the only free parameter to write

$$\begin{aligned} &\underset{\mathbf{w} \in \mathbb{R}^{M_{t+1}}}{\text{argmin}} \ \frac{1}{2\eta_t} \Big\| \sum_{n=1}^{M_{t+1}} w_n \kappa(\mathbf{d}_n, \cdot) - \sum_{m=1}^{\tilde{M}} \tilde{w}_m \kappa(\tilde{\mathbf{d}}_m, \cdot) \Big\|_{\mathcal{H}}^2 \qquad (6.19) \\ &= \underset{\mathbf{w} \in \mathbb{R}^{M_{t+1}}}{\text{argmin}} \frac{1}{2\eta_t} \left(\sum_{n,n'=1}^{M_{t+1}} w_n w_{n'} \kappa(\mathbf{d}_n, \mathbf{d}_{n'}) - 2 \sum_{n,m=1}^{M_{t+1},\tilde{M}} w_n \tilde{w}_m \kappa(\mathbf{d}_n, \tilde{\mathbf{d}}_m) + \sum_{m,m'=1}^{\tilde{M}} \tilde{w}_m \tilde{w}_{m'} \kappa(\tilde{\mathbf{d}}_m, \tilde{\mathbf{d}}_{m'}) \right) \\ &= \underset{\mathbf{w} \in \mathbb{R}^{M_{t+1}}}{\text{argmin}} \frac{1}{2\eta_t} \Big(\mathbf{w}^T \mathbf{K}_{\mathbf{D}_{t+1}, \mathbf{D}_{t+1}} \mathbf{w} - 2\mathbf{w}^T \mathbf{K}_{\mathbf{D}_{t+1}, \tilde{\mathbf{D}}_{t+1}} \tilde{\mathbf{w}}_{t+1} + \tilde{\mathbf{w}}_{t+1} \mathbf{K}_{\tilde{\mathbf{D}}_{t+1}, \tilde{\mathbf{D}}_{t+1}} \tilde{\mathbf{w}}_{t+1} \Big) := \mathbf{w}_{t+1} \ . \end{aligned}$$

In (6.19), the first equality comes from expanding the square, and the second comes from defining the cross-kernel matrix $\mathbf{K}_{\mathbf{D}_{t+1}, \tilde{\mathbf{D}}_{t+1}}$ whose $(n,m)^{\text{th}}$ entry is given by $\kappa(\mathbf{d}_n, \tilde{\mathbf{d}}_m)$. The other kernel matrices $\mathbf{K}_{\tilde{\mathbf{D}}_{t+1}, \tilde{\mathbf{D}}_{t+1}}$ and $\mathbf{K}_{\mathbf{D}_{t+1}, \mathbf{D}_{t+1}}$ are similarly defined. Note that M_{t+1} is the number of columns in $\mathbf{D}_{t+1}$, while $\tilde{M} = M_t + 1$ is the number of columns in $\tilde{\mathbf{D}}_{t+1}$ [cf. (6.18)]. The explicit solution of (6.19) may be obtained by noting that the last term is

a constant independent of $\mathbf{w}$, and thus by computing gradients and solving for $\mathbf{w}_{t+1}$ we obtain

$$\mathbf{w}_{t+1} = \mathbf{K}_{\mathbf{D}_{t+1}\mathbf{D}_{t+1}}^{-1} \mathbf{K}_{\mathbf{D}_{t+1}\tilde{\mathbf{D}}_{t+1}} \tilde{\mathbf{w}}_{t+1} \ , \tag{6.20}$$

Given that the projection of $\tilde{f}_{t+1}$ onto the stochastic subspace $\mathcal{H}_{\mathbf{D}_{t+1}}$, for a fixed dictionary $\mathbf{D}_{t+1}$, amounts to a simple least-squares multiplication, we turn to detailing how the kernel dictionary $\mathbf{D}_{t+1}$ is selected from the data sample path $\{\mathbf{x}_u, y_u\}_{u \leq t}$.

Dictionary Update The selection procedure for the kernel dictionary $\mathbf{D}_{t+1}$ is based upon greedy sparse approximation, a topic studied extensively in the compressive sensing community [137]. The function $\tilde{f}_{t+1} = (1 - \eta_t) f_t - \eta_t \nabla_f \ell(f_t; \mathbf{x}_t, \mathbf{y}_t)$ defined by stochastic gradient method without projection is parameterized by dictionary $\tilde{\mathbf{D}}_{t+1}$ [cf. (6.18)], whose model order is $\tilde{M} = M_t + 1$. We form $\mathbf{D}_{t+1}$ by selecting a subset of M_{t+1} columns from $\tilde{\mathbf{D}}_{t+1}$ that are best for approximating $\tilde{f}_{t+1}$ in terms of error with respect to the Hilbert norm. As previously noted, numerous approaches are possible for seeking a sparse representation. We make use of *kernel orthogonal matching pursuit* (KOMP) [203] with allowed error tolerance ϵ_t to find a kernel dictionary matrix $\mathbf{D}_{t+1}$ based on the one which adds the latest sample point $\tilde{\mathbf{D}}_{t+1}$. This choice is due to the fact that we can tune its stopping criterion to guarantee stochastic descent, and guarantee the model order of the learned function remains finite – see Section 6.3 for details.

We now describe the variant of KOMP we propose using, called Destructive KOMP with Pre-Fitting (see [203], Section 2.3), which is summarized in Algorithm 5. This flavor of KOMP takes as an input a candidate function $\tilde{f}$ of model order $\tilde{M}$ parameterized by its kernel dictionary $\tilde{\mathbf{D}} \in \mathbb{R}^{p \times \tilde{M}}$ and coefficient vector $\tilde{\mathbf{w}} \in \mathbb{R}^{\tilde{M}}$. The method then seeks to approximate $\tilde{f}$ by a parsimonious function $f \in \mathcal{H}$ with a lower model order. Initially, this sparse approximation is the original function $f = \tilde{f}$ so that its dictionary is initialized with that of the original function $\mathbf{D} = \tilde{\mathbf{D}}$, with corresponding coefficients $\mathbf{w} = \tilde{\mathbf{w}}$. Then, the algorithm sequentially removes dictionary elements from the initial dictionary $\tilde{\mathbf{D}}$, yielding a sparse approximation f of $\tilde{f}$, until the error threshold $\|f - \tilde{f}\|_{\mathcal{H}} \leq \epsilon_t$ is violated, in which case it terminates.

At each stage of KOMP, a single dictionary element j of $\mathbf{D}$ is selected to be removed which contributes the least to the Hilbert-norm approximation error $\min_{f \in \mathcal{H}_{\mathbf{D} \setminus \{j\}}} \|\tilde{f} - f\|_{\mathcal{H}}$ of the original function $\tilde{f}$, when dictionary $\mathbf{D}$ is used. Since at each stage the kernel dictionary is fixed, this amounts to a computation involving weights $\mathbf{w} \in \mathbb{R}^{M-1}$ only; that is, the error of removing dictionary point $\mathbf{d}_j$ is computed for each j as $\gamma_j = \min_{\mathbf{w}_{\mathcal{I} \setminus \{j\}} \in \mathbb{R}^{M-1}} \|\tilde{f}(\cdot) - \sum_{k \in \mathcal{I} \setminus \{j\}} w_k \kappa(\mathbf{d}_k, \cdot)\|$. We use the notation $\mathbf{w}_{\mathcal{I} \setminus \{j\}}$ to denote the entries of $\mathbf{w} \in \mathbb{R}^M$ restricted to the sub-vector associated with indices $\mathcal{I} \setminus \{j\}$. Then, we define the dictionary element which contributes the least to the approximation error as $j^* = \operatorname{argmin}_j \gamma_j$. If the error associated with removing this kernel dictionary element exceeds the given

Algorithm 5 Destructive Kernel Orthogonal Matching Pursuit (KOMP)

Require: function $\tilde{f}$ defined by dict. $\tilde{\mathbf{D}} \in \mathbb{R}^{p \times \tilde{M}}$, coeffs. $\tilde{\mathbf{w}} \in \mathbb{R}^{\tilde{M}}$, approx. budget $\epsilon_t > 0$

initialize $f = \tilde{f}$, dictionary $\mathbf{D} = \tilde{\mathbf{D}}$ with indices $\mathcal{I}$, model order $M = \tilde{M}$, coeffs. $\mathbf{w} = \tilde{\mathbf{w}}$.
while candidate dictionary is non-empty $\mathcal{I} \neq \emptyset$ **do**
  for $j = 1, \ldots, \tilde{M}$ **do**
    Find minimal approximation error with dictionary element $\mathbf{d}_j$ removed

$$\gamma_j = \min_{\mathbf{w}_{\mathcal{I}\setminus\{j\}} \in \mathbb{R}^{M-1}} \|\tilde{f}(\cdot) - \sum_{k \in \mathcal{I}\setminus\{j\}} w_k \kappa(\mathbf{d}_k, \cdot)\|_{\mathcal{H}} \, .$$

  end for
  Find dictionary index minimizing approximation error: $j^* = \operatorname{argmin}_{j \in \mathcal{I}} \gamma_j$
  if minimal approximation error exceeds threshold $\gamma_{j^*} > \epsilon_t$
    stop
  else
    Prune dictionary $\mathbf{D} \leftarrow \mathbf{D}_{\mathcal{I}\setminus\{j^*\}}$
    Revise set $\mathcal{I} \leftarrow \mathcal{I} \setminus \{j^*\}$ and model order $M \leftarrow M - 1$.
    Compute updated weights $\mathbf{w}$ defined by current dictionary $\mathbf{D}$

$$\mathbf{w} = \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^M} \|\tilde{f}(\cdot) - \mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\|_{\mathcal{H}}$$

  end
end while
return $f, \mathbf{D}, \mathbf{w}$ of model order $M \leq \tilde{M}$ such that $\|f - \tilde{f}\|_{\mathcal{H}} \leq \epsilon_t$

approximation budget $\gamma_{j^*} > \epsilon_t$, the algorithm terminates. Otherwise, this dictionary element $\mathbf{d}_{j^*}$ is removed, the weights $\mathbf{w}$ are revised based on the pruned dictionary as $\mathbf{w} = \operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^M} \|\tilde{f}(\cdot) - \mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\|_{\mathcal{H}}$, and the process repeats as long as the current function approximation is defined by a nonempty dictionary.

With Algorithm 5 stated, we may summarize the key steps of the proposed method in Algorithm 6 for solving (6.1) while maintaining a finite model order, thus breaking the "curse of kernelization." The method, Parsimonious Online Learning with Kernels (POLK), executes the stochastic projection of the functional stochastic gradient iterates onto sparse subspaces $\mathcal{H}_{\mathbf{D}_{t+1}}$ stated in (6.16). The initial function is set to null $f_0 = 0$, meaning that it has empty kernel dictionary $\mathbf{D}_0 = []$ and coefficient vector $\mathbf{w}_0 = []$. The notation [] is used to denote the empty matrix or vector respective size $p \times 0$ or 0. Then, at each step, given an independent training example $(\mathbf{x}_t, y_t)$ and step-size η_t, we compute the *unconstrained* functional stochastic gradient iterate $\tilde{f}_{t+1}(\cdot) = (1 - \eta_t \lambda) f_t - \eta_t \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t) \kappa(\mathbf{x}_t, \cdot)$ which admits the parametric representation $\tilde{\mathbf{D}}_{t+1}$ and $\tilde{\mathbf{w}}_{t+1}$ as stated in (6.18). These parameters are then fed into KOMP with approximation budget ϵ_t, such that $(f_{t+1}, \mathbf{D}_{t+1}, \mathbf{w}_{t+1}) = \text{KOMP}(\tilde{f}_{t+1}, \tilde{\mathbf{D}}_{t+1}, \tilde{\mathbf{w}}_{t+1}, \epsilon_t)$.

In the next section, we discuss the analytical properties of Algorithm 6 for solving

Algorithm 6 Parsimonious Online Learning with Kernels (POLK)

Require: $\{\mathbf{x}_t, \mathbf{y}_t, \eta_t, \epsilon_t\}_{t=0,1,2,\ldots}$

initialize $f_0(\cdot) = 0, \mathbf{D}_0 = [], \mathbf{w}_0 = []$, i.e. initial dictionary, coefficient vectors are empty

for $t = 0, 1, 2, \ldots$ **do**

Obtain independent training pair realization $(\mathbf{x}_t, y_t)$

Compute unconstrained functional stochastic gradient step [cf. (6.17)]

$$\tilde{f}_{t+1}(\cdot) = (1 - \eta_t\lambda) f_t - \eta_t \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)\kappa(\mathbf{x}_t, \cdot)$$

Revise dictionary $\tilde{\mathbf{D}}_{t+1} = [\mathbf{D}_t, \quad \mathbf{x}_t]$ and weights $\tilde{\mathbf{w}}_{t+1} \leftarrow [(1 - \eta_t\lambda)\mathbf{w}_t, \ -\eta_t\ell'(f_t(\mathbf{x}_t), y_t)]$

Compute sparse function approximation via Algorithm 5

$$(f_{t+1}, \mathbf{D}_{t+1}, \mathbf{w}_{t+1}) = \mathbf{KOMP}(\tilde{f}_{t+1}, \tilde{\mathbf{D}}_{t+1}, \tilde{\mathbf{w}}_{t+1}, \epsilon_t)$$

end for

online nonparametric regression problems of the form (6.1). We close here with an example algorithm derivation for the kernel logistic regression problem stated in Example 4.

Example 5 *(Kernel Logistic Regression)* Returning to the case of kernel logistic regression stated in Example 4, with feature vectors $\mathbf{x}_n \in \mathcal{X} \subseteq \mathbb{R}^p$ and binary class labels $y_n \in \{0, 1\}$, we may perform sparse function estimation in $\mathcal{H}$ that fits a a training example $\mathbf{x}$ to its associated label y under the logistic model [cf. (7.4)] of the odds-ratio of the given class label. The associated λ-regularized maximum-likelihood estimation (MLE) is given as (6.8). Provided that a particular kernel map $\kappa(\cdot, \cdot)$, regularizer λ, and step-size η_t have been chosen, the only specialization of Algorithm 6 to this case is the computation of $\tilde{f}_t$, which requires computing the stochastic gradient of (7.4) with respect to an instantaneous training example $(\mathbf{x}_t, y_t)$. Doing so specializes (6.17) to

$$\tilde{f}_{t+1}(\cdot) = (1 - \eta_t\lambda) f_t - \eta_t \frac{\exp\{-\tilde{f}_t(\mathbf{x}_t)\}}{[1 + \exp\{-\tilde{f}_t(\mathbf{x}_t)\}]^2}\kappa(\mathbf{x}_t, \cdot) \ . \tag{6.21}$$

The resulting dictionary and parameter updates implied by (6.21), given in (6.18), are then fed into KOMP (Algorithm 5) which returns their greedy sparse approximation for a fixed budget ϵ_t.

6.3 POLK Convergence

We turn to studying the theoretical performance of Algorithm 6 developed in Section 6.2. In particular, we establish that the method, when a diminishing step-size is chosen, is guaranteed to converge to the optimum of (6.1). We further obtain that when a sufficiently

small constant step-size is chosen, the limit infimum of the iterate sequence is within a neighborhood of the optimum. In both cases, the convergence behavior depends on the approximation budget used in the online sparsification procedure detailed in Algorithm 5.

We also perform a worst-case analysis of the model order of the instantaneous iterates resulting from Algorithm 6, and show that asymptotically the model order depends on that of the optimal $f^* \in \mathcal{H}$. Before analyzing the proposed method developed in Section 6.2, we define key quantities to simplify the analysis and introduce standard assumptions which are necessary to establish convergence. First, define the regularized stochastic functional gradient as

$$\hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) = \nabla_f \ell(f_t(\mathbf{x}_t), y_t) + \lambda f_t \tag{6.22}$$

Further define the projected stochastic functional gradient associated with the update in (6.16) as

$$\tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) = \Big(f_t - \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \Big[f_t - \eta_t \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \Big] \Big) / \eta_t \tag{6.23}$$

such that the Hilbert space update of Algorithm 6 [cf. (6.16)] may be expressed as a stochastic descent using projected functional gradients

$$f_{t+1} = f_t - \eta_t \tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \; . \tag{6.24}$$

The definitions (6.23) - (6.22) will be used to analyze the convergence behavior of the algorithm. Before doing so, observe that the stochastic functional gradient in (6.22), based upon the fact that $(\mathbf{x}_t, y_t)$ are independent and identically distributed realizations of the random pair $(\mathbf{x}, y)$, is an unbiased estimator of the true functional gradient of the regularized expected risk $R(f)$ in (6.1), i.e.

$$\mathbb{E}[\hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \,\big|\, \mathcal{F}_t] = \nabla_f R(f_t) \tag{6.25}$$

for all t. Next, we formally state technical conditions on the loss functions, data domain, and stochastic approximation errors that are necessary to establish convergence.

AS14 *The feature space $\mathcal{X} \subset \mathbb{R}^p$ and target domain $\mathcal{Y} \subset \mathbb{R}$ are compact, and the reproducing kernel map may be bounded as*

$$\sup_{\mathbf{x} \in \mathcal{X}} \sqrt{\kappa(\mathbf{x}, \mathbf{x})} = X < \infty \tag{6.26}$$

AS15 *The instantaneous loss $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is uniformly C-Lipschitz continuous for all $z \in \mathbb{R}$ for a fixed $y \in \mathcal{Y}$*

$$|\ell(z, y) - \ell(z', y)| \leq C|z - z'| \tag{6.27}$$

AS16 *The loss function $\ell(f(\mathbf{x}), y)$ is convex and differentiable with respect to its first (scalar) argument $f(\mathbf{x})$ on $\mathbb{R}$ for all $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$.*

AS17 *Let $\mathcal{F}_t$ denote the sigma algebra which measures the algorithm history for times $u \leq t$, i.e. $\mathcal{F}_t = \{\mathbf{x}_u, y_u, u_u\}_{u=1}^t$. The projected functional gradient of the regularized instantaneous risk in* (6.22) *has finite conditional second moments for each t, that is,*

$$\mathbb{E}[\|\tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}}^2 \mid \mathcal{F}_t] \leq \sigma^2 \tag{6.28}$$

Assumption 14 holds in most practical settings by the data domain itself, and justifies the bounding of the loss in Assumption 15. Taken together, these conditions permit bounding the optimal function f^* in the Hilbert norm, and imply that the worst-case model order is guaranteed to be finite. Variants of Assumption 15 appear in the analysis of stochastic descent methods in the kernelized setting [150, 221]. Assumption 16 is satisfied for supervised learning problems such as logistic regression, support vector machines with the square-hinge-loss, the square loss, among others. Moreover, it is a standard condition in the analysis of descent methods (see [33]). Assumption 17 is common in stochastic approximation literature, and ensures that the variance of the stochastic approximation error is finite.

Next we establish an auxiliary result needed to prove Theorems 5 and 6 which bounds the magnitude of the iterates of Algorithm 6 in the Hilbert norm.

Proposition 3 *Let Assumptions 14-17 hold and denote $\{f_t\}$ as the sequence generated by Algorithm 6 with $f_0 = 0$. Further denote f^* as the optimum defined by* (6.1)*. Both quantities are bounded by the constant $K := CX/\lambda$ in Hilbert norm for all t as*

$$\|f_t\|_{\mathcal{H}} \leq \frac{CX}{\lambda}, \qquad \|f^*\|_{\mathcal{H}} \leq \frac{CX}{\lambda} \tag{6.29}$$

Proof: First, since we repeatedly use the Cauchy-Schwartz inequality together with the reproducing kernel property in the following analysis, we here note that for all $g \in \mathcal{H}$, $|g(\mathbf{x}_t)| \leq |\langle g, \kappa(\mathbf{x}_t, \cdot)\rangle_{\mathcal{H}}| \leq X\|g\|_{\mathcal{H}}$. Now, consider the magnitude of f_1 in the Hilbert norm, given $f_0 = 0$

$$\begin{aligned}
\|f_1\|_{\mathcal{H}} &= \left\| \mathcal{P}_{\mathcal{H}_{\mathbf{D}_1}} \left[\eta_0 \nabla_f \ell(0, y_0) \right] \right\|_{\mathcal{H}} \\
&\leq \eta_0 \|\nabla_f \ell(0, y_0)\|_{\mathcal{H}} \leq \eta_0 |\ell'(0, y_0)| \|\kappa(\mathbf{x}_0, \cdot)\|_{\mathcal{H}} \\
&\leq \eta_0 CX < \frac{CX}{\lambda}
\end{aligned} \tag{6.30}$$

The first equality comes from substituting in $f_0 = 0$ and the second inequality comes from the definition of optimality condition of the projection operator and the homogeneity of

the Hilbert norm, and the third uses the derivation of the functional stochastic gradient in (6.10) with the Cauchy-Schwartz inequality. Lastly, we make use of Assumptions 14 and 15 to bound the scalar derivative ℓ' using the Lipschitz constant, and the boundedness of the kernel map [cf. (6.26)]. The final strict inequality in (6.30) comes from applying the step-size condition $\eta_0 < 1/\lambda$.

Now we consider the induction step. Given the induction hypothesis $\|f_t\|_{\mathcal{H}} \leq CX/\lambda$, consider the magnitude of the iterate at the time $t+1$ as

$$\begin{aligned}\|f_{t+1}\|_{\mathcal{H}} &= \left\|\mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}}\left[(1-\eta_t\lambda)f_t - \eta_t\nabla_f\ell(f_t(\mathbf{x}_t),y_t)\right]\right\|_{\mathcal{H}} \\ &\leq \|(1-\eta_t\lambda)f_t - \eta_t\nabla_f\ell(f_t(\mathbf{x}_t),y_t)\|_{\mathcal{H}} \\ &\leq (1-\eta_t\lambda)\|f_t\| + \eta_t\|\nabla_f\ell(f_t(\mathbf{x}_t),y_t)\|_{\mathcal{H}}\ , \end{aligned} \tag{6.31}$$

where we have applied the non-expansion property of the projection operator for the first inequality on the right-hand side of (6.31), and the triangle inequality for the second. Now, apply the induction hypothesis $\|f_t\|_{\mathcal{H}} \leq CX/\lambda$ to the first term on the right-hand side of (6.31), and the chain rule together with the triangle inequality to the second to obtain

$$\begin{aligned}\|f_{t+1}\|_{\mathcal{H}} &\leq (1-\eta_t\lambda)\frac{CX}{\lambda} + \eta_t|\ell'(f_t(\mathbf{x}_t),y_t)|\|\kappa(\mathbf{x}_t,\cdot)\|_{\mathcal{H}} \\ &\leq (\frac{1}{\lambda}-\eta_t)CX + \eta_t CX = \frac{CX}{\lambda}\end{aligned} \tag{6.32}$$

where we have made use of Assumptions 14 and 15 to bound the scalar derivative ℓ' using the Lipschitz constant, and the boundedness of the kernel map [cf. (6.26)] as in the base case for f_1, as well as the fact that $\eta_t < 1/\lambda$. The same bound holds for f^* by applying the result of Section V-B of [89] with $m \to \infty$.

■

Next we introduce a proposition which quantifies the error due to our sparse stochastic projection scheme in terms of the ratio of the sparse approximation budget to the algorithm step-size.

Proposition 4 *Given independent identical realizations $(\mathbf{x}_t, y_t)$ of the random pair $(\mathbf{x}, y)$, the difference between the projected stochastic functional gradient and the stochastic functional gradient of the regularized instantaneous risk defined by* (6.23) *and* (6.22)*, respectively, is bounded for all t as*

$$\|\tilde{\nabla}_f\ell(f_t(\mathbf{x}_t),y_t) - \hat{\nabla}_f\ell(f_t(\mathbf{x}_t),y_t)\|_{\mathcal{H}} \leq \frac{\epsilon_t}{\eta_t} \tag{6.33}$$

where $\eta_t > 0$ denotes the algorithm step-size and $\epsilon_t > 0$ is the approximation budget param-

eter of Algorithm 5.

Proof: Consider the square-Hilbert-norm difference of $\tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)$ and $\hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)$ defined in (6.22) and (6.23), respectively,

$$\begin{aligned} &\|\tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) - \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}}^2 \\ &\quad = \left\| \left(f_t - \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \left[f_t - \eta_t \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \right] \right) / \eta_t - \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \right\|_{\mathcal{H}}^2 \end{aligned} \tag{6.34}$$

Multiply and divide $\hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)$, the last term, by η_t, and reorder terms to write

$$\begin{aligned} &\left\| \left(f_t - \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \left[f_t - \eta_t \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \right] \right) / \eta_t - \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \right\|_{\mathcal{H}}^2 \\ &\qquad = \left\| \frac{1}{\eta_t} \left(f_t - \eta_t \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \right) - \frac{1}{\eta_t} \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \left[f_t - \eta_t \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \right] \right\|_{\mathcal{H}}^2 \\ &\qquad = \frac{1}{\eta_t^2} \|\tilde{f}_{t+1} - f_{t+1}\|_{\mathcal{H}}^2 \end{aligned} \tag{6.35}$$

where we have substituted the definition of $\tilde{f}_{t+1}$ and f_{t+1} in (6.17) and (6.15), respectively, and pulled the nonnegative scalar η_t outside the norm. Now, observe that the KOMP residual stopping criterion in Algorithm 5 is $\|\tilde{f}_{t+1} - f_{t+1}\|_{\mathcal{H}} \leq \epsilon_t$, which we may apply to the last term on the right-hand side of (6.35) to conclude (6.33).

■

Lemma 5 *(Stochastic Descent) Consider the sequence generated $\{f_t\}$ by Algorithm 6 with $f_0 = 0$. Under Assumptions 14-17, the following expected descent relation holds.*

$$\mathbb{E}\left[\|f_{t+1} - f^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t \right] \leq \|f_t - f^*\|_{\mathcal{H}}^2 - 2\eta_t [R(f_t) - R(f^*)] + 2\epsilon_t \|f_t - f^*\|_{\mathcal{H}} + \eta_t^2 \sigma^2 \,. \tag{6.36}$$

Proof: Begin by considering the square of the Hilbert-norm difference between f_{t+1} and f^* defined by (6.1), and expand the square to write

$$\begin{aligned} \|f_{t+1} - f^*\|_{\mathcal{H}}^2 &= \|f_t - \eta_t \tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}}^2 \\ &= \|f_t - f^*\|_{\mathcal{H}}^2 - 2\eta_t \langle f_t - f^*, \tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \rangle_{\mathcal{H}} + \eta_t^2 \|\tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}}^2 \end{aligned} \tag{6.37}$$

Add and subtract the gradient of the regularized instantaneous risk $\hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)$ defined in (6.22) to the second term on the right-hand side of (6.37) to obtain

$$\begin{aligned} \|f_{t+1} - f^*\|_{\mathcal{H}}^2 = \|f_t - f^*\|_{\mathcal{H}}^2 &- 2\eta_t \langle f_t - f^*, \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \rangle_{\mathcal{H}} \\ &- 2\eta_t \langle f_t - f^*, \tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) - \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \rangle_{\mathcal{H}} + \eta_t^2 \|\tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}}^2 \end{aligned} \tag{6.38}$$

We deal with the third term on the right-hand side of (6.38), which represents the directional error associated with the sparse stochastic projections, by applying the Cauchy-Schwartz inequality together with Proposition 7 to obtain

$$\begin{aligned}\|f_{t+1} - f^*\|_{\mathcal{H}}^2 \leq \|f_t - f^*\|_{\mathcal{H}}^2 - 2\eta_t \langle f_t - f^*, \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)\rangle_{\mathcal{H}} \\ + 2\epsilon_t \|f_t - f^*\|_{\mathcal{H}} + \eta_t^2 \|\tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}}^2\end{aligned} \tag{6.39}$$

Now compute the expectation of (6.39) conditional on the algorithm history $\mathcal{F}_t$

$$\mathbb{E}\left[\|f_{t+1} - f^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq \|f_t - f^*\|_{\mathcal{H}}^2 - 2\eta_t \langle f_t - f^*, \nabla_f R(f_t)\rangle_{\mathcal{H}} + 2\epsilon_t \|f_t - f^*\|_{\mathcal{H}} + \eta_t^2 \sigma^2 \;, \tag{6.40}$$

where we have applied the fact that the stochastic functional gradient in (6.22) is an unbiased estimator [cf. (6.25)] for the functional gradient of the expected risk in (6.1), as well as the fact that the variance of the functional projected stochastic gradient is finite stated in (6.28) (Assumption 17). Observe that since $R(f)$ is an expectation of a convex function, it is also convex, which allows us to write

$$R(f_t) - R(f^*) \leq \langle f_t - f^*, \nabla_f R(f_t)\rangle_{\mathcal{H}} \;, \tag{6.41}$$

which we substitute into the second term on the right-hand side of the relation given in (6.40) to obtain

$$\mathbb{E}\left[\|f_{t+1} - f^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq \|f_t - f^*\|_{\mathcal{H}}^2 - 2\eta_t [R(f_t) - R(f^*)] + 2\epsilon_t \|f_t - f^*\|_{\mathcal{H}} + \eta_t^2 \sigma^2 \;. \tag{6.42}$$

Thus the claim in Lemma 7 is valid. ■

6.3.1 Iterate Convergence

As is customary in the analysis of stochastic algorithms, we establish that under a diminishing algorithm step-size scheme (non-summable and square-summable), with the sparse approximation budget selection

$$\sum_{t=1}^{\infty} \eta_t = \infty \;, \quad \sum_{t=1}^{\infty} \eta_t^2 < \infty \;, \quad \epsilon_t = \eta_t^2 \;, \tag{6.43}$$

Algorithm 6 converges exactly to the optimal function f^* in stated in (6.1) almost surely.

Theorem 5 *Consider the sequence generated $\{f_t\}$ by Algorithm 6 with $f_0 = 0$, and denote f^* as the minimizer of the regularized expected risk stated in* (6.1)*. Let Assumptions 14-17*

hold and suppose the step-size rules and approximation budget are diminishing as in (6.43) *with regularizer such that* $\eta_t < 1/\lambda$ *for all* t*. Then the objective function error sequence converges to null in infimum almost surely as*

$$\liminf_{t\to\infty} R(f_t) - R(f^*) = 0 \qquad a.s. \tag{6.44}$$

Moreover, the sequence of functions $\{f_t\}$ *converges almost surely to the optimum* f^* *as*

$$\lim_{t\to\infty} \|f_t - f^*\|_{\mathcal{H}}^2 = 0 \qquad a.s. \tag{6.45}$$

Proof: Apply the iterate bound stated in Proposition 3 to the third term on the right-hand side of (6.36) (Lemma 7) to write

$$\mathbb{E}\left[\|f_{t+1} - f^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq \|f_t - f^*\|_{\mathcal{H}}^2 - 2\eta_t[R(f_t) - R(f^*)] + \eta_t^2\left(\frac{4CX}{\lambda} + \sigma^2\right) . \tag{6.46}$$

where we also have applied the approximation budget condition $\epsilon_t = \eta_t^2$. We use the relation in (6.46) to construct a martingale difference sequence. In particular, define the nonnegative stochastic processes α_t and β_t as

$$\alpha_t = \|f_t - f^*\|_{\mathcal{H}}^2 + \left(\frac{4CX}{\lambda} + \sigma^2\right)\sum_{u=t}^{\infty}\eta_u^2 , \qquad \beta_t = 2\eta_t[R(f_t) - R(f^*)] \tag{6.47}$$

Observe that α_t is finite almost surely, since $\sum_{u=t}^{\infty}\eta_u^2 \leq \sum_{u=0}^{\infty}\eta_u^2$. Given the definitions of α_t and β_t in (6.47), we may write

$$\mathbb{E}\left[\alpha_{t+1} \,\middle|\, \mathcal{F}_t\right] \leq \alpha_t - \beta_t , \tag{6.48}$$

together with the fact that α_t and β_t are nonnegative, whereby the conditions of the Supermartingale Convergence Theorem [183] are satisfied. Therefore, we obtain that (i) α_t has a finite limit almost surely; and (ii) the series $\sum_{t=1}^{\infty}\beta_t < \infty$ is almost surely finite. The later result, taken together with the non-summability of the step-sizes stated in (6.43), implies that the almost surely the limit infimum of $R(f_t) - R(f^*)$ is null, i.e.

$$\liminf_{t\to\infty} R(f_t) - R(f^*) = 0 \qquad \text{a.s.} \tag{6.49}$$

Now, using the consequence of the Supermartingale Convergence Theorem, α_t almost surely has a limit. Observe that the sum $\sum_{u=t}^{\infty}\eta_u^2$ is a deterministic quantity whose limit is null (we sum over less and less terms over time, asymptotically summing over zero terms). Taken

with the finiteness of the limit of α_t, we conclude

$$\lim_{t\to\infty} \|f_t - f^*\|_{\mathcal{H}}^2 = 0 \qquad \text{a.s.} \tag{6.50}$$

■

The result in Theorem 5 states that when a diminishing algorithm step-size is chosen as, e.g. $\eta_t = \mathcal{O}(1/t)$, and the approximation budget that dictates the size of the sparse stochastic subspaces onto which the iterates are projected is selected as $\epsilon_t = \eta_t^2$, we obtain exact convergence to the optimizer of the regularized expected risk in (6.1). However, in obtaining exact convergence behavior, we require the approximation budget to approach null asymptotically, which means that the model order of the resulting function sequence may grow arbitrarily, unless f^* is sparse and the magnitude of the stochastic gradient reduces sufficiently quickly, i.e., comparable to $\epsilon_t = \mathcal{O}(1/t^2)$.

If instead we consider a constant algorithm step-size $\eta_t = \eta$ and the approximation budget $\epsilon_t = \epsilon$ is chosen as a constant which satisfies $\epsilon_t = \epsilon = \mathcal{O}(\eta^{3/2})$, we obtain that the iterates converge in infimum to a neighborhood of the optimum, as we state next.

Theorem 6 *Denote $\{f_t\}$ as the sequence generated by Algorithm 6 with $f_0 = 0$, and denote f^* as the minimizer of the regularized expected risk stated in* (6.1)*. Let Assumptions 14-17 hold, and given regularizer $\lambda > 0$, suppose a constant algorithm step-size $\eta_t = \eta$ is chosen such that $\eta < 1/\lambda$, and the sparse approximation budget satisfies $\epsilon = K\eta^{3/2} = \mathcal{O}(\eta^{3/2})$, where K is a positive scaler. Then the algorithm converges to a neighborhood almost surely as*

$$\liminf_{t\to\infty} \|f_t - f^*\|_{\mathcal{H}} \leq \frac{\sqrt{\eta}}{\lambda}\Big(K + \sqrt{K^2 + \lambda\sigma^2}\Big) = \mathcal{O}(\sqrt{\eta}) \qquad \textit{a.s.} \tag{6.51}$$

Proof: The use of the regularizing term $(\lambda/2)\|f\|_{\mathcal{H}}^2$ in (6.1) implies that the regularized expected risk is λ-strongly convex with respect to $f \in \mathcal{H}$, which allows us to write

$$\frac{\lambda}{2}\|f_t - f^*\|_{\mathcal{H}}^2 \leq R(f_t) - R(f^*) \tag{6.52}$$

Substituting the relation (6.52) into the second term on the right-hand side of the expected descent relation stated in Lemma 7, with constant step-size $\eta_t = \eta$ and approximation budget $\epsilon_t = \epsilon$, yields

$$\mathbb{E}\left[\|f_{t+1} - f^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq (1 - \eta\lambda)\|f_t - f^*\|_{\mathcal{H}}^2 + 2\epsilon\|f_t - f^*\|_{\mathcal{H}} + \eta^2\sigma^2 \,. \tag{6.53}$$

We use the expression in (6.53) to construct a stopping stochastic process, which tracks the suboptimality of $\|f_t - f^*\|_{\mathcal{H}}^2$ until it reaches a specific threshold. In doing so, we obtain convergence to a neighborhood. We aim to define a stochastic process δ_t that qualifies

as a supermartingale, i.e. $\mathbb{E}\left[\delta_{t+1} \,\middle|\, \mathcal{F}_t\right] \leq \delta_t$. To do so, consider (6.53) and solve for the appropriate threshold by analyzing when the following holds true

$$\mathbb{E}\left[\|f_{t+1} - f^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq (1-\eta\lambda)\|f_t - f^*\|_{\mathcal{H}}^2 + 2\epsilon\|f_t - f^*\|_{\mathcal{H}} + \eta^2\sigma^2 \leq \|f_t - f^*\|_{\mathcal{H}}^2 \,. \quad (6.54)$$

Re-arrange the above expression to obtain the sufficient condition

$$-\eta\lambda\|f_t - f^*\|_{\mathcal{H}}^2 + 2\epsilon\|f_t - f^*\|_{\mathcal{H}} + \eta^2\sigma^2 \leq 0 \,. \quad (6.55)$$

Observe that (6.55) defines a quadratic polynomial in $\|f_t - f^*\|_{\mathcal{H}}$, which, using the quadratic formula, has roots

$$\|f_t - f^*\|_{\mathcal{H}} = \frac{-2\epsilon \pm \sqrt{4\epsilon^2 - (-4\lambda\eta)(\eta^2\sigma^2)}}{-2\lambda\eta} = \frac{\epsilon \pm \sqrt{\epsilon^2 + \lambda\eta^3\sigma^2}}{\lambda\eta} \quad (6.56)$$

The quadratic polynomnial defined by (6.55) opens downward, and $\|f_t - f^*\|_{\mathcal{H}} \geq 0$, so we focus on the positive root, substituting the approximation budget selection $\epsilon = K\eta^{3/2}$ to define the radius of convergence as

$$\Delta := \frac{\epsilon + \sqrt{\epsilon^2 + \lambda\eta^3\sigma^2}}{\lambda\eta} = \frac{\sqrt{\eta}}{\lambda}\Big(K + \sqrt{K^2 + \lambda\sigma^2}\Big) \quad (6.57)$$

The definition (6.57) allows us to construct a stopping process. In particular, define the stochastic process δ_t as

$$\delta_t = \|f_t - f^*\|_{\mathcal{H}} \mathbb{1}\Big\{ \min_{u \leq t} -\eta\lambda\|f_u - f^*\|_{\mathcal{H}}^2 + 2\epsilon\|f_u - f^*\|_{\mathcal{H}} + \eta^2\sigma^2 > \Delta \Big\} \quad (6.58)$$

where $\mathbb{1}\{E\}$ denotes the indicator process of event $E \in \mathcal{F}_t$. Note that $\delta_t \geq 0$ for all t, since both $\|f_t - f^*\|_{\mathcal{H}}$ and the indicator function are nonnegative. Observe that, given the definition (6.58), either $\min_{u \leq t} -\eta\lambda\|f_u - f^*\|_{\mathcal{H}}^2 + 2\epsilon\|f_u - f^*\|_{\mathcal{H}} + \eta^2\sigma^2 > \Delta$ holds, in which case we may compute the square root of the condition in (6.54) to write

$$\mathbb{E}[\delta_{t+1} \,|\, \mathcal{F}_t] \leq \delta_t \quad (6.59)$$

Alternatively, $\min_{u \leq t} -\eta\lambda\|f_u - f^*\|_{\mathcal{H}}^2 + 2\epsilon\|f_u - f^*\|_{\mathcal{H}} + \eta^2\sigma^2 \leq \Delta$, in which case the indicator function is null for all subsequent times, due to the use of the minimum inside the indicator in the definition of (6.58). Thus in either case, (6.59) holds, which implies that δ_t converges almost surely to null, which, as a consequence we obtain the fact that either $\lim_{t\to\infty} \|f_t - f^*\|_{\mathcal{H}} - \Delta = 0$ or the indicator function is null for large t, i.e. $\lim_{t\to\infty} \mathbb{1}\{\min_{u \leq t} -\eta\lambda\|f_u -$

$f^*\|_{\mathcal{H}}^2 + 2\epsilon\|f_u - f^*\|_{\mathcal{H}} + \eta^2\sigma^2 > \Delta\} = 0$ almost surely. Therefore, we obtain that

$$\liminf_{t\to\infty} \|f_t - f^*\|_{\mathcal{H}} \leq \Delta = \frac{\sqrt{\eta}}{\lambda}\Big(K + \sqrt{K^2 + \lambda\sigma^2}\Big) \qquad \text{a.s.} \tag{6.60}$$

which is as stated in Theorem 6.

■

Theorem 6 states that when a sufficiently small constant step-size is used together with a bias tolerance induced by sparsification chosen as $\epsilon = \mathcal{O}(\eta^{3/2})$, Algorithm 6 converges in infimum to a neighborhood of the optimum which depends on the chosen step-size, the parsimony constant K which scales the approximation budget ϵ, the regularization parameter λ, as well as the variance of the stochastic gradient σ^2. This result again is typical of convergence results in stochastic gradient methods. However, the use of a constant learning rate allows use to guarantee the model order of the resulting function sequence is always bounded, as we establish in the following subsection.

6.3.2 Model Order Control

In this subsection, we establish that the sequence of functions $\{f_t\}$ generated by Algorithm 6, when a constant algorithm step-size is selected, is parameterized by a kernel dictionary which is guaranteed to have finitely many elements, i.e., its the model order remains bounded. We obtain that the worst-case bound on the model order of f_t is depends by the topological properties of the feature space $\mathcal{X}$, the Lipschitz constant of the instantaneous loss, and the radius of convergence $\Delta = (\sqrt{\eta}/\lambda)(K + \sqrt{K^2 + \lambda\sigma^2})$ defined in Theorem 6. To do so, we first require a lemma which allows us to relate the stopping criterion of our sparsification procedure to a Hilbert subspace distance.

Lemma 6 *Define the distance of an arbitrary feature vector* $\mathbf{x}$ *evaluated by the feature transformation* $\phi(\mathbf{x}) = \kappa(\mathbf{x}, \cdot)$ *to* $\mathcal{H}_{\mathbf{D}} = span\{\kappa(\mathbf{d}_n, \cdot)\}_{n=1}^{M}$*, the subspace of the Hilbert space spanned by a dictionary* $\mathbf{D}$ *of size* M*, as*

$$dist(\kappa(\mathbf{x}, \cdot), \mathcal{H}_{\mathbf{D}}) = \min_{f \in \mathcal{H}_{\mathbf{D}}} \|\kappa(\mathbf{x}, \cdot) - \mathbf{v}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\|_{\mathcal{H}} \,. \tag{6.61}$$

This set distance simplifies to following least-squares projection when $\mathbf{D} \in \mathbb{R}^{p\times M}$ *is fixed*

$$dist(\kappa(\mathbf{x}, \cdot), \mathcal{H}_{\mathbf{D}}) = \Big\|\kappa(\mathbf{x}, \cdot) - [\mathbf{K}_{\mathbf{D},\mathbf{D}}^{-1}\boldsymbol{\kappa}_{\mathbf{D}}(\mathbf{x})]^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\Big\|_{\mathcal{H}} \,. \tag{6.62}$$

Proof: The distance to the subspace $\mathcal{H}_{\mathbf{D}}$ is defined as

$$\text{dist}(\kappa(\mathbf{x}, \cdot), \mathcal{H}_{\mathbf{D}_t}) = \min_{f \in \mathcal{H}_{\mathbf{D}}} \|\kappa(\mathbf{x}, \cdot) - \mathbf{v}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\|_{\mathcal{H}} = \min_{\mathbf{v} \in \mathbb{R}^M} \|\kappa(\mathbf{x}, \cdot) - \mathbf{v}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\|_{\mathcal{H}} \,, \tag{6.63}$$

where the first equality comes from the fact that the dictionary $\mathbf{D}$ is fixed, so $\mathbf{v} \in \mathbb{R}^M$ is the only free parameter. Now plug in the minimizing weight vector $\tilde{\mathbf{v}}^* = \mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}^{-1} \boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t)$ into (6.63) which is obtained in an analogous manner to the logic which yields (6.19) - (6.20). Doing so simplifies (6.63) to the following

$$\text{dist}(\kappa(\mathbf{x}_t, \cdot), \mathcal{H}_{\mathbf{D}_t}) = \left\| \kappa(\mathbf{x}_t, \cdot) - [\mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}^{-1} \boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t)]^T \boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot) \right\|_{\mathcal{H}} . \tag{6.64}$$

■

Theorem 7 *Denote f_t as the function sequence defined by Algorithm 6 with constant step-size $\eta_t = \eta < 1/\lambda$ and approximation budget $\epsilon = K\eta^{3/2}$ where $K > 0$ is an arbitrary positive scalar. Let M_t be the model order of f_t i.e., the number of columns of the dictionary $\mathbf{D}_t$ which parameterizes f_t. Then there exists a finite upper bound M^∞ such that, for all $t \geq 0$, the model order is always bounded as $M_t \leq M^\infty$. Consequently, the model order of the limiting function $f^\infty = \lim_t f_t$ is finite.*

Proof: The proof proceeds by the following logic. We begin by considering the model order at two arbitrary subsequent iterates of Algorithm 6, and reduce model order growth at a given time to a criterion involving the approximation error γ_{M_t+1} associated with removing the most recent feature vector $\mathbf{x}_t$, and then analyze the conditions under which this simplified criterion is not satisfied for all subsequent times, meaning that the model order does not grow beyond a certain point. To do so, we prove that this quantity reduces to a weighted set distance to the Hilbert subspace $\mathcal{H}_{\mathbf{D}_t}$ defined by dictionary $\mathbf{D}_t$, and thus we are able to invoke point-set topological properties of the compact feature space $\mathcal{X}$, specifically, its packing number, which guarantee that the number of dictionary elements remains finite, in a manner similar to the proof of Theorem 3.1 in [59].

Consider the model order of the function iterates f_t and f_{t+1} generated by Algorithm 6 denoted by M_t and M_{t+1}, respectively, at two arbitrary subsequent times t and $t+1$. Assume a constant algorithm step-size η has been chosen such that $\eta < 1/\lambda$ and the approximation budget ϵ satisfies $\epsilon = K\eta^{3/2}$ for some positive scalar $K > 0$. Suppose the model order of the function f_{t+1} is less than or equal to that of f_t, i.e. $M_{t+1} \leq M_t$. This relation holds when the stopping criterion of KOMP (Algorithm 5), stated as $\min_{j=1,\ldots,M_t+1} \gamma_j > \epsilon$, *is not* satisfied for the kernel dictionary matrix with the newest sample point $\mathbf{x}_t$ appended: $\tilde{\mathbf{D}}_{t+1} = [\mathbf{D}_t; \mathbf{x}_t]$ [cf. (6.18)], which is of size $M_t + 1$. Thus, the negation of the termination condition of Algorithm 5 must hold for this case, stated as

$$\min_{j=1,\ldots,M_t+1} \gamma_j \leq \epsilon . \tag{6.65}$$

Observe that the left-hand side of (6.65) lower bounds the approximation error γ_{M_t+1}

of removing the most recent feature vector $\mathbf{x}_t$ due to the minimization over j, that is, $\min_{j=1,\ldots,M_t+1} \gamma_j \leq \gamma_{M_t+1}$. Consequently, if $\gamma_{M_t+1} \leq \epsilon$, then (6.65) holds and the model order does not grow. Thus it suffices to consider γ_{M_t+1}.

The definition of γ_{M_t+1} with the substitution of $\tilde{f}_{t+1}$ in (6.17) allows us to write

$$\begin{aligned}\gamma_{M_t+1} &= \min_{\mathbf{u}\in\mathbb{R}^{M_t}} \left\|(1-\eta\lambda)f_t - \eta\ell'(f_t(\mathbf{x}_t),\mathbf{y}_t)\kappa(\mathbf{x}_t,\cdot) - \sum_{k\in\mathcal{I}\setminus\{M_t+1\}} u_k\kappa(\mathbf{d}_k,\cdot)\right\|_{\mathcal{H}} \qquad (6.66)\\ &= \min_{\mathbf{u}\in\mathbb{R}^{M_t}} \left\|(1-\eta\lambda)\sum_{k\in\mathcal{I}\setminus\{M_t+1\}} w_k\kappa(\mathbf{d}_k,\cdot) - \eta\ell'(f_t(\mathbf{x}_t),\mathbf{y}_t)\kappa(\mathbf{x}_t,\cdot) - \sum_{k\in\mathcal{I}\setminus\{M_t+1\}} u_k\kappa(\mathbf{d}_k,\cdot)\right\|_{\mathcal{H}},\end{aligned}$$

where we denote the k^{th} column of $\mathbf{D}_t$ as $\mathbf{d}_k$. The minimal error is achieved by considering the square of the expression inside the minimization and expanding terms to obtain

$$\begin{aligned}&\left\|(1-\eta\lambda)\sum_{k\in\mathcal{I}\setminus\{M_t+1\}} w_k\kappa(\mathbf{d}_k,\cdot) - \eta\ell'(f_t(\mathbf{x}_t),\mathbf{y}_t)\kappa(\mathbf{x}_t,\cdot) - \sum_{k\in\mathcal{I}\setminus\{M_t+1\}} u_k\kappa(\mathbf{d}_k,\cdot)\right\|_{\mathcal{H}}^2 \qquad (6.67)\\ &\quad= (1-\eta\lambda)^2\mathbf{w}^T\mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}\mathbf{w} + \eta^2\ell'(f_t(\mathbf{x}_t),\mathbf{y}_t)^2\kappa(\mathbf{x}_t,\mathbf{x}_t) + \mathbf{u}^T\mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}\mathbf{u}\\ &\qquad -2(1-\eta\lambda)\eta\ell'(f_t(\mathbf{x}_t),\mathbf{y}_t)^2\mathbf{w}^T\boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t) + 2\eta^2\ell'(f_t(\mathbf{x}_t),\mathbf{y}_t)\mathbf{u}^T\boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t) - 2(1-\eta\lambda)\mathbf{w}^T\mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}\mathbf{u}.\end{aligned}$$

To obtain the minimum, we compute the stationary solution of (6.67) with respect to $\mathbf{u}\in\mathbb{R}^{M_t}$ and solve for the minimizing $\tilde{\mathbf{u}}^*$, which in a manner similar to the logic in (6.19) - (6.20), is given as

$$\tilde{\mathbf{u}}^* = (1-\eta\lambda)\mathbf{w} - \eta\ell'(f_t(\mathbf{x}_t),\mathbf{y}_t)\mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}^{-1}\boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t)\,. \qquad (6.68)$$

Plug $\tilde{\mathbf{u}}^*$ in (6.68) into the expression in (6.66) and using the short-hand notation $f_t(\cdot) = \mathbf{w}^T\boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot)$ and $\sum_k u_k\kappa(\mathbf{d}_k,\cdot) = \mathbf{u}^T\boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot)$. Doing so simplifies (6.66) to

$$\begin{aligned}&\left\|(1-\eta\lambda)\mathbf{w}^T\boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot) - \eta\ell'(f_t(\mathbf{x}_t),\mathbf{y}_t)\kappa(\mathbf{x}_t,\cdot) - \mathbf{u}^T\boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot)\right\|_{\mathcal{H}} \qquad (6.69)\\ &\quad= \Big\|(1-\eta\lambda)\mathbf{w}^T\boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot) - \eta\ell'(f_t(\mathbf{x}_t),\mathbf{y}_t)\kappa(\mathbf{x}_t,\cdot)\\ &\qquad - \left[(1-\eta\lambda)\mathbf{w} - \eta\ell'(f_t(\mathbf{x}_t),\mathbf{y}_t)\mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}^{-1}\boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t)\right]^T\boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot)\Big\|_{\mathcal{H}}.\end{aligned}$$

The above expression may be simplified by cancelling like terms $(1-\eta\lambda)\mathbf{w}^T\boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot)$ and pulling out a common factor of $\eta|\ell'(f_t(\mathbf{x}_t),\mathbf{y}_t)|$ outside the norm as

$$\begin{aligned}&\left\| - \eta\ell'(f_t(\mathbf{x}_t),\mathbf{y}_t)\kappa(\mathbf{x}_t,\cdot) - \eta\ell'(f_t(\mathbf{x}_t),\mathbf{y}_t)[\mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}^{-1}\boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t)]^T\boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot)\right\|_{\mathcal{H}}\\ &\qquad = \eta|\ell'(f_t(\mathbf{x}_t),\mathbf{y}_t)|\left\|\kappa(\mathbf{x}_t,\cdot) - [\mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}^{-1}\boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t)]^T\boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot)\right\|_{\mathcal{H}}. \qquad (6.70)\end{aligned}$$

Notice that the right-hand side of (6.70) may be identified as the distance to the subspace

$\mathcal{H}_{\mathbf{D}_t}$ in (6.64) defined in Lemma 6 scaled by a factor of $\eta|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)|$. Using this identification, we transform the sufficient condition for the stopping criterion of KOMP to be violated, stated as $\gamma_{M_t+1} \leq \epsilon$, into a criterion on $\text{dist}(\kappa(\mathbf{x}_t, \cdot), \mathcal{H}_{\mathbf{D}_t})$, the subspace distance of $\kappa(\mathbf{x}_t, \cdot)$ to the span of kernel evaluations of the current dictionary $\mathcal{H}_{\mathbf{D}_t}$. Substituting the definition (6.64) into $\gamma_{M_t+1} \leq \epsilon$ and dividing both sides by $\eta|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)|$ yields

$$\text{dist}(\kappa(\mathbf{x}_t, \cdot), \mathcal{H}_{\mathbf{D}_t}) \leq \frac{\epsilon}{\eta|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)|} \, . \tag{6.71}$$

Now use the approximation budget selection in terms of the learning rate η as $\epsilon = K\eta^{3/2}$. Furthermore, the C-Lipschitz continuity of ℓ [cf. (6.27)] in Assumption 15 allows us to bound the instantaneous gradient by this same constant. Inverting this expression yields $1/|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)| \geq 1/C$. Substituting in this lower bound and selection of ϵ, we obtain that if

$$\text{dist}(\kappa(\mathbf{x}_t, \cdot), \mathcal{H}_{\mathbf{D}_t}) \leq \frac{K\sqrt{\eta}}{C} \tag{6.72}$$

holds, then (6.71) holds, and consequently $M_{t+1} \leq M_t$. The contrapositive of the aforementioned logic tells us that growth in the model order ($M_{t+1} = M_t + 1$) implies that the condition

$$\text{dist}(\kappa(\mathbf{x}_t, \cdot), \mathcal{H}_{\mathbf{D}_t}) > \frac{K\sqrt{\eta}}{C} \tag{6.73}$$

holds. Therefore, each time a new point is added to the model, the corresponding kernel function is guaranteed to be at least a distance of $\frac{K\sqrt{\eta}}{C}$ from every other kernel function in the current model, i.e., for distinct dictionary points $\mathbf{d}_k$ and $\mathbf{d}_j$ for $j, k \in \{1, \ldots, M_t\}$, $\|\phi(\mathbf{d}_j) - \phi(\mathbf{d}_k)\|_2 > \frac{K\sqrt{\eta}}{C}$. We shall now proceed in a manner similar to the proof of Theorem 3.1 in [59]. Since $\mathcal{X}$ is compact and κ is continuous, the range $\phi(\mathcal{X})$ (where $\phi(\mathbf{x}) = \kappa(\mathbf{x}, \cdot)$ for $\mathbf{x} \in \mathcal{X}$) of the kernel transformation of feature space $\mathcal{X}$ is compact. Therefore, the number of balls of radius δ (here, $\delta = \frac{K\sqrt{\eta}}{C}$) needed to cover the set $\phi(\mathcal{X})$ is finite (see, e.g., [7]). Therefore, for some finite M^∞, if $M_t = M^\infty$, the left-hand side of (6.72) holds, which implies (6.65) is true for all t. We conclude that $M_t \leq M^\infty$ for all t. ■

The number of kernel dictionary elements in the function sequence f_t generated by Algorithm 6 is in the worst-case determined by the packing number of the kernel transformation of the feature space $\phi(\mathcal{X}) = \kappa(\mathcal{X}, \cdot)$, as shown in the proof of Theorem 7. Moreover, the online sparsification procedure induced by KOMP reduces to a condition on the scale of the packing number of $\phi(\mathcal{X})$ as stated in (6.73). Specifically, as the radius $\frac{K\sqrt{\eta}}{C}$ increases, the packing number of the kernelized feature space decreases, and hence the required model order to fill $\phi(\mathcal{X})$ decreases. This radius depends on the constant K which scales the approx-

Table 6.1: Summary of convergence results for different parameter selections.

	Diminishing	Constant
Step-size/Learning rate	$\eta_t = \mathcal{O}(1/t)$	$\eta_t = \eta > 0$
Sparse Approximation Budget	$\epsilon_t = \eta_t^2$	$\epsilon = \mathcal{O}(\eta^{3/2})$
Regularization Condition	$\eta_t < 1/\lambda$	$\eta < 1/\lambda$
Convergence Result	$f_t \to f^*$ a.s.	$\liminf_t \|f_t - f^*\| = \mathcal{O}(\sqrt{\eta})$ a.s.
Model Order Guarantee	None	Finite

imation budget selection η, the learning rate η, and the constant C bounding the gradient of the regularized instantaneous loss.

We have established that Algorithm 6 yields convergent behavior for the problem (6.1) in both diminishing and constant step-size regimes. When the learning rate η_t satisfies $\eta_t < 1/\lambda$, where $\lambda > 0$ is the regularization parameter, and is attenuating such that $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$, i.e., $\eta_t = \mathcal{O}(1/t)$, the approximation budget ϵ_t of Algorithm 5 must satisfy $\epsilon_t = \eta_t^2$ [cf. (6.43)]. Practically speaking, this means that asymptotically the iterates generated by Algorithm 6 may have a very large model order in the diminishing step-size regime, since the approximation budget is vanishing as $\epsilon_t = \mathcal{O}(1/t^2)$. On the other hand, when a constant algorithm step-size $\eta_t = \eta$ is chosen to satisfy $\eta < 1/\lambda$, then we only require the constant approximation budget $\epsilon_t = \epsilon$ to satisfy $\epsilon = \mathcal{O}(\eta^{3/2})$. This means that in the constant learning rate regime, we obtain a function sequence which converges to a neighborhood of the optimal f^* defined by (6.1) and is guaranteed to have a finite model order. These results are summarized in Table 8.1.

Remark 6 *(Sparsity of f^*)* Algorithm 6 provides a method to avoid keeping an unnecessarily large number of kernel dictionary elements along the convergence path towards f^* [cf. (6.1)], solving the classic scalability problem of kernel methods in stochastic programming. However, if the optimal function admits a low dimensional representation $|\mathcal{I}| << \infty$, then in addition to extracting memory efficient instantaneous iterates, POLK will obtain the optimal function exactly. In Section 6.4, we illustrate this property via a multi-class classification problem where the data is generated from Gaussian mixture models.

6.4 Experiments with Efficient Nonparametric Methods

In this section, we evaluate POLK by considering its performance on two supervised learning tasks trained for three streaming data sets. The specific tasks we consider are those of (a)

training a multi-class kernel logistic regressor (KLR), and (b) training a multi-class kernel support vector machine (KSVM). The three data sets we use are (i) `multidist`, a synthetic data set we constructed using two-dimensional Gaussian mixture models; (ii) `mnist`, the MNIST handwritten digits [105]; and (iii) `brodatz`, image textures drawn from a subset of the Brodatz texture database [37]. Where possible, we compare our technique with competing methods. Specifically, for the online support vector machine case, we compare with budgeted stochastic gradient descent (BSGD) [207], which requires a maximum model order *a priori*. For off-line (batch) KLR, we compare with the import vector machine (IVM) [227], a sparse second-order method. We also compare with the batch techniques of LIBSVM [44], applicable to KSVM only, and an L-BFGS solver [143].

6.4.1 Tasks

The tasks we consider are those of multi-class classification, which is a problem that admits approaches based on probabilistic and geometric criteria. In what follows, we use $\mathbf{x}_n \in \mathcal{X} \subset \mathbb{R}^p$ to denote the n^{th} feature vector in a given data set, and $y_n \in \{1, \ldots, D\}$ to denote its corresponding label.

Multi-class Kernel Support Vector Machines (Multi-KSVM) The first task we consider is that of training a multi-class kernel support vector machine, in which the merit of a particular regressor is defined by its ability to maximize its classification margin. In particular, define a set of class-specific activation functions $f_d : \mathcal{X} \to \mathbb{R}$, and denote them jointly as $\mathbf{f} \in \mathcal{H}^D$. In Multi-KSVM, points are assigned the class label of the activation function that yields the maximum response. KSVM is trained by taking the instantaneous loss ℓ to be the multi-class hinge function which defines the margin separating hyperplane in the kernelized feature space, i.e.,

$$\ell(\mathbf{f}, \mathbf{x}_n, y_n) = \max(0, 1 + f_r(\mathbf{x}_n) - f_{y_n}(\mathbf{x}_n)) + \lambda \sum_{d'=1}^{D} \|f_{d'}\|_{\mathcal{H}}^2 \,, \tag{6.74}$$

where $r = \operatorname{argmax}_{d' \neq y} f_{d'}(\mathbf{x})$. Further details may be found in [133].

Multi-class Kernel Logistic Regression (Multi-KLR) The second task we consider is that of kernel logistic regression, wherein, instead of maximizing the margin which separates sample points in the kernelized feature space, we instead adopt a probabilistic model on the odds ratio that a sample point has a specific label relative to all others. Using the same notation as above for the class-specific activation functions, we adopt the probabilistic model:

$$P(y = d \,|\, \mathbf{x}) \triangleq \frac{\exp(f_d(\mathbf{x}))}{\sum_{d'} \exp(f_{d'}(\mathbf{x}))}. \tag{6.75}$$

(a) `multidist` data. (b) `brodatz` example texture (c) `mnist` examples.

Figure 6.1: Visualizations of the data sets used in experiments.

which models the odds ratio of a given sample point being in class c versus all others. We use the negative log likelihood pertaining to the above model as the instantaneous loss (see, e.g., [133]), i.e.,

$$\begin{aligned}\ell(\mathbf{f}, \mathbf{x}_n, y_n) &= -\log P(y = y_n|\mathbf{x}_n) + \frac{\lambda}{2}\sum_d \|f_d\|_{\mathcal{H}}^2 \\ &= \log\left(\sum_{d'} \exp(f_{d'}(\mathbf{x}_n))\right) - f_{y_n}(\mathbf{x}_n) + \frac{\lambda}{2}\sum_d \|f_d\|_{\mathcal{H}}^2 \,. \end{aligned} \tag{6.76}$$

Observe that the loss (6.76) substituted into the empirical risk minimization problem in Example 4 is its generalization to multi-class classification. For a given set of activation functions, the classification decision $\tilde{d}$ for $\mathbf{x}$ is given by the class that yields the maximum likelihood, i.e., $\tilde{d} = \text{argmax}_{d\in\{1,\ldots,D\}} f_d(\mathbf{x})$.

6.4.2 Data Sets

We evaluate Algorithm 6 for the Multi-KLR and Multi-KSVM tasks described above using the `multidist`, `mnist`, `brodatz` data sets.

`multidist`

In a manner similar to [227], we generate the `multidist` data set using a set of Gaussian mixture models. The data set consists $N = 5000$ feature-label pairs for training and 2500 for testing. Each label y_n was drawn uniformly at random from the label set. The corresponding feature vector $\mathbf{x}_n \in \mathbb{R}^p$ was then drawn from a planar ($p = 2$), equitably-weighted Gaussian mixture model, i.e., $\mathbf{x}\,|\,y \;\sim\; (1/3)\sum_{j=1}^{3}\mathcal{N}(\boldsymbol{\mu}_{y,j}, \sigma_{y,j}^2\mathbf{I})$ where $\sigma_{y,j}^2 = 0.2$ for all values of y and j. The means $\boldsymbol{\mu}_{y,j}$ are themselves realizations of their own Gaussian distribution with class-dependent parameters, i.e., $\boldsymbol{\mu}_{y,j} \sim \mathcal{N}(\boldsymbol{\theta}_y, \sigma_y^2\mathbf{I})$, where $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_D\}$ are equitably spaced around the unit circle, one for each class label, and $\sigma_y^2 = 1.0$. We fix the number of classes $D = 5$, meaning that the feature distribution has, in total, 15 distinct modes. The data points are plotted in Figure 6.1(a).

`mnist`

The `mnist` data set we use is the popular MNIST data set [105], which consists of $N = 60000$ feature-label pairs for training and 10000 for testing. Feature vectors are $p = 784$-dimensional, where each dimension captures a single grayscale pixel value (scaled to lie within the unit interval) that corresponds to a unique location in a 28-pixel-by-28-pixel image of a cropped, handwritten digit. Labels indicate which digit is written, i.e., there are $C = 10$ classes total, corresponding to digits $0, \ldots, 9$ – examples are given in Figure 6.1(c).

`brodatz`

We generated the `brodatz` data set using a subset of the images provided in [37]. Specifically, we used 13 texture images (i.e., $C = 13$), and from them generated a set of 256 textons [107]. Next, for each overlapping patch of size 24-pixels-by-24-pixels within these images, we took the feature to be the associated $p = 256$-dimensional texton histogram. The corresponding label was given by the index of the image from which the patch was selected. When then randomly selected $N = 10000$ feature-label pairs for training and 5000 for testing. An example texture image can be seen in Figure 6.1(b).

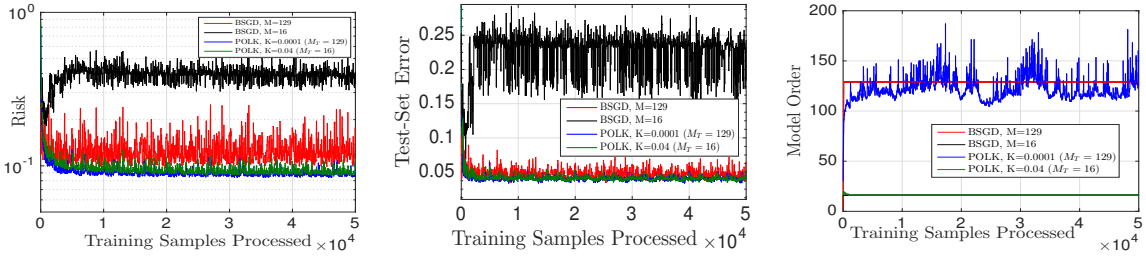
6.4.3 Results

For each task and data set described above, we implemented POLK (Algorithm 6) along with the competing methods described at the beginning of the section. For some of the tasks, only a subset of the competing methods are applicable, and in some cases such as online logistic regression, none are. Here, we shall describe the details of each experimental setting and the corresponding results.

`multidist` Results

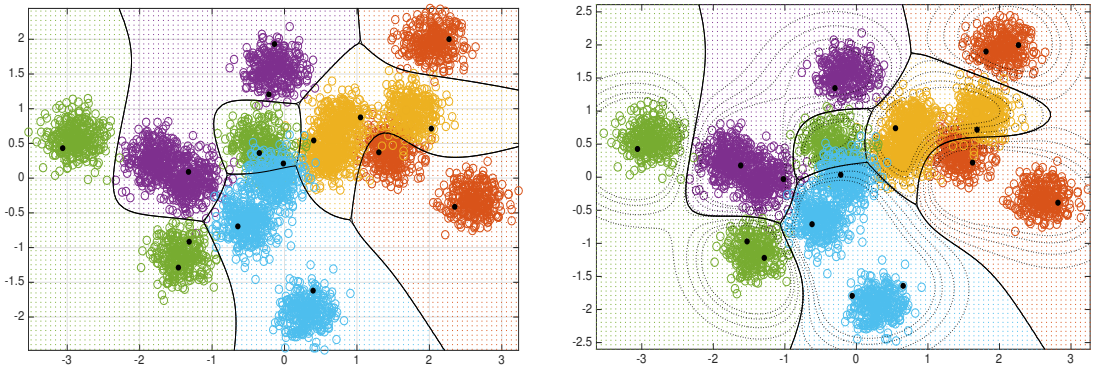
Due to the small size of our synthetic `multidist` data set, we were able to generate results for the Multi-KSVM task using each of the methods specified earlier except for IVM. For POLK, we used the following specific parameter values: we select the Gaussian/RBF kernel with bandwidth $\tilde{\sigma}^2 = 0.6$, constant learning rate $\eta = 6.0$, parsimony constant $K \in \{10^{-4}, 0.04\}$, and regularization constant $\lambda = 10^{-6}$. Further, we processed streaming samples in mini-batches of size 32. For BSGD, we used the same $\tilde{\sigma}^2$ and λ, but achieved the best results with smaller constant learning rate $\eta = 1.0$ (perhaps due, in part, to the fact that BSGD does not support mini-batching). In order to compare with POLK, we set BSGD's pre-specified model orders to be $\{16, 129\}$, i.e., the steady-state model orders of POLK parameterized with the values of K specified above.

In Figure 6.3 we plot the empirical results of this experiment for POLK and BSGD, and observe that POLK outperforms the competing method by an order of magnitude in terms of objective evaluation (Fig. 6.2(a)) and test-set error rate (Fig 6.2(b)). Moreover,



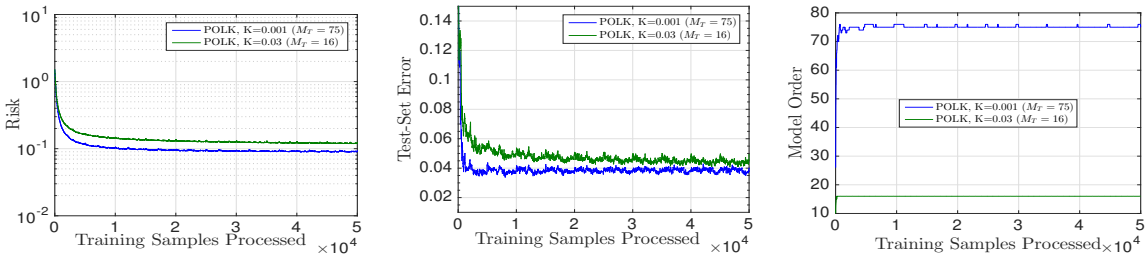
(a) Empirical risk $R(\mathbf{f}_t)$. (b) Error rate (c) Model order M_t.

Figure 6.2: Comparison of POLK and BSGD on the `multidist` data set for the Multi-KSVM task. Observe that POLK achieves lower risk and higher accuracy for a fixed model order. More accurate POLK regressors require use of a smaller parsimony constant K, although we observe diminishing benefit of increasing the model order via reducing K.



(a) $\mathbf{f}_T$ (hinge loss). (b) $\mathbf{f}_T$ (logistic loss)

Figure 6.3: Visualization of the decision surfaces yielded by POLK for the Multi-KSVM and Multi-Logistic tasks on the `multidist` data set. Training examples from distinct classes are assigned a unique color. Grid colors represent the classification decision by $\mathbf{f}_T$. Bold black dots are kernel dictionary elements, which concentrate at the modes of the joint data distribution. Solid lines are drawn to denote class label boundaries, and additional dashed lines in 6.3(b) are drawn to denote confidence intervals.



(a) Empirical risk $R(\mathbf{f}_t)$. (b) Error rate (c) Model order M_t.

Figure 6.4: Empirical behavior of the POLK algorithm applied to the `multidist` data set for the Multi-Logistic task. Observe that the algorithm converges to a low risk value ($R(f_t) < 10^{-1}$) and achieves test set accuracy between 4% and 5% depending on choice of parsimony constant K, which respectively corresponds to a model order between 75 and 16.

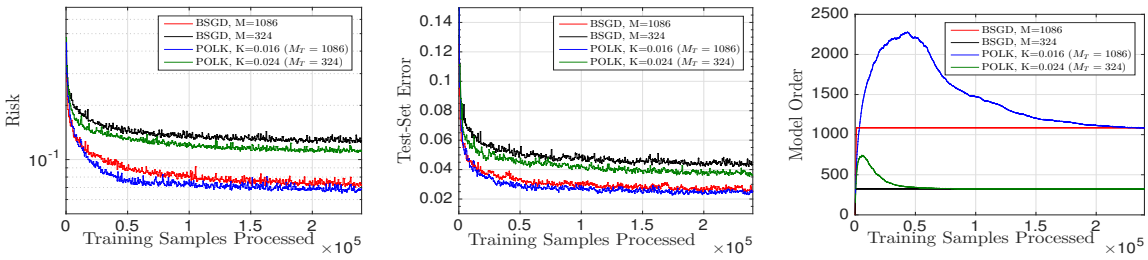
because the marginal feature density of `multidist` contains 15 modes, the optimal model order is $M^* = 15$, which is approximately *learned* by POLK for $K = 0.04$ (i.e., $M_T = 16$) (Fig. 6.2(c)). The corresponding trial of BSGD, on the other hand, initialized with this parameter, does not converge. Observe that for this task POLK exhibits a state of the art trade off between test set accuracy and number of samples processed – reaching below 4% error after only 1249 samples. The final decision surface $\mathbf{f}_T$ of this trial of POLK is shown in Fig. 6.3(a), where it can be seen that the selected kernel dictionary elements concentrate near the modes of the marginal feature density.

We can also see from Table 6.2 that POLK compares favorably to the batch techniques for Multi-KSVM on the `multidist` data set. It achieves approximately the same error rate as LIBSVM with significantly fewer model points (support vectors) and even outperforms our (dense) L-BFGS batch solver in terms of test-set error, while adding the ability to process data in an online fashion.

For the Multi-Logisitic task on this data set, we were able to generate results for each method except BSGD and LIBSVM, which are specifically tailored to the SVM task. For POLK, we used the following parameter values: Gaussian kernel with bandwidth $\tilde{\sigma}^2 = 0.6$, constant learning rate $\eta = 6.0$, parsimony constant $K \in \{0.001, 0.03\}$, and regularization constant $\lambda = 10^{-6}$. As in Multi-KSVM, we processed the streaming samples in mini-batches of size 32. The empirical behavior of POLK for the Multi-Logistic task can be seen in Figure 6.4 and the final decision surface is presented in Figure 6.3(b). Observe that POLK is exhibits comparable convergence to the SVM problem, but a smoother descent due to the differentiability of the multi-logistic loss. In Table 6.2 we present final accuracy and risk values on the logistic task, and note that it performs comparably, or in some cases, favorably, to the batch techniques (IVM and L-BFGS), while processing streaming data.

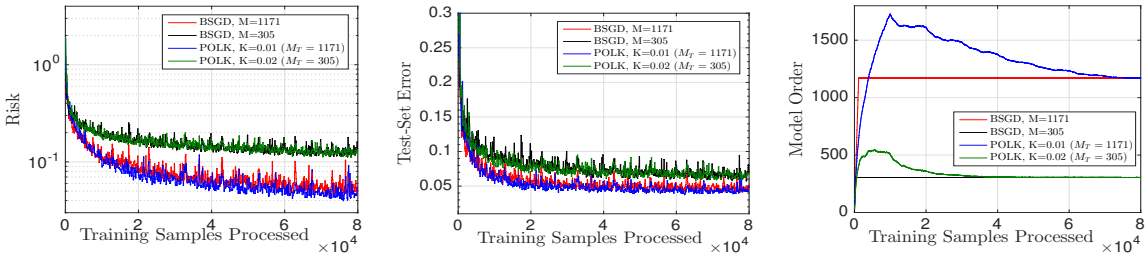
`mnist` and `brodatz` Results

By construction, the `multidist` data set above yields optimal activation functions that



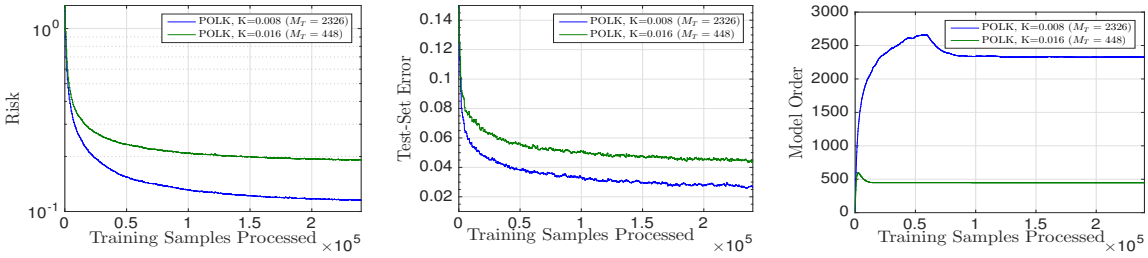
(a) Empirical risk $R(\mathbf{f}_t)$. (b) Error rate (c) Model order M_t.

Figure 6.5: Comparison of POLK and BSGD on `mnist` data set for the Multi-KSVM task. Observe that POLK achieves lower risk and higher accuracy on this task, and extracts a model order directly from the feature space that yields convergence.



(a) Empirical risk $R(\mathbf{f}_t)$. (b) Error rate (c) Model order M_t.

Figure 6.6: Comparison of POLK and BSGD on `brodatz` data set for the Multi-KSVM task. We observe that POLK behaves similarly to BSGD for this task, stabilizing at an accuracy near 96%. For this dense data domain, larger model orders are needed to achieve convergence.

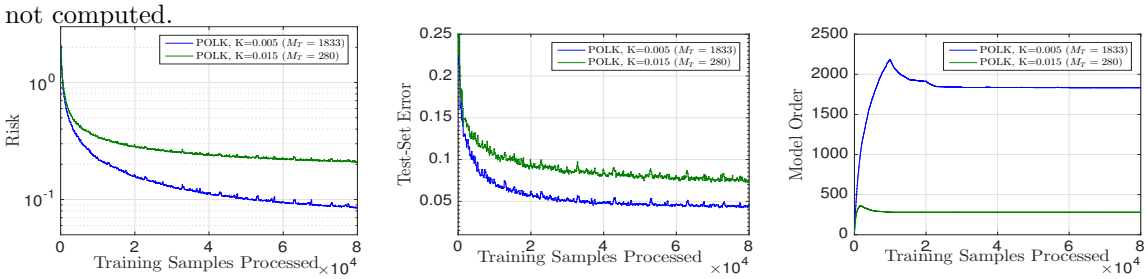


(a) Empirical risk $R(\mathbf{f}_t)$. (b) Error rate (c) Model order M_t.

Figure 6.7: Empirical behavior of the POLK algorithm applied to `mnist` data set for the Multi-Logistic task. The algorithm exhibits smoother convergence due to the differentiability of the logistic loss, and achieves asymptotic test error 2.6%. We again observe due to the dense data domain, larger model orders are needed to exhibit competitive classification performance.

	multidist	
Algorithm	Multi-KSVM (risk/error/model order)	Multi-Logistic (risk/error/model order)
LIBSVM	−/3.92/656	−/ − /−
L-BFGS	0.0854/4.08/5000	0.0854/4.04/5000
IVM	−/ − /−	0.0894/4.08/16
BSGD	0.385/21.8/16	−/ − /−
POLK	0.0919/3.98/16	0.120/4.36/16

Table 6.2: Comparison of POLK, BSGD, IVM, L-BFGS, and LIBSVM results on the `multidist` data set. Reported risk and error values for POLK and BSGD were averaged over the final 5% of processed training examples. Dashes indicate where the method could not be used to generate results because it is not defined for that task. LIBSVM is used as a baseline, but note that it uses a fundamentally different model for multi-class problems (a separate one-vs-all classifier is trained for each class, and then at test time, a majority vote is executed), and so a comparable risk value can not computed.



(a) Empirical risk $R(\mathbf{f}_t)$. (b) Error rate (c) Model order M_t.

Figure 6.8: Empirical behavior of the POLK algorithm applied to the `brodatz` data set for the Multi-Logistic task. We observe convergent behavior, and a clear trade off between higher model order and increased accuracy. Due to this data domain being a more challenging task than the `mnist` digits, we observe asymptotic test accuracy of approximately 4.4%.

are themselves sparse (i.e., $\mathbf{f}^*$ has a low model order due to the marginal feature density). Here, we analyze the performance of POLK on more realistic data sets where the optimal solutions are not sparse, i.e., where one might desire a sparse approximation. Due to the increased size and dimensionality of these data sets, we were unable to generate results for `mnist` using the batch L-BFGS technique, and unable to generate results for either data set using IVM.

For Multi-KSVM on `mnist`, we used the following parameter values for POLK: Gaussian kernel with bandwidth $\tilde{\sigma}^2 = 4.0$, constant learning rate $\eta = 24.0$, parsimony constant $K \in \{0.16, 0.24\}$, and regularization constant $\lambda = 10^{-6}$. We again processed data in mini-batches of size 32. For `brodatz`, we used identical parameters except for changing the kernel bandwith $\tilde{\sigma}^2 = 0.1$ and parsimony constant $K \in \{0.01, 0.02\}$. For BSGD, we again found $\eta = 1.0$ to yield the best results on both datasets, and pre-specified model orders of $\{324, 1086\}$ and $\{305, 1171\}$ on `mnist` and `brodatz`, respectively, for comparison to POLK.

	mnist		brodatz	
Algorithm	Multi-KSVM (risk/error/model order)	Multi-Logistic (risk/error/model order)	Multi-KSVM (risk/error/model order)	Multi-Logistic (risk/error/model order)
LIBSVM	−/1.50/16118	−/−/−	−/3.72/4777	−/−/−
L-BFGS	−/−/−	−/−/−	0.0319/4.44/10000	0.0572/4.00/10000
BSGD	0.0731/2.67/1086	−/−/−	0.0560/4.72/1171	−/−/−
POLK	0.0684/2.46/1086	0.116/2.68/2326	0.0507/4.53/1171	0.0871/4.41/1833

Table 6.3: Comparison of POLK, BSGD, IVM, L-BFGS, and LIBSVM results on the `mnist` and `brodatz` data sets. Reported risk and error values for POLK and BSGD were averaged over the final 5% of processed training examples. Dashes indicate where the method could not be used to generate results either because it is not defined for the task or because the size of the problem was too large for that data set. For these reasons, IVM was not able to generate results for these data sets on either task, and so is omitted here. LIBSVM is used as a baseline, but note that it uses a fundamentally different model for multi-class problems (1v1 + majority vote), and so a comparable risk value can not be computed.

In Figures 6.5 and 6.6 we plot the empirical results of these experiments for POLK and BSGD. We observe that POLK is able to outperform the comparable BSGD trial in terms of convergence speed and steady-state risk and test-set error. The strength of the proposed technique is further demonstrated in Table 6.3, where we can see that POLK is able to achieve test-set error within 1-2% of LIBSVM while requiring a number of support vectors (model points) that is significantly-less than LIBSVM, *while* adding the ability to process streaming data.

For the Multi-Logistic task on `mnist`, we ran POLK using a Gaussian kernel with bandwidth $\tilde{\sigma}^2 = 4.0$, constant learning rate $\eta = 24.0$, parsimony constant $K \in \{0.08, 0.16\}$, and regularization constant $\lambda = 10^{-6}$. Data was processed in mini-batches of size 32 here as well. For `brodatz`, we again change the kernel bandwidth $\tilde{\sigma}^2 = 0.1$ and used different parsimony constants $K = \{0.005, 0.015\}$. The empirical behavior of POLK on this task can be seen in Figures 6.7 and 6.8. Observe that for this task the descent is smoother due to the differentiability of the logistic loss, although the asymptotic test accuracy is lower than that of KSVM.

The overall performance is summarized in Table 6.3. Note that the only other technique that was able to generate results for this task was L-BFGS, and even there only on the `brodatz` data set, since the complexity bottleneck in the sample size for `mnist` is prohibitive for batch optimization. We see from this comparison that POLK yields a test-set error within 0.5% of the batch solution while using an order of magnitude fewer model points. Additionally, POLK is able to run *online*, with streaming data, whereas L-BFGS requires all the data points to be operated on at each step.

6.5 On the Promise and State of Memory-Efficient Kernel Methods

Over the past several years, parametric function approximation has largely dominated the machine learning landscape. Deep learning is perhaps currently the most prominent parametric paradigm [72]. One must first specify a network structure, thereby fixing the parametric representation of the function to be learned, before proceeding to determine the coefficients linking neurons in different layers. Given this parametric representation, training techniques proceed by searching over the predefined parameter space for the optimal parameter values that minimize the error between the function and observed input-output pairs. The main reason for the popularity of parametric function approximation is its success in solving practical problems, but there are other factors that have fostered their adoption. One such factor is the availability of workable, if not necessarily efficient, optimization techniques for the determination of the optimal parameter values, in the form of stochastic gradient descent and its variants. Parametric stochastic gradient descent processes training examples sequentially and has a per-iteration complexity that is linear on the number of parameters but independent of the size of the training set.

Despite the success of parametric techniques, nonparametric function approximation has the advantage of expressive power in the sense that they are allowed to select the approximating function from a more general set of functions than those that admit a parametric form chosen *a priori*. This advantage is seen, e.g., in the improved classification accuracy of (nonparametric) kernel support vector machines (SVMs) relative to (parametric) linear SVMs [62]. This is not to say that nonparametric methods are necessarily better. Neural networks, e.g., have proven to be very adept parametric representations in image classification problems [103]. However, it is nonetheless true that the better expressive power of nonparametric representations is of importance in some applications.

The importance of expressiveness notwithstanding, nonparametric approaches are relatively less popular. This is partly explained by the fact that, contrary to parametric approaches, workable algorithms for the minimization of functional costs are not as well-developed. Indeed, nonparametric models involve function representations that depend on an infinite number of parameters. This is a challenge not only because optimal function descriptions can become computationally intractable but, more importantly, because finding these optimal representations is itself intractable.

This work represents the first attempt at comprehensively addressing this intractability. In particular, we have proposed solving general convex expected risk minimization problems over a Hilbert space that defines nonparametric regression functions in a way that guarantees the model order of the learned function does not grow unnecessarily large. In doing so, we

addressed challenges (1) - (2) as follows: we considered the generalization of stochastic gradient descent to the kernelized expected risk setting and we compressed the learned decision function in a way that guarantees stochastic descent by tuning a greedy sparse approximation error criterion to the underlying optimization sequence. The result is an almost-sure convergent function sequence with moderate complexity that is able to operate in true online settings.

Indeed, our experiments have shown that POLK performs comparably to batch kernel methods in terms of accuracy, while its model complexity is reduced by orders of magnitude. Additionally, we observe state-of-the-art performance in terms of test-set accuracy relative to the number of samples processed. Such performance is key to achieving reasonable performance in many applications of interest, e.g., when learning on robotics platforms operating in unknown environments. In this case, the online nature of the problem is intrinsic and due to a lack of prior information on their operating domain [92].

On the other hand, it must be noted that POLK, and even batch kernel methods, for certain large-scale supervised learning tasks, have not met the high bar of asymptotic test set accuracy set forth by batch approaches to deep learning [103]. We believe this discrepancy is on account of the single-layer nature of the nonparametric regressor, which is tied to the choice of reproducing kernel used in our experiments. Of course, more complicated multi-layer composite kernels may be used, based on the fact that a composition and positive linear combination of kernels is still a kernel [192, Ch. 11]. However, the scalable development of online nonparametric methods based on such composite kernels is not straight-forward, and left to future work.

Setting aside the issue of how to develop kernel methods that attain comparable accuracy to deep neural networks, in this chapter, we observe that properly sparsified nonparametric methods yield a stable framework for accurate statistical learning from streaming data, and suggest a path forward for statistical inference in collaborative decentralized systems. In the next chapter, we build upon the framework set forth by POLK to develop such a method.

Chapter 7

Decentralized Efficient Nonparametric Stochastic Optimization

In this chapter, we build upon the mathematical framework for greedily sparsified nonparametric statistical estimation developed in 6 to devise the first provably stable and memory-efficient method for collaborative multi-agent learning. To do so, we consider decentralized online optimization problems: a network $\mathcal{G} = (V, \mathcal{E})$ of agents aims to minimize a global objective that is a sum of local convex objectives available only to each node. The problem is online because data samples upon which the local objectives depend are sequentially and locally observed by each agent. In this setting, agents aim to make inferences as well as one which has access to all data at a centralized location in advance. Here instead of assuming agents seek a common parameter vector $\mathbf{w} \in \mathbb{R}^p$, we instead focus on the case where agents seek to learn a common *decision function* $f(\mathbf{x})$ that belong to a reproducing kernel Hilbert space (RKHS). Such functions represent, e.g., nonlinear statistical models [7] or trajectories in a continuous space [123]. Nonlinear interpolators typically perform far better than their linear counterparts induced by the vector-valued convex problems [109] (see the numerical experiments of Chapters 2-3 as compared with that of Chapters 4 and 6), but little work [67, 141] has been done to extend them to streaming decentralized settings that are increasingly important in Internet of Things [69,111] and multi-robot [95,172] applications.

To contextualize this work, first consider centralized vector-valued stochastic convex programming, which has classically been solved with stochastic gradient descent (SGD) [161]. SGD involves descending along the negative of the stochastic gradient rather than the true gradient to avoid the fact that computing the gradient of the average objective has complexity comparable to the training sample size, which could be infinite. In contrast, the

setting considered in this work is a stochastic program defined over a function space, which is in general an intractable variational inference problem. However, when the function space is a RKHS [88], the Representer Theorem allows us to transform a search over an infinite space into one over a set of weights and data samples [169]. Unfortunately, the feasible set of the resulting problem has complexity comparable to the sample size N, and thus is intractable for $N \to \infty$ [144].

Efforts to mitigate the complexity of the function representation, colloquially called the curse of kernelization, have been developed that combine functional extensions of stochastic gradient method with compressions of the function sequence parameterization [50, 59, 89, 112, 227]. Mostly, such methods compress the function representation independent of the optimization problem to which they are applied. In contrast, in Chapter 6, we develop a method that combines greedily constructed [148] sparse subspace projections with functional stochastic gradient descent and guarantees exact convergence to the minimizer of the average risk functional. This technique, called parsimonious online learning with kernels (POLK), tailors the parameterization compression to not violate descent properties of the underlying RKHS-valued stochastic process [98], and inspires the approach considered in this chapter.

In this chapter, we extend the ideas in [98] to multi-agent settings. Multiple tools from distributed optimization may be used to develop such an extension; however, we note that the Representer Theorem [169] has not been established for general stochastic saddle point problems in RKHSs. Therefore, we adopt an approximate primal-only approach based on penalty methods [82, 157], which in the context of decentralized optimization is known as distributed gradient descent (DGD). Using functional stochastic extensions of DGD, together with the greedy Hilbert subspace projections designed in POLK, we develop a method such that each agent, through its local data stream and message passing with only its neighbors, learns a memory-efficient approximation to the globally optimal regression function with probability 1. Such global stability guarantees are in contrast to other methods for nonlinear function estimation in distributed online settings such as dictionary learning [43, 95, 116] or neural networks [103], which suffer from instability due to the non-convexity of the optimization problem their training defines.

The result of the chapter is organized as follows. In Section 7.1 we clarify the problem setting of stochastic programming in the RKHS setting in the centralized and decentralized case. In Section 7.2, we propose a new penalty function that permits deriving a decentralized online method for kernel regression without any complexity bottleneck by making use of functional stochastic gradient method (Section 7.2.1) combined with greedy subspace projections (Section 7.2.2). In Section 7.3 we present our main theoretical results, which establish function of sequence of each agent generated by the proposed technique converges to a neighborhood of the globally optimal regression function with probability 1. In Section

7.4, we present numerical examples of decentralized online multi-class kernel logistic regression and kernel support vector classification with data generated from Gaussian mixtures, and observe a state of the art trade-off between stability and accuracy.

7.1 Decentralized Functional Stochastic Programming

Consider the problem of expected risk minimization problem, where the goal is to learn a regressor that minimizes a loss function quantifying the merit of a statistical model averaged over a data set. We focus on the case when the number of training examples N is very large or infinite. In this chapter, input-output examples, $(\mathbf{x}_n, \mathbf{y}_n)$, are i.i.d. realizations drawn from a stationary joint distribution over the random pair $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subset \mathbb{R}^p$ and $\mathcal{Y} \subset \mathbb{R}$. Here, we consider finding regressors that are not vector valued parameters, but rather functions $\tilde{f} \in \mathcal{H}$ in a hypothesized function class $\mathcal{H}$, which allows for learning nonlinear statistical models rather than generalized linear models that rarely achieve satisfactory statistical error rates in practice [109, 131]. The merit of the function $\tilde{f}$ is evaluated by the convex loss function $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that quantifies the merit of the estimator $\tilde{f}(\tilde{\mathbf{x}})$ evaluated at feature vector $\tilde{\mathbf{x}}$. This loss is averaged over all possible training examples to define the statistical loss $\tilde{L}(\tilde{f}) := \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(\tilde{f}(\mathbf{x}), y)]$, which we combine with a Tikhonov regularizer to construct the regularized loss $\tilde{R}(\tilde{f}) := \operatorname{argmin}_{\tilde{f} \in \mathcal{H}} \tilde{L}(\tilde{f}) + (\lambda/2)\|\tilde{f}\|^2_{\mathcal{H}}$ [62, 178]. We then define the optimal function as

$$\tilde{f}^* = \operatorname*{argmin}_{\tilde{f} \in \mathcal{H}} \tilde{R}(\tilde{f}) := \operatorname*{argmin}_{\tilde{f} \in \mathcal{H}} \mathbb{E}_{\tilde{\mathbf{x}},\tilde{y}} \Big[\ell\big(\tilde{f}\big(\tilde{\mathbf{x}}\big), \tilde{y}\big) \Big] + \frac{\lambda}{2} \|\tilde{f}\|^2_{\mathcal{H}} \tag{7.1}$$

In this chapter, we focus on extensions of the formulation in (7.1) to the case where data is scattered across an interconnected network that represents, for instance, robotic teams [95], communication systems [159], or sensor networks [101]. To do so, we define a symmetric, connected, and directed network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = V$ nodes and $|\mathcal{E}| = E$ edges and denote as $n_i := \{j : (i,j) \in \mathcal{E}\}$ the neighborhood of agent i. For simplicity we assume that the number of edges E is even. Each agent $i \in \mathcal{V}$ observes a local data sequence as realizations $(\mathbf{x}_{i,n}, y_{i,n})$ from random pair $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ and seeks to learn a common globally optimal regression function f. This setting may be mathematically captured by associating to each node i is a convex loss functional $\ell_i : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that quantifies the merit of the estimator $f_i(\mathbf{x}_i)$ evaluated at feature vector $\mathbf{x}_i$, and defining the goal for each node as the minimization of the common global loss

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}} \sum_{i \in \mathcal{V}} \left(\mathbb{E}_{\mathbf{x}_i, y_i} \Big[\ell_i\big(f\big(\mathbf{x}_i\big), y_i\big) \Big] + \frac{\lambda}{2} \|f\|^2_{\mathcal{H}} \right) \tag{7.2}$$

Observe that this global loss is a network-wide average (scaled by V) of all local losses, and therefore the minimizers of (7.1) and (7.2) coincide when $(\mathbf{x}_i, y_i)$ have a common joint distribution for each i. However, in multi-agent optimization, we assume that there is no global coordination among the agents in selecting regression function f, but rather, each agent, based upon its locally observed data and message passing with its neighbors, seeks to learn f^*. Therefore, we allow agent i to select a distinct function f_i, but due to the uniqueness of the optimizer of (7.2) (since the regularizer makes the problem strongly convex), at optimality all function estimates f_i coincide with one another, we constrain functions to be equal to one another $f_i = f_j$, $(i,j) \in \mathcal{E}$. Thus we consider the nonparametric decentralized stochastic program:

$$\begin{aligned} f^* = \quad & \underset{f_i \in \mathcal{H}}{\operatorname{argmin}} \qquad \sum_{i \in \mathcal{V}} \left(\mathbb{E}_{\mathbf{x}_i, y_i} \Big[\ell_i(f_i(\mathbf{x}), y_i) \Big] + \frac{\lambda}{2} \|f_i\|_{\mathcal{H}}^2 \right) \\ & \text{such that} \quad f_i = f_j \ , (i,j) \in \mathcal{E} \end{aligned} \tag{7.3}$$

For further reference we define the stacked Hilbert space $\mathcal{H}^V$ of functions aggregated over the network whose elements are stacked functions $f(\cdot) = [f_1(\cdot); \cdots ; f_V(\cdot)]$ that yield vectors of length V when evaluated at local random vectors $f(\mathbf{x}) = [f_1(\mathbf{x}_1); \cdots ; f_V(\mathbf{x}_V)] \in \mathbb{R}^V$. Moreover, define stacked the random vectors $\mathbf{x} = [\mathbf{x}_1; \cdots ; \mathbf{x}_V] \in \mathcal{X}^V \subset \mathbb{R}^{Vp}$ and $\mathbf{y} = [y_1; \cdots y_V] \in \mathbb{R}^V$ that represents V labels or physical measurements, for instance.

The goal of this chapter is to develop an algorithm to solve (7.3) in distributed online settings where nodes don't know the distribution of the random pair $(\mathbf{x}_i, y_i)$ but observe local independent training examples $(\mathbf{x}_{i,n}, y_{i,n})$ sequentially. Before discussing necessary details of the function space $\mathcal{H}^V$ to make (7.3) tractable and discussing algorithmic solutions, we present a representative example.

Example (Distributed Online Kernel Logistic Regression). Consider the case of *kernel logistic regression* (KLR), with feature vectors $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^p$ and binary class labels $y_i \in \{0, 1\}$. Each agent seeks to learn a common global function $f_i \in \mathcal{H}$ that allows us to best approximate the distribution of an unknown class label given a training example $\mathbf{x}_i$ under the assumed model

$$\mathbb{P}\left(y_i = 0 \mid \mathbf{x}_i\right) = \frac{\exp\left\{f_i(\mathbf{x}_i)\right\}}{1 + \exp\left\{f_i(\mathbf{x})\right\}} \ . \tag{7.4}$$

In classical logistic regression, we assume that f_i is linear, i.e., $f_i(\mathbf{x}_i) = \mathbf{c}_i^T \mathbf{x}_i + b$. In KLR, on the other hand, we instead seek a nonlinear regression function. By making use of (7.4), we may formulate a maximum-likelihood estimation (MLE) problem to find the optimal functions $\{f_i\}_{i=1}^V$ on the basis of training examples $\{\mathbf{x}_i\}_{i=1}^V$ by solving for the function that maximizes the λ-regularized average negative log likelihood over the network-aggregated

data

$$f^* = \underset{f \in \mathcal{H}^V}{\operatorname{argmin}} \sum_{i \in \mathcal{V}} \left(\mathbb{E}_{\mathbf{x}_i, y_i} \left[\log \left(1 + e^{\{f_i(\mathbf{x}_i)\}} \right) - \mathbb{1}(y_i = 1) - f(\mathbf{x}_i) \mathbb{1}(y_i = 0) \right] + \frac{\lambda}{2} \| f_i \|_{\mathcal{H}}^2 \right)$$
$$\text{such that} \quad f_i = f_j \, , (i,j) \in \mathcal{E} \tag{7.5}$$

where $\mathbb{1}(\cdot)$ represents the indicator function. Solving (7.5) amounts to finding a function f_i for each agent i that, given a feature vector $\mathbf{x}_i$ and the probabilistic model outlined by (7.4), best represents the class-conditional probabilities that the corresponding label y_i is either 0 or 1.

7.1.1 Function Estimation in Reproducing Kernel Hilbert Spaces

The optimization problem in (7.1), and hence (7.3), is intractable in general, since it defines a variational inference problem integrated over the unknown joint distribution $\mathbb{P}(\mathbf{x}, y)$. However, when $\mathcal{H}$ is equipped with a *reproducing kernel* $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (see [109, 132]), a function estimation problem of the form (7.1) may be reduced to a parametric form via the Representer Theorem [144, 212]. Thus, we restrict the Hilbert space in Section 7.1 to be one equipped with a kernel κ that satisfies for all functions $\tilde{f} : \mathcal{X} \to \mathbb{R}$ in $\mathcal{H}$:

$$\begin{aligned} &(i) \ \langle \tilde{f}, \kappa(\mathbf{x}_i, \cdot) \rangle_{\mathcal{H}} = \tilde{f}(\mathbf{x}_i) \quad \text{for all } \mathbf{x}_i \in \mathcal{X} \, , \\ &(ii) \ \mathcal{H} = \overline{\operatorname{span}\{\kappa(\mathbf{x}_i, \cdot)\}} \quad \text{for all } \mathbf{x}_i \in \mathcal{X} \, . \end{aligned} \tag{7.6}$$

Here $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the Hilbert inner product for $\mathcal{H}$. We further assume that the kernel is positive semidefinite, i.e. $\kappa(\mathbf{x}_i, \mathbf{x}_i') \geq 0$ for all $\mathbf{x}_i, \mathbf{x}_i' \in \mathcal{X}$. Function spaces with this structure are called reproducing kernel Hilbert spaces (RKHS).

In (7.6), property (i) is the reproducing property (via Riesz Representation Theorem [212]). Replacing $\tilde{f}$ by $\kappa(\mathbf{x}_i', \cdot)$ in (7.6) (i) yields $\langle \kappa(\mathbf{x}_i', \cdot), \kappa(\mathbf{x}_i, \cdot) \rangle_{\mathcal{H}} = \kappa(\mathbf{x}_i, \mathbf{x}_i')$ which is the origin of the term "reproducing kernel." This property induces a nonlinear transformation of the input space $\mathcal{X}$: denote by $\phi(\cdot)$ a nonlinear map of the feature space that assigns to each $\mathbf{x}_i$ the kernel function $\kappa(\cdot, \mathbf{x}_i)$. The reproducing property yields that the inner product of the image of distinct feature vectors $\mathbf{x}_i$ and $\mathbf{x}_i'$ under the map ϕ requires only kernel evaluations: $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_i') \rangle_{\mathcal{H}} = \kappa(\mathbf{x}_i, \mathbf{x}_i')$ (the 'kernel trick').

Moreover, property (7.6) (ii) states that any function $\tilde{f} \in \mathcal{H}$ may be written as a linear combination of kernel evaluations. For kernelized and regularized empirical risk minimization (ERM), the Representer Theorem [88, 169] establishes that the optimal $\tilde{f}$ in the hypothesis function class $\mathcal{H}$ may be written as an expansion of kernel evaluations *only*

at elements of the training set as

$$\tilde{f}(\mathbf{x}_i) = \sum_{n=1}^{N} w_{i,n}\kappa(\mathbf{x}_{i,n}, \mathbf{x}_i) \ . \tag{7.7}$$

where $\mathbf{w}_i = [w_{i,1}, \cdots, w_{i,N}]^T \in \mathbb{R}^N$ denotes a set of weights. The upper index N in (7.7) is referred to as the model order, and for ERM the model order and training sample size are equal. Common choices κ include the polynomial and the radial basis kernel, i.e., $\kappa(\mathbf{x}_i, \mathbf{x}_i') = \left(\mathbf{x}_i^T\mathbf{x}_i' + q\right)^b$ and $\kappa(\mathbf{x}_i, \mathbf{x}_i') = \exp\{-\|\mathbf{x}_i - \mathbf{x}_i'\|_2^2/2\tilde{\sigma}^2\}$, respectively, where $\mathbf{x}_i, \mathbf{x}_i' \in \mathcal{X}$.

Suppose, for the moment, that we have access to N i.i.d. realizations of the random pairs $(\mathbf{x}_i, y_i)$ for each agent i such that the expectation in (7.3) is computable, and we further ignore the consensus constraint. Then the objective in (7.3) becomes:

$$f^* = \underset{f\in\mathcal{H}^V}{\operatorname{argmin}} \frac{1}{N}\sum_{n=1}^{N}\sum_{i\in\mathcal{V}} \ell(f_i(\mathbf{x}_{i,n}), y_{i,n}) + \frac{\lambda}{2}\|f_i\|_{\mathcal{H}}^2 \tag{7.8}$$

Then, by substituting the Representer Theorem [cf. (7.7)] into (7.3), we obtain that optimizing in $\mathcal{H}^V$ reduces to optimizing over the set of NV weights:

$$f^* = \underset{\{\mathbf{w}_i\}\in\mathbb{R}^N}{\operatorname{argmin}} \frac{1}{N}\sum_{n=1}^{N}\sum_{i\in\mathcal{V}} \ell_i(\mathbf{w}_i^T\boldsymbol{\kappa}_{\mathbf{X}_i}(\mathbf{x}_{i,n}), y_{i,n}) + \frac{\lambda}{2}\mathbf{w}_i^T\mathbf{K}_{\mathbf{X}_i,\mathbf{X}_i}\mathbf{w}_i \ , \tag{7.9}$$

where we have defined the Gram (or kernel) matrix $\mathbf{K}_{\mathbf{X}_i,\mathbf{X}_i} \in \mathbb{R}^{N\times N}$, with entries given by the kernel evaluations between $\mathbf{x}_{i,m}$ and $\mathbf{x}_{i,n}$ as $[\mathbf{K}_{\mathbf{X}_i,\mathbf{X}_i}]_{m,n} = \kappa(\mathbf{x}_{i,m}, \mathbf{x}_{i,n})$. We further define the vector of kernel evaluations $\boldsymbol{\kappa}_{\mathbf{X}_i}(\cdot) = [\kappa(\mathbf{x}_{i,1}, \cdot)\ldots\kappa(\mathbf{x}_{i,N}, \cdot)]^T$, which are related to the kernel matrix as $\mathbf{K}_{\mathbf{X}_i,\mathbf{X}_i} = [\boldsymbol{\kappa}_{\mathbf{X}_i}(\mathbf{x}_{i,1})\ldots\boldsymbol{\kappa}_{\mathbf{X}_i}(\mathbf{x}_{i,N})]$. The dictionary of training points associated with the kernel matrix is defined as $\mathbf{X}_i = [\mathbf{x}_{i,1}, \ \ldots \ , \mathbf{x}_{i,N}]$.

By exploiting the Representer Theorem, we transform a nonparametric infinite dimensional optimization problem in $\mathcal{H}^V$ (7.8) into a finite NV-dimensional parametric problem (7.9). Thus, for empirical risk minimization, the RKHS provides a principled framework to solve nonparametric regression problems as a search over $\mathbb{R}^{VN}$ for an optimal set of coefficients.

However, is to solve problems of the form (7.8) when training examples $(\mathbf{x}_{i,n}, y_{i,n})$ become sequentially available or their total number N is not finite, the objective in (7.8)

becomes an expectation over random pairs $(\mathbf{x}_i, y_i)$ as [181]

$$f^* = \underset{\mathbf{w}_i \in \mathbb{R}^{\mathcal{I}}, \{\mathbf{x}_{i,n}\}_{n\in\mathcal{I}}}{\operatorname{argmin}} \sum_{i\in\mathcal{V}} \mathbb{E}_{\mathbf{x}_i, y_i}[\ell_i(\sum_{n\in\mathcal{I}} w_{i,n}\kappa(\mathbf{x}_{i,n}, \mathbf{x}_i), y_i)] + \frac{\lambda}{2}\| \sum_{n,m\in\mathcal{I}} w_{i,n}w_{i,m}\kappa(\mathbf{x}_{i,m}, \mathbf{x}_{i,n})\|^2_{\mathcal{H}} \,, \tag{7.10}$$

where we substitute the Representer Theorem generalized to the infinite sample-size case established in [144] into the objective (7.3) with $\mathcal{I}$ as some countably infinite indexing set. That is, as the data sample size $N \to \infty$, the representation of f_i becomes infinite as well. Thus, our goal is to solve (7.10) in an approximate manner such that each f_i admits a finite representation near f_i^*, while satisfying the consensus constraints $f_i = f_j$ for $(i,j) \in \mathcal{E}$ (which were omitted for the sake of discussion between (7.8) - (7.10)).

7.2 Greedily Projected Penalty Method

We turn to developing an online iterative and decentralized solution to solving (7.3) when the functions $\{f_i\}_{i\in\mathcal{V}}$ are elements of a RKHS, as detailed in Section 7.1.1. To exploit the properties of this function space, we require the applicability of the Representer Theorem [cf. (7.7)], but this result holds for any regularized minimization problem with a convex functional. Thus, we may address the consensus constraint $f_i = f_j \ , (i,j) \in \mathcal{E}$ in (7.3) by enforcing approximate consensus on estimates $f_i(\mathbf{x}_i) = f_j(\mathbf{x}_i)$ in expectation. This specification may be met by introducing the penalty function

$$\psi_c(f) = \sum_{i\in\mathcal{V}} \Big(\mathbb{E}_{\mathbf{x}_i, \mathbf{y}_i}\Big[\ell_i(f_i\big(\mathbf{x}_i\big), y_i)\Big] + \frac{\lambda}{2}\|f_i\|^2_{\mathcal{H}} + \frac{c}{2}\sum_{j\in n_i} \mathbb{E}_{\mathbf{x}_i}\left\{[f_i(\mathbf{x}_i) - f_j(\mathbf{x}_i)]^2\right\}\Big) \tag{7.11}$$

The reasoning for the definition (7.11) rather than one that directly addresses the consensus constraint deterministically is given in Remark 7, motivated by following the algorithm derivation. For future reference, we also define the local penalty as

$$\psi_{i,c}(f_i) = \mathbb{E}_{\mathbf{x}_i, \mathbf{y}_i}\Big[\ell_i(f_i\big(\mathbf{x}_i\big), y_i)\Big] + \frac{\lambda}{2}\|f_i\|^2_{\mathcal{H}} + \frac{c}{2}\sum_{j\in n_i} \mathbb{E}_{\mathbf{x}_i}\left\{[f_i(\mathbf{x}_i) - f_j(\mathbf{x}_i)]^2\right\} \tag{7.12}$$

and we easily observe from (7.11) - (7.12) that $\psi_c(f) = \sum_i \psi_{i,c}(f_i)$. Further define $f_c^* = \operatorname{argmin}_{f\in\mathcal{H}^V} \psi_c(f)$. We note that in the vector-valued decision variable case, other techniques to address the constraint in (7.3) are possible such as primal-dual methods [93] or dual methods [187], but the Representer Theorem has not been established for RKHS-valued stochastic saddle point problems. It is an open question whether expressions of the form (7.7) apply to problems with general functional constraints, but this matter is beyond the

scope of this thesis. Therefore, these other approaches which make use of Lagrange duality do not readily extend to the nonparametric setting considered here.

7.2.1 Functional Stochastic Gradient Method

Given that the data distribution $\mathbb{P}(\mathbf{x}, \mathbf{y})$ is unknown, minimizing $\psi_c(f)$ directly via variational inference is not possible. Rather than postulate a specific distribution for $(\mathbf{x}, \mathbf{y})$, we only assume access to sequentially available (streaming) independent and identically distributed samples $(\mathbf{x}_t, \mathbf{y}_t)$ from their joint density. Then, we may wield tools from stochastic approximation to minimize (7.11), which in turn yields a solution to (7.3). Begin by defining, $\hat{\psi}_c(f(\mathbf{x}_t), \mathbf{y}_t)$, the stochastic approximation of the penalty function $\psi_c(f)$, evaluated at a realization $(\mathbf{x}_t, \mathbf{y}_t)$ of the stacked random pair $(\mathbf{x}, \mathbf{y})$:

$$\hat{\psi}_c(f(\mathbf{x}_t), \mathbf{y}_t) = \sum_{i \in \mathcal{V}} \Big(\ell_i(f_i\big(\mathbf{x}_{i,t}\big), y_{i,t}\big) + \frac{\lambda}{2} \|f_i\|_{\mathcal{H}}^2 + \frac{c}{2} \sum_{j \in n_j} (f_i(\mathbf{x}_i) - f_j(\mathbf{x}_i))^2 \Big) \tag{7.13}$$

and the local instantaneous penalty function $\hat{\psi}_{i,c}(f_i(\mathbf{x}_{i,t}), \mathbf{y}_{i,t})$ similarly. To compute the functional stochastic gradient of $\psi_c(f)$ evaluated at a sample point $(\mathbf{x}_t, \mathbf{y}_t)$, we first address the local data-dependent term $\ell_i(f_i\big(\mathbf{x}_{i,t}\big), y_{i,t})$ in (7.13) as [89, 98]:

$$\nabla_{f_i} \ell_i(f_i(\mathbf{x}_{i,t}), y_{i,t})(\cdot) = \frac{\partial \ell_i(f_i(\mathbf{x}_{i,t}), y_{i,t})}{\partial f_i(\mathbf{x}_{i,t})} \frac{\partial f_i(\mathbf{x}_{i,t})}{\partial f_i}(\cdot) \tag{7.14}$$

where we have applied the chain rule. Now, define the short-hand notation $\ell_i'(f_i(\mathbf{x}_{i,t}), y_{i,t}) := \partial \ell_i(f_i(\mathbf{x}_{i,t}), y_{i,t}) / \partial f_i(\mathbf{x}_{i,t})$ for the derivative of $\ell_i(f(\mathbf{x}_{i,t}), y_{i,t})$ with respect to its first scalar argument $f_i(\mathbf{x}_{i,t})$ evaluated at $\mathbf{x}_{i,t}$. To evaluate the second term on the right-hand side of (7.14), differentiate both sides of the expression defining the reproducing property of the kernel [cf. (7.6)(i)] with respect to f_i to obtain

$$\frac{\partial f_i(\mathbf{x}_{i,t})}{\partial f_i} = \frac{\partial \langle f_i, \kappa(\mathbf{x}_{i,t}, \cdot) \rangle_{\mathcal{H}}}{\partial f_i} = \kappa(\mathbf{x}_{i,t}, \cdot) \tag{7.15}$$

Then, given (7.14) - (7.15), we may compute the overall gradient of the instantaneous penalty function $\hat{\psi}_c(f(\mathbf{x}_t), \mathbf{y}_t)$ in (7.13) as

$$\nabla_f \hat{\psi}_c(f(\mathbf{x}_t), \mathbf{y}_t) = \mathrm{vec}\Big[\ell_i'(f_i(\mathbf{x}_{i,t}), y_{i,t}) \kappa(\mathbf{x}_{i,t}, \cdot) + \lambda f_i + c \sum_{j \in n_i} (f_i(\mathbf{x}_{i,t}) - f_j(\mathbf{x}_{i,t})) \kappa(\mathbf{x}_{i,t}, \cdot) \Big] \tag{7.16}$$

where on the right-hand side of (7.16), we have defined the vector stacking notation $\mathrm{vec}[\cdot]$ to denote the stacking of V component-wise functional gradients, each associated with

function f_i, $i \in \mathcal{V}$, and used the fact that the variation of the instantaneous approximate of the cross-node term, $[f_i(\mathbf{x}_i) - f_j(\mathbf{x}_i)]^2$, by the same reasoning as (7.14) - (7.15), is $2[f_i(\mathbf{x}_{i,t}) - f_j(\mathbf{x}_{i,t})]\kappa(\mathbf{x}_{i,t}, \cdot)$. With this computation in hand, we present the stochastic gradient method for the kernelized λ-regularized multi-agent expected risk minimization problem in (7.3) as

$$f_{t+1} = (1 - \eta_t\lambda)f_t - \eta_t \text{vec}\Big[\ell_i'(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})\kappa(\mathbf{x}_{i,t}, \cdot) + c\sum_{j \in n_i}(f_{i,t}(\mathbf{x}_{i,t}) - f_{j,t}(\mathbf{x}_{i,t}))\kappa(\mathbf{x}_{i,t}, \cdot)\Big] , \tag{7.17}$$

where $\eta_t > 0$ is an algorithm step-size either chosen as diminishing with $\mathcal{O}(1/t)$ or a small constant – see Section 7.3. We may glean from (7.17) that the update for the network-wide function f_t decouples into ones for each agent $i \in \mathcal{V}$, using the node-separability of the penalty $\psi_c(f) = \sum_i \psi_{i,c}(f_i)$, i.e.,

$$f_{i,t+1} = (1 - \eta_t\lambda)f_{i,t} - \eta_t\Big[\ell_i'(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})\kappa(\mathbf{x}_{i,t}, \cdot) + c\sum_{j \in n_i}(f_{i,t}(\mathbf{x}_{i,t}) - f_{j,t}(\mathbf{x}_{i,t}))\kappa(\mathbf{x}_{i,t}, \cdot)\Big] . \tag{7.18}$$

We further require that, given $\lambda > 0$, the step-size satisfies $\eta_t < 1/\lambda$ and the global sequence is initialized as $f_0 = 0 \in \mathcal{H}^V$. With this initialization, the Representer Theorem (7.7) implies that, at time t, the function $f_{i,t}$ admits an expansion in terms of feature vectors $\mathbf{x}_{i,t}$ observed thus far as

$$f_{i,t}(\mathbf{x}) = \sum_{n=1}^{t-1} w_{i,n}\kappa(\mathbf{x}_{i,n}, \mathbf{x}) = \mathbf{w}_{i,t}^T \boldsymbol{\kappa}_{\mathbf{X}_{i,t}}(\mathbf{x}) . \tag{7.19}$$

On the right-hand side of (7.19) we have introduced the notation $\mathbf{X}_{i,t} = [\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,t-1}] \in \mathbb{R}^{p \times (t-1)}$, $\boldsymbol{\kappa}_{\mathbf{X}_{i,t}}(\cdot) = [\kappa(\mathbf{x}_{i,1}, \cdot), \ \ldots \ , \kappa(\mathbf{x}_{i,t-1}, \cdot)]^T$, and $\mathbf{w}_{i,t} = [w_{i,1}, \ \ldots \ w_{i,t-1}] \in \mathbb{R}^{t-1}$. Moreover, observe that the kernel expansion in (7.19), taken together with the functional update (7.17), yields the fact that performing the stochastic gradient method in $\mathcal{H}^V$ amounts to the following V parallel parametric updates on the kernel dictionaries $\mathbf{X}_i$ and coefficients $\mathbf{w}_i$:

$$\mathbf{X}_{i,t+1} = [\mathbf{X}_{i,t}, \ \ \mathbf{x}_{i,t}] , \tag{7.20}$$

$$[\mathbf{w}_{i,t+1}]_u = \begin{cases} (1 - \eta_t\lambda)[\mathbf{w}_{i,t}]_u & \text{for } 0 \leq u \leq t-1 \\ -\eta_t\Big(\ell_i'(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t}) + c\sum_{j \in n_i}(f_{i,t}(\mathbf{x}_{i,t}) - f_{j,t}(\mathbf{x}_{i,t}))\Big), & \text{for } u = t \end{cases}$$

Observe that this update causes $\mathbf{X}_{i,t+1}$ to have one more column than $\mathbf{X}_{i,t}$. We define the *model order* as number of data points $M_{i,t}$ in the dictionary of agent i at time t (the number of columns of $\mathbf{X}_t$). FSGD is such that $M_{i,t} = t - 1$, and hence grows unbounded

with iteration index t. In the following subsection, we address this intractable memory growth such that we may execute stochastic descent through low-dimensional projections of the stochastic gradient, inspired by [98]. Before doing so, we clarify the motivation for the choice of the penalty function (7.11).

Remark 7 In principle, it is possible to address the RKHS-valued consensus constraint in (7.3) directly, through primal-only stochastic methods, by introducing the penalty function

$$\tilde{\psi}_c(f) = \sum_{i\in\mathcal{V}} \Big(\mathbb{E}_{\mathbf{x}_i,\mathbf{y}_i}\Big[\ell_i(f_i(\mathbf{x}_i), y_i)\Big] + \frac{\lambda}{2}\|f_i\|^2_{\mathcal{H}} + \frac{c}{2}\sum_{j\in n_i}\|f_i - f_j\|^2_{\mathcal{H}}\Big) \tag{7.21}$$

Observe, however, that FSGD applied to (7.21), using comparable reasoning to that which leads to (7.18) from (7.11), yields

$$f_{i,t+1} = (1-\eta_t\lambda)f_{i,t} - \eta_t\Big[\nabla_{f_i}\ell_i'(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})\kappa(\mathbf{x}_{i,t},\cdot) + c\sum_{j\in n_i}(f_{i,t} - f_{j,t})\Big]. \tag{7.22}$$

Unfortunately, we cannot inductively define a parametric representation of (7.22) for node i in terms of its own kernel dictionaries and weights independently of the *entire function* associated to node j, since the last term in (7.22) lives directly in the Hilbert space. Thus, to implement (7.22) each agent would need to store the entire kernel dictionary and weights of all its neighbors at each step, which is impractically costly. The use of (7.11) rather than (7.21) is further justified that under a hypothesis regarding the mean transformation of the local data spaces, $\mathbb{E}_{\mathbf{x}_i}[\kappa(\mathbf{x}_i,\cdot)]$, consensus with respect to the Hilbert norm, in addition to the mean square sense, is achieved.

7.2.2 Local Sparse Subspace Projections

To mitigate the complexity growth noted in Section 7.2.1, we approximate the function sequence (7.17) by one that is orthogonally projected onto subspaces $\mathcal{H}_{\mathbf{D}} \subseteq \mathcal{H}$ that consist only of functions that can be represented using some dictionary $\mathbf{D} = [\mathbf{d}_1, \ \ldots, \ \mathbf{d}_M] \in \mathbb{R}^{p\times M}$, i.e., $\mathcal{H}_{\mathbf{D}} = \{f \ : \ f(\cdot) = \sum_{n=1}^M w_n\kappa(\mathbf{d}_n,\cdot) = \mathbf{w}^T\boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\} = \text{span}\{\kappa(\mathbf{d}_n,\cdot)\}_{n=1}^M$, and $\{\mathbf{d}_n\} \subset \{\mathbf{x}_u\}_{u\leq t}$. For convenience we define $[\boldsymbol{\kappa}_{\mathbf{D}}(\cdot) = \kappa(\mathbf{d}_1,\cdot)\ldots\kappa(\mathbf{d}_M,\cdot)]$, and $\mathbf{K}_{\mathbf{D},\mathbf{D}}$ as the resulting kernel matrix from this dictionary. We enforce efficiency in function representation by selecting dictionaries $\mathbf{D}_i$ that $M_{i,t} << \mathcal{O}(t)$ for each i, following [98].

To be specific, we propose replacing the local update (7.18) in which the dictionary

grows at each iteration by its projection onto subspace $\mathcal{H}_{\mathbf{D}_{i,t+1}} = \text{span}\{\kappa(\mathbf{d}_{i,n}, \cdot)\}_{n=1}^{M_{t+1}}$ as

$$f_{i,t+1} = \underset{f \in \mathcal{H}_{\mathbf{D}_{i,t+1}}}{\text{argmin}} \left\| f - \Big(f_{i,t} - \eta_t \nabla_{f_i} \hat{\psi}_{i,c}(f_i(\mathbf{x}_{i,t}), y_{i,t}) \Big) \right\|_{\mathcal{H}}^2$$
$$:= \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}} \Big[(1 - \eta_t \lambda) f_{i,t} - \eta_t \Big(\nabla_{f_i} \ell_i(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t}) + c \sum_{j \in n_i} (f_{i,t}(\mathbf{x}_{i,t}) - f_{j,t}(\mathbf{x}_{i,t})) \kappa(\mathbf{x}_{i,t}, \cdot) \Big) \Big]. \tag{7.23}$$

where we define the projection operator $\mathcal{P}$ onto subspace $\mathcal{H}_{\mathbf{D}_{i,t+1}} \subset \mathcal{H}$ by the update (7.23).

Coefficient update The update (7.23), for a fixed dictionary $\mathbf{D}_{i,t+1} \in \mathbb{R}^{p \times M_{t+1}}$, yields one in the coefficient space only. This fact may be observed by defining the un-projected stochastic gradient step starting at function $f_{i,t}$ parameterized by dictionary $\mathbf{D}_{i,t}$ and coefficients $\mathbf{w}_{i,t}$:

$$\tilde{f}_{i,t+1} = f_{i,t} - \eta_t \nabla_{f_i} \hat{\psi}_{i,c}(f_i(\mathbf{x}_{i,t}), y_{i,t}) \ . \tag{7.24}$$

This update may be represented using dictionary and weights

$$\tilde{\mathbf{D}}_{i,t+1} = [\mathbf{D}_{i,t}, \ \ \mathbf{x}_{i,t}] \ , \tag{7.25}$$
$$[\tilde{\mathbf{w}}_{i,t+1}]_u = \begin{cases} (1 - \eta_t \lambda)[\mathbf{w}_{i,t}]_u & \text{for } 0 \leq u \leq t-1 \\ -\eta_t \Big(\ell_i'(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t}) + c \sum_{j \in n_i} (f_{i,t}(\mathbf{x}_{i,t}) - f_{j,t}(\mathbf{x}_{i,t})) \Big), & \text{for } u = t \end{cases}$$

Note that $\tilde{\mathbf{D}}_{i,t+1}$ has $\tilde{M} = M_{i,t} + 1$ columns, which is also the length of $\tilde{\mathbf{w}}_{i,t+1}$. For a fixed $\mathbf{D}_{i,t+1}$, the stochastic projection (7.23) is a least-squares update on the coefficient vector: the Representer Theorem allows us to rewrite (7.23) in terms of kernel expansions as in Section 3.2 of [98], which yields

$$\mathbf{w}_{i,t+1} = \mathbf{K}^{-1}_{\mathbf{D}_{i,t+1}\mathbf{D}_{i,t+1}} \mathbf{K}_{\mathbf{D}_{i,t+1}\tilde{\mathbf{D}}_{i,t+1}} \tilde{\mathbf{w}}_{i,t+1} \ , \tag{7.26}$$

where we define the cross-kernel matrix $\mathbf{K}_{\mathbf{D}_{i,t+1},\tilde{\mathbf{D}}_{i,t+1}}$ whose $(n,m)^{\text{th}}$ entry is given by $\kappa(\mathbf{d}_{i,n}, \tilde{\mathbf{d}}_{i,m})$. The other kernel matrices $\mathbf{K}_{\tilde{\mathbf{D}}_{i,t+1},\tilde{\mathbf{D}}_{i,t+1}}$ and $\mathbf{K}_{\mathbf{D}_{i,t+1},\mathbf{D}_{i,t+1}}$ are defined similarly. Observe that $M_{i,t+1}$ is the number of columns in $\mathbf{D}_{i,t+1}$, while $\tilde{M}_i = M_{i,t} + 1$ is the number of columns in $\tilde{\mathbf{D}}_{t+1}$ [cf. (7.25)]. Given that the local projections of $\tilde{f}_{i,t+1}$ onto stochastic subspaces $\mathcal{H}_{\mathbf{D}_{i,t+1}}$, for a fixed node-specific dictionaries $\mathbf{D}_{i,t+1}$, is a least-squares multiplication, we now detail how the kernel dictionary $\mathbf{D}_{i,t+1}$ is selected from past data $\{\mathbf{x}_{i,u}, y_{i,u}\}_{u \leq t}$.

Dictionary Update The selection procedure for the kernel dictionary $\mathbf{D}_{i,t+1}$ is based upon greedy compression [137]: function $\tilde{f}_{i,t+1}$ defined by the stochastic gradient method without projection is parameterized by dictionary $\tilde{\mathbf{D}}_{i,t+1}$ [cf. (7.25)] of model order $\tilde{M}_i =$

$M_{i,t}+1$. We form $\mathbf{D}_{i,t+1}$ by selecting a subset of $M_{i,t+1}$ columns from $\tilde{\mathbf{D}}_{i,t+1}$ that best approximate $\tilde{f}_{i,t+1}$ in terms of Hilbert norm error, which may be done by executing *kernel orthogonal matching pursuit* (KOMP) [148, 203] with error tolerance ϵ_t to find a kernel dictionary matrix $\mathbf{D}_{i,t+1}$ based on the one which adds the latest sample point $\tilde{\mathbf{D}}_{i,t+1}$. This choice is due to the fact that we can tune its stopping criterion to guarantee stochastic descent, and guarantee the model order of the learned function remains finite – see Section 7.3 for details.

We now describe the variant of KOMP we propose using, called Destructive KOMP with Pre-Fitting (see [203], Section 2.3, and Algorithm 5). Begin with an input a candidate function $\tilde{f}$ of model order $\tilde{M}$ parameterized by kernel dictionary $\tilde{\mathbf{D}} \in \mathbb{R}^{p\times\tilde{M}}$ and coefficients $\tilde{\mathbf{w}} \in \mathbb{R}^{\tilde{M}}$. The method then approximates $\tilde{f}$ by a function $f \in \mathcal{H}$ with a lower model order. Initially, this sparse approximation is the original function $f = \tilde{f}$ so that its dictionary is initialized with that of the original function $\mathbf{D} = \tilde{\mathbf{D}}$, with corresponding coefficients $\mathbf{w} = \tilde{\mathbf{w}}$. Then, the algorithm sequentially removes dictionary elements from the initial dictionary $\tilde{\mathbf{D}}$, yielding a sparse approximation f of $\tilde{f}$, until the error threshold $\|f - \tilde{f}\|_{\mathcal{H}} \leq \epsilon_t$ is violated, in which case it terminates.

We summarize the key steps of the proposed method in Algorithm 7 for solving (7.3) while maintaining a finite model order, thus allowing for the memory-efficient learning of nonparametric regression functions online in multi-agent systems. The method, Greedy Projected Penalty Method, executes the stochastic projection of the functional stochastic gradient iterates onto sparse subspaces $\mathcal{H}_{\mathbf{D}_{i,t+1}}$ stated in (7.23). Initial functions are set to null $f_{i,0} = 0$, i.e., it has empty dictionary $\mathbf{D}_{i,0} = []$ and coefficient vector $\mathbf{w}_{i,0} = []$. The notation [] is used to denote the empty matrix or vector respective size $p \times 0$ or 0. Then, at each step, given an independent training example $(\mathbf{x}_{i,t}, y_{i,t})$ and step-size η_t, we compute the *unconstrained* functional stochastic gradient iterate (7.24) with respect to the instantaneous penalty function (7.13) which admits the parameterization $\tilde{\mathbf{D}}_{i,t+1}$ and $\tilde{\mathbf{w}}_{i,t+1}$ as stated in (7.25). These parameters are then fed into KOMP with approximation budget ϵ_t, such that $(f_{i,t+1}, \mathbf{D}_{i,t+1}, \mathbf{w}_{i,t+1}) = \text{KOMP}(\tilde{f}_{i,t+1}, \tilde{\mathbf{D}}_{i,t+1}, \tilde{\mathbf{w}}_{i,t+1}, \epsilon_t)$.

7.3 Convergence of Multi-Agent Efficient Kernel Learning

We turn to establishing that the method presented in Algorithm 7 converges with probability 1 to the minimizer of the penalty function $\psi_c(f)$ [cf. (7.11)] when attenuating algorithm step-sizes are used, and to a neighborhood of the minimizer along a subsequence when constant step-sizes are used. Moreover, for the later case, the kernel dictionary that parameterizes the regression function f_i for each agent i remains finite in the worst case. This analysis is an application of Section IV of [98], but these results, together with the properties of the penalty function $\psi_c(f)$ allow us to establish bounds on the deviation for each

Algorithm 7 Greedy Projected Penalty Method

Require: $\{\mathbf{x}_t, \mathbf{y}_t, \eta_t, \epsilon_t\}_{t=0,1,2,\ldots}$

initialize $f_{i,0}(\cdot) = 0, \mathbf{D}_{i,0} = [], \mathbf{w}_0 = []$, i.e. initial dictionary, coefficients are empty for each $i \in \mathcal{V}$

for $t = 0, 1, 2, \ldots$ **do**

loop in parallel for agent $i \in \mathcal{V}$

Observe local training example realization $(\mathbf{x}_{i,t}, y_{i,t})$

Send obs. $\mathbf{x}_{i,t}$ to nodes $j \in n_i$, receive scalar $f_{j,t}(\mathbf{x}_{i,t})$

Receive obs. $\mathbf{x}_{j,t}$ from nodes $j \in n_i$, send $f_{i,t}(\mathbf{x}_{j,t})$

Compute unconstrained stochastic grad. step [cf. (7.24)]

$$\tilde{f}_{i,t+1}(\cdot) = (1 - \eta_t \lambda) f_{i,t} - \eta_t \nabla_{f_i} \hat{\psi}_{i,c}(f_i(\mathbf{x}_{i,t}), \mathbf{y}_{i,t}) \; .$$

Update params: $\tilde{\mathbf{D}}_{i,t+1} = [\mathbf{D}_{i,t}, \;\; \mathbf{x}_{i,t}]$, $\tilde{\mathbf{w}}_{i,t+1}$ [cf. (7.25)]

Greedily compress function using matching pursuit

$$(f_{i,t+1}, \mathbf{D}_{i,t+1}, \mathbf{w}_{i,t+1}) = \mathbf{KOMP}(\tilde{f}_{i,t+1}, \tilde{\mathbf{D}}_{i,t+1}, \tilde{\mathbf{w}}_{i,t+1}, \epsilon_t)$$

end loop

end for

individual in the network from the common globally optimal regression function.

Before analyzing the proposed method developed in Section 7.2, we define key quantities to simplify the analysis and introduce standard assumptions which are necessary to establish convergence. Define the local projected stochastic functional gradient associated with the update in (7.23) as

$$\tilde{\nabla}_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t}) = \Big(f_{i,t} - \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}} \Big[f_{i,t} - \eta_t \nabla_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t}) \Big] \Big) / \eta_t \tag{7.27}$$

such that the local update of Algorithm 7 [cf. (7.23)] may be expressed as a stochastic descent using projected functional gradients $f_{i,t+1} = f_{i,t} - \eta_t \tilde{\nabla}_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})$. The definitions of (7.27) and the local stochastic gradient $\nabla_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})$ may be stacked to analyze the global convergence behavior of the algorithm. For further reference, we define the stacked projected functional stochastic gradient of the penalty function as $\tilde{\nabla}_f \hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t) = [\tilde{\nabla}_{f_1} \hat{\psi}_{1,c}(f_{1,t}(\mathbf{x}_{1,t}), y_{1,t}); \cdots ; \tilde{\nabla}_{f_V} \hat{\psi}_{V,c}(f_{V,t}(\mathbf{x}_{V,t}), y_{V,t})]$. Then the stacked global update of the algorithm is

$$f_{t+1} = f_t - \eta_t \tilde{\nabla}_f \hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t) \; . \tag{7.28}$$

Moreover, observe that the stochastic functional gradient in (7.16), based upon the fact that $(\mathbf{x}_t, y_t)$ are independent and identically distributed realizations of the random pair $(\mathbf{x}, y)$, is an unbiased estimator of the true functional gradient of the penalty function $\psi_c(f)$ in

(7.11), i.e.

$$\mathbb{E}[\nabla_f \hat{\psi}_c(f(\mathbf{x}_t), \mathbf{y}_t) \,\big|\, \mathcal{F}_t] = \nabla_f \psi_c(f) \tag{7.29}$$

for all t. Next, we formally state technical conditions on the loss functions, data domain, and stochastic approximation errors that are necessary to establish convergence.

AS18 *The sets $\mathcal{H}_{\mathbf{D}_{i,t}}$ onto which the functions $f_{i,t}$ are projected in* (7.23) *are intersected with some finite Hilbert-norm ball $\|f\|_{\mathcal{H}} \leq K$ for all t.*

AS19 *The feature space $\mathcal{X} \subset \mathbb{R}^p$ and target domain $\mathcal{Y} \subset \mathbb{R}$ are compact, and the reproducing kernel map may be bounded as*

$$\sup_{\mathbf{x} \in \mathcal{X}} \sqrt{\kappa(\mathbf{x}, \mathbf{x})} = X < \infty \tag{7.30}$$

Moreover, the instantaneous losses $\ell_i : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ are C_i-Lipschitz continuous for all $z \in \mathbb{R}$ for a fixed $y \in \mathcal{Y}$

$$|\ell_i(z, y) - \ell_i(z', y)| \leq C_i |z - z'| \tag{7.31}$$

with $C := \max_i C_i$ as the largest modulus of continuity.

AS20 *The local losses $\ell_i(f_i(\mathbf{x}), y)$ are convex and differentiable with respect to the first (scalar) argument $f_i(\mathbf{x})$ on $\mathbb{R}$ for all $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$.*

AS21 *Let $\mathcal{F}_t$ denote the sigma algebra which measures the algorithm history for times $u \leq t$, i.e. $\mathcal{F}_t = \{\mathbf{x}_u, y_u, u_u\}_{u=1}^t$. The projected functional gradient of the instantaneous penalty function defined by stacking* (7.27) *has finite conditional second moments:*

$$\mathbb{E}[\|\tilde{\nabla}_f \hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \mid \mathcal{F}_t] \leq \sigma^2 \tag{7.32}$$

Assumption 18 is necessary to ensure that the algorithm iterates have finite Hilbert-norm for all time. Moreover, Assumption 19 holds in most practical settings by the data domain itself, and justifies the bounding of the loss. Taken together, these conditions permit bounding the optimal function f_c^* in the Hilbert norm, and imply that the worst-case model order is guaranteed to be finite. Variants of Assumption 19 appear in the analysis of stochastic descent methods in the kernelized setting [150,221]. Assumption 20 is satisfied for supervised learning problems such as logistic regression, support vector machines with the square-hinge-loss, the square loss, among others. Moreover, it is a standard condition in the analysis of descent methods (see [139]). Assumption 21 is common in stochastic methods, and ensures that the stochastic approximation error has finite variance.

Next we establish a few auxiliary results needed in the proof of the main results. First, note that the sequence generated by Algorithm 7 and f_c^*, the minimizer of (7.11), are

bounded in Hilbert norm for all t as

$$\|f_t\|_{\mathcal{H}} \leq K \ , \qquad \|f_c^*\|_{\mathcal{H}} \leq K \tag{7.33}$$

by Assumption 18. Next we introduce a proposition which quantifies the error due to sparse projections in terms of the ratio of the sparse approximation budget to the algorithm step-size.

Proposition 5 *Given independent realizations* $(\mathbf{x}_t, \mathbf{y}_t)$ *of the random pair* $(\mathbf{x}, \mathbf{y})$*, the difference between the stacked projected stochastic functional gradient and the its un-projected variant defined by* (7.27) *and* (7.16)*, respectively, is bounded as*

$$\|\tilde{\nabla}_f \hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t) - \nabla_f \hat{\psi}_c(f(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}} \leq \frac{\epsilon_t V}{\eta_t} \tag{7.34}$$

where $\eta_t > 0$ *denotes the algorithm step-size and* $\epsilon_t > 0$ *is the approximation budget parameter of Algorithm 5.*

Proof:

Consider the square-Hilbert-norm difference of the stacked projected stochastic gradient $\tilde{\nabla}_f \hat{\psi}_c(f_t(\mathbf{x}_t), y_t)$ and its un-projected variant $\nabla_f \hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t)$ defined in (7.27) and (7.16), respectively,

$$\begin{aligned}
&\|\tilde{\nabla}_f \hat{\psi}_c(f_t(\mathbf{x}_t), y_t) - \nabla_f \hat{\psi}_c(f(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \qquad (7.35)\\
&= \Big\|\mathrm{vec}\Big(f_{i,t} - \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big[f_{i,t} - \eta_t \nabla_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})\Big]\Big)/\eta_t - \mathrm{vec}\left(\nabla_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})\right)\Big\|_{\mathcal{H}}^2 \\
&\leq V^2 \max_{i \in \mathcal{V}} \Big\|\Big(f_{i,t} - \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big[f_{i,t} - \eta_t \nabla_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})\Big]\Big)/\eta_t - \nabla_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})\Big\|_{\mathcal{H}}^2
\end{aligned}$$

where we apply the fact that the functional gradient is a concatenation of functional gradients associated with each agent in (7.56) for the first equality, and for the second inequality we consider the worst-case estimate across the network. Now, let's focus on the term inside the Hilbert-norm on the right-hand side. Multiply and divide $\nabla_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})$, the last term, by η_t, and reorder terms to write

$$\begin{aligned}
&\Big\|\Big(f_{i,t} - \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big[f_{i,t} - \eta_t \nabla_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})\Big]\Big)/\eta_t - \nabla_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})\Big\|_{\mathcal{H}}^2 \\
&\qquad = \Big\|\frac{1}{\eta_t}\Big(f_{i,t} - \eta_t \nabla_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})\Big) - \frac{1}{\eta_t} \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big[f_{i,t} - \eta_t \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})\Big]\Big\|_{\mathcal{H}}^2 \\
&\qquad\qquad = \frac{1}{\eta_t^2} \|\tilde{f}_{i,t+1} - f_{i,t+1}\|_{\mathcal{H}}^2 \qquad (7.36)
\end{aligned}$$

where we have substituted the definition of $\tilde{f}_{i,t+1}$ and $f_{i,t+1}$ in (7.24) and (7.23), respectively,

and pulled the nonnegative scalar η_t outside the norm. Now, observe that the KOMP residual stopping criterion in Algorithm 5 is $\|\tilde{f}_{i,t+1} - f_{i,t+1}\|_{\mathcal{H}} \leq \epsilon_t$, which we may apply to the last term on the right-hand side of (7.57). This result together with the inequality (7.56) yields (7.34).

■

With the error induced by sparse projections quantified, we may now shift focus to analyzing the Hilbert-norm sub-optimality of the stacked iterates generated by Algorithm 7. We begin by establishing a stochastic descent property of the sequence $\{f_t\}$.

Lemma 7 *(Stochastic Descent) Consider the sequence generated $\{f_t\}$ by Algorithm 7 with $f_0 = 0$. Under Assumptions 18-21, the following expected descent relation holds.*

$$\mathbb{E}\left[\|f_{t+1} - f_c^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq \|f_t - f_c^*\|_{\mathcal{H}}^2 - 2\eta_t[\psi_c(f_t) - \psi_c(f_c^*)] + 2\epsilon_t V\|f_t - f_c^*\|_{\mathcal{H}} + \eta_t^2\sigma^2 \tag{7.37}$$

Proof: Begin by considering the square of the Hilbert-norm difference between f_{t+1} and $f_c^* = \operatorname{argmin} \psi_c(f)$ which minimizes (7.11), and expand the square to write

$$\begin{aligned} \|f_{t+1} - f_c^*\|_{\mathcal{H}}^2 &= \|f_t - \eta_t\tilde{\nabla}_f\hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \\ &= \|f_t - f^*\|_{\mathcal{H}}^2 - 2\eta_t\langle f_t - f_c^*, \tilde{\nabla}_f\hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t)\rangle_{\mathcal{H}} + \eta_t^2\|\tilde{\nabla}_f\hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \end{aligned} \tag{7.38}$$

Add and subtract the functional stochastic gradient of the penalty function $\nabla_f\hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t)$ defined in (7.16) to the second term on the right-hand side of (7.38) to obtain

$$\begin{aligned} \|f_{t+1} - f_c^*\|_{\mathcal{H}}^2 = \|f_t - f_c^*\|_{\mathcal{H}}^2 &- 2\eta_t\langle f_t - f_c^*, \nabla_f\hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t)\rangle_{\mathcal{H}} \\ &- 2\eta_t\langle f_t - f_c^*, \tilde{\nabla}_f\hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t) - \nabla_f\hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t)\rangle_{\mathcal{H}} + \eta_t^2\|\tilde{\nabla}_f\ell(f_t(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \end{aligned} \tag{7.39}$$

We deal with the third term on the right-hand side of (7.39), which represents the directional error associated with the sparse stochastic projections, by applying the Cauchy-Schwartz inequality together with Proposition 5 to obtain

$$\begin{aligned} \|f_{t+1} - f_c^*\|_{\mathcal{H}}^2 = \|f_t - f_c^*\|_{\mathcal{H}}^2 &- 2\eta_t\langle f_t - f_c^*, \nabla_f\hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t)\rangle_{\mathcal{H}} \\ &+ 2\epsilon_t V\|f_t - f_c^*\|_{\mathcal{H}} + \eta_t^2\|\tilde{\nabla}_f\ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}}^2 \end{aligned} \tag{7.40}$$

Now compute the expectation of (7.40) conditional on the algorithm history $\mathcal{F}_t$

$$\mathbb{E}\left[\|f_{t+1} - f_c^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] = \|f_t - f_c^*\|_{\mathcal{H}}^2 + 2\epsilon_t V\|f_t - f_c^*\|_{\mathcal{H}} + \eta_t^2\sigma^2 - 2\eta_t\langle f_t - f_c^*, \nabla_f\psi_c(f_t)\rangle_{\mathcal{H}} \tag{7.41}$$

where we have applied the fact that the stochastic functional gradient in (7.16) is an unbiased estimator [cf. (7.29)] for the functional gradient of the penalty function in (7.11), as well as the fact that the variance of the functional projected stochastic gradient is finite stated in (7.32) (Assumption 21). Observe that since $\psi_c(f)$ is an expectation of a convex function, it is also convex, which allows us to write

$$\psi_c(f_t) - \psi_c(f_c^*) \leq \langle f_t - f_c^*, \nabla_f \psi_c(f_t) \rangle_{\mathcal{H}} \ , \tag{7.42}$$

which we substitute into the second term on the right-hand side of the relation given in (7.41) to obtain

$$\mathbb{E}\left[\|f_{t+1} - f_c^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq \|f_t - f_c^*\|_{\mathcal{H}}^2 - 2\eta_t[\psi_c(f_t) - \psi_c(f_c^*)] + 2\epsilon_t V \|f_t - f_c^*\|_{\mathcal{H}} + \eta_t^2 \sigma^2 \ . \tag{7.43}$$

Thus the claim in Lemma 7 is valid.

■

Now that Lemma 7 establishes a descent-like property, we may apply the proof of Theorem 1 in [98] for the sequence $\|f_t - f_c^*\|_{\mathcal{H}}$, and thus we have the following as a corollary.

Corollary 2 *Consider the sequence $\{f_t\}$ generated by Algorithm 7 with $f_0 = 0$ and regularizer $\lambda > 0$. Under Assumptions 18-21, with diminishing step-sizes and compression budget, i.e.,*

$$\sum_{t=0}^{\infty} \eta_t = \infty \ , \quad \sum_{t=0}^{\infty} \eta_t^2 < \infty \ , \quad \epsilon_t = \eta_t^2 \ , \tag{7.44}$$

and $\eta_t < 1/\lambda$, the sequence converges exactly to the minimizer to the penalty function [cf. (7.11)]: $f_t \to f_c^*$ with probability 1.

To attain exact convergence to the minimizer of the penalty, f_c^*, we require the compression budget determining the error ϵ_t incurred by sparse projections to approach null. This means that to have exact convergence, we require the function representation to require an increasing amount of memory which is, in the limit, of infinite complexity. In contrast, when constant step-size and compression budget are used, then the algorithm settles to a neighborhood, as we state next.

Theorem 8 *Consider the sequence $\{f_t\}$ generated by Algorithm 7 with $f_0 = 0$ and regularizer $\lambda > 0$. Suppose Assumptions 18-21 hold, and we select a constant step-size $\eta_t = \eta < 1/\lambda$ and compression budget $\epsilon_t = \epsilon$ chosen such that $\epsilon = K\eta^{3/2}$ for an arbitrary positive constant*

K. Then we have convergence to a neighborhood with probability 1 *as*

$$\liminf_t \|f_t - f_c^*\|_{\mathcal{H}} \leq \frac{\sqrt{\eta}}{\lambda}\left[KV + \sqrt{K^2V^2 + \lambda\sigma^2}\right] = \mathcal{O}(\sqrt{\eta}) \ a.s. \tag{7.45}$$

Proof:

The use of the regularizing term $(\lambda/2)\|f\|_{\mathcal{H}}^2$ in (7.11) implies that the penalty is λ-strongly convex with respect to $f \in \mathcal{H}$, which allows us to write

$$\frac{\lambda}{2}\|f_t - f_c^*\|_{\mathcal{H}}^2 \leq \psi_c(f_t) - \psi_c(f_c^*) \tag{7.46}$$

Substituting the relation (7.46) into the second term on the right-hand side of the expected descent relation stated in Lemma 7, with constant step-size $\eta_t = \eta$ and approximation budget $\epsilon_t = \epsilon$, yields

$$\mathbb{E}[\|f_{t+1} - f_c^*\|_{\mathcal{H}}^2 \,\big|\, \mathcal{F}_t] \leq (1 - \eta\lambda)\|f_t - f_c^*\|_{\mathcal{H}}^2 + 2\epsilon V\|f_t - f_c^*\|_{\mathcal{H}} + \eta^2\sigma^2 \ . \tag{7.47}$$

The expression in (7.47) may be used to construct a stopping stochastic process , which tracks the suboptimality of $\|f_t - f_c^*\|_{\mathcal{H}}^2$ until it reaches a specific threshold, as in the proof of Theorem 2 of [98]. In doing so, we obtain convergence to a neighborhood. We may define a stochastic process δ_t that qualifies as a supermartingale, i.e. $\mathbb{E}\left[\delta_{t+1} \,\middle|\, \mathcal{F}_t\right] \leq \delta_t$ by considering (7.47) and solving for the appropriate threshold by analyzing when the following holds true

$$\begin{aligned}\mathbb{E}[\|f_{t+1} - f_c^*\|_{\mathcal{H}}^2 \,\big|\, \mathcal{F}_t] &\leq (1 - \eta\lambda)\|f_t - f_c^*\|_{\mathcal{H}}^2 + 2\epsilon V\|f_t - f_c^*\|_{\mathcal{H}} + \eta^2\sigma^2 \\ &\leq \|f_t - f_c^*\|_{\mathcal{H}}^2 \ .\end{aligned} \tag{7.48}$$

which may be rearranged to obtain the sufficient condition

$$-\eta\lambda\|f_t - f_c^*\|_{\mathcal{H}}^2 + 2\epsilon V\|f_t - f_c^*\|_{\mathcal{H}} + \eta^2\sigma^2 \leq 0 \ . \tag{7.49}$$

Note that (7.49) defines a quadratic polynomial in $\|f_t - f_c^*\|_{\mathcal{H}}$, which, using the quadratic formula, has roots

$$\|f_t - f_c^*\|_{\mathcal{H}} = \frac{\epsilon V \pm \sqrt{\epsilon^2V^2 + \lambda\eta^3\sigma^2}}{\lambda\eta} \tag{7.50}$$

Observe (7.49) is a downward-opening polynomial in $\|f_t - f_c^*\|_{\mathcal{H}}$ which is nonnegative. Thus, focus on the positive root, substituting the approximation budget selection $\epsilon = K\eta^{3/2}$ to

define the radius of convergence as

$$\Delta := \frac{\epsilon V + \sqrt{\epsilon^2 V^2 + \lambda\eta^3\sigma^2}}{\lambda\eta} = \frac{\sqrt{\eta}}{\lambda}\Big(KV + \sqrt{K^2V^2 + \lambda\sigma^2}\Big) \tag{7.51}$$

(7.51) allows us to construct a stopping process: define the process δ_t as

$$\delta_t = \|f_t - f_c^*\|_{\mathcal{H}} \mathbb{1}\Big\{ \min_{u\leq t} -\eta\lambda\|f_u - f_c^*\|_{\mathcal{H}}^2 + 2\epsilon V\|f_u - f_c^*\|_{\mathcal{H}} + \eta^2\sigma^2 > \Delta \Big\} \tag{7.52}$$

where $\mathbb{1}\{E\}$ denotes the indicator process of event $E \in \mathcal{F}_t$. Note that $\delta_t \geq 0$ for all t, since both $\|f_t - f^*\|_{\mathcal{H}}$ and the indicator function are nonnegative. The rest of the proof applies the same reasoning as that of Theorem 2 in [98]: in particular, given the definition (7.52), either $\min_{u\leq t} -\eta\lambda\|f_u - f_c^*\|_{\mathcal{H}}^2 + 2\epsilon V\|f_u - f_c^*\|_{\mathcal{H}} + \eta^2\sigma^2 > \Delta$ holds, in which case we may compute the square root of the condition in (7.48) to write

$$\mathbb{E}[\delta_{t+1} \,\big|\, \mathcal{F}_t] \leq \delta_t \tag{7.53}$$

Alternatively, $\min_{u\leq t} -\eta\lambda\|f_u - f_c^*\|_{\mathcal{H}}^2 + 2\epsilon V\|f_u - f_c^*\|_{\mathcal{H}} + \eta^2\sigma^2 \leq \Delta$, in which case the indicator function is null for all $s \geq t$ from the use of the minimum inside the indicator in (7.52). Thus in either case, (7.53) is valid, implying δ_t converges almost surely to null, which, as a consequence we obtain the fact that either $\lim_{t\to\infty} \|f_t - f_c^*\|_{\mathcal{H}} - \Delta = 0$ or the indicator function is null for large t, i.e. $\lim_{t\to\infty} \mathbb{1}\{\min_{u\leq t} -\eta\lambda\|f_u - f_c^*\|_{\mathcal{H}}^2 + 2\epsilon V\|f_u - f_c^*\|_{\mathcal{H}} + \eta^2\sigma^2 > \Delta\} = 0$ almost surely. Therefore, we obtain that

$$\liminf_{t\to\infty} \|f_t - f_c^*\|_{\mathcal{H}} \leq \Delta = \frac{\sqrt{\eta}}{\lambda}\Big(KV + \sqrt{K^2 + \lambda\sigma^2}\Big) \;\text{ a.s. }, \tag{7.54}$$

as stated in Theorem 8.

■

Empirically, the use of constant step-sizes has the effect of maintaining consistent algorithm adaptivity in the face of new data, at the cost of losing exact convergence. But this drawback is more than compensated for by the fact that in this case we may apply Theorem 3 of [98], which guarantees the model order of the function sequence remains finite, and in the worst case, is related to the covering number of the data domain

Corollary 3 *Denote $f_t \in \mathcal{H}^V$ as the stacked function sequence defined by Algorithm 7 with constant step-size $\eta_t = \eta < 1/\lambda$ and approximation budget $\epsilon = K\eta^{3/2}$ where $K > 0$ is an arbitrary positive scalar. Let M_t be the model order of the stacked function f_t i.e., the number of columns of the dictionary $\mathbf{D}_t$ which parameterizes f_t. Then there exists a finite upper bound M^∞ such that, for all $t \geq 0$, the model order is always bounded as $M_t \leq M^\infty$.*

Consequently, the model order of the limiting function $f_c^\infty = \lim_t f_t$ is finite.

Thus, only constant step-sizes attain a reasonable tradeoff between performance relative to f_c^* and the complexity of storing the function sequence $\{f_t\}$: in this setting, we obtain approximate convergence to f_c^* while ensuring the memory requirements are always finite, as stated in Corollary 3.

We are left to analyze the goodness of the solution f_c^* as an approximation of the solution of the original problem (7.3). In particular, we establish consensus in the mean square sense. Let us start by establishing that the penalty term is bounded by a p^*/c, where p^* is the primal value of the optimization problem (7.3) and c is the barrier parameter introduced in (7.11). We formalize this result next.

Proposition 6 *Let Assumptions 18 - 21 hold. Let f_c^* be the minimizer of the penalty function* (7.11) *and let p^* be the primal optimal value of* (7.3)*. Then, it holds that*

$$\frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in n_i}\mathbb{E}_{\mathbf{x}_i}\left\{[f_{c,i}^*(\mathbf{x}_i)-f_{c,j}^*(\mathbf{x}_i)]^2\right\}\leq\frac{p^*}{c}. \tag{7.55}$$

Proof: Consider the square-Hilbert-norm difference of the stacked projected stochastic gradient $\tilde{\nabla}_f\hat{\psi}_c(f_t(\mathbf{x}_t),y_t)$ and its un-projected variant $\nabla_f\hat{\psi}_c(f_t(\mathbf{x}_t),\mathbf{y}_t)$ defined in (7.27) and (7.16), respectively,

$$\begin{aligned}
&\|\tilde{\nabla}_f\hat{\psi}_c(f_t(\mathbf{x}_t),y_t)-\nabla_f\hat{\psi}_c(f(\mathbf{x}_t),\mathbf{y}_t)\|_{\mathcal{H}}^2 \qquad (7.56)\\
&=\left\|\mathrm{vec}\Big(f_{i,t}-\mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big[f_{i,t}-\eta_t\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\Big]\Big)/\eta_t-\mathrm{vec}\left(\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\right)\right\|_{\mathcal{H}}^2\\
&\leq V^2\max_{i\in\mathcal{V}}\left\|\Big(f_{i,t}-\mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big[f_{i,t}-\eta_t\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\Big]\Big)/\eta_t-\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\right\|_{\mathcal{H}}^2
\end{aligned}$$

where we apply the fact that the functional gradient is a concatenation of functional gradients associated with each agent in (7.56) for the first equality, and for the second inequality we consider the worst-case estimate across the network. Now, let's focus on the term inside the Hilbert-norm on the right-hand side. Multiply and divide $\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})$, the last term, by η_t, and reorder terms to write

$$\begin{aligned}
&\left\|\Big(f_{i,t}-\mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big[f_{i,t}-\eta_t\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\Big]\Big)/\eta_t-\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\right\|_{\mathcal{H}}^2\\
&\quad=\left\|\frac{1}{\eta_t}\Big(f_{i,t}-\eta_t\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\Big)-\frac{1}{\eta_t}\mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big[f_{i,t}-\eta_t\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\Big]\right\|_{\mathcal{H}}^2\\
&\quad=\frac{1}{\eta_t^2}\|\tilde{f}_{i,t+1}-f_{i,t+1}\|_{\mathcal{H}}^2 \qquad (7.57)
\end{aligned}$$

where we have substituted the definition of $\tilde{f}_{i,t+1}$ and $f_{i,t+1}$ in (7.24) and (7.23), respectively,

and pulled the nonnegative scalar η_t outside the norm. Now, observe that the KOMP residual stopping criterion in Algorithm 5 is $\|\tilde{f}_{i,t+1} - f_{i,t+1}\|_{\mathcal{H}} \leq \epsilon_t$, which we may apply to the last term on the right-hand side of (7.57). This result together with the inequality (7.56) yields (7.34). ■

Proposition 6 establishes a relationship between the choice of penalty parameter c and constraint satisfaction. This result may be used to attain convergence in mean square of each individual agent's regression function to ones which coincide with one another. Under an additional hypothesis, we obtain exact consensus, as we state next.

Theorem 9 *Let Assumptions 18 - 21 hold. Let f_c^* be the minimizer of the penalty function* (7.11)*. Then, suppose the penalty parameter c in* (7.11) *approaches infinity $c \to \infty$, and that the node-pair differences $f_{i,c}^* - f_{j,c}^*$ are not orthogonal to mean transformation $\mathbb{E}_{\mathbf{x}_i}[\kappa(\mathbf{x}_i, \cdot)]$ of the local input spaces $\mathbf{x}_i$ for all $(i,j) \in \mathcal{E}$. Then $f_{i,c}^* = f_{j,c}^*$ for all $(i,j) \in \mathcal{E}$.*

Proof: As a consequence, the limit of (7.55) when c tends to infinity yields consensus in L^2 sense, i.e.,

$$\lim_{c\to\infty} \frac{1}{2} \sum_{i\in\mathcal{V}} \sum_{j\in n_i} \mathbb{E}_{\mathbf{x}_i} \left\{ [f_{c,i}^*(\mathbf{x}_i) - f_{c,j}^*(\mathbf{x}_i)]^2 \right\} = 0, \tag{7.58}$$

which, by puling the limit outside the sum in (7.58), is equivalent to

$$\lim_{c\to\infty} \mathbb{E}_{\mathbf{x}_i} \left\{ [f_{c,i}^*(\mathbf{x}_i) - f_{c,j}^*(\mathbf{x}_i)]^2 \right\} = 0 \tag{7.59}$$

for all i, j. Consensus in the mean square sense is a less stringent constraint that equality in the Hilbert norm as desired in (7.3). In particular, for any $(i,j) \in \mathcal{E}$ if $f_i = f_j$ consensus in the mean square sense is satisfied as well. Moreover, by applying the reproducing property of the kernel (7.6)(i), we have

$$\begin{aligned} 0 &= \lim_{c\to\infty} \mathbb{E}_{\mathbf{x}_i} \left\{ \left| < f_{c,i}^* - f_{c,j}^*, k(\mathbf{x}_i, \cdot) > \right| \right\} \\ &\geq \lim_{c\to\infty} \left| \mathbb{E}_{\mathbf{x}_i} \left\{ < f_{c,i}^* - f_{c,j}^*, k(\mathbf{x}_i, \cdot) > \right\} \right| \\ &= \lim_{c\to\infty} \left| < f_{c,i}^* - f_{c,j}^*, \mathbb{E}_{\mathbf{x}_i} k(\mathbf{x}_i, \cdot) > \right| \end{aligned} \tag{7.60}$$

where in the previous expression we pull the absolute value outside the expectation, and in the later we apply linearity of the expectation. Thus, (7.60) implies consensus is achieved with respect to the Hilbert norm, whenever the function differences $f_{c,i}^* - f_{c,j}^*$ are not orthogonal to $\mathbb{E}_{\mathbf{x}_i}[\kappa(\mathbf{x}_i, \cdot)]$, the mean of the transformation of the local input data $\mathbf{x}_i$. ■

We have established that Algorithm 7 yields convergent behavior to an approximate solution to the problem (7.3) defined by the penalty function (7.11) when both diminishing

Table 7.1: Stability for decentralized kernel methods for different parameter selections.

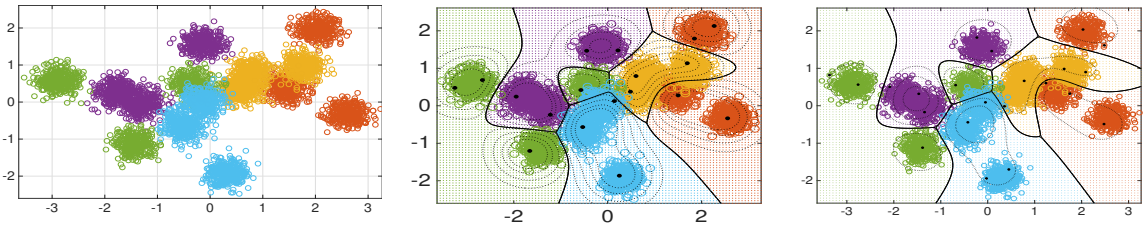
	Diminishing	Constant
Learning rate	$\eta_t = \mathcal{O}(1/t)$	$\eta_t = \eta > 0$
Compression Budget	$\epsilon_t = \eta_t^2$	$\epsilon = \mathcal{O}(\eta^{3/2})$
Regularizer	$\eta_t < 1/\lambda$	$\eta < 1/\lambda$
Convergence Result	$f_t \to f_c^*$ a.s.	$\liminf_t \|f_t - f_c^*\| = \mathcal{O}(\sqrt{\eta})$ a.s.
Model Order	None	Finite
Consensus	Attained for $c \to \infty$	Fixed $c > 0$: not achieved

and constant learning rates are used. When the learning rate η_t satisfies $\eta_t < 1/\lambda$ with $\lambda > 0$ as the regularizer, and is attenuating in the classical stochastic approximation conditions $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$, i.e., $\eta_t = \mathcal{O}(1/t)$, the compression budget ϵ_t of Algorithm 5 must satisfy $\epsilon_t = \eta_t^2$ [cf. (6.43)]. Practically speaking, this means that asymptotically the iterates of each agent generated by Algorithm 7 may have a very large model order in the diminishing step-size regime, since the approximation budget is vanishing as $\epsilon_t = \mathcal{O}(1/t^2)$. In contrast, when a constant algorithm step-size $\eta_t = \eta$ is chosen to satisfy $\eta < 1/\lambda$, then we only require the approximation budget $\epsilon_t = \epsilon$ to satisfy $\epsilon = \mathcal{O}(\eta^{3/2})$. This means that in the constant learning rate regime, we obtain a function sequence which converges to a neighborhood of the optimal f_c^* defined by (7.11) and is guaranteed to have a finite model order. Furthermore, when we send the penalty parameter $c \to \infty$, the function estimates of each agent satisfy the consensus constraints, provided that $f_{i,c}^* - f_{j,c}^*$ are not orthogonal to $\kappa(\mathbf{x}_i, \cdot)$ for each $(i, j) \in \mathcal{E}$ (Theorem 9). These results are summarized in Table 8.1.

7.4 Experiments with Decentralized Kernel Learning

Kernel Logistic Regression We consider the task of kernel logistic regression from multi-class training data that is scattered across a multi-agent system. In this case, the merit of a particular regressor for agent i is quantified by its contribution to the class-conditional probability. We define a set of class-specific activation functions $f_{i,d} : \mathcal{X} \to \mathbb{R}$, and denote them jointly as $\mathbf{f}_i \in \mathcal{H}^D$, where $\{1, \ldots, D\}$ denotes the set of classes. Then, as in the example in Section 7.1, define the probabilistic model

$$P(y_i = d \,|\, \mathbf{x}_i) := \frac{\exp(f_{i,d}(\mathbf{x}_i))}{\sum_{d'} \exp(f_{i,d'}(\mathbf{x}_i))}. \tag{7.61}$$



(a) Gaussian Mixtures data. (b) Logistic Decision surface. (c) Hinge Decision surface.

Figure 7.1: Visualizations of the Gaussian mixture data set (Figure 7.1(a)) as in [227] and the learned low-memory multi-class kernel logistic regressor of a randomly chosen agent in the network (Figure 7.1(b)), which attains 95.2% classification accuracy on a hold-out test set. Curved black lines denote decision boundaries between classes; dotted lines denote confidence intervals; bold black dots denote kernel dictionary elements. Kernel dictionary elements concentrate at the modes of the Gaussian clusters and near points of overlap between classes. In Figure 7.1(c) we plot the resulting decision surface learned by kernel SVM which attains 95.7% accuracy – the state of the art.

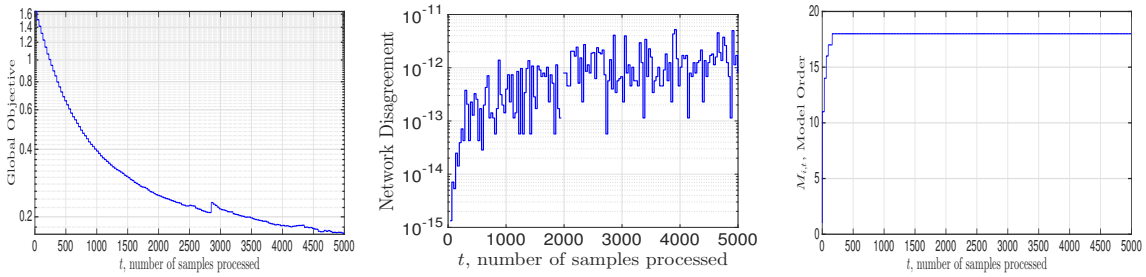
which models the odds ratio of a sample being in class d versus all others. The negative log likelihood defined by (7.61) is the instantaneous loss (see, e.g., [133]) at sample $(\mathbf{x}_{i,n}, y_{i,n})$:

$$\ell_i(\mathbf{f}_i, \mathbf{x}_{i,n}, y_{i,n}) = -\log P(y_i = y_{i,n}|\mathbf{x}_{i,n}) + \frac{\lambda}{2}\sum_d \|f_{i,d}\|^2_{\mathcal{H}} \tag{7.62}$$

The loss (7.62) substituted into the empirical risk minimization problem in the example in Section 7.1 is its generalization to multi-class problems. For a given set of activation functions, classification decisions $\tilde{d}$ for $\mathbf{x}_i$ is given by the maximum likelihood estimate, i.e., $\tilde{d} = \text{argmax}_{d\in\{1,\dots,D\}} f_{i,d}(\mathbf{x})$.

Following [98,227], we generate a data set from Gaussian mixture models, which consists $N = 5000$ feature-label pairs for training and 2500 for testing. Each label y_n was drawn uniformly at random from the label set. The corresponding feature vector $\mathbf{x}_n \in \mathbb{R}^p$ was then drawn from a planar ($p = 2$), equitably-weighted Gaussian mixture model, i.e., $\mathbf{x}\,\big|\,y \sim (1/3)\sum_{j=1}^3 \mathcal{N}(\boldsymbol{\mu}_{y,j}, \sigma^2_{y,j}\mathbf{I})$ where $\sigma^2_{y,j} = 0.2$ for all values of y and j. The means $\boldsymbol{\mu}_{y,j}$ are themselves realizations of their own Gaussian distribution with class-dependent parameters, i.e., $\boldsymbol{\mu}_{y,j} \sim \mathcal{N}(\boldsymbol{\theta}_y, \sigma^2_y\mathbf{I})$, where $\{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_D\}$ are equitably spaced around the unit circle, one for each class label, and $\sigma^2_y = 1.0$. We fix the number of classes $D = 5$, meaning that the feature distribution has, in total, 15 distinct modes. The data points are plotted in Figure 7.1(a).

Each agent in a $V = 20$ network observes a unique stream of training examples from this common data set. Here the communications graph is a random network with edges generated randomly between nodes with probability 1/5 repeatedly until we obtain one that is connected, and then symmetrize it. We run Algorithm 7 when the entire training set is fed to each agent in a streaming fashion, a Gaussian kernel is used with bandwidth



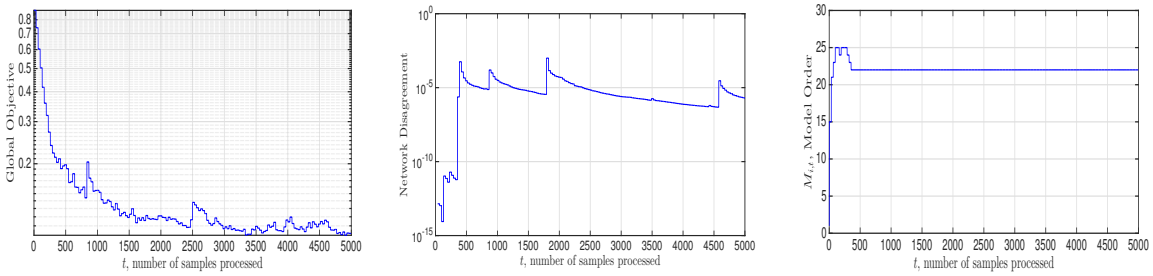
(a) Global objective vs. samples (b) Disagreement vs. samples (c) Model Order $M_{i,t}$ vs. samples

Figure 7.2: In Fig. 7.2(a), we plot the global objective $\sum_{i\in\mathcal{V}}(\mathbb{E}_{\mathbf{x}_i,y_i}[\ell_i(f_{i,t}(\mathbf{x}),y_i)])$ versus the number of samples processed, and observe convergence. In Fig. 7.2(b) we display the Hilbert-norm network disagreement $\sum_{(i,j)\in\mathcal{E}}\|f_{i,t}-f_{j,t}\|_{\mathcal{H}}^2$ with a penalty parameter c that doubles every 200 samples. As c increases, agents attain consensus. In Fig. 7.2(c), we plot the model order of a randomly chosen agent's regression function, which stabilizes to 18 after 162 samples.

$\tilde{\sigma}^2 = 0.6$, with constant learning rate $\eta = 3$, compression budget chosen as $\epsilon = \eta^{3/2}$ with parsimony constant $K = 0.04$, mini-batch size 32, and regularizer $\lambda = 10^{-6}$. The penalty coefficient is initialized as $c = 0.01$ and doubled after every 200 training examples.

We plot the results of this implementation in Figures 7.1(b) and 7.2. In Figure 7.2(a), we plot the global objective $\sum_{i\in\mathcal{V}}(\mathbb{E}_{\mathbf{x}_i,y_i}[\ell_i(f_{i,t}(\mathbf{x}),y_i)])$ relative to the number of training examples processed, and observe stable convergence to a global minimum. In Figure 7.2(b) we display Hilbert-norm network disagreement $\sum_{(i,j)\in\mathcal{E}}\|f_{i,t}-f_{j,t}\|_{\mathcal{H}}^2$ versus observed sample points. Since each regression function is initialized as null, initially the disagreement is trivially null, but it remains small over the function sample path as model training occurs. Moreover, the model order of an arbitrarily chosen agent $i = 15$ versus samples processed is given in Figure 7.2(c): observe that the model order stabilizes after only a couple hundred training examples to 18, which is only a couple more than 15, the number of modes of the joint data density function. The resulting decision surface of node 15 is given in Figure 7.1(b), which achieves 95.2% classification accuracy on the test set which is comparable to existing centralized batch approaches (see Table 2 of [98]) to kernel logistic regression.

Kernel Support Vector Machines Now we address the problem of training a multi-class kernel support vector machine online in a multi-agent systems. The merit of a particular regressor is defined by its ability to maximize its classification margin, which may be formulated by first defining a set of class-specific activation functions $f_{i,d} : \mathcal{X} \to \mathbb{R}$, and denote them jointly as $\mathbf{f}_i \in \mathcal{H}^D$. In Multi-KSVM, points are assigned the class label of the activation function that yields the maximum response. KSVM is trained by taking the instantaneous loss ℓ to be the multi-class hinge function which defines the margin separating



(a) Global objective vs. samples (b) Disagreement vs. samples (c) Model Order $M_{i,t}$ vs. samples

Figure 7.3: In Fig. 7.3(a), we plot the global objective $\sum_{i\in\mathcal{V}}(\mathbb{E}_{\mathbf{x}_i,y_i}[\ell_i(f_{i,t}(\mathbf{x}),y_i)])$ versus the number of samples processed, and observe convergence, albeit more noisily than for the differentiable logistic loss. In Fig. 7.3(b) we display the Hilbert-norm network disagreement $\sum_{(i,j)\in\mathcal{E}}\|f_{i,t}-f_{j,t}\|_{\mathcal{H}}^2$ with a penalty parameter c that doubles every 200 samples. As c increases, agents attain consensus with respect to the Hilbert norm. In Fig. 7.3(c), we plot the model order of a randomly chosen agent's regression function, which stabilizes to 22 after 354 samples. Here we obtain a slightly higher complexity classifier that achieves slightly better accuracy.

hyperplane in the kernelized feature space, i.e.,

$$\ell_i(\mathbf{f}_i,\mathbf{x}_n,y_n)=\max(0,1+f_{i,r}(\mathbf{x}_n)-f_{i,y_n}(\mathbf{x}_n))+\lambda\sum_{d'=1}^{D}\|f_{i,d'}\|_{\mathcal{H}}^2\,, \tag{7.63}$$

where $r=\operatorname{argmax}_{d'\neq y}f_{i,d'}(\mathbf{x})$. Further details may be found in [133].

We again consider an implementation where each agent in a $V=20$ network observes a unique stream of training examples from the Gaussian mixtures data set (see Figure 7.1(a)). Moreover, the communications graph is fixed as a random network with edges generated randomly between nodes with probability 1/5 repeatedly until we obtain one that is connected, and then symmetrize it. We run Algorithm 7 when the entire training set is fed to each agent in a streaming fashion, a Gaussian kernel is used with bandwidth $\tilde{\sigma}^2=0.6$, with constant learning rate $\eta=3$, compression budget chosen as $\epsilon=\eta^{3/2}$ with parsimony constant $K=0.04$, mini-batch size 32, and regularizer $\lambda=10^{-6}$. The penalty coefficient is initialized as $c=0.01$ and doubled after every 200 training examples.

We plot the results of this implementation in Figures 7.1(c) and 7.3. In Figure 7.3(a), we observe that the global objective $\sum_{i\in\mathcal{V}}(\mathbb{E}_{\mathbf{x}_i,y_i}[\ell_i(f_{i,t}(\mathbf{x}),y_i)])$ converges stably to a global minimum as the number of samples processed increases. In Figure 7.3(b) we display Hilbert-norm network disagreement $\sum_{(i,j)\in\mathcal{E}}\|f_{i,t}-f_{j,t}\|_{\mathcal{H}}^2$ versus observed sample points. Since each regression function is initialized as null, initially the disagreement is trivially null, but it remains small over the function sample path as model training occurs, and periodically spikes when the penalty parameter is increased. Moreover, the model order of an arbitrarily chosen agent $i=6$ versus samples processed is given in Figure 7.3(c): the model order

stabilizes after only a couple hundred training examples to 22, which is only a couple more than 15, the number of modes of the joint data density function. The resulting decision surface of node 6 is given in Figure 7.1(c), which achieves 95.7% classification accuracy on the test set which is comparable to existing centralized batch approaches.

7.5 Perspectives on Efficient Multi-Agent Kernel Learning

In this chapter, we extended the ideas in Chapter 6 to multi-agent settings with the intent of developing a method such that a network of autonomous agents, based on their local data stream, may learn a kernelized statistical model which optimal with respect to information aggregated across the entire network. To do so, we proposed an unusual penalty function whose structure is amenable to efficient parameterizations when developing stochastic approximation-based updates. By applying functional stochastic gradient method to this node-separable penalty combined with greedily constructed subspace projections, we obtain a decentralized online algorithm for memory-efficient nonparametric function approximation that is globally convergent. We obtain a controllable trade-off between optimality and memory requirements through the design of the greedy subspace projections. Moreover, under specific selections of the penalty parameter, agents achieve consensus.

The empirical performance of this protocol, the Greedy Projected Penalty Method, yields state of the art statistical accuracy for a team of interconnected agents learning from streaming data for both multi-class kernel logistic regression and multi-class kernel support vector machines problems. The importance of these results is that they provide a mathematical and empirical foundation for accurate and stable multi-agent statistical inference in streaming data settings while preserving memory-efficiency.

In the following chapter, we shift gears towards applying the nonparametric function approximation techniques developed in Chapter 6 to address deal with long-standing issues related to statistical control via dynamic programming. The motivation for this shift comes from the fact that to design truly intelligent behavior in an autonomous system, it is not enough to make accurate inferences, but rather, we would like it to augment its behavior over time based on rewards and incentives. It is left to future work to develop a formulation for multi-agent statistical control based on decentralized nonparametric stochastic optimization.

Chapter 8

From Inference to Control: Markov Decision Processes

In this chapter, we change course from all previous chapters. Up until now, we have addressed the problem of statistical inference/learning from streaming data, that is, from a training example $\mathbf{x}_n$, we seek to predict y_n as $\hat{y}_n = f(\mathbf{x}_n)$. Based on the function class $\mathscr{F}$ to which f belongs, the difficulty of the optimization problem that defines finding f varies, as does its statistical performance. In this chapter, we build upon the lessons learned regarding how to use reproducing kernel Hilbert spaces for statistical inference to develop a provably stable method for *statistical control*. Our motivation for changing course comes from the fact that for an autonomous system learn truly intelligent behavior, making good predictions is not enough. In addition, we would like an autonomous agent to augment its behavior over time based on rewards and incentives. A framework to begin doing so is developed in this chapter by building upon the memory-efficient methods for nonparametric stochastic programming developed in Chapter 6.

8.1 Policy Evaluation in Markov Decision Processes

We consider an autonomous agent acting in an environment defined by a Markov decision process (MDP) [185] with continuous spaces, which is increasingly relevant to emerging technologies such as robotics [90], power systems [173], and others. A MDP is a quintuple $(\mathcal{X}, \mathcal{A}, \mathbb{P}, r, \gamma)$, where $\mathbb{P}$ is the action-dependent transition probability of the process: when the agent starts in state $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^p$ at time t and takes an action $\mathbf{a}_t \in \mathcal{A}$, a transition to next state $\mathbf{y}_t \in \mathcal{X}$ is distributed according to $\mathbf{y}_t \sim \mathbb{P}(\cdot \,\big|\, \mathbf{x}_t, \mathbf{a}_t)$. After the agent transitions to a particular $\mathbf{y}_t$, the MDP provides to it an instantaneous reward $r(\mathbf{x}_t, \mathbf{a}_t, \mathbf{y}_t)$, where the reward function is a map $r : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$.

We focus on the problem of *policy evaluation*: control decisions $\mathbf{a}_t$ are chosen according to

a fixed stationary stochastic policy $\pi : \mathcal{X} \to \rho(\mathcal{A})$, where $\rho(\mathcal{A})$ denotes the set of probability distributions over $\mathcal{A}$. Policy evaluation underlies methods that seek optimal policies through repeated evaluation and improvement [104]. In policy evaluation, we seek to compute the *value* of a policy when starting in state $\mathbf{x}$, quantified by the discounted expected sum of rewards, or value function $V^\pi(\mathbf{x})$:[1]

$$V^\pi(\mathbf{x}) = \mathbb{E}_{\mathbf{y}}\Big[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{x}_t, \mathbf{a}_t, \mathbf{y}_t) \,\Big|\, \mathbf{x}_0 = \mathbf{x}, \{\mathbf{a}_t = \pi(\mathbf{x}_t)\}_{t=0}^{\infty}\Big]. \tag{8.1}$$

For a single trajectory through the state space $\mathcal{X}$, $\mathbf{y}_t = \mathbf{x}_{t+1}$. The value function (8.1) is parameterized by a discount factor $\gamma \in (0, 1)$, which determines the agent's farsightedness. Decomposing the summand in (8.1) into its first and subsequent terms, and using both the stationarity of the transition probability and the Markov property yields the Bellman evaluation equation [20]:

$$V^\pi(\mathbf{x}) = \int_{\mathcal{X}} [r(\mathbf{x}, \pi(\mathbf{x}), \mathbf{y}) + \gamma V^\pi(\mathbf{y})] \mathbb{P}(d\mathbf{y} \,|\, \mathbf{x}, \pi(\mathbf{x})) \;\text{ for all } \mathbf{x} \in \mathcal{X}, \tag{8.2}$$

The right-hand side of (8.2) defines a Bellman evaluation operator $\mathscr{B}^\pi : \mathcal{B}(\mathcal{X}) \to \mathcal{B}(\mathcal{X})$ over $\mathcal{B}(\mathcal{X})$, the space of bounded continuous value functions $V : \mathcal{X} \to \mathbb{R}$:

$$(\mathscr{B}^\pi V)(\mathbf{x}) = \int_{\mathcal{X}} [r(\mathbf{x}, \pi(\mathbf{x}), \mathbf{y}) + \gamma V(\mathbf{y})] \mathbb{P}(d\mathbf{y} \,|\, \mathbf{x}, \pi(\mathbf{x})) \;\text{ for all } \mathbf{x} \in \mathcal{X}, \tag{8.3}$$

[23][Proposition 4.2(b)] establishes that the stationary point of (8.3) is V^π, i.e., $(\mathscr{B}^\pi V^\pi)(\mathbf{x}) = V^\pi(\mathbf{x})$. As a stepping stone to finding optimal policies in infinite MDPs, we seek here to find the fixed point of (8.3). Specifically, the goal of this work is stable value function estimation in infinite MDPs, with nonlinear parameterizations that are allowed to be infinite, but are nonetheless memory-efficient.

Challenges To solve (8.3), fixed point methods, i.e., value iteration ($V_{k+1} = \mathscr{B}^\pi V_k$), have been proposed [23], but only apply when the value function can be represented by a vector whose length is defined by the number of states and the state space is small enough that the expectation[2] in $\mathscr{B}$ can be computed. For large spaces, stochastic approximations of value iteration, i.e., temporal difference (TD) learning [184], have been utilized to circumvent this intractable expectation. Incremental methods (least-squares TD) provide an alternative when $V(\mathbf{x})$ has a finite linear parameterization [35], but their extensions to infinite representations require infinite memory [153] or elude stability [214].

[1] In MDPs more generally, we choose actions $\{\mathbf{a}_t\}_{t=1}^{\infty}$ to maximize the reward accumulation starting from state $\mathbf{x}$, i.e., $V(\mathbf{x}, \{\mathbf{a}_t\}_{t=0}^{\infty}) = \mathbb{E}_{\mathbf{y}}\Big[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{x}_t, \mathbf{a}_t, \mathbf{y}_t) \,\Big|\, \mathbf{x}_0 = \mathbf{x}, \{\mathbf{a}_t\}_{t=0}^{\infty}\Big]$. For fixed π, this simplifies to (8.1).

[2] The integral in (8.2) defines a conditional expectation: $V^\pi(\mathbf{x}) = \mathbb{E}_{\mathbf{y}}[r(\mathbf{x}, \pi(\mathbf{x}), \mathbf{y}) + \gamma V^\pi(\mathbf{y})] \,|\, \mathbf{x}, \pi(\mathbf{x})]$.

Solving the fixed point problem defined by (8.3) requires surmounting the fact that this expression is defined for each $\mathbf{x} \in \mathcal{X}$, which for continuous $\mathcal{X} \subset \mathbb{R}^p$ has *infinitely many* unknowns. This phenomenon is one example of Bellman's curse of dimensionality [20], and it is frequently sidestepped by parameterizing the value function using a finite linear [125,200] or nonlinear [24] basis expansion. Such methods have paved the way for the recent success of neural networks in value function-based approaches to MDPs [127], but combining TD learning with different parameterizations may cause divergence [17, 200]: in general, the representation must be tied to the stochastic update [85] to ensure both the parameterization and the stochastic process are stable.

Contributions Our main result is a memory-efficient, non-parametric, stochastic method that converges to the Bellman fixed point almost surely when it belongs to a reproducing kernel Hilbert space (RKHS). Our approach is to reformulate (8.2) as a compositional stochastic program (Section 8.2), a topic studied in operations research [179] and probability [91, 100]. These problems motivate stochastic *quasi-gradient* (SQG) methods which use two time-scale stochastic approximation to mitigate the fact that the objective's stochastic gradient is biased with respect to its average [61]. Here, we use SQG for policy evaluation in infinite MDPs (finite MDPs addressed in [24, 186]).

In (8.2), the decision variable is a continuous function, which we address by hypothesizing the Bellman fixed point belongs to a RKHS [88, 181]. However, a function in a RKHS has comparable complexity to the number of training samples processed, which could be infinite (an issue ignored in many kernel methods for MDPs [48, 64, 70, 146, 153, 190, 214]). We will tackle this memory bottleneck by requiring memory efficiency in both the function sample path and in its limit.

To find a memory-efficient sample path in the function space, we generalize SQG to RKHSs (Section 8.3), and combine this generalization with greedily-constructed sparse subspace projections (Section 8.3.1). These subspaces are constructed via matching pursuit [108, 148], a procedure motivated by the facts that (a) kernel matrices induced by arbitrary data streams likely violate requirements for convex-relaxation-based sparsity [40], and (b) parsimony is more important than exact recovery since SQG iterates are not the target signal but rather a point along the convergence path to Bellman fixed point. Rather than unsupervised forgetting [60], we tie the projection-induced error to stochastic descent [98] which keeps only those dictionary points needed for convergence (Sec. 8.4).

As a result, we conduct functional SQG descent via sparse projections of the SQG. This maintains a moderate-complexity sample path exactly towards V^*, which may be made arbitrarily close to the Bellman fixed point by decreasing the regularizer. By generalizing the relationship between SQG and supermartingales in [206] to Hilbert spaces, we establish that the sparse projected SQG sequence converges almost surely to the Bellman fixed point

with decreasing learning rates, and converges in mean while maintaining finite complexity when constant learning rates are used (Section 8.4). This is the first almost sure convergence result for policy evaluation in infinite MDPs with *nonlinear value function parameterizations* of moderate complexity, which is in the worst-case comparable to the data domain covering number [149, 226]. This work has been submitted as [99].

8.2 Policy Evaluation as Compositional Stochastic Programming

We turn to reformulating the functional fixed point problem (8.3) defined by Bellman's equation so that it may be identified with a nested stochastic program. We note that the resulting domain of this problem is intractable, and address this by hypothesizing that the Bellman fixed point belongs to a RKHS, which, in turn, requires the introduction of regularization.

We proceed with reformulating (8.3): subtract the value function $V^\pi(\mathbf{x})$ that satisfies the fixed point relation from both sides, and then pull it inside the expectation:

$$0 = \mathbb{E}_{\mathbf{y}}[r(\mathbf{x}, \pi(\mathbf{x}), \mathbf{y}) + \gamma V^\pi(\mathbf{y}) - V^\pi(\mathbf{x}) \,|\, \mathbf{x}, \pi(\mathbf{x})] \quad \text{for all } \mathbf{x} \in \mathcal{X} \,. \tag{8.4}$$

Value functions satisfying (8.4) are equivalent to those which satisfy the quadratic expression $0 = \frac{1}{2}(\mathbb{E}_{\mathbf{y}}[r(\mathbf{x}, \pi(\mathbf{x}), \mathbf{y}) + \gamma V^\pi(\mathbf{y}) - V^\pi(\mathbf{x}) \,|\, \mathbf{x}, \pi(\mathbf{x})])^2$, which is null for all $\mathbf{x} \in \mathcal{X}$. Solving this expression for every $\mathbf{x}$ may be achieved by considering this expression in an initialization-independent manner. That is, integrating out $\mathbf{x}$, the starting point of the trajectory defining the value function (8.1), as well as policy $\pi(\mathbf{x})$, yields the compositional stochastic program:

$$V^\pi = \operatorname*{argmin}_{V \in \mathcal{B}(\mathcal{X})} J(V) := \operatorname*{argmin}_{V \in \mathcal{B}(\mathcal{X})} \mathbb{E}_{\mathbf{x}, \pi(\mathbf{x})} \Big\{ \frac{1}{2} (\mathbb{E}_{\mathbf{y}}[r(\mathbf{x}, \pi(\mathbf{x}), \mathbf{y}) + \gamma V(\mathbf{y}) - V(\mathbf{x}) \,|\, \mathbf{x}, \pi(\mathbf{x})])^2 \Big\} \,, \tag{8.5}$$

whose solutions coincide exactly with the fixed points of (8.3).

(8.5) defines a functional optimization problem which is intractable when we search over all bounded continuous functions $\mathcal{B}(\mathcal{X})$. However, when we restrict $\mathcal{B}(\mathcal{X})$ to a Hilbert space $\mathcal{H}$ equipped with a unique *reproducing kernel*, i.e., an inner product-like map $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that

$$(i)\ \langle f, \kappa(\mathbf{x}, \cdot) \rangle_{\mathcal{H}} = f(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathcal{X} \,, \qquad (ii)\ \mathcal{H} = \overline{\operatorname{span}\{\kappa(\mathbf{x}, \cdot)\}} \quad \text{for all } \mathbf{x} \in \mathcal{X} \,, \tag{8.6}$$

we may apply the Representer Theorem to transform the functional problem (8.5) into a parametric one [88, 144, 169] In a RKHS, the optimal function $f \in \mathcal{H}$ of (8.5) then takes the

form

$$f(\mathbf{x}) = \sum_{n=1}^{N} w_n \kappa(\mathbf{x}_n, \mathbf{x}) \ , \tag{8.7}$$

where $\mathbf{x}_n$ is a realization of the random variable $\mathbf{x}$. Thus, $f \in \mathcal{H}$ is an expansion of kernel evaluations *only* at training samples. We refer to the upper summand index N in (8.7) in the kernel expansion of $f \in \mathcal{H}$ as the model order, which here coincides with the training sample size. In (8.6), property (i) is called the reproducing property, which follows from Riesz Representation Theorem [212]. Replacing f by $\kappa(\mathbf{x}', \cdot)$ in (8.6) (i) yields the expression $\langle \kappa(\mathbf{x}', \cdot), \kappa(\mathbf{x}, \cdot) \rangle_{\mathcal{H}} = \kappa(\mathbf{x}, \mathbf{x}')$, the origin of the term "reproducing kernel." Moreover, property (8.6) (ii) states that functions $f \in \mathcal{H}$ admit a basis expansion in terms of kernel evaluations (8.7). Function spaces of this type are referred to as reproducing kernel Hilbert spaces (RKHSs). For universal kernels the kernel is universal [126], e.g., a Gaussian, a continuous function over a compact set may be approximated uniformly by one in a RKHS.

Subsequently, we seek to solve (8.5) with the restriction that $V \in \mathcal{H}$, and independent and identically distributed samples $(\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)$ from the triple $(\mathbf{x}, \pi(\mathbf{x}), \mathbf{y})$ are sequentially available, yielding

$$V^* = \underset{V \in \mathcal{H}}{\operatorname{argmin}} \, \mathbb{E}_{\mathbf{x}, \pi(\mathbf{x})} \big\{ \frac{1}{2} (\mathbb{E}_{\mathbf{y}}[r(\mathbf{x}, \pi(\mathbf{x}), \mathbf{y}) + \gamma V(\mathbf{y}) - V(\mathbf{x}) \,\big|\, \mathbf{x}, \pi(\mathbf{x})])^2 \big\} + \frac{\lambda}{2} \|V\|_{\mathcal{H}}^2 \tag{8.8}$$

Hereafter, define $L(V) := \mathbb{E}_{\mathbf{x}, \pi(\mathbf{x})} \{ \frac{1}{2} (\mathbb{E}_{\mathbf{y}}[r(\mathbf{x}, \pi(\mathbf{x}), \mathbf{y}) + \gamma V(\mathbf{y}) - V(\mathbf{x}) \,\big|\, \mathbf{x}, \pi(\mathbf{x})])^2 \}$ and $J(V) = L(V) + (\lambda/2)\|V\|_{\mathcal{H}}^2$. The regularization term $(\lambda/2)\|V\|_{\mathcal{H}}^2$ in (8.8) is needed to apply the Representer Theorem (8.7) [169]. Thus, policy evaluation in infinite MDPs (8.8) is both a specialization of compositional stochastic programming [206] to an objective defined by dynamic programming, and a generalization to the case where the decision variable is not vector-valued but is instead a function.

8.3 Functional Stochastic Quasi-Gradient Method

To apply functional SQG to (8.8), we differentiate the compositional objective $L(V)$, which is of the form $L = g \circ h$, with $g(u) = \mathbb{E}_{\mathbf{x}, \pi(\mathbf{x})}[(1/2)u^2]$ and $h(V) = \mathbb{E}_{\mathbf{y}}[r(\mathbf{x}, \pi(\mathbf{x}), \mathbf{y}) + \gamma V(\mathbf{y}) - V(\mathbf{x}) \,\big|\, \mathbf{x}, \pi(\mathbf{x})]$, and then consider its stochastic estimate. Consider the Frechét derivative of $L(V)$:

$$\begin{aligned} \nabla_V L(V) &= \mathbb{E}_{\mathbf{x}, \pi(\mathbf{x})} \big\{ \nabla_V \frac{1}{2} (\mathbb{E}_{\mathbf{y}} \left[r(\mathbf{x}, \pi(\mathbf{x}), \mathbf{y}) + \gamma V(\mathbf{y}) - V(\mathbf{x}) \,\big|\, \mathbf{x}, \pi(\mathbf{x}) \right])^2 \big\} \\ &= \mathbb{E}_{\mathbf{x}, \pi(\mathbf{x})} \big\{ \mathbb{E}_{\mathbf{y}} \left[\gamma \kappa(\mathbf{y}, \cdot) - \kappa(\mathbf{x}, \cdot) \,\big|\, \mathbf{x}, \pi(\mathbf{x}) \right] \mathbb{E}_{\mathbf{y}} \left[r(\mathbf{x}, \pi(\mathbf{x}), \mathbf{y}) + \gamma V(\mathbf{y}) - V(\mathbf{x}) \,\big|\, \mathbf{x}, \pi(\mathbf{x}) \right] \big\} \end{aligned} \tag{8.9}$$

On the first line, we pull the differential operator inside the expectation, and on the second line we make use of the chain rule and reproducing property of the kernel (8.6)(i). Now, we can use the law of total expectation to simplify the last line of (8.9) to

$$\nabla_V J(V) = \mathbb{E}_{\mathbf{x},\pi(\mathbf{x}),\mathbf{y}} \left[\gamma\kappa(\mathbf{y},\cdot) - \kappa(\mathbf{x},\cdot)\right] \mathbb{E}_{\mathbf{x},\pi(\mathbf{x}),\mathbf{y}} \left[r(\mathbf{x},\pi(\mathbf{x}),\mathbf{y}) + \gamma V(\mathbf{y}) - V(\mathbf{x})\right] \tag{8.10}$$

For future reference, we define the expression $\mathbb{E}_{\mathbf{y}}[r(\mathbf{x},\pi(\mathbf{x}),\mathbf{y}) + \gamma V(\mathbf{y}) - V(\mathbf{x}) \,\big|\, \mathbf{x},\pi(\mathbf{x})] = \bar{\delta}$ as the average temporal difference [184]. To perform stochastic descent in function space $\mathcal{H}$, we need a stochastic approximate of (8.9) evaluated at a state-action-state triple $(\mathbf{x},\pi(\mathbf{x}),\mathbf{y})$, which together with the regularizer yields

$$\nabla_V J(V,\delta;\mathbf{x},\pi(\mathbf{x}),\mathbf{y}) = \left[\gamma\kappa(\mathbf{y},\cdot) - \kappa(\mathbf{x},\cdot)\right] \left[r(\mathbf{x},\pi(\mathbf{x}),\mathbf{y}) + \gamma V(\mathbf{y}) - V(\mathbf{x})\right] + \lambda V \tag{8.11}$$

where $\delta := r(\mathbf{x},\pi(\mathbf{x}),\mathbf{y}) + \gamma V(\mathbf{y}) - V(\mathbf{x})$ is defined as the (instantaneous) temporal difference. Observe that we cannot obtain unbiased samples of $\nabla_V J(V,\delta;\mathbf{x},\pi(\mathbf{x}),\mathbf{y})$ due to the fact that the terms inside the inner expectations in (8.9) are *dependent*, a problem first identified in [186] for finite MDPs. Therefore, we require a method that constructs a *coupled* stochastic descent procedure by considering noisy estimates of both terms in the product-of-expectations expression in (8.9).

Due to the fact that the first term $[\gamma\kappa(\mathbf{y},\cdot) - \kappa(\mathbf{x},\cdot)]$ in (8.11) is a difference of kernel maps, building up its total expectation will, in the limit, be of infinite complexity [89]. Thus, we propose instead to construct a sequence based on samples of the second term. That is, based on realizations of δ, we consider a fixed point recursion that builds up an estimate of $\bar{\delta}$ by defining a scalar sequence z_t as

$$\delta_t = r(\mathbf{x}_t,\pi(\mathbf{x}_t),\mathbf{y}_t) + \gamma V_t(\mathbf{y}_t) - V_t(\mathbf{x}_t)\ , \quad z_{t+1} = (1-\beta_t)z_t + \beta_t\delta_t \tag{8.12}$$

where we define δ_t [184] as the temporal difference at time t in (8.12) Thus, (8.12) approximately averages the temporal difference sequence δ_t: z_t estimates $\bar{\delta}_t$, and $\beta_t \in (0,1)$ is a learning rate.

To define a stochastic descent step, we replace the first term inside the outer expectation in (8.9) with its instantaneous approximate, i.e., $[\gamma\kappa(\mathbf{y}_t,\cdot) - \kappa(\mathbf{x}_t,\cdot)]$, evaluated at a sample triple $(\mathbf{x}_t,\pi(\mathbf{x}_t),\mathbf{y}_t)$, which yields the stochastic quasi-gradient step [61, 206]

$$\hat{V}_{t+1} = (1-\alpha_t\lambda)\hat{V}_t - \alpha_t(\gamma\kappa(\mathbf{y}_t,\cdot) - \kappa(\mathbf{x}_t,\cdot))z_{t+1}\ . \tag{8.13}$$

where the coefficient $(1-\alpha_t\lambda)$ comes from the regularizer, and α_t is a positive scalar learning rate. This update is a stochastic quasi-gradient step because the true stochastic gradient of $J(V)$ is $(\gamma\kappa(\mathbf{y}_t,\cdot) - \kappa(\mathbf{x}_t,\cdot))\delta_t$, but this estimator is biased with respect to its average

$\nabla_V J(V)$ since the terms in this product are correlated. By replacing δ_t by auxiliary variable z_{t+1} this issue may be circumvented in the construction of coupled supermartingales (Section 8.4).

Kernel Parameterization Suppose $V_0 = 0 \in \mathcal{H}$. Then the update in (8.13) at time t, making use of the Representer Theorem (8.7), implies the function $\tilde{V}_t$ is a kernel expansion of past states $(\mathbf{x}_t, \mathbf{y}_t)$ as

$$\hat{V}_t(\mathbf{x}) = \sum_{n=1}^{2(t-1)} w_n \kappa(\mathbf{v}_n, \mathbf{x}) = \mathbf{w}_t^T \boldsymbol{\kappa}_{\mathbf{X}_t}(\mathbf{x}) \; . \tag{8.14}$$

On the right-hand side of (8.14) we introduce the notation $\mathbf{v}_n = \mathbf{x}_n$ for n even and $\mathbf{v}_n = \mathbf{y}_n$ for n odd, and: $\mathbf{w}_t = [w_1, \cdots, w_{2(t-1)}] \in \mathbb{R}^{2(t-1)}$, $\mathbf{X}_t = [\mathbf{x}_1, \mathbf{y}_1, \ldots, \mathbf{x}_{t-1}, \mathbf{y}_{t-1}] \in \mathbb{R}^{p \times 2(t-1)}$, and $\boldsymbol{\kappa}_{\mathbf{X}_t}(\cdot) = [\kappa(\mathbf{x}_1, \cdot), \; \kappa(\mathbf{y}_1, \cdot), \; \ldots \; , \kappa(\mathbf{x}_{t-1}, \cdot), \kappa(\mathbf{y}_{t-1}, \cdot)]^T$. The kernel expansion in (8.14), together with the functional update (8.13), yields the fact that functional SQG in $\mathcal{H}$ amounts to the following updates on the kernel dictionary $\mathbf{X}$ and coefficient vector $\mathbf{w}$:

$$\mathbf{X}_{t+1} = [\mathbf{X}_t, \;\; \mathbf{x}_t, \mathbf{y}_t], \quad \mathbf{w}_{t+1} = [(1 - \alpha_t \lambda)\mathbf{w}_t, \quad \alpha_t z_{t+1}, -\alpha_t \gamma z_{t+1}] \; , \tag{8.15}$$

Observe that this update causes $\mathbf{X}_{t+1}$ to have two more columns than $\mathbf{X}_t$. We define the *model order* as number of data points M_t in the dictionary at time t, which for functional stochastic quasi-gradient descent is $M_t = 2(t-1)$. Asymptotically, then, the complexity of storing $\hat{V}_t(\mathbf{x})$ is infinite.

8.3.1 Sparse Projection of Stochastic Quasi-Gradient Method

Since the update (8.13) has complexity $\mathcal{O}(t)$ due to the parameterization induced by RKHS [89, 98], it is impractical in settings with streaming data or arbitrarily large training sets. We address this issue by replacing the stochastic descent step (8.13) with an orthogonally projected variant [98], where the projection is onto a low-dimensional functional subspace $\mathcal{H}_{\mathbf{D}_{t+1}}$ of $\mathcal{H}$, i.e.,

$$V_{t+1} = \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}}[(1 - \alpha_t \lambda)V_t - \alpha_t(\gamma \kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot))z_{t+1}] \; , \tag{8.16}$$

where α_t again is a scalar step-size, and $\mathcal{H}_{\mathbf{D}_{t+1}} = \text{span}\{\kappa(\mathbf{d}_n, \cdot)\}_{n=1}^{M_t}$ for some collection of sample instances $\{\mathbf{d}_n\} \subset \{\mathbf{x}_u\}_{u \leq t}$. The interpretation of the un-projected function SQG method (8.13) (Section 8.3) in terms of subspace projections is comparable to the derivation of Algorithm 6 in Chapter 6, motivating (8.16).

We proceed to describe the construction of these subspace projections. Consider sub-

spaces $\mathcal{H}_{\mathbf{D}} \subseteq \mathcal{H}$ that consist of functions that can be represented using some dictionary $\mathbf{D} = [\mathbf{d}_1, \ \ldots, \ \mathbf{d}_M] \in \mathbb{R}^{p \times M}$, i.e., $\mathcal{H}_{\mathbf{D}} = \text{span}\{\kappa(\mathbf{d}_n, \cdot)\}_{n=1}^{M}$. For convenience, we define $\boldsymbol{\kappa}_{\mathbf{D}}(\cdot) = [\kappa(\mathbf{d}_1, \cdot) \ldots \kappa(\mathbf{d}_M, \cdot)]$, and $\mathbf{K}_{\mathbf{D},\mathbf{D}}$ as the resulting kernel matrix from this dictionary. We enforce function parsimony by selecting dictionaries $\mathbf{D}$ that $M_t << \mathcal{O}(t)$.

Observe that by selecting $\mathbf{D} = \mathbf{X}_{t+1}$ at each step, the sequence (8.13) may be interpreted as a sequence of orthogonal projections. To see this, rewrite (8.13) as the quadratic minimization

$$\begin{aligned}\hat{V}_{t+1} &= \underset{V \in \mathcal{H}}{\text{argmin}} \left\| V - \Big((1 - \alpha_t \lambda) \hat{V}_t - \alpha_t (\gamma \kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot)) z_{t+1} \Big) \right\|_{\mathcal{H}}^2 \\ &= \underset{V \in \mathcal{H}_{\mathbf{X}_{t+1}}}{\text{argmin}} \left\| V - \Big((1 - \alpha_t \lambda) \hat{V}_t - \alpha_t (\gamma \kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot)) z_{t+1} \Big) \right\|_{\mathcal{H}}^2, \end{aligned} \tag{8.17}$$

where the first equality in (8.17) comes from ignoring constant terms which vanish upon differentiation with respect to V, and the second comes from observing that V_{t+1} can be represented using only the points $\mathbf{X}_{t+1}$, using (8.15). Notice now that (8.17) expresses V_{t+1} as the orthogonal projection of the update $(1 - \alpha_t \lambda) V_t - \alpha_t (\gamma \kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot)) z_{t+1}$ onto the subspace defined by dictionary $\mathbf{X}_{t+1}$.

Rather than select dictionary $\mathbf{D} = \mathbf{X}_{t+1}$, we propose instead to select a different dictionary, $\mathbf{D} = \mathbf{D}_{t+1}$, which is extracted from the data points observed thus far, at each iteration. The process by which we select $\mathbf{D}_{t+1}$ is using matching pursuit (Algorithm 5), and is of dimension $p \times M_{t+1}$, with $M_{t+1} << \mathcal{O}(t)$. As a result, the sequence V_t differs from the functional stochastic quasi-gradient method $\hat{V}_t$ presented in Section 8.3.

The function V_{t+1} is parameterized dictionary $\mathbf{D}_{t+1}$ and weight vector $\mathbf{w}_{t+1}$. We denote columns of $\mathbf{D}_{t+1}$ as $\mathbf{d}_n$ for $n = 1, \ldots, M_{t+1}$, where the time index is dropped for notational clarity but may be inferred from the context. We replace the update (8.17) in which the dictionary grows at each iteration by the functional stochastic quasi-gradient sequence projected onto the subspace $\mathcal{H}_{\mathbf{D}_{t+1}} = \text{span}\{\kappa(\mathbf{d}_n, \cdot)\}_{n=1}^{M_{t+1}}$ as

$$\begin{aligned} V_{t+1} &= \underset{V \in \mathcal{H}_{\mathbf{D}_{t+1}}}{\text{argmin}} \left\| V - \Big((1 - \alpha_t \lambda) V_t - \alpha_t (\gamma \kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot)) z_{t+1} \Big) \right\|_{\mathcal{H}}^2 \\ &:= \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \Big[(1 - \alpha_t \lambda) V_t - \alpha_t (\gamma \kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot)) z_{t+1} \Big]. \end{aligned} \tag{8.18}$$

where we define the projection operator $\mathcal{P}$ onto subspace $\mathcal{H}_{\mathbf{D}_{t+1}} \subset \mathcal{H}$ by the update (8.18). This orthogonal projection is the modification of the functional SQG iterate [cf. (8.13)] defined at the beginning of this subsection (8.16). Next we discuss how this update amounts to modifications of the parametric updates (8.15) defined by functional SQG.

Coefficient update The update (8.16), for a fixed dictionary $\mathbf{D}_{t+1} \in \mathbb{R}^{p \times M_{t+1}}$, may be expressed in terms of the parameter space of coefficients only. To do so, first define

the stochastic quasi-gradient update *without projection*, given function V_t parameterized by dictionary $\mathbf{D}_t$ and coefficients $\mathbf{w}_t$, as

$$\tilde{V}_{t+1} = (1 - \alpha_t\lambda)V_t - \alpha_t(\gamma\kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot))z_{t+1} \ . \tag{8.19}$$

This update may be represented using dictionary and weight vector

$$\tilde{\mathbf{D}}_{t+1} = [\mathbf{D}_t, \ \ \mathbf{x}_t, \mathbf{y}_t], \quad \tilde{\mathbf{w}}_{t+1} = [(1 - \alpha_t\lambda)\mathbf{w}_t, \ \ \alpha_t z_{t+1} \ , -\alpha_t\gamma z_{t+1}] \ , \tag{8.20}$$

Observe that $\tilde{\mathbf{D}}_{t+1}$ has $\tilde{M}_{t+1} = M_t + 2$ columns, which is the length of $\tilde{\mathbf{w}}_{t+1}$. For a fixed dictionary $\mathbf{D}_{t+1}$, the stochastic projection in (8.18) is a least-squares problem on the coefficient vector, i.e.,

$$\mathbf{w}_{t+1} = \mathbf{K}^{-1}_{\mathbf{D}_{t+1}\mathbf{D}_{t+1}}\mathbf{K}_{\mathbf{D}_{t+1}\tilde{\mathbf{D}}_{t+1}}\tilde{\mathbf{w}}_{t+1} \ , \tag{8.21}$$

where we define the cross-kernel matrix $\mathbf{K}_{\mathbf{D}_{t+1},\tilde{\mathbf{D}}_{t+1}}$ whose $(n,m)^{\text{th}}$ entry is $\kappa(\mathbf{d}_n, \tilde{\mathbf{d}}_m)$. Kernel matrices $\mathbf{K}_{\tilde{\mathbf{D}}_{t+1},\tilde{\mathbf{D}}_{t+1}}$ and $\mathbf{K}_{\mathbf{D}_{t+1},\mathbf{D}_{t+1}}$ are similarly defined. Here M_{t+1} is the number of columns in $\mathbf{D}_{t+1}$, while $\tilde{M}_{t+1} = M_t + 2$ is that of in $\tilde{\mathbf{D}}_{t+1}$ [cf. (8.20)]. The derivation of this identity is as follows.

We use the notation that V_{t+1} is the sequence of projected quasi-FGSD iterates [cf. (8.16)] and $\tilde{V}_{t+1}$ is the update [cf. (8.19)] without projection in Section 8.3.1. The later is parameterized by dictionary $\tilde{\mathbf{D}}_{t+1}$ and weights $\tilde{\mathbf{w}}_{t+1}$ (8.20). When the dictionary defining V_{t+1} is assumed fixed, we may use use of the Representer Theorem to rewrite (8.18) in terms of kernel expansions, and note that the coefficient vector is the only free parameter to write

$$\begin{aligned}
&\operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{M_{t+1}}} \frac{1}{2\eta_t}\Big\| \sum_{n=1}^{M_{t+1}} w_n\kappa(\mathbf{d}_n, \cdot) - \sum_{m=1}^{\tilde{M}} \tilde{w}_m\kappa(\tilde{\mathbf{d}}_m, \cdot)\Big\|^2_{\mathcal{H}} \qquad (8.22)\\
&= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{M_{t+1}}} \frac{1}{2\eta_t}\left(\sum_{n,n'=1}^{M_{t+1}} w_n w_{n'}\kappa(\mathbf{d}_n, \mathbf{d}_{n'}) - 2\sum_{n,m=1}^{M_{t+1},\tilde{M}} w_n\tilde{w}_m\kappa(\mathbf{d}_n, \tilde{\mathbf{d}}_m) + \sum_{m,m'=1}^{\tilde{M}} \tilde{w}_m\tilde{w}_{m'}\kappa(\tilde{\mathbf{d}}_m, \tilde{\mathbf{d}}_{m'})\right)\\
&= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{M_{t+1}}} \frac{1}{2\eta_t}\Big(\mathbf{w}^T\mathbf{K}_{\mathbf{D}_{t+1},\mathbf{D}_{t+1}}\mathbf{w} - 2\mathbf{w}^T\mathbf{K}_{\mathbf{D}_{t+1},\tilde{\mathbf{D}}_{t+1}}\tilde{\mathbf{w}}_{t+1} + \tilde{\mathbf{w}}_{t+1}\mathbf{K}_{\tilde{\mathbf{D}}_{t+1},\tilde{\mathbf{D}}_{t+1}}\tilde{\mathbf{w}}_{t+1}\Big) := \mathbf{w}_{t+1} \ .
\end{aligned}$$

In (8.22), the first equality comes from expanding the square, and the second comes from defining The explicit solution of (8.22) may be obtained by noting that the last term is a constant independent of $\mathbf{w}$, and thus by computing gradients and solving for $\mathbf{w}_{t+1}$ we obtain (8.21).

We now turn to selecting the dictionary $\mathbf{D}_{t+1}$ from the MDP trajectory $\{\mathbf{x}_u, \pi(\mathbf{x}_u), \mathbf{y}_u\}_{u\le t}$.

Dictionary Update We select kernel dictionary $\mathbf{D}_{t+1}$ via greedy compression, a topic studied in compressive sensing [137]. The function $\tilde{V}_{t+1} = (1 - \alpha_t)V_t - \alpha_t(\gamma\kappa(\mathbf{y}_t, \cdot) -$

Algorithm 8 PKGTD: Parsimonious Kernel Gradient Temporal Difference

Require: $\{\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t, \alpha_t, \beta_t, \epsilon_t\}_{t=0,1,2,\ldots}$

initialize $V_0(\cdot) = 0, \mathbf{D}_0 = [], \mathbf{w}_0 = [], z_0 = 0$, i.e. initial dict., coeffs., and aux. variable null

for $t = 0, 1, 2, \ldots$ **do**

Obtain trajectory realization $(\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)$

Compute the temporal difference and update the auxiliary sequence z_{t+1} [cf. (8.12)]:

$$\delta_t = r(\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) + \gamma V_t(\mathbf{y}_t) - V_t(\mathbf{x}_t) \ , \quad z_{t+1} = (1 - \beta_t) z_t + \beta_t \delta_t$$

Compute unconstrained functional stochastic quasi-gradient step [cf. (8.13)]

$$\tilde{V}_{t+1}(\cdot) = (1 - \alpha_t \lambda) \tilde{V}_t(\cdot) - \alpha_t (\gamma \kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot)) z_{t+1}$$

Revise dictionary $\tilde{\mathbf{D}}_{t+1} = [\mathbf{D}_t, \mathbf{x}_t \ , \mathbf{y}_t]$,
and weights $\tilde{\mathbf{w}}_{t+1} \leftarrow [(1 - \alpha_t \lambda)\mathbf{w}_t, \ \ \alpha_t z_{t+1}, -\alpha_t \gamma z_{t+1}]$

Obtain greedy compression of function parameterization via Algorithm 5

$$(V_{t+1}, \mathbf{D}_{t+1}, \mathbf{w}_{t+1}) = \mathbf{KOMP}(\tilde{V}_{t+1}, \tilde{\mathbf{D}}_{t+1}, \tilde{\mathbf{w}}_{t+1}, \epsilon_t)$$

end for

$\kappa(\mathbf{x}_t, \cdot))z_{t+1}$ defined by SQG method without projection (8.19) is parameterized by dictionary $\tilde{\mathbf{D}}_{t+1}$ [cf. (8.20)]. We form $\mathbf{D}_{t+1}$ by selecting a subset of M_{t+1} columns from $\tilde{\mathbf{D}}_{t+1}$ that best approximate $\tilde{V}_{t+1}$ in terms of Hilbert norm error. To accomplish this, we use *kernel orthogonal matching pursuit* (KOMP, Algorithm 5) [203] with error tolerance ϵ_t to find a dictionary $\mathbf{D}_{t+1}$ based that which adds the latest samples $\tilde{\mathbf{D}}_{t+1}$. We tune ϵ_t to ensure both stochastic descent (Lemma 8(2)) and finite model order (Corollary 4).

With respect to the KOMP procedure above, we specifically use a variant called destructive KOMP with pre-fitting (see [203], Section 2.3). This flavor of KOMP takes as an input a candidate function $\tilde{V}$ of model order $\tilde{M}$ parameterized by its dictionary $\tilde{\mathbf{D}} \in \mathbb{R}^{p \times \tilde{M}}$ and coefficients $\tilde{\mathbf{w}} \in \mathbb{R}^{\tilde{M}}$. The method then approximates $\tilde{V}$ by $V \in \mathcal{H}$ with a lower model order. Initially, the candidate is the original $V = \tilde{V}$ so that its dictionary is initialized with $\mathbf{D} = \tilde{\mathbf{D}}$, with coefficients $\mathbf{w} = \tilde{\mathbf{w}}$. Then, we sequentially and greedily remove model points from initial dictionary $\tilde{\mathbf{D}}$ until threshold $\|V - \tilde{V}\|_{\mathcal{H}} \leq \epsilon_t$ is violated. The result is a sparse approximation V of $\tilde{V}$.

We summarize the proposed method, Parsimonious Kernel Gradient Temporal Difference (PKGTD) in Algorithm 8: we execute the stochastic projection of the functional SQG iterates onto sparse subspaces $\mathcal{H}_{\mathbf{D}_{t+1}}$ stated in (8.18). With initial function null $V_0 = 0$ (empty dictionary $\mathbf{D}_0 = []$ and coefficients $\mathbf{w}_0 = []$),at each step, given an i.i.d. sample $(\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)$ and step-sizes α_t, β_t, we compute the *unconstrained* functional SQG iterate $\tilde{V}_{t+1}(\cdot) = (1 - \alpha_t \lambda)\tilde{V}_t(\cdot) - \alpha_t(\gamma \kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot))z_{t+1}$ parameterized by $\tilde{\mathbf{D}}_{t+1}$ and

$\tilde{\mathbf{w}}_{t+1}$ as stated in (8.20), which are fed into KOMP (Algorithm 5) with budget ϵ_t, i.e., $(V_{t+1}, \mathbf{D}_{t+1}, \mathbf{w}_{t+1}) = \text{KOMP}(\tilde{V}_{t+1}, \tilde{\mathbf{D}}_{t+1}, \tilde{\mathbf{w}}_{t+1}, \epsilon_t)$.

8.4 Convergence Analysis via Coupled Supermartingales

We now analyze the stability and memory requirements of Algorithm 8 developed in Section 8.3. Our approach is fundamentally different from stochastic fixed point methods such as TD learning, which are not descent techniques, and thus exhibit delicate convergence. The interplay between the Bellman operator contraction [23] and expectations prevents the construction of supermartingales underlying stochastic descent stability [161]. Attempts to mitigate this issue, such as those based on stochastic backward-differences [87] ([75, 199]) or Lyapunov approaches [30], e.g., [186], require the state space to be completely explored in the limit *per step* (intractable when $|\mathcal{X}| = \infty$), or stipulate that data dependent matrices be non-singular, respectively. Thus, there is a long-standing question of how to perform policy evaluation in MDPs under conditions applicable to practitioners while also guaranteeing stability. We provide an answer by connecting RKHS-valued stochastic quasi-gradient methods (Algorithm 8) with coupled supermartingale theory [205].

Iterate Convergence Under the technical conditions stated below, it is possible to derive the fact that the auxiliary variable z_t and value function estimate V_t satisfy supermartingale-type relationships, but their behavior is intrinsically coupled to one another. We generalize recently developed coupled supermartingale tools in [205], i.e., Lemma 9 to establish almost sure convergence when the step-sizes and compression budget are diminishing. To do so, some technical assumptions and auxiliary results are needed which we state next.

For further reference, we define the functional stochastic quasi-gradient of the regularized objective as

$$\hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) = (\gamma\kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot))z_{t+1} + \lambda V_t \ , \tag{8.23}$$

and its sparse-subspace projected variant as

$$\tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) = \Big(V_t - \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}}\Big[V_t - \alpha_t \hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)\Big]\Big)/\alpha_t \ , \tag{8.24}$$

Note that the update (8.16), using (8.24), may be rewritten as a stochastic projected quasi-gradient step rather than a stochastic quasi-gradient step followed by set projection, i.e.,

$$V_{t+1} = V_t - \alpha_t \tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) \ , \tag{8.25}$$

AS22 *The state space $\mathcal{X} \subset \mathbb{R}^p$ and action space $\mathcal{A} \subset \mathbb{R}^q$ are compact, and the reproducing*

kernel map may be bounded as

$$\sup_{\mathbf{x}\in\mathcal{X}} \sqrt{\kappa(\mathbf{x},\mathbf{x})} = X < \infty \tag{8.26}$$

Furthermore, the subspaces $\mathcal{H}_{\mathbf{D}_t}$ output by Algorithm 5 are intersected with some finite Hilbert-norm ball: $\|f\|_{\mathcal{H}} \leq K$ for all t.

AS23 *The temporal difference δ and auxiliary sequence z [cf. (8.12)] satisfy the zero-mean, finite conditional variance, and Lipschitz continuity conditions, respectively,*

$$\mathbb{E}\left[\delta \,\middle|\, \mathbf{x}, \pi(\mathbf{x})\right] = \bar{\delta}\ , \qquad \mathbb{E}\left[(\delta-\bar{\delta})^2\right] \leq \sigma_\delta^2\ , \qquad \mathbb{E}\left[z^2 \,\middle|\, \mathbf{x}, \pi(\mathbf{x})\right] \leq G_\delta^2\ . \tag{8.27}$$

where σ_δ and G_δ are positive scalars.

AS24 *The functional gradient of the temporal difference is an unbiased estimate for $\nabla_V J(V)$ [cf. (8.9)], and the difference of reproducing kernels expression (the first term in the product expression (8.11)) has finite conditional variance:*

$$\mathbb{E}\left[(\gamma\kappa(\mathbf{y},\cdot)-\kappa(\mathbf{x},\cdot))\delta\right] = \nabla_V J(V)\ , \qquad \mathbb{E}\left[\|\gamma\kappa(\mathbf{y}_t,\cdot)-\kappa(\mathbf{x}_t,\cdot)\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq G_V^2\ . \tag{8.28}$$

Moreover, the projected stochastic quasi-gradient of the objective [cf. (8.24)] has finite second conditional moment as

$$\mathbb{E}\left[\|\tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq \sigma_V^2\ , \tag{8.29}$$

and the temporal difference is Lipschitz continuous with respect to the value function V, i.e for any two distinct δ and $\tilde{\delta}$, we have

$$|\delta - \tilde{\delta}| \leq L_V \|V - \tilde{V}\|_{\mathcal{H}} \tag{8.30}$$

where $V, \tilde{V} \in \mathcal{H}$ are distinct value functions in the RKHS, and $L_V > 0$ is a scalar.

Assumption 22 regarding the compactness of the state and action spaces of the Markov Decision Process intrinsically hold for most application settings and limit the radius of the set from which the MDP trajectory is sampled. Similar boundedness conditions on the reproducing kernel map have been considered in supervised learning applications [89]. The mean and variance properties of the temporal difference stated in Assumption 23 are necessary to bound the error in the descent direction associated with stochastic approximations, and are necessary to establish stability properties of stochastic methods. Assumption 24 is similar to Assumption 23 but instead of establishing bounds on the stochastic approximation error of the temporal difference, limits stochastic error variance in the reproducing

kernel Hilbert space. These are natural extensions of the conditions needed for convergence of stochastic compositional gradient methods with vector-valued decision variables [206].

The stipulation that the subspaces $\mathcal{H}_{\mathbf{D}_t}$ of $\mathcal{H}$ have non-empty intersection with some finite Hilbert-norm ball (Assumption 22), mean V_t is contained within compact sets for all t due to the use of set projections in the value function update (8.16), which allows us to write

$$\|V_t\|_{\mathcal{H}} \leq K \ , \qquad \|V^*\|_{\mathcal{H}} \leq K \ , \quad \text{for all } t \tag{8.31}$$

where $K > 0$ is some constant. The boundedness of V^* follows from the fact that since $\mathcal{X}$ is compact and $J(V)$ is a continuous convex function over a compact set, its minimizer is achieved over this compact set [36][Corrolary 3.23].

Next we turn to establishing some technical results which are necessary precursor to the proofs of the main stability results.

Proposition 7 *Given independent identical realizations* $(\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)$ *of the random triple* $(\mathbf{x}, \pi(\mathbf{x}), \mathbf{y})$*, the difference between the projected stochastic functional quasi-gradient and the stochastic functional quasi-gradient of the instantaneous cost instantaneous risk defined by* (8.23) *and* (8.24)*, respectively, is bounded for all* t *as*

$$\|\tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) - \hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}} \leq \frac{\epsilon_t}{\alpha_t} \tag{8.32}$$

where $\alpha_t > 0$ *denotes the algorithm step-size and* $\epsilon_t > 0$ *is the compression budget parameter of Algorithm 5.*

Proof: As in Proposition 6 of [98], consider the square-Hilbert-norm difference of $\tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)$ and $\hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)$ defined in (8.23) and (8.24), respectively,

$$\begin{aligned} &\|\tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) - \hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)\|^2_{\mathcal{H}} \qquad (8.33) \\ &= \Big\| \Big(V_t - \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \Big[V_t - \alpha_t \hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) \Big] \Big) / \alpha_t - \hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) \Big\|^2_{\mathcal{H}} \end{aligned}$$

Multiply and divide $\hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)$, the last term, by α_t, and reorder terms to write

$$\begin{aligned} &\Big\| \frac{\Big(V_t - \alpha_t \hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) \Big)}{\alpha_t} - \frac{\mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \Big[V_t - \alpha_t \hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) \Big] \Big)}{\alpha_t} \Big\|^2_{\mathcal{H}} \\ &= \frac{1}{\alpha_t^2} \Big\| (V_t - \alpha_t \hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) - \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \Big[V_t - \alpha_t \hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) \Big] \Big) \Big\|^2_{\mathcal{H}} \\ &= \frac{1}{\alpha_t^2} \|\tilde{V}_{t+1} - V_{t+1}\|^2_{\mathcal{H}} \leq \frac{\epsilon_t^2}{\alpha_t^2} \qquad (8.34) \end{aligned}$$

where we have pulled the nonnegative scalar α_t outside the norm on the second line and substituted the definition of $\tilde{V}_{t+1}$ and V_{t+1} in (8.13) and (8.16), respectively, in the last one. These facts combined with the KOMP residual stopping criterion in Algorithm 5 is $\|\tilde{V}_{t+1} - V_{t+1}\|_{\mathcal{H}} \leq \epsilon_t$ applied to the last term on the right-hand side of (8.34) yields (8.32). ■

Lemma 8 *Denote the filtration $\mathcal{F}_t$ as the time-dependent sigma-algebra containing the algorithm history $\mathcal{F}_t \supset (\{V_u, z_u\}_{u=0}^t \cup \{\mathbf{x}_s, \pi(\mathbf{x}_s), \mathbf{y}_s\}_{s=0}^{t-1})$ Let Assumptions 22 - 24 hold true and consider the sequence of iterates defined by Algorithm 8. Then:*

1. *The conditional expectation of the Hilbert-norm difference of value functions at the next and current iteration satisfies the relationship*

$$\mathbb{E}\left[\|V_{t+1} - V_t\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq 2\alpha_t^2(G_\delta^2 G_V^2 + \lambda^2 K^2) + 2\epsilon_t^2 \tag{8.35}$$

2. *The conditional expectation of the Hilbert-norm difference of value functions at the next and current iteration satisfies the relationship*

$$\begin{aligned}\mathbb{E}\left[\|V_{t+1} - V^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq \left(1 + \frac{\alpha_t^2}{\beta_t} G_V^2\right) \|V_t - V^*\|_{\mathcal{H}}^2 + 2\epsilon_t \|V_t - V^*\|_{\mathcal{H}} \\ - 2\alpha_t \left[J(V_t) - J(V^*)\right] + \alpha_t^2 \sigma_V^2 + \beta_t \mathbb{E}\left[(z_{t+1} - \bar{\delta}_t)^2 \,\middle|\, \mathcal{F}_t\right] .\end{aligned} \tag{8.36}$$

3. *Define the expected value of the temporal difference given the state variable $\mathbf{x}$ and policy π as $\bar{\delta}_t = \mathbb{E}[\delta_t \,|\, \mathbf{x}_t, \pi(\mathbf{x}_t)]$. Then the evolution of the auxiliary sequence z_t with respect to $\bar{\delta}_t$ satisfies*

$$\mathbb{E}\left[(z_{t+1} - \bar{\delta}_t)^2 \,\middle|\, \mathcal{F}_t\right] \leq (1 - \beta_t)(z_t - \bar{\delta}_{t-1})^2 + \frac{L_V}{\beta_t} \|V_t - V_{t-1}\|_{\mathcal{H}}^2 + 2\beta_t^2 \sigma_\delta^2 \tag{8.37}$$

Proof of Lemma 8(1): Consider the Hilbert-norm difference of value functions at the next and current iteration, and use the definition of V_{t+1} in (8.25), i.e.,

$$\begin{aligned}\|V_{t+1} - V_t\|_{\mathcal{H}}^2 &= \alpha_t^2 \|\tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \\ &\leq 2\alpha_t^2 \|\hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \\ &\quad + 2\alpha_t^2 \|\hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) - \tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 ,\end{aligned} \tag{8.38}$$

where we add and subtract the functional stochastic quasi-gradient $\hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)$ on the first line of (8.38) and apply the triangle inequality $(a+b)^2 \leq 2a^2 + 2b^2$ which holds

for any a, b. Now, we may apply Proposition 7 to the second term. Doing so and computing the expectation conditional on the filtration $\mathcal{F}_t$ yields

$$\mathbb{E}[\|V_{t+1} - V_t\|_{\mathcal{H}}^2 \,|\, \mathcal{F}_t] = 2\alpha_t^2 \mathbb{E}[\|\hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \,|\, \mathcal{F}_t] + 2\epsilon_t^2 \,. \tag{8.39}$$

Use the Cauchy-Schwartz inequality together with Law of Total Expectation and the definition of the functional stochastic quasi-gradient (8.23) to upper-estimate the right-hand side of (8.39) as

$$\mathbb{E}[\|V_{t+1} - V_t\|_{\mathcal{H}}^2 \,|\, \mathcal{F}_t] \leq 2\alpha_t^2 \mathbb{E}\Big\{\|\gamma\kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot))\|_{\mathcal{H}}^2 \mathbb{E}[z_{t+1}^2 \big|\, \mathbf{x}_t, \pi(\mathbf{x}_t)] \,\big|\, \mathcal{F}_t\Big\} + 2\alpha_t^2 \lambda \|V_t\|_{\mathcal{H}}^2 + 2\epsilon_t^2 \,, \tag{8.40}$$

which together with Assumption 8.27 regarding fact that z_{t+1} has a finite second conditional moment, yields

$$\begin{aligned} \mathbb{E}[\|V_{t+1} - V_t\|_{\mathcal{H}}^2 \,|\, \mathcal{F}_t &\leq 2\alpha_t^2 G_\delta^2 \mathbb{E}\Big[\|\gamma\kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot))\|_{\mathcal{H}}^2 \,|\, \mathcal{F}_t\Big] + 2\alpha_t^2 \lambda \|V_t\| + 2\epsilon_t^2 \\ &\leq 2\alpha_t^2 (G_\delta^2 G_V^2 + \lambda^2 K^2) + 2\epsilon_t^2 \,, \end{aligned} \tag{8.41}$$

where we have also applied the fact that the functional gradient of the temporal difference $\gamma\kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot))$ has a finite second conditional moment and the bound on the function sequence [cf. (8.31)], allowing us to conclude (8.35). ■

Proof of Lemma 8(2): This proof is a generalization of Lemma 3 in Appendix G.2 in the Supplementary Material of [206] to a function-valued stochastic quasi-gradient step combined with bias induced by the sparse subspace projections $\mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}}[\cdot]$ in (8.16). Begin by considering the square-Hilbert norm sub-optimality of V_{t+1}, i.e.,

$$\begin{aligned} \|V_{t+1} - V^*\|_{\mathcal{H}}^2 &= \|V_t - \alpha_t \tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) - V^*\|_{\mathcal{H}}^2 \\ &= \|V_t - V^*\|_{\mathcal{H}}^2 - 2\alpha_t \langle \tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t), V_t - V^* \rangle_{\mathcal{H}} \\ &\quad + \alpha_t^2 \|\tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \,, \end{aligned} \tag{8.42}$$

where we use the reformulation of the projected functional stochastic quasi-gradient step defined in (8.25) for the first equality, and expand the square in the second. Now, adding and subtracting $\hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)$ the (un-projected) functional stochastic quasi-

gradient (8.23) yields

$$\begin{aligned}\|V_{t+1} - V^*\|_{\mathcal{H}}^2 = \|V_t - V^*\|_{\mathcal{H}}^2 &- 2\alpha_t \langle \hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t), V_t - V^* \rangle_{\mathcal{H}} \\ &+ 2\alpha_t \langle \hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) - \tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t), V_t - V^* \rangle_{\mathcal{H}} \\ &+ \alpha_t^2 \|\tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \ . \end{aligned} \tag{8.43}$$

Apply the Cauchy-Schwartz inequality to the third term on the right-hand side of (8.43) together with the bound on the difference between unprojected and projected stochastic quasi-gradients in Proposition 7 to obtain

$$\begin{aligned}\|V_{t+1} - V^*\|_{\mathcal{H}}^2 = \|V_t - V^*\|_{\mathcal{H}}^2 &- 2\alpha_t \langle \hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t), V_t - V^* \rangle_{\mathcal{H}} \\ &+ 2\epsilon_t \|V_t - V^*\|_{\mathcal{H}} + \alpha_t^2 \|\tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \ . \end{aligned} \tag{8.44}$$

Now, with $\bar{\delta}_t = \mathbb{E}[\delta_t \,|\, \mathbf{x}_t, \pi(\mathbf{x}_t)]$, add and subtract $\hat{\nabla}_V J(V_t, \bar{\delta}_t; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)$, the stochastic quasi-gradient evaluated at $(V_t, \bar{\delta}_t)$ rather than (V_t, z_{t+1}), inside the inner-product term on the right-hand side of (8.44), to write

$$\begin{aligned}\|V_{t+1} - V^*\|_{\mathcal{H}}^2 = \|V_t - V^*\|_{\mathcal{H}}^2 &- 2\alpha_t \langle \hat{\nabla}_V J(V_t, \delta_t; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t), V_t - V^* \rangle_{\mathcal{H}} + 2\epsilon_t \|V_t - V^*\|_{\mathcal{H}} \\ &+ 2\alpha_t \langle (\gamma \kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot))(\bar{\delta}_t - z_{t+1}), V_t - V^* \rangle_{\mathcal{H}} \\ &+ \alpha_t^2 \|\tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \ , \end{aligned} \tag{8.45}$$

where we substitute in the definitions of $\hat{\nabla}_V J(V_t, \bar{\delta}_t; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)$ and $\hat{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)$ [cf. (8.11), (8.23), respectively] in (8.45), and cancel out the common regularization term λV_t. We define the directional error associated with difference between the stochastic quasi-gradient and the stochastic gradient as

$$v_t = 2\alpha_t \langle (\gamma \kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot))(\bar{\delta}_t - z_{t+1}), V_t - V^* \rangle_{\mathcal{H}} \tag{8.46}$$

From here, compute the expectation conditional on the algorithm history $\mathcal{F}_t$:

$$\begin{aligned}\mathbb{E}\left[\|V_{t+1} - V^*\|_{\mathcal{H}}^2 \,|\, \mathcal{F}_t\right] = \|V_t - V^*\|_{\mathcal{H}}^2 &- 2\alpha_t \langle \mathbb{E}\left[\hat{\nabla}_V J(V_t, \bar{\delta}_t; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) \,|\, \mathcal{F}_t\right], V_t - V^* \rangle_{\mathcal{H}} \\ &+ 2\epsilon_t \|V_t - V^*\|_{\mathcal{H}} + \mathbb{E}\left[v_t \,|\, \mathcal{F}_t\right] \\ &+ \alpha_t^2 \mathbb{E}\left[\|\tilde{\nabla}_V J(V_t, z_{t+1}; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \,|\, \mathcal{F}_t\right] \ . \end{aligned} \tag{8.47}$$

Note that the compositional objective $J(V)$ is convex with respect to V, which allows us to write

$$\langle \mathbb{E}\left[\hat{\nabla}_V J(V_t, \bar{\delta}_t; \mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) \,|\, \mathcal{F}_t\right], V_t - V^* \rangle_{\mathcal{H}} \geq J(V_t) - J(V^*) \ . \tag{8.48}$$

Now, we may use Assumption 23 [cf. (8.29)] regarding the finite conditional moments of the projected stochastic quasi-gradient to the last term in (8.47) so that it may be replaced by its upper-estimate, which together with (8.48) simplifies to

$$\mathbb{E}\left[\|V_{t+1} - V^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] = \|V_t - V^*\|_{\mathcal{H}}^2 - 2\alpha_t \left[J(V_t) - J(V^*)\right] + 2\epsilon_t \|V_t - V^*\|_{\mathcal{H}} + \alpha_t^2 \sigma_V^2 + \mathbb{E}\left[v_t \,\middle|\, \mathcal{F}_t\right] . \tag{8.49}$$

It remains to analyze v_t, the directional error associated with using stochastic quasi-gradients rather than stochastic gradients. In doing so, we derive the fact that the sub-optimality $\|V_t - V^*\|$ is intrinsically coupled to the auxiliary sequence $(z_{t+1} - \bar{\delta}_t)$, which is the focus of Lemma 8(3). Proceed by applying the Cauchy-Schwartz inequality to (8.46), which allows us to write

$$v_t \leq 2\alpha_t \|\gamma\kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot)\|_{\mathcal{H}}^2 |z_{t+1} - \bar{\delta}_t| \|V_t - V^*\|_{\mathcal{H}} \tag{8.50}$$

Note that $2ab \leq \rho a^2 + b^2/\rho$ for $\rho, a, b > 0$, which we apply to (8.50) with $a = |z_{t+1} - \bar{\delta}_t|$, $b = \alpha_t \|\gamma\kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot)\|_{\mathcal{H}} \|V_t - V^*\|_{\mathcal{H}}$, and $\rho = \beta_t$ so that (8.50) becomes

$$v_t \leq \beta_t (z_{t+1} - \bar{\delta}_t)^2 + \frac{\alpha_t^2}{\beta_t} \|\gamma\kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot)\|_{\mathcal{H}}^2 \|V_t - V^*\|_{\mathcal{H}}^2 . \tag{8.51}$$

The conditional mean of v_t [cf. (8.46)], using (8.51), is then

$$\begin{aligned} \mathbb{E}\left[v_t \,\middle|\, \mathcal{F}_t\right] &\leq \beta_t \mathbb{E}\left[(z_{t+1} - \bar{\delta}_t)^2 \,\middle|\, \mathcal{F}_t\right] + \frac{\alpha_t^2}{\beta_t} \mathbb{E}\left[\|\gamma\kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot)\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \|V_t - V^*\|_{\mathcal{H}}^2 \\ &\leq \beta_t \mathbb{E}\left[(z_{t+1} - \bar{\delta}_t)^2 \,\middle|\, \mathcal{F}_t\right] + \frac{\alpha_t^2}{\beta_t} G_V^2 \|V_t - V^*\|_{\mathcal{H}}^2 , \end{aligned} \tag{8.52}$$

where we apply the finite variance property of the functional component of the stochastic gradient [cf. (8.28)] for the final inequality (8.52). Now, substitute (8.52) into the right-hand side of (8.49) and gather like terms:

$$\begin{aligned} \mathbb{E}\left[\|V_{t+1} - V^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq \left(1 + \frac{\alpha_t^2}{\beta_t} G_V^2\right) \|V_t - V^*\|_{\mathcal{H}}^2 + 2\epsilon_t \|V_t - V^*\|_{\mathcal{H}} \\ - 2\alpha_t \left[J(V_t) - J(V^*)\right] + \alpha_t^2 \sigma_V^2 + \beta_t \mathbb{E}\left[(z_{t+1} - \bar{\delta}_t)^2 \,\middle|\, \mathcal{F}_t\right] . \end{aligned} \tag{8.53}$$

which is as stated in Lemma 8(2). ■

Proof of Lemma 8(3): This proof is an adaptation of Lemma 2 in Appendix G.1 in the Supplementary Material of [206] to the recursively averaged temporal difference sequence z_t defined in (8.12). Begin by defining the scalar quantity e_t as the difference of mean

temporal differences scaled by the forgetting factor β_t, i.e. $e_t = (1-\beta_t)(\bar{\delta}_t - \bar{\delta}_{t-1})$. Then we consider the difference of the evolution of the auxiliary variable z_{t+1} with respect to the conditional mean temporal difference $\bar{\delta}_t$, plus the difference of mean temporal differences:

$$\begin{aligned} z_{t+1} - \bar{\delta}_t + e_t &= (1-\beta_t)z_t + \beta_t\delta_t - [(1-\beta_t)\bar{\delta}_t + \beta_t\bar{\delta}_t] + (1-\beta_t)(\bar{\delta}_t - \bar{\delta}_{t-1}) \\ &= (1-\beta_t)\left(z_t - \bar{\delta}_{t-1}\right) + \beta_t(\delta_t - \bar{\delta}_t) \end{aligned} \tag{8.54}$$

where we make use of the definition of z_{t+1} in (8.12), the fact that $\bar{\delta}_t = [(1-\beta_t)\bar{\delta}_t + \beta_t\bar{\delta}_t]$, and the definition of e_t on the first line of (8.54), and in the second we gather terms with respect to coefficients $(1-\beta_t)$ and β_t, and cancel the redundant $\bar{\delta}_t$ term. Now, consider the square of the expression (8.54), using it's simplification on the right-hand side of the preceding expression

$$\begin{aligned} (z_{t+1} - \bar{\delta}_t + e_t)^2 &= [(1-\beta_t)\left(z_t - \bar{\delta}_{t-1}\right) + \beta_t(\delta_t - \bar{\delta}_t)]^2 \\ &= [(1-\beta_t)^2\left(z_t - \bar{\delta}_{t-1}\right)^2 + \beta_t^2(\delta_t - \bar{\delta}_t)^2 + 2(1-\beta_t)\beta_t\left(z_t - \bar{\delta}_{t-1}\right)(\delta_t - \bar{\delta}_t)\,. \end{aligned} \tag{8.55}$$

where we expand the square to obtain the second line in the previous expression. Now, compute the expectation of (8.55) conditional on the filtration $\mathcal{F}_t$, which yields

$$\begin{aligned} \mathbb{E}[(z_{t+1} - \bar{\delta}_t + e_t)^2 \,|\, \mathcal{F}_t] &= [(1-\beta_t)^2\left(z_t - \bar{\delta}_{t-1}\right)^2 + \beta_t^2\mathbb{E}[(\delta_t - \bar{\delta}_t)^2 \,|\, \mathcal{F}_t] \\ &\quad + 2(1-\beta_t)\beta_t\left(z_t - \bar{\delta}_{t-1}\right)\mathbb{E}[(\delta_t - \bar{\delta}_t) \,|\, \mathcal{F}_t]\,. \end{aligned} \tag{8.56}$$

Now we apply the assumption [cf. (8.27)] that the fact that the temporal difference δ_t is an unbiased estimator for its conditional mean $\bar{\delta}_t$ (so that the last term in the previous expression is null), with finite variance $\mathbb{E}[(\delta_t - \bar{\delta}_t)^2 \,|\, \mathcal{F}_t] \leq \sigma_\delta^2$ (Assumption 23), to write

$$\mathbb{E}[(z_{t+1} - \bar{\delta}_t + e_t)^2 \,|\, \mathcal{F}_t] = (1-\beta_t)^2\left(z_t - \bar{\delta}_{t-1}\right)^2 + \beta_t^2\sigma_\delta^2;\,. \tag{8.57}$$

We may use the relationship in (8.57) to obtain an upper estimate on the conditional mean square of $z_{t+1} - \bar{\delta}_t$ by using the inequality $\|a+b\|^2 \leq (1+\rho)\|a\|^2 + (1+1/\rho)\|b\|^2$ which holds for any $\rho > 0$: set $a = z_{t+1} - \bar{\delta}_t + e_t$, $b = -e_t$, and $\rho = \beta_t$. Therefore, we obtain

$$(z_{t+1} - \bar{\delta}_t)^2 \leq (1+\beta_t)(z_{t+1} - \bar{\delta}_t + e_t)^2 + \left(1 + \frac{1}{\beta_t}\right)e_t^2\,. \tag{8.58}$$

Now, we may use the conditionally expected value of (8.58) in lieu of (8.57), while gaining a multiplicative factor of $(1+\beta_t)$ on the right-hand side of (8.57) plus the error term

$(1 + 1/\beta_t)e_t$, yielding

$$\mathbb{E}[(z_{t+1} - \bar{\delta}_t)^2 \,\big|\, \mathcal{F}_t] = (1+\beta_t)[(1-\beta_t)^2 \left(z_t - \bar{\delta}_{t-1}\right)^2 + \beta_t^2 \sigma_\delta^2] + \left(\frac{1+\beta_t}{\beta_t}\right) e_t^2; . \tag{8.59}$$

Use the fact that $(1-\beta_t^2)(1-\beta_t) \leq (1-\beta_t)$ to the first term in (8.59) and $(1+\beta_t)\beta_t^2 \leq 2\beta_t^2$ to the second (since $\beta_t \in (0,1)$) to simplify (8.59) as

$$\mathbb{E}[(z_{t+1} - \bar{\delta}_t)^2 \,\big|\, \mathcal{F}_t] = (1-\beta_t) \left(z_t - \bar{\delta}_{t-1}\right)^2 + 2\beta_t^2 \sigma_\delta^2 + \left(\frac{1+\beta_t}{\beta_t}\right) e_t^2; . \tag{8.60}$$

From here, we turn to controlling the term involving e_t, which represents the difference of mean temporal differences. By definition, we have

$$|e_t| = (1-\beta_t)|(\bar{\delta}_t - \bar{\delta}_{t-1})| \leq (1-\beta_t) L_V \|V_t - V_{t-1}\|_{\mathcal{H}} \tag{8.61}$$

where we apply the Lipschitz continuity of the temporal difference with respect to the value function [cf. (8.30)] stated in Assumption 24. Substitute the right-hand side of (8.61) into (8.60), and simplify the expression in the last term as $(1-\beta_t^2)/\beta_t \leq 1/\beta_t$ to conclude (8.37). ■

Theorem 10 *Consider the sequence z_t [cf. (8.12)] and $\{V_t\}$ [cf. (8.16)] as stated in Algorithm 8. Assume the regularizer is positive $\lambda > 0$, Assumptions 22 - 24 hold, and the step-size conditions hold:* [3]

$$\sum_{t=1}^{\infty} \alpha_t = \infty \,, \quad \sum_{t=1}^{\infty} \beta_t = \infty \,, \quad \sum_{t=1}^{\infty} \alpha_t^2 + \beta_t^2 + \frac{\alpha_t^2}{\beta_t} < \infty \,, \quad \epsilon_t = \alpha_t^2 \tag{8.62}$$

Then $V_t \to V^*$ defined by (8.8) with probability 1, and thus achieves the regularized Bellman fixed point (8.4) restricted to the reproducing kernel Hilbert space.

Proof: First, we state a lemma which provides the technical foundation of this result:

Lemma 9 *(Coupled Supermartingale Theorem [205][Lemma 6]) Let $\{\xi_k\}$, $\{\zeta_k\}$, $\{u_k\}$, $\{\bar{u}_k\}$,*

[3] One step-size sequence satisfying (8.62) is $\alpha_t = \mathcal{O}(t^{-(3/4+\zeta/2)})$, $\beta_t = \mathcal{O}(t^{-(1+\zeta)/2})$, $\epsilon_t = \mathcal{O}(\alpha_t^2) = \mathcal{O}(t^{-(3/2+\zeta)})$, where $\zeta > 0$ is an arbitrarily small constant so that series $\sum_t \alpha_t$ and $\sum_t \beta_t$ diverge. Generally, satisfying (8.62), requires: $\alpha_t = \mathcal{O}(t^{-p_\alpha})$, $\beta_t = \mathcal{O}(t^{-p_\beta})$ with $p_\alpha \in (3/4, 1)$ and $p_\beta \in (1/2, 2p_\alpha - 1)$.

$\{\eta_k\}$, $\{\theta_k\}$, $\{\varepsilon_k\}$, $\{\mu_k\}$, $\{\nu_k\}$ be sequences of nonnegative random variables such that

$$\mathbb{E}[\xi_{k+1} \,|\, \mathcal{G}_k] \leq (1+\eta_k)\xi_k - u_k + c\theta_k\zeta_k + \mu_k \;, \tag{8.63}$$

$$\mathbb{E}[\zeta_{k+1} \,|\, \mathcal{G}_k] \leq (1-\theta_k)\zeta_k - \bar{u}_k + \varepsilon_k\xi_k + \nu_k \;, \tag{8.64}$$

where $\mathcal{G}_k = \{\xi_s, \zeta_s, u_s, \bar{u}_s, \eta_s, \theta_s, \varepsilon_s, \mu_s, \nu_s\}_{s=0}^k$ is the filtration, and $c > 0$ is a scalar. Suppose the following summability conditions hold:

$$\sum_{k=0}^{\infty} \eta_k < \infty \;, \quad \sum_{k=0}^{\infty} \varepsilon_k < \infty \;, \; \sum_{k=0}^{\infty} \mu_k < \infty \;, \; \sum_{k=0}^{\infty} \nu_k < \infty \;, \; \textit{almost surely.} \tag{8.65}$$

Then ξ_k and ζ_k converge almost surely to two respective nonnegative random variables, and we may conclude that

$$\sum_{k=0}^{\infty} u_k < \infty \;, \quad \sum_{k=0}^{\infty} \bar{u}_k < \infty \;, \; \sum_{k=0}^{\infty} \theta_k\zeta_k < \infty \;, \; \textit{almost surely.} \tag{8.66}$$

We can use Lemma 9 to establish convergence with probability 1 of Algorithm 8 by constructing a coupled supermartingale of the form in Lemma 9. First, consider the expression (8.36) for the value function sub-optimality, using approximation budget $\epsilon_t = \alpha_t^2$ and the fact that the value function is bounded in Hilbert norm [cf. (8.31)] to obtain $\|V_t - V^*\|_{\mathcal{H}} \leq 2K$:

$$\begin{aligned}\mathbb{E}\left[\|V_{t+1} - V^*\|_{\mathcal{H}}^2 \,|\, \mathcal{F}_t\right] \leq \left(1 + \frac{\alpha_t^2}{\beta_t} G_V^2\right) \|V_t - V^*\|_{\mathcal{H}}^2 - 2\alpha_t \left[J(V_t) - J(V^*)\right] \\ + \alpha_t^2(\sigma_V^2 + 4K) + \beta_t \mathbb{E}\left[(z_{t+1} - \bar{\delta}_t)^2 \,|\, \mathcal{F}_t\right] \;.\end{aligned} \tag{8.67}$$

and then substitute (8.37) regarding the evolution of z_t with respect to its conditional expectation into (8.67) to obtain :

$$\begin{aligned}\mathbb{E}\left[\|V_{t+1} - V^*\|_{\mathcal{H}}^2 \,|\, \mathcal{F}_t\right] \leq \left(1 + \frac{\alpha_t^2}{\beta_t} G_V^2\right) \|V_t - V^*\|_{\mathcal{H}}^2 - 2\alpha_t \left[J(V_t) - J(V^*)\right] \\ + \alpha_t^2(\sigma_V^2 + 4K) + \beta_t(1-\beta_t)(z_t - \bar{\delta}_{t-1})^2 + L_V\|V_t - V_{t-1}\|_{\mathcal{H}}^2 + 2\beta_t^3\sigma_\delta^2 \;.\end{aligned} \tag{8.68}$$

Assume that $\beta_t \in (0,1)$ for all t, so that the right-hand side of (8.68) may be simplified to

$$\begin{aligned}\mathbb{E}\left[\|V_{t+1} - V^*\|_{\mathcal{H}}^2 \,|\, \mathcal{F}_t\right] \leq \left(1 + \frac{\alpha_t^2}{\beta_t} G_V^2\right) \|V_t - V^*\|_{\mathcal{H}}^2 - 2\alpha_t \left[J(V_t) - J(V^*)\right] \\ + \beta_t(z_t - \bar{\delta}_{t-1})^2 + \alpha_t^2(\sigma_V^2 + 4K) + L_V\|V_t - V_{t-1}\|_{\mathcal{H}}^2 + 2\beta_t^2\sigma_\delta^2 \;.\end{aligned} \tag{8.69}$$

We may identify (8.69) with the first supermartingale relationship in Lemma 9 [cf. (8.63)] via the identifications

$$\xi_t = \|V_t - V^*\|_{\mathcal{H}}^2 \, , \eta_t = \frac{\alpha_t^2}{\beta_t} G_V^2 \, , u_t = 2\alpha_t \left[J(V_t) - J(V^*) \right] \, , \; c = 1 \, , \tag{8.70}$$
$$\zeta_t = (z_t - \bar{\delta}_{t-1})^2 \, , \theta_t = \beta_t \, , \; \mu_t = \alpha_t^2(\sigma_V^2 + 4K) + L_V \|V_t - V_{t-1}\|_{\mathcal{H}}^2 + 2\beta_t^2 \sigma_\delta^2$$

where $u_t \geq 0$ by the definition of the optimal objective $J(V^*)$. The summability of μ_t may be established as follows: consider summing the expression in Lemma 8(1) for all t, which by the fact that $\sum_t \alpha_t^2 < \infty$ [cf. (8.62)], implies that the conditional mean series is finite. Consequently, $\sum_{t=0}^{\infty} \|V_t - V_{t-1}\|_{\mathcal{H}}^2 < \infty$ with probability 1 using the fact that $\|V_t - V_{t-1}\|_{\mathcal{H}}$ is bounded. Thus $\sum_t \mu_t < \infty$.

Now, let's connect the evolution of the auxiliary temporal difference sequence z_t (8.12) in Lemma 8(3). In particular, (8.37) is related to (8.64) via the identifications:

$$\bar{u}_t = 0 \, , \; \varepsilon_t = 0 \, , \; \nu_t = \frac{L_V}{\beta_t} \|V_t - V_{t-1}\|_{\mathcal{H}}^2 + 2\beta_t^2 \sigma_\delta^2 \, , \tag{8.71}$$

with $\zeta_t = (z_t - \bar{\delta}_{t-1})^2$ and $\theta_t = \beta_t$ as in (8.70). The summability of ν_t follows the following logic: consider the expression $\|V_t - V_{t-1}\|_{\mathcal{H}}^2 / \beta_t$ which is of order $\mathcal{O}(\alpha_t^2/\beta_t)$ in conditional expectation by Lemma 8(1). Sum the resulting conditional expectation for all t, which by the summability of the sequence $\sum_t \alpha_t^2/\beta_t < \infty$ is finite. Therefore, $\sum_t \|V_t - V_{t-1}\|_{\mathcal{H}}^2/\beta_t < \infty$ almost surely.

Together with the conditions on the step-size sequences α_t and β_t (8.62), the summability conditions (8.65) of Lemma 9, the Coupled Supermartingale Theorem, are satisfied, which allows us to conclude that $\xi_t = \|V_t - V^*\|_{\mathcal{H}}^2$ and $\zeta_t = (z_t - \bar{\delta}_{t-1})^2$ converge to two nonnegative random variables with probability 1, and that:

$$\sum_t \alpha_t \left[J(V_t) - J(V^*) \right] < \infty \, , \quad \sum_t \beta_t (z_{t+1} - \bar{\delta}_t)^2 < \infty \, , \quad \text{almost surely.} \tag{8.72}$$

The non-summability of the step-size sequences α_t and β_t (8.62) allows us to conclude that:

$$\liminf_{t \to \infty} J(V_t) = J(V^*) \, , \quad \liminf_{t \to \infty} (z_{t+1} - \bar{\delta}_t)^2 = 0 \, , \quad \text{almost surely.} \tag{8.73}$$

Then, the convergence of the whole sequence $\|V_t - V^*\|_{\mathcal{H}}^2$ implies that this sequence is bounded with probability 1. Then, since $J(V_t) \to J(V^*)$ almost surely along a subsequence, $V_t \to V^*$ almost sure along a subsequence using the continuity of $J(V)$. However, since the whole sequence $\|V_t - V^*\|_{\mathcal{H}}^2$ converges to a unique limit, the whole sequence $\{V_t\}$ converges to V^* with probability 1.

■

Theorem 10 states that the value functions generated by Algorithm 8 converge almost surely to the optimal V^* defined by (8.8). With regularizer λ made arbitrarily small but nonzero, using a universal kernel (e.g., a Gaussian), V_t converges arbitrarily close to a function satisfying Bellman's equation in *infinite MDPs* (8.3). This is the first guarantee w.p.1 for a true stochastic descent method with an infinitely and nonlinearly parameterized value function. Theorem 10 requires attenuating step-sizes such that the stochastic approximation error approaches null. In contrast, constant learning rates allow for the perpetual revision of the value function estimates without diminishing algorithm adaptivity, motivating the following result.

Theorem 11 *Suppose Algorithm 8 is run with constant positive learning rates $\alpha_t = \alpha$ and $\beta_t = \beta$ and constant compression budget $\epsilon_t = \epsilon$ with sufficiently large regularization, i.e.*

$$0 < \beta < 1 \, , \alpha = \beta, \epsilon = C\alpha^2, \lambda = G_V^2 \frac{\alpha}{\beta} + \lambda_0 \tag{8.74}$$

where $C > 0$ is a scalar, and $0 < \lambda_0 < 1$. Then, under Assumptions 22 - 24, the sub-optimality sequence $\|V_t - V^*\|_{\mathcal{H}}^2$ converges in mean to a neighborhood:

$$\limsup_{t \to \infty} \mathbb{E}\left[\|V_t - V^*\|_{\mathcal{H}}^2\right] = \mathcal{O}\left(\alpha + \alpha^2 + \alpha^3\right) \, . \tag{8.75}$$

Proof:

Before analyzing the mean convergence behavior of the value function, we consider the mean sub-optimality of the auxiliary variable z_t with respect to the conditional mean of the temporal difference $\bar{\delta}_t$. To do so, compute the total expectation of Lemma 8(3), stated as

$$\mathbb{E}\left[(z_{t+1} - \bar{\delta}_t)^2\right] \leq (1-\beta)\mathbb{E}\left[(z_t - \bar{\delta}_{t-1})^2\right] + \frac{L_V}{\beta}\mathbb{E}\left[\|V_t - V_{t-1}\|_{\mathcal{H}}^2\right] + 2\beta^2\sigma_\delta^2 \, , \tag{8.76}$$

where we have substituted in constant learning rate $\beta_t = \beta$ in (8.76). The total expectation of Lemma 8(1) regarding $\|V_t - V_{t-1}\|_{\mathcal{H}}^2$, the difference of value functions in Hilbert-norm, may be substituted into (8.76), with constant step-size $\alpha_t = \alpha$ and compression budgets $\epsilon_t = \epsilon$ to obtain

$$\mathbb{E}\left[(z_{t+1} - \bar{\delta}_t)^2\right] \leq (1-\beta)\mathbb{E}\left[(z_t - \bar{\delta}_{t-1})^2\right] + \frac{2L_V}{\beta}\left[\alpha^2(G_\delta^2 G_V^2 + \lambda^2 K^2) + \epsilon^2\right] + 2\beta^2\sigma_\delta^2 \, , \tag{8.77}$$

Observe that (8.79) gives a relationship between the sequence $\mathbb{E}\left[(z_{t+1} - \bar{\delta}_t)^2\right]$ and its value at the previous iterate. We can substitute $t+1$ by t in (8.79) to write

$$\mathbb{E}\left[(z_t - \bar{\delta}_{t-1})^2\right] \leq (1-\beta)\mathbb{E}\left[(z_{t-1} - \bar{\delta}_{t-2})^2\right] + \frac{2L_V}{\beta}\left[\alpha^2(G_\delta^2 G_V^2 + \lambda^2 K^2) + \epsilon^2\right] + 2\beta^2\sigma_\delta^2 \, , \tag{8.78}$$

Substituting (8.78) into the right-hand side of (8.79) yields

$$\mathbb{E}\big[(z_{t+1}-\bar{\delta}_t)^2\big] \leq (1-\beta)^2\mathbb{E}\left[(z_{t-1}-\bar{\delta}_{t-2})^2\right]+[1+(1-\beta)]\Big\{\frac{2L_V}{\beta}\big[\alpha^2(G_\delta^2 G_V^2+\lambda^2K^2)+\epsilon^2\big]+2\beta^2\sigma_\delta^2\Big\}. \tag{8.79}$$

We can recursively apply the previous two steps backwards in time to the initialization to obtain

$$\mathbb{E}\left[(z_{t+1}-\bar{\delta}_t)^2\right] \leq (1-\beta)^{t+1}(z_0-\bar{\delta}_{-1})^2+\sum_{u=0}^{t}(1-\beta)^u\Big\{\frac{2L_V}{\beta}\big[\alpha^2(G_\delta^2 G_V^2+\lambda^2K^2)+\epsilon^2\big]+2\beta^2\sigma_\delta^2\Big\}\,, \tag{8.80}$$

In (8.80), the first term on the left-hand side vanishes due to the initialization $z_0 = 0$ and the convention $\delta_{-1} = 0$. Moreover, the finite geometric sum may be evaluated, provided $\beta < 1$, as $\sum_{u=0}^{t}(1-\beta)^u = [1-(1-\beta)^t]/\beta$. The numerator in this simplification is strictly less than unit, which means that the right-hand side of (8.80) simplifies to

$$\mathbb{E}\left[(z_{t+1}-\bar{\delta}_t)^2\right] \leq \frac{2L_V}{\beta^2}\big[\alpha^2(G_\delta^2 G_V^2+\lambda^2K^2)+\epsilon^2\big]+2\beta\sigma_\delta^2 = \mathcal{O}\left(\frac{\alpha^2+\epsilon^2}{\beta^2}+\beta\right) \tag{8.81}$$

With this relationship established for the auxiliary sequence z_t, we shift gears to addressing the evolution of the value function sub-optimality $\|V_t - V^*\|_{\mathcal{H}}$ in expectation. Begin by using the fact that the Hilbert-norm regularizer $(\lambda/2)\|V\|_{\mathcal{H}}^2$ in (8.8) implies the objective $J(V)$ is strongly convex, i.e.

$$\frac{\lambda}{2}\|V_t - V^*\|_{\mathcal{H}}^2 \leq J(V_t) - V(V^*)\,, \tag{8.82}$$

together with the expression in Lemma 8(2) regarding the evolution of the value function sub-optimality, assuming constant learning rates and compression budget, i.e. $\alpha_t = \alpha, \beta_t = \beta, \epsilon_t = \epsilon$, to write

$$\begin{aligned}\mathbb{E}\left[\|V_{t+1}-V^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq \left(1+\frac{\alpha^2}{\beta}G_V^2-\alpha\lambda\right)\|V_t-V^*\|_{\mathcal{H}}^2+2\epsilon\|V_t-V^*\|_{\mathcal{H}} \\ +\alpha^2\sigma_V^2+\beta\mathbb{E}\left[(z_{t+1}-\bar{\delta}_t)^2\,\middle|\,\mathcal{F}_t\right]\,.\end{aligned} \tag{8.83}$$

Consider the total expectation of (8.83), using choice of compression budget $\epsilon = C\alpha^2$ for some arbitrary constant $C > 0$, the fact that $\|V_t - V^*\|_{\mathcal{H}} \leq 2K$, and applying (8.81) to the

last term on the right-hand side of the preceding expression to obtain:

$$\mathbb{E}\left[\|V_{t+1}-V^*\|_{\mathcal{H}}^2\right] \leq \left(1+\frac{\alpha^2}{\beta}G_V^2-\alpha\lambda\right)\mathbb{E}\left[\|V_t-V^*\|_{\mathcal{H}}^2\right]+\alpha^2(\sigma_V^2+4CK)+2\beta^2\sigma_\delta^2 \quad (8.84)$$
$$+\frac{2L_V}{\beta}\left[\alpha^2(G_\delta^2G_V^2+\lambda^2K^2)+C^2\alpha^4\right] \ .$$

From (8.84), substitute in the regularizer selection $\lambda = G_V^2\alpha/\beta+\lambda_0$ for $\lambda_0<1$. We may establish asymptotic convergence to a neighborhood by analyzing the conditions for which we have a decreasing sequence, i.e., the following holds

$$\begin{aligned}\mathbb{E}\left[\|V_{t+1}-V^*\|_{\mathcal{H}}^2\right] &\leq (1-\lambda_0)\,\mathbb{E}\left[\|V_t-V^*\|_{\mathcal{H}}^2\right]+\alpha^2(\sigma_V^2+4CK)+2\beta^2\sigma_\delta^2 \\ &\quad+\frac{2L_V}{\beta}\left[\alpha^2(G_\delta^2G_V^2+(G_V^2\alpha/\beta+\lambda_0)^2K^2)+C^2\alpha^4\right] \\ &\leq \mathbb{E}\left[\|V_t-V^*\|_{\mathcal{H}}^2\right]\end{aligned} \quad (8.85)$$

Partition the set of time indices $\{t\geq 0\}$ into two disjoint sets $\{t_k\}$ and $\{t_j\}$, and suppose that (8.85) holds along subsequence $\mathbb{E}\left[\|V_{t_k}-V^*\|_{\mathcal{H}}^2\right]$ associated with time indices $\{t_k\}$. We may simplify the condition in (8.85) for this subsequence to

$$\begin{aligned}&\lambda_0^{-1}\left(\alpha^2(\sigma_V^2+4K)+2\beta^2\sigma_\delta^2+\frac{2L_V}{\beta}\left[\alpha^2(G_\delta^2G_V^2+(G_V^2\alpha/\beta+\lambda_0)^2K^2)+C^2\alpha^4\right]\right) \\ &\quad=\mathcal{O}\left(\alpha^2+\beta^2+\frac{\alpha^2}{\beta}\left[1+\alpha^2+\frac{\alpha}{\beta}+\frac{\alpha^2}{\beta^2}\right]\right)\leq \mathbb{E}\left[\|V_{t_k}-V^*\|_{\mathcal{H}}^2\right] \ .\end{aligned} \quad (8.86)$$

For this subsequence, since (8.85) holds, $\mathbb{E}\left[\|V_{t_k}-V^*\|_{\mathcal{H}}^2\right]$ is decreasing, and since it is bounded, it thus converges to its infimum by the Monotone Convergence Theorem. The infimum of $\mathbb{E}\left[\|V_{t_k}-V^*\|_{\mathcal{H}}^2\right]$ is the left-hand side of (8.86), so that we may write

$$\lim_{t\to\infty}\mathbb{E}\left[\|V_{t_k}-V^*\|_{\mathcal{H}}^2\right]=\mathcal{O}\left(\alpha^2+\beta^2+\frac{\alpha^2}{\beta}\left[1+\alpha^2+\frac{\alpha}{\beta}+\frac{\alpha^2}{\beta^2}\right]\right) \quad (8.87)$$

For all elements of the sequence $\mathbb{E}\left[\|V_t-V^*\|_{\mathcal{H}}^2\right]$ not part of the subsequence of indices $\{t_k\}$, i.e., those associated with $\{t_j\}$, the condition in (8.86) fails to hold:

$$\mathbb{E}\left[\|V_{t_j}-V^*\|_{\mathcal{H}}^2\right]<\mathcal{O}\left(\alpha^2+\beta^2+\frac{\alpha^2}{\beta}\left[1+\alpha^2+\frac{\alpha}{\beta}+\frac{\alpha^2}{\beta^2}\right]\right) \ . \quad (8.88)$$

The statements in (8.87) and (8.88) taken together imply

$$\limsup_{t\to\infty}\mathbb{E}\left[\|V_t-V^*\|_{\mathcal{H}}^2\right]=\mathcal{O}\left(\alpha^2+\beta^2+\frac{\alpha^2}{\beta}\left[1+\alpha^2+\frac{\alpha}{\beta}+\frac{\alpha^2}{\beta^2}\right]\right) \ . \quad (8.89)$$

When $\alpha = \beta$, the posynomial of the learning rates on the right-hand side of (8.89) simplifies to be $\mathcal{O}(\alpha + \alpha^2 + \alpha^3)$, which is as stated in (8.75) (Theorem 11).

■

Theorem 11 establishes that the value function estimates generated by Algorithm 8 converge in expectation to a neighborhood when constant step-sizes α and β and sparsification budget ϵ in Algorithm 5 are small constants. In particular, the bias ϵ induced by sparsification does not cause instability even when it is *not going to null*. Moreover, this result only holds when the regularizer λ is chosen large enough, which numerically induces a forgetting factor on past kernel dictionary weights (8.20). We may make the learning rates α and β arbitrarily small, which yield a proportional decrease in the radius of convergence to a neighborhood of the Bellman fixed point (8.3).

Remark 8 (Aggressive Constant Learning Rates) In practice, one may obtain better performance by using larger constant step-sizes. To do so, the criterion (8.74) may be relaxed: we require $0 < \beta < 1$ but $\alpha > 0$ may be any positive scalar. Then, the radius of convergence is

$$\limsup_{t\to\infty} \mathbb{E}\left[\|V_t - V^*\|_{\mathcal{H}}^2\right] = \mathcal{O}\left(\alpha^2 + \beta^2 + \frac{\alpha^2}{\beta}\left[1 + \alpha^2 + \frac{\alpha}{\beta} + \frac{\alpha^2}{\beta^2}\right]\right) . \tag{8.90}$$

The ratios α^2/β and α^2/β^2 dominate (8.90) and must be made small to obtain accurate solutions.

Theorem 11 is the first constant learning rate result for nonparametric compositional stochastic programming of which we are aware, and allows for repeatedly revising value function without the need for stochastic approximation error to approach null. Use of constant learning rates yields the fact that value function estimates have moderate complexity even in the worst case, as we detail next.

Model Order Control As noted in Section 8.3, the complexity of functional stochastic quasi-gradient method in a RKHS is of order $\mathcal{O}(2(t-1))$ which grows without bound. To mitigate this issue, we develop the sparse subspace projection in Section 8.3.1. We formalize here that this projection does indeed limit the complexity of the value function when constant learning rates and compression budget are used. This result is a corollary, since it is an extension of Theorem 3 in [98]. To obtain this result, the reward function must be bounded, as we state next.

AS25 *The reward function* $r : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$ *is bounded for all* $\mathbf{x}, \mathbf{a}, \mathbf{y}$*, i.e.,*

$$r(\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t) \leq R_{\max} \quad \textit{for all } t \tag{8.91}$$

Assumption 25 holds whenever the reward function is continuous and the state and action spaces are compact, and thus is not restrictive as these conditions are met in most practical settings. In this setting, we have the following finite-memory property of Algorithm 8.

Corollary 4 *Denote V_t as the value function sequence defined by Algorithm 8 with constant step-sizes $\alpha_t = \alpha$ and $\beta_t = \beta \in (0,1)$ with compression budget $\epsilon_t = \epsilon = C\alpha^2$ and regularization parameter $\lambda = (\alpha/\beta)G_V^2 + \lambda_0 = \mathcal{O}(\alpha\beta^{-1} + 1)$ as in Remark 8. Let M_t be the model order of the value function V_t i.e., the number of columns of the dictionary $\mathbf{D}_t$ which parameterizes V_t. Then there exists a finite upper bound M^∞ such that, for all $t \geq 0$, the model order is always bounded as $M_t \leq M^\infty$. Consequently, the model order of the limiting function $V^\infty = \lim_t V_t$ is finite.*

Proof:

The proof of Corollary 4 is similar to that of Theorem 3 of [98][Appendix D.1]. In that result, it is established for a nonparametric stochastic program without any compositional structure that the effect of sparse subspace projections on the functional stochastic gradient sequence in an RKHS is to yield a function sequence of finite model order, provided a constant algorithm step-size and compression budget are used. The proof of Corollary 4 is nearly identical: the same projection operator is used and the same compactness properties of the state and action spaces apply. The only point of departure is that a distinct deterministic bound is needed on the functional stochastic quasi-gradient for all $\{\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t\}$, i.e., to apply the reasoning following equations (74) in [98][Appendix D.1], we require the existence of a deterministic constant D such that $|[\gamma\kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot)]z_{t+1}| \leq D$ for all $\{\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t\}$. We turn to establishing such an upper-estimate. To do so, we first establish that the auxiliary sequence z_t stated in (8.12) is bounded, i.e.

Proposition 8 *The auxiliary sequence z_t [cf. (8.12)] satisfies the following upper bound when constant step-size $\beta_t = \beta$ is used :*

$$|z_t| = (\gamma + 1)K + R_{\max} \text{ for all } t \tag{8.92}$$

Proof: We pursue a proof by induction. First, the base case: with $V_0 = 0$, we have $|z_1| \leq \beta R_{\max} \leq (\gamma + 1)K + R_{\max}$ making use of the bound on V_t for all t in (8.31) and the fact that the step-size is less than unit. Now we consider the induction step: assume the prior bound holds for z_u for $u \leq t$. Write for z_{t+1}

$$|z_{t+1}| = (1 - \beta)|z_t| + \beta|\delta_t| \leq (\gamma + 1)K + R_{\max} \tag{8.93}$$

where in the last inequality we apply the induction hypothesis together with the upper-estimate on the temporal difference $\delta_t \leq (\gamma+1)K + R_{\max}$. ■

By making use of Proposition 8 together with the bound on the reproducing kernel map (Assumption 22), we have the following uniform deterministic bound:

$$|[\gamma\kappa(\mathbf{y}_t, \cdot) - \kappa(\mathbf{x}_t, \cdot)]z_{t+1}| \leq X(\gamma+1)[(\gamma+1)K + R_{\max}] := D \text{ for all } \{\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{y}_t\} \quad (8.94)$$

Then, we may apply the same reasoning as that of Appendix D.1 of [98] which allows us to conclude that the number of Euclidean balls of radius $d = \epsilon/D$ needed to cover the space $\phi(\mathcal{X}) = \kappa(\mathcal{X}, \cdot)$ is finite, where ϵ is a constant as in (8.74). See [7, 59] for further details. Therefore, for Algorithm 8, there exists a finite $M^\infty < \infty$ such that the model order $M_t \leq M^\infty$ for all t. ■

The results above establish that Algorithm 8 yields convergent behavior for the problem (8.8) in both diminishing and constant step-size regimes. With diminishing step-sizes [cf. (8.62)] and compression budget $\epsilon_t = \mathcal{O}(\alpha_t^2)$, we obtain exact convergence with probability 1 of the function sequence $\{V_t\}$ in the RKHS to that of the regularized Bellman fixed point of the evaluation equation V^* (Theorem 10). This result holds for any positive regularizer $\lambda > 0$, and thus can be made arbitrarily close to the *true Bellman fixed point* V^π [cf. (8.2)] by decreasing λ. However, an exact solution requires increasing the complexity of the function estimate such that its limiting memory becomes infinite. This drawback motivates us to consider the case where both the learning rates $\alpha_t = \alpha$, $\beta_t = \beta$ and the compression budget $\epsilon_t = \epsilon$ are constant. Under specific selections (8.74), the algorithm converges to a neighborhood of the optimal value function, whose radius depends on the step-sizes, and may be made small by decreasing α at the cost of a decreasing learning rate. Moreover, the use of constant step-sizes and compression budget with large enough regularization yields a value function parameterized by a dictionary whose model order is always bounded (Corollary 4). These results are summarized in Table 8.1.

8.5 Experiments with Stochastic Quasi-Gradient-Based Policy Evaluation

Our experiments aim to compare PKGTD to other policy evaluation techniques in this domain. Because it seeks memory-efficient solutions over an RKHS, we expect PKGTD to obtain accurate estimates of the value function using only a fraction of the memory required by the other methods. We perform experiments on the classical Mountain Car domain [185]: an agent applies discrete actions $\mathcal{A} = \{\texttt{reverse}, \texttt{coast}, \texttt{forward}\}$ to a car that starts at the

Table 8.1: Summary of convergence results for different parameter selections.

	Diminishing	Constant
Learning rate	$\sum_{t=1}^{\infty} \alpha_t^2 + \beta_t^2 + \frac{\alpha_t^2}{\beta_t} < \infty$	$\alpha > 0,\ \beta \in (0, 1)$
Compression Budget	$\epsilon_t = \mathcal{O}(\alpha_t^2)$	$\epsilon = \mathcal{O}(\alpha^2)$
Regularization	$0 < \lambda$	$\lambda = \mathcal{O}(\alpha\beta^{-1} + 1)$
Convergence Result	$V_t \to V^*$ a.s.	$\sup \mathbb{E}\left[\|V_t - V^*\|_{\mathcal{H}}^2\right] \to \mathcal{O}\left(\alpha + \alpha^2 + \alpha^3\right)$
Model Order	None	Finite

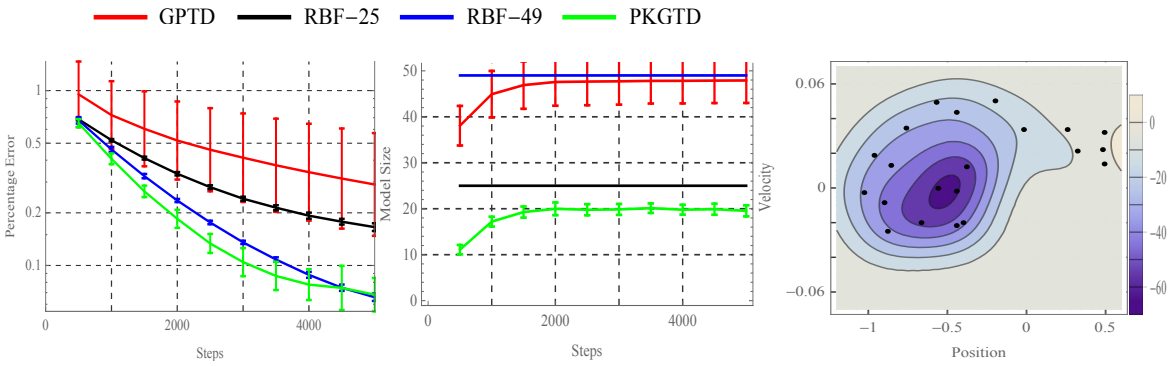


Figure 8.1: Experimental comparison of PKGTD to existing kernel methods for policy evaluation on the Mountain Car task. Test set error (left), and the parameterization complexity (center) vs. iterations. PKGTD learns fastest and most stably with the least complexity (best viewed in color). We plot the contour of the learned value function (right): its minimal value is in the valley, and states near the goal are close to null. Bold black dots are kernel dictionary elements, or retained instances.

bottom of a valley and attempts to climb up to a goal at the top of one of the mountain sides. The state space is continuous, consisting of the car's scalar position and velocity, i.e., $\mathcal{X} = \mathbb{R}^2$. The reward function $r(\mathbf{x}_t, \mathbf{a}_t, \mathbf{y}_t)$ is -1 unless $\mathbf{y}_t$ is the goal state at the mountain top, in which case it is 0 and the episode terminates.

To obtain a benchmark policy for this task, we make use of trust region policy optimization [171]. To evaluate value function estimates, we form an offline training set of state transitions and associated rewards by running this policy through consecutive episodes until we had one training trajectory of 5000 steps and then repeat this for 100 training trajectories to generate sample statistics. For ground truth, we generate one long trajectory of 10000 steps and randomly sample 2000 states from it. From each of these 2000 states, we apply the policy until episode termination and use the observed discounted return as $\hat{V}_\pi(\mathbf{x})$. Since our policy was deterministic, we only performed this procedure

once per sampled state. For value function V, we define the percentage error metric: Percentage Error$(V) = (1/2000)\sum_{i=1}^{2000} |(V(\mathbf{x}_i) - \hat{V}_\pi(\mathbf{x}_i))/\hat{V}_\pi(\mathbf{x}_i)|$ We compared PKGTD with a Gaussian kernel to two other techniques for policy evaluation that also use kernel-based value function representations: (1) Gaussian process temporal difference (GPTD) [60], and (2) gradient temporal difference (GTD) [186] using radial basis function (RBF) network features.

The Mountain Car environment has a two-dimensional state space (position and velocity) with bounds of $[-1.2, 0.6]$ in position, and $[-0.07, 0.07]$ in velocity. We chose not to normalize this state space to $[0, 1]$ intervals, choosing instead to handle the scale difference by using non-isotropic kernels. The ratio of the kernel variances is equal to the ratio of the lengths of their corresponding bounds, so they would be isotropic kernels if we normalized the state space.

We used a fixed non-isotropic kernel bandwidth of $\sigma_1 = 0.2, \sigma_2 = 0.0156$ in all cases. By fixing the kernel bandwidth across all algorithms, we are basically enforcing that the learned functions all belong to the same Kernel Hilbert Space.

For PKGTD, the relevant parameters are the step size, α, the rate of expectation update, β, the regularizer, λ, and the approximation error, K. For GPTD, the relevant parameters are the gaussian process noise standard deviation, σ_0, the linear independence test bound, ν, and the regularizer, λ. For the RBF grids fit using GTD, the relevant parameters are the grid spacing in the position and velocity directions, h_1 and h_2, respectively, the step size, α, and the rate of expectation update, β. Our values are summarized in Table. 8.2.

Table 8.2: Experiment Parameters

	α	β	λ	K	σ_0	ν	h_1	h_2
PKGTD	8.0	0.2	1e-6	0.02				
GPTD			1e-6		0.01	0.2		
RBF-25	10.0	0.25					0.44	0.0343
RBF-49	1.5	0.35					0.26	0.0203

Figure 8.1 depicts the results of our experiment. We fix a kernel bandwidth across all techniques, and select parameter values that yield the best results for each method. For RBF feature generation, we use two fixed grids with different spacing. The first was one for which GTD yielded a value function estimate with percentage error similar to that which we obtained using PKGTD (RBF-49), and the second was one which yielded a number of basis functions that was similar to what PKGTD selected (RBF-25). Observe that GTD with fixed RBF features requires a much denser grid in order to reach the same Percentage Error as Algorithm 8. Moreover, PKGTD's adaptive instance selection results in both faster initial learning and smaller error. Compared to GPTD, which chooses model points online

according to a fixed linear-dependence criterion, PKGTD requires fewer model points and converges to a better estimate of the value function more quickly and stably.

8.6 Implications of Gradient Temporal Difference Learning in infinite MDPs

In this chapter, we considered the problem of policy evaluation in infinite MDPs with value functions that belong to a RKHS. To solve this problem, we extended recent SQG methods for compositional stochastic programming to a RKHS, and used the result, combined with greedy sparse subspace projection, in a new policy-evaluation procedure called PKGTD (Algorithm 8). Under diminishing step sizes, PKGTD solves Bellman's evaluation equation exactly under the hypothesis that its fixed point belongs to a RKHS (Theorem 10). Under constant step sizes, we can further guarantee finite-memory approximations (Corollary 4) that still exhibit mean convergence to a neighborhood of the optimal value function (Theorem 11). In our Mountain Car experiments, PKGTD yields excellent sample efficiency and model complexity, and therefore holds promise for large state space problems common in robotics where fixed state-action space tiling may prove impractical.

We believe PKGTD will provide a solid foundation for making policy iteration affordable in infinite MDPs by exploiting the properties of RKHS. Moreover, the framework of nonparametric compositional stochastic optimization is sufficiently general as to possibly be used to solve Bellman's optimality equation, and thus find the *optimal* action-value function. Further generalizations of nonparametric compositional stochastic optimization are possible, and hold important implications for robust statistical estimation that could allow one to encode data volatility directly into an estimate, an important property in financial time series analysis and econometrics.

Chapter 9

Conclusions and Future Directions

This dissertation focused on the mathematical underpinnings of adaptive statistical learning in multi-agent systems. In Part I, we developed decentralized stochastic optimization algorithms based on primal-dual method to solve learning problems where each agent seeks to learn a common linear statistical model. We obtained theoretical stability and observed it translates into practice, but on the other hand, we noted that GLMs are too restrictive to obtain satisfactory statistical accuracy on problems we actually want to solve. Nonetheless, the lessons learnt from the approaches for these simple problems inform subsequent development of more effective statistical learning techniques.

In Part II, we used dictionary learning methods to go beyond GLMs and solve a real problem: a ground robot platform was able to anticipate control uncertainty based on its past experience using task-driven dictionary learning. Heartened by this successful implementation, we considered the use of dictionary learning methods in multi-agent settings, but here the limitations associated with non-convexity caused computational problems. In order to obtain stability for multi-agent dictionary learning, we required restrictive assumptions on stochastic approximation errors, and substantially smaller learning rates, which renders them impractical (Chapter 5). This limitation suggests the need for new decentralized stochastic *non-convex* optimization tools which are provably stable under general conditions. Initial contributions towards this class of problems have appeared recently [52], and their extensions to online settings may be more promising than primal-dual method for developing online task-driven dictionaries in multi-agent systems.

Parsimonious Online Learning with Kernels Observing the computational benefits of convexity in Parts I relative to Part II, as well as the advantages of nonlinear statistical models relative to linear ones in Part II versus Part I, we shifted focus in Part III to stochastic optimization problems defined over reproducing kernel Hilbert spaces (RKHS). This problem class allows for the learning of nonlinear statistical models while preserving convexity. This increased descriptive power comes at the cost of increased complexity

(infinite, to be exact), which motivated the contribution of Chapter 6: how to design memory-efficient and globally convergent stochastic approximation algorithms in RKHSs. By trailoring the memory compression of the statistical model representation during training to guarantee stochastic descent, we obtained exact convergence with probability 1. The performance of this approach in practice far surpassed methods considered in Parts I and II, and achieved comparable accuracy to batch nonparametric methods, with improved adaptability and scalability.

One limitation of the memory-efficient kernel methods in Chapter 6, and most nonparametric stochastic approximation algorithms, is the requisite that the choice of kernel is fixed during training, which means that the choice of the RKHS is set a priori. In practice, it is beneficial to train hyperparameters alongside statistical model parameters [182], but theoretically this has the effect of "warping" the function space [217]. In particular, this would amount to solving nonparametric stochastic optimization problems over a family of RKHSs rather than a single RKHS, and the determination of how to guarantee descent in a union/intersection of RKHSs while executing descent steps with respect to hyperparameters (kernel dictionary elements or Gaussian bandwidth, for instance) has not yet been established. However, we note that for special cases, there is a well-developed literature that connects the model order of a nonparametric regressor inversely with a Gaussian kernel bandwidth [56][Chapter 19]: larger bandwidth yields smaller model order. The generalization of this idea to problems of the form considered in Chapter 6, however, remains open. Experimentally, we have observed that incorporation of stochastic descent steps with respect to hyperparameters allows the learning of more flexible nonparametric regressors that attain better accuracy with fewer model points, suggesting the practical utility of this idea.

Additionally, as noted in Section 6.5, the inference accuracy attained by kernel methods has for the most part not met benchmarks set by neural networks [103]. One explanation that has been offered for the success of neural networks is that they analyze data smoothly at multiple scales before outputting estimates [121], an explanation that has been lent credibility by the recent success of multi-layer wavelet scattering [38] which has similar structural properties. Therefore, we believe the accuracy discrepancy between nonparametric methods and neural networks is a consequence of the fact that nonparametric regressors are single-layer maps from data to estimates. However, more complicated multi-layer composite kernels may be developed, based on the fact that a composition and positive linear combination of kernels is still a kernel [192, Ch. 11]. The scalable development of online nonparametric methods based on such composite kernels is not straight-forward, though; on the other hand, some initial off-line attempts in this direction have recently appeared [115, 120].

Multi-agent Efficient Online Kernel Learning The favorable trade-off between computational efficiency and statistical accuracy achieved by the method proposed in Chap-

ter 6 motivated us to develop a decentralized extension of sparsified online kernel learning based on penalty method that allows a network of interconnected autonomous agents, via their local data stream and message passing, to learn a close approximation to the globally optimal kernelized regressor that is memory-efficient. This approach yielded *state of the the art* performance for a multi-agent network attempting to solve online inference problems. We must emphasize, however, that this penalty method-based approach to multi-agent memory-efficient kernelized learning is not the only optimization tool that could be wielded to solve the problem formulated at the outset of Chapter 7. In principle, it is also possible to use sparse projections of primal-dual method or others which exploit Lagrange duality. To do so, however, first *the Representer Theorem must be extended to stochastic saddle point problems defined over RKHS*, since it does not yet apply to that setting. We believe such an extension is possible via use of functional extensions of the dual-regularized augmented Lagrangian proposed in Chapter 3. We further note that the method proposed in Chapter 7 only applies to the hypothesis where each agent is observing data from a common distribution, and hence consensus is the goal. This hypothesis, in general, may be violated, as noted in Chapter 3, in which case the proposed penalty method in Chapter 7 may be similarly used to address proximity constraints that incentivize nearby nodes estimates to be similar.

Value Function-Based Reinforcement Learning In Chapter 8, we shifted focus from statistical inference to stochastic control, or reinforcement learning, motivated by the fact that machine intelligence is more than just learning to make good predictions. More generally, we would like an autonomous agent to adapt its behavior over time to temporal incentives. To model this situation, we considered the formalism of Markov Decision Processes, in which every action an agent takes causes a random transition to a new state of the world and the collection of a random reward. The actual task of determining the choice of the optimal action sequence in a continuous state-action space setting has been deemed mathematically intractable due to fundamental complexity issues [152], but in Chapter 8, we began to repudiate this view by developing an efficient nonparametric stochastic method that nearly exactly determines the value of an action sequence when the action sequence follows a fixed stochastic stationary distribution called a policy. This setting, called policy evaluation, is a necessary precursor to *policy iteration* in which alternating policy evaluation and improvement steps are used to find the optimal policy. Moreover, the framework of nonparametric compositional stochastic optimization is sufficiently general as to possibly be used to solve Bellman's optimality equation, and thus find the *optimal* action-value function. We believe that this approach can further lay the foundation for stable collaborative multi-agent learning based control systems, building upon ideas proposed in Chapter 7.

Nonparametric Compositional Stochastic Optimization Underlying the afore-

mentioned proposed approaches to solving previously untenable problems in reinforcement learning (Chapter 8) are greedy projections of stochastic quasi-gradient algorithms in RKHS as solutions to nonparametric compositional stochastic optimization problems. Special cases of these problems have only recently been solved [206], and their memory-efficient extensions to RKHSs hold additional important implications for robust statistical estimation that could allow one to encode data volatility directly into an estimate [51, 71], a beneficial property in financial time series, econometrics, and robust formulations of model predictive control with uncertainty. In particular, this mathematical formulation would yield a problem whose solution would attain satisfactory inference accuracy in practice and do so with confidence.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning."

[2] M. Aharon, M. Elad, and A. Bruckstein, "k-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, 2006.

[3] I. F. Akyildiz and X. Wang, "A survey on wireless mesh networks," *IEEE Communications magazine*, vol. 43, no. 9, pp. S23–S30, 2005.

[4] A. Angelova, L. Matthies, D. Helmick, and P. Perona, "Learning and prediction of slip from visual information," *Journal of Field Robotics*, vol. 24, no. 3, pp. 205–231, 2007.

[5] A. Angelova, L. Matthies, D. Helmick, G. Sibley, and P. Perona, "Learning to predict slip for ground robots," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 3324–3331.

[6] K. M. Anstreicher, "On convex relaxations for quadratically constrained quadratic programming," *Mathematical programming*, vol. 136, no. 2, pp. 233–251, 2012.

[7] M. Anthony and P. L. Bartlett, *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

[8] L. Armijo, "Minimization of functions having lipschitz continuous first partial derivatives," *Pacific Journal of Mathematics*, vol. 16, no. 1, 1966.

[9] K. Arrow, L. Hurwicz, and H. Uzawa, *Studies in linear and non-linear programming*, ser. Stanford Mathematical Studies in the Social Sciences. Stanford University Press, Stanford, Dec. 1958, vol. II.

[10] K. J. Aström and R. M. Murray, *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2010.

[11] N. Ayanian, V. Kumar, and D. Koditschek, "Synthesis of controllers to create, maintain, and reconfigure robot formations with communication constraints," in *Robotics Research*. Springer, 2011, pp. 625–642.

[12] F. Bach, J. Mairal, and J. Ponce, "Task-driven dictionary learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 791–804, 2012.

[13] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, "Convex optimization with sparsity-inducing norms," in *Optimization for Machine Learning*. MIT Press, 2011.

[14] F. R. Bach, "Adaptivity of averaged stochastic gradient descent to local strong convexity for logistic regression." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 595–627, 2014.

[15] K. Bache and M. Lichman, "KDD Cup 1999 Data," 2013. [Online]. Available: http://archive.ics.uci.edu/ml/databases/kddcup99/kddcup99.html

[16] S. Bahrampour, N. Nasrabadi, A. Ray, and W. Jenkins, "Multimodal task-driven dictionary learning for image classification," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 24–38, 2016.

[17] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *In Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann, 1995, pp. 30–37.

[18] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 138–156, 2006.

[19] A. Beck, P. Stoica, and J. Li, "Exact and approximate solutions of source localization problems," *Signal Processing, IEEE Transactions on*, vol. 56, no. 5, pp. 1770–1778, May 2008.

[20] R. Bellman, *Dynamic Programming*, 1st ed. Princeton, NJ, USA: Princeton University Press, 1957. [Online]. Available: http://books.google.com/books?id=fyVtp3EMxasC&pg=PR5&dq=dynamic+programming+richard+e+bellman&client=firefox-a#v=onepage&q=dynamic%20programming%20richard%20e%20bellman&f=false

[21] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.

[22] D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar, *Convex Analysis and Optimization*. Athena Scientific, 2003.

[23] D. P. Bertsekas and S. E. Shreve, *Stochastic optimal control: The discrete time case*. Academic Press, 1978, vol. 23.

[24] S. Bhatnagar, D. Precup, D. Silver, R. S. Sutton, H. R. Maei, and C. Szepesvári, "Convergent temporal-difference learning with arbitrary smooth function approximation," in *Advances in Neural Information Processing Systems*, 2009, pp. 1204–1212.

[25] K. Bimbraw, "Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous

vehicle technology," in *Informatics in Control, Automation and Robotics (ICINCO), 2015 12th International Conference on*, vol. 1. IEEE, 2015, pp. 191–198.

[26] E. G. Birgin, J. . M. Martnez, and M. Raydan, "Nonmonotone spectral projected gradient methods on convex sets," *SIAM Journal on Optimization*, pp. 1196–1211, 2000.

[27] H. D. Block, "The perceptron: A model for brain functioning. i," *Rev. Mod. Phys.*, vol. 34, pp. 123–135, Jan. 1962.

[28] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Learnability and the vapnik-chervonenkis dimension," *Journal of the ACM (JACM)*, vol. 36, no. 4, pp. 929–965, 1989.

[29] A. Bordes, L. Bottou, and P. Gallinari, "Sgd-qn: Careful quasi-newton stochastic gradient descent," *The Journal of Machine Learning Research*, vol. 10, pp. 1737–1754, 2009.

[30] V. S. Borkar and S. P. Meyn, "The ode method for convergence of stochastic approximation and reinforcement learning," *SIAM Journal on Control and Optimization*, vol. 38, no. 2, pp. 447–469, 2000.

[31] L. Bottou, "Online algorithms and stochastic approximations," in *Online Learning and Neural Networks*, D. Saad, Ed. Cambridge, UK: Cambridge University Press, 1998.

[32] O. Bousquet and L. Bottou, "The tradeoffs of large scale learning," in *Advances in neural information processing systems*, 2008, pp. 161–168.

[33] S. Boyd and L. Vanderberghe, *Convex Programming*. New York, NY: Wiley, 2004.

[34] D. Bradley and Bagnell, "Differentiable Sparse Coding," in *Proceedings of Neural Information Processing Systems 22*, Dec. 2008.

[35] S. J. Bradtke and A. G. Barto, "Linear least-squares algorithms for temporal difference learning," *Machine learning*, vol. 22, no. 1-3, pp. 33–57, 1996.

[36] H. Brezis, *Functional analysis, Sobolev spaces and partial differential equations*. Springer Science & Business Media, 2010.

[37] P. Brodatz, *Textures: A Photographic Album for Artists and Designers*. Dover, 1966.

[38] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013.

[39] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, Jun. 1998.

[40] E. J. Candes, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, vol. 346, no. 9, pp. 589–592, 2008.

[41] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, Jul. 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1007379606734

[42] V. Cevher, S. Becker, and M. Schmidt, "Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics," *Signal Processing Magazine, IEEE*, vol. 31, no. 5, pp. 32–43, Sept 2014.

[43] P. Chainais and C. Richard, "Learning a common dictionary over a sensor network," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2013 IEEE 5th International Workshop on*. IEEE, 2013, pp. 133–136.

[44] C. Chang and C. Lin, "LIBSVM : A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, pp. 1–39, 2011.

[45] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4129–4144, 2014.

[46] K. Cheung, W.-K. Ma, and H. So, "Accurate approximation algorithm for toa-based maximum likelihood mobile location using semidefinite programming," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, vol. 2, May 2004, pp. ii–145–8 vol.2.

[47] F. R. K. Chung, *Spectral Graph Theory*. American Mathematical Society, 1997.

[48] B. Dai, N. He, Y. Pan, B. Boots, and L. Song, "Learning from conditional distributions via dual kernel embeddings," *arXiv preprint arXiv:1607.04579*, 2016.

[49] A. Defazio, F. Bach, and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Advances in Neural Information Processing Systems*, 2014, pp. 1646–1654.

[50] O. Dekel, S. Shalev-Shwartz, and Y. Singer, "The forgetron: A kernel-based perceptron on a fixed budget," in *Advances in Neural Information Processing Systems 18*. MIT Press, 2006, p. 259266. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=78226

[51] D. Dentcheva, S. Penev, and A. Ruszczyński, "Statistical estimation of composite risk functionals and risk optimization problems," *Annals of the Institute of Statistical Mathematics*, vol. 69, no. 4, pp. 737–760, Aug 2017.

[52] P. Di Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

[53] A. Dieuleveut, F. Bach *et al.*, "Nonparametric stochastic approximation with large step-sizes," *The Annals of Statistics*, vol. 44, no. 4, pp. 1363–1399, 2016.

[54] C. B. Do, Q. V. Le, and C.-S. Foo, "Proximal regularization for online and batch learning," in *Proc. 26th Int. Conf. Machine Learning*. New York: ACM, Jun. 14-18 2009, pp. 257–264.

[55] M. Dong, L. Tong, and B. M. Sadler., "Information retrieval and processing in sensor networks: deterministic scheduling vs. random access," in *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, June 2004, pp. 79–.

[56] B. Efron and T. Hastie, *Computer Age Statistical Inference*. Cambridge University Press, 2016, vol. 5.

[57] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, pp. 407–499, 2004.

[58] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *Trans. Img. Proc.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIP.2006.881969

[59] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, Aug 2004.

[60] ——, "Bayes meets bellman: The gaussian process approach to temporal difference learning," in *Proc. of the 20th International Conference on Machine Learning*, 2003.

[61] Y. Ermoliev, "Stochastic quasigradient methods and their application to system optimization," *Stochastics: An International Journal of Probability and Stochastic Processes*, vol. 9, no. 1-2, pp. 1–36, 1983.

[62] T. Evgeniou, M. Pontil, and T. Poggio, "Regularization networks and support vector machines," *Advances in computational mathematics*, vol. 13, no. 1, pp. 1–50, 2000.

[63] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Aug. 2008.

[64] A.-m. Farahmand, C. Ghavamzadeh, Mohammadand Szepesvári, and S. Mannor, "Regularized policy iteration with nonparametric function spaces," *Journal of Machine Learning Research*, vol. 17, no. 139, pp. 1–66, 2016. [Online]. Available: http://jmlr.org/papers/v17/13-016.html

[65] J. Fink and E. Stump, "Experimental analysis of models for trajectory generation on tracked vehicles," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, Sept 2014, pp. 1970–1977.

[66] R. A. Fisher, "On the mathematical foundations of theoretical statistics," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 222, pp. 309–368, 1922.

[67] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1663–1707, 2010.

[68] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Mach. Learn.*, vol. 37, no. 3, pp. 277–296, Dec. 1999.

[69] A. Ghosh and S. Sarkar, "Pricing for profit in internet of things," in *Information Theory (ISIT), 2015 IEEE International Symposium on.* IEEE, 2015, pp. 2211–2215.

[70] S. Grünewälder, G. Lever, L. Baldassarre, M. Pontil, and A. Gretton, "Modelling transition dynamics in mdps with rkhs embeddings," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, vol. 1, 2012, pp. 535–542.

[71] V. Guigues, V. Krätschmer, and A. Shapiro, "Statistical inference and hypotheses testing of risk averse stochastic programs," *arXiv preprint arXiv:1603.07384*, 2016.

[72] S. Haykin, "Neural networks: A comprehensive foundation," 1994.

[73] P. Honeine, "Online kernel principal component analysis: A reduced-order model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1814–1826, 2012.

[74] S. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The internet of things for health care: a comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.

[75] T. Jaakkola, M. I. Jordan, and S. P. Singh, "On the convergence of stochastic iterative dynamic programming algorithms," *Neural computation*, vol. 6, no. 6, pp. 1185–1201, 1994.

[76] A. Jadbabaie, J. Yu, and J. Hauser, "Unconstrained receding-horizon control of nonlinear systems," *Automatic Control, IEEE Transactions on*, vol. 46, no. 5, pp. 776–783, 2001.

[77] D. Jakovetic, J. M. F. Xavier, and J. M. F. Moura, "Fast distributed gradient methods," *CoRR*, vol. abs/1112.2972, Apr. 2011.

[78] F. Jakubiec and A. Ribeiro, "D-map: Distributed maximum a posteriori probability estimation of dynamic systems," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 450–466, Feb. 2013.

[79] R. Jenatton, J.-Y. Audibert, and F. Bach, "Structured variable selection with sparsity-inducing norms," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2777–2824, 2011.

[80] R. Jenatton, J. Huang, and C. Archambeau, "Online optimization and regret guarantees for non-additive long-term constraints," *arXiv preprint arXiv:1602.05394*, 2016.

[81] T. Joachims and C.-N. J. Yu, "Sparse kernel svms via cutting-plane training," *Machine Learning*, vol. 76, no. 2-3, pp. 179–193, 2009.

[82] B. Johansson, T. Keviczky, M. Johansson, and K. Johansson, "Subgradient methods and consensus algorithms for solving convex optimization problems," in *Proc. of the 47th IEEE Conference on Decision and Control*, Cancun, Mexico, 2008, pp. 4185–4190.

[83] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems*, 2013, pp. 315–323.

[84] I. Jolliffe, *Principal Component Analysis*. Springer Verlag, 1986.

[85] N. K. Jong and P. Stone, "Model-based function approximation in reinforcement learning," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM, 2007, p. 95.

[86] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[87] J. Kiefer, J. Wolfowitz *et al.*, "Stochastic estimation of the maximum of a regression function," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, 1952.

[88] G. Kimeldorf and G. Wahba, "Some results on tchebycheffian spline functions," *Journal of mathematical analysis and applications*, vol. 33, no. 1, pp. 82–95, 1971.

[89] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online Learning with Kernels," *IEEE Transactions on Signal Processing*, vol. 52, pp. 2165–2176, August 2004.

[90] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, p. 0278364913495721, 2013.

[91] V. R. Konda and J. N. Tsitsiklis, "Convergence rate of linear two-time-scale stochastic approximation," *Annals of applied probability*, pp. 796–819, 2004.

[92] A. Koppel, J. Fink, G. Warnell, E. Stump, and A. Ribeiro, "Online learning for characterizing unknown environments in ground robot vehicle modles," in *2016 IEEE International Conference in Intelligent Robots and Systems (IROS)*. IEEE, 2016.

[93] A. Koppel, F. Jakubiec, and A. Ribeiro, "A saddle point algorithm for networked online convex optimization," *IEEE Trans. Signal Process.*, p. 15, Oct 2015.

[94] A. Koppel, G. Warnell, and E. Stump, "Task-driven dictionary learning in distributed online settings," in *Asilomar Conf. on Signals, Systems, and Computers*. Pacific Grove, CA, November 8 - 11 2015.

[95] A. Koppel, G. Warnell, E. Stump, and A. Ribeiro, "D4l: Decentralized dynamic discriminative dictionary learning," *IEEE Trans. Signal and Info. Process. over Networks*, vol. (submitted), June 2016, available at http://www.seas.upenn.edu/ aribeiro/wiki.

[96] A. Koppel, S. Paternain, C. Richard, and A. Ribeiro, "Decentralized efficient nonparametric stochastic optimization," in *Signal and Information Processing (GlobalSIP), 2016 IEEE Global Conference on (submitted)*. IEEE, 2017.

[97] A. Koppel, B. Sadler, and A. Ribeiro, "Proximity without consensus in online multiagent optimization," *IEEE Transactions on Signal Processing*, 2017.

[98] A. Koppel, G. Warnell, E. Stump, and A. Ribeiro, "Parsimonious online learning with kernels via sparse projections in function space," *arXiv preprint arXiv:1612.04111*, 2016.

[99] ——, "Breaking bellman's curse of dimensionality: Efficient kernel gradient temporal difference," in *Advances in Neural Information Processing Systems (under review)*, 2017.

[100] A. Korostelev, "Stochastic recurrent procedures: Local properties," *Nauka: Moscow (in Russian)*, 1984.

[101] R. J. Kozick and B. M. Sadler, "Source localization with distributed sensor arrays and partial spatial coherence," *IEEE Transactions on Signal Processing*, vol. 52, no. 3, pp. 601–616, 2004.

[102] R. Kozick and B. Sadler, "Accuracy of source localization based on squared-range least squares (sr-ls) criterion," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2009 3rd IEEE International Workshop on*, Dec 2009, pp. 37–40.

[103] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[104] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, no. Dec, pp. 1107–1149, 2003.

[105] Y. Lecun and C. Cortes, "The MNIST database of handwritten digits." [Online]. Available: http://yann.lecun.com/exdb/mnist/

[106] B. R. Leffler, C. R. Mansley, and M. L. Littman, "Efficient learning of dynamics models using terrain classification," in *Proceedings of the 1st International Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems, ERLARS*, vol. 2008. Citeseer, 2008, pp. 41–46.

[107] T. Leung and J. Malik, "Representing and Recognizing the Visual Appearence of Materials using Three-dimensional Textons," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 29–44, 1999.

[108] G. Lever, J. Shawe-Taylor, R. Stafford, and C. Szepesvari, "Compressed conditional mean embeddings for model-based reinforcement learning," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[109] J.-B. Li, S.-C. Chu, and J.-S. Pan, *Kernel Learning Algorithms for Face Recognition*. Springer, 2014.

[110] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "Dlm: Decentralized linearized alternating direction method of multipliers," *IEEE Trans. Signal Process.*, Aug. 2014.

[111] J. Liu, Q. Chen, and H. D. Sherali, "Algorithm design for femtocell base station placement in commercial building environments," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2951–2955.

[112] W. Liu, P. P. Pokharel, and J. C. Principe, "The kernel least-mean-square algorithm," *Signal Processing, IEEE Transactions on*, vol. 56, no. 2, pp. 543–554, 2008.

[113] I. Lobel and A. Ozdaglar, "Distributed subgradient methods for convex optimization over random networks," *IEEE Trans. on Autom. Control,*, vol. 56, no. 6, pp. 1291–1306, Jun. 2011.

[114] M. Mahdavi, R. Jin, and T. Yang, "Trading regret for efficiency: online convex optimization with long term constraints," *Journal of Machine Learning Research*, vol. 13, no. Sep, pp. 2503–2528, 2012.

[115] J. Mairal, "End-to-end kernel learning with supervised convolutional kernel networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[116] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 4, pp. 791–804, 2012.

[117] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, Mar. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1756008

[118] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative learned dictionaries for local image analysis." in *CVPR*. IEEE Computer Society, 2008. [Online]. Available: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2008.html#MairalBPSZ08

[119] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," in *the IEEE Trans. on Image Processing*. ITIP, 2007, pp. 53–69.

[120] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 2627–2635.

[121] S. Mallat, "Understanding deep convolutional networks," *Phil. Trans. R. Soc. A*, vol. 374, no. 2065, p. 20150203, 2016.

[122] ——, *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*, 3rd ed. Academic Press, 2008.

[123] Z. Marinho, B. Boots, A. Dragan, A. Byravan, G. J. Gordon, and S. Srinivasa, "Functional gradient motion planning in reproducing kernel hilbert spaces," in *Proceedings of Robotics: Science and Systems*, Ann Arbor, MI, July 2016.

[124] M. A. Medina, "A pilot study: Exploring the acceptability, concerns, and preferences of older adults and family members regarding remotely controlled telemonitoring robots," Ph.D. dissertation, University of California, Davis, 2014.

[125] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, "An analysis of reinforcement learning with function approximation," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 664–671.

[126] C. A. Micchelli, Y. Xu, and H. Zhang, "Universal kernels," *Journal of Machine Learning Research*, vol. 7, no. Dec, pp. 2651–2667, 2006.

[127] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[128] A. Mokhtari and A. Ribeiro, "Res: Regularized stochastic bfgs algorithm," *Signal Processing, IEEE Transactions on*, vol. 62, no. 23, pp. 6089–6104, 2014.

[129] ——, "Global convergence of online limited memory bfgs," *Journal of Machine Learning Research*, vol. 16, pp. 3151–3181, 2015.

[130] J. J. Mor, "Generalizations of the trust region problem," *OPTIMIZATION METHODS AND SOFTWARE*, vol. 2, pp. 189–209, 1993.

[131] S. Mukherjee and S. K. Nayar, "Automatic generation of rbf networks using wavelets," *Pattern Recognition*, vol. 29, no. 8, pp. 1369–1383, 1996.

[132] K. Müller, T. Adali, K. Fukumizu, J. C. Principe, and S. Theodoridis, "Special issue on advances in kernel-based learning for signal processing [from the guest editors]," *IEEE Signal Process. Mag.*, vol. 30, no. 4, pp. 14–15, 2013. [Online]. Available: http://dx.doi.org/10.1109/MSP.2013.2253031

[133] K. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

[134] A. Nedić and A. Ozdaglar, "Subgradient methods for saddle-point problems," *J Optimiz. Theory App.*, vol. 142, no. 1, pp. 205–228, Aug. 2009.

[135] ——, "Approximate primal solutions and rate analysis for dual subgradient methods," *SIAM J. Optimiz.*, vol. 19, no. 4, pp. 1757–1780, Feb. 2009.

[136] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[137] D. Needell, J. Tropp, and R. Vershynin, "Greedy signal recovery review," in *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*. IEEE, 2008, pp. 1048–1050.

[138] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

[139] Y. Nesterov, "Introductory lectures on convex programming volume i: Basic course," 1998.

[140] ——, "Primal-dual subgradient methods for convex problems," *Math. Program.*, vol. 120, no. 1, pp. 221–259, Apr. 2009.

[141] X. Nguyen, M. J. Wainwright, and M. I. Jordan, "Nonparametric decentralized detection using kernel methods," *IEEE Transactions on Signal Processing*, vol. 53, no. 11, pp. 4053–4066, 2005.

[142] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive Processing*, vol. 12, no. 4, pp. 319–340, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10339-011-0404-1

[143] J. Nocedal, "Updating quasi-Newton matrices with limited storage," *Mathematics of Computation*, vol. 35, no. 151, pp. 773–773, 1980.

[144] V. Norkin and M. Keyzer, "On stochastic optimization and statistical learning in reproducing kernel hilbert spaces by support vector machines (svm)," *Informatica*, vol. 20, no. 2, pp. 273–292, 2009.

[145] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals &Amp; Systems (2Nd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[146] D. Ormoneit and Ś. Sen, "Kernel-based reinforcement learning," *Machine learning*, vol. 49, no. 2-3, pp. 161–178, 2002.

[147] C. Ostafew, A. Schoellig, and T. Barfoot, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 4029–4036.

[148] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition," in *Proceedings of the Asilomar Conference on Signals, Systems and Computers*, 1993.

[149] M. Pontil, "A note on different covering numbers in learning theory," *Journal of Complexity*, vol. 19, no. 5, pp. 665–671, 2003.

[150] M. Pontil, Y. Ying, and D. xuan Zhou, "Error analysis for online gradient descent algorithms in reproducing kernel hilbert spaces," Tech. Rep., 2005.

[151] C. E. Powell, "Numerical methods for generating gaussian random fields," in *The Porous Media-Processes and Mathematics (PMPM) Annual Meeting*, Oct 2014.

[152] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality.* John Wiley & Sons, 2007, vol. 703.

[153] W. B. Powell and J. Ma, "A review of stochastic algorithms with continuous value function approximation and some new approximate policy iteration algorithms for multidimensional continuous applications," *Journal of Control Theory and Applications*, vol. 9, no. 3, pp. 336–352, 2011.

[154] M. Rabbat, R. Nowak, and J. Bucklew, "Generalized consensus computation in networked systems with erasure links," in *IEEE 6th Workshop Signal Process. Adv. in Wireless Commun Process.*, Jun. 5-8 2005, pp. 1088–1092.

[155] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: ACM, 2007, pp. 759–766. [Online]. Available: http://doi.acm.org/10.1145/1273496.1273592

[156] H. Raja and W. U. Bajwa, "Cloud k-svd: A collaborative dictionary learning algorithm for big, distributed data," *IEEE Transactions on Signal Processing*, vol. 64, no. 1, pp. 173–188, Jan 2016.

[157] S. Ram, A. Nedić, and V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *J Optimiz. Theory App.*, vol. 147, no. 3, pp. 516–545, Sep. 2010.

[158] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in neural information processing systems*, 2011, pp. 693–701.

[159] A. Ribeiro, "Ergodic stochastic optimization algorithms for wireless communication and networking," *IEEE Transactions on Signal Processing*, vol. 58, no. 12, pp. 6369–6386, 2010.

[160] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1058–1067, 2009.

[161] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, 09 1951.

[162] F. Rogers-Marcovitz, N. Seegmiller, and A. Kelly, "Continuous vehicle slip model identification on changing terrains," in *RSS 2012 Workshop on Long-term Operation of Autonomous Robotic Systems in Changing Environments*, 2012.

[163] G. Ross, "The efficient use of function minimization in non-linear maximum-likelihood estimation," *Applied Statistics*, pp. 205–221, 1970.

[164] P. Roux, M. Corciulo, M. Campillo, and D. Dubuq, "Source localization analysis using seismic noise data acquired in exploration geophysics," *AGU Fall Meeting Abstracts*, p. C2249, Dec. 2011.

[165] G. Sampson, R. Haigh, and E. Atwell, "Natural language analysis by stochastic optimization: A progress report on project april," *J. Exp. Theor. Artif. Intell.*, vol. 1, no. 4, pp. 271–287, Oct. 1990. [Online]. Available: http://dx.doi.org/10.1080/09528138908953710

[166] A. Sayed, A. Tarighat, and N. Khajehnouri, "Network-based wireless location: challenges faced in developing techniques for accurate wireless location information," *Signal Processing Magazine, IEEE*, vol. 22, no. 4, pp. 24–40, July 2005.

[167] I. Schizas, A. Ribeiro, and G. Giannakis, "Consensus in ad hoc wsns with noisy links - part i: distributed estimation of deterministic signals," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 350–364, Jan. 2008.

[168] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, pp. 1–30.

[169] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," *Subseries of Lecture Notes in Computer Science Edited by JG Carbonell and J. Siekmann*, p. 416.

[170] N. N. Schraudolph, J. Yu, and S. Günter, "A stochastic quasi-newton method for online convex optimization," in *International Conference on Artificial Intelligence and Statistics*, 2007, pp. 436–443.

[171] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 1889–1897.

[172] M. Schwager, P. Dames, D. Rus, and V. Kumar, "A multi-robot control policy for information gathering in the presence of unknown hazards," in *Robotics Research*. Springer, 2017, pp. 455–472.

[173] W. R. Scott, W. B. Powell, and S. Moazehi, "Least squares policy iteration with instrumental variables vs. direct policy search: Comparison against optimal benchmarks using energy storage," *arXiv preprint arXiv:1401.0843*, 2014.

[174] N. Seegmiller, F. Rogers-Marcovitz, G. Miller, and A. Kelly, "Vehicle model identification by integrated prediction error minimization," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 912–931, 2013.

[175] N. Seegmiller, F. Rogers-Marcovitz, G. A. Miller, and A. Kelly , "A unified perturbative dynamics approach to online vehicle model identification," in *International Symposium on Robotics Research*, August 2011.

[176] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, Feb. 2012.

[177] S. Shalev-Shwartz and Y. Singer, "Logarithmic regret algorithms for strongly convex repeated games," The Hebrew University, Jerusalem, Israel, Tech. Rep., 2007.

[178] S. Shalev-Shwartz, O. Shamir, N. Srebro, and K. Sridharan, "Learnability, stability and uniform convergence," *Journal of Machine Learning Research*, vol. 11, no. Oct, pp. 2635–2670, 2010.

[179] A. Shapiro, D. Dentcheva *et al.*, *Lectures on stochastic programming: modeling and theory*. Siam, 2014, vol. 16.

[180] P. Shi and C.-L. Tsai, "Regression model selection-a residual likelihood approach," *J. Roy. Statist. Soc. Ser. B*, vol. 64, no. 2, pp. 237–252, May 2002.

[181] K. Slavakis, P. Bouboulis, and S. Theodoridis, "Online learning in reproducing kernel hilbert spaces," *Signal Processing Theory and Machine Learning*, pp. 883–987, 2013.

[182] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," in *Proceedings of the 18th International Conference on Neural Information Processing Systems*. MIT Press, 2005, pp. 1257–1264.

[183] V. Solo and X. Kong, *Adaptive Signal Processing Algorithms: Stability and Performance*, ser. Prentice-Hall information and system sciences series. Prentice Hall, 1995. [Online]. Available: https://books.google.com/books?id=3AkfAQAAIAAJ

[184] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[185] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[186] R. S. Sutton, H. R. Maei, and C. Szepesvári, "A convergent $o(n)$ temporal-difference algorithm for off-policy learning with linear function approximation," in *Advances in neural information processing systems*, 2009, pp. 1609–1616.

[187] T. Suzuki, "Dual averaging and proximal gradient descent for online alternating direction multiplier method," in *Proc. 30th Int. Conf. Machine Learning*, vol. 28, no. 1, Atlanta, GA, USA, Jun. 16-21 2013, pp. 392–400.

[188] M. Taşan, G. Musso, T. Hao, M. Vidal, C. A. MacRae, and F. P. Roth, "selecting causal genes from genome-wide association studies via functionally coherent subnetworks," *Nature methods*, 2014.

[189] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *IEEE Proc. Int. Sym. on Comp. Intelligence for Sec. and Def. App.*, Ottawa, Ontario, Canada, Jul. 8-10 2009, pp. 53–58.

[190] G. Taylor and R. Parr, "Kernelized value function approximation for reinforcement learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 1017–1024.

[191] A. Tewari and P. L. Bartlett, "Learning theory," in *Academic Press Library in Signal Processing: Volume 1 Signal Processing Theory and Machine Learning*, ser. Academic Press Library in Signal Processing, P. S. Diniz, J. A. Suykens, R. Chellappa, and S. Theodoridis, Eds. Elsevier, 2014, vol. 1, ch. 14, pp. 775–816. [Online]. Available: http://dx.doi.org/10.1016/B978-0-12-396502-8.00014-0

[192] S. Theodoridis, *Machine learning: a Bayesian and optimization perspective*. Academic Press, 2015.

[193] S. Theodoridis, K. Slavakis, and I. Yamada, "Adaptive learning in a world of projections," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 97–123, 2011.

[194] Z. J. Towfic and A. H. Sayed, "Stability and performance limits of adaptive primal-dual networks," *IEEE Transactions on Signal Processing*, vol. 63, no. 11, pp. 2888–2903.

[195] ——, "Adaptive penalty-based distributed stochastic convex optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 15, pp. 3924–3938, 2014.

[196] P. Tseng and C. O. L. Mangasarian, "Convergence of a block coordinate descent method for nondifferentiable minimization," *J. Optim Theory Appl*, pp. 475–494, 2001.

[197] K. I. Tsianos and M. G. Rabbat, "Distributed strongly convex optimization," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*. IEEE, 2012, pp. 593–600.

[198] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. 31, no. 9, pp. 803–812, 1986.

[199] J. N. Tsitsiklis, "Asynchronous stochastic approximation and q-learning," *Machine Learning*, vol. 16, no. 3, pp. 185–202, 1994.

[200] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE transactions on automatic control*, vol. 42, no. 5, pp. 674–690, 1997.

[201] J. W. Tukey, "The future of data analysis," *Annals of Mathematical Statistics*, vol. 33, no. 1, pp. 1–67, March 1962. [Online]. Available: http://projecteuclid.org/DPubS/Repository/1.0/Disseminate?handle=euclid.aoms/1177704711&view=body&content-type=pdf_1

[202] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[203] P. Vincent and Y. Bengio, "Kernel matching pursuit," *Machine Learning*, vol. 48, no. 1, pp. 165–187, 2002.

[204] D. D. Wackerly, W. M. III, and R. L. Scheaffer, *Mathematical Statistics with Applications*, sixth edition ed. Duxbury Advanced Series, 2002.

[205] M. Wang and D. P. Bertsekas, "Incremental constraint projection-proximal methods for nonsmooth convex optimization," *SIAM Journal on Optimization (to appear)*, 2014.

[206] M. Wang, E. X. Fang, and H. Liu, "Stochastic compositional gradient descent: Algorithms for minimizing compositions of expected-value functions," *Mathematical Programming*, vol. 161, no. 1-2, pp. 419–449, 2017.

[207] Z. Wang, K. Crammer, and S. Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 3103–3131, 2012.

[208] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998.

[209] E. Wei and A. Ozdaglar, "Distributed alternating direction method of multipliers," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 5445–5450.

[210] M. N. Wernick, Y. Yang, J. G. Brankov, G. Yourganov, and S. C. Strother, "Machine learning in medical imaging," *IEEE signal processing magazine*, vol. 27, no. 4, pp. 25–38, 2010.

[211] D. B. West, *Introduction to Graph Theory*, 2nd ed. Prentice Hall, Sept. 2000.

[212] R. Wheeden, R. Wheeden, and A. Zygmund, *Measure and Integral: An Introduction to Real Analysis*, ser. Chapman & Hall/CRC Pure and Applied Mathematics. Taylor & Francis, 1977. [Online]. Available: https://books.google.com/books?id=YDkDmQ_hdmcC

[213] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2008.79

[214] X. Xu, T. Xie, D. Hu, and X. Lu, "Kernel least-squares temporal difference learning," *International Journal of Information Technology*, vol. 11, no. 9, pp. 54–63, 2005.

[215] Y. Xu and W. Yin, "Block stochastic gradient iteration for convex and nonconvex optimization," *SIAM Journal on Optimization*, vol. 25, no. 3, pp. 1686–1716, 2015.

[216] ——, "A globally convergent algorithm for nonconvex optimization based on block coordinate update," *Journal of Scientific Computing*, pp. 1–35, 2017. [Online]. Available: http://dx.doi.org/10.1007/s10915-017-0376-0

[217] Y. Xu and H. Zhang, "Refinement of reproducing kernels," *Journal of Machine Learning Research*, vol. 10, no. Jan, pp. 107–140, 2009.

[218] B. Yamauchi, "Packbot: A versatile platform for military robotics," in *SPIE*, 2004, pp. 228—-237.

[219] F. Yan, S. Sundaram, S. V. N. Vishwanathan, and Y. Qi, "Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties," *IEEE Trans. on Knowl. and Data Eng.*, vol. 25, no. 11, pp. 2483–2493, Nov. 2013.

[220] F. Yan, S. V. N. Vishwanathan, and Y. Qi, "Cooperative autonomous online learning," *CoRR*, vol. abs/1006.4039, Sep. 2010.

[221] Y. Ying and D. X. Zhou, "Online regularized classification algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 4775–4788, Nov 2006.

[222] W. Yu, O. Chuy, J. Collins, E.G., and P. Hollis, "Analysis and experimental verification for dynamic modeling of a skid-steered wheeled vehicle," *Robotics, IEEE Transactions on*, vol. 26, no. 2, pp. 340–353, April 2010.

[223] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *ArXiv e-prints 1310.7063*, Oct. 2013.

[224] M. Zargham, A. Ribeiro, A. Jadbabaie, and A. Ozdaglar, "Accelerated dual descent for network optimization," *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 905 – 920, Apr. 2014.

[225] L. Zhang, J. Yi, R. Jin, M. Lin, and X. He, "Online kernel learning with a near optimal sparsity bound." in *ICML (3)*, 2013, pp. 621–629.

[226] D.-X. Zhou, "The covering number in learning theory," *Journal of Complexity*, vol. 18, no. 3, pp. 739–767, 2002.

[227] J. Zhu and T. Hastie, "Kernel Logistic Regression and the Import Vector Machine," *Journal of Computational and Graphical Statistics*, vol. 14, no. 1, pp. 185–205, 2005.

[228] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. 20th Int. Conf. on Machine Learning*, vol. 20, no. 2, Washington DC, USA, Aug. 21-24 2003, pp. 928–936.