Wasserstein-Splitting Gaussian Process Regression for Heterogeneous Online Bayesian Inference

Michael E. Kepler[1], Alec Koppel[2], Amrit Singh Bedi[2], and Daniel J. Stilwell[1]

Abstract— **Gaussian processes (GPs) are a canonical nonparametric Bayesian inference technique, but they suffer from scalability problems for large sample sizes, and their performance breaks down for non-stationary or spatially heterogeneous data. In this work, we seek to overcome these issues through (i) employing variational free energy approximations of GPs operating in tandem with online expectation propagation steps; and (ii) introducing a local splitting step which instantiates a new GP whenever the posterior distribution changes significantly as quantified by the Wasserstein metric over posterior distributions. Over time, then, this yields an ensemble of sparse GPs which may be updated incrementally, and adapts to locality, heterogeneity, and non-stationarity in training data. We provide a 1-dimensional example to illustrate the motivation behind our approach, and compare the performance of our approach to other Gaussian process methods across various data sets, which often achieves competitive, if not superior predictive performance, relative to other locality-based GP regression methods in which hyperparameters are learned in an online manner.**

I. Introduction

Gaussian Processes (GPs) are a nonparametric Bayesian inference technique that has been widely used across science and engineering to provide nonlinear interpolation and uncertainty estimates [1], as in terrain elevation [2], temperature [3], the inverse mapping of sensor outputs to joint inputs for the control of a robotic arm [4], [5], systems identification [6], [7] and in uncertainty-aware mapping [8], [9]. In its most essential form, a posterior distribution is fit to an unknown nonlinear map $f : \mathscr{X} \to \mathbb{R}$ upon the basis of feature vectors $\mathbf{x}_k \in \mathscr{X} \subset \mathbb{R}^d$ and noisy observations $y_k = f(\mathbf{x}_k) + \varepsilon_k$ where ε_k is observation noise, typically Gaussian. The descriptive power of this framework is hamstrung by two important attributes of the GP posterior: (i) computing the conditional mean and covariance require computational effort cubic $\mathscr{O}(N^3)$ in the sample size N due to the presence of a kernel matrix inversion; and (ii) consistency guarantees require data to be independent and identically distributed (i.i.d) [10]. In robotics, data is typically arriving incrementally and exhibits locality or drift [11]. Augmenting GPs to address these issues is the goal of this work.

Challenge (i), the poor scaling with sample size N, originates from the fact that the posterior mean and covariance depend on a data matrix, or kernel dictionary, that accumulates all past observations. A long history of works have studied this issue, and typically select a subset of $M \ll N$ possible model points called inducing [12] or pseudo-inputs [13], and optimize GP hyperparameters along this subspace. One may select points via information gain [14], greedy compression [15], Nyström sampling [16], or other probabilistic criteria [17], [18]. A related question is the objective for inducing input optimization, which may be done according to the posterior likelihood along the M-dimensional subspace [13], or a KL-divergence lower bound thereof called the variational free energy (VFE) [12]. These methods are predominately offline, i.e., $N < \infty$ is a fixed finite number.

To augment these approaches to apply to settings with incrementally arriving data, dynamic point selection schemes have been proposed [19]; however, it is difficult to incorporate hyperparameter search, especially for inducing inputs, into these schemes, which are essential to obtaining competitive performance in practice. By contrast, fixed-memory VFE approximations may gracefully incorporate online Expectation Propagation (EP) steps [20], as pointed out in [21], and may be subsumed into the streaming variational Bayes framework [22]. See [7], [23] for a review of recent advances in Variational Bayes as it pertains to GPs. For this reason, to mitigate the sample complexity bottleneck, we adopt a VFE approach with online EP steps [21], whose approximate consistency (for i.i.d. settings) is recently established [24].

Attempts to address issue (ii), i.e., to broaden use of GPs to non-stationary or spatially heterogeneous data, have taken inspiration from vector-valued time-series analysis as well as ensemble methods. Specifically, time-series approaches seek to define covariance kernels that vary with time and space [25]–[27]. These approaches have also given rise to use of GPs in high-dimensional problems through use of convolutional kernels [28]. However, doing so typically requires offline training, and numerical conditioning issues make it challenging to adapt them to the online setting, which is inherent to continually learning autonomous robots.

Our approach is, instead, inspired by ensemble methods, where one builds a family of "weak learners" and forms the inference by appropriately weighting the constituents [29], [30], as in boosting [31], and local regression, i.e., the Nadaraya-Watson estimator [32]–[34]. Two design decisions are crucial to formulating an ensemble GP: (a) how to form the aggregate estimate, and (b) when to instantiate a new model. The line of research related to ensemble GPs began with local Gaussian Process regression (LGPR) [4], which (a) formulates estimates by weighted majority voting akin to Nadaraya-Watson, and (b) assigns samples to a local model based on a similarity measure defined in terms of the covariance kernel to the geometric mean of a local model's samples. If no local models are sufficiently close,

[1] Virginia Tech; mkepler@vt.edu, stilwell@vt.edu
[2] U.S. ARL; alec.e.koppel.civ@mail.mil, amrit0714@gmail.com

then a new local model is instantiated. We note that this approach requires hyperparameters to be fixed *in advance* of training. Most similar to this work is sparse *online* locally adaptive regression using Gaussian processes (SOLAR GP) [35], where each local GP is approximated using VFE and updated online using EP [21].

The aforementioned approaches to ensemble GPs have two key limitations [4], [35], [36]: (a) they form aggregate estimates by weighted voting, and (b) use distance in feature space to quantify whether a new constituent model is needed. Weighted voting, as inspired by local regression, hinges upon the validity of an i.i.d. hypothesis [32], [33], and distance in feature space may be confuse far away samples that are similar in distribution to a current local model. To address these issues, we propose Wasserstein-splitting Gaussian Process Regression (WGPR), whose main merits are as follows:

- the notion of similarity is specified as the Wasserstein distance, a rigorous metric over distributions which is available in closed-form for Gaussian Processes. This distance is used to quantify the change in the each model's predictive distribution caused by an update with the new training data, and hence defines a decision rule for when to instantiate a new local model: a new local model is instantiated only if the change in the posterior predictive model exceeds a threshold in the Wasserstein distance for every model, otherwise it is incorporated into the most similar, i.e. smallest Wasserstein distance, model via expectation propagation (EP) steps;
- WGPR forms predictive distributions upon the basis of only the the nearest local model in terms of Euclidean distance in the input space, rather than a weighted vote, in order to more strictly encapsulate local heterogeneity in the data, and hence only operates upon a *local* i.i.d. hypotheses;
- experimentally WGPR identifies the number of data phase transitions on a 1-dimensional data set and achieves competitive performance with the state of the art for a variety of real data sets.

II. Background and Notation

In WGPR, we employ a sparse approximation of the posterior predictive distribution for each model. In this section, we introduce notation and provide the background material on sparse Gaussian processes required to implement WGPR.

A. Gaussian Process Regression

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution [37]. We use a Gaussian process to model the spatial field of interest $f : \mathscr{X} \to \mathbb{R}$, which we write as $f(\cdot) \sim GP(m(\cdot), k(\cdot,\cdot))$, where $m : \mathscr{X} \to \mathbb{R}$ and $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ denote the mean and covariance functions, respectively. Practical application involves evaluating a Gaussian process at a finite set of inputs $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\} \subset \mathscr{X}$, and with a slight abuse of notation we write $f(\mathbf{X}) \sim \mathscr{N}(m(\mathbf{X}), k(\mathbf{X},\mathbf{X}))$ where the mean vector $m(\mathbf{X}) \in \mathbb{R}^N$ and covariance matrix $k(\mathbf{X},\mathbf{X}) \in \mathbb{R}^{N\times N}$ are defined element-wise via $[m(\mathbf{X})]_i = m(\mathbf{x}_i)$ and $[k(\mathbf{X},\mathbf{X})]_{i,j} = k(\mathbf{x}_i,\mathbf{x}_j)$. More generally, for $\mathbf{Z} = \{\mathbf{z}_1, \cdots, \mathbf{z}_M\} \subset \mathscr{X}$, the matrix $k(\mathbf{X},\mathbf{Z}) \in \mathbb{R}^{N\times M}$ is defined by $[k(\mathbf{X},\mathbf{Z})]_{i,j} = k(\mathbf{x}_i,\mathbf{z}_j)$ for all $i = 1 : N$, and all $j = 1 : M$.

Gaussian process regression follows a Bayesian approach which begins with specifying a prior distribution $f(\cdot) \sim GP_0(m_0(\cdot), k_0(\cdot,\cdot))$. As is customarily done for the sake notational simplicity [37, Ch. 2.2], we select the zero-mean function as the prior mean function $m_0(\cdot)$. The covariance kernel $k_0(\cdot,\cdot)$ is user-specified and encodes prior assumptions, such as smoothness, about the underlying function f. Throughout this paper, we employ the commonly-used squared exponential covariance kernel given by

$$k_0(\mathbf{x},\mathbf{x}') = \sigma_f^2 \exp\{-(1/2)(\mathbf{x}-\mathbf{x}')^\top \Lambda^{-1}(\mathbf{x}-\mathbf{x}')\}, \qquad (1)$$

where σ_f^2 denotes the signal variance, and $\Lambda = \operatorname{diag}(\lambda_1^2, ..., \lambda_d^2)$ contains the length-scale parameter λ_i of each input dimension $i = 1, ..., d$. The entire conceptual development proceeds analogous for other covariance kernels. The assumption that each measurement is corrupted by independent identically distributed Gaussian noise $\varepsilon \sim N(0, \sigma_n^2)$ gives rise to a Gaussian likelihood, and given a collection of training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ the posterior process is

$$f(\mathbf{x}) \sim GP(m_{post}(\mathbf{x}), k_{post}(\mathbf{x},\mathbf{x}')) \qquad (2)$$

where

$$m_{post}(\mathbf{x}) = k_0(\mathbf{x},\mathbf{X})(k_0(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I})^{-1}\mathbf{y},$$
$$k_{post}(\mathbf{x},\mathbf{x}') = k_0(\mathbf{x},\mathbf{x}') + k_0(\mathbf{x},\mathbf{X})(k_0(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I})^{-1}k_0(\mathbf{X},\mathbf{x}'),$$

for all $\mathbf{x},\mathbf{x}' \in \mathscr{X}$. Performing inference with the exact form of the posterior (2), is only feasible for small to moderate data sets, e.g. sample size $N < 10{,}000$, as the necessary matrix inversion in the preceding expression for the posterior mean and covariance functions incurs a computational cost that scales $\mathscr{O}(N^3)$. To mitigate the complexity bottleneck in the sample size, we employ a sparse approximation of the posterior distribution for each Gaussian process parameterization henceforth considered. Next we define the specific sparse approximation scheme we consider.

B. Variational Free Energy Approximation

We reduce computational and memory costs using the posterior approximation developed by Titsias [12]. The key idea underpinning the approximation is to express the posterior in terms of $M \ll N$ pseudo (inducing) points, which are as close to possible as the full posterior likelihood associated with N training points. With M fixed (by the user), the pseudo points $\mathbf{Z} = \{\mathbf{z}_1, ..., \mathbf{z}_M\} \subset \mathscr{X}$, specified as variational parameters, are selected so as to maximize a lower bound on the true log marginal likelihood. This bound is referred to as the variational free energy and is given by

$$F_{VFE} = \log[\mathscr{N}(\mathbf{y}; \mathbf{0}, \sigma_n^2\mathbf{I} + \mathbf{Q}_{NN})] - \frac{1}{2\sigma_n^2}\operatorname{tr}(k_0(\mathbf{X},\mathbf{X}) - \mathbf{Q}_{NN}), \qquad (3)$$

where $\mathbf{Q}_{NN} = k_0(\mathbf{X},\mathbf{Z})k_0(\mathbf{Z},\mathbf{Z})^{-1}k_0(\mathbf{Z},\mathbf{X})$. Equivalently, by maximizing the variational free energy, the Kullback-Leibler divergence between the variational distribution and exact posterior distribution over $f(\mathbf{Z})$ is minimized. Given the pseudo inputs $\mathbf{Z}$, the optimal variational distribution over $f(\mathbf{Z}) \triangleq \mathbf{f}_Z$ has an analytic form $\mathbf{f}_Z \sim N(\mu_Z, \mathbf{S}_Z)$ with

$$\mu_Z = \sigma_n^{-2} k_0(\mathbf{Z},\mathbf{Z})\Sigma k_0(\mathbf{Z},\mathbf{X})\mathbf{y},$$
$$\mathbf{S}_Z = k_0(\mathbf{Z},\mathbf{Z}) \underbrace{\left(k_0(\mathbf{Z},\mathbf{z}) + \sigma_n^{-2}k_0(\mathbf{Z},\mathbf{X})k_0(\mathbf{X},\mathbf{Z})\right)^{-1}}_{\triangleq\Sigma} k_0(\mathbf{Z},\mathbf{Z}).$$

Given the optimal pseudo-inputs $\mathbf{Z} \subset \mathscr{X}$, the approximate posterior is given by $f(\cdot) \sim GP(m_{VFE}(\cdot), k_{VFE}(\cdot,\cdot))$, where

$$\begin{aligned} m_{VFE}(\mathbf{x}) &= k_0(\mathbf{x},\mathbf{Z})k_0(\mathbf{Z},\mathbf{Z})^{-1}\mu_Z, \qquad (4)\\ k_{VFE}(\mathbf{x},\mathbf{x}') &= k_0(\mathbf{x},\mathbf{x}') - k_0(\mathbf{x},\mathbf{Z})k_0(\mathbf{Z},\mathbf{Z})^{-1}k_0(\mathbf{Z},\mathbf{x}') \\ &\quad + k_0(\mathbf{x},\mathbf{Z})k_0(\mathbf{Z},\mathbf{Z})^{-1}\mathbf{S}_Z k_0(\mathbf{Z},\mathbf{Z})^{-1}k_0(\mathbf{Z},\mathbf{x}'). \end{aligned}$$

In practice, we first obtain the pseudo-inputs $\mathbf{Z} \subset \mathscr{X}$ and hyperparameters $\theta = \{\sigma_f, \Lambda, \sigma_n\}$ via gradient-based minimization of the negative variational free energy (3). With the (locally optimal)[1] hyperparameters and pseudo-inputs in hand, we then form the posterior predictive distribution (4).

As new training data is collected, we use the principled *expectation propagation* framework of Bui et al. [21] to recursively update the existing sparse approximation, along with the hyperparameters and pseudo points. For the recursive update, it is assumed that previous measurements are inaccessible so knowledge about previously collected measurements is inferred through existing pseudo points. More precisely, let GP_{old} denote the current sparse posterior approximation with associated pseudo inputs $\mathbf{Z}_a \subset \mathscr{X}$ and hyperparameters θ_{old}. The updated posterior approximation follows from an optimization problem, where given the new data $\mathscr{D}_{new} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{new}}$ and $\mathbf{Z}_a \subset \mathscr{X}$ we maximize the online log marginal likelihood F_{OVFE}. The optimization gives rise to the updated posterior predictive distribution

$$GP_{upd}(m_{upd}(\cdot), k_{upd}(\cdot,\cdot)), \qquad (5)$$

which depends on the new pseudo inputs $\mathbf{Z}_b$ and hyperparameters θ_{new} that jointly optimize F_{OVFE}. We omit the exact expressions for F_{OVFE} and the posterior mean and covariance as they are prohibitively long, but they can be readily accessed from the supplementary appendix of [21]. In summary, the recursive update is performed by first obtaining the pseudo-inputs $\mathbf{Z}_b$ and hyperparameters θ_{new} that optimize F_{OVFE}, and then using them to form the posterior (5).

III. Wasserstein-Splitting Gaussian Processes

In this section, we present Wasserstein-Splitting Gaussian Process Regression (WGPR). The high-level structure of WGPR mirrors that of LGPR [4] and SOLAR GP [35], however our method deviates from the existing methods in two notable ways: (i) we instantiate new models on the basis of the their posterior predictive distribution, rather than their locality in the input space, and (ii) we propose an alternative approach for making predictions.

A. Training

Initialization: Training the overall prediction model begins with an empty collection of models, and given the initial batch of training data $\mathscr{D}_{new} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{new}}$ of N_{new} samples, we instantiate the first local model using the VFE approximation as detailed in Section II-B yielding GP_1.

Updating the Current Ensemble: Now, suppose we have a collection $\{GP_j\}_{j=1}^J$ of J Gaussian Process models. The key question is upon obtaining a new batch of training data $\mathscr{D}_{new}$, how to decide which model is most similar, and thus is a suitable candidate for executing an update. Previous approaches [4], [34], [35] define similarity in terms of the covariance kernel [cf. (1)]. With this definition of similarity, two distant regions of the spatial field, characterized by the same set of underlying hyperparameters, gives rise to two different models. There are indeed apparent structural similarities to local kernel regression estimates of conditional expectations such as the Nadaraya-Watson estimator [32], [33] or ensemble techniques [31] inspired by majority voting [29], [30]. However, the former approach hinges upon training examples are i.i.d. from a stationary distribution, and the latter collapses distributional inference to point estimates in feature space, which belies the fact that we are in the Bayesian setting.

We, instead, base similarity on the intuition that if $\mathscr{D}_{new}$ is similar to an existing model, then the new data should minimally change the posterior predictive distribution according to some metric over probability distributions. In other words, predictions of the updated individual model (i) remain (approximately) unchanged across previously seen data, and (ii) agree with the predictions of a newly instantiated model across the new measurements. To formalize this intuition and assess how the new data would affect each posterior distribution in the collection $\{GP_j(m_j(\cdot), k_j(\cdot,\cdot))\}_{j=1}^J$, we update each model using the recursive VFE framework [21] using the new mini-batch $\mathscr{D}_{new}$, as outlined in Section II, which yields the updated collection $\{GP_j^{upd}\}_{j=1}^J$ where

$$GP_j^{upd}\left(m_j^{upd}(\cdot), k_j^{upd}(\cdot,\cdot)\right), \quad j = 1, \ldots, J.$$

Let $\mathbf{Z}_j \subset \mathscr{X}$ denote the pseudo-inputs of model j, prior to performing the update with $\mathscr{D}_{new}$. We then measure the change in predictions over previous data by computing the squared Wasserstein distance [38] between the original and updated posterior evaluated at $\mathbf{Z}_j$, i.e.

$$w_j^{old} = \|\mathbf{m}_j - \mathbf{m}_{j,Z}^{upd}\|^2 + \mathrm{tr}\left(\mathbf{S}_j + \mathbf{S}_{j,Z}^{upd} - 2\left(\mathbf{S}_j^{1/2}\mathbf{S}_{j,Z}^{upd}\mathbf{S}_j^{1/2}\right)^{1/2}\right),$$

where $\|\cdot\|$ denotes the Euclidean norm, and

$$\begin{aligned} \mathbf{m}_j &\triangleq m_j(\mathbf{Z}_j), & \mathbf{S}_j &\triangleq k_j(\mathbf{Z}_j, \mathbf{Z}_j) \\ \mathbf{m}_{j,Z}^{upd} &\triangleq m_j^{upd}(\mathbf{Z}_j), & \mathbf{S}_{j,Z}^{upd} &\triangleq k_j^{upd}(\mathbf{Z}_j, \mathbf{Z}_j). \end{aligned}$$

[1]Observe that in general, since the VFE is non-convex with respect to its hyperparameters θ and pseudo-inputs $\mathbf{Z}$, the best pointwise limit one may hope to achieve via gradient-based search is a local minimizer.

Note that the Wasserstein distance is a valid metric over probability measures, and its salient feature is that it is computable in closed-form for Gaussians, as well as it avoids some of the computational issues of comparable choices such as the Total Variation or Hellinger metrics – see [39].

Now, with $\{GP_j^{upd}\}_{j=1}^{J}$ in hand, we quantify how well each updated model agrees with the new data. To do so, we take $\mathscr{D}_{new}$ and instantiate a new approximate GP via the batch variational free energy sparse approximation (3) - (4):

$$GP_{J+1}\left(m_{J+1}(\cdot), k_{J+1}(\cdot,\cdot)\right)$$

that is free from bias of any previous data. Ideally, if model j is similar to the new data, then the distributions GP_j^{upd} and GP_{J+1} should agree well over the new data. Let $\mathbf{X}_{new} \subset \mathscr{X}$ denote the locations of the new measurements. We measure the agreement by computing the squared Wasserstein distance between the newly instantiated and updated posterior evaluated at $\mathbf{X}_{new}$, given by

$$w_j^{new} = \|\mathbf{m}_{J+1} - \mathbf{m}_{j,X}^{upd}\|^2 + \mathrm{tr}\left(\mathbf{S}_{J+1} + \mathbf{S}_{j,X}^{upd} - 2\left(\mathbf{S}_{J+1}^{1/2}\mathbf{S}_{j,X}^{upd}\mathbf{S}_{J+1}^{1/2}\right)^{1/2}\right)$$

where

$$\mathbf{m}_{J+1} \triangleq m_{J+1}(\mathbf{X}_{new}), \qquad \mathbf{S}_{J+1} \triangleq k_{J+1}(\mathbf{X}_{new}, \mathbf{X}_{new})$$
$$\mathbf{m}_{j,X}^{upd} \triangleq m_j^{upd}(\mathbf{X}_{new}), \qquad \mathbf{S}_{j,X}^{upd} \triangleq k_j^{upd}(\mathbf{X}_{new}, \mathbf{X}_{new}).$$

Thus, the net measure of similarity w_j is the aggregation of the squared-Wasserstein distance between the previous model GP_j and its updated variant GP_j^{upd} evaluated at the old pseudo-inputs $\mathbf{Z}_j$, which serve as a proxy for previously seen data, and the squared distance between GP_j^{upd} and the new model GP_{J+1} evaluated at the new measurement locations $\mathbf{X}_{new}$:

$$w_j = w_j^{new} + w_j^{old}, \qquad j = 1, \ldots, J. \tag{6}$$

Once we compute w_j for all j, we define the most similar model GP_{j^*} as the one that minimizes (6) , i.e., $j^* = \arg\min_{j=1,\ldots,J} w_j$ and provided it is sufficiently similar, meaning w_{j^*} is less than a user-defined threshold ε, we retain the updated model GP_{j*}^{upd}. All other models revert back to their previous state, prior to the update, and the newly instantiated model GP_{J+1} is discarded. However, if GP_* is not sufficiently similar, then we retain GP_{J+1} and revert all local models back to their state prior to the update.

To be more precise, denote J_k as the number of ensemble models at time k. Upon observing mini-batch $\mathscr{D}_{new}^k$, we update model GP_{j^*} if (6) is less than a threshold ε, otherwise we add GP_{J_K+1} to the collection; succinctly stated as

$$\begin{aligned} \{GP_j\}_{j=1}^{J_{k+1}} &\leftarrow \left\{\{GP_j\}_{j=1}^{J_k} \setminus \{GP_{j*}\}\right\} \cup \{GP_{j*}^{upd}\} && \text{if } w_{j^*} \le \varepsilon \\ \{GP_j\}_{j=1}^{J_{k+1}} &\leftarrow \{GP_j\}_{j=1}^{J_k} \cup \{GP_{J_k+1}\} && \text{if } w_{j^*} > \varepsilon \end{aligned} \tag{7}$$

We summarize the process of iteratively developing the collection of models $\{GP_j\}_{j=1}^{J_k}$ in Algorithm 1.

Remark 1: Given that each model needs to be temporarily updated with the new data, the computation of the similarity metric (6) across all models can be computationally prohibitive. In practice, we reduce complexity by only computing the similarity metric over the $\hat{J} < J$ models closest in terms of Euclidean distance in input space given by $\|\mathbf{c}_j - \mathbf{c}_{new}\|^2$ for $j = 1, \ldots, J$, where $\mathbf{c}_j \in \mathscr{X}$ is the geometric center of the pseudo-inputs of model j and $\mathbf{c}_{new} \in \mathscr{X}$ is the geometric center of the new batch of data.

```
Input  : Instantiation threshold $\varepsilon$
Output: Collection of models $\{GP\}_{j=1}^{J_{k+1}}$
$\mathscr{D}_{new}^1 \leftarrow$ Receive 1st batch of data;
$GP_1 \leftarrow$ Init. new $GP$ with $\mathscr{D}_{new}^1$ (Eq. (4));
for $k$ = 2,3,... do
    $\mathscr{D}_{new}^k \leftarrow$ Receive new batch of data;
    $GP_{J_k+1} \leftarrow$ Init. new $GP$ with $\mathscr{D}_{new}^k$ (Eq. (4));
    for $j = 1 : J_k$ do
        $GP_j^{upd} \leftarrow$ Update $GP_j$ with $\mathscr{D}_{new}^k$ (Eq. (5));
        $w_j^{old} \leftarrow$ Similarity wrt previous predictions;
        $w_j^{new} \leftarrow$ Similarity wrt new data;
        $w_j = w_j^{old} + w_j^{new}$ ;
    end
    $w_{j^*} = \min w_j$ ;
    if $w_{j^*} \le \varepsilon$ then
        $\{GP_j\}_{j=1}^{J_{k+1}} \leftarrow \{GP_j\}_{j=1}^{J_k} \setminus \{GP_{j*}\} \cup \{GP_{j*}^{upd}\}$;
    else
        $\{GP_j\}_{j=1}^{J_{k+1}} \leftarrow \{GP_j\}_{j=1}^{J_k} \cup \{GP_{J_k+1}\}$.
    end
end
```

Algorithm 1: Wasserstein-Splitting GP Regression (WGPR): train online approximate GP ensemble.

B. Model Evaluation and Prediction

The previous approaches [4], [34], [35] propose combining predictions from various models to form a single weighted prediction, as mentioned in the preceding section, mostly inspired by ensemble [31] and local kernel smoothing techniques [32], [33]. However, the validity of doing so hinges upon an i.i.d. stationarity hypothesis for the unknown data distribution. This makes these approaches well-suited to capturing global patterns (long-term spatial correlations) in data at the expense of filtering out local patterns [40]. In an effort to preserve the non-stationary features of a heterogeneous spatial field, we predict the spatial field at a location of interest $\mathbf{x}_s$ by considering the model with the pseudo-input closest to $\mathbf{x}_s$ evaluated as $GP_* = \arg\min_{j=1:J}\left\{\min_{\mathbf{z}_i \in \mathbf{Z}_j} \|\mathbf{z}_i - \mathbf{x}_s\|^2\right\}$, and using $GP_*(m_*(\cdot), k_*(\cdot,\cdot))$ to predict $f(\mathbf{x}_s) \sim \mathscr{N}(\mu_s, \sigma_s^2)$ where $\mu_s = m_*(\mathbf{x}_s)$ and $\sigma_s^2 = k_*(\mathbf{x}_s, \mathbf{x}_s)$, with the posterior given by the VFE approximation (4) for models that have not undergone a recursive update. Models that have been recursively updated have the posterior mean and covariance functions given by (5). This better encompasses non-stationarity and heterogeneous nonlinearities, although it may be susceptible to sub-sampling bias or discontinuities as a respective function of the the number of pseudo-inputs in a local model or the kernel hyperparameters.

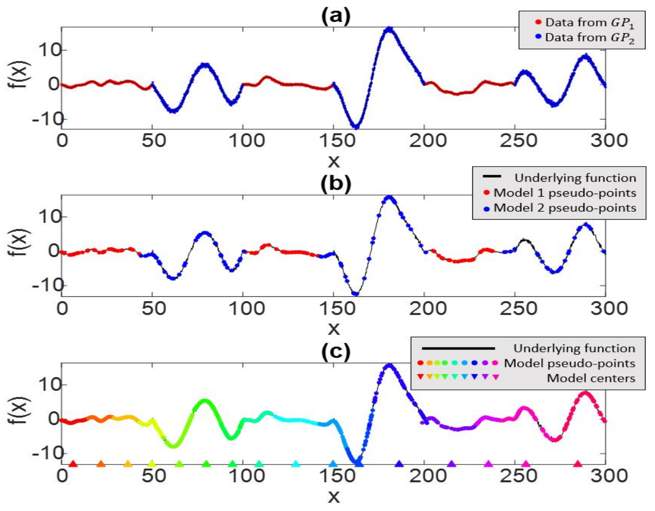

Fig. 1. Toy example motivating our definition of similarity. The training data in (a) is generated from two distinct Gaussian processes with known hyperparameters. By defining similarity in terms of the posterior predictive model, WGPR identifies the two models as seen in (b). Other methods that define similarity based on proximity in input space fail to capture to the two underlying models succinctly as indicated by the 16 models in (c).

IV. Empirical Assessment

In this section, we explore the performance of WGPR across various numerical experiments. We start with a 1-dimensional toy example that motivates defining model similarity in terms of the Wasserstein metric over posterior distributions rather than simply distance in feature space (or a positive definite map thereof). Next, the performance of WGPR is compared to SOLAR-GP [35], which has the same algorithmic structure and Gaussian process model representation, but defines similarity in terms of Euclidean distance. The purpose of this comparison is to highlight the difference in performance due to the new similarity metric and using the prediction of the nearest model. Lastly, we compare the performance of several methods, that learn hyperparameters online, with LGPR, which is optimal in the sense that hyperparameters are learned offline on a training set characteristic of the entire domain. In all comparisons, we evaluate the methods on a real-world bathymetry dataset collected at Claytor Lake in Southwest Virginia, as well as, four other publicly accessible datasets[2]. Our MATLAB-based implementation uses the gradient optimizer and covariance kernel functionality provided by the GPML toolbox [41].

A. Toy-example

To illustrate the motivation behind our definition of similarity, consider the non-stationary dataset depicted in Figure 1(a). This dataset was constructed from two different Gaussian process models with known hyperparameters. Ideally, this data set could be concisely described with two models. The dataset is processed in mini-batches of size $N_{new} = 100$ in spatial order from $\mathbf{x} = 0$ to $\mathbf{x} = 300$ and each local model employs $M = 50$ pseudo points. Provided the instantiation threshold ε is set appropriately, WGPR is able to identify the two underlying models as seen in 1(b). Regardless of how the instantiation threshold is set, SOLAR GP is incapable of succinctly identifying the two underlying models in 1(a) because a new model is instantiated provided new data is sufficiently far from all other existing models. This phenomenon is demonstrated by the 16 models in 1(c). WGPR, with 2 models, achieves a root mean squared error (RMSE) of 0.26. SOLAR GP, with 16 models, achieves an RMSE of 0.39.

B. Comparison of Online Local GPs on Real Data

To examine how our new definition of similarity and making predictions based on the nearest model impact performance, we conducted experiments to examine the difference in performance between WGPR and SOLAR GP [35]. In theory, the model instantiation thresholds can be set to achieve anywhere from a single model to the maximum allowable number of models (as dictated by the number of batches of training data). So ideally, the comparison should highlight the performance difference between the two methods as a function of the number of models used. Thus, the instantiation thresholds are set separately for WGPR and SOLAR to yield approximately the same number of models. Training data was processed in batches of size $N_{new} = 100$ and we employed $M = 50$ pseudo points for each individual Gaussian process model.

We began by setting the instantiation thresholds to yield the maximum number of allowable models and progressively adjusted the thresholds to yield fewer models. For SOLAR GP, despite meticulous tuning of the number of pseudo points and mini-batch size, we regularly observed divergence in the Claytor lake, Abalone, and Sarcos data sets. By divergence, we mean that the posterior mean evaluated at the pseudo-inputs, i.e $m(\mathbf{Z}_b)$, tended to infinity for some local models. This is due to the online update of the VFE posterior approximation [21], which only guarantees we recover the batch posterior and marginal likelihood approximation [12] when the hyperparameters and pseudo-inputs are fixed in advance. When operating with online updates, however, it is possible that pseudo-inputs become very concentrated in a small region of the feature space, causing kernel matrices to become near-singular.

With our definition of similarity (6), on the other hand, we can identify when a stable update can be performed, as the effect of the posterior update is reflected in our similarity metric. More specifically, we only execute EP updates when there is a significant degree of distributional change, and hence statistical independence between pseudo-inputs and the new mini-batch. This precludes the possibility of pseudo-inputs concentrating in a small region of feature space. Numerically, this has the effect of preserving convergence, and overall requiring significantly fewer models on a consistent basis than SOLAR GP. For example, given the batch size of $N_{new} = 100$, the maximum number of models that can be instantiated with the Claytor Lake dataset is 127. We are regularly able to achieve as few as 14.6 models, which is

[2]Kin40k: https://www.cs.toronto.edu/ delve/data/kin/desc.html
Abalone: https://archive.ics.uci.edu/ml/datasets/abalone
Sarcos Joint 1: http://www.gaussianprocess.org/gpml/data/
Pumadyn(8nm): https://www.cs.toronto.edu/ delve/data/pumadyn/desc.html

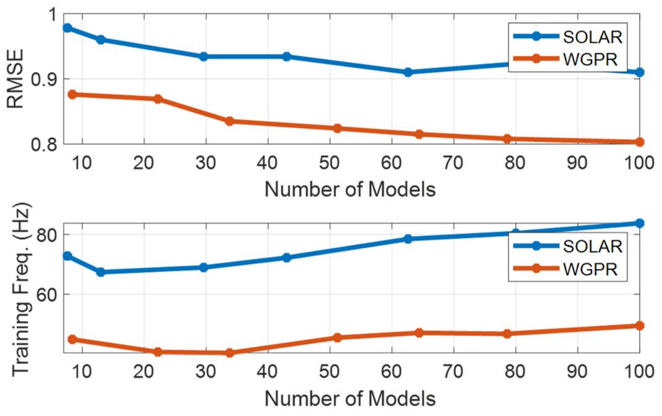

Fig. 2. Comparison of the performance difference between our method (WGPR) and SOLAR GP [35] on the kin40k dataset. Regardless of the number of models used, WGPR outperforms the other online methods at the expense of additional computations required to compute new similarity metric – see Remark 1.

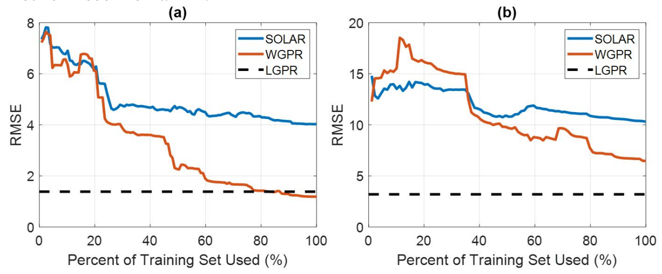

Fig. 3. Evolution of the RMSE as more training batches are processed for Claytor Lake (a) and the Sarcos (b) datasets for SOLAR GP [35], WGPR, and the offline benchmark LGPR [4]. Observe that as more training examples are processed, we approach the performance of the offline method.

the average across 5 trials, with standardized MSE of 0.097, whereas SOLAR GP only achieves sensible predictive performance when there are more than 108.8 models, on average, with standarized MSE of 0.183. Thus, for the Claytor Lake, Sarcos, and Abalone data sets, we can only non-trivially compare predictive performance between SOLAR GP and WGPR in the setting where a new model is instantiated with each new batch of data. We observed that WGPR outperformed SOLAR on the Claytor Lake and Sarcos data sets, and both methods achieved approximately the same predictive performance on the Abalone data set.

For the Kin40k dataset, however, we were able to achieve stable performance for SOLAR GP across various numbers of models employed. We illustrate the overall performance difference between WGPR and SOLAR GP in Figure 2. We report the root MSE, along with the training and predictive frequencies achieved by both methods. Observe that for a given number of models, WGPR outperforms SOLAR GP in terms of predictive performance. This is at the expense of additional computational complexity, as reflected by the lower training frequency. We conclude by stating that we were also able to achieve stable performance for SOLAR GP on the Pumadyn dataset and observed that SOLAR GP (RMSE = 1.088) outperformed WGPR (RMSE = 1.142). However, performance was approximately constant across the number of models used, and thus we omit the performance plots.

C. *Optimal Offline Benchmarking*

In various practical applications, there is no *a priori* access to a dataset, characteristic of the entire state space, to learn the hyperparameters of the predictive model. To cope with these situations, we require that our method have the ability to iteratively update the hyperparameters of the predictive model as new training data is received. As local Gaussian process regression (LGPR) [4] learns the hyperparameters on a characteristic subset of all training data prior to building the prediction model, we consider this an appropriate offline benchmark to assess the predictive performance of WGPR and SOLAR GP. Given that there are parameters unique to each method and LGPR employs the exact posterior (2) for each model, we manually tune the parameters to achieve the best possible predictive performance (averaged across 5 trials) for all algorithms and compare them.

We depict the results for Claytor Lake and the Sarcos dataset in Figure 3 (a) and (b), respectively. Here, we see that as more training data is collected, the predictive performance of WGPR tends to LGPR and in both cases WGPR outperforms SOLAR GP. For all datasets, we collect the RMSE after processing the entire dataset in Table I, where we have delineated LGPR from all other methods that iteratively learn hyperparameters using a dashed lined. We also included the online VFE method of Bui et al. [21], which we denote by OVFE. Note that entries in the table containing a ”-” indicate the algorithm failed to converge. By and large, we see WGPR either outperforms or attains comparable performance with all other methods that iteratively learn the hyperparameters, and gets within striking distance of offline benchmarks (with the exception of the Sarcos).

TABLE I
COMPARISON OF ONLINE METHODS THE OFFLINE BENCHMARK LGPR

	Claytor Lake	Sarcos	kin40k	Abalone	Pumadyn
LGPR [4]	1.381	3.195	0.231	2.070	1.072
WGPR	**1.185**	**6.287**	**0.815**	2.470	1.142
SOLAR [35]	4.030	9.359	0.926	**2.458**	**1.088**
OVFE [21]	-	-	0.996	-	1.204

V. Conclusion

Our empirical results demonstrate that WGPR achieves comparable, if not superior, predictive performance as compared to other methods that iteratively learn hyperparameters. Our new measure of similarity not only allows us to succinctly characterize a non-stationary spatial field, but it also aids in identifying when we can update the posterior approximation of an existing model in a stable manner. For WGPR, there are two user-specified parameters: the instantiation threshold ε, and the number of inducing inputs M in each Gaussian process model. To make our approach more broadly applicable, as an area future research, we seek principled methods to select the threshold ε, perhaps on the basis of a trade-off between memory (number of models) and accuracy. Additionally, we seek principled methods to determine the number of pseudo-points in each local model to ensure a certain quality of approximation.

References

[1] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced lectures on machine learning*. Springer, 2004, pp. 63–71.

[2] S. Vasudevan, F. Ramos, E. Nettleton, and H. Durrant-Whyte, "Gaussian process modeling of large-scale terrain," *Journal of Field Robotics*, vol. 26, no. 10, pp. 812–840, 2009.

[3] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 235–284, 2008.

[4] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model learning with local gaussian process regression," *Advanced Robotics*, vol. 23, no. 15, pp. 2015–2034, 2009.

[5] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 2, pp. 408–423, 2013.

[6] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung, "Kernel methods in system identification, machine learning and function estimation: A survey," *Automatica*, vol. 50, no. 3, pp. 657–682, 2014.

[7] M. Liu, G. Chowdhary, B. C. Da Silva, S.-Y. Liu, and J. P. How, "Gaussian processes for learning and control: A tutorial with examples," *IEEE Control Systems Magazine*, vol. 38, no. 5, pp. 53–86, 2018.

[8] R. Senanayake and F. Ramos, "Bayesian hilbert maps for dynamic continuous occupancy mapping," in *Conference on Robot Learning*, 2017, pp. 458–471.

[9] E. Zobeidi, A. Koppel, and N. Atanasov, "Dense incremental metric-semantic mapping via sparse gaussian process regression," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[10] A. W. van der Vaart, J. H. van Zanten *et al.*, "Rates of contraction of posterior distributions based on gaussian process priors," *The Annals of Statistics*, vol. 36, no. 3, pp. 1435–1463, 2008.

[11] F. Meier and S. Schaal, "Drifting gaussian processes with varying neighborhood sizes for online model learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 264–269.

[12] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Artificial Intelligence and Statistics*, 2009, pp. 567–574.

[13] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," in *Advances in neural information processing systems*, 2006, pp. 1257–1264.

[14] M. Seeger, C. Williams, and N. Lawrence, "Fast forward selection to speed up sparse gaussian process regression," in *Artificial Intelligence and Statistics 9*, no. EPFL-CONF-161318, 2003.

[15] A. J. Smola and P. L. Bartlett, "Sparse greedy gaussian process regression," in *Advances in neural information processing systems*, 2001, pp. 619–625.

[16] C. K. Williams and M. Seeger, "Using the nyström method to speed up kernel machines," in *Advances in neural information processing systems*, 2001, pp. 682–688.

[17] A. Solin and S. Särkkä, "Hilbert space methods for reduced-rank gaussian process regression," *Statistics and Computing*, pp. 1–28, 2014.

[18] M. McIntire, D. Ratner, and S. Ermon, "Sparse gaussian processes for bayesian optimization," in *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2016, pp. 517–526.

[19] A. Koppel, H. Pradhan, and K. Rajawat, "Consistent online gaussian process regression without the sample complexity bottleneck," *arXiv preprint arXiv:2004.11094*, 2020.

[20] L. Csató, "Gaussian processes: iterative sparse approximations," Ph.D. dissertation, 2002.

[21] T. D. Bui, C. Nguyen, and R. E. Turner, "Streaming sparse gaussian process approximations," in *Advances in Neural Information Processing Systems*, 2017, pp. 3299–3307.

[22] T. Broderick, N. Boyd, A. Wibisono, A. C. Wilson, and M. I. Jordan, "Streaming variational bayes," in *Advances in neural information processing systems*, 2013, pp. 1727–1735.

[23] J. Shi, M. Titsias, and A. Mnih, "Sparse orthogonal variational inference for gaussian processes," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 1932–1942.

[24] D. Burt, C. E. Rasmussen, and M. Van Der Wilk, "Rates of convergence for sparse variational gaussian process regression," in *International Conference on Machine Learning*, 2019, pp. 862–871.

[25] A. Wilson and H. Nickisch, "Kernel interpolation for scalable structured gaussian processes (kiss-gp)," in *International Conference on Machine Learning*, 2015, pp. 1775–1784.

[26] S. Remes, M. Heinonen, and S. Kaski, "Non-stationary spectral kernels," in *Advances in neural information processing systems*, 2017, pp. 4642–4651.

[27] C. Toth and H. Oberhauser, "Variational gaussian processes with signature covariances," *arXiv preprint arXiv:1906.08215*, 2019.

[28] K. Blomqvist, S. Kaski, and M. Heinonen, "Deep convolutional gaussian processes," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2019, pp. 582–597.

[29] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Information and computation*, vol. 108, no. 2, pp. 212–261, 1994.

[30] D. Fudenberg and D. Levine, "Consistency and cautious fictitious play," *Journal of Economic Dynamics and Control*, 1995.

[31] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[32] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[33] G. S. Watson, "Smooth regression analysis," *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 359–372, 1964.

[34] S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[35] B. Wilcox and M. C. Yip, "Solar-gp: Sparse online locally adaptive regression using gaussian processes for bayesian robot model learning and control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2832–2839, 2020.

[36] F. Meier, P. Hennig, and S. Schaal, "Incremental local gaussian regression," in *Advances in Neural Information Processing Systems*, 2014, pp. 972–980.

[37] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[38] V. M. Panaretos and Y. Zemel, "Statistical aspects of wasserstein distances," *Annual review of statistics and its application*, vol. 6, pp. 405–431, 2019.

[39] L. Wasserman, *All of nonparametric statistics*. Springer Science & Business Media, 2006.

[40] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When gaussian process meets big data: A review of scalable gps," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[41] C. E. Rasmussen and H. Nickisch, "Gaussian processes for machine learning (gpml) toolbox," *The Journal of Machine Learning Research*, vol. 11, pp. 3011–3015, 2010.