# Escaping Saddle Points in Successive Convex Approximation*

Amrit Singh Bedi, Ketan Rajawat, Vaneet Aggarwal, and Alec Koppel †

**Abstract.** Optimizing non-convex functions is of primary importance in modern machine learning due to the fact that it underlies the training of deep networks and nonlinear dimensionality reduction. While the capability of first-order algorithms to converge to local extrema rather than saddle points has gained attention recently, it is well known that their convergence rates are inferior to those which exploit additional structure. In particular, empirically, successive convex approximation (SCA) converges faster than first-order methods. However, SCA in non-convex settings without additional structure in general converge to first-order stationary points, which could either be local extrema or saddle points whose performance is typically inferior. To mitigate this issue, we propose a perturbation of SCA at appropriate times, which inherits the same convergence rate results as the non-perturbed version, while enhancing the quality of its limit points. In particular, we establish that SCA with perturbations converges to a second order stationary points. Experiments on multi-dimensional scaling, a machine learning problem whose training objective is non-convex, substantiates the performance gains associated with employing random perturbations.

**Key words.** Stochastic optimization, non-convex optimization, first order methods.

**AMS subject classifications.** 46N10, 93E35, 68W27.

## 1. Introduction.

Optimization of non-convex continuous functions is an increasingly salient problem as it underlies the training of deep networks [11], clustering, matrix completion [34], phase retrieval [33], tensor decomposition [9], tensor completion [16, 37], dimensionality reduction [38], among others. These tools are employed to discern patterns in data sets whose scale grows larger by the day. Despite the prevalence of non-convex optimization, methods for solving it efficiently is an ongoing challenge [12]. The most common approaches are gradient-based search [9], convex relaxation [17], and characterizing which geometries exhibit "benign landscapes" that may be exploited through an appropriately chosen metric.

In this work, we adopt the spirit of gradient search for general non-convex problems. Gradient methods are popular due to their ease of implementation, simple analysis, and ability to serve as a template to mitigate the challenges of non-convexity. These challenges include the fact that finding the global extrema of a non-convex problem is NP hard in general, and that the best limit point one may hope for in general (in an almost sure sense) is convergence to local extrema. We note that even attaining acceptable local extrema is non-trivial, since gradient methods converge to stationary points, which could either be saddle points or local extrema. Ensuring convergence to local extrema can be ensured by perturbing the optimization iterates by random noise [10, 24] or augmented step-size rules [7]. While such favorable limiting performance make perturbed gradient search attractive as a template upon which to study finite-time performance [13], it is well-known that successive convex approximation (SCA) experimentally converges far faster [29, 28, 15].

*Submitted to the editors October 2, 2019.

†A. S. Bedi and A. Koppel are with US Army Research Laboratory, Adelphi, MD, USA {Email: amrit0714@gmail.com, alec.e.koppel.civ@mail.mil . K. Rajawat is with the Department of Electrical Engineering, IIT Kanpur, India {Email: ketan@iitk.ac.in}. V. Aggarwal is with the School of Industrial Engineering and the School of Electrical and Computer Engineering, Purdue University, West Lafayette IN 47907, USA {Email: vaneet@purdue.edu}.

SCA, rather than computing its local approximation of the non-convex objective function using the first-order Taylor expansion, as in gradient methods, instead uses any convex surrogate function whose gradient coincides with the original objective. Thus, one may encode any higher-order algorithm through an appropriate choice of convex surrogate. As a result, SCA has been widely used to (a) separate the function in its variables, achieving parallel and distributed implementation, (b) discern computationally cheap updates either because the surrogate admits a closed-form or computationally affordable minimizer [30, 28]. In practice, the chosen surrogate could be well-behaved and capable of navigating narrow valleys, i.e., exploiting higher-order structure such as curvature, unlike gradient descent. The empirical performance of convex surrogates has been well-investigated within the context of majorization-minimization [19] and successive convex approximation methods [1, 26, 30]. Unfortunately, existing uses of SCA for non-convex problems are only theoretically guaranteed to converge to stationary points, which may be degenerate saddle points [28]. Thus, one would like to employ SCA due to its favorable convergence in practice, while guaranteeing that it does not become stuck at undesirable critical points of the objective. Doing so is the goal of this work through the use of randomized perturbations.

**2. Related Work and Contributions.** First-order methods have become widespread in machine learning, but their performance on non-convex problems can be problematic, as they may become stuck at saddle points. Inspired from statistical mechanics, augmentations of stochastic search that add random perturbations date back to at least [10, 24]. [24] establishes global convergence in distribution to the set of minimizers. The corresponding nonasymptotic analysis in the context of non-convex learning problems is provided in [25]. We focus on convergence to local extrema in finite time with high probability, a weaker condition but for a more stringent time horizon. The reason for this focus is that finite-time performance may be translated into sample complexity, a salient issue in training learning models. Alternatively, when favorable geometry is present, as in matrix completion or phase retrieval [23, 5, 34], faster local convergence may be attained. However, these performance gains break down when moving from specialized to general non-convex optimization problems.

Beyond first-order methods, majorization-minimization (MM) [35, 19], SCA [29], and quasi-newton methods [39] have been proposed for non-convex non-convex optimization in order to reduce time to convergence. The first two approaches break the non-convex problem into a sequence of minimization problems involving surrogate functions. At the point of approximation, MM minimizes a convex upper-bound for the non-convex objective, while SCA employs a surrogate that is convex without any upper bound condition, but instead gradient consistency. Therefore, SCA better approximates the non-convex objective and results in better algorithm accuracy [15]. Quasi-Newton methods, by contrast, improve upon gradient methods by estimating the Hessian in order to implement approximate Newton steps.

Motivated by the favorable attributes and generality of SCA, in this work we propose augmentations that facilitate extending its favorable learning rates to favorable limiting behavior. To do so, we note that for non-convex objectives, the convergence to first-order stationary points (FOSP) is established in [28, 15]. By contrast, here we employ perturbation methods to escape saddle points and converge to the second order stationary point (SOSP), which under suitable conditions coincide with local extrema [13].

To escape the saddle points, two popular techniques, namely, random perturbations [13] and second-order based methods [22] have been used. A cubic regularization based algorithm is proposed

in [22] with convergence to $\epsilon$ SOSP in $\mathcal{O}(\frac{1}{\epsilon^{3/2}})$ number of iterations. In practice, the second order information based methods require Hessian inversion at each algorithm iteration, which is computationally expensive. On the other hand, in perturbation based methods, noise from a given distribution (uniform in this paper) is added to the algorithm update (when in the neighborhood of a saddle point) to escape the saddle point which is computationally inexpensive. This work proposes SCA with random perturbations in order to inherit the favorable experimental convergence of SCA together with improved limiting performance.

To do so, we interpret SCA algorithms as an inexact version of gradient descent, which permits us to establish its convergence is comparable to [3, 31]. Then, building upon this foundation, we augment its limit points through random perturbation arguments pioneered by [13], which in particular. yields its convergence to SOSP. The result is a unique modification of SCA [28], which we call perturbed successive convex approximation (P-SCA), that converges to $\epsilon$ second order stationary point in $\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$ with an extra multiplicative factor of *polylog(d)* where $d$ is the underlying parameter dimension. More broadly, our analysis extends convergence of random perturbation methods to SOSP to any arbitrary inexact gradient descent algorithm under an additional hypothesis that the norm of the error in the gradient is bounded. Thus, our main contributions are as follows:

- To date, convergence rates of successive convex approximation is missing, i.e., existing analyses focus on asymptotic convergence [28]. By contrast, we present the convergence rate analysis for SCA.
- By virtue of these rates, we are then able to employ random perturbation based arguments to establish conditions under which randomization yields convergence to second-order stationary points. Doing so requires extending the ideas in [13] methods which employ inexact gradient directions. The reason for this is that we establish that SCA may be interpreted as a member of the family of inexact gradient algorithms, and hence is no longer a first order method.
- With this foundation laid, the results of [13] are verified and proved for gradient descent with in errors.
- Experimentally, we demonstrate the advantages of the proposed algorithm via solving a multi-dimensional scaling (MDS) problem in Sec. 6.

## 3. Preliminaries.

In this paper, we are solving a non-convex optimization problem with the help of gradient based methods. To follow the analysis, we describe some useful definitions which will be referred throughout the paper.

**Definition 1** For a differentiable function $U(\mathbf{x})$, a point $\mathbf{x}$ is $\epsilon$ first order stationary point (FOSP) if $\|\nabla U(\mathbf{x})\| \leq \epsilon$.

**Definition 2** (Strict saddle point) For a twice differentiable function, a point $\mathbf{x}$ is a strict saddle point if $\mathbf{x}$ is a FOSP and $\lambda_{\min}(\nabla^2 U(\mathbf{x})) < 0$. For a strict saddle point, the second order stationary point (SOCP) is a local minima of the non-convex objective as defined next.

**Definition 3** ($\epsilon$ second order order stationarity) For a non-convex function $U(\mathbf{x})$, a point $\mathbf{x}$ is $\epsilon$ second order stationary if $\|\nabla U(\mathbf{x})\| \leq \epsilon \quad \text{and} \quad \lambda_{\min}(\nabla^2 U(\mathbf{x})) \geq -\sqrt{L_2\epsilon}$, where $L_2$ is Hessian Lipschitz constant.

## 4. Problem Formulation.

In this work, we focus on the minimization of a non-convex objective function $U(\mathbf{x})$ given by

$$\min_{\mathbf{x}} \; U(\mathbf{x}) \tag{4.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the optimization variable, $U(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is a twice differentiable non-convex function. For a general non-convex optimization problem, it is well known that an algorithm cannot achieve global minima using first order algorithms [9, 13]. In literature, there are various algorithms proposed which converges to the approximate FOSP of the problem (4.1). A FOSP could be a local minima, local maxima, or a saddle point. It is sufficient to achieve convergence to FOSP in convex settings because it coincides with the global minima of the function. However, for the non-convex functions, it may be highly suboptimal to achieve FOSP if it denotes the local maxima and we are minimizing the function and vice versa. In addition, the converged FOSP can be a saddle point for the unconstrained optimization problem and algorithm might get stuck there. Hence, it becomes important to look for the convergence to a second order stationary point. In literature, various algorithms are proposed for problem (4.1) with the guarantee of convergence to a second order stationary point [6, 13, 14]. However, they are based on the standard gradient descent algorithm.

Recently, an SCA based algorithm is proposed by [28] to solve the problem of (4.1) in an iterative manner. We utilize the similar ideas and use SCA to solve the unconstrained version of the problem. The SCA algorithm is shown to outperform the other first order methods in literature [28]. In this method, the non-convex objective function $U(\mathbf{x})$ is approximated by a surrogate convex function $\tilde{U}(\mathbf{x};\mathbf{y})$ approximated at $\mathbf{y}$. Utilizing this idea, the standard SCA algorithm to solve the problem in (4.1) is summarized in Algorithm 4.1.

**Algorithm 4.1** Successive convex approximation (SCA)

1: **Set** $t = 0$, initialize $\eta = (0, 1]$,
2: **STOP** if $\mathbf{x}_t$ is a FOSP
3: Solve the following optimization problem to get $\hat{\mathbf{x}}(\mathbf{x}_t)$

$$\hat{\mathbf{x}}(\mathbf{x}_t) := \arg\min_{\mathbf{x}\in\mathbb{R}^n} \tilde{U}(\mathbf{x};\mathbf{x}_t), \tag{4.2}$$

4: Set $\mathbf{x}_{t+1} = \mathbf{x}_t + \eta(\hat{\mathbf{x}}(\mathbf{x}_t) - \mathbf{x}_t)$
5: Set $t = t + 1$ and go to step 2

In Algorithm 4.1, at each step $t$, a convex approximation $\tilde{U}(\mathbf{x};\mathbf{x}_t)$ of the non-convex objective function $U(\mathbf{x})$ is used to perform the update in step 3. The main idea to construct an approximation function $\tilde{U}(\cdot)$ is to utilize a simple convex function whose gradient calculation is easier. A detailed discussion about the selection of $\tilde{U}(\cdot)$ is provided in [28]. The approximated convex function $\tilde{U}$ should preserve the first order condition of the original non-convex function such that the gradient of the both functions is same; see Assumption 2. It is important to note that unlike the standard successive approximation techniques, the proposed algorithm does not require the $\tilde{U}(\cdot)$ to upper bound the non-convex function $U$. This provides a lot of freedom in the selection of $\tilde{U}(\cdot)$ which depends upon the application of interest. Some specific examples are provided next.

• **Block wise convex:** If the objective function $U(\mathbf{x};\mathbf{y})$ is block wise convex with respect to each block $i$, the following approximation can be used

$$\tilde{U}_i(\mathbf{x}_i;\mathbf{y}) := U(\mathbf{x}_i,\mathbf{y}_{-i}) + \frac{\tau_i}{2}(\mathbf{x}_i - \mathbf{y}_i)^T\mathbf{H}_i(\mathbf{y})(\mathbf{x}_i - \mathbf{y}_i), \tag{4.3}$$

where $\mathbf{y} := \{\mathbf{y}_i\}_{i=1}^I, \mathbf{y}_{-i} := \{\mathbf{y}_j\}_{j\neq i}$.

• **Product of functions:** If the objective function is of the type $U(\mathbf{x}) = f_1(\mathbf{x})f_2(\mathbf{x})$ where $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are convex and positive, we can use

$$\tilde{U}(\mathbf{x};\mathbf{y}) = f_1(\mathbf{x})f_2(\mathbf{y}) + f_1(\mathbf{y})f_2(\mathbf{x}) + \frac{\tau_i}{2}(\mathbf{x}-\mathbf{y})^T\mathbf{H}(\mathbf{y})(\mathbf{x}-\mathbf{y}).$$

Algorithm 4.1 converges to the first order stationary point as derived in [28, Theorem 2], which states that for a constant step size $\eta$, the sequence $\{\mathbf{x}_t\}$ is bounded and each of its limit points is stationary point of the optimization problem in (4.1). But the results presented in [28] are asymptotic in nature and does not provide the number of iterations required to converge to an $\epsilon$ optimal solution. This results is important because it characterizes the convergence behavior of the proposed algorithm. Moreover, as stated earlier, the convergence to FOSP is not a sufficient condition of convergence to local minima. This is because a FOSP could be a saddle point and the proposed algorithm might get stuck there.

In this paper, we are interested in deriving the number of iterations required to achieve an $\epsilon$ FOSP. In addition to that, we want to characterize the number of iterations required to achieve an $\epsilon$ second order stationary point. In literature there are different techniques proposed to escape the saddle points and converge to SOSP. For instant, some of the approached include methods utilizing second order information [20], and random perturbation based methods [13]. Motivated from the advantages of perturbation based methods as discussed in [13] and advantages of SCA methods [28], we use the perturbed method to make the SCA algorithm escape saddle points effectively. A perturbed successive convex approximation (P-SCA) algorithm is summarized in Algorithm 4.2.

**Algorithm 4.2** :P-SCA Algorithm

1: **Set** $t = 0$, initialize $\chi \leftarrow 3\max\{\log\left(\frac{dL_1\triangle_U}{c\epsilon^2\delta}, 4\right)\}$, $\eta \leftarrow \frac{c}{L_1}$, $r \leftarrow \frac{\epsilon\sqrt{c}}{L_1\chi^2}$, $g_{\text{th}} \leftarrow \frac{\epsilon\sqrt{c}}{\chi^2}$, $f_{\text{th}} \leftarrow \frac{c}{\chi^3}\sqrt{\frac{\epsilon^3}{L_2}}$, $t_{\text{th}} = \frac{(1-s)\chi L_1}{c^2\sqrt{\epsilon L_2}}$, $t_{\text{noise}} = -t_{\text{th}} - 1$, $s \in (0,1)$, and $\mathbf{x}_0$

2: If $\|\nabla U(\mathbf{x}_t)\| \leq g_{\text{th}}$ and $t - t_{\text{noise}} > t_{\text{th}}$ then
$\tilde{\mathbf{x}}_t \leftarrow \mathbf{x}_t$, and $t_{\text{noise}} \leftarrow t$
$\mathbf{x}_t \leftarrow \tilde{\mathbf{x}}_t + \boldsymbol{\xi}_t$, where $\boldsymbol{\xi}$ is uniformly $\sim \mathbb{B}_0(r)$ with radius $r$

3: If $t - t_{\text{noise}} = t_{\text{th}}$ and $U(\mathbf{x}_t) - U(\tilde{\mathbf{x}}_{t_{\text{noise}}}) > -(1-s)f_{\text{th}}$ then
**return** $\tilde{\mathbf{x}}_{t_{\text{noise}}}$

4: Solve the following optimization problem to get $\hat{\mathbf{x}}(\mathbf{x}_t)$

$$\hat{\mathbf{x}}(\mathbf{x}_t) := \arg\min_{\mathbf{x}} \tilde{U}(\mathbf{x};\mathbf{x}_t) \tag{4.4}$$

5: Set $\mathbf{x}_{t+1} = \mathbf{x}_t + \eta(\hat{\mathbf{x}}(\mathbf{x}_t) - \mathbf{x}_t)$

6: Set $t = t + 1$ and go to Step 2

In Algorithm 4.2, the standard SCA algorithm updates are used to reach FOSP. If the condition for FOSP are satisfied (which is nothing but the gradient norm is smaller than a threshold value $g_{\text{th}}$), a random perturbation from uniform ball $\mathbb{B}_0(r)$ of radius $r$ around $\tilde{\mathbf{x}}_t$ is added. This happens at most once every $t_{\text{th}}$ number of iterations. After adding perturbation, if the function value is not decreased by the threshold $(1-s)f_{\text{th}}$ (where $s \in (0,1)$ is a constant and defined later in the paper) then $\tilde{\mathbf{x}}_{t_{\text{noise}}}$ is returned as the output of Algorithm 4.2. Here $t_{\text{noise}}$ denotes the last iteration at which the optimization variable was perturbed with respect to current iteration of the algorithm. It is proved in [28] that after

the addition of this random perturbations, the standard gradient descent iterates escapes the saddle points efficiently. We extend the analysis in [28] and [13] papers to prove that the proposed P-SCA algorithm also escapes the saddle points in a similar manner and $\tilde{\mathbf{x}}_{\text{noise}}$ will be in the neighborhood of $\epsilon$ SOSP. We note that the steps in Algorithm 4.2 do not exhibit a gradient descent like update. Hence, it is difficult to analyze the dynamics of the proposed algorithm. To perform the analysis, it becomes important to look at the proposed algorithm from the perspective of inexact gradient descent algorithms [32, 2, 8, 31]. The meaning of term 'inexact' means that the gradient value used for the algorithm update is in error. The error may arise due to stochastic nature of the gradient or partial availability of the gradient [3] etc. In the above mentioned references, all the inexact gradient algorithms are considered for convex objective function settings. The analysis for asymptotic convergence to FOSP under non-convex objective is performed in [3] but with diminishing step size. The work in this paper focuses on more general scenario of non-convex objective and inexactness of the gradient [40].

Hence, the main idea for the inexact version is to interpret the algorithm as a gradient descent algorithm with error. It is proved in literature that under some regularity conditions on the error term in the gradient, the algorithm converges to (or provide close approximation to) the optimal value under convexity assumption . In order to proceed with analysis for the proposed P-SCA algorithm, we first show that the SCA algorithm is nothing but an inexact version of the standard gradient descent algorithm. From step 5 of the Algorithm 4.2, we have

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta(\hat{\mathbf{x}}(\mathbf{x}_t) - \mathbf{x}_t), \tag{4.5}$$

Next, add and subtract the original gradient $\nabla U(\mathbf{x}_t)$ as follows

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta(\nabla U(\mathbf{x}_t) - \nabla U(\mathbf{x}_t) + \hat{\mathbf{x}}(\mathbf{x}_t) - \mathbf{x}_t). \tag{4.6}$$

Finally, we can write the P-SCA algorithm update in the following alternative form

$$\mathbf{x}_{t+1} = \underbrace{\mathbf{x}_t - \eta\nabla U(\mathbf{x}_t)}_{\text{GD update}} + \eta \underbrace{\left[\nabla U(\mathbf{x}_t) + \mathbf{x}_t - \hat{\mathbf{x}}(\mathbf{x}_t)\right]}_{:=\mathbf{e}_t}. \tag{4.7}$$

The first term in (4.7) is similar to the standard gradient descent algorithm. The second term in (4.7) corresponds to the error in the gradient and denoted by $\mathbf{e}_t$. Hence the final gradient used for the update is $\nabla U(\mathbf{x}_t) + \mathbf{e}_t$ as follows

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta(\nabla U(\mathbf{x}_t) + \mathbf{e}_t). \tag{4.8}$$

For the further analysis in this paper, we will utilize the algorithm update form as presented in (4.7).

## 5. Convergence analysis.

This section details about the convergence rate analysis of the proposed P-SCA algorithm and show that the number of iterations required to converge to a second order stationary point are of the order of $\mathcal{O}\left((\text{polylog}(d))\frac{L_1(U(\mathbf{x}_0)-U(\mathbf{x}^\star))}{\epsilon^2}\right)$. This section also establishes the result that the proposed algorithm indeed escapes the saddle point with probability $1-\delta$ for any $\delta > 0$. Note that the updates in the proposed algorithm utilize a convex approximations $\tilde{U}(\mathbf{x};\mathbf{x}_t)$ at each step $t$ for the non-convex function $U(\mathbf{x})$ [28]. Before discussing the mathematical results, some assumptions are required to hold for the objective function and its convex approximation. All the required assumptions are provided next.

Assumption 1. *The continuously differentiable objective function $U(\mathbf{x})$ and its gradient $\nabla U(\mathbf{x})$ are Lipschitz continuous with parameter $L_0 > 0$ and $L_1 > 0$, respectively. This implies that*

$$\|U(\mathbf{x}) - U(\mathbf{y})\| \leq L_0 \|\mathbf{x} - \mathbf{y}\|, \tag{5.1}$$

$$\|\nabla U(\mathbf{x}) - \nabla U(\mathbf{y})\| \leq L_1 \|\mathbf{x} - \mathbf{y}\| \tag{5.2}$$

*for all $\mathbf{x}$ and $\mathbf{y}$.*

Apart from the objective function, the convex surrogate function needs to satisfy the following assumption.

Assumption 2. *The convex approximation function $\tilde{U}(\mathbf{x}; \mathbf{y})$ to non-convex objective function $U(\mathbf{x})$ at $\mathbf{y}$ is continuously differentiable with respect to its first argument such that*

*B1) $\tilde{U}(\cdot; \mathbf{y})$ is uniformly strongly convex with parameter $C$, which implies that*

$$\left(\nabla \tilde{U}(\mathbf{x}; \mathbf{y}) - \nabla \tilde{U}(\mathbf{z}; \mathbf{y})\right)^T (\mathbf{x} - \mathbf{z}) \geq C \|\mathbf{x} - \mathbf{z}\|^2 \tag{5.3}$$

*for all $\mathbf{x}$ and $\mathbf{z}$.*

*B2) The approximation function gradient $\nabla \tilde{U}(\cdot; \mathbf{y})$ is equal to the original function gradient at the approximation point $\mathbf{y}$*

$$\nabla \tilde{U}(\mathbf{y}; \mathbf{y}) = \nabla U(\mathbf{y}). \tag{5.4}$$

*This property is the key to represent the P-SCA algorithm as an inexact gradient descent algorithm.*

Another assumption required for the objective function which upper bounds the rate of change of the Hessian.

Assumption 3. *The objective function $U(\mathbf{x})$ is Hessian Lipschitz with parameter $L_2$, which implies that*

$$\left\|\nabla^2 U(\mathbf{x}) - \nabla^2 U(y)\right\| \leq L_2 \|\mathbf{x} - \mathbf{y}\| \tag{5.5}$$

*for all $\mathbf{x}$ and $\mathbf{y}$.*

All of the above mentioned assumptions are standard and have been considered in literature [28]. Assumption 1 states that the non-convex objective function is Lipschitz which means that the gradient of the objective function is smooth and does not changes arbitrarily. This is an important assumption since it provides an upper bound on the gradient at two difference points in terms of the difference between the points itself. This is a standard assumption in the analysis of inexact gradient descent algorithms, for instance see [27]. Next, assumption 2 is for the convex approximation $\tilde{U}(\mathbf{x}; \mathbf{y})$ to the non-convex objective function $U(\mathbf{x})$ at $\mathbf{y}$, and states that $\tilde{U}(\mathbf{x}; \mathbf{y})$ is strongly convex at given $\mathbf{y}$ with parameter $C$. This assumption assures that the convex optimization problem in (4.4) has a unique solution. The last assumption 3 states that the gradient of $U(\mathbf{x})$ is smooth and hence the Hessian of the function does not changes abruptly. All of the above mentioned assumptions are utilized to prove the convergence results in this paper.

Before proceeding with the analysis, we remark that the analysis performed in [13] holds for the gradient descent algorithm which is a first order method and is not applicable to SCA based methods.

This is because there is a minimization step included in the proposed Algorithm 4.2 which is not present in [13]. To proceed with the analysis, we first prove that the proposed algorithm is an inexact version of the gradient descent algorithm. Therefore, Assumption 1 is required to perform the analysis and handle the error in the inexact version of the gradient. This is a standard assumption in the analysis of inexact gradient descent algorithms [27]. Then, all the results of [13] are verified and proved for the gradient descent algorithm with gradient in error.

We now show that it is possible for the SCA algorithm to converge to the second order stationary point with a simple modification. In the SCA algorithm updates, once the gradient norm is less than a threshold value $g_{\text{th}}$, we add a small random perturbation to current iterate $\tilde{\mathbf{x}}_t$. In other words, a random perturbation is added to the algorithm updates at most once after every $t_{\text{th}}$ iterations. To make the analysis simple, similar to [13] we assume that the random perturbation $\boldsymbol{\xi}_t$ is uniformly sampled from a ball $\mathbb{B}_0(r)$ of radius $r$ centered around $\tilde{\mathbf{x}}_t$. After adding the perturbation at iteration $t$, if the function value does not decrease by $(1-s)f_{\text{th}}$ after $t_{\text{th}}$ iterations, then the algorithm stops and return $\tilde{\mathbf{x}}_{t_{\text{noise}}}$. This paper proves that the output $\tilde{\mathbf{x}}_{t_{\text{noise}}}$ is essentially the second order stationary point for the problem in (4.1). The main result is presented next in Theorem 5.1.

**Theorem 5.1.** *Let the assumptions 1-3 are satisfied, there exits a constant $c_{\max}$ such that, for any $\delta > 0$, $\epsilon \leq \frac{L_1^2}{L_2}$, $\triangle_U \geq U(\mathbf{x}_0) - U(\mathbf{x}^\star)$, and $c \leq c_{\max}$, the output of Algorithm 4.2 is a $\epsilon$ second order stationary point with probability $(1-\delta)$ for problem (4.1). To achieve the $\epsilon$ second order stationary point, the number of iterations required for the algorithm is given by*

$$\mathcal{O}\left(\frac{L_1 \triangle_U}{\epsilon^2} \log^4\left(\frac{d L_1 \triangle_U}{\epsilon^2 \delta}\right)\right). \tag{5.6}$$

It is interesting to note that the convergence rate of the proposed algorithm remains the same as that of the one in [13, Theorem 3]. The proof of the Theorem 5.1 and succeeding lemmas follows the similar ideas to [13]. Theorem 5.1 establishes the fact that the algorithm terminates in finite number of iterations. Before discussing the proof of Theorem 5.1, there are some other results which we need to discuss first. The algorithm operation can be divided in two scenarios 1) the iterate is not close to FOSP and the gradient $\nabla U(\mathbf{x}_t)$ is large and 2) when gradient is small but the Hessian $\nabla U(\mathbf{x}_t)$ has a very small minimum eigenvalue. First, to prove the convergence to FOSP, we need to show that a single iteration of the proposed algorithm results in a descent direction for the objective function $U(\mathbf{x})$ which is stated in Lemma 5.2, whose proof is provided in the Appendix A.

**Lemma 5.2.** *Under the Assumptions 1-2, and for a $L_2$ Hessian Lipschitz objective function $U(\mathbf{x})$, one iterate of Algorithm 4.2 is a descent direction, which implies that*

$$U(\mathbf{x}_{t+1}) \leq U(\mathbf{x}_t) - \eta' \left\|(\tilde{\mathbf{x}}(\mathbf{x}_t) - \mathbf{x}_t)\right\|^2 \tag{5.7}$$

*where $\eta' := \eta C - \eta^2 \frac{L_1}{2}$ with $\eta \leq \frac{2C}{L_1}$.*

The above mentioned analysis helps us to derive an upper bound on the gradient error norm. Consider the additional term in (4.7), we have

$$\|\mathbf{e}_t\| = \|\nabla U(\mathbf{x}_t) + \mathbf{x}_t - \hat{\mathbf{x}}(\mathbf{x}_t)\| \leq \|\nabla U(\mathbf{x}_t)\| + \|\hat{\mathbf{x}}(\mathbf{x}_t) - \mathbf{x}_t\|. \tag{5.8}$$

From the upper bound in (A.3) and Lipschitz property of the objective function, we have

$$\|\mathbf{e}_t\| \leq L_0\left(1 + \frac{1}{C}\right) =: \mathcal{D}. \tag{5.9}$$

Before proceeding, let us define the following parameters as specified in [28] which are used for deriving the mathematical results next.

$$F := \frac{\eta L_1}{L_2^2}\gamma^3 \cdot \log^{-3}\left(\frac{d\kappa}{\delta}\right),\ G := \frac{\sqrt{\eta L_1}}{L_2}\gamma^2 \cdot \log^{-2}\left(\frac{d\kappa}{\delta}\right)$$

$$L := \sqrt{\eta L_1}\frac{\gamma}{L_2} \cdot \log^{-1}\left(\frac{d\kappa}{\delta}\right),\quad \mathcal{T} := \frac{\log\left(\frac{d\kappa}{\delta}\right)}{\eta\gamma} \tag{5.10}$$

where $\gamma$ is the negative eigenvalue and the condition number $\kappa$ is is given by $\kappa = \frac{L_1}{\gamma} \geq 1$. Once the algorithm iterate $\mathbf{x}_t$ is near a FOSP, it holds that $\|\nabla U(\mathbf{x}_t)\| \leq g_{\text{th}}$ (gradient is small) and the minimum eigen value of Hessian is largely negative $\lambda_{\min}(\nabla^2 U(\mathbf{x}_t)) \leq -\sqrt{L_2\epsilon}$. When $\mathbf{x}_t$ is near a FOSP, then we add perturbation to $\mathbf{x}_t$ followed by SCA updates for $t_{\text{th}}$ steps. We prove that after these $t_{\text{th}}$ steps, the function value will decrease by at least $f_{\text{th}}$ with high probability. This result is formalized in Lemma 5.3.

Lemma 5.3. *Under the Assumption 1-2, for any $\delta \in (0, d\kappa/e]$, if for given $\tilde{\mathbf{x}}$, $\|\nabla U(\tilde{\mathbf{x}})\| \leq G$ (gradient is small) and $\lambda_{\min}(\nabla^2 U(\tilde{\mathbf{x}})) \leq -\gamma$ (sufficiently negative), then if we perform $\mathbf{x}_0 = \tilde{\mathbf{x}} + \boldsymbol{\xi}$ where $\boldsymbol{\xi} \in \mathbb{B}_{\tilde{\mathbf{x}}}(r)$ with radius $r = \frac{L}{\kappa \log\left(\frac{d\kappa}{\delta}\right)}$, it holds that*

$$U(\mathbf{x}_T) - U(\tilde{\mathbf{x}}) \leq -F + 16L_2\eta^3\mathcal{D}^3 \tag{5.11}$$

*for any $T \geq \frac{1}{c_{\max}\mathcal{T}}$ with probability $1-\delta$ when the step size is selected as $\eta \leq \frac{C_{\max}}{L_1}$ and $\delta = \frac{dL_1}{\sqrt{L_2\epsilon}}\exp(-\chi)$.*

The result in Lemma 5.3 states that adding the perturbation decreases the function values further. By selecting $\eta = \frac{c}{L_1}$, $\gamma = \sqrt{L_2\epsilon}$, and $\delta = \frac{dL_1}{\sqrt{L_2\epsilon}}\exp(-\chi)$. We can restate the result in Lemma 5.3 follows. If $\tilde{\mathbf{x}}_t$ is a FOSP, after adding perturbation $\mathbf{x}_t = \tilde{x}_t + \boldsymbol{\xi}_t$, it holds with probability $1-\delta$ that

$$U(\mathbf{x}_{t+th}) - U(\tilde{\mathbf{x}}_t) \leq -f_{\text{th}} + \frac{16L_2c^3\mathcal{D}^3}{L_1^3} \tag{5.12}$$

with $t_{\text{th}} = \frac{\chi}{c^2}\frac{L_1}{\sqrt{L_2\epsilon}}$. Next lemma states that addition of perturbation indeed results in escaping the saddle points.

Lemma 5.4. *Under the conditions similar to Lemma 5.3, let $\mathbf{g}_1$ is the minimum eigen vector of $\nabla^2 U(\tilde{\mathbf{x}})$. If we consider two sequences $\mathbf{u}_t$ and $\mathbf{w}_t$ such that*

$$\|\mathbf{u}_0 - \tilde{\mathbf{x}}\| \leq r,\quad \mathbf{w}_0 = \mathbf{u}_0 + \mu r\mathbf{g}_1\ ;\ \mu \in [\delta/(2\sqrt{d}), 1], \tag{5.13}$$

*then it holds that*

$$\min\{U(\mathbf{u}_T) - U(\mathbf{u}_0), U(\mathbf{w}_T) - U(\mathbf{w}_0)\} \leq -2.5F + 16L_2\eta^3\mathcal{D}^3 \tag{5.14}$$

*for any $\eta \leq \frac{c_{\max}}{L_1}$ and any $T \geq \frac{\mathcal{T}}{c_{\max}}$.*

Lemma 5.4 states that for two points $\mathbf{u}_0$ and $\mathbf{w}_0$, where $\mathbf{w}_0$ lies in the direction of the minimum eigenvector, one of both sequences $\mathbf{u}_t$ and $\mathbf{w}_t$ will result in a further decrement of the objective value

and hence escapes the saddle point. To prove the statement of Lemma 5.4, we need another two results stated next. In Lemma 5.5, we show that if the function value does not decrease after adding perturbation, then all the iterates belongs to a small ball around $\mathbf{u}_0$. The next Lemma 5.6 shows that if the iterates starting from $\mathbf{u}_0$ get stuck, then the sequence $\mathbf{w}_t$ will result in decreasing the function value and hence escapes the saddle point. Here, $\mathbf{w}_t$ is the sequence obtained after applying SCA algorithm steps to $\mathbf{w}_0$ which by taking a step in the direction of minimum eigenvalue of the Hessian denoted by $\mathbf{z}_1$ and starting from $\mathbf{u}_0$.

Lemma 5.5. *For any constant* $\hat{c} \geq \frac{6}{\left(12+\frac{2\mathcal{D}}{\gamma L}\log\frac{d\kappa}{\delta}\right)^2}$, *there exists a constant* $c_{\max}$ *for any* $\delta \in (0, \frac{d\kappa}{e}]$, $f(\cdot)$ *satisfying the assumptions, and* $\|f(\tilde{\mathbf{x}})\| \leq G$, *for an initial point* $\mathbf{u}_0 = \tilde{\mathbf{x}} + \zeta$ *with* $\|\mathbf{u}_0 - \tilde{\mathbf{x}}\| \leq 2r$, *let*

$$T := \min\{\inf_t\{t|\tilde{f}_{\mathbf{u}_0}(\mathbf{u}_t) - f(\mathbf{u}_0) \leq -3F\}, \hat{c}F\}. \tag{5.15}$$

*Then for any* $\eta \leq c_{\max}/L_1$, *it holds that for all* $t < T$ *we have* $\|\mathbf{u}_t - \tilde{\mathbf{x}}\| \leq Z(L\hat{c})$, *where* $Z := 48 + \frac{8\mathcal{D}}{\gamma L}\log\left(\frac{d\kappa}{\delta}\right)$ *with* $\mathcal{D} = L_0\left(1+\frac{1}{C}\right)$.

Lemma 5.6. *There exists* $\hat{c}$, $c_{\max}$ *for any* $\delta \in (0, \frac{d\kappa}{e}]$, $f(\cdot)$ *satisfying the assumptions, and* $\|f(\tilde{\mathbf{x}})\| \leq G$, *for an initial point* $\mathbf{u}_0 = \tilde{\mathbf{x}} + \zeta$ *and sequence* $\{\mathbf{u}_t\}, \{\mathbf{w}_t\}$ *where* $\mathbf{w}_0 = \mathbf{u}_0 + \mu r\mathbf{z}_1$ *with* $\mu \in [\frac{\delta}{2\sqrt{2}}, 1]$, *let us define*

$$T = \min\{\inf_t\{t|\tilde{f}_{\mathbf{w}_0}(\mathbf{w}_t) - f(\mathbf{w}_0) \leq -3F\}, \hat{c}F\} \tag{5.16}$$

*then for any* $\eta \leq c_{\max}/L_1$, *if* $\|\mathbf{u}_t - \tilde{\mathbf{x}}\| \leq Z(L\hat{c})$ *for all* $t < T$, *it holds that* $T < \hat{c}F$.

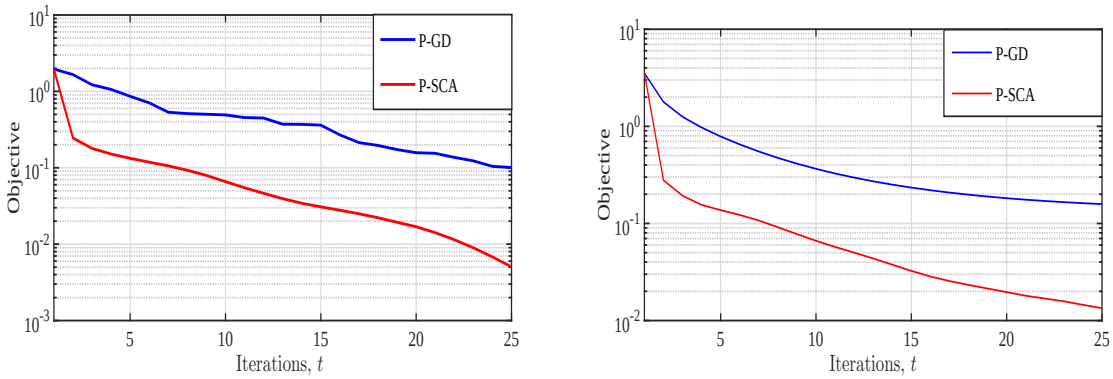
The proof of Lemmas 5.3 - Lemma 5.6 are provided in the Appendix B-E. These intermediate results are then used to prove Theorem 5.1; see Appendix F for the detailed proof.

## 6. Numerical Results.

As a demonstration of the speedup afforded from SCA algorithms, we consider the multi-dimensional scaling (MDS) problem. Here, the goal is to embed relational data onto a low-dimensional euclidean space [18, 36, 4]. Given pairwise dissimilarities $\delta_{mn}$ where $(m,n) \in \mathcal{E}$ for all $\mathcal{E} \subset \{(m,n)|1 \leq m < n \leq N\}$, between $N$ objects, the embedding vectors are given by:

$$\{x_n\}_{n=1}^N = \arg\min_{\{\mathbf{x}_n\}_{n=1}^N} \sum_{1\leq m<n\leq N} w_{mn}(\delta_{mn} - \|\mathbf{x}_m - \mathbf{x}_n\|)^2 \tag{6.1}$$

where $w_{mn}$ is the weight associated with the corresponding dissimilarity measurement, usually set to zero if it is not available. It can be see that (6.1) is non-convex and even the global optimum is not unique, allowing rotational, translational, and reflectional ambiguity. Nevertheless, the objective in (6.1) can be expanded into a difference of convex terms, allowing us to apply the proposed SCA approach, where the surrogate is constructed by linearizing the concave term. The same approach has been widely used within the context of MDS and is referred to as the SMACOF algorithm. We consider a simple example with $N = 200$ and dissimilarities generated from calculating pairwise distances between $N$ euclidean vectors in $[0,1]^2$, keeping only 20% of the dissimilarities, and adding Gaussian noise to the distances with zero mean and variance $0.01$. The non-zero weights are selected as inverse of the corresponding distances, which is the so-called Sammon mapping. The proposed




(a) Larger step size, $\eta = 0.5$ (b) Smaller step size, $\eta = 0.5$

Figure 1: Performance comparisons for multidimensional scaling.

P-SCA as well as the P-GD algorithm have been applied to the MDS problem and the evolution of the objective function in 6.1, normalized by $N^2$ is shown in Fig. 1a and Fig. 1b for larger and smaller step sizes, respectively. It can be seen that the P-SCA is at least several times faster than P-GD algorithm, while retaining its saddle-point escaping properties. The performance of GD becomes worse for the smaller step sizes while SCA algorithm still performs better. It is remarked that for MDS problem, the per-iteration computational complexity of the two algorithms is exactly same.

**7. Conclusions and Future Directions.** This paper considers an algorithm for solving non-convex optimization formed by perturbation of successive convex approximation based method. The proposed perturbed successive convex approximation (P-SCA) algorithm is shown to converge to second order stationary point with a rate similar to the vanilla gradient descent (convergence to first order stationary point) up to a constant factor. Hence, the proposed algorithm can escape the saddle point efficiently utilizing the perturbed variant of the successive convex optimization algorithm. To perform the analysis, the proposed algorithm is proved to be a special case of the standard gradient descent algorithm with the gradient in error, which is called inexact gradient descent in literature. The idea of adding perturbation to the algorithms updates is utilized to escape the saddle points. As a future work, we are interested in considering the constrained version of the proposed algorithm which also escapes the saddle points.

**Appendix A. Proof of Lemma 5.2.** Note that $\hat{\mathbf{x}}(\mathbf{x}_t)$ is the solution of strongly convex problem in (4.4), hence from the first order optimality condition, it holds that

$$(\mathbf{y} - \hat{\mathbf{x}}(\mathbf{x}_t))^T \nabla \tilde{U}(\hat{\mathbf{x}}(\mathbf{x}_t); \mathbf{x}_t) \geq 0. \tag{A.1}$$

By selecting $\mathbf{y} = \mathbf{x}_t$ and add subtract $\nabla \tilde{U}(\mathbf{x}_t; \mathbf{x}_t)$ to the gradient term, we get

$$(\mathbf{x}_t - \hat{\mathbf{x}}(\mathbf{x}_t))^T \Big( \nabla \tilde{U}(\hat{\mathbf{x}}(\mathbf{x}_t); \mathbf{x}_t) - \nabla \tilde{U}(\mathbf{x}_t; \mathbf{x}_t) + \nabla \tilde{U}(\mathbf{x}_t; \mathbf{x}_t) \Big) \geq 0. \tag{A.2}$$

Following the inequality in Assumption 2, it holds that $\nabla U(\mathbf{x}_t) = \nabla \tilde{U}(\mathbf{x}_t; \mathbf{x}_t)$. This implies

$$(\mathbf{x}_t - \hat{\mathbf{x}}(\mathbf{x}_t))^T \nabla U(\mathbf{x}_t) \geq C \left\| \hat{\mathbf{x}}(\mathbf{x}_t) - \mathbf{x}_t \right\|^2. \tag{A.3}$$

From the statement of Assumption 1, it holds that the objective function $U(\mathbf{x})$ is Lipschitz continuous gradient with parameter $L_1$, which implies that

$$U(\mathbf{x}_{t+1}) \leq U(\mathbf{x}_t) + \nabla U(\mathbf{x}_t)^T (\mathbf{x}_{t+1} - \mathbf{x}_t) + \frac{L_1}{2} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 .$$

$$U(\mathbf{x}_{t+1}) \leq U(\mathbf{x}_t) + \eta \nabla U(\mathbf{x}_t)^T (\tilde{\mathbf{x}}(\mathbf{x}_t) - \mathbf{x}_t) + \frac{\eta^2 L_1}{2} \|\tilde{\mathbf{x}}(\mathbf{x}_t) - \mathbf{x}_t\|^2 , \tag{A.4}$$

where (A.4) holds by utilizing the update for $\mathbf{x}_{t+1}$ from Algorithm 4.2. Applying the upper bound in (A.3), we get

$$U(\mathbf{x}_{t+1}) \leq U(\mathbf{x}_t) - \eta C \|\hat{\mathbf{x}}(\mathbf{x}_t) - \mathbf{x}_t\|^2 + \frac{L_1}{2} \|\eta(\tilde{\mathbf{x}}(\mathbf{x}_t) - \mathbf{x}_t)\|^2 \tag{A.5}$$

$$= U(\mathbf{x}_t) + \left(-\eta C + \frac{\eta^2 L_1}{2}\right) \|(\tilde{\mathbf{x}}(\mathbf{x}_t) - \mathbf{x}_t)\|^2 = U(\mathbf{x}_t) - \eta' \|(\tilde{\mathbf{x}}(\mathbf{x}_t) - \mathbf{x}_t)\|^2 . \tag{A.6}$$

where $\eta' := \eta C - \eta^2 \frac{L_1}{2}$. Hence, one run of the Algorithm 4.2 results in a function value decrement with $\eta < \frac{2C}{L_1}$.

**Appendix B. Proof of Lemma 5.3.** From Lipschitz continuous gradient property

$$U(\mathbf{x}_0) \leq U(\tilde{\mathbf{x}}) + \nabla U(\tilde{\mathbf{x}})^T (\mathbf{x}_0 - \tilde{\mathbf{x}}) + \frac{L_1}{2} \|\mathbf{x}_0 - \tilde{\mathbf{x}}\|^2 . \tag{B.1}$$

From the statement of Lemma 5.3 and Cauchy-Schwarz inequality, it holds that

$$U(\mathbf{x}_0) - U(\tilde{\mathbf{x}}) \leq \|\nabla U(\tilde{\mathbf{x}})\| \|\boldsymbol{\xi}\| + \frac{L_1}{2} \|\boldsymbol{\xi}\|^2 \tag{B.2}$$

$$\leq Gr + \frac{L_1}{2} r^2 \leq \frac{GL}{\kappa \log\left(\frac{d\kappa}{\delta}\right)} + \frac{L_1}{2} \left(\frac{L}{\kappa \log\left(\frac{d\kappa}{\delta}\right)}\right)^2 \leq \frac{3}{2} F.$$

The above mentioned bound represents the maximum value by which the function value can increase in the worst case after adding the perturbation. Next, following exactly the similar steps as performed in the proof of Lemma 14 in [28], we obtain

$$U(\mathbf{x}_T) - U(\tilde{x}) = U(\mathbf{x}_T) - U(\mathbf{x}_0) + U(\mathbf{x}_0) - U(\tilde{x})$$

$$\leq -2.5F + 16 L_2 \eta^3 \mathcal{D}^3 + 1.5F \leq -F + 16 L_2 \eta^3 \mathcal{D}^3. \tag{B.3}$$

**Appendix C. Proof of Lemma 5.4.** We consider $\tilde{\mathbf{x}} = 0$ without loss of generality, $T^\star = \hat{c}\mathcal{T}$, and $T'$ defined as $T' = \inf_t \{t | \tilde{U}_{\mathbf{u}_0}(\mathbf{u}_t) - U(\mathbf{u}_0) \leq -3F\}$. In order to prove the lemma statement, let us consider the two cases.

**Case I** ($T' \leq T^\star$)**:** From the statement of Lemma 5.5, we have $\|\mathbf{u}_{T'-1}\| \leq \mathcal{O}(L)$ which implies that

$$\|\mathbf{u}_{T'}\| = \|\mathbf{u}_{T'-1}\| + \eta \|\nabla U(\mathbf{u}_{T'-1})\| + \eta \|\mathbf{e}_{T'-1}\|$$

$$\leq \|\mathbf{u}_{T'-1}\| + \eta \|\nabla U(\tilde{\mathbf{x}})\| + \eta L_1 \|\mathbf{u}_{T'-1}\| + \eta \|\mathbf{e}_{T'-1}\| \leq (1 + \eta L_1)\hat{c} Z L + \eta G + \eta \mathcal{D}. \tag{C.1}$$

From the equality $\frac{G\log(\frac{d\kappa}{\delta})}{\gamma} = L$, we can write $\|\mathbf{u}_{T'}\| \leq Z_1 L + \eta\mathcal{D}$, where $Z_1 := \left((1+\eta L_1)\hat{c}Z + \eta\frac{\gamma}{\log(\frac{d\kappa}{\delta})}\right)$. By choosing a small $c_{\max}$ and $\eta L_1 \leq c_{\max}$, for the Hessian Lipschitz function, we have

$$U(\mathbf{u}_{T'}) - U(\mathbf{u}_0) \leq \nabla U(\mathbf{u}_0)^T(\mathbf{u}_{T'} - \mathbf{u}_0) + \frac{1}{2}(\mathbf{u}_{T'} - \mathbf{u}_0)^T\nabla^2 U(\mathbf{u}_0)(\mathbf{u}_{T'} - \mathbf{u}_0) + \frac{L_2}{6}\|\mathbf{u}_{T'} - \mathbf{u}_0\|^3.$$

Consider the following quadratic approximation

$$\tilde{U}_{\mathbf{y}}(\mathbf{x}) := U(\mathbf{y}) + \nabla U(\mathbf{y})^T(\mathbf{x} - \mathbf{y}) + \frac{1}{2}(\mathbf{x} - \mathbf{y})^T\nabla^2 U(\tilde{\mathbf{x}})(\mathbf{x} - \mathbf{y}).$$

For a Hessian Lipschitz function, it holds that

$$\begin{aligned}
U(\mathbf{u}_{T'}) - U(\mathbf{u}_0) \leq& \nabla U(\mathbf{u}_0)^T(\mathbf{u}_{T'} - \mathbf{u}_0) + \frac{1}{2}(\mathbf{u}_{T'} - \mathbf{u}_0)^T\nabla^2 U(\mathbf{u}_0)(\mathbf{u}_{T'} - \mathbf{u}_0) + \frac{L_2}{6}\|\mathbf{u}_{T'} - \mathbf{u}_0\|^3 \\
\leq& \nabla U(\mathbf{u}_0)^T(\mathbf{u}_{T'} - \mathbf{u}_0) + \frac{1}{2}(\mathbf{u}_{T'} - \mathbf{u}_0)^T\nabla^2 U(\tilde{\mathbf{x}})(\mathbf{u}_{T'} - \mathbf{u}_0) \\
&+ \frac{1}{2}(\mathbf{u}_{T'} - \mathbf{u}_0)^T(\nabla^2 U(\mathbf{u}_0) - \nabla^2 U(\tilde{\mathbf{x}}))(\mathbf{u}_{T'} - \mathbf{u}_0) + \frac{L_2}{6}\|\mathbf{u}_{T'} - \mathbf{u}_0\|^3 \\
\leq& \tilde{U}_{\mathbf{u}_0}(\mathbf{u}_{T'}) - U(\mathbf{u}_0) + \frac{L_2}{2}\|\mathbf{u}_{T'} - \tilde{\mathbf{x}}\|\|\mathbf{u}_{T'} - \mathbf{u}_0\|^2 \qquad \text{(C.2)} \\
&+ \frac{1}{2}(\mathbf{u}_{T'} - \mathbf{u}_0)^T\nabla^2 U(\mathbf{u}_0)(\mathbf{u}_{T'} - \mathbf{u}_0) + \frac{L_2}{6}\|\mathbf{u}_{T'} - \mathbf{u}_0\|^3 \\
\leq& -3F + \mathcal{O}(L_2 L^3) + 16L_2\eta^3\mathcal{D}^3 \\
=& -3F + \mathcal{O}(\sqrt{\eta L_1}F) + 16L_2\eta^3\mathcal{D}^3 \leq -2.5F + 16L_2\eta^3\mathcal{D}^3 \qquad \text{(C.3)}
\end{aligned}$$

by selecting $c_{\max} \leq \min\{1, \frac{1}{\hat{c}}\}$ and $\eta < \frac{1}{L_1}$. From Lemma 1, it holds that for any $T \geq \frac{1}{c_{\max}}\mathcal{T} \geq \hat{c}\mathcal{T} = T^\star \geq T'$, we can write

$$U(\mathbf{u}_T) - U(\mathbf{u}_0) \leq U(\mathbf{u}_{T^\star}) - U(\mathbf{u}_0) \leq U(\mathbf{u}_{T'}) - U(\mathbf{u}_0) \leq -2.5F + 16L_2\eta^3\mathcal{D}^3.$$

**Case II ($T' > T^\star$):** For this case also, we know that $\|\mathbf{u}_t\| \leq \mathcal{O}(L) + \eta\mathcal{D}$ for all $t \leq T^\star$. Let us define $T''$ as $T'' = \inf_t\{t \mid \tilde{U}_{\mathbf{w}_0}(\mathbf{w}_t) - U(\mathbf{w}_0) \leq -3F\}$. From Lemma 5.6, we have $T'' \leq T^\star$. Using the similar arguments as in case 1, we conclude that for all $T \geq \frac{1}{c_{\max}}\mathcal{T}$, it holds that

$$U(\mathbf{w}_T) - U(\mathbf{w}_0) \leq U(\mathbf{w}_{T^\star}) - U(\mathbf{w}_0) \leq -2.5F + 16L_2\eta^3\mathcal{D}^3. \qquad \text{(C.4)}$$

This concludes the proof.

## Appendix D. Proof of Lemma 5.5.

The idea here is to show that if function value does not decrease after performing SCA iterations then it must be bounded in a small ball. This is performed by analyzing the update dynamics of the proposed algorithm via decomposing $d$ dimensional update into two components: one along the subspace $\mathcal{S}$ which is span of significantly negative eigenvalues of the Hessian and second to the subspace compliment to it. Consider the second order Taylor series approximation of the objective function $U(\mathbf{x})$ evaluated at $\tilde{\mathbf{x}}$ given by

$$\tilde{U}_{\tilde{\mathbf{x}}}(\mathbf{x}) := U(\tilde{\mathbf{x}}) + \nabla U(\tilde{\mathbf{x}})^T(\mathbf{x} - \tilde{\mathbf{x}}) + \frac{1}{2}(\mathbf{x} - \tilde{\mathbf{x}})^T\mathcal{H}(\mathbf{x} - \tilde{\mathbf{x}}) \qquad \text{(D.1)}$$

where $\mathcal{H} = \nabla^2 U(\tilde{\mathbf{x}})$. It is emphasized that the gradient of $U$ can be approximated by gradient of $\tilde{U}_{\tilde{\mathbf{x}}}$ provided $\mathbf{x}$ and $\tilde{\mathbf{x}}$ are close. This result is stated as [21]

$$\left\|\nabla U(\mathbf{x}) - \nabla \tilde{U}_{\tilde{\mathbf{x}}}(\mathbf{x})\right\| \leq \frac{L_2}{2} \|\mathbf{x} - \tilde{\mathbf{x}}\|^2 \tag{D.2}$$

for a $L_2$-Hessian Lipschitz function $U$. Set $\mathbf{u}_0 = 0$, from the SCA steps in (4.7), we can write

$$\begin{aligned}
\mathbf{u}_{t+1} =& \mathbf{u}_t - \eta \nabla U(\mathbf{u}_t) + \eta \mathbf{e}_t && \text{(D.3)}\\
=& \mathbf{u}_t - \eta \nabla U(0) - \eta \left[\int_0^1 \nabla^2 U(\theta \mathbf{u}_t) d\theta\right] \mathbf{u}_t + \eta \mathbf{e}_t && \text{(D.4)}\\
=& \mathbf{u}_t - \eta \nabla U(0) - \eta(\mathcal{H} + \triangle_t)\mathbf{u}_t + \eta \mathbf{e}_t && \text{(D.5)}\\
=& (\mathbf{I} - \eta\mathcal{H} - \eta\triangle_t)\mathbf{u}_t - \eta(\nabla U(0) - \mathbf{e}_t) && \text{(D.6)}
\end{aligned}$$

where $\triangle_t := \int_0^1 \nabla^2 U(\theta \mathbf{u}_t) d\theta - \mathcal{H}$. Next, from the Hessian Lipschitz in (5.5), we have $\|\triangle_t\| \leq L_2(\|\mathbf{u}_t\| + \|\tilde{\mathbf{x}}\|)$. From the smoothness of the gradient in Assumption 1, we have

$$\begin{aligned}
\|\nabla U(0) - \mathbf{e}_t\| \leq& \|\nabla U(0) - \nabla U(\tilde{\mathbf{x}}) + \nabla U(\tilde{\mathbf{x}}) + \mathbf{e}_t\| && \text{(D.7)}\\
\leq& \|\nabla U(\tilde{\mathbf{x}})\| + L_1 \|\tilde{\mathbf{x}}\| + \mathcal{D} \leq G + L_1 \cdot 2\frac{L}{\kappa \cdot \log\left(\frac{d\kappa}{\delta}\right)} + \mathcal{D} \leq 3G + \mathcal{D}. && \text{(D.8)}
\end{aligned}$$

Next, we will calculate projections of $\mathbf{u}_t$ on to the subspaces $S$ and $S_c$, where $S$ is the subspace of eigen vectors whose corresponding eigen values are less than $-\frac{\gamma}{\hat{c}\log\frac{d\kappa}{\delta}}$. Further, $S_c$ defines the subspace of the remaining eigen vectors. Next, from the definition of $T$, we know that $\tilde{U}_{\mathbf{u}_0}(\mathbf{u}_t) - U(\mathbf{u}_0) > -3F$. Let $\mathbf{u}_0 = 0$, we get

$$-3F < \tilde{U}_0(\mathbf{u}_t) - U(0) = \nabla U(0)^T \mathbf{u}_t + \frac{1}{2}\mathbf{u}_t^T \mathcal{H} \mathbf{u}_t \tag{D.9}$$

Now decomposing the vector $\mathbf{u}_t$ into $\alpha_t$ (projection on to $S$) and $\beta_t$ (projection on to $S_c$), we get

$$\begin{aligned}
\alpha_{t+1} =& (\mathbf{I} - \eta\mathcal{H})\alpha_t - \eta \mathcal{P}_S \triangle_t \mathbf{u}_t - \eta \mathcal{P}_S(\nabla U(0) - \mathbf{e}_t) && \text{(D.10)}\\
\beta_{t+1} =& (\mathbf{I} - \eta\mathcal{H})\beta_t - \eta \mathcal{P}_{S^c} \triangle_t \mathbf{u}_t - \eta \mathcal{P}_{S^c}(\nabla U(0) - \mathbf{e}_t). && \text{(D.11)}
\end{aligned}$$

Utilizing these definitions, we get $-3F \leq \nabla U(0)^T \mathbf{u}_t - \frac{\gamma}{2}\frac{\|\alpha_t\|^2}{\hat{c}\log\frac{d\kappa}{\delta}} + \frac{1}{2}\beta_t^T \mathcal{H} \beta_t$, where the negative sign comes for the second term comes from the upper bound on the eigen values in subspace $S$. Note that $\|\mathbf{u}_t\|^2 = \|\alpha_t\|^2 + \|\beta_t\|^2$, we get

$$-3F \leq \nabla U(0)^T \mathbf{u}_t - \frac{\gamma}{2}\frac{\|\mathbf{u}_t\|^2 - \|\beta_t\|^2}{\hat{c}\log\frac{d\kappa}{\delta}} + \frac{1}{2}\beta_t^T \mathcal{H} \beta_t. \tag{D.12}$$

After rearranging the terms, we get

$$\begin{aligned}
\|\mathbf{u}_t\|^2 \leq& \frac{2\hat{c}\log\frac{d\kappa}{\delta}}{\gamma}\left(3F + \nabla U(0)^T \mathbf{u}_t + \frac{1}{2}\beta_t^T \mathcal{H} \beta_t\right) + \|\beta_t\|^2\\
\leq& 4\max\left\{\frac{6F\hat{c}\log\frac{d\kappa}{\delta}}{\gamma}, \frac{6G\hat{c}\log\frac{d\kappa}{\delta}}{\gamma}\|\mathbf{u}_t\|, \frac{\beta_t^T \mathcal{H} \beta_t \hat{c}\log\frac{d\kappa}{\delta}}{\gamma}, \|\beta_t\|^2\right\}.
\end{aligned}$$

Taking square root on both sides results in

$$\|\mathbf{u}_t\| \leq 4\max\left\{\sqrt{\frac{6F\hat{c}\log\frac{d\kappa}{\delta}}{\gamma}}, \frac{6G\hat{c}\log\frac{d\kappa}{\delta}}{\gamma}, \sqrt{\frac{\beta_t^T\mathcal{H}\beta_t\hat{c}\log\frac{d\kappa}{\delta}}{\gamma}}, \|\beta_t\|\right\}.$$

In order to prove the lemma statement

$$\|\mathbf{u}_t\| \leq \hat{c}LZ \tag{D.13}$$

we proceed by induction. It holds trivially for $t = 0$ since $\mathbf{u}_0 = 0$. Assume that (D.13) holds for $\tau \leq t$, and then it remains to show that (D.13) holds for $t+1 < T$. It is sufficient to bound the last two terms of (D) which are $\beta_t^T\mathcal{H}\beta_t$ and $\|\beta_t\|$. Consider the update equation for $\beta_t$ from (D.11), we have

$$\beta_t = (\mathbf{I} - \eta\mathcal{H})\beta_t + \eta\delta_t \tag{D.14}$$

where $\|\delta_t\|$ is bounded as

$$\|\delta_t\| \leq \|\triangle_t\|\,\|\mathbf{u}_t\| + \|\nabla U(0) - \mathbf{e}_t\| \tag{D.15}$$

$$\leq L_2(\|\mathbf{u}_t\| + \|\tilde{\mathbf{x}}\|)\,\|\mathbf{u}_t\| + \|\nabla U(0) - \mathbf{e}_t\| \tag{D.16}$$

$$\leq L_2\hat{c}Z\left(\hat{c}Z + \frac{2}{\kappa\cdot\log(\frac{d\kappa}{\delta})}\right)L^2 + 3G + \mathcal{D}$$

$$\leq \left[\hat{c}Z(\hat{c}Z+2)\sqrt{\eta L_1} + 3\right]G + \mathcal{D} \leq 4G + \mathcal{D} \tag{D.17}$$

where the last step follows by choosing $c_{\max} \leq \frac{1}{Z\hat{c}(Z\hat{c}+2)}$ and step size $\eta < c_{\max}/L$.

**Bounding $\|\beta_{t+1}\|$:** We have

$$\|\beta_{t+1}\| \leq \left(1 + \frac{\eta\gamma}{\hat{c}\log\left(\frac{d\kappa}{\delta}\right)}\right)\|\beta_t\| + \eta(4G + \mathcal{D}) \tag{D.18}$$

Applying recursively, we get

$$\|\beta_{t+1}\| \leq \sum_{\tau=0}^{t}(4G+\mathcal{D})\left(1 + \frac{\eta\gamma}{\hat{c}\log\left(\frac{d\kappa}{\delta}\right)}\right)^{\tau}\eta \leq 3(4G+\mathcal{D})\eta T \leq (12L + \eta\mathcal{D}\mathcal{T})\hat{c}. \tag{D.19}$$

Now, from the definition we have $T \leq \hat{c}F$, so that $\left(1 + \frac{\eta\gamma}{\hat{c}\log\left(\frac{d\kappa}{\delta}\right)}\right)^{T} \leq 3$.

**Bounding $\beta_{t+1}^T\mathcal{H}\beta_{t+1}$** : From (D.14), we can write $\beta_t = \sum_{\tau=0}^{t-1}(\mathbf{I} - \eta\mathcal{H})^{\tau}\delta_{\tau}$, which implies that

$$\beta_{t+1}^T\mathcal{H}\beta_{t+1} = \eta^2\sum_{\tau_1=0}^{t}\sum_{\tau_2=0}^{t}\delta_{\tau_1}^T(\mathbf{I}-\eta\mathcal{H})^{\tau_1}\mathcal{H}(\mathbf{I}-\eta\mathcal{H})^{\tau_2}\delta_{\tau_2} \tag{D.20}$$

$$\leq \eta^2\sum_{\tau_1=0}^{t}\sum_{\tau_2=0}^{t}\|\delta_{\tau_1}\|\,\|(\mathbf{I}-\eta\mathcal{H})^{\tau_1}\mathcal{H}(\mathbf{I}-\eta\mathcal{H})^{\tau_2}\|\,\|\delta_{\tau_2}\| \tag{D.21}$$

$$\leq (4G+\mathcal{D})^2\eta^2\sum_{\tau_1=0}^{t}\sum_{\tau_2=0}^{t}\|(\mathbf{I}-\eta\mathcal{H})^{\tau_1}\mathcal{H}(\mathbf{I}-\eta\mathcal{H})^{\tau_2}\|\,. \tag{D.22}$$

If $\{\lambda_i\}$ denotes eigen values of $\mathcal{H}$, then $\lambda_i(1-\eta\lambda_i^{\tau_1+\tau_2})$ denotes the corresponding eigen value of $(\mathbf{I}-\eta\mathcal{H})^{\tau_1}\mathcal{H}(\mathbf{I}-\eta\mathcal{H})^{\tau_2}$. The maxima is achieved for $\lambda^\star_{i,t} = \frac{1}{(1+t)\eta}$. Note that the function $\lambda_i(1-\eta\lambda_i^{\tau_1+\tau_2})$ is monotonically increasing between $(-\infty, \lambda^\star_{i,t}]$. Hence,

$$\begin{aligned}\|(\mathbf{I}-\eta\mathcal{H})^{\tau_1}\mathcal{H}(\mathbf{I}-\eta\mathcal{H})^{\tau_2}\| &= \max_i \lambda_i(1-\eta\lambda_i)^{\tau_1+\tau_2} \\ &\le \hat{\lambda}(1-\eta\hat{\lambda})^{\tau_1+\tau_2} \le \frac{1}{(1+\tau_1+\tau_2)\eta} \end{aligned} \tag{D.23}$$

where $\hat{\lambda} = \min\{L_1, \lambda^\star_{\tau_1+\tau_2}\}$. Hence, we get

$$\begin{aligned}\beta_{t+1}^T\mathcal{H}\beta_{t+1} &\le 2(4G+\mathcal{D})^2\eta T \\ &\le 4(4G^2\eta\hat{c}\mathcal{T}+\eta\hat{c}\mathcal{T}\mathcal{D}^2) \le 2(4\hat{c}L^2\gamma\log^{-1}\left(\frac{d\kappa}{\delta}\right)+\eta\hat{c}\mathcal{T}\mathcal{D}^2)\end{aligned} \tag{D.24}$$

which follows from

$$\begin{aligned}\sum_{\tau_1=0}^{t}\sum_{\tau_2=0}^{t}\frac{1}{(1+\tau_1+\tau_2)\eta} &= \sum_{\tau=0}^{2t}\min\{1+\tau, 2t+1-\tau\}\cdot\frac{1}{1+\tau} \\ &\le 2t+1 < 2T.\end{aligned} \tag{D.25}$$

Finally, substituting the upper bounds in (D.19) and (D.24) into the right hand side of (D), and further simplifying the bounds, we get $\|\mathbf{u}_t\| \le ZL\hat{c}$ where $Z := 48 + \frac{8\mathcal{D}}{\gamma L}\log\left(\frac{d\kappa}{\delta}\right)$ is a constant and $\hat{c}$ is selected such that $\hat{c} \ge \frac{6}{\left(12+\frac{2\mathcal{D}}{\gamma L}\log\frac{d\kappa}{\delta}\right)^2}$. This concludes the proof.

## Appendix E. Proof of Lemma 5.6.

Consider the update equation for $\mathbf{w}_t$, we have

$$\begin{aligned}\mathbf{u}_{t+1}+\mathbf{v}_{t+1} &= \mathbf{w}_{t+1} - \eta\nabla U(\mathbf{w}_t) + \eta\mathbf{e}_t = \mathbf{u}_t+\mathbf{v}_t - \eta\nabla U(\mathbf{u}_t+\mathbf{v}_t) \\ &= \mathbf{u}_t+\mathbf{v}_t - \eta\nabla U(\mathbf{u}_t) - \eta\left[\int_0^1 \nabla^2 U(\mathbf{u}_t+\theta\mathbf{v}_t)d\theta\right]\mathbf{v}_t + \eta\mathbf{e}_t \end{aligned} \tag{E.1}$$

$$= \mathbf{u}_t+\mathbf{v}_t - \eta\nabla U(\mathbf{u}_t) - \eta(\mathcal{H}+\triangle_t')\mathbf{v}_t + \eta\mathbf{e}_t \tag{E.2}$$

$$= \mathbf{u}_t - \eta\nabla U(\mathbf{u}_t) + \eta\mathbf{e}_t + \eta(\mathbf{I}-\eta\mathcal{H}-\triangle_t')\mathbf{v}_t. \tag{E.3}$$

From now onwards, the proof is exactly similar to that of Lemma 17 in [28] but provided here for completeness. As in the earlier proofs, , we have $\|\triangle_t'\| \le L_2(\|\mathbf{u}_t\|+\|\mathbf{v}_t\|+\|\tilde{\mathbf{x}}\|)$. Now, we can write the update for $\mathbf{v}_{t+1}$ as

$$\mathbf{v}_{t+1} = (\mathbf{I}-\eta\mathcal{H}-\triangle_t')\mathbf{v}_t \tag{E.4}$$

Note that we have $\|\mathbf{w}_0-\tilde{\mathbf{x}}\| \le \|\mathbf{u}_0-\tilde{\mathbf{x}}\|+\|\mathbf{v}_0\| \le \frac{L}{\kappa\cdot\log\left(\frac{d\kappa}{\delta}\right)}$, from Lemma (5.5), we have $\|\mathbf{w}_t\| \le ZL\hat{c}$ for all $t \le T$. From the condition in Lemma 5.6, we note that $\|\mathbf{u}_t\| \le ZL\hat{c}$ for all $t < T$. This implies that

$$\|\mathbf{v}_t\| \le \|\mathbf{u}_t\| + \|\mathbf{w}_t\| \le 2ZL\hat{c} \tag{E.5}$$

for all $t < T$. From here, we can derive that

$$\left\|\triangle_t'\right\| \leq L_2(\|\mathbf{u}_t\| + \|\mathbf{v}_t\| + \|\tilde{\mathbf{x}}\|) \leq L_2\left(2ZL\hat{c} + \frac{L}{\kappa \log \frac{d\kappa}{\delta}}\right) \leq L_2 L\left(2Z\hat{c}+1\right). \tag{E.6}$$

Next, we develop a lower bound on $\|\mathbf{v}_t\|$. Let $\psi_t$ be the norm of the projection of $\mathbf{v}_t$ onto $\mathbf{z}_1$ direction and $\phi_t$ be the norm of the projection of $\mathbf{v}_t$ on to the remaining subspace. From (E.4), we can write

$$\psi_t \geq (1+\gamma\eta)\psi_t - \mu\sqrt{\psi_t^2+\phi_t^2} \tag{E.7}$$

$$\phi_t \leq (1+\gamma\eta)\Psi_t + \mu\sqrt{\psi_t^2+\phi_t^2} \tag{E.8}$$

where $\mu = \eta L_2 L\left(2ZL\hat{c}+1\right)$. By induction, next we prove that the following inequality holds for all $t < T$

$$\phi_t \leq 4\mu t \psi_t. \tag{E.9}$$

For $t = 1$ the base case of the induction holds for $t = 0$ from the definition of $\mathbf{v}_0$. Now, let us assume that (E.9) holds for $t \leq T$, we need to show that it holds for $t+1 \leq T$, we have

$$4\mu(t+1)\psi_t \geq 4\mu(t+1)\left((1+\gamma\eta)\psi_t - \mu\sqrt{\psi_t^2+\phi_t^2}\right) \tag{E.10}$$

$$\phi_{t+1} \leq 4\mu t(1+\gamma\eta)\psi_t + \mu\sqrt{\psi_t^2+\phi_t^2}. \tag{E.11}$$

Next, we only need to show that

$$(1+4\mu(t+1))\sqrt{\psi_t^2+\phi_t^2} \leq 4(1+\gamma\eta)\psi_t. \tag{E.12}$$

By selecting $\sqrt{c_{\max}} \leq \frac{1}{2Z\hat{c}+1}\min\{\frac{1}{2\sqrt{2}}, \frac{1}{4\hat{c}}\}$ and $\eta \leq \frac{c_{\max}}{L_1}$, we get

$$4\mu(t+1) \leq 4\mu T \leq 4\eta L_2 L(2Z\hat{c}+1)\hat{c}\mathcal{T} = 4\sqrt{\eta L_1}(2Z\hat{c}+1)\hat{c} \leq 1. \tag{E.13}$$

This implies that

$$4(1+\gamma\eta)\psi_t \geq 4\psi_t \leq 2\sqrt{2\phi_t^2} \geq (1+4\mu(t+1))\sqrt{\psi_t^2+\phi_t^2} \tag{E.14}$$

which proves the induction. From (E.9), we have $\phi_t \leq 4\mu t\psi_t \leq \psi_t$, which gives

$$\psi_{t+1} \geq (1+\gamma\eta)\psi_t - \sqrt{2}\mu\psi_t \geq \left(1+\frac{\gamma\eta}{2}\right)\psi_t \tag{E.15}$$

where the last inequality follows from $\mu = \eta L_2 L(2ZL\hat{c}+1) \leq \sqrt{c_{\max}}(2ZL\hat{c}+1)\gamma\eta\log^{-1}\frac{d\kappa}{\delta} < \frac{\gamma\eta}{2\sqrt{2}}$. From (E.5) and (E.15), we obtain for all $t < T$

$$\begin{aligned} 2ZL\hat{c} \geq \|\mathbf{v}_t\| \geq \psi_t &\geq \left(1+\frac{\gamma\eta}{2}\right)^t \psi_0 \\ &= \left(1+\frac{\gamma\eta}{2}\right)^t c_0 \frac{L}{\kappa}\log^{-1}\frac{d\kappa}{\delta} \geq \left(1+\frac{\gamma\eta}{2}\right)^t \frac{\delta}{2\sqrt{d}}\frac{L}{\kappa}\log^{-1}\left(\frac{d\kappa}{\delta}\right). \end{aligned} \tag{E.16}$$

The above result implies that

$$T \leq \frac{1}{2}\frac{\log\left(4Z\frac{\kappa\sqrt{d}}{\delta}\cdot\hat{c}\log\left(\frac{d\kappa}{\delta}\right)\right)}{\log\left(1+\frac{\gamma\eta}{2}\right)} \leq \frac{1}{2}\frac{\log\left(4Z\frac{\kappa\sqrt{d}}{\delta}\cdot\hat{c}\log\left(\frac{d\kappa}{\delta}\right)\right)}{\gamma\eta} \leq (2+\log(4Z\hat{c}))\mathcal{T}. \tag{E.17}$$

The inequality follows from the selection $\delta \in (0, \frac{d\kappa}{\delta}]$ and we have $\log\left(\frac{d\kappa}{\delta}\right) \geq 1$. We choose $\hat{c}$ large enough such that $2+\log(4Z\hat{c}) \leq \hat{c}$, we get $T < \hat{c}\mathcal{T}$, which concludes the proof.

**Appendix F. Proof of Theorem 5.1.** Let $c < \min\{c_{\max}, 2C\}$, the goal of this proof is to achieve a point $\mathbf{x}$ for which it holds that

$$\|\nabla U(\mathbf{x})\| \leq g_{\text{th}} = \frac{\epsilon\sqrt{c}}{\chi^2}, \quad \lambda_{\min}(\nabla^2 U(\mathbf{x})) \geq -\sqrt{\epsilon L_2}. \tag{F.1}$$

Note that $c \leq 1$, $\chi \geq 1$ as defined in Algorithm 4.2, which implies that $\frac{\sqrt{c}}{\chi^2} \leq 1$ and hence any $\mathbf{x}$ satisfying (F.1) is an $\epsilon$ second order stationary point. To proceed with the proof, let us start from $\mathbf{x}_0$, if $\mathbf{x}_0$ does not satisfy (F.1), then there are two possible outcomes

1. $\|\nabla U(\mathbf{x}_0)\| > g_{\text{th}}$. This means that the first order stationary point is not reached yet and hence Algorithm 2 will not perform the perturbation step. From Lemma 5.2, we have

$$U(\mathbf{x}_1) - U(\mathbf{x}_0) \leq -\frac{\eta}{2}g_{\text{th}}^2 = -\frac{c^2\epsilon^2}{2\chi^4 L_1}. \tag{F.2}$$

2. $\|\nabla U(\mathbf{x}_0)\| \leq g_{\text{th}}$. Under this condition, a perturbation step is performed by Algorithm 2, then SCA steps are performed for the next $t_{\text{th}}$ iterations, and then termination condition is checked. If termination condition is not satisfied, it holds that

$$U(\mathbf{x}_{\text{th}}) - U(\mathbf{x}_0) \leq -f_{\text{th}} + \frac{16L_2c^3\mathcal{D}^3}{L_1^3}. \tag{F.3}$$

where we have substituted step size $\eta = \frac{c}{L_1}$. Now we select $c^3 = \frac{f_{\text{th}}}{16L_2\mathcal{D}^3}$ with $s \in (0,1)$ and introduce $s$ as an adjustment parameter to make $c < \min\{c_{\max}, 2C\}$. Note that we cannot make $s$ arbitrarily small because that would result in s smaller step size $\eta$ which is proportional to $c$. Next, we obtain

$$U(\mathbf{x}_{\text{th}}) - U(\mathbf{x}_0) \leq -(1-s)f_{\text{th}}. \tag{F.4}$$

The above results implies that on an average with each step of the algorithm, the function value reduces by

$$\frac{U(\mathbf{x}_{\text{th}}) - U(\mathbf{x}_0)}{t_{\text{th}}} \leq -\frac{c^3\epsilon^2}{\chi^4 L_1}. \tag{F.5}$$

The function values decreases at least by $\frac{c^3\epsilon^2}{\chi^4 L_1}$ on an average after running for $t_{\text{th}}$ iterations. Since the maximum amount by which the function value may decrease is upper bound by $U(\mathbf{x}_0) - U^\star$, the algorithm must terminate in the following number of iterations

$$\frac{U(\mathbf{x}_0) - U^\star}{\frac{c^3\epsilon^2}{\chi^4 L_1}} = \frac{\chi^4}{c^3}\cdot\frac{L_1(U(\mathbf{x}_0) - U^\star)}{\epsilon^2} = \mathcal{O}\left(\frac{L_1(U(\mathbf{x}_0) - U^\star)}{\epsilon^2}\log\left(\frac{dL_1\triangle_U}{\epsilon^2\delta}\right)\right). \tag{F.6}$$

From Algorithm 4.2, note that the perturbation is added at iteration $t$ if the gradient is small or $\nabla U(\tilde{x}_t) \leq g_{\text{th}}$. From Lemma 5.3, the probability of happening this at each time is at least $1-\delta$ where $\delta = \frac{dL_1}{\sqrt{L_2\epsilon}}\exp(-\chi)$. In other words, the number of times perturbation is added for one run of the algorithm is given by

$$\text{(F.7)} \qquad \frac{1}{t_{\text{th}}} \cdot \frac{\chi^4}{c^3} \cdot \frac{L_1(U(\mathbf{x}_0)-U^\star)}{\epsilon^2} = \frac{\chi^3}{c} \cdot \frac{L_1(U(\mathbf{x}_0)-U^\star)}{\epsilon^2}.$$

Using the union bound, it holds that Lemma 5.3 holds for each of the iterations after adding perturbations. This result makes sure that the Algorithm 4.2 converges to the $\epsilon$ second order stationary point with probability

$$\text{(F.8)} \qquad 1 - \frac{dL_1}{\sqrt{L_2\epsilon}}\exp(-\chi) \cdot \frac{\chi^3}{c}\frac{L_1(U(\mathbf{x}_0)-U^\star)}{\epsilon^2} = 1 - \frac{\chi^3\exp(-\chi)}{c} \cdot \frac{dL_1(U(\mathbf{x}_0)-U^\star)}{\epsilon^2}.$$

Note that we choose $\mathcal{X} = 3\max\{\log\left(\frac{dL_1\triangle_U}{c\epsilon^2}\right), 4\}$, which implies that $\chi \geq 12$, and $\chi^3\exp(-\chi) \leq \exp^{-\chi/3}$. Therefore, we have

$$\text{(F.9)} \qquad \frac{\chi^3\exp(-\chi)}{c} \cdot \frac{dL_1(U(\mathbf{x}_0)-U^\star)}{\epsilon^2} \leq \frac{\exp^{-\chi/3}}{c} \cdot \frac{dL_1(U(\mathbf{x}_0)-U^\star)}{\epsilon^2} \leq \delta.$$

which completes the proof.

## REFERENCES


[1] A. Beck, A. Ben-Tal, and L. Tetruashvili, *A sequential parametric convex approximation method with applications to nonconvex truss topology design problems*, J. Global Opt., 47 (2010), pp. 29–51.

[2] A. S. Bedi, P. Sarma, and K. Rajawat, *Tracking moving agents via inexact online gradient descent algorithm*, IEEE J. Sel. Topics Signal Process., 12 (2018), pp. 202–217.

[3] D. P. Bertsekas and J. N. Tsitsiklis, *Gradient convergence in gradient methods with errors*, SIAM J. Opt., 10 (2000), pp. 627–642.

[4] I. Borg and P. Groenen, *Modern multidimensional scaling: Theory and applications*, Journal of Educational Measurement, 40 (2003), pp. 277–280.

[5] E. J. Candás, X. Li, and M. Soltanolkotabi, *Phase retrieval via wirtinger flow: Theory and algorithms*, IEEE Trans. Inf. Theory, 61 (2015), pp. 1985–2007.

[6] F. E. Curtis, D. P. Robinson, and M. Samadi, *A trust region algorithm with a worst-case iteration complexity of $\mathcal{O}(\epsilon^{-3/2})$ for nonconvex optimization*, Math. Prog., 162 (2017), pp. 1–32.

[7] H. Daneshmand, J. Kohler, A. Lucchi, and T. Hofmann, *Escaping saddles with stochastic gradients*, in International Conference on Machine Learning, 2018, pp. 1163–1172.

[8] R. Dixit, A. S. Bedi, R. Tripathi, and K. Rajawat, *Online learning with inexact proximal online gradient descent algorithms*, IEEE Trans. Signal Process., 67 (2019), pp. 1338–1352.

[9] R. Ge, F. Huang, C. Jin, and Y. Yuan, *Escaping from saddle points — online stochastic gradient for tensor decomposition*, in Proceedings of The 28th Conference on Learning Theory, 2015, pp. 797–842.

[10] S. B. Gelfand and S. K. Mitter, *Recursive stochastic algorithms for global optimization in r^d*, SIAM J. Control Opt., 29 (1991), pp. 999–1018.

[11] S. S. Haykin et al., *Neural networks and learning machines/Simon Haykin.*, New York: Prentice Hall,, 2009.

[12] P. Jain, P. Kar, et al., *Non-convex optimization for machine learning*, Foundations and Trends® in Machine Learning, 10 (2017), pp. 142–336.

[13] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan, *How to escape saddle points efficiently*, in Proc. ICML, 2017, pp. 1724–1732.

[14] K. Y. Levy, *The power of normalization: Faster evasion of saddle points*, arXiv preprint arXiv:1611.04831, (2016).
[15] A. Liu, V. Lau, and B. Kananian, *Stochastic successive convex approximation for non-convex constrained stochastic optimization*, arXiv preprint arXiv:1801.08266, (2018).
[16] X.-Y. Liu, S. Aeron, V. Aggarwal, and X. Wang, *Low-tubal-rank tensor completion using alternating minimization*, arXiv preprint arXiv:1610.01690, (2016).
[17] W.-K. K. Ma, *Semidefinite relaxation of quadratic optimization problems and applications*, IEEE Signal Processing Magazine, 1053 (2010).
[18] L. v. d. Maaten and G. Hinton, *Visualizing data using t-sne*, Journal of machine learning research, 9 (2008), pp. 2579–2605.
[19] J. Mairal, *Stochastic majorization-minimization algorithms for large-scale optimization*, in Adv. Neural Inf. Process. Syst., 2013, pp. 2283–2291.
[20] Y. Nesterov, *Introductory lectures on convex programming volume i: Basic course*, Lecture notes, (1998).
[21] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*, vol. 87, Springer Science & Business Media, 2013.
[22] Y. Nesterov and B. Polyak, *Cubic regularization of newton method and its global performance*, Mathematical Programming, 108 (2006), pp. 177–205.
[23] P. Netrapalli, P. Jain, and S. Sanghavi, *Phase retrieval using alternating minimization*, in Adv. Neural Inf. Process. Sys., 2013, pp. 2796–2804.
[24] R. Pemantle et al., *Nonconvergence to unstable points in urn models and stochastic approximations*, Annals of Prob., 18 (1990), pp. 698–712.
[25] M. Raginsky, A. Rakhlin, and M. Telgarsky, *Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis*, in Proc. of COLT, S. Kale and O. Shamir, eds., vol. 65 of Proceedings of Machine Learning Research, Amsterdam, Netherlands, 07–10 Jul 2017, PMLR, pp. 1674–1703.
[26] M. Razaviyayn, M. Hong, and Z.-Q. Luo, *A unified convergence analysis of block successive minimization methods for nonsmooth optimization*, SIAM J. Opt., 23 (2013), pp. 1126–1153.
[27] M. Schmidt, N. L. Roux, and F. R. Bach, *Convergence rates of inexact proximal-gradient methods for convex optimization*, in Advances in neural information processing systems, 2011, pp. 1458–1466.
[28] G. Scutari, F. Facchinei, and L. Lampariello, *Parallel and distributed methods for constrained nonconvex optimization?part i: Theory*, IEEE Trans. Signal Process., 65 (2017), pp. 1929–1944.
[29] G. Scutari, F. Facchinei, P. Song, D. P. Palomar, and J. Pang, *Decomposition by partial linearization: Parallel optimization of multi-agent systems*, IEEE Trans. Signal Process., 62 (2014), pp. 641–656.
[30] G. Scutari, F. Facchinei, P. Song, D. P. Palomar, and J.-S. Pang, *Decomposition by partial linearization: Parallel optimization of multi-agent systems*, IEEE Trans. Signal Process., 62 (2013), pp. 641–656.
[31] A. M.-C. So and Z. Zhou, *Non-asymptotic convergence analysis of inexact gradient methods for machine learning without strong convexity*, Opt. Methods Soft., 32 (2017), pp. 963–992.
[32] A. M.-C. So and Z. Zhou, *Non-asymptotic convergence analysis of inexact gradient methods for machine learning without strong convexity*, Opt. Methods Software, 32 (2017), pp. 963–992.
[33] J. Sun, Q. Qu, and J. Wright, *A geometric analysis of phase retrieval*, Foundations of Computational Mathematics, 18 (2018), pp. 1131–1198.
[34] R. Sun and Z. Luo, *Guaranteed matrix completion via non-convex factorization*, IEEE Trans. Inf. Theory, 62 (2016), pp. 6535–6579.
[35] Y. Sun, P. Babu, and D. P. Palomar, *Majorization-minimization algorithms in signal processing, communications, and machine learning*, IEEE Trans. Signal Process., 65 (2017), pp. 794–816.
[36] J. Tzeng, H. H.-S. Lu, and W.-H. Li, *Multidimensional scaling for large genomic data sets*, BMC bioinformatics, 9 (2008), p. 179.
[37] W. Wang, V. Aggarwal, and S. Aeron, *Efficient low rank tensor ring completion*, in Proc. IEEE Int. Conf. Computer Vision, 2017, pp. 5697–5705.
[38] W. Wang, V. Aggarwal, and S. Aeron, *Tensor train neighborhood preserving embedding*, IEEE Trans. Signal Process., 66 (2018), pp. 2724–2732.
[39] X. Wang, S. Ma, D. Goldfarb, and W. Liu, *Stochastic quasi-newton methods for nonconvex stochastic optimization*, SIAM J. Opt., 27 (2017), pp. 927–956.
[40] L. Xiao and T. Zhang, *A proximal stochastic gradient method with progressive variance reduction*, SIAM J. Opt., 24 (2014), pp. 2057–2075.