PERTURBED PROJECTED GRADIENT DESCENT CONVERGES TO APPROXIMATE SECOND-ORDER POINTS FOR BOUND CONSTRAINED NONCONVEX PROBLEMS

Songtao Lu[†]*, Ziping Zhao*[‡]*, Kejun Huang*[*]*, and Mingyi Hong*[†]

[†] Department of Electrical and Computer Engineering, University of Minnesota Twin Cities, Minneapolis, MN, 55455, USA

[‡] Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong

[*]Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, 32611, USA

ABSTRACT

In this paper, a gradient-based method for bound constrained nonconvex problems is proposed. By leveraging both projected gradient descent and perturbed gradient descent, the proposed algorithm, named perturbed projected gradient descent (PP-GD), converges to some approximate second-order stationary (SS2) points (which satisfy certain approximate second-order necessary conditions) with provable convergence rate guarantees. The proposed algorithm is suitable for a large-scale problem since it only uses the gradient information of the objective function. It also seamlessly incorporates variable constraints such as nonnegativity, which is commonly seen in many practical machine learning problems. We provide a concrete theoretical analysis showing that PP-GD is able to obtain *approximate second-order solutions* by extracting the negative curvature of the objective function around the strict saddle points. Numerical results demonstrate that PP-GD indeed converges faster compared to other first-order methods in the presence of strict saddle points.

Index Terms— Strict saddle points, projected gradient descent, convergence rate, second-order stationary (SS2) points

1. INTRODUCTION

In this paper, we consider a class of nonconvex optimization problems under box constraints on the variables. To be more specific, the problem is shown as the following:

$$\underset{x\in\mathcal{X}}{\text{minimize}} \quad f(x) \tag{1}$$

where $f:\mathbb{R}^d\to\mathbb{R}$ is twice differentiable (possibly nonconvex), and $\mathcal{X}\in\mathbb{R}^d$ denotes the bound constraint. Such problem finds many applications in machine learning and signal processing, including:

• *Nonnegative matrix factorization.* Given a data matrix $M\in\mathbb{R}^{n\times m}$, NMF seeks two nonnegative matrices $X\in\mathbb{R}^{n\times r}$ and $Y\in\mathbb{R}^{m\times r}$ such that $\|XY^T-M\|_F^2$ is minimized [1]. It is also of interest to consider the symmetric case $\min_{X\geq 0}\|XX^T-M\|_F^2$ where $M\in\mathbb{R}^{n\times n}$ [2].

• *Power allocation in wireless communications.* In a wireless network, each transmitter has a limited power budget which needs to be optimized so that the sum achievable rate is maximized. Assume there are K transmitters. This problem can be formulated as

$$\max_{p_1,\ldots,p_K}\sum_{k=1}^{K}\log\left(1+\frac{|h_{kk}|^2p_k}{\sum_{j\neq k}|h_{jk}|^2p_j+\sigma_k^2}\right) \tag{2}$$

$$\text{s.t.}\quad 0\leq p_k\leq P_{\max},\quad \forall k=1,\ldots,K, \tag{3}$$

where $h_{jk}\in\mathbb{C}$ denotes the interference channel fading from the jth transmitter to the kth receiver and $h_{kk}\in\mathbb{C}$ denotes the desired link channel fading, σ_k^2 stands for the noise power at the kth receiver, and $P_{\max}$ is the maximum peak power.

1.1. Motivation

Algorithms that escape strict saddle points—stationary points that have negative curvatures—gained a lot of interest recently. For example, it was shown that when learning shallow networks, a notoriously nonconvex problem, the stationary points are either global minimum or strict saddle points [3, 4]. Similar results were shown that all saddle points are *strict* saddle points in tensor decomposition [5], dictionary learning [6], phase retrieval [7], and a broad class of low-rank matrix factorization problems [8]. These theoretical works showed the landscape of the objective functions in these applications, and illustrated that escaping strict saddle points are paramount in obtaining good solutions to these computationally hard problems.

A way of avoiding strict saddle points is to seek algorithms that are guaranteed to converge to some properly defined *second-order stationary point* (SS2), a point whose Hessian matrix does not have any negative eigenvalue (in some subspace defined by the active constraint set).

1.2. State of the art

For unconstrained smooth problems, it has been shown that gradient descent (GD) converges to second-order stationary (SS2) points with random initializations [9]. One way of finding negative curvature is to occasionally add noise to the iterates. A perturbed version of GD was proposed with convergence guarantees to SS2 points [10], and the convergence rate is provably faster than the ordinary gradient descent algorithm with random initializations. In a follow-up work [11], the authors proposed NEgative-curvature-Originated-from-Noise (NEON), and showed that the perturbed gradient descent is essentially implementing the power method around the saddle point so that a negative curvature of the Hessian matrix is extracted. An accelerated version of NEON that adopts momentum methods to extract the negative curvature, called NEON$^+$, is proposed in [11] as well. For block structured nonconvex problems, it has been shown that perturbed alternating proximal point is also able to converge to the SS2 points but with a faster rate compared with perturbed GD numerically [12].

Despite the exciting developments, none of the existing methods are able to incorporate constraints (as simple as bound constraints) while preserving the advantages convergence property. In practical machine learning and data mining problems, however, bound constraints are ubiquitous due to physical concerns. Examples include

The first two authors contributed equally.

neural networks training with the nonnegative constraint [13], nonnegative matrix factorization (NMF) [1], nonnegative tensor factorization [14], resource allocation in wireless networks [15], image restoration, non-convex quadratic programming with box constraints (QPB) [16], to name just a few. Existing work rely on *second-order* information of the objective function to guarantee convergence to a SS2 point [17, 18], including the trust region method [19], cubic regularized Newton's method [20, 21], and a mixed approach of first-order and second-order methods [22]. The convergence rate of second-order methods to SS2 points is only recently studied to handle certain classes of constraints, such as quadratic ones [23]. Nevertheless, second-order methods are oftentimes unfavorable for large-scale problems due to scalability issues. It is of pressing demand to develop first-order methods for bound constrained problems with provable convergence to an SS2 point.

1.3. Contributions of this work

In this paper, we propose a gradient-based algorithm, termed perturbed projected gradient descent (PP-GD). The core of PP-GD is related to the interior descent algorithm, which updates the iterates by the designed line search algorithm so that the objective value is decreased either by exploiting the gradient of the objective function or the negative curvature around strict saddle points without any projection operation.

The main contributions of this work are listed as follows:

1. To the best of our knowledge, this is the first algorithm that only uses the gradient (first-order information) of the objective function such that it is guaranteed that the algorithm can converge to some properly defined SS2 point under bound constraints.
2. We also provide a convergence rate analysis, showing that the PP-GD algorithm converges to an SS2 point at a sublinear rate with respect to the error, and only linearly scaled by the problem dimension.
3. Numerical simulations demonstrate that the proposed algorithm efficiently converges to SS2 points in terms of both the objective value and the trajectory of the iterates.

2. PROPOSED ALGORITHM

In this section, we introduce the proposed first-order algorithm that exploits both the projected and perturbed gradient descent algorithms such that the active variables can be identified and free variables are updated by either the gradient or negative curvature. We start by assuming that the objective function is gradient Lipschitz continuous with constant L_1 and Hessian Lipschitz continuous with constant L_2.

2.1. Principles of the Proposed Algorithm

In this algorithm, there is an important concept related to a set $\mathcal{F}(x)$, which is defined as the following. Let set $\mathcal{X}$ be a compact d-dimensional box, i.e., $\mathcal{X} = \{x \in \mathbb{R}^d | l \le x \le u\}$. We define

$$\mathcal{F}(x) \triangleq \left\{ z \in \mathcal{X} | z_i = \begin{cases} z_i = l_i, \text{if } x_i = l_i \\ z_i = u_i, \text{if } x_i = u_i \\ l_i < z_i < u_i, \text{otherwise} \end{cases} \right\}, \quad (4)$$

as an open *in-face* that x belongs to. Variables x_i that are strictly $l_i < x_i < u_i$ are called free variable and the remaining ones are called active variables. Let $\mathcal{S}(x)$ be the set that contains free variables of x. Therefore, the dimension of $\mathcal{F}(x)$ denoted by $\dim(\mathcal{F}(x))$ is same as the dimension of $\mathcal{S}(x)$ (i.e., the number of free variables).

When the box constraints are considered, we have the following notations to distinguish the gradient of the free variables and the active variables. Specifically, let

$$g_P(x) = P_{\mathcal{X}}\big(x - \frac{1}{L_1}\nabla f(x)\big) - x \quad (5)$$

denote the projected gradient, where

$$P_{\mathcal{X}}(x) = \begin{cases} l_i, & \text{if } x_i \le l_i, \\ x_i, & \text{if } l_i < x_i < u_i, \\ u_i, & \text{if } x_i \ge u_i, \end{cases} \quad (6)$$

and $g(x) \triangleq g_{\mathcal{F}}(x) = P_{\mathcal{S}(x)}(\nabla f(x))$ denote the gradient of the inactive variables, meaning that $P_{\mathcal{S}(x)}(\nabla f(x))$ keeps the gradient of the free variables and set the gradient of the active variables as 0. Assume that $\mathcal{F}(x)$ has at least one free variable. Define the (i,j)th entry of reduced Hessian $H_{\mathcal{F}}(x)$ as the $d \times d$ matrix whose the (i,j)th entry is the (i,j)th entry of $\nabla^2 f(x)$ if both x_i, x_j are free variables and the identity $d \times d$ matrix otherwise.

According to the definition of the second-order stationary condition for constrained optimization problems [24, Proposition 3.3.2], we define SS2 points for problem (1) as follows:
A point x^* is an (ϵ_G, ϵ_H)-SS2 point of problem (1) if the following conditions are satisfied.

$$\|g_P(x^*)\| \le \epsilon_G \quad \text{and} \quad \lambda_{\min}(H_{\mathcal{F}}(x^*)) \ge -\epsilon_H \quad (7)$$

where $\epsilon_G, \epsilon_H > 0$ are sufficiently small, and $\lambda_{\min}(H_{\mathcal{F}}(x^*))$ denotes the smallest eigenvalue of matrix $H_{\mathcal{F}}(x^*)$. Note that if a point x^* only satisfies $\|g_P(x^*)\| \le \epsilon_G$, we call it an ϵ_G-first order stationary (SS1) point.

The details of implementing the algorithm are given in Algorithm 1, where r denotes the index of the iterates. The PP-GD algorithm basically includes two parts, projected gradient descent and interior descent algorithm, which will be fleshed out in the sequel.

Algorithm 1 Perturbed Projected Gradient Descent (PP-GD)

1: **Input:** $x^0, \epsilon_G, \epsilon_H, L_1, L_2, \alpha_P, T, \bar{f}, R, U, \delta$
2: **for** $r = 0, 1, \ldots$ **do**
3: **if** $\|g_P(x^r)\| \ge \epsilon_G$ **then**
4: $$x^{r+1} = P_{\mathcal{X}}(x^r - \alpha_P \nabla f(x^r)) \quad (8)$$
5: **else**
6: $[v(x^r), \mathsf{flag}] = \mathsf{NEON}(x^r, T, \bar{f}, R, U, \delta)$ by Algorithm 2
7: **if** $\mathsf{flag} = \emptyset$ or $\dim(\mathcal{F}(x^r)) = 0$ **then**
8: Output x^r
9: **else** ▷ $\mathsf{flag} = \Diamond$ and $\dim(\mathcal{F}(x^r)) \ge 1$
10: Choose $v(x^r)$ such that $g(x^r)^T v(x^r) \le 0$
11: **if** $g(x^r)^T v(x^r) - \frac{1}{2}\epsilon'_H \ge -\|g(x^r)\|$ **then**
12: $d^r = -g(x^r)$ ▷ Choose gradient direction
13: **else**
14: $d^r = v(x^r)$ ▷ Choose negative curvature direction
15: **end if**
16: Compute x^{r+1} by the line search algorithm
17: **end if**
18: **end if**
19: **end for**

2.2. Projected Gradient Descent (PGD, shown in line 3–4)

PGD is implemented through line 3–4 of Algorithm 1 when the size of the projected gradient is large. A constant stepsize is adopted with $0 < \alpha_P \le 1/L_1$. This procedure guarantees that the objective function is decreasing.

2.3. Interior descent algorithm (IDA, shown in line 10–21)

In Algorithm 1, line 10–21 is called the interior descent algorithm, which handles the case when $\|g_P(x)\|$ is small. A new iterate x^{r+1} generated by the remaining parts of the algorithm will be in the closure of $\mathcal{F}(x^r)$, i.e., $x^{r+1} \in \overline{\mathcal{F}(x^r)}$. In other words, after an appropriate step-size is chosen by line search, iterates x^{r+1} will always stay in the feasible set without any projection. Also, the interior algorithm only updates the free variables within the feasible set while fixing the active variables on the boundaries. The direction of the gradient and negative curvature (if exits) are both computed and the choice of $g(x^r)$ or $v(x^r)$ is made on line 16, which is equivalent to

$$-\frac{g(x^r)^T g(x^r)}{\|g(x^r)\|} < \frac{g(x^r)^T v(x^r)}{\|v(x^r)\|} + \frac{v(x^r)^T H_{\mathcal{F}}(x^r) v(x^r)}{\|v(x^r)\|^2}, \quad (9)$$

since $\|v(x^r)\| = 1$. The above equation shows that after normalizing the size of the direction, we choose the direction that gives more descent of the objective value. To be more specific, the left-hand side of (9) shows the descent of only using the gradient and the right-hand side of (9) shows the one given by the negative curvature.

The interior descent algorithm includes three parts: NEON , line search, and backtracking algorithms.

On the use of negative curvature (by NEON). The NEON algorithm is a first-order method that can extract the negative curvature of the (reduced) Hessian matrix around the strict saddle points efficiently. The theoretical guarantees of the sequence generated by NEON are restated as follows, where the details of implementing NEON are shown in Algorithm 2. From [11, Theorem 2], we know that NEON can obtain the negative curvature with high probability. Let x be a point whose $\lambda_{\min}(H_{\mathcal{F}}(x)) \geq -\epsilon_H$. The theorem says that there exits a constant $c_{\max}$ depended on c such that if NEON is called with stepsize $\beta = c_{\max}/L_1$, $T \triangleq c\eta/(\beta\epsilon_H)$, $\bar{f} \triangleq \beta\epsilon_H^3 L_1/(L_2^2\eta^3)$, $R \triangleq \sqrt{\beta}\epsilon_H^2/(\sqrt{L_1}\eta^2)$, $U \triangleq 4c(\frac{\sqrt{\beta L_1}\bar{f}}{L_2})^{1/3}$, $\eta \triangleq \log(dL_1/(\epsilon_H\delta))$ and $c \geq 18$, then with probability $1-\delta$ it returns $\Diamond$ and a vector $u \triangleq v^T/\|v^T\|$ such that $u^T\nabla^2 f(x)u \leq -\frac{\epsilon_H}{8c^2\log(dL_1/(\epsilon_H\delta))} \triangleq -\epsilon'_H$. If NEON returns $\emptyset$, we can conclude that $\lambda_{\max}(H_{\mathcal{F}}(x)) \geq -\epsilon_H$ with probability $1-\mathcal{O}(\delta)$. By leveraging NEON , we can to obtain an approximate negative value ϵ'_H and the corresponding eigenvector u within a finite number of steps if there exits $\lambda_{\min}(H_{\mathcal{F}}(x)) \geq -\epsilon_H$.

Remark 1. Note that the obtained eigenvector here $v(x^r)$ is orthogonal to the space spanned by $C_{\mathcal{F}}(x^r)^T$ where $C_{\mathcal{F}}(x^r) \triangleq [\ldots, e_i, \ldots], \forall i \in \mathcal{I}(x^r)$ and e_i denotes the standard basis and $\mathcal{I}(x^r)$ denotes the index set of $\mathcal{S}(x^r)$ (see [24, Proposition 3.3.2]).

Remark 2. If the dimension of the reduced Hessian matrix is very small at some iteration, e.g., $\dim(\mathcal{F}(x^r)) \leq d^{\frac{1}{3}}$, we can implement the eigenvalue decomposition directly.

On the use of the line search and backtracking algorithms. Line search algorithm is exploited here for keeping the free variable updated with the feasible set. Specifically, let $\alpha^r_{\max} \triangleq \max\{\alpha \geq 0 | [x^r, x^r + \alpha d^r] \in \mathcal{X}\}$ and choose $x^{r+1} = x^r + \alpha^r_{\max} d^r$. If $f(x^{r+1}) < f(x^r)$, then the algorithm returns x^{r+1} (in this case, the iterate is very close to the boundary.). Otherwise, the algorithm will call backtracking algorithm by $\alpha \leftarrow \alpha_{\max}$ to obtain a sufficient descent. Note that when $\alpha^r_{\max}$ is chosen, we will have $x^{r+1} \notin \mathcal{F}(x^r)$.

Backtracking algorithm is used for finding a α such that iterates x^{r+1} will be stayed in the feasible set without any projection operation. If $f(x^r + \alpha d^r) > f(x^r) + \lambda\rho(\alpha)$, we will implement $\alpha \leftarrow \frac{1}{2}\alpha$ until a sufficient descent is satisfied (see Lemma 2 and Lemma 3).

Algorithm 2 NEON (Finding Negative Curvature)

1: **Input:** $x^r, T, \bar{f}, R, U, \delta$
2: Generate vector y randomly from the sphere of an Euclidean ball of radius R and initialize $v^0 = P_{\mathcal{S}(x^r)}(y)$.
3: **for** $\tau = 0, 1, \ldots, T$ **do**
4:
$$v^{\tau+1} = v^\tau - \beta(g(x^r + v^\tau) - g(x^r)) \quad (10)$$
5: **end for**
6: **if** $f(x^r + v^T) - f(x^r) - g(x^r)^T v^T \leq -2.5\bar{f}$ where $\|v^T\| \leq U$ **then**
7: return $[v^T/\|v^T\|, \Diamond]$
8: **else**
9: return $[0, \emptyset]$
10: **end if**

Here, $\rho(\alpha) = -\alpha\|g(x^r)\|^2$ if $d^r = -g(x^r)$ and $\rho(\alpha) = -\alpha^2\epsilon'_H/4$ if $d^r = v(x^r)$. Without loss of generality, we can choose $\lambda = 0.5$ in this paper.

3. CONVERGENCE ANALYSIS

The convergence analysis will be given in this section, which shows that PP-GD converges to SS2 points in a finite number of steps. In Algorithm 1, it can be observed that d^r could be chosen by gradient $\nabla f(x^r)$, projected gradient $g(x^r)$ and negative curvature $v(x^r)$. Using the line search algorithm ensures that the iterates stay in the feasible set. When $\alpha^r_{\max}$ is chosen, the objective function will not increase. When $\alpha^r_{\max}$ is not chosen, we will have a sufficient descent. We will give the following three lemmas that quantify the minimum decrease of the objective value by implementing one step of the algorithm, i.e., $x^{r+1} = x^r + \alpha^r d^r$. They serve as the stepping stones for the main result that follows. The details of the proof will be given in the journal version.

Lemma 1. *If x^{r+1} is computed by projected gradient descent with step-size chosen by $0 < \alpha_P \leq \frac{1}{L_1}$, then $f(x^{r+1}) \leq f(x^r) - \frac{L_1}{2}\epsilon_G^2$.*

Lemma 2. *If d^r is chosen by $-g(x^r)$ and x^{r+1} is computed by the interior descent algorithm of Algorithm 1, then there exits an $\alpha \geq 1/(2L_1)$ such that $f(x^{r+1}) \leq f(x^r) - \frac{3L_1}{8}\epsilon'^2_H$.*

Lemma 3. *If d^r is chosen by $v(x^r)$ and x^{r+1} is computed by the interior algorithm of Algorithm 1, then there exits an $\alpha \geq 9/(4L_2)$ and $f(x^{r+1}) \leq f(x^r) - 1.5L_2^2\epsilon'_H$ with high probability .*

Since the designed algorithm can sufficient descent of the objective value with high probability, by applying the Boole's inequality (a.k.a. union bound) we can give the following convergence rate result of the proposed algorithm immediately.

Theorem 1. *(Convergence rate) Suppose the objective function satisfies assumption A. The sequence $\{x^r\}$ generated by Algorithm 1 satisfies optimality condition* (7) *in the following number of iterations with high probability.*

$$\widetilde{\mathcal{O}}\left(\frac{d(f(x^0) - f^\star)}{\min\left\{\frac{L_1\epsilon_G^2}{2}, \frac{3L_1\epsilon'^2_H}{8}, 1.5L_2^2\epsilon'_H\right\}}\right) \quad (11)$$

where $f^\star$ denotes the global minimum value of the objective function and $\widetilde{\mathcal{O}}$ hides the number of iterations run by NEON .

Remark 3. NEON in Algorithm 1 is not needed for every step. Also, the accelerated version of NEON (i.e., NEON $^+$) can be used such that we can have a faster convergence rate of PP-GD.

Remark 4. The proposed algorithm can be also applied to deal with general linear constraints, which will involve more complicated projection operators and convergence analysis. Due to the page limit, this part is not included in this paper.

4. ILLUSTRATIVE EXAMPLE

In this part, we use simulations to verify our theoretical results in various scenarios. The results are based on a construction example. The objective function $f(x)$ takes the form of "kelp" as given in the following

$$f(x) = (x_1^2 + x_2^2)\sin(\pi x_1) \tag{12}$$

where the constraint set is $\mathcal{X} = \{-1.5 \le x_1 \le 0.3, -2 \le x_2 \le 2\}$.

The general landscape of the "kelp" function. The general landscape of the construction function $f(x)$ is shown in Figure 1. The

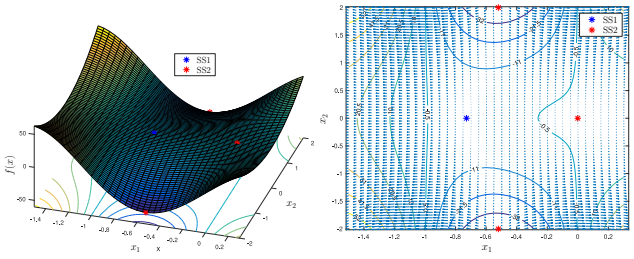

Fig. 1. Landscape and contour with negative gradient flow of $f(x)$.

$f(x)$ is a 2-dimensional function and is symmetric with respect to $x_2 = 0$. There is one SS1 on $x_2 = 0$ shown by the blue aster in Figure 1 and we denote it by $x^{\mathsf{SS1}} = (x_1^{\mathsf{SS1}}, 0)$. There are three SS2s denoted by the red asters in Figure 1. One SS2 is in the interior of $\mathcal{X}$ and is given by $x_{\mathsf{int}}^{\mathsf{SS2}} = (0, 0)$. The other two are on the boundary which are denoted by $x_{\mathsf{bd},-}^{\mathsf{SS2}} = (x_{\mathsf{bd},1}^{\mathsf{SS2}}, -2)$ and $x_{\mathsf{bd},+}^{\mathsf{SS2}} = (x_{\mathsf{bd},1}^{\mathsf{SS2}}, +2)$.

Let "conv." denote the shortcut of converging. The trajectories of iterations on the contour maps and the rate of convergence will be presented in the following figures.

Case 1: Escaping the strict saddle point. As shown in Figure 2, we initialize PGD and PP-GD at $x_{\mathsf{case1}} = (-1.3, 0)$. It can be observed that PGD finally converges to the SS1 (x^{SS1}). PP-GD, however, firstly "lingers" at the SS1 (x^{SS1}) for some time, but finally escapes this point and converges to an SS2 ($x_{\mathsf{bd},+}^{\mathsf{SS2}}$).

Another interesting observation is that since PP-GD can exploit the second-order information of $f(x)$, it obtains a faster convergence rate during the convergence stage from x_{case1} to x^{SS1}.

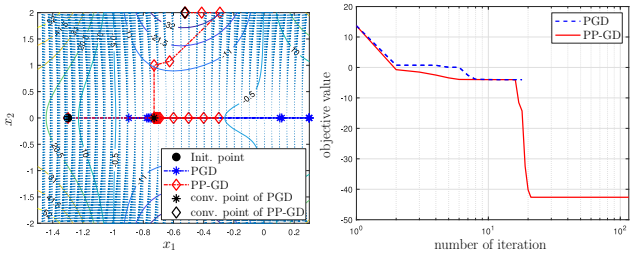

Fig. 2. Convergence comparison of PGD and PP-GD on Case 1.

Case 2: A faster convergence rate of PP-GD over PGD. As a continuing example of **Case 1**, to further look into the faster convergence property of PP-GD, in Figure 3, the two algorithms are initialized at $x_{\mathsf{case2}} = (-1.3, 0.6)$. Notice that in the first several steps, PP-GD takes the same steps as PGD. About after the 10th iteration, PP-GD achieves a faster convergence rate since PP-GD starts to use the second-order curvature information of $f(x)$ to choose the descent direction. But, PGD is still taking the gradient (first-order) descent direction. Although PGD has a similar convergence rate as PP-GD theoretically, this result illustrates that PP-GD in practice can be expected to give a faster convergence result than PGD.

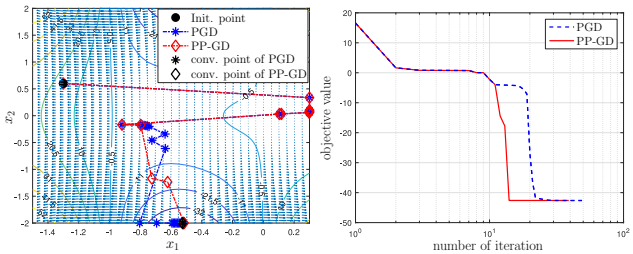

Fig. 3. Convergence comparison of PGD and PP-GD on Case 2.

Case 3: SS1 on the boundary with the perpendicular negative curvature. As another continuing example of **Case 1**, we look into a "hard" case where SS1 is restrained on the boundary. In Figure 4, PP-GD and PGD are still initialized at $x_{\mathsf{case1}} = (-1.3, 0)$, but we shrink $\mathcal{X}$ to be a more restrained space given by $\mathcal{X}_H = \{-1.5 \le x_1 \le 0.3 \text{ and } -2 \le x_2 \le 0\} \subset \mathcal{X}$. The set $\mathcal{X}_H^{\complement} = \mathcal{X} \setminus \mathcal{X}_H$ is denoted as the reddish area in Figure 4. Unlike **Case 1** in Figure 2, both PP-GD and PGD are finally stuck at the SS1 (x^{SS1}). In fact, as shown in Algorithm 1, for this "hard" case, the algorithm PP-GD cannot succeed in escaping from SS1 to SS2.

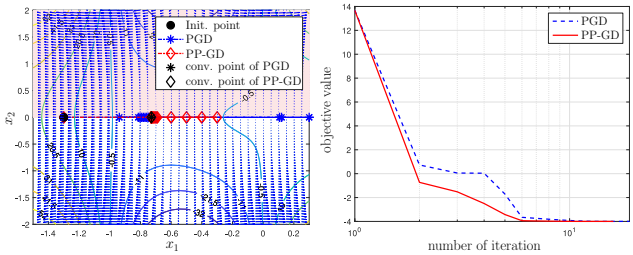

Fig. 4. Convergence comparison of PGD and PP-GD on Case 3.

Case 4: SS1 on the boundary that can be escaped. Continuing **Case 3**, we can further examine what will happen if we initialize the PP-GD and PGD algorithms with SS1 (x^{SS1}) by considering the shrunken set $\mathcal{X}_V = \{x_1^{\mathsf{SS1}} \le x_1 \le 0.3 \text{ and } -2 \le x_2 \le 0\} \subset \mathcal{X}$. In this case, SS1 is still located on the boundary of the set. Since a negative curvature can be found by PP-GD, similar to **Case 1**, PP-GD can finally converge to SS2 ($x_{\mathsf{bd},+}^{\mathsf{SS2}}$), while PGD cannot move at the SS1. This result is shown in Figure 5.

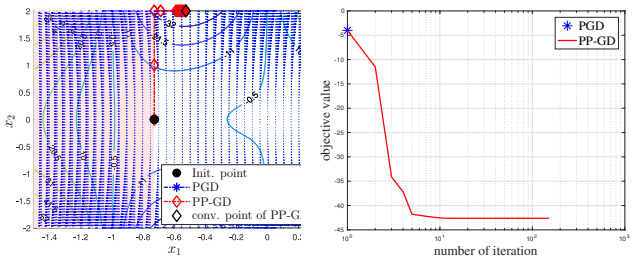

Fig. 5. Convergence comparison of PGD and PP-GD on Case 4.

5. CONCLUSION AND FUTURE WORK

In this work, we have proposed the perturbed projected gradient descent (PP-GD) algorithm for bound constrained nonconvex problems, and we have shown that it is guaranteed to converge to a second-order stationary (SS2) point with a sublinear rate. In the upcoming journal version, we will provide detailed proof for the convergence result, as well as applications to the aforementioned NMF and power allocation problems in machine learning and signal processing to showcase the superiority of the proposed PP-GD.

6. REFERENCES

[1] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788, 1999.

[2] S. Lu, M. Hong, and Z. Wang, "A nonconvex splitting method for symmetric nonnegative matrix factorization: Convergence analysis and optimality," *IEEE Transactions on Signal Processing*, vol. 65, no. 12, pp. 3120–3135, June 2017.

[3] K. Kawaguchi, "Deep learning without poor local minima," in *Proceedings of Neural Information Processing Systems (NIPS)*, 2016, pp. 586–594.

[4] M. Soltanolkotabi, A. Javanmard, and J. D. Lee, "Theoretical insights into the optimization landscape of over-parameterized shallow neural networks," *IEEE Transactions on Information Theory*, 2018.

[5] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle points — online stochastic gradient for tensor decomposition," in *Proceedings of Annual Conference on Learning Theory (COLT)*, 2015, pp. 797–842.

[6] J. Sun, Q. Qu, and J. Wright, "When are nonconvex problems not scary?," in *Proceedings of NIPS Workshop on Nonconvex Optimization for Machine Learning: Theory and Practice*, 2015.

[7] J. Sun, Q. Qu, and J. Wright, "A geometric analysis of phase retrieval," *arXiv:1602.06664 [cs.IT]*, 2017.

[8] R. Ge, C. Jin, and Y. Zheng, "No spurious local minima in nonconvex low rank problems: A unified geometric analysis," in *Proceedings of International Conference on Machine Learning (ICML)*, 2017, pp. 1233–1242.

[9] J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht, "Gradient descent only converges to minimizers," in *Proceedings of Annual Conference on Learning Theory (COLT)*, 2016, pp. 1246–1257.

[10] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan, "How to escape saddle points efficiently," in *Proceedings of International Conference on Machine Learning (ICML)*, 2017, pp. 1724–1732.

[11] Y. Xu and T. Yang, "First-order stochastic algorithms for escaping from saddle points in almost linear time," in *Proceedings of Neural Information Processing Systems (NIPS)*, 2018.

[12] S. Lu, M. Hong, and Z. Wang, "Fast and global optimal nonconvex matrix factorization via perturbed alternating proximal point," in *Proceedings of the 44th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2019.

[13] J. Chorowski and J. M. Zurada, "Learning understandable neural networks with nonnegative weight constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 1, pp. 62–69, Jan. 2015.

[14] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582.

[15] S. Lu and Z. Wang, "Training optimization and performance of single cell uplink system with massive-antennas base station," *IEEE Transactions on Communications*, vol. 67, no. 2, pp. 1570–1585, Feb. 2019.

[16] S. Burer and A. N. Letchford, "On nonconvex quadratic programming with box constraints," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 1073–1089, 2009.

[17] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt, "Second-order negative-curvature methods for box-constrained and general constrained optimization," *Computational Optimization and Applications*, vol. 45, no. 2, pp. 209–236, 2010.

[18] C. W. Royer and S. J. Wright, "Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization," *SIAM Journal on Optimization*, vol. 28, no. 2, pp. 1448–1477, 2018.

[19] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust region methods*, SIAM, 2000.

[20] Y. Nesterov and B. T. Polyak, "Cubic regularization of Newton method and its global performance," *Mathematical Programming*, vol. 108, no. 1, pp. 177–205, 2006.

[21] Y. Carmon and J. C. Duchi, "Gradient descent efficiently finds the cubic-regularized non-convex Newton step," in *Proceedings of NIPS Workshop on Nonconvex Optimization for Machine Learning: Theory and Practice*, 2016, [arXiv preprint arXiv:1612.00547].

[22] S. J. Reddi, M. Zaheer, S. Sra, B. Póczos, F. Bach, R. Salakhutdinov, and A. J. Smola, "A generic approach for escaping saddle points," in *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018, pp. 1230–1242.

[23] A. Mokhtari, A. Ozdaglar, and A. Jadbabaie, "Escaping saddle points in constrained optimization," in *Proceedings of Neural Information Processing Systems (NIPS)*, 2018.

[24] D. P. Bertsekas, *Nonlinear Programming, 2nd ed*, Athena Scientific, Belmont, MA, 1999.