PA-GD: On the Convergence of Perturbed Alternating Gradient Descent to Second-Order Stationary Points for Structured Nonconvex Optimization

Songtao Lu [1] **Mingyi Hong** [1] **Zhengdao Wang** [2]

Abstract

Alternating gradient descent (A-GD) is a simple but popular algorithm in machine learning, which updates two blocks of variables in an alternating manner using gradient descent steps. In this paper, we consider a smooth unconstrained nonconvex optimization problem, and propose a **p**erturbed **A-GD** (PA-GD) which is able to converge (with high probability) to the second-order stationary points (SOSPs) with a global sublinear rate. Existing analysis on A-GD type algorithm either only guarantees convergence to first-order solutions, or converges to second-order solutions asymptotically (without rates). To the best of our knowledge, this is the first alternating type algorithm that takes $\mathcal{O}(\text{polylog}(d)/\epsilon^2)$ iterations to achieve an $(\epsilon, \sqrt{\epsilon})$-SOSP with high probability, where $\text{polylog}(d)$ denotes the polynomial of the logarithm with respect to problem dimension d.

1. Introduction

In this paper, we consider a smooth and unconstrained nonconvex optimization problem

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta}) \tag{1}$$

where $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is twice differentiable (possibly nonconvex). Suppose $\boldsymbol{\theta}$ can be further divided into two blocks: $\boldsymbol{\theta} = [\mathbf{x} \quad \mathbf{y}]^\intercal$, with $d = d_\mathbf{x} + d_\mathbf{y}$. The block-structured equivalent form of (1) is then given by

$$\min_{\mathbf{x}\in\mathbb{R}^{d_\mathbf{x}}, \mathbf{y}\in\mathbb{R}^{d_\mathbf{y}}} f(\mathbf{x}, \mathbf{y}). \tag{2}$$

[1]Department of Electrical and Computer Engineering, University of Minnesota Twin Cities, Minneapolis, MN, USA [2]Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA. Correspondence to: Songtao Lu <lus@umn.edu>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

There are many ways of solving problem (1), such as **g**radient **d**escent (GD), **a**ccelerated **g**radient **d**escent (AGD), etc. When the problem has some naturally defined block structures, as given in (2), it is common to adopt the **a**lternating **g**radient **d**escent (A-GD) algorithm, or **b**lock **c**oordinate **g**radient **d**ecent (BC-GD). These algorithms typically solve the (smaller dimensional) subproblems in a sequential manner, and they enjoy the significant benefit of fast empirical convergence compared with GD, because they are able to use much larger stepsizes when updating each block variables (Jain et al., 2013; Xu & Yin, 2013; Zhao et al., 2015).

More specifically, many machine learning related applications have the aforementioned "block structure", such as matrix factorization (Zhao et al., 2015; Lu et al., 2017), tensor decomposition (Ge et al., 2015), matrix completion/decomposition (Xu & Yin, 2013; Jain et al., 2013). Under relatively mild conditions, the convergence of block structured algorithms such as BC-GD and A-GD to the first-order stationary points (FOSPs) have been broadly investigated (Tseng, 2001). In particular, it is known that under mild conditions, these algorithms also achieve global sublinear rates (Razaviyayn et al., 2014). However, despite the popularity of these block-structured algorithms and significant recent progress in understanding its behavior, it remains unclear whether, for generic block-structured nonconvex problems, they can converge to the set of second-order stationary points (SOSPs) with a provable global rate, even for the simplest problem with two blocks of variables.

1.1. Motivation

Algorithms that can escape from strict saddle points – those stationary points that have negative eigenvalues – have wide applications. Many recent works have analyzed the saddle points in machine learning problems (Kawaguchi, 2016). In two-layer porcupine neural networks (PNNs), it has been shown that most local optima of PNN optimizations are also global optimizers (Feizi et al., 2018). Previous work in (Ge et al., 2015) has shown that the saddle points in tensor decomposition are indeed strict saddle points. Also, it has been shown that any saddle points are strict in dictionary learning and phase retrieval problems (Sun et al., 2015). More recently, (Ge et al., 2017) proposed a unified analysis of saddle points for a broad class of low rank matrix factorization problems, and they proved that these saddle points are strict.

In nonconvex block structured problems (2), using A-GD is advantageous compared with GD because it can potentially use a much larger stepsize. Consider a simple quadratic function $f(\boldsymbol{\theta}) = \boldsymbol{\theta}^{\mathsf{T}}\mathbf{A}\boldsymbol{\theta}$ where $\mathbf{A} = [1\ a; a\ 1] \in \mathbb{R}^{2\times 2}$ and $a > 0$ is number. In this case, the block-wise gradient Lipschitz constant is $L_{\max} = 1$ and the gradient Lipschitz constant of the entire function is $L = 1 + a$ (since the largest eigenvalue of matrix $\mathbf{A}$ is $1 + a$). When $\mathbf{A}$ is semi-positive definite, problem $\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$ is convex. The difference between L and $L_{\max}$ is limited since $a \leq 1$. When $\mathbf{A}$ is indefinite, the problem becomes nonconvex and a could be very large, resulting a large difference between L and $L_{\max}$. If A-GD is used to solve this problem, the adopted stepsizes, which are inversely proportional to the block-wise Lipschitz constants, could be much larger than the one using GD. This simple example motivates us to use A-GD instead of GD to solve block structured problems. Simple numerical examples to demonstrate this phenomenon will be shown in Fig. 1, while other more complicated examples are given in Sec. 6.

1.2. Related Work

Many recent works have been focused on the performance analysis and/or design of algorithms with convergence guarantees to local minimum points/SOSPs for nonconvex optimization problems. These include the trust region method (Conn et al., 2000), cubic regularized methods (Nesterov & Polyak, 2006; Carmon & Duchi, 2016; Agarwal et al., 2017), a mixed approach of the first-order and second-order methods (Reddi et al., 2018), negative curvature methods (Goldfarb et al., 2017), **ada**ptive **n**egative **c**urvature **d**escent (AdaNCD) (Liu et al., 2018), an accelerated gradient method (Carmon et al., 2018), etc. Also, the advantages of exploiting the negative curvature compared with only using gradient descent have been shown by extensive numerical evidence in (Curtis & Robinson, 2018). However, these algorithms typically require second-order information, therefore they incur high computational complexity when problem dimension becomes large.

There has been a line of work on stochastic gradient descent algorithms, where properly scaled Gaussian noise is added to the iterates of the gradient at each time (also known as stochastic gradient Langevin dynamics (SGLD)). Some theoretical works have pointed out that SGLD not only converges to the local minimum points asymptotically but also may escape from local minima (Zhang et al., 2017). However, these algorithms require a large number of iterations with $\mathcal{O}(d^4/\epsilon^4)$ steps to achieve the optimal point. On the other hand, there is also a line of work analyzing the deterministic GD type method. With random initializations, it has been shown that GD asymptotically converges to SOSPs for unconstrained smooth problems (Lee et al., 2016). More recently, accelerated gradient descent (O'Neill & Wright, 2017), block coordinate descent, alternating minimization (Li et al., 2019), block mirror descent and proximal block coordinate descent (Lee et al., 2019; Song et al., 2017) have been proven to almost always converge to SOSPs with random initializations, but there is no convergence rate reported. Unfortunately, a follow-up study indicated that GD requires exponential time to escape from saddle points for certain pathological problems (Du et al., 2017).

Adding some noise occasionally to the iterates of the algorithm is another way of finding the negative curvature. A perturbed version of GD has been proposed with convergence guarantees to SOSPs (Jin et al., 2017), which shows a faster provable convergence rate than the ordinary gradient descent algorithm with random initializations. There are fruitful follow-up results that show some carefully designed stochastic algorithms can escape from strict saddle points efficiently, such as perturbed stochastic gradient descent (PSGD) (Jin et al., 2018a), **ne**gative-curvature-**o**riginated-from **n**oise (NEON), NEON^{+}(Xu et al., 2018), and NEON2 (Allen-Zhu & Li, 2018). Furthermore, the accelerated version of PGD is proposed in (Jin et al., 2018b), which shows the fastest convergence rate among all Hessian free algorithms.

1.3. Scope of This Paper

In this work, we consider a smooth unconstrained optimization problem, and develop a **p**erturbed **A-GD** algorithm (PA-GD) which converges (with high probability) to the set of SOSPs with a global sublinear rate. Our work is inspired by the existing works (Jin et al., 2017; Ge et al., 2015), which developed novel perturbed GD methods that can escape from strict saddle points efficiently. Similarly as in (Jin et al., 2017), we also divide the entire iterates of GD into three types of points: those whose gradients are large, those that are local minimum, and those that are strict saddle points. At a given point, when the size of the gradient is large enough, we just implement the ordinary A-GD. When the gradient norm is small, which may be either strict saddle or local minimum, a perturbation will be added on the iterates to help to escape from the saddle points.

From the above discussion, we know that many works have been developed to make use of negative curvature information around the saddle points. Unfortunately, these techniques cannot be directly applied to A-GD type of algorithms. The *key challenge* here is that at each iteration only part of the variables are updated, therefore we have

Table 1. Convergence rates of algorithms to SOSPs with the first-order information of the objective function, where $p \geq 4$, $\mathsf{poly}(\cdot)$ denotes a polynomial function and $\widetilde{\mathcal{O}}$ hides a polylogarithmic factor.

ALGORITHM	TYPE	ITERATIONS	(ϵ,γ)-SOSP
SGD (GE ET AL., 2015)	STOCHASTIC	$\mathcal{O}(d^p/\epsilon^4)$	$(\epsilon,\epsilon^{\frac{1}{4}})$
SGLD (ZHANG ET AL., 2017)	STOCHASTIC	$\mathcal{O}(d^p/\epsilon^4)$	$(\epsilon,\epsilon^{\frac{1}{2}})$
NEON+NATASHA (XU ET AL., 2018)	STOCHASTIC	$\widetilde{\mathcal{O}}(1/\epsilon^{\frac{13}{4}})$	$(\epsilon,\epsilon^{\frac{1}{4}})$
PSGD (JIN ET AL., 2018A)	STOCHASTIC	$\mathcal{O}(\mathsf{POLY}(d,1/\epsilon))$	$(\epsilon,\epsilon^{\frac{1}{2}})$
NEON2+SGD (ALLEN-ZHU & LI, 2018)	STOCHASTIC	$\widetilde{\mathcal{O}}(1/\epsilon^4)$	$(\epsilon,\epsilon^{\frac{1}{2}})$
NEON$^+$ (XU ET AL., 2018)	DETERMINISTIC	$\widetilde{\mathcal{O}}(1/\epsilon^{\frac{7}{4}})$	$(\epsilon,\epsilon^{\frac{1}{2}})$
ACCELERATED PGD (JIN ET AL., 2018B)	DETERMINISTIC	$\widetilde{\mathcal{O}}(1/\epsilon^{\frac{7}{4}})$	$(\epsilon,\epsilon^{\frac{1}{2}})$
PGD (JIN ET AL., 2017)	DETERMINISTIC	$\widetilde{\mathcal{O}}(1/\epsilon^2)$	$(\epsilon,\epsilon^{\frac{1}{2}})$
PA-GD (THIS WORK)	DETERMINISTIC	$\widetilde{\mathcal{O}}(1/\epsilon^2)$	$(\epsilon,\epsilon^{\frac{1}{2}})$

access only to partial second-order information at the points of interest. For example, consider a quadratic objective function shown in Figure 1 (Section 4.1). While fixing one block, the problem is strongly convex with respect to the other block, but the entire problem is nonconvex. Even if the iterates converge for each block to the minimum points within the block, the stationary point could still be a saddle point for the overall objective function. Therefore, the analysis of how A-GD type of algorithms exploit the negative curvature is one of the main tasks in this paper.

Table 1 gives a summary of the iteration complexity of the algorithms which only use the first-order information for escaping from strict saddle points as a literature review. To the best of our knowledge, there has been no work on using A-GD algorithms to escape from strict saddle points with provable convergence rate. The main contributions of this work are as follows.

1.4. Contributions of This Work

In this paper, we design and analyze a perturbed A-GD algorithm for solving a class of block structured unconstrained nonconvex problems. By adding the perturbation to A-GD, the algorithm is guaranteed to converge to a set of SOSPs of a nonconvex problem with high probability. By utilizing the matrix perturbation theory, convergence rate of the proposed algorithm is also established, which shows that the algorithm takes $\mathcal{O}(\text{polylog}(d)/\epsilon^2)$ iterations to achieve an $(\epsilon,\sqrt{\epsilon})$-SOSP with high probability.

The main contributions of the paper are highlighted below:
1) To the best of our knowledge, it is the first theoretical results that shows A-GD type method can converge to SOSPs for nonconvex optimization problems.

2) We show that our proposed PA-GD algorithm converges to SOSPs with a rate of $\widetilde{\mathcal{O}}(1/\epsilon^2)$; Further, the algorithm can use a larger stepsize which is inversely proportional to the maximum Lipschitz constant over the two blocks, rather than being inversely proportional to the Lipschitz constants of the entire function. This is one of the major differences between GD and A-GD.

3) By further exploiting the landscape of the block structured objective functions, we show that when PA-GD is applied in certain machine learning problems the convergence rate of PA-GD to SOSPs still holds for finding the global optimal solution of these problems.

2. Motivating Examples of This Work

From a geometric view of the loss functions in machine learning problems, there are two types of undesired critical points: (1) local minima that are not global minima; (2) saddle points. If all critical points of a function $f(\boldsymbol{\theta})$ are either global minima or strict saddle points, we say that $f(\boldsymbol{\theta})$ has *benign* landscape (Chi et al., 2018), which is the main property interested in this paper.

Matrix factorization: We consider a general low-rank matrix factorization problem as follows,

$$\underset{\mathbf{U}\in\mathbb{R}^{n\times r},\mathbf{V}\in\mathbb{R}^{m\times r}}{\text{minimize}} \; \frac{1}{2}\|\mathbf{U}\mathbf{V}^{\mathsf{T}}-\mathbf{Z}^*\|_F^2, \tag{3}$$

where $\mathbf{Z}^*\in\mathbb{R}^{n\times m}$ denotes the data matrix, $\mathbf{U}$ and $\mathbf{V}$ represent the feature matrices in the latent space, and superscript $^{\mathsf{T}}$ stands for the standard matrix/vector transpose. It is not hard to see that there is a scaling ambiguity between $\mathbf{U}$ and $\mathbf{V}$. Recent works (Zhu et al., 2018) have shown that after adding a proper regularizer, the reformulated problem will not change the global optimal solution of the original one. In order to make the notation of the function concise, let $\mathbf{W}\triangleq[\mathbf{U}\;\;\mathbf{V}]^{\mathsf{T}}$. Then, the reformulated problem is given by

$$\underset{\mathbf{U}\in\mathbb{R}^{n\times r},\mathbf{V}\in\mathbb{R}^{m\times r}}{\text{minimize}} \; \frac{1}{2}\|\mathbf{U}\mathbf{V}^{\mathsf{T}}-\mathbf{Z}^*\|_F^2+\rho(\mathbf{U},\mathbf{V}), \tag{4}$$

where $\rho(\mathbf{U},\mathbf{V}) \triangleq \frac{\nu}{4}\|\mathbf{U}^\mathsf{T}\mathbf{U} - \mathbf{V}^\mathsf{T}\mathbf{V}\|_F^2,\ \nu > 0$. This regularizer is able to enforce the size difference between $\mathbf{U}$ and $\mathbf{V}$ to be as small as possible.

The matrix factorization problem has a wide application in areas of machine learning and signal processing; see a recent survey paper (Chen & Chi, 2018) and the references therein. For example, the objective functions of the following two interesting examples also have this *benign* landscape.

Matrix sensing: One popular formulation of the matrix sensing problem is given by:

$$\underset{\mathbf{U}\in\mathbb{R}^{n\times r},\mathbf{V}\in\mathbb{R}^{m\times r}}{\text{minimize}} \ \frac{1}{2}\|\mathcal{A}(\mathbf{U}\mathbf{V}^\mathsf{T} - \mathbf{Z}^*)\|^2 + \rho(\mathbf{U},\mathbf{V}), \tag{5}$$

where the mapping $\mathcal{A}(\cdot)$ satisfies the restricted isometry property (Recht et al., 2010).

Two-layer linear neural networks: given a set of data points $\{\widehat{\boldsymbol{x}}_i, \widehat{\boldsymbol{y}}_i\}_{i=1}^m$ of size m, we wish to fit a two-layer linear network using the quadratic loss as follows,

$$\underset{\mathbf{U}\in\mathbb{R}^{n\times r},\mathbf{V}\in\mathbb{R}^{m\times r}}{\text{minimize}} \sum_{i=1}^{k}\|\widehat{\boldsymbol{y}}_i - \mathbf{U}\mathbf{V}^\mathsf{T}\widehat{\boldsymbol{x}}_i\|_2^2 = \|\widehat{\boldsymbol{Y}} - \mathbf{U}\mathbf{V}^\mathsf{T}\widehat{\boldsymbol{X}}\|_F^2, \tag{6}$$

where $\widehat{\boldsymbol{X}} \triangleq [\widehat{\boldsymbol{x}}_1,\ldots,\widehat{\boldsymbol{x}}_k] \in \mathbb{R}^{m\times k}$ denotes the data matrix and $\widehat{\boldsymbol{Y}} \triangleq [\widehat{\boldsymbol{y}}_1,\ldots,\widehat{\boldsymbol{y}}_k] \in \mathbb{R}^{n\times k}$ represents the label matrix.

3. Definition and Assumption

The objective function has the following properties.

Definition 1. *A differentiable function $f(\cdot)$ is L-smooth with gradient Lipschitz constant L (uniformly Lipschitz continuous), if*

$$\|\nabla f(\boldsymbol{\theta}) - \nabla f(\boldsymbol{\theta}')\| \le L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|, \quad \forall \boldsymbol{\theta}, \boldsymbol{\theta}'.$$

The function is block-wise smooth with gradient Lipschitz constant $L_\mathbf{x}$, that is:

$$\|\nabla_\mathbf{x} f(\mathbf{x},\mathbf{y}) - \nabla_\mathbf{x} f(\mathbf{x}',\mathbf{y})\| \le L_\mathbf{x}\|\mathbf{x} - \mathbf{x}'\|,\ \forall \mathbf{x},\mathbf{x}',\mathbf{y}$$

or with gradient Lipschitz constant $L_\mathbf{y}$, that is:

$$\|\nabla_\mathbf{y} f(\mathbf{x},\mathbf{y}) - \nabla_\mathbf{y} f(\mathbf{x},\mathbf{y}')\| \le L_\mathbf{y}\|\mathbf{y} - \mathbf{y}'\|,\ \forall \mathbf{y},\mathbf{y}',\mathbf{x}.$$

Further, let $L_{\max} \triangleq \max\{L_\mathbf{x}, L_\mathbf{y}\}$. It can be easily shown that $L_{\max} \le L$. In most block structured problems, we have $L_{\max} \ll L$.

Definition 2. *For a differentiable function $f(\cdot)$, if $\|\nabla f(\boldsymbol{\theta})\| = 0$, then $\boldsymbol{\theta}$ is a FOSP. If $\|\nabla f(\boldsymbol{\theta})\| \le \epsilon$, then $\boldsymbol{\theta}$ is an ϵ-FOSP.*

Definition 3. *If $\|\nabla f(\boldsymbol{\theta})\| = 0$ and $\lambda_{\min}(\nabla^2 f(\boldsymbol{\theta})) < 0$, $\boldsymbol{\theta}$ is a strict (non-degenerate) saddle point, where $\lambda_{\min}(\nabla^2 f(\boldsymbol{\theta}))$ denotes the minimum eigenvalue of matrix $\nabla^2 f(\boldsymbol{\theta})$.*

Remark 1. For a differentiable function $f(\cdot)$, if $\boldsymbol{\theta}$ is a FOSP, and there exists $\epsilon > 0$ so that for any $\boldsymbol{\theta}'$ in the ϵ-neighborhood of $\boldsymbol{\theta}$, we have $f(\boldsymbol{\theta}) \le f(\boldsymbol{\theta}')$, then $\boldsymbol{\theta}$ is a local minimum. A saddle point $\boldsymbol{\theta}$ is a FOSP that is not a local minimum.

Definition 4. *A twice-differentiable function $f(\cdot)$ is ρ-Hessian Lipschitz if*

$$\|\nabla^2 f(\boldsymbol{\theta}) - \nabla^2 f(\boldsymbol{\theta}')\| \le \rho\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|, \quad \forall \boldsymbol{\theta}, \boldsymbol{\theta}'. \tag{7}$$

Definition 5. *For a ρ-Hessian Lipschitz function $f(\cdot)$, $\boldsymbol{\theta}$ is a SOSP if $\|\nabla f(\boldsymbol{\theta})\| = 0$ and $\lambda_{\min}(\nabla^2 f(\boldsymbol{\theta})) \ge 0$. If the following holds*

$$\|\nabla f(\boldsymbol{\theta})\| \le \epsilon, \quad \text{and} \quad \lambda_{\min}(\nabla^2 f(\boldsymbol{\theta})) \ge -\gamma \tag{8}$$

where $\epsilon, \gamma > 0$, then $\boldsymbol{\theta}$ is an (ϵ,γ)-SOSP.

Assumption 1. *Function $f(\cdot)$ is L-smooth, block-wise smooth with gradient Lipschitz constants $L_\mathbf{x}, L_\mathbf{y}$, and ρ-Hessian Lipschitz.*

4. Perturbed Alternating Gradient Descent

4.1. Algorithm Description

A-GD is a classical algorithm that optimizes the variables of an optimization problem in an alternating manner (Bertsekas, 1999), meaning that when one block of variables is updated, the remaining block is fixed to be the same as its previous solution. Mathematically, the iterates of A-GD are updated by the following rule

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta\nabla_\mathbf{x} f(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}), \tag{9a}$$

$$\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)} - \eta\nabla_\mathbf{y} f(\mathbf{x}^{(t+1)}, \mathbf{y}^{(t)}) \tag{9b}$$

where $\boldsymbol{\theta} \triangleq [\mathbf{x}\ \mathbf{y}]^\mathsf{T}$ is partitioned by two blocks, superscript (t) denotes the iteration counter and $\eta \le 1/L$ stands for the stepsize. Our proposed algorithm is based on A-GD, but modified in a way similar to the recent work (Jin et al., 2017), which adds some noise in PGD.

The details of the implementation of PA-GD are shown in Algorithm 1, where Δ_f denotes the difference of the objective value at the initial point and global optimal solution, ϵ represents the predefined target error, δ is the predefined target probability, and more definitions of the parameters used in Algorithm 1 are shown in Table 2. The algorithm works in the following way:

1) Algorithm 1 just implements the ordinary A-GD before the iterates achieve an approximate FOSP, which is defined by the condition: $\|\nabla_\mathbf{x} f(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})\|^2 + \|\nabla_\mathbf{y} f(\mathbf{x}^{(t+1)}, \mathbf{y}^{(t)})\|^2 \le g_{\mathsf{th}}^2$;

2) When some additional conditions are satisfied, that is, the algorithm has waited long enough from the last time that any noise is added (indicated by the condition $t - t_\mathsf{p} >$

Table 2. Definition of Parameters in Algorithm 1.

PARAMETERS	EXPRESSION/RANGE
η	$(0, 1/L]$
p_0	$\max\left\{6, 4(L/L_{\max})^2 + 1\right\}$
p_1	$1 + L/L_{\max}$
p_2	$1 + L\log(4d)/(2L_{\max})$
$\widehat{c}$	$\min\{136, 8(3p_0^2 + 12p_0 + 12)\}$
δ	$(0, 1]$
ϵ	$(0, L_{\max}^2/\rho]$
χ	$\max\{\log(\frac{\widehat{c}^5 p_1^3 p_2^2 d L_{\max}\Delta_f}{\epsilon^2\delta}), 4\}$
g_{TH}	$\frac{\epsilon}{(\chi p_1)^2 p_2}$
f_{TH}	$\frac{\sqrt{\epsilon^3/\rho}}{\widehat{c}^5(\chi p_1)^3 p_2^2}$
t_{TH}	$\frac{\widehat{c} L_{\max}\chi p_1}{\sqrt{\rho\epsilon}} + 3$
r	$\frac{\rho\epsilon}{L_{\max}\widehat{c}^5(\chi p_1)^2 p_2}$

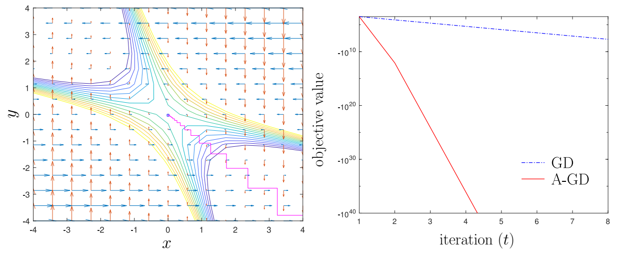

Figure 1. The objective function is $f(\boldsymbol{\theta}) = \boldsymbol{\theta}^\mathsf{T}\mathbf{A}\boldsymbol{\theta}, \boldsymbol{\theta} = [x\ y]^\mathsf{T} \in \mathbb{R}^{2\times 1}$. (left) Contour of the objective values and the trajectory (pink color) of PA-GD started near strict saddle point $[0, 0]$, where $\mathbf{A} = [1\ 2; 2\ 1] \in \mathbb{R}^{2\times 2}$, and the length of the arrows indicate the strength of $-\nabla f(\boldsymbol{\theta})$ projected onto directions x, y respectively. (right) Convergence rate comparison between GD and A-GD with random initialization, where $\mathbf{A} = [1\ \ 1000; 1000\ \ 1]$, the stepsize of A-GD is $1/L_{\max} = 1$, and the stepsize of GD is $1/L = 1/1001$.

t_{th} in Algorithm 1), we add some random noise uniformly taken from $\mathbb{B}_0(r)$ (a d-dimensional ball centered at 0 with radius r);

3) If the objective value is not decreased sufficiently after adding the perturbation for t_{th} steps (specifically, not decreased by at least f_{th}), then Algorithm 1 stops and returns an approximate SOSP solution.

It is worth noting that the structure of the algorithm is similar to the PGD in (Jin et al., 2017). The perturbation at certain iterates plays an important role in showing that the iterates converge to SOSPs. The key difference between PGD and PA-GD lies in the way of updating variables. In each update of variables, we implement one step of gradient descent on block $\mathbf{x}$, and then proceed to block $\mathbf{y}$. Once the algorithm has sufficient decrease of the objective value, it implies that the algorithm converges to some good solution. Otherwise, some perturbation may be needed to help the iterates escape from the saddle

Algorithm 1 Perturbed Alternating Gradient Descent

```
Input: $\boldsymbol{\theta}^{(1)}, L_{\max}, L, \eta = 1/L_{\max}, \rho, \epsilon, \delta, \Delta_f$
for $t = 1, \ldots$ do
    Update $\mathbf{x}^{(t+1)}$ via (9a).
    if $\|\nabla_\mathbf{x} f(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})\|^2 + \|\nabla_\mathbf{y} f(\mathbf{x}^{(t+1)}, \mathbf{y}^{(t)})\|^2 \le g_{\mathsf{th}}^2$
    and $t - t_{\mathsf{p}} > t_{\mathsf{th}}$ then
        $\widetilde{\boldsymbol{\theta}}^{(t)} \leftarrow \boldsymbol{\theta}^{(t)}$ and $t_{\mathsf{p}} \leftarrow t$
        $\boldsymbol{\theta}^{(t)} = \widetilde{\boldsymbol{\theta}}^{(t)} + \xi^{(t)}$        ◁ Add perturbation
        Update $\mathbf{x}^{(t+1)}$ via (9a) again.
    end if
    Update $\mathbf{y}^{(t+1)}$ via (9b).
    if $t - t_{\mathsf{p}} > t_{\mathsf{th}}$ and $f(\boldsymbol{\theta}^{(t)}) - f(\widetilde{\boldsymbol{\theta}}^{(t_{\mathsf{p}})}) > -f_{\mathsf{th}}$ then
        return $\widetilde{\boldsymbol{\theta}}^{(t_{\mathsf{p}})}$
    end if
end for
```

points. If after the perturbation the objective value does not decrease sufficiently after a number of further iterations, the algorithm terminates and returns the iterate before the last perturbation.

To illustrate the practical behavior of the algorithm, we provide an example that shows the trajectory of A-GD after a small perturbation at a stationary point. In Figure 1, it is clear that $\boldsymbol{\theta} = [0\ \ 0]^\mathsf{T}$ is a FOSP and also a strict saddle point since the eigenvalues of $\mathbf{A}$ are -1 and 3 respectively. When x is fixed, function $f(\boldsymbol{\theta})$ is convex with respect to y and vice versa, however, the objective function is nonconvex. It can be observed that PA-GD can escape from the strict saddle point efficiently while A-GD will not move at $\boldsymbol{\theta} = [0\ 0]^\mathsf{T}$ without perturbation.

4.2. Convergence Rate Analysis

Despite the fact that PA-GD exploits a different way of updating variables compared with GD, SGD, and their variants, we will show that it can still escape from strict saddle points with high probability with suitable perturbation. The main theorem is presented as follows.

Theorem 1. *Under Assumption 1, there exists a constant* $\widehat{c} \ge \min\{136, 8(3p_0^2 + 12p_0 + 12)\}$ *such that: for any* $\eta \le 1/L_{\max}$, $\delta \in (0, 1]$, $\epsilon \le \frac{L_{\max}^2}{\rho}$, *and* $\Delta_f \triangleq f(\boldsymbol{\theta}^{(1)}) - f^\star$, *with probability* $1 - \delta$, *the iterates generated by PA-GD converge to an* $(\epsilon, \sqrt{\epsilon})$*-SOSP* $\boldsymbol{\theta}$ *satisfying*

$$\|\nabla f(\boldsymbol{\theta})\| \le \epsilon, \quad \text{and} \quad \lambda_{\min}(\nabla^2 f(\boldsymbol{\theta})) \ge -\sqrt{\rho\epsilon}$$

in the following number of iterations:

$$\mathcal{O}\left(\frac{\widehat{c}^6 L_{\max} p_1^4 p_2^2 \Delta_f}{\epsilon^2} \log^4\left(\frac{\widehat{c}^5 p_1^3 p_2^2 d L_{\max}\Delta_f}{\epsilon^2\delta}\right)\right) \tag{10}$$

where $f^\star$ *denotes the global minimum value of the objective function.*

Remark 2. When $\eta = 1/L$ is used, the ratio $L/L_{\max}$ becomes 1, so the convergence rate of PA-GD becomes $\mathcal{O}\left(\frac{L\Delta_f \log^2(4d)}{\epsilon^2}\log^4\left(\frac{dL\Delta_f}{\epsilon^2\delta}\right)\right)$. This result shows that if a smaller stepsize is used, the worst-case convergence rate of PA-GD is faster (with smaller constants) . This property is consistent with the known result when block coordinate descent is used in convex optimization problems, e.g., see (Sun & Hong, 2015, Theorem 2.1). Also, in this case the convergence rate is reduced to the one of PGD up to some logarithmic factors, although the analysis is still different.

5. Convergence Analysis

In this section, we will present the main proof steps of convergence analysis of PA-GD.

5.1. The Main Difficulty of the Proof

First, let us understand the challenge in the analysis of PA-GD compared with that of P-GD.

Gradient Descent: GD searches the descent direction of the objective function in the entire space $\mathbb{R}^d$. Without loss of generality, we assume $\boldsymbol{\theta}^{(1)} = 0$. According to the mean value theorem, the GD update can be expressed as

$$\begin{aligned}\boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \eta\nabla f(\boldsymbol{\theta}^{(t)})\\ &= \boldsymbol{\theta}^{(t)} - \eta\nabla f(0) - \eta\left(\int_0^1 \nabla^2 f(\alpha\boldsymbol{\theta}^{(t)})d\alpha\right)\boldsymbol{\theta}^{(t)} \quad (11)\end{aligned}$$

where $0 < \alpha < 1$. It can be observed that the update rule of GD contains the information of the Hessian matrix at point $\boldsymbol{\theta}^{(t)}$ in the integral, i.e., $\nabla^2 f(\alpha\boldsymbol{\theta}^{(t)})$. To be more specific, letting $\boldsymbol{\mathcal{H}} \triangleq \nabla^2 f(\boldsymbol{\theta}^*)$ where $\boldsymbol{\theta}^*$ denotes a saddle point, we can rewrite (11) as

$$\boldsymbol{\theta}^{(t+1)} = (\mathbf{I} - \eta\boldsymbol{\mathcal{H}})\boldsymbol{\theta}^{(t)} - \eta\Delta^{(t)}\boldsymbol{\theta}^{(t)} - \eta\nabla f(0) \quad (12)$$

where $\Delta^{(t)} \triangleq \int_0^1(\nabla^2 f(\alpha\boldsymbol{\theta}^{(t)}) - \boldsymbol{\mathcal{H}})d\alpha$.

Based on the ρ-Hessian Lipschitz property, we can show that $\|\Delta^{(t)}\|$ is upper bounded by the difference of iterates. By exploiting the negative curvature of the Hessian matrix at saddle point $\boldsymbol{\theta}^*$, we can project the iterate onto the direction where the eigenvalue of $\mathbf{I} - \eta\boldsymbol{\mathcal{H}}$ is greater than 1. This leads to the fact that the norm of the iterates projected along this direction will be increasing exponentially as the algorithm proceeds around point $\boldsymbol{\theta}^*$, implying that the sequence generated by GD is able to escape from the saddle point efficiently. The details of characterizing the convergence rate have been analyzed previously in (Jin et al., 2017).

Alternating Gradient Descent: However, the A-GD algorithm only updates partial variables of vector $\boldsymbol{\theta}$. Similarly, from the mean value theorem we can express the A-GD rule of updating variables with assuming $\boldsymbol{\theta}^{(1)} = 0$ as follows:

$$\begin{aligned}\boldsymbol{\theta}^{(t+1)} =& \boldsymbol{\theta}^{(t)} - \eta\begin{bmatrix}\nabla_{\mathbf{x}} f(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})\\ \nabla_{\mathbf{y}} f(\mathbf{x}^{(t+1)}, \mathbf{y}^{(t)})\end{bmatrix}\\ =& \boldsymbol{\theta}^{(t)} - \eta\nabla f(0) - \eta\int_0^1 \boldsymbol{\mathcal{H}}_l^{(t)}(\alpha)d\alpha\boldsymbol{\theta}^{(t+1)}\\ &- \eta\int_0^1 \boldsymbol{\mathcal{H}}_u^{(t)}(\alpha)d\alpha\boldsymbol{\theta}^{(t)} \quad (13)\end{aligned}$$

where

$$\boldsymbol{\mathcal{H}}_l^{(t)}(\alpha) \triangleq \begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \nabla^2_{\mathbf{yx}} f(\alpha\mathbf{x}^{(t+1)}, \alpha\mathbf{y}^{(t)}) & \mathbf{0}\end{bmatrix},$$

$$\boldsymbol{\mathcal{H}}_u^{(t)}(\alpha) \triangleq \begin{bmatrix}\nabla^2_{\mathbf{xx}} f(\alpha\mathbf{x}^{(t)}, \alpha\mathbf{y}^{(t)}) & \nabla^2_{\mathbf{xy}} f(\alpha\mathbf{x}^{(t)}, \alpha\mathbf{y}^{(t)})\\ \mathbf{0} & \nabla^2_{\mathbf{yy}} f(\alpha\mathbf{x}^{(t+1)}, \alpha\mathbf{y}^{(t)})\end{bmatrix}.$$

From the above expression, it can be seen clearly that the update rule of A-GD does not include a full Hessian matrix at any point but only partial ones.

Furthermore, the right hand side of (13) not only contains the second-order information of the previous point, i.e., $[\mathbf{x}^{(t)}, \mathbf{y}^{(t)}]$ but also the one of the most recently updated point, i.e., $[\mathbf{x}^{(t+1)}, \mathbf{y}^{(t)}]$. These features of A-GD make the convergence analysis fundamentally different from the GD-and/or SGD-type of algorithms. The split block structure of the second-order information of the objective function and dependence between the two blocks represent the main challenges in understanding the behavior of the sequence generated by the A-GD algorithm.

5.2. The Sketch of the Proof

First, we split $\boldsymbol{\mathcal{H}}$ as two parts, which are

$$\boldsymbol{\mathcal{H}}_u = \begin{bmatrix}\nabla^2_{\mathbf{xx}} f(\boldsymbol{\theta}^*) & \nabla^2_{\mathbf{xy}} f(\boldsymbol{\theta}^*)\\ \mathbf{0} & \nabla^2_{\mathbf{yy}} f(\boldsymbol{\theta}^*)\end{bmatrix}, \quad (14a)$$

$$\boldsymbol{\mathcal{H}}_l = \begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \nabla^2_{\mathbf{yx}} f(\boldsymbol{\theta}^*) & \mathbf{0}\end{bmatrix}, \quad (14b)$$

and obviously we have $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{H}}_l + \boldsymbol{\mathcal{H}}_u$.

Then, recursion (13) can be written as

$$\begin{aligned}&\boldsymbol{\theta}^{(t+1)} + \eta\boldsymbol{\mathcal{H}}_l\boldsymbol{\theta}^{(t+1)}\\ &= \boldsymbol{\theta}^{(t)} - \eta\boldsymbol{\mathcal{H}}_u\boldsymbol{\theta}^{(t)} - \eta\Delta_u^{(t)}\boldsymbol{\theta}^{(t)} - \eta\Delta_l^{(t)}\boldsymbol{\theta}^{(t+1)} \quad (15)\end{aligned}$$

where $\Delta_u^{(t)} \triangleq \int_0^1(\boldsymbol{\mathcal{H}}_u^{(t)}(\alpha) - \boldsymbol{\mathcal{H}}_u)d\alpha$, $\Delta_l^{(t)} \triangleq \int_0^1(\boldsymbol{\mathcal{H}}_l^{(t)}(\alpha) - \boldsymbol{\mathcal{H}}_l)d\alpha$. However, it is still unclear from (15) how the iteration evolves around the strict saddle point.

To highlight ideas, let us define

$$\mathbf{M} \triangleq \mathbf{I} + \eta\boldsymbol{\mathcal{H}}_l, \quad \mathbf{T} \triangleq \mathbf{I} - \eta\boldsymbol{\mathcal{H}}_u. \quad (16)$$

It can be observed that $\mathbf{M}$ is a lower triangular matrix where the diagonal entries are all 1s; therefore it is invertible. After taking the inverse of matrix $\mathbf{M}$ on both sides of (15), we can obtain

$$\boldsymbol{\theta}^{(t+1)} = \mathbf{M}^{-1}\mathbf{T}\boldsymbol{\theta}^{(t)} - \eta\mathbf{M}^{-1}\Delta_u^{(t)}\boldsymbol{\theta}^{(t)} - \eta\mathbf{M}^{-1}\Delta_l^{(t)}\boldsymbol{\theta}^{(t+1)}.$$

Our goal of analyzing the recursion of $\boldsymbol{\theta}^{(t)}$ becomes to find the maximum eigenvalue of $\mathbf{M}^{-1}\mathbf{T}$. With the help of the matrix perturbation theory, we can quantify the difference between the eigenvalues of matrix $\boldsymbol{\mathcal{H}}$ that contains the negative curvature and matrix $\mathbf{M}^{-1}\mathbf{T}$ that we are interested in analyzing. To be more precise, we give the following lemma.

Lemma 1. *Under Assumption 1, let $\boldsymbol{\mathcal{H}} \triangleq \nabla^2 f(\boldsymbol{\theta})$ denote the Hessian matrix at $\boldsymbol{\theta}$ where $\lambda_{\min}(\boldsymbol{\mathcal{H}}) \le -\gamma$ and $\gamma > 0$. We have*

$$\lambda_{\max}(\mathbf{M}^{-1}\mathbf{T}) > 1 + \frac{\eta\gamma}{1 + \frac{L}{L_{\max}}} \tag{17}$$

where $\mathbf{M}, \mathbf{T}$ are defined in (16) and $\lambda_{\max}(\mathbf{M}^{-1}\mathbf{T})$ denotes the maximum eigenvalue of matrix $\mathbf{M}^{-1}\mathbf{T}$.

Lemma 1 illustrates that there exits a subspace spanned by the eigenvector of $\mathbf{M}^{-1}\mathbf{T}$ whose eigenvalue is greater than 1, indicating that the sequence generated by A-GD can still potentially escape from the strict saddle point by leveraging such negative curvature information. Next, we can give a sketch of the proof of Theorem 1.

PA-GD updates the variables block by block, so we have to provide the new proofs different from the previous work in (Jin et al., 2017) to show that PA-GD can still escape from saddle points with the perturbation technique.

First, if the size of the gradient is large enough, Algorithm 1 just implements the ordinary A-GD. We give the descent lemma of A-GD as follows.

Lemma 2. *Under Assumption 1, for the A-GD algorithm with step-size $\eta \le 1/L_{\max}$, we have*

$$f(\boldsymbol{\theta}^{(t+1)}) \le f(\boldsymbol{\theta}^{(t)}) - \frac{\eta}{2}\left(\|\nabla_{\mathbf{x}} f(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})\|^2 + \|\nabla_{\mathbf{y}} f(\mathbf{x}^{(t+1)}, \mathbf{y}^{(t)})\|^2\right).$$

Second, if the iterates are close to a strict saddle point, we can show that the A-GD algorithm after a perturbation can give a sufficient objective decrease with high probability in terms of the objective value. To be more precise, the statement is given as follows.

Lemma 3. *Under Assumption 1, let $\chi \ge 1$, and η, $\widehat{c}$, r, g_{th}, t_{th} calculated as Algorithm 1 describes. Let $\widetilde{\boldsymbol{\theta}}^{(t)} \triangleq [\widetilde{\mathbf{x}}^{(t)}\ \widetilde{\mathbf{y}}^{(t)}]^{\mathsf{T}}$ be a strict saddle point, which satisfies*

$$\|\nabla f(\widetilde{\boldsymbol{\theta}}^{(t)})\|^2 \le \left(1 + 2\left(\frac{L}{L_{\max}}\right)^2\right)\left(\|\nabla_{\widetilde{\mathbf{x}}} f(\widetilde{\mathbf{x}}^{(t)}, \widetilde{\mathbf{y}}^{(t)})\|^2 + \|\nabla_{\widetilde{\mathbf{y}}} f(\widetilde{\mathbf{x}}^{(t+1)}, \widetilde{\mathbf{y}}^{(t)})\|^2\right) < 2p_1^2 g_{th}^2 < \epsilon \tag{18}$$

and $\lambda_{\min}(\nabla^2 f(\widetilde{\boldsymbol{\theta}}^{(t)})) \le -\gamma$.

Let $\boldsymbol{\theta}^{(t)} \triangleq \widetilde{\boldsymbol{\theta}}^{(t)} + \xi^{(t)}$ where $\xi^{(t)}$ is generated randomly which follows the uniform distribution over $\mathbb{B}_0(r)$, and let $\boldsymbol{\theta}^{(t+t_{th})}$ be the iterates of PA-GD. With at least probability $1 - \frac{dL_{\max}}{\sqrt{\rho\epsilon}}e^{-\chi}$, we have $f(\boldsymbol{\theta}^{(t+t_{th})}) - f(\widetilde{\boldsymbol{\theta}}^{(t)}) \le -f_{th}$.

We remark that Lemma 2 is well-known, while the main challenges lie in proving Lemma 3. In the following, we outline the main idea used in proving the latter. The formal statements of these steps are shown in the appendix; see Lemma 7–Lemma 9 therein.

We emphasize that the main contributions of this paper lies in the analysis of the first two steps, where the special update rule of PA-GD is analyzed so that the negative curvature of $\boldsymbol{\mathcal{H}}$ around the saddle points can be utilized.

Step 1 (Lemma 7) Consider a generic sequence $\mathbf{u}^{(t)}$ generated by PA-GD. We characterize the relation between the distance between $\mathbf{u}^{(t)}$ and $\widetilde{\boldsymbol{\theta}}^{(t)}$ and the decrease of the objective value.

Step 2 (Lemma 8) Consider two generic sequences generated by PA-GD, $\mathbf{u}^{(t)}, \mathbf{w}^{(t)}$ initialized around the saddle point. Leveraging the negative curvature around the strict saddle point, by Lemma 1 we know that there exits a direction, i.e., $\vec{\mathbf{e}}$, which is spanned by the eigenvector of $\mathbf{M}^{-1}\mathbf{T}$ whose corresponding eigenvalue is greater than 1. When the initial points of these two iterates are separated apart away from each other along direction $\vec{\mathbf{e}}$ with a small distance, meaning that $\mathbf{w}^{(1)} = \mathbf{u}^{(1)} + \upsilon r\vec{\mathbf{e}}$, $\upsilon \in [\delta/(2\sqrt{d}), 1]$, we can show that if iterate $\mathbf{u}^{(t_{th})}$ is still near the saddle point, the other sequence $\mathbf{w}^{(t)}$ will give a sufficient decrease of the objective value with less than t_{th} steps, implying that iterates $\mathbf{w}^{(t)}$ can escape from the saddle point with less than t_{th} steps.

Step 3 (Lemma 9) We can quantify the probability that

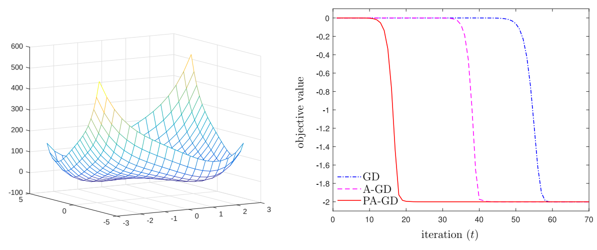

Figure 2. Convergence comparison between A-GD and PA-GD, where $\epsilon = 10^{-4}$, $g_{th} = \epsilon/10$, $\eta = 0.02$, $t_{th} = 10/\sqrt{\epsilon}$, $r = \epsilon/10$.

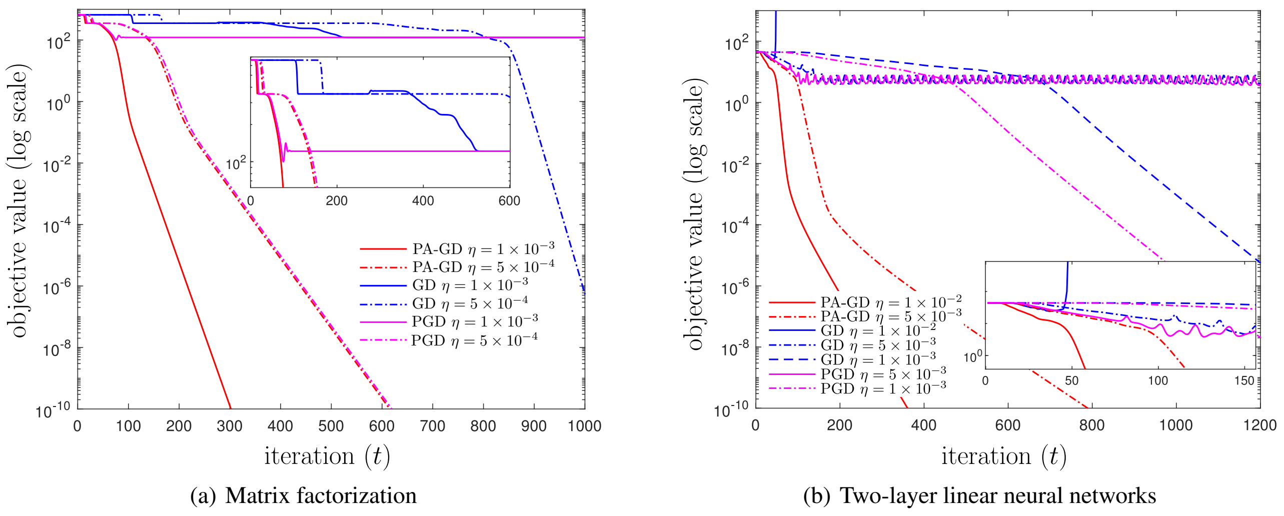

Figure 3. Convergence comparison among GD, PGD and PA-GD, where $\epsilon = 10^{-10}$, $g_{\text{th}} = \epsilon/10$, $t_{\text{th}} = 10/\sqrt{\epsilon}$, $r = \epsilon/10$.

the A-GD sequence will give a sufficient decrease of the objective value within T iterations after the perturbation (Jin et al., 2017, Lemma 14,15).

6. Numerical Results

In this section, we will use several numerical results to illustrate the effectiveness of the proposed algorithm on escaping strict saddle points.

6.1. A toy example

First, we present a simple example that shows there are two equal local optimal solutions, i.e., the global optimal points. Consider a nonconvex objective function, i.e.,

$$f(\boldsymbol{\theta}) = \boldsymbol{\theta}^\intercal \mathbf{A}\boldsymbol{\theta} + \frac{1}{4}\|\boldsymbol{\theta}\|_4^4, \tag{19}$$

where $\boldsymbol{\theta} = [x\ y]^\intercal$. Here, we can easily show the shape of objective function (19) in the two dimensional (2D) case in Figure 2 (left), where $\mathbf{A} = [1\ 2; 2\ 1] \in \mathbb{R}^{2\times 2}$. It can be observed clearly that there exists a strict saddle point at $[0\ 0]^\intercal$ and two other local optimal points. We randomly initialize the algorithms around strict saddle point $[0\ 0]^\intercal$. The convergence comparison between GD and PA-GD is shown in Figure 2 (right). It can be observed that PA-GD converges faster than GD to the global optimal point. Note that if we initialize the iterates exactly at the origin, GD will not move but PA-GD can still converge to the global optimal solution.

6.2. Matrix factorization

We test the algorithms for the problem of matrix factorization. We randomly generate matrix $\mathbf{Z}^* = \mathbf{U}^*(\mathbf{V}^*)^\intercal$ with dimension $n = 800, m = 200, r = 10$ and initialize GD, PGD and PA-GD around saddle point 0. The step-sizes of the GD, PGD, PA-GD algorithms are denoted as η. All perturbation related parameters of PA-GD and PGD, e.g., $g_{\text{th}}, t_{\text{th}}, r$, are the same for fair comparison at some extent. Figure 3(a) shows the superiority of PA-GD in the matrix factorization problem. When the step-size is large, GD and PGD cannot decrease the objective value monotonically or sufficiently but PA-GD can, since the regularizer (or step-size) of PA-GD depends on $L_{\max}$ rather than L. PA-GD and PGD converge faster since the negative curvature can be captured by adding the random noise. Also, it can be observed that PA-GD converges to the global optimal solution of this problem.

6.3. Two-layer linear neural networks

We also implement the algorithms in the application of two-layer linear neural networks, where the data matrices are randomly generated with dimension $n = 100, m = 40, r = 20, k = 20$. The other settings are the same as the above section in solving the matrix factorization problem. In can be observed from Figure 3(b) that PA-GD converges faster compared with other ones by the largest stepsize to the global optimal solution of this problem. When GD uses a large stepsize, e.g., $\eta = 1 \times 10^{-2}$, it will diverge, which is the reason why PA-GD is preferred.

7. Concluding Remarks

In this paper, a perturbed A-GD algorithm is proposed, with the objective of finding SOSPs of nonconvex smooth problems. The main contribution of this work is a new analysis that takes into consideration the block structure of the updates for the perturbed A-GD algorithm. By exploiting the negative curvature, it is established that with high probability the algorithms can converge to an $(\epsilon, \sqrt{\epsilon})$-SOSP with $\widetilde{\mathcal{O}}(1/\epsilon^2)$ iterations.

References

Agarwal, N., Allen-Zhu, Z., Bullins, B., Hazan, E., and Ma, T. Finding approximate local minima faster than gradient descent. In *Proceedings of Annual ACM Symposium on the Theory of Computing (STOC)*, pp. 1195–1199, 2017.

Allen-Zhu, Z. and Li, Y. Neon2: Finding local minima via first-order oracles. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2018.

Angelos, J. R., Cowen, C. C., and Narayan, S. K. Triangular truncation and finding the norm of a Hadamard multiplier. *Linear Algebra and its Applications*, 170:117–135, 1992.

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

Carmon, Y. and Duchi, J. C. Gradient descent efficiently finds the cubic-regularized non-convex Newton step. *arXiv preprint arXiv:1612.00547*, 2016.

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. Accelerated methods for nonconvex optimization. *SIAM Journal on Optimization*, 28(2):1751–1772, 2018.

Chen, Y. and Chi, Y. Harnessing structures in big data via guaranteed low-rank matrix estimation: Recent theory and fast algorithms via convex and nonconvex optimization. *IEEE Signal Processing Magazine*, 35(4): 14–31, July 2018.

Chi, Y., Lu, Y. M., and Chen, Y. Nonconvex optimization meets low-rank matrix factorization: An overview. *arXiv:1809.09573*, 2018.

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

Curtis, F. E. and Robinson, D. P. Exploiting negative curvature in deterministic and stochastic optimization. *Mathematical Programming*, 2018.

Du, S. S., Jin, C., Lee, J. D., Jordan, M. I., Póczos, B., and Singh, A. Gradient descent can take exponential time to escape saddle points. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2017.

Feizi, S., Javadi, H., Zhang, J., and Tse, D. Porcupine neural networks: Approximating neural network landscapes. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2018.

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

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

Goldfarb, D., Mu, C., Wright, J., and Zhou, C. Using negative curvature in solving nonlinear programs. *Computational Optimization and Applications*, 68(3): 479–502, Dec 2017.

Holbrook, J. A. Spectral variation of normal matrices. *Linear Algebra and its Applications*, 174:131–144, 1992.

Jain, P., Netrapalli, P., and Sanghavi, S. Low-rank matrix completion using alternating minimization. In *Proceedings of Annual ACM Symposium on the Theory of Computing (STOC)*, pp. 665–674, 2013.

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

Jin, C., Liu, L. T., Ge, R., and Jordan, M. I. On the local minima of the empirical risk. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 4901–4910. 2018a.

Jin, C., Netrapalli, P., and Jordan, M. I. Accelerated gradient descent escapes saddle points faster than gradient descent. In *Proceedings of Annual Conference on Learning Theory (COLT)*, 2018b.

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

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

Lee, J. D., Panageas, I., Piliouras, G., Simchowitz, M., Jordan, M. I., and Recht, B. First-order methods almost always avoid saddle points. *Mathematical Programming, Series B*, 2019.

Li, Q., Zhu, Z., and Tang, G. Alternating minimizations converge to second-order optimal solutions. In *Proceedings of International Conference on Machine Learning (ICML)*, 2019.

Liu, M., Li, Z., Wang, X., Yi, J., and Yang, T. Adaptive negative curvature descent with applications in non-convex optimization. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 4854–4863, 2018.

Lu, S., Hong, M., and Wang, Z. A stochastic nonconvex splitting method for symmetric nonnegative matrix factorization. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54, pp. 812–821, 20–22 Apr. 2017.

Nesterov, Y. and Polyak, B. T. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

O'Neill, M. and Wright, S. J. Behavior of accelerated gradient methods near critical points of nonconvex problems. 2017.

Razaviyayn, M., Hong, M., Luo, Z.-Q., and Pang, J.-S. Parallel successive convex approximation for nonsmooth nonconvex optimization. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 1440–1448, 2014.

Recht, B., Fazel, M., and Parrilo, P. A. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3): 471–501, 2010.

Reddi, S. J., Zaheer, M., Sra, S., Póczos, B., Bach, F., Salakhutdinov, R., and Smola, A. J. A generic approach for escaping saddle points. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

Song, E., Shen, Z., and Shi, Q. Block coordinate descent almost surely converges to a stationary point satisfying the second-order necessary condition. *optimization-online*, 2017.

Sun, J., Qu, Q., and Wright, J. When are nonconvex problems not scary? In *Proceedings of NIPS Workshop on Non-convex Optimization for Machine Learning: Theory and Practice*, 2015.

Sun, R. and Hong, M. Improved iteration complexity bounds of cyclic block coordinate descent for convex problems. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 1306–1314, 2015.

Tseng, P. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

Weyl, H. Das asymptotische verteilungsgesetz der eigenwerte linearer partieller differentialgleichungen (mit einer anwendung auf die theorie der hohlraumstrahlung). *Mathematische Annalen*, 71 (4):441–479, 1912.

Xu, Y. and Yin, W. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013.

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

Zhang, Y., Liang, P., and Charikar, M. A hitting time analysis of stochastic gradient langevin dynamics. In *Proceedings of Annual Conference on Learning Theory (COLT)*, pp. 1980–2022, 2017.

Zhao, T., Wang, Z., and Liu, H. A nonconvex optimization framework for low rank matrix estimation. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 559–567, 2015.

Zhu, Z., Li, Q., Tang, G., and Wakin, M. B. The global optimization geometry of low-rank matrix optimization. *arXiv:1703.01256*, 2018.