Escaping Saddle Points Efficiently with Occupation-Time-Adapted Perturbations

Xin Guo * 1 Jiequn Han * 2 Mahan Tajrobehkar * 1 Wenpin Tang * 3

Abstract
Motivated by the super-diffusivity of self-repelling random walk, which has roots in statistical physics, this paper develops a new perturbation mechanism for optimization algorithms. In this mechanism, perturbations are adapted to the history of states via the notion of occupation time. After integrating this mechanism into the framework of perturbed gradient descent (PGD) and perturbed accelerated gradient descent (PAGD), two new algorithms are proposed: perturbed gradient descent adapted to occupation time (PGDOT) and its accelerated version (PAGDOT). PGDOT and PAGDOT are shown to converge to second-order stationary points at least as fast as PGD and PAGD, respectively, and thus they are guaranteed to avoid getting stuck at non-degenerate saddle points. The theoretical analysis is corroborated by empirical studies in which the new algorithms consistently escape saddle points and outperform not only their counterparts, PGD and PAGD, but also other popular alternatives including stochastic gradient descent, Adam, AMSGrad, and RMSProp.

1. Introduction
Gradient descent (GD), which dates back to (Cauchy, 1847), aims to minimize a function $f : \mathbb{R}^d \to \mathbb{R}$ via the iteration:
$$x_{t+1} = x_t - \eta \nabla f(x_t), \quad t = 0, 1, 2, \ldots,$$
where $\eta > 0$ is the step size and $\nabla f$ is the gradient of $f$. Due to its simple form and fine computational properties, GD and its variants (e.g., stochastic gradient descent) are essential for many machine learning tools: principle component analysis (Candès et al., 2011), phase retrieval (Candès et al., 2015), and deep neural network (Rumelhart et al., 1986), just to name a few. In the era of data deluge, many problems are concerned with large-scale optimization in which the intrinsic dimension $d$ is large. GD turns out to be efficient in dealing with high-dimensional convex optimization, where the first-order stationary point $\nabla f(x) = 0$ is necessarily the global minimum point. Algorithmically, it involves finding a point with small gradient $||\nabla f(x)|| < \epsilon$. A classical result of (Nesterov, 2004) showed that the time required by GD to find such a point in a possibly non-convex problem is of order $\epsilon^{-2}$, independent of the dimension $d$.

In non-convex settings, applying GD will still lead to an approximate first-order stationary point. However, this is not sufficient: for non-convex functions, first-order stationary points can be either global minimum, local minimum, local maximum, or saddle points. As we will explain, saddle points are the main bottleneck for GD in many non-convex problems. The goal of this paper is therefore to develop efficient algorithms to escape saddle points in high-dimensional non-convex problems, and hence overcome the curse of dimensionality.

Escape local minima: Inspired by annealing in metallurgy, (Kirkpatrick et al., 1983) developed simulated annealing to approximate the global minimum of a given function. (Geman & Hwang, 1986) proposed a diffusion simulated annealing and proved that it converges to the set of global minimum points. However, subsequent works (Holley et al., 1989; Menz et al., 2018; Miclo, 1992; Monmarché, 2018; Tang & Zhou, 2021) revealed that it might take an exponentially long time (of order $\exp(d)$) for diffusion simulated annealing to get close to the global minimum. Some work, e.g., methods based on Lévy flights (Pavlyukevich, 2007) or Cuckoo’s search (Yang & Deb, 2009) showed empirically faster convergence to the global minimum. Yet the theory of these approaches is far-fetched. There are recent efforts in approximating the global

1Department of Industrial Engineering and Operations Research, UC Berkeley, Berkeley, CA, USA 2Center of Computational Mathematics, Flatiron Institute, New York, NY, USA 3Department of Industrial Engineering and Operations Research, Columbia University, New York City, NY, USA. Correspondence to: Xin Guo <xinguo1@berkeley.edu>, Jiequn Han <jiequnh@princeton.edu>, Mahan Tajrobehkar <mahan_tajrobehkar@berkeley.edu>, Wenpin Tang <wt2319@columbia.edu>.
Escaping Saddle Points Efficiently with Occupation-Time-Adapted Perturbations

minimum in non-convex problems via Langevin dynamics-based stochastic gradient descent (Raginsky et al., 2017; Chen et al., 2020), along with its variants using non-reversibility (Hu et al., 2020) and replica exchange (Chen et al., 2019; Dong & Tong, 2021). Typically, these algorithms take polynomial time in the dimension $d$, and thus may scale poorly when $d$ is large.

**Escape saddle points**: Fortunately, in many non-convex problems, it suffices to find a local minimum. Indeed, there has been a line of recent work arguing that local minima are less problematic, and that for many non-convex problems there are no spurious local minima. That is, all local minima are comparable in value with the global minimum. Examples include tensor decomposition (Ge et al., 2015; 2018; Ge & Ma, 2017; Sanjabi et al., 2019), semidefinite programming (Bandeira et al., 2016; Mei et al., 2017), dictionary learning (Sun et al., 2017), phase retrieval (Sun et al., 2018), robust regression (Mei et al., 2018), low-rank matrix factorization (Bhojanapalli et al., 2016; Ge et al., 2017; 2016; Park et al., 2017), and certain classes of deep neural networks (Choromanska et al., 2015; Draxler et al., 2018; Kawaguchi, 2016; Kazemipour et al., 2019; Liang et al., 2018; Nguyen & Hein, 2017; Venturi et al., 2019; Wu et al., 2018). Nevertheless, as shown in (Dauphin et al., 2014b; Du et al., 2017; Jain et al., 2017), saddle points may correspond to suboptimal solutions, and it may take exponentially long time to move from saddle points to a local minimum point. Meanwhile, it has been observed in empirical studies (Dauphin et al., 2014a; Swirszcz et al., 2016) that GD and its variants with momentum such as Adam (Kingma & Ba, 2015) may be trapped in saddle points.

(Ge et al., 2015) took the first step to show that by adding noise at each iteration, GD can escape all saddle points in polynomial time. Additionally, (Du et al., 2018; Lee et al., 2016) proved that with random initialization, GD converges to a local minimizer. Moreover, (Jin et al., 2017) proposed the perturbed gradient descent (PGD) algorithm, which (Jin et al., 2018) further improved to the perturbed accelerated gradient descent (PAGD) algorithm. They showed that PGD and PAGD are efficient – the time complexity is almost independent of the dimension $d$. See also (Jin et al., 2021) for a summary of results in this direction.

**Our idea.** Motivated by the “fast exploration” of self-repelling random walk, this paper develops a new perturbation mechanism by adapting the perturbations to the history of states. Recall that (Jin et al., 2017; 2021) used the following perturbation update when perturbation conditions hold:

$$x'_t = x_t + \text{Unif}(B^d(0, r)),$$

$$x_{t+1} = x'_t - \eta \nabla f(x'_t),$$

where $\text{Unif}(B^d(0, r))$ is a point picked uniformly in the ball of radius $r$. On the empirical side, (Neelakantan et al., 2015; Zhou et al., 2019) applied this idea of GD with noise to train deep neural networks.

Our idea is to replace $\text{Unif}(B^d(0, r))$ with non-uniform perturbations, whose mechanism depends on the current state $x_t$ and the history of states $\{x_s; s \leq t\}$. There are conceivably many ways to add non-uniform perturbation based on the current and previous states; here we choose to adapt perturbations to the “occupation time”.

The intuition is illustrated by the one-dimensional function $f(x) = x^3$ (see Figure 1).

![Figure 1. Illustration of occupation-time-adapted perturbation using $f(x) = x^3$.](image)

There is a saddle point at 0, and imagine GD approaches 0 from the right. It can be shown that GD converges monotonically to a stationary point (see Appendix A). The uniform perturbation will add noise with probability $1/2$ both to the right and to the left. To the right, GD will again get stuck at the saddle point 0. However, to the left, there is a possibility of escaping from 0 and finding a local minimum ($-\infty$ in this case). Therefore, it is reasonable to add noise with a larger probability to the left, since it has spent a long time on the right and has yet to explore the left side.

The previous intuition can be quantified via the notion of occupation times $L_t$ (the number of $\{x_s\}_{s \leq t}$ to the left of $x_t$) and $R_t$ (the number of $\{x_s\}_{s \leq t}$ to the right of $x_t$). By definition, $R_t + L_t = t$, for each $t = 0, 1, \ldots$. If $L_t$ is larger, the
Escaping Saddle Points Efficiently with Occupation-Time-Adapted Perturbations

Perturbation will push the iterate $x_t$ to the right; and if $R_t$ is larger, push to the left. More precisely,

$$x_{t+1} = \begin{cases} 
  x_t - r \text{Unif}(0, 1) & \text{with probability } p, \\
  x_t + r \text{Unif}(0, 1) & \text{with probability } 1 - p,
\end{cases} \quad (1)$$

where $p = \frac{w(R_t)}{w(L_t)+w(R_t)}$ and $w : \{0, 1, \ldots\} \to (0, \infty)$ is an increasing weight function on the nonnegative integers (e.g., $w(n) = 1 + n^\alpha$ for $\alpha > 0$).

The dynamics (1) is closely related to the vertex-repelling random walk defined by

$$Z_{t+1} = \begin{cases} 
  Z_t - 1 & \text{with probability } \frac{w(R_t)}{w(L_t)+w(R_t)}, \\
  Z_t + 1 & \text{with probability } \frac{w(L_t)}{w(L_t)+w(R_t)},
\end{cases} \quad (2)$$

where $R_t := \{ s < t : Z_s = Z_t + 1 \}$ and $L_t := \{ s < t : Z_s = Z_t - 1 \}$. This (non-Markovian) random walk model was introduced by (Peliti & Pietronero, 1987) in the statistical physics literature. Based on the scaling arguments and simulations, it was conjectured that the walk $(Z_t, t \geq 0)$ is recurrent and is further super-diffusive in the sense that $E[Z_t^2] \sim t^{\frac{4}{3}}$, whereas for a simple random walk $(S_t, t \geq 0)$ its exploration range is $E[S_t^2] \sim t^2$. These properties have only been proved rigorously for a simpler variant – the edge-repelling random walk, see (Davis, 1990; Tóth, 1995). A counterpart to the vertex-repelling walk is the vertex-reinforced walk (Pemantle, 1992; Volkov, 2006) defined by $Z_{t+1} = Z_t - 1$ with probability $\frac{w(L_t)}{w(L_t)+w(R_t)}$, and $Z_{t+1} = Z_t + 1$ with probability $\frac{w(R_t)}{w(L_t)+w(R_t)}$. It is well known (Tarrés, 2004; Volkov, 2006) that vertex-reinforced random walk exhibits localization at a finite number of points for some choices of $w(\cdot)$, e.g., $w(n) \sim n^\alpha$ with $\alpha \geq 1$.

Our results. We will first show that vertex-repelling walk will never be localized or stuck at some points in contrast with vertex-reinforced walk (see Theorem 3.1 below). The non-localization and the (conjectured) super-diffusive properties of the vertex-repelling walk facilitate exploration, and thus the corresponding perturbation scheme (1) makes it more likely to escape from saddle points.

We will then propose a new perturbation mechanism based on the dynamics (1), which can be integrated into the framework of (any) perturbation-based optimization algorithms. In particular, integrating the above-mentioned mechanism into the framework of PGD and PAGD, we propose two new algorithms: perturbed gradient descent adapted to occupation time (PGDOT, Algorithm 1) and its accelerated version, perturbed accelerated gradient descent adapted to occupation time (PAGDOT, Algorithm 2).

We will prove that Algorithm 1 (resp. Algorithm 2) converges to a second-order stationary point at least as fast as PGD (resp. PAGD).

**Algorithm 1** Perturbed Gradient Descent Adapted to Occupation Time (Meta Algorithm)

```plaintext
for $t = 0, 1, \ldots$ do
  if perturbation condition holds then
    for $i = 1, \ldots, d$ do
      $L_t^i \leftarrow \# \{ s < t : x_s^i \leq x_t^i \}$
      $R_t^i \leftarrow \# \{ s < t : x_s^i > x_t^i \}$
      $x_t^i \leftarrow \begin{cases} 
        x_t^i - \frac{\sqrt{d}}{\sqrt{2}} \text{Unif}(0, 1) & \text{w.p. } p, \\
        x_t^i + \frac{\sqrt{d}}{\sqrt{2}} \text{Unif}(0, 1) & \text{w.p. } 1 - p,
      \end{cases}$
    where $p = \frac{w(R_t)}{w(L_t)+w(R_t)}$
    $x_{t+1} \leftarrow x_t - \eta \nabla f(x_t)$
  end if
end for
```

Algorithms 1 and 2 are state-dependent adaptive algorithms, perturbing GD and accelerated gradient descent (AGD) (Nesterov, 1983) non-uniformly according to the history of states.

We will finally corroborate our theoretical analysis by experimental results. In particular, we will demonstrate that Algorithms 1 and 2 escape saddle points faster than not only their counterparts, PGD and PAGD, but also several momentum methods such as Adam, AMSGrad, and RMSProp in training multilayer perceptrons (MLPs) on some well-studied datasets such as MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky et al., 2009).
We say that a differentiable function $f$ is strict if $\lambda_{\min}(\nabla^2 f(x)) > 0$ for all $x$ in some neighborhood of $x^*$. A popular approach is to consider the first-order stationary points instead.

**Definition 2.1.** Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function. We say that (i) $x$ is a first-order stationary point of $f$ if $\nabla f(x) = 0$; (ii) $x$ is an $\epsilon$-first-order stationary point of $f$ if $\|\nabla f(x)\| < \epsilon$.

We say that a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is $\ell$-gradient Lipschitz if $\|\nabla f(x_1) - \nabla f(x_2)\| \leq \ell \|x_1 - x_2\|$ for all $x_1, x_2 \in \mathbb{R}^d$.

**Theorem 2.2.** Assume that $f : \mathbb{R}^d \to \mathbb{R}$ is $\ell$-gradient Lipschitz. For any $\epsilon > 0$, if we run GD with step size $\eta = \ell^{-1}$, then the number of iterations to find an $\epsilon$-first-order stationary point is $\frac{\ell(f(x_0) - f^*)}{\epsilon^2}$.

Note that in Theorem 2.2, the time complexity of GD is independent of the dimension $d$. For a non-convex function, a first-order stationary point can be either a local minimum, a saddle point, or a local maximum. The following definition is taken from (Jin et al., 2017)[Definition 4].

**Definition 2.3.** Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function. We say that (i) $x$ is a local minimum if $x$ is a first-order stationary point, and $f(x) \leq f(y)$ for all $y$ in some neighborhood of $x$; (ii) $x$ is a saddle point if $x$ is a first-order stationary point but not a local minimum. Assume further that $f$ is twice differentiable. We say a saddle point $x$ is strict if $\lambda_{\min}(\nabla^2 f(x)) < 0$.

For a twice differentiable function $f$, note that $\lambda_{\min}(\nabla^2 f(x)) \leq 0$ for any saddle point $x$. So by assuming a saddle point $x$ to be strict, we rule out the case $\lambda_{\min}(\nabla^2 f(x)) = 0$. The next subsection will review two perturbation-based algorithms that allow jumping out of strict saddle points.

### 2.2. Results of PGD and PAGD

One drawback of GD in non-convex optimization is that it may get stuck at saddle points. (Jin et al., 2017) and (Jin et al., 2018) proposed PGD and PAGD, respectively, to escape saddle points, which we review here. To proceed further, we need...
Theorem 3.1. Let
\( \text{Theorem 2.6.} \) Assume that \( \text{Theorem 2.5.} \) Assume that
\( (\text{Jin et al., 2018}) \) Assume that \( (\text{Jin et al., 2017}) \)
\( \text{Theorem 2.5.} \) Assume that
\( (\text{Jin et al., 2017}) \)
\( \text{The proof of this theorem is given in Appendix C.} \)

3.2. Perturbed Gradient Descent Adapted to Occupation Time

Compared with Theorem 2.2, PGD takes almost the same order of time to find a second-order stationary point as GD does to find a first-order stationary point.

Theorem 2.6. (\text{Jin et al., 2018}) Assume that \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) is \( \ell \)-gradient Lipschitz and \( \rho \)-Hessian Lipschitz. Then there exists \( c_{\text{max}} > 0 \) such that for any \( \delta > 0, \epsilon \leq \ell^2 / \rho, \Delta_f \geq f(x_0) - f^* \), and \( c \leq c_{\text{max}}, \) PGD outputs an \( \epsilon \)-second-order stationary point with probability \( 1 - \delta \), terminating within the following number of iterations:
\[
O \left( \frac{\ell(f(x_0) - f^*)}{\epsilon^2} \log^4 \left( \frac{d\ell \Delta_f}{\epsilon^2 \delta} \right) \right).
\]

The main result for PGD, Theorem 3 in (\text{Jin et al., 2017}), and for PAGD, Theorem 3 in (\text{Jin et al., 2018}), are stated below showing that the time complexity of these two algorithms are almost dimension-free (with a log factor).

Theorem 2.5. (\text{Jin et al., 2017}) Assume that \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) is \( \ell \)-gradient Lipschitz and \( \rho \)-Hessian Lipschitz. Then there exists \( c_{\text{max}} > 0 \) such that for any \( \delta > 0, \epsilon \leq \ell^2 / \rho, \Delta_f \geq f(x_0) - f^* \), and \( c \leq c_{\text{max}}, \) PGD outputs an \( \epsilon \)-second-order stationary point with probability \( 1 - \delta \), terminating within the following number of iterations:
\[
O \left( \frac{\ell(f(x_0) - f^*)}{\epsilon^2} \log^4 \left( \frac{d\ell \Delta_f}{\epsilon^2 \delta} \right) \right).
\]

3. Main Results

In this section, we first prove the non-localization property of the vertex-repelling random walk. Then, we formalize the idea of perturbations adapted to occupation time and provide the full version of PGDOT and PAGDOT in Algorithms 3 and 4, respectively. Our main results show that these algorithms converge rapidly to second-order stationary points.

3.1. Non-Localization Property of Vertex-Repelling Random Walk

The following theorem suggests that the new perturbation mechanism helps perturbation-based algorithms to avoid getting stuck at saddle points, as the dynamics of the vertex-repelling random walk prescribed in (1) does not localize.

Theorem 3.1. Let \( \{Z_t, \ t = 0, 1, \ldots\} \) be the vertex-repelling random walk defined by (2), where \( w : \{0, 1, \ldots\} \rightarrow (0, \infty) \) is an increasing function such that \( w(n) \rightarrow \infty \) as \( n \rightarrow \infty \). Then
\[
P(\exists t_0 > 0, k \leq \ell : Z_t \in \{k, \ldots, \ell\} \text{ for all } t \geq t_0) = 0.
\]

The proof of this theorem is given in Appendix C.

3.2. Perturbed Gradient Descent Adapted to Occupation Time

PGD adds a uniform random perturbation when stuck at saddle points. From the discussion in the introduction, it is more reasonable to perturb with non-uniform noise whose distribution depends on the occupation times. Recall that \( w : \{0, 1, \ldots\} \rightarrow (0, \infty) \) is an increasing weight function on the nonnegative integers. The following algorithm adapts PGD to random perturbation depending on the occupation dynamics. We follow the parameter setting as in (\text{Jin et al., 2017}). Our algorithm performs GD with step size \( \eta \) and gets a perturbation of amplitude \( \frac{\tau}{\sqrt{d}} \) near saddle points at most once every \( t_{\text{thres}} \) iterations. The threshold \( t_{\text{thres}} \) ensures that the dynamics of the algorithm is mostly GD. The threshold \( g_{\text{thres}} \) determines if a perturbation is needed, and the threshold \( f_{\text{thres}} \) decides when the algorithm terminates.
The next theorem gives the convergence rate of Algorithm 3: PGDOT finds a second-order stationary point in the same number of iterations (up to a constant factor) as PGD does.

**Theorem 3.2.** Assume that \( f : \mathbb{R}^d \to \mathbb{R} \) is \( \ell \)-gradient Lipschitz and \( \rho \)-Hessian Lipschitz. Then there exists \( c_{\text{max}} > 0 \) such that for any \( \delta > 0 \), \( \epsilon \leq \ell / \rho, \Delta^2 \geq f(\mathbf{x}_0) - f^* \), and \( c \leq c_{\text{max}} \), PGDOT (Algorithm 3) outputs an \( \epsilon \)-second-order stationary point with probability \( 1 - \delta \) terminating within the following number of iterations:

\[
O \left( \frac{\ell(f(\mathbf{x}_0) - f^*)}{\epsilon^2} \log^4 \left( \frac{d\ell \Delta}{\epsilon^3 \delta} \right) \right).
\]

The proof of Theorem 3.2 is based on a geometric characterization of saddle points – thin pancake property (Jin et al., 2017). In Appendix D, we will discuss this property, and show how it is used to prove Theorem 3.2.

### 3.3. Perturbed Accelerated Gradient Descent Adapted to Occupation Time

Similar to the way we combined our perturbation mechanism with PGD, we can adapt PAGD to this mechanism as well resulting in the accelerated version of PGDOT (Algorithm 4). We follow the parameter setting as in (Jin et al., 2018).

**Algorithm 4** Perturbed Accelerated Gradient Descent Adapted to Occupation Time: PAGDOT(\( \mathbf{x}_0, \ell, \rho, \epsilon, c, \delta, \Delta^2 \))

\[
\begin{align*}
\chi &\leftarrow 3 \max \left\{ \log\left( \frac{d\Delta}{\rho \ell^2}, 1 \right), 0 \right\}, \quad \eta \leftarrow \frac{c}{\sqrt{d}}, \quad \gamma \leftarrow \frac{1}{4t} \\
\eta_0 &\leftarrow 0, \quad \theta \leftarrow \frac{1}{\sqrt{\gamma}}, \quad \gamma \leftarrow \frac{2}{\sqrt{4d}}, \quad s \leftarrow \frac{\rho}{\chi^2 d}, \quad r \leftarrow \frac{\rho}{\chi^2 d}, \quad \tau \leftarrow \chi \sqrt{\tau} \\
\text{for } t = 0, 1, \ldots, & \text{ do} \\
\quad & \text{if } ||\nabla f(\mathbf{x}_t)|| \leq \epsilon \text{ and no perturbation in last } \tau \text{ steps then} \\
\quad & \quad \mathbf{x}_t \leftarrow \mathbf{x}_t \\
\quad & \quad \text{for } i = 1, \ldots, d \text{ do} \\
\quad & \quad \quad \mathbf{L}_i \leftarrow \# \{ s < t : x^i_s \leq x^i_t \} \\
\quad & \quad \quad \mathbf{R}_i \leftarrow \# \{ s < t : x^i_s > x^i_t \} \\
\quad & \quad \quad x^i_t \leftarrow \begin{cases} 
\mathbf{x}^i_t - \frac{\chi^2}{\tau d} \text{Unif}(0, 1) & \text{w.p. } p, \quad \text{ where } p = \frac{\mathbf{w}(\mathbf{R}_i)}{\mathbf{w}(\mathbf{L}_i) + \mathbf{w}(\mathbf{R}_i)} \\
\mathbf{x}^i_t + \frac{\chi^2}{\tau d} \text{Unif}(0, 1) & \text{w.p. } 1 - p
\end{cases} \\
\quad & \quad \text{if } f(\mathbf{x}_t) \leq f(\mathbf{y}_t) + (\nabla f(\mathbf{y}_t), \mathbf{x}_t - \mathbf{y}_t) - \frac{\gamma}{2} ||\mathbf{x}_t - \mathbf{y}_t||^2 \text{ then} \\
\quad & \quad \quad (\mathbf{x}_{t+1}, \mathbf{v}_{t+1}) \leftarrow \text{NCE}(\mathbf{x}_t, \mathbf{v}_t, s)
\end{align*}
\]

Algorithm 4, similar to PAGD, enjoys a feature enabling it to reset the momentum and decide whether to exploit the negative curvature when the function becomes “too convex” (see Algorithm 5).
As shown in these examples, integrating our new perturbation mechanism into the framework of perturbation-based algorithms boosts their performance: PGDOT and PAGDOT escape saddle points or plateaus faster than their counterparts. Specifically, example 4 shows that in training MLPs on the MNIST and CIFAR-10 datasets, PGDOT and PAGDOT are robust against different initialization and manage to escape saddle points efficiently; in contrast, PGD and PAGD as well as other popular algorithms such as stochastic gradient descent (SGD), Adam, AMSGrad, and RMSProp fail to do so. In addition, PAGDOT converges faster than PGDOT in all these examples, which is in line with the theoretical results in Theorems 3.2 and 3.3.

In these experiments, we use $L_i^t(h) := \# \{ t - t_{\text{count}} \leq s < t : x_i^s - h \leq x_i^t \leq x_i^s + h \}$ and $R_i^t(h) := \# \{ t - t_{\text{count}} \leq s < t : x_i^s < x_i^t \leq x_i^s + h \}$ instead of $L_i^t$ and $R_i^t$ in Algorithms 3 and 4. Here $h$ is a hyperparameter characterizing the occupation time over a small interval. $t_{\text{count}}$ is another hyperparameter prescribing how long one should keep track of the history of $x_i$ in order to approximate the occupation time with a constant memory cost. We choose the weight function in Algorithms 3 and 4 as $w(n) = 1 + n^5$. All other hyperparameters used in the numerical examples are reported in Appendix E.

Example 1 Given $N \in \mathbb{Z}^+$, $L \in \mathbb{R}^+$, define a function $\hat{f} : \mathbb{R}^+ \to \mathbb{R}^+$ as

$$\hat{f}(r) = \begin{cases} r^3, & r \in [0, \frac{1}{2} L), \\ (r - nL)^3 + \frac{1}{4} nL^3, & r \in [a(n), b(n)), 1 \leq n \leq N, \\ (r - NL)^3 + \frac{1}{4} NL^3, & r \in [NL + \frac{1}{2} L, \infty), \end{cases}$$

where $a(n) = nL - \frac{1}{2} L$ and $b(n) = nL + \frac{1}{2} L$. For $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$, we define $f(x) = \hat{f} \left( \frac{1}{2} \sum_{i=1}^{d} x_i^2 \right)$. Figure 2 gives the visualization of the case $N = 4, L = 1$ and also the training curves of $f$ given by 5 different algorithms when $d = 4$. The initial values are all the same, and all the algorithms except for GD are run 3 times considering the randomness of perturbations. We can see that while GD gets stuck at the saddle points, all the rest of the algorithms escape from them. Moreover, the new algorithms PGDOT and PAGDOT outperform their counterparts.
Example 2  We consider a nonlinear regression problem, adapted from learning time series data with a continuous dynamical system (Li et al., 2021). The loss function is defined as $f(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}(s_i; \mathbf{x}) - y^*(s_i))^2$, where $\{s_i\}_{i=1}^{N}$ are $N$ sample points, $y^*(s)$ is the target function, and $\hat{y}(s)$ is the function to fit with the form $\hat{y}(s; \mathbf{x}) = \sum_{m=1}^{M} (a_m \cos (\lambda_m s) + b_m \sin (\lambda_m s)) e^{w_m s}$. Here $\mathbf{x} = \{a_m, b_m, \lambda_m, w_m\}_{m=1}^{M}$ and the optimization problem is non-convex. We assume $y^*(s) = \text{Ai}(\omega|s - s_0|)$, where $\omega = 3.2$, $s_0 = 3.0$, and $\text{Ai}(s)$ is the Airy function of the first kind, given by the improper integral $\text{Ai}(s) = \frac{1}{\pi} \int_{0}^{\infty} \cos \left( \frac{u^3}{3} + su \right) \, du$.

For the specific regression model, we assume $M = 4$ and use $N = 50$ data points with $s_i = i/10$, $i = 0, \ldots, 49$. Figure 3 shows the target function and the fitted function obtained by PGDOT. Also, the learning curves of 5 different algorithms are plotted. Again, PGDOT and PAGDOT escape the saddle point faster than GD and outperform PGD and PAGD, respectively.

Example 3  The next two non-convex optimization problems are taken from (Wang et al., 2019). The first problem is a regularized linear-quadratic problem (Reddi et al., 2018a), whose loss function is

$$f_1(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \left( \frac{1}{2} \mathbf{x}^T H \mathbf{x} + \mathbf{b}_i^T \mathbf{x} + ||\mathbf{x}||_{10} \right),$$
where we take $N = 10$, $H = \text{diag}([1, -0.1])$ and $b_i$’s instances of $\mathcal{N}(0, \text{diag}([0.1, 0.001]))$. The second problem is the phase retrieval problem (Candès et al., 2013) with loss function

$$f_2(x) = \frac{1}{N} \sum_{i=1}^{N} ((a_i^T x)^2 - (a_i^T x^*)^2),$$

where we choose $N = 200$, $x^*$ an instance of $\mathcal{N}(0, I_d/d)$ and $a_i$’s instances of $\mathcal{N}(0, I_d)$ with $d = 10$.

We initialize the regularized linear-quadratic problem with $x_0 = 0$, and the phase retrieval problem with $x_0$ sampled from $\mathcal{N}(0, I_d/(10000d))$. Figure 4 presents the learning curves of 5 different algorithms. In both problems, all other algorithms escape saddle points faster than GD, with PGDOT and PAGDOT outperforming their counterparts.

**Figure 4.** The performance of different algorithms for $f_1$ (left) and $f_2$ (right) in Example 3.

**Example 4** (Dauphin et al., 2014a) observed that in training simple MLPs (MLPs with only one hidden layer) on the MNIST and CIFAR-10 datasets, SGD might get stuck at saddle points. Moreover, as demonstrated in (Swirszcz et al., 2016), Adam also gets stuck and performs poorly when a simple MLP with specific initialization is trained on the MNIST dataset. Inspired by (Swirszcz et al., 2016), we conduct two sets of experiments on the MNIST and CIFAR-10 datasets, in which we train several simple MLPs using the mini-batch version of our proposed algorithms as well as the mini-batch version of other popular alternatives such as SGD, Adam, AMSGrad, and RMSProp. For both of the datasets, the batch size is set to 128 and all the images are downsized to be of size $10 \times 10$.

In the first set of experiments, we train several simple MLPs whose weights and biases are initialized with $\mathcal{N}(0, 0.01)$ on the aforementioned datasets. The top two rows in Figure 5 show the training curves of SGD, Adam, PGD, PGDOT, PAGD, and PAGDOT. Note that $n_{\text{hidden}}$ is the number of neurons in the hidden layer of simple MLP. Observe that all algorithms manage to escape saddle points in all the cases with the exception of Adam, which fails on the CIFAR-10 dataset.

For the second set of experiments, we consider several simple MLPs whose weights and biases are initialized with $\mathcal{N}(-1, 0.01)$. The bottom two rows in Figure 5 show the training curves of different algorithms. For both of the datasets, while SGD and Adam are stuck at the saddle points, the new algorithms PGDOT and PAGDOT escape the saddle points and significantly outperform their counterparts. Note that for the CIFAR-10 dataset, the training curves of SGD, Adam, PGD, and PAGD are almost identical.

It is also worth mentioning that other variants of Adam such as AMSGrad (Reddi et al., 2018b) and RMSProp also fail to escape the saddle points in the training process (see Figure 6 in Appendix F). Moreover, comparing these results with that of the first set of experiments (Figure 5, top two rows) we conclude that the new perturbation mechanism helps the algorithms to be robust against different initialization.

**5. Conclusion**

In this paper, we develop a new perturbation mechanism in which the perturbations are adapted to the history of states via the notion of occupation time. This mechanism is integrated into the framework of PGD and PAGD resulting in two new algorithms: PGDOT and PAGDOT. We prove that PGDOT and PAGDOT converge rapidly to second-order stationary points, which is corroborated by empirical studies ranging from time series analysis and the phase retrieval problem to neural networks.
Figure 5. Training simple MLPs with two different initialization on the MNIST and CIFAR-10 datasets. First row: \( N(0, 0.01) \) initialization, MNIST; Second row: \( N(0, 0.01) \) initialization, CIFAR-10; Third row: \( N(-1, 0.01) \) initialization, MNIST; Fourth row: \( N(-1, 0.01) \) initialization, CIFAR-10.

References

Ball, K. Cube slicing in \( \mathbb{R}^n \). Proc. Amer. Math. Soc., 97(3):465–473, 1986.

Bandeira, A. S., Boumal, N., and Voroninski, V. On the low-rank approach for semidefinite programs arising in synchronization and community detection. In Conference on Learning Theory, pp. 361–382, 2016.

Bhojanapalli, S., Neyshabur, B., and Srebro, N. Global optimality of local search for low rank matrix recovery. In Advances in Neural Information Processing Systems, pp. 3873–3881, 2016.

Bubeck, S. Convex optimization: Algorithms and complexity. Foundations and Trends® in Machine Learning, 8(3-4):231–357, 2015.

Candès, E. J., Li, X., Ma, Y., and Wright, J. Robust principal component analysis? J. ACM, 58(3):Art. 11, 37, 2011.

Candès, E. J., Eldar, Y. C., Strohmer, T., and Voroninski, V. Phase retrieval via matrix completion. SIAM J. Imaging Sci., 6 (1):199–225, 2013.
Candès, E. J., Li, X., and Soltanolkotabi, M. Phase retrieval via Wirtinger flow: theory and algorithms. *IEEE Trans. Inform. Theory*, 61(4):1985–2007, 2015.

Cauchy, A. Méthode générale pour la résolution des systemes d’équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847): 536–538, 1847.

Chen, X., Du, S. S., and Tong, X. T. On stationary-point hitting time and ergodicity of stochastic gradient Langevin dynamics. *J. Mach. Learn. Res.*, 21:Paper No. 68, 41, 2020.

Chen, Y., Chen, J., Dong, J., Peng, J., and Wang, Z. Accelerating nonconvex learning via replica exchange Langevin diffusion. In *International Conference on Learning Representations (ICLR)*, 2019.

Choromanska, A., Henaff, M., Mathieu, M., Ben Arous, G., and LeCun, Y. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pp. 192–204, 2015.

Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. 2014a. arXiv:1406.2572.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, pp. 2933–2941, 2014b.

Davis, B. Reinforced random walk. *Probab. Theory Related Fields*, 84(2):203–229, 1990.

Dong, J. and Tong, X. T. Replica exchange for non-convex optimization. *J. Mach. Learn. Res.*, 22:Paper No. 173, 59, 2021.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. Essentially no barriers in neural network energy landscape. In *International Conference on Machine Learning*, pp. 1309–1318, 2018.

Du, S. S., Jin, C., Lee, J. D., Jordan, M. I., Singh, A., and Poczos, B. Gradient descent can take exponential time to escape saddle points. In *Advances in Neural Information Processing Systems*, pp. 1067–1077, 2017.

Du, S. S., Lee, J. D., Tian, Y., Singh, A., and Poczos, B. Gradient descent learns one-hidden-layer CNN: Don’t be afraid of spurious local minima. In *International Conference on Machine Learning*, pp. 1339–1348, 2018.

Ge, R. and Ma, T. On the optimization landscape of tensor decompositions. In *Advances in Neural Information Processing Systems*, pp. 3653–3663, 2017.

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

Ge, R., Lee, J. D., and Ma, T. Matrix completion has no spurious local minimum. In *Advances in Neural Information Processing Systems*, pp. 2973–2981, 2016.

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

Ge, R., Lee, J. D., and Ma, T. Learning one-hidden-layer neural networks with landscape design. In *International Conference on Learning Representations*, 2018.

Geman, S. and Hwang, C.-R. Diffusions for global optimization. *SIAM J. Control Optim.*, 24(5):1031–1043, 1986.

Holley, R. A., Kusuoka, S., and Stroock, D. W. Asymptotics of the spectral gap with applications to the theory of simulated annealing. *J. Funct. Anal.*, 83(2):333–347, 1989.

Hu, Y., Wang, X., Gao, X., Gürbüzbalaban, M., and Zhu, L. Non-convex optimization via non-reversible stochastic gradient Langevin dynamics. 2020. arXiv:2004.02823.

Jain, P., Jin, C., Kakade, S., and Netrapalli, P. Global convergence of non-convex gradient descent for computing matrix squareroot. In *Artificial Intelligence and Statistics*, pp. 479–488, 2017.
Jin, C., Ge, R., Netrapalli, P., Kakade, S., and Jordan, M. I. How to escape saddle points efficiently. In *International Conference on Machine Learning*, pp. 1724–1732, 2017.

Jin, C., Netrapalli, P., and Jordan, M. I. Accelerated gradient descent escapes saddle points faster than gradient descent. In *Conference On Learning Theory*, pp. 1042–1085, 2018.

Jin, C., Netrapalli, P., Ge, R., Kakade, S. M., and Jordan, M. I. On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points. *Journal of the ACM*, 68(2):1–29, 2021.

Kawaguchi, K. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pp. 586–594, 2016.

Kazemipour, A., Larsen, B., and Druckmann, S. Avoiding spurious local minima in deep quadratic networks. 2019. arXiv:2001.00098.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kirkpatrick, S., Gelatt, J. C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

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

Li, Z., Han, J., E. W., and Li, Q. On the curse of memory in recurrent neural networks: Approximation and optimization analysis. In *International Conference on Learning Representations*, 2021.

Liang, S., Sun, R., Lee, J. D., and Srikanth, R. Adding one neuron can eliminate all bad local minima. In *Advances in Neural Information Processing Systems*, pp. 4350–4360, 2018.

Mei, S., Misiakiewicz, T., Montanari, A., and Oliveira, R. I. Solving SDPs for synchronization and MaxCut problems via the Grothendieck inequality. In *Conference on Learning Theory*, pp. 1476–1515, 2017.

Mei, S., Bai, Y., and Montanari, A. The landscape of empirical risk for nonconvex losses. *Ann. Statist.*, 46(6A):2747–2774, 2018.

Menz, G., Schlichting, A., Tang, W., and Wu, T. Ergodicity of the infinite swapping algorithm at low temperature. 2018. arXiv:1811.10174.

Micol, L. Recuit simulé sur $\mathbb{R}^n$. Étude de l’évolution de l’énergie libre. *Ann. Inst. H. Poincaré Probab. Statist.*, 28(2):235–266, 1992.

Monmarché, P. Hypocoercivity in metastable settings and kinetic simulated annealing. *Probab. Theory Related Fields*, 172 (3-4):1215–1248, 2018.

Neelakantan, A., Vilnis, L., Le, Q. V., Sutskever, I., Kaiser, L., Kurach, K., and Martens, J. Adding gradient noise improves learning for very deep networks. 2015. arXiv:1511.06807.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, Boston, MA, 2004.

Nesterov, Y. E. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In *Dokl. Akad. Nauk SSSR*, volume 269, pp. 543–547, 1983.

Nguyen, Q. and Hein, M. The loss surface of deep and wide neural networks. In *International Conference on Machine Learning*, pp. 2603–2612, 2017.
Escaping Saddle Points Efficiently with Occupation-Time-Adapted Perturbations

Park, D., Kyrillidis, A., Carmanis, C., and Sanghavi, S. Non-square matrix sensing without spurious local minima via the burer-monteiro approach. In Artificial Intelligence and Statistics, pp. 65–74, 2017.

Pavlyukevich, I. Lévy flights, non-local search and simulated annealing. Journal of Computational Physics, 226(2):1830–1844, 2007.

Peliti, L. and Pietronero, L. Random walks with memory. La Rivista del Nuovo Cimento, 10(6):1–33, 1987.

Pemantle, R. Vertex-reinforced random walk. Probab. Theory Related Fields, 92(1):117–136, 1992.

Polyak, B. T. Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics, 4(5):1–17, 1964.

Raginsky, M., Rakhlin, A., and Telgarsky, M. Non-convex learning via stochastic gradient Langevin dynamics: a nonasymptotic analysis. In Conference On Learning Theory, pp. 1674–1703, 2017.

Reddi, S., Zaheer, M., Sra, S., Poczos, B., Bach, F., Salakhutdinov, R., and Smola, A. A generic approach for escaping saddle points. In International Conference on Artificial Intelligence and Statistics, pp. 1233–1242, 2018a.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of Adam and beyond. In International Conference on Learning Representations, 2018b.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. Nature, 323 (6088):533–536, 1986.

Sanjabi, M., Baharlouei, S., Razaviyayn, M., and Lee, J. D. When does non-orthogonal tensor decomposition have no spurious local minima? 2019. arXiv:1911.09815.

Sun, J., Qu, Q., and Wright, J. Complete dictionary recovery over the sphere I: Overview and the geometric picture. IEEE Trans. Inform. Theory, 63(2):853–884, 2017.

Sun, J., Qu, Q., and Wright, J. A geometric analysis of phase retrieval. Found. Comput. Math., 18(5):1131–1198, 2018.

Swirszcz, G., Czarnecki, W. M., and Pascanu, R. Local minima in training of deep networks. 2016. Available at https://openreview.net/pdf?id=Syoiqwcxx.

Tang, W. and Zhou, X. Y. Simulated annealing from continuum to discretization: a convergence analysis via the Eyring–Kramers law. 2021. arXiv:2102.02339.

Tarrès, P. Vertex-reinforced random walk on $\mathbb{Z}$ eventually gets stuck on five points. Ann. Probab., 32(3B):2650–2701, 2004.

Tóth, B. The “true” self-avoiding walk with bond repulsion on $\mathbb{Z}$: limit theorems. Ann. Probab., 23(4):1523–1556, 1995.

Venturi, L., Bandeira, A. S., and Bruna, J. Spurious valleys in one-hidden-layer neural network optimization landscapes. J. Mach. Learn. Res., 20:Paper No. 133, 34, 2019.

Volkov, S. Phase transition in vertex-reinforced random walks on $\mathbb{Z}$ with non-linear reinforcement. J. Theoret. Probab., 19 (3):691–700, 2006.

Wang, J.-K., Lin, C.-H., and Abernethy, J. Escaping saddle points faster with stochastic momentum. In International Conference on Learning Representations, 2019.

Wu, C., Luo, J., and Lee, J. D. No spurious local minima in a two hidden unit ReLU network. 2018. Available at https://openreview.net/forum?id=B14uJzW0b.

Yang, X.-S. and Deb, S. Cuckoo search via Lévy flights. In 2009 World congress on nature & biologically inspired computing (NaBIC), pp. 210–214, 2009.

Zhou, M., Liu, T., Li, Y., Lin, D., Zhou, E., and Zhao, T. Toward understanding the importance of noise in training neural networks. In International Conference on Machine Learning, 2019.
A. Monotone convergence of gradient descent

Here we prove a property of gradient descent applied to a function $f : \mathbb{R} \to \mathbb{R}$, as mentioned in the introduction. This property of gradient descent supports the use of our proposed perturbation mechanism.

**Proposition A.1.** Let $f \in C^2(\mathbb{R})$. Assume that we start gradient descent at some arbitrary point $x_0$, and the corresponding iterates $\{x_n\}_{n \geq 0}$ converge to the point $x_s$ with $f''(x_s) \neq 0$. Then, if $f$ is $\ell$-smooth and the step size is less than $\frac{1}{\ell}$, the sequence $\{x_n\}_{n \geq 0}$ converges monotonically to $x_s$.

In order to prove this proposition, we break it down into two lemmas.

**Lemma A.2.** Let $f \in C^2(\mathbb{R})$. Assume that we start gradient descent at some arbitrary point $x_0$, and the corresponding iterates $\{x_n\}_{n \geq 0}$ converge to the point $x_s$ with $f''(x_s) \neq 0$. Then, if $f$ is $\ell$-smooth and the step size is less than $\frac{1}{\ell}$, there exists $M > 0$ such that the sequence $\{x_n\}_{n \geq M}$ converges monotonically to $x_s$.

**Proof.** Note that for $n \geq 0$, $x_{n+1} = x_n - \eta f'(x_n)$, where $0 < \eta < \frac{1}{\ell}$ is the step size. Also, it is easy to show that $f'(x_s) = 0$. Assume that at some point $x_n \geq x_s$. Then, since $|f'(x_n) - f'(x_s)| = |f'(x_n)| \leq \ell |x_n - x_s|$, we have

$$x_s \leq x_n - \frac{1}{\ell} |f'(x_n)| \leq x_n - \eta f'(x_n) \leq x_{n+1}.$$

Similarly, if $x_n \leq x_s$, then we get $x_{n+1} \leq x_s$. This implies that the sequence $\{x_n\}_{n \geq 0}$ is entirely either on the left hand side of $x_s$ or on its right hand side (including $x_s$).

Without loss of generality, assume that the entire sequence of iterations lies on the right hand side of $x_s$. If at some iteration, $x_m = x_s$, then since $f'(x_s) = 0$, $x_n = x_s$ for all $n \geq m$, which yields the desired result. So we can assume that $x_n \neq x_s$ for all $n \geq 0$. Using a similar argument, we can also assume that $f''(x_n) \neq 0$ for all $n \geq 0$. Suppose by contradiction that there is no such $M$ as described in the lemma. Then there exist infinitely many $n$ such that $x_n < x_{n+1}$ implying that for infinitely many $n$, $f'(x_n) < 0$. Since $\lim_{n \to \infty} x_n = x_s$ and the entire sequence is on the right hands side of $x_s$, we also have infinitely many $n$ such that $f'(x_n) > 0$. Combining these results, one can construct a strictly decreasing sub-sequence $\{y_n\}_{n \geq 0}$ of the iterations such that $\lim_{n \to \infty} y_n = x_s$, $f'(y_{2m}) > 0$, and $f'(y_{2m+1}) < 0$ for all $m \geq 0$. Since $f'$ is continuous, there exists $y_{2m+1} < y_m < y_{2m}$ such that $f'(y_m) = 0$, for each $m \geq 0$. It is easy to see that $\{z_n\}_{n \geq 0}$ is also strictly decreasing and $\lim_{n \to \infty} z_n = x_s$. Note that since $f''$ is continuous, by the mean value theorem, one can find a sequence $\{t_n\}_{n \geq 0}$ such that for each $n \geq 0$, $z_{n+1} < t_n < z_n$ and $f''(t_n) = 0$. Since $\{z_n\}_{n \geq 0}$ converges to $x_s$, then so does $\{t_n\}_{n \geq 0}$. But this implies that $f''(x_s) = \lim_{n \to \infty} f''(t_n) = 0$ contradicting with the fact that $f''(x_s) \neq 0$. \hfill $\square$

**Lemma A.3.** Given the setting in Lemma A.2, $\{x_n\}_{n \geq 0}$ converges monotonically to $x_s$.

**Proof.** Without loss of generality, assume that $x_0 \geq x_s$, then using what we obtained during the proof of Lemma A.2, we know that the entire sequence $\{x_n\}_{n \geq 0}$ lies on the right hand side of $x_s$ (including $x_s$). Let $M$ be the minimum index that satisfies the condition in Lemma A.2. Suppose by contradiction that $M > 0$. So $x_s < x_{M-1} < x_M$, which implies $f'(x_{M-1}) < 0$ considering $x_M = x_{M-1} - \eta f'(x_{M-1})$. Since the sequence converges to $x_s$, there should be a $k \geq 0$ such that $x_{M+k+1} < x_{M-1} < x_{M+k}$. Note that $x_{M+k+1} = x_{M+k} - \eta f'(x_{M+k})$, so

$$\eta f'(x_{M+k}) = x_{M+k} - x_{M+k+1} > x_{M+k} - x_{M-1}.$$

Since $f'(x_{M-1}) < 0$, we have $\eta f'(x_{M+k}) - f'(x_{M-1}) > \eta f'(x_{M+k}) - x_{M+k} - x_{M-1}$. This contradicts the fact that $\eta f'(x_{M+k}) - f'(x_{M-1}) \leq \ell(x_{M+k} - x_{M-1}) < x_{M+k} - x_{M-1}$. \hfill $\square$

B. Background on convex optimization

We provide some context of gradient descent applied to convex functions.

**Definition B.1.**

1. A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is $\ell$-gradient Lipschitz if $||\nabla f(x_1) - \nabla f(x_2)|| \leq \ell ||x_1 - x_2||$ for all $x_1, x_2 \in \mathbb{R}^d$. 

Escaping Saddle Points Efficiently with Occupation-Time-Adapted Perturbations
2. A twice differentiable function \( f : \mathbb{R}^d \to \mathbb{R} \) is \( \alpha \)-strongly convex if \( \lambda_{\min}(\nabla^2 f(x)) \geq \alpha \) for all \( x \in \mathbb{R}^d \).

The gradient Lipschitz condition controls the amount of decay in each iteration, and the strong convexity condition guarantees that the unique stationary point is the global minimum. The ratio \( \ell/\alpha \) is often called the condition number of the function \( f \).

The following theorem shows the local convergence of gradient descent to the global minimum \( x^* \), see (Bubeck, 2015)[Theorem 3.10] and (Nesterov, 2004)[Theorem 2.1.15].

**Theorem B.2.** (Bubeck, 2015; Nesterov, 2004) Assume that \( f : \mathbb{R}^d \to \mathbb{R} \) is \( \ell \)-gradient Lipschitz and \( \alpha \)-strongly convex. For any \( \epsilon > 0 \), if we run gradient descent with step size \( \eta = \ell^{-1} \), then the number of iterations to be \( \epsilon \)-close to \( x^* \) is
\[
\frac{2\ell^2}{\alpha} \log \left( \frac{\|x_0 - x^*\|}{\epsilon} \right).
\]

**C. Proof of Theorem 3.1**

Suppose by contradiction that with positive probability, the walk is localized at some points \( \{k, \ldots, \ell\} \). We focus on the left end \( k \). Let \( r_n^k \) be the time at which the point \( k \) is visited \( n \) times. For \( n \) sufficiently large, the point \( k+1 \) is visited approximately at least \( n \) times by \( r_n^k \). So at time \( r_n^k \), the walk moves from \( k \) to \( k+1 \) with probability bounded from above by \( C/w(n) \) for some constant \( C > 0 \). Consequently, the probability that the walk is localized at \( \{k, \ldots, \ell\} \) is less than \( \prod_{n>0} \frac{C}{w(n)} \). By standard analysis, \( \prod_{n>0} \frac{C}{w(n)} = 0 \) if \( w(n) \to \infty \) as \( n \to \infty \). This leads to the desired result.

**D. Proof of Theorem 3.2**

We show how the thin-pancake property of saddle points is used to prove Theorem 3.2. Recall that an \( \epsilon \)-second-order stationary point is a point with a small gradient, and where the Hessian does not have a large negative eigenvalue. Let us put down the basic idea in Section 2.2 with the parameters in Algorithm 3 (PGDOT). If we are currently at an iterate \( x_t \) which is not an \( \epsilon \)-second-order stationary point, there are two cases: (1) The gradient is large: \( \|\nabla f(x_t)\| \geq g_{\text{thres}} \) (2) \( x_t \) is close to a saddle point: \( \|\nabla f(x_t)\| \leq g_{\text{thres}} \) and \( \lambda_{\min}(\nabla^2 f(x_t)) \leq -\sqrt{\rho \epsilon} \). The case (1) is easy to deal with by the following elementary lemma.

**Lemma D.1.** Assume that \( f : \mathbb{R}^d \to \mathbb{R} \) is \( \ell \)-gradient Lipschitz. Then for GD with step size \( \eta < \ell^{-1} \), we have \( f(x_{t+1}) - f(x_t) \leq -\frac{\eta}{2}\|\nabla f(x_t)\|^2 \).

The case (2) is more subtle, and the following lemma gives the decay of the function value after a random perturbation described in Algorithm 3 (PGDOT).

**Lemma D.2.** Assume that \( f : \mathbb{R}^d \to \mathbb{R} \) is \( \ell \)-gradient Lipschitz and \( \rho \)-Hessian Lipschitz. If \( \|\nabla f(x_t)\| \leq g_{\text{thres}} \) and \( \lambda_{\min}(\nabla^2 f(x_t)) \leq -\sqrt{\rho \epsilon} \), then adding one perturbation step as in Algorithm 3 followed by \( t_{\text{thres}} \) steps of GD with step size \( \eta \), we have \( f(x_{t+t_{\text{thres}}} - f(x_t) \leq -f_{\text{thres}} \) with probability at least \( 1 - \frac{d t_{\text{thres}}}{\sqrt{\rho \epsilon}} e^{-\chi} \).

(Jin et al., 2017) proved Lemma D.2 for PGD, and used it together with Lemma D.1 to prove Theorem 2.5. We will use the same argument, with Lemmas D.1 and D.2, leading to Theorem 3.2 for PGDOT.

Now, let us explain how to prove Lemma D.2 via a purely geometric property of saddle points. Consider a point \( \bar{x} \) satisfying the condition \( \|\nabla f(\bar{x})\| \leq g_{\text{thres}} \) and \( \lambda_{\min}(\nabla^2 f(\bar{x})) \leq -\sqrt{\rho \epsilon} \). After adding the perturbation in Algorithm 3, the resulting vector can be viewed as a distribution over the cube \( C(d)(\bar{x}, r/\sqrt{d}) \). Similar as in (Jin et al., 2017), we call \( C(d)(\bar{x}, r/\sqrt{d}) \) the perturbation cube which is divided into two regions: (1) escape region \( \chi_{\text{escape}} \) which consists of all points \( x \in C(d)(\bar{x}, r/\sqrt{d}) \) whose function value decreases by at least \( f_{\text{thres}} \) after \( t_{\text{thres}} \) steps; (2) stuck region \( \chi_{\text{stuck}} \) which is the complement of \( \chi_{\text{escape}} \) in \( C(d)(\bar{x}, r/\sqrt{d}) \). The key idea is that the stuck region \( \chi_{\text{stuck}} \) looks like a non-flat thin pancake, which has a very small volume compared to that of \( C(d)(\bar{x}, r/\sqrt{d}) \). This claim can be formalized by the following lemma, which is a direct corollary of (Jin et al., 2017)[Lemma 11] as \( C(d)(\bar{x}, r/\sqrt{d}) \subseteq B(d)(\bar{x}, r) \).

**Lemma D.3.** Assume that \( \bar{x} \) satisfies \( \|\nabla f(\bar{x})\| \leq g_{\text{thres}} \) and \( \lambda_{\min}(\nabla^2 f(\bar{x})) \leq -\sqrt{\rho \epsilon} \). Let \( e_1 \) be the smallest eigendirection of \( \nabla^2 f(\bar{x}) \). For any \( \delta < 1/3 \) and any \( u, v \in C(d)(\bar{x}, r/\sqrt{d}) \), if \( u - v = \mu e_1 \) and \( \mu \geq \delta/(2\sqrt{d}) \), then at least one of \( u \) and \( v \) is not in the stuck region \( \chi_{\text{stuck}} \).

To prove Lemma D.2, it suffices to check that \( \mathbb{P}(\chi_{\text{stuck}}) \leq C\delta \) for some \( C > 0 \). This criterion is general for any (random) perturbation. Let \( O_1, \ldots, O_{2^d} \) be the orthants centered at \( \bar{x} \); that is, the space \( \mathbb{R}^d \) is divided into \( 2^d \) subspaces according to the coordinate signs of \( \bar{x} \). The symbol \( \text{sgn}(O_i) \in \{-1, 1\}^d \) denotes the coordinate signs of \( y - \bar{x} \) for any \( y \in O_i \). For
1 ≤ i ≤ 2^d, let
\[ p_i = \prod_{\text{sgn}(O_i)_k = -1} \frac{w(R^k_i)}{w(L^k_i) + w(R^k_i)} \prod_{\text{sgn}(O_i)_k = 1} \frac{w(L^k_i)}{w(L^k_i) + w(R^k_i)} \]
be the probability that the random perturbation drives \( \tilde{x} \) into \( C^{(d)}(\tilde{x}, r/\sqrt{d}) \cap O_i \). Consequently, \( P(\chi_{\text{stuck}}) = \sum_{i=1}^{2^d} p_i \frac{\text{Vol}(\chi_{\text{stuck}} \cap O_i)}{\text{Vol}(C^{(d)}(\tilde{x}, r/\sqrt{d}) \cap O_i)} \), where \( \text{Vol}(\cdot) \) denotes the volume of a domain. It is easy to see that \( \text{Vol}(C^{(d)}(\tilde{x}, r/\sqrt{d}) \cap O_i) = (r/\sqrt{d})^d \). By Lemma D.3 and the slicing volume bound (Ball, 1986), \( \text{Vol}(\chi_{\text{stuck}} \cap O_i) \leq \sqrt{2}(r/\sqrt{d})^{d-1} \delta r/\sqrt{d} \). Therefore, \( \text{Vol}(\chi_{\text{stuck}} \cap O_i) \leq \sqrt{2} \delta \) implying that \( P(\chi_{\text{stuck}}) \leq \sqrt{2} \delta \).

Note that this proof does not rely on the full history of states for \( L_t \) and \( R_t \). Thus, one can restrict the number of previous iterations as is done in Section 4 using the hyperparameter \( t_{\text{count}} \).

E. Hyperparameters in the numerical examples

Table 1 summarizes the hyperparameters used for the empirical studies in Section 4.

| Hyperparameters | d | # of steps | \( h \) | \( t_{\text{count}} \) | \( \eta \) | \( t_{\text{thres}} \) | \( g_{\text{thres}} \) | \( r \) | momentum | batch size |
|-----------------|---|------------|-----|----------------|----|----------------|------------|---|----------|---------|
| Example 1       | 4 | 2000       | 0.04| 200           | 0.1| 10            | 0.01       | 0.04 | 0.5      | -       |
| Example 2       | 16| 14000      | 0.04| 200           | 0.1| 50            | 0.1        | 0.1  | 0.5      | -       |
| Example 3 (1)   | 2 | 3000       | 1   | 200           | 0.01| 50            | 0.01       | 0.01 | 0.5      | -       |
| Example 3 (2)   | 10| 1200       | 1   | 200           | 0.001| 50           | 1          | 0.01 | 0.5      | -       |
| Example 4 (MNIST)|- | 200 (epochs)| 10^{12} | 50       | 0.01| 10            | 0.1        | 0.5  | 0.9      | 128     |
| Example 4 (CIFAR-10)|- | 200 (epochs)| 10^{12} | 50       | 0.01| 10            | 0.1        | 0.5  | 0.9      | 128     |

Note that the implementation of PAGDOT is almost the same as the implementation of PGDOT with one difference: instead of GD, Nesterov’s AGD is deployed. Also, for the last two sets of experiments, \( h \) is set to be \( 10^{12} \), a very large number. This way we are basically ignoring the hyperparameter \( h \).

All the experiments are conducted on either a local machine or Google Colab using a CPU. Each of the first three examples takes a few minutes to run, and each of the experiments in example 4 takes a few hours.

F. AMSGrad and RMSProp also fail

Figure 6 confirms that besides SGD and Adam, some of their variants such as AMSGrad and RMSProp also fail in the training process when the weights and biases of the simple MLPs are initialized with \( \mathcal{N}(-1, 0.01) \). Note that \( n_{\text{hidden}} \) is the number of neurons in the hidden layer of simple MLP.
Figure 6. Training simple MLPs with $\mathcal{N}(-1, 0.01)$ initialization in Example 4. Top three: MNIST; Bottom three: CIFAR-10.