Overview of Saddle Point Escaping Problem

Jingqi Zhu
Department of Mathematics
The University of Manchester
Oxford Road, Manchester, M13 9PL
jingli.zhu@student.manchester.ac.uk

Abstract

Gradient descent and it variations are workhorses of modern machine learning. A large number of works studied their performance of escaping saddle points in non-convex optimization. For gradient-based method, adding perturbation can make gradient descent escape saddle point efficiently, provided that the function is strict saddle and satisfies some smooth assumptions. A nearly dimension-free bound of number of iteration allows gradient descent escape saddle points almost for free. However, while using random initialization can guarantee an asymptotic escape from saddle point, algorithm can be slowed significantly by saddle points without additional perturbation. It is also provable that adaptive methods like Adam and Rmsprop can escape saddle points quickly, even faster than SGD.

1 Introduction

1.1 Optimization in machine learning setting

Minimizing population risk is the prime concern in machine learning. However, people do not have access to population risk, an important class of algorithms we usually consider is based on empirical risk minimization. i.e. consider empirical risk $R(a) = \frac{1}{n}\sum_{i=1}^{n} l(a, Z_i)$ as a proxy for population risk $r(a) = \mathbf{E}\, l(a, Z)$.

From the optimization point of view, finding local minima of population risk and finding local minima empirical risk is different, as the empirical risk is usually more poorly behaved due to sampling and has many shallow local minimum while population risk is more smooth (Figure 1). However, it is still widely used to use ERM framework to approximate population risk minimization because of the following two theorems.

Theorem 1 *Let $R : \mathcal{B} \to \mathbb{R}_+$ be empirical risk function $R(a) = \frac{1}{n}\sum_{i=1}^{n} l(a, Z_i)$. For any fixed $a \in \mathcal{B}$, the sequence $(R(a))_n$ converges almost surely to population risk $r(a)$:*

$$(R(a))_n \overset{a.s.}{\to} r(a),\ as\ n \to \infty \tag{1}$$

Theorem 2 *Under certain assumptions on the sampling process, $r(a)$ and $R(a)$ are uniformly close:*

$$||r(a) - R(a)||_\infty \leq \nu \tag{2}$$

where ν decreases with sample size n.

Recent study has found a way to approximate local minima of the smooth population risk r, while avoiding the local minima that exist only in the sampled empirical risk R by adding small perturbations

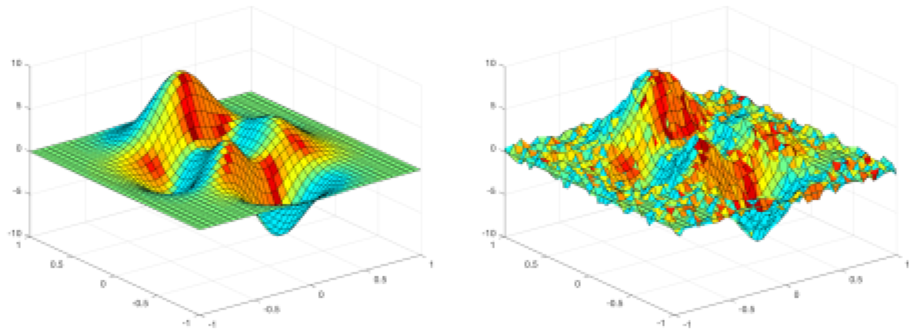

Figure 1: Population risk vs empirical risk.

[Jin et al., 2018]. Overall, ERM is still an optimal framework as a computable proxy of population risk minimization.

Under the ERM framework, the ultimate goal is

$$\underset{x \in \mathcal{C}}{minimize}\, f(x) \tag{3}$$

where $f(x)$ is the empirical risk and $\mathcal{C}$ is constraint set. The general solution to this optimization problem is gradient descent algorithm:

$$x_{k+1} = x_k - \eta \nabla f(x_k) \tag{4}$$

1.2 Problem of saddle point escaping

However, the gradient descent update is guaranteed to converge to a stationary point, which mean the output of algorithm can be local minima, saddle point or local maxima. Convex models, like linear regression and softmax, can easily avoid this problem as there is only local minima which is also the global minima and the convergence rate is provable by classic convex optimization theory. Unfortunately, most of the problems in modern machine leaning is non-convex, like MLP,CNN,RNN,etc., where there are a large number of unidentified stationary points. If we stick to traditional gradient descent, the algorithm may get stuck in one of those point forever. So the great challenge is how to get rid of these points efficiently and force gradient descent converge to global minima quickly on non-convex landscape.

Bad local minima used to thought to be the main problem on optimization of machine learning, however it is provable that modern machine learning models either have no local minima or stochastic gradient descent seems easily find global minima. For example, if a neural network is big and redundant enough, which is a usual case, we can easily optimize the function to zero and obtain the global minima. Likewise, local maxima can be avoided as well.

Actually, Saddle points are our real nemesis, because saddle points abound in modern machine learning architectures. A stationary point is a local minima if all eigenvalues of Hessian matrix is positive, while a stationary point is a saddle point as long as there is one eigenvalue is negative. So, we have a greater probability of getting saddle points, compared to local minima. Another point is that saddle points severely cause the learning curve to flatten out. We often see a learning curve that drops quickly, and then remains flat for a long time. This is the performance of being close to a saddle point. After running away from the saddle point we may run into another saddle point. That is why we usually see a learning curve going step down like this picture. To some extent, this is not a problem if we finally get the correct answer. But it is possible to come across a saddle point and stay there for such a long time that we do not know where we can find a better answer, especially when running time of algorithm is limited.

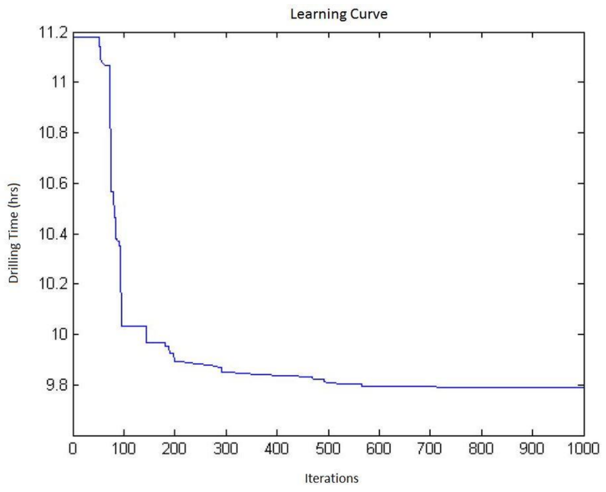

Figure 2: Step shape learning curve caused by saddle points.

The problem is more obvious when the model has millions of dimensions, as it takes time to find a decreasing way out from so many directions. Admittedly, if we have access to Hessian matrix, we can know all directions and easily distinguish which direction towards the way out. But computing Hessian matrix at each point can be very computationally costly. So the real problem lies in how to escape saddle points only using gradient-based method.

2 Preliminaries

2.1 Gradient Lipschitz and Hessian Lipschitz

Definition 1 *A differentiable function f is l-gradient Lipschitz if:*

$$\|\nabla f(\mathrm{x}_1) - \nabla f(\mathrm{x}_2)\| \leq \ell \|\mathrm{x}_1 - \mathrm{x}_2\| \quad \forall \mathrm{x}_1, \mathrm{x}_2 \tag{5}$$

.

Definition 2 *A twice-differentiable function f is ρ-Hessian Lipschitz if:*

$$\left\|\nabla^2 f(\mathrm{x}_1) - \nabla^2 f(\mathrm{x}_2)\right\| \leq \rho \|\mathrm{x}_1 - \mathrm{x}_2\| \quad \forall \mathrm{x}_1, \mathrm{x}_2 \tag{6}$$

2.2 Stationary point

Definition 3 *For a differentiable function f, x is a first-order stationary poin) if $\nabla f(\mathrm{x}) = 0$.*

Definition 4 *For a differentiable function f, x is an ϵ-first-order stationary point if $\|\nabla f(\mathrm{x})\| \leq \epsilon$.*

Definition 5 *For twice-differentiable function $f(\cdot), \mathrm{x}$ is a second-order stationary point if*

$$\nabla f(x) = 0, \; and \; \nabla^2 f(x) \geq 0. \tag{7}$$

Definition 6 *For a ρ-Hessian Lipschitz function $f(\cdot), x$ is an ϵ-second-order stationary point if*

$$\|\nabla f(x)\| \leq \epsilon \; and \; \nabla^2 f(x) \geq -\sqrt{\rho\epsilon} \cdot \mathbf{I}. \tag{8}$$

2.3 Strict saddle point

Definition 7 *For a twice-differentiable function f, x is a strict saddle point if x is a stationary point and $\lambda_{\min}\left(\nabla^2 f(x)\right) < 0$.*

3 Main Result

To solve the saddle point escaping problem, mathematicians put forward two main variation schemes based on the traditional gradient descent method: (1) adding perturbations; (2) random initialization.

3.1 Adding perturbation

For a twice differentiable function $f(x)$, a point x is a stationary point, if $\nabla f(x) = 0$. For a stationary point, it is a saddle point, if $\nabla^2 f(x)$ has both positive and negative eigenvalues. This criteria causes a identification problem in degenerate scenarios: $\nabla^2 f(x)$ can be positive semi-definite, where the point can be a local minima or a saddle point. To get rid of this degenerate case, we define the strict saddle property.

Definition 8 (strict saddle property) *A twice differentiable function $f(x)$ is strict saddle, if all its local minima have $\nabla^2 f(x) > 0$ and all its other stationary points satisfy $\lambda_{min}(\nabla^2 f(x)) < 0$.*

As strict saddle property guarantees any saddle point a decreasing direction, we still do not know the local improvement around the saddle point. To be safe, we define a robust version of strict saddle property.

Definition 9 (robust strict saddle property) *A twice differentiable function $f(x)$ is $(\alpha, \gamma, \epsilon, \delta)$-strict saddle, if for any point x at least one of the following holds*
1. $||\nabla f(x)|| \geq \epsilon$
2. $\lambda_{min}(\nabla^2 f(w)) \leq -\gamma$
3. There is a local minimum x^*, s.t. $||x - x^*|| \leq \delta$ and the function $f(x')$ restricted to $||x' - x^*|| \leq 2\delta$ is α-strongly convex

Under this definition, the local region of every point x looks like one of the following pictures:

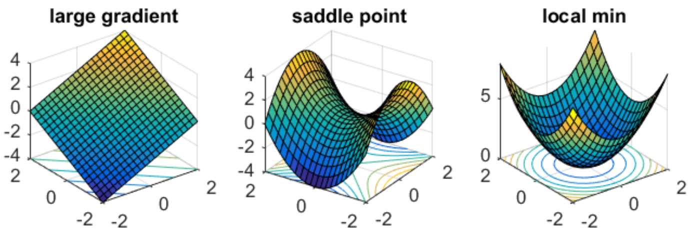

Figure 3: Three kinds of local landscape.

The study of strict saddle function is meaningful, as in many scenarios we are dealing with strict saddle functions. Ge et al. 2015 showed a tensor decomposition problem is strict saddle. Sun et al. 2015 observed that problems like complete dictionary learning, phase retrieval are also strict saddle.

Inspired by the fact that saddle points are unstable is dynamical systems, Ge et al. add perturbation at each step of update, making gradient descent 'fall' from saddle points, which is formalized as following:

Definition 10 (noisy gradient descent, a.k.a. perturbed gradient descent) *Adding a perturbation in each update*

$$x_{k+1} = x_k - \eta \nabla f(x_k) + \epsilon \tag{9}$$

where ϵ is a noise vector with mean 0.

It can be shown that, by adding noise at each step, the noise does not interfere with convergence and noisy gradient descent can escape from all saddle points in polynomial number of iterations, provided that the objective function satisfies strict saddle property [Ge et al. 2015].

However, the bound $O(poly(\frac{d}{\epsilon}))$ is highly dependent on dimension d and the smallest eigenvalue of the Hessian, thus is not very practical. For example, when people are dealing with millions number of dimension, $O(d^4)$ is catastrophic and not helpful indeed. An optimal convergence rate for strict saddle problems is an open problem.

3.2 Random initialization

Lee et al. 2016 published a novel idea that people can choose initial point randomly to avoid saddle point, without adding additional perturbation. Intuitively, under such randomness, even simple algorithms like gradient descent with constant step sizes is unlikely to converge to saddle points.

Theorem 3 *Let $f(x)$ be a twice differentiable function and x^* is a strict saddle point. Assume that step-size $0 < \alpha < \frac{1}{L}$, then*

$$P(\lim_{k \to \infty} x_k = x^*) = 0 \tag{10}$$

where $L = \max_i |\lambda_i|$, λ_i are eigenvalues of Hessian matrix of the strict saddle point.

This result indicates that with random initialization and non-aggressive step-size, the gradient descent can avoid saddle points and converge to local minima with probability one. To illustrate intuitively, let's consider the case of a non-convex quadratic $f(x) = \frac{1}{2} \sum_{i=1}^{d} a_i x_i^2$, where a_i is non-negative for k values and strictly negative for $d - k$ values. Then $x = 0$ is the unique stationary point and Hessian is $diag(a_1, a_2, ..., a_d)$. Consider gradient descent updates:

$$x^{(k+1)} = x^k - \eta \nabla f(x^k) \tag{11}$$

Moving term, it can be transformed to component-wise form

$$x_i^{k+1} = (1 - \eta a_i) x_i^k \tag{12}$$

For any initial point $x^{(0)}$,

$$x_i^k = (1 - \eta a_i)^k x_i^0 \tag{13}$$

This expression makes it easier to observe that when all a_i are positive and $\eta |a_i| < 1$, the iteration converge to 0, the stationary point. But if there is a single negative a_i,the function diverges to negative infinity exponentially quickly from any randomly chosen initial point. This example shows that the case of converging to saddle point can be fragile intuitively by random initialization and gradient descent can asymptotically converge to local minima.

Lee et al. made this result provable by the Stable Manifold Theorem from dynamical systems.

Theorem 4 (stable manifold theorem) *Let 0 be a fixed point for the C^r local diffeomorphism $\phi : U \to E$, where U is a neighborhood of 0 in the Banach space E. Suppose that $E = E_s \oplus E_w$, where E_s is the span of the eigenvectors corresponding to eigenvalues less than or equal to 1 of $D\phi(0)$, and E_u is the span of the eigenvectors corresponding to eigenvalues greater than 1 of $D\phi(0)$. Then there exists a C^r embedded disk W_{loc}^{cs} that is tangent to E_s at 0 called the local stable center manifold. Moreover, there exists a neighborhood B of 0, such that $\phi(W_{loc}^{cs}) \cap B \subset W_{loc}^{cs}$ and $\cap_{k=0}^{\infty} \phi^{-k}(B) \subset W_{loc}^{cs}$.*

This stable manifold theorem says that if there is a map that diffeomorphically deforms a neighborhood of a stationary point, then there exist a local stable center manifold W_{loc}^{cs} which contains the stationary point. This manifold has dimension equal to the number of eigenvalues of the Jacobian of the stationary points that are less than 1. W_{loc}^{cs} contains all points that are locally forward non-escaping, which means in a smaller neighborhood B, point converges to x^* after iterating ϕ only if it is in $B \cap W_{loc}^{cs}$.

They were able to show that the gradient descent algorithm satisfied the assumptions of the Stable Manifold Theorem, and, moreover, that the set of points that converge to strict saddles always has measure zero.

Although the result that gradient descent can eventually fall to local minima seems exciting, it is still unsatisfactory, as there is no quantitative convergence bound. How many numbers of iteration it make take to escape saddle point and converge to local minima is yet unclear. In this sense, Ge et al. 2015's work seems more satisfactory, they can gave a polynomial bound of iteration number after all, even though it was still not good enough. In fact, the worry about convergence bound was right, Du et al. 2017 have proved that even with random initialization, GD can take exponential time to escape saddle points. This result must have disappointed those people who was favor of the idea of random initialization, but it stressed the need of understanding the saddle point escaping problem once more and raised a new wave of finding iteration bound of saddle escaping with various methods.

3.3 Nearly dimension free bound

Asymptotic, even polynomial results are generally important, but they do not explain well why many gradient-based algorithms work very well on actual non-convex problems. And they can't guarantee that when the user observes a relatively flat learning curve, whether it is near the saddle point or has converged to a local minimum. Finally, they cannot guarantee that GD can solve high-dimensional problems as quickly as in convex optimization in non-convex optimization. In other words, previous works explained why gradient descent method can avoid saddle points, but not why it is efficient. In general, a really useful result would be a bound that is better than polynomial or even dimension free. This advance should be possible in theory, because otherwise we can not explain why gradient descent and its variations can have good performance in actual experiments.

In 2017, Jin et al. published their work *How to Escape Saddle Points Efficiently* on ICML, announced that gradient descent can escape saddle points and converge to local minima in a number of iterations that is almost dimension free.

Their work is the first sharp analysis that shows that perturbed gradient descent finds an approximate second-order stationary point in at most $polylog(d)$ iterations. This result is still poly-logarithmically dependent on dimension d, but it is a huge advance from Ge et al. 2015's bound $O(poly(d/\epsilon))$ and Levy 2016's bound $O(d^3poly(1/\epsilon))$. For instance, when d of a neural network is a million, d^3 is 10^{18}, however $log^4(d)$ is equal to 1296, about 10^{15} times smaller than the polynomial bound. Though it is not dimension-free, the bound grows super slow with the number of dimension, especially in high dimension scenarios. That is good enough for backing up the theory of modern machine learning in practice. That is why it is called almost dimension-free.

Table 1: Iteration complexity of gradient-based algorithms.

Algorithm	Iteration	Oracle
Ge et al.(2015)	$O(poly(d/\epsilon))$	Gradient
Levy (2016)	$O(d^3poly(1/\epsilon))$	Gradient
Jin et al.(2017)	$O(log^4(d)/\epsilon^2)$	Gradient

In the paper, they formally assume two standard conditions for smoothness of function:

Assumption 1 *f is l-gradient Lipschitz, i.e.*

$$\forall x_1, x_2, |\nabla f(x_1) - \nabla f(x_2)| \leq l|x_1 - x_2| \tag{14}$$

Assumption 2 *f is ρ-Hessian Lipschitz, i.e.*

$$\forall x_1, x_2, |\nabla^2 f(x_1) - \nabla^2 f(x_2)| \leq \rho|x_1 - x_2| \tag{15}$$

Recall that when we want to study the convergence to a first order stationary point, we try to bound the number of iterations to find a ϵ-first order stationary point, $|\nabla f(x)| \leq \epsilon$. Likewise, when dealing with the convergence to a second order stationary point, $\nabla f(x) = 0$, $\lambda_{min}(\nabla^2 f(x)) \geq 0$, we can define an ϵ-version of definition:

Definition 11 *A point x is an ϵ-second order stationary point if*

$$|\nabla f(x)| \leq \epsilon \tag{16}$$

$$\lambda_{min}(\nabla^2 f(x)) \geq -\sqrt{\rho\epsilon} \tag{17}$$

where ρ is consistent to the constant in Hessian Lipschitz assumption.

Regarding the bound of number of iteration, in classical analysis of first order stationary points, there is a very favorable property for gradient descent:

Theorem 5 (Nesterov 1998) *If f if l-gradient Lipschitz, then gradient descent with $\eta = \frac{1}{l}$ can find an ϵ-first order stationary point in $\frac{2l(f(x_0)-f^*)}{\epsilon^2}$ number of iterations, where x_0 is the initial point and f^* is the global minimum.*

This theorem suggests that a stationary point can be found in $O(\frac{1}{\epsilon^2})$ iterations, a completely dimension-free result. Now that we deal with a similar problem for second order stationary points, we wish those good properties hold for second order case as well. Thus it is natural to put forward these three questions:
1. Can we achieve a dimension-free number of iterations?
2. Can we obtain an $O(\frac{1}{\epsilon^2})$ convergence rate?
3. Can we have the same dependence on l and $f(x_0) - f^*$ as the Nesterov's theorem?
Surprisingly, up to small log factors, the answer is Yes to all three questions.

Theorem 6 *If assumption 1 and 2 hold, the perturbed gradient descent with $O(\frac{1}{l})$ can find an ϵ-second order stationary point in $\tilde{O}(\frac{l(f(x_0)-f^*)}{\epsilon^2})$ iterations with high probability, where $\tilde{O}$ hides some logarithmic factors.*

In fact, their dimension dependence is only $log^4(d)$. The theorem indicates that under the Hessian-Lipschitz condition, perturbed gradient descent converges to a second-order stationary point efficiently, with almost the same time for gradient descent to converge to a first-order stationary point. Hence, they claim perturbed method can escape saddle points almost for free.

3.4 Stuck region of noise

The official proof of the Jin et al.'s poly-logarithmic bound is rather complicated, they used 17 lemmas and some differential geometry to prove the result. The general idea is that, since the idea of random initialization seems to be a dead end, Jin et al. utilized the perturbed gradient method and made some adaptions. Instead of adding perturbation at each step, they only add it when it satisfies a certain condition: 1.The norm of gradient is smaller than a constant g, which means the current point may be near a saddle point; 2. There was no additional perturbation added in last t iterations, which means they add perturbation at least every t step. Assume that t steps after the last perturbation, the decreased amount compared to last perturbed time is less than a constant f, then it is believed that it has arrived near a local minima, the algorithm stops.

Their work also asserts that only if the perturbation x_0 lands in the set $\left\{ x||e_1^\top x\right|^2 \leq O(1/d)\right\} \cap \mathcal{B}_0(1)$, will we take a long time to escape the saddle point. They call it pancake-shape stuck region, the green object in the graph. It is also provable that the stuck region it very thin: the thickness can be at most $O(\frac{1}{\sqrt{d}})$. So, in general, there is little chance for random perturbation to land in the stuck region.

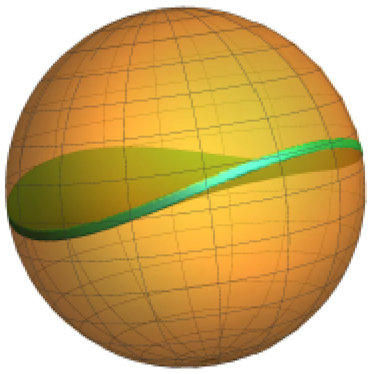

Figure 4: Perturbation ball and stuck region.

3.5 Necessity of perturbations

Their advanced version of paper (Jin et al. 2019) also discussed the necessity of adding perturbations. Although gradient descent with only random initialization can be significantly slowed down by saddle points, the behaviour of perturbed gradient descent is strikingly good: it can escape saddle points almost for free under some smooth assumptions. It shows that perturbations are necessary indeed. This necessity can be confirmed by experiments: Let both gradient descent and perturbed gradient descent with random initialization travel in same landscape, both of them visit the vicinity of d strict saddle points and reach a local minimum in the end. The gradient descent without perturbation gets closer and closer to the later saddle points, thus takes more time to escape. However, perturbed gradient descent is always able to run away from saddle points in a relatively small number of steps.

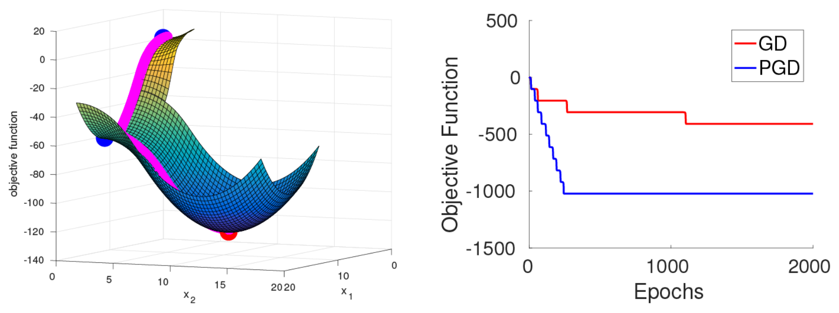

Figure 5: Decreasing path and learning curve: perturbed gradient descent vs gradient descent.

3.6 Adaptive method saddle point escaping

The work of Jin et al. 2019 is a beautiful summary of recent year advance in saddle point escaping topic. But how about adaptive methods, like Adam and RMSProp. Those are widely used in large-scale machine leaning but not well understood. In 2020, Staib et al. provided the first second-order convergence result of adaptive methods, and showed that adaptive methods escape saddle points faster than SGD.

Their novel idea is to view adaptive methods as preconditioned SGD, where the preconditioner is estimated in an online manner. They prove that the preconditioner rescales the stochastic gradient noise to be isotropic near stationary points, which helps escape saddle points. On the other hand, adaptive methods can efficiently estimate the aforementioned preconditioner. Combining the two parts, they proved the reuslt of the second-order convergence of adaptive methods.

4 Conclusion

Gradient descent and it variations are workhorses of modern machine learning. A large number of works studied their performance of escaping saddle points in non-convex optimization. For gradient-based method, adding perturbation can make gradient descent escape saddle point efficiently, provided that the function is strict saddle and satisfies some smooth assumptions. A nearly dimension-free bound of number of iteration allows gradient descent escape saddle points almost for free. However, while using random initialization can guarantee an asymptotic escape from saddle point, algorithm can be slowed significantly by saddle points without additional perturbation. It is also provable that adaptive methods like Adam and Rmsprop can escape saddle points quickly, even faster than SGD.

References

[1] Simon S Du, Chi Jin, Jason D Lee, Michael I Jordan, and Aarti Singh, Barnabas Poczos. Gradient descent can take exponential time to escape saddle points. In NIPS, 2017.

[2]Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In COLT, 2015.

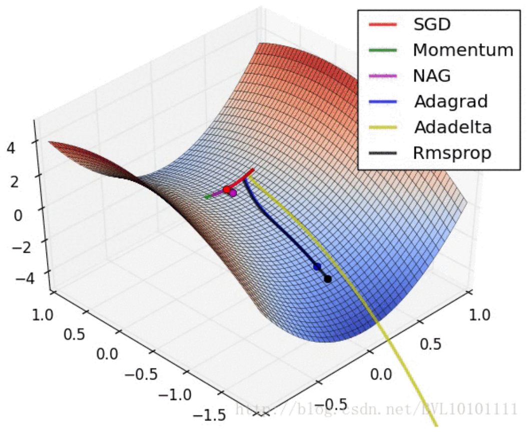

Figure 6: Comparison of escaping rate among different methods.

[3]Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. arXiv preprint arXiv:1703.00887, 2017.

[4]Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent converges to minimizers. University of California, Berkeley, 1050:16, 2016.

[5]Chi Jin, Lydia T. Liu, Rong Ge, and Michael I. Jordan. On the Local Minima of the Empirical Risk. In NIPS, 2018.

[6]Shuang Li, Gongguo Tang, and Michael B. Wakin. The Landscape of Non-convex Empirical Risk with Degenerate Population Risk. arXiv: 1907.05520, 2019.

[7]Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M. Kakade, and Michael I. Jordan. On Nonconvex Optimization for Machine Learning: Gradients, Stochasticity, and Saddle Points. arXiv: 1902.04811, 2019.

[8]Matthew Staib, Sashank J. Reddi, Satyen Kale, Sanjiv Kumar, and Suvrit Sra. Escaping Saddle Points with Adaptive Gradient Methods. arXiv:1901.09149, 2020.

[9]Catherine F. Higham, and Desmond J. Higham. Deep Learning: An Introduction for Applied Mathematicians. arXiv:1801.05894, 2018.