How to Escape Saddle Points Efficiently

CS 149: 21st Century Algorithms

Barry Yang and Devina Kumar

November 7, 2018

Problem Statement

This paper is concerned with how to escape saddle points efficiently, even in cases pertaining to non-convex functions. Ultimately, it shows that gradient descent modified with appropriate perturbations escapes saddle points efficiently.

In convex optimization, gradient descent finds a first-order stationary point independent of the number of dimensions. In non-convex settings, convergence to first-order stationary points is more difficult, as these points could be local maxima or saddle points. Past literature suggests finding local minima is sufficient in place of finding global minima. However, saddle points correspond to suboptimal solutions and occur frequently and in high-dimension, non-convex optimization problems. Though some papers suggest workarounds to this problem, such as adding noise at each step of gradient descent, these solutions often are inefficient and have runtimes related to the number of dimensions, considerably slowing runtimes compared to rates of convergence to first-order stationary stationary points.

Given the above, the authors of the paper are concerned with whether gradient descent can escape saddle points and converge to local minima in time independent of the number of the dimensions. Specifically, the paper investigates the time complexity of converging to second-order stationary points, as second-order stationary points are local minima under the assumption that all saddle points are strict.

Relevance

Many of problems in machine learning rely on optimization of some kind. Gradient descent is one of the most widely utilized optimization algorithms in machine learning models. In fact, it is one of the most widely used algorithms in machine learning, period. Part of the appeal of gradient descent lies in its efficiency even when applied in high-dimensional settings. Gradient descent

can reach a point with a small gradient within a number of iterations that is independent of dimension. In fact, for functions that are l-smooth (see Definitions section below), gradient descent can find a point x with $||\nabla f(x)|| \leq \epsilon$—known as an ϵ-first-order stationary point—within $l(f(x_0) - f^*)/\epsilon^2$ iterations, where x_0 is the initial point and f^* is the optimal value of f.

While this bound is highly useful for convex optimization, in which finding an ϵ-first-order stationary point is akin to finding a global optimum, first-order stationary points in non-convex settings can indicate a variety of point types, including local or global minima, local maxima, and saddle points. Many problems of interest in the field of machine learning—matrix completion, principle component analysis, low-rank models, tensor decomposition, and deep neural networks—deal with non-convex settings, which makes non-convex optimization a highly salient topic.

Despite the numerous possible things a first-order stationary point can indicate in a non-convex setting, research has shown that some types of first-order stationary points are perhaps more useful than others. For some non-convex problems, it suffices to find a local minimum instead of a global minimum (Ge et al., 2015; Sun et al., 2016b; Bhojanapalli et al., 2016). Saddle points, however, can correspond to highly suboptimal solutions for non-convex problems. Research indicates that non-convex optimization problems in high dimensions almost requisitely contain saddle points (Dauphin et al., 2014). Given that saddle points can cause gradient descent to get "stuck" at a highly suboptimal points, methods to escape saddle points are highly useful in non-convex optimization problems.

Under certain conditions, methods to escape saddle points exist, such as adding noise to the step (Ge et al., 2015) or random initialization (Lee et al., 2016). However, the bound on the number of iterations for these methods to converge is still dependent on dimension, leaving us with the problem of gradient descent getting stuck or being very slow in high-dimensional, non-convex optimization settings. Thus, this paper's examination of escaping saddle-points and converging to local minima in a number of iterations that is almost dimension-free is highly relevant to ongoing work in the field of machine learning and addresses an issue that is ubiquitous across high dimensional, non-convex problems.

Related Work

According to the authors, there has been much past literature regarding finding convergence for non-convex optimization problems. However, many of these solutions are often problem-specific and hard to generalize to other

non-convex optimization problems. Nevertheless, the authors categorized past bodies of literature that focused on solutions for for general non-convex optimization problems: Hessian-based solutions, Hessian-vector-product-based solutions, and gradient-based solutions.

Hessian-based Solutions. These algorithms rely on computing the Hessian, or a second-order derivative matrix, to find second-order stationary points as opposed to first-order stationary points. The idea is that second-order optimization methods converge to second-order stationary points. When the eigenvalues are positive for a given $\nabla f(x)$, then x must be a local minimum. Otherwise, x may be a local maximum or a saddle point. The authors focus on two algorithms that rely on computing the Hessian to find second-order stationary points: cubic regularization (Nesterov and Polyak, 2006) and trust region (Curtis et al., 2014) algorithms, both of which can find local minima efficiently. However, the paper points out that because these algorithms rely on computing the Hessian per iteration, they are often times too expensive to be put into practice, as many matrix operations would have to be applied onto a large data set per iteration.

Hessian-vector-product-based Solutions. Instead of calculating the full Hessian, some past papers explore using only Hessian-vector products in order to find second-order stationary points. This would entail using a Hessian-vector product oracle: given function f, point x and direction u, the oracle returns $\nabla^2 f(x) \cdot u$. Some examples the paper uses are Agarwal et al. (2016), Carmon et al (2016), and Carmon and Duchi (2016), all of whom use the Hessian vector product to find second order stationary points. The paper suggests that the these algorithms can be implemented efficiently in roughly the same time complexity as gradient-based algorithms. However, the paper does not focus on these Hessian-based algorithms and instead focuses on gradient-based Oracles.

Gradient-based Solutions. Finally, the paper also discusses work that explores convergence to second-order stationary points without computing the Hessian. These solutions include algorithms such as stochastic gradient descent and perturbed gradient descent. For example, the idea behind stochastic gradient descent is that, given only the gradient of the function, randomly adding noise to a point ("nudging it") is enough to escape a saddle point. This is often extremely efficient as it is less expensive to randomly calculate a noisy gradient than to calculate a true gradient, and it is obviously less expensive than computing the Hessian matrix. Previously, the complexity of stochastic gradient descent was unknown, but Ge et al. (2015) showed that convergence to a second-order stationary point is bound in polynomial time, specifically $\text{poly}(d/\epsilon)$ iterations. Levy (2016) expanded on this by showing that normalized gradient descent converges in $O(d^3 \cdot \text{poly}(1/\epsilon))$ time. This paper finally expands upon latter's run time by showing that perturbed gradient

descent can converge to second-order stationary points up to polylog factors, which matches the run time of converging to first-order stationary points.

Analysis

The general result of this paper is an analysis that shows that perturbed gradient descent can be used to find an approximate second-order stationary point in at most $polylog(d)$ iterations. Its main contributions include the following:

- For l-gradient Lipschitz, ρ-Hessian Lipschitz functions, gradient descent finds an ϵ-second-order stationary point within $\tilde{O}(l(f(x_0) - f^*/\epsilon^2)$ iterations, where $\tilde{O}(\cdot)$ hides polylog factors.
- Under a strict-saddle condition, the convergence rate from above can be directly applied for finding local minima.

Algorithm

The paper analyzes the the "Perturbed Gradient Descent" algorithm, which is a modification of gradient descent that adds a small perturbation when likely near a saddle point.

Perturbed Gradient Descent

for $t = 0, 1, ...$ **do**
  if $||\nabla f(x_t)|| \leq g_{\text{thres}} \wedge t - t_{\text{noise}} > t_{\text{thres}}$ **then**
    $t_{\text{noise}} \leftarrow t$
    $\tilde{x_t} \leftarrow x_t$
    $x_t \leftarrow \tilde{x_t} + \xi_t$, ξ_t uniformly $\sim B_o(r)$
  if $t - t_{\text{noise}} = t_{\text{thres}} \wedge f(x_t) - f(\tilde{x}_{t_{\text{noise}}}) > -f_{\text{thres}}$ **then**
    return $\tilde{x}_{t_{\text{noise}}}$
  $x_{t+1} \leftarrow x_t - \eta \nabla f(x_t)$

How the Algorithm Works. At each time step, the algorithm checks for several conditions. When the norm of the current gradient is below a certain threshold (which would suggest that we might be close to a saddle point), the algorithm adds a small random perturbation to x_t, where the perturbation is uniformly sampled from a d-dimensional ball B centered at 0 with a suitably small radius. If the function value does not decrease enough after a certain threshold of iterations, then the algorithm returns the current value of x_t. This indicates that x_t should be a sufficiently "close" to a second-order stationary point and not a saddle point.

Definitions

To analyze the complexity of the perturbed gradient descent algorithm, it is helpful to review certain definitions. These definitions were taken from the paper.

- **l-smoothness.** A differential function f is l-smooth if

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

- **α-strongly convex.** A twice differentiable function f is $\alpha-$strongly convex if

$$\forall x, \lambda_{\min}(\nabla^2 f(x)) \geq \alpha$$

- **$\rho-$Hessian Lipschitz.** A twice differentiable function f is $\rho-$Hessian Lipschitz if

$$\forall x_1, x_2, \frac{||\nabla^2 f(x_1) - \nabla^2 f(x_2)||}{||x_1 - X_2||} \leq \rho$$

A strongly convex condition ensures that descent converges linearly to a unique stationary point (the minima). l-smoothness and ρ-Hessian Lipschitz ensure that the function behaves properly around saddle points (the gradient descent as well as the spectral norm both do not change drastically).

- **Saddle Point.** A point x is a saddle point if it is a first-order stationary point but not a local minimum.

  For convenience, the paper describes saddle points as both traditional saddle points and local maxima.

- **Strict Saddle Point.** For a twice differentiable function f, x is a strict saddle point if satisfies either of the following: (1) $\nabla f(x)$ is very large, or (2) if $\lambda_{\min}(\nabla^2 f(x)) < 0$.

- **Second Order Stationary Point.** For a $\rho-$Hessian Lipschitz function f, point x is a second-order stationary point if:

$$||\nabla f(x)|| \leq 0 \text{ and } \lambda_{\min}(\nabla^2 f(x)) \geq 0$$

  A point x is a $\epsilon-$second order stationary point if

$$||\nabla f(x)|| \leq \epsilon \text{ and } \lambda_{\min}(\nabla^2 f(x)) \geq -\sqrt{\rho\epsilon}$$

  According to the paper, when all saddle points are strict, all second-order stationary points are local minima.

Main Results

Given that the function f is both l-smooth and p-Hessian Lipschitz, one can say that f is well-behaved near a saddle point, making it possible to escape from saddle points with a small perturbation. "Well-behaved" refers to the shape around the saddle point: (1) l-smooth ensures the gradient does not change too rapidly, and (2) ρ-Hessian Lipschitz ensures the spectral norm also does not change rapidly. The paper formalizes this notion with the following theorem: given that f is both l-smooth and ρ-Hessian Lipschitz, there exists an absolute constant c_{max} such that for any $\delta > 0, \epsilon \leq \frac{l^2}{}$

, $\triangle_f \geq f(x_0) - f^*$, and constant $c \leq c_{max}$, the perturbed gradient descent algorithm described above will output an ϵ-second order stationary point, with probability $1 - \delta$, and terminate in the following number of iterations:

$$O(\frac{l(f(x_0) - f^*)}{\epsilon^2} \cdot \log^4 \frac{dl\triangle_f}{\epsilon^2 \delta}).$$

c_{max} represents a bound from which a range of parameters could be selected from. In addition, the algorithm is only concerned in situations where the condition $\epsilon \leq l^2/\rho$ applies. In cases where $\epsilon > l^2/\rho$, standard gradient descent suffices (as $\epsilon > l^2/\rho \implies \sqrt{\epsilon\rho} \leq -l \leq \lambda_{\min}(\nabla^2 f(x))$ under the definition of a ϵ-second order stationary point, which is also a strict saddle point).

Under a perturbed form of gradient descent with an added Hessian-Lipschitz condition, we can converge to a second-order stationary point in

$$\tilde{O}(\frac{l(f(x_0) - f^*)}{\epsilon^2})$$

This runtime complexity is almost the same time as is required for gradient descent to converge to a first-order stationary point. For comparison, standard gradient descent finds a first-order stationary point within $l(f(x_0) - f*)/\epsilon^2$ iterations.

This theorem holds true if we consider the behavior of the point that we are iterating. Suppose we have a point x. This point is either a second-order stationary point (in which case we are done), or it is not a second-order stationary point. If x is not a second-order stationary point and we iterate on it via this perturbed gradient descent algorithm, then by our definition of x, we know that either the gradient of x is large or the Hessian of x has a significant negative eigenvalue. If the gradient is large, then we will not have met our perturbation condition and will continue traditionally. However, if the gradient meets our perturbation threshold and the Hessian has a significantly negative eigenvalue, then by adding a perturbation to our gradient descent step and then running normal gradient descent for a couple iterations, our function value will will decrease by a certain value (which we represent in the algorithm as our function threshold) with high probability.

Functions with Strict Saddle Property

In addition to assuming the function is Hessian-Lipschitz, if we assume that all saddle points are strict, then we can say that not only does our perturbed gradient descent algorithm terminate in $\tilde{O}(l(f(x_0) - f^*/\epsilon^2)$, but also the algorithm converges to local minima (in the case of strict saddle points, all second-order stationary points are local minima). While it may seem unrealistic to assume this property, the paper claims that for many many practical non-convex problems, all saddle points are strict (including problems concerning orthogonal tensor decomposition, matrix completion, and principal components analysis). In these kinds of non-convex problems, it has been shown that local minima are also global minima. Thus, efficiently finding ϵ-second-order stationary points, in these cases, is akin to finding solutions to non-convex problems with global guarantees.

Extensions

The above results can be further improved to linear convergence when the function has local structure. The paper presents a way of applying gradient descent on functions that uphold standard β-smooth (where $\beta \leq l$) and α-strongly convex conditions. Certain low-rank problems, such as matrix sensing and matrix completion, often see the appearance of this regularity condition.

The final contribution of the paper is a new technique to bound the volume of the "band" of points where gradient descent gets stuck around a saddle point; by adding a random perturbation, the paper shows that the resulting point is very unlikely to be in the band, thus making efficient escape from a saddle point possible. The authors observe that near a saddle point, there exists a certain set of points for which it is very easy to get "stuck" using gradient descent. Approximating this "stuck" region as a flat set mandates using a very small step size, resulting in suboptimal runtimes. However, the paper notes that instead of trying to characterize the shape of the stuck region (which varies with dimensionality), one can note that the stuck region is likely very thin and then try to characterize the thinness of the stuck region. The paper bounds the thickness of the stuck region by $O(1/\sqrt{d})$. This bound on the thickness of the stuck region allows the perturbation algorithm to escape saddle points with high probability, which is foundational for the paper's main analysis.

Conclusion

Overall, the authors present an interesting and novel analysis of perturbed gradient descent that provides a nearly dimension-free result for the algorithm in a general non-convex setting. Although the assumptions the authors make may limit the applicability of this method for extremely complicated

non-convex settings, the paper's analysis is helpful in considering non-convex optimization methodologies and prompts research into ways that other modifications of gradient descent can lead to dimension-free guarantees.

Note: all references come from the References section of the paper.