Efficiently avoiding saddle points with zero order methods: No gradients required

Lampros Flokas*, Emmanouil V. Vlatakis-Gkaragkounis*, Georgios Piliouras

Columbia University & Singapore University of Technology and Design

Introduction

⇒ Given function $f : \mathbb{R}^d \to \mathbb{R}$, solving the problem

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$$

is one of the building blocks that many machine learning algorithms are based on.

⇒ Unfortunately access to gradient evaluations may be either computationally inefficient or even impossible.

Examples: *Hyper-parameter tuning, black-box adversarial attacks on DNN, computer network control, Simulation or Bandit optimization.*

Are there scalable zero order methods that can safely and efficiently avoid strict saddle points and always converge to local minima of $f(\mathbf{x})$?

Saddle points

⇒ Strict saddle points: $\nabla f = \mathbf{0}$ & $\lambda_{\min}(\nabla^2 f) < 0$

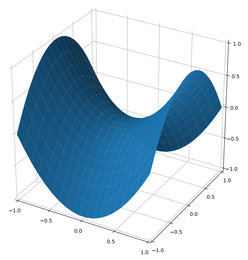

Figure: Strict saddle points are not second order stationary

⇒ Non strict saddle points $\nabla f = \mathbf{0}$ & $\lambda_{\min}(\nabla^2 f) = 0$
Can be local minimum or just second order stationary point.
If only strict saddles, convergence to SOSPs is enough.

Zero Order Convergence to SOSPs

⇒ Zero order trust region methods with quadratic models
Can take $\mathcal{O}(d^4)$ operations per iteration
⇒ Gaussian smoothing reduction to first order methods
Leads to $\mathrm{poly}(d, 1/\epsilon)$ slowdown compared to first order methods

Is that really necessary?

Finite differences

$$r_f(\mathbf{x}, h) = \begin{cases} \sum_{l=0}^{d} \dfrac{f(\mathbf{x} + h\mathbf{e}_l) - f(\mathbf{x})}{h}\mathbf{e}_l \text{ when } h \neq 0 \\ \nabla f(\mathbf{x}) \text{ if } h = 0 \end{cases}$$

Properties for ℓ-gradient Lipschitz f
⇒ Approximates well the gradient

$$\|\nabla f(\mathbf{x}) - r_f(\mathbf{x}, h)\| \leq \sqrt{d}\ell|h|$$

⇒ Is $\ell\sqrt{d}$ Lipschitz

$$\|r_f(\mathbf{y}, h) - r_f(\mathbf{x}, h)\| \leq \sqrt{d}\ell\|\mathbf{y} - \mathbf{x}\|$$

Approximate gradient descent (AGD)

$$\boldsymbol{\chi}_{k+1} = g_0(\boldsymbol{\chi}_k) \triangleq \begin{pmatrix} \mathbf{x}_{k+1} \\ h_{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_k - \eta r_f(\mathbf{x}_k, h_k) \\ \beta h_k \end{pmatrix}$$

→ When $\beta \in (0, 1)$ $\mathbf{x}_k$ same fixed points as GD.
But: Does it converge to SOSPs?

Differences with Gradient Descent

⇒ Non autonomous dynamical system: h_k changes
⇒ Does h_0 affect convergence to SOSPs?
⇒ Point-wise convergence for non-isolated fixed points?

Theorem (Convergence to minimizers)

Let $f : \mathbb{R}^d \to \mathbb{R} \in C^2$ be a ℓ-gradient Lipschitz function. Let us also assume that f is analytic, has compact sub-level sets and all of its saddle points are strict. Let $\eta < \min\{\frac{1}{\ell\sqrt{d}}, \frac{1}{2\ell}\}$ and $\beta < 1 - 2\eta\ell$. If we pick a random initialization point $\mathbf{x}_0$, then we have that for the $\mathbf{x}_k$ iterates of g_0

$$\forall h_0 \in \mathbb{R}: \quad \Pr(\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}^*) = 1$$

where $\mathbf{x}^*$ is a local minimizer of f.

Full version at `arxiv:TBA`

What about efficiency?

Gradient descent can take exponential time to converge to SOSPs
⇒ Perturbed GD (PGD): add noise to iterates when gradient is low
⇒ PGD converges to SOSPs in $\mathrm{polylog}(d)$ time
⇒ Does it work for AGD?

Challenges compared to PGD

We address:
⇒ Detecting low gradient only with function evaluations
⇒ Escaping strict saddles even if value of f is not decreasing
⇒ Escaping strict saddles without h_k going to zero
⇒ $\mathrm{polylog}(d)$ calls to r_f, same as ∇f calls in PGD up to constants

Theorem (Analysis of Perturbed AGD (PAGD))

If f is ℓ-gradient Lipschitz and ρ-Hessian Lipschitz, then for any $\delta > 0$, $\epsilon \leq \frac{\ell^2}{\rho}$, $\Delta_f \geq f(\mathbf{x}_0) - f^\star$, with probability $1 - \delta$, the output of PAGD will be an ϵ-SOSP, and use

$$\mathcal{O}\left(d\frac{\ell(f(\mathbf{x}_0) - f^\star)}{\epsilon^2}\log^4\left(\frac{d\ell\Delta_f}{\epsilon^2\delta}\right)\right)$$

evaluations of f.

Experiments

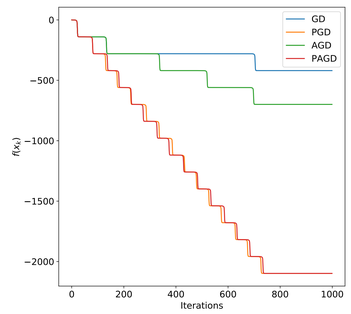

Figure: Octopus function with $d = 15$.

⇒ Octopus-like function with sequences of strict saddle points
⇒ GD and AGD take progressively longer time to escape each saddle
⇒ PAGD takes the same number of iterations to converge as PGD