Optimization for Deep Learning

Tutorial for OSU TDAI Deep Learning Summer School

Jia (Kevin) Liu

Assistant Professor
Department of Electrical and Computer Engineering
The Ohio State University, Columbus, OH, USA

June 1, 2022

Tutorial Outline

- Introduction
- Convexity
- First-Order Methods
- Zeroth-Order Methods
- First-Order Optimization for ML Problems with Special Geometric Structures

Part I

Introduction

Mathematical Optimization

Mathematical optimization problem:

$$\begin{array}{ll} \text{Minimize} & f_0(\mathbf{x}) \\ \text{subject to} & f_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m \end{array}$$

- $\mathbf{x} = [x_1, \ldots, x_N]^\top \in \mathbb{R}^N$: decision variables
- $f_0 : \mathbb{R}^N \to \mathbb{R}$: objective function
- $f_i : \mathbb{R}^N \to \mathbb{R}, i = 1, \ldots, m$: constraint fucntions

Solution or **optimal point** $\mathbf{x}^*$ has the smallest value of f_0 among all vectors that satisfy the constraints

Brief History of Optimization

Theory:

- Early foundations laid by many all-time great mathematicians (e.g., Newton, Gauss, Lagrange, Euler, Fermat, ...)
- Convex analysis 1900–1970 (Duality by von Neumann, KKT conditions...)

Algorithms

- 1947: simplex algorithm for linear programming (Dantzig)
- 1970s: ellipsoid method [Khachiyan 1979], 1st polynomial-time alg. for LP
- 1980s & 90s: polynomial-time interior-point methods for convex optimization [Karmarkar 1984, Nesterov & Nemirovski 1994]
- since 2000s: many methods for large-scale convex optimization

Applications

- before 1990: mostly in operations research, a few in engineering
- since 1990: many applications in engineering (control, signal processing, networking and communications, circuit design,...)
- since 2000s: machine learning

Solving Optimization Problems

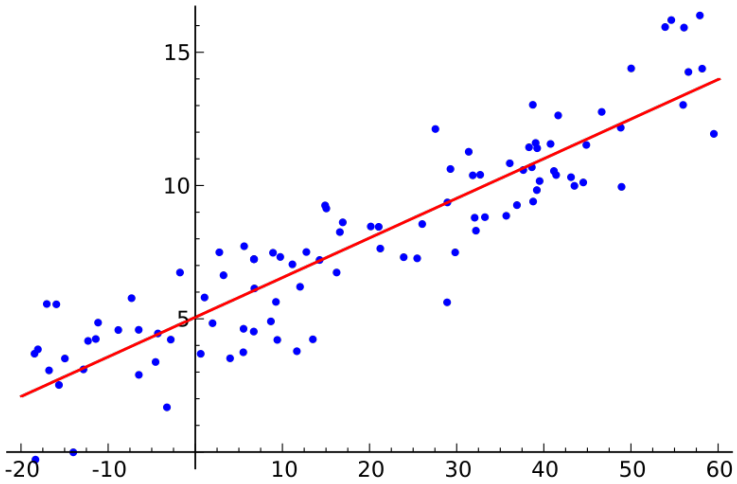
- General optimization problems
  - Very difficult to solve (NP-hard in general)
  - Often involve trade-offs: long computation time, may not find an optimal solution (approximation may be acceptable in practice)
- Exceptions: Problems with special structures
  - Linear programming problems
  - Convex optimization problems
  - Some non-convex optimization problems with strong-duality
- Watershed between Problem Hardness: Convexity
  - This course focuses on nonconvex problems arising from ML context

Applying Optimization Tools in Machine Learning

- Linear Regression
- Variable Selection & Compressed Sensing
- Support Vector Machine
- Logistic Regression (+ Regularization)
- Matrix Completion
- Deep Neural Network Training
- Reinforcement Learning
- Distributed/Federated/Decentralized Learning
- ...

Example 1: Linear Regression (Convex)

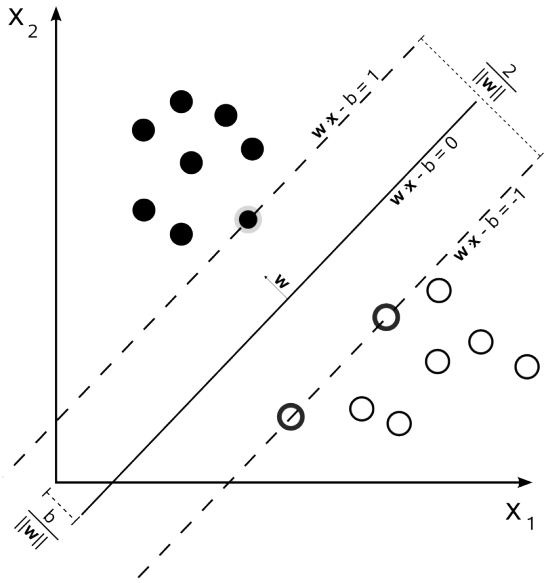
$$\text{Minimize}_{\beta} \quad \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$$



- Given data samples: $\{(\mathbf{x}_i, y_i), i = 1, \ldots, m\}$, where $\mathbf{x}_i \in \mathbb{R}^n$, $\forall i$
- Find a linear estimator: $y = \boldsymbol{\beta}^\top \mathbf{x}$, so that "error" is small in some sense
- Let $\mathbf{X} \triangleq [\mathbf{x}_1, \ldots, \mathbf{x}_m]^\top \in \mathbb{R}^{m \times n}$, $\mathbf{y} \triangleq [y_1, \ldots, y_m]^\top \in \mathbb{R}^m$
- Linear algebra for $\|\cdot\|_2$: $\boldsymbol{\beta}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ (analytical solution)
- Computation time proportional to $n^2 m$ (less if structured)
- Stochastic gradient if m, n are large

Example 2: Support Vector Machine (Convex)

- Given data samples: $\{(\mathbf{x}_i, y_i), i = 1, \ldots, m\}$
  - $\mathbf{x}_i \in \mathbb{R}^n$ called "feature vectors", $\forall i$
  - $y_i \in \{-1, +1\}$ are "labels"
- Linear classifier: $f(\mathbf{x}) = \text{sgn}(\mathbf{w}^\top \mathbf{x} + b)$:
  - $\mathbf{w} \in \mathbb{R}^n$: weight vector for features
  - $b \in \mathbb{R}$: Some "bias"



- Goal: To find a pair $(\mathbf{w}, b)$ to minimize a weighted sum such that
  - Minimize classification error on training samples
  - Robust to random noise in the training samples

$$\underset{\mathbf{w},b,\boldsymbol{\epsilon}}{\text{Minimize}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \epsilon_i$$

$$\text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \epsilon_i, \quad \epsilon_i \geq 0, \quad i = 1, \ldots, m$$

Nonconvex Optimization Problems in ML

- Lower complexity bound for solving general nonconvex problems
  - Consider, w.l.o.g., $\min_{\mathbf{x}\in[0,1]^d} f(\mathbf{x})$
  - f is nonconvex and L-Lipschitz-continuous, with global optimal $f^* > -\infty$
  - To find an ϵ-approximate solution $\hat{\mathbf{x}}$ (i.e., $f(\hat{\mathbf{x}}) - f^* \leq \epsilon$), number of iterations required: $\Omega(L^d \epsilon^{-d})$ (exponential)

Nonconvex Optimization Problems in ML

- Lower complexity bound for solving general nonconvex problems
  - Consider, w.l.o.g., $\min_{\mathbf{x}\in[0,1]^d} f(\mathbf{x})$
  - f is nonconvex and L-Lipschitz-continuous, with global optimal $f^* > -\infty$
  - To find an ϵ-approximate solution $\hat{\mathbf{x}}$ (i.e., $f(\hat{\mathbf{x}}) - f^* \leq \epsilon$), number of iterations required: $\Omega(L^d \epsilon^{-d})$ (exponential)
- Several ways to relax this challenging goal:
  - Finding hidden convexity or reformulate into an equivalent convex problem
    - ⋆ Need to exploit special problem structure as much as possible
    - ⋆ However, solution approaches cannot be generalized
  - Change the goal to finding a stationary point or a local extremum
    - ⋆ Often possible to obtain FO methods with polynomial dependence of the complexity on the dimension of the problem and desired accuracy
  - Identify a class of problems:
    - ⋆ General enough to characterize a wide range of applications (in ML)
    - ⋆ Allow one to obtain global performance guarantees of an algorithm
    - ⋆ E.g., Polyak-Lojasiewicz condition (linear convergence), α-weakly-quasi-convexity (sublinear convergence), etc.

Nonconvex Optimization Problems in ML

- Lower complexity bound for solving general nonconvex problems
  - Consider, w.l.o.g., $\min_{\mathbf{x}\in[0,1]^d} f(\mathbf{x})$
  - f is nonconvex and L-Lipschitz-continuous, with global optimal $f^* > -\infty$
  - To find an ϵ-approximate solution $\hat{\mathbf{x}}$ (i.e., $f(\hat{\mathbf{x}}) - f^* \leq \epsilon$), number of iterations required: $\Omega(L^d \epsilon^{-d})$ (exponential)
- Several ways to relax this challenging goal:
  - Finding hidden convexity or reformulate into an equivalent convex problem
    - ⋆ Need to exploit special problem structure as much as possible
    - ⋆ However, solution approaches cannot be generalized
  - Change the goal to finding a stationary point or a local extremum
    - ⋆ Often possible to obtain FO methods with polynomial dependence of the complexity on the dimension of the problem and desired accuracy
  - Identify a class of problems:
    - ⋆ General enough to characterize a wide range of applications (in ML)
    - ⋆ Allow one to obtain global performance guarantees of an algorithm
    - ⋆ E.g., Polyak-Lojasiewicz condition (linear convergence), α-weakly-quasi-convexity (sublinear convergence), etc.
  - But what if gradients are hard to obtain?
    - ⋆ E.g., reinforcement learning, blackbox adversarial attacks on DNN?
    - ⋆ Zeroth-order or derivative-free methods

Tractable Nonconvex Optimization Problems in ML

- Problems with hidden convexity or analytic solutions
  - Eigen-problems (e.g., PCA, multi-dimensional scaling, ...)
  - Non-convex proximal operators (e.g., Hard-thresholding, Potts minimization)
  - Some discrete problems (binary graph segmentation, discrete Potts minimization, nearly optimal K-means)
  - Infinite-dimensional problems (smoothing splines, locally adaptive regression splines, reproducing kernel Hilbert spaces)
  - Non-negative matrix factorization (NMF)
  - Compressive sensing with ℓ_1 regularization
- Problems with (global) convergence results
  - Phase retrieval problem
  - Low-rank matrix completion
  - Deep learning
- Problems with certain properties of symmetry
  - Rotational symmetry, discrete symmetry, etc.

Example 3: Deep Learning (Nonconvex)

- Example: Train an L-layer fully-connected NN for supervised learning:

$$\min_{\mathbf{W}} \left\{ F(\mathbf{W}) \triangleq \frac{1}{m} \sum_{i=1}^{m} \ell(\mathbf{y}_i, f(\mathbf{x}_i, \mathbf{W})) \right\},$$

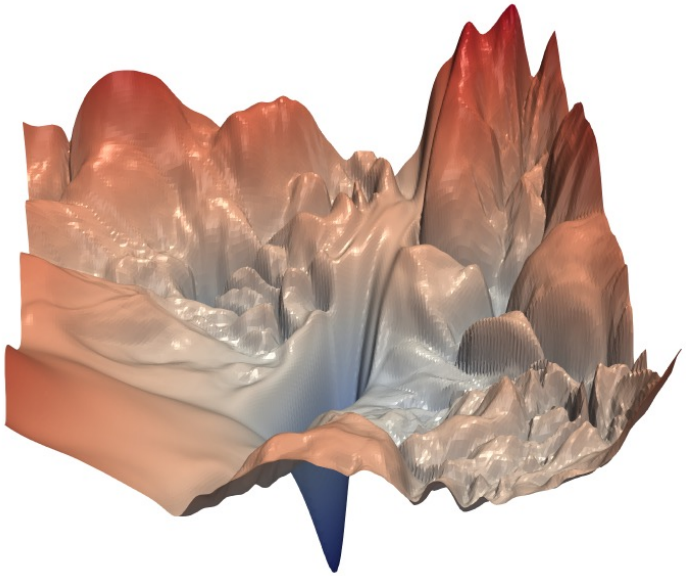
  - $\mathbf{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_L\}$, with $\mathbf{W}_i \in \mathbb{R}^{n_i \times n_{i-1}}$, are weights of NN model
  - $\{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^m\}$, $\mathbf{x}_i \in \mathbb{R}^{n_0}$, are training samples
  - $\ell(\cdot, \cdot)$ is a loss function (e.g., quadratic or logistic loss)
  - NN model can be written as:

$$f(\mathbf{x}_i, \mathbf{W}) = \mathbf{W}_L \sigma(\mathbf{W}_{L-1} \sigma(\ldots, \sigma(\mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{x}_i)) \ldots)),$$

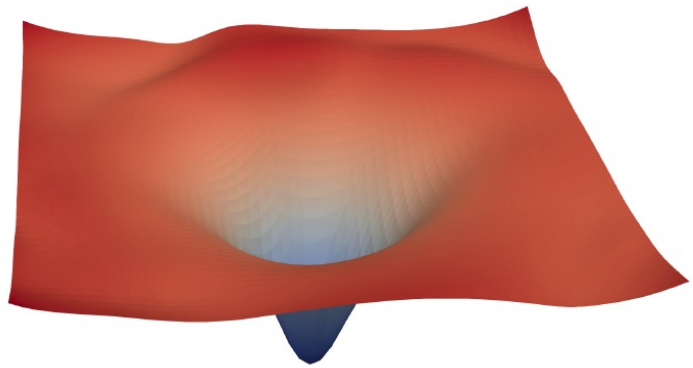
    where $\sigma(\cdot)$ is scalar-valued and called activation function.

Example 6: Deep Learning (Nonconvex)

- Landscape of deep neural networks
  - Loss surfaces of ResNet-56 with/without skip connections [Li et al. '18]



(a) without skip connections



(b) with skip connections

- Training NN is NP-complete in general [Blum and Rivest, '89], but:
  - All local minima are global for 1-layer NN: [Soltanolkotabi et al. '18], [Haeffele and Vidal, '17], [Feizi et al. '17]
  - GD/SGD converge to global min for linear networks [Arora et al. '18], [Ji and Telgarsky, '19], [Shin, '19], wide over-parameterized networks [Allen-Zhu et al., '19], and pyramid networks [Nguyen and Mondelli, '19]

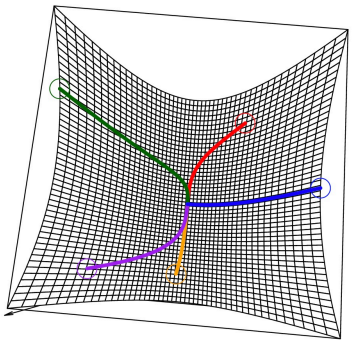
Part II

Convexity

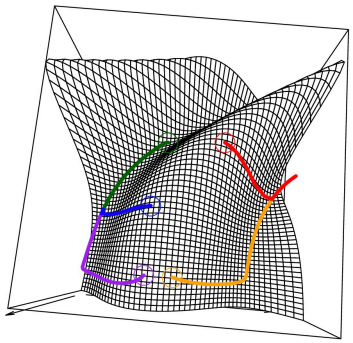
Why Do We Care About Convexity?

For convex optimization problem, local minima are global minima

Formally: Let $\mathcal{D}$ be the feasible domain defined by the constraints. If $\mathbf{x} \in \mathcal{D}$ satisfies the following local condition: $\exists\ d > 0$ such that for all $\mathbf{y} \in \mathcal{D}$ satisfying $\|\mathbf{x} - \mathbf{y}\|_2 \leq d$, we have $f_0(\mathbf{x}) \leq f_0(\mathbf{y})$. $\Rightarrow f_0(\mathbf{x}) \leq f_0(\mathbf{y})$ for all $\mathbf{y} \in \mathcal{D}$.



Convex



Nonconvex

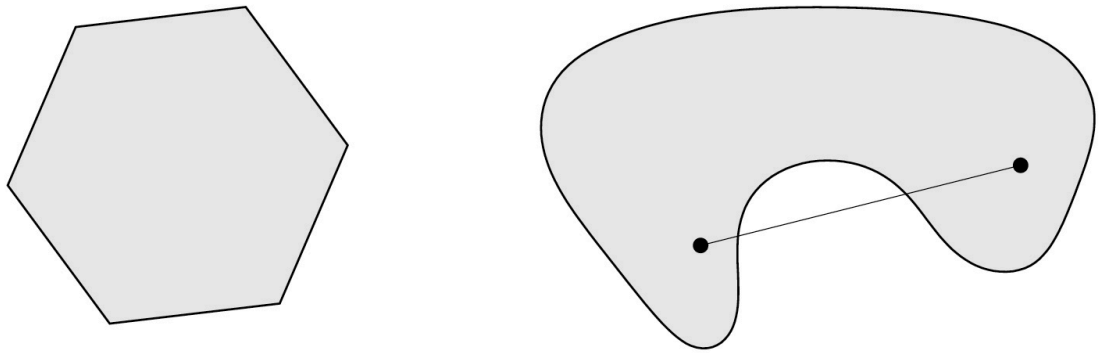
A crucial fact that would significantly reduce the complexity in optimization!

Convex Sets

Convex set: A set $\mathcal{D} \in \mathbb{R}^n$ such that

$$\forall \mathbf{x}, \mathbf{y} \in \mathcal{D} \quad \Rightarrow \quad \mu\mathbf{x} + (1-\mu)\mathbf{y} \in \mathcal{D}, \quad \forall 0 \leq \mu \leq 1$$

Geometrically, line segment joining any two points in $\mathcal{D}$ lies in entirely in $\mathcal{D}$

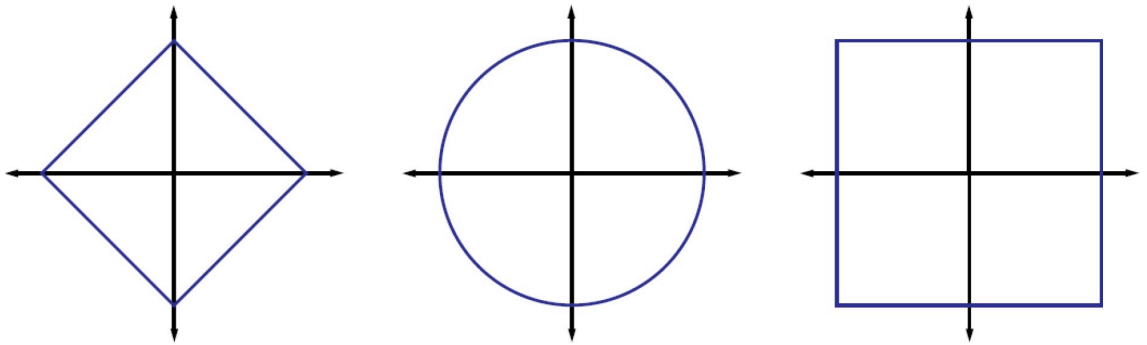


Convex combination: A linear combination $\mu_1\mathbf{x}_1 + \cdots + \mu_k\mathbf{x}_k$ for $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^n$, with $\mu_i \geq 0$, $i = 1, \ldots, k$ and $\sum_{i=1}^{k} \mu_i = 1$.

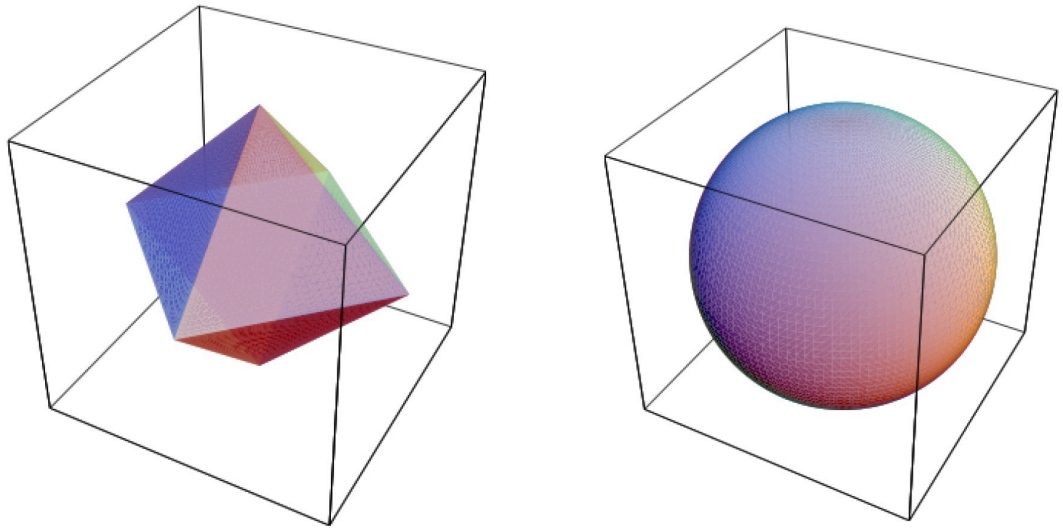
Convex hull: A set defined by all convex combinations of elements in a set $\mathcal{D}$.

Examples of Convex Sets

1) Norm balls: Radius r ball in l_p norm $\mathcal{B}_p = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_p \leq r\}$



$p = 1$ $p = 2$ $p = \infty$

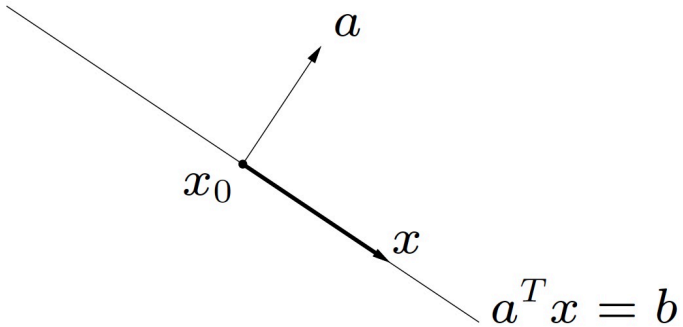


$p = 1$ $p = 2$

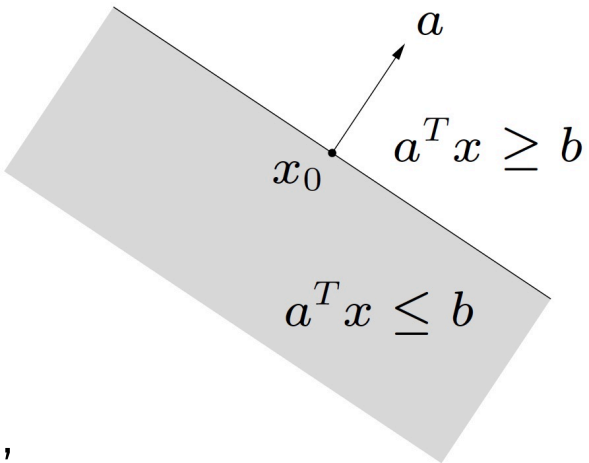
Examples of Convex Sets

2) Hyperplane and haflspaces

- Hyperplane: Set of the form $\{\mathbf{x}|\mathbf{a}^\top\mathbf{x} = b\}$ with $\mathbf{a} \neq \mathbf{0}$



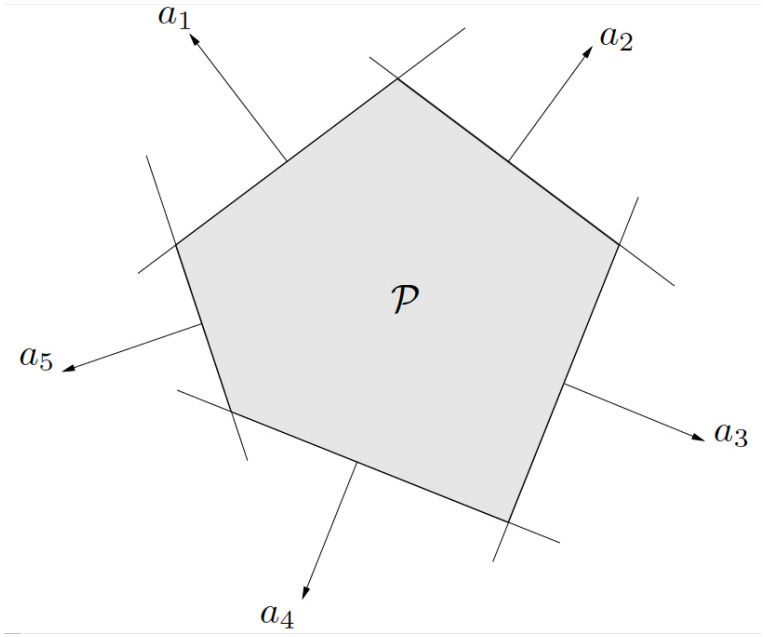
- Halfspace: Set of the form $\{\mathbf{x}|\mathbf{a}^\top\mathbf{x} \leq b\}$ with $\mathbf{a} \neq \mathbf{0}$



- $\mathbf{a}$ is called "normal vector"

Examples of Convex Sets

3) Polyhedron: $\{\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$, whre $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\leq$ is component-wise inequality



Note:

- $\{\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{C}\mathbf{x} = \mathbf{d}\}$ is also a polyhedron (Why?)
- Polyhedron is an intersection of finite number of halfspaces and hyperplanes

Operations That Preserve Convexity of Sets

- Intersection: The intersection of convex sets is convex
- Scaling and Translation: If $\mathcal{C}$ is convex, then $a\mathcal{C} + \mathbf{b} \triangleq \{a\mathbf{x} + \mathbf{b} : \mathbf{x} \in \mathcal{C}\}$ is also convex for any a and $\mathbf{b}$.
- Affine image and preimage: If $f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$ and $\mathcal{C}$ is convex, then

$$f(\mathcal{C}) \triangleq \{f(\mathbf{x}) : \mathbf{x} \in \mathcal{C}\}$$

is also convex. If $\mathcal{D}$ is convex, then

$$f^{-1}(\mathcal{D}) \triangleq \{\mathbf{x} : f(\mathbf{x}) \in \mathcal{D}\}$$

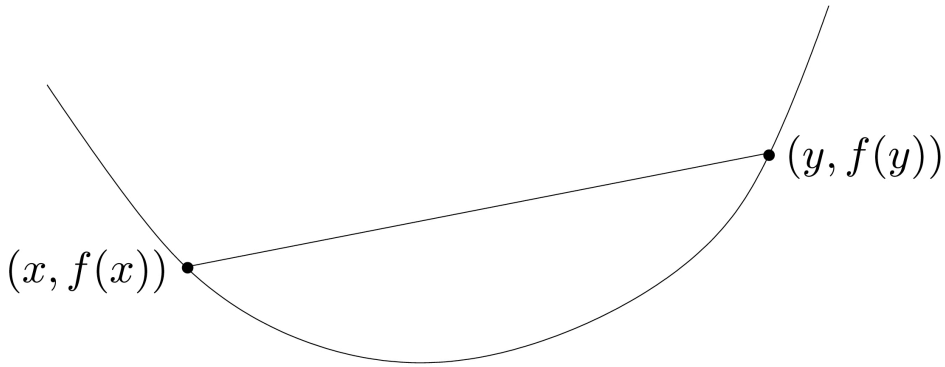
is also convex

Convex Functions

- Convex function: $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is convex if $\mathrm{dom}(f) \in \mathbb{R}^n$ is convex and

$$f\big(\mu\mathbf{x} + (1-\mu)\mathbf{y}\big) \leq \mu f(\mathbf{x}) + (1-\mu) f(\mathbf{y})$$

for all $\mu \in [0, 1]$ and for all $\mathbf{x}, \mathbf{y} \in \mathrm{dom}(f)$.



In words, f lies below the line segment that joins any $f(\mathbf{x})$ and $f(\mathbf{y})$.

- Concave function: f concave $\iff -f$ convex

Other Important Characterizations of Convex Functions

- First-order characterization: If f is differentiable, then f is convex if and only if $\mathrm{dom}(f)$ is convex, and

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f^{\top}(\mathbf{x})(\mathbf{y} - \mathbf{x})$$

  for all $\mathbf{x}, \mathbf{y} \in \mathrm{dom}(f)$.

- Implying an important consequence: $\nabla f(\mathbf{x}) = 0 \Longrightarrow \mathbf{x}$ minimizes f

- Second-order characterization: If f is twice differentiable, then f is convex if and only if $\mathrm{dom}(f)$ is convex, and $\mathbf{H}(\mathbf{x}) = \nabla^2 f(\mathbf{x}) \succeq 0$ for all $\mathbf{x} \in \mathrm{dom}(f)$

Important Convexity Notions

- Strictly convex: $f\big(\mu\mathbf{x}+(1-\mu)\mathbf{y}\big) < \mu f(\mathbf{x})+(1-\mu)f(\mathbf{y})$, i.e., f is convex and has greater curvature than a linear function
- Strongly convex with parameter m: $f(\mathbf{x}) - \frac{m}{2}\|\mathbf{x}\|^2$ is convex, i.e., f is at least as curvy as a m-parameterized quadratic function
- Note: strongly convex $\Rightarrow$ strictly convex $\Rightarrow$ convex, (converse is not true)
- Similar notions for concave functions

Important Examples of Convex/Concave Functions

- Univariate functions:
  - Exponential functions: e^{ax} is convex for all $a \in \mathbb{R}$
  - Power functions: x^a is convex if $a \in (-\infty, 0] \cup [1, \infty)$ and concave if $a \in [0, 1]$
  - Logarithmic functions: $\log(x)$ is concave for $x > 0$
- Affine function: $\mathbf{a}^\top \mathbf{x} + \mathbf{b}$ is both concave and convex
- Quadratic function: $\frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ is convex if $\mathbf{Q} \succeq 0$ (positive semidefinite)
- Least square loss function: $\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$ is always convex (since $\mathbf{A}^\top \mathbf{A} \succeq 0$)
- Norm: $\|\mathbf{x}\|$ is always convex for any norm, e.g.,
  - l_p norm: $\|\mathbf{x}\|_p = (\sum_{i=1}^n x_i^p)^{\frac{1}{p}}$ for $p \geq 1$, $\|\mathbf{x}\|_\infty = \max_{i=1,\ldots,n}\{|x_i|\}$
  - Matrix operator (spectral) norm $\|\mathbf{X}\|_{\text{op}} = \sigma_1(\mathbf{X})$
    Matrix trace (nuclear) norm $\|\mathbf{X}\|_{\text{tr}} = \sum_{i=1}^r \sigma_r(\mathbf{X})$, where
    $\sigma_1(\mathbf{X}) \geq \cdots \geq \sigma_r(\mathbf{X}) \geq 0$ are the singular values of $\mathbf{X}$

Operations That Preserve Convexity of Functions

- Nonnegative linear combinations: $f_1, \ldots, f_m$ being convex implies $\mu_1 f_1 + \cdots + \mu_m f_m$ is convex for any $\mu_1, \ldots, \mu_m \geq 0$

- Pointwise maximization: If f_i is convex for any index $i \in \mathcal{I}$, then

$$f(\mathbf{x}) = \max_{i \in \mathcal{I}} f_i(\mathbf{x})$$

  is convex. Note that the index set $\mathcal{I}$ can be infinite

- Partial minimization: If $g(\mathbf{x}, \mathbf{y})$ is convex in $\mathbf{x}, \mathbf{y}$ and $\mathcal{C}$ is convex, then

$$f(\mathbf{x}) = \min_{\mathbf{y} \in \mathcal{C}} g(\mathbf{x}, \mathbf{y})$$

  is convex (the basis for ADMM, coordinate descent, ...)

More Operations That Preserve Convexity of Functions

- Affine composition: f is convex $\Longrightarrow g(\mathbf{x}) = f(\mathbf{A}\mathbf{x} + \mathbf{b})$ is convex

- General composition: Suppose $f = h \circ g$, where $g : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R} \to \mathbb{R}$, $f : \mathbb{R}^n \to \mathbb{R}$. Then:
  - f is convex if h is convex & nondecreasing, g is convex
  - f is convex if h is convex & nonincreasing, g is concave
  - f is concave if h is concave & nondecreasing, g is concave
  - f is concave if h is concave & nonincreasing, g is convex

  How to remember these? Think of the chain rule when $n = 1$

$$f''(x) = h''(g(x))g'(x)^2 + h'(g(x))g''(x)$$

Part III

First-Order Methods

Outline for First-Order Methods

- Convergence Rate Concept
- The Gradient Descent Method
- The Stochastic Gradient Descent Method
- Variance-Reduced Stochastic First-Order Methods
- Adaptive First-Order Methods

Iterative Algorithms for Optimization

We consider the following iterative algorithms:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + s_k \mathbf{d}_k,$$

where s_k is step-size, and $\mathbf{d}_k$ is search direction depending on $(\mathbf{x}_k, \mathbf{x}_{k-1}, \ldots)$.

For now: assume f smooth, $f(\mathbf{x}_k)$ and $\nabla f(\mathbf{x}_k)$ is easy to evaluate

Complications from ML:

- Nonconvex f
- Nonsmooth f
- f not available (or too expensive to evaluate exactly)
- Only an estimate of $\nabla f(\mathbf{x}_k)$ is available
- A constraint $\mathbf{x} \in \Omega$ (usually a relatively simple Ω, e.g., ball, box, simplex...)
- Nonsmooth regularization, i.e., instead of $f(\mathbf{x})$, we want $\min f(\mathbf{x}) + \tau\psi(\mathbf{x})$

How to Evaluate the Speed of an Iterative Algorithm?

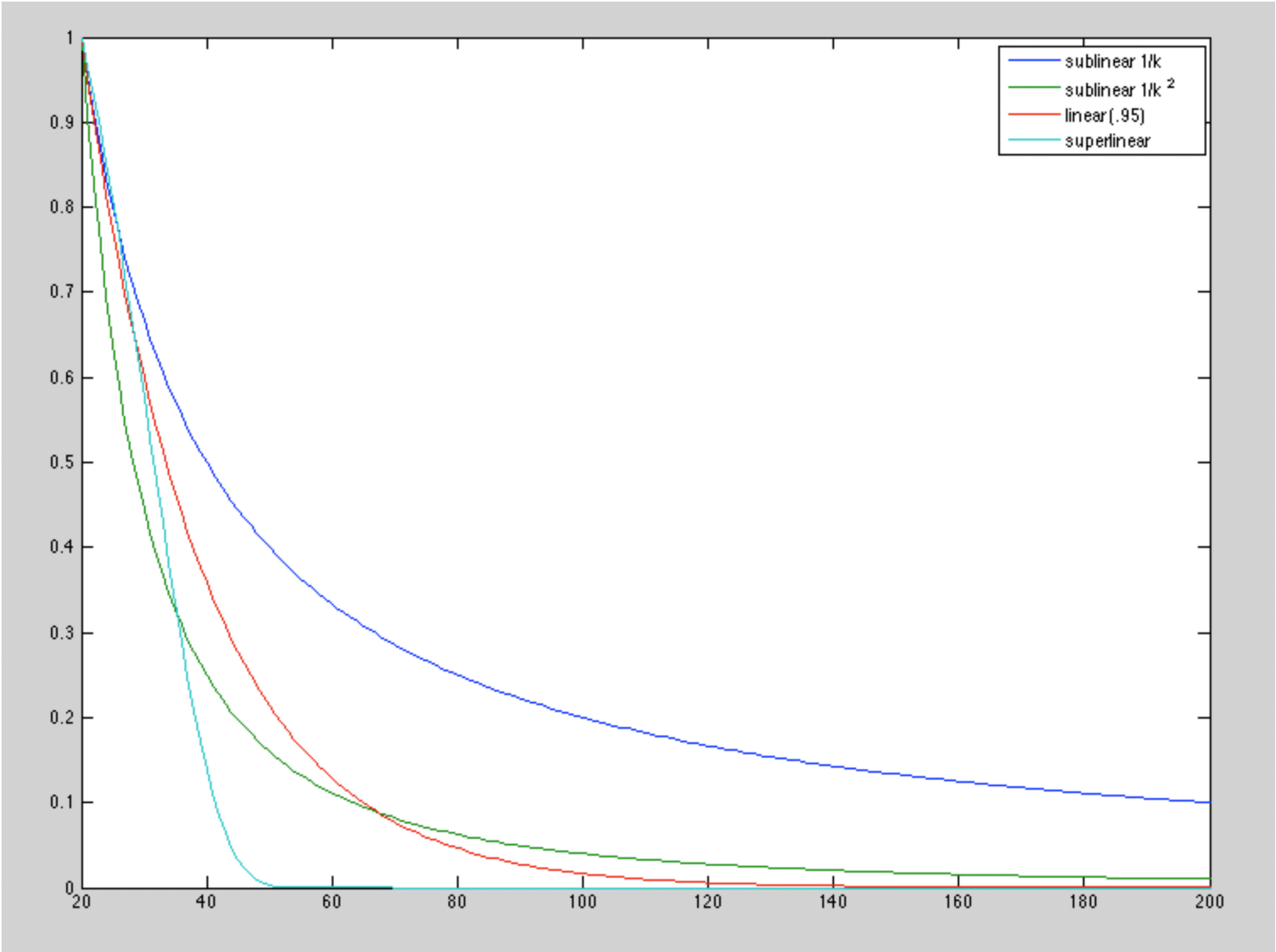
Definition 1 (Convergence rate)

A sequence $\{r_k\} \to r^*$ and $r_k \neq r^*$ for all k. The rate (or order) of convergence p is a nonnegative number satisfying

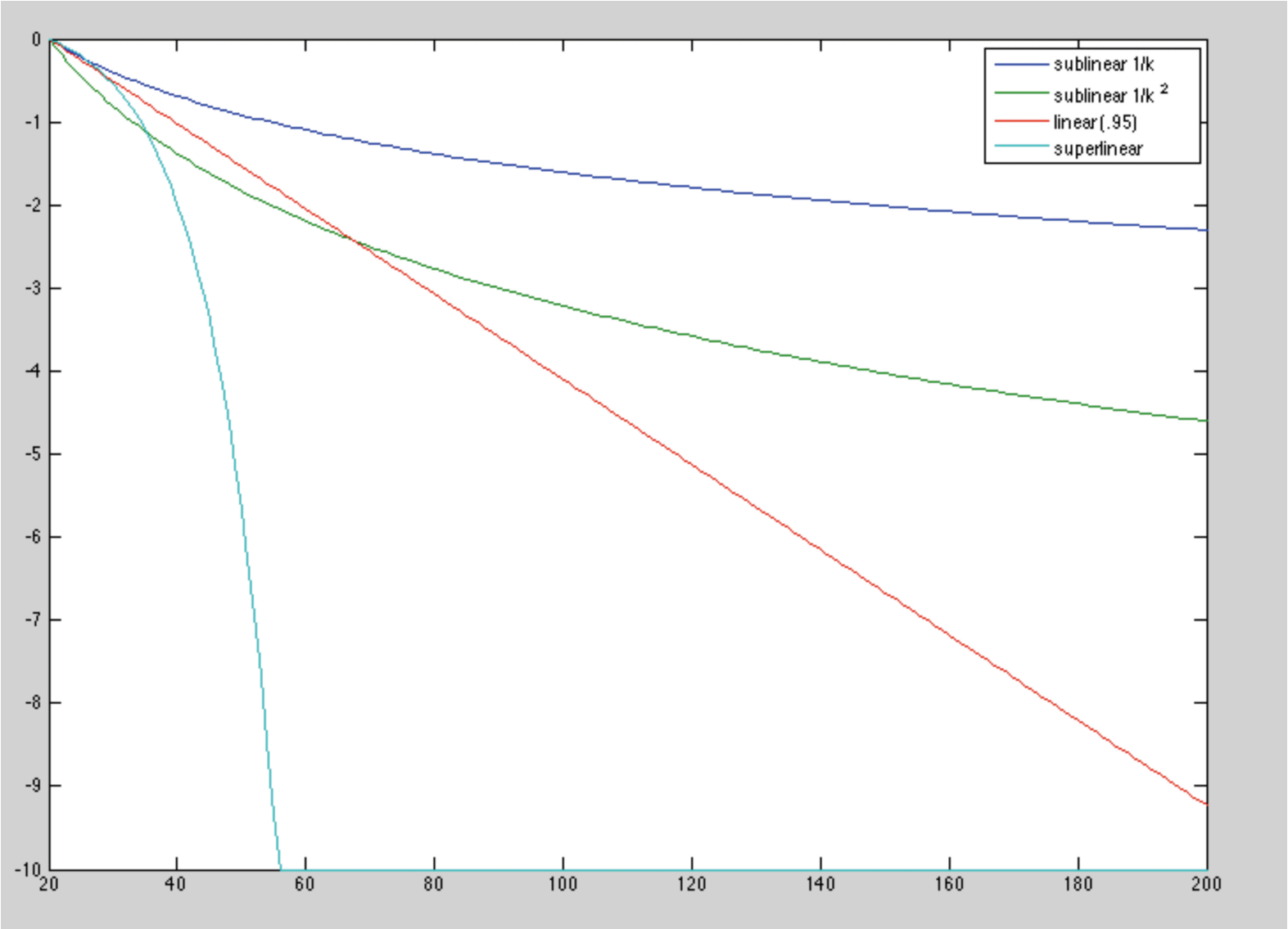
$$\limsup_{k\to\infty} \frac{\|r_{k+1} - r^*\|}{\|r_k - r^*\|^p} = \beta < \infty.$$

- Sublinear: $p = 1$ and $\beta = 1$ (e.g., $O(1/k)$ rate, kind of slow but still OK)
- Linear or geometric: $p = 1$ and $0 < \beta < 1$ (i.e., $\|r_{k+1} - r^*\| \leq \beta\|r_k - r^*\|$ for some $\beta \in (0, 1)$, or $\|r_k - r^*\| = O(\beta^k)$, which is quite fast)
- Superlinear: $p > 1$ and $\beta < \infty$, or $p = 1$ and $\beta = 0$ (i.e., $\frac{\|r_{k+1}-r^*\|}{\|r_k-r^*\|} \to 0$, that's very fast!)
- Quadratic: $p = 2$ and $\beta < \infty$ ($\|r_{k+1} - r^*\| \leq \beta\|r_k - r^*\|^2$, # of correct significant digits doubles per iteration. Rarely need anything faster than this!)

Convergence Rates Comparisons



Convergence Rates Comparisons: Log-Scale



Outline for First-Order Methods

- Convergence Rate Concept
- The Gradient Descent Method
- The Stochastic Gradient Descent Method
- Variance-Reduced Stochastic First-Order Methods
- Adaptive First-Order Methods

Gradient Descent

Back to the unconstrained optimization problem, with f smooth and convex:

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$$

Denote the optimal value as $f^* = \min_{\mathbf{x}} f(\mathbf{x}^*)$ and an optimal solution as $\mathbf{x}^*$

Gradient Descent

Choose initial point $\mathbf{x}_0 \in \mathbb{R}^n$. Repeat:

$$\mathbf{x}_k = \mathbf{x}_{k-1} - s_k \nabla f(\mathbf{x}_{k-1}), \quad k = 1, 2, 3, \ldots$$

Stop if some stopping criterion is satisfied.

Gradient Descent: Geometric Interpretation

Gradient descent is a first-order method: Consider the following quadratic Taylor approximation:

$$f(\mathbf{y}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{1}{2}(\mathbf{y} - \mathbf{x})^\top \nabla^2 f(\mathbf{x})(\mathbf{y} - \mathbf{x})$$

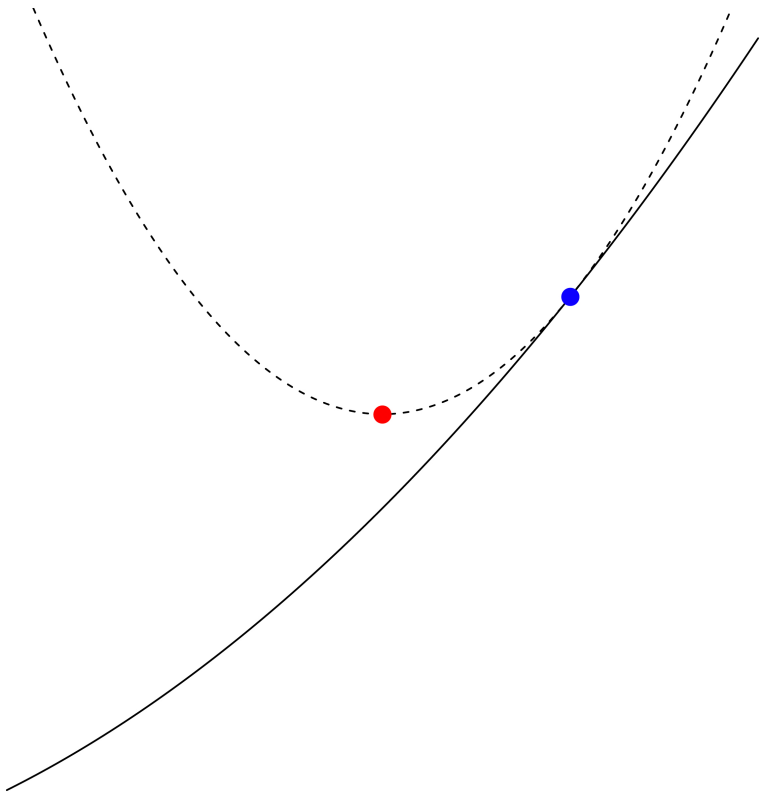
No, we replace Hessian $\nabla^2 f(\mathbf{x})$ by $\frac{1}{s}\mathbf{I}$ to obtain:

$$f(\mathbf{y}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{1}{2s}\|\mathbf{y} - \mathbf{x}\|^2$$

Can be viewed as a linear approximation to f, with proximity term to $\mathbf{x}$ weighted by $\frac{1}{2s}$. Choose next point $\mathbf{y} = \mathbf{x}^+$ to minimize this approximation:

$$\mathbf{x}^+ = \mathbf{x} - s\nabla f(\mathbf{x})$$

Gradient Descent: Geometric Interpretation



$$\mathbf{x}^+ = \arg\min_y f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{1}{2s}\|\mathbf{y} - x\|_2^2$$

Questions:

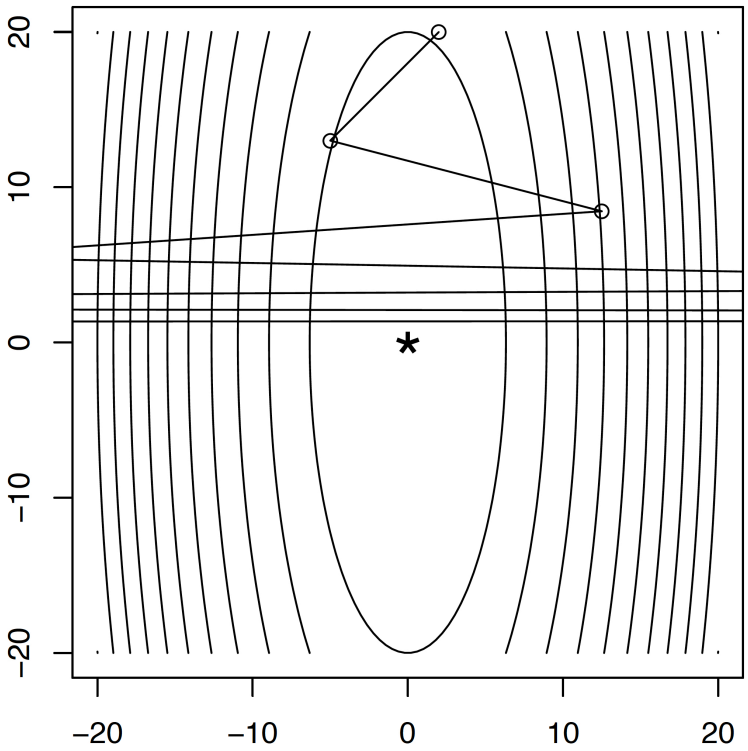
- How to choose step sizes $\{s_k\}$?
- What is the according convergence rate? Or does it depend on $\{s_k\}$?

Strategy 1: Fixed Step Size

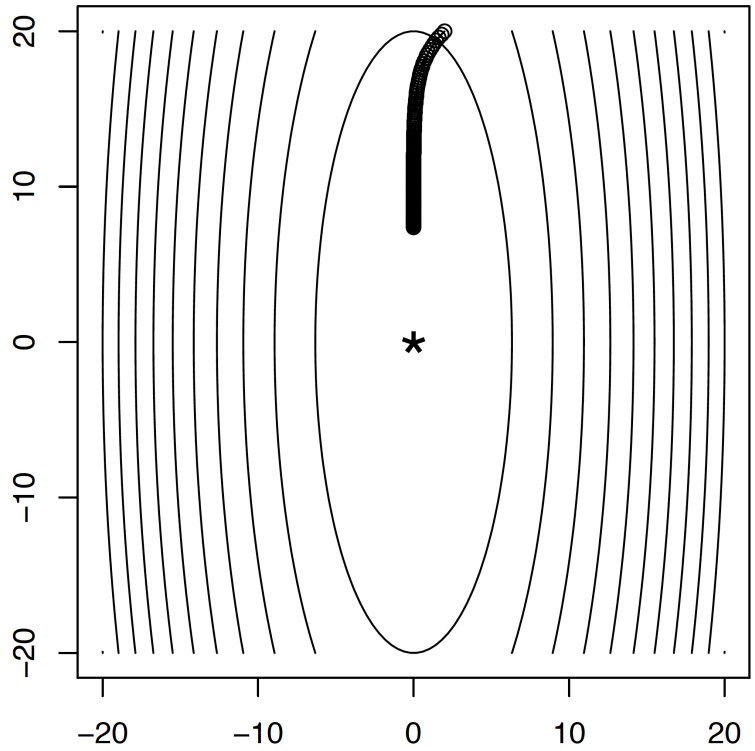
Simply set $s_k = s$ for all $k = 1, 2, 3, \ldots$.

Limitations: May diverge if s is too large, Can be slow if s is too small.

Example: Consider $f(\mathbf{x}) = (10x_1^2 + x_2^2)/2$:



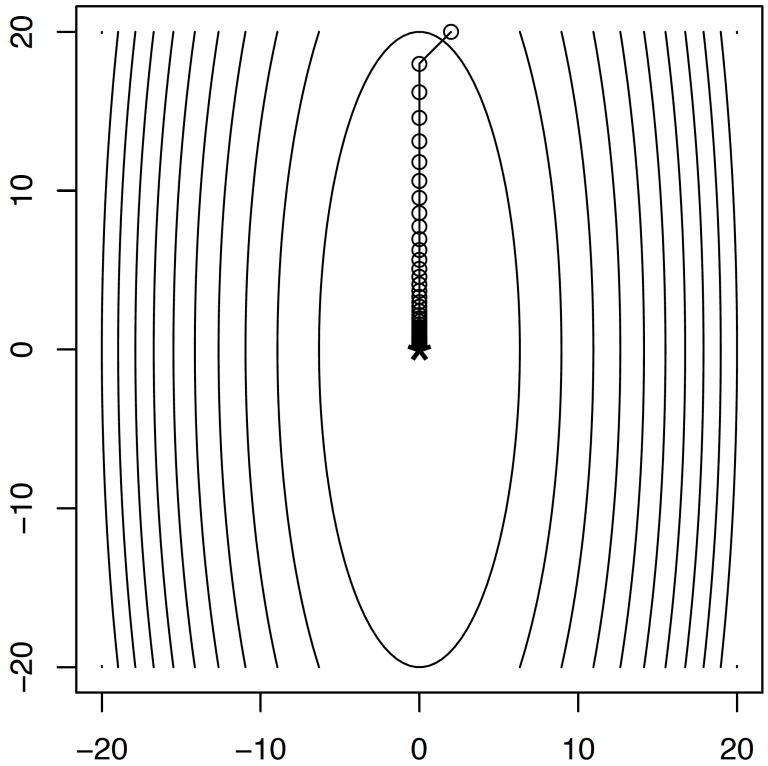
8 iterations



100 iterations

Strategy 1: Fixed Step Size

Converges nicely when s is "just right." Same example, GD after 40 iterations:



Will be clear what we mean by "just right" in convergence rate analysis later

Convergence Rate Analysis (Convex): Fixed Step Size

Assume that f is convex & differentiable, with $\mathrm{dom}(f) = \mathbb{R}^n$ and additionally

$$\|\nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\|_2 \leq L\|\mathbf{y} - \mathbf{x}\|_2, \quad \forall \mathbf{x}, \mathbf{y}$$

That is, ∇f is Lipschitz continuous with constant $L > 0$ (L-Lipschitz continuous)

Theorem 1 (Optimality Gap)

If f is convex, differentiable, and L-smooth, gradient descent with fixed step size $s \leq 1/L$ satisfies

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \frac{\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2}{2sk},$$

i.e., gradient descent method has sublinear convergence rate $O(1/k)$.

Remark:

- To get $f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \epsilon$, it takes $O(1/\epsilon)$ iterations.

Convergence Rate Analysis (Nonconvex): Fixed Step Size

Assume that f is nonconvex & differentiable, and L-smooth

Theorem 2 (Stationarity Gap)

If f is nonconvex, differentiable, and L-smooth, then gradient descent with fixed step size $s \leq 1/L$ satisfies

$$\min_{t=0,\ldots,k-1} \|\nabla f(\mathbf{x}_t)\|_2^2 \leq \frac{2(f(\mathbf{x}_0) - f^*)}{sk}$$

i.e., gradient descent method has sublinear convergence rate $O(1/k)$.

Remark:

- To get $\|\nabla f(\mathbf{x}_k)\|_2 \leq \epsilon$ for some k, it takes $O(\epsilon^{-2})$ iterations.

Strategy 2: Exact Line Search

Choose the step size s to do the "best" we can along the direction of $-\nabla f(\mathbf{x})$:

$$s = \arg\min_{t \geq 0} f(\mathbf{x} - t\nabla f(\mathbf{x}))$$

Limitations:

- Usually it's too expensive to do this in each iteration.

Strategy 3: Inexact Line Search – Backtracking

One way to adaptively choose step size is to use backtracking line search

1. First fix parameters $0 < \beta < 1$ and $0 < \alpha \leq \frac{1}{2}$
2. At each iteration, start with $s = 1$, and while

$$f(\mathbf{x} - s\nabla f(\mathbf{x})) > f(\mathbf{x}) - \alpha s \|\nabla f(\mathbf{x})\|_2^2$$

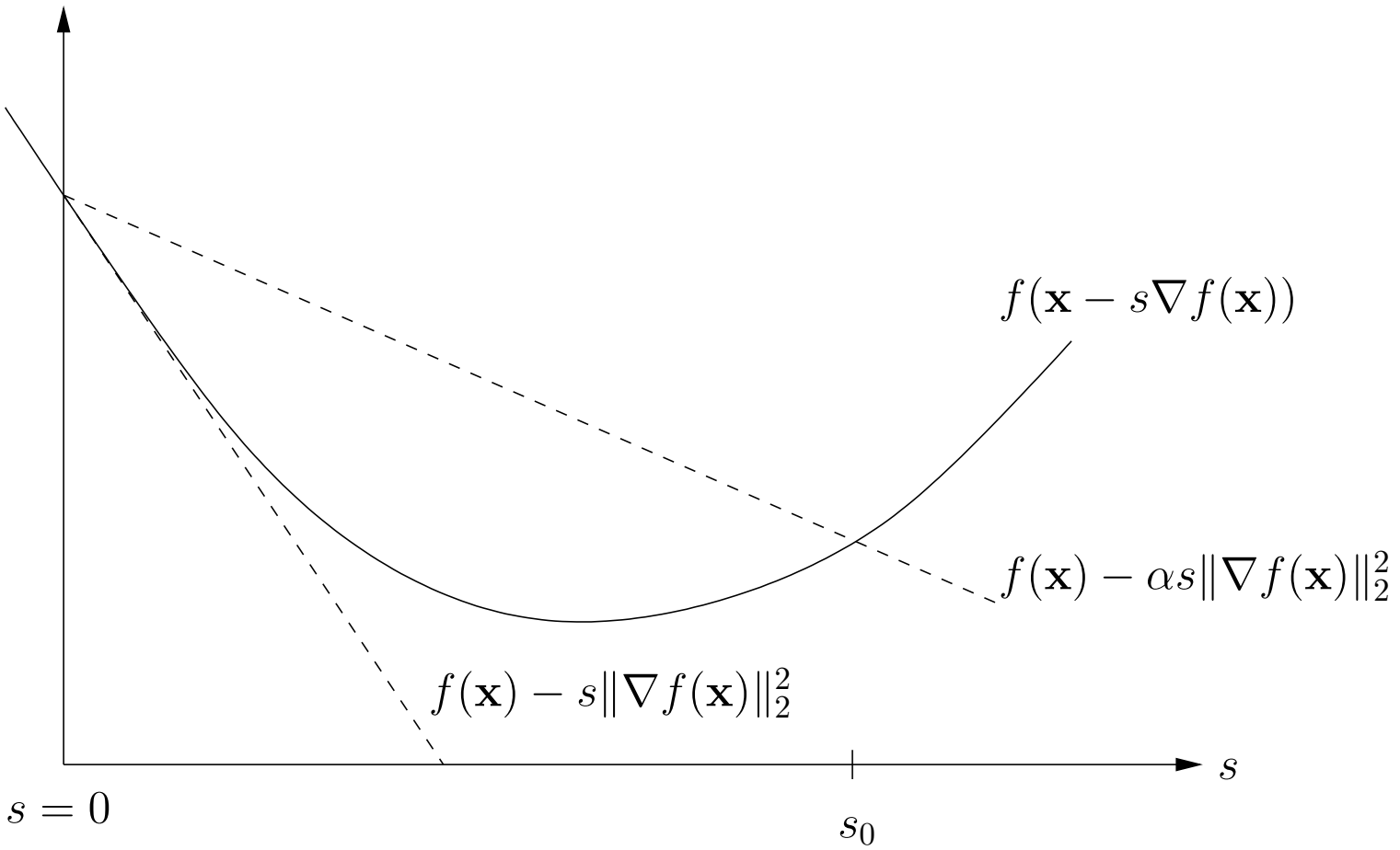
shink $s = \beta s$. Else, perform gradient descent update:

$$\mathbf{x}^+ = \mathbf{x} - s\nabla f(\mathbf{x})$$

Remarks:

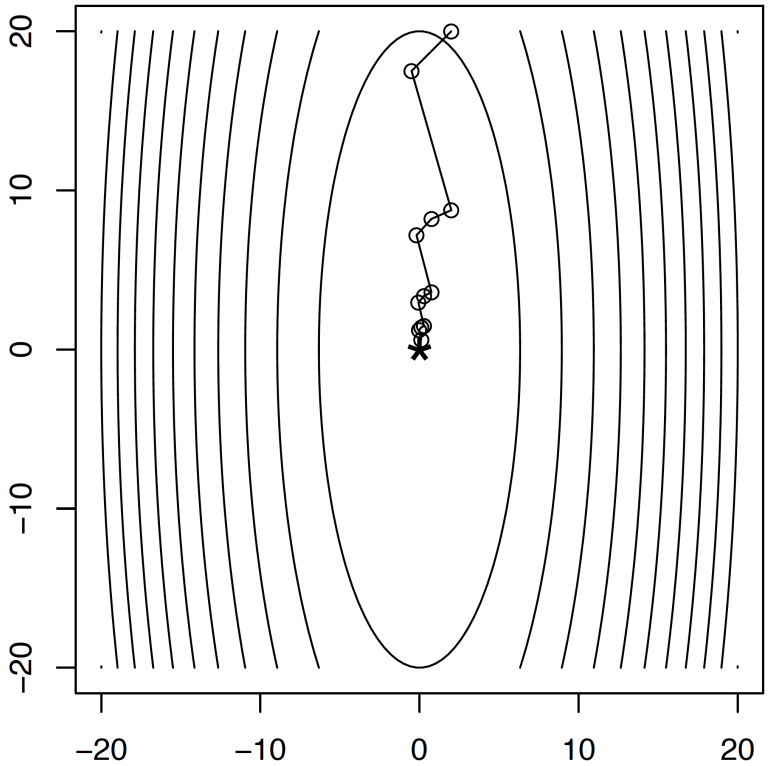
- Simple and tends to work well in practice (further simplification: just take $\alpha = \beta = 1/2$). But doesn't work for f nonsmooth
- Also referred to as Armijo's rule. Step size shrinking very aggressively

Backtracking Interpretation



Backtracking Example

Backtracking picks up roughly the right step size (12 outer iterations, 40 iterations in total):



Outline for First-Order Methods

- Convergence Rate Concept
- The Gradient Descent Method
- The Stochastic Gradient Descent Method
- Variance-Reduced Stochastic First-Order Methods
- Adaptive First-Order Methods

Unbiased Stochastic Gradient

- Random vector $\tilde{\mathbf{g}} \in \mathbb{R}^n$ is a unbiased stochastic gradient if it can be written as $\tilde{\mathbf{g}} = \mathbf{g} + \mathbf{n}$, where $\mathbf{g}$ is the true gradient and $\mathbb{E}[\mathbf{n}] = \mathbf{0}$
- $\mathbf{n}$ can be interpreted as error in computing $\mathbf{g}$, measurement noise, Monte Carlo sampling errors, etc.
- If $f(\cdot)$ is non-smooth, $\tilde{\mathbf{g}}$ is a noisy unbiased subgradient at $\mathbf{x}$ if

$$f(\mathbf{z}) \geq f(\mathbf{x}) + (\mathbb{E}[\tilde{\mathbf{g}}|\mathbf{x}])^\top (\mathbf{z} - \mathbf{x}), \quad \forall \mathbf{z}$$

  holds almost surely.

Stochastic Gradient Descent Method

- Consider $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$. Following standard GD, we should do:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k \mathbb{E}[\tilde{\mathbf{g}}_k | \mathbf{x}_k]$$

- However, $\mathbb{E}[\tilde{\mathbf{g}}_k | \mathbf{x}_k]$ is difficult to compute: Unknown distribution, too costly to sample at each iteration k, etc.

- Idea: Simply use a noisy unbiased subgradient to replace $\mathbb{E}[\tilde{\mathbf{g}}_k | \mathbf{x}_k]$

- The stochastic subgradient method works as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k \tilde{\mathbf{g}}_k$$

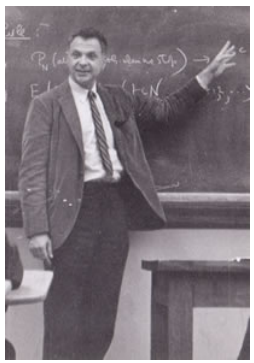
  - $\mathbf{x}_k$ is the k-th iterate
  - $\tilde{\mathbf{g}}_k$ is any noisy gradient of at $\mathbf{x}_k$, i.e., $\mathbb{E}[\tilde{\mathbf{g}}_k | \mathbf{x}_k] = \nabla f(\mathbf{x}_k)$
  - s_k is the step size
  - Let $f_{\text{best}}^{(k)} \triangleq \min_{i=1,\ldots,k} \{f(\mathbf{x}_i)\}$ and $\|\nabla f_{\text{best}}^{(k)}\| \triangleq \min_{i=1,\ldots,k} \{\|\nabla f(\mathbf{x}_i)\|\}$

Historical Perspective

- Also referred to as stochastic approximation in the literature, first introduced by [Robbins, Monro '51] and [Keifer, Wolfowitz '52]
- The original work [Robbins, Monro '51] is motivated by finding a root of a continuous function:

$$f(\mathbf{x}) = \mathbb{E}[F(\mathbf{x}, \theta)] = 0,$$

where $F(\cdot, \cdot)$ is unknown and depends on a random variable θ. But the experimenter can take random samples (noisy measurements) of $F(\mathbf{x}, \theta)$



Herbert Robbins

Sutton Monro

Historical Perspective

- Robbins-Monro: $\mathbf{x}_{k+1} = \mathbf{x}_k + s_k Y(\mathbf{x}_k, \theta)$, where:
  - $\mathbb{E}[Y(\mathbf{x}, \theta)|\mathbf{x} = \mathbf{x}_k] = f(\mathbf{x}_k)$ is an unbiased estimator of $f(\mathbf{x}_k)$
  - Robbins-Monro originally showed convergence in L^2 and in probability
  - Blum later prove convergence is actually w.p.1. (almost surely)
  - Key idea: Diminishing step-size provides implicit averaging of the observations
- Robbins-Monro's scheme can also be used in stochastic optimization of the form $f(\mathbf{x}^*) = \min_{\mathbf{x}} \mathbb{E}[F(\mathbf{x}, \theta)]$ (equivalent to solving $\nabla f(\mathbf{x}^*) = 0$)
- Stochastic approximation, or more generally, stochastic gradient has found applications in many areas
  - Adaptive signal processing
  - Dynamic network control and optimization
  - Statistical machine learning
  - Workhorse algorithm for training deep neural networks

Assumptions and Step Size Rules

- $f^* = \inf_x f(\mathbf{x}_k) > -\infty$, with $f(\mathbf{x}^*) = f^*$
- $\mathbb{E}[\|\tilde{\mathbf{g}}_k\|_2^2] \leq G^2$, for all k
- $\mathbb{E}[\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2] \leq R^2$

Commonly used step-size strategies:

- Constant step-size: $s_k = s$, $\forall k$
- Step-size is square summable, but not summable

$$s_k > 0, \ \forall k, \qquad \sum_{k=1}^{\infty} s_k^2 < \infty, \qquad \sum_{k=1}^{\infty} s_k = \infty$$

Note: This is stronger than needed, but just to simplify proof

Convergence of SGD (Convex)

- Convergence in expectation:

$$\lim_{k\to\infty} \mathbb{E}[f_{\text{best}}^{(k)}] = f^*$$

- Convergence in probability: for any $\epsilon > 0$,

$$\lim_{k\to\infty} \Pr\{|f_{\text{best}}^{(k)} - f^*| > \epsilon\} = 0$$

- Almost sure convergence

$$\Pr\Big\{ \lim_{k\to\infty} f_{\text{best}}^{(k)} = f^* \Big\} = 1$$

- See [Kushner, Yin '97] for a complete treatment on convergence analysis

Convergence Rate (Nonconvex) – Finite Sum

- Consider the following finite-sum minimization

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) = \min_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x})$$

  where N is typically large, e.g., empirical risk minimization (ERM) in ML

- Consider using SGD to solve this problem under the following assumptions:
  - $f(\cdot)$ is nonconvex and bounded from below
  - ∇f is differentiable with L-Lipschitz continuous gradients (L-smooth)
  - $\mathbb{E}[\|\nabla f_i(\mathbf{x})\|^2] \leq \sigma^2$ for some σ^2 and all $\mathbf{x}$ (bounded gradient, can be relaxed)

Convergence Rate (Nonconvex) – Finite Sum

Theorem 3 (Stationarity Gap)

If the finite-sum problem $f(\cdot)$ is nonconvex, differentiable, and L-smooth, then the SGD method with step-sizes $\{s_k\}$ satisfies

$$\min_{k=0,1,\ldots,t-1} \{\|\nabla f(\mathbf{x}_k)\|_2^2\} \leq \frac{f(\mathbf{x}_0) - f^*}{\sum_{k=0}^{t-1} s_k} + \frac{L\sigma^2}{2} \frac{\sum_{k=0}^{t-1} s_k^2}{\sum_{k=0}^{t-1} s_k}.$$

Remark:

- If $\sigma^2 = 0$ (GD), then a constant step-size recovers an $O(1/t)$ rate.
- Classical diminishing step-sizes $s_k = \alpha/k$ for some $\alpha > 0$: $\sum_k s_k = O(\log(t))$ and $\sum_k s_k^2 = O(1)$. So convergence rate is $O(1/\log(t))$
- Diminishing step-sizes $s_k = \alpha/\sqrt{k}$ for some $\alpha > 0$: $\sum_k s_k = O(\sqrt{t})$ and $\sum_k s_k^2 = O(\log(t))$. So convergence rate is $O(\log(t)/\sqrt{t}) = \tilde{O}(1/\sqrt{t})$
- Constant step-sizes $s_k = \alpha$ for some $\alpha > 0$: $\sum_k s_k = k\alpha$ and $\sum_k s_k^2 = k\alpha^2$. So convergence rate is $O(1/t) + O(\alpha)$

Convergence Rate (Nonconvex) - Finite Sum+Time Oracle

Theorem 4 ([Ghadimi & Lan '13])

Suppose $f(\cdot)$ is L-smooth and has σ-bounded gradients and it is known a priori that the SGD algorithm will be executed for T iterations. Let $s_k = c/\sqrt{T}$, where

$$c = \sqrt{\frac{2(f(\mathbf{x}_0) - f^*)}{L\sigma^2}}.$$

Then, the iterates of SGD satisfy

$$\min_{0 \le t \le T-1} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2] \le \sqrt{\frac{2(f(\mathbf{x}_0) - f^*)L}{T}}\sigma.$$

Convergence Rate (Nonconvex) - General Expectation Minimization with Batching

- Consider the following general expectation minimization problem

$$f(\mathbf{x}) = \mathbb{E}_{\xi}[f(\mathbf{x}, \xi)],$$

  where ξ is a random vaiable with distribution $\mathcal{D}$.

- Consider using SGD to solve this problem under the following assumptions:
  - $f(\cdot)$ is nonconvex and bounded from below
  - ∇f is differentiable with L-Lipschitz continuous gradients (L-smooth)
  - $\mathbb{E}_{\xi}[f(\mathbf{x}, \xi)] = \nabla f(\mathbf{x})$ and $\mathbb{E}_{\xi}[\|f(\mathbf{x}, \xi) - \nabla f(\mathbf{x})\|_2^2] \leq \sigma^2$

- A common approach in SGD: Rather than choosing one training sample randomly at a time, use a larger random mini-batch of samples $\mathcal{B}_k$, with $|\mathcal{B}_k| = B_k$. Then, $\mathbf{g}_k = \frac{1}{B_k} \sum_{i=1}^{B_k} \nabla f(\mathbf{x}, \xi_i)$. SGD becomes:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k \mathbf{g}_k = \mathbf{x}_k - \frac{s_k}{B_k} \sum_{i=1}^{B_k} \nabla f(\mathbf{x}, \xi_i),$$

  where $\xi_1, \ldots, \xi_{B_k}$ are i.i.d. sampled from $\mathcal{D}$

Convergence Rate (Nonconvex) - General Expectation Minimization with Batching

Theorem 5 (Stationarity Gap)

In the expectation minimization problem, supposed that $f(\cdot)$ is nonconvex, differentiable, and L-smooth. For any given $\epsilon > 0$, then the SGD method with mini-batch size $B_k = B = \max\{1, \frac{2\sigma^2}{\epsilon^2}\}$, $\forall k$, and step-sizes $s_k \leq \frac{1}{2L}$, $\forall k$, satisfies

$$\mathbb{E}[\|\nabla f(\hat{\mathbf{x}}_t)\|_2^2] \leq \frac{4L(f(\mathbf{x}_0) - f^*)}{t} + \frac{\epsilon^2}{2}, \tag{1}$$

where $\hat{\mathbf{x}}_t$ is chosen uniformly at random from $\mathbf{x}_0, \ldots, \mathbf{x}_{t-1}$. Thus, Eq. (1) implies that taking $t = \lceil \frac{8L(f(\mathbf{x}_0)-f^*)}{\epsilon^2} \rceil$ yields $\mathbb{E}[\|\nabla f(\hat{\mathbf{x}}_t)\|_2^2] \leq \epsilon^2$.

Sample Complexity Bound:

$$\sum_{k=0}^{t-1} B_k = \frac{2\sigma^2}{\epsilon^2} t = \left\lceil \frac{16L(f(\mathbf{x}_0) - f^*)\sigma^2}{\epsilon^4} \right\rceil = O(\epsilon^{-4})$$

- Optimal up to constant factors (see [Arjevani et al. 2019] for lower bound)

Mini-Batching SGD as Gradient Descent with Error

- SGD with mini-batcch:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{s_k}{B_k} \sum_{i=1}^{B_k} \nabla f(\mathbf{x}, \xi_i)$$

- This can be viewed as a "gradient descent with error"

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k(\nabla f(\mathbf{x}_k) + \mathbf{e}_k),$$

  where $\mathbf{e}_k$ is the difference between approximation and true gradient

- By setting $s_k = 1/L$, it can be shown that

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) - \underbrace{\frac{1}{2L}\|\nabla f(\mathbf{x}_k)\|^2}_{\text{good}} + \underbrace{\frac{1}{2L}\|\mathbf{e}_k\|^2}_{\text{bad}}$$

Mini-Batching SGD as Gradient Descent with Error

- SGD progress bound with $s_k = 1/L$ and error is:

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) - \underbrace{\frac{1}{2L}\|\nabla f(\mathbf{x}_k)\|^2}_{\text{good}} + \underbrace{\frac{1}{2L}\|\mathbf{e}_k\|^2}_{\text{bad}}$$

- Relationship between "error-free" rate and "with error" rate:
  - If "error-free" rate is $O(1/k)$, you maintain this rate if $\|\mathbf{e}_k\|^2 = O(1/k)$
  - If "error-free" rate is $O(\rho^k)$, you maintain this rate if $\|\mathbf{e}_k\|^2 = O(\rho^k)$
  - If error goes to zero more slowly, error vanishing rate is the "bottleneck"

- So, need to know how batch-size B_k affects $\|\mathbf{e}_k\|^2$

Mini-Batching SGD as Gradient Descent with Error

- Sample with replacement:

$$\mathbb{E}[\|\mathbf{e}_k\|^2] = \frac{1}{B_k}\sigma^2,$$

  where σ^2 is the variance of the stochastic gradient norm (i.e., doubling the batch-size cuts the error in half)

- Sample without replacement (from a dataset of size N):

$$\mathbb{E}[\|\mathbf{e}_k\|^2] = \frac{N - B_k}{N - 1}\frac{1}{B_k}\sigma^2,$$

  i.e., driving error to zero as batch size approaches N

- Growing batch-size:
  - For $O(\rho^k)$ linear convergence: need $B_{k+1} = B_k/\rho$
  - For $O(1/k)$ sublinear convergence: need $B_{k+1} = B_k + \text{const.}$

Mini-Batching SGD as Gradient Descent with Error

- SGD with mini-batcch:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{s_k}{B_k} \sum_{i=1}^{B_k} \nabla f(\mathbf{x}, \xi_i)$$

- For a fixed B_k: sublinear convergence rate
  - Fixed step-size: sublinear convergence to an error ball around a stationary point
  - Diminishing step-size: sublienar convergence to a stationary point

- Can grow B_k to achieve faster rate:
  - Early iterations: cheap SG iterations
  - Later iterations: Use larger batch-sizes (no need to play with step-sizes)

Outline for First-Order Methods

- Convergence Rate Concept
- The Gradient Descent Method
- The Stochastic Gradient Descent Method
- Variance-Reduced Stochastic First-Order Methods
- Adaptive First-Order Methods

Recap: Stochastic Gradient Descent

- SGD Convergence Performace
  - Constant step-size: SGD converges quickly to an approximation
    - Step-size s and batch size B, converges to a $\frac{s\sigma^2}{B}$-error ball
  - Decreasing step-size: SGD converges slowly to exact solution

- Two "control knobs" to improve SGD convergence performance
  - Decrease (gradually) step-sizes:
    - Improves convergence accuracy
    - Make convergence too slow
  - Increase batch-sizes:
    - Leads to faster rate of iterations
    - Makes setting step-sizes easier
    - But increases the iteration cost

- Question: Could we achieve fast convergence rate with small batch-size?

Stochastic Average Gradient (SAG)

- Growing batch-size B_k eventually requires $O(N)$ samples per iteration
- Question: Can we achieve one sample per iteration and same iteration complexity as deterministic first-order methods?
- Answer: Yes, the first method was the stochastic average gradient (SAG) method [Le Roux et al. 2012]
- To understand SAG, it's insightful to view GD as performing the following iteration in solving the finite-sum problem:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{s_k}{N} \sum_{i=1}^{N} \mathbf{v}_k^i$$

  where in each step we set $\mathbf{v}_k^i = \nabla f_i(\mathbf{x}_k)$ for all i

- SAG method: Only set $\mathbf{v}_k^{i_k} = \nabla f_{i_k}(\mathbf{x}_k)$ for randomly chosen i_k
  - All other $\mathbf{v}_k^{i_k}$ are kept at their previous values (a lazy update approach)

Stochastic Average Gradient (SAG)

- One can think of SAG as having a memory:

$$\begin{bmatrix} \text{——} & \mathbf{v}^1 & \text{——} \\ \text{——} & \mathbf{v}^2 & \text{——} \\ & \vdots & \\ \text{——} & \mathbf{v}^N & \text{——} \end{bmatrix},$$

  where $\mathbf{v}^i$ is the gradient $\nabla f_i(\mathbf{x}_{k'})$ from the last k' where i is selected

- In each iteration:
  - Randomly choose one of the $\mathbf{v}^i$ and update it to the current gradient
  - Take a step in the direction of the average of these $\mathbf{v}^i$

Stochastic Average Gradient (SAG)

- Basic SAG algorithm (maintains $\mathbf{g} = \sum_{i=1}^{N} \mathbf{v}^i$):
  - Set $\mathbf{g} = \mathbf{0}$ and gradient approximation $\mathbf{v}^i = \mathbf{0}$ for $i = 1, \ldots, N$.
  - while (1):
    1. Sample i from $\{1, 2, \ldots, N\}$
    2. Compute $\nabla f_i(\mathbf{x})$
    3. $\mathbf{g} = \mathbf{g} - \mathbf{v}^i + \nabla f_i(\mathbf{x})$
    4. $\mathbf{v}^i = \nabla f_i(\mathbf{x})$
    5. $\mathbf{x}^+ = \mathbf{x} - \frac{s}{N}\mathbf{g}$

- Iteration cost is $O(d)$ (one sample)

- Memory complexity is $O(Nd)$
  - Could be less if the model is sparse
  - Could reduce to $O(N)$ for linear models $f_i(\mathbf{x}) = h(\mathbf{x}^\top \boldsymbol{\xi}^i)$:

$$\nabla f_i(\mathbf{x}) = \underbrace{h'(\mathbf{x}^\top \boldsymbol{\xi}^i)}_{\text{scalar}} \underbrace{\mathbf{x}^i}_{\text{data}}$$

  - But for neural networks, would still need to store all activations (typically impractical)

Stochastic Average Gradient (SAG)

- The SAG algorithm:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{s_k}{N} \sum_{i=1}^{N} \mathbf{v}_k^i,$$

  where in each iteration, $\mathbf{v}_k^{i_k} = \nabla f_{i_k}(\mathbf{x}_k)$ for a randomly chosen i_k

- Unlike batching in SGD, use a "gradient" for every sample
  - But the gradient might be out of date due to lazy update

- Intuition: $\mathbf{v}_k^i \to \nabla f_i(\mathbf{x}^*)$ at the same rate that $\mathbf{x}_k \to \mathbf{x}^*$
  - so the variance $\|\mathbf{e}_k\|^2$ ("bad term") converges linearly to 0

Convergence Rate of SAG

Theorem 6 ([Le Roux et al. 2012])

If each ∇f_i is L-Lipschitz continuous and f is strongly convex, with $s_k = 1/16L$, SAG satisfies:

$$\mathbb{E}[f(\mathbf{x}_k) - f^*] = O\left(\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^k\right)$$

- Sample Complexity: Number of ∇f_i evaluations to reach accuracy ϵ:
  - Stochastic: $O(\frac{L}{\mu}(1/\epsilon))$
  - Gradient: $O(n\frac{L}{\mu}\log(1/\epsilon))$
  - Nesterov: $O(n\sqrt{\frac{L}{\mu}}\log(1/\epsilon))$
  - SAG: $O(\max\{n, \frac{L}{\mu}\}\log(1/\epsilon))$

- Note: L values are different between algorithms

Stochastic Variance-Reduced Gradient (SVRG)

Idea: Get rid of memory by periodically computing full gradient [Johnson&Zhang,'13]

- Start with some $\tilde{\mathbf{x}}^0 = \mathbf{x}_m^0 = \mathbf{x}_0$, where m is a parameter. Let $S = \lceil T/m \rceil$
- for $s = 0, 1, 2, \ldots, S-1$
  - $\mathbf{x}_0^{s+1} = \mathbf{x}_m^s$
  - $\nabla f(\tilde{\mathbf{x}}^s) = \frac{1}{N} \sum_{i=1}^N \nabla f_i(\tilde{\mathbf{x}}^s)$
  - for $k = 0, 1, 2, \ldots, m-1$
    - Uniformly pick a batch $I_k \subset \{1, 2, \ldots, N\}$ at random (with replacement), with batch size $|I_k| = B$
    - Let $\mathbf{v}_k^{s+1} = \frac{1}{B} \sum_{i=1}^B [\nabla f_{i_k}(\mathbf{x}_k^{s+1}) - \nabla f_{i_k}(\tilde{\mathbf{x}}^s)] + \nabla f(\tilde{\mathbf{x}}^s)$
    - $\mathbf{x}_{k+1} = \mathbf{x}_k - s_k \mathbf{v}_k^{s+1}$
  - $\tilde{\mathbf{x}}^{s+1} = \mathbf{x}_m^{s+1}$
- Output: Chose $\mathbf{x}_a$ uniformly at random from $\{\{\mathbf{x}_k^{s+1}\}_{k=0}^{m-1}\}_{s=0}^{S-1}$

Convex settings: Convergence properties similar to SAG for suitable m

- Unbiased: $\mathbb{E}[\mathbf{v}_k^{s+1}] = \nabla f(\mathbf{x}_k^{s+1})$
- Theoretically m depends on L, μ, and N ($m = N$ works well empirically)
- $O(d)$ storage complexity (2B+1 gradients per iteration on average)
- Last step $\tilde{\mathbf{x}}^{s+1}$ in outer loop can be randomly chosen from inner loop iterates

Convergence Rate of SVRG (Nonconvex)

- Consider finite-sum problem $\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \triangleq \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x})$, where both $f(\cdot)$ and $f_i(\cdot)$ are nonconvex, differentiable, and L-smooth.
- Define a sequence $\{\Gamma_k\}$ with $\Gamma_k \triangleq s_k - \frac{c_{k+1}s_k}{\beta_k} - s_k^2 L - 2c_{k+1}s_k^2$, where parameters c_{k+1} and β_k are TBD shortly.

Theorem 7 ([Reddi et al. '16])

Let $c_m = 0$, $s_k = s > 0$, $\beta_k = \beta > 0$, and $c_k = c_{k+1}(1 + s\beta + 2s^2L^2/B) + s^2L^3/B$ such that $\Gamma_k > 0$ for $k = 0, \ldots, m-1$. Let $\gamma = \min_k \Gamma_k$. Also, let T be a multiple of m. Then, the output $\mathbf{x}_a$ of SVRG satisfies:

$$\mathbb{E}[\|\nabla f(\mathbf{x}_a)\|^2] \leq \frac{f(\mathbf{x}_0) - f^*}{T\gamma}.$$

SAGA (SAG Again?)

Basic SAGA algorithm [Defazio et al. 2014]: Similar in spirit to SAG

- Initialize $\mathbf{x}_0$; Create a table, containing gradients and $\mathbf{v}_0^i = \nabla f_i(\mathbf{x}_0)$
- In iterations $k = 0, 1, 2, \ldots$:
  1. Pick a random $i_k \in \{1, \ldots, N\}$ uniformly at random and compute $\nabla f_{i_k}(\mathbf{x}_k)$.
  2. Update $\mathbf{x}_{k+1}$ as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k \left(\nabla f_{i_k}(\mathbf{x}_k) - \mathbf{v}_k^{i_k} + \frac{1}{N} \sum_{i=1}^{N} \mathbf{v}_k^i \right)$$

  3. Update table entry $\mathbf{v}_k^{i_{k+1}} = \nabla f_i(\mathbf{x}_k)$. Set all other $\mathbf{v}_{k+1}^i = \mathbf{v}_k^i$, $i \neq i_k$, i.e., other table entries remain the same

SAGA (SAG Again?)

- SAGA basically matches convergence rates of SAG (for both convex and strongly convex cases), but the proof is simpler (due to unbiasedness)

- Another strength of SAGA is that it can extend to composite problems:

$$\min_{\mathbf{x}} \frac{1}{N} \sum_{i=1}^{N} f_i(\mathbf{x}) + h(\mathbf{x}),$$

where each $f_i(\cdot)$ is L-smooth, and h is convex and non-smooth, but has a known proximal operator

$$\mathbf{x}_{k+1} = \text{prox}_{h,s_k} \left\{ \mathbf{x}_k - s_k \left(\nabla f_{i_k}(\mathbf{x}_k) - \mathbf{v}_k^{i_k} + \frac{1}{N} \sum_{i=1}^{N} \mathbf{v}_k^i \right) \right\}.$$

But it is unknown whether SAG is convergent or not under proximal operator

SAGA Variance Reduction

- Stochastic gradient in SAGA:

$$\underbrace{\nabla f_{i_k}(\mathbf{x}_k)}_{X} - \underbrace{\left(\mathbf{v}_k^{i_k} - \frac{1}{N}\sum_{i=1}^{N}\mathbf{v}_k^i\right)}_{Y}$$

- Note: $\mathbb{E}[X] = \nabla f(\mathbf{x}_k)$ and $\mathbb{E}[Y] = 0 \Rightarrow$ we have an unbiased estimator

- Note: $X - Y \to 0$ as $k \to \infty$, since $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$ converges to some $\bar{\mathbf{x}}$, the difference between the first two terms converges to zero. The last term converges to gradient at stationarity, i.e., also zero

- Thus, the overall ℓ_2 norm estimator (i.e., variance) decays to zero

Comparisons between SAG, SVRG, and SAGA

A general variance reduction approach: Want to estimate $\mathbb{E}[X]$

- Suppose we can compute $\mathbb{E}[Y]$ for a r.v. Y that is highly correlated with X
- Consider the estimator θ_a as an approximation to $\mathbb{E}[X]$:

$$\theta_\alpha \triangleq \alpha(X - Y) + \mathbb{E}[Y], \text{ for some } \alpha \in (0, 1]$$

- Observations:
  - $\mathbb{E}[\theta_\alpha] = \alpha\mathbb{E}[X] + (1-\alpha)\mathbb{E}[Y]$, i.e., a convex combination of $\mathbb{E}[X]$ and $\mathbb{E}[Y]$.
  - Standard VR: $\alpha = 1$ and hence $\mathbb{E}[\theta_\alpha] = \mathbb{E}[X]$
  - Variance of θ_α: $\mathrm{Var}(\theta_\alpha) = \alpha^2[\mathrm{Var}(X) + \mathrm{Var}(Y) - 2\mathrm{Cov}(X, Y)]$
  - If $\mathrm{Cov}(X, Y)$ is large, variance of θ_α is reduced compared to X
  - Letting α from 0 to 1, $\mathrm{Var}(X) \uparrow$ to max value while decreasing bias to zero

- SAG, SVRG, and SAGA can be derived from this VR viewpoint:
  - SAG: Let $X = \nabla f_{i_k}(\mathbf{x}_k)$ and $Y = \mathbf{v}_k^{i_k}$, $\alpha = 1/N$ (biased)
  - SAGA: Let $X = \nabla f_{i_k}(\mathbf{x}_k)$ and $Y = \mathbf{v}_k^{i_k}$, $\alpha = 1$ (unbiased)
  - SVRG: Let $X = \nabla f_{i_k}(\mathbf{x}_k)$ and $Y = \nabla f_{i_k}(\tilde{\mathbf{x}})$, $\alpha = 1$ (unbiased)
  - Variance of SAG is $1/N^2$ times of that of SAGA

Comparisons between SAG, SVRG, and SAGA

- Update rules:

$$\text{(SAG)} \qquad \mathbf{x}_{k+1} = \mathbf{x}_k - s\left[\frac{1}{N}(\nabla f_{i_k}(\mathbf{x}_k) - \mathbf{v}_k^{i_k}) + \frac{1}{N}\sum_{i=1}^{N}\mathbf{v}_k^i\right]$$

$$\text{(SAGA)} \qquad \mathbf{x}_{k+1} = \mathbf{x}_k - s\left[\nabla f_{i_k}(\mathbf{x}_k) - \mathbf{v}_k^{i_k} + \frac{1}{N}\sum_{i=1}^{N}\mathbf{v}_k^i\right]$$

$$\text{(SVRG)} \qquad \mathbf{x}_{k+1} = \mathbf{x}_k - s\left[\nabla f_{i_k}(\mathbf{x}_k) - \nabla f_{i_k}(\tilde{\mathbf{x}}) + \frac{1}{N}\sum_{i=1}^{N}\nabla f_i(\tilde{\mathbf{x}})\right]$$

- SVRG: $\tilde{\mathbf{x}}$ is not updated very step (only updated in the start of outer loops)
- SAG & SAGA: Update $\mathbf{v}_k^{i_k}$ in the table each time index i_k is picked
- SVRG vs. SAGA:
  - SVRG: Low memory cost, slower convergence (same convergence rate order)
  - SAGA: High memory cost, (arguably) faster convergence
- SAGA can be viewed as a midpoint between SAG and SVRG

Stochastic Recursive Gradient Algorithm (SARAH)

- Sample complexity of GD, SGD, SVRG, and SAGA for ϵ-stationarity:
  - GD and SGD require $O(N\epsilon^{-2})$ and $O(\epsilon^{-4})$, respectively[1]
  - $B = 1$: Both SVRG and SARAH guarantee only $O(N\epsilon^{-2})$, same as GD
  - $B = N^{\frac{2}{3}}$: Both SVRG and SAGA achieve $O(N^{\frac{2}{3}}\epsilon^{-2})$, $N^{\frac{1}{3}}$ times better than GD in terms of dependence on N

- However, the sample complexity lower bound is $\Omega(\sqrt{N}\epsilon^{-2})$
  - There exist sample complexity order-optimal algorithms (e.g., SPIDER [Fang et al. 2018] and PAGE [Li et al. 2020])

- These order-optimal algorithms are variants of SARAH [Nguyen et al. 2017]
  - Sample complexity for convex and strongly convex problems: $O(N + 1/\epsilon^2)$ and $O((N + \kappa)\log(1/\epsilon))$, respectively ($\kappa = L/\mu$, a single outer loop)
  - Sample complexity for nonconvex problems: $O(N + L^2/\epsilon^4)$ (step size $s = O(1/L\sqrt{T})$, non-batching, a single outer loop)

[1] For simplicity, we ignore all other parameters except N and ϵ here.

Stochastic Recursive Gradient Algorithm (SARAH)

The SARAH algorithm:

- Pick learning rate $\eta > 0$ and inner loop size m
- for $s = 0, 1, 2, \ldots, S-1$
  - $\mathbf{x}_0^{s+1} = \tilde{\mathbf{x}}^s$
  - $\mathbf{v}_0^{s+1} = \frac{1}{N} \sum_{i=1}^N \nabla f_i(\mathbf{x}_0^{s+1})$
  - $\mathbf{x}_1^{s+1} = \mathbf{x}_0^{s+1} - \eta \mathbf{v}_0^{s+1}$
  - for $k = 1, 2, \ldots, m-1$
    - Uniformly pick a batch $I_k \subset \{1, 2, \ldots, N\}$ at random (with replacement), with batch size $|I_k| = B$
    - Let $\mathbf{v}_k^{s+1} = \frac{1}{B} \sum_{i \in I_k} [\nabla f_{i_k}(\mathbf{x}_k^{s+1}) - \nabla f_{i_k}(\mathbf{x}_{k-1}^{s+1})] + \mathbf{v}_{k-1}^{s+1}$
    - $\mathbf{x}_{k+1}^{s+1} = \mathbf{x}_k^{s+1} - \eta \mathbf{v}_k^{s+1}$
  - $\tilde{\mathbf{x}}^{s+1} = \mathbf{x}_k^{s+1}$ with k chosen uniformly at random from $\{0, 1, \ldots, m\}$
- Output: Chose $\mathbf{x}_a$ uniformly at random from $\{\{\mathbf{x}_k^{s+1}\}_{k=0}^{m-1}\}_{s=0}^{S-1}$

Comparison to SVRG (ignoring outer loop index s):

- SVRG: $\mathbf{v}_k = \nabla f_{i_k}(\mathbf{x}_k) - \nabla f_{i_k}(\mathbf{x}_0) + \mathbf{v}_0$ (unbiased)
- SARAH: $\mathbf{v}_k = \nabla f_{i_k}(\mathbf{x}_k) - \nabla f_{i_k}(\mathbf{x}_{k-1}) + \mathbf{v}_{k-1}$ (biased)

SPIDER/SpiderBoost

- SPIDER [Fang et al. 2018]: Provides the first sample complexity lower bound and the first sample complexity order-optimal algorithm
  - SPIDER stands for "stochastic path-integrated differential estimator"
  - Lower bound $O(\sqrt{N}\epsilon^{-2})$ for small data regime $N = O(L^2(f(\mathbf{x}_0) - f^*)\epsilon^{-4})$
  - SPIDER achieves sample complexity $O(\sqrt{N}\epsilon^{-2})$
  - However, requires very small step-size $O(\epsilon/L)$, poor convergence in practice
  - Original proof of SPIDER is technically too complex and hence hard to generalize the method to composite optimization problems

- SpiderBoost [Wang et al. 2018] [Wang et al. NeurIPS'19]:
  - Same algorithm, same sample complexity, but relax the step-size to $O(1/L)$
  - Simpler proof and can be generalized to composite optimization problems
  - Also works well with heavy-ball momentum

SPIDER/SpiderBoost

The SpiderBoost Algorithm

- Pick learning rate $s = 1/2L$, epoch length m, starting point $\mathbf{x}_0$, batch size B, number of iteration T
- **for** $k = 0, 1, 2, \ldots, T-1$
  - **if** $k \mod m = 0$ **then**
    - Compute full gradient $\mathbf{v}_k = \nabla f(\mathbf{x}_k)$
  - **else**
    - Uniformly randomly pick $I_k \subset \{1, \ldots, N\}$ (with replacement) with $|I_k| = B$. Compute

$$\mathbf{v}_k = \frac{1}{B} \sum_{i \in I_k} [\nabla f_i(\mathbf{x}_k) - \nabla f_i(\mathbf{x}_{k-1})] + \mathbf{v}_{k-1}$$

  - **end if**
  - Let $\mathbf{x}_{k+1} = \mathbf{x}_k - s\mathbf{v}_k$
- **end for**
- **Output:** $\mathbf{x}_\xi$, where ξ is picked uniformly at random from $\{0, \ldots, T-1\}$

Probabilistic Gradient Estimator (PAGE)

- SPIDER/SpiderBoost: Sample complexity LB is for small data regime
- PAGE [Li et al. ICML'21]: Proved the lower bound $\Omega(N + \sqrt{N}\epsilon^{-2})$ without any assumption on data set size N and provided a new order-optimal method
  - A variant of SPIDER with random length of inner loop, making the algorithm easier to analyze

Probabilistic Gradient Estimator (PAGE)

The PAGE Algorithm

- Pick $\mathbf{x}_0$, step-size s, mini-batch sizes B and $B' < B$, probabilities $\{p_k\}_{k\geq 0} \in (0,1]$, number of iterations T
- Let $\mathbf{g}_0 = \frac{1}{B}\sum_{i\in I} \nabla f_i(\mathbf{x}_0)$, where I is a random mini-batch with $|I| = B$
- **for** $k = 0, 1, 2, \ldots, T-1$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s\mathbf{g}_k,$$

$$\mathbf{g}_{k+1} = \begin{cases} \frac{1}{B}\sum_{i\in I_k} \nabla f_i(\mathbf{x}_{k+1}), & \text{w.p. } p_k, \\ \mathbf{g}_k + \frac{1}{B'}\sum_{i\in I'_k} [\nabla f_i(\mathbf{x}_{k+1}) - \nabla f_i(\mathbf{x}_k)], & \text{w.p. } 1-p_k, \end{cases}$$

  where $|I_k| = B$ and $|I'_k| = B'$
  end for
- **Output:** $\hat{\mathbf{x}}_T$ chosen uniformly from $\{\mathbf{x}_k\}_{k=1}^T$

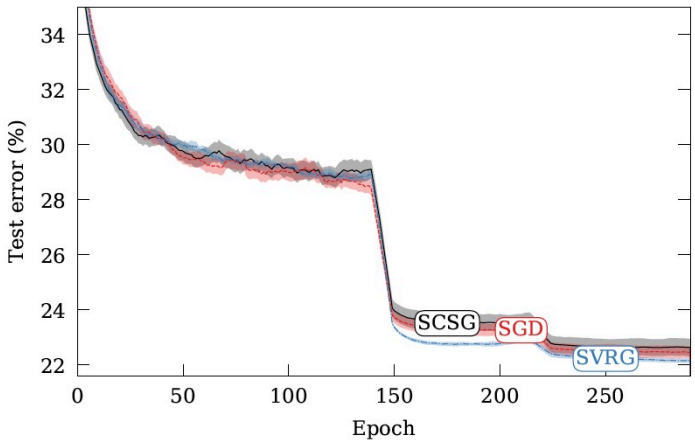
Summary of Sample Complexity Results for VR Methods

Method	References	Sample Complexity
Lower Bound	[Fang et al. NeurIPS'18]	$L\Delta_0 \min\{\sigma\epsilon^{-3}, \sqrt{N}\epsilon^{-2}\}$
GD		$NL\Delta_0\epsilon^{-2}$
SGD (bnd. var.)	[Ghadimi & Lan, SIAM-JO'13]	$L\Delta_0 \max\{\epsilon^{-2}, \sigma^2\epsilon^{-4}\}$
SGD (ubd. var.)	[Khaled & Richtarik, '20]	$\frac{L^2\Delta_0}{\epsilon^4} \max\{\Delta_0, \Delta_*\}$
SVRG ($B=1$)	[Reddi et al. NeurIPS'16]	$NL\Delta_0\epsilon^{-2}$
SVRG ($B=\lceil N^{\frac{2}{3}} \rceil$)	[Reddi et al. NeurIPS'16]	$N^{\frac{2}{3}} L\Delta_0\epsilon^{-2}$
SAGA ($B=1$)	[Reddi et al. NeurIPS'16]	$NL\Delta_0\epsilon^{-2}$
SAGA ($B=\lceil N^{\frac{2}{3}} \rceil$)	[Reddi et al. NeurIPS'16]	$N^{\frac{2}{3}} L\Delta_0\epsilon^{-2}$
SpiderBoost	[Wang et al. NeurIPS'19]	$N^{\frac{1}{2}} L\Delta_0\epsilon^{-2}$
SPIDER	[Fang et al. NeurIPS'18]	$L\Delta_0 \min\{\sigma\epsilon^{-3}, \sqrt{N}\epsilon^{-2}\}$
PAGE	[Li et al. ICML'21]	$L\Delta_0 \min\{\sigma\epsilon^{-3}, \sqrt{N}\epsilon^{-2}\}$

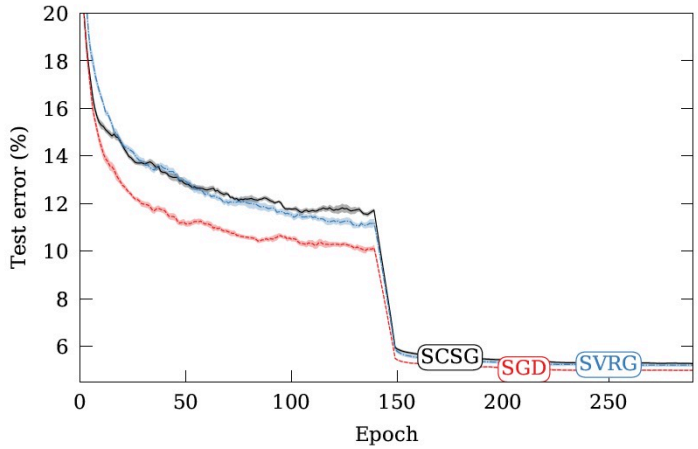
- Notation: $\Delta_0 = f(\mathbf{x}_0) - f^*$, $\Delta_* = \frac{1}{N}\sum_{i=1}^N (f^* - f_i^*)$, σ^2 is a uniform bound for the variance of stochastic gradient, B is batch size
- All results are for finite-sum with L-smooth summands. Sample complexity means the overall number of stochastic first-order oracle calls to find an ϵ-stationary point

Caveat of Variance-Reduced Methods

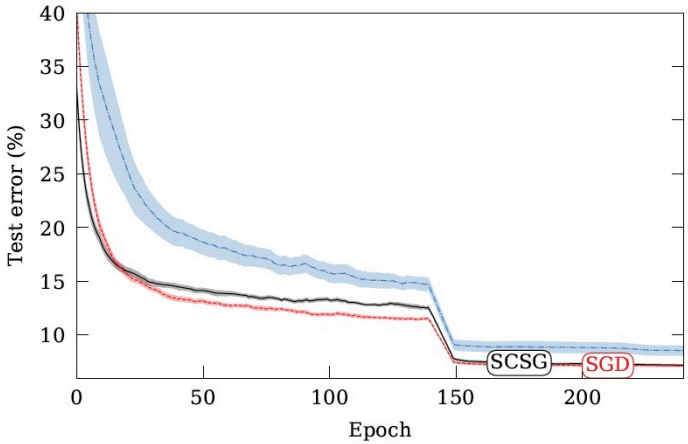
- In deep neural networks training, VR methods work typically worse than SGD or SGD+Momentum [Defazio & Bottou, NeurIPS'19]
  - Bad behavior of VR methods with several widely used deep learning tricks (e.g., batch normalization, data augmentation and dropout)



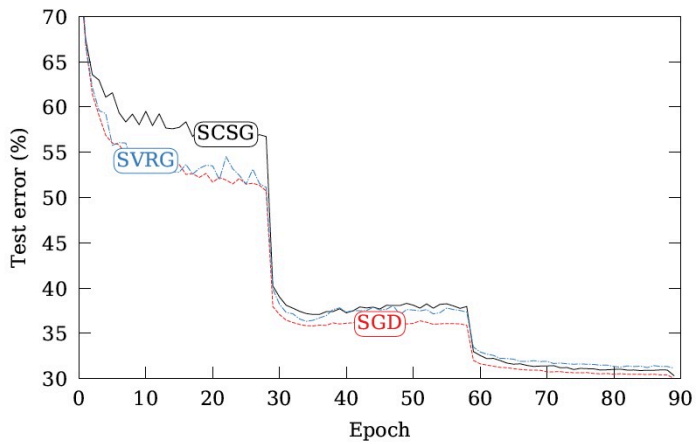
(a) LeNet on CIFAR10



(b) DenseNet on CIFAR10



(c) ResNet-110 on CIFAR10



(d) ResNet-18 on ImageNet

Outline for First-Order Methods

- Convergence Rate Concept
- The Gradient Descent Method
- The Stochastic Gradient Descent Method
- Variance-Reduced Stochastic First-Order Methods
- Adaptive First-Order Methods

Motivation

- Recall that SGD has two hyber-parameter "control knobs" for convergence performance
  - Step-size
  - Batch-size

- A significant issue in SGD and variance-reduced versions: Tuning parameters
  - Time-consuming, particularly for training deep neural networks
  - Thus, adaptive first-order methods have received a lot of attention

- The most popular ones that spawn many variants:
  - AdaGrad: [Duchi et al. JMLR'11]
  - RMSProp: [Hinton, '12]
  - Adam: [Kingma & Ba, ICLR'15] (AMSGrad [Reddi et al. ICLR'18])
  - All of these methods still depend on some hyper-parameters, but they are more robust than other variants of SGD or variance-reduced methods
  - One can find PyTorch implementations of these popular adaptive first-order meth methods

AdaGrad

- AdaGrad stands for "adaptive gradient." It is the first algorithm aiming to remove the need for turning the step-size in SGD:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s(\delta \mathbf{I} + \mathrm{Diag}\{\mathbf{G}_k\})^{-\frac{1}{2}} \mathbf{g}_k,$$

  where $\mathbf{G}_k = \sum_{t=1}^{k} \mathbf{g}_t \mathbf{g}_t^\top$, s is an initial learning rate, and $\delta > 0$ is a small value to prevent from the division by zero (typically on the order of 10^{-8})

- Entry-wise version: ($\mathbf{a}_{k,i}$ denotes the i-th entry of $\mathbf{a}_k$)

$$\mathbf{x}_{k+1,i} = \mathbf{x}_{k,i} - \frac{s_k}{\sqrt{\delta + G_{k,i}}} \mathbf{g}_{k,i},$$

  where $G_{k,i} = \sum_{t=1}^{k} (\mathbf{g}_{t,i})^2$. Typically, $s_k = s$, $\forall k$.

- AdaGrad can be viewed as a special case of SGD with an adaptively scaled step-size (learning rate) for each dimension (feature).

RMSProp

- A major limitation of AdaGrad:
  - Step-sizes could rapidly diminishing (particularly in dense settings), may get stuck in saddle points in nonconvex optimization

- RMSProp (root mean squared propagation)
  - First appeared in Hinton's Lecture 6 notes of the online course "Neural Networks for Machine Learning."
  - Motivated by RProp [Igel & Hüsken, NC'00] (resolving the issue that gradients may vary widely in magnitudes, only using the sign of the gradient)
  - Unpublished (and being famous because of this! ☺)
  - Idea: Keep an exponential moving average of squared gradient of each weight

$$\mathbb{E}[\mathbf{g}^2_{k+1,i}] = \beta\mathbb{E}[\mathbf{g}^2_{k,i}] + (1-\beta)(\nabla_i f(\mathbf{x}_k))^2,$$
$$\mathbf{x}_{k+1,i} = \mathbf{x}_{k,i} - \frac{s_k}{(\delta + \mathbb{E}[\mathbf{g}^2_{k+1,i}])^{\frac{1}{2}}}\nabla_i f(\mathbf{x}_k).$$

- RMSProp vs. AdaGrad
  - AdaGrad: Keep a running sum of squared gradients
  - RMSProp: Keep an exponential moving average of squared gradients

Adam

- Stands for adaptive momentum estimation [Kingma & Ba, ICLR'15]
- Motivated by RMSProp, also aims to address the limitation of AdaGrad
- Algorithm: $(\mathbf{g}_k \triangleq \nabla f(\mathbf{x}_k))$

$$\mathbf{m}_{k,i} = \beta_1 \mathbf{m}_{k-1,i} + (1-\beta_1)\mathbf{g}_{k,i}, \qquad \hat{\mathbf{m}}_{k,i} = \frac{\mathbf{m}_{k,i}}{1-(\beta_1)^k},$$

$$\mathbf{v}_{k,i} = \beta_2 \mathbf{v}_{k-1,i} + (1-\beta_2)(\mathbf{g}_{k,i})^2, \qquad \hat{\mathbf{v}}_{k,i} = \frac{\mathbf{v}_{k,i}}{1-(\beta_2)^2},$$

$$\mathbf{x}_{k+1,i} = \mathbf{x}_{k,i} - \frac{s_k}{\sqrt{\hat{\mathbf{v}}_{k,i}} + \delta}\hat{\mathbf{m}}_{k,i}, \qquad i = 1, \ldots, d.$$

- Parameters:
  - $\beta_1 \in [0, 1)$: momentum parameter ($\beta_1 = 0.9$ by default, $\beta_1 = 0 \Rightarrow$ RMSProp)
  - $\beta_2 \in (0, 1)$: exponential average parameter ($\beta_2 = 0.999$ in the original paper)
- A flaw in convergence proof spotted by [Reddi et al. ICLR'18], leading to...

AMSGrad

- Idea: Use a smaller learning rate and incorporate the intuition of slowly decaying the effect of past gradient
- The algorithm: In iteration k:

$$\mathbf{g}_k = \nabla f_k(\mathbf{x}_k)$$
$$\mathbf{m}_k = \beta_{1,k}\mathbf{m}_{k-1} + (1-\beta_{1,k})\mathbf{g}_k,$$
$$\mathbf{v}_k = \beta_2\mathbf{v}_{k-1} + (1-\beta_2)\mathbf{g}_k \circ \mathbf{g}_k,$$
$$\hat{\mathbf{v}}_k = \max(\hat{\mathbf{v}}_{k-1}, \mathbf{v}_k), \text{ and } \hat{\mathbf{V}}_k = \mathrm{Diag}(\hat{\mathbf{v}}_k)$$
$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k\hat{\mathbf{V}}_k^{-\frac{1}{2}}\mathbf{m}_k$$

- Maintain the maximum of all $\mathbf{v}_k$ until the present iteration and use the maximum to ensure non-increasing learning rate

Convergence of Adaptive First-Order Methods

- While faster convergence of adaptive methods over SGD has been widely observed, their best-known convergence rate bounds so far are the same (or even worse) than those of SGD
- We adopt the proof in [Défossez et al. '20] due to generality and simplicity
- A unified formulation used in [Défossez et al. '20] for AdaGrad and Adam ($0 < \beta_2 \leq 1$ and $0 \leq \beta_1 < \beta_2$):

$$\begin{aligned}
\mathbf{m}_{k,i} &= \beta_1 \mathbf{m}_{k-1,i} + \nabla_i f_k(\mathbf{x}_{k-1}), \\
\mathbf{v}_{k,i} &= \beta_2 \mathbf{v}_{k-1,i} + \left(\nabla_i f_k(\mathbf{x}_{k-1})\right)^2, \\
\mathbf{x}_{k,i} &= \mathbf{x}_{k-1,i} - s_k \frac{\mathbf{m}_{k,i}}{\sqrt{\delta + \mathbf{v}_{k,i}}},
\end{aligned}$$

  - AdaGrad: $\beta_1 = 0$, $\beta_2 = 1$, and $s_k = s$
  - Adam: Take $s_k = s(1-\beta_1)\sqrt{\frac{1-\beta_2^k}{1-\beta_2}}$

Convergence of Adaptive First-Order Methods

- Consider a general expectation optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} F(\mathbf{x}) \triangleq \min_{\mathbf{x}\in\mathbb{R}^d} \mathbb{E}[f(\mathbf{x})]$$

- Notation: For a given time horizon $T \in \mathbb{N}$, let τ_T be a random index with value in $\{0, \ldots, T-1\}$ so that $\Pr[\tau_T = j] \propto 1 - \beta_1^{T-j}$
  - $\beta_1 = 0$: Sampling τ_T uniformly in $\{0, \ldots, T-1\}$ (note: no momentum)
  - $\beta_1 > 0$: The fast few $\frac{1}{1-\beta_1}$ iterations are sampled relatively rarely and older iterations are sampled approximately uniformly

- Assumptions:
  - F is bounded from below: $F(\mathbf{x}) \geq F^*$, $\mathbf{x} \in \mathbb{R}^d$
  - ℓ_∞ norm of stochastic gradients is uniformly bounded almost surely: $\exists \epsilon > 0$ s.t. $\|\nabla f(\mathbf{x})\|_\infty \leq R - \sqrt{\epsilon}$ a.s.
  - L-smoothness: $\|\nabla F(\mathbf{x}) - F(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2$, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$

Convergence of Adaptive First-Order Methods

Theorem 8 (Adam w/o Momentum, (AdaGrad))

Let the iterates $\{\mathbf{x}_k\}$ *be generated with* $\beta_2 = 1$, $s_k = s > 0$, *and* $\beta_1 = 0$. *Then for any* $T \in \mathbb{N}$, *we have:*

$$\mathbb{E}[\|\nabla F(\mathbf{x}_{\tau_T})\|^2] \leq 2R\frac{F(\mathbf{x}_0) - F^*}{s\sqrt{T}} + \frac{1}{\sqrt{T}}(4dR^2 + sdRL)\ln\left(1 + \frac{TR^2}{\epsilon}\right).$$

Theorem 9 (Adam w/o Momentum (RMSProp))

Let the iterates $\{\mathbf{x}_k\}$ *be generated with* $\beta_2 \in (0,1)$, $s_k = s\sqrt{\frac{1-\beta_2^k}{1-\beta_2}}$ *with* $s > 0$, *and* $\beta_1 = 0$. *Then for any* $T \in \mathbb{N}$, *we have:*

$$\mathbb{E}[\|\nabla F(\mathbf{x}_{\tau_T})\|^2] \leq 2R\frac{F(\mathbf{x}_0) - F^*}{sT} + C\left(\frac{1}{T}\ln\left(1 + \frac{R^2}{(1-\beta_2)\epsilon}\right) - \ln(\beta_2)\right),$$

where constant $C \triangleq \frac{4dR^2}{\sqrt{1-\beta_2}} + \frac{sdRL}{1-\beta_2}$.

Convergence of Adaptive First-Order Methods

Theorem 10 (AdaGrad w/ Momentum)

Let the iterates $\{\mathbf{x}_k\}$ *be generated with* $\beta_2 = 1$, $s_k = s > 0$, *and* $\beta_1 \in (0, 1)$. *Then for any* $T \in \mathbb{N}$ *such that* $T > \frac{\beta_1}{1-\beta_1}$, *we have:*

$$\mathbb{E}[\|\nabla F(\mathbf{x}_{\tau_T})\|^2] \leq 2R\sqrt{T}\frac{F(\mathbf{x}_0) - F^*}{s\tilde{T}} + \frac{\sqrt{T}}{\tilde{T}} C \ln\left(1 + \frac{TR^2}{\epsilon}\right).$$

where $\tilde{T} = T - \frac{\beta_1}{1-\beta_1}$ *and* $C = sdRL + \frac{12dR^2}{1-\beta_1} + \frac{2s^2dL^2\beta_1}{1-\beta_1}$.

Theorem 11 (Adam w/ Momentum)

Let $\{\mathbf{x}_k\}$ *be generated with* $\beta_2 \in (0, 1)$, $\beta_1 \in [0, \beta_2)$, *and* $s_k = s(1-\beta_1)\sqrt{\frac{1-\beta_2^k}{1-\beta_2}}$ *with* $s > 0$. *Then for any* $T \in \mathbb{N}$ *such that* $T > \frac{\beta_1}{1-\beta_1}$, *we have:*

$$\mathbb{E}[\|\nabla F(\mathbf{x}_{\tau_T})\|^2] \leq 2R\frac{F(\mathbf{x}_0) - F^*}{sT} + C\left(\frac{1}{T}\ln\left(1 + \frac{R^2}{(1-\beta_2)\epsilon}\right) - \ln(\beta_2)\right),$$

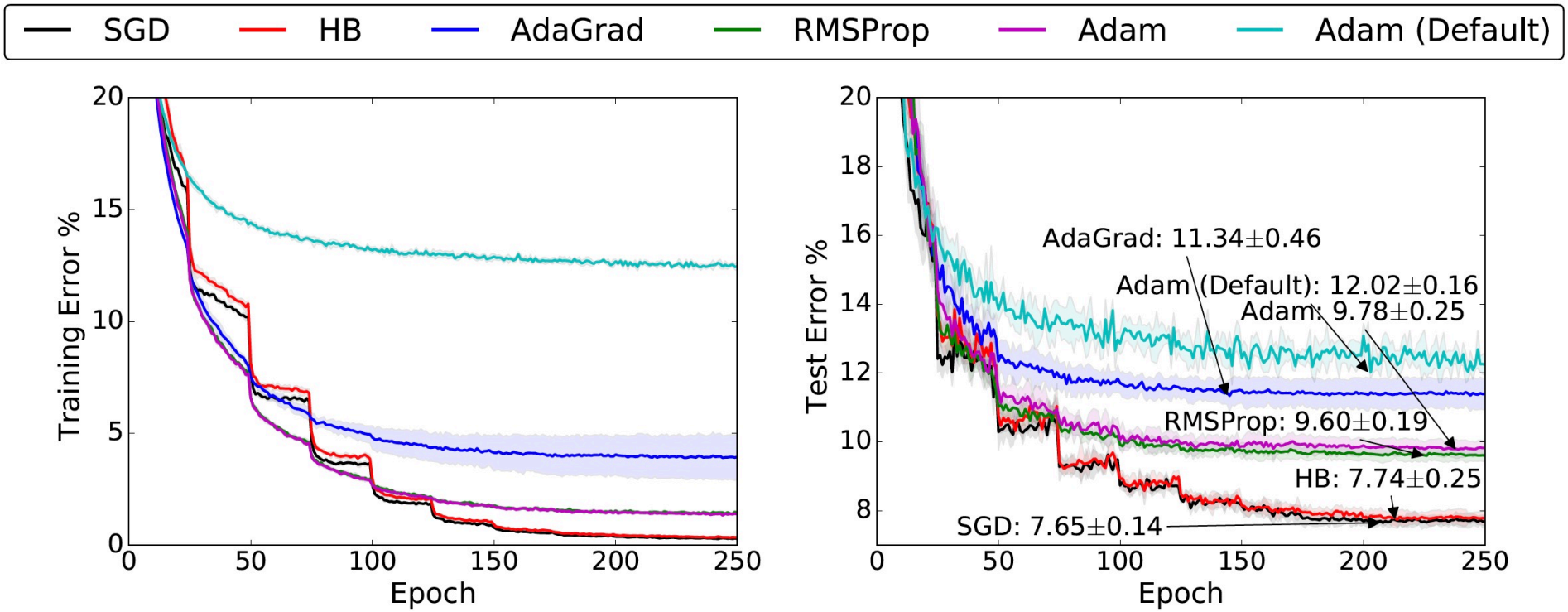
where $\tilde{T} = T - \frac{\beta_1}{1-\beta_1}$ *and* $C = \frac{sdRL(1-\beta_1)}{(1-\frac{\beta_1}{\beta_2})(1-\beta_2)} + \frac{12dR^2\sqrt{1-\beta_1}}{(1-\frac{\beta_1}{\beta_2})^{3/2}\sqrt{1-\beta_2}} + \frac{2s^2dL^2\beta_1}{(1-\frac{\beta_1}{\beta_2})(1-\beta_2)^{3/2}}$.

Theoretical Understanding of Adaptive Methods

- Pros:
  - [Zhang et al. NeurIPS'20]: Adam performs better than SGD when stochastic gradients are heavy-tailed since Adam does an "adaptive gradient clipping"
  - [Zhang et al. NeurIPS'20]: Also shows that SGD can fail to converge under heavy-tailed situations, while clipped-SGD can.
  - [Goodfellow & Bengio, '16]: Clipped-SGD works better than SGD in vicinity of extremely steep cliffs
  - [Zhang et al. ICML'20]: Clipped-GD converges without L-smoothness (with rate ϵ^{-2} while GD may converge arbitrarily slower
- Cons:
  - [Wilson et al. NeurIPS'17]: While converging faster in general, adaptive first-order methods does not have good test error and generalization performances in the over-parameterized regime. Adaptive methods often generalize significantly worse than SGD. So one may need to reconsider the use of adaptive methods to train deep neural networks

Limitations of Adaptive Methods

- [Wilson et al. NeurIPS'17]: VGG+BN+Dropout network for CIFAR-10



(a) CIFAR-10 (Train)

(b) CIFAR-10 (Test)

Part IV

Zeroth-Order Methods for Learning

Overview of Zeroth-Order Methods

- Zeroth-order (gradient free) method: Use only function values
  - Reinforcement learning [Malik et al., AISTATS'20]
  - Blackbox adversarial attacks on DNN [Papernot et al., CCS'17]
  - Or problems with structure making gradients difficult or infeasible to obtain

- Two major classes of zeroth-order methods
  - Methods that do not have any connections to gradient
    - ⋆ Random search algorithm [Schumer and Steiglitz, TAC'68]
    - ⋆ Nelder-Mead algorithm [Nelder and Mead, Comp J. '65]
    - ⋆ Model-based methods [Conn et al., SIAM'09]
    - ⋆ Stochastic three points methods (STP) [Bergou et al., SIAM J. Opt. '20]
    - ⋆ STP with momentum [Gorbunov et al., ICLR'20]
  - Methods that rely on gradient estimations
    - ⋆ More modern approach, the focus of this course

Outline for Zeroth-Order Methods

- Representative Techniques for Random Directions of Gradient Estimations
- Representative Variance-Reduced Zeroth-Order Methods

Basic Idea of (Deterministic) Gradient Estimation

- Gradient estimation with finite-difference directional derivative estimation:

$$\text{(Forward version):}\ \mathbf{g}(\mathbf{x}) = \sum_{i=1}^{d} \frac{f(\mathbf{x}+\mu\mathbf{e}_i) - f(\mathbf{x})}{\mu}\mathbf{e}_i,$$

$$\text{(Centered version):}\ \mathbf{g}(\mathbf{x}) = \sum_{i=1}^{d} \frac{f(\mathbf{x}+\mu\mathbf{e}_i) - f(\mathbf{x}-\mu\mathbf{e}_i)}{2\mu}\mathbf{e}_i,$$

  where $\mathbf{e}_i$ is the i-th natural basis vector of $\mathbb{R}^n$ and μ is the sampling radius

- For the gradient estimation above, it can be shown that for $f \in C_L^{1,1}$ (i.e., continuously differentiable with Lipschitz-continuous gradient)

$$\|\mathbf{g}(\mathbf{x}) - \nabla f(\mathbf{x})\|_2 \leq \mu L\sqrt{d}$$

- Natural idea: Replace actual gradient with gradient estimation in any first-order optimization scheme (deterministic ZO methods)
  - Pro: Use Lipschitz-like bound above to characterize convergence performance
  - Con: Suffer from problem dimensionality for large d ($O(d)$ ZO-oracle calls)

Randomized Gradient Estimation

- Two-point random gradient estimator

$$\hat{\nabla} f(\mathbf{x}) = (d/\mu)[f(\mathbf{x} + \mu\mathbf{u}) - f(\mathbf{x})]\mathbf{u},$$

  where $\mathbf{u}$ is an i.i.d. random direction

- $(q+1)$-point random gradient estimator

$$\hat{\nabla} f(\mathbf{x}) = (d/(\mu q)) \sum_{i=1}^{q} [f(\mathbf{x} + \mu\mathbf{u}_i) - f(\mathbf{x})]\mathbf{u}_i,$$

  which is also referred to as average random gradient estimator

- Benefits:
  - Make every iteration simpler
  - Easy convergence proof
  - For problems limited to only several (or even one) ZO oracle queries

Formalization of Stochastic Zeroth-Order Methods

- Consider the problem of the following form:

$$\min_{\mathbf{x} \in Q \subseteq \mathbb{R}^d} f(\mathbf{x})$$

- A stochastic ZO method generates $\{\mathbf{x}_k\}$ as follows:

$$\mathbf{x}_{k+1} = \mathcal{A}\left(\hat{f}, \mathbf{X}, P, \{\mathbf{x}_i\}_{i=0}^k, \{\mathbf{u}_i\}_{i=0}^k\right)$$

  - $\hat{f}$: ZO-oracle (could be noisy, i.e., $\hat{f}$ is not necessarily equal to f; e.g., $\hat{f}(\mathbf{x}) = f(\mathbf{x}) + \epsilon(\mathbf{x})$ or $\hat{f}(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}) + \epsilon(\mathbf{x}, \mathbf{u})$ with $\mathbb{E}_{\mathbf{u}}[\hat{f}(\mathbf{x}, \mathbf{u})] = f(\mathbf{x})$)
  - $\{\mathbf{x}_i\}_{i=0}^k$: history of $\mathbf{x}$-variables
  - $\{\mathbf{u}_i\}_{i=0}^k$: random sampling directions
  - P: parameters (dimension d of $\mathbf{x}$, L-Lipschitz constant, etc.)

- This lecture: Focus on non-convex objective function

Random Directions Gradient Estimations

- Consider the following ZO scheme using gradient approximation:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k),$$

where $\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k)$ follows the two-point random gradient estimator:

$$\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) = \frac{\hat{f}(\mathbf{x}_k + \mu \mathbf{u}_k) - \hat{f}(\mathbf{x}_k)}{\mu} \mathbf{u}_k$$

- It makes sense to use centrally symmetric distributions for $\mathbf{u}_k$:
  - Uniformly distributed over unit Euclidean sphere [Flaxman et al. SODA'05], [Gorbunov et al. SIOPT'18], [Dvurechensky et al., E. J. OR'21]:

$$\mathbf{u}_k \sim \mathcal{U}\{S^{d-1}\}, \text{ where } S^{d-1} = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 = 1\}$$

  - Gaussian smoothing [Nesterov and Spokoiny, Math Prog.'06]:

$$\mathbf{u}_k \sim \mathcal{N}(0, \mathbf{I}_d)$$

Gaussian Smoothing [Nesterov and Spokoiny, FCM'17]

- Gaussian smoothing approximation:

$$f_\mu(\mathbf{x}) = \frac{1}{\kappa} \int_{\mathbb{R}^d} f(\mathbf{x} + \mu \mathbf{u}) e^{-\frac{1}{2}\|\mathbf{u}\|_2^2} d\mathbf{u},$$

  where $\kappa = \int_{\mathbb{R}^d} e^{-\frac{1}{2}\|\mathbf{u}\|_2^2} d\mathbf{u} = (2\pi)^{d/2}$.

- Good properties:
  - Convexity preservation: If f is convex, so is f_μ
  - Differentiability
  - If $f \in C^{0,0}_{L_0}$ (or $f \in C^{1,1}_{L_1}$), the same holds for f_μ with $L_0(f_\mu) \leq L_0(f)$ (or $L_1(f_\mu) \leq L_1(f)$)
  - $|f_\mu(\mathbf{x}) - f(\mathbf{x})| \leq \mu L_0 \sqrt{d}$ if $f \in C^{0,0}_{L_0}$

Gaussian Smoothing [Nesterov and Spokoiny, FCM'17]

- Consider the following algorithm:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k), \text{ and } \mathbf{u}_k \sim \mathcal{N}(0, \mathbf{I}_d).$$

- For nonconvex $f \in C^{1,1}_{L_1}$, we have (let $U = \{\mathbf{u}_k\}_{k=0}^{K-1}$):

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}_U[\|\nabla f_\mu(\mathbf{x}_k)\|_2^2] \leq 8(d+4)L_1 \left[\frac{f_\mu(\mathbf{x}_0) - f^*}{K} + \frac{3\mu^2(d+4)}{32} L_1\right]$$

- Using the facts that $\|f_\mu(\mathbf{x}) - \nabla f(\mathbf{x})\|_2 \leq \frac{\mu L_1}{2}(d+3)^{\frac{3}{2}}$ and $\|\nabla f(\mathbf{x})\|_2^2 \leq 2\|\nabla f_\mu(\mathbf{x}) - \nabla f(\mathbf{x})\|_2^2 + 2\|\nabla f_\mu(\mathbf{x})\|_2^2$, we obtain:

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}_U[\|\nabla f(\mathbf{x}_k)\|_2^2] \leq 2\frac{\mu^2 L_1^2}{4}(d+3)^3 + 16(d+4)L_1 \left[\frac{f_\mu(\mathbf{x}_0) - f^*}{K} + \frac{3\mu^2(d+4)}{32} L_1\right]$$

Gaussian Smoothing [Nesterov and Spokoiny, FCM'17]

- Choosing $\mu = O(\epsilon/[d^3 L_1])$ ensures $\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}_U[\|\nabla f(\mathbf{x}_k)\|_2^2] \leq \epsilon^2$, which implies the following sample complexity:

$$K = O(d\epsilon^{-2}).$$

- For $f \in C_{L_0}^{0,0}$, we have (let $S_K = \sum_{k=0}^{K-1} s_k$):

$$\frac{1}{S_K}\sum_{k=0}^{K-1} s_k \mathbb{E}_U[\|\nabla f_\mu(\mathbf{x}_k)\|_2^2] \leq \frac{1}{S_K}\left[(f_\mu(\mathbf{x}_0) - f^*) + \frac{1}{\mu} d^{\frac{1}{2}}(d+4)^2 L_0^3 \sum_{k=0}^{K-1} s_k^2\right]$$

- Consider a bounded domain Q with $\mathrm{diam}(Q) \leq R$. To ensure $\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}_U[\|\nabla f_\mu(\mathbf{x}_k)\|_2^2] \leq \epsilon^2$ and $|f_\mu(\mathbf{x}) - f(\mathbf{x})| \leq \delta$, we have the following sample complexity:

$$K = O\left(\frac{d(d+4)^2 L_0^5 R}{\epsilon^4 \delta}\right).$$

- If $s_k \to 0$ and $\mu \to 0$, convergence of $\mathbb{E}_U[\|\nabla f(\mathbf{x}_k)\|_2]$ can also be proved.

Extensions of Gaussian Smoothing to Noisy $\hat{f}$

Consider the following:

- Noisy $\hat{f}$: $|\hat{f}(\mathbf{x}) - f(\mathbf{x})| \leq \delta$
- Hölder continuous gradient (intermediate smoothness)

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L_\nu \|\mathbf{x} - \mathbf{y}\|_2^\nu, \text{ for some } \nu \in [0, 1],$$

which implies the following generalized descent lemma:

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{L_\nu}{1+\nu} \|\mathbf{y} - \mathbf{x}\|^{1+\nu}$$

- To ensure $\frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E}_U[\|\nabla f(\mathbf{x}_k)\|_2^2] \leq \epsilon^2$, we have the following sample complexity [Shibaev et al., Opt. Lett. '21]:

$$K = O\left(\frac{d^{2+\frac{1-\nu}{2\nu}}}{\epsilon^{\frac{2}{\nu}}}\right) \text{ if } \delta = O\left(\frac{\epsilon^{\frac{3+\nu}{2\nu}}}{d^{\frac{3+7\nu}{4\nu}}}\right).$$

Extensions of Gaussian Smoothing to Noisy $\hat{f}$

- Special case of $\nu = 1$ (i.e., $f \in C^{1,1}_{L_1}$): Sample complexity is improved to

$$K = O(d\epsilon^{-2}),$$

  which is d times better than [Nesterov and Spokoiny, FCM'17]

- If $|\hat{f}(\mathbf{x}) - f(\mathbf{x})| \leq \epsilon_f$, where f is convex and 1-Lipschitz and $\epsilon_f \sim \max\{\epsilon^2/\sqrt{d}, \epsilon/d\}$, then [Risteski and Li, NeurIPS'16] showed that there exists an algorithm that finds ϵ-optimal solution (i.e., $\hat{f}(\mathbf{x}) - \hat{f}^* \leq \epsilon$) with sample complexity Poly(d, ϵ^{-1}). Also, the dependence $\epsilon_f(\epsilon)$ is optimal

Randomized Stochastic Gradient-Free Methods

Gaussian smoothing is extended to [Ghadimi and Lan, SIAM J. Opt. '13] [Ghadimi et al., Math Prog. '16] (unconstrained case, i.e., $Q = \mathbb{R}^d$):

- $\hat{f} = F(\mathbf{x}, \xi)$ such that $\mathbb{E}_\xi[F(\mathbf{x}, \xi)] = f(\mathbf{x})$, where ξ is a random variable whose distribution P is supported on $\Xi \subseteq \mathbb{R}^d$
- $F(\cdot, \xi)$ has L_1-Lipschitz continuous gradient
- Consider the following randomized stochastic gradient-free method (RSGF):

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k G(\mathbf{x}_k, \xi_k, \mathbf{u}_k),$$

$$G(\mathbf{x}_k, \xi_k, \mathbf{u}_k) = \frac{\hat{f}(\mathbf{x}_k + \mu \mathbf{u}_k, \xi_k) - \hat{f}(\mathbf{x}_k, \xi_k)}{\mu} \mathbf{u}_k$$

- It follows from $\mathbb{E}_\xi[F(\mathbf{x}, \xi)] = f(\mathbf{x})$ that $\mathbb{E}_{\xi, \mathbf{u}}[G(\mathbf{x}, \xi, \mathbf{u})] = \nabla f_\mu(\mathbf{x})$
- Similar to FO-SGD in [Ghadimi and Lan, SIAM J. Opt. '13], RSGF chooses $\mathbf{x}_R$ from $\{\mathbf{x}_k\}_{k=1}^K$ where R is a r.v. with p.m.f. P_R supported on $\{1, \ldots, K\}$

Randomized Stochastic Gradient-Free Methods

- For $f \in C_{L_1}^{1,1}$, smoothing parameter μ, $D_f = (2(f(\mathbf{x}_1) - f^*)/L)^{\frac{1}{2}}$, and $\mathbb{E}_\xi[\|\nabla \hat{f}(\mathbf{x}, \xi) - \nabla f(\mathbf{x})\|_2^2] \leq \sigma^2$ and p.m.f. of R being:

$$P_R(k) = \frac{s_k - 2L(d+4)s_k^2}{\sum_{i=1}^K (s_i - 2L(d+4)s_i^2)},$$

it then holds that:

$$\frac{1}{L_1}\mathbb{E}[\|\nabla f(\mathbf{x}_R)\|_2^2] \leq \frac{1}{\sum_{k=1}^K [s_k - 2L_1(d+4)s_k^2]} \Bigg[D_f^2 + 2\mu^2(d+4)\times \left(1 + L_1(d+4)^2 \sum_{k=1}^K (\frac{s_k}{4} + Ls_k^2)\right) + 2(d+4)\sigma^2 \sum_{k=1}^K s_k^2 \Bigg],$$

where the expectation is taken w.r.t. R and $\{\xi_k\}$.

Randomized Stochastic Gradient-Free Methods

- Choose constant step-size $s_k = \frac{1}{\sqrt{d+4}} \min\{\frac{1}{4L\sqrt{d+4}}, \frac{\tilde{D}}{\sigma\sqrt{K}}\}$ for some $\tilde{D} > 0$ (some estimation of D_f):

$$\frac{1}{L_1}\mathbb{E}[\|\nabla f(\mathbf{x}_R)\|_2^2] \leq \frac{12(d+4)L_1 D_f^2}{K} + \frac{2\sigma\sqrt{d+4}}{\sqrt{K}}\left(\tilde{D} + \frac{D_f^2}{\tilde{D}}\right)$$

- To ensure $\Pr\{\|\nabla f(\mathbf{x}_R)\|_2^2 \leq \epsilon\} \geq 1-\delta$ (i.e., (ϵ, δ)-solution), the zeroth-order oracle sample complexity is:

$$O\left(\frac{dL_1^2 D_f^2}{\delta\epsilon} + \frac{dL_1^2}{\delta^2}\left(\tilde{D} + \frac{D_f^2}{\tilde{D}}\right)\frac{\sigma^2}{\epsilon^2}\right)$$

Randomized Stochastic Gradient-Free Methods

Two-phase randomized stochastic gradient-free method (2-RSGF) [Ghadimi and Lan, SIAM J. Opt. '13]

- Run RSGF $S = \log(1/\delta)$ times as a subroutine producing a list $\{\bar{\mathbf{x}}_k\}_{k=1}^S$
- Output point $\bar{\mathbf{x}}^*$ is chosen in such a way that:

$$\|\mathbf{g}(\bar{\mathbf{x}}^*)\|_2 = \min_{s=1,\ldots,S} \|\mathbf{g}(\bar{\mathbf{x}}_s)\|_2, \text{ where } \mathbf{g}(\bar{\mathbf{x}}_s) = \frac{1}{T}\sum_{k=1}^{T} G_\mu(\bar{\mathbf{x}}_s, \xi_k, \mathbf{u}_k),$$

  where $G_\mu(\bar{\mathbf{x}}_s, \xi_k, \mathbf{u}_k)$ is defined as in RSGF

- The zeroth-order oracle sample complexity for achieving (ϵ, δ)-solution:

$$O\left(\frac{dL_1^2 D_f^2 \log(1/\delta)}{\epsilon} + dL_1^2\left(\tilde{D} + \frac{D_f^2}{\tilde{D}}\right)^2 \frac{\sigma^2}{\epsilon^2}\log(1/\delta) + \frac{d\log^2(1/\delta)}{\delta}\left(1 + \frac{\sigma^2}{\epsilon}\right)\right)$$

- A more general problem $\min_{\mathbf{x}\in Q\subseteq\mathbb{R}^d} \Psi(\mathbf{x}) = f(\mathbf{x}) + h(\mathbf{x})$ is also solved in [Ghadimi et al., Math Prog.'16]
  - $f \in C_L^{1,1}$: nonconvex; $h(\mathbf{x})$ is simple convex and possibly non-smooth

RSGF Based on Uniform Sampling over Unit Sphere

- Consider the problem $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \triangleq \mathbb{E}_\xi[F(\mathbf{x},\xi)] = \mathbb{E}_\xi[\hat{f}(\mathbf{x},\xi)]$
  - $f(\mathbf{x})$ is L-Lipschitz and μ-smooth
  - $|F(\mathbf{x},\xi)| \leq \Omega$ and F's variance is bounded by V_f
- Stochastic gradient estimation based on uniform sampling over unit sphere:

$$\mathbf{g}(\mathbf{x}_k,\xi_k,\mathbf{u}_k) = n\frac{\hat{f}(\mathbf{x}_k+\mu\mathbf{u}_k,\xi_k) - \hat{f}(\mathbf{x}_k-\mu\mathbf{u}_k,\xi_k)}{2\mu},$$

  where $\mathbf{u}_k \sim \mathcal{U}(S^{n-1})$. The update process is $\mathbf{x}_{k+1} = \mathbf{x}_k - s\mathbf{g}(\mathbf{x}_k,\xi_k,\mathbf{u}_k)$
- After K steps, we have [Sener and Koltun, ICML'20]:

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\|\nabla f(\mathbf{x}_k)\|_2^2] = O\left(\frac{n}{K^{1/2}} + \frac{n^{2/3}}{K^{1/3}}\right)$$

RSGF Based on Uniform Sampling over Unit Sphere

- Consider the case for a given ξ, $F(\mathbf{x}, \xi) = g(r(\mathbf{x}, \theta^*), \Psi^*)$, where $g(\cdot, \Psi)$ and $r(\cdot, \theta)$ are parameterized function classes
  - $r(\cdot, \theta^*) : \mathbb{R}^n \to \mathbb{R}^d$, where $d \ll n$
  - $F(\cdot, \xi) : \mathbb{R}^n \to \mathbb{R}$ is actually defined on a d-dimensional manifold $\mathcal{M}$ for all ξ
- Thus, if one knows the manifold (i.e., θ^*) and g and r are smooth, we have from chain rule: $\nabla f(\mathbf{x}) = J(\mathbf{x}, \theta^*) \nabla_r g(r, \Psi)$, where $J(\mathbf{x}, \theta^*) = \frac{\partial r(\mathbf{x}, \theta^*)}{\partial \mathbf{x}}$. This leads to [Sener and Koltun, ICML'20]:

$$G(\mathbf{x}_k, \xi_k, \mathbf{u}_k) = d \frac{\hat{f}(\mathbf{x}_k + \mu J_q \mathbf{u}_k, \xi_k) - \hat{f}(\mathbf{x}_k - \mu J_q \mathbf{u}_k, \xi_k)}{2\mu} \mathbf{u}_k,$$

where J_q is the orthonomalized $J(\mathbf{x}_k, \theta^*)$ and $\mathbf{u}_k \sim \mathcal{U}(S^{d-1})$. It follows that

$$\frac{1}{K} \sum_{k=1}^{K} \mathbb{E}[\|\nabla f(\mathbf{x}_k)\|_2^2] = O\left(\frac{n^{1/2}}{K} + \frac{n^{1/2} + d + dn^{1/2}}{K^{1/2}} + \frac{d^{2/3} + n^{1/2} d^{2/3}}{K^{1/3}} \right).$$

which is much better than the previous bound for $d \leq n^{1/2}$.

Which Gradient Estimation Works Better?

- Gradient estimations with random directions are worse than finite differences in terms of # of samples required to ensure the norm condition:

$$\|\mathbf{g}(\mathbf{x}) - \nabla f(\mathbf{x})\|_2 \leq \theta \|\nabla f(\mathbf{x})\|_2, \text{ for some } \theta \in [0, 1)$$

- Gradient estimation methods are studied in [Berahas et al., FCM'21]: Compare the # of calls r (i.e., "batch size") to ensure norm condition
  - FFD (Forward Finite Differences): $\sum_{i=1}^{d} \frac{\hat{f}(\mathbf{x}+\mu\mathbf{e}_i)-\hat{f}(\mathbf{x})}{\mu}\mathbf{e}_i$
  - CFD (Centered Finite Differences): $\sum_{i=1}^{d} \frac{\hat{f}(\mathbf{x}+\mu\mathbf{e}_i)-\hat{f}(\mathbf{x}-\mu\mathbf{e}_i)}{2\mu}\mathbf{e}_i$
  - LI (Linear Interpolation): $\sum_{i=1}^{d} \frac{\hat{f}(\mathbf{x}+\mu\mathbf{u}_i)-\hat{f}(\mathbf{x})}{\mu}\mathbf{u}_i,\ \mathbf{u}_i = [\mathbf{Q}]_i$
  - GSG (Gaussian Smoothed Gradients): $\frac{1}{r}\sum_{i=1}^{r} \frac{\hat{f}(\mathbf{x}+\mu\mathbf{u}_i)-\hat{f}(\mathbf{x})}{\mu}\mathbf{u}_i,\ \mathbf{u}_i \sim \mathcal{N}(0, \mathbf{I}_d)$
  - cGSG (Centered GSG): $\frac{1}{r}\sum_{i=1}^{r} \frac{\hat{f}(\mathbf{x}+\mu\mathbf{u}_i)-\hat{f}(\mathbf{x}-\mu\mathbf{u}_i)}{2\mu}\mathbf{u}_i,\ \mathbf{u}_i \sim \mathcal{N}(0, \mathbf{I}_d)$
  - SSG (Sphere Smoothed Gradients): $\frac{d}{r}\sum_{i=1}^{r} \frac{\hat{f}(\mathbf{x}+\mu\mathbf{u}_i)-\hat{f}(\mathbf{x})}{\mu}\mathbf{u}_i,\ \mathbf{u}_i \sim \mathcal{U}(S^{d-1})$
  - cSSG (Centered SSG): $\frac{d}{r}\sum_{i=1}^{r} \frac{\hat{f}(\mathbf{x}+\mu\mathbf{u}_i)-\hat{f}(\mathbf{x}-\mu\mathbf{u}_i)}{2\mu}\mathbf{u}_i,\ \mathbf{u}_i \sim \mathcal{U}(S^{d-1})$

Which Gradient Estimation Works Better?

- Consider an unconstrained problem $\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$ [Berahas et al., FCM'21]:
  - Noisy ZO oracle: $\hat{f}(\mathbf{x}) = f(\mathbf{x}) + \epsilon(\mathbf{x})$
  - Noise ϵ is bounded uniformly: $|\epsilon(\mathbf{x})| \leq \epsilon_f$ (noise not neccessarily random)
  - $f(\mathbf{x}) \in C_L^{1,1}$ or $f(\mathbf{x}) \in C_M^{2,2}$ (twice continuously differentiable with M-Lipschitz Hessian)

Method	Number of calls r	$\|\nabla f(\mathbf{x})\|_2$
FFD	d	$\frac{2\sqrt{dL\epsilon_f}}{\theta}$
CFD	d	$\frac{2\sqrt{d}\sqrt[3]{M\epsilon_f^2}}{\sqrt[3]{6}\theta}$
LI	d	$\frac{2\|Q^{-1}\|\sqrt{dL\epsilon_f}}{\theta}$
GSG	$\frac{12d}{\sigma\theta^2} + \frac{d+20}{16\delta}$	$\frac{6d\sqrt{L\epsilon_f}}{\theta}$
cGSG	$\frac{12d}{\sigma\theta^2} + \frac{d+30}{144\delta}$	$\frac{12\sqrt[3]{d^{7/2}M\epsilon_f^2}}{\theta}$
SSG	$[\frac{8d}{\theta^2} + \frac{8d}{3\theta} + \frac{11d+104}{24}]\log\frac{d+1}{\delta}$	$\frac{4d\sqrt{L\epsilon_f}}{\theta}$
cSSG	$[\frac{8d}{\theta^2} + \frac{8d}{3\theta} + \frac{9d+192}{27}]\log\frac{d+1}{\delta}$	$\frac{4\sqrt[3]{d^{7/2}M\epsilon_f^2}}{\theta}$

- LI is essentially FFD with directions given as columns of a nonsingular matrix $\mathbf{Q}$
- For GSG, cGSG, SSG, and cSSG, results are w.p. $1-\delta$

Outline for Zeroth-Order Methods

- Representative Techniques for Random Directions of Gradient Estimations
- Representative Variance-Reduced Zeroth-Order Methods

Finite-Sum Minimization with VR Zeroth-Order Methods

- Consider ZO methods for special case of $\min f(\mathbf{x})$: finite-sum minimization

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x})$$

  - We have studied finite-sum minimization with VR first-order methods

- Need for solving finite-sum minimization problem with ZO methods:
  - Reinforcement learning (e.g., [Fazel et al., ICML'18])
  - Non-stationary online optimization problems [Zhang et al., arXiv:2010.07378]

- We have seen that SGD-type ZO methods with noisy $\hat{f}$ have sample complexity $O(d\epsilon^{-4})$ in the last lecture

Can we do better (at least for finite-sum minimization)?

Variance Reduction in First-Order Methods

- SAG
- SVRG
- SAGA
- SARAH
- SPIDER/SpiderBoost
- PAGE

We will develop their ZO counterparts

ZO-SVRG [Liu et al., NeurIPS'18]

- A zeroth-order version of SVRG
- Consider a non-convex finite-sum problem:

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x})$$

  - $f_i \in C_L^{1,1}$ ($\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\|_2 \leq L\|\mathbf{x}-\mathbf{y}\|_2$, $\forall \mathbf{x},\mathbf{y}\in\mathbb{R}^d$, $\forall i \in \{1,\ldots,N\}$)
  - Bounded variance of stochastic gradient: $\frac{1}{N}\sum_{i=1}^{N}\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|_2^2 \leq \sigma^2$
- The following gradient estimations are used in [Liu, et al., NeurIPS'18]:

$$\text{RandGradEst:}\quad \hat{\nabla} f_i(\mathbf{x}) = \frac{d}{\mu}[f_i(\mathbf{x}+\mu\mathbf{u}_i) - f_i(\mathbf{x})]\mathbf{u}_i$$

$$\text{Avg-RandGradEst:}\quad \hat{\nabla} f_i(\mathbf{x}) = \frac{d}{\mu q}\sum_{j=1}^{q}[f_i(\mathbf{x}+\mu\mathbf{u}_{i,j}) - f_i(\mathbf{x})]\mathbf{u}_{i,j}$$

$$\text{CoordGradEst:}\quad \hat{\nabla} f_i(\mathbf{x}) = \frac{1}{2\mu}\sum_{j=1}^{d}[f_i(\mathbf{x}+\mu_j\mathbf{e}_j) - f_i(\mathbf{x}-\mu_j\mathbf{e}_j)]\mathbf{e}_j$$

ZO-SVRG [Liu et al., NeurIPS'18]

The ZO-SVRG Algorithm

- **Required:** Step-sizes $\{\eta_s^t\}$, epoch length T, starting point $\mathbf{x}_0 \in \mathbb{R}^d$, smoothing parameter μ, number of iterations $K = S \cdot T$, $\phi_0 = \mathbf{x}_0^0$
- **for** $s = 0, 1, 2, \ldots, S-1$

  Compute ZO full gradient estimate $\hat{\nabla} f(\phi_s)$

  for $t = 0, 1, 2, \ldots, T-1$ **then**

  Uniformly randomly pick $I_t \subset \{1, \ldots, N\}$ with $|I_t| = B$ with replacement. Compute:

$$\mathbf{v}_s^t = \frac{1}{B} \sum_{i \in I_t} [\hat{\nabla} f_i(\mathbf{x}_s^t) - \hat{\nabla} f_i(\phi_s)] + \hat{\nabla} f(\phi_s)$$

$$\mathbf{x}_s^{t+1} = \mathbf{x}_s^t - \eta_s^t \mathbf{v}_s^t$$

  end for

  Let $\phi_{s+1} = \mathbf{x}_{s+1}^0 = \mathbf{x}_s^t$

end for

Output: $\mathbf{x}_\xi$, where ξ is picked uniformly at random from $\{0, \ldots, K-1\}$

ZO-SVRG [Liu et al., NeurIPS'18]

- Compared to FO-SVRG, the only difference is:

$$\text{FO-SVRG: } \mathbf{x}_s^{t+1} = \mathbf{x}_s^t - \eta_s^t \mathbf{v}_s^t, \ \mathbf{v}_s^t = \nabla f_{I_t}(\mathbf{x}_s^t) - \nabla f_{I_t}(\mathbf{x}_s^0) + \nabla f(\mathbf{x}_s^0)$$

$$\text{ZO-SVRG: } \mathbf{x}_s^{t+1} = \mathbf{x}_s^t - \eta_s^t \hat{\mathbf{v}}_s^t, \ \hat{\mathbf{v}}_s^t = \hat{\nabla} f_{I_t}(\mathbf{x}_s^t) - \hat{\nabla} f_{I_t}(\mathbf{x}_s^0) + \hat{\nabla} f(\mathbf{x}_s^0)$$

  where $\hat{\nabla} f_I(\mathbf{x}) = \frac{1}{B} \sum_{i \in I} \hat{\nabla} f_i(\mathbf{x})$

- Key Problem: $\hat{\nabla} f(\mathbf{x}_s^0)$ is no longer unbiased ZO gradient estimate
- Under stated assumptions, ZO-SVRG after $K = ST$ steps achieves:

$$\text{RandGradEst: } \mathbb{E}[\|\nabla f(\mathbf{x}_\xi)\|_2^2] = O\left(\frac{d}{T} + \frac{1}{B}\right)$$

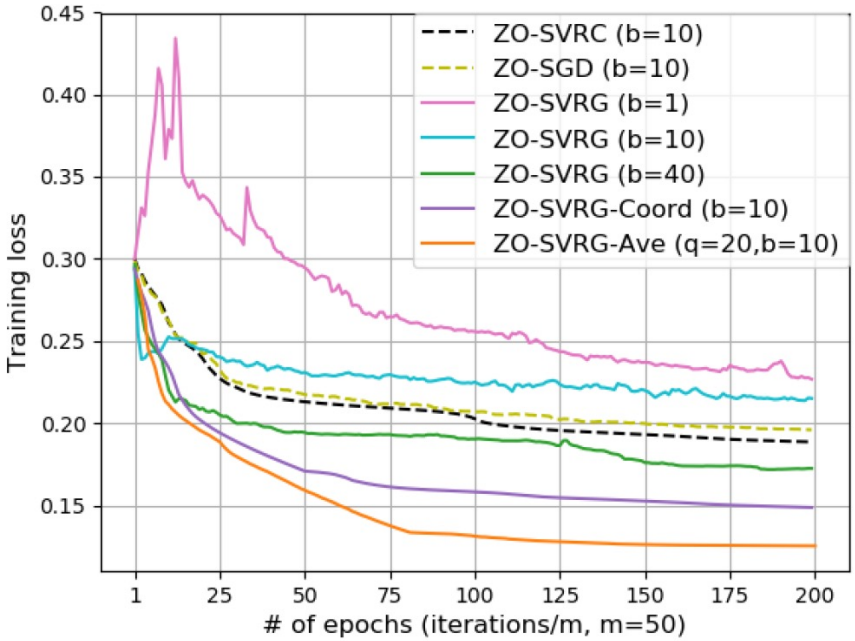
$$\text{Avg-RandGradEst: } \mathbb{E}[\|\nabla f(\mathbf{x}_\xi)\|_2^2] = O\left(\frac{d}{T} + \frac{1}{B \min\{d, q\}}\right)$$

$$\text{CoordGradEst: } \mathbb{E}[\|\nabla f(\mathbf{x}_\xi)\|_2^2] = O\left(\frac{d}{T}\right)$$

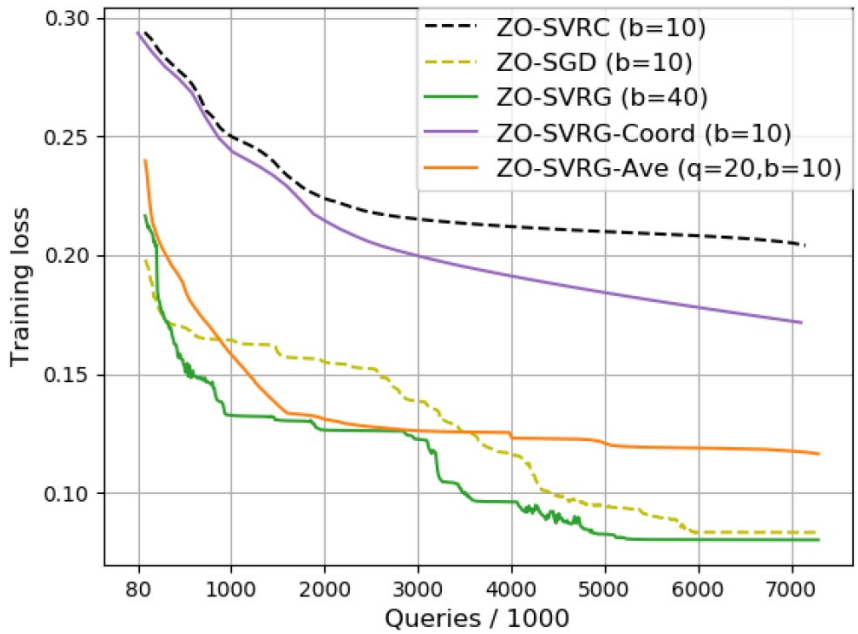
- Insight: CoordGradEst (i.e., deterministic gradient estimation) achieves same convergence rate as FO-SVRG

ZO-SVRG [Liu et al., NeurIPS'18]

- Blackbox classification problem motivated by material science:
  - A nonlinear least square problem $f_i(\mathbf{x}) = (y_i - \phi(\mathbf{x}; \mathbf{a}_i))^2$ for $i \in [N]$, where $\phi(\mathbf{x}, \mathbf{a}_i)$ is a blackbox function that only returns function value
  - $N = 1,000$ crystalline materials/compounds extracted from Open Quantum Materials Database; each compound has $d = 145$ chemical features



(a) Training loss versus iterations



(b) Training loss versus function queries

SpiderSZO [Fang et al., NeurIPS'18]

- **Required:** $n_0 = [1, \frac{30(2d+9)\sigma}{\epsilon}]$, Lipschitz constant L, epoch T, initial $\mathbf{x}_0 \in \mathbb{R}^d$, outer and inner batch-sizes B_1 and B_2, num. of iterations $K = ST$.
- **for** $k = 0, 1, 2, \ldots, K-1$

  if $\mod(k, T) = 0$ **then**

  Uniformly randomly pick $I_k \subset \{1, \ldots, N\}$ with $|I_k| = B_1$ with replacement. Compute:

$$\mathbf{v}_k = \sum_{j=1}^{d} \left(\frac{1}{B_1} \sum_{i \in I_k} \frac{[f_i(\mathbf{x}_k + \mu \mathbf{e}_j) - f_i(\mathbf{x}_k)]}{\mu} \right) \mathbf{e}_j$$

  else

  Create set of pairs $I_k = \{(i, \mathbf{u}_i)\}$ w/ $|I_k| = B_2$, where $i \sim \mathcal{U}[N]$ (with replacement) and indep. $\mathbf{u}_i \sim \mathcal{N}(0, \mathbf{I}_d)$. Compute:

$$\mathbf{v}_k = \frac{1}{B_2} \sum_{(i, \mathbf{u}_i) \in I_k} \left(\frac{f_i(\mathbf{x}_k + \mu \mathbf{u}_i) - f_i(\mathbf{x}_k)}{\mu} \mathbf{u}_i - \frac{f_i(\mathbf{x}_{k-1} + \mu \mathbf{u}_i) - f_i(\mathbf{x}_{k-1})}{\mu} \mathbf{u}_i \right) + \mathbf{v}_{k-1}$$

  end if

  Let $\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \mathbf{v}_k$, where $\eta_k = \min(\frac{\epsilon}{L n_0 \|\mathbf{v}_k\|}, \frac{1}{2Ln_0})$

end for

Output: $\mathbf{x}_\xi$, where ξ is picked uniformly at random from $\{0, \ldots, K-1\}$

SpiderSZO [Fang et al., NeurIPS'18]

- Learning rate $\eta_k = \min(\frac{\epsilon}{Ln_0\|\mathbf{v}_k\|}, \frac{1}{2Ln_0})$:
  - Follows from normalized gradient descent (NGD) [Nesterov, Book'04]
  - Inversely proportional to norm of "gradient"

Theorem 12 ([Fang et al., NeurIPS'18])

After $K = O(\epsilon^{-2})$ iterations, with $O(d\min\{N^{1/2}\epsilon^{-2}, \epsilon^{-3}\})$ incremental zeroth-order oracle (IZO, i.e., returning the value of $f_i(\mathbf{x})$ given $\mathbf{x}$ and i) calls, SpiderSZO ensures that:

$$\mathbb{E}[\|\nabla f(\mathbf{x}_\xi)\|_2] \leq 6\epsilon.$$

- This result is better than the sample complexity of [Nesterov and Spokoiny, FCM'17] by a factor of $N^{1/2}$

Improved ZO-SVRG and ZO-SPIDER [Ji et al., ICML'19]

- A tighter analysis for ZO-SVRG in [Ji et al., ICML'19]:
  - ZO-SVRG-Coord has a better convergence rate $\mathbb{E}[\|\nabla f(\mathbf{x}_\xi)\|_2^2] = O(1/K)$
  - d times better than the previous analysis in [Liu et al., NeurIPS'18]
  - To achieve an ϵ-stationary point (i.e., $\mathbb{E}[\|\nabla f(\mathbf{x}_\xi)\|_2^2] \leq \epsilon^2$), ZO-SVRG-Coord's function query complexity is $O(\min\{N^{2/3}d\epsilon^{-2}, d\epsilon^{-10/3}\})$

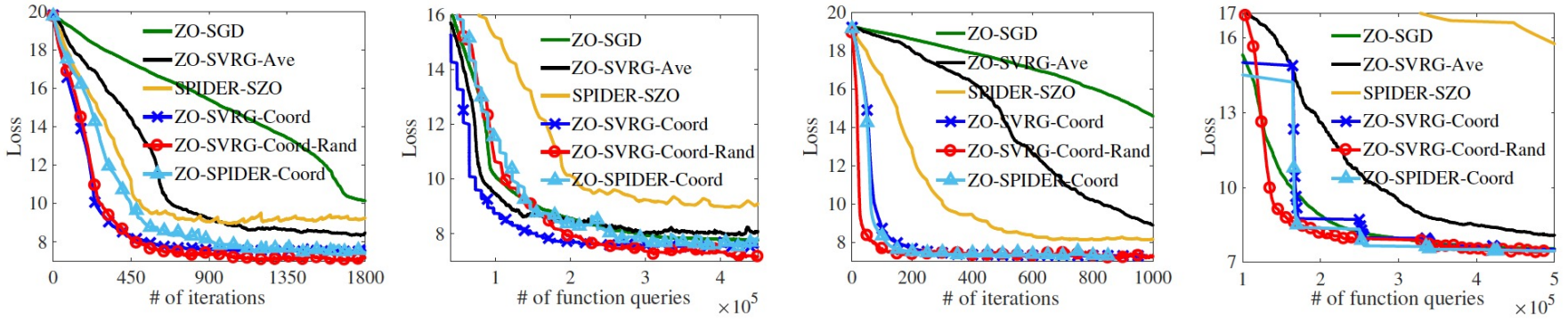
- Proof Sketch:
  1. Consider an intermediate variant of ZO-SVRG-Coord and ZO-SVRG-Ave called ZO-SVRG-Coord-Rand that uses CFD and SSG for the $\hat{\nabla} f(\phi_s)$ and $\hat{\nabla} f_i(\mathbf{x}_s^t) - \hat{\nabla} f_i(\phi_s)$ parts in $\mathbf{v}_s^t = \frac{1}{B}\sum_{i\in I_t}[\hat{\nabla} f_i(\mathbf{x}_s^t) - \hat{\nabla} f_i(\phi_s)] + \hat{\nabla} f(\phi_s)$, respectively, as opposed to [Liu et al., NeurIPS'18] that used only one type of gradient estimation at once.
  2. [Ji et al., ICML'19] showed that, although the replacement of SSG with CFD requires d more oracle calls, it achieves more accurate gradient estimation, which yields a convergence rate $\mathbb{E}[\|\nabla f(\mathbf{x}_\xi)\|_2^2] = O(1/K)$. So, the convergence rate stays the same for ZO-SVRG-Coord.

Improved ZO-SVRG and ZO-SPIDER [Ji et al., ICML'19]

- A new variant of ZO-SPIDER in [Ji et al., ICML'19]: ZO-SPIDER-Coord:
  - Similar to ZO-SVRG-Coord: Use CFD instead of GSG in SpiderSZO
  - Show that ZO-SPIDER-Coord has the same convergence rate as SpiderSZO, but with a bigger size-size $\eta_k = 1/4L$ and doesn't depend on ϵ (using similar idea as in SpiderBoost)
  - With appropriate choices of learning rate, sampling radius parameters, outer batch size, ZO-SPIDER-Coord achieves a convergence rate $O(\sqrt{B_1}/K)$
  - To achieve an ϵ-stationary point (i.e., $\mathbb{E}[\|\nabla f(\mathbf{x}_\xi)\|_2^2] \leq \epsilon^2$), ZO-SVRG-Coord's function query complexity is $O(\min\{N^{1/2}d\epsilon^{-2}, d\epsilon^{-3}\})$

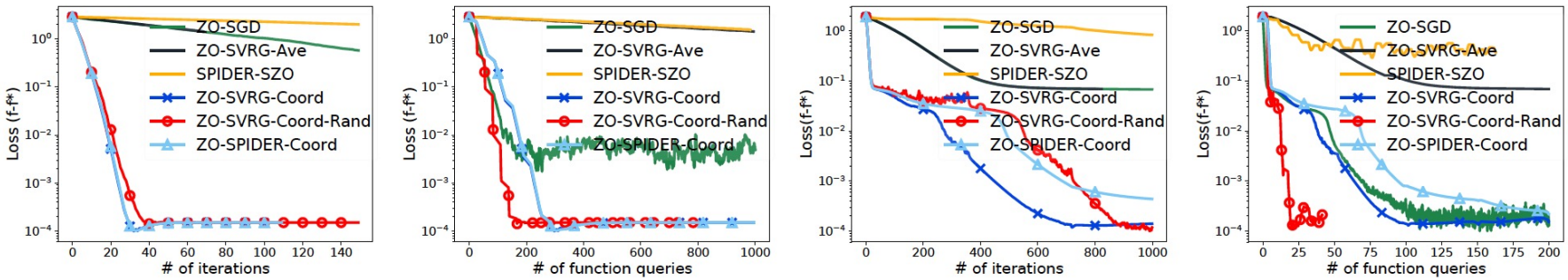
Improved ZO-SVRG and ZO-SPIDER [Ji et al., ICML'19]

- Numerical result comparisons:
  - Generation of black-box adversarial examples (DNN for MNIST handwritten digit classification, use the blackbox attacking loss in [Liu et al. NeurIPS'18])
  - Nonconvex logistic regression on LIBSVM [Chang and Lin, ACM TIST'11]



(a) Loss vs iterations: $n = 10$ (b) Loss vs queries: $n = 10$ (c) Loss vs iterations: $n = 100$ (d) Loss vs queries: $n = 100$

Figure 1. Comparison of different zeroth-order algorithms for generating black-box adversarial examples for digit "1" class



(a) Loss vs iterations: german (b) Loss vs queries: german (c) Loss vs iterations: ijcnn1 (d) Loss vs queries: ijcnn1

Figure 2. Comparison of different zeroth-order algorithms for logistic regression problem with a nonconvex regularizer

Part V

First-Order Optimization with Special Geometric Structure

Outline

- The Polyak-Łojasiewicz (PL) Condition and Convergence of Various Methods under the PL Condition
- The PL Condition and the Over-parameterized Regime
- Star-Convexity and α-Weak-Quasi-Convexity

Convergence Results of Methods We Learned Thus Far

- First-order and zeroth-order methods for nonconvex optimization in learning:
  - GD/SGD-style algorithms
  - Only focus on stationarity gap
  - Typically sublinear convergence rates: $O(1/K)$, $O(1/\sqrt{K})$, ... ($O(1/K^2)$ is order-optimal)
- Meanwhile, it's well-known from convex optimization that:
  - GD achieves linear convergence rate under strong convexity
  - Convergence of global optimality

Can global linear convergence to optimality happen under weaker conditions?

The Polyak-Łojasiewicz Condition

Definition 13 ([Polyak, '63], [Łojasiewicz, '63])

A function $f(\mathbf{x})$ is said to satisfy the Polyak-Łojasiewicz (PL) condition if for all $\mathbf{x} \in \mathbb{R}^d$, there exists a constant $\mu > 0$ such that:

$$2\mu(f(\mathbf{x}) - f(\mathbf{x}^*)) \leq \|\nabla f(\mathbf{x})\|_2^2.$$

Remarks

- Aka "gradient dominated" condition (e.g., [Reddi et al., ICML'16])
- Implies any stationary point is a global min, although not necessarily unique
- Automatically holds for strongly convex functions
- Many nonconvex functions satisfy PL condition, especially in the over-parameterized regime
- PL condition means that the optimality gap $f(\mathbf{x}) - f^*$ is upper bounded by a quadratic function of the stationarity gap

Nice Features of the PL Condition

- Ease of verification compared to strong convexity (SC):
  - One only needs to access $\|\nabla f(\mathbf{x})\|$ and $f(\mathbf{x})$. In comparison, SC requires checking PD of the Hessian matrix $\mathbf{H}$ (accurate estimation of $\lambda_{\min}(\mathbf{H})$)
- Robustness of the condition
  - $\|\nabla f(\mathbf{x})\|$ is more resilient to perturbation of the obj function than $\lambda_{\min}(\mathbf{H})$
- Allows multiple global minima:
  - Modern ML problems are over-parameterized and have manifolds of global minima, not compatible with SC in general but compatible with PL
- Invariance under transformation:
  - PL is invariant under a broad class of nonlinear coordinate transformations arising from feature extraction/transformation of many ML applications
- PL on manifolds:
  - PL allows for efficient optimization on manifolds, while being invariant under the choice of coordinates (see [Weber and Sra, arXiv:1710:10770]
- Linear convergence of GD and SGD:
  - PL is sufficient not only for GD but also for SGD

Gradient Descent under the PL Condition

Theorem 14 (Linear Convergence Rate for GD)

Consider the unconstrained optimization problem $\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$, *where* f *has an* L*-Lipschitz continuous gradient, a non-empty solution set* $\mathcal{X}^*$, *and satisfies the PL condition. Then, the gradient descent method with a step-size of* $1/L$, *i.e.,* $\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{1}{L}\nabla f(\mathbf{x}_k)$, *has a global linear convergence rate:*

$$f(\mathbf{x}_k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k (f(\mathbf{x}_0) - f^*).$$

Remarks

- For twice differentiable functions, L-smoothness means eigenvalues of $\nabla^2 f(\mathbf{x})$ are bounded from above by L (curvature upper bound)

Stochastic Gradient Descent under the PL Condition

- The finite-sum minimization problem: $\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x})$
- Consider the SGD method that uses the iteration: $\mathbf{x}_{k+1} = \mathbf{x}_k - s_k \nabla_{i_k} f(\mathbf{x}_k)$

Theorem 15 (Convergence Rate for SGD)

Assume that f has L-Lipschitz continuous gradients and a non-empty solution set $\mathcal{X}^*$, and it satisfies the PL condition, and f satisfies $\|\nabla f_{i_k}(\mathbf{x}_k)\| \leq C^2$ for all $\mathbf{x}_k$ and some constant $C > 0$. Then, it holds that:

- *SGD with diminishing step-size $s_k = \frac{2k+1}{2\mu(k+1)^2}$ has a convergence rate of:*

$$\mathbb{E}[f(\mathbf{x}_k) - f^*] \leq \frac{LC^2}{2\mu^2 k}.$$

- *SGD with constant step-size $s_k = s \leq \frac{1}{2\mu}$ has a convergence rate of:*

$$\mathbb{E}[f(\mathbf{x}_k) - f^*] \leq (1 - 2s\mu)^k [f(\mathbf{x}_0) - f^*] + \frac{LC^2 s}{4\mu}.$$

SGD under PL Condition in Over-parameterized Regime

- Consider ERM in over-parameterized regime: $\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^N f_i(\mathbf{x})$
  - $f(\mathbf{x})$ is L-smooth: $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x}-\mathbf{y}\|$, $\forall \mathbf{x}, \mathbf{y}$
  - $f_i(\mathbf{x})$ satisfies: $\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\| \leq \tilde{L}|f_i(\mathbf{x}) - f_i(\mathbf{y})|$ for some $\tilde{L} > 0$
  - In ML problems, w.l.o.g., we can assume that $\inf_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) = 0$ and so the PL condition can be modified as μ-PL*: $2\mu f(\mathbf{x}) \leq \|\nabla f(\mathbf{x})\|_2^2$

- Over-parameterized regime: $d \gg N$
  - The interpolation effect: for every sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots$ such that $\lim_{k\to\infty} f(\mathbf{x}_k) = 0$, we have

$$\lim_{k\to\infty} f_i(\mathbf{x}_k) = 0, \quad 1 \leq i \leq N.$$

  - Meaning: In the over-parameterized regime, the richness of the model is so high such that fit all training samples

SGD under PL Condition in Over-parameterized Regime

- Consider the general mini-batched version of SGD with constant step-size s:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{s}{B} \sum_{j=1}^{B} \nabla f_{i_k^j}(\mathbf{x}_k),$$

  - B: the mini-batch size; the sample indices $\{i_k^1, \ldots, i_k^B\}$ in the mini-batch are drawn uniformly with replacement in each iteration k from $\{1, \ldots, N\}$

Theorem 16 ([Bassily et al., arXiv:1811.02564])

Consider the mini-batch SGD with smooth losses as stated. Suppose the interpolation condition holds. Suppose that the ERM function $f(\mathbf{x})$ is μ-PL* for some $\mu > 0$. For any mini-batch size $B \in \mathbb{N}$, the mini-batch SGD with constant step-size $s^*(B) \triangleq \frac{2\mu B}{L(\tilde{L}+L(B-1))}$ guarantees that:

$$\mathbb{E}[f(\mathbf{x}_k)] \leq \left(1 - \mu s^*(B)\right)^k f(\mathbf{x}_0)$$

Other Methods under the PL Condition

Similar linear convergence rate results can be shown for other methods under the μ-PL, L-smoothness, and uniform variance bound conditions, which implies the following sample complexity results:

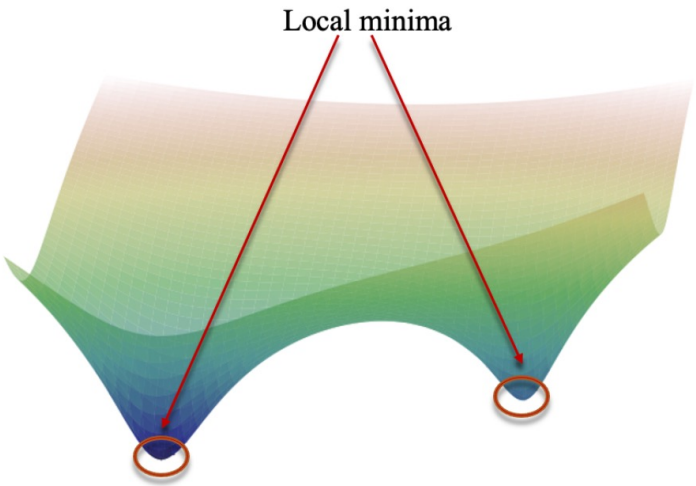
- GD [Polyak, '63]: $\frac{L}{\mu}\log\frac{\Delta_0}{\epsilon}$
- SGD [Karimi et al., ECML-KDD'16]: $\frac{L}{\mu}(\frac{\max_i L_i}{\mu}\log(\frac{\Delta_0}{\epsilon})+\frac{\max_i L_i\Delta_*}{\mu\epsilon})$
- SVRG [Reddi et al., NeurIPS'16]: $(N+\frac{N^{2/3}\max_i L_i}{\mu})\log(\frac{\Delta_0}{\epsilon})$
- SAGA [Reddi et al., NeurIPS'16]: $(N+\frac{N^{2/3}\max_i L_i}{\mu})\log(\frac{\Delta_0}{\epsilon})$
- PAGE [Li et al., ICML'21]: $(b+\sqrt{b}\frac{L_{\text{avg}}}{\mu})\log(\frac{\Delta_0}{\epsilon})$, where $b=\min\{\frac{\sigma^2}{\mu\epsilon},N\}$

Outline

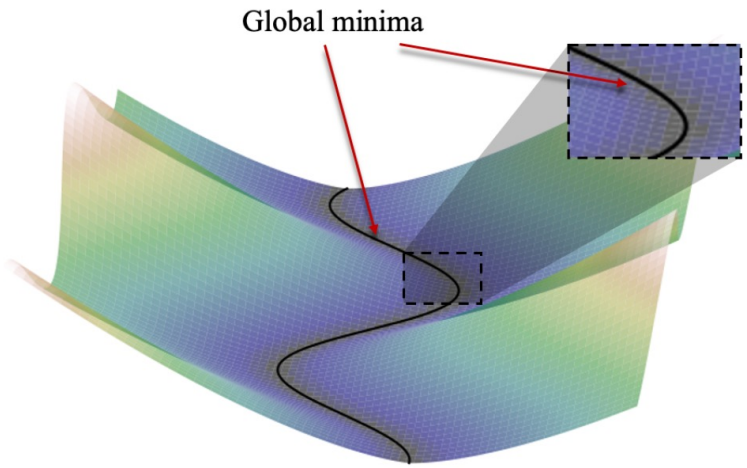
- The Polyak-Łojasiewicz (PL) Condition and Convergence of Various Methods under the PL Condition
- The PL Condition and the Over-parameterized Regime
- Star-Convexity and α-Weak-Quasi-Convexity

PL Condition and Over-parameterized Regime

- Landscape of under-parameterized and over-parameterized models (figure from [Liu et al., arXiv:2003:00307]



(a) Loss landscape of under-parameterized models



(b) Loss landscape of over-parameterized models

- Key Insight:
  - Convexity is not the right framework for analyzing the loss landscape of over-parameterized systems, even locally
  - Instead, the μ-PL* condition (i.e., $\|\nabla f(\mathbf{w})\|_2^2 \geq 2\mu f(\mathbf{w})$, $\forall \mathbf{w}$) is a more appropriate framework

PL Condition and Over-parameterized Regime

The essence of supervised learning:

- Given a dataset of size N, $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, $\mathbf{x}_i \in \mathbb{R}^d$, $y \in \mathbb{R}$
- A parametric family of models $f(\mathbf{w}, \mathbf{x})$ (e.g., a neural network)
- Goal: To find a model with parameter $\mathbf{w}^*$ that fits the training data:

$$f(\mathbf{w}^*, \mathbf{x}_i) \approx y_i, \quad i = 1, 2, \ldots, N$$

- Mathematically: Equivalent to solving (exactly or approximately) a system of N nonlinear equations:

$$\mathcal{F}(\mathbf{w}) = \mathbf{y},$$

where $\mathbf{w} \in \mathbb{R}^d$, $\mathbf{y} \in \mathbb{R}^N$, and $\mathcal{F}(\cdot) : \mathbb{R}^d \to \mathbb{R}^N$ with $(\mathcal{F}(\mathbf{w}))_i = f(\mathbf{w}, \mathbf{x}_i)$.

- The system of equations is solved by minimizing a certain loss function $\mathcal{L}(\mathbf{w})$
  - E.g., the square loss: $\mathcal{L}(\mathbf{w}) = \frac{1}{2}\|\mathcal{F}(\mathbf{w}) - \mathbf{y}\|^2 = \frac{1}{2}\sum_{i=1}^N (f(\mathbf{w}, \mathbf{x}_i) - y_i)^2$

PL Condition and Over-parameterized Regime

μ-PL* condition emerges through the spectrum of the tangent kernel

- Let $D\mathcal{F}(\mathbf{w}) \in \mathbb{R}^{N \times d}$ be the differential of the mapping $\mathcal{F}$ at $\mathbf{w}$
- The tangent kernel of $\mathcal{F}$ is defined as an $N \times N$ matrix:

$$\mathbf{K}(\mathbf{w}) \triangleq D\mathcal{F}(\mathbf{w}) D\mathcal{F}^\top(\mathbf{w})$$

  - It follows from the definition that $\mathbf{K}(\mathbf{w})$ is PSD

- The square loss $\mathcal{L}$ is μ-PL* at $\mathbf{w}$ [Liu, et al., arXiv:2003:00307], where

$$\mu = \lambda_{\min}(\mathbf{K}(\mathbf{w})),$$

  is the smallest eigenvalue of the kernel matrix

Thus, the PL* condition is inherently tied to the spectrum of the tangent kernel matrix associated with $\mathcal{F}$

PL Condition and Over-parameterized Regime

Wide (hence over-parameterized) neural networks satisfy PL* condition:

- A powerful tool: the neural tangent kernel (NTK)
  - First appeared in a landmark paper [Jacot et al., NeurIPS'18]
  - Tangent kernel of a single-layer wide neural networks with linear output layer ($f(\mathbf{x}) = \sum_{i=1}^{d} \sigma(\mathbf{w}^\top \mathbf{x})$) are nearly constant in a ball $\mathcal{B}$ of a certain radius around the ball with a random center (note: d is also the width of the NN):

$$\|\mathbf{H}_{\mathcal{F}}(\mathbf{w})\| = O^*(1/\sqrt{d}),$$

    where $\mathbf{H}_{\mathcal{F}}(\mathbf{w})$ is a $N \times d \times d$ tensor with $(\mathbf{H}_{\mathcal{F}})_{ijk} = \frac{\partial^2 \mathcal{F}_i}{\partial w_j \partial w_k}$
  - Constancy of NTK implies training dynamic of wide NNs is approximately a linear model $\Rightarrow$ linear convergence of gradient flow (hence GD)
  - It can be shown that [Liu, et al., arXiv:2003:00307]:

$$|\lambda_{\min}(\mathbf{K}(\mathbf{w})) - \lambda_{\min}(\mathbf{K}(\mathbf{w}_0))| < O\left(\sup_{\mathbf{w}\in\mathcal{B}} \|\mathbf{H}_{\mathcal{F}}(\mathbf{w})\|\right) = O(1/\sqrt{d})$$

Thus, the PL* condition holds for single-layer wide NN

Outline

- The Polyak-Łojasiewicz (PL) Condition and Convergence of Various Methods under the PL Condition
- The PL Condition and the Over-parameterized Regime
- Star-Convexity and α-Weak-Quasi-Convexity

Star-Convex Function

Definition 17 ([Nesterov and Polyak, Math Prog'06])

A function $f(\mathbf{x})$ is called star-convex if for some global minimizer $\mathbf{x}^*$ and for all $\lambda \in [0,1]$ and $\mathbf{x} \in \mathbb{R}^d$

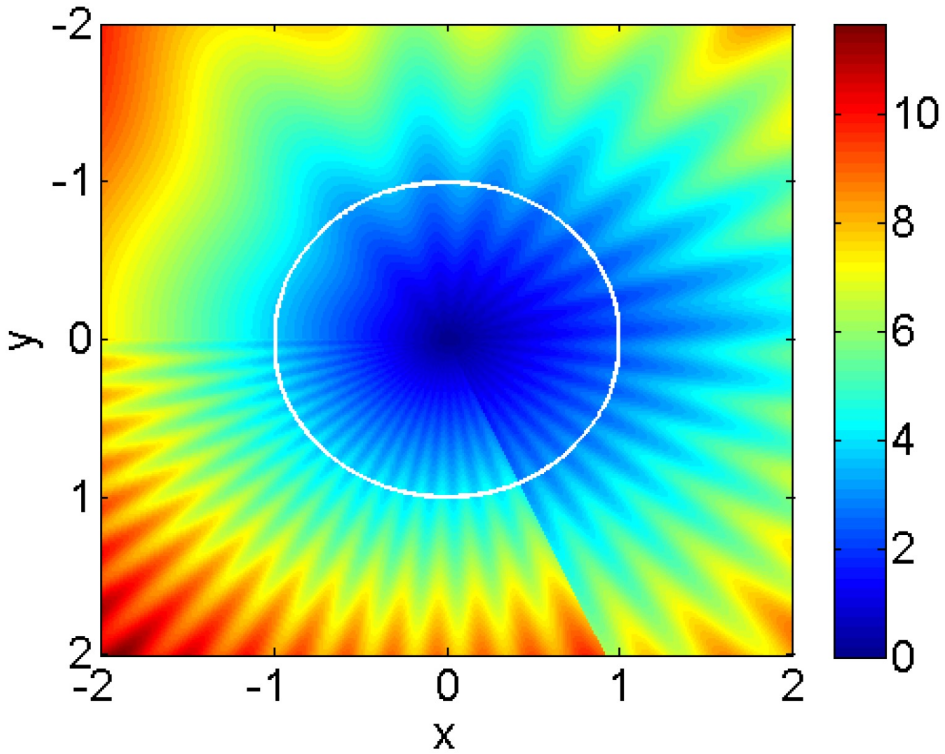
$$f(\lambda \mathbf{x} + (1-\lambda)\mathbf{x}^*) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{x}^*)$$

Remarks

- Any interval connecting some point $\mathbf{x}$ and some global minimizer $\mathbf{x}^*$ lies not lower than the graph
- Considerably weaker than convexity
- For example, $|x|(1 - e^{-|x|})$ is a nonconvex star-convex function.

An Example of Star-Convex Landscape

- Intuitively, if we visualize the objective function as a landscape, star-convexity means that the global optimum is "visible" from every point (i.e., "no blocking ridges", figure from [Lee and Valiant, FOCS'16])



Optimal First-Order Algorithms under Star-Convexity

- AGMsDR [Nesterov et al., arXiv:1809.05895]
  - Accelerated Gradient Method with Small-Dimensional Relaxation (AGMsDR)
  - For star-convex L-smooth functions, AGMsDR achieves

$$\min_{k=\lceil T/2 \rceil, \ldots, T} \|\nabla f(\mathbf{y}_k)\|_*^2 \leq \frac{64 L^2 \Delta_0}{T^3},$$

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq \frac{4 L \Delta_0}{T^2}$$

α-Weak-Quasi-Convex Function

A more general class of functions:

Definition 18

A function $f(\mathbf{x})$ is called α-weakly-quasi-convex function if for some global minimizer $\mathbf{x}^*$, some some $\alpha \in (0, 1]$, and $\mathbf{x} \in \mathbb{R}^d$, $f(\mathbf{x})$ satisfies

$$f(\mathbf{x}) - f(\mathbf{x}^*) \leq \frac{1}{\alpha} \langle \nabla f(\mathbf{x}), \mathbf{x} - \mathbf{x}^* \rangle$$

Remarks

- Continuously differentiable 1-weakly-quasi-convex functions are exactly the star-convex functions [Guminov et al., arXiv:1710.00797]

Optimal FO Methods under α-Weak-Quasi-Convexity

Iteration complexity:

- AGMsDR [Nesterov et al., arXiv:1809.05895]: $O(\alpha^{-3/2}L^{1/2}\Delta_0\epsilon^{-1/2})$
  - AGMsDR requires exact line search
- SESOP [Guminov et al., arXiv:1710.00797]: $O(\alpha^{-1}L^{1/2}\Delta_0\epsilon^{-1/2})$
  - SESOP requires exact line search
- GAGD [Hinder et al., COLT'20]: $O(\alpha^{-1}L^{1/2}\Delta_0\epsilon^{-1/2})$
  - GAGD only requires simple backtracking and binary line search
  - Also provided iteration complexity lower bound, thus proving GAGD being order-optimal in terms of iteration complexity

(α, μ)-Strongly Quasi-Convex Function

A more general class of functions:

Definition 19

A function $f(\mathbf{x})$ is called (α, μ)-strongly-quasi-convex function if for some global minimizer $\mathbf{x}^*$, some some $\alpha \in (0, 1]$, $\mu > 0$, and $\mathbf{x} \in \mathbb{R}^d$, $f(\mathbf{x})$ satisfies

$$f(\mathbf{x}) - f(\mathbf{x}^*) \leq \frac{1}{\alpha}\langle \nabla f(\mathbf{x}), \mathbf{x} - \mathbf{x}^* \rangle - \frac{\mu}{2}\|\mathbf{x} - \mathbf{x}^*\|^2$$

Iteration complexity:

- GAGD [Hinder et al., COLT'20]: $O(\alpha^{-1} L^{1/2} \Delta_0 \log(\alpha^{-1}\epsilon^{-1}))$

Stochastic Methods under α-Weak-Quasi-Convexity

- SGD [Gower et al., AISTATS'21]: finite-sum minimization:
  - Sample complexity bound under "expected residual" assumption: $\mathbb{E}[\|g(\mathbf{x}) - g(\mathbf{x}^*) - (\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}^*))\|^2] \leq 2\rho(f(\mathbf{x}) - f(\mathbf{x}^*))$ for some $\rho > 0$:

$$O\left(\frac{(\rho + L)\Delta_0^2}{\alpha^2\epsilon} + \frac{\sigma_*^2\Delta_0^2}{\alpha^2\epsilon^2}\right)$$

  - Under interpolation condition and with Polyak step-size:

$$O\left(\frac{\bar{L}\Delta_0^2}{\alpha^2\epsilon}\right)$$

    - ⋆ $\bar{L}$ is the expected smoothness constant
    - ⋆ In full batch case (i.e., $g(\mathbf{x}) = \nabla f(\mathbf{x})$), we have $\bar{L} = L$
    - ⋆ in importance sampling case (i.e., $g(\mathbf{x}) = \nabla f_j(\mathbf{x})$ where $j = i$ with prob. $L_i / \sum_{k=1}^N L_k$), we have $\bar{L} = \frac{1}{N}\sum_{i=1}^N L_i$

Thank You!