Optimization Methods

GU 4241/GR 5241

Statistical Machine Learning

Xiaofei Shi

Optimization problems

Optimization problems underlie nearly everything we do in Machine Learning and Statistics. In many courses, you learn how to:

translate into $P \; : \; \min_{x \in D} \; f(x)$

Conceptual idea *Optimization problem*

Examples of this? Examples of the contrary?

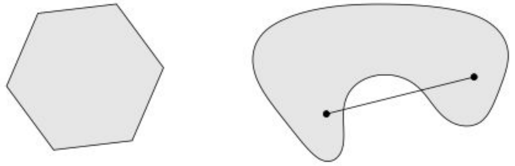
Optimization problems

● lasso , ridge regression	● MSE + L_p penalty with p>=1
● Mixture of Gaussian	● -Likelihood or -log likelihood
● ……	
● PCA	
● Basis pursuit	● MSE + L_0 penalty

Convex sets and functions

Convex set: $C \subseteq \mathbb{R}^n$ such that

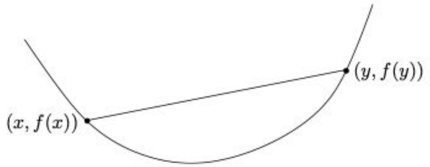
$$x, y \in C \implies tx + (1-t)y \in C \text{ for all } 0 \le t \le 1$$



Convex function: $f : \mathbb{R}^n \to \mathbb{R}$ such that $\mathrm{dom}(f) \subseteq \mathbb{R}^n$ convex, and

$$f(tx + (1-t)y) \le tf(x) + (1-t)f(y) \text{ for all } 0 \le t \le 1$$

and all $x, y \in \mathrm{dom}(f)$



Convex optimization problems

Optimization problem:

$$\begin{aligned} \min_{x \in D} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \; i = 1, \ldots m \\ & h_j(x) = 0, \; j = 1, \ldots r \end{aligned}$$

Here $D = \text{dom}(f) \cap \bigcap_{i=1}^{m} \text{dom}(g_i) \cap \bigcap_{j=1}^{p} \text{dom}(h_j)$, common domain of all the functions

This is a convex optimization problem provided the functions f and $g_i, i = 1, \ldots m$ are convex, and $h_j, j = 1, \ldots p$ are affine:

$$h_j(x) = a_j^T x + b_j, \quad j = 1, \ldots p$$

Local minimum

For convex optimization problems, local minima are global minima

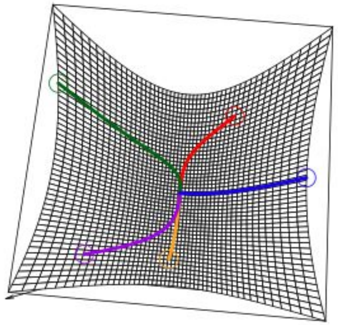
Formally, if x is feasible—$x \in D$, and satisfies all constraints—and minimizes f in a local neighborhood,

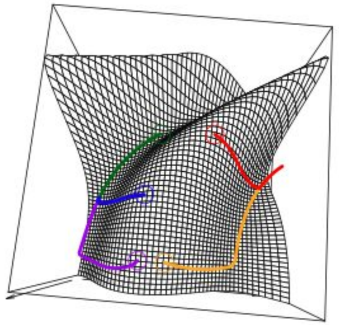
$$f(x) \leq f(y) \ \text{ for all feasible } y, \ \|x - y\|_2 \leq \rho,$$

then

$$f(x) \leq f(y) \ \text{ for all feasible } y$$

This is a very useful fact and will save us a lot of trouble!


Convex


Nonconvex

Linear program

A linear program or LP is an optimization problem of the form

$$\begin{aligned} \min_x \quad & c^T x \\ \text{subject to} \quad & Dx \leq d \\ & Ax = b \end{aligned}$$

Observe that this is always a convex optimization problem

- First introduced by Kantorovich in the late 1930s and Dantzig in the 1940s
- Dantzig's simplex algorithm gives a direct (noniterative) solver for LPs (later in the course we'll see interior point methods)
- Fundamental problem in convex optimization. Many diverse applications, rich history

Linear program: diet selection

Find cheapest combination of foods that satisfies some nutritional requirements (useful for graduate students!)

$$\begin{aligned} \min_x \quad & c^T x \\ \text{subject to} \quad & Dx \geq d \\ & x \geq 0 \end{aligned}$$

Interpretation:

- c_j : per-unit cost of food j
- d_i : minimum required intake of nutrient i
- D_{ij} : content of nutrient i per unit of food j
- x_j : units of food j in the diet

Quadratic program

A convex quadratic program or QP is an optimization problem of the form

$$\begin{aligned} \min_{x} \quad & c^T x + \frac{1}{2} x^T Q x \\ \text{subject to} \quad & Dx \leq d \\ & Ax = b \end{aligned}$$

where $Q \succeq 0$, i.e., positive semidefinite

Note that this problem is not convex when $Q \not\succeq 0$

From now on, when we say quadratic program or QP, we implicitly assume that $Q \succeq 0$ (so the problem is convex)

COLUMBIA UNIVERSITY

Quadratic program: portfolio selection

Construct a financial portfolio, trading off performance and risk:

$$\begin{aligned} \max_{x} \quad & \mu^T x - \frac{\gamma}{2} x^T Q x \\ \text{subject to} \quad & 1^T x = 1 \\ & x \geq 0 \end{aligned}$$

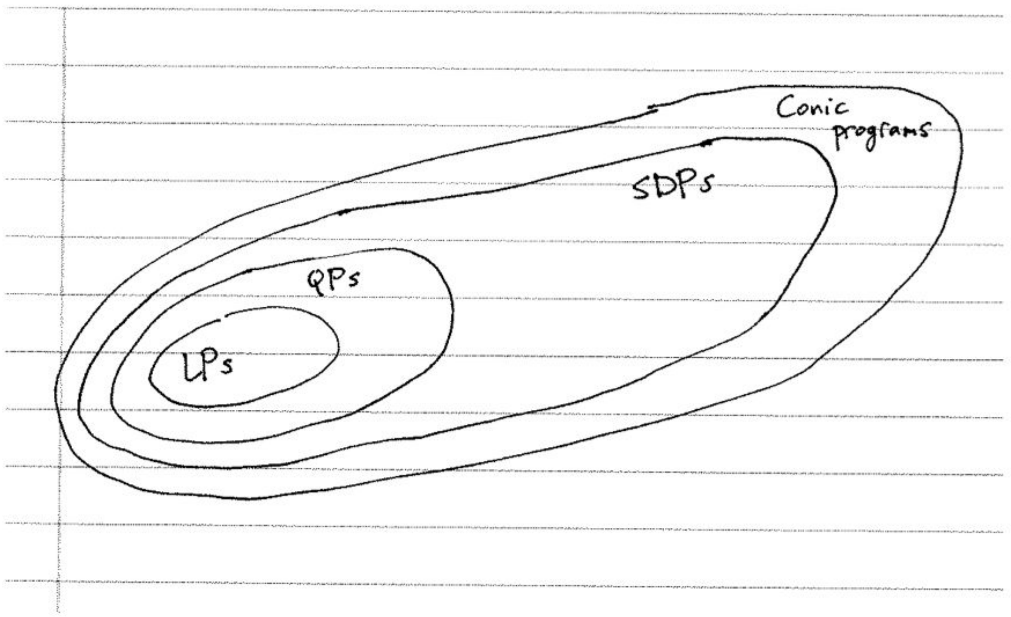
Interpretation:

- μ : expected assets' returns
- Q : covariance matrix of assets' returns
- γ : risk aversion
- x : portfolio holdings (percentages)

COLUMBIA UNIVERSITY

Standard formulation

- Linear programs
- Quadratic programs
- Semidefinite programs
- Conic programs



Gradient descent

Consider unconstrained, smooth convex optimization

$$\min_x \ f(x)$$

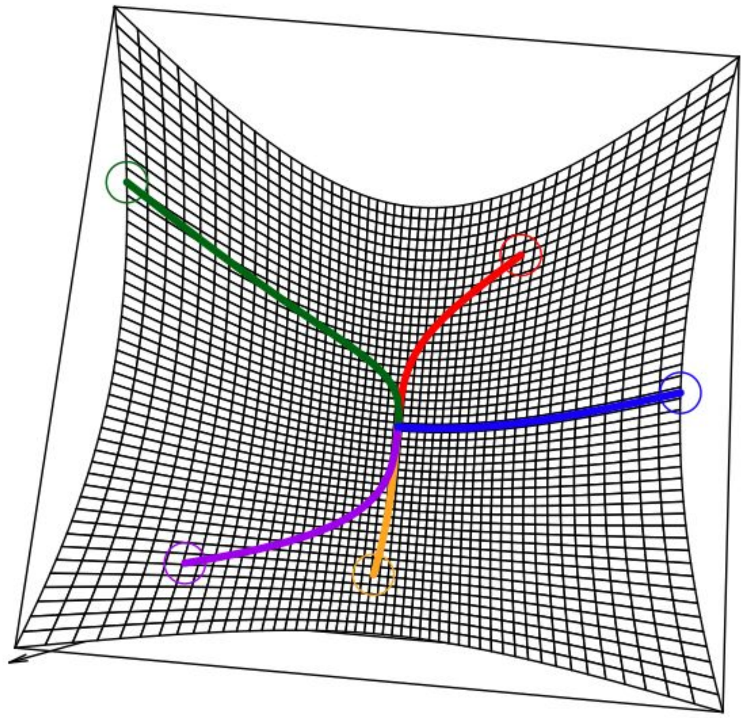
i.e., f is convex and differentiable with $\mathrm{dom}(f) = \mathbb{R}^n$. Denote the optimal criterion value by $f^\star = \min_x \ f(x)$, and a solution by $x^\star$

Gradient descent: choose initial point $x^{(0)} \in \mathbb{R}^n$, repeat:

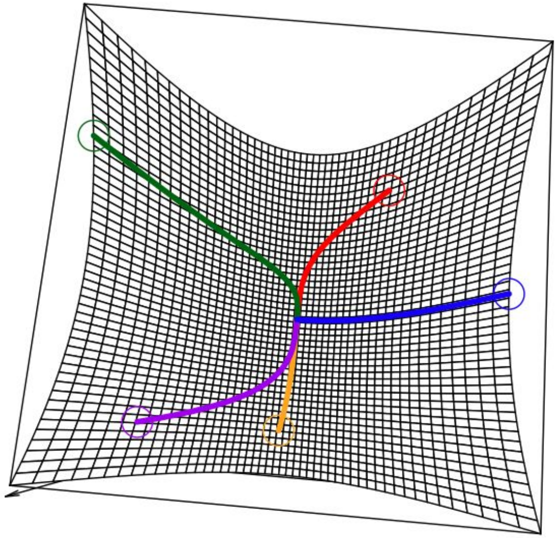
$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

Stop at some point

Gradient descent



Gradient descent



At each iteration, consider the expansion

$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2t}\|y - x\|_2^2$$

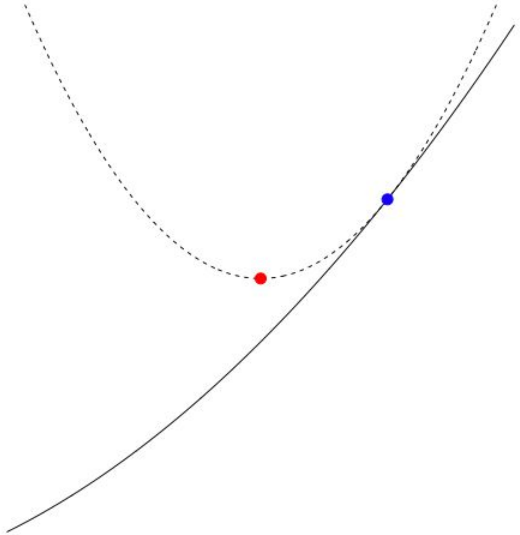
Quadratic approximation, replacing usual Hessian $\nabla^2 f(x)$ by $\frac{1}{t}I$

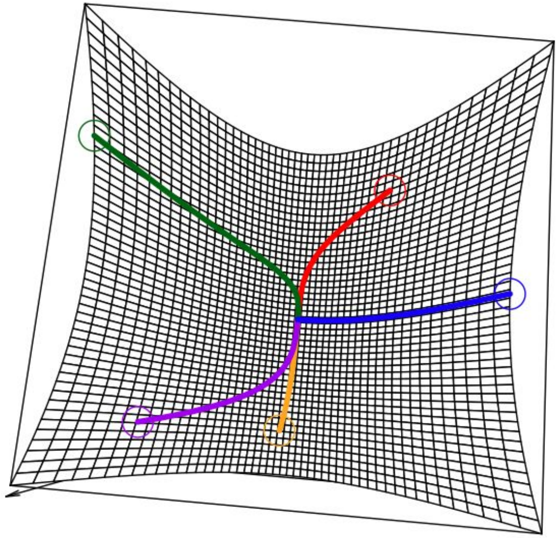
$f(x) + \nabla f(x)^T (y - x)$	linear approximation to f
$\frac{1}{2t}\|y - x\|_2^2$	proximity term to x, with weight $1/(2t)$

Choose next point $y = x^+$ to minimize quadratic approximation:

$$x^+ = x - t\nabla f(x)$$

Gradient descent



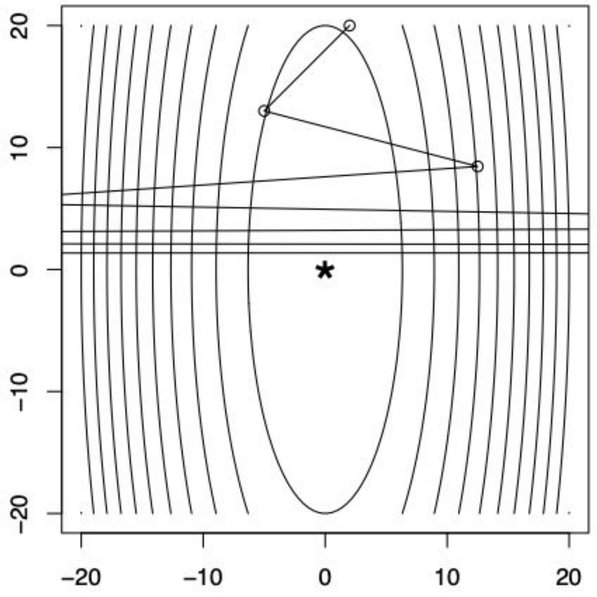


Blue point is x, red point is

$$x^+ = \underset{y}{\operatorname{argmin}} \; f(x) + \nabla f(x)^T (y - x) + \frac{1}{2t}\|y - x\|_2^2$$

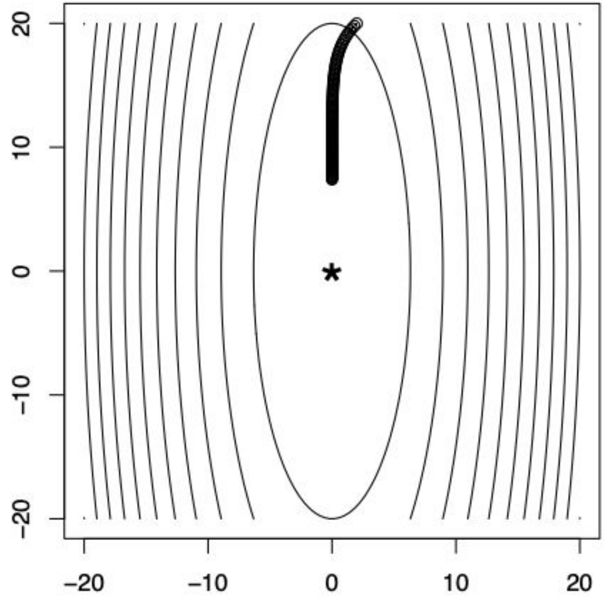
Gradient descent: fixed step size

Simply take $t_k = t$ for all $k = 1, 2, 3, \ldots$, can diverge if t is too big. Consider $f(x) = (10x_1^2 + x_2^2)/2$, gradient descent after 8 steps:



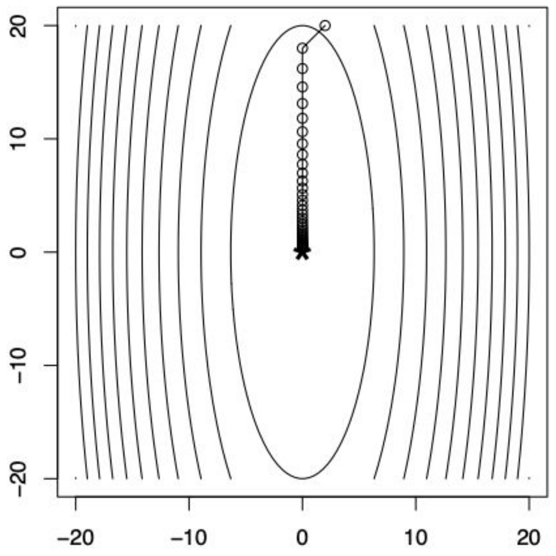
Gradient descent: fixed step size

Can be slow if t is too small. Same example, gradient descent after 100 steps:



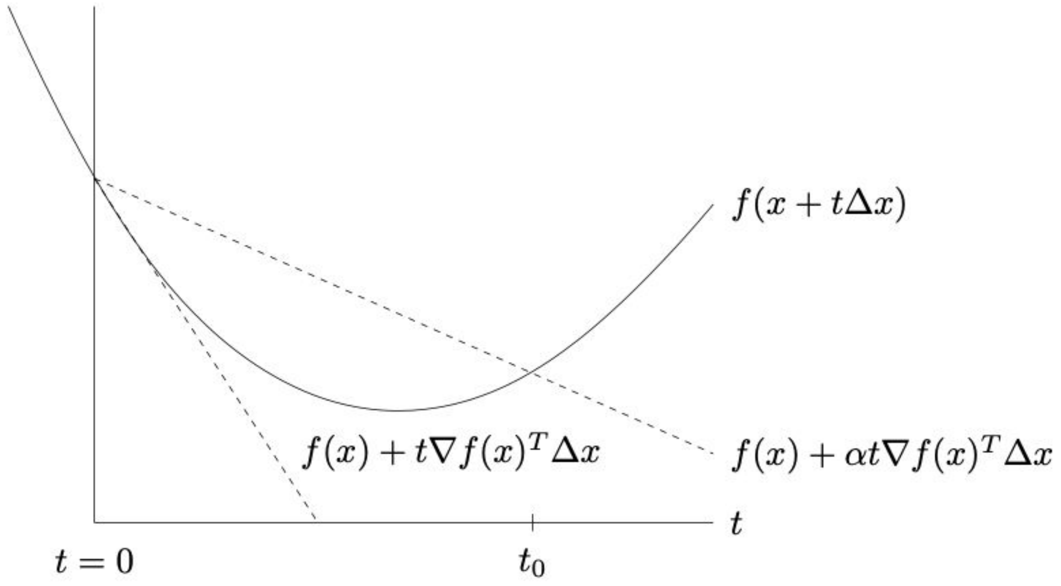
Gradient descent: fixed step size

Converges nicely when t is "just right". Same example, 40 steps:



Convergence analysis later will give us a precise idea of "just right"

Gradient descent


$f(x + t\Delta x)$
$f(x) + t\nabla f(x)^T \Delta x$
$f(x) + \alpha t\nabla f(x)^T \Delta x$
t
$t = 0$
t_0

For us $\Delta x = -\nabla f(x)$

Gradient descent

Stopping rule: stop when $\|\nabla f(x)\|_2$ is small

- Recall $\nabla f(x^\star) = 0$ at solution $x^\star$
- If f is strongly convex with parameter m, then

$$\|\nabla f(x)\|_2 \leq \sqrt{2m\epsilon} \implies f(x) - f^\star \leq \epsilon$$

Pros and cons of gradient descent:

- Pro: simple idea, and each iteration is cheap (usually)
- Pro: fast for well-conditioned, strongly convex problems
- Con: can often be slow, because many interesting problems aren't strongly convex or well-conditioned
- Con: can't handle nondifferentiable functions

Gradient descent: convergence

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

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2 \quad \text{for any } x, y$$

I.e., ∇f is Lipschitz continuous with constant $L > 0$

Theorem: Gradient descent with fixed step size $t \leq 1/L$ satisfies

$$f(x^{(k)}) - f^\star \leq \frac{\|x^{(0)} - x^\star\|_2^2}{2tk}$$

and same result holds for backtracking, with t replaced by β/L

We say gradient descent has convergence rate $O(1/k)$. I.e., it finds ϵ-suboptimal point in $O(1/\epsilon)$ iterations

Columbia University

Gradient descent: can we do better?

Gradient descent has $O(1/\epsilon)$ convergence rate over problem class of convex, differentiable functions with Lipschitz gradients

First-order method: iterative method, which updates $x^{(k)}$ in

$$x^{(0)} + \mathrm{span}\{\nabla f(x^{(0)}), \nabla f(x^{(1)}), \ldots \nabla f(x^{(k-1)})\}$$

Theorem (Nesterov): For any $k \leq (n-1)/2$ and any starting point $x^{(0)}$, there is a function f in the problem class such that any first-order method satisfies

$$f(x^{(k)}) - f^\star \geq \frac{3L\|x^{(0)} - x^\star\|_2^2}{32(k+1)^2}$$

Can attain rate $O(1/k^2)$, or $O(1/\sqrt{\epsilon})$? Answer: yes (we'll see)!

Gradient descent: can we do better?

Gradient descent has $O(1/\epsilon)$ convergence rate over problem class of convex, differentiable functions with Lipschitz gradients

First-order method: iterative method, which updates $x^{(k)}$ in

$$x^{(0)} + \text{span}\{\nabla f(x^{(0)}), \nabla f(x^{(1)}), \ldots \nabla f(x^{(k-1)})\}$$

Theorem (Nesterov): For any $k \le (n-1)/2$ and any starting point $x^{(0)}$, there is a function f in the problem class such that any first-order method satisfies

$$f(x^{(k)}) - f^\star \ge \frac{3L\|x^{(0)} - x^\star\|_2^2}{32(k+1)^2}$$

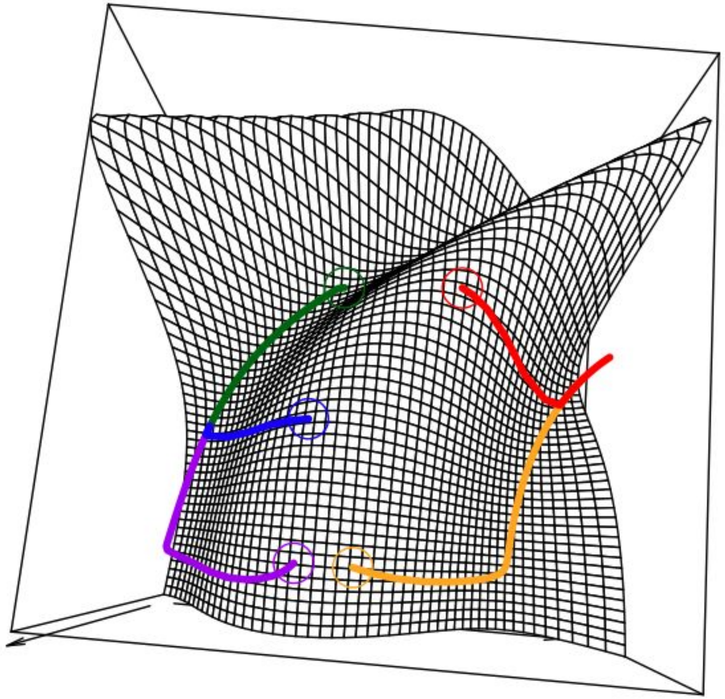
Can attain rate $O(1/k^2)$, or $O(1/\sqrt{\epsilon})$? Answer: yes (we'll see)!

Nesterov's

Accelerated

Gradient Descent!

Gradient descent: non-convex function



COLUMBIA UNIVERSITY

Gradient descent

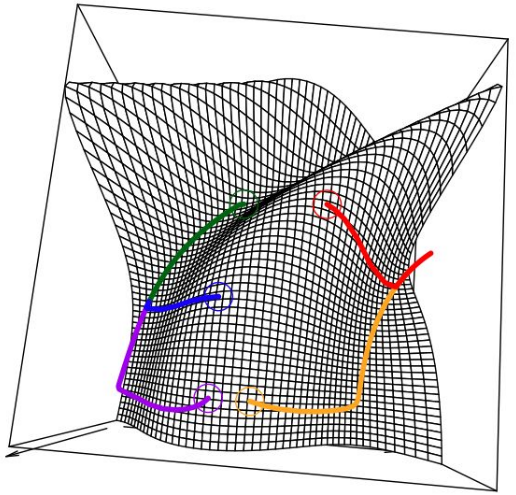
Assume f is differentiable with Lipschitz gradient as before, but now nonconvex. Asking for optimality is too much. So we'll settle for x such that $\|\nabla f(x)\|_2 \le \epsilon$, called ϵ-stationarity

Theorem: Gradient descent with fixed step size $t \le 1/L$ satisfies

$$\min_{i=0,\dots,k} \|\nabla f(x^{(i)})\|_2 \le \sqrt{\frac{2(f(x^{(0)}) - f^\star)}{t(k+1)}}$$

Thus gradient descent has rate $O(1/\sqrt{k})$, or $O(1/\epsilon^2)$, even in the nonconvex case for finding stationary points

This rate cannot be improved (over class of differentiable functions with Lipschitz gradients) by any deterministic algorithm[1]



Modern Stochastic Methods

Consider minimizing an average of functions

$$\min_x \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

As $\nabla \sum_{i=1}^{n} f_i(x) = \sum_{i=1}^{n} \nabla f_i(x)$, gradient descent or GD repeats:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x^{(k-1)}), \quad k = 1, 2, 3, \dots$$

In comparison, stochastic gradient descent or SGD repeats:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f_{i_k}(x^{(k-1)}), \quad k = 1, 2, 3, \dots$$

where $i_k \in \{1, \dots n\}$ is randomly chosen index at iteration k. Note $\mathbb{E}[\nabla f_{i_k}(x)] = \nabla f(x)$, so we use unbiased estimate of full gradient

COLUMBIA UNIVERSITY

Stochastic gradient descent: mini-batches

Also common is mini-batch stochastic gradient descent, where we choose a random subset $I_k \subseteq \{1, \dots n\}$, of size $|I_k| = b \ll n$, and repeat:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \frac{1}{b} \sum_{i \in I_k} \nabla f_i(x^{(k-1)}), \quad k = 1, 2, 3, \dots$$

Again, we are approximating full graident by an unbiased estimate:

$$\mathbb{E}\left[\frac{1}{b} \sum_{i \in I_k} \nabla f_i(x)\right] = \nabla f(x)$$

Using mini-batches reduces the variance of our gradient estimate by a factor $1/b$, but is also b times more expensive

SGD for logistic regression

Given $(x_i, y_i) \in \mathbb{R}^p \times \{0, 1\}$, $i = 1, \ldots n$, recall logistic regression:

$$\min_{\beta} \; f(\beta) = \frac{1}{n} \sum_{i=1}^{n} \underbrace{\Big(-y_i x_i^T \beta + \log(1 + \exp(x_i^T \beta)) \Big)}_{f_i(\beta)}$$

Gradient computation $\nabla f(\beta) = \frac{1}{n} \sum_{i=1}^{n} \big(y_i - p_i(\beta) \big) x_i$ is doable when n is moderate, but not when n is huge

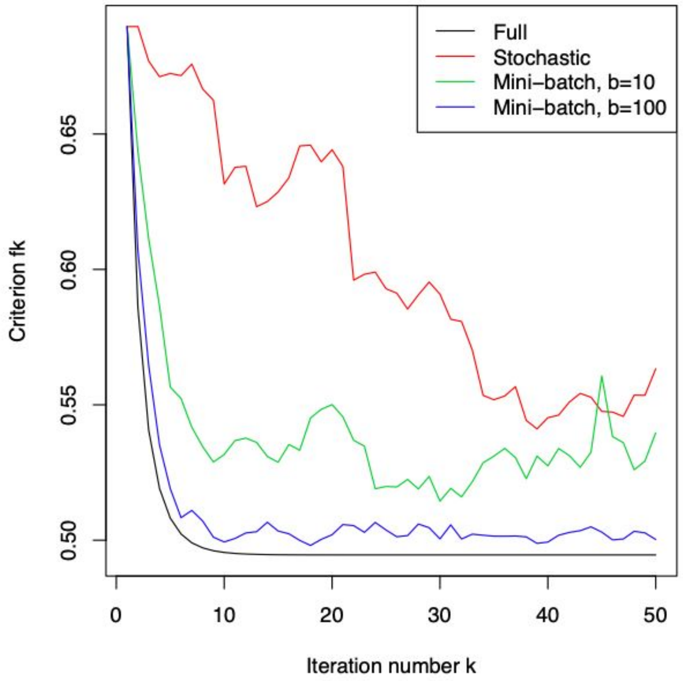
Full gradient (also called batch) versus stochastic gradient:

- One batch update costs $O(np)$
- One mini-batch update costs $O(bp)$
- One stochastic update costs $O(p)$

COLUMBIA UNIVERSITY

SGD for logistic regression

Example with $n = 10,000$, $p = 20$, all methods use fixed step sizes:

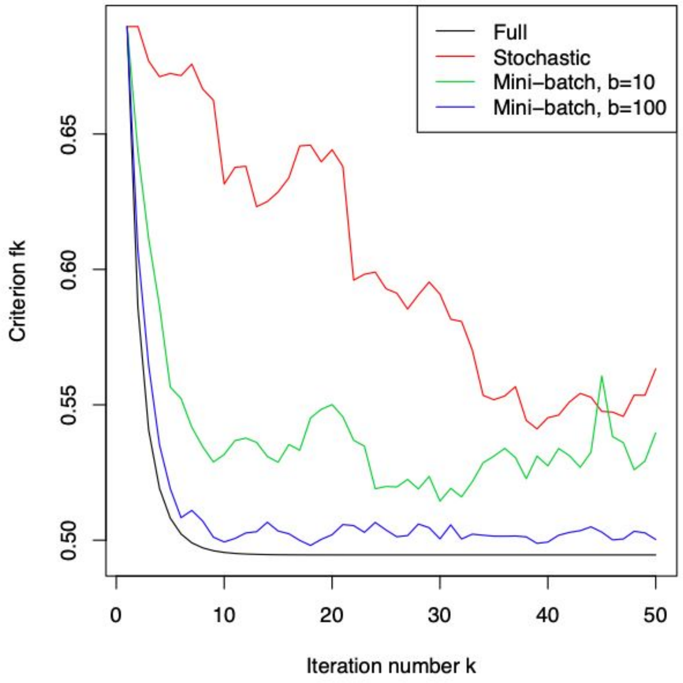


SGD for logistic regression

Example with $n = 10,000$, $p = 20$, all methods use fixed step sizes:

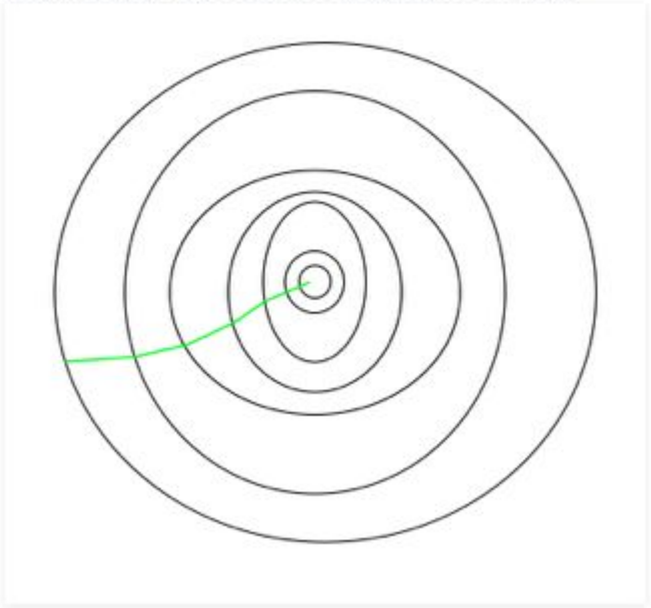
No free lunch!

We give up the steepest descent to trade for faster calculation speed!

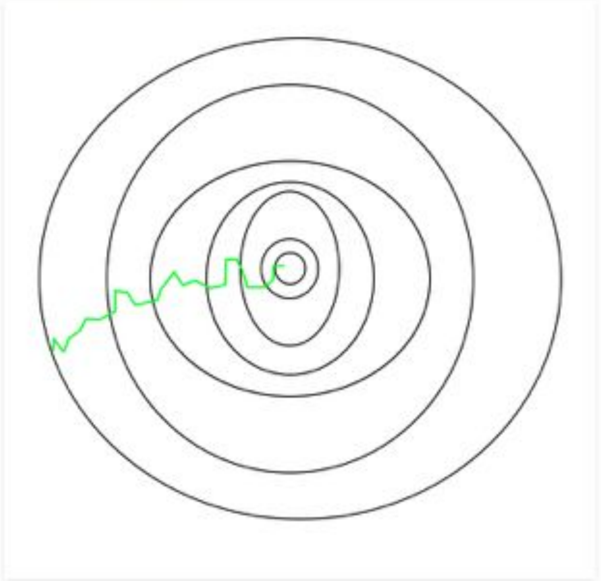


Comparison

Path taken by Batch Gradient Descent –



Path taken by Stochastic Gradient Descent –



Convergence rate

Recall the following:

Condition	GD rate	SGD rate
Convex	$O(1/\sqrt{k})$	$O(1/\sqrt{k})$
+ Lipschitz gradient	$O(1/k)$	$O(1/\sqrt{k})$
+ Strongly convex	$O(c^k)$	$O(1/k)$

Notes:

- In GD, we can take fixed step sizes in the latter two cases
- In SGD, we always take diminishing step sizes to control the variance (of the gradient estimate)
- Mini-batches are a wash in terms of flops (but still popular practice)

Stochastic average gradient

Stochastic average gradient or SAG (Schmidt, Le Roux, and Bach 2013) is a breakthrough method in stochastic optimization:

- Maintain table, containing gradient g_i of f_i, $i = 1, \ldots n$
- Initialize $x^{(0)}$, and $g_i^{(0)} = \nabla f_i(x^{(0)})$, $i = 1, \ldots n$
- At steps $k = 1, 2, 3, \ldots$, pick random $i_k \in \{1, \ldots n\}$, then let

$$g_{i_k}^{(k)} = \nabla f_{i_k}(x^{(k-1)}) \quad \text{(most recent gradient of } f_{i_k}\text{)}$$

$$\mathbb{E}\left[\frac{1}{b}\sum_{i \in I_k} \nabla f_i(x)\right] = \nabla f(x)$$

Set all other $g_i^{(k)} = g_i^{(k-1)}$, $i \neq i_k$, i.e., these stay the same

- Update

$$x^{(k)} = x^{(k-1)} - t_k \cdot \frac{1}{n}\sum_{i=1}^{n} g_i^{(k)}$$

Columbia University

Stochastic average gradient

- Key of SAG is to allow each f_i, $i = 1, \ldots n$ to communicate a part of the gradient estimate at each step
- This basic idea can be traced back to incremental aggregated gradient (Blatt, Hero, Gauchman, 2006)
- SAG gradient estimates are no longer unbiased, but they have greatly reduced variance
- Isn't it expensive to average all these gradients? Basically just as efficient as SGD, as long we're clever:

$$\mathbb{E}\left[\frac{1}{b}\sum_{i\in I_k}\nabla f_i(x)\right] = \nabla f(x)$$

$$x^{(k)} = x^{(k-1)} - t_k \cdot \underbrace{\left(\frac{g_{i_k}^{(k)}}{n} - \frac{g_{i_k}^{(k-1)}}{n} + \underbrace{\frac{1}{n}\sum_{i=1}^{n} g_i^{(k-1)}}_{\text{old table average}}\right)}_{\text{new table average}}$$

Columbia University

And many, many others...

A lot of recent work revisiting stochastic optimization:

- SDCA (Shalev-Schwartz, Zhang, 2013): applies coordinate ascent to the dual of ridge regularized problems, and uses randomly selected coordinates. Effective primal updates are similar to SAG/SAGA
- SVRG (Johnson, Zhang, 2013): like SAG/SAGA, but does not store a full table of gradients, just an average, and updates this occasionally
- There's also S2GD (Konecny, Richtarik, 2014), MISO (Mairal, 2013), Finito (Defazio, Caetano, Domke, 2014), etc.
- Both the SAG and SAGA papers give very nice reviews and discuss connections

COLUMBIA UNIVERSITY

Acceleration, momentum and beyond

Variance reduction + acceleration completely solve the finite sum case. Beyond this, the story is much more complicated ...

- Recall, for general stochastic setting, the performance of SGD cannot be improved (matching lower bounds in Nemirovski et al. 2009)
- Acceleration is less used for nonconvex problems (?), but a related technique is often used: momentum
- Predates acceleration by nearly two decades (Polyak, 1964). In practice, Polyak's heavy ball method can work really well:

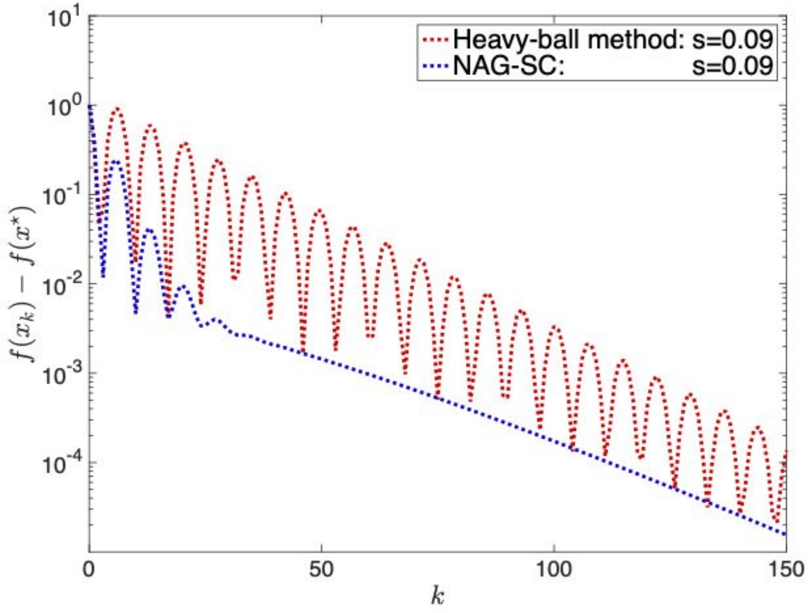
$$x^{(k)} = x^{(k-1)} + \alpha(x^{(k-1)} - x^{(k-2)}) - t_k \nabla f_{i_k}(x^{(k-1)})$$

  but it can also be somewhat fragile
- Open problem: when and why does this work?

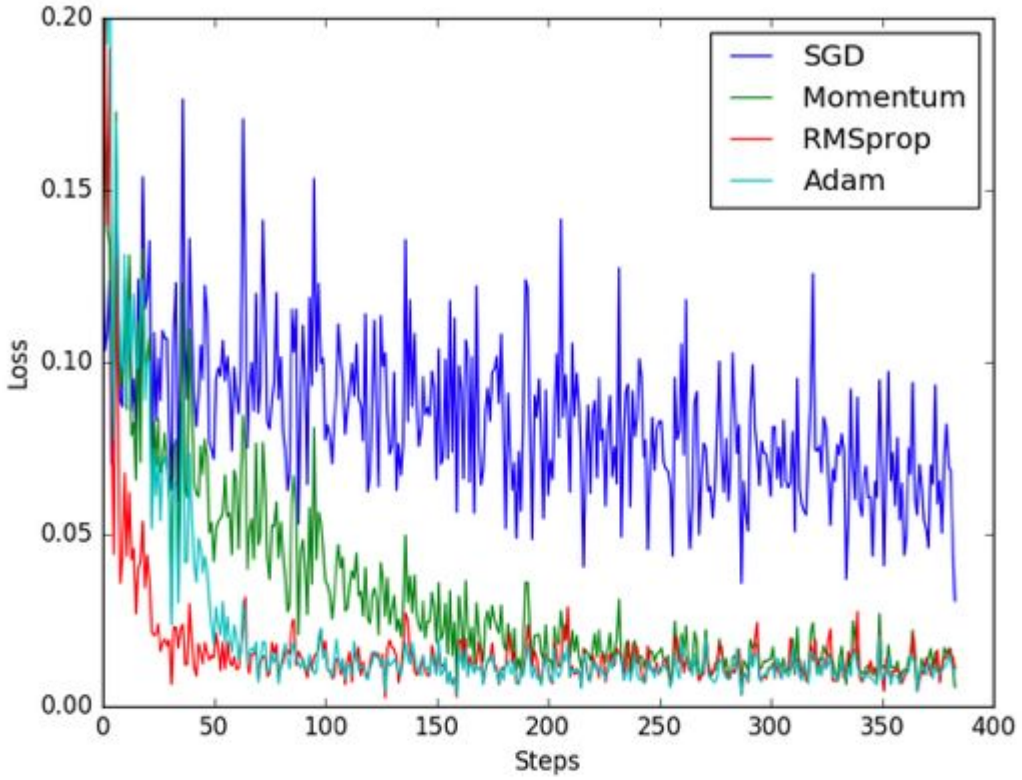
COLUMBIA UNIVERSITY

Acceleration, momentum and beyond

Polyak's heavy ball versus Nesterov acceleration, in optimizing a convex quadratic (from Shi et al., 2018):



Comparison



SGD
Momentum
RMSprop
Adam
Loss
Steps
0.00
0.05
0.10
0.15
0.20
0
50
100
150
200
250
300
350
400

References

- S. Boyd and L. Vandenberghe: Convex Optimization, Chapter 9
- Trevor Hastie, Robert Tibshirani, Jerome Friedman: The Elements of Statistical Learning: Data Mining, Inference and Prediction, Chapter 10, 16
- Ziv Bar-Joseph, Tom Mitchell, Pradeep Ravikumar and Aarti Singh: CMU 10-701
- Ryan Tibshirani: CMU 10-725
- https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/