

# SGD and GD Revisited

David Rosenberg

New York University

February 17, 2016

# Terminology

# Iterative Optimization Methods

## Iterative Optimization (Generic Version)

- ① Pick some starting point  $x^{(0)} \in \mathbf{R}^d$ .
  - ② For  $k = 0, 1, \dots$ 
    - ① Choose a **step** or **search direction**  $v^{(k)}$ .
    - ② Choose a **step size**  $t^{(k)}$ .
    - ③ Set  $x^{(k+1)} = x^{(k)} + t^{(k)} v^{(k)}$
- Despite the names,
    - $v^{(k)}$  is **not** generally a unit vector.
    - $t^{(k)}$  is **not**  $\|x^{(k+1)} - x^{(k)}\|$  (unless  $\|v^{(k)}\| = 1$ ).

# Descent Directions

## Definition

A **[one-sided] directional derivative** of  $f$  at  $x$  in the direction  $v$  is

$$f'(x; v) = \lim_{h \downarrow 0} \frac{f(x + hv) - f(x)}{h}.$$

[Note: Can be  $\pm\infty$ , e.g. for discontinuous functions.]

## Definition

$v$  is a **descent direction** for  $f$  at  $x$  if  $f'(x; v) < 0$ .

# Descent Directions

- If  $f$  is differentiable, then

$$f'(x, v) = \nabla f(x)^T v.$$

- So if  $f$  is differentiable, then  $v$  is a descent direction at  $x$  iff

$$\nabla f(x)^T v < 0.$$

- **Newton** step is a descent direction for strictly convex functions:

$$v_{\text{newton}} = -[\nabla^2 f(x)]^{-1} \nabla f(x)$$

- Much faster convergence close to the optimum.
- Computing/storing inverse Hessian is too much in high dimensions.
- Quasi-Newton methods approximate Newton step, without Hessian (e.g. L-BFGS).

# Descent Method

## Definition

An iterative optimization method is a **descent method** if every step is a descent direction.

- Equivalently, we have a descent method if

$$f(x^{(k+1)}) < f(x^{(k)}),$$

except when  $x^{(k)}$  is optimal.

- Is SGD a descent method?

# Stochastic Gradient Descent

# Gradient Descent

## Gradient Descent

- Initialize  $x = 0$
- repeat
  - $x \leftarrow x - \eta \nabla f(x)$



# “Noisy” Gradient Descent

## “Noisy” Gradient Descent (not a standard name)

- Initialize  $x = 0$
- repeat
  - $x \leftarrow x - \eta v$

Where  $v$  is some estimate of  $\nabla f(x)$

- In minibatch SGD, we have  $\mathbb{E}v = \nabla f(x)$ .
- With large batches, we get better estimates. (  $\text{Var}(v)$  decreases.)

# SGD on Regularized Empirical Risk

- Our typical objective function is of the form

$$\begin{aligned} J(w) &= \lambda\Omega(w) + \frac{1}{n} \sum_{i=1}^n \ell(f_w(x_i), y_i) \\ &= \frac{1}{n} \sum_{i=1}^n h_i(w) \end{aligned}$$

where

$$h_i(w) = \lambda\Omega(w) + \ell(f_w(x_i), y_i).$$

- SGD on  $J(w)$ :
  - Choose  $i \in \{1, \dots, n\}$  uniformly at random
  - Approximate  $\nabla J(w)$  by  $\nabla h_i(w)$ .
- Step is unbiased for gradient:

$$\mathbb{E}_{i \sim \text{Unif}(1, \dots, n)} \nabla h_i(w) = \nabla J(w)$$

# SGD on Risk

- Suppose  $(x, y) \sim P_{\mathcal{X} \times \mathcal{Y}}$  and objective is the expected loss:

$$J(w) = \mathbb{E} \ell(f_w(x), y).$$

- SGD on  $J(w)$ :
  - Choose  $(x, y) \sim P_{\mathcal{X} \times \mathcal{Y}}$ .
  - Approximate  $\nabla_w J(w)$  by  $\nabla_w \ell(f_w(x), y)$ .
- Step is unbiased for gradient:

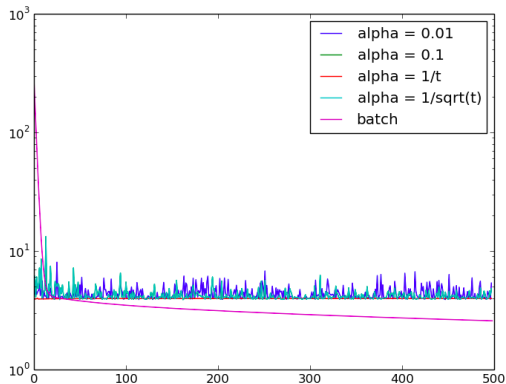
$$\mathbb{E}_{(x,y) \sim P_{\mathcal{X} \times \mathcal{Y}}} \nabla_w \ell(f_w(x), y) = \nabla_w \mathbb{E} \ell(f_w(x), y)$$

- To implement this, need fresh samples from  $P_{\mathcal{X} \times \mathcal{Y}}$ .
- If we're resampling from training set,  $(x, y) \sim \hat{P}_{\mathcal{X} \times \mathcal{Y}}$  we get back SGD.

# Convergence Rates

# Does SGD Catch Up to GD?

- Loss on ridge regression for GD and SGD with various stepsizes



- Why doesn't SGD catch up to batch GD?
- Short answer: It does, just takes a very long time.

# Convergence Rates for Gradient Descent

Assume  $f : \mathbf{R}^n \rightarrow \mathbf{R}$  is convex and differentiable, and

- $\nabla f$  is **Lipschitz continuous** with constant  $L > 0$ , that is:

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

## Theorem

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

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

- So GD has convergence rate  $O(1/k)$ .
- To get  $f(x^{(k)}) - f(x^*) \leq \varepsilon$ , need  $O(1/\varepsilon)$  iterations.
- Same rate with backtracking line search.

# Convergence Rates for GD with Strong Convexity

## Definition

A differentiable function  $f$  is strongly convex if there is some  $d > 0$  for which

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{d}{2} \|y - x\|^2 \text{ all } x, y.$$

(e.g. ridge regression because of the  $\ell_2$  regularization term)

## Theorem

*Under same Lipschitz condition as before and strong convexity, GD with fixed step size or with backtracking line search satisfies*

$$f(x^{(k)}) - f(x^*) \leq c^k \frac{L}{2} \|x^{(0)} - x^*\|^2,$$

where  $0 < c < 1$ .

# Convergence Rates for GD with Strong Convexity

## Theorem

*Under same Lipschitz condition as before and strong convexity, GD with fixed step size or with backtracking line search satisfies*

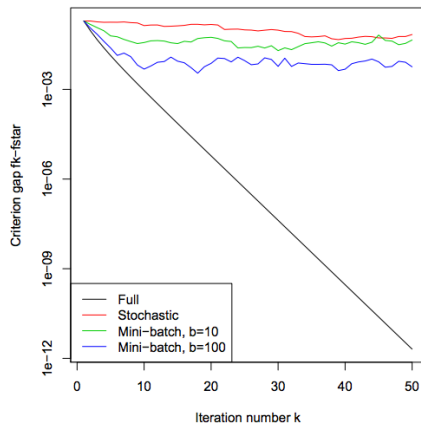
$$f(x^{(k)}) - f(x^*) \leq c^k \frac{L}{2} \|x^{(0)} - x^*\|^2,$$

where  $0 < c < 1$ .

- So with strong convexity, GD converges at rate  $O(c^k)$ . (exponentially fast!)
- To get  $f(x^{(k)}) - f(x^*) \leq \varepsilon$ , need  $O(\log[1/\varepsilon])$  iterations.
- Called “**linear convergence**” because looks linear on semi-log plot.



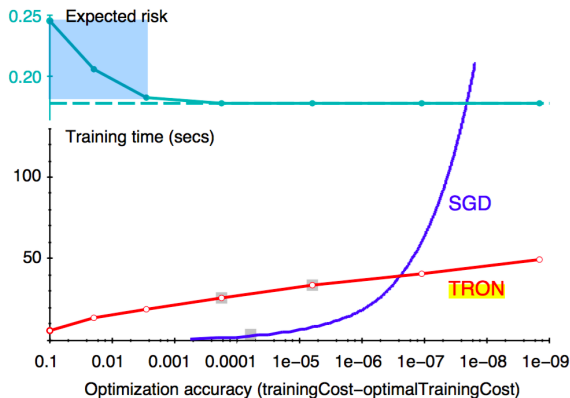
## SGD vs GD on Log Scale



- Shows **linear convergence** for “Full” GD; **sublinear** for others.
- Note: logarithmic y-axis

# SGD is Slow Close to the Optimum – Does it Matter?

- TRON is a 2nd order method (very fast close to the optimum)



Plots from Bottou and Bousquet's "The Tradeoffs of Large Scale Learning."