Optimization Methods for Large-Scale Machine Learning

Frank E. Curtis, Lehigh University

presented at

East Coast Optimization Meeting
George Mason University

Fairfax, Virginia

April 2, 2021

References



- ⋆ Léon Bottou, Frank E. Curtis, and Jorge Nocedal.
  Optimization Methods for Large-Scale Machine Learning.
  SIAM Review, 60(2):223–311, 2018.

- ⋆ Frank E. Curtis and Katya Scheinberg.
  Optimization Methods for Supervised Machine Learning: From Linear Models to Deep Learning.
  In *INFORMS Tutorials in Operations Research*, chapter 5, pages 89–114. Institute for Operations Research and the Management Sciences (INFORMS), 2017.

Motivating questions

- How do optimization problems arise in machine learning applications, and what makes them challenging?
- What have been the most successful optimization methods for large-scale machine learning, and why?
- What recent advances have been made in the design of algorithms, and what are open questions in this research area?

Outline

Outline

Learning problems and (surrogate) optimization problems

Learn a prediction function $h : \mathcal{X} \to \mathcal{Y}$ to solve

$$\max_{h \in \mathcal{H}} \int_{\mathcal{X} \times \mathcal{Y}} \mathbb{1}[h(x) \approx y] dP(x, y)$$

Various meanings for $h(x) \approx y$ depending on the goal:

- Binary classification, with $y \in \{-1, +1\}$: $y \cdot h(x) > 0$.
- Regression, with $y \in \mathbb{R}^{n_y}$: $\|h(x) - y\| \leq \delta$.

Parameterizing h by $w \in \mathbb{R}^d$, we aim to solve

$$\max_{w \in \mathbb{R}^d} \int_{\mathcal{X} \times \mathcal{Y}} \mathbb{1}[h(w; x) \approx y] dP(x, y)$$

Now, common practice is to replace the indicator with a smooth loss...

Stochastic optimization

Over a parameter vector $w \in \mathbb{R}^d$ and given

$$\ell(\cdot; y) \circ h(w; x) \quad \text{(loss w.r.t. “true label” } \circ \text{ prediction w.r.t. “features”)},$$

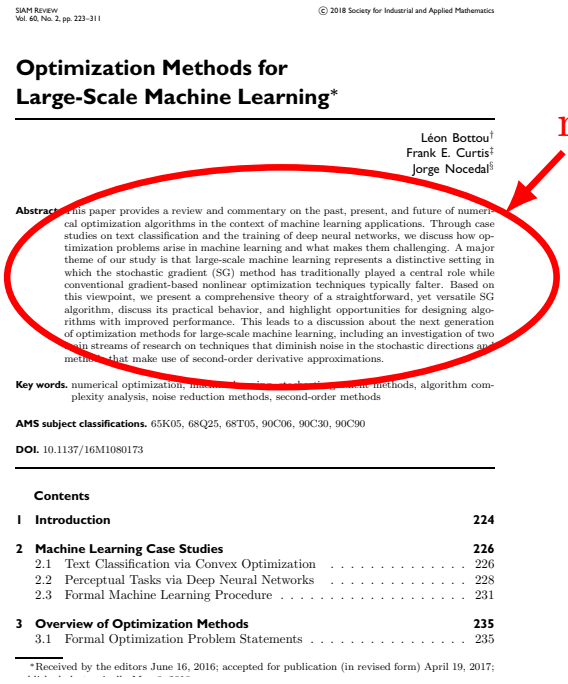
consider the unconstrained optimization problem

$$\min_{w \in \mathbb{R}^d} \; f(w), \quad \text{where} \quad f(w) = \mathbb{E}_{(x,y)}[\ell(h(w; x), y)].$$

Given training set $\{(x_i, y_i)\}_{i=1}^n$, approximate problem given by

$$\min_{w \in \mathbb{R}^d} \; f_n(w), \quad \text{where} \quad f_n(w) = \frac{1}{n} \sum_{i=1}^n \ell(h(w; x_i), y_i).$$

Text classification



SIAM REVIEW
Vol. 60, No. 2, pp. 223–311

Optimization Methods for Large-Scale Machine Learning*

Léon Bottou†
Frank E. Curtis‡
Jorge Nocedal§

Abstract. This paper provides a review and commentary on the past, present, and future of numerical optimization algorithms in the context of machine learning applications. Through case studies on text classification and the training of deep neural networks, we discuss how optimization problems arise in machine learning and what makes them challenging. A major theme of our study is that large-scale machine learning represents a distinctive setting in which the stochastic gradient (SG) method has traditionally played a central role while conventional gradient-based nonlinear optimization techniques typically falter. Based on this viewpoint, we present a comprehensive theory of a straightforward, yet versatile SG algorithm, discuss its practical behavior, and highlight opportunities for designing algorithms with improved performance. This leads to a discussion about the next generation of optimization methods for large-scale machine learning, including an investigation of two main streams of research on techniques that diminish noise in the stochastic directions and methods that make use of second-order derivative approximations.

Key words. numerical optimization, [illegible] methods, algorithm complexity analysis, noise reduction methods, second-order methods

AMS subject classifications. 65K05, 68Q25, 68T05, 90C06, 90C30, 90C90

DOI. 10.1137/16M1080173

Contents

*Received by the editors June 16, 2016; accepted for publication (in revised form) April 19, 2017;

math

poetry

The New York Times

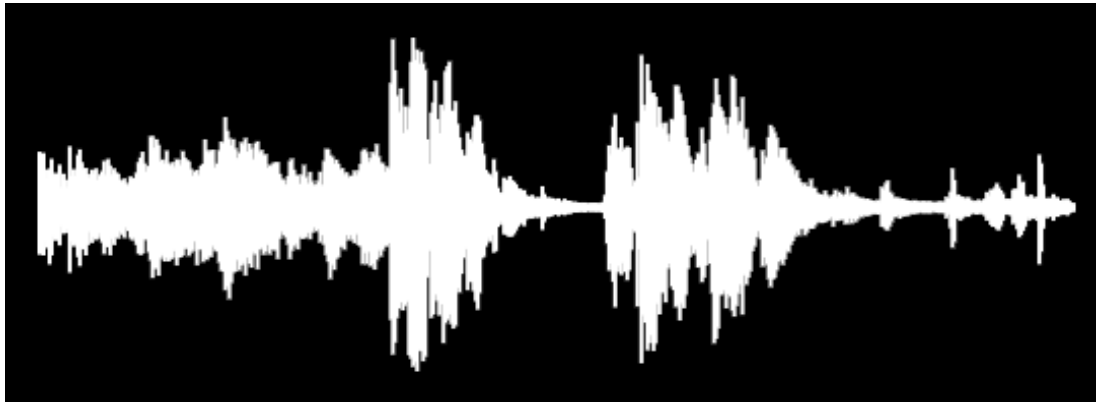
UP NEXT

Meet Amanda Gorman, America's First Youth Poet Laureate

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-(w^T x_i) y_i)) + \frac{\lambda}{2} \|w\|_2^2$$

Image / speech recognition

What pixel combinations represent the number 4?



What sounds are these? ("Here comes the sun" – The Beatles)

Deep neural networks

$$h(w;x) = a_l(W_l \dots (a_2(W_2(a_1(W_1 x + \omega_1)) + \omega_2)) \dots)$$

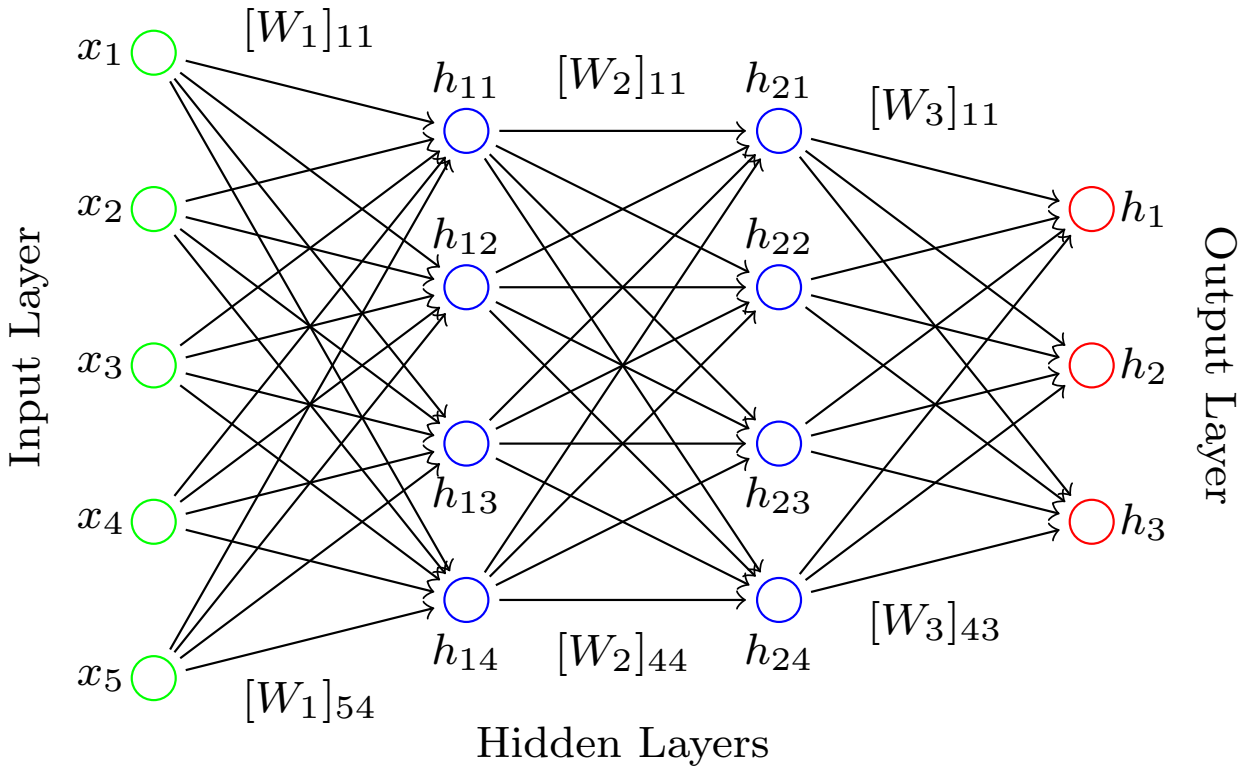


Figure: Illustration of a DNN

Tradeoffs of large-scale learning

Bottou, Bousquet (2008) and Bottou (2010)

Notice that we went from our true problem

$$\max_{h \in \mathcal{H}} \int_{\mathcal{X} \times \mathcal{Y}} \mathbb{1}[h(x) \approx y] dP(x, y)$$

to say that we'll find our solution $h \equiv h(w; \cdot)$ by (approximately) solving

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(h(w; x_i), y_i).$$

Three sources of error:

- approximation
- estimation
- optimization

Approximation error

Choice of prediction function family $\mathcal{H}$ has important implications; e.g.,

$$\mathcal{H}_C := \{h \in \mathcal{H} : \Omega(h) \leq C\}.$$

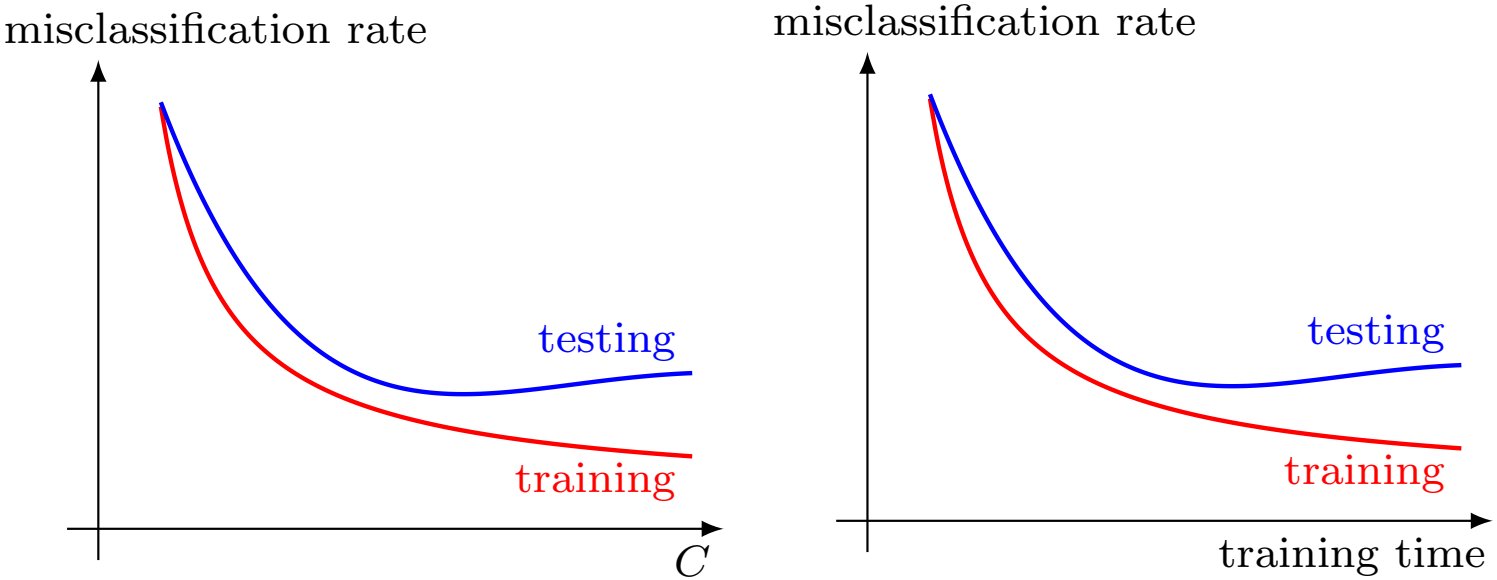


Figure: Illustration of C and training time vs. misclassification rate

Problems of interest

Let's focus on the expected loss/risk problem

$$\min_{w \in \mathbb{R}^d} \; f(w), \quad \text{where} \quad f(w) = \mathbb{E}_{(x,y)}[\ell(h(w;x), y)]$$

and the empirical loss/risk problem

$$\min_{w \in \mathbb{R}^d} \; f_n(w), \quad \text{where} \quad f_n(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(w;x_i), y_i).$$

For this talk, let's assume

- f is continuously differentiable, bounded below, and potentially nonconvex;
- ∇f is L-Lipschitz continuous, i.e., $\|\nabla f(w) - \nabla f(\overline{w})\|_2 \leq L\|w - \overline{w}\|_2$.

Gradient descent

Aim: Find a stationary point, i.e., w with $\nabla f(w) = 0$.

Algorithm GD : Gradient Descent

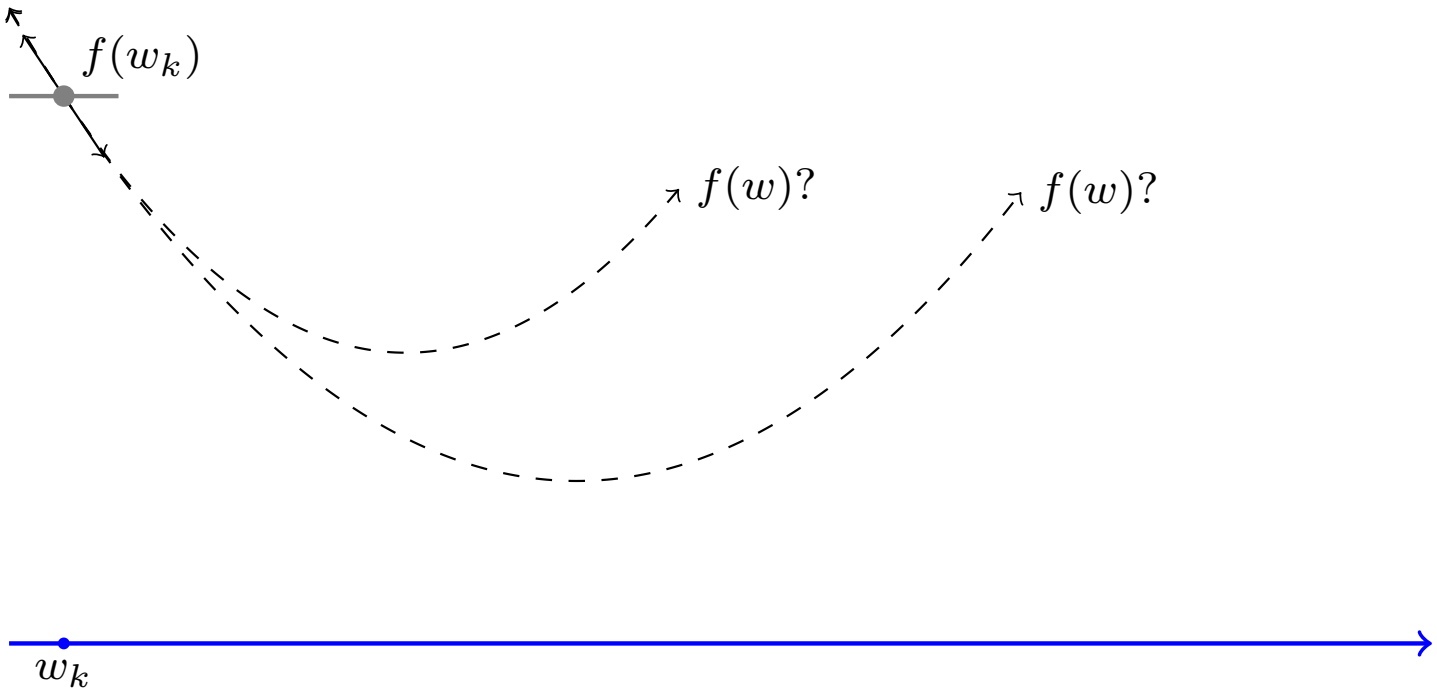
1: choose an initial point $w_0 \in \mathbb{R}^n$ and stepsize $\alpha > 0$
2: **for** $k \in \{0, 1, 2, \dots\}$ **do**
3: set $w_{k+1} \leftarrow w_k - \alpha \nabla f(w_k)$
4: **end for**

Gradient descent

Aim: Find a stationary point, i.e., w with $\nabla f(w) = 0$.

Algorithm GD : Gradient Descent

1: choose an initial point $w_0 \in \mathbb{R}^n$ and stepsize $\alpha > 0$
2: **for** $k \in \{0, 1, 2, \dots\}$ **do**
3: set $w_{k+1} \leftarrow w_k - \alpha \nabla f(w_k)$
4: **end for**

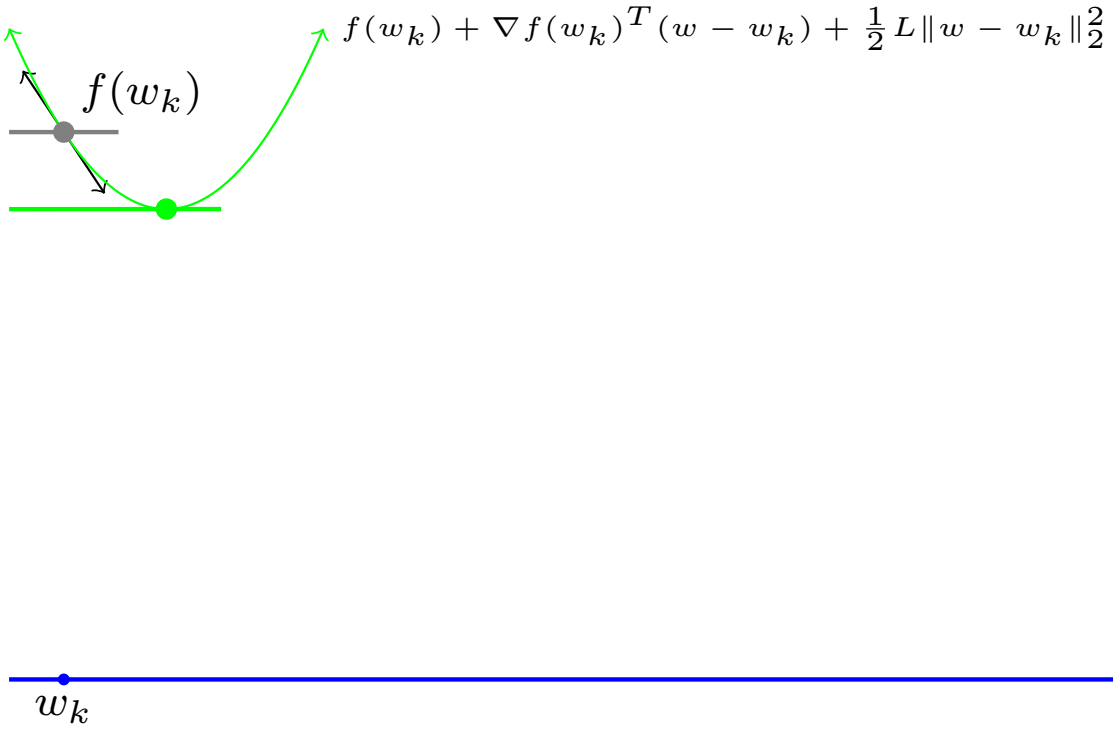


Gradient descent

Aim: Find a stationary point, i.e., w with $\nabla f(w) = 0$.

Algorithm GD : Gradient Descent

1: choose an initial point $w_0 \in \mathbb{R}^n$ and stepsize $\alpha > 0$
2: **for** $k \in \{0, 1, 2, \dots\}$ **do**
3: set $w_{k+1} \leftarrow w_k - \alpha \nabla f(w_k)$
4: **end for**

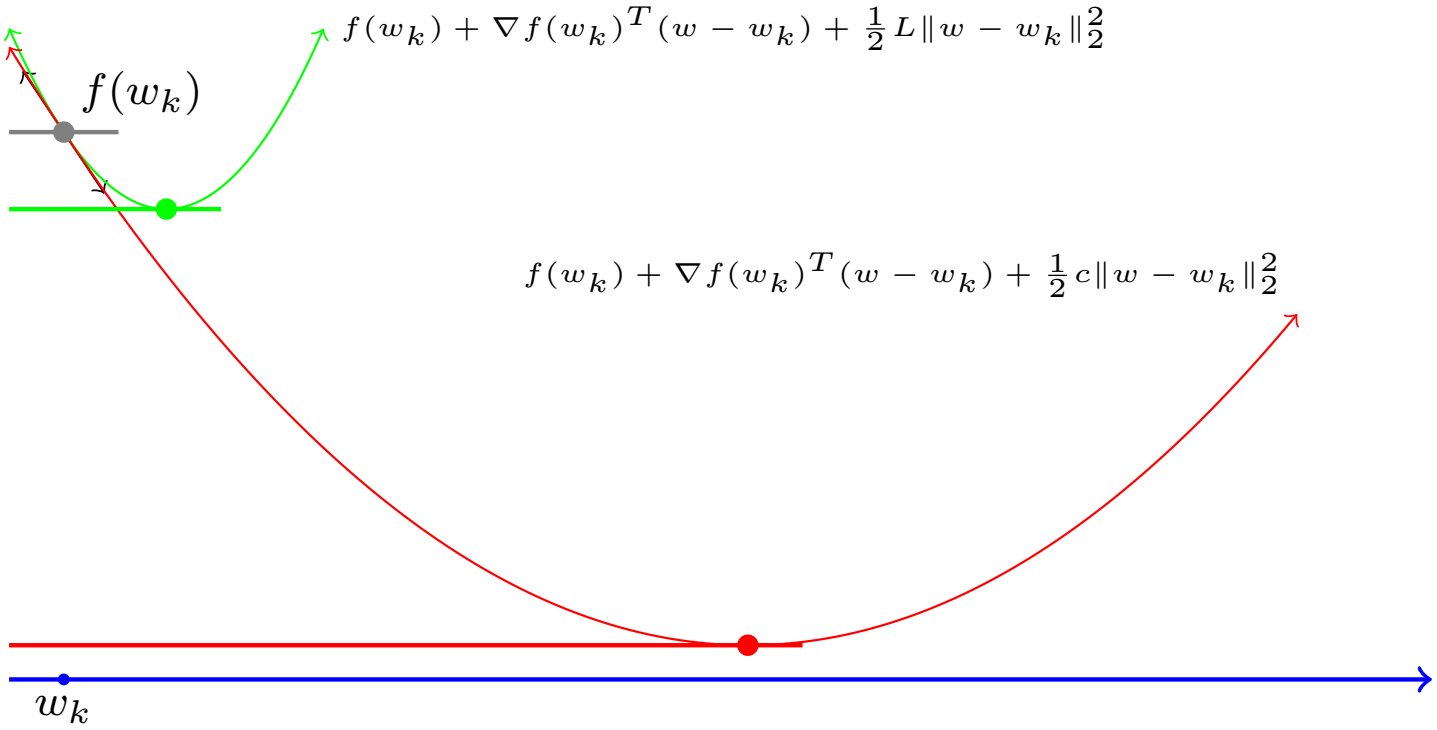


Gradient descent

Aim: Find a stationary point, i.e., w with $\nabla f(w) = 0$.

Algorithm GD : Gradient Descent

1: choose an initial point $w_0 \in \mathbb{R}^n$ and stepsize $\alpha > 0$
2: **for** $k \in \{0, 1, 2, \dots\}$ **do**
3: set $w_{k+1} \leftarrow w_k - \alpha \nabla f(w_k)$
4: **end for**



GD theory

Theorem GD

If $\alpha \in (0, 1/L]$, *then* $\sum_{k=0}^{\infty} \|\nabla f(w_k)\|_2^2 < \infty$, *which implies* $\{\nabla f(w_k)\} \to 0$.

If, in addition, f *is* c*-strongly convex, then for all* $k \geq 1$*:*

$$f(w_k) - f_* \leq (1 - \alpha c)^k (f(x_0) - f_*).$$

Proof.

$$\begin{aligned} f(w_{k+1}) &\leq f(w_k) + \nabla f(w_k)^T (w_{k+1} - w_k) + \tfrac{1}{2} L \|w_{k+1} - w_k\|_2^2 \\ &\cdots \text{(due to stepsize choice)} \\ &\leq f(w_k) - \tfrac{1}{2}\alpha \|\nabla f(w_k)\|_2^2 \\ &\leq f(w_k) - \alpha c (f(w_k) - f_*). \\ &\implies f(w_{k+1}) - f_* \leq (1 - \alpha c)(f(w_k) - f_*). \end{aligned}$$

GD illustration

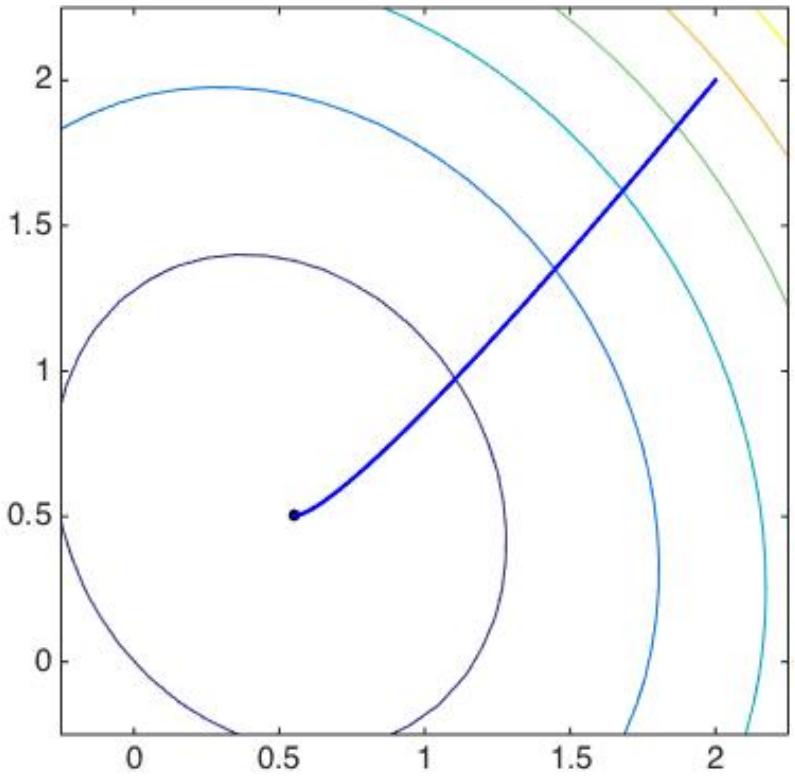


Figure: GD with fixed stepsize

Stochastic gradient method (SG)

Invented by Herbert Robbins and Sutton Monro in 1951.

Sutton Monro, former Lehigh faculty member

Stochastic gradient ~~descent~~

Approximate gradient only; e.g., random i_k so $\mathbb{E}[\nabla_w \ell(h(w; x_{i_k}), y_{i_k})|w] = \nabla f(w)$.

Algorithm SG : Stochastic Gradient

```
choose an initial point w_0 ∈ ℝ^n and stepsizes {α_k} > 0
for k ∈ {0, 1, 2, ...} do
    set w_{k+1} ← w_k − α_k g_k, where g_k ≈ ∇f(w_k)
end for
```

Not a descent method!

...but can guarantee *eventual descent in expectation* (with $\mathbb{E}_k[g_k] = \nabla f(w_k)$):

$$\begin{aligned} f(w_{k+1}) &\leq f(w_k) + \nabla f(w_k)^T (w_{k+1} - w_k) + \tfrac{1}{2} L \|w_{k+1} - w_k\|_2^2 \\ &= f(w_k) - \alpha_k \nabla f(w_k)^T g_k + \tfrac{1}{2} \alpha_k^2 L \|g_k\|_2^2 \\ \implies \mathbb{E}_k[f(w_{k+1})] &\leq f(w_k) - \alpha_k \|\nabla f(w_k)\|_2^2 + \tfrac{1}{2} \alpha_k^2 L \mathbb{E}_k[\|g_k\|_2^2]. \end{aligned}$$

Markov process: w_{k+1} depends only on w_k and random choice at iteration k.

SG theory

Theorem SG

If $\mathbb{E}_k[\|g_k\|_2^2] \leq M + \|\nabla f(w_k)\|_2^2$, *then:*

$$\alpha_k = \frac{1}{L} \qquad \Longrightarrow \mathbb{E}\left[\frac{1}{k}\sum_{j=1}^{k}\|\nabla f(w_j)\|_2^2\right] \leq M$$

$$\alpha_k = \mathcal{O}\left(\frac{1}{k}\right) \qquad \Longrightarrow \mathbb{E}\left[\sum_{j=1}^{k}\alpha_j\|\nabla f(w_j)\|_2^2\right] < \infty.$$

If, in addition, f *is c-strongly convex, then:*

$$\alpha_k = \frac{1}{L} \qquad \Longrightarrow \mathbb{E}[f(w_k) - f_*] \leq \mathcal{O}\left(\frac{(\alpha L)(M/c)}{2}\right)$$

$$\alpha_k = \mathcal{O}\left(\frac{1}{k}\right) \qquad \Longrightarrow \mathbb{E}[f(w_k) - f_*] = \mathcal{O}\left(\frac{(L/c)(M/c)}{k}\right).$$

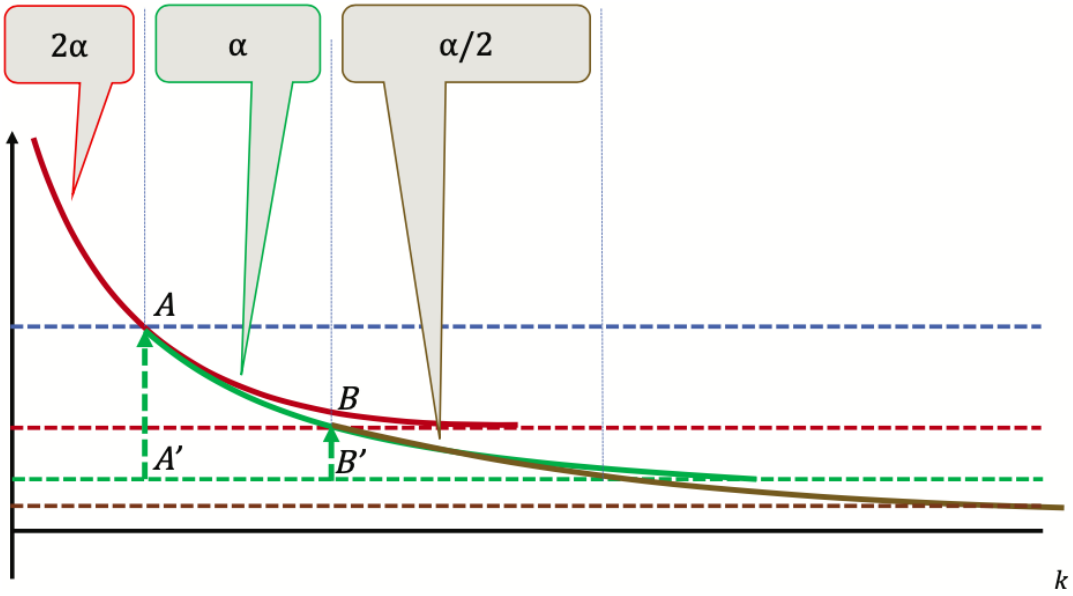
(*Assumed unbiased gradient estimates; see paper for more generality.)

Why $\mathcal{O}(1/k)$?

Mathematically:

$$\sum_{k=1}^{\infty} \alpha_k = \infty \quad \text{while} \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty$$

Graphically (sequential version of constant stepsize result):



SG illustration

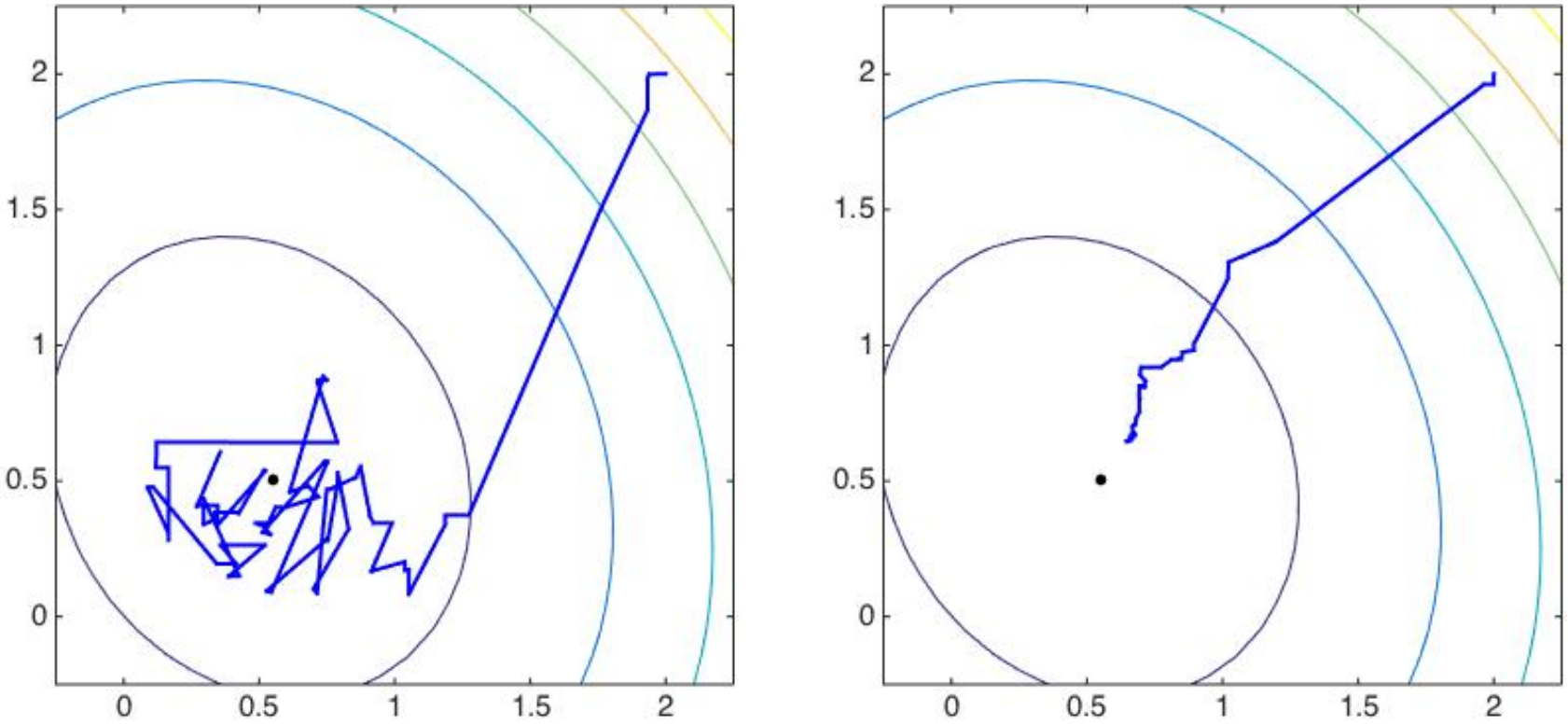


Figure: SG with fixed stepsize (left) vs. diminishing stepsizes (right)

Outline

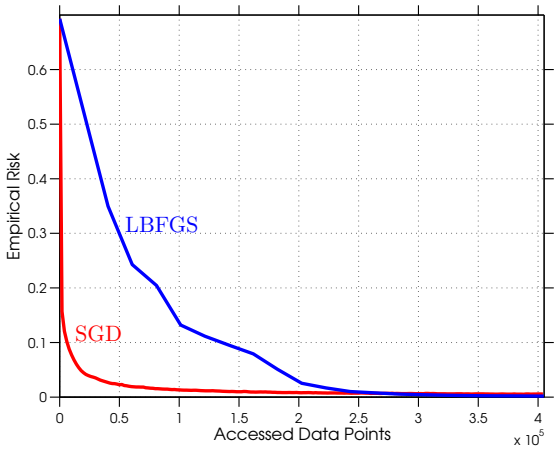
Why SG over GD for large-scale machine learning?

GD: $\mathbb{E}[f_n(w_k) - f_{n,*}] = \mathcal{O}(\rho^k)$ linear convergence

SG: $\mathbb{E}[f_n(w_k) - f_{n,*}] = \mathcal{O}(1/k)$ sublinear convergence

So why SG?

Motivation	Explanation
Intuitive	data "redundancy"
Empirical	SG vs. L-BFGS with batch gradient (below)
Theoretical	$\mathbb{E}[f_n(w_k) - f_{n,*}] = \mathcal{O}(1/k)$ and $\mathbb{E}[f(w_k) - f_*] = \mathcal{O}(1/k)$



Work complexity

Time, not data, as limiting factor; Bottou, Bousquet (2008) and Bottou (2010).

	Convergence rate		Time per iteration		Time for ϵ-optimality
GD:	$\mathbb{E}[f_n(w_k) - f_{n,*}] = \mathcal{O}(\rho^k)$	$+$	$\mathcal{O}(n)$	$\Longrightarrow$	$n \log(1/\epsilon)$
SG:	$\mathbb{E}[f_n(w_k) - f_{n,*}] = \mathcal{O}(1/k)$	$+$	$\mathcal{O}(1)$	$\Longrightarrow$	$1/\epsilon$

Considering total (estimation + optimization) error as

$$\mathcal{E} = \mathbb{E}[f(w^n) - f(w^*)] + \mathbb{E}[f(\tilde{w}^n) - f(w^n)] \sim \tfrac{1}{n} + \epsilon$$

and a time budget $\mathcal{T}$, one finds:

- SG: Process as many samples as possible ($n \sim \mathcal{T}$), leading to

$$\mathcal{E} \sim \frac{1}{\mathcal{T}}.$$

- GD: With $n \sim \mathcal{T}/\log(1/\epsilon)$, minimizing $\mathcal{E}$ yields $\epsilon \sim 1/\mathcal{T}$ and

$$\mathcal{E} \sim \frac{\log(\mathcal{T})}{\mathcal{T}} + \frac{1}{\mathcal{T}}.$$

Outline

End of the story?

SG is great! Let's keep proving how great it is!

- SG is "stable with respect to inputs"
- SG avoids "steep minima"
- SG avoids "saddle points"
- ...(many more)

No, we should want more...

- SG requires a lot of "hyperparameter" tuning
- Sublinear convergence is not satisfactory
- ... "linearly" convergent method eventually wins
- ...with higher budget, faster computation, parallel?, distributed?

Also, any "gradient"-based method is not scale invariant.