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], stochastic gradient 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

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

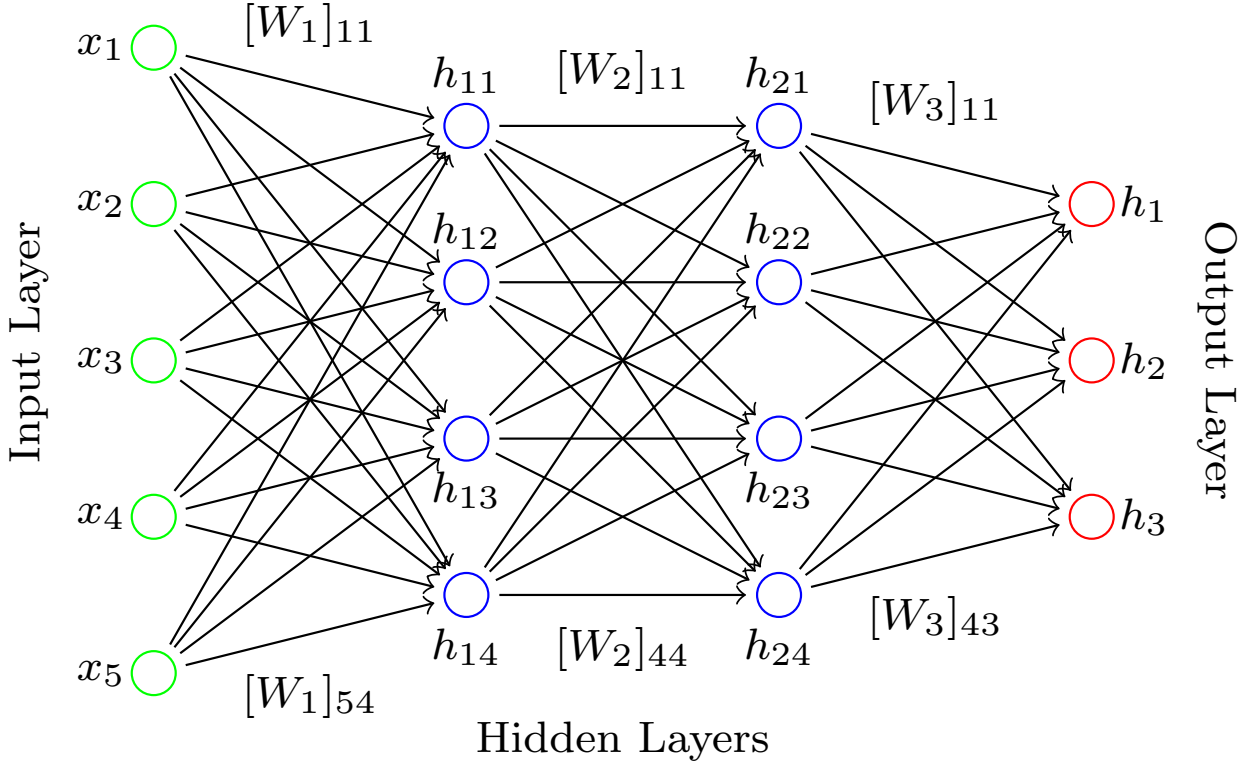


Figure: Illustration of a DNN

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.,

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

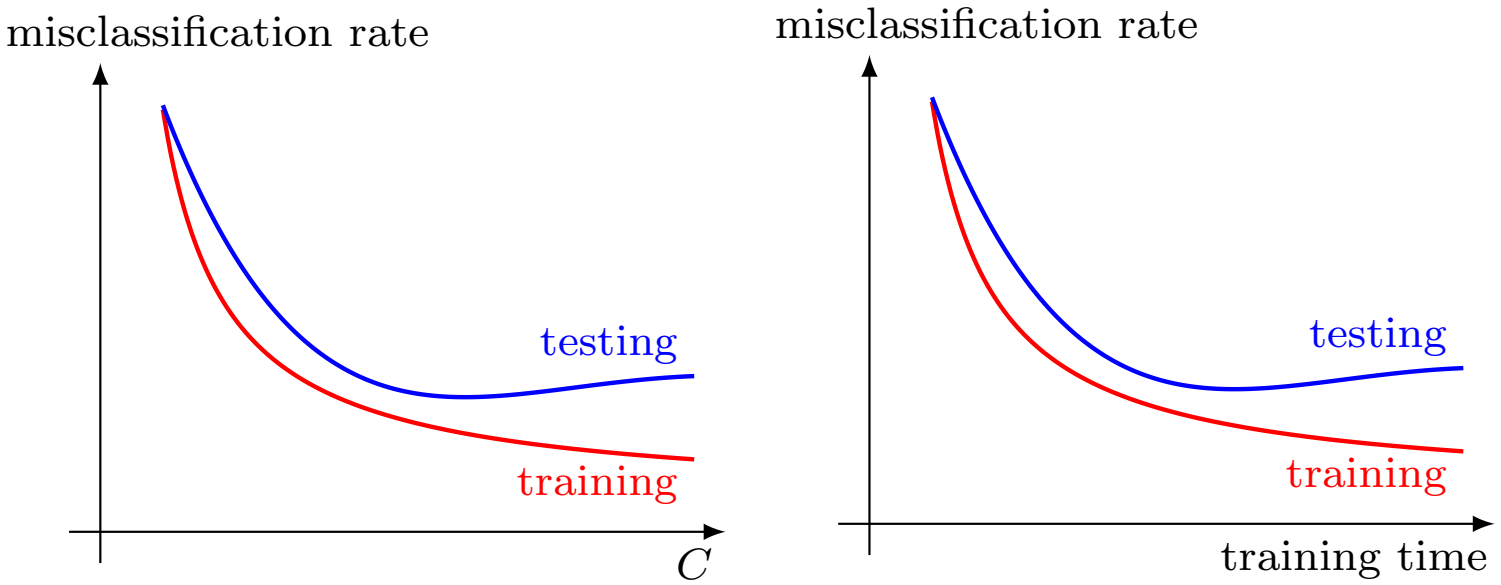


Figure: Illustration of C and training time vs. misclassification rate

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

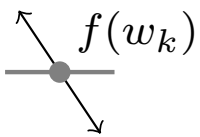
- f is continuously differentiable, bounded below, and potentially nonconvex;
- ∇f is L-Lipschitz continuous, i.e., $\|\nabla f(w) - \nabla f(\overline{w})\|_2 \leq L\|w - \overline{w}\|_2$.

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**

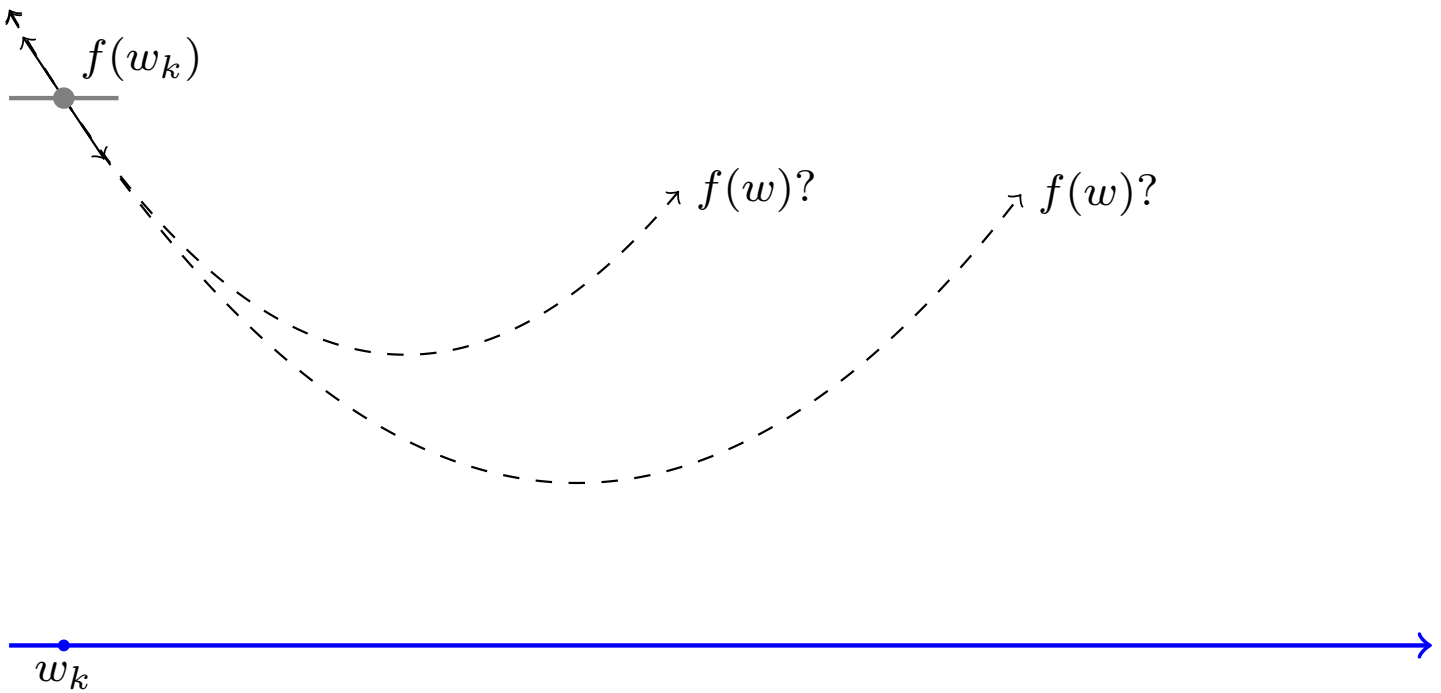


Gradient descent

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**

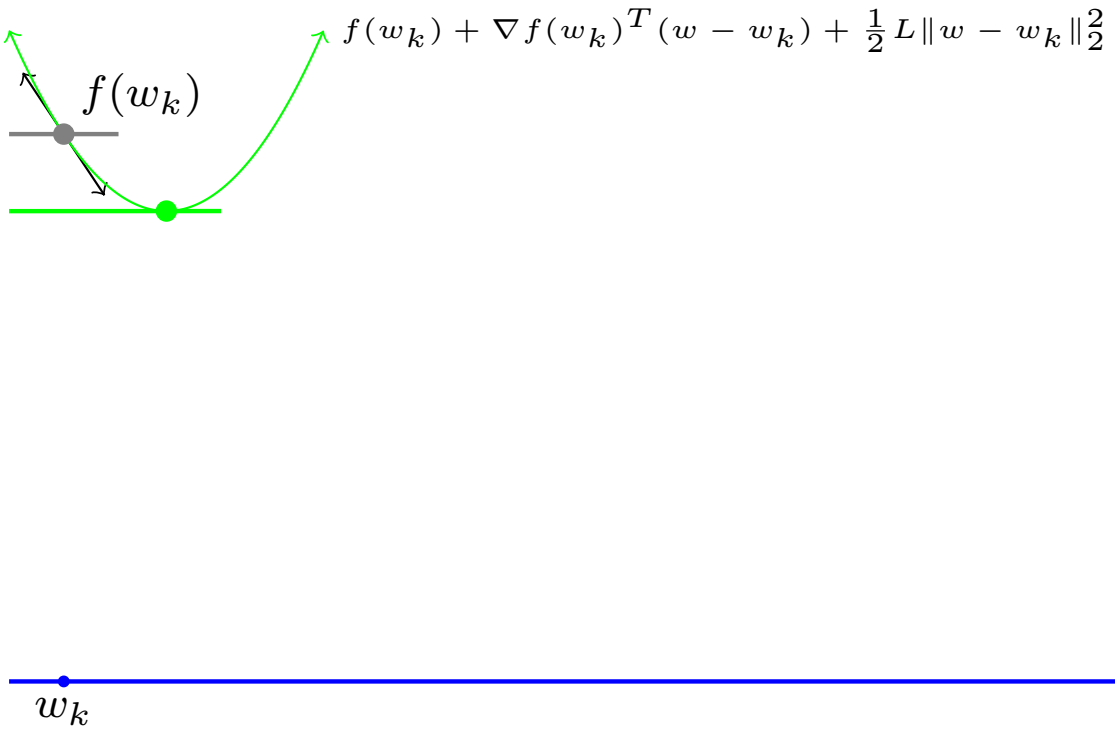


Gradient descent

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**

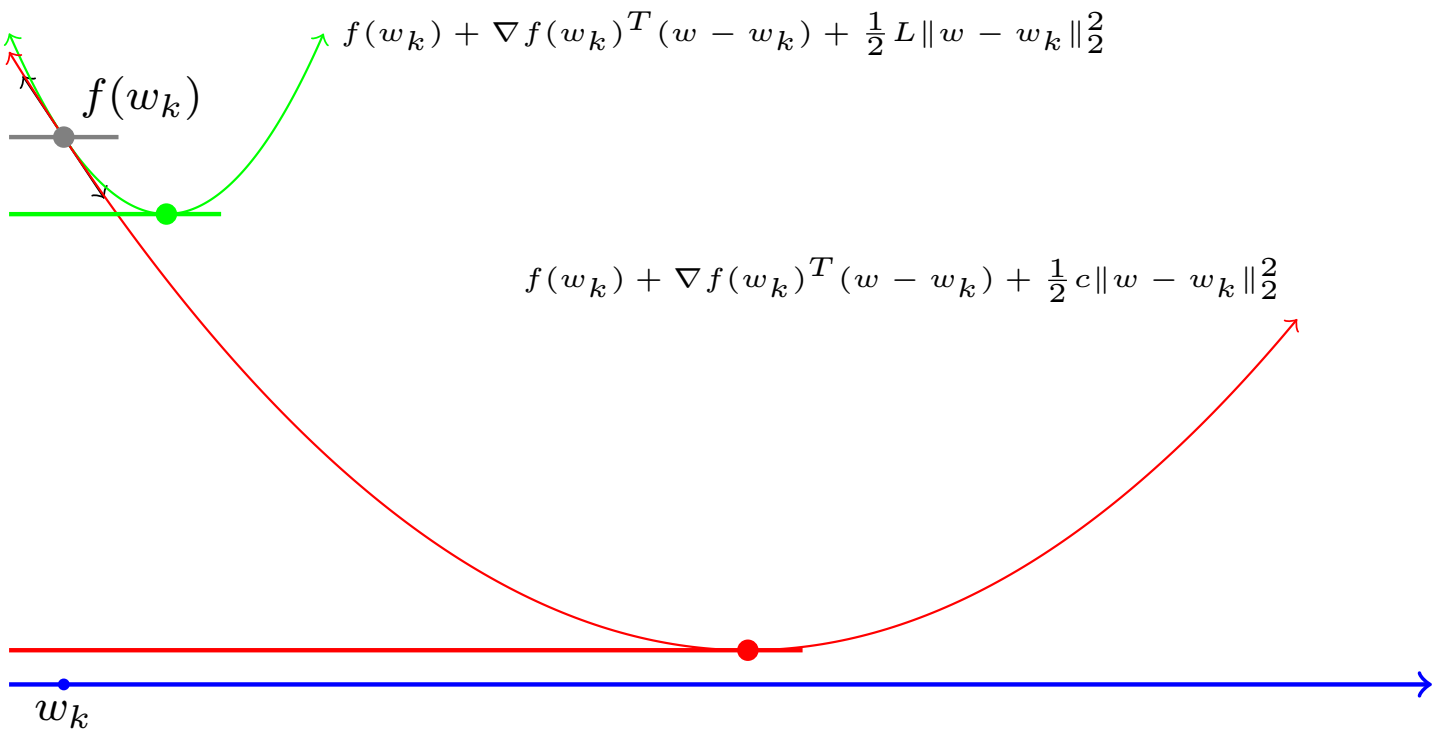


Gradient descent

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

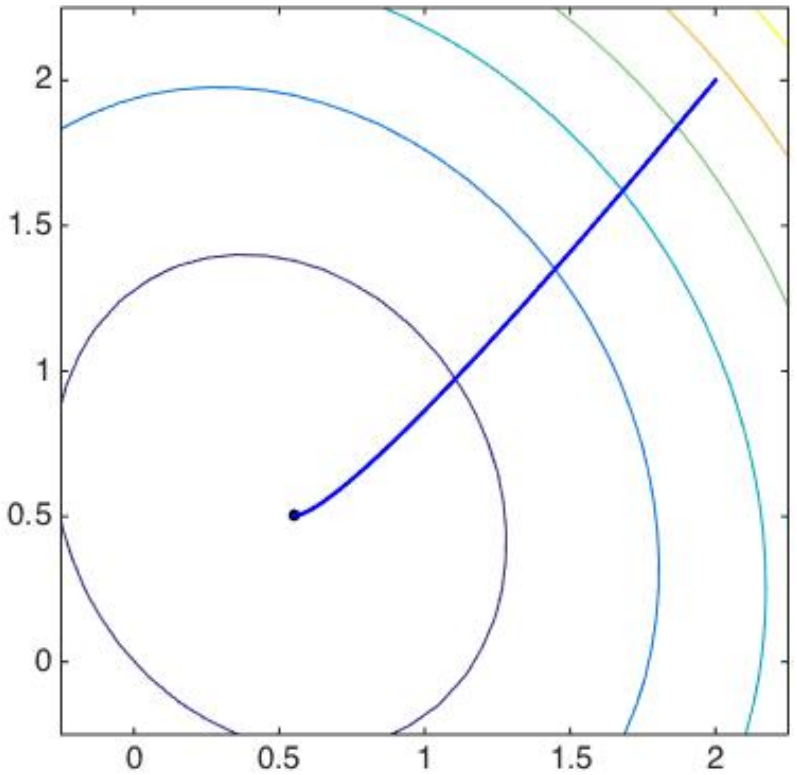


Figure: GD with fixed stepsize

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

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

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

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

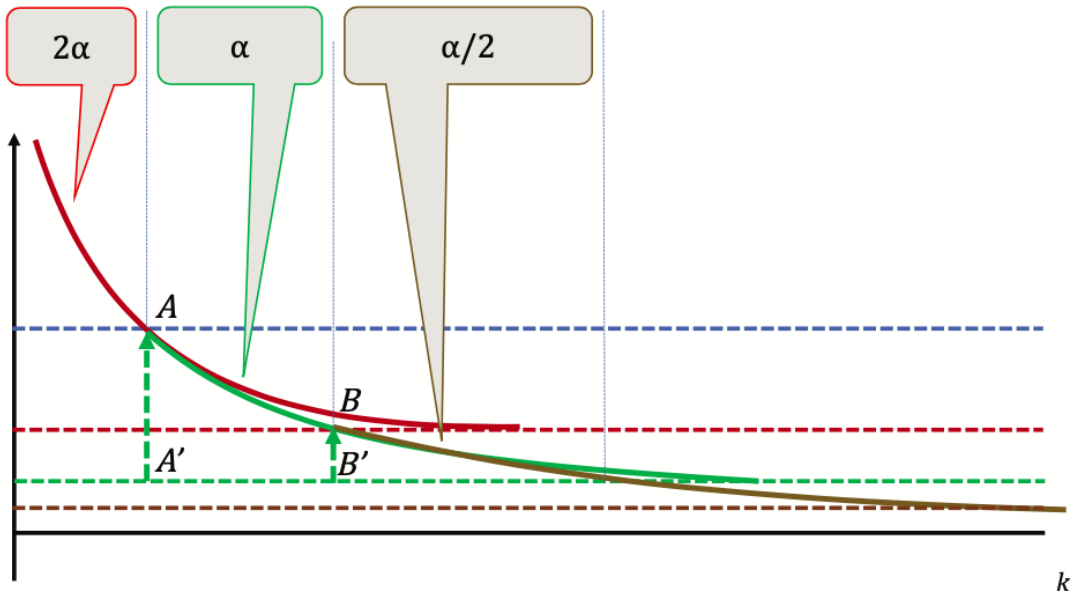
(*Assumed unbiased gradient estimates; see paper for more generality.)

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

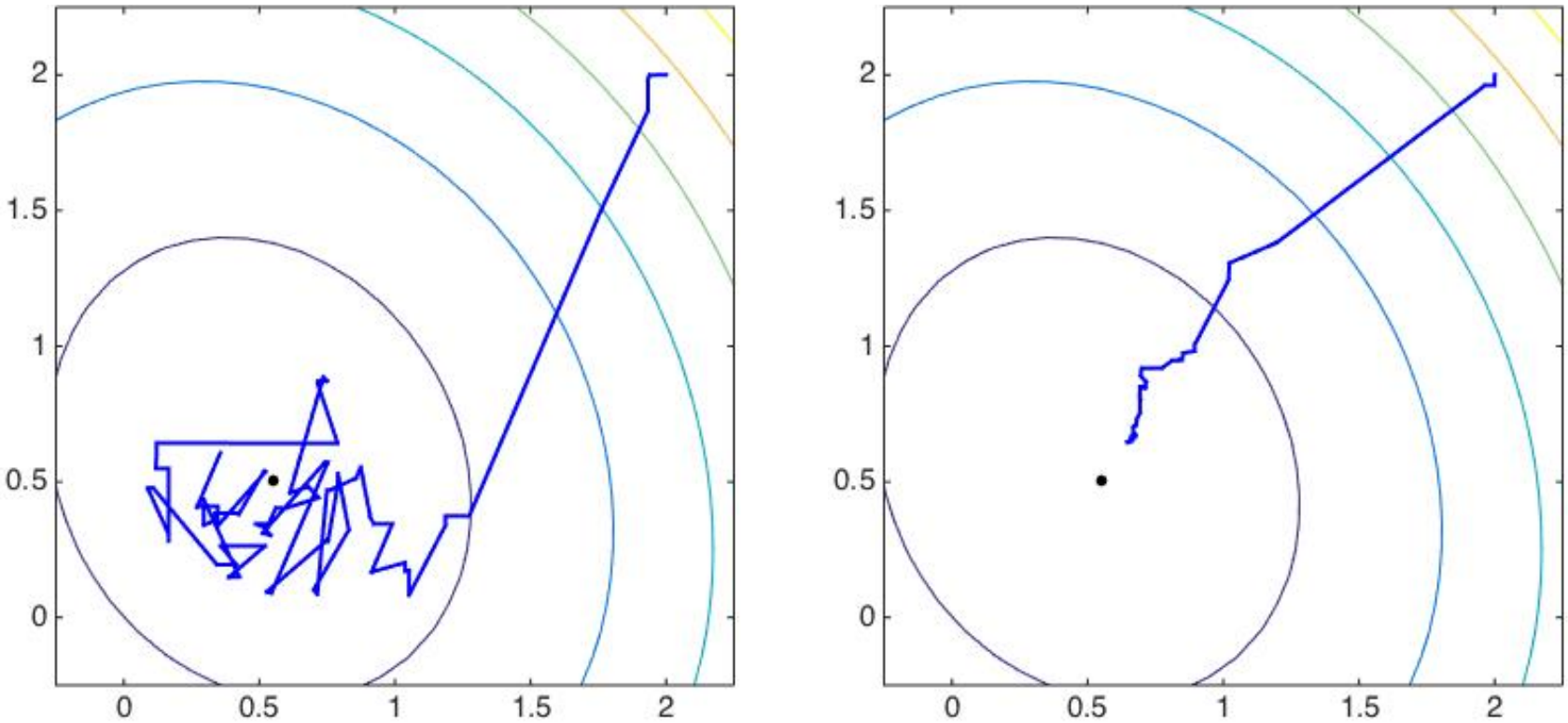


Figure: SG with fixed stepsize (left) vs. diminishing stepsizes (right)

Outline

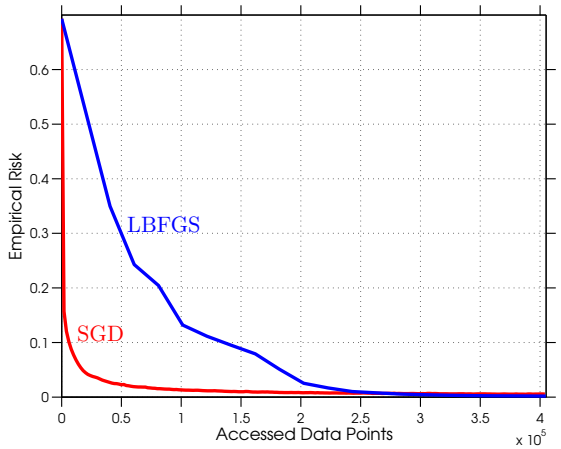
Why SG over GD for large-scale machine learning?

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)

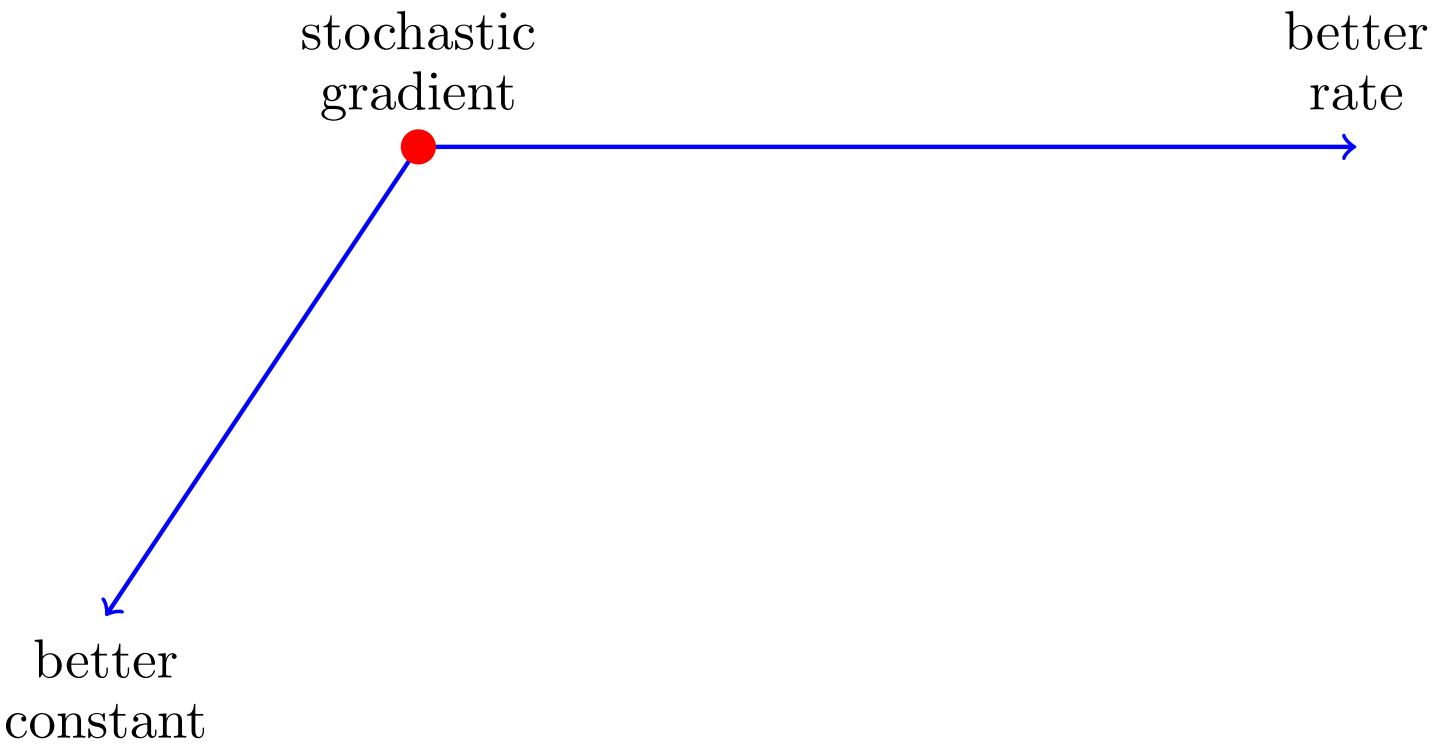
No, we should want more. . .

- 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.

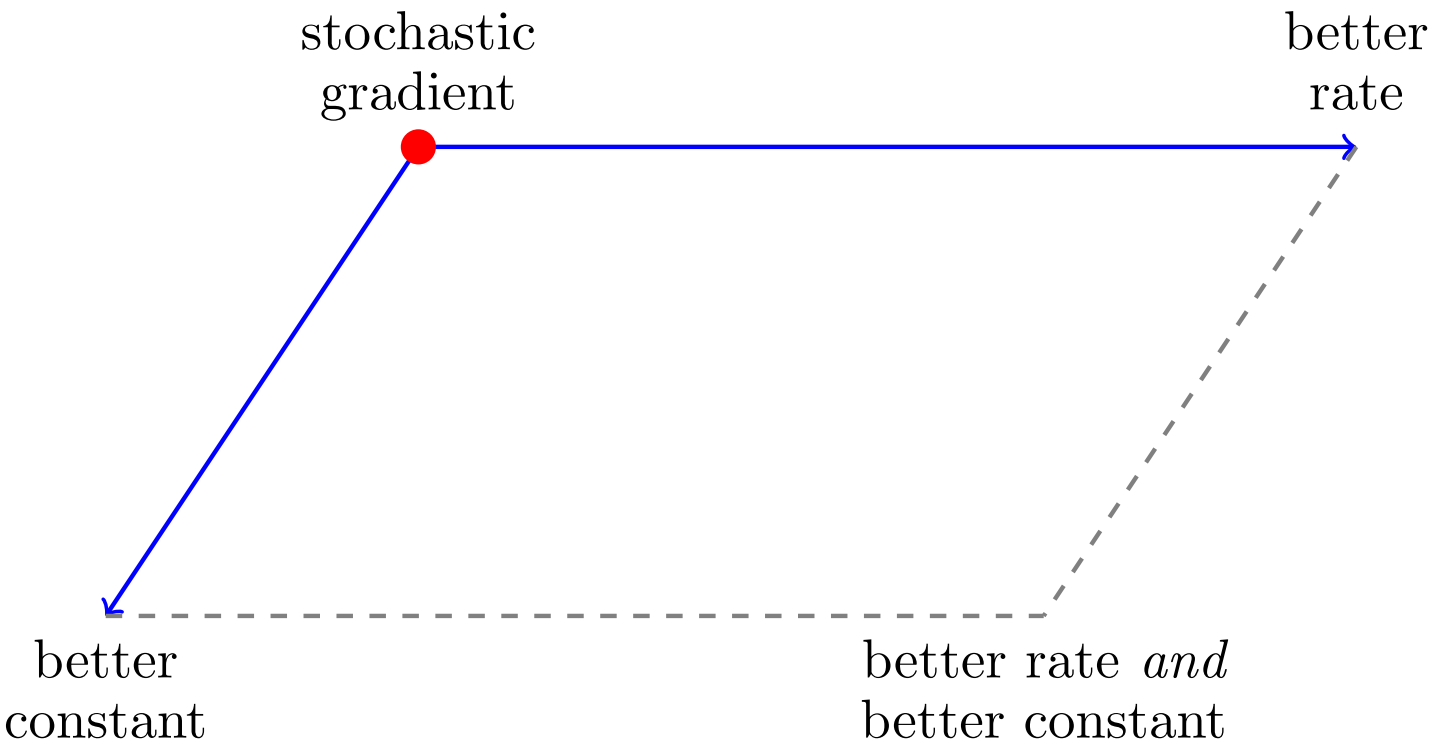
What can be improved?

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

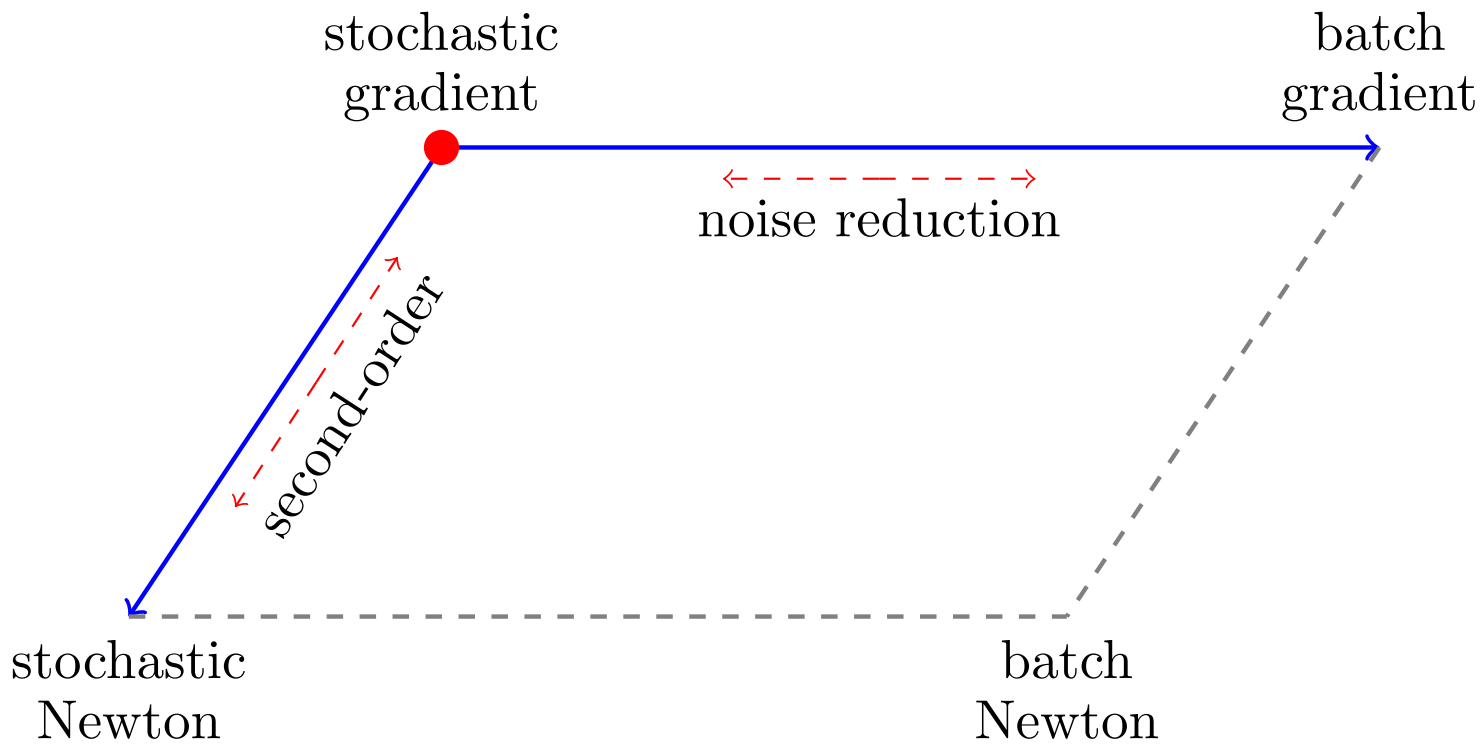


What can be improved?

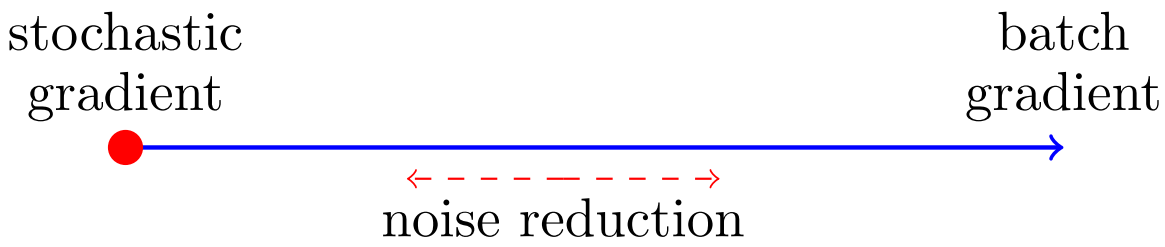
$$\mathbb{E}[f(w_k) - f_*] = \mathcal{O}\left(\frac{(L/c)(M/c)}{k}\right)$$



Two-dimensional schematic of methods

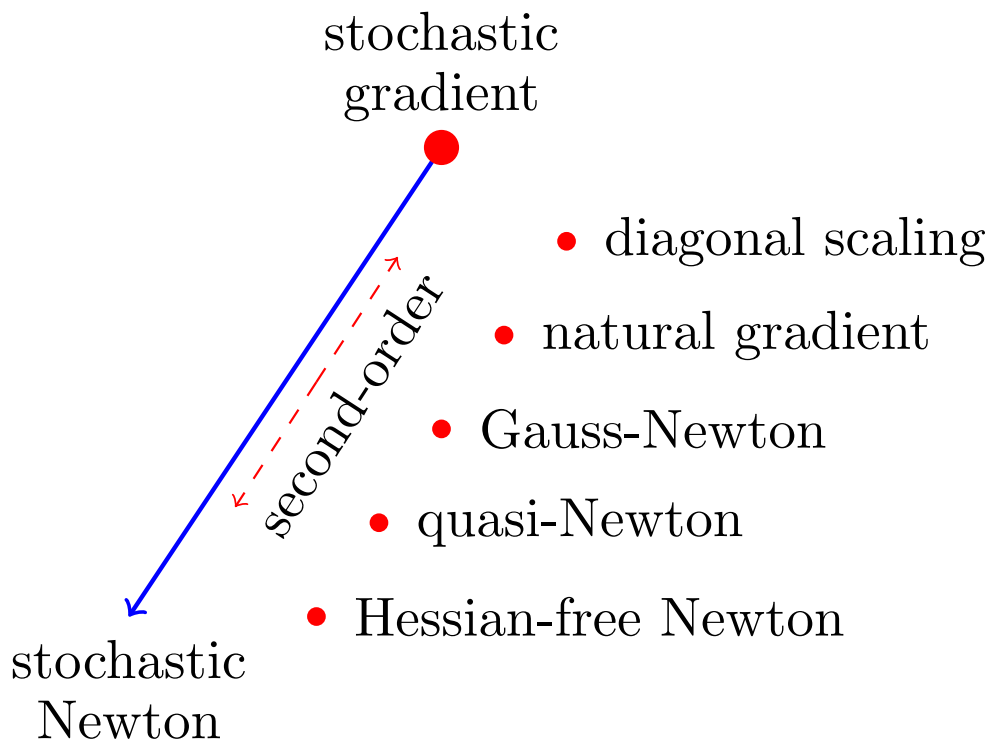


2D schematic: Noise reduction methods



- dynamic sampling
- gradient aggregation
- iterate averaging

2D schematic: Second-order methods



Even more...

- momentum
- acceleration
- (dual) coordinate descent
- trust region / step normalization
- exploring negative curvature
- ...

Outline

Idea #1: Dynamic sampling

We have seen

- fast initial improvement by SG
- long-term linear rate achieved by batch gradient

$\Longrightarrow$ accumulate increasingly accurate gradient information during optimization.

But at what rate?

- too slow: won't achieve linear convergence
- too fast: loss of optimal work complexity

Geometric decrease

Correct balance achieved by decreasing noise at a geometric rate.

Theorem 3

Suppose f is c-strongly convex and L-smooth and that

$$\mathbb{V}_k[g_k] \leq M\zeta^{k-1} \quad \text{for some} \quad M \geq 0 \quad \text{and} \quad \zeta \in (0,1).$$

Then, the SG method with a fixed stepsize $\alpha = 1/L$ yields

$$\mathbb{E}[f(w_k) - f_*] \leq \omega\rho^{k-1},$$

where

$$\omega := \max\left\{\frac{M}{c}, f(w_0) - f_*\right\}$$

$$\text{and} \quad \rho := \max\left\{1 - \frac{c}{2L}, \zeta\right\} < 1.$$

Effectively ties rate of noise reduction with convergence rate of optimization.

Geometric decrease

Proof.

The now-familiar inequality

$$\mathbb{E}_k[f(w_{k+1})] - f(w_k) \leq -\alpha\|\nabla f(w_k)\|_2^2 + \tfrac{1}{2}\alpha^2 L\mathbb{E}_k[\|g_k\|_2^2],$$

strong convexity, and the stepsize choice lead to

$$\mathbb{E}[f(w_{k+1}) - f_*] \leq \left(1 - \frac{c}{L}\right)\mathbb{E}[f(w_k) - f_*] + \frac{M}{2L}\zeta^{k-1}.$$

- Exactly as for batch gradient (in expectation) except for the last term.
- An inductive argument completes the proof.

Practical geometric decrease (unlimited samples)

How can geometric decrease of the variance be achieved in practice?

$$g_k := \frac{1}{|\mathcal{S}_k|} \sum_{i \in \mathcal{S}_k} \nabla f_i(w_k) \quad \text{with} \quad |\mathcal{S}_k| = \lceil \tau^{k-1} \rceil \quad \text{for} \quad \tau > 1,$$

since, for all $i \in \mathcal{S}_k$,

$$\mathbb{V}_k[g_k] \leq \frac{\mathbb{V}_k[\nabla f_i(w_k)]}{|\mathcal{S}_k|} \leq M(\lceil \tau \rceil)^{k-1}.$$

But is it too fast? What about work complexity?

$$\text{same as SG as long as} \quad \tau \in \left(1, \left(1 - \frac{c}{2L}\right)^{-1}\right].$$

Illustration

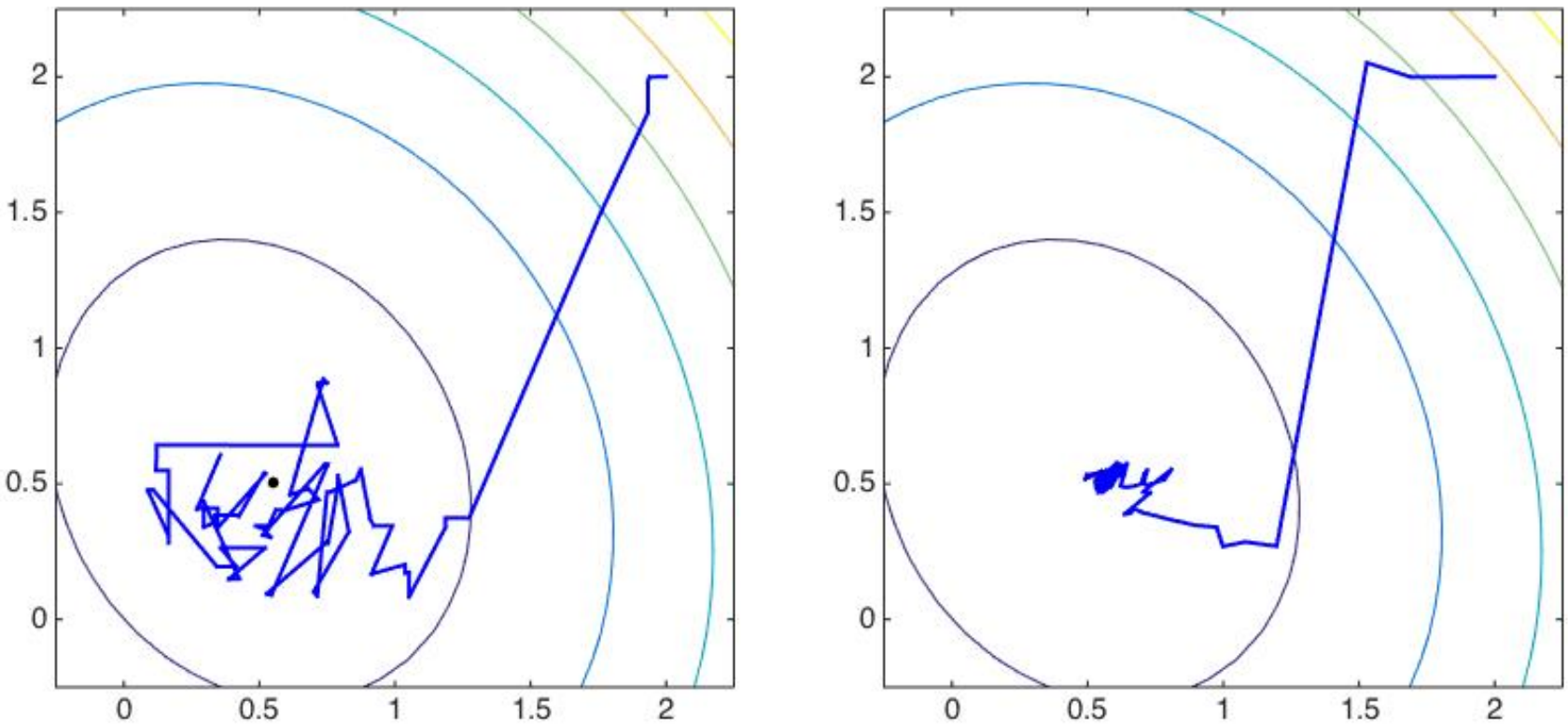


Figure: SG run with a fixed stepsize (left) vs. dynamic SG with fixed stepsize (right)

Additional considerations

In practice, choosing τ is a challenge.

- What about an adaptive technique?
- Guarantee descent in expectation
- Methods exist, but need geometric sample size increase as backup

Idea #2: Gradient aggregation

"I'm minimizing a finite sum and am willing to store previous gradient(s)."

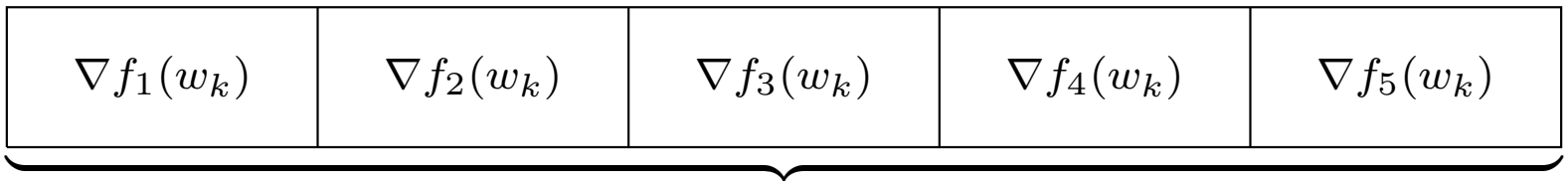
$$F(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w).$$

Idea: reuse and/or revise previous gradient information in storage.

- SVRG: store full gradient, correct sequence of steps based on perceived bias
- SAGA: store *elements* of full gradient, revise as optimization proceeds
- SARAH: stochastic recursive gradient method

Stochastic variance reduced gradient (SVRG) method

At $w_k =: w_{k,1}$, compute a batch gradient:

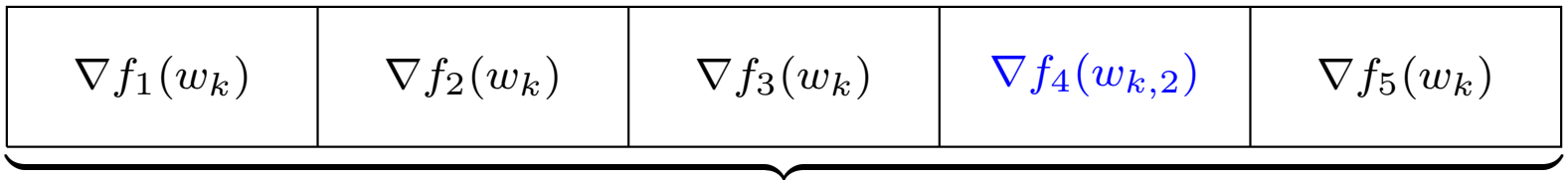


then step

$$w_{k,2} \leftarrow w_{k,1} - \alpha g_{k,1}$$

Stochastic variance reduced gradient (SVRG) method

Now, iteratively, choose an index *randomly* and correct bias:



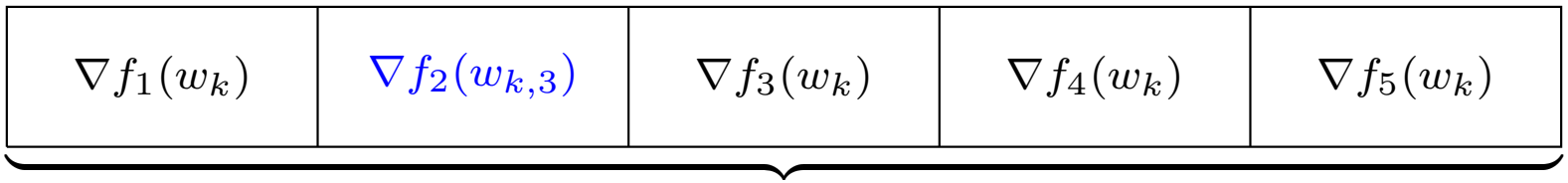
$$g_{k,2} \leftarrow \nabla F(w_k) - \nabla f_4(w_k) + \nabla f_4(w_{k,2})$$

then step

$$w_{k,3} \leftarrow w_{k,2} - \alpha g_{k,2}$$

Stochastic variance reduced gradient (SVRG) method

Now, iteratively, choose an index *randomly* and correct bias:



$$g_{k,3} \leftarrow \nabla F(w_k) - \nabla f_2(w_k) + \nabla f_2(w_{k,3})$$

then step

$$w_{k,4} \leftarrow w_{k,3} - \alpha g_{k,3}$$

Stochastic variance reduced gradient (SVRG) method

Each $g_{k,j}$ is an unbiased estimate of $\nabla F(w_{k,j})$!

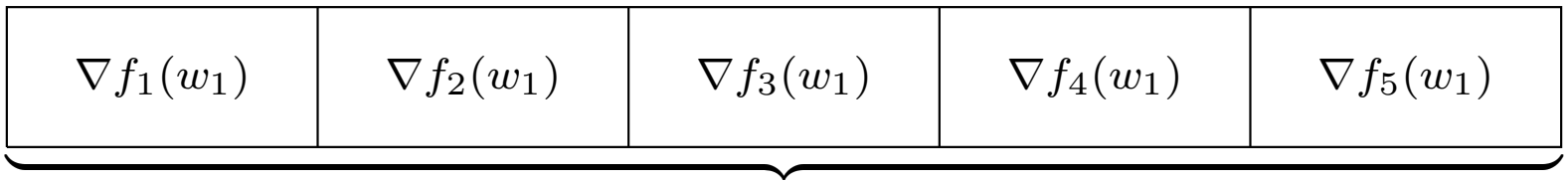
Algorithm SVRG

Choose an initial iterate $w_1 \in \mathbb{R}^d$, stepsize $\alpha > 0$, and positive integer m.
for $k = 1, 2, \ldots$ **do**
  Compute the batch gradient $\nabla F(w_k)$.
  Initialize $w_{k,1} \leftarrow w_k$.
  for $j = 1, \ldots, m$ **do**
    Chose i uniformly from $\{1, \ldots, n\}$.
    Set $g_{k,j} \leftarrow \nabla f_i(w_{k,j}) - (\nabla f_i(w_k) - \nabla F(w_k))$.
    Set $w_{k,j+1} \leftarrow w_{k,j} - \alpha g_{k,j}$.
  end for
  Option (a): Set $w_{k+1} = \tilde{w}_{m+1}$
  Option (b): Set $w_{k+1} = \frac{1}{m} \sum_{j=1}^{m} \tilde{w}_{j+1}$
  Option (c): Choose j uniformly from $\{1, \ldots, m\}$ and set $w_{k+1} = \tilde{w}_{j+1}$.
end for

If f is c-strongly convex and L-smooth, then options (b) and (c) are linearly convergent for certain (α, m)

Stochastic average gradient (SAGA) method

At w_1, compute a batch gradient:



$$g_1 \leftarrow \nabla F(w_1)$$

then step

$$w_2 \leftarrow w_1 - \alpha g_1$$

Stochastic average gradient (SAGA) method

Now, iteratively, choose an index *randomly* and revise table entry:

$\nabla f_1(w_1)$	$\nabla f_2(w_1)$	$\nabla f_3(w_1)$	$\nabla f_4(w_2)$	$\nabla f_5(w_1)$

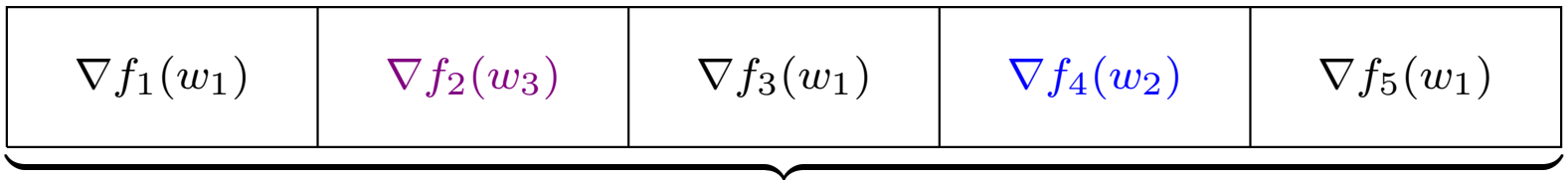
$g_2 \leftarrow$ new entry $-$ old entry $+$ average of entries (before replacement)

then step

$$w_3 \leftarrow w_2 - \alpha g_2$$

Stochastic average gradient (SAGA) method

Now, iteratively, choose an index *randomly* and revise table entry:



$g_3 \leftarrow$ new entry $-$ old entry $+$ average of entries (before replacement)

then step

$$w_4 \leftarrow w_3 - \alpha g_3$$

Stochastic average gradient (SAGA) method

Each g_k is an unbiased estimate of $\nabla F(w_k)$!

Algorithm SAGA

Choose an initial iterate $w_1 \in \mathbb{R}^d$ and stepsize $\alpha > 0$.
for $i = 1, \ldots, n$ **do**
    Compute $\nabla f_i(w_1)$.
    Store $\nabla f_i(w_{[i]}) \leftarrow \nabla f_i(w_1)$.
end for
for $k = 1, 2, \ldots$ **do**
    Choose j uniformly in $\{1, \ldots, n\}$.
    Compute $\nabla f_j(w_k)$.
    Set $g_k \leftarrow \nabla f_j(w_k) - \nabla f_j(w_{[j]}) + \frac{1}{n} \sum_{i=1}^n \nabla f_i(w_{[i]})$.
    Store $\nabla f_j(w_{[j]}) \leftarrow \nabla f_j(w_k)$.
    Set $w_{k+1} \leftarrow w_k - \alpha g_k$.
end for

If f is c-strongly convex and L-smooth, then linearly convergent for certain α

- storage of gradient vectors reasonable in some applications
- with access to feature vectors, need only store n scalars

Idea #3: Iterative averaging

Averages of SG iterates are less noisy:

$$w_{k+1} \leftarrow w_k - \alpha_k g_k$$

$$\tilde{w}_{k+1} \leftarrow \frac{1}{k+1} \sum_{j=1}^{k+1} w_j \quad \text{(in practice: running average)}$$

Unfortunately, no better theoretically when $\alpha_k = \mathcal{O}(1/k)$, but

- long steps (say, $\alpha_k = \mathcal{O}(1/\sqrt{k})$) *and* averaging
- lead to a better sublinear rate (like a second-order method?)

See also

- mirror descent
- primal-dual averaging

Idea #3: Iterative averaging

Averages of SG iterates are less noisy:

$$w_{k+1} \leftarrow w_k - \alpha_k g_k$$

$$\tilde{w}_{k+1} \leftarrow \frac{1}{k+1} \sum_{j=1}^{k+1} w_j \quad \text{(in practice: running average)}$$

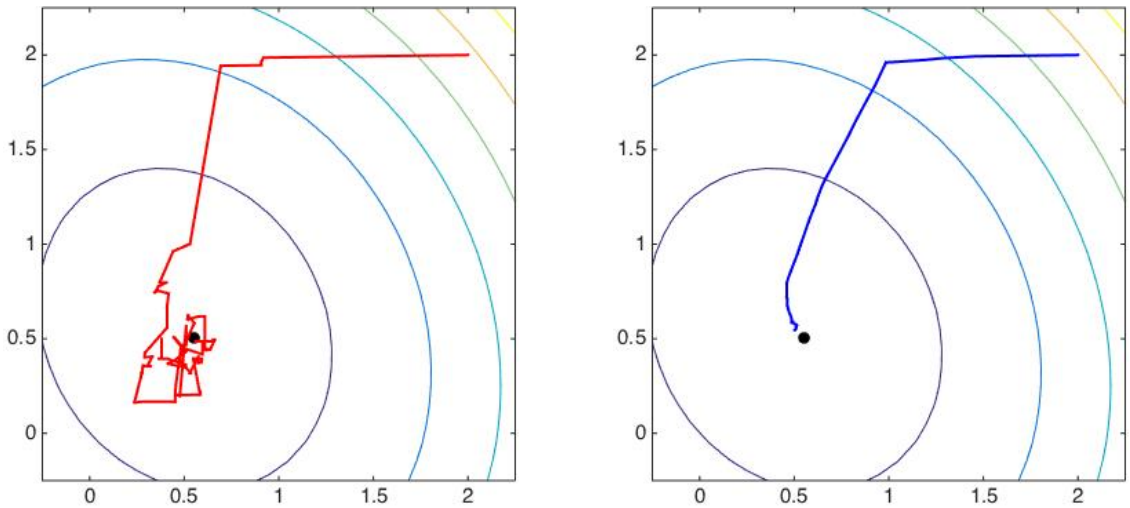
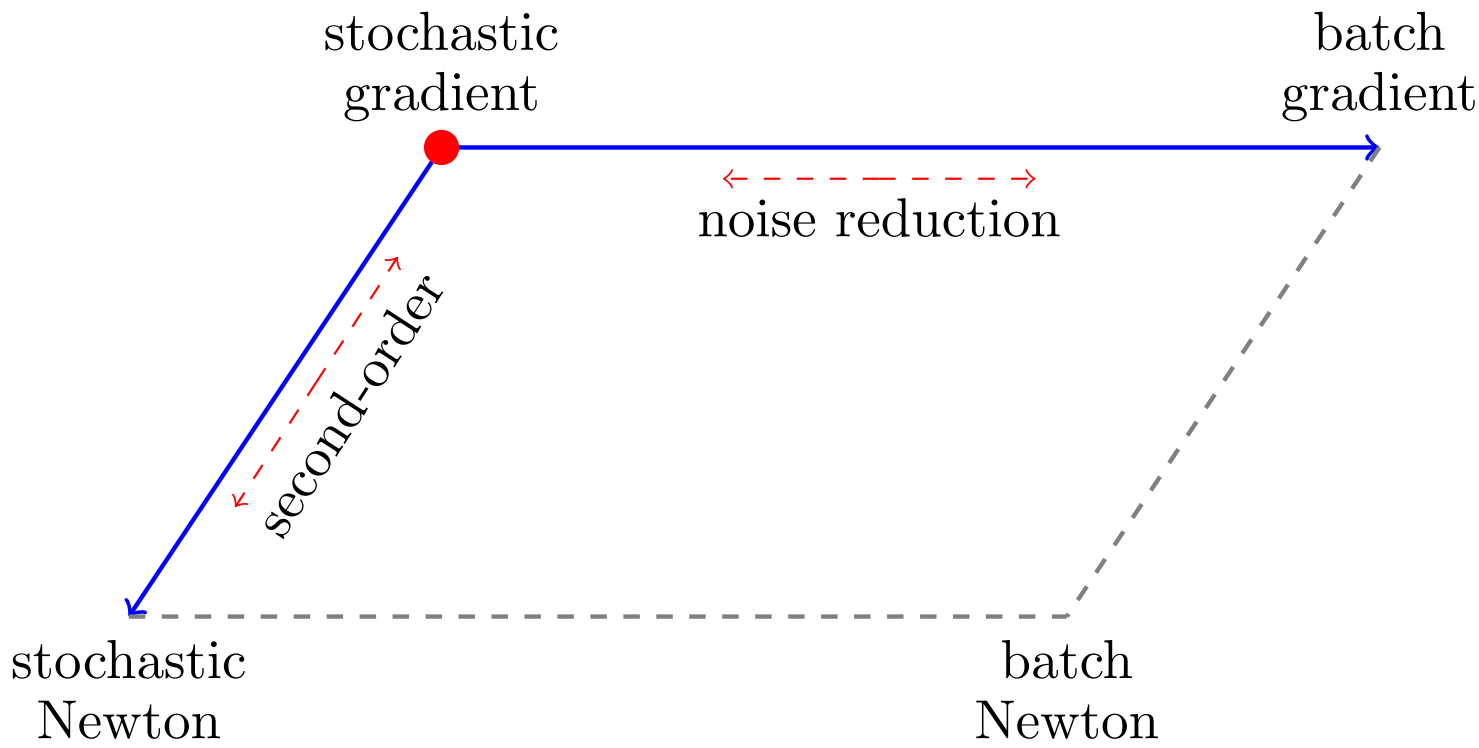


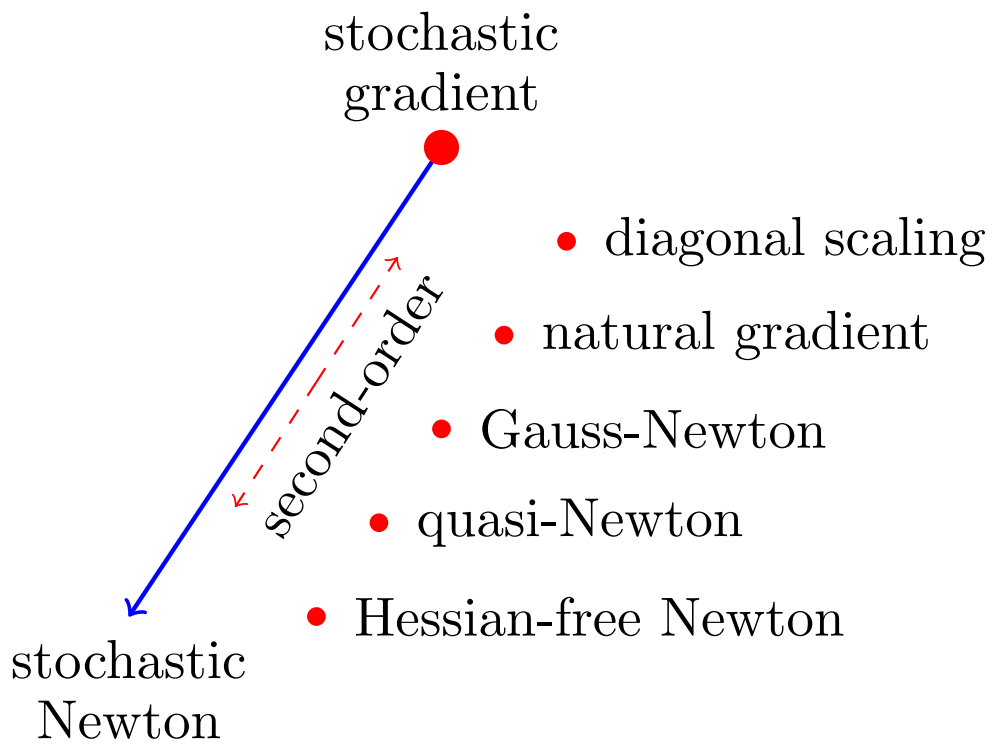
Figure: SG run with $\mathcal{O}(1/\sqrt{k})$ stepsizes (left) vs. sequence of averages (right)

Outline

Two-dimensional schematic of methods



2D schematic: Second-order methods



Ideal: Scale invariance

Neither SG nor batch gradient are invariant to linear transformations!

$$\min_{w \in \mathbb{R}^d} f(w) \quad \Longrightarrow \quad w_{k+1} \leftarrow w_k - \alpha_k \nabla f(w_k)$$

$$\min_{\tilde{w} \in \mathbb{R}^d} f(B\tilde{w}) \quad \Longrightarrow \quad \tilde{w}_{k+1} \leftarrow \tilde{w}_k - \alpha_k B \nabla f(B\tilde{w}_k) \quad \text{(for given } B \succ 0)$$

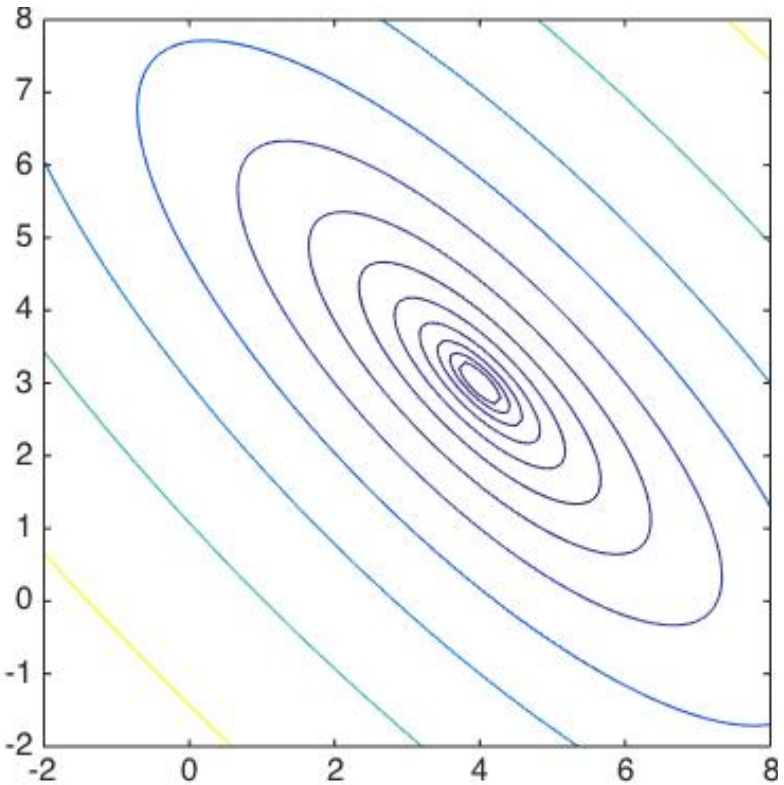
Scaling latter by B and defining $\{w_k\} = \{B\tilde{w}_k\}$ yields

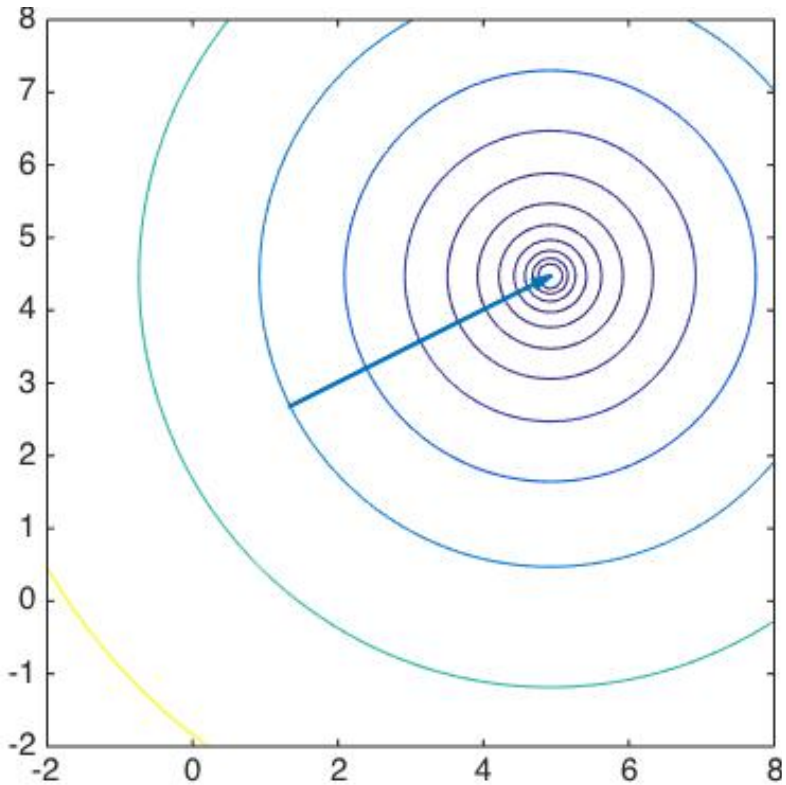
$$w_{k+1} \leftarrow w_k - \alpha_k B^2 \nabla f(w_k)$$

- Algorithm is clearly affected by choice of B
- Surely, some choices may be better than others (in general?)

Newton scaling

Consider the function below and suppose that $w_k = (0, 3)$:

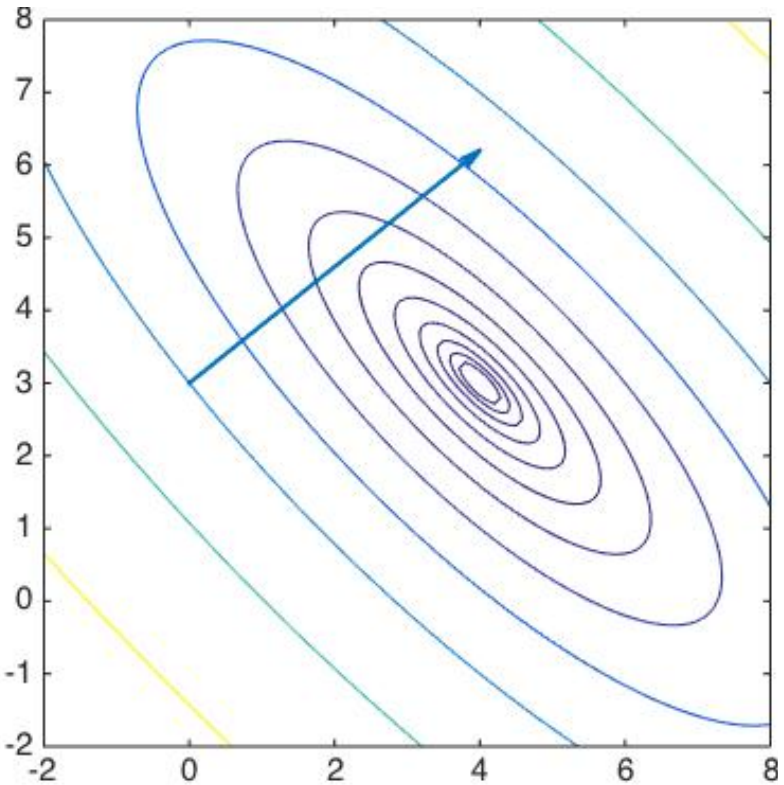


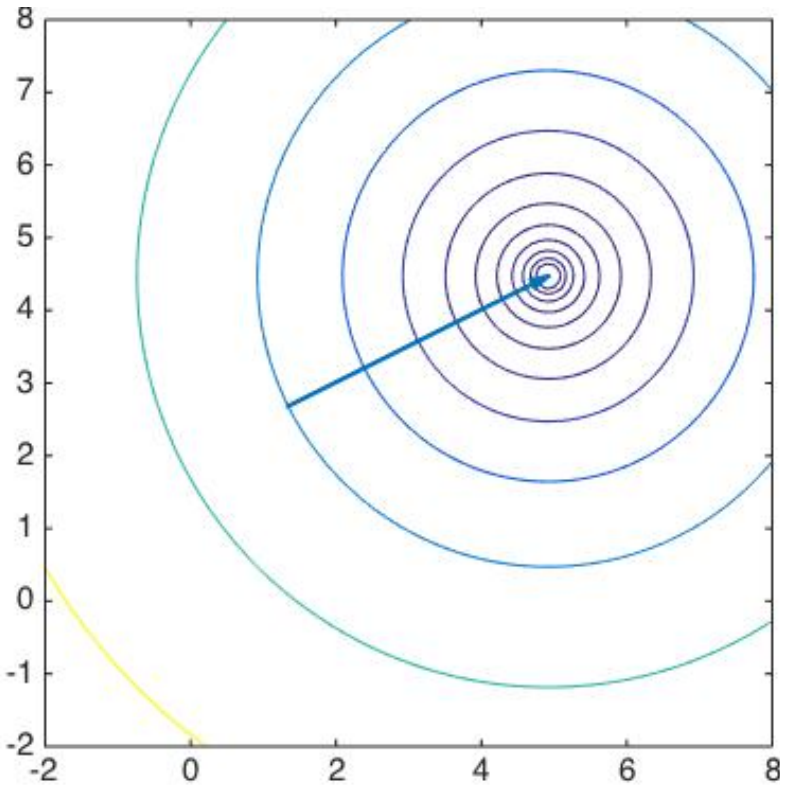


$$w_{k+1} \leftarrow w_k + \alpha_k s_k \quad \text{where} \quad \nabla^2 f(w_k) s_k = -\nabla f(w_k)$$

Newton scaling

Batch gradient step $-\alpha_k \nabla f(w_k)$ ignores curvature of the function:

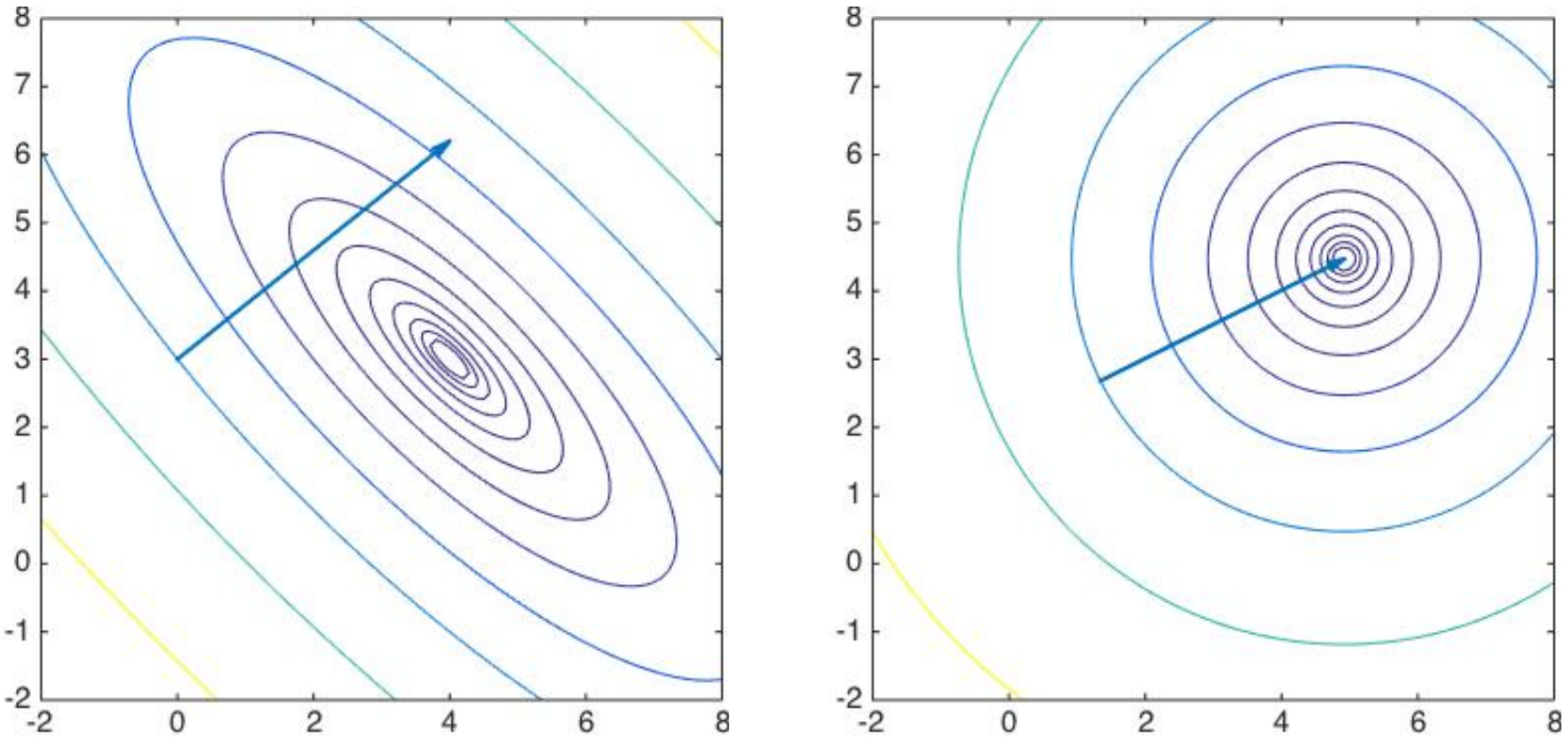




$$w_{k+1} \leftarrow w_k + \alpha_k s_k \quad \text{where} \quad \nabla^2 f(w_k) s_k = -\nabla f(w_k)$$

Newton scaling

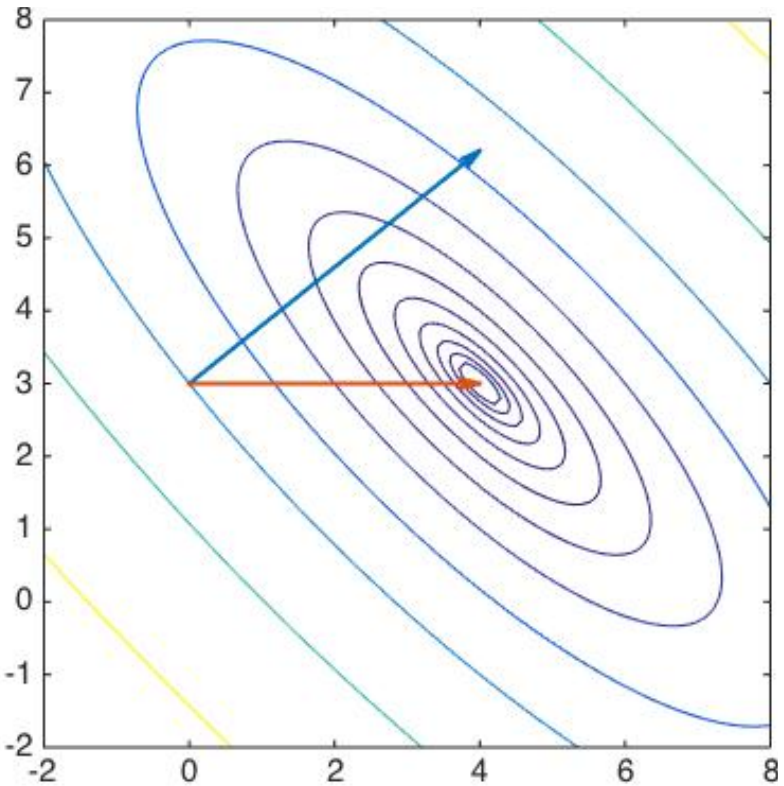
Newton scaling ($B = (\nabla f(w_k))^{-1/2}$): gradient step moves to the minimizer:

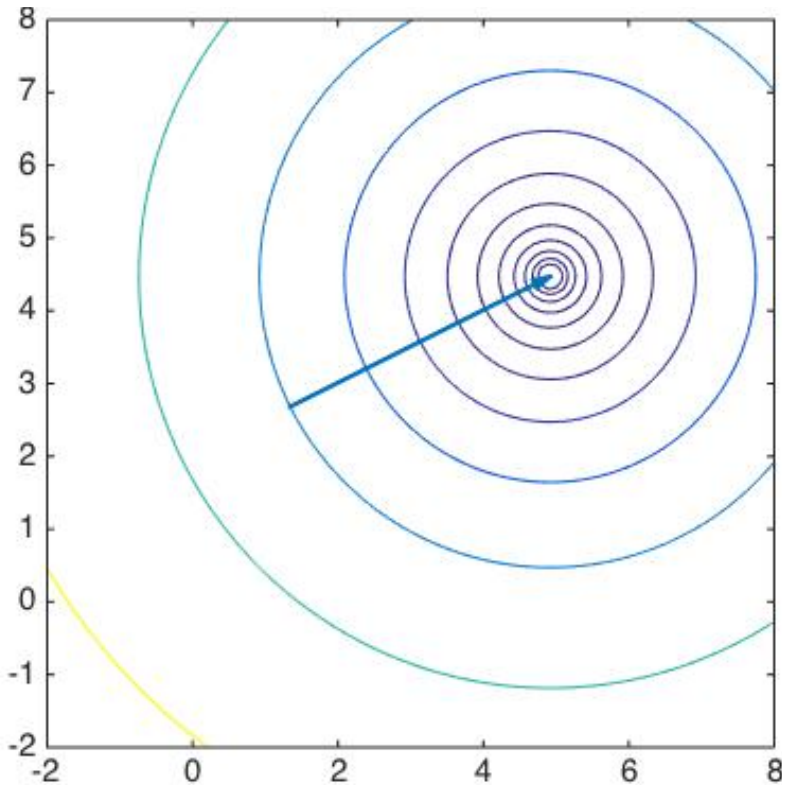


$$w_{k+1} \leftarrow w_k + \alpha_k s_k \quad \text{where} \quad \nabla^2 f(w_k) s_k = -\nabla f(w_k)$$

Newton scaling

. . . corresponds to minimizing a quadratic model of f in the original space:





$$w_{k+1} \leftarrow w_k + \alpha_k s_k \quad \text{where} \quad \nabla^2 f(w_k) s_k = -\nabla f(w_k)$$

Deterministic case to stochastic case

What is known about Newton's method for deterministic optimization?

- local rescaling based on inverse Hessian information
- locally quadratically convergent near a strong minimizer
- global convergence rate better than gradient method (*when regularized*)

However, it is way too expensive in our case.

- But all is not lost: scaling is viable.
- Wide variety of scaling techniques improve performance.
- Our convergence theory for SG still holds with B-scaling.
- . . . could hope to remove condition number (L/c) from convergence rate!
- Added costs can be minimal when coupled with noise reduction.

Idea #1: Inexact Hessian-free Newton

Compute Newton-like step

$$\nabla^2 f_{\mathcal{S}_k^H}(w_k) s_k = -\nabla f_{\mathcal{S}_k^g}(w_k)$$

- mini-batch size for Hessian =: $|\mathcal{S}_k^H| < |\mathcal{S}_k^g|$:= mini-batch size for gradient
- cost for mini-batch gradient: g_{cost}
- use CG and terminate early: max_{cg} iterations
- in CG, cost for each Hessian-vector product: $factor \times g_{cost}$
- choose $max_{cg} \times factor \approx$ small constant so total per-iteration cost:

$$max_{cg} \times factor \times g_{cost} = \mathcal{O}(g_{cost})$$

- convergence guarantees for $|\mathcal{S}_k^H| = |\mathcal{S}_k^g| = n$ are well-known

Idea #2: (Generalized) Gauss-Newton

Classical approach for nonlinear least squares, linearize inside of loss/cost:

$$\begin{aligned} f(w;\xi) &= \tfrac{1}{2}\|h(x_\xi;w) - y_\xi\|_2^2 \\ &\approx \tfrac{1}{2}\|h(x_\xi;w_k) + J_h(w_k;\xi)(w-w_k) - y_\xi\|_2^2 \end{aligned}$$

Leads to Gauss-Newton approximation for second-order terms:

$$G_{\mathcal{S}_k^H}(w_k;\xi_k^H) = \frac{1}{|\mathcal{S}_k^H|} J_h(w_k;\xi_{k,i})^T J_h(w_k;\xi_{k,i})$$

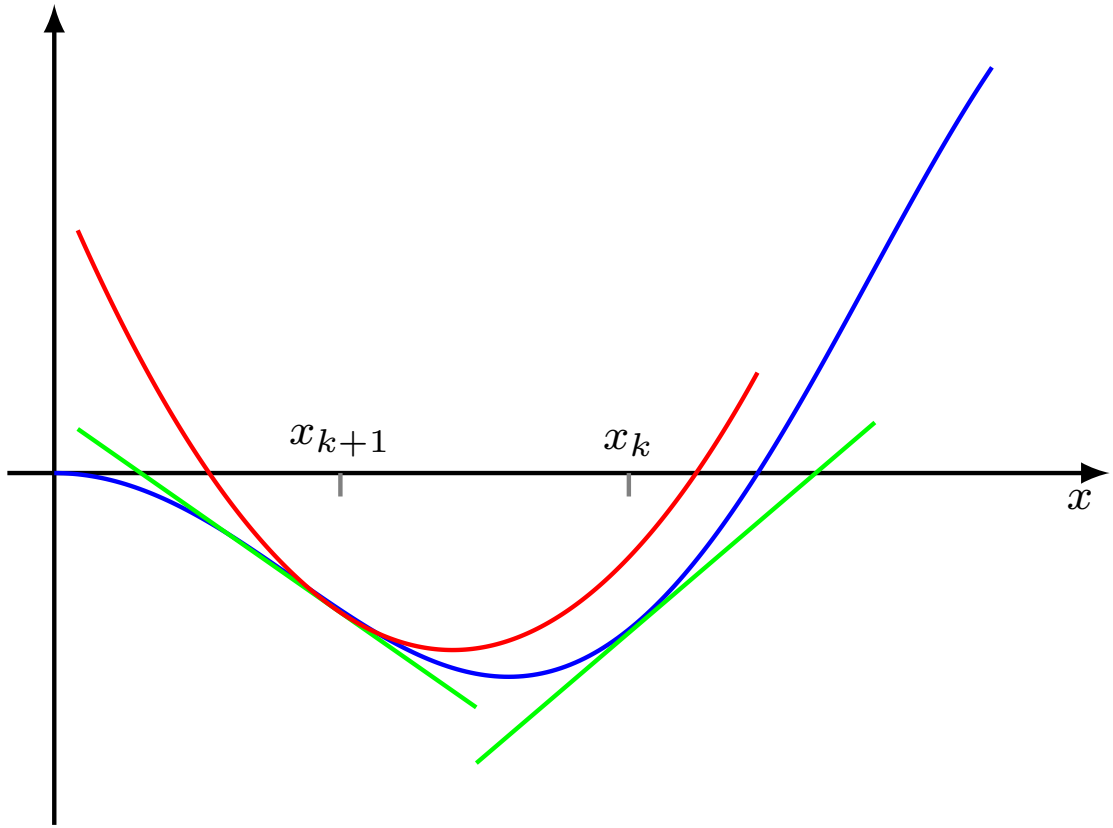
Can be generalized for other (convex) losses:

$$\widetilde{G}_{\mathcal{S}_k^H}(w_k;\xi_k^H) = \frac{1}{|\mathcal{S}_k^H|} J_h(w_k;\xi_{k,i})^T \underbrace{H_\ell(w_k;\xi_{k,i})}_{=\frac{\partial^2 \ell}{\partial h^2}} J_h(w_k;\xi_{k,i})$$

- costs similar as for inexact Newton
- ... but scaling matrices are always positive (semi)definite
- see also *natural gradient*, invariant to more than just linear transformations

Idea #3: (Limited memory) quasi-Newton

Only *approximate* second-order information with gradient displacements:



Secant equation $H_k v_k = s_k$ to match gradient of f at w_k, where

$$s_k := w_{k+1} - w_k \quad \text{and} \quad v_k := \nabla f(w_{k+1}) - \nabla f(w_k)$$

Deterministic case to stochastic case

Standard update for inverse Hessian ($w_{k+1} \leftarrow w_k - \alpha_k H_k g_k$) is BFGS:

$$H_{k+1} \leftarrow \left(I - \frac{v_k s_k^T}{s_k^T v_k}\right)^T H_k \left(I - \frac{v_k s_k^T}{s_k^T v_k}\right) + \frac{s_k s_k^T}{s_k^T v_k}$$

What is known about quasi-Newton methods for deterministic optimization?

- local rescaling based on iterate/gradient displacements
- strongly convex function $\implies$ positive definite (p.d.) matrices
- only first-order derivatives, no linear system solves
- locally superlinearly convergent near a strong minimizer

Extended to stochastic case? How?

- Noisy gradient estimates $\implies$ challenge to maintain p.d.
- Correlation between gradient and Hessian estimates
- Overwriting updates $\implies$ poor scaling that plagues!

Proposed methods

- gradient displacements using same sample:

$$v_k := \nabla f_{\mathcal{S}_k}(w_{k+1}) - \nabla f_{\mathcal{S}_k}(w_k)$$

  (requires two stochastic gradients per iteration)
- gradient displacement replaced by action on subsampled Hessian:

$$v_k := \nabla^2 f_{\mathcal{S}_k^H}(w_k)(w_{k+1} - w_k)$$

- decouple iteration and Hessian update to amortize added cost
- limited memory approximations (e.g., L-BFGS) with per-iteration cost $4md$

Idea #4: Diagonal scaling

Restrict added costs through only diagonal scaling:

$$w_{k+1} \leftarrow w_k - \alpha_k D_k g_k$$

Ideas:

- $D_k^{-1} \approx$ diag(Hessian (approximation))
- $D_k^{-1} \approx$ diag(Gauss-Newton approximation)
- $D_k^{-1} \approx$ running average/sum of gradient components

Last approach can be motivated by minimizing regret.

- RMSProp
- ADAGRAD
- ADAM
- Batch normalization
- TRish

Outline

Why should we care?

Mathematical optimization is one of the foundations of machine learning.

- Understanding machine learning requires understanding optimization!
- . . . after all, the effectiveness of that model that you trained depends greatly on the optimization algorithm that produced it.

Why is optimization for machine learning difficult?

- We're using randomized algorithms to "solve" an unknown problem
- . . . and somehow it can be argued that's the best thing to do!

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.