Beyond SG: Noise Reduction and Second-Order Methods

https://arxiv.org/abs/1606.04838

Frank E. Curtis, Lehigh University

joint work with

Léon Bottou, Facebook AI Research
Jorge Nocedal, Northwestern University

International Conference on Machine Learning (ICML)
New York, NY, USA

19 June 2016

Outline

What have we learned about SG?

Assumption $\langle L/c \rangle$

The objective function $F : \mathbb{R}^d \to \mathbb{R}$ is

- *c-strongly convex ($\Rightarrow$ unique minimizer) and*
- *L-smooth (i.e., ∇F is Lipschitz continuous with constant L).*

Theorem SG (sublinear convergence)

Under Assumption $\langle L/c \rangle$ and $\mathbb{E}_{\xi_k}[\|g(w_k, \xi_k)\|_2^2] \leq M + \mathcal{O}(\|\nabla F(w_k)\|_2^2)$,

$$w_{k+1} \leftarrow w_k - \alpha_k g(w_k, \xi_k)$$

yields

$$\alpha_k = \frac{1}{L} \qquad \Longrightarrow \mathbb{E}[F(w_k) - F_*] \to \frac{M}{2c};$$

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

(*Let's assume unbiased gradient estimates; see paper for more generality.)

Illustration

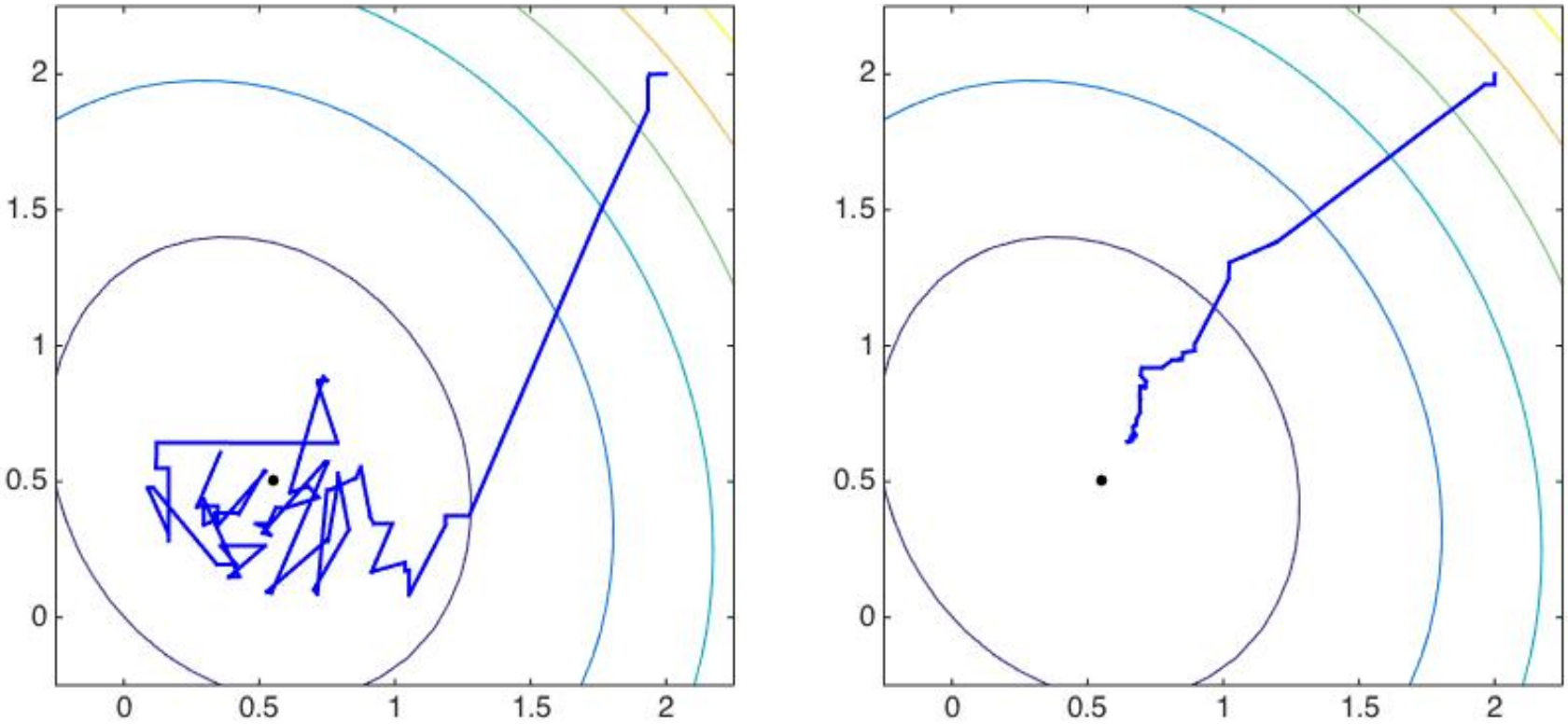
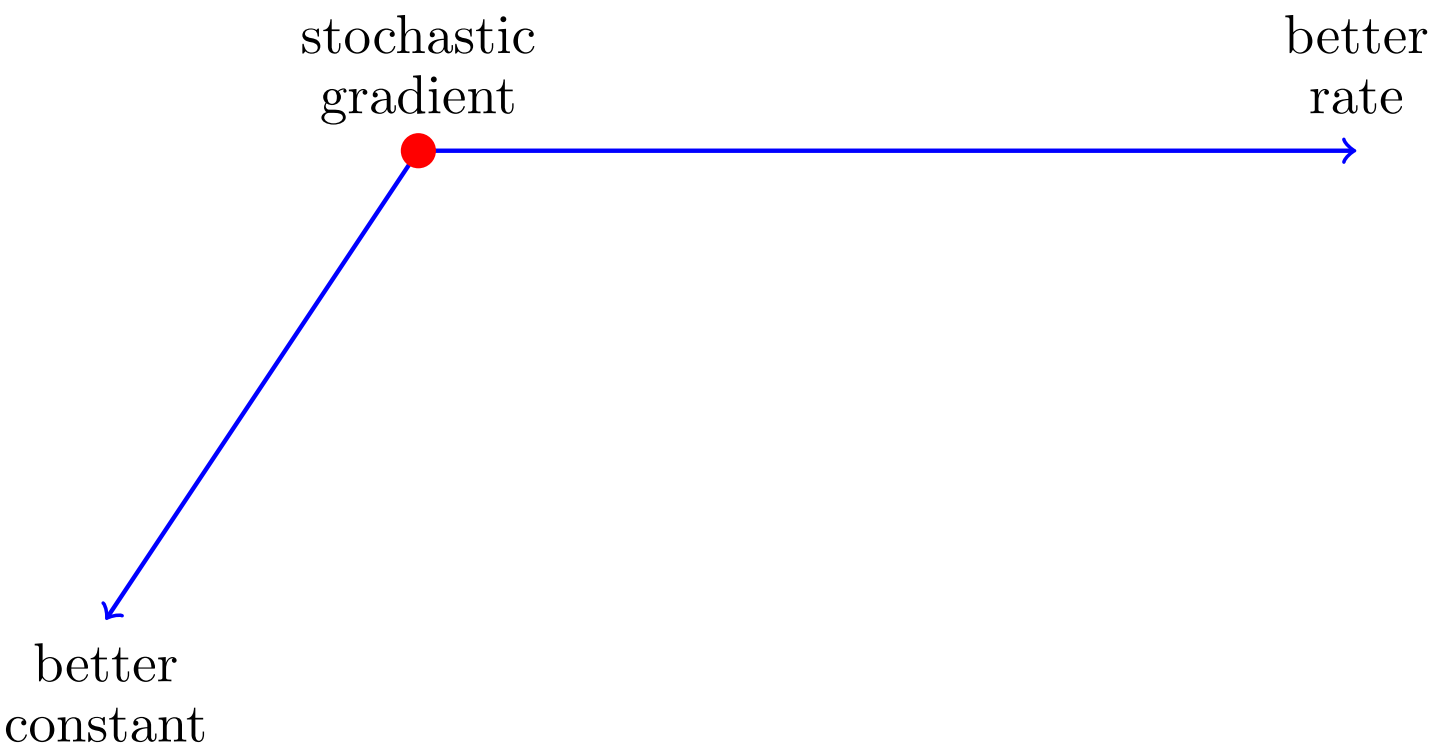
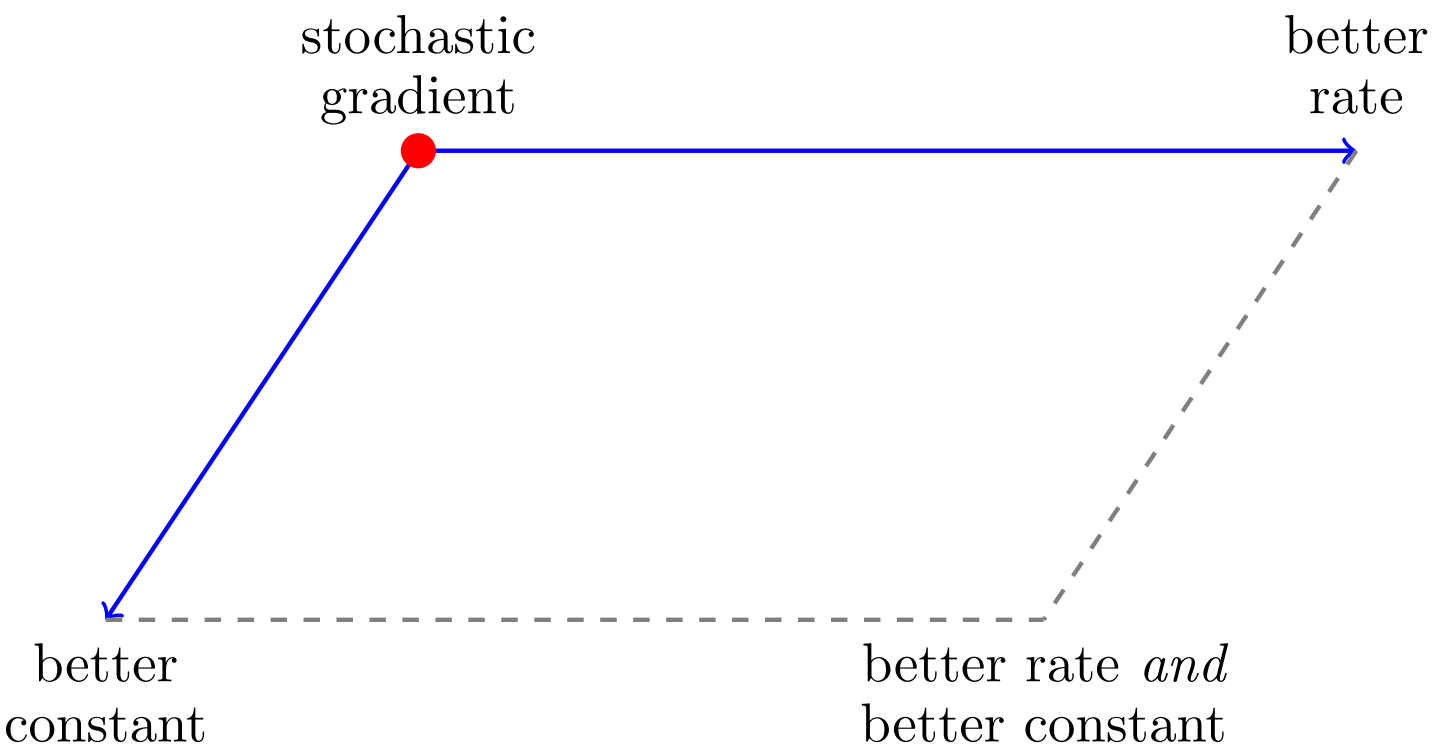


Figure: SG run with a fixed stepsize (left) vs. diminishing stepsizes (right)

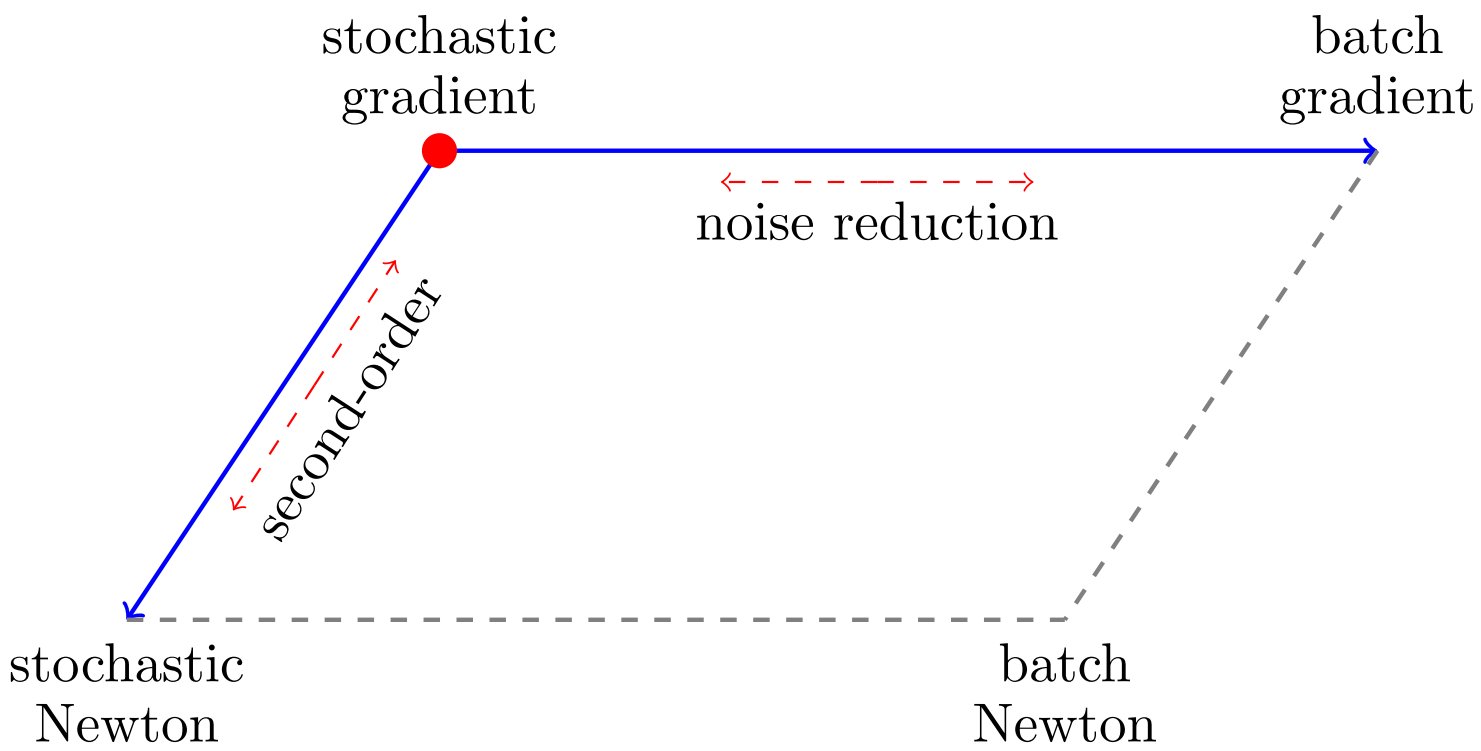
What can be improved?



What can be improved?



Two-dimensional schematic of methods



Nonconvex objectives

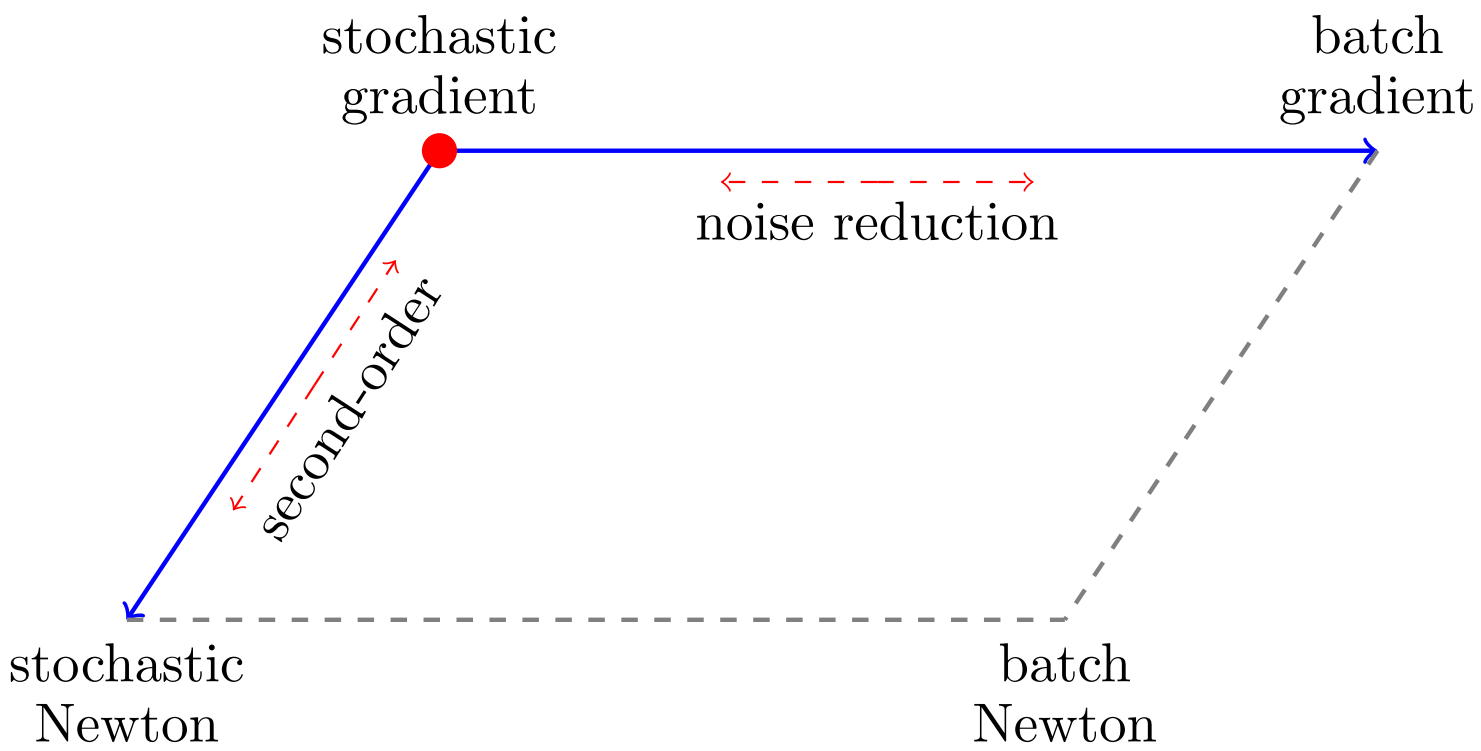
Despite loss of convergence rate, motivation for nonconvex problems as well:

- Convex results describe behavior near strong local minimizer
- Batch gradient methods are unlikely to get trapped near saddle points
- Second-order information can
  - avoid negative effects of nonlinearity and ill-conditioning
  - *require* mini-batching (noise reduction) to be efficient

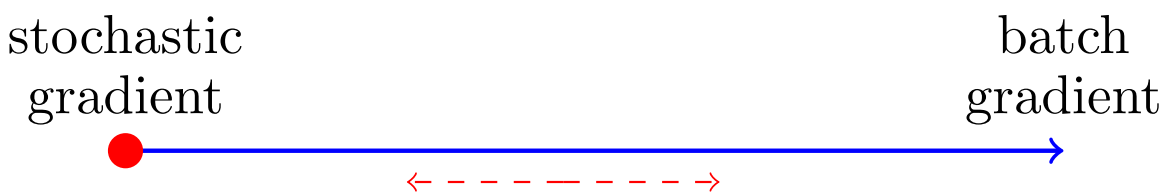
Conclusion: explore entire plane, not just one axis

Outline

Two-dimensional schematic of methods



2D schematic: Noise reduction methods

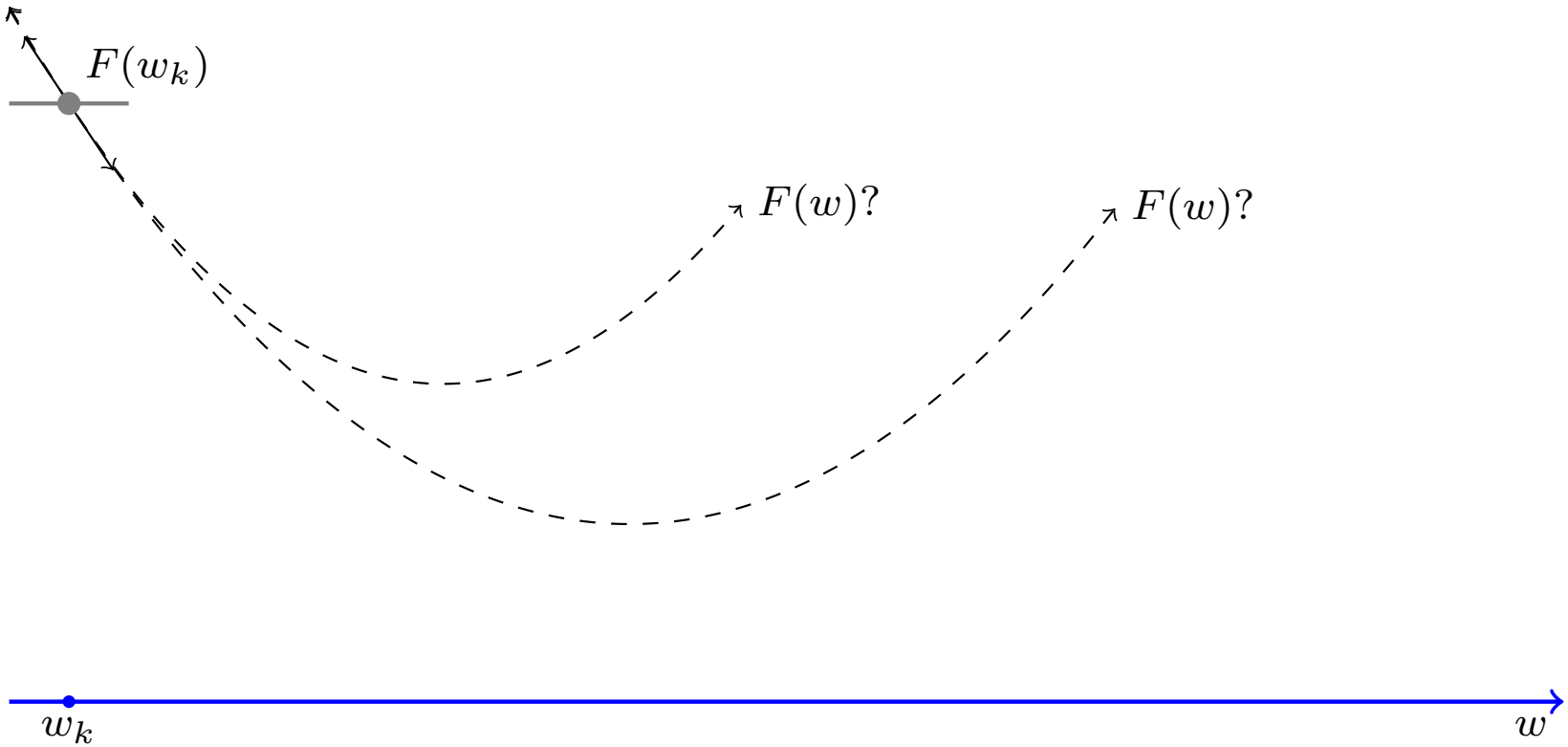


noise reduction

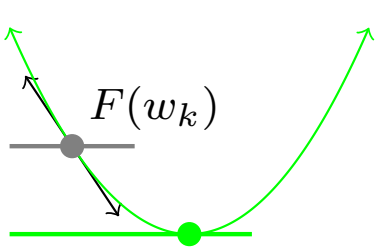
- dynamic sampling
- gradient aggregation
- iterate averaging

Ideal: Linear convergence of a batch gradient method

Ideal: Linear convergence of a batch gradient method

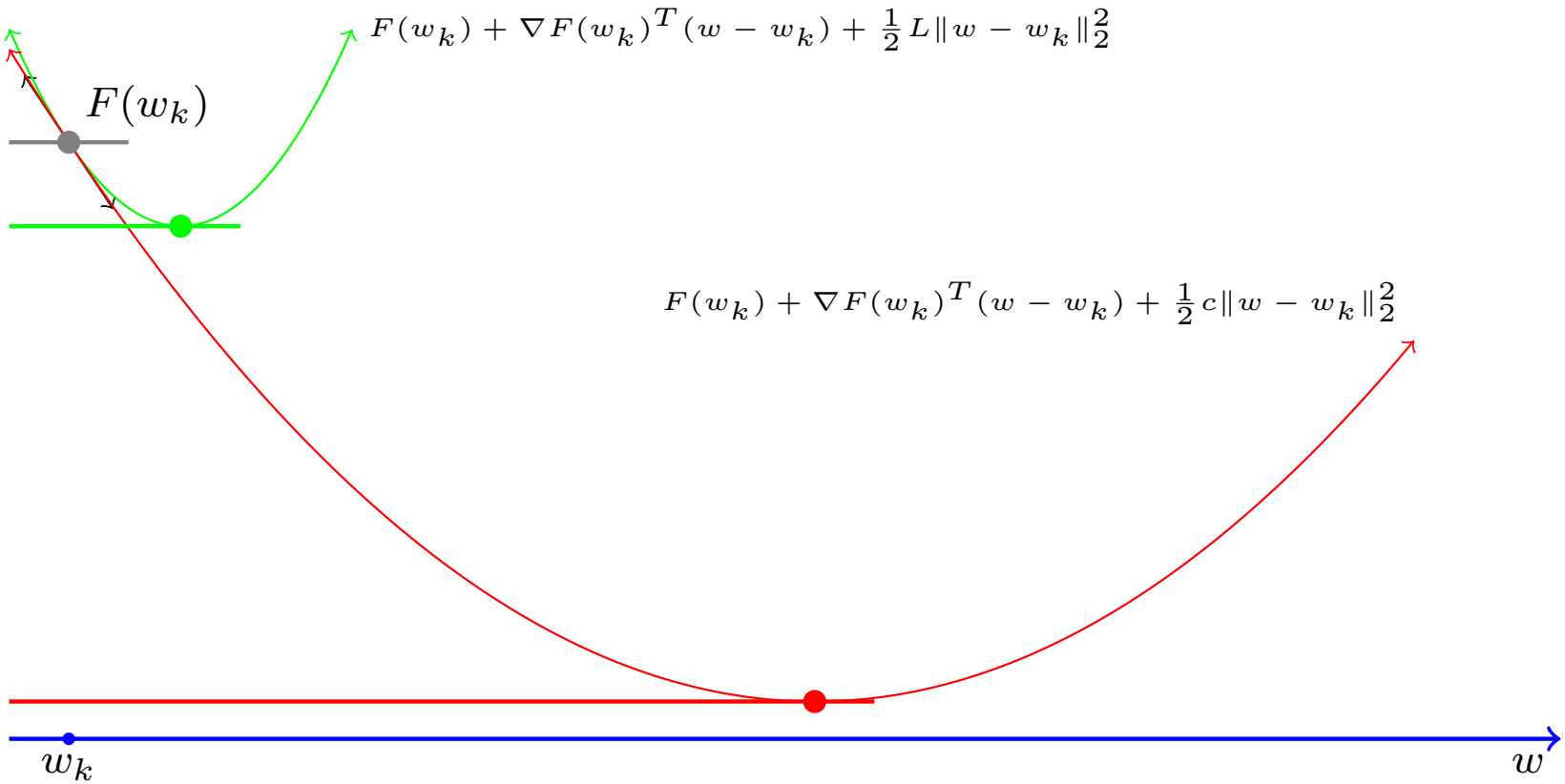


Ideal: Linear convergence of a batch gradient method

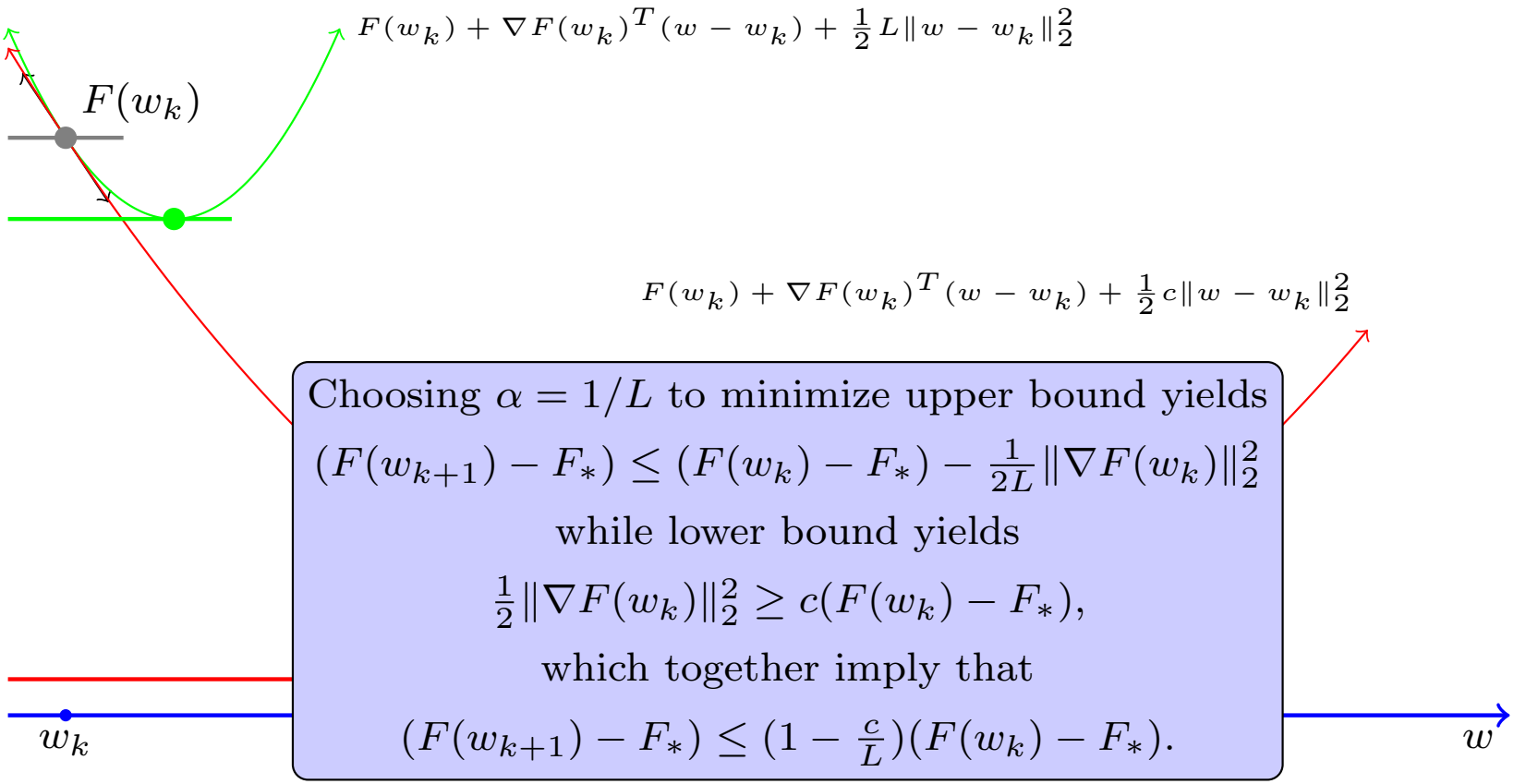


$$F(w_k) + \nabla F(w_k)^T (w - w_k) + \tfrac{1}{2} L \| w - w_k \|_2^2$$

Ideal: Linear convergence of a batch gradient method



Ideal: Linear convergence of a batch gradient method



Choosing $\alpha = 1/L$ to minimize upper bound yields

$$(F(w_{k+1}) - F_*) \leq (F(w_k) - F_*) - \frac{1}{2L}\|\nabla F(w_k)\|_2^2$$

while lower bound yields

$$\frac{1}{2}\|\nabla F(w_k)\|_2^2 \geq c(F(w_k) - F_*),$$

which together imply that

$$(F(w_{k+1}) - F_*) \leq (1 - \tfrac{c}{L})(F(w_k) - F_*).$$

Illustration

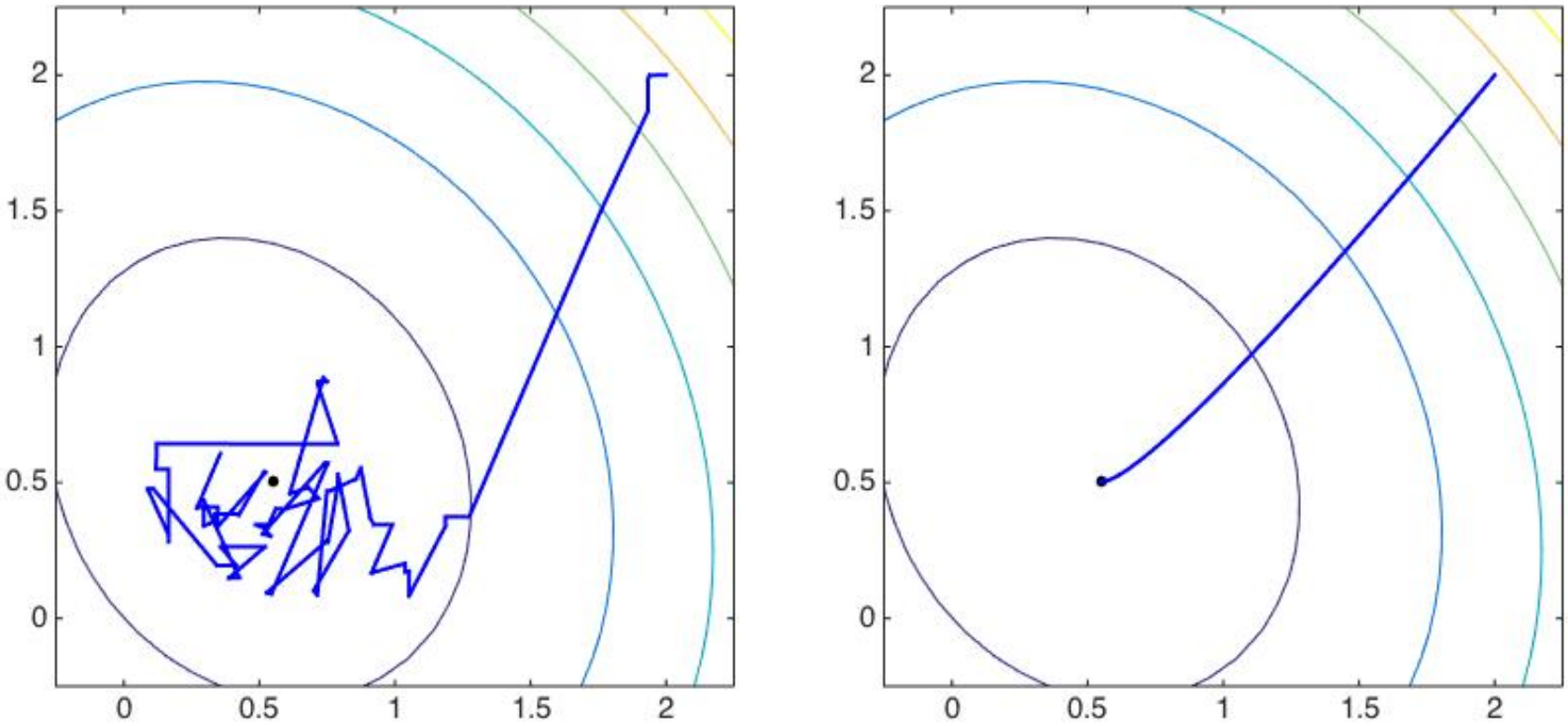


Figure: SG run with a fixed stepsize (left) vs. batch gradient with fixed stepsize (right)

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.

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 Assumption $\langle L/c \rangle$ *holds and that*

$$\mathbb{V}_{\xi_k}[g(w_k, \xi_k)] \leq M\zeta^{k-1} \quad \textit{for some} \quad M \geq 0 \quad \textit{and} \quad \zeta \in (0, 1).$$

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

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

where

$$\omega := \max\left\{\frac{M}{c}, F(w_1) - F_*\right\}$$

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

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

Geometric decrease

Proof.

The now familiar inequality

$$\mathbb{E}_{\xi_k}[F(w_{k+1})] - F(w_k) \leq -\alpha\|\nabla F(w_k)\|_2^2 + \tfrac{1}{2}\alpha^2 L\mathbb{E}_{\xi_k}[\|g(w_k,\xi_k)\|_2^2],$$

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?

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

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

$$\mathbb{V}_{\xi_k}[g_k] \leq \frac{\mathbb{V}_{\xi_k}[\nabla f(w_k; \xi_{k,i})]}{|\mathcal{S}_k|} \leq M(\lceil \tau \rceil)^{k-1}.$$

Practical geometric decrease (unlimited samples)

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

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

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

$$\mathbb{V}_{\xi_k}[g_k] \leq \frac{\mathbb{V}_{\xi_k}[\nabla f(w_k; \xi_{k,i})]}{|\mathcal{S}_k|} \leq M(\lceil \tau \rceil)^{k-1}.$$

But is it too fast? What about work complexity?

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

Illustration

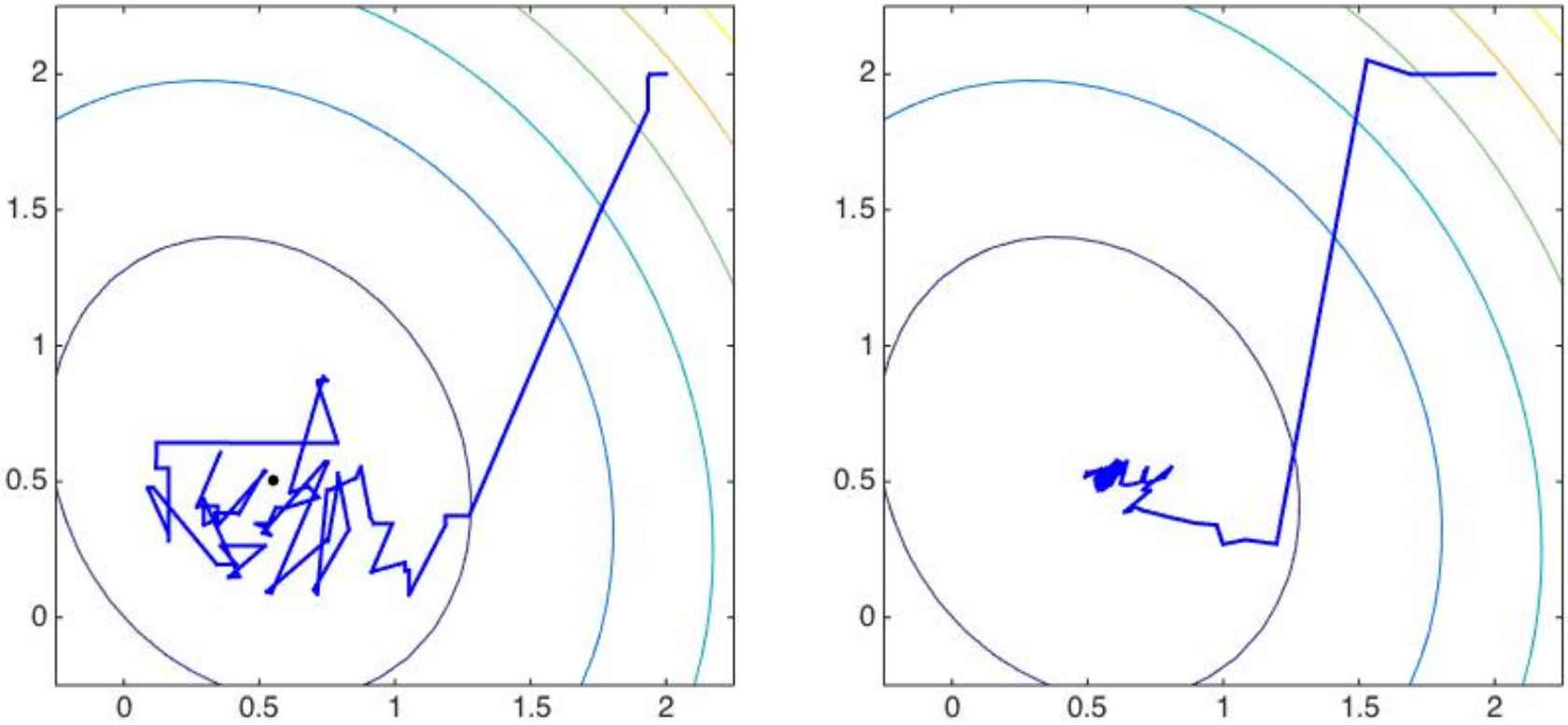


Figure: SG run with a fixed stepsize (left) vs. dynamic SG with fixed stepsize (right)

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)."

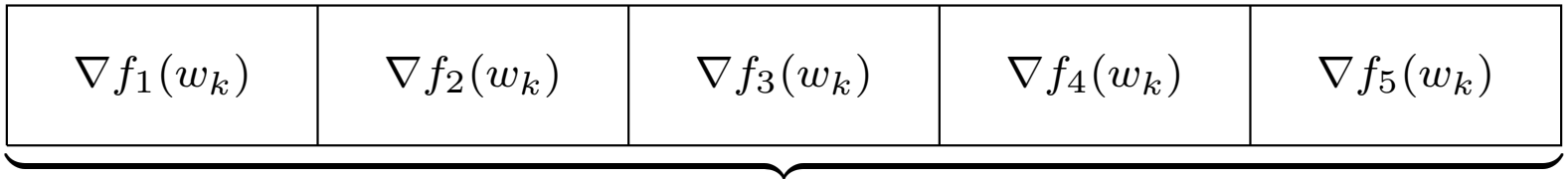
$$F(w) = R_n(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w).$$

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

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:

$\nabla f_1(w_k)$	$\nabla f_2(w_k)$	$\nabla f_3(w_k)$	$\nabla f_4(w_{k,2})$	$\nabla f_5(w_k)$

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

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:

$\nabla f_1(w_k)$	$\nabla f_2(w_{k,3})$	$\nabla f_3(w_k)$	$\nabla f_4(w_k)$	$\nabla f_5(w_k)$

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

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 $\nabla F(w_{k,j})$!

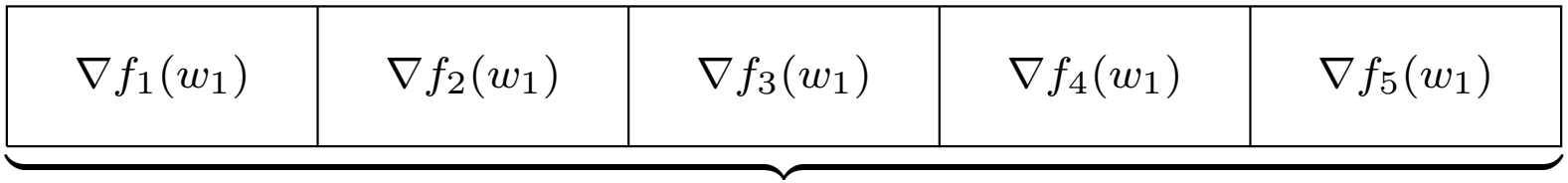
Algorithm SVRG

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

Under Assumption $\langle L/c \rangle$, options (b) and (c) 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:

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

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

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:

$\nabla f_1(w_1)$	$\nabla f_2(w_3)$	$\nabla f_3(w_1)$	$\nabla f_4(w_2)$	$\nabla f_5(w_1)$

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

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

Under Assumption $\langle L/c \rangle$, 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:

$$w_{k+1} \leftarrow w_k - \alpha_k g(w_k, \xi_k)$$

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

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:

$$w_{k+1} \leftarrow w_k - \alpha_k g(w_k, \xi_k)$$

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

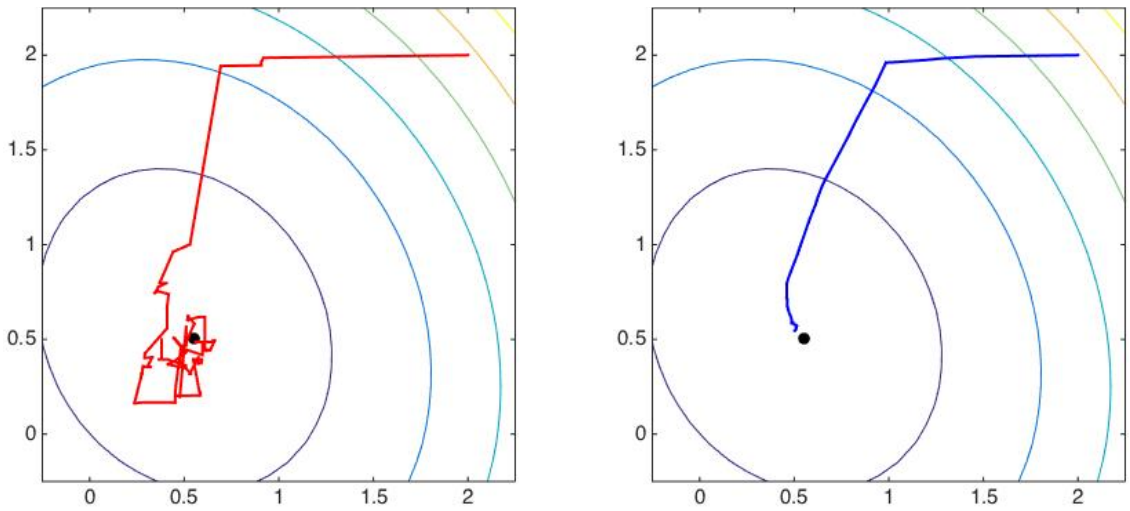
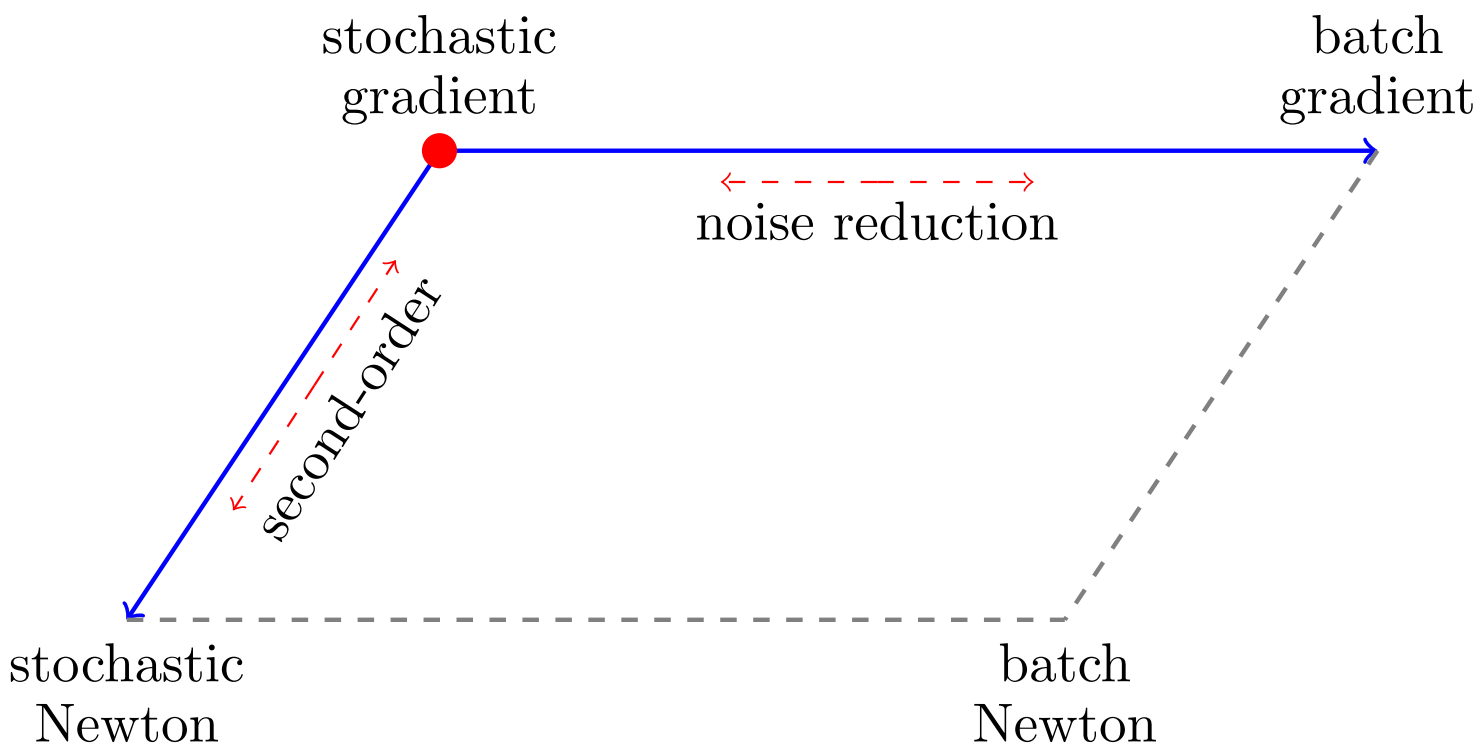


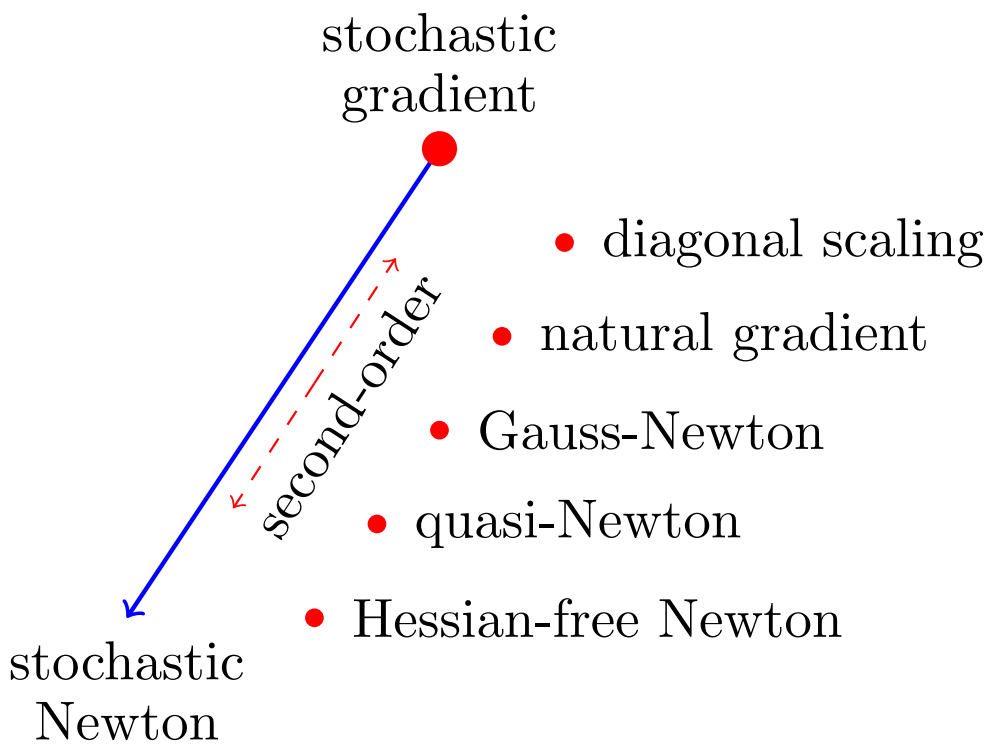
Figure: SG run with $\mathcal{O}(1/\sqrt{k})$ stepsizes (left) vs. sequence of averages (right)

Outline

Two-dimensional schematic of methods



2D schematic: Second-order methods



Ideal: Scale invariance

Neither SG nor batch gradient are invariant to linear transformations!

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

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

Ideal: Scale invariance

Neither SG nor batch gradient are invariant to linear transformations!

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

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

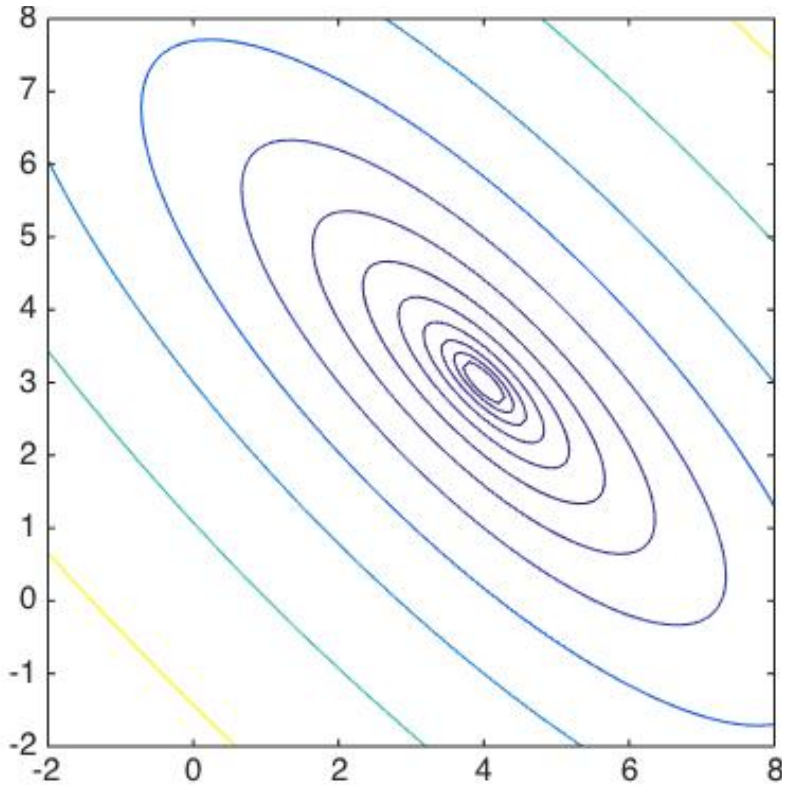
Scaling latter by B and defining $\{w_k\} = \{B\tilde{w}_k\}$ yields

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

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

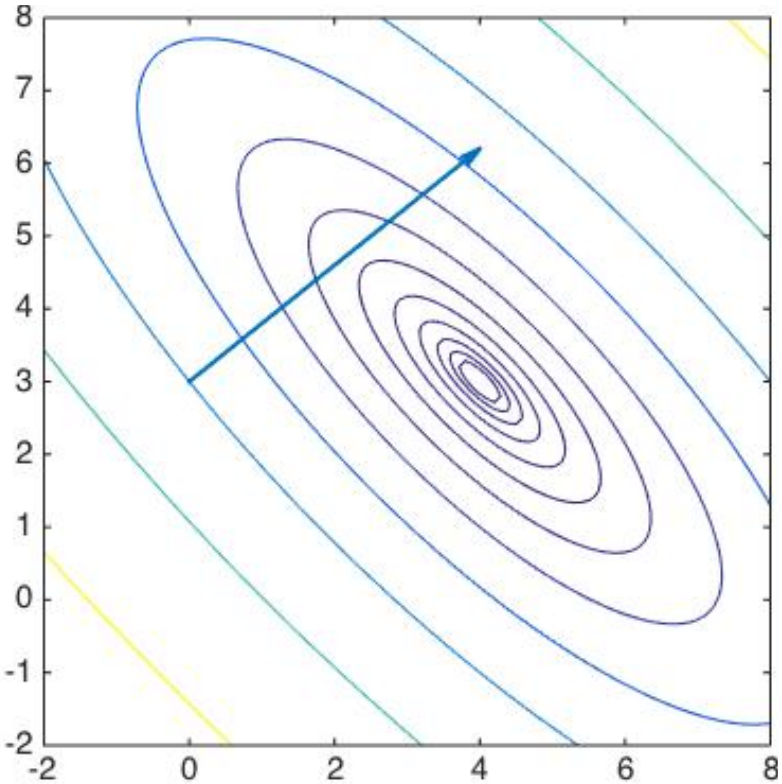
Newton scaling

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



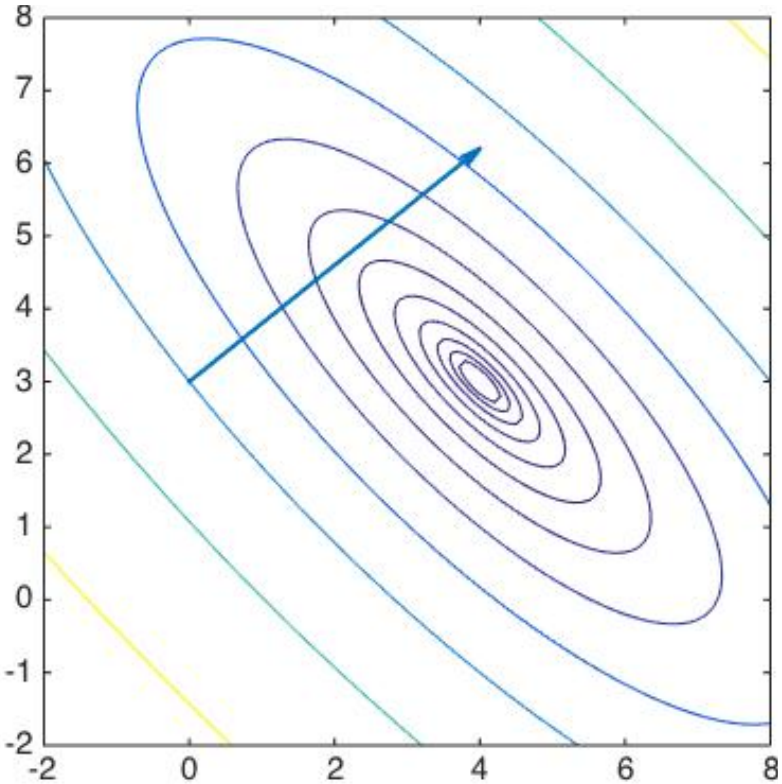
Newton scaling

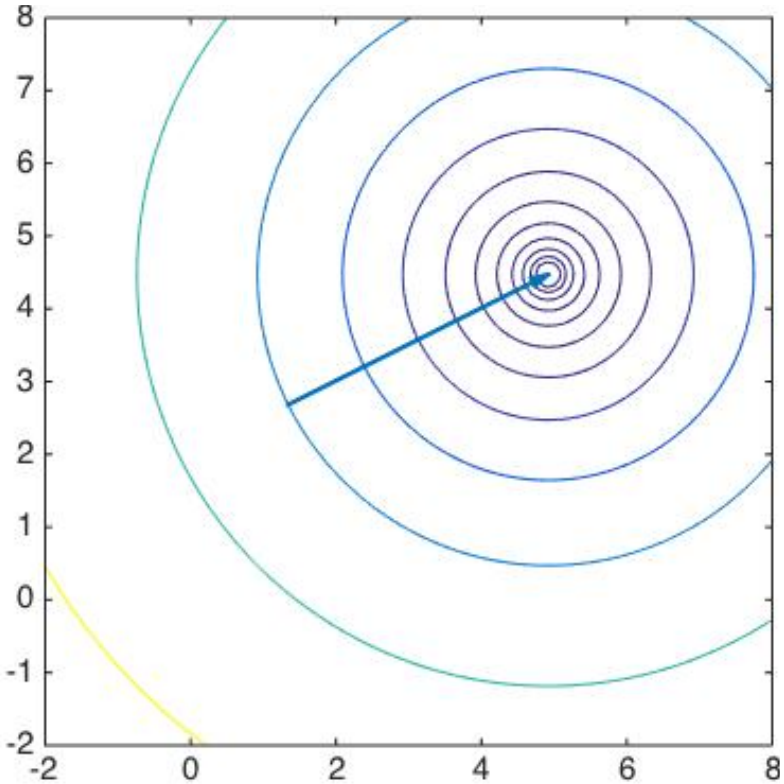
Batch gradient step $-\alpha_k \nabla F(w_k)$ ignores curvature of the function:



Newton scaling

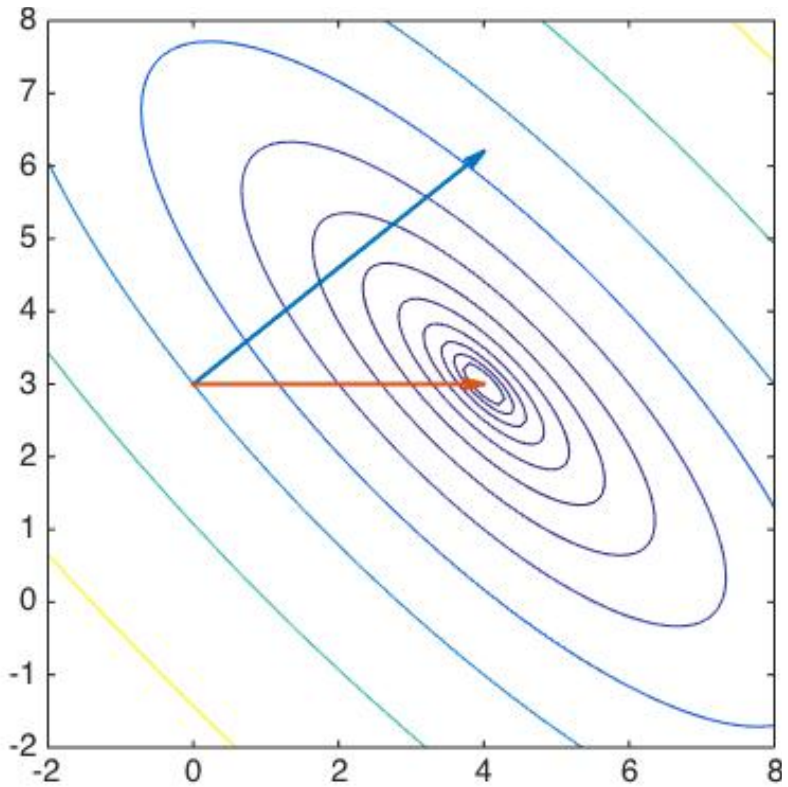
Newton scaling ($B = (\nabla F(w_k))^{-1/2}$): gradient step moves to the minimizer:

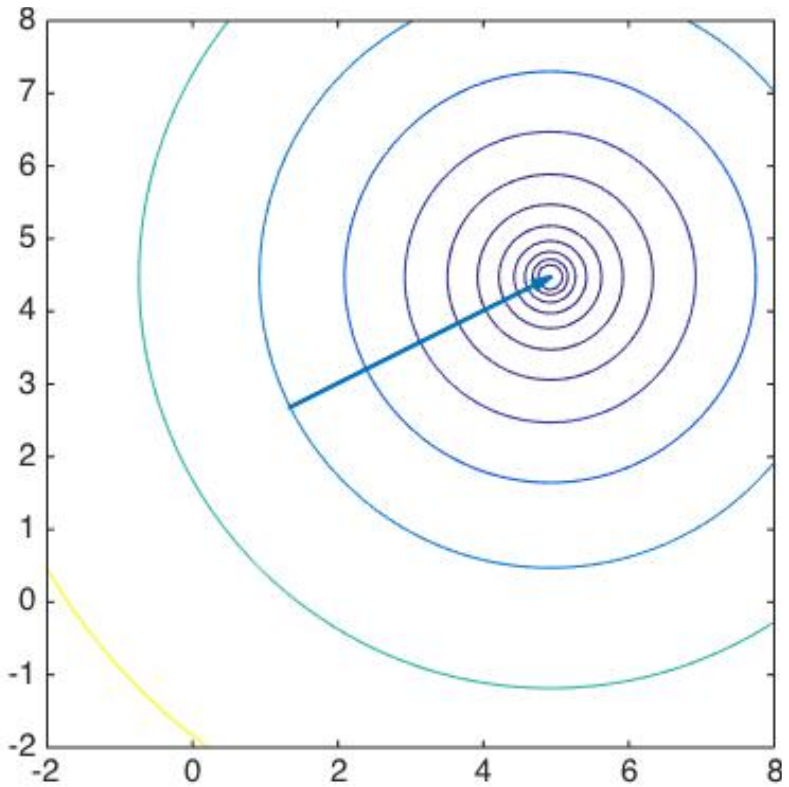




Newton scaling

. . . 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

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*)

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 minimial 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:

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

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:

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

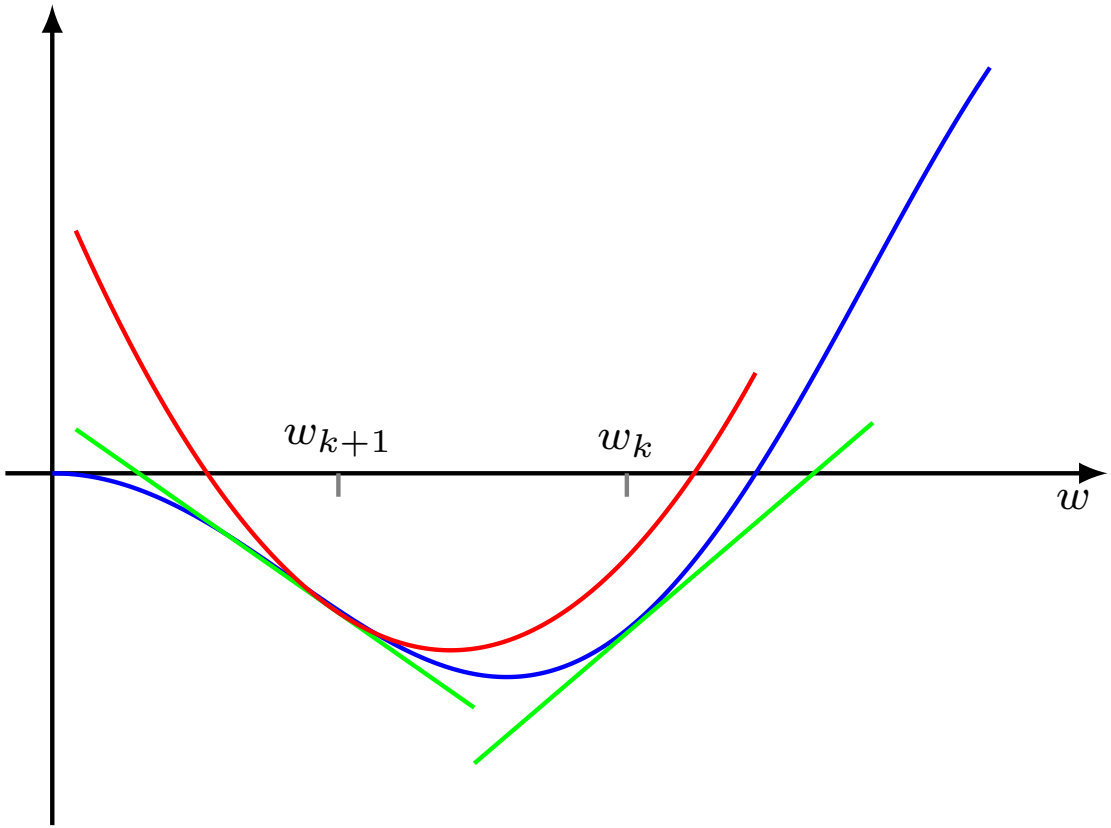
Can be generalized for other (convex) losses:

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

- 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

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

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.

Outline

Plenty of ideas not covered here!

- gradient methods with momentum
- gradient methods with acceleration
- coordinate descent/ascent in the primal/dual
- proximal gradient/Newton for regularized problems
- alternating direction methods
- expectation-maximization
- ...