Second-Order Methods for Stochastic Optimization

Frank E. Curtis, Lehigh University

involving joint work with

Léon Bottou, Facebook AI Research
Jorge Nocedal, Northwestern University
"Optimization Methods for Large-Scale Machine Learning"
http://arxiv.org/abs/1606.04838

Columbia University, Department of IEOR

16 October 2017

Outline

Outline

Perspectives on Nonconvex Optimization

GD, SG, and Beyond

Stochastic Quasi-Newton

Self-Correcting Properties of BFGS

Proposed Algorithm: SC-BFGS

Summary

Problem statement

Consider the problem to find $w \in \mathbb{R}^d$ to minimize f subject to being in $\mathcal{W} \subseteq \mathbb{R}^d$:

$$\min_{w \in \mathcal{W}} f(w). \tag{P}$$

Interested in algorithms for solving (P) when f might not be convex.

Nonconvex optimization is experiencing a heyday!

- nonlinear least squares
- training deep neural networks
- subspace clustering
- ...

History

Nonlinear optimization has had parallel developments

convexity	smoothness
Rockafellar Fenchel Nemirovski Nesterov	Powell Fletcher Goldfarb Nocedal
subgradient inequality	sufficient decrease
convergence, complexity guarantees	convergence, fast local convergence

These worlds are (finally) colliding! Where should emphasis be placed?

First- versus second-order

First-order methods follow a steepest descent methodology:

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

Second-order methods follow Newton's methodology:

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

which one should view as minimizing a quadratic model of f at w_k:

$$f(w_k) + \nabla f(w_k)^T (w - w_k) + \tfrac{1}{2}(w - w_k)^T \nabla^2 f(w_k)(w - w_k)$$

First- versus quasi-second-order

First-order methods follow a steepest descent methodology:

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

Second-order methods follow Newton's methodology:

$$w_{k+1} \leftarrow w_k - \alpha_k M_k \nabla f(w_k),$$

which one should view as minimizing a quadratic model of f at w_k:

$$f(w_k) + \nabla f(w_k)^T (w - w_k) + \frac{1}{2}(w - w_k)^T H_k (w - w_k)$$

Might also replace the Hessian with an approximation H_k with inverse M_k

Why second-order???

Traditional motivation:

- Fast local convergence guarantees

Recent motivation:

- Better complexity properties

Why second-order???

Traditional motivation:

- Fast local convergence guarantees

Recent motivation:

- Better complexity properties

However, I believe these convey the wrong message, especially when problems

- . . . involve stochasticity / randomness
- . . . involve nonsmoothness

I believe there are other more appropriate motivations

Early 2010's

Complexity guarantees for nonconvex optimization algorithms

- Iterations or function/derivative evaluations to achieve

$$\|\nabla f(w_k)\|_2 \leq \epsilon$$

- Steepest descent (first-order): $\mathcal{O}(\epsilon^{-2})$
- Line search (second-order): $\mathcal{O}(\epsilon^{-2})$
- Trust region (second-order): $\mathcal{O}(\epsilon^{-2})$
- Cubic regularization (second-order): $\mathcal{O}(\epsilon^{-3/2})$

Cubic regularization has longer history, but *picks up steam* in early 2010's:

- Griewank (1981)
- Nesterov & Polyak (2006)
- Weiser, Deuflhard, Erdmann (2007)
- Cartis, Gould, Toint (2011), the `ARC` method

Theory vs. practice

Researchers have been gravitating to adopt and build on cubic regularization:

- Agarwal, Allen-Zhu, Bullins, Hazan, Ma (2017)
- Carmon, Duchi (2017)
- Kohler, Lucchi (2017)
- Peng, Roosta-Khorasan, Mahoney (2017)

However, *there remains a large gap between theory and practice*!

Theory vs. practice

Researchers have been gravitating to adopt and build on cubic regularization:

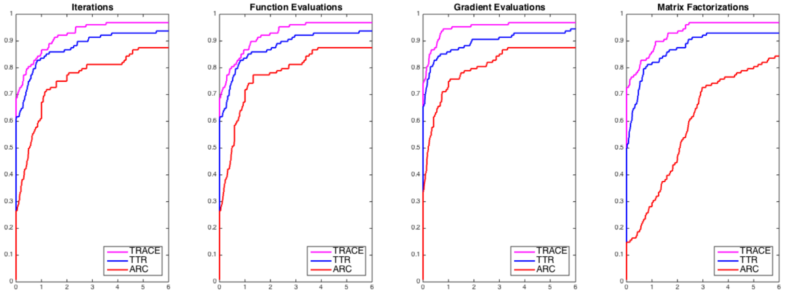
- Agarwal, Allen-Zhu, Bullins, Hazan, Ma (2017)
- Carmon, Duchi (2017)
- Kohler, Lucchi (2017)
- Peng, Roosta-Khorasan, Mahoney (2017)

However, *there remains a large gap between theory and practice*!

Little evidence that cubic regularization methods offer improved performance:

- Trust region (TR) methods remain the state-of-the-art
- TR-like methods can achieve the same complexity guarantees

Trust region methods with optimal complexity



So, why second-order?

For better complexity properties?

- Eh, not really...
- Many are no better than first-order methods in terms of complexity
- ...and ones with better complexity aren't necessarily best in practice (yet)

So, why second-order?

For better complexity properties?

- Eh, not really...
- Many are no better than first-order methods in terms of complexity
- ...and ones with better complexity aren't necessarily best in practice (yet)

For fast local convergence guarantees?

- Eh, probably not...
- Hard to achieve, especially in large-scale, nonsmooth, or stochastic settings

So, why second-order?

For better complexity properties?

- Eh, not really. . .
- Many are no better than first-order methods in terms of complexity
- . . . and ones with better complexity aren't necessarily best in practice (yet)

For fast local convergence guarantees?

- Eh, probably not. . .
- Hard to achieve, especially in large-scale, nonsmooth, or stochastic settings

Then why?

- Adaptive, natural scaling (gradient descent $\approx 1/L$ while Newton ≈ 1)
- Mitigate effects of ill-conditioning
- Easier to tune parameters(?)
- Better at avoiding saddle points(?)
- Better trade-off in parallel and distributed computing settings

(Also, opportunities for NEW algorithms! Not analyzing the same old. . .)

Message of this talk

People want to solve more complicated, nonconvex problems

- . . . involving stochasticity / randomness
- . . . involving nonsmoothness

We might waste this spotlight on nonconvex optimization if we do not. . .

- Make clear the gap between theory and practice (and close it!)
- Learn from advances that have already been made
- . . . and adapt them *appropriately* for modern problems

Outline

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

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

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

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

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

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

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

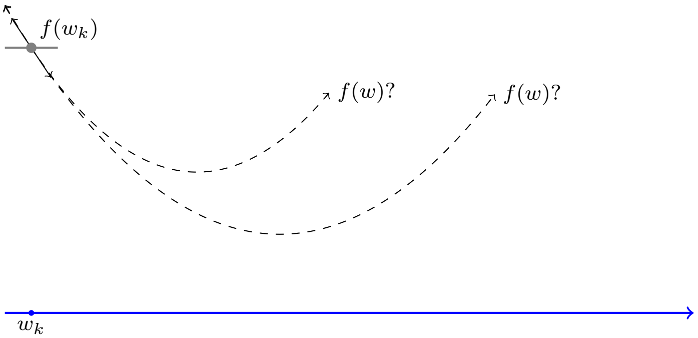
Algorithm GD : Gradient Descent

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

Gradient descent

Algorithm GD : Gradient Descent

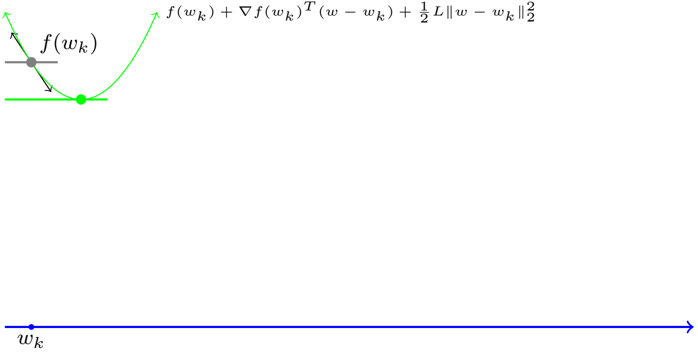
1: choose an initial point $w_0 \in \mathbb{R}^n$ and stepsize $\alpha > 0$
2: **for** $k \in \{0, 1, 2, \ldots\}$ **do**
3: set $w_{k+1} \leftarrow w_k - \alpha \nabla f(w_k)$
4: **end for**



Gradient descent

Algorithm GD : Gradient Descent

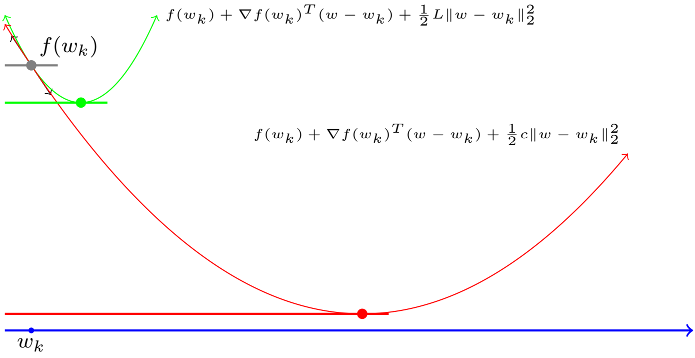
choose an initial point $w_0 \in \mathbb{R}^n$ and stepsize $\alpha > 0$
for $k \in \{0, 1, 2, \dots\}$ **do**
    set $w_{k+1} \leftarrow w_k - \alpha \nabla f(w_k)$
end for



Gradient descent

Algorithm GD : Gradient Descent

1: choose an initial point $w_0 \in \mathbb{R}^n$ and stepsize $\alpha > 0$
2: **for** $k \in \{0, 1, 2, \ldots\}$ **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* $\{\nabla f(w_k)\} \to 0$.

Proof.

$$\begin{aligned} f(w_{k+1}) &\leq f(w_k) + \nabla f(w_k)^T (w_{k+1} - w_k) + \tfrac{1}{2} L \|w_{k+1} - w_k\|_2^2 \\ &\leq f(w_k) - \tfrac{1}{2}\alpha \|\nabla f(w_k)\|_2^2 \end{aligned}$$

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.

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

$$\implies f(w_{k+1}) - f_* \leq (1 - \alpha c)(f(w_k) - f_*).$$

GD illustration

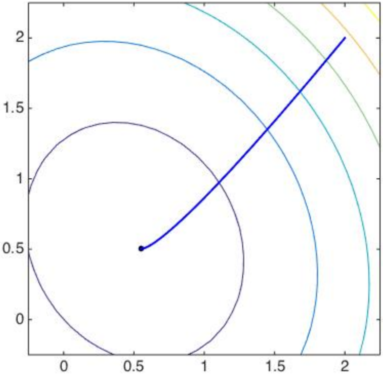


Figure: GD with fixed stepsize

Stochastic gradient ~~descent~~

Approximate gradient only; e.g., random i_k and $\nabla_w \ell(h(w; x_{i_k}), y_{i_k}) \approx \nabla f(w)$.

Algorithm SG : Stochastic Gradient

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

Stochastic gradient ~~descent~~

Approximate gradient only; e.g., random i_k and $\nabla_w \ell(h(w; x_{i_k}), y_{i_k}) \approx \nabla f(w)$.

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 $\mathbb{E}_k[g_k] = \nabla f(w_k)$):

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

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:*

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

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

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

SG theory

Theorem SG

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

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

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

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

$$\alpha_k = \frac{1}{L} \implies \mathbb{E}[f(w_k) - f_*] \overset{k\to\infty}{\leq} \frac{(M/c)}{2}$$

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

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

SG illustration

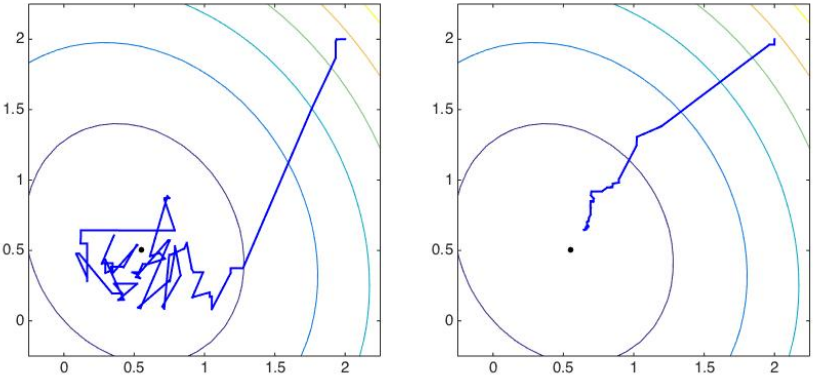


Figure: SG with fixed stepsize (left) vs. diminishing stepsizes (right)

Why SG over GD for large-scale machine learning?

We have seen:

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?

Why SG over GD for large-scale machine learning?

We have seen:

$$\begin{array}{lll} \text{GD:} & \mathbb{E}[f_n(w_k) - f_{n,*}] = \mathcal{O}(\rho^k) & \text{linear convergence} \\ \text{SG:} & \mathbb{E}[f_n(w_k) - f_{n,*}] = \mathcal{O}(1/k) & \text{sublinear convergence} \end{array}$$

So why SG?

Motivation	Explanation
Intuitive	data "redundancy"
Empirical	SG works well in practice vs. GD
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		Cost per iteration		Cost 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:	$\mathbb{E}[f_n(w_k) - f_{n,*}] = \mathcal{O}(1/k)$	$+$	$\mathcal{O}(1)$	$\Longrightarrow$	$1/\epsilon$

Work complexity

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

	Convergence rate		Cost per iteration		Cost 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

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

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

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

End of the story?

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

- Stability of SG; Hardt, Recht, Singer (2015)
- SG avoids steep minima; Keskar, Mudigere, Nocedal, Smelyanskiy (2016)
- . . . (many more)

End of the story?

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

- Stability of SG; Hardt, Recht, Singer (2015)
- SG avoids steep minima; Keskar, Mudigere, Nocedal, Smelyanskiy (2016)
- ... (many more)

No, we should want more...

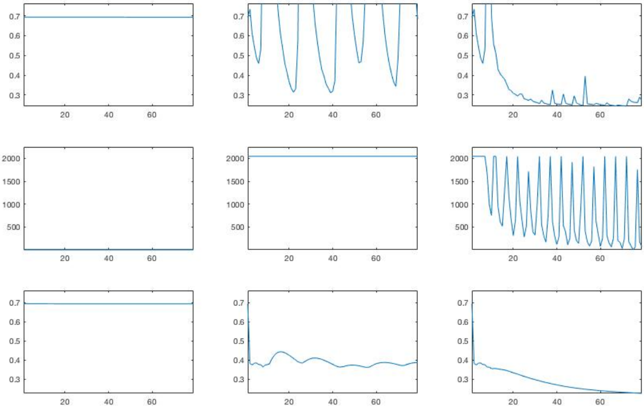
- SG requires a lot of tuning
- Sublinear convergence is not satisfactory
- ... "linearly" convergent method eventually wins
- ... with higher budget, faster computation, parallel?, distributed?

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

Stochastic Optimization: No Parameter Tuning

Limited memory stochastic gradient method (extends Barzilai-Borwein):

$$x_{k+1} \leftarrow x_k - \alpha_k g_k \quad \text{where} \quad \alpha_k > 0 \text{ chosen adaptively}$$

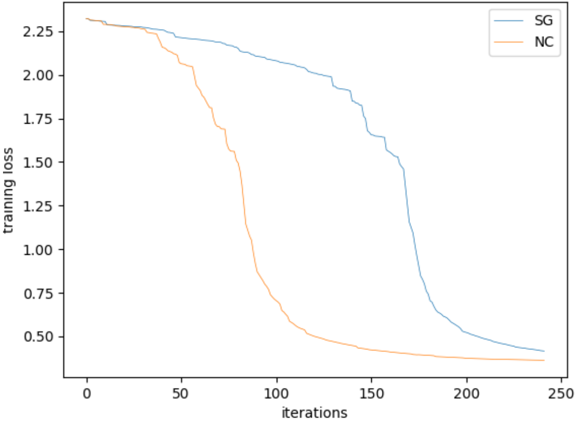


Minimizing logistic loss for binary classification with RCV1 dataset

Stochastic Optimization: Avoiding Saddle Points / Stagnation

Training a convolutional neural network for classifying digits in `mnist`:

Stochastic-gradient-type method versus one that follows negative curvature:

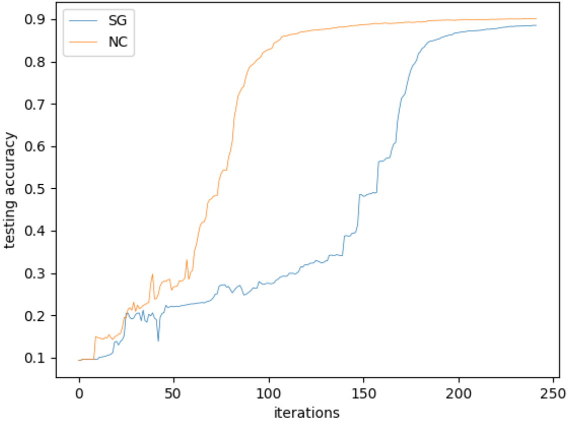


Overcomes slow initial progress by SG-type method. . .

Stochastic Optimization: Avoiding Saddle Points / Stagnation

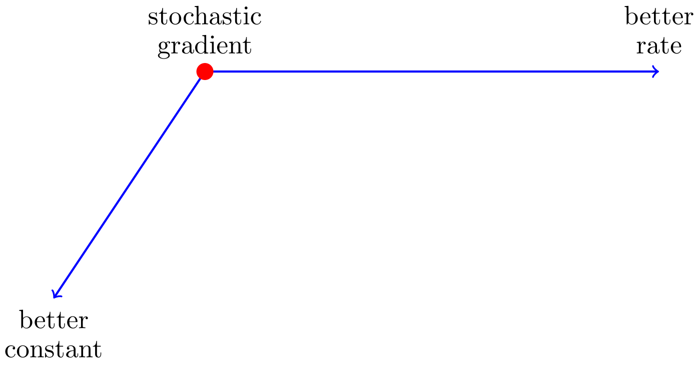
Training a convolutional neural network for classifying digits in `mnist`:

Stochastic-gradient-type method versus one that follows negative curvature:

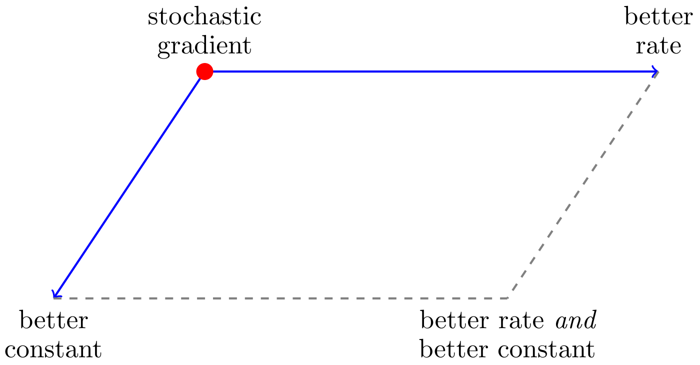


. . . while still yielding good behavior in terms of testing accuracy

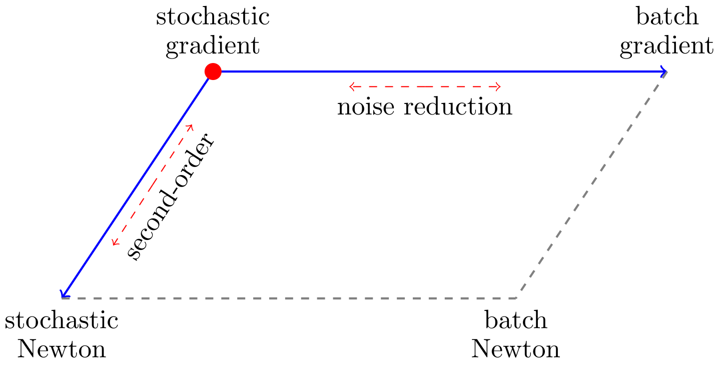
What can be improved?



What can be improved?



Two-dimensional schematic of methods

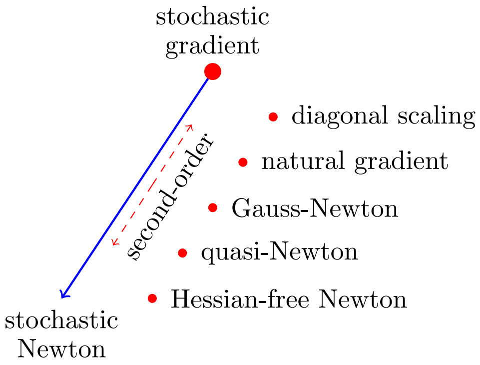


2D schematic: Noise reduction methods

noise reduction

- dynamic sampling
- gradient aggregation
- iterate averaging

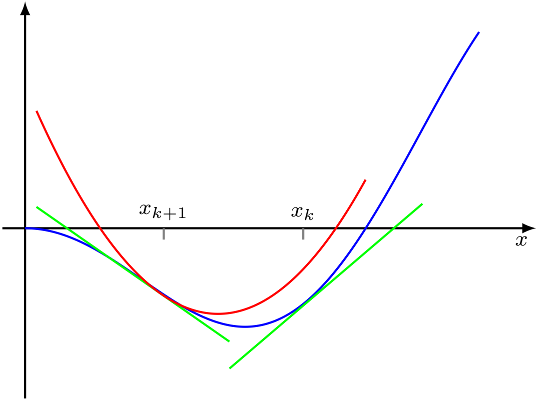
2D schematic: Second-order methods



Outline

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

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

Previous work: BFGS-type methods

Much focus on the secant equation ($H_{k+1} \sim$ Hessian approximation)

$$H_{k+1}s_k = y_k \quad \text{where} \quad \begin{cases} s_k := w_{k+1} - w_k \\ y_k := \nabla f(w_{k+1}) - \nabla f(w_k) \end{cases}$$

and an appropriate replacement for the gradient displacement:

$$y_k \leftarrow \underbrace{\nabla f(w_{k+1}, \xi_k) - \nabla f(w_k, \xi_k)}_{\text{use same seed}}$$

oLBFGS, Schraudolph et al. (2007)
SGD-QN, Bordes et al. (2009)
RES, Mokhtari & Ribeiro (2014)

$$\text{or} \quad y_k \leftarrow \underbrace{\left(\sum_{i \in \mathcal{S}_k^H} \nabla^2 f(w_{k+1}, \xi_{k+1,i}) \right) s_k}_{\text{use action of step on subsampled Hessian}}$$

SQN, Byrd et al. (2015)
Goldfarb et al. (2016)

Is this the right focus? Is there a better way (especially for nonconvex f)?

Proposal

Propose a quasi-Newton method for stochastic (nonconvex) optimization

- exploit self-correcting properties of BFGS-type updates
  - Powell (1976)
  - Ritter (1979, 1981)
  - Werner (1978)
  - Byrd, Nocedal (1989)

Outline

BFGS-type updates

Inverse Hessian and Hessian approximation updating formulas ($s_k^T v_k > 0$):

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

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

- These satisfy secant-type equations

$$M_{k+1} v_k = s_k \;\text{ and }\; H_{k+1} s_k = v_k,$$

but these are not critical for this talk.

Geometric properties of Hessian update: Burke, Lewis, Overton (2007)

Consider the matrices (which only depend on s_k and H_k, not g_k!)

$$P_k := \frac{s_k s_k^T H_k}{s_k^T H_k s_k} \text{ and } Q_k := I - P_k.$$

Both H_k-orthogonal projection matrices (i.e., idempotent and H_k-self-adjoint).

- P_k yields H_k-orthogonal projection onto span(s_k).
- Q_k yields H_k-orthogonal projection onto span$(s_k)^{\perp_{H_k}}$.

Geometric properties of Hessian update: Burke, Lewis, Overton (2007)

Consider the matrices (which only depend on s_k and H_k, not g_k!)

$$P_k := \frac{s_k s_k^T H_k}{s_k^T H_k s_k} \quad \text{and} \quad Q_k := I - P_k.$$

Both H_k-orthogonal projection matrices (i.e., idempotent and H_k-self-adjoint).

- P_k yields H_k-orthogonal projection onto $\text{span}(s_k)$.
- Q_k yields H_k-orthogonal projection onto $\text{span}(s_k)^{\perp_{H_k}}$.

Returning to the Hessian update:

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

- Curvature projected out along $\text{span}(s_k)$
- Curvature corrected by $\frac{v_k v_k^T}{s_k^T v_k} = \left(\frac{v_k v_k^T}{\|v_k\|_2^2}\right)\left(\frac{\|v_k\|_2^2}{v_k^T M_{k+1} v_k}\right)$ (inverse Rayleigh).

Self-correcting properties of Hessian update

Since curvature is constantly projected out, what happens after many updates?

Self-correcting properties of Hessian update

Since curvature is constantly projected out, what happens after many updates?

Theorem 3 (Byrd, Nocedal (1989))

Suppose that, for all k, there exists $\{\eta, \theta\} \subset \mathbb{R}_{++}$ such that

$$\eta \leq \frac{s_k^T v_k}{\|s_k\|_2^2} \quad \text{and} \quad \frac{\|v_k\|_2^2}{s_k^T v_k} \leq \theta. \tag{$\star$}$$

Then, for any $p \in (0,1)$, there exist constants $\{\iota, \kappa, \lambda\} \subset \mathbb{R}_{++}$ such that, for any $K \geq 2$, the following relations hold for at least $\lceil pK \rceil$ values of $k \in \{1, \ldots, K\}$:

$$\iota \leq \frac{s_k^T H_k s_k}{\|s_k\|_2 \|H_k s_k\|_2} \quad \text{and} \quad \kappa \leq \frac{\|H_k s_k\|_2}{\|s_k\|_2} \leq \lambda.$$

Proof technique.

Building on work of Powell (1976), involves bounding growth of

$$\gamma(H_k) = \mathrm{tr}(H_k) - \ln(\det(H_k)).$$

Self-correcting properties of inverse Hessian update

Rather than focus on superlinear convergence results, we care about the following.

Corollary 4

Suppose the conditions of Theorem 3 hold. Then, for any $p \in (0,1)$, there exist constants $\{\mu, \nu\} \subset \mathbb{R}_{++}$ such that, for any $K \geq 2$, the following relations hold for at least $\lceil pK \rceil$ values of $k \in \{1, \dots, K\}$:

$$\mu \|\bar{g}_k\|_2^2 \leq \bar{g}_k^T M_k \bar{g}_k \quad \textit{and} \quad \|M_k \bar{g}_k\|_2^2 \leq \nu \|\bar{g}_k\|_2^2$$

Here $\bar{g}_k$ is the vector such that the iterate displacement is

$$w_{k+1} - w_k = s_k = -M_k \bar{g}_k$$

Proof sketch.

Follows simply after algebraic manipulations from the result of Theorem 3, using the facts that $s_k = -M_k \bar{g}_k$ and $M_k = H_k^{-1}$ for all k.

Summary

Our main idea is to use a carefully selected type of damping:

- Choosing $v_k \leftarrow y_k := g_{k+1} - g_k$ yields standard BFGS, but we consider

$$v_k \leftarrow \beta_k H s_k + (1-\beta_k)\tilde{y}_k \ \text{ for some } \ \beta_k \in [0,1] \ \text{ and } \ \tilde{y}_k \in \mathbb{R}^n.$$

This scheme preserves the self-correcting properties of BFGS.

Outline

Algorithm SC : Self-Correcting BFGS Algorithm

```
Choose $w_1 \in \mathbb{R}^d$.
Set $g_1 \approx \nabla f(w_1)$.
Choose a symmetric positive definite $M_1 \in \mathbb{R}^{d \times d}$.
Choose a positive scalar sequence $\{\alpha_k\}$.
for $k = 1, 2, \ldots$ do
    Set $s_k \leftarrow -\alpha_k M_k g_k$.
    Set $w_{k+1} \leftarrow w_k + s_k$.
    Set $g_{k+1} \approx \nabla f(w_{k+1})$.
    Set $y_k \leftarrow g_{k+1} - g_k$.
    Set $\beta_k \leftarrow \min\{\beta \in [0,1] : v(\beta) := \beta s_k + (1-\beta)\alpha_k y_k$ satisfies $(\star)\}$.
    Set $v_k \leftarrow v(\beta_k)$.
    Set
        $$M_{k+1} \leftarrow \left(I - \frac{v_k s_k^T}{s_k^T v_k}\right)^T M_k \left(I - \frac{v_k s_k^T}{s_k^T v_k}\right) + \frac{s_k s_k^T}{s_k^T v_k}.$$
end for
```

Global convergence theorem

Theorem (Bottou, Curtis, Nocedal (2016))

Suppose that, for all k, there exists a scalar constant $\rho > 0$ such that

$$-\nabla f(w_k)^T \mathbb{E}_{\xi_k}[M_k g_k] \leq -\rho \|\nabla f(w_k)\|_2^2,$$

and there exist scalars $\sigma > 0$ and $\tau > 0$ such that

$$\mathbb{E}_{\xi_k}[\|M_k g_k\|_2^2] \leq \sigma + \tau \|\nabla f(w_k)\|_2^2.$$

Then, $\{\mathbb{E}[f(w_k)]\}$ converges to a finite limit and

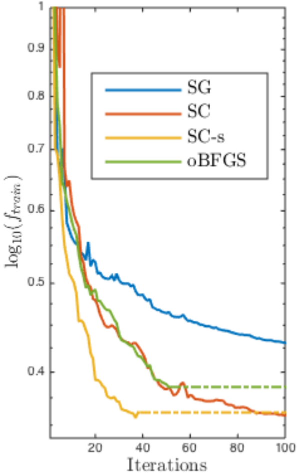
$$\liminf_{k\to\infty} \mathbb{E}[\nabla f(w_k)] = 0.$$

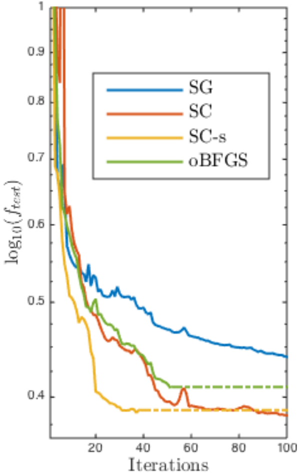
Proof technique.

Follows from the critical inequality

$$\mathbb{E}_{\xi_k}[f(w_{k+1})] - f(w_k) \leq -\alpha_k \nabla f(w_k)^T \mathbb{E}_{\xi_k}[M_k g_k] + \alpha_k^2 L \mathbb{E}_{\xi_k}[\|M_k g_k\|_2^2].$$

Numerical Experiments: a1a

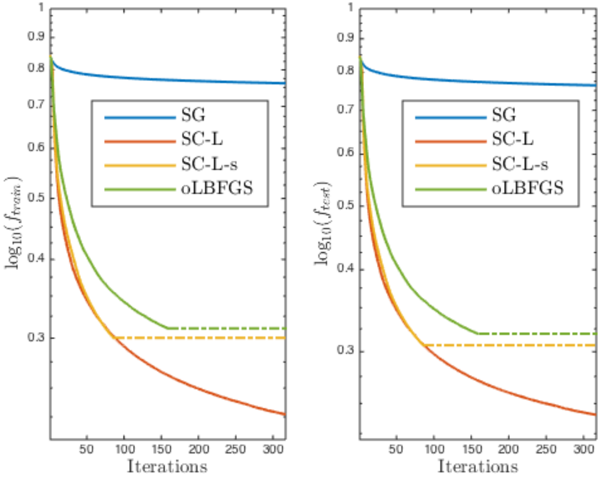




logistic regression, data `a1a`, diminishing stepsizes

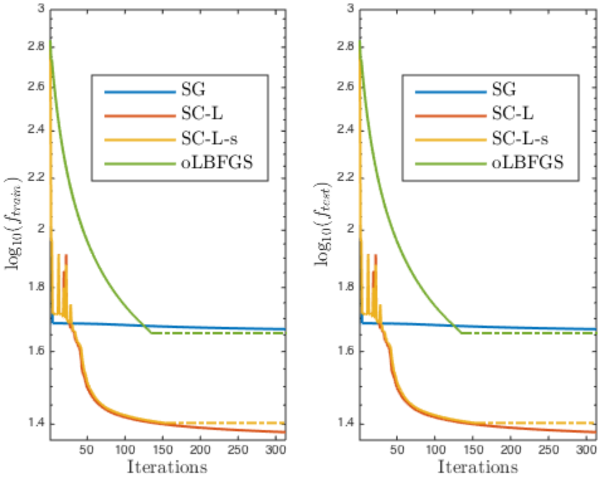
Numerical Experiments: rcv1

SC-L and SC-L-s: limited memory variants of SC and SC-s, respectively:



logistic regression, data `rcv1`, diminishing stepsizes

Numerical Experiments: mnist



deep neural network, data `mnist`, diminishing stepsizes

Outline

Why second-order?

For better complexity properties?

- Eh, not really...
- Many are no better than first-order methods in terms of complexity
- ...and ones with better complexity aren't necessarily best in practice (yet)

For fast local convergence guarantees?

- Eh, probably not...
- Hard to achieve, especially in large-scale, nonsmooth, or stochastic settings

Then why?

- Adaptive, natural scaling (gradient descent $\approx 1/L$ while Newton ≈ 1)
- Mitigate effects of ill-conditioning
- Easier to tune parameters(?)
- Better at avoiding saddle points(?)
- Better trade-off in parallel and distributed computing settings

(Also, opportunities for NEW algorithms! Not analyzing the same old...)

Message of this talk

People want to solve more complicated, nonconvex problems

- ...involving stochasticity / randomness
- ...involving nonsmoothness

We might waste this spotlight on nonconvex optimization if we do not...

- Make clear the gap between theory and practice (and close it!)
- Learn from advances that have already been made
- ...and adapt them *appropriately* for modern problems

References

For references, please see

- http://coral.ise.lehigh.edu/frankecurtis/publications

Please also visit the OptML @ Lehigh website!

- http://optml.lehigh.edu