Second-Order Methods for Stochastic and Nonsmooth Optimization

Frank E. Curtis, Lehigh University

USC Department of ISE

10 October 2017

Outline

Outline

Problem statement

Consider the problem to find $x \in \mathbb{R}^n$ to minimize f subject to being in $\mathcal{X} \subseteq \mathbb{R}^n$:

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{s.t.} \quad x \in \mathcal{X}. \tag{P}$$

Interested in algorithms for solving (P) when f and/or $\mathcal{X}$ might not be convex.

Nonconvex optimization is experiencing a heyday!

- nonlinear least squares
- training deep neural networks
- PDE-constrained optimization

History

Nonlinear optimization theory and algorithms have had parallel developments

convexity

Rockafellar
Fenchel
Nemirovski
Nesterov

subgradient inequality

convergence, complexity guarantees

smoothness

Powell
Fletcher
Goldfarb
Nocedal

sufficient decrease

convergence, fast local convergence

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

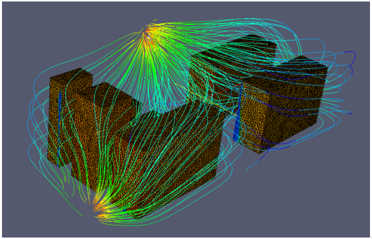
My work: Inexact Newton Methods

Doctoral work, postdoc, and first few years as asst. prof.:

- Inexact Newton and interior-point methods for solving large-scale problems
- Motivated primarily by PDE-constrained optimization
- Software available in `Ipopt/Pardiso`

$$\begin{bmatrix} W_k & J_k^T \\ J_k & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \delta_k \end{bmatrix} = - \begin{bmatrix} g_k + J_k^T \lambda_k \\ c_k \end{bmatrix}$$

- Iterative Krylov methods
- Inexactness conditions

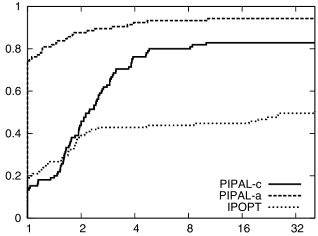


- Theory: Emphasis on preserving global and fast local convergence guarantees
- Have to deal with nonconvexity and rank deficiency issues

My work: Infeasibility Detection

Postdoc and first few years as asst. prof.:

- State-of-the-art packages fail at infeasibility detection!
- `IPOPT`, `KNITRO`, `LOQO`, `SNOPT`, etc.
- Designed additional steps / new algorithms that overcome this deficiency



- Theory: Emphasis on completing the table...

	Convergence	Fast Local Convergence
Feasible	✓	✓
Infeasible	✓	✓

My work: Nonconvex, Nonsmooth Optimization

Postdoc and first few years as asst. prof.:

- Adaptive gradient sampling and other types of methods
- More on this later...

Early 2010's

Back to the colliding worlds. . .

Complexity guarantees for nonconvex optimization algorithms

- Iterations or function/derivative evaluations to achieve

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

- 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): $\mathcal{O}(\epsilon^{-3/2})$

Early 2010's

Back to the colliding worlds...

Complexity guarantees for nonconvex optimization algorithms

- Iterations or function/derivative evaluations to achieve

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

- 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

My work: Trust Region Methods with Optimal Complexity

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*!

My work: Trust Region Methods with Optimal Complexity

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

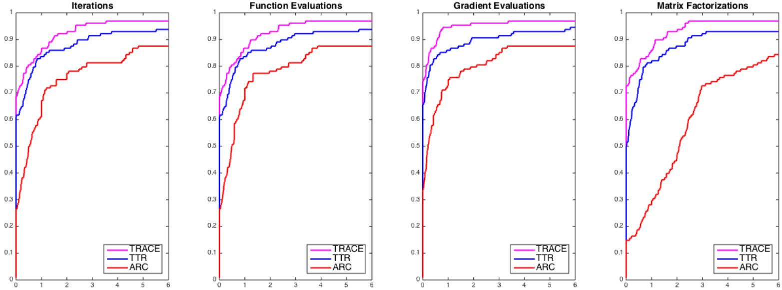
- Agarwal, Allen-Zhu, Bullins, Hazan, Ma (2017)
- Carmon, Duchi (2017)
- Kohler, Lucchi (2017)
- Peng, Roosta-Khorasan, Mahoney (2017)

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

My work: Trust Region Methods with Optimal Complexity



My view: Message of this Talk

Nonconvex optimization is experiencing a heyday!

- People want to solve more complicated problems
- . . . involving nonsmoothness
- . . . involving stochasticity

My view: Message of this Talk

Nonconvex optimization is experiencing a heyday!

- People want to solve more complicated problems
- …involving nonsmoothness
- …involving stochasticity

However, we might waste this opportunity 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

First- versus Second-Order

First-order methods follow a steepest descent methodology:

$$x_{k+1} \leftarrow x_k - \alpha_k \nabla f(x_k)$$

Second-order methods follow Newton's methodology:

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

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

$$f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k)$$

First- versus Quasi-Second-Order

First-order methods follow a steepest descent methodology:

$$x_{k+1} \leftarrow x_k - \alpha_k \nabla f(x_k)$$

Second-order methods follow Newton's methodology:

$$x_{k+1} \leftarrow x_k - \alpha_k W_k \nabla f(x_k),$$

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

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

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

Why Second-Order?

For better complexity properties?

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)

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?

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

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

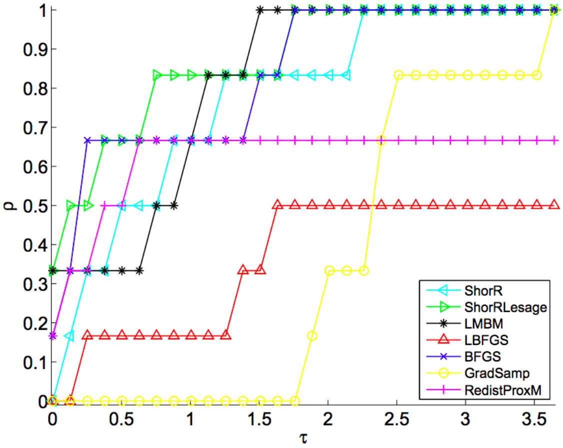
Then why?

- 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...)

Nonsmooth Optimization

Few comparisons between first- and second-order methods, but here's one:

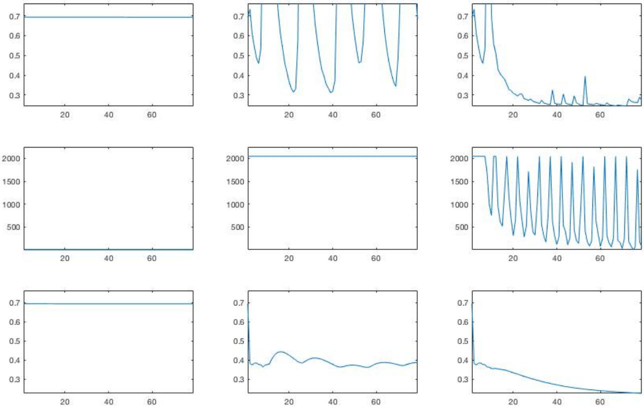


Skajaa (2010) (Master's thesis advised by Overton)

Stochastic Optimization: No Parameter Tuning

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

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

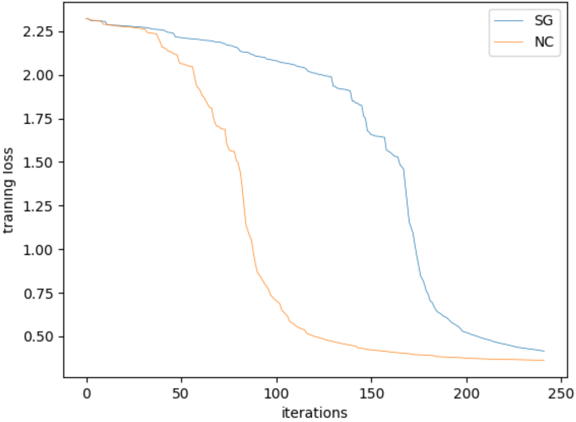


Minimizing logistic loss for binary classification with RCV1 dataset

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:

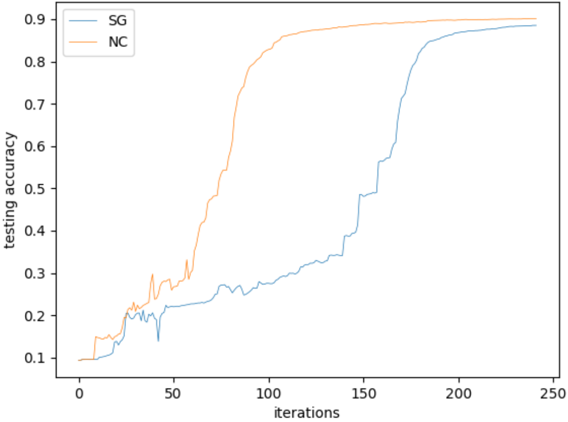


Overcomes slow initial progress by SG-type method…

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

Outline

Quasi-Newton Methodology

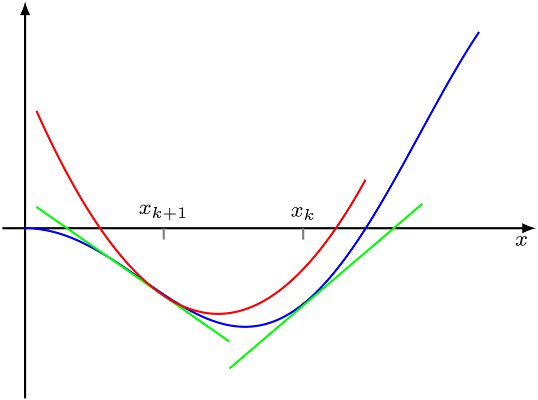
Quasi-Newton step:

$$x_{k+1} \leftarrow x_k - \alpha_k W_k \nabla f(x_k)$$

How should we choose W_k?

Standard Motivation

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



Secant equation $H_k y_k = s_k$ to match gradient of f at x_k, where

$$s_k := x_{k+1} - x_k \quad \text{and} \quad y_k := \nabla f(x_{k+1}) - \nabla f(x_k)$$

But BFGS offers more!

All quasi-Newton methods use this idea, but all are not equal!

- Broyden (1970)
- Fletcher (1970)
- Goldfarb (1970)
- Shanno (1970)

The critical properties of BFGS took a few extra years to come into focus:

- Powell (1976)
- Ritter (1979, 1981)
- Werner (1978)
- Byrd, Nocedal (1989)

BFGS-type updates

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

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

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

- These satisfy secant-type equations

$$W_{k+1} v_k = s_k \quad \text{and} \quad H_{k+1} s_k = v_k,$$

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

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 $\frac{v_k v_k^T}{s_k^T v_k} = \left(\frac{v_k v_k^T}{\|v_k\|_2^2}\right)\left(\frac{\|v_k\|_2^2}{v_k^T W_{k+1} v_k}\right)$ (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 (Byrd, Nocedal (1989))

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

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

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 $k \in \{1, \dots, K\}$:

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

Proof technique.

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

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

□

Self-correcting properties of inverse Hessian update

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

Corollary

Suppose the conditions of Theorem 1 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 $k \in \{1, \ldots, K\}$:

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

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

$$x_{k+1} - x_k = s_k = -W_k \bar{g}_k$$

Proof sketch.

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

Subproblems in nonsmooth optimization algorithms

With sets of points, scalars, and (sub)gradients

$$\{x_{k,j}\}_{j=1}^m, \quad \{f_{k,j}\}_{j=1}^m, \quad \{g_{k,j}\}_{j=1}^m,$$

nonsmooth optimization methods involve the primal subproblem

$$\begin{aligned} \min_{x\in\mathbb{R}^n} & \left(\max_{j\in\{1,\dots,m\}} \{f_{k,j} + g_{k,j}^T(x - x_{k,j})\} + \tfrac{1}{2}(x-x_k)^T H_k (x - x_k) \right) \\ \text{s.t. } & \|x - x_k\| \leq \delta_k, \end{aligned} \tag{P}$$

but, with $G_k \leftarrow [g_{k,1} \ \cdots \ g_{k,m}]$, it is typically more efficient to solve the dual

$$\begin{aligned} \sup_{(\omega,\gamma)\in\mathbb{R}_+^m\times\mathbb{R}^n} & -\tfrac{1}{2}(G_k\omega + \gamma)^T W_k (G_k\omega + \gamma) + b_k^T\omega - \delta_k\|\gamma\|_* \\ \text{s.t. } & \mathbb{1}_m^T\omega = 1. \end{aligned} \tag{D}$$

The primal solution can then be recovered by

$$x_k^* \leftarrow x_k - W_k \underbrace{(G_k\omega_k + \gamma_k)}_{\tilde{g}_k}.$$

Algorithm Self-Correcting BFGS for Nonsmooth Optimization

Choose $x_1 \in \mathbb{R}^n$.
Choose a symmetric positive definite $W_1 \in \mathbb{R}^{n \times n}$.
Choose $\alpha \in (0, 1)$
for $k = 1, 2, \ldots$ **do**
Solve (P)–(D) such that setting

$$G_k \leftarrow \begin{bmatrix} g_{k,1} & \cdots & g_{k,m} \end{bmatrix},$$
$$s_k \leftarrow -W_k(G_k\omega_k + \gamma_k),$$
$$\text{and } \; x_{k+1} \leftarrow x_k + s_k$$

yields

$$f(x_{k+1}) \leq f(x_k) - \tfrac{1}{2}\alpha(G_k\omega_k + \gamma_k)^T W_k(G_k\omega_k + \gamma_k).$$

Choose $\tilde{y}_k \in \mathbb{R}^n$.
Set $\beta_k \leftarrow \min\{\beta \in [0,1] : v(\beta) := \beta s_k + (1-\beta)\tilde{y}_k$ satisfies $(\star)\}$.
Set $v_k \leftarrow v(\beta_k)$.
Set

$$W_{k+1} \leftarrow \left(I - \frac{v_k s_k^T}{s_k^T v_k}\right)^T W_k \left(I - \frac{v_k s_k^T}{s_k^T v_k}\right) + \frac{s_k s_k^T}{s_k^T v_k}.$$

end for

Instances of the framework

Cutting plane / bundle methods

- Points added incrementally until sufficient decrease obtained
- Finite number of additions until accepted step

Gradient sampling methods

- Points added randomly / incrementally until sufficient decrease obtained
- Sufficient number of iterations with "good" steps

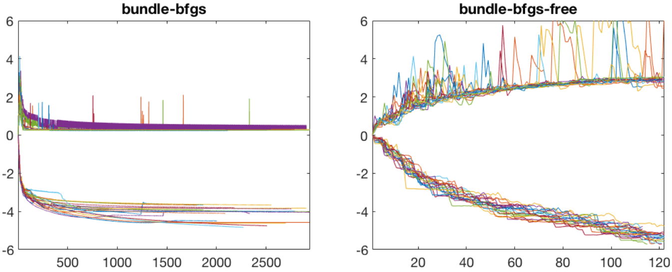
In any case: convergence guarantees require $\{W_k\}$ to be uniformly positive definite and bounded *on a sufficient number of accepted steps*

C++ implementation: NonOpt (sabbatical project)

BFGS w/ weak Wolfe line search							
Name	Exit	ϵ_{end}	$f(x_{\text{end}})$	#iter	#func	#grad	#subs
maxq	Stationary	+9.77e-05	+2.26e-07	450	1017	452	451
mxhilb	Stepsize	+3.13e-03	+9.26e-02	101	1886	113	102
chained lq	Stepsize	+5.00e-02	-6.93e+01	205	4754	207	206
chained cb3 1	Stepsize	+1.00e-01	+9.80e+01	347	7469	348	348
chained cb3 2	Stepsize	+1.00e-01	+9.80e+01	64	1496	69	65
active faces	Stepsize	+2.50e-02	+2.22e-16	24	672	27	25
brown function 2	Stepsize	+1.00e-01	+2.04e-05	395	17259	396	396
chained mifflin 2	Stepsize	+5.00e-02	-3.47e+01	476	10808	508	477
chained crescent 1	Stepsize	+1.00e-01	+2.18e-01	74	2278	91	75
chained crescent 2	Stepsize	+1.00e-01	+5.86e-02	313	7585	334	314

Bundle method with self-correcting properties							
Name	Exit	ϵ_{end}	$f(x_{\text{end}})$	#iter	#func	#grad	#subs
maxq	Stationary	+9.77e-05	+1.04e-06	193	441	635	440
mxhilb	Stationary	+9.77e-05	+2.25e-05	39	338	351	137
chained lq	Stationary	+9.77e-05	-6.93e+01	29	374	398	366
chained cb3 1	Stationary	+9.77e-05	+9.80e+01	50	1038	1069	1017
chained cb3 2	Stationary	+9.77e-05	+9.80e+01	29	174	204	173
active faces	Stationary	+9.77e-05	+2.09e-02	17	387	165	32
brown function 2	Stationary	+9.77e-05	+2.49e-03	232	10094	9674	9438
chained mifflin 2	Stationary	+9.77e-05	-3.48e+01	393	24410	19493	18924
chained crescent 1	Stationary	+9.77e-05	+2.73e-04	30	66	92	59
chained crescent 2	Stationary	+9.77e-05	+4.36e-05	137	6679	6140	5997

Minimum and maximum eigenvalues



Outline

Perspectives on Nonconvex Optimization

Motivation for Second-Order Methods

Self-Correcting Properties of BFGS

Nonsmooth Optimization

Stochastic Optimization

Summary

Stochastic Gradient (SG)

SG and its variants are the state-of-the-art:

$$x_{k+1} \leftarrow x_k - \alpha_k g_k \quad \text{where} \quad \mathbb{E}_k[g_k] = \nabla f(x_k)$$

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

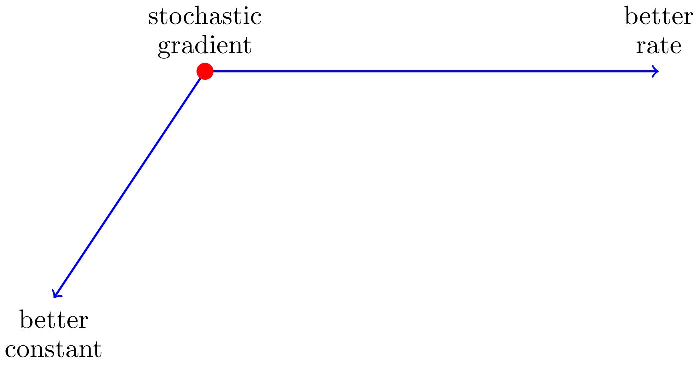
- Stability of SG; Hardt, Recht, Singer (2015)
- SG avoids steep minima; Keskar, Mudigere, Nocedal, Smelyanskiy (2016)
- . . . (many more)

No, we should want more. . .

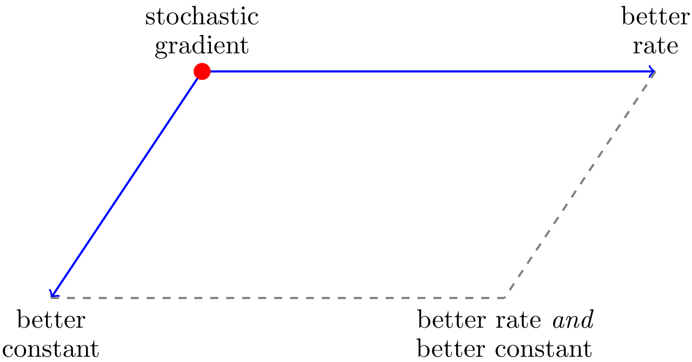
- SG requires a lot of tuning
- Sublinear convergence is not satisfactory
- . . . "linearly" convergent method eventually wins
- . . . with higher budget, faster computation, parallel?, distributed?

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

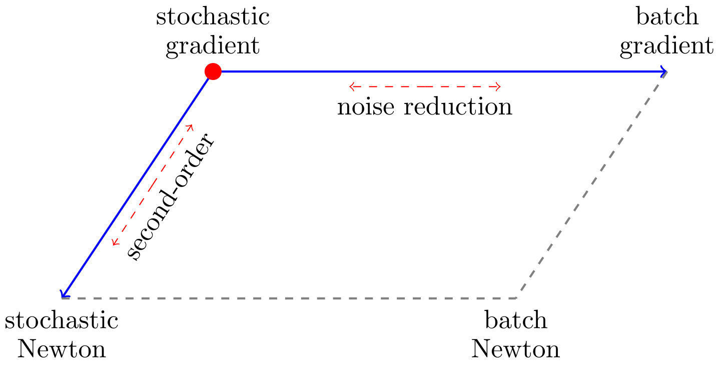
What can be improved?



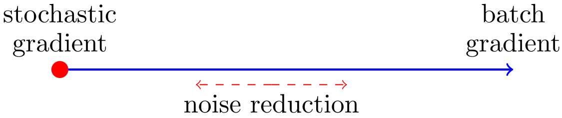
What can be improved?



Two-dimensional schematic of methods

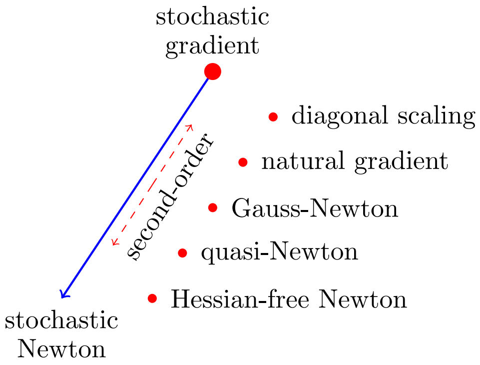


2D schematic: Noise reduction methods



- dynamic sampling
- gradient aggregation
- iterate averaging

2D schematic: Second-order methods



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:

$$y_k \leftarrow \underbrace{\nabla f(w_{k+1}, \xi_k) - \nabla f(w_k, \xi_k)}_{\substack{\text{use same seed} \\ \text{oLBFGS, Schraudolph et al. (2007)} \\ \text{SGD-QN, Bordes et al. (2009)} \\ \text{RES, Mokhtari \& Ribeiro (2014)}}}$$

$$\text{or} \quad y_k \leftarrow \underbrace{\left(\sum_{i \in \mathcal{S}_k^H} \nabla^2 f(w_{k+1}, \xi_{k+1,i}) \right) s_k}_{\substack{\text{use action of step on subsampled Hessian} \\ \text{SQN, Byrd et al. (2015)}}}$$

I believe this is the wrong focus

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 $k = 1, 2, \dots$ **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

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

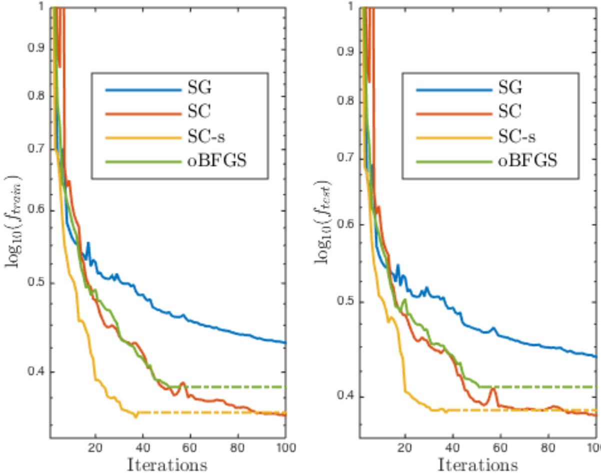
Proof technique.

Follows from the critical inequality

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

□

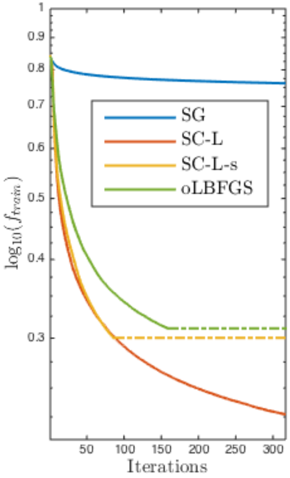
Numerical Experiments: a1a



logistic regression, data `a1a`, diminishing stepsizes

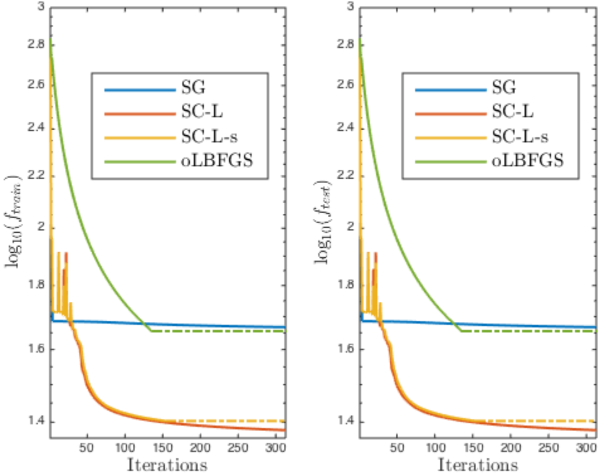
Numerical Experiments: rcv1

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

Summary

Nonconvex optimization is experiencing a heyday!

- People want to solve more complicated problems
- . . . involving nonsmoothness
- . . . involving stochasticity

However, we might waste this opportunity 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

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...)

References

For references, please see

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

Please also visit the OptML @ Lehigh website!

- http://optml.lehigh.edu