ORIE 6326: Convex Optimization

Stochastic subgradient methods

Professor Udell

Operations Research and Information Engineering
Cornell

May 8, 2017

Minimizing a sum

$$\text{minimize} \quad \frac{1}{m} \sum_{i=1}^{m} f_i(x)$$

examples:

- least squares: $f_i(x) = (a_i^T x - b_i)^2$
- logistic regression: $f_i(x) = -\log(1 + \exp(-b_i a_i^T x))$
- maximum likelihood estimation: $f_i(x)$ is -loglik of observation i given parameter x
- machine learning: f_i is misfit of model x on example i

Minimizing a sum

solve

$$\text{minimize} \quad \frac{1}{m}\sum_{i=1}^{m} f_i(x)$$

with variable $x \in \mathbf{R}^n$

quandary:

- solving a problem with **more data** should be **easier**
- but complexity of algorithms increases with m!

goal: find algorithms that work **better** given **more** data
(or at least, not worse)

Minimizing a sum

solve

$$\text{minimize} \quad \frac{1}{m}\sum_{i=1}^{m} f_i(x)$$

with variable $x \in \mathbf{R}^n$

quandary:

- solving a problem with **more data** should be **easier**
- but complexity of algorithms increases with m!

goal: find algorithms that work **better** given **more** data
(or at least, not worse)
idea:

Minimizing a sum

solve

$$\text{minimize} \quad \frac{1}{m}\sum_{i=1}^{m} f_i(x)$$

with variable $x \in \mathbf{R}^n$

quandary:

- solving a problem with **more data** should be **easier**
- but complexity of algorithms increases with m!

goal: find algorithms that work **better** given **more** data
(or at least, not worse)
idea: throw away data! (cleverly)

Minimizing an expectation

solve

$$\text{minimize} \quad \mathbf{E} f(x) = \mathbf{E}_\omega f(x;\omega)$$

with variable $x \in \mathbf{R}^n$

- random loss function f
- or equivalently, function $f(\cdot;\omega)$ of random variable ω

examples: **observations** $\omega = (a, b)$ is random

- least squares: $f(x;\omega) = (a^T x - b)^2$
- logistic regression: $f(x;\omega) = -\log(1 + \exp(-ba^T x))$
- maximum likelihood estimation: $f(x;\omega)$ is -loglik of observation ω given parameter x
- machine learning: $f(x;\omega)$ is misfit of model x on example ω

Outline

Stochastic subgradient

random vector $\tilde{g} \in \mathbf{R}^n$ is a **stochastic subgradient** of $f : \mathbf{R}^n \to \mathbf{R}^n$ at $x \in \mathbf{R}^n$ if

$$\mathbf{E}\tilde{g} \in \partial f(x)$$

ie, for all z

$$f(z) \geq f(x) + (\mathbf{E}\tilde{g})^T(z - x).$$

($\tilde{g}$ is a stochastic gradient if $\mathbf{E}\tilde{g} \in \nabla f(x)$)

- equivalently, $\tilde{g} = g + v$, where $g \in \partial f(x)$, $\mathbf{E}v = 0$
- v can represent error in computing g, measurement noise, Monte Carlo sampling error, etc.

Stochastic subgradient method

stochastic subgradient method is the subgradient method, using stochastic subgradients

$$x^{(k+1)} = x^{(k)} - t_k \tilde{g}^{(k)}$$

- $x^{(k)}$ is kth iterate
- $\tilde{g}^{(k)}$ is any noisy unbiased subgradient of (convex) f at $x^{(k)}$, *i.e.*,

$$\mathbf{E}(\tilde{g}^{(k)} | x^{(k)}) = g^{(k)} \in \partial f(x^{(k)})$$

- $t_k > 0$ is the kth step size

Stochastic gradient descent

Q: is stochastic subgradient method a descent method?

Stochastic gradient descent

Q: is stochastic subgradient method a descent method?
A: **no!** stochastic subgradient could even be an **ascent** direction

$$(\tilde{g}^{(k)})^T g^{(k)} > 0$$

Stochastic gradient descent

Q: is stochastic subgradient method a descent method?
A: **no!** stochastic subgradient could even be an **ascent** direction

$$(\tilde{g}^{(k)})^T g^{(k)} > 0$$

unfortunately,

- stochastic subgradient methods are widely used in machine learning
- especially to (approximately) optimize (nonconvex, nonsmooth) deep neural networks
- machine learning practitioners are less careful about nomenclature than optimizers
- they call it **stochastic gradient descent** (SGD)
- it is neither a descent method nor a gradient method

use the term SGD at your own peril.

Assumptions

- $f^\star = \inf_x f(x) > -\infty$, with $f(x^\star) = f^\star$
- $\mathbf{E}\|g^{(k)}\|_2^2 \leq G^2$ for all k
- $\mathbf{E}\|x^{(1)} - x^\star\|_2^2 \leq R^2$ (can take $=$ here)
- step sizes are square-summable but not summable

$$t_k \geq 0, \qquad \sum_{k=1}^{\infty} t_k^2 = \|t\|_2^2 < \infty, \qquad \sum_{k=1}^{\infty} t_k = \infty$$

these assumptions are stronger than needed, just to simplify proofs

Notation

as before, define

- best objective value $f_{\text{best}}^{(k)} := \min\{f(x^{(1)}), \ldots, f(x^{(k)})\}$
- averaged iterate $\bar{x}^{(k)} = \frac{1}{k}\sum_{i=1}^{k} x^{(i)}$

by optimality,

$$f_{\text{best}}^{(k)} = \min\{f(x^{(1)}), \ldots, f(x^{(k)})\} \leq \frac{1}{k}\sum_{i=1}^{k} f(x^{(i)})$$

and by convexity,

$$f(\bar{x}^{(k)}) \leq \frac{1}{k}\sum_{i=1}^{k} f(x^{(i)})$$

Convergence results

- convergence in expectation:

$$\lim_{k\to\infty} f_{\text{best}}^{(k)} = f^\star$$

- convergence in probability: for any $\epsilon > 0$,

$$\lim_{k\to\infty} \textbf{prob}(f_{\text{best}}^{(k)} \geq f^\star + \epsilon) = 0$$

- almost sure convergence:

$$\lim_{k\to\infty} f_{\text{best}}^{(k)} = f^\star$$

  a.s. (we won't show this)

Convergence proof

key quantity: expected square Euclidean distance to the optimal set

$$\mathbf{E}\left(\|x^{(k+1)}-x^\star\|_2^2 \;\middle|\; x^{(k)}\right) = \mathbf{E}\left(\|x^{(k)}-t_k\tilde{g}^{(k)}-x^\star\|_2^2 \;\middle|\; x^{(k)}\right)$$

$$\begin{aligned}
&= \|x^{(k)}-x^\star\|_2^2 - 2t_k\mathbf{E}\left(\tilde{g}^{(k)T}(x^{(k)}-x^\star)\,\middle|\, x^{(k)}\right) + t_k^2\mathbf{E}\left(\|\tilde{g}^{(k)}\|_2^2\,\middle|\, x^{(k)}\right)\\
&= \|x^{(k)}-x^\star\|_2^2 - 2t_k\mathbf{E}(\tilde{g}^{(k)}|x^{(k)})^T(x^{(k)}-x^\star) + t_k^2\mathbf{E}\left(\|\tilde{g}^{(k)}\|_2^2 \;\middle|\; x^{(k)}\right)\\
&\leq \|x^{(k)}-x^\star\|_2^2 - 2t_k(f(x^{(k)})-f^\star) + t_k^2\mathbf{E}\left(\|\tilde{g}^{(k)}\|_2^2 \;\middle|\; x^{(k)}\right)
\end{aligned}$$

using $\mathbf{E}(\tilde{g}^{(k)}|x^{(k)}) \in \partial f(x^{(k)})$

now take full expectation:

$$\mathbf{E}\|x^{(k+1)} - x^\star\|_2^2 \leq \mathbf{E}\|x^{(k)} - x^\star\|_2^2 - 2t_k(\mathbf{E}f(x^{(k)}) - f^\star) + t_k^2\mathbf{E}\|\tilde{g}^{(k)}\|_2^2$$

apply recursively, and use $\mathbf{E}\|\tilde{g}^{(k)}\|_2^2 \leq G^2$ to get

$$\mathbf{E}\|x^{(k+1)} - x^\star\|_2^2 \leq \mathbf{E}\|x^{(1)} - x^\star\|_2^2 - 2\sum_{i=1}^k t_i(\mathbf{E}f(x^{(i)}) - f^\star) + G^2\sum_{i=1}^k t_i^2$$

and so

$$\min_{i=1,\ldots,k}(\mathbf{E}f(x^{(i)}) - f^\star) \leq \frac{R^2 + G^2\|t\|_2^2}{2\sum_{i=1}^k t_i}$$

if $t_i = t$ constant, also get

$$(\mathbf{E}f(\bar{x}^{(i)}) - f^\star) \leq \frac{R^2 + G^2\|t\|_2^2}{2tk}$$

- we conclude $\min_{i=1,\ldots,k} \mathbf{E} f(x^{(i)}) \to f^\star$
- Jensen's inequality and concavity of minimum yields

$$\mathbf{E} f_{\rm best}^{(k)} = \mathbf{E} \min_{i=1,\ldots,k} f(x^{(i)}) \leq \min_{i=1,\ldots,k} \mathbf{E} f(x^{(i)})$$

  so $\mathbf{E} f_{\rm best}^{(k)} \to f^\star$ (convergence in expectation)
- Markov's inequality: for $\epsilon > 0$

$$\mathbf{prob}(f_{\rm best}^{(k)} - f^\star \geq \epsilon) \leq \frac{\mathbf{E}(f_{\rm best}^{(k)} - f^\star)}{\epsilon}$$

  righthand side goes to zero, so we get convergence in probability

Example

piecewise linear minimization

$$\text{minimize} \quad f(x) = \max_{i=1,\ldots,m}(a_i^T x + b_i)$$

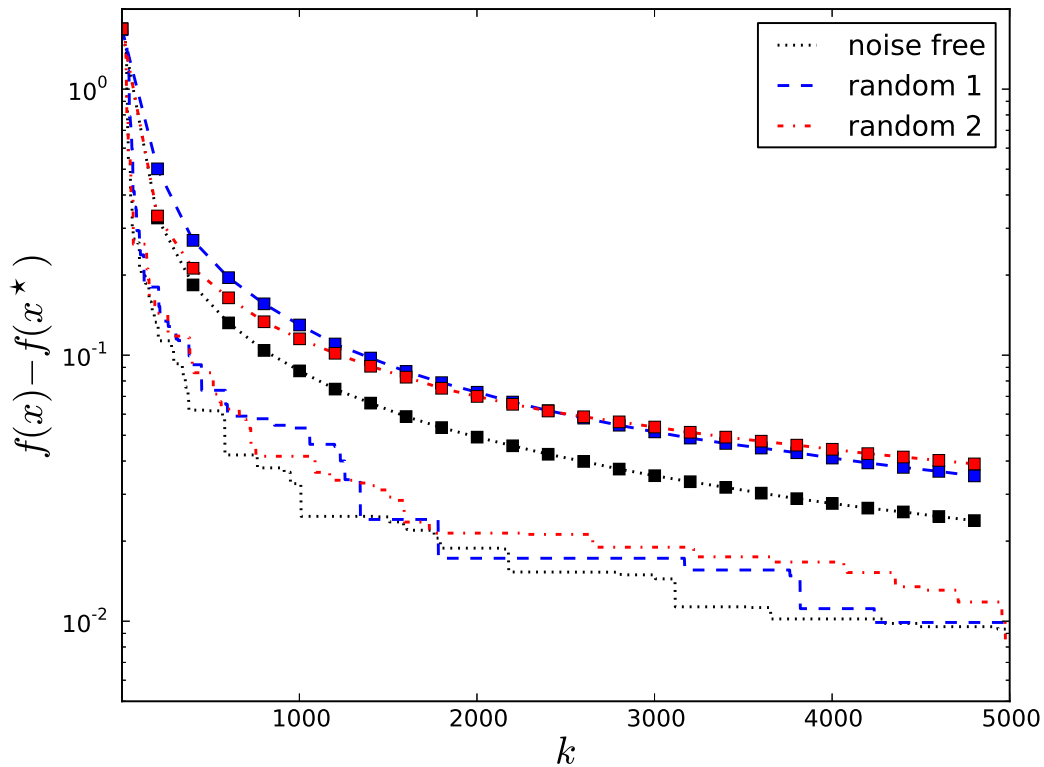
we use stochastic subgradient algorithm with noisy subgradient

$$\tilde{g}^{(k)} = g^{(k)} + v^{(k)}, \qquad g^{(k)} \in \partial f(x^{(k)})$$

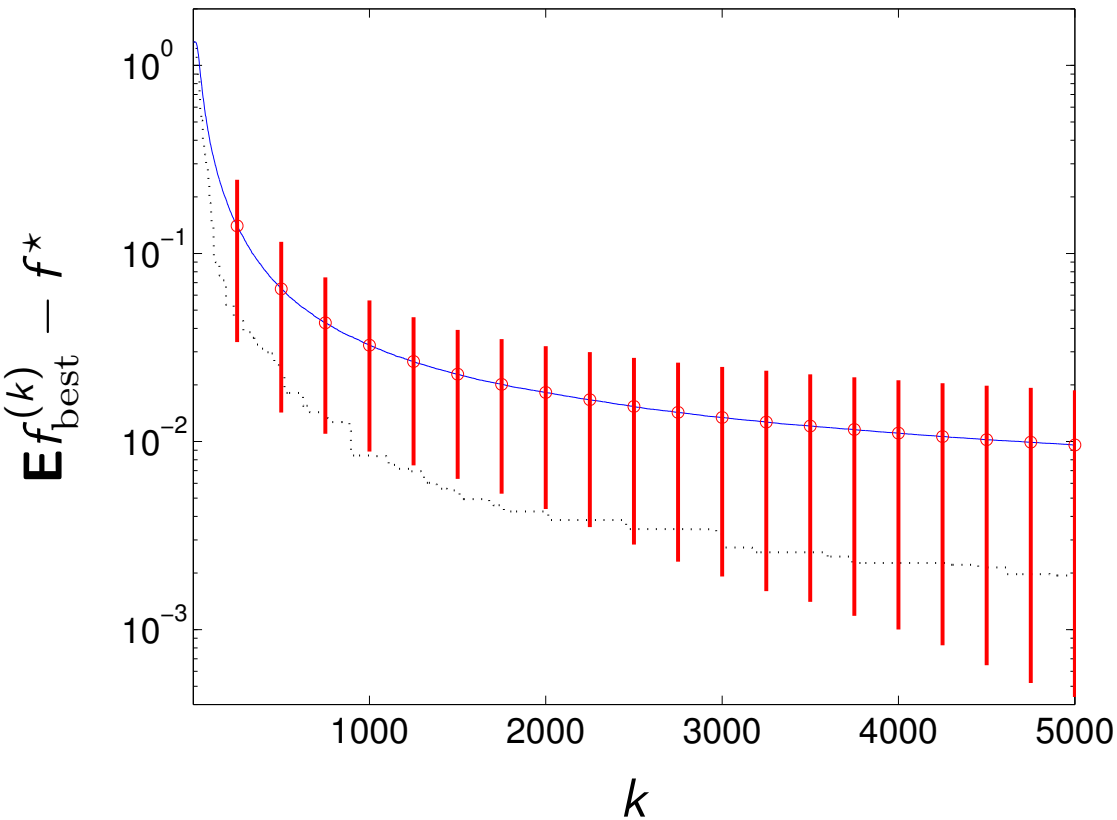
$v^{(k)}$ independent zero mean random variables

problem instance: $n = 20$ variables, $m = 100$ terms, $f^\star \approx 1.1$, $t_k = 1/k$

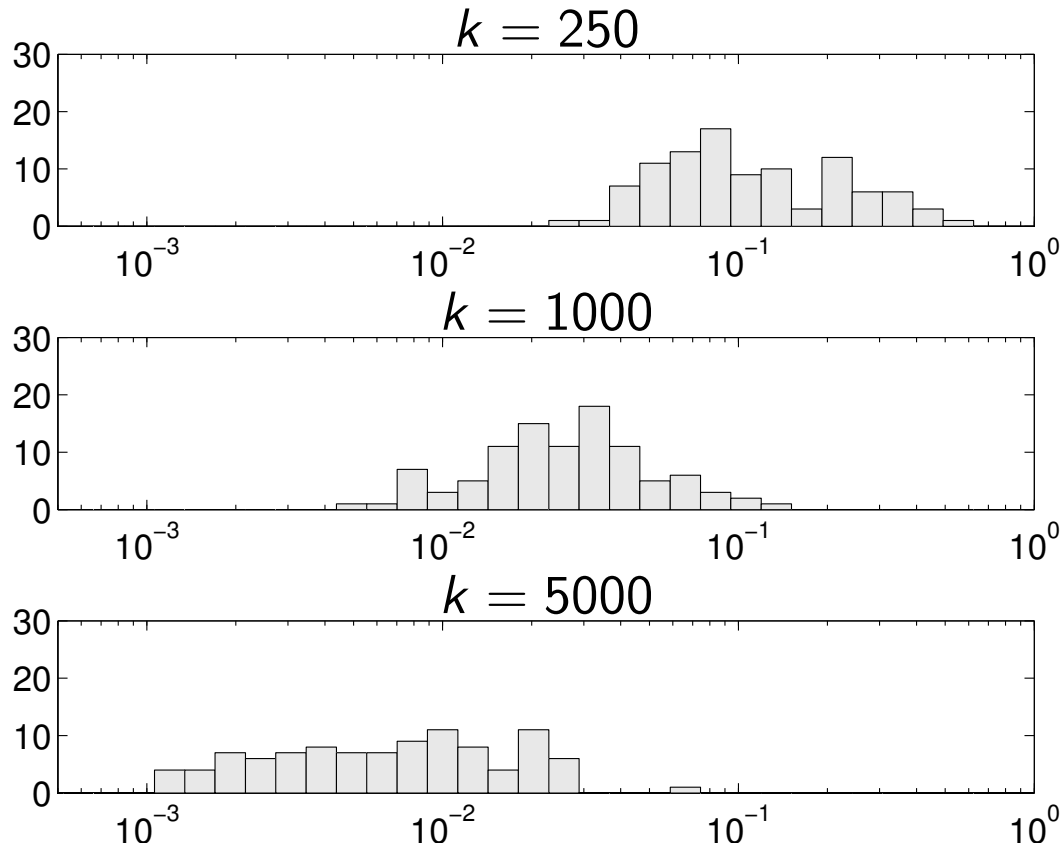
$v^{(k)}$ are IID $\mathcal{N}(0, 0.5I)$ (25% noise since $\|g\| \approx 4.5$)



average and one std. dev. for $f_{\text{best}}^{(k)} - f^\star$ over 100 realizations



empirical distributions of $f_{\mathrm{best}}^{(k)} - f^\star$ at $k = 250$, $k = 1000$, and $k = 5000$



Expected value of convex function

suppose $F(x, w)$ is convex in x for each w and $G(x, w) \in \partial_x F(x, w)$

- $f(x) = \mathbf{E}F(x, w) = \int F(x, w)p(w)\, dw$ is convex
- a subgradient of f at x is

$$g = \mathbf{E}G(x, w) = \int G(x, w)p(w)\, dw \in \partial f(x)$$

- a noisy unbiased subgradient of f at x is

$$\tilde{g} = \frac{1}{M} \sum_{i=1}^{M} G(x, w_i)$$

where $w_1, \ldots, w_M$ are M independent samples (Monte Carlo)

Example: Expected value of piecewise linear function

$$\text{minimize} \quad f(x) = \mathbf{E} \max_{i=1,\ldots,m}(a_i^T x + b_i)$$

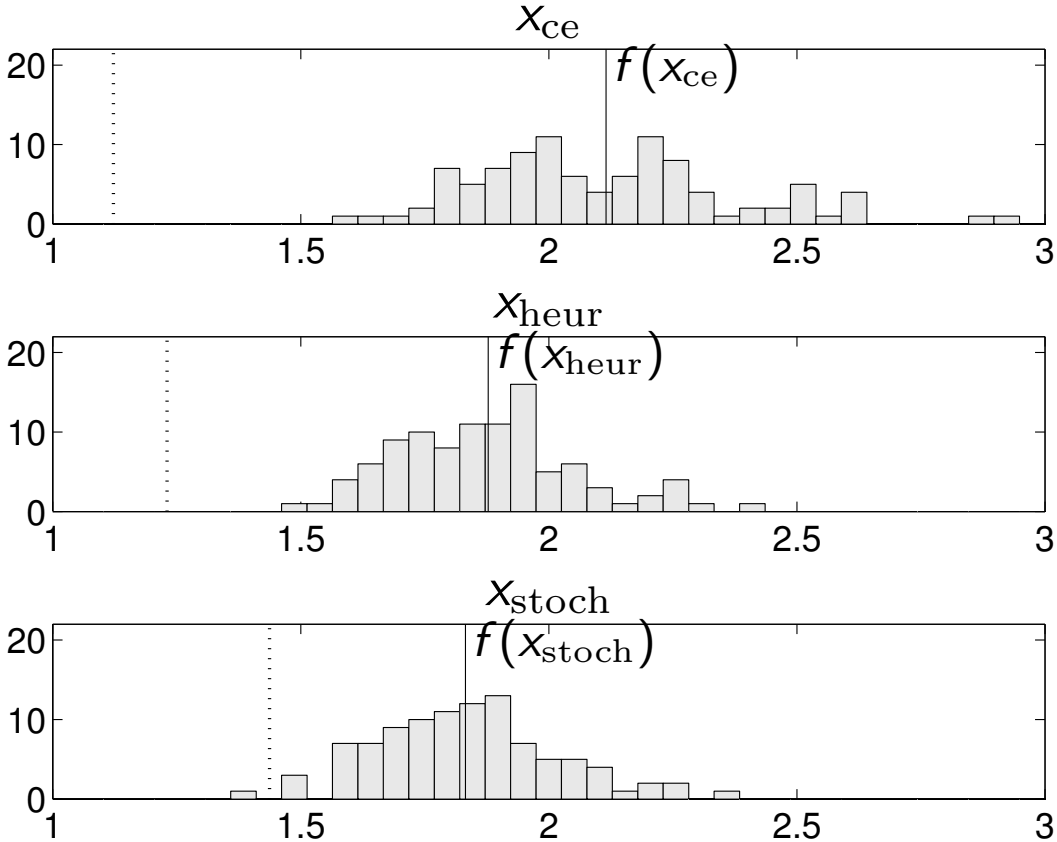
where a_i and b_i are random

evaluate noisy subgradient using Monte Carlo method with M samples, and run stochastic subgradient method

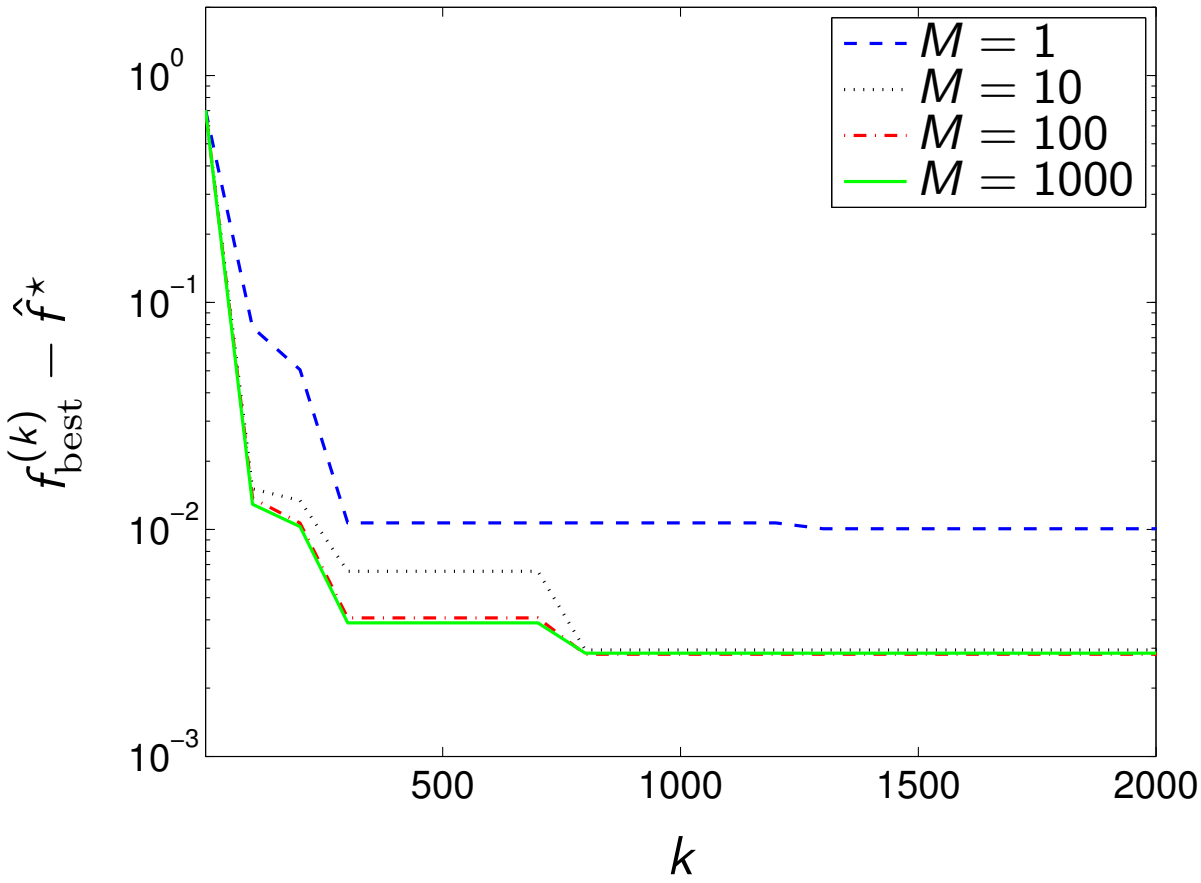
compare to:

- certainty equivalent: minimize $f_{\mathrm{ce}}(x) = \max_{i=1,\ldots,m}(\mathbf{E}a_i^T x + \mathbf{E}b_i)$
- heuristic: minimize $f_{\mathrm{heur}}(x) = \max_{i=1,\ldots,m}(\mathbf{E}a_i^T x + \mathbf{E}b_i + \lambda\|x\|_2)$

problem instance: $n = 20$, $m = 100$, $a_i \sim \mathcal{N}(\bar{a}_i, 5I)$, $b \sim \mathcal{N}(\bar{b}, 5I)$, $\|a_i\|_2 \approx 5$, $\|b\|_2 \approx 10$, x_{stoch} computed using $M = 100$



$f^\star \approx 1.34$ estimated by running the method with $M = 1000$ for long time



Online learning and adaptive signal processing

- $(x, y) \in \mathbf{R}^n \times \mathbf{R}$ have some joint distribution
- find weight vector $w \in \mathbf{R}^n$ for which $w^T x$ is a good estimator of y
- choose w to minimize expected value of a convex **loss function** ℓ

$$f(w) = \mathbf{E}\ell(w^T x - y)$$

  - $\ell(u) = u^2$: mean square error
  - $\ell(u) = |u|$: mean absolute error
- at each step (*e.g.*, time sample), we are given a sample $(x^{(k)}, y^{(k)})$ from the distribution

noisy unbiased subgradient of f at $w^{(k)}$, based on sample $x^{(k)}, y^{(k)}$:

$$g^{(k)} = \ell'(w^{(k)T}x^{(k)} - y^{(k)})x^{(k)}$$

where ℓ' is the derivative (or a subgradient) of ℓ

online algorithm:

$$w^{(k+1)} = w^{(k)} - t_k\ell'(w^{(k)T}x^{(k)} - y^{(k)})x^{(k)}.$$

- for $\ell(u) = u^2$, gives the LMS (least mean-square) algorithm
- for $\ell(u) = |u|$, gives the **sign** algorithm
- $w^{(k)T}x^{(k)} - y^{(k)}$ is the prediction error

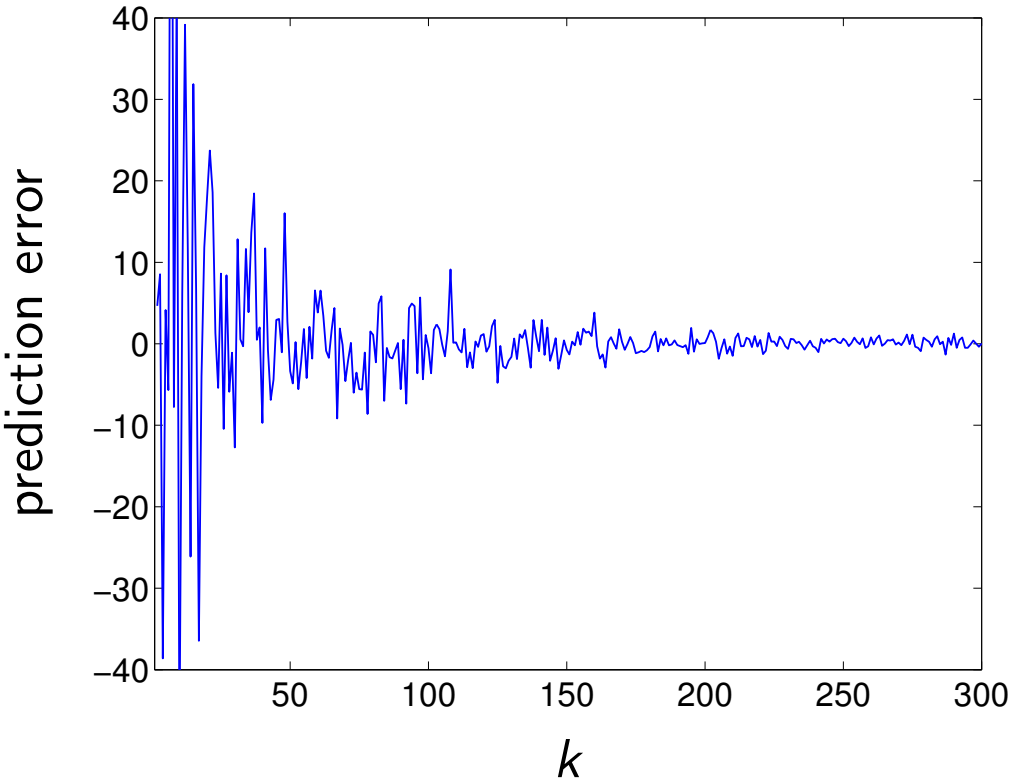
generalization guarantee: online algorithm converges to minimizer of

$$f(w) = \mathbf{E}\ell(w^Tx - y),$$

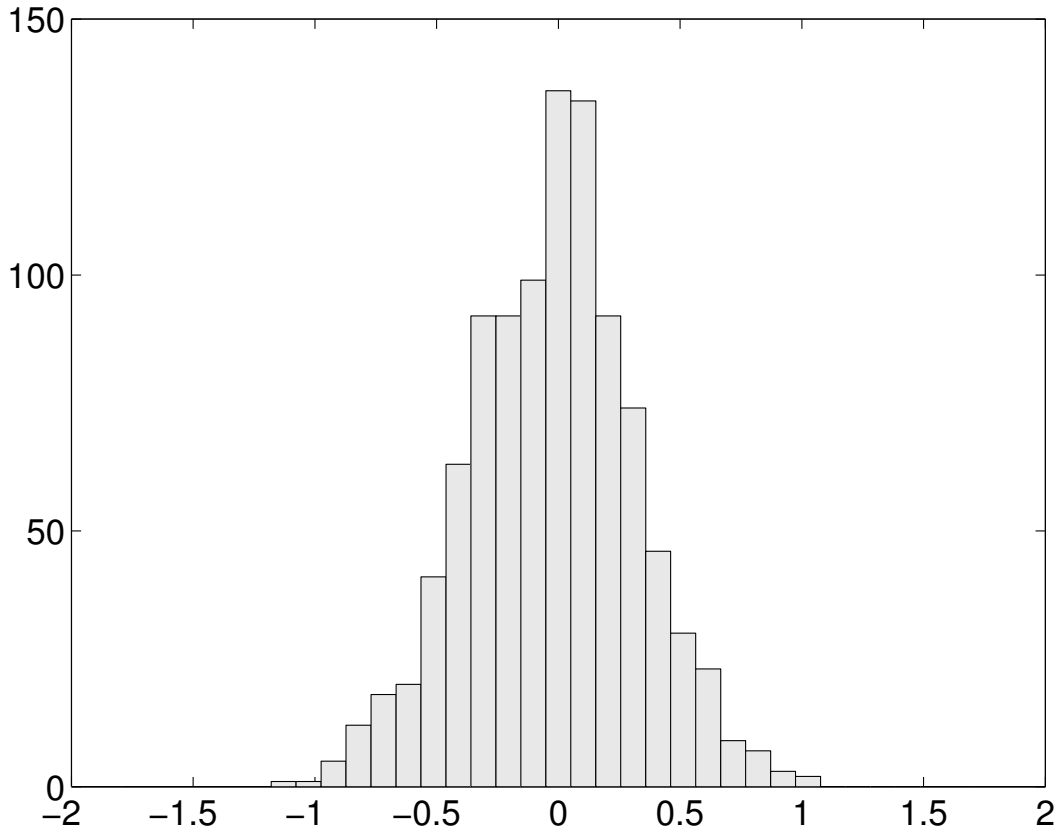
not minimizer of sample average $\frac{1}{k}\sum_{i=1}^{k}\ell(w^Tx^{(i)} - y^{(i)})$

Example: Mean absolute error minimization

problem instance: $n = 10$, $(x, y) \sim \mathcal{N}(0, \Sigma)$, Σ random with $\mathbf{E}(y^2) \approx 12$, $t_k = 1/k$



empirical distribution of prediction error for $w^\star$ (over 1000 samples)



Outline

Stochastic gradients

Reduced variance

Stochastic splitting methods

Stochastic adaptive stepsizes

Coordinate descent and asynchronous optimization

SGD requires stepsize $\to 0$

SGD requires $t_k \to 0$ to compensate for variance of gradients

example: suppose u is uniformly distributed on unit sphere $\|u\| = 1$

$$\text{minimize} \quad \mathbf{E}_u \left[\frac{1}{2} \|x - u\|^2 \right]$$

- solution is $x^\star = 0$
- stochastic gradient at solution has norm 1

$$\nabla \left(\frac{1}{2} \|0 - u\|^2 \right) = u$$

- so stochastic gradient step moves away from solution
- solution: need either $t_k \to 0$ or $\mathbf{var}(\tilde{g}) \to 0$
  (can achieve $\mathbf{var}(\tilde{g}) \to 0$ by taking larger minibatches as $k \to \infty$)

Variance reduction

consider minimizing the average of smooth functions f_i

$$\text{minimize} \quad \frac{1}{m}\sum_{i=1}^{m} f_i(x)$$

variance reduction [Johnson Zhang, 2014]:

- every K iterations, record iterate $\hat{x}$ and full gradient $\hat{g} = \nabla f(\hat{x})$.
- if i is chosen randomly from $\{1, \ldots, m\}$,

$$\mathbf{E}\left[\nabla f_i(\hat{x}) - \hat{g}\right] = 0$$

- use this to form a stochastic gradient of f at x

$$\mathbf{E}\left[\nabla f_i(x) - \nabla f_i(\hat{x}) + \hat{g}\right] = \nabla f(x)$$

- variance of this gradient goes to 0 as iterates converge!
  - if $\hat{x} \to x^\star$, then $\hat{g} \to 0$
  - if $x \to x^\star$, then $\nabla f_i(x),\ \nabla f_i(\hat{x}) \to \nabla f_i(x^\star)$

  so

$$\nabla f_i(x) - \nabla f_i(\hat{x}) + \hat{g} \to \nabla f_i(x) - \nabla f_i(\hat{x}) \to 0$$

Stochastic Variance Reduced Gradient (SVRG)

for $f_i : \mathbf{R}^n \to \mathbf{R}$ smooth,

$$\text{minimize} \quad f(x) = \frac{1}{m} \sum_{i=1}^{m} f_i(x)$$

Algorithm 1 SVRG [Johnson Zhang, 2014].

Input: a starting point $x \in \mathbf{dom}\, f$, update frequency K, stepsize t

for $k = 1, 2, \ldots$ **do**
  Snapshot. $\hat{x} := x$, $\hat{g} = \nabla f(\hat{x})$
  for $\ell = 1, \ldots, K$ **do**
    Select function. Randomly choose $i \in \{1, \ldots, m\}$.
    Update. $x := x - t(\nabla f_i(x) - \nabla f_i(\hat{x}) + \hat{g})$.

Guarantee: if each f_i is smooth and f is strongly convex, get linear convergence [Johnson Zhang, 2014].

Outline

Do they play nice?

- with splitting methods: yes
- with quasi-Newton methods: no
- with linesearch: no

Stochastic splitting methods

$f_i : \mathbf{R}^n \to \mathbf{R}$ smooth, $h : \mathbf{R}^n \to \mathbf{R}$ proxable

$$\text{minimize} \quad \frac{1}{m} \sum_{i=1}^{m} f_i(x) + h(x)$$

- take any splitting method (prox grad, dual prox grad, . . .)
- replace gradient ∇f by stochastic subgradient $\tilde{g}$
- generally, still get same convergence results
  (in expectation and in probability) (with step size depending on properties of f_i)

SAGA

$f_i : \mathbf{R}^n \to \mathbf{R}$ smooth, $h : \mathbf{R}^n \to \mathbf{R}$ proxable

$$\text{minimize} \quad \frac{1}{m}\sum_{i=1}^{m} f_i(x) + h(x)$$

Algorithm 2 SAGA [Defazio Bach Lacoste-Julien, 2014].

Input: a starting point $x \in \mathbf{dom}\, f$, stepsize t

1 *Store.* $g_i := \nabla f_i(x)$ for $i = 1, \ldots, m$

2 **for** $k = 1, 2, \ldots$ **do**

3 *Select.* Randomly choose $i \in \{1, \ldots, m\}$.

4 *Update.*

$$\begin{aligned} w &:= x - t\left(\nabla f_i(x) - g_i + \frac{1}{m}\sum_{j=1}^{m} g_j\right) \\ x &:= \mathbf{prox}_{th}(w) \end{aligned}$$

5 *Store.* $g_i := \nabla f_i(x)$

SAGA: guarantees

Guarantees: [Defazio Bach Lacoste-Julien, 2014]

- if f is strongly convex and smooth and $t = \frac{1}{3(\alpha m + \beta)}$, get linear convergence
$$\mathbf{E}\|x^{(k)} - x^\star\| \leq c^k \|x^{(0)} - x^\star\|$$
with rate $c < 1$
- if each f_i is strongly convex too and $t = \frac{1}{2(\alpha m + \beta)}$, get better rate $c' < c < 1$
- if f is smooth but not strongly convex and $t = \frac{1}{3L}$, get $\frac{1}{k}$ convergence of values
$$\mathbf{E} f(x^{(k)}) - f(x^\star) \leq \frac{C}{k}$$

Outline

Stochastic quasi-Newton

with $f_i : \mathbf{R}^n \to \mathbf{R}$ smooth, $i = 1, \ldots, m$, solve

$$\text{minimize} \quad \frac{1}{m} \sum_{i=1}^{m} f_i(x)$$

- idea: use stochastic gradients to form stochastic Hessian approximation?
- main difficulty: how to ensure positive-definiteness of update?

Stochastic quasi-Newton

with $f_i : \mathbf{R}^n \to \mathbf{R}$ smooth, $i = 1, \ldots, m$, solve

$$\text{minimize} \quad \frac{1}{m} \sum_{i=1}^{m} f_i(x)$$

- idea: use stochastic gradients to form stochastic Hessian approximation?
- main difficulty: how to ensure positive-definiteness of update?

recall BFGS update

$$H^+ = H + \frac{yy^T}{y^T s} - \frac{Hss^T H}{s^T Hs}$$

where $s = x^+ - x$, $y = \nabla f(x^+) - \nabla f(x)$

- $y^T s > 0$ if f is convex and gradients are exact
- simple approach: if $(\tilde{g}^+ - \tilde{g})^T (x^+ - x) \leq 0$, don't update Hessian approximation
- more sophisticated approach: [Byrd Hansen Nocedal Singer, 2015]

Stochastic quasi-Newton

with $f_i : \mathbf{R}^n \to \mathbf{R}$ smooth, $i = 1, \ldots, m$, solve

$$\text{minimize} \quad \frac{1}{m} \sum_{i=1}^{m} f_i(x)$$

- idea: use stochastic gradients to form stochastic Hessian approximation?
- main difficulty: how to ensure positive-definiteness of update?

recall BFGS update

$$H^+ = H + \frac{yy^T}{y^T s} - \frac{Hss^T H}{s^T Hs}$$

where $s = x^+ - x$, $y = \nabla f(x^+) - \nabla f(x)$

- $y^T s > 0$ if f is convex and gradients are exact
- simple approach: if $(\tilde{g}^+ - \tilde{g})^T (x^+ - x) \leq 0$, don't update Hessian approximation
- more sophisticated approach: [Byrd Hansen Nocedal Singer, 2015]

sadly, seems not to work well despite valiant attempt

Adaptive step sizes that work

other ideas for adaptive step sizes:

- AdaGrad
- AdaDelta (= AdaGrad with stepsize $\not\to 0$)
- RMSProp (= AdaGrad with stepsize $\not\to 0$)
- Adam (= AdaGrad + momentum)
- ...

comparisons: `http://imgur.com/a/Hqolp`

these are mostly used for deep learning, so lots of info online!

AdaGrad

solve (f_i not necessarily smooth)

$$\text{minimize} \quad f(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x)$$

Algorithm 3 AdaGrad [Duchi Singer Hazan, 2010].

Input: a starting point $x^{(0)} \in \mathbf{dom}\, f$, stepsize t, small constant $\delta > 0$

1 **for** $k = 0, 1, 2, \ldots$ **do**

2 *Compute stochastic subgradient.* $g^{(k)} \in \partial f_i(x^{(k)})$

3 *Update metric.*

- set $s_k = \sum_{i=1}^{k} (g^{(i)})^2$
- set $h_k = \delta \mathbf{1} + \sqrt{s_k}$
- set $H_k = \frac{1}{t}\mathrm{diag}(h_k)$

4 *Update iterate.* $x^{(k+1)} := x^{(k)} - H_k^{-1} g^{(k)}$

AdaGrad – motivation

- for fixed $H_k = H$ we have estimate:

$$f_{\text{best}}^{(k)} - f^\star \leq \frac{R^2 + \sum_{i=1}^{k} \|g^{(i)}\|_{H^{-1}}^2}{2k}$$

- **idea**: Choose $H_k \succ 0$ that minimizes this estimate in hindsight:

$$H_k = \operatorname*{argmin}_H \sum_{i=1}^{k} \|g^{(i)}\|_{H^{-1}}^2$$

  subject to $\operatorname{trace}(H) = c$.

- optimal $H_k = \frac{1}{t}\operatorname{diag}\left(\sqrt{\sum_{i=1}^{k}[g^i]_{11})^2}, \ldots, \sqrt{\sum_{i=1}^{k}[g^i]_{nn})^2}\right)$
- **intuition**: adapt step-length based on historical step lengths

AdaGrad – convergence

- since $h_i \geq h_{i-1}$ and $H_i = \frac{1}{t}\mathrm{diag}(h_i)$, AdaGrad converges as

$$f_{\mathrm{best}}^k - f^\star \leq \frac{\sum_{i=1}^k \|g^{(i)}\|_{H_i^{-1}}^2}{2k} + \frac{R^2 \frac{1}{t}\|h_k\|_1}{2k}$$

- it can be shown that

$$\sum_{i=1}^k \|g^{(i)}\|_{H_i^{-1}}^2 \leq 2t\|h_k\|_1$$

  which gives

$$f_{\mathrm{best}}^k - f^\star \leq \frac{(2t + \frac{R^2}{t})|h_k\|_1}{2k}$$

- if $\|g\|_\infty \leq G$ it can be shown that

$$\|h_k\|_1 \leq n(\delta + G\sqrt{k})$$

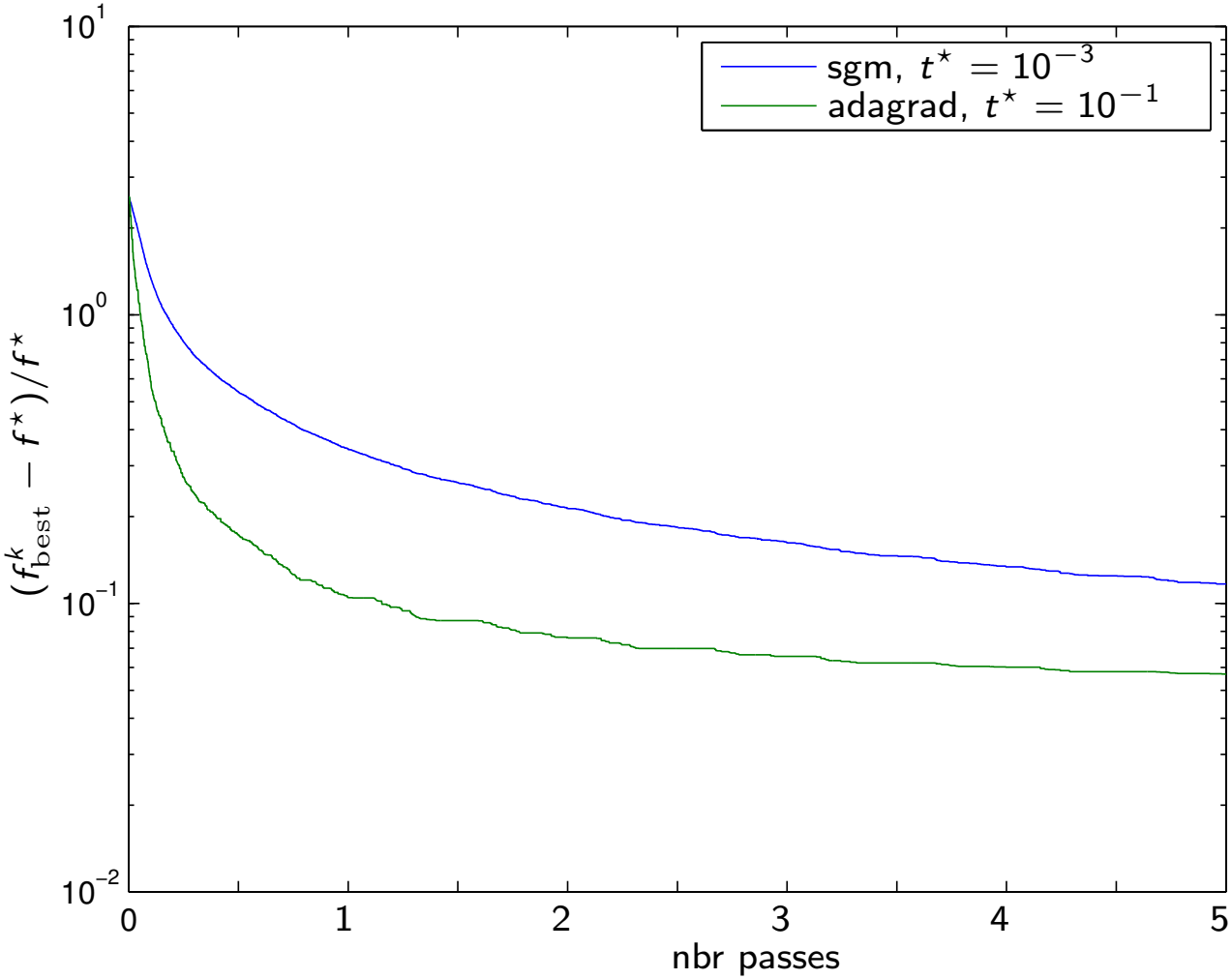
  which for $R < \infty$ implies $f_{\mathrm{best}}^k - f^\star \to 0$ for any $t > 0$

Example

Classification problem:

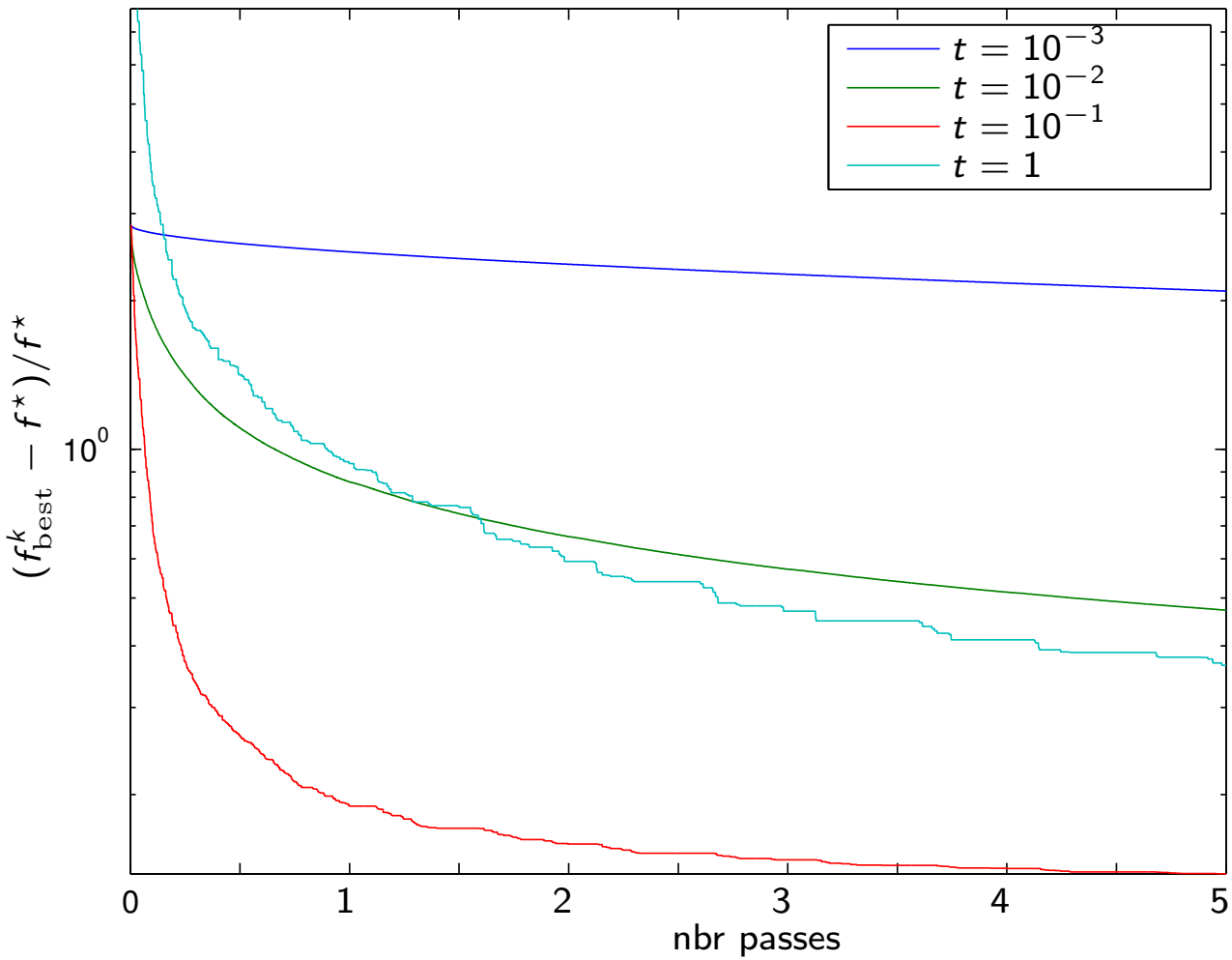
- **Data**: $\{a_i, b_i\}$, $i = 1, \ldots, 50000$
  - $a_i \in \mathbf{R}^{1000}$
  - $b \in \{-1, 1\}$
  - Data created with 5% mis-classifications w.r.t. $w = \mathbf{1}$, $v = 0$
- **Objective**: find classifiers $w \in \mathbf{R}^{1000}$ and $v \in \mathbf{R}$ such that
  - $a_i^T w + v > 1$ if $b = 1$
  - $a_i^T w + v < 1$ if $b = -1$
- **Optimization method**:
  - Minimize hinge-loss: $\sum_i \max(0, 1 - b_i(a_i^T w + v))$
  - Choose example uniformly at random, take sub-gradient step w.r.t. that example

Best subgradient method vs best AdaGrad



Often best AdaGrad performs better than best subgradient method

AdaGrad with different step-sizes t:



Sensitive to step-size selection (like standard subgradient method)
improvements: AdaDelta, RMSProp, ADAM, ...

Outline

Coordinate descent

solve (f_i not necessarily smooth)

$$\text{minimize} \quad f(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x)$$

Algorithm 4 Coordinate descent.

Input: $x \in \mathbf{R}^n$

1. **loop**
2. Randomly select a coordinate $j \in \{1, \ldots, n\}$
3. Compute g_j, (scaled) jth coordinate of subgradient, so

$$\mathbf{E} g_j e_j \in \partial f(x))$$

4. Update $x_j := (x_j - t g_j e_j)$

Stochastic coordinate descent

Algorithm 5 Stochastic coordinate descent.

Input: $x \in \mathbf{R}^n$

1 **loop**

2 Randomly select a function $i \in \{1, \ldots, m\}$

3 Randomly select a coordinate $j \in \{1, \ldots, n\}$

4 Read x from shared memory

5 Compute g_j, (scaled) jth coordinate of subgradient, so

$$\mathbf{E} g_j e_j \in \partial f(x))$$

6 Update $x_j := (x_j - t g_j e_j)$

to be more efficient on sparse data, choose coordinate j so $g_j \neq 0$

Hogwild: asynchronous stochastic coordinate descent

each update touches only one coordinate of x
so why not execute many different updates in parallel?

Algorithm 6 Hogwild: local view [Niu Recht Re Wright, 2011].

Input: $x \in \mathbf{R}^n$

All processors in parallel do
loop
    Randomly select a function $i \in \{1, \ldots, m\}$
    Randomly select a coordinate $j \in \{1, \ldots, n\}$
    Read x from shared memory
    Compute g_j, (scaled) jth coordinate of subgradient, so

$$\mathbf{E} g_j e_j \in \partial f(x))$$

    Update $x_j := (x_j - t g_j e_j)$

The delayed iterate

- each processor reads, computes, then writes to x
- in the meantime, x may have changed!

for the analysis (not needed by the algorithm):

- define the iteration counter k (updated whenever x is updated)
- consider the kth update, and the worker executing it
- count the number of updates $d_{k,j}$ made to x_j between reading and writing x_j
- define the (inconsistently) **delayed iterate** $x^{(k-d_k)} = (x_1^{(k-d_{k,1})}, \ldots, x_n^{(k-d_{k,n})})$
- define the maximum delay $\tau = \max_{k,j} d_{k,j}$
- convergence results depend on maximum delay

note: $x^{(k-d_k)}$ need never have existed in memory!

Hogwild: global view

Algorithm 7 Hogwild: global view [Niu Recht Re Wright, 2011].

Input: $x^{(0)} \in \mathbf{R}^n$

1 **for** $k = 1, \ldots$ **do**

2 Randomly select a function $i_k \in \{1, \ldots, m\}$

3 Randomly select a coordinate $j_k \in \{1, \ldots, n\}$

4 Read $x^{(k-d_k)} = (x_1^{(k-d_{k,1})}, \ldots, x_n^{(k-d_{k,n})})$ from shared memory

5 Compute j_kth coordinate of stochastic subgradient g so

$$\mathbf{E} g e_{j_k} \in \partial f_{i_k}(x^{(k-d_k)}))$$

6 Update $x^{(k+1)} = x^{(k)}$, $x_{j_k}^{(k+1)} := (x_j^{(k)} - t g e_j)$

Guarantee: if maximum delay τ bounded and f_i are all smooth, get $\frac{1}{k}$ convergence rate [Niu Recht Re Wright 2011]

Recipe for new SGD-type algorithm

$f_i : \mathbf{R}^n \to \mathbf{R}$ smooth, $h : \mathbf{R}^n \to \mathbf{R}$ proxable (possibly 0)

$$\text{minimize} \quad f(x) + h(x) = \frac{1}{m} \sum_{i=1}^{m} f_i(x) + h(x)$$

to form new algorithm, combine existing ingredients

- stochastic
- online
- variance-reduced
- regularized
- adaptive
- accelerated
- dual
- coordinate
- asynchronous
- distributed

with one new ingredient (see current-year NIPS)

Recipe for new SGD-type algorithm

$f_i : \mathbf{R}^n \to \mathbf{R}$ smooth, $h : \mathbf{R}^n \to \mathbf{R}$ proxable (possibly 0)

$$\text{minimize} \quad f(x) + h(x) = \frac{1}{m} \sum_{i=1}^{m} f_i(x) + h(x)$$

to form new algorithm, combine existing ingredients

- stochastic
- online
- variance-reduced
- regularized
- adaptive
- accelerated
- dual
- coordinate
- asynchronous
- distributed

with one new ingredient (see current-year NIPS)

expect convergence rate

- linear convergence if f is SSC

Recipe for new SGD-type algorithm

$f_i : \mathbf{R}^n \to \mathbf{R}$ smooth, $h : \mathbf{R}^n \to \mathbf{R}$ proxable (possibly 0)

$$\text{minimize} \quad f(x) + h(x) = \frac{1}{m} \sum_{i=1}^{m} f_i(x) + h(x)$$

to form new (publishable) algorithm, combine existing ingredients

- stochastic (like SGD)
- online (like SGD)
- variance-reduced (like SVRG)
- regularized (like SAGA)
- adaptive (like AdaGrad)
- accelerated (like Nesterov's method)
- dual (like dual proximal gradient)
- coordinate (like coordinate descent)
- asynchronous (like Hogwild)
- distributed (like Chambolle-Pock)

with one new ingredient (see current-year NIPS)

Recipe for convergence rate of new SGD-type algorithm

expect convergence rate

- linear convergence if f is SSC
- $\frac{1}{k}$ convergence if f is smooth
- $\frac{1}{\sqrt{k}}$ convergence if f is nonsmooth

References

- SVRG [Johnson Zhang 2014]
- SAGA [Defazio Bach Lacoste-Julien, 2014]
- AdaGrad [Duchi Singer Hazan, 2010]
- Hogwild [Niu Recht Re Wright, 2011]
- Overview of GD-based optimization methods, Sebastian Ruder. `http://sebastianruder.com/optimizing-gradient-descent/`