Stochastic Subgradient Methods

Yu-Xiang Wang
CS292A

(Based on Ryan Tibshirani's 10-725)

Last time: proximal map, Moreau envelope, interpretations of proximal algorithms

1. Properties of prox_f and M_f.
   - Moreau decomposition: $\mathrm{prox}_f + \mathrm{prox}_{f^*} = I$.
   - Gradient: $\nabla M_{t,f}(x) = \frac{x - \mathrm{prox}_{t,f}(x)}{t}$.
2. Two interpretations of the proximal point algorithm for minimizing f
   - Fixed point iterations: $x^{k+1} = (\mathrm{I} + t\partial f)^{-1} x^k$.
   - Gradient Descent on Moreau Envelope: $x^{k+1} = x^k - t\nabla M_f(x^k)$.
3. Two interpretation of the proximal gradient algorithm $f + g$.
   - Majorization-minimization.
   - Gradient Descent on Moreau Envelope of a locally linearized objective,
   - Fixed Point iterations

Outline

Today:

- Stochastic subgradient descent
- Convergence rates
- Mini-batches
- Early stopping

Stochastic gradient descent

Consider minimizing an average of functions

$$\min_x \frac{1}{m} \sum_{i=1}^m f_i(x)$$

As $\nabla \sum_{i=1}^m f_i(x) = \sum_{i=1}^m \nabla f_i(x)$, gradient descent would repeat:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \frac{1}{m} \sum_{i=1}^m \nabla f_i(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

In comparison, stochastic gradient descent or SGD (or incremental gradient method) repeats:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f_{i_k}(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

where $i_k \in \{1, \ldots m\}$ is some chosen index at iteration k.
(Robbins and Monro, 1951, Annals of Mathematical Statistics)

Two rules for choosing index i_k at iteration k:

- Randomized rule: choose $i_k \in \{1, \dots m\}$ uniformly at random
- Cyclic rule: choose $i_k = 1, 2, \dots m, 1, 2, \dots m, \dots$

Randomized rule is more common in practice. For randomized rule, note that

$$\mathbb{E}[\nabla f_{i_k}(x)] = \nabla f(x)$$

so we can view SGD as using an unbiased estimate of the gradient at each step

Main appeal of SGD:

- Iteration cost is independent of m (number of functions)
- Can also be a big savings in terms of memory useage

Example: stochastic logistic regression

Given $(x_i, y_i) \in \mathbb{R}^p \times \{0, 1\}$, $i = 1, \ldots n$, recall logistic regression:

$$\min_\beta \; f(\beta) = \frac{1}{n} \sum_{i=1}^n \underbrace{\Big(- y_i x_i^T \beta + \log(1 + \exp(x_i^T \beta)) \Big)}_{f_i(\beta)}$$

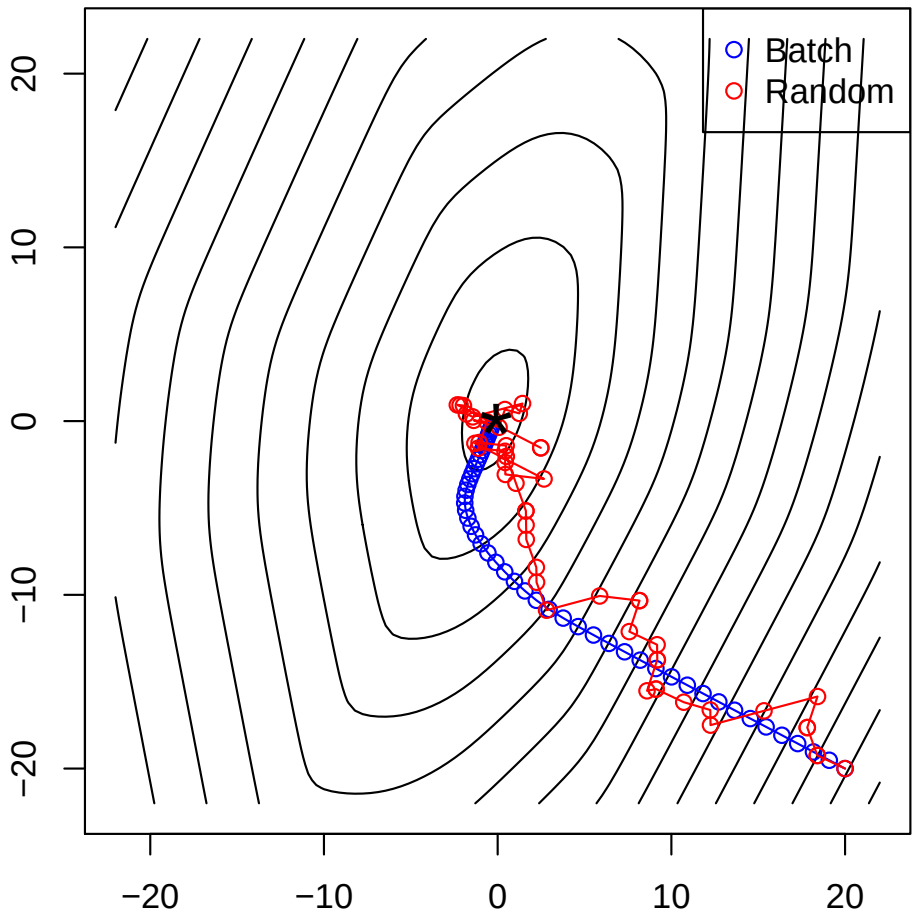
Gradient computation $\nabla f(\beta) = \frac{1}{n} \sum_{i=1}^n \big(y_i - p_i(\beta)\big) x_i$ is doable when n is moderate, but not when n is huge

Full gradient (also called batch) versus stochastic gradient:

- One batch update costs $O(np)$
- One stochastic update costs $O(p)$

Clearly, e.g., 10K stochastic steps are much more affordable

Small example with $n = 10$, $p = 2$ to show the "classic picture" for batch versus stochastic methods:



Blue: batch steps, $O(np)$
Red: stochastic steps, $O(p)$

Rule of thumb for stochastic methods:

- generally thrive far from optimum
- generally struggle close to optimum

Step sizes

Standard in SGD is to use diminishing step sizes, e.g., $t_k = 1/k$, for $k = 1, 2, 3, \ldots$

Why not fixed step sizes? Here's some intuition. Suppose we take cyclic rule for simplicity. Set $t_k = t$ for m updates in a row, we get:

$$x^{(k+m)} = x^{(k)} - t \sum_{i=1}^{m} \nabla f_i(x^{(k+i-1)})$$

Meanwhile, full gradient with step size t would give:

$$x^{(k+1)} = x^{(k)} - t \sum_{i=1}^{m} \nabla f_i(x^{(k)})$$

The difference here: $t \sum_{i=1}^{m} [\nabla f_i(x^{(k+i-1)}) - \nabla f_i(x^{(k)})]$, and if we hold t constant, this difference will not generally be going to zero

Convergence rates

Recall: for convex f, gradient descent with diminishing step sizes satisfies

$$f(x^{(k)}) - f^\star = O(1/\sqrt{k})$$

When f is differentiable with Lipschitz gradient, we get for suitable fixed step sizes

$$f(x^{(k)}) - f^\star = O(1/k)$$

What about SGD? For convex f, SGD with diminishing step sizes satisfies[1]

$$\mathbb{E}[f(x^{(k)})] - f^\star = O(\log(k)/\sqrt{k})$$

Unfortunately this almost does not improve[2] when we further assume f has Lipschitz gradient.

---

[1] E.g., Shamir and Zhang, ICML'2012.

[2] We may improve a $\log k$ factor when assuming smoothness. But there are algorithms that is not the last iterate of SGD that does not need the $\log k$ factor in the first place even without smoothness.

Even worse is the following discrepancy!

When f is strongly convex and has a Lipschitz gradient, gradient descent satisfies

$$f(x^{(k)}) - f^\star = O(c^k)$$

where $c < 1$. But under same conditions, SGD gives us[3]

$$\mathbb{E}[f(x^{(k)})] - f^\star = O(\log k/k)$$

So stochastic methods do not enjoy the linear convergence rate of gradient descent under strong convexity.

What can we do to improve SGD?

[3] E.g., Shamir and Zhang, ICML'2012.

Mini-batches

Also common is mini-batch stochastic gradient descent, where we choose a random subset $I_k \subseteq \{1, \ldots m\}$, of size $|I_k| = b \ll m$, and repeat:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \frac{1}{b} \sum_{i \in I_k} \nabla f_i(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

Again, we are approximating full graident by an unbiased estimate:

$$\mathbb{E}\left[\frac{1}{b} \sum_{i \in I_k} \nabla f_i(x)\right] = \nabla f(x)$$

Using mini-batches reduces the variance of our gradient estimate by a factor $1/b$, but is also b times more expensive

Back to logistic regression, let's now consider a regularized version:

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \Big(- y_i x_i^T \beta + \log(1 + e^{x_i^T \beta}) \Big) + \frac{\lambda}{2} \|\beta\|_2^2$$

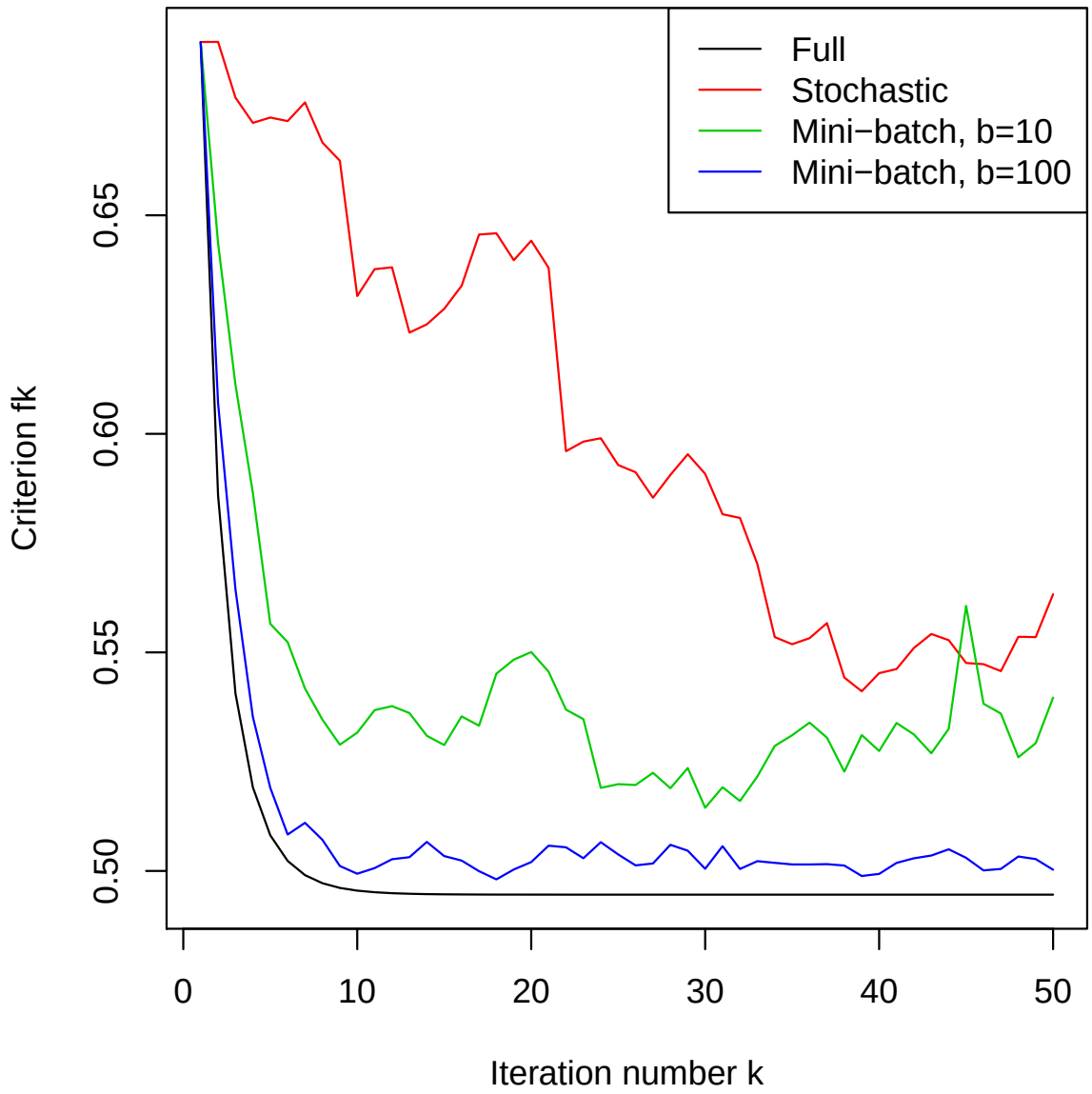
Write the criterion as

$$f(\beta) = \frac{1}{n} \sum_{i=1}^{n} f_i(\beta), \quad f_i(\beta) = -y_i x_i^T \beta + \log(1 + e^{x_i^T \beta}) + \frac{\lambda}{2} \|\beta\|_2^2$$

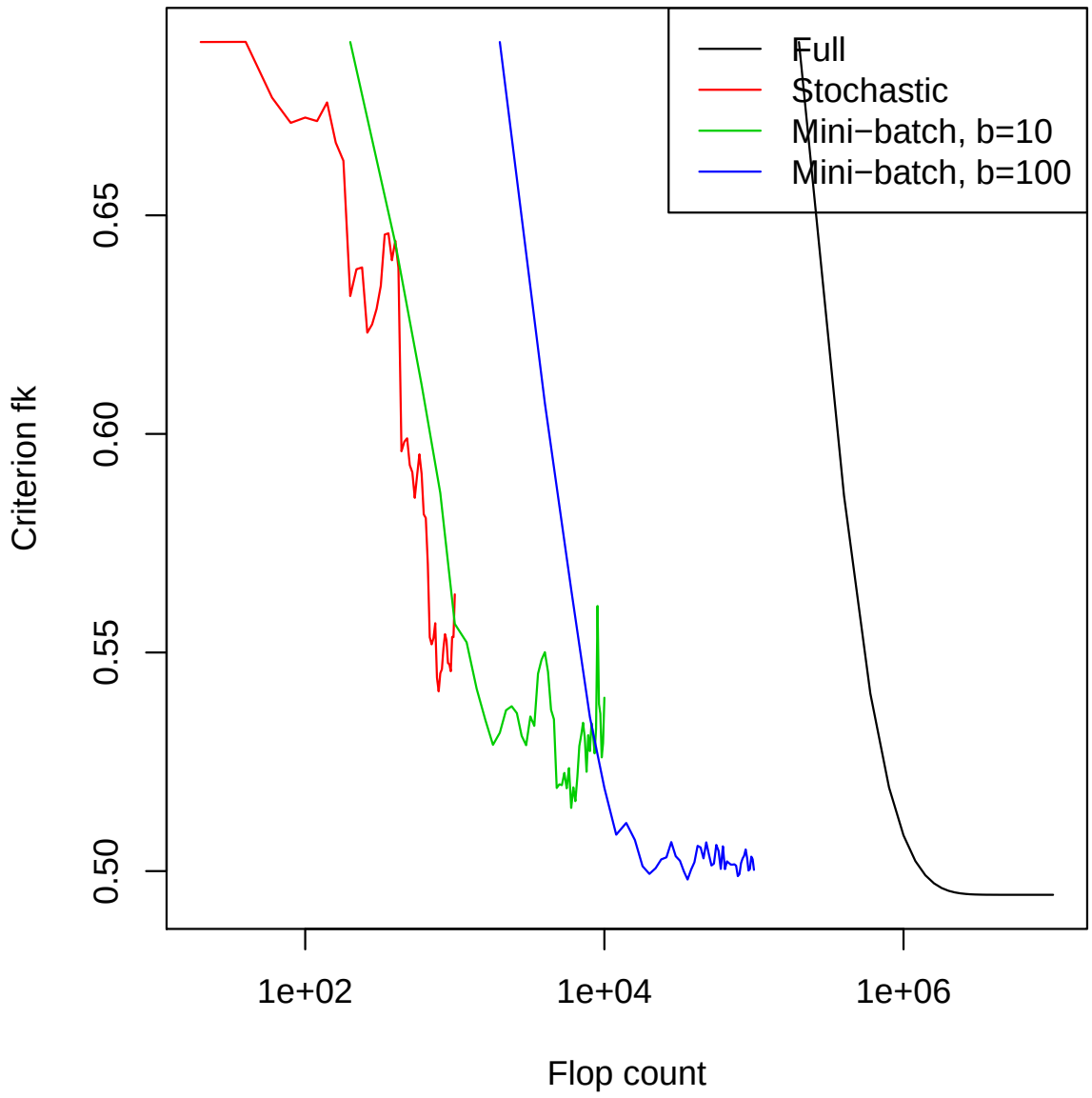
Full gradient computation is $\nabla f(\beta) = \frac{1}{n} \sum_{i=1}^{n} \big(y_i - p_i(\beta)\big) x_i + \lambda \beta$.
Comparison between methods:

- One batch update costs $O(np)$
- One mini-batch update costs $O(bp)$
- One stochastic update costs $O(p)$

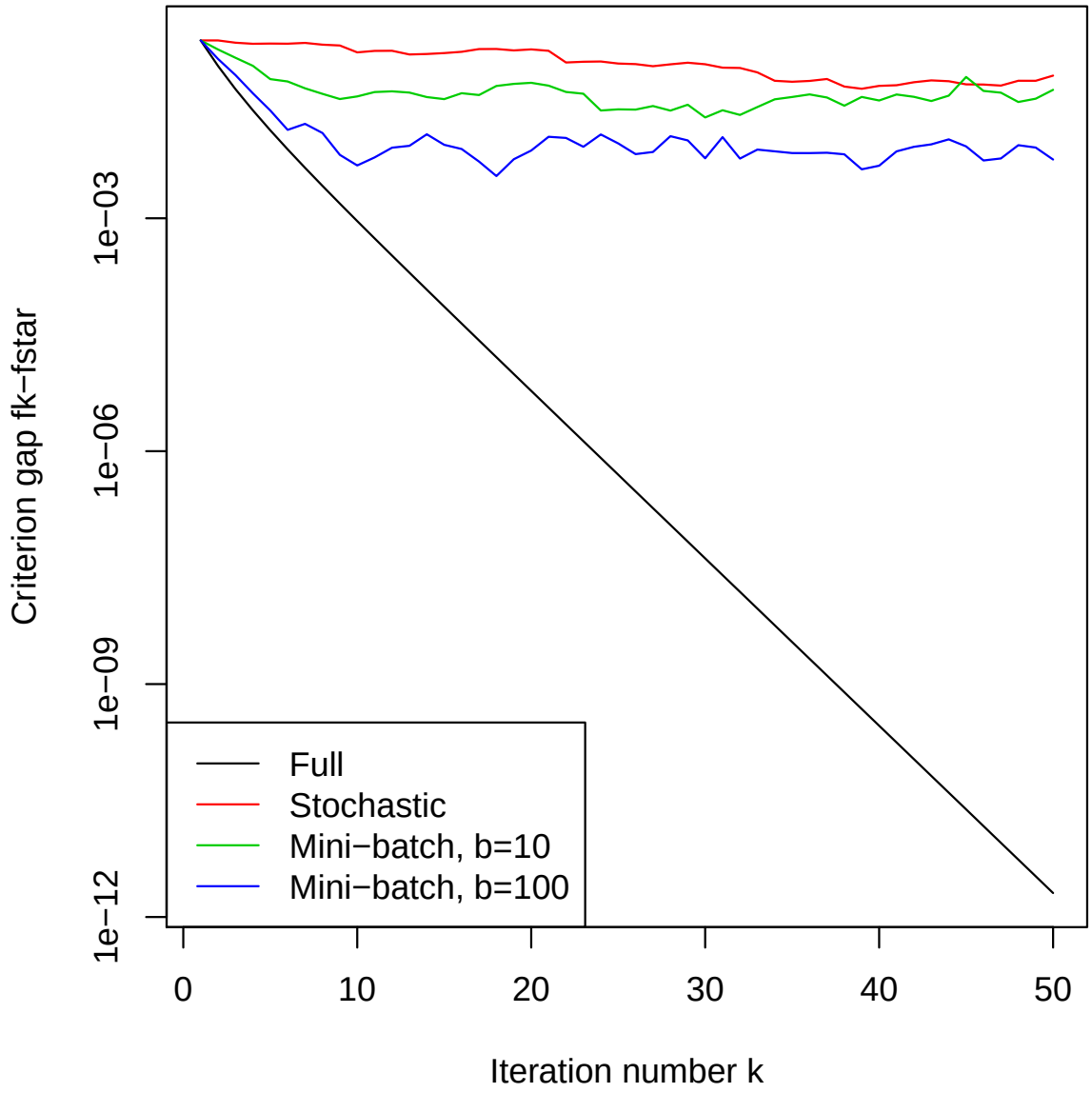
Example with $n = 10,000$, $p = 20$, all methods use fixed step sizes:



What's happening? Now let's parametrize by flops:



Finally, looking at suboptimality gap (on log scale):



Convergence rate proofs

Algorithm: $x^{k+1} = x^k - t_k g^k$.
Assumptions: (1) Unbiased subgradient: $\mathbb{E}[g^k | x^k] \in \partial f(x^k)$.
(2) Bounded variance: $\mathbb{E}[\|g^k - \mathbb{E}[g^k|x^k]\|^2 | x^k] \leq \sigma^2$

- Convex and G-Lipschitz (Proof this!)

$$\min_{i=1,\ldots,k} \mathbb{E}\left[f(x^i)\right] - f^* \leq \frac{\|x^1 - x^*\|^2 + (G^2 + \sigma^2)\sum_{i=1}^k t_i^2}{2\sum_{i=1}^k t_i}$$

- Nonconvex but L-smooth with $t_i = 1/(\sqrt{k}L)$ (Proof this!)

$$\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^k \|\nabla f(x^i)\|^2\right] \leq \frac{2(f(x^1) - f^*)L + \sigma^2}{\sqrt{k}}$$

- m-Strongly convex and G-Lipschitz: $O(G^2 \log(T)/mT)$ rate.
  we will prove this when we talk about online learning!

Averaging Stochastic (Sub)gradient Descent

One drawbacks of SGD: the guarantees are for $\min_{i=1,\ldots,k} \mathbb{E}\left[f(x^i)\right]$.
Idea: Let's output the online averages of the iterates.

$$\bar{x}^k = \frac{k-1}{k}\bar{x}^{k-1} + \frac{1}{k}x^k \text{(Polyak-Rupert Averaging)}^4$$

Convergence bound:

$$\mathbb{E}[f(\bar{x}^k)] - f^* \leq \begin{cases} O(1/\sqrt{k}) & \text{if convex (We just proved that!)}^5 \\ O(\log k/k) & \text{if strongly convex} \end{cases}$$

Can the $\log k$ be removed? No. Use α-Suffix averaging. (Rakhlin, Shamir, Sridharan, ICML'2012)
But that doesn't have an online implementation. Solution by (Shamir and Zhang, ICML'2013) : $\bar{x}_\eta^k = (1 - \frac{1+\eta}{k+\eta})\bar{x}_\eta^{k-1} + \frac{1+\eta}{k+\eta}x^k$.

[4] See, Polyak,1990; Rupert,1988; Polyak and Juditsky, 1992.
[5] Using ideas from Nemirovski et al. (2009).

ASGD Comparison in Practice

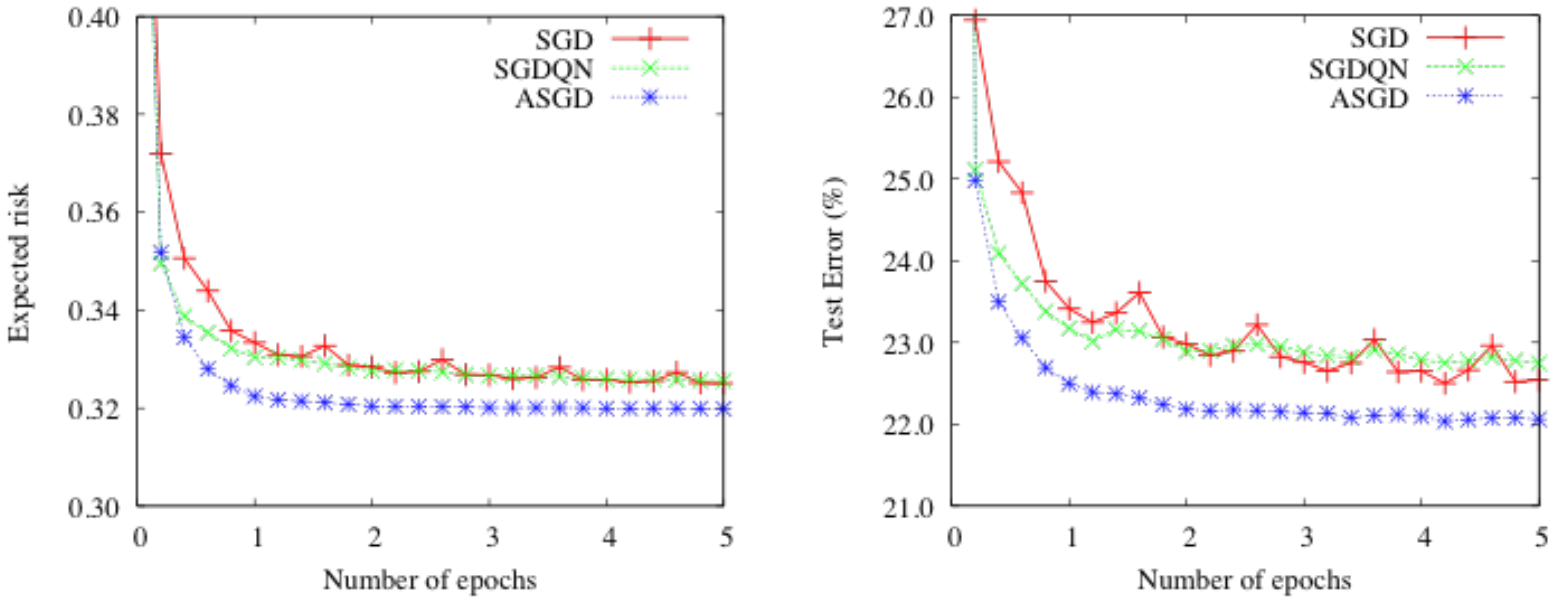


Fig. 2. Comparaison of the test set performance of SGD, SGDQN, and ASGD for a linear squared hinge SVM trained on the ALPHA task of the 2008 Pascal Large Scale Learning Challenge. ASGD nearly reaches the optimal expected risk after a single pass.

(Figure from Leon Bottou, 2010)

Stochastic Programming

Stochastic programming:

$$\underset{x}{\text{minimize}}\, \mathbb{E} f(x)$$

where the expectation is taken over f f is a random function of x. More generally,

$$\underset{x}{\text{minimize}}\, \mathbb{E} f(x), \text{ Subject to } \mathbb{E} g(x) \leq 0.$$

Chance constrained stochastic progamming:

$$\underset{x}{\text{minimize}}\, \mathbb{E} f(x) \text{ Subject to }, \mathbb{P}(g_i(x) \leq 0) \geq \eta.$$

Stochastic Programming: Examples

1. Machine Learning / Stochastic Convex Optimization

$$\underset{h \in \mathcal{H}}{\text{minimize}}\, \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(h,(x,y))]$$

where ℓ is a loss function, $\mathcal{H}$ is a hypothesis class (a class of classifiers), x, y are feature and label pairs.

2. Portfolio Optimization with Value At Risk (VaR) constraint.

$$\max_{x: x\geq 0, \sum_i x_i = \text{ Budget}} \quad \mathbb{E}\left[\sum_i R_i x_i\right],$$

$$\text{Subject to} \qquad \mathbb{P}\left(\sum_i R_i x_i \leq -\$1 \text{ Million}\right) \leq 0.05$$

where R_i is the return of Stock i, x_i are the allocated budget in the portfolio.

Optimality Guarantees of SGD

Under the gradient oracle: $\mathbb{E}[g_i] = \nabla \mathbb{E} f(x)$, let us minimize the following stochastic objective over $\hat{\theta} \in \mathbb{R}$

$$\mathbb{E}_{X \sim \mathcal{N}(\mu, \sigma^2)}[(\theta - X)^2] = (\theta - \mu)^2 + \sigma^2.$$

This function is 1-strongly convex in $\hat{\theta}$. The observation is $X_1, ..., X_n \sim \mathcal{N}(\mu, \sigma^2)$, equivalent to observing a stochastic gradient $\theta - X_i$, which approximates the gradient $\theta - \mu$.

Theorem: Any algorithm $\hat{\mu}$ that takes random variables $X_1, ..., X_n$ as an input obeys that:

$$\max_{\theta \in \mathbb{R}} \mathbb{E}[(\hat{\mu} - \mu)^2] \geq \frac{\sigma^2}{n}$$

If we can solve the stochastic progamming problems, then we can solve the estimation problem beyond its information-theoretic limit.

Optimality Guarantees of SGD

Similarly, consider

$$\min_{\theta \in [-1,1]} \mathbb{E}[X\theta] = p\theta - (1-p)\theta = (-1+2p)\theta$$

where $\mathbb{P}(X=1) = p$ and $\mathbb{P}(X=-1) = 1-p$.
This is a convex and 1-Lipschitz objective. Observing samples $X_1, ..., X_n$ from that distribution can be considered stochastic gradient.s
Statistical lower bound $1/\sqrt{n}$ on estimating p suggest that we cannot distinguish between the world when $p = 0.5 - 1/\sqrt{n}$ and the world when $p = 0.5 + 1/\sqrt{n}$, which implies a lower bound of $1/\sqrt{k}$ for the convergence rate of SGD for non-strongly convex stochastic objective.

End of the story?

Short story:

- SGD can be super effective in terms of iteration cost, memory
- But SGD is slow to converge, can't adapt to strong convexity
- And mini-batches seem to be a wash in terms of flops (though they can still be useful in practice)
- Averaging trick helps to remove $\log k$ terms in cases without smoothness.
- Lower bound from stochastic programming that says $1/\sqrt{k}$ is optimal in general and $1/k$ for strongly convex.

Is this the end of the story for SGD?

For a while, the answer was believed to be yes. But this was for a more general stochastic optimization problem, where $f(x) = \int F(x, \xi)\, dP(\xi)$.

New wave of "variance reduction" work shows we can modify SGD to converge much faster for finite sums (more later?)

SGD in large-scale ML

SGD has really taken off in large-scale machine learning

- In many ML problems we don't care about optimizing to high accuracy, it doesn't pay off in terms of statistical performance
- Thus (in contrast to what classic theory says) fixed step sizes are commonly used in ML applications
- One trick is to experiment with step sizes using small fraction of training before running SGD on full data set ... many other heuristics are common[6]
- Many variants provide better practical stability, convergence: momentum, acceleration, averaging, coordinate-adapted step sizes, variance reduction ...
- See AdaGrad, Adam, AdaMax, SVRG, SAG, SAGA ... (more later?)

[6] E.g., Bottou (2012), "Stochastic gradient descent tricks"

Early stopping

Suppose p is large and we wanted to fit (say) a logistic regression model to data $(x_i, y_i) \in \mathbb{R}^p \times \{0, 1\}$, $i = 1, \ldots n$

We could solve (say) ℓ_2 regularized logistic regression:

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n \Big(- y_i x_i^T \beta + \log(1 + e^{x_i^T \beta}) \Big) \quad \text{subject to} \quad \|\beta\|_2 \leq t$$

We could also run gradient descent on the unregularized problem:

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n \Big(- y_i x_i^T \beta + \log(1 + e^{x_i^T \beta}) \Big)$$

and stop early, i.e., terminate gradient descent well-short of the global minimum

Consider the following, for a very small constant step size ϵ:

- Start at $\beta^{(0)} = 0$, solution to regularized problem at $t = 0$
- Perform gradient descent on unregularized criterion

$$\beta^{(k)} = \beta^{(k-1)} - \epsilon \cdot \frac{1}{n} \sum_{i=1}^{n} (y_i - p_i(\beta^{(k-1)}))x_i, \quad k = 1, 2, 3, \ldots$$

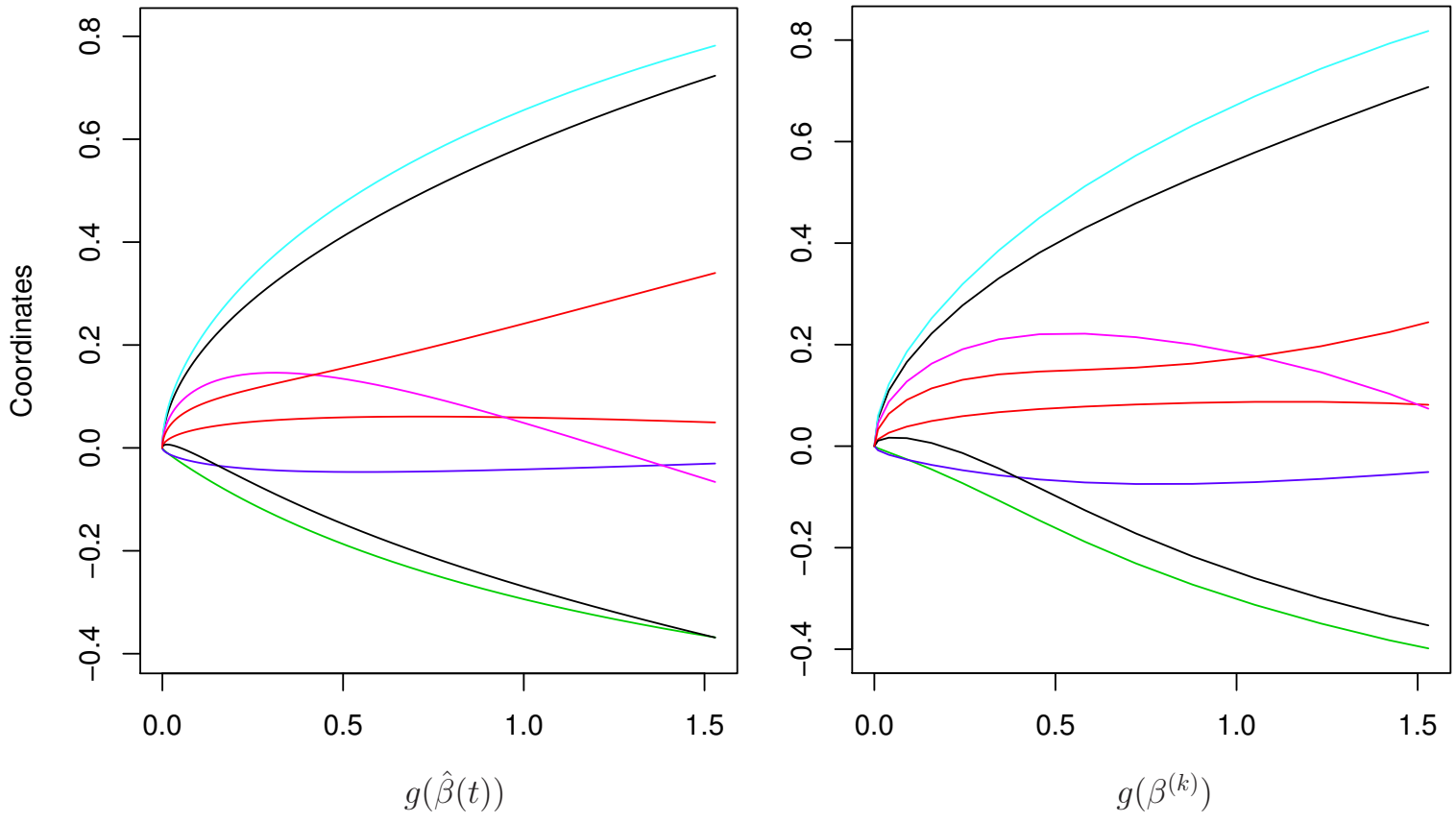
  (we could equally well consider SGD)
- Treat $\beta^{(k)}$ as an approximate solution to regularized problem with $t = \|\beta^{(k)}\|_2$

This is called early stopping for gradient descent. Why would we ever do this? It's both more convenient and potentially much more efficient than using explicit regularization

An intruiging connection

When we solve the ℓ_2 regularized logistic problem for varying t ... solution path looks quite similar to gradient descent path!

Example with $p = 8$, solution and grad descent paths side by side:



Lots left to explore

- Connection holds beyond logistic regression, for arbitrary loss
- In general, the grad descent path will not coincide with the ℓ_2 regularized path (as $\epsilon \to 0$). Though in practice, it seems to give competitive statistical performance
- Can extend early stopping idea to mimick a generic regularizer (beyond ℓ_2)[7]
- There is a lot of literature on early stopping, but it's still not as well-understood as it should be
- Early stopping is just one instance of implicit or algorithmic regularization ... many others are effective in large-scale ML, they all should be better understood

[7] Tibshirani (2015), "A general framework for fast stagewise algorithms"

References and further reading

- D. Bertsekas (2010), "Incremental gradient, subgradient, and proximal methods for convex optimization: a survey"
- A. Nemirovski and A. Juditsky and G. Lan and A. Shapiro (2009), "Robust stochastic optimization approach to stochastic programming"
- O. Shamir, T. Zhang. (2013). "Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes". In International Conference on Machine Learning.
- R. Tibshirani (2015), "A general framework for fast stagewise algorithms"