Convex Optimization Theory and Applications

Topic 20 - Stochastic Gradient Descent

Li Li

Department of Automation
Tsinghua University

Fall, 2009-2021.

20.0. Outline

20.1. 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 descent) 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

20.1. Stochastic Gradient Descent

Two rules for choosing index i_k at iteration k:

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

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

20.1. Stochastic Gradient Descent

Example: stochastic logistic regression

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

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

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

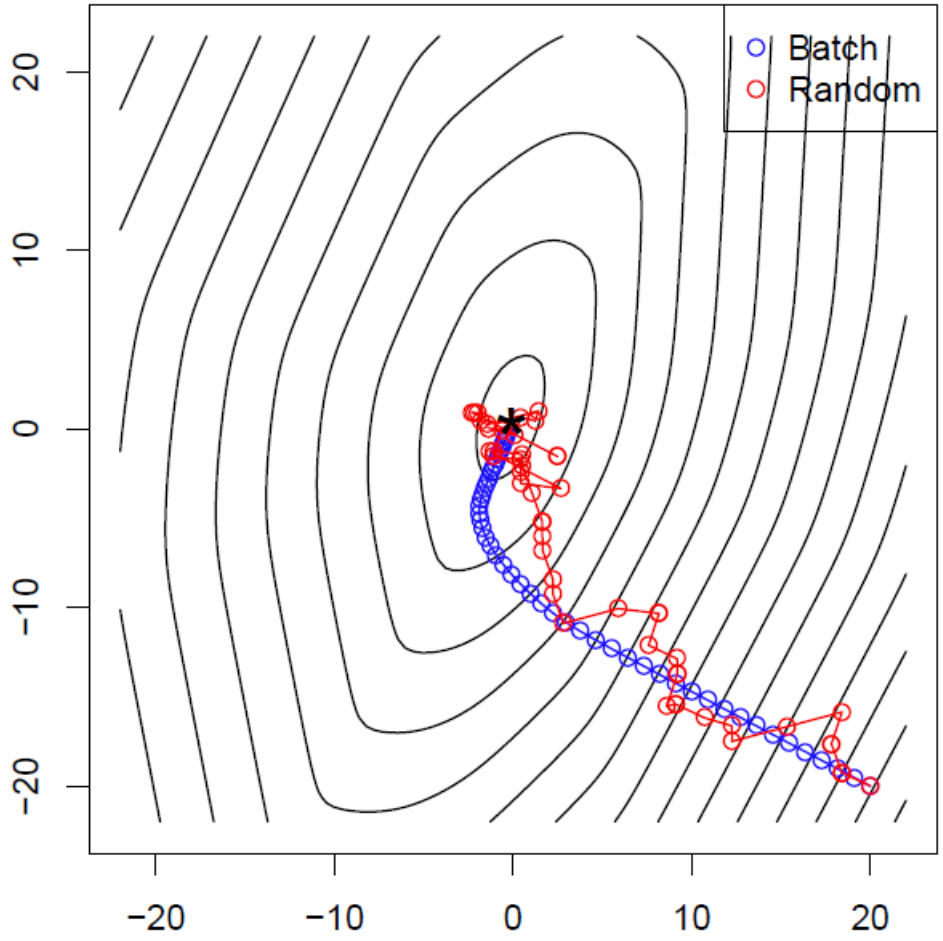
Full gradient (also called batch) versus stochastic gradient:

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

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

20.1. Stochastic Gradient Descent

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

20.1. Stochastic Gradient Descent

Standard in SGD is to use diminishing step sizes, e.g., $t_k = 1/k$

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 mt 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

20.1. Stochastic Gradient Descent

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]

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

Unfortunately this does not improve when we further assume f has Lipschitz gradient

[1] For example, Nemirosvki et al. (2009), "Robust stochastic optimization approach to stochastic programming"

20.1. Stochastic Gradient Descent

Even worse is the following discrepancy!

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

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

where $0 < \gamma < 1$. But under same conditions, SGD gives us[2]

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

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

What can we do to improve SGD?

---

[2]For example, Nemirosvki et al. (2009), "Robust stochastic optimization approach to stochastic programming"

20.1. Stochastic Gradient Descent

Also common is mini-batch stochastic gradient descent, where we choose a random subset $I_k \subseteq \{1, \ldots, m\}$, $|I_k| = b \ll m$, 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 gradient 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 variance by a factor $1/b$, but is also b times more expensive. Theory is not convincing: under Lipschitz gradient, rate goes from $O(1/\sqrt{k})$ to $O(1/\sqrt{bk} + 1/k)$[3]

[3]For example, Dekel et al. (2012), "Optimal distributed online prediction using mini-batches"

20.1. Stochastic Gradient Descent

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

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

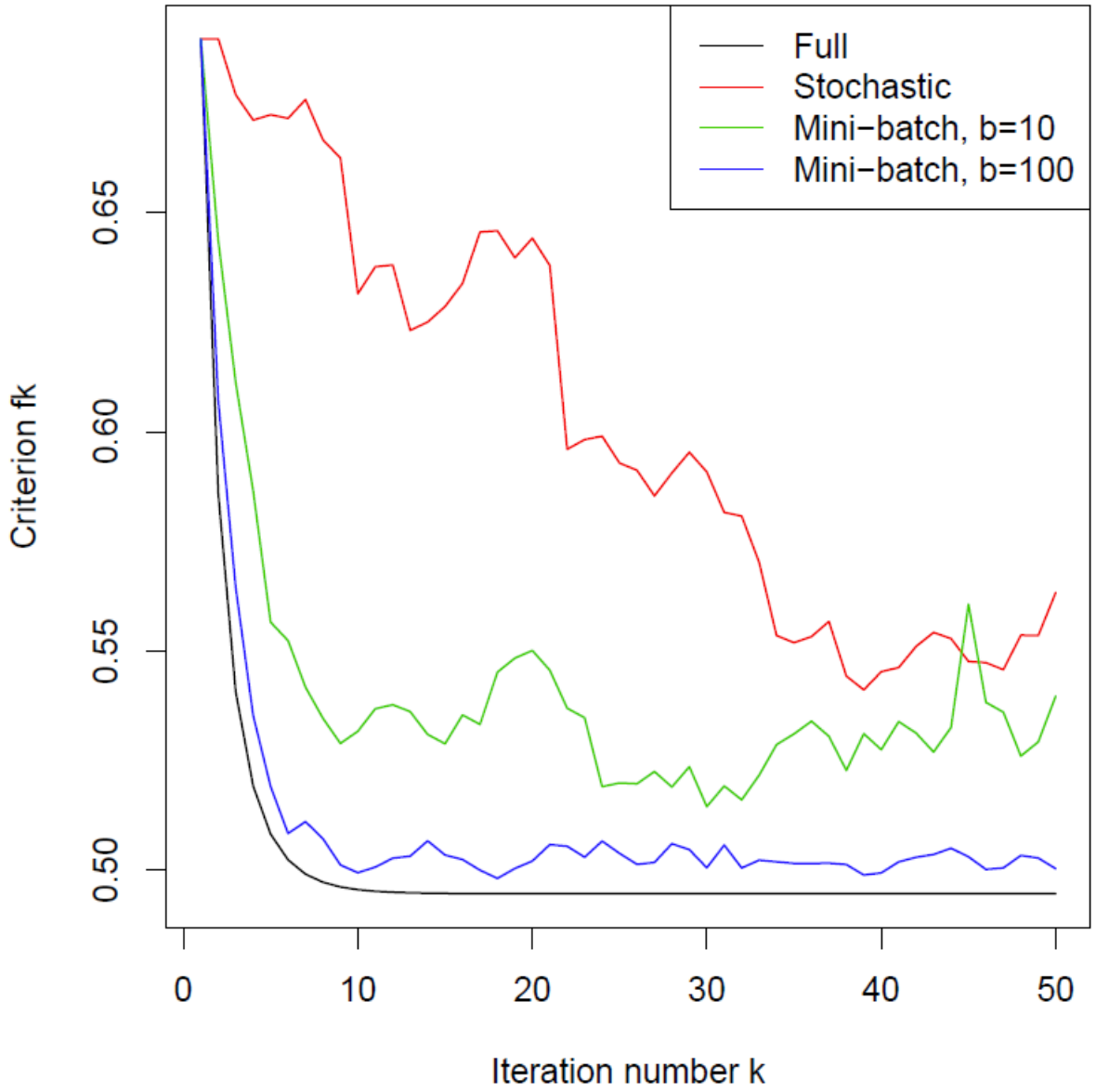
Write the criterion as

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

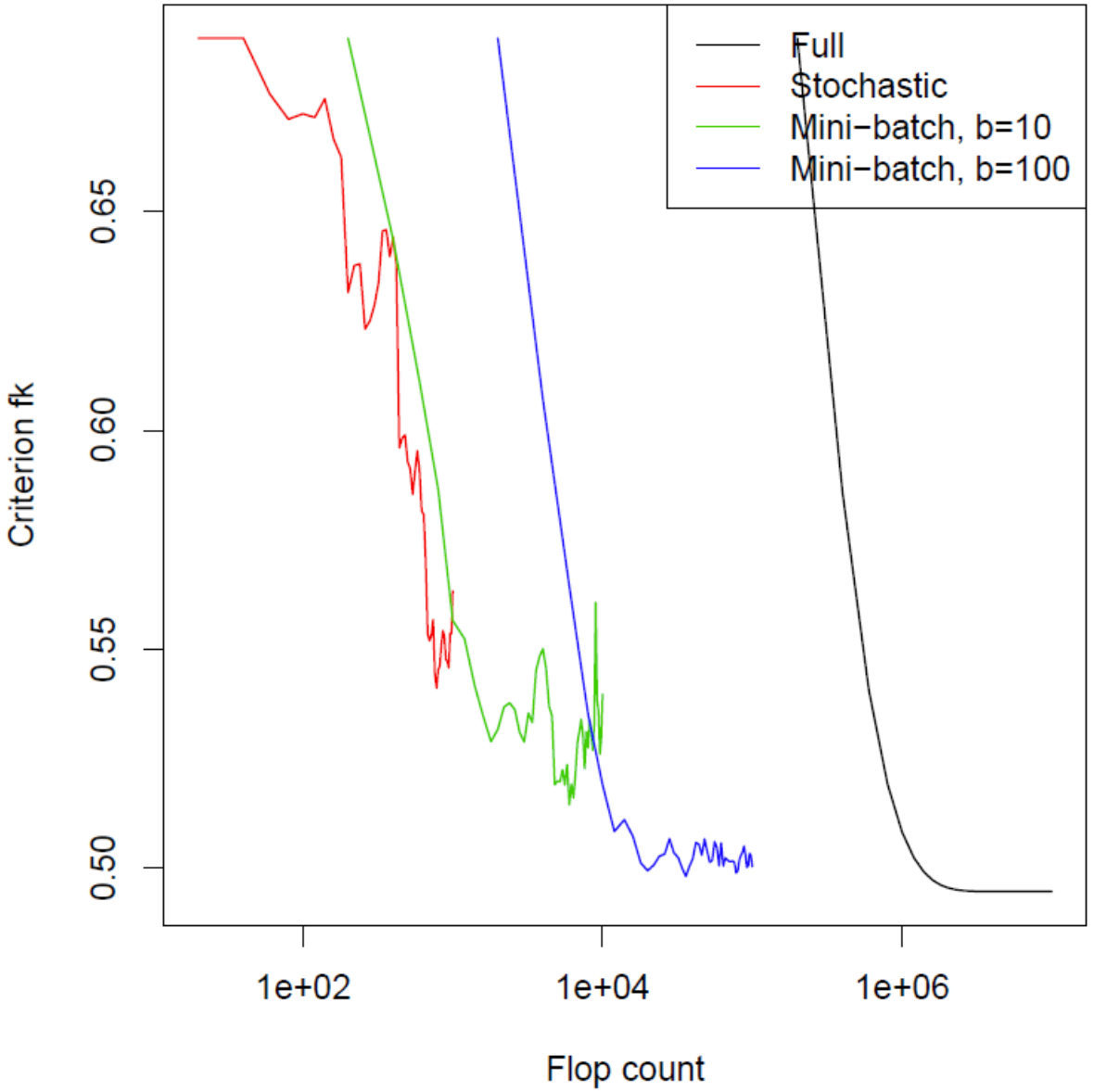
Full gradient computation is $\nabla f(\beta) = \frac{1}{n} \sum_{i=1}^{n} \big(y_i - p_i(\beta)\big) x_i + \lambda \beta$.
Comparison between methods:

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

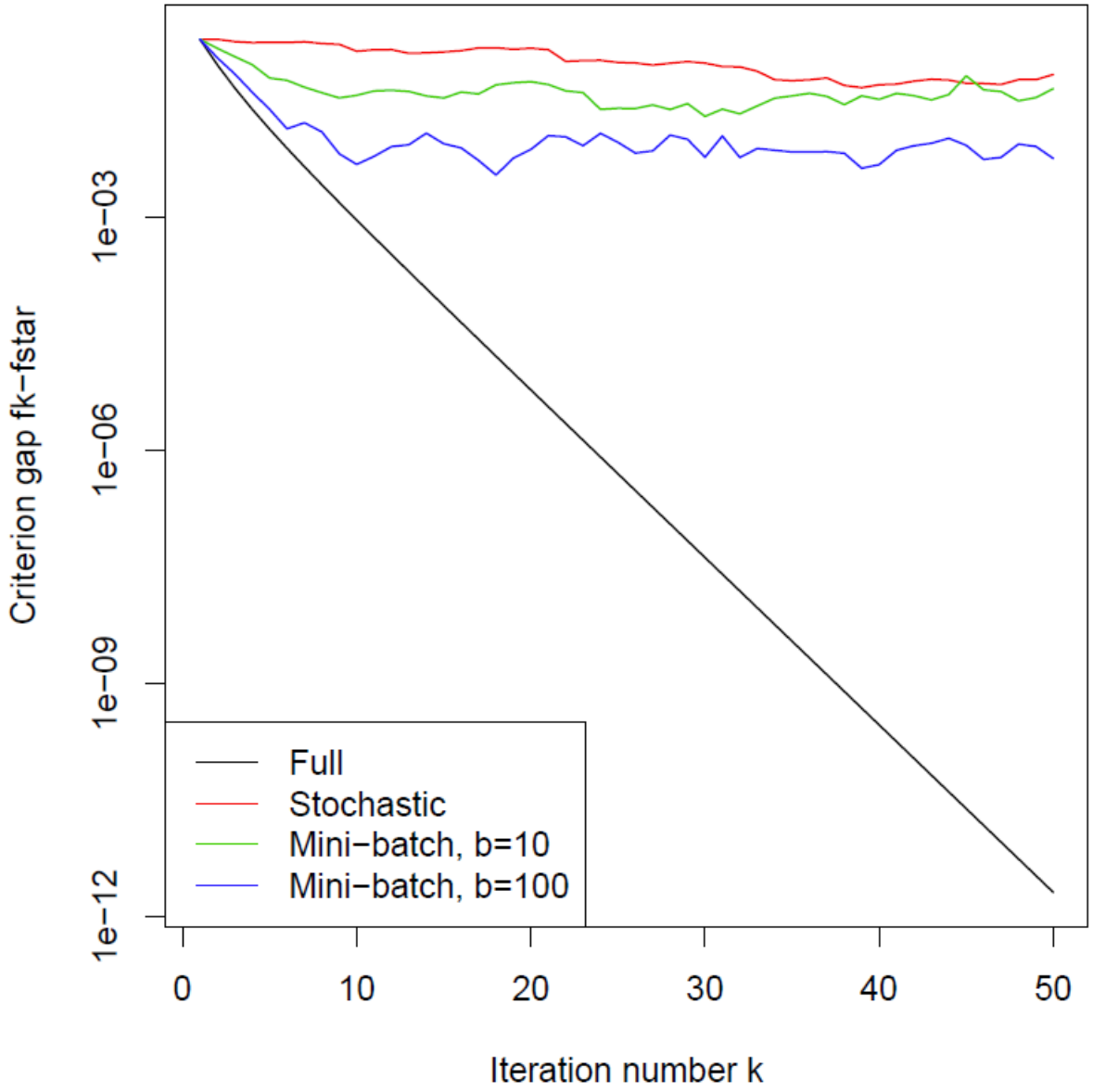
Example with $n = 10,000$, $p = 20$, all methods use fixed step sizes:



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



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



20.1. Stochastic Gradient Descent

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)

Is this the end of the story for SGD?

For a while, the answer was believed to be yes. Slow convergence for strongly convex functions was believed inevitable, as Nemirovski and others established matching lower bounds ... but this was for a more general stochastic 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?)

20.1. Stochastic Gradient Descent

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[4]
- Momentum/acceleration, averaging, adaptive step sizes are all popular variants in practice
- SGD is especially popular in large-scale, continous, nonconvex optimization, but it is still not particular well-understood there (a big open issue is that of implicit regularization)

[4] For example, Bottou (2012), "Stochastic gradient descent tricks"

20.2. Modified SGD

- Typically we make a (uniform) random choice $i_k \in \{1, \ldots n\}$
- Also common: mini-batch stochastic gradient descent, where we choose a random subset $I_k \subset \{1, \ldots n\}$, of size $b \ll n$, and update according to

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

- In both cases, we are approximating the full graident by a noisy estimate, and our noisy estimate is unbiased

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

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

The mini-batch reduces the variance by a factor $1/b$, but is also b times more expensive!

20.2. Modified SGD

For solving $\min_x f(x)$, stochastic gradient is actually a class of algorithms that use the iterates:

$$x^{(k)} = x^{(k-1)} - \eta_k \, g(x^{(k-1)}; \xi_k),$$

where $g(x^{(k-1)}, \xi_k)$ is a *stochastic gradient* of the objective $f(x)$ evaluated at $x^{(k-1)}$.

Bias: The bias of the stochastic gradient is defined as:

$$\text{bias}(g(x^{(k-1)}; \xi_k)) := \mathbb{E}_{\xi_k}(g(x^{(k-1)}; \xi_k)) - \nabla f(x^{(k-1)}).$$

Unbiased: When $\mathbb{E}_{\xi_k}(g(x^{(k-1)}; \xi_k)) = \nabla f(x^{(k-1)})$, the stochastic gradient is said to be unbiased. (e.g. the stochastic gradient scheme discussed so far)
Biased: We might also be interested in biased estimators, but where the bias is small, so that $\mathbb{E}_{\xi_k}(g(x^{(k-1)}; \xi_k)) \approx \nabla f(x^{(k-1)})$.

20.2. Modified SGD

Variance. In addition to small (or zero) bias, we also want the variance of the estimator to be small:

$$\begin{aligned}\text{variance}(g(x^{(k-1)}, \xi_k)) &:= \mathbb{E}_{\xi_k}(g(x^{(k-1)}, \xi_k) - \mathbb{E}_{\xi_k}(g(x^{(k-1)}, \xi_k)))^2 \\ &\leq \mathbb{E}_{\xi_k}(g(x^{(k-1)}, \xi_k))^2.\end{aligned}$$

The caveat with the stochastic gradient scheme we have seen so far is that its variance is large, and in particular doesn't decay to zero with the iteration index.

Loosely: because of above, we have to decay the step size η_k to zero, which in turn means we can't take "large" steps, and hence the convergence rate is slow.

Can we get the variance to be small, and decay to zero with iteration index?

20.2. Modified SGD

Consider an estimator X for a parameter θ.
Note that for an unbiased estimator, $\mathbb{E}(X) = \theta$.

Now consider the following modified estimator: $Z := X - Y$, such that $\mathbb{E}(Y) \approx 0$. Then the bias of Z is also close to zero, since

$$\mathbb{E}(Z) = \mathbb{E}(X) - \mathbb{E}(Y) \approx \theta.$$

If $\mathbb{E}(Y) = 0$, then Z is unbiased iff X is unbiased.

What about the **variance** of estimator X?
$\mathrm{Var}(X - Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) - 2\,\mathrm{Cov}(X, Y)$. This can be seen to much less than $\mathrm{Var}(X)$ if Y is highly correlated with X.
Thus, given any estimator X, we can reduce its variance, if we can construct a Y that (a) has expectation (close to) zero, and (b) is highly correlated with X. This is the abstract template followed by SAG, SAGA, SVRG, SDCA, ...

20.2. Modified SGD

Stochastic average gradient or SAG (Schmidt, Le Roux, Bach 2013) is a breakthrough method in stochastic optimization. Idea is fairly simple:

- Maintain table, containing gradient g_i of f_i, $i = 1, \ldots n$
- Initialize $x^{(0)}$, and $g_i^{(0)} = x^{(0)}$, $i = 1, \ldots n$
- At steps $k = 1, 2, 3, \ldots$, pick a random $i_k \in \{1, \ldots n\}$ and then let

$$g_{i_k}^{(k)} = \nabla f_i(x^{(k-1)}) \quad \text{(most recent gradient of } f_i\text{)}$$

  Set all other $g_i^{(k)} = g_i^{(k-1)}$, $i \neq i_k$, i.e., these stay the same
- Update

$$x^{(k)} = x^{(k-1)} - t_k \cdot \frac{1}{n} \sum_{i=1}^{n} g_i^{(k)}$$

20.2. Modified SGD

- Key of SAG is to allow each f_i, $i = 1, \ldots n$ to communicate a part of the gradient estimate at each step
- This basic idea can be traced back to incremental aggregated gradient (Blatt, Hero, Gauchman, 2006)
- SAG gradient estimates are no longer unbiased, but they have greatly reduced variance
- Isn't it expensive to average all these gradients? (Especially if n is huge?) This is basically just as efficient as stochastic gradient descent, as long we're clever:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \underbrace{\left(\frac{g_{i_k}^{(k)}}{n} - \frac{g_{i_k}^{(k-1)}}{n} + \underbrace{\frac{1}{n} \sum_{i=1}^{n} g_i^{(k-1)}}_{\text{old table average}} \right)}_{\text{new table average}}$$

20.2. Modified SGD

Stochastic gradient in SAG:

$$\underbrace{g_{i_k}^{(k)}}_{X} - \underbrace{(g_{i_k}^{(k-1)} - \sum_{i=1}^{n} g_i^{(k-1)})}_{Y}.$$

It can be seen that $\mathbb{E}(X) = \nabla f(x^{(k)})$.
But that $\mathbb{E}(Y) \neq 0$, so that we have a *biased* estimator.

But we do have that Y seems correlated with X (in line with variance reduction template). In particular, we have that $X - Y \to 0$, as $k \to \infty$, since $x^{(k-1)}$ and $x^{(k)}$ converge to $\bar{x}$, the difference between first two terms converges to zero, and the last term converges to gradient at optimum, i.e. also to zero.

Thus, the overall estimator ℓ_2 norm (and accordingly its variance) decays to zero.

1

20.2. Modified SGD

Assume that $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$, where each f_i is differentiable, and ∇f_i is Lipschitz with constant L

Denote $\bar{x}^{(k)} = \frac{1}{k}\sum_{\ell=0}^{k-1} x^{(\ell)}$, the average iterate after $k-1$ steps

Theorem (Schmidt, Le Roux, Bach): SAG, with a fixed step size $t = 1/(16L)$, and the initialization

$$g_i^{(0)} = \nabla f_i(x^{(0)}) - \nabla f(x^{(0)}), \quad i = 1, \ldots n$$

satisfies

$$\mathbb{E}[f(\bar{x}^{(k)})] - f^\star \leq \frac{48n}{k}\left(f(x^{(0)}) - f^\star\right) + \frac{128L}{k}\|x^{(0)} - x^\star\|_2^2$$

where the expectation is taken over the random choice of index at each iteration

20.2. Modified SGD

- Result stated in terms of the average iterate $\bar{x}^{(k)}$, but also can be shown to hold for best iterate $x^{(k)}_{\text{best}}$ seen so far
- This is $O(1/k)$ convergence rate for SAG. Compare to $O(1/k)$ rate for FG, and $O(1/\sqrt{k})$ rate for SG
- But, the constants are different! Bounds after k steps:

$$\text{SAG}: \quad \frac{48n}{k}\big(f(x^{(0)}) - f^\star\big) + \frac{128L}{k}\|x^{(0)} - x^\star\|_2^2$$

$$\text{FG}: \quad \frac{L}{2k}\|x^{(0)} - x^\star\|_2^2$$

$$\text{SG}^*: \quad \frac{L\sqrt{5}}{\sqrt{2k}}\|x^{(0)} - x^\star\|_2 \quad \text{(*not a real bound, loose translation)}$$

- So first term in SAG bound suffers from factor of n; authors suggest smarter initialization to make $f(x^{(0)}) - f^\star$ small (e.g., they suggest using result of n SG steps)

20.2. Modified SGD

Assume further that each f_i is strongly convex with parameter m

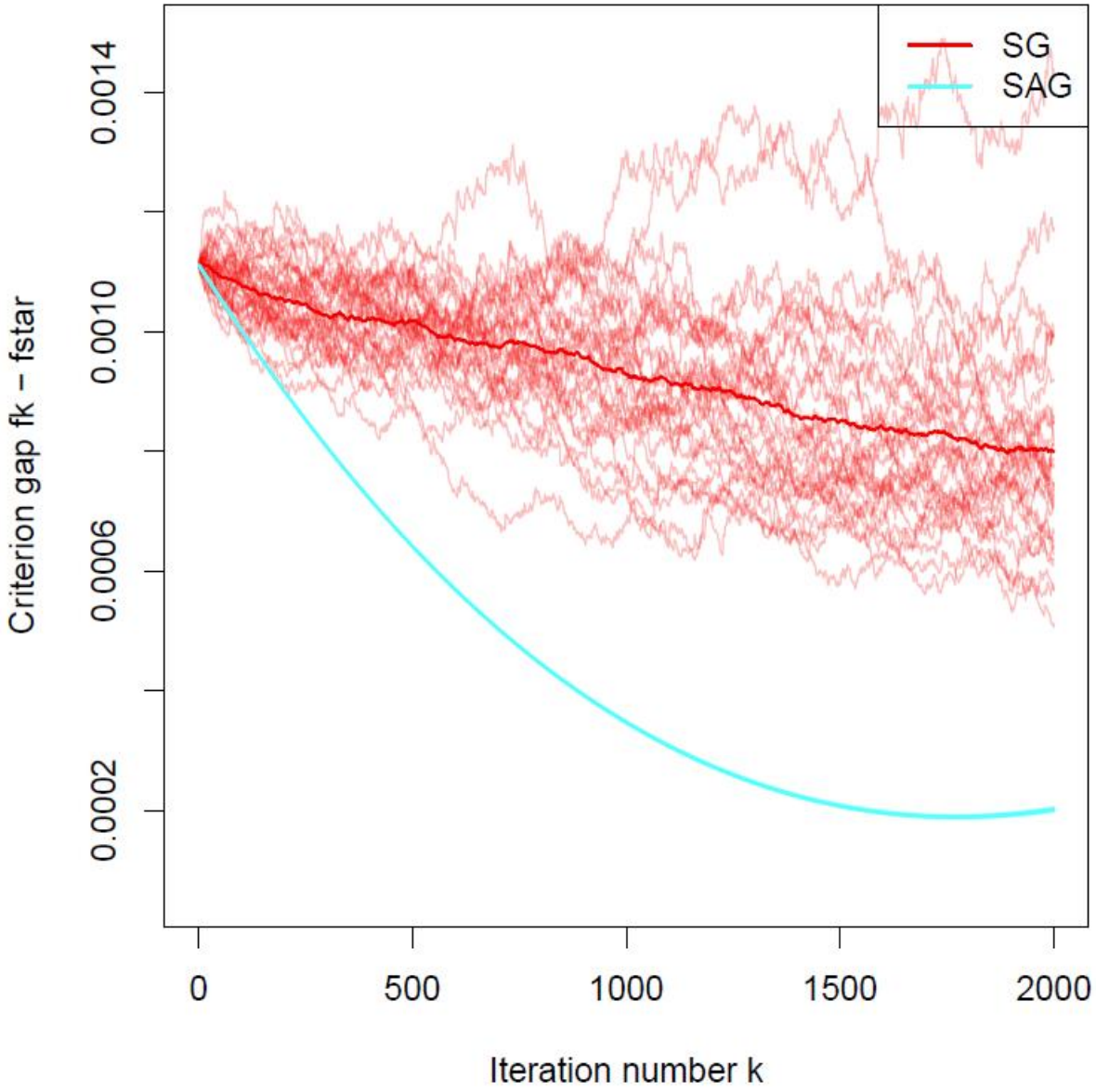
Theorem (Schmidt, Le Roux, Bach): SAG, with a step size $t = 1/(16L)$ and the same initialization as before, satisfies

$$\mathbb{E}[f(x^{(k)})] - f^\star \leq \left(1 - \min\left\{\frac{m}{16L}, \frac{1}{8n}\right\}\right)^k \cdot \left(\frac{3}{2}(f(x^{(0)}) - f^\star) + \frac{4L}{n}\|x^{(0)} - x^\star\|_2^2\right)$$

More notes:

- This is linear convergence rate $O(\rho^k)$ for SAG. Compare this to $O(\rho^k)$ for FG, and only $O(1/k)$ for SG
- Like FG, we say SAG is adaptive to strong convexity (achieves better rate with same settings)
- Proofs of these results not easy: 15 pages, computed-aided!

Back to our ridge logistic regression example, SG versus SAG, over 30 reruns of these randomized algorithms:



20.2. Modified SGD

- SAG does well, but did not work out of the box; required a specific setup
- Took one full cycle of SG (one pass over the data) to get $\beta^{(0)}$, and then started SG and SAG both from $\beta^{(0)}$. This warm start helped a lot
- SAG initialized at $g_i^{(0)} = \nabla f_i(\beta^{(0)})$, $i = 1, \ldots n$, computed during initial SG cycle. Centering these gradients was much worse (and so was initializing them at 0)
- Tuning the fixed step sizes for SAG was very finicky; here now hand-tuned to be about as large as possible before it diverges

20.2. Modified SGD

SAGA (Defazio, Bach, Lacoste-Julien, 2014) is another recent stochastic method, similar in spirit to SAG. Idea is again simple:

- Maintain table, containing gradient g_i of f_i, $i = 1, \ldots n$
- Initialize $x^{(0)}$, and $g_i^{(0)} = x^{(0)}$, $i = 1, \ldots n$
- At steps $k = 1, 2, 3, \ldots$, pick a random $i_k \in \{1, \ldots n\}$ and then let

$$g_{i_k}^{(k)} = \nabla f_i(x^{(k-1)}) \quad \text{(most recent gradient of } f_i\text{)}$$

  Set all other $g_i^{(k)} = g_i^{(k-1)}$, $i \neq i_k$, i.e., these stay the same
- Update

$$x^{(k)} = x^{(k-1)} - t_k \cdot \left(g_{i_k}^{(k)} - g_{i_k}^{(k-1)} + \frac{1}{n} \sum_{i=1}^{n} g_i^{(k-1)} \right)$$

20.2. Modified SGD

- SAGA gradient estimate $g_{i_k}^{(k)} - g_{i_k}^{(k-1)} + \frac{1}{n}\sum_{i=1}^n g_i^{(k-1)}$, versus SAG gradient estimate $\frac{1}{n}g_{i_k}^{(k)} - \frac{1}{n}g_{i_k}^{(k-1)} + \frac{1}{n}\sum_{i=1}^n g_i^{(k-1)}$
- Recall, SAG estimate is biased; remarkably, SAGA estimate is unbiased!

20.2. Modified SGD

Stochastic gradient in SAGA:

$$\underbrace{g_{i_k}^{(k)}}_{X} \underbrace{- \left(g_{i_k}^{(k-1)} - \frac{1}{n}\sum_{i=1}^{n} g_i^{(k-1)}\right)}_{Y}.$$

It can be seen that $\mathbb{E}(X) = \nabla f(x^{(k)})$.
And that $\mathbb{E}(Y) \neq 0$, so that we have an *unbiased* estimator.

Moreover, we have that Y seems correlated with X (in line with variance reduction template). In particular, we have that $X - Y \to 0$, as $k \to \infty$, since $x^{(k-1)}$ and $x^{(k)}$ converge to $\bar{x}$, the difference between first two terms converges to zero, and the last term converges to gradient at optimum, i.e. also to zero.

Thus, the overall estimator ℓ_2 norm (and accordingly its variance) decays to zero.

- SAGA basically matches strong convergence rates of SAG (for both Lipschitz gradients, and strongly convex cases), but the proofs here much simpler
- Another strength of SAGA is that it can extend to composite problems of the form

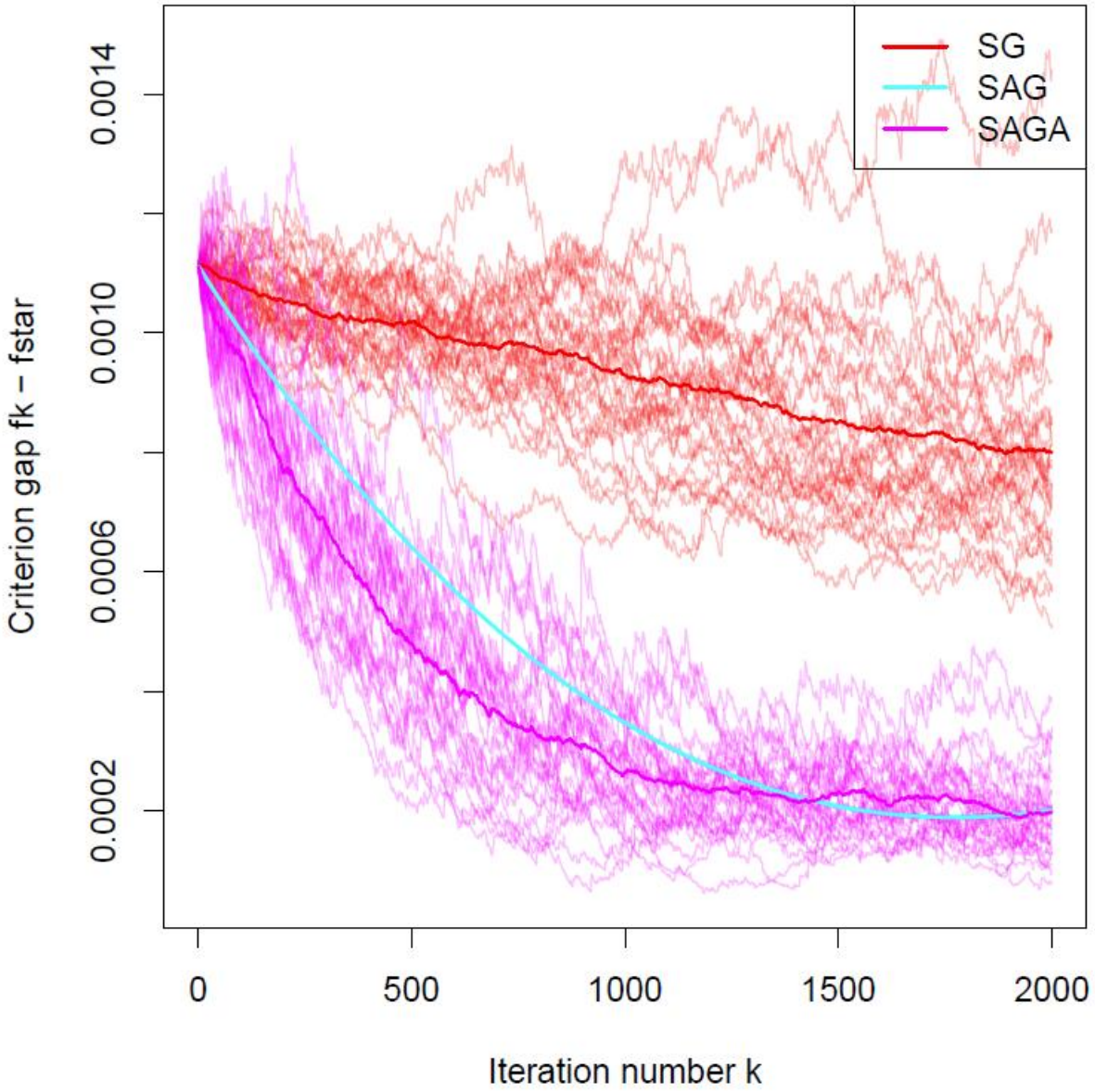
$$\min_x \frac{1}{n}\sum_{i=1}^m f_i(x) + h(x)$$

where each f_i is smooth and convex, and h is convex and nonsmooth but has a known prox. The updates are now

$$x^{(k)} = \text{prox}_{h,t_k}\left(x^{(k-1)} - t_k \cdot \left(g_i^{(k)} - g_i^{(k-1)} + \frac{1}{n}\sum_{i=1}^n g_i^{(k-1)}\right)\right)$$

- It is not known whether SAG is generally convergent under such a scheme

Back to our ridge logistic regression example, now adding SAGA to the mix:



20.2. Modified SGD

- SAGA does well, but again it required somewhat specific setup
- As before, took one full cycle of SG (one pass over the data) to get $\beta^{(0)}$, and then started SG, SAG, SAGA all from $\beta^{(0)}$. This warm start helped a lot
- SAGA initialized at $g_i^{(0)} = \nabla f_i(\beta^{(0)})$, $i = 1, \ldots n$, computed during initial SG cycle. Centering these gradients was much worse (and so was initializing them at 0)
- Tuning the fixed step sizes for SAGA was fine; seemingly on par with tuning for SG, and more robust than tuning for SAG
- Interestingly, the SAGA criterion curves look like SG curves (realizations being jagged and highly variable); SAG looks very different, and this really emphasizes the fact that its updates have much lower variance

20.2. Modified SGD

The Stochastic Variance Reduced Gradient (SVRG) algorithm (Johnson, Zhang, 2013) runs in epochs:

- Initialize $\widetilde{x}^{(0)}$.
- For $k = 1, \ldots$:
  - Set $\widetilde{x} = \widetilde{x}^{(k-1)}$.
  - Compute $\widetilde{\mu} := \nabla f(\tilde{x})$.
  - Set $x^{(0)} = \widetilde{x}$. For $\ell = 1, \ldots, m$:
    - Pick coordinate i_ℓ at random from $\{1, \ldots, n\}$.
    - Set:

$$x^{(\ell)} = x^{(\ell-1)} - \eta \left(\nabla f_{i_\ell}(x^{(\ell-1)}) - \nabla f_{i_\ell}(\widetilde{x}) + \tilde{\mu} \right).$$

  - Set $\widetilde{x}^{(k)} = x^{(m)}$.

Just like SAG/SAGA, but does not store a full table of gradients, just an average, and updates this occasionally.

SAGA: $\nabla f_{i_k}^{(k)} - \nabla f_{i_k}^{(k-1)} + \frac{1}{n}\sum_{i=1}^n \nabla f_i^{(k-1)}$,
SVRG: $\nabla f_{i_\ell}(x^{(\ell-1)}) - \nabla f_{i_\ell}(\widetilde{x}) + \frac{1}{n}\sum_{i=1}^n \nabla f_i(\widetilde{x})$.

Can be shown to achieve variance reduction similar to SAGA.

Convergence rates similar to SAGA, but formal analysis much simpler.

A lot of recent work revisiting stochastic optimization:

- SDCA (Shalev-Schwartz, Zhang, 2013): applies randomized coordinate ascent to the dual of ridge regularized problems. Effective primal updates are similar to SAG/SAGA.
- There's also S2GD (Konecny, Richtarik, 2014), MISO (Mairal, 2013), Finito (Defazio, Caetano, Domke, 2014), etc.

20.3. SGD for Deep Learning

优化理论里大家更在乎的是到 critical points 的收敛性，梯度逐渐收敛到0即可。深度学习里大家在乎是找到的是global minima，还是local minima，因为我们面临的是非凸优化。local minima 意味着 training loss 还比较高，还没优化到位。

所以深度学习动力学有两个非常值得研究的核心问题：

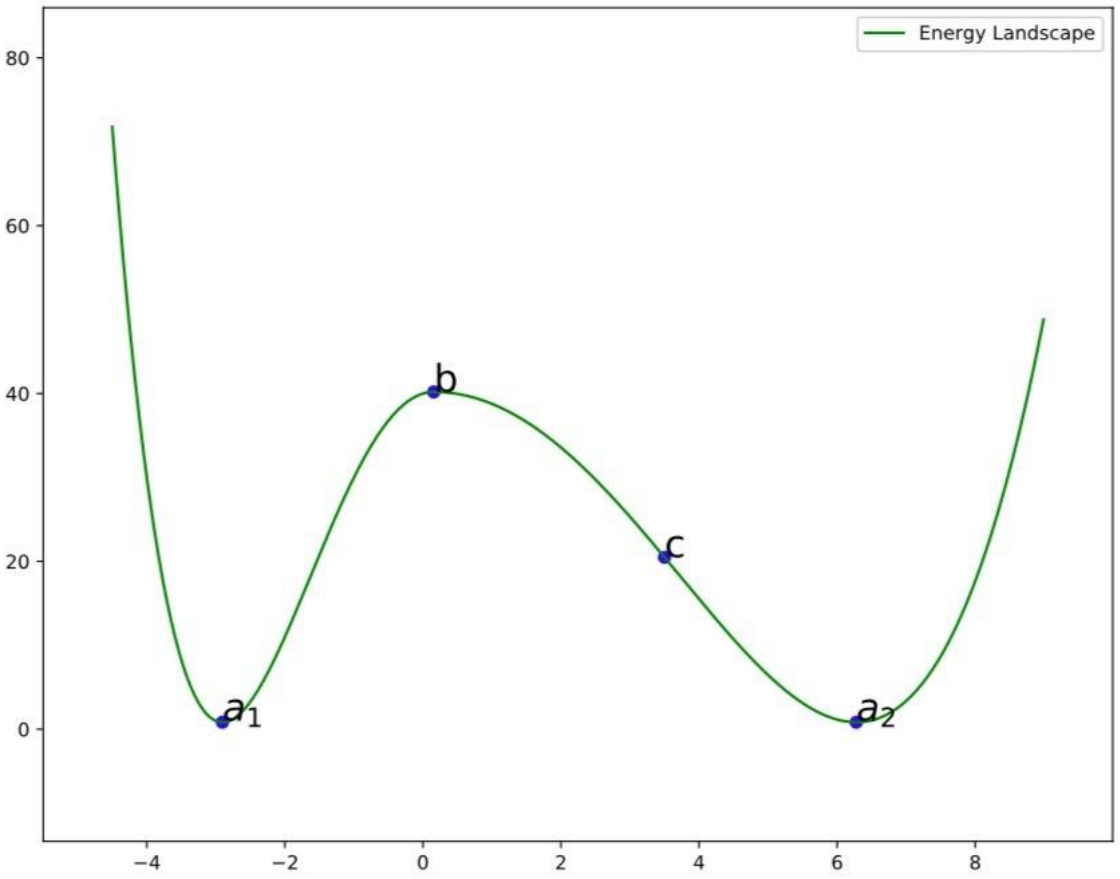
1) 怎么快速逃离鞍点；（鞍点逃逸问题一般指的是逃离 ε-first-order stationary points，其实是鞍点附近、梯度很小的区域，而不是梯度严格等于 0 的点。）

2) 怎么逃离 sharp minima 找到 flat minima。

上述问题的答案帮助我们更好地理解深度神经网络的训练过程，更有依据地调参或者设计新的随机优化器。SGD 为代表的随机优化器在这两个问题里都有相当好的性质。

20.3. SGD for Deep Learning

先来看一个一维示意图（高维空间也不难想象）。假如一个粒子初始时刻在能阱 a1 里，那么：Q1.它需要多长时间才能越过鞍点 b 进入势阱 a2？Q2.经过足够长的时间，这个粒子落入陷阱 a1 和 a2 的概率分别有多大？



20.3. SGD for Deep Learning

抽象一下，深度学习的训练过程其实就对应着这么个经典的 Kramers Escape 问题。统计物理学家 Kramers 提出了这么一个布朗运动粒子（服从朗之万动力学 Langevin Dynamics）的逃逸问题。经过一些统计物理学里的经典近似手段，得到了热噪音下（即各向同性的高斯噪音）的平均逃逸时间公式

$$\tau = 2\pi \sqrt{\frac{-\det(H_b)}{\det(H_a)}} \frac{1}{H_{be}} \exp\left(\frac{\Delta L_{ab}}{k_B T}\right)$$

ΔL是势阱深度；H_a和H_b分别是a和b处的$Hessian$；T是温度；k_b是统计物理里的玻尔兹曼常数。

很容易发现，在最简单的热噪音的情况，我们已经能看到随机动力学是偏好 flat minima 的，这种偏好是多项式级的。

20.3. SGD for Deep Learning

这里的$\det(H_a)$（一维情况下的二阶导数或者高维情况下的 Hessian 的行列式）就是 minima sharpness 的一种度量。在深度学习里，“类似”的方法则可以计算出 SGD 逃离 minima 的速率和 SGD 对 flat minima 的偏好。

大致的做法是从 SGD 算法得到对应的 Generalized Langevin Dynamics，再得到对应 Fokker-Planck Equation。从 Fokker-Planck Equation 可以解出粒子的概率密度。再借助 Smoluchowski Equation 可以解出粒子的概率密度从一个势阱流向另一个势陷的概率流大小。

可以直观的理解为，一堆概率云最开始在一个势阱内，渐渐地通过鞍点流向了另一个势阱，并逐渐达到平衡。达到平衡的特征时间就是平均逃逸时间τ，最终概率云的分布则反映了找到不同的解的概率。

Xie 等 2021 年文章表明：

$$\log(\tau) = O\left(\frac{B\Delta L_{ab}}{\eta H_a}\right)$$ （简单起见，这里只写一维空间的情况。高维空间类似。）

其中 B 是batch size, η 是学习率。这里可以看到，平均逃逸时间对minima sharpness的依赖是指数级的。

（1）热力学对于 flat minima 的偏好是多项式级的，而 SGD 对 flat minima 的偏好是指数级的。这就是为什么随机优化对深度学习如此重要。

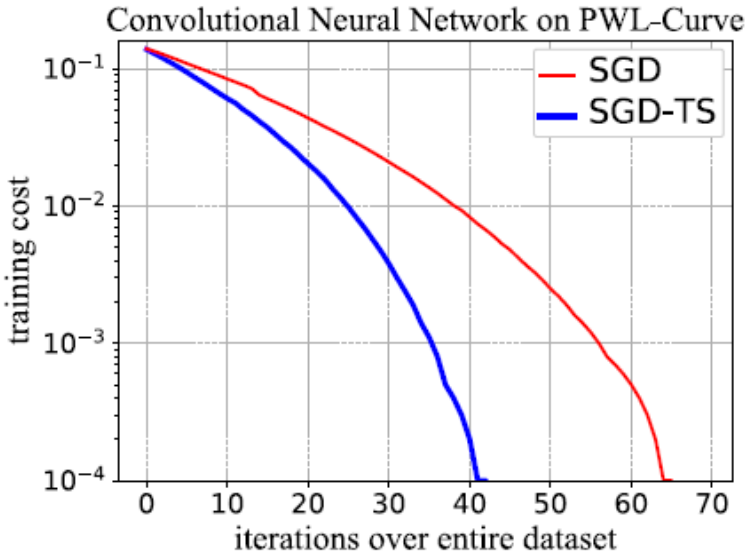
（2）这个 batch size 和学习率的比值 η / B 也是指数级的重要的。这个也解释为什么 large batch training 时 η / B 需要保持在一个稳定的值。另外，这个比值还可以影响深度学习的泛化。

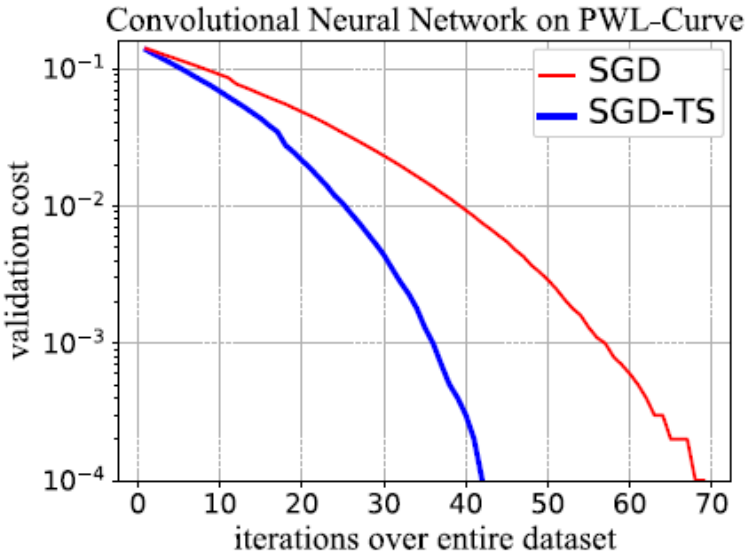
（3）深度学习的参数空间虽然很高维，但是学习动力学主要是发生在一个低维空间的。参数几乎不会沿着在 Hessian 的本征值接近 0 的那些方向学习。

20.3. SGD for Deep Learning

我们的新工作发现，其实每个样本的权重是不一样的，在计算 SGD 时，偏向重要的样本，也即重要的 $f_i(x)$，可以加速

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



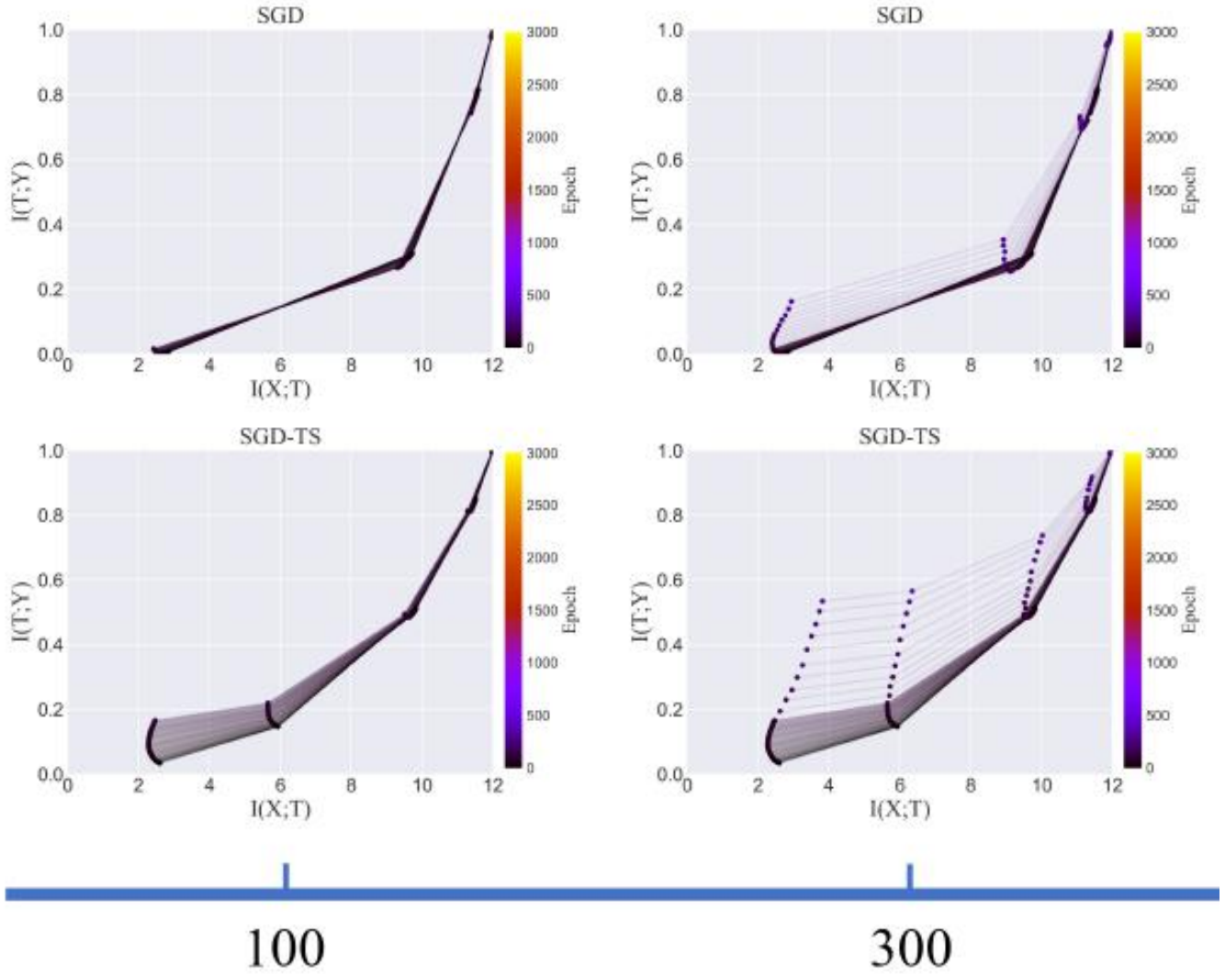


20.3. SGD for Deep Learning

Naftali Tishby
Dec 28, 1952 Jerusalem, State of Palestine - Aug 09, 2021, Jerusalem, State of Palestine

20.3. SGD for Deep Learning

偏向重要的样本可以帮助我们尽快通过信息瓶颈



20.4. References

[1] A. Nemirovski, A. Juditsky, G. Lan, A. Shapiro, "Robust stochastic optimization approach to stochastic programming," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1574-1609, 2009.

[2] L. Bottou, Stochastic Gradient Descent Tricks, https://leon.bottou.org/publications/pdf/tricks-2012.pdf

[3] O. Dekel, R. Gilad-Bachrach, O. Shamir, L. Xiao, "Optimal distributed online prediction using mini-batches," *Journal of Machine Learning Research*, vol. 13, pp. 165-202, 2012.

[4] M. Schmidt, N. Le Roux, F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, vol. 162, pp. 83-112, 2017.

[5] S. Shalev-Shwartz, T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *Journal of Machine Learning Research*, vol. 14, pp. 567-599, 2013.

[6] http://www.cs.cmu.edu/~aarti/Class/10725_Fall17/Lecture_Slides/stochastic_optimization_methods.pdf

[7] http://www.stat.cmu.edu/~ryantibs/convexopt/lectures/stochastic-gd.pdf

[8] Z. Xie, I. Sato, M. Sugiyama, "A diffusion theory For deep learning dynamics: Stochastic gradient descent exponentially favors flat minima," https://arxiv.org/abs/2002.03495v14

[9] https://www.zhihu.com/question/68109802/answer/1677007564

[10] X. Peng, L. Li, F.-Y. Wang, "Accelerating minibatch stochastic gradient descent using typicality sampling," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4649-4659, 2020.

[11] X. Peng, J. Zhang, F.-Y. Wang, L. Li, "Drill the cork of information bottleneck by inputting the most important data," *IEEE Transactions on Neural Networks and Learning Systems*, https://ieeexplore.ieee.org/document/9439803