Stochastic gradient descent

The main taks of empirical risk minimization for supervised learning is to minimize an average of functions,

$$\min_x \quad f(x) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(x), \qquad x \in \mathbb{R}^p$$

where we assume each f_i is *convex and differentiable*, $\mathrm{dom}(f_i) = \mathbb{R}^p$.

- $f_i(x)$ refers to the loss function from an individual sample labeled by i; there are totally n samples.
- x is the parameter in the model. Example: in linear regression, it is usually denoted as β and

$$f_i(\beta) = |y_i - X_i^T \beta|^2$$

- n is usually very large.
- We have not yet add the regularization term such as $\lambda\|x\|_2^2$ or $\lambda\|x\|_1$ in f.

Stochastic Optimization

We may use a generic random variable ξ to refer to the data point (y_i, X_i) which follows certain distribution ρ. Then the *population* risk is

$$\mathbb{E}_{\xi\sim\rho}[f(x;\xi)] = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} f(x;\xi_i), \qquad \xi_i \sim \rho$$

The Stochastic Optimization is to solve

$$\min_x \mathbb{E}_{\xi\sim\rho}[f(x;\xi)]$$

where ρ, the distribution of ξ, is fixed, i.e., independent of x (*exogenous* noise).

Sample Average Approximation (SAA)

$$\mathbb{E}_{\xi\sim\rho}[f(x;\xi)] \approx \frac{1}{n}\sum_{i=1}^{n} f(x;\xi_i)$$

for a finite n and solve

$$\min_x \frac{1}{n}\sum_{i=1}^{n} f(x;\xi_i)$$

Recall **Gradient Descent**:

- Choose initial $x^{(0)} \in \mathbb{R}^p$
- Repeat:

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

  where the step size is t_k.

> In this type of Gradient Descent, the whole training data set is used to compute the gradient of the cost function. It is also called (Full) Gradient Descent.
> This whole process of visiting the full batch of n data is called a cycle and a training **epoch**.

Main Challenge:

- It takes n operations to compute the gradient exactly: ∇f_i for all $1 \le i \le n$. This huge cost is not affordable for millions of data.

- IMAGENET [ImageNet](#) has a total number of images: 14,197,122
- The data points ξ_i, and f_i are not available in batch with all n, but online in a sequential manner of one observation at a time.

Stochastic gradient descent (SGD)

Here we introduce **Stochastic Gradient Descent**:

- Choose initial $x^{(0)} \in \mathbb{R}^p$
- Repeat:

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

  where the step size t_k is chosen to be fixed and small, $i_k \in \{1, \ldots, n\}$ is a ***randomly chosen index*** at iteration k.

> Note: $\mathbb{E}\left[\nabla f_{i_k}(x)\right] = \nabla f(x)$. $\nabla f_{i_k}(x)$ is an unbiased estimate of full gradient.

For the stochastic optimization

$$\min_x \mathbb{E}[f(x;\xi)],$$

the stochastic gradient $\nabla f(x;\xi)$ is used to replace the true gradient $\nabla \mathbb{E} f(x;\xi) = \mathbb{E}[\nabla f(x;\xi)]$:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f\left(x^{(k-1)}; \xi_k\right), \quad k = 1, 2, 3, \ldots$$

Choose index

Two rules for choosing index i_k at iteration k :

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

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

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

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 n (the number of functions)
- Can also be a big saving in terms of memory usage

Mini-batches

Also common is **Mini-batch Stochastic Gradient Descent**. Instead of choosing one single index, we choose a random **subset** $I_k \subseteq \{1, \dots, n\}$ of size $|I_k| = b \ll n$, and repeat:

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

> 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 of our gradient estimate by a factor $1/b$, but is also b times more expensive

Step size

The general rule in SGD is to use *diminishing* step sizes, e.g., $t_k = 1/k$ (or more general $1/k^\alpha$ with $\alpha \in [0.5, 1]$).
Why not the fixed step sizes? Here's some intuition.
Suppose we take cyclic rule for simplicity. Set $t_k = t$ for n updates in a row for SGD, we get:

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

Meanwhile, full gradient with step size t would give:

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

The difference here:

$$t \sum_{i=1}^{n} \left[\nabla f_i \left(x^{(k+i-1)} \right) - \nabla f_i \left(x^{(k)} \right) \right],$$

and if we hold t constant, this difference will not generally be going to zero.
For the randomized index rule, the difference with random i:

$$t \nabla f(x) - t \nabla f_i(x)$$

has zero mean, but non-zero variance:

$$t^2 \mathbb{E} \big(\nabla f(x) - \nabla f_i(x) \big)^2 = t^2 \operatorname{Var}(\nabla f_i(x))$$

where the variance is respect to the random index i.

A diminishing step size can help reduce the variance (fluctuation) of the solution.

Illustrative Example of SGD:

Consider

$$\min_x f(x) = \frac{1}{2}\mathbb{E}(x-\xi)^2 = \frac{1}{2}\int (x-\xi)^2 \rho(\xi) d\xi$$

The minimizer $x^\star$ satisfies $\nabla f(x^\star) = 0$, i.e.,

$$\mathbb{E}(x^\star - \xi) = 0, \qquad \Rightarrow x^\star = \mathbb{E}(\xi)$$

- Applied SGD with constant step size η to this problem. Since $\nabla f(x;\xi) = (x - \xi)$, then the SGD iteration is

  $$x^{(k)} = x^{(k-1)} - \eta(x^{(k-1)} - \xi^{(k)}) = (1-\eta)x^{(k-1)} + \eta\xi^{(k)}$$

  where $\xi^{(k)}$ are iid. So

  $$\begin{aligned} x^{(k)} &= \left((1-\eta)^2 x^{(k-2)} + (1-\eta)\eta\xi^{(k-1)}\right) + \eta\xi^{(k)} \\ &= (1-\eta)^3 x^{(k-3)} + (1-\eta)^2\eta\xi^{(k-2)} + (1-\eta)\eta\xi^{(k-1)} + \eta\xi^{(k)} \\ &= (1-\eta)^k x^{(0)} + \eta\sum_{j=1}^{k}(1-\eta)^{j-1}\xi^{(k+1-j)} \end{aligned}$$

  Let $1-\eta \in (0,1)$, then as $k \to \infty$,

  $$\mathbb{E}(x^{(k)}) \to \eta\sum_{j=1}^{\infty}(1-\eta)^{j-1} \times \mathbb{E}(\xi) = \mathbb{E}(\xi)$$

  The variance is then

  $$\mathrm{Var}[x^{(k)}] = \eta^2\sum_{j=1}^{k}(1-\eta)^{2(j-1)}\,\mathrm{Var}(\xi_{k-j}) \to \frac{\eta}{2-\eta}\,\mathrm{Var}(\xi)$$

  So, the limit of x_k is a *random variable* with

  - The mean $x^\star = \mathbb{E}(\xi)$

  - The *standard deviation* = $\sqrt{\frac{\eta}{2-\eta}}\,\mathrm{std}(\xi)$.

> $\frac{\eta}{2-\eta} \leq 1/2$. For small η, $\frac{\eta}{2-\eta} \approx \frac{1}{2}\eta + O(\eta^2)$

- Applied SGD with the diminishing step size $\eta^{(k)} = \frac{1}{k}$ to this problem with $x^{(0)} = 0$. Then the SGD iteration is

$$x^{(k)} = x^{(k-1)} - \eta_k\left(x^{(k-1)} - \xi^{(k)}\right) = \frac{k-1}{k}x^{(k-1)} + \frac{1}{k}\xi^{(k)}, \quad k \geq 2$$

where $\xi^{(k)}$ are iid. So

$$x^{(k)} = \frac{1}{k}\sum_{j=1}^{k}\xi^{(j)}$$

is just the empirical average (unbiased estimator) of the random variable ξ. And as $k \to \infty$, we know

$$x^{(k)} \to \mathbb{E}(\xi)$$

(converges to a *constant*) almost surely and $\sqrt{k}x_k$converges to the Gaussian variable $\mathcal{N}(\mathbb{E}(\xi), \mathrm{Var}(\xi))$ by the central limit theorem.

Example: logistic regression

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

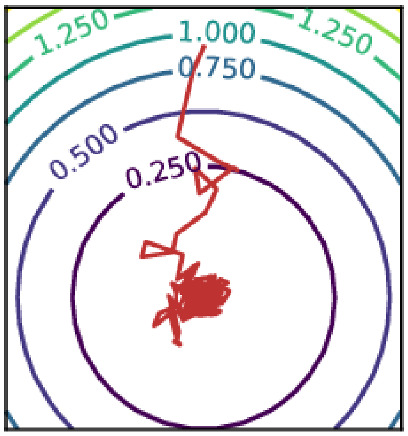
$$\min_{\beta} \frac{1}{n}\sum_{i=1}^{n}\underbrace{\left(-y_i x_i^T\beta + \log\left(1 + \exp\left(x_i^T\beta\right)\right)\right)}_{f_i(\beta)}$$

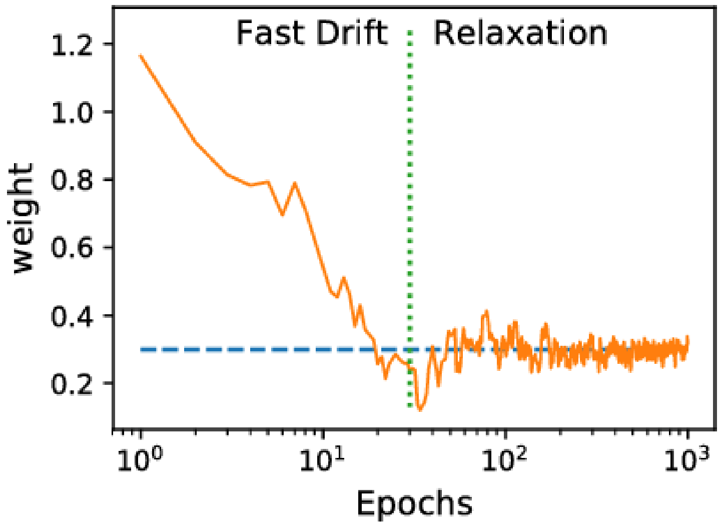
Gradient computation $\nabla f(\beta) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - p_i(\beta)\right)x_i$ is doable when n is moderate, but not when n is huge
Full gradient (also called batch) versus stochastic gradient:

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

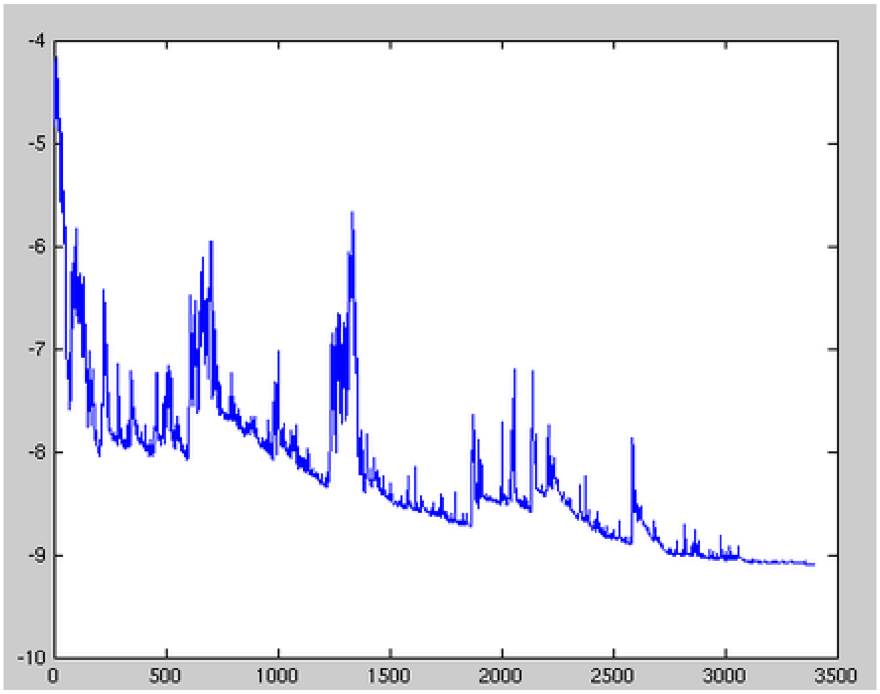
Drift vs Fluctuation

- Convex function





- Non-convex function
  Fluctuation in the total objective function



Convergence rates

We have the following table

Condition	GD rate	SGD rate
Convex	$O(1/\sqrt{k})$	$O(1/\sqrt{k})$
+ Lipschitz gradient	$O(1/k)$	$O(1/\sqrt{k})$
+ Strongly convex	$O\left(\gamma^k\right)$	$O(1/k)$

Notes:

- GD takes n (the number of functions) operations while SGD only takes one in each iteration from k to $k+1$;
- In GD, we can take fixed step sizes in the latter two cases
- In SGD, we always take diminishing step sizes $t_k \sim 1/k$ to control the variance (of the gradient estimate)

- The current theory is not convincing on whether mini-batches help (but still popular practice)
- reference: Nemirovski et al (2009), Bach & Moulines (2011)

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)

> Is this the end of the story for SGD?

For a while, the answer was believed to be yes.

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

SGD in practice

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** in the **training** process, 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[1]
- Momentum/acceleration, averaging, adaptive step sizes are all popular variants in practice
- SGD is especially popular in large-scale, continuous, nonconvex optimization, but it is still not particular well-understood there (a big open issue is that of implicit regularization)

Early stopping

- stop early, i.e., terminate gradient descent well-short of the global minimum
- it was thought of an implicit **regularization** or more advanced/deep understanding[2]
- some discussion on this trick

Variance reduction methods

Iterate Averaging (Polayk-Ruppert averaging)

Averaged stochastic gradient descent, invented independently by Ruppert and Polyak in the late 1980s, is ordinary stochastic gradient descent that records an average of its parameter vector over time. That is, the update is the same as for ordinary stochastic gradient descent for $x^{(k)}$, but the algorithm also keeps track of the average

$$\bar{x}^{(k)} = \frac{1}{k}\sum_{i=1}^{k} x^{(i)}, \quad \text{constant stepsize}$$

If the step size t_k is not fixed, then the output is the weighted average

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

> Write the recursive relation from $\bar{x}^{(k-1)}$ to $\bar{x}^{(k)}$.

- In general, using the step size $t^{(k)} \sim \frac{1}{\sqrt{k}}$ and iterate averaging is a good choice because it is adaptive to strong convexity while being robust to non-smoothness and non-strong convexity.

Stochastic Average Gradient (SAG)

Stochastic average gradient [3] is a breakthrough method in stochastic optimization. The SAG method has the low iteration cost of SGD methods, but achieves the convergence rates stated above for the full GD method.

- Maintain table, containing gradient g_i of $f_i, i = 1, \dots, n$
- Initialize $x^{(0)}$, and $g_i^{(0)} = \nabla f_i\left(x^{(0)}\right), i = 1, \dots, n$
- At steps $k = 1, 2, 3, \dots,$ pick random $i_k \in \{1, \dots n\}$, then only update

$$g_{i_k}^{(k)} = \nabla f_{i_k}\left(x^{(k-1)}\right)$$

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

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

Notes:

- The gradient used to update $x^{(k)}$ still consists of n gradient functions; but only one (or a mini-batch) of them is really computed and updated while all others are unchanged as *history*.
- 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? Basically just as efficient as SGD, as long we're clever:

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

  We do not need to recompute the whole n gradient function actually !

- But the memory is an issue: a table of the gradient g_i for many $1 \leq i \leq n$ is needed.

SAG convergence analysis

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

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

Then the iterate average $\bar{x}^{(k)}$
satisfies

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

Notes:

- n is the number of functions in the sum $f=\sum_{i=1}^{n} f_i$.
- Result stated in terms of the average iterate $\bar{x}^{(k)}$, but also can be shown to hold for best iterate $x_{\text{best}}^{(k)}$ seen so far.
- This is $O(1/k)$ convergence rate for SAG. Compare to $O(1/k)$ rate for GD, and $O(1/\sqrt{k})$ rate for SGD
- But, the constants are different! Bounds after k steps:

$$\begin{aligned}\text{GD}: &\quad \frac{L}{2k}\left\|x^{(0)}-x^{\star}\right\|_2^2 \\ \text{SAG}: &\quad \frac{48n}{k}\left(f\left(x^{(0)}\right)-f^{\star}\right)+\frac{128L}{k}\left\|x^{(0)}-x^{\star}\right\|_2^2\end{aligned}$$

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

Convergence under strong convexity

Assume further that each f_i is strongly convex with parameter m
Theorem: SAG with a step size $t=1/(16L)$ and the same initialization as before, satisfies

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

> Notes:

> - This is linear convergence rate $O\left(\gamma^k\right)$ for SAG. Compare this to $O\left(\gamma^k\right)$ for GD, and only $O(1/k)$ for SGD.
> - Like GD, we say SAG is adaptive to strong convexity
> - Proofs of these results not easy: Schmidt, M., Le Roux, N. & Bach, F. Minimizing finite sums with the stochastic average gradient. *Math. Program.* **162**, 83–112 (2017). https://doi.org/10.1007/s10107-016-1030-6

SAG does well when it requires a specific setup.

> - Took one full cycle of SGD (one pass over the data) to get $x^{(0)}$, and then started SGD and SAG both from $x^{(0)}$. This ***warm start*** helped a lot
> - SAG initialized at $g_i^{(0)} = \nabla f_i\left(x^{(0)}\right), i = 1, \ldots, n$, computed during initial SGD 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

Authors of SAG conveyed that this algorithm will work the best, relative to SGD, for ill-conditioned problems.

SAGA

SAGA[4] (Defazio, Bach, and Lacoste-Julien 2014) is a follow-up on the **SAG** work:

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

$$g_{i_k}^{(k)} = \nabla f_{i_k}\left(x^{(k-1)}\right)$$

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

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

> Note that there is no $\frac{1}{n}$ in front of the term $g_{i_k}^{(k)} - g_{i_k}^{(k-1)}$.

Comparation between SAG and SAGA

	SAG	**SAGA**
Gradient	$\frac{1}{n} g_{i_k}^{(k)} - \frac{1}{n} g_{i_k}^{(k-1)} + \frac{1}{n} \sum_{i=1}^{n} g_i^{(k-1)}$	$g_{i_k}^{(k)} - g_{i_k}^{(k-1)} + \frac{1}{n} \sum_{i=1}^{n} g_i^{(k-1)}$
Bias or not	Biased	Unbiased
Convergence rate	$O\left(\frac{1}{k}\right)$	$O\left(\frac{1}{k}\right)$
Convergence rate under strong convexity	$O\left(\gamma^k\right)$	$O\left(\gamma^k\right)$

SAGA matches convergence rates of SAG, with simpler proof

> - Why SAG estimate is biased while SAGA estimate is unbiased? Consider family of estimators
>
> $$\theta_\alpha = \alpha(X - Y) + \mathbb{E}(Y)$$
>
> for $\mathbb{E}(X)$, where $\alpha \in [0, 1]$, and X, Y are presumed correlated. We have

> $$\mathbb{E}(\theta_\alpha) = \alpha\mathbb{E}(X) + (1-\alpha)\mathbb{E}(Y)$$
> $$\operatorname{Var}(\theta_\alpha) = \alpha^2(\operatorname{Var}(X) + \operatorname{Var}(Y) - 2\operatorname{Cov}(X,Y))$$
>
> SAGA uses $\alpha = 1$ (unbiased), SAG uses $\alpha = 1/n$ (biased)

SAGA does well, but again it required specific setup:

> - As before, **took one full cycle of SGD to get** $x^{(0)}$, and then started SGD, SAG, SAGA all from $x^{(0)}$.
> - **Initial** $g_i^{(0)}, i = 1, \ldots, n$, **at 0**.
> - Tuning the fixed step sizes for SAGA was fine, seemingly on par with tuning for SGD, and more robust than tuning for SAG.

Interestingly, the SAGA criterion curves look like SGD curves (realizations being jagged and highly variable); SAG looks very different, and this really emphasizes the fact that its updates have much lower variance

SVRG

Stochastic variance reduced gradient[5] (Johnson & Tong 2013) is another variance reduction method.

- Initialize $\hat{x}^{(0)}$.
- At step $k = 1, 2, \ldots$, set:

$$\hat{x} = \hat{x}^{(k-1)}$$

  - For $j = 1, \ldots, m$ (m is the update frequency of each step), randomly pick $i_j \in \{1, \ldots, n\}$ with initial

$$x_0 = \hat{x}$$

    and update the x_t in this inner loop

$$x_j = x_{j-1} - t \cdot \left(\nabla f_{i_j}(x_{j-1}) - \nabla f_{i_j}(\hat{x}) + \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\hat{x}) \right)$$

- Option 1: Set $\hat{x}^{(k)} = x_m$.
- Option 2: Set $\hat{x}^{(k)} = x_t$ for randomly chosen $t \in \{0, \ldots, m-1\}$.

> In the inner m loops, the average gradient $\frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\hat{x})$ is unchanged: in other words, it is updated only every m steps.

There is ***no need of a table*** to save n gradients.

Convergence analysis of SVRG

Theorem: Consider SVRG with option 2. Assume that $f_i(x)$ are convex and smooth, $\sum_{i=1}^{n} f_i(x)$ is strongly convex. i.e.

$$\begin{aligned} f_i(x) - f_i(x') - 0.5L\|x - x'\|_2 &\leq \nabla f_i(x')(x - x') \\ f(x) - f(x') - 0.5\gamma\|x - x'\|_2^2 &\leq \nabla f(x')(x - x') \end{aligned}$$

where $f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$ and $L \geq \gamma \geq 0$.
Let $x^\star$ be the optimal solution. Assume that m is sufficiently large so that

$$\gamma \triangleq \frac{1}{\gamma t(1 - 2Lt)m} + \frac{2Lt}{1 - 2Lt} < 1$$

then we have geometric convergence in expectation for SVRG:

$$\mathbb{E} f(\hat{x}^{(k)}) \leq \mathbb{E} f(x^\star) + \gamma^k (f(\hat{x}^{(0)}) - f(x^\star))$$

> Note:
>
> - SVRG algorithm can be applied to smooth but not strongly convex problems. A convergence rate of $O(1/k)$ may be obtained.
> - If the system is locally (strongly) convex, the theorem can be directly applied, which implies local geometric convergence rate with a constant learning rate.

Many, many others methods

Many other ideas for variance reduction in stochastic optimization:

- SDCA (Shalev-Schwartz, Zhang, 2013): applies coordinate ascent to the dual of ridge regularized problems, and uses randomly selected coordinates. Effective primal updates are similar to SAG/SAGA.
- S2GD (Konecny, Richtarik, 2014), MISO (Mairal, 2013) , Finito (Defazio, Caetano, Domke, 2014), etc.
- Both the SAG and SAGA papers give very nice reviews and discuss connections
- The list is still growing...

References and further reading

- J. Duchi and E. Hazan and Y. singer (2010), "Adaptive subgradient methods for online learning and stochastic optimization".
- A. Defasio and F. Bach and S. Lacoste-Julien (2014), "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives".
- G. Lan and Y. Zhou (2015), "An optimal randomized incremental gradient method"
- A. Nemirovski and A. Juditsky and G. Lan and A. Shapiro (2009) "Robust stochastic optimization approach to stochastic programming"
- M. Schmidt and N. Le Roux and F. Bach (2013), "Minimizing finite sums with the stochastic average gradient".

---

1. Bottou (2012), "Stochastic gradient descent tricks" ↩
2. http://proceedings.mlr.press/v51/duvenaud16.pdf ↩
3. https://arxiv.org/pdf/1309.2388.pdf ;
   https://link.springer.com/article/10.1007/s10107-016-1030-6 ↩

4. https://papers.nips.cc/paper/5258-saga-a-fast-incremental-gradient-method-with-support-for-non-strongly-convex-composite-objectives.pdf ↩

5. https://papers.nips.cc/paper/4937-accelerating-stochastic-gradient-descent-using-predictive-variance-reduction.pdf "Accelerating Stochastic Gradient Descent using Predictive Variance Reduction: ↩