**EE C227C Convex Optimization and Approximation** Lecture 25 - 4/19/2022

# Lecture 25: Stochastic Variance Reduced Gradient Method

*Lecturer: Jiantao Jiao* *Scribe: Vivek Bharadwaj and Sohom Paul*

## 1 Review of Stochastic Optimization

Last time, we reviewed some classical results in stochastic gradient methods. To recall the problem setup, we consider some minimization problem

$$\min_x f(x) := \mathbb{E}_\xi[F(x, \xi)]$$

where $\xi$ is some random variable. Rather than querying our gradient oracle for $\nabla f(x)$ directly, we instead have access to a randomized oracle that samples $\xi \sim \mathcal{D}_\xi$, where $\mathcal{D}_\xi$ is some distribution, and returns $(F(x, \xi), \nabla_x F(x, \xi))$.

However, in many cases, the function we want to minimize can be expressed as a sum[1]

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

This "stochastic oracle" formulation is very common in machine learning, where the overall loss of our model can be expressed as a sum of losses over the training point. In such a setting, it is common to have our stochastic gradient step be sampling $i \sim \text{Uniform}[n]$ and then we get gradient $\nabla f_i$. For example, this is the case for stochastic gradient descent on set of discrete datapoints, where $\nabla f_i$ is the gradient of the loss function at a point $p_i$. We have $\mathbb{E}_i \nabla f_i = \nabla f$, and we can pass over these data points as many times as we wish when performing the optimization. Note this approach minimizes the empirical loss $\frac{1}{m}\sum_{i=1}^{m} \ell(x, \xi)$ rather than the *expected loss* $\mathbb{E}_\xi \ell(x, \xi)$. A stochastic oracle can also be used to minimize the latter expression (which asks us to minimize expected out-of-sample loss based on samples taken from the distribution) but convergence guarantees require that we only make a single pass over the datapoints [1].

It turns out that this formulation lets us have much more powerful optimization techniques. Intuitively, this is because drawing a large number of samples $i$ lets us deduce the overall structure of $f$ due to the strong structure requirements on $f$.

Let us recall some of the results from the previous lecture. In the general setting where our gradient queries are $\nabla F(x, \xi)$, we found that for convex, Lipshitz functions we had $O(1/\sqrt{T})$ convergence and similarly for $\mu$-strongly convex Lipschitz functions (with known $\mu$) we have $O\left(1/(\mu T)\right)$ convergence. There are similar bounds for when the function is $L$-smooth instead of $L$-Lipschitz.

However, in the non-stochastic setting we were able to achieve $O(1/T)$ and $O(1/T^2)$ convergence for gradient descent and accelerated gradient descent respectively, which is much faster than the bounds above. We will work to tighten this gap, although we cannot tighten it completely. For general stochastic oracles, a lower bound by Tsybakov (2003) shows that smoothness of the function $f$ will cannot yield any acceleration compared to the non-smooth case, ruling out quadratic $O(1/T^2)$ convergence [2, 1]. Nevertheless, we can interpolate between $O(1/\sqrt{T})$ and $O(1/T)$ convergence using the technique of stochastic variance reduction.

## 2 Stochastic Variance Reduced Gradient Method

### Assumptions

Throughout the rest of this lecture, we impose the following assumptions:

[1]We have a small notation change here; in previous lectures we had $f(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x)$. This change is to be consistent with the machine learning literature.

1. Each $f_i$ is $L$-smooth (e.g. $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x-y\|$).
2. The overall function $f$ is $\mu$-strongly convex.
3. Querying $\nabla f_i(x)$ takes $O(1)$ time and $\nabla f(x)$ takes $O(n)$ time. Note that this differs from our previous lectures, where we assumed querying $\nabla f(x)$ takes $O(1)$ time.

Define the condition number $\kappa := L/\mu$ to be some measure of the difficulty of our problem. For sake of comparison, we consider two algorithms we analyzed previously:

| | Gradient Descent | Stochastic Gradient Descent |
|---|---|---|
| Update Rule | $x_{k+1} = x_k - \eta\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(x)$ | $x_{k+1} = x_k - \eta\nabla f_{i_k}(x_k)$ |
| Iteration Complexity | $n\kappa\log(1/\varepsilon)$ | $1/(\mu\varepsilon)$ |

(As a technical aside, note that this setting differs from last lecture because we have Lipschitz gradient rather than bounded gradient; however, the same bound holds.)

Both gradient descent (GD) and stochastic gradient descent (SGD) correspond to batch gradient descent (BGD) with different batch sizes, so we expect the run time for BGD to be an interpolation between these two. However, we have $n$ in the iteration complexity and linear convergence for GD, while for SGD we have dependence on $\mu$ and a much slower convergence rate for small $\varepsilon$. It is rather confusing and we would ideally like an algorithm that does not depend on knowing $\mu$ and $L$.

## 2.1 Variance Reduction

We will present an algorithm with iteration complexity $(n+\kappa)\log(1/\varepsilon)$. Note that this is much better than the iteration complexity for GD because we are adding $n$ and $\kappa$ instead of multiplying. Both of these numbers can be large in practice.

Recall that for GD we can maintain a fixed learning rate as the step size will naturally decrease as the gradients decrease. However, for SGD, the noise in the gradient forces us to decrease the learning rate as we are converging. We ultimately want an algorithm that maintains a low variance gradient estimate while maintaining the small number of calls to the gradient oracle that SGD affords.

We use the following gradient estimate:

$$\nabla f_i(x) - \nabla f_i(y) + \nabla f(y)$$

for $i \sim \text{Uniform}[n]$. Note that the first two terms above require only 1 call to the gradient oracle each, while the last term requires $n$ calls. The sequence of $y$'s is referred to as a *snapshot sequence* or *centering sequence* and is updated less frequently than $x$ (since it requires the computation of a full gradient of $f$ with respect to $y$, $O(n)$ time).

We have two important properties of the gradient estimate above:

1. It is unbiased. We can compute

$$\mathbb{E}[\nabla f_i(x) - \nabla f_i(y) + \nabla f(y)] = \nabla f(x) - \nabla f(y) + \nabla f(y) = \nabla f(x)$$

2. If $x$ and $y$ are both very close to the global min of the function, we can use the Lipschitz property of $f_i$ to observe that $\nabla f_i(x) - \nabla f_i(y) \leq L\|x-y\| \approx 0$. This makes the estimator low variance when we are close to convergence.

The full algorithm is provided in Algorithm 1.

Note that there is a strange random sampling step to choose snapshots, which is not done in practice but is required for the proof to work. In practice $m$ is increased for each outer loop iteration, because as we get close to the optimum, we don't want to recompute $\nabla f(\tilde{x}_s)$ too often.

```
initialize $\tilde{x}_0$ and inner loop size $m$;
for $s = 0, 1, 2, \dots$ do
    sample $N \sim \text{Uniform}\{0, \dots, m-1\}$;
    start at $x_0 = \tilde{x}_s$;
    for $k = 0, 1, 2, \dots$ do
        sample $I_k \sim \text{Uniform}[n]$;
        $x_{k+1} \leftarrow x_k - \eta\,[\nabla f_{I_k}(x_k) - \nabla f_{I_k}(\tilde{x}_s) - \nabla f(\tilde{x}_s)]$;
    end
    $\tilde{x}_{s+1} = x_N$;
end
```

**Algorithm 1:** SVRG Algorithm

## 2.2 Analysis

We have the following performance guarantee for the SVRG algorithm:

**Theorem 1** (Johnson & Zhang, '13)**.** *For $s \geq 0$, and the above algorithm, we have*

$$2\eta(1-2\eta L)(\mathbb{E}f(\tilde{x}_{s+1}) - f^*) + \eta\mu\mathbb{E}\left\|\tilde{x}_{s+1} - x^*\right\|^2 \leq 4L\eta^2(\mathbb{E}f(\tilde{x}_s) - f^*) + \frac{1}{m}\mathbb{E}\left\|\tilde{x}_s - x^*\right\|^2 \tag{1}$$

*If $\eta = 1/(8L)$ and $m \geq 2/(\eta\mu) = 16L/\mu = 16\kappa$, then*

$$\mathbb{E}[f(\tilde{x}_{s+1})] - f^* + \mu\mathbb{E}\left\|\tilde{x}_{s+1} - x^*\right\|^2 \leq \left(\frac{1}{2}\right)^s \left[f(\tilde{x}_0) - f^* + \mu\left\|\tilde{x}_0 - x^*\right\|^2\right] \tag{2}$$

Note the final line tells us that our measure of suboptimality (which combines the absolute suboptimality and the distance of the optimizer to the optimum) decreases geometrically, so we have $O(\log(1/\varepsilon))$ outer loop iterations.

**Proof** Define

$$g_i(x) := f_i(x) - f_i(x^*) - \langle \nabla f_i(x^*), x - x^* \rangle$$

and observe $g_i(x) \geq 0$ using the definition of convexity. Using the fact that $f_i$ is $L$-smooth and Theorem 2.1.5 of [3], we deduce

$$g_i(x) \geq \frac{1}{2L}\left\|\nabla g_i(x)\right\|^2 = \frac{1}{2L}\left\|\nabla f_i(x) - \nabla f_i(x^*)\right\|^2$$

This immediately implies the following inequality

$$\left\|\nabla f_i(x) - \nabla f_i(x^*)\right\|^2 \leq 2L\left(f_i(x) - f_i(x^*) - \langle \nabla f_i(x^*), x - x^* \rangle\right)$$

Taking expectations of both sides, using $i \sim \text{Uniform}[n]$,

$$\begin{aligned}\mathbb{E}\left\|\nabla f_i(x) - \nabla f_i(x^*)\right\|^2 &\leq \mathbb{E}\left[2L\left(f_i(x) - f_i(x^*) - \langle \nabla f_i(x^*), x - x^* \rangle\right)\right] \\ &= 2L(f(x) - f(x^*))\end{aligned}$$

where we use the fact that $\mathbb{E}[\nabla f_i(x^*)] = \nabla f(x^*) = 0$. The above fact is known as the *Fundamental Lemma of SVRG* because it can be used to relate the variance of the observed gradient to the suboptimality gap. We can still apply a version of SVRG to optimize non-convex functions, but the above lemma no longer holds and weakens the convergence guarantee. Now, define

$$h_k := \nabla f_{I_k}(x_k) - \nabla f_{I_k}(\tilde{x}_s) + \nabla f(\tilde{x}_s)$$

where we are in the inner loop where we start indexing at $x_0 = \tilde{x}_s$. All expectations will be with respect to $I_k$.

$$\begin{aligned}\mathbb{E}\left\|h_k\right\|^2 &= \mathbb{E}\left\|\nabla f_{I_k}(x_k) - \nabla f_{I_k}(x^*) + \nabla f_{I_k}(x^*) - \nabla f_{I_k}(\tilde{x}_s) + \nabla f(\tilde{x}_s)\right\|^2 \\ &\leq 2\mathbb{E}\left\|\nabla f_{I_k}(x_k) - \nabla f_{I_k}(x^*)\right\|^2 + 2\mathbb{E}\left\|\nabla f_{I_k}(\tilde{x}_s) - \nabla f_{I_k}(x^*) - \nabla f(\tilde{x}_s) + \nabla f(x^*)\right\|^2\end{aligned}$$

using the identity $(a+b)^2 \leq 2a^2 + 2b^2$. We continue bounding using the Fundamental Lemma of SVRG

$$\begin{aligned}&\leq 4L(f(x_k) - f^*) + 2\operatorname{Var}[\nabla f_{I_k}(\tilde{x}_s) - \nabla f_{I_k}(x^*)] \\ &\leq 4L(f(x_k) - f^*) + 2\mathbb{E}\left\|\nabla f_{I_k}(\tilde{x}_s) - \nabla f_{I_k}(x^*)\right\|^2 \\ &\leq 4L(f(x_k) - f^*) + 4L(f(\tilde{x}_s) - f^*)\end{aligned}$$

Now, we bound the distance $\|x_{k+1} - x^*\|$:

$$\begin{aligned}\mathbb{E}\|x_{k+1} - x^*\|^2 &\leq \|x_k - x^*\| - 2\eta\mathbb{E}\langle x_k - x^*, h_k\rangle + \eta^2\mathbb{E}\|h_k\|^2 \\ &= \left\|x_k - x^*\right\|^2 - 2\eta\langle x_k - x^*, \nabla f(x_k)\rangle + \eta^2\mathbb{E}\left\|h_k\right\|^2 \\ &\leq \left\|x_k - x^*\right\|^2 - 2\eta\left(f(x_k) - f(x^*) + \frac{\mu}{2}\left\|x_k - x^*\right\|^2\right) + 4L\eta^2(f(x_k) - f(\tilde{x}_s) - 2f(x^*)) \\ &= \left\|x_k - x^*\right\|^2 - 2\eta(1 - 2L\eta)(f(x_k) - f(x^*)) - \mu\eta\left\|x_k - x^*\right\|^2 + 4L\eta^2(f(\tilde{x}_s - f(x^*)))\end{aligned}$$

Using a telescoping argument, we get

$$\begin{aligned}\mathbb{E}\left\|x_m - x^*\right\|^2 \leq \mathbb{E}\left\|x_0 - x^*\right\|^2 &- 2\eta(1 - 2L\eta)m(\mathbb{E}f(\tilde{x}_{s+1} - f(x^*)) \\ &- \mu\eta m\mathbb{E}\left\|\tilde{x}_{s+1} - x^*\right\|^2 + 4L\eta^2 m(\mathbb{E}f(\tilde{x}_s) - f(x^*))\end{aligned}$$

Dropping the $\mathbb{E}\left\|x_m - x^*\right\|^2$ term and rearranging gives (1). Plugging in $\eta = \frac{1}{8L}$ and $m \geq 2\eta\mu$ gives

$$\frac{3\eta}{2}(\mathbb{E}f(\tilde{x}_{s+1}) - f(x^*)) + \mu\eta\mathbb{E}\left\|\tilde{x}_{s+1} - x^*\right\|^2 \leq \frac{\eta}{2}(\mathbb{E}f(\tilde{x}_s) - f(x^*)) + \frac{\mu\eta}{2}\mathbb{E}\left\|\tilde{x}_s - x^*\right\|^2$$

which implies (2). □

## 3 Additional Notes

## References

[1] S. Bubeck, "Convex Optimization: Algorithms and Complexity," *arXiv:1405.4980 [cs, math, stat]*, Nov. 2015, arXiv: 1405.4980. [Online]. Available: http://arxiv.org/abs/1405.4980

[2] A. B. Tsybakov, "Optimal Rates of Aggregation," in *Learning Theory and Kernel Machines*, G. Goos, J. Hartmanis, J. van Leeuwen, B. Schölkopf, and M. K. Warmuth, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, vol. 2777, pp. 303–313, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-540-45167-9_23

[3] Y. Nesterov, *Lectures on convex optimization*, 2nd ed. Springer, 2018.