IFT 6132: Advanced Strutured Prediction and Optimization **Winter 2019**

Lecture 20 — March 21

Lecturer: Simon Lacoste-Julien *Scribe: Frédéric Boileau, Adel Nabli*

20.1 Motivation

Let's suppose we have a dataset of n observations $(x_i, y_i)_{i \in [\![1,n]\!]} \in \mathcal{X} \times \mathcal{Y}$ and we want to learn a parametrized prediction function $h(x, w)$ with $w \in \mathbb{R}^d$. One way of doing that would be finding $\hat{w}$ solution of

$$\min_{w \in \mathbb{R}^d} f(w) = \min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(w) \tag{20.1}$$

with f some empirical or regularized convex risk function. A wide variety of optimization methods are applicable to solve this problem, but the complexity of second order algorithms being prohibitive for large datasets, we only want to consider the gradient based methods. In this family of algorithms, we can distinguish two main branches: the **deterministic gradient descent** (GD) which requires the computation of n derivatives at each step and the **stochastic gradient descent** (SGD) where only one derivative is needed at each iteration.

But it turns out there is a tradeoff between those two methods: either we have *fast computation but slow convergence* (SGD) or we have *slow computation but fast convergence* (GD). Even if the fast computation of SGD allows us to reach an approximate solution quickly, the convergence slows down eventually.

The question is thus: *Is it possible to think of a new method that combines the best of both worlds ?*

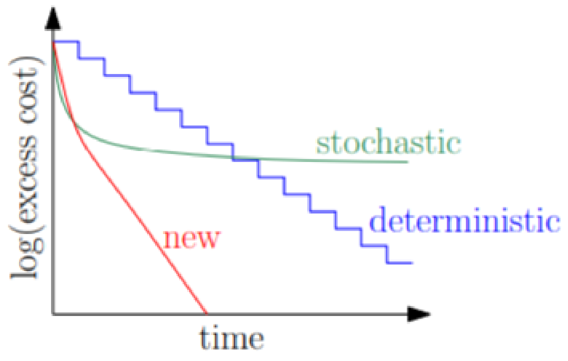


Figure 20.1. The desired convergence *(image taken from [1])*

20.2 Variance Reduced Stochastic Gradient Descent

20.2.1 Main idea

In order to find this new method, we will modify the usual update rule of the SGD. Usually, in the SGD algorithm, we have:

$$w_t = w_{t-1} - \gamma_t \nabla f_{i_t}(w_{t-1}) \tag{20.2}$$

with γ_t the learning rate and $i_t \in [\![1, n]\!]$ the randomly picked index at iteration t. If we consider the random variables $i_t \sim \mathcal{U}([\![1, n]\!])$, then the $X^{(t)}(i_t) \triangleq \nabla f_{i_t}(w_{t-1})$ are also random variables. We then have $\mathbb{E}_{i_t \sim \mathcal{U}}[X^{(t)}] = \frac{1}{n}\sum_i X^{(t)}(i_t = i)$ is the batch gradient in the GD method. We would like to use $\mathbb{E}[X^{(t)}]$ instead of the $X^{(t)}$, the problem being it is expensive to compute. One way to solve that would be to use Monte Carlo samples to estimate $\mathbb{E}[X^{(t)}]$ using random variables $Y^{(t)}$ **highly correlated** with the $X^{(t)}$ but for which the **expectation is cheap to compute**. Let's consider the following parametrized estimator of $\mathbb{E}[X^{(t)}]$:

$$\forall \alpha \in [0, 1],\ \theta_\alpha^{(t)} \triangleq \alpha\big(X^{(t)}(i_t) - Y^{(t)}(i_t)\big) + \mathbb{E}[Y^{(t)}] \tag{20.3}$$

Then, we have the following properties:

$$\mathbb{E}[\theta_\alpha^{(t)}] = \alpha\mathbb{E}[X^{(t)}] + (1-\alpha)\mathbb{E}[Y^{(t)}] \tag{20.4}$$

$$\mathbb{V}[\theta_\alpha^{(t)}] = \alpha^2\Big[\mathbb{V}[X^{(t)}] + \mathbb{V}[Y^{(t)}] - 2Cov(X^{(t)}, Y^{(t)})\Big] \tag{20.5}$$

- From (20.4) we deduce that $\theta_\alpha^{(t)}$ is **unbiased** if $\mathbb{E}[X^{(t)}] = \mathbb{E}[Y^{(t)}]$ *(which is not interesting as we wouldn't need sampling)* or if $\alpha = 1$
- From (20.5) we deduce that $\theta_\alpha^{(t)}$ has **low variance** if α is small or if $Y^{(t)}$ is highly correlated with $X^{(t)}$

Then, in this setting, what remains to be done is **setting** α and **finding good** $Y^{(t)}$ to use $\theta_\alpha^{(t)}$ instead of $X^{(t)}$ in (20.2). In the next subsection, we will present the 3 different choices made for the variance reduced techniques SAG *(Stochastic Average Gradient)*, SAGA *(Stochastic Average Gradient Amélioré)* and SVRG *(Stochastic Variance Reduced Gradient)*.

20.2.2 3 update rules

Let's introduce $g^{(t)}$ the past stored gradients *(in practice it is a table containing the last gradient computed at each training example)*. If we set an other random variable $i_t^{'} \sim \mathcal{U}([\![1, n]\!])$, we can identify $g^{(t)}(i_t^{'}|i_t)$ as the following r.v:

$$g^{(t)}(i_t^{'}|i_t) \triangleq \begin{cases} X^{(t)} = \nabla f_{i_t}(w_{t-1}) & \text{if } i_t^{'} = i_t \\ g^{(t-1)}(i_t^{'}) & \text{otherwise} \end{cases}$$

Then, at time step t, the only element that changes in $g^{(t)}$ compared to $g^{(t-1)}$ is g_{i_t} which is updated. With this definition we have that $g^{(t)}(i_t) = X^{(t)}$ and $\mathbb{E}_{i'_t \sim \mathcal{U}}[g^{(t)}] = \frac{1}{n}\sum_i g^{(t)}(i'_t = i)$.

- **SAG/SAGA** : $Y^{(t)} = g^{(t-1)}$:
  - **SAG**: $\alpha = \frac{1}{n}$. Then, in this setting, combining (20.3) and (20.2) gives:

$$w_t = w_{t-1} - \gamma_t\Big(\overbrace{\frac{1}{n}}^{\alpha}\,\big[\overbrace{\nabla f_{i_t}(w_{t-1})}^{X^{(t)}(i_t)=g_{i_t}^{(t)}} - \overbrace{g_{i_t}^{(t-1)}}^{Y^{(t)}(i_t)}\big] + \overbrace{\frac{1}{n}\sum_j g_j^{(t-1)}}^{\mathbb{E}[Y^{(t)}]}\Big) \tag{20.6}$$

$$w_t = w_{t-1} - \gamma_t\Big(\frac{1}{n}\sum_j g_j^{(t)}\Big) \tag{20.7}$$

  Thus, SAG is **biased** ($\alpha \neq 1$) but its variance goes to 0 as $n \to \infty$ and we have **low variance**.

  - **SAGA**: $\alpha = 1$. Then, the update rule is:

$$w_t = w_{t-1} - \gamma_t\Big(\overbrace{1}^{\alpha} \times \big[\overbrace{\nabla f_{i_t}(w_{t-1})}^{X^{(t)}(i_t)=g_{i_t}^{(t)}} - \overbrace{g_{i_t}^{(t-1)}}^{Y^{(t)}(i_t)}\big] + \overbrace{\frac{1}{n}\sum_j g_j^{(t-1)}}^{\mathbb{E}[Y^{(t)}]}\Big) \tag{20.8}$$

$$w_t = w_{t-1} - \gamma_t\Big(g_{i_t}^{(t)} - g_{i_t}^{(t-1)} + \frac{1}{n}\sum_j g_j^{(t-1)}\Big) \tag{20.9}$$

  Here, we see that SAGA is **unbiased** ($\alpha = 1$), but then the variance is n^2 higher than in SAG. The "variance reduction" only comes from the correlation between $X^{(t)}$ and $Y^{(t)}$

- **SVRG**: Here, we pose $\alpha = 1$ and $Y^{(t)}(i_t) = \nabla f_{i_t}(w_{old})$. The idea is to compute the n gradients every few steps and use them without updating them during T_{max} steps. This procedure then allows us to store in memory only $g_{ref} = \mathbb{E}[Y^{(t)}]$ and w_{old} *(which is significantly less than a table of n gradients)* at the cost of computing at least 2 derivatives at each iteration ($\nabla f_{i_t}(w_{t-1})$ and $\nabla f_{i_t}(w_{old})$).

$$w_t = w_{t-1} - \gamma_t\Big(\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(w_{old}) + \underbrace{\frac{1}{n}\sum_j \nabla f_j(w_{old})}_{g_{ref}}\Big) \tag{20.10}$$

  Then, the method is **unbiased** and the **memory footprint of SVRG is much lower** than for SAG/SAGA. But the computation is a bit heavier in average at each step and the user has to tune T_{max}. In the next subsection, we will discuss a bit the properties of those methods.

Note: *SAG/SAGA can easily be used for generalized linear models where the dependence of the f_i on the x_i is of the form $< w, x_i >$, allowing the table of gradients to only store scalars. But for more complex loss function it may not be suitable (e.g for deep neural networks).*

20.2.3 Properties

Algorithm 1: SVRG algorithm

```
Result: $w^{(m)}$
initialization of $w^{(0)}$;
for $k = 0, ..., m$ do
    $g_{ref} \triangleq \frac{1}{n}\sum_j \nabla f_j(w^{(k)})$;
    for $t = 0, ..., T_{max}$ do
        sample $i_t \sim \mathcal{U}([\![1, n]\!])$;
        $w_t^{(k)} = w_{t-1}^{(k)} - \gamma_t\Big(\nabla f_{i_t}(w_{t-1}^{(k)}) - \nabla f_{i_t}(w^{(k)}) + g_{ref}\Big)$;
    end
    $w^{(k+1)} = w_{T_{max}}^{(k)}$;
end
```

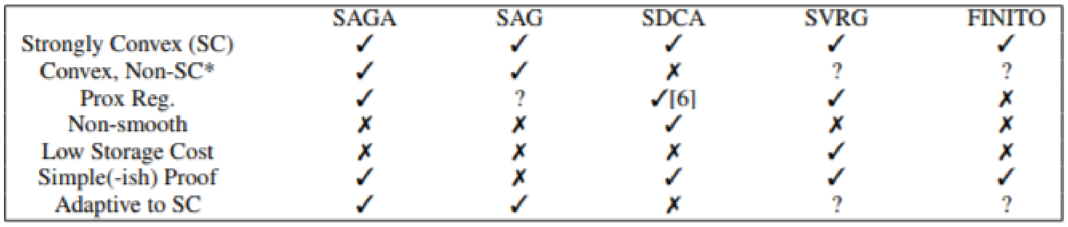
From the SVRG algorithm, two questions main questions arise: *What should be γ_t ?* and *What should be T_{max} ?*

The original SVRG convergence results give:

- $\gamma_t \leq \dfrac{1}{L}$ with L the Lipschitz constant
- $T_{max} \geq \dfrac{L}{\mu} = \kappa$ with κ the condition number

Then, as κ is needed, the method is not adaptive to local strong convexity: there is a need to know the strong convexity properties of the loss before running the method, which is more than the SAG/SAGA methods demand *(they are adaptive)*. But with a tweak on SVRG, better properties fall [3]:

If $T_{max} \sim Geom(\frac{1}{n})$ *(i.e at each inner loop iteration the inner loop is ended with probability $\frac{1}{n}$)*, then the same convergence properties as SAGA fall. Moreover, we have $\mathbb{E}[T_{max}] = n$ and the overall cost of SVRG $\simeq 3\times$ (SGD for each n updates).
With this tweak, then for the three algorithms SAG/SAGA/SVRG the rate of convergence for convex functions ($\mu = 0$) get $\min_t \left[\mathbb{E}f(w_t) - f^*\right] = O(1/t)$ which contrast with the $O(1/\sqrt{t})$ rate for the SGD.



	SAGA	SAG	SDCA	SVRG	FINITO
Strongly Convex (SC)	✓	✓	✓	✓	✓
Convex, Non-SC*	✓	✓	✗	?	?
Prox Reg.	✓	?	✓[6]	✓	✗
Non-smooth	✗	✗	✓	✗	✗
Low Storage Cost	✗	✗	✗	✓	✗
Simple(-ish) Proof	✓	✗	✓	✓	✓
Adaptive to SC	✓	✓	✗	?	?

Figure 20.2. Summary of a few variance reduced optimization methods *(image taken from [2]). This photo is older than the results in [3]*

20.3 Application to CRF

20.4 Proximal gradient method

Bibliography

[1] Francis Bach. Tutorial on stochastic optimization. https://www.di.ens.fr/~fbach/fbach_tutorial_vr_nips_2016.pdf, 2016. [NIPS].

[2] Aaron Defazio, Francis R. Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. *CoRR*, abs/1407.0202, 2014.

[3] Thomas Hofmann, Aurélien Lucchi, Simon Lacoste-Julien, and Brian McWilliams. Variance reduced stochastic gradient descent with neighbors. In *NIPS*, 2015.