Sorbonne Université
Master DAC

Online Convex Optimization

Project Report

Tan Khiem HUYNH

3 janvier 2023

Gradient Descent

Unconstrained Gradient Descent

Performance of Unconstrained Gradient Descent with different values of λ :

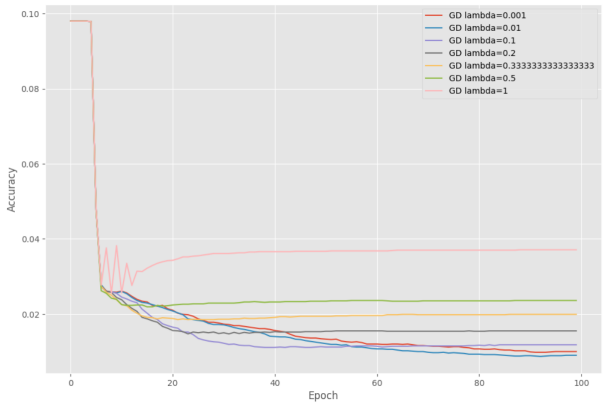

FIGURE 1 – GD's performance with different values of λ

We can see that smaller λ give better convergence rate and final accuracy. The role of λ is to control the compromise between maximizing the margin of the separating plane and the number of misclassified instance. Since in our case the data is not quite well separable, a small λ allow more data points to be misclassified, hence reduce the chance of overfitting on the training set.

Projected Gradient Descent

Performance of Projected Gradient Descent with different values of the l^1 ball radius and $\lambda = \frac{1}{3}$

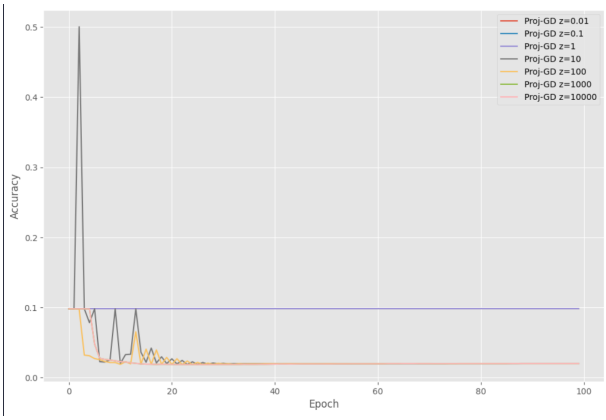

FIGURE 2 – Projected GD's performance with different values of z

We can see that the accuracy of the Projected Gradient Descent algorithm with a z too small is stuck at 0.1% - the accuracy of a random guess. With $z >= 10$, the algorithm is able to get out of the random guess accuracy quicker and give a better final accuracy (≈ 0.02).

Comparison between two versions of gradient descent :

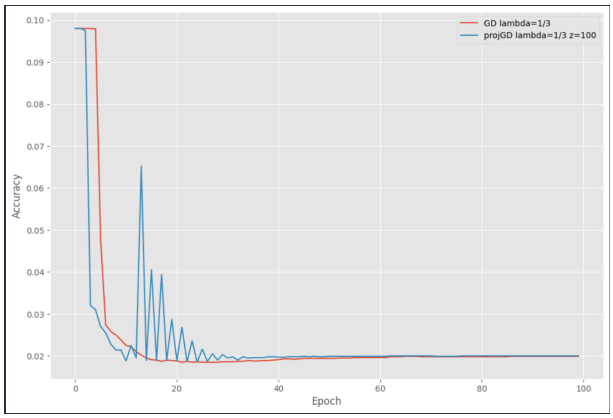

FIGURE 3 – Comparison between two versions of GD

We can see that the projected version takes advantage of the sparsity in the pixel space of hand-writing digit and has a better convergence rate. But its accuracy is unstable, maybe due to overfitting

Stochastic Gradient Descent

We study the performance of two versions of the Stochastic Gradient Descent algorithm with two different learning rate : $\eta_t = \frac{1}{\lambda t}$ and $\eta_t = \frac{1}{\sqrt{t}}$

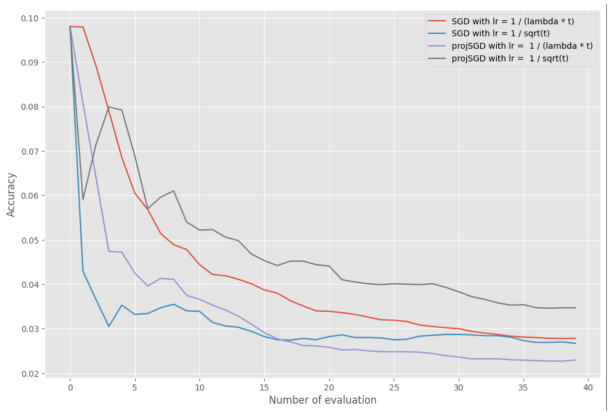

FIGURE 4 – Performance of SGD

The projected version has a slightly better convergence rate than the unconstrained version. The learning rate $\eta_t = \frac{1}{\lambda t}$ has a slower convergence rate, since the learning rate use to establish the theoretical regret bound is $\eta_t = \frac{1}{\sqrt{t}}$, but its final accuracy is better.

About the running time, 10000 epochs of SGD is 3 times faster that 100 epochs of GD.

Regularized Follow the Leader

We study the performance of 3 algorithms : Stochastic Mirror Descent, Stochastic Exponentiated Gradient +/-, Adaptive Gradient and compare them to the Projected SGD.

Four algorithm use a projection onto $\mathcal{B}_1(100)$

The learning rate of SMD and SEG is $\eta_t = \frac{1}{\sqrt{t}}$. The learning rate of Projected SGD is $\eta_t = \frac{1}{\lambda t}$. The learning rate of AdaGrad is 1.

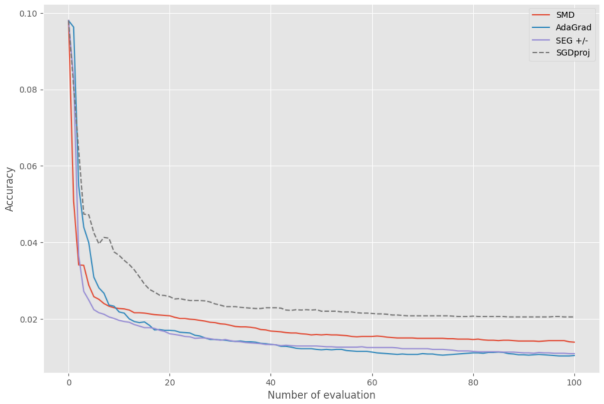

FIGURE 5 – Regularized Follow the Leader

The Regularized Follow the Leader methods take advantage of implicit regularization, hence they have a better accuracy than the Projected SGD, with use a explicit strongly convex regularization. AdaGrad exploits the sparsity in the pixel space in a more efficient way by using an adaptive projection, so its performance is better than the others.

Online Newton Step

We study the performance of the Online Newton Step acceleration algorithm with the regularization parameter $\lambda = \frac{1}{3}$ and $\gamma = \frac{1}{8}$. The projection is done only using the diagonal of A to save time, since a generalized projection using a solver takes too much time with dimension 785.

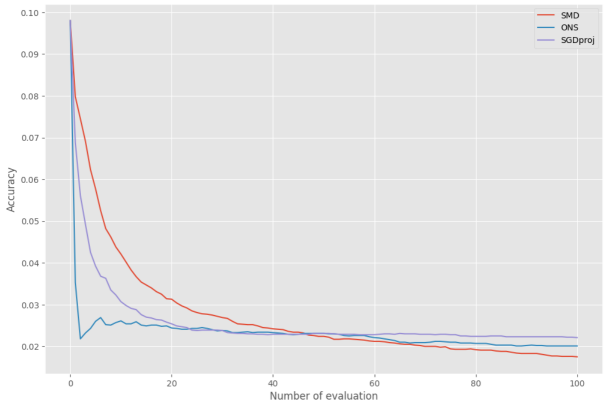

FIGURE 6 – Performance of ONS

We can see that ONS can not gain any significant improvement on the performance in comparison with other methods like Projected SGD or SMD. The running time is much slower than all others algorithms, since ONS require computing and inverting a large matrix A of shape $785x785$.

Exploration Methods

We study the performance of two algorithms : Stochastic Randomized Exponentiated Gradient +/- and Stochastic Bandit Exponentiated Gradient. The learning rate is $\eta_t = \frac{1}{\sqrt{dn}}$ for the two methods.

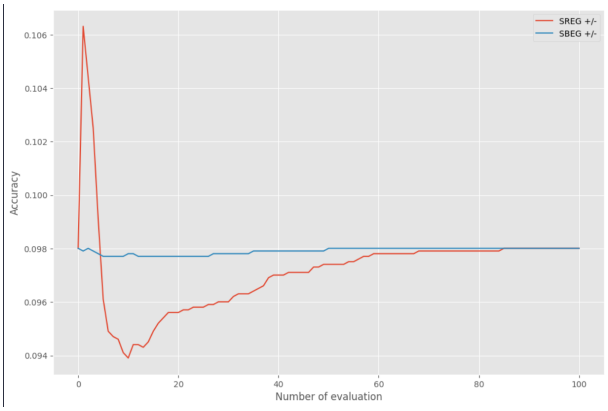

FIGURE 7 – Regularized Follow the Leader

The two algorithms performance is stuck at the random guess accuracy. This is not corresponding to the theoretical regret bound established in the lecture. The algorithms explore too much (for this simple problem) and may not achieve a good compromise between exploration and exploitation.

Proximal-Stochastic Variance Reduction Gradient

Motivation

We consider the problem of minimizing the sum of two convex functions :

$$\underset{x\in\mathbb{R}^d}{minimize} \quad \{P(x) = F(x) + R(x)\} \tag{1}$$

where $F(x)$ is the average of many component functions :

$$F(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) \tag{2}$$

and $R(x)$ is a convex regularized function that may be non differentiable.

In the big data setting, where the number of components n is very large, it can be advantageous to use incremental methods (such as stochastic gradient method) which operate on a single component at each iteration.

A standard method for solving problem 1 is the proximal gradient method :

$$x_k = \underset{x\in\mathbb{R}^d}{argmin} \quad \{\nabla F(x_{k-1})^T x + \frac{1}{2\eta_k}||x - x_{k-1}||^2 + R(x)\}$$

With the definition of proximal mapping :

$$prox_R(y) = \underset{x\in\mathbb{R}^d}{argmin} \quad \{\frac{1}{2}||x - y||^2 + R(x)\}$$

the proximal gradient method can be rewritten as :

$$x_k = prox_{\eta_k R}(x_{k-1} - \eta_k \nabla F(x_{k-1})) \tag{3}$$

3 is refer to as the proximal full gradient (Prox-FG) method, which can be very expensive when n is large. An effective alternative is the proximal stochastic gradient (Prox-SG) method. At each iteration $k = 1, 2, ...$, we draw i_k randomly and take the update :

$$x_k = prox_{\eta_k R}(x_{k-1} - \eta_k \nabla f_{i_k}(x_{k-1})) \tag{4}$$

If the conditions below are satisfied :

- The function R is lower semi-continuous and convex, and its effective domain is closed, and each $f_i(x)$ is differentiable on an open set that contains $dom(R)$, and their gradients are L_i-Lipschitz
- The overall cost function P is μ strongly convex. The strong convexity may come from either $F(x)$ or $R(x)$ or both. More precisely, let $F(x)$ and $R(x)$ be μ_F and μ_R strong convex respectively, then $\mu \geq \mu_F + \mu_R$

then the Prox-FG method with a constant learning rate $\eta = \frac{1}{L}$ generates sequences that satisfy :

$$P(x_k) - P(x^*) \leq O((\frac{L - \mu_F}{L + \mu_R})^k) \tag{5}$$

with $x^* = argmin_x P(x)$

For large-scale problem when $\mu << L$, the ratio $\frac{L}{\mu}$ is often called the condition number of the problem 1. In this case, the Prox-FG method need $O((L/\mu)log(1/\epsilon))$ iterations to ensure $P(x_k) - P(x^*) \leq \epsilon$. Thus the overall complexity in term of the total number of component gradients to find an ϵ-accurate solution is $O(n(L/\mu)log(1/\epsilon))$. Some acceleration method can reduce this complexity to $O(n\sqrt{L/\mu}log(1/\epsilon))$

On the other hand, with a diminishing step size $\eta_k = 1/(\mu k)$, the Prox-SG method have a sublinear convergence rate :

$$P(x_k) - P(x^*) \leq O(\frac{1}{\mu k}) \tag{6}$$

So, the total number of component gradients to find an ϵ-accuracte solution is $O(1/(\mu\epsilon)$.

Both two methods Prox-FG and Prox-SG do not fully exploit the problem structure defined by **??** and **??**. Prox-FG method ignores the fact that $F(x)$ is the average of n component function, while Prox-SG does not exploit the fact that the objective function is a deterministic function, and often leads to high variance and non stable solution.

Several recent work developed algorithm that have the total complexity :

$$O((n + L_{max}/\mu)log(1/\epsilon)) \tag{7}$$

where $L_{max} = max\{L_1, ..., L_n\}$. This complexity is far superior than that of both Prox-FG and Prox-SG introduced above.

The Prox-SVRG method introduced in the paper is an extension of the SVRG method, a method that employs a multi-stage scheme to progressively reduce the variance of the stochastic gradient for the special case $R(x) \equiv 0$. The Prox-SVRG method solve a more general class of problem defined in 1. When the component function is sampled uniformly, Prox-SVRG achieves the same complexity as in 7. When the sampling probabilities for f_i are proportional to their Lipschitz constants L_i, the complexity is :

$$O((n + L_{avg}/\mu)log(1/\epsilon)) \tag{8}$$

where $L_{avg} = \sum_{i=1}^{n} L_i$. This bound is an improvement of 7, especially in the setting where the component functions vary substantially in smoothness.

Algorithm

In Prox-SG method, with uniform sampling of the component function at each iteration, we have unbiased estimation of the full gradient. But in order to ensure asymptotic convergence, the learning rate has to decay to zero to mitigate the effect of high variance introduced by random sampling. This may leads to slow convergence. Prox-SVRG reduces gradually the variance is estimating the full gradient, hence a much bigger (even constant) learning rate is allowed and the algorithm achieves much faster convergence rate. There are some other methods exploit this idea by using mini-batches with exponentially growing sizes, but their overall computational cost is still on the same order as the full gradient method.

The variance reduction technique consists in compute the full gradient periodically. More specifically, an estimation $\tilde{x}$ of the point optimal x^* is maintained and is update periodically, say after every m Prox-SG iterations. Whenever $\tilde{x}$ is updated, the full gradient is also computed :

$$\nabla F(\tilde{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\tilde{x})$$

This full gradient is used to guide the next m stochastic gradient directions. Suppose the next m iterations are initialized with $x_0 = \tilde{x}$ and indexed by $k = 1, 2, ...$ For each $k \geq 1$, a randomly component function f_{i_k} is sampled and we compute :

$$v_k = \nabla f_{i_k}(x_{k-1}) - \nabla f_{i_k}(\tilde{x}) + \nabla F(\tilde{x})$$

Then $\nabla f_{i_k}(x_{k-1})$ in the proximal mapping of Prox-SG method is replaced by v_k :

$$x_k = prox_{\eta_k R}(x_{k-1} - \eta_k v_k)$$

Convergence Analysis

We can prove that just like the stochastic gradient $\nabla f_{i_k}(x_{k-1})$, v_k is an unbiased estimation of ∇F at x_{k-1} :

$$\begin{aligned}\mathbb{E}[v_k] &= \mathbb{E}[\nabla f_{i_k}(x_{k-1})] - \mathbb{E}[\nabla f_{i_k}(\tilde{x})] + \nabla F(\tilde{x}) \\ &= \nabla F(x_{k-1}) - \nabla F(\tilde{x}) + \nabla F(\tilde{x}) \\ &= \nabla F(x_{k-1})\end{aligned}$$

However the variance $\mathbb{E}[||v_k - \nabla F(x_{k-1})||^2]$ can be much smaller than $\mathbb{E}[||\nabla f_{i_k}(x_{k-1}) - \nabla F(x_{k-1})||^2]$. In fact, the variance is bounded :

$$\mathbb{E}[||v_k - \nabla F(x_{k-1})||^2] \leq 4L_{max}[P(x_{k-1}) - P(x^*) + P(\tilde{x}) - P(x^*)] \tag{9}$$

Therefore, when both x_{k-1} and $\tilde{x}$ converge to x^*, the variance of v_k also converges to 0. So we can use a constant learning rate and still obtain much faster convergence.

Let $Q = (q_1, q_2, ..., q_n)$ probability distribution on $(1, 2, ..., n)$. This is the probability distribution used to sample component function at each iteration of stochastic gradient

in the Prox-SVRG method. Then let $L_Q = max_i L_i/(q_i n)$. Assume that $0 < \eta < 1/(4L_Q)$ and m sufficient large so that :

$$\rho = \frac{1}{\mu\eta(1-4L_Q\eta)m} + \frac{4L_Q\eta(m+1)}{(1-4L_Q\eta)m} < 1 \tag{10}$$

Then the Prox-SVRG method has geometric convergence :

$$\mathbb{E}[P(\tilde{x}_s) - P(x^*)] \leq \rho^s [P(\tilde{x}_0) - P(x^*)] \tag{11}$$

10 implies that setting m on the same order as L_Q/μ is sufficient to have geometric convergence.

In order to achieve an ϵ-accurate solution, i.e $\mathbb{E}[P(\tilde{x}_s) - P(x^*)] \leq \epsilon$, the number of effective iteration need to satisfy :

$$s \geq log\rho^{-1} log\frac{P(\tilde{x}_0) - P(x^*)}{\epsilon}$$

Since each effective iteration requires $n + 2m$ component gradient evaluation, and it is sufficient to set $m = \Theta(L_Q/\mu)$, the overall complexity is :

$$O((n + L_Q/\mu)log(1/\epsilon))$$

For uniform sampling, $q_i = 1/n$ for all $i = 1, 2, ..., n$, so $L_Q = max_i L_i$ and the above complexity become 7. The smallest possible value of L_Q is $L_Q = \frac{1}{n}\sum_{i=1}^n L_i$, achieved at $q_i = L_i/\sum_{j=1}^n L_j$, i.e the sampling probabilities for the component functions are proportional to their Lipschitz constants. In this case, the complexity becomes **??**

Numerical experiments

We implements the Prox-SVRG method to solve the MNIST classification problem. To take advantage of the sparsity in the pixel space of hand-writting digits, we choose the regularized function $R(x)$ to be the indicator function of the l^1-ball of radius $z = 100$:

$$R(x) = \begin{cases} 0 & \text{if } x \in \mathcal{B}_1(100) \\ +\infty & \text{otherwise} \end{cases}$$

With this choice of the regularized function, the proximal mapping becomes the projection onto the l^1 ball :

$$x_k = prox_{\eta_k R}(x_{k-1} - \eta_k v_k) = \Pi_{\mathcal{B}_1(100)}(x_k - \eta_k v_k)$$

$F(x)$ in this case is the average of the hinge loss on all over the training set. There is no variance in smoothness between component function so at each iteration, the component function is sampled uniformly.

We run Prox-SVRG for $n = 70$ effective iteration, $m = 2n = 140$ stochastic gradient update at each iteration to have about 10000 component gradient evaluation in total like the others algorithms studied above. The learning rate is 0.01. The figure below show

the performance of Prox-SVRG in comparison with GDproj, SGDproj, SMD, SEG +/-, AdaGrad.

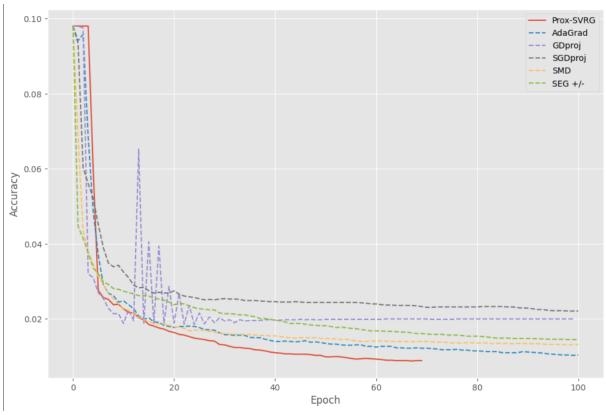

FIGURE 8 – Performance of Prox-SVRG

We can see that Prox-SVRG has the best convergence, even better than AdaGrad.