Non-Asymptotic Analysis of Stochastic Approximation Algorithms for Machine Learning

Laure Ferraris, Éloïse Milin, Grégoire Salha

Sorbonne University, Paris, France

Abstract

Online Convex Optimization is the study of recursive algorithm and their theoretical guarantees called regret bounds. In this document, we recall algorithms seen during the OCO lectures, and we put them in practice, presenting some graphics. Next we describe in a short presentation the main results of [1].

Contents

1 Framework **3**

2 Algorithm seen during OCO lectures **3**
2.1 Gradient Descent 4
2.2 Stochastic Gradient Descent 6
2.3 Regularized Follow the Leader 7
2.4 Online Newton Step 9
2.5 Exploration methods 9

3 Main results of Bach&Moulines **10**
3.1 Problem set-up 10
3.2 Hypothesis 11
3.3 Theorems 12
3.3.1 Stochastic gradient descent 12
3.3.2 Polyak-Ruppert averaging 13

4 Experiments **14**

5 Conclusion **14**

6 Proof **15**

Références **17**

1. Framework

Here, consider a supervised classification problem of two classes $\{+1, -1\}$ and one observes labels $b_i \in \{+1, -1\}$, for $i \in \{1, \dots, n\}$, together with explanatory variables $a_i \in \mathbb{R}^d$. In order to use the following algorithms, it is right to consider the regularized CO problem called the soft-margin problem

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} \ell_{a_i, b_i}(x) + \frac{\lambda}{2} \|x\|^2 \tag{1}$$

where $\lambda > 0$ and $\ell_{a,b}(x) = \text{hinge}(bx^T a) := \max(0, 1 - bx^T a)$. The problem (1) is then a λ-strongly convex CO.

2. Algorithm seen during OCO lectures

In the notebook file, we implemented several algorithms seen during OCO lectures :

- Gradient Descent projected and not projected (GD and GDproj)
- Stochastic Gradient Descent projected and not projected (SGD and SGDproj)
- Stochastic Mirror Descent (SMD)
- Stochastic Exponentiated Gradient $+/-$ (SEGpm)
- Stochastic Adagrad (Adaproj)
- Online Newton Step (ONS)
- Stochastic Randomized Exponentiated Gradient $+/-$ (SREGpm)
- Stochastic Bandit Exponentiated Gradient $+/-$ (SBEGpm)

To put in practice the algorithms above, we use the MNIST dataset (available at this link) which is a handwritten digit database (here, $n = 60000$ and $d = 784$), and we consider two classes : 0 vs other digits ($b_i = 1$ if the digit is 0, else $b_i = -1$).

Next, in order to compare the algorithms, we will display the accuracy for each of them.

2.1. Gradient Descent

Discuss the choice of the hyperparameters λ and z, the parameters of the l^2-regularization and the radius of the ℓ^1-ball, respectively.

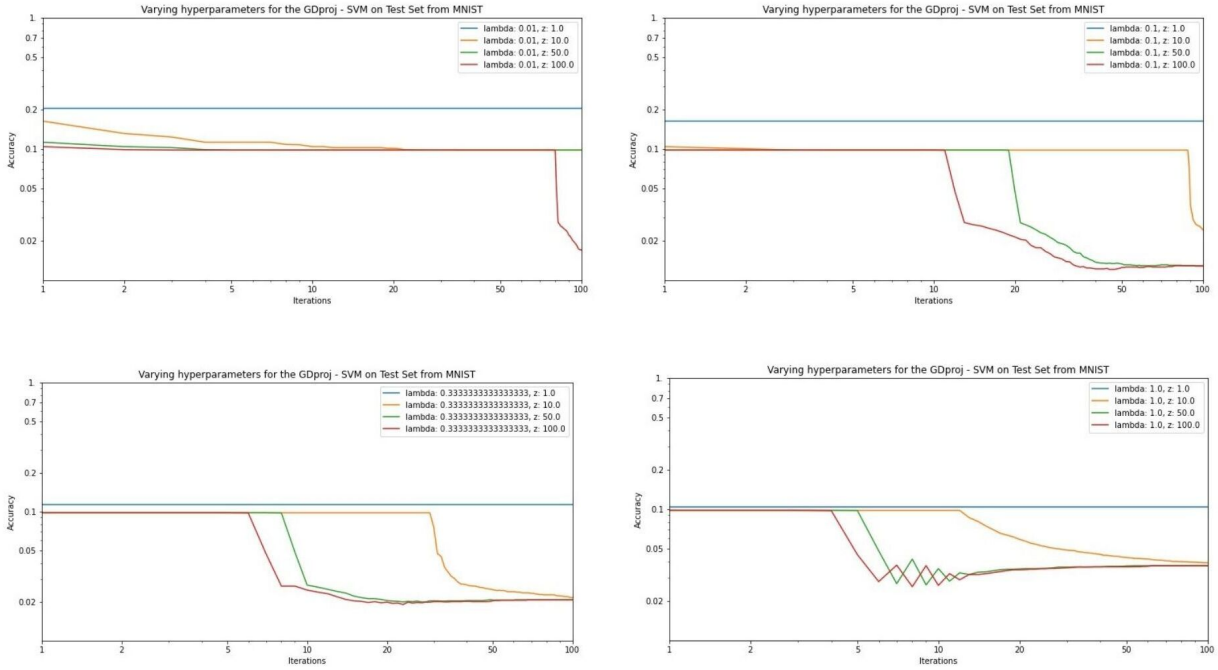

Figure 1: Optimization of the GDproj hyperparameters, λ and z.

We observe that the setting of the parameters is critical for a good classification as for an efficient speed of convergence.

It seems that the **radius z** shouldn't be too small. Increasing **z** from 1 to 100 improves performances of the GDproj algorithm in terms of speed of convergence. This makes us want to observe what happens when $z > 100$. **Lambda** set to 0.1 gives the best accuracy for 100 iterations.

We now explore the algorithm with $z \geq 100$ and $\lambda \leq 0.1$:

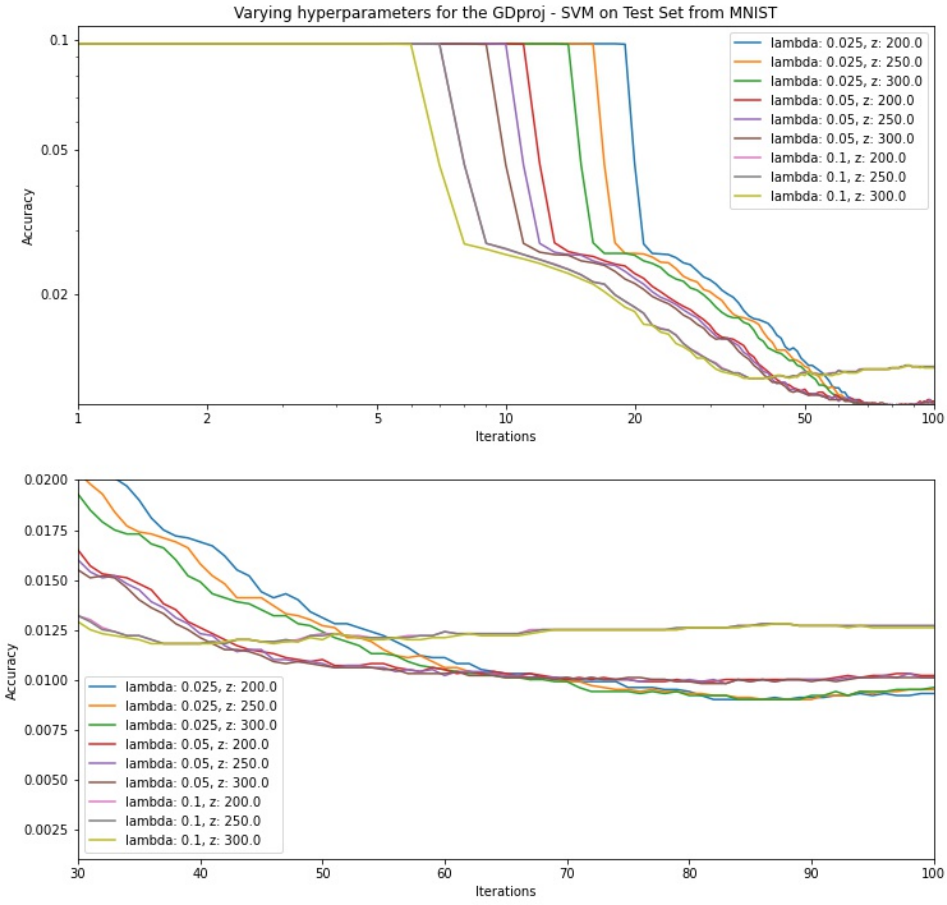

Figure 2: Optimization of the GDproj hyperparameters, λ and z.

Performances are quite similar. We set λ to $1/40$ and z to 300.

2.2. Stochastic Gradient Descent

Compare the accuracy and the running time of the stochastic versus non-stochastic (projected) GD.

We set the parameters as follow :

$$\textbf{GDproj} \,\&\, \textbf{SGDproj} : \text{Epoch } T = 10000,\ \eta_t = \frac{1}{\lambda t},\ \lambda = 1/40,\ z = 300.$$

We recall that $\mathbf{n = 60000}$ is the size of the training set and $\mathbf{d = 784}$ is the dimension of the problem, i.e. the number of features per digit.

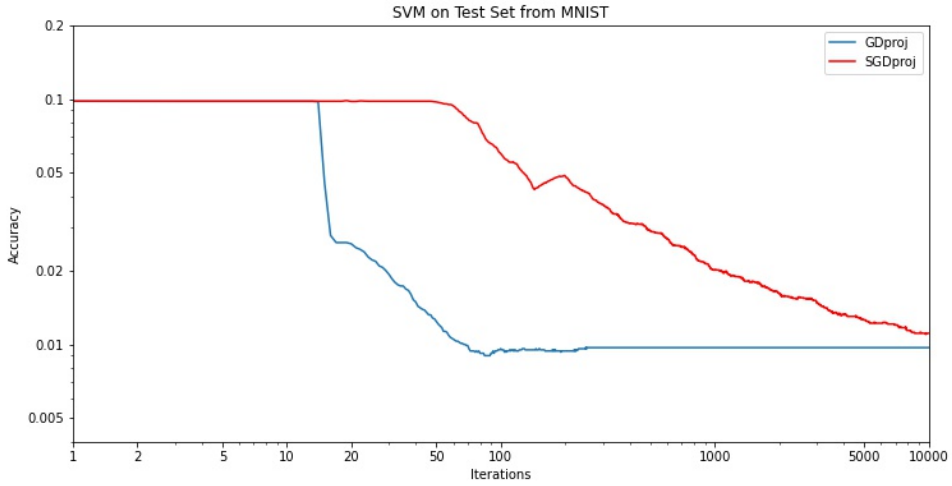

Figure 3: Comparison between GDproj and SDGproj

The **Projected Stochastic Gradient Descent (SGDproj)** algorithm is dramatically faster than the **Projected Gradient Descent (GDproj)**. SGDproj operates 10000 iterations in 5 seconds when GDproj needs 3 hours 5 seconds.

Indeed as studied during the lectures, each iteration of the GDproj costs $\mathcal{O}(nd + P)$ as it needs to compute n gradients of dimension d plus the projection on the ℓ^1-ball of complexity P. Each iteration of the SGDproj computes a single gradient of dimension d plus the projection, then the cost is $\mathcal{O}(d + P)$. From what is specified in the lecture's notes, we can expect a relative speed $\frac{SGDproj}{GDproj}$ of $\frac{1}{1000}$. Here we observe $\frac{1}{2000}$.

After 100 iterations, the accuracy of the GDproj remains constant around 0.01. The SGDproj almost reaches the 0.01 accuracy at 10000 iterations.

2.3. Regularized Follow the Leader

1. Compare the Stochastic Mirror Descent with the projected SGD.

We set the parameters as follow.

$$\textbf{SMDproj} : \text{Epoch} T = 10000,\ \eta = \sqrt{z/T},\ \lambda = 0,\ z = 300.$$

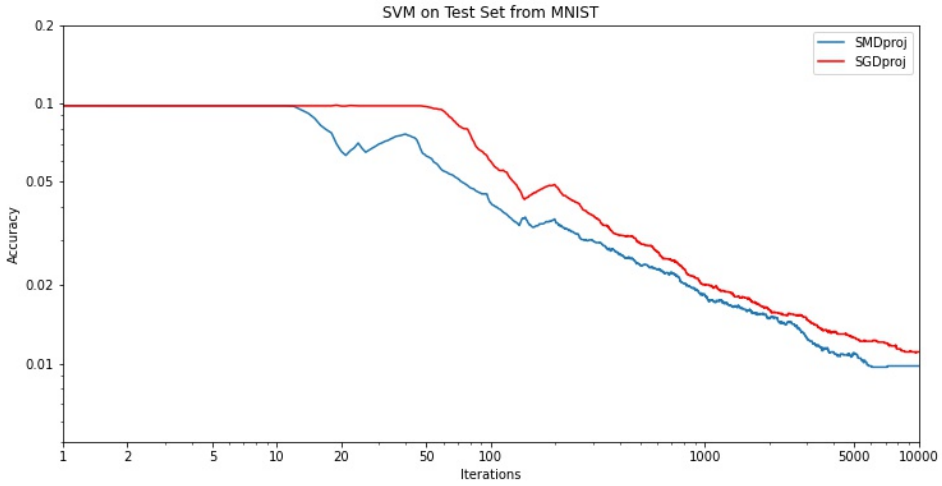

Figure 4: Comparison between SMDproj and SDGproj

We observe similar accuracies despite the slower theoretical rate of convergence of the SMDproj. Indeed, from the OCO lectures we know that the averaging regret bound of a **SGD** algorithm for a λ-strongly convex problem and step size $\eta = 1/\lambda t$ is bounded by a term in the order of $\log T$. Whereas the regret bound of a **SMD** is supposed to be bounded by a term in the order of $\sqrt{T}$ for an optimal constant step size η.

2. Compare the Stochastic Exponentiated Gradient +/- with the projected SGD

We set the parameters as follow.

$$\textbf{SEGpm} : \text{Epoch} T = 10000,\ \eta = \sqrt{2/T},\ \lambda = 1/40,\ z = 300.$$

After 3000 iterations accuracies are quite similar. We remark that if we change the parameters λ and z for **SEGpm** to $\lambda = 1/130$, $z = 800$ it then outperforms both **SMDproj** and **SGDproj**.

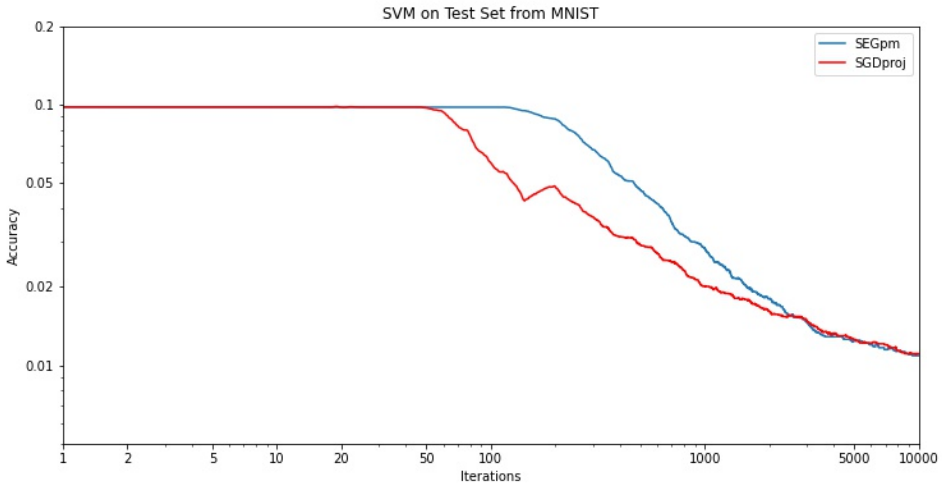

Figure 5: Comparison between SEGpm and SGDproj

3. Compare the Stochastic AdaGrad with the projected SGD. We set the parameters as follow.

$$\textbf{Adaproj} : \text{Epoch} T = 10000,\ \lambda = 1/150,\ z = 110.$$

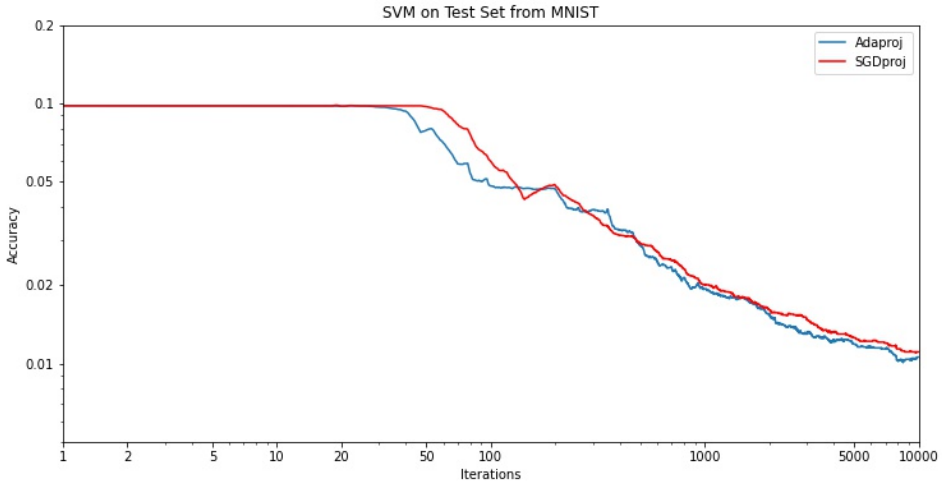

Figure 6: Comparison between Adaproj and SDGproj

At first **SGDproj** gets better performances, but after 500 iterations, **Adaproj** obtains a better accuracy. We need to set different parameters for **Adaproj**, in particular the radius z. **Adaproj** is an adaptive algorithm, here it probably takes advantage because it learns the strong sparsity of the pixel whereas the LASSO constraint (with the parameter z) in **SGDproj** is fixed a priori.

2.4. Online Newton Step

Compare the performances of ONS in terms of accuracy and running time with the other methods.

We set the parameters as follow.

$$\textbf{ONS} : \text{Epoch} T = 10000,\ \gamma = 1/6900, \lambda = 1/50,\ z = 200.$$

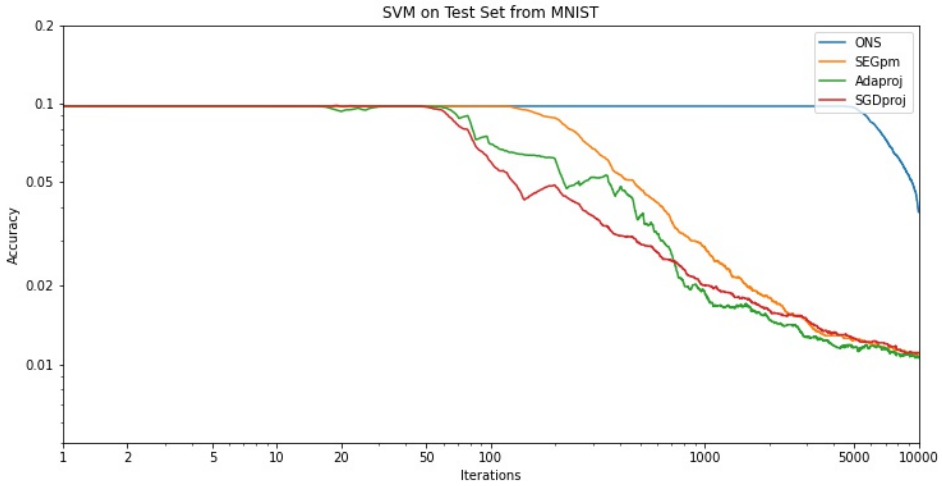

Figure 7: Comparison between ONS and other methods

The **ONS** algorithm is much slower than the others. Its accuracy gets stuck at 0.1 and after 5000 iterations it starts to improve. The parameter gamma is tricky to calibrate, we had to set different z and lambda to obtain an accuracy lower than 0.1.

2.5. Exploration methods

1. Discuss the convergence of the Stochastic Exponentiated Gradient +/- and the Stochastic Bandit Exponentiated Gradient +/-.

We set the parameters as follow.

$$\textbf{SREGpm\&SBEGpm} : \text{Epoch} T = 10000,\ \lambda = 1/40,\ z = 300.$$

Both algorithm do not converge. The accuracies get stuck at 1. We observed that the $(x_t)_{t\geq 1}$ are all equal to $0 \in \mathbb{R}^d$. We think it is probably due to the strong sparsity of the problem. If most of the features are equal to 0, then the exploration is not big enoug even after 10^6 iterations.

2. Is it in accordance with the regret bounds established during OCO lectures?.

We are supposed to observe better performances for the **SBEGpm** algorithm as a result of the OCO lectures assures that we gain a $\log d$ in the averaged learning rate. Here the algorithm doesn't converge and therefore doesn't demonstrate better accuracy than **SREGpm**.

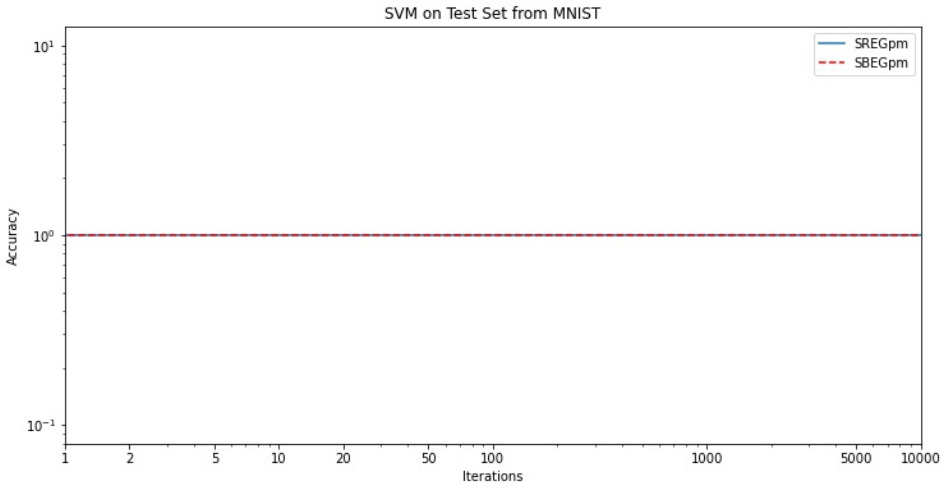

Figure 8: Comparison between SREGpm and SBEGpm

3. Main results of Bach&Moulines

Notations. We consider a Hilbert space $\mathcal{H}$ with a scalar product $\langle \cdot, \cdot \rangle$. We denote by $\|\cdot\|$ the associated norm and use the same notation for the operator norm on bounded linear operators from $\mathcal{H}$ to $\mathcal{H}$, defined as $\|A\| = \sup_{\|x\| \leq 1} \|Ax\|$ (if $\mathcal{H}$ is a Euclidean space, then $\|A\|$ is the largest singular value of A). We also use the notation "w.p.1" to mean "with probability one". We denote by $\mathbb{E}$ the expectation or conditional expectation with respect to underlying probability space.

3.1. Problem set-up

The paper [1] relies on main algorithms in the minimization of an objective function : the stochastic gradient descent (a.k.a. Robbins-Monro algorithm) and a simple modification where iterates are averaged (a.k.a. Polyak-Ruppert averaging).

Assume that we want to minimize f on a convex set $\mathcal{K} \subset \mathcal{H}$. Then the Robbins-Monro algorithm is described below :

Algorithm 1 Stochastic Gradient Descent, Robbins and Monro(1951)

Parameters: Epoch N, step-size $(\gamma_n)_{n \in \{1,\dots,N\}}$
Initialization: Initial point $\theta_1 \in \mathbb{K}$
for $n \in \{1, \dots, N\}$ **do**
  Sample ∇f_n
  Update

$$\theta_n = \theta_{n-1} - \gamma_n \nabla f_n(\theta_{n-1}) \tag{2}$$

end for
Return: θ_N

where $(\gamma_n)_{n\in\{1,\dots,N\}}$ is a deterministic sequence of positive scalars, wich we refer to as the *learning rate sequence*, and $(f_n)_{n\in\{1,\dots,N\}}$ is a sequence of convex differentiable random functions whom its gradients are an unbiased estimate of the gradient of f :

(H1). Let $(\mathcal{F}_n)_{n\geq 0}$ be an incresing family of σ-fields. θ_0 is $\mathcal{F}_0$-measurable, and for each $\theta \in \mathcal{H}$, the random variable $\nabla f_n(\theta)$ is square-integrable, $\mathcal{F}_n$-measurable and

$$\forall \theta \in \mathcal{H}, \forall n \geq 1, \mathbb{E}[\nabla f_n(\theta) \mid \mathcal{F}_{n-1}] = \nabla f(\theta), \text{ w.p.1} \tag{3}$$

Then, the Polyak-Ruppert averaging procedure consists in returning a Cesaro average over the past iterations of the SGD, instead of θ_N :

$$\bar{\theta}_N = \frac{1}{N}\sum_{n=1}^{N} \theta_n$$

This procedure averaging is a way to improve the convergence properties of the original SGD $(\theta_n)_{n\geq 1}$.

We focus here on the learning rate of the form $\gamma_n = Cn^{-\alpha}$, with $C > 0$ and $\alpha \in [0, 1]$. We aim at provide some explicit and non asymptotic regret bounds in order to estimate the efficiency of the algorithm.

3.2. Hypothesis

First, let introduce some hypothesis useful for next.

(H2). For each $n \geq 1$, the function f_n is almost surely convex, differentiable, and:

$$\forall n \geq 1, \forall \theta_1, \theta_2 \in \mathcal{H}, \mathbb{E}[\|\nabla f_n(\theta_1) - \nabla f_n(\theta_2)\|^2 \mid \mathcal{F}_{n-1}] \leq L^2\|\theta_1 - \theta_2\|^2, \text{ w.p.1} \tag{4}$$

(H2'). For each $n \geq 1$, the function f_n is almost surely convex, differentiable with Lipschitz-continuous gradient ∇f_n, with constant L, that is:

$$\forall n \geq 1, \forall \theta_1, \theta_2 \in \mathcal{H}, \|\nabla f_n(\theta_1) - \nabla f_n(\theta_2)\| \leq L\|\theta_1 - \theta_2\|, \text{ w.p.1.} \tag{5}$$

If f_n is twice differentiable, this corresponds to having the norm operator of the Hessian operator of f_n bounded by L.

(H3). The function f is strongly convex with respect to the norm $\|\cdot\|$, with convexity constant $\mu > 0$:

$$\forall \theta_1, \theta_2 \in \mathcal{H}, f(\theta_1) \geq f(\theta_2) + \langle \nabla f(\theta_2), \theta_1 - \theta_2\rangle + \frac{\mu}{2}\|\theta_1 - \theta_2\|^2$$

Note that this last hypothesis simply needs to be satisfied for $\theta_2 = \theta^*$ being the unique global minimizer of f (such that $\nabla f(\theta^*) = 0$). In the context of machine learning, assumption (**H3**) is satisfied as soon as $\frac{\mu}{2}\|\theta\|^2$ is used as an additional regulizer.

(H4). There exists $\sigma^2 \geq 0$ such that $\mathbb{E}[\|\nabla f_n(\theta^*)\|^2 \mid \mathcal{F}_{n-1}] \leq \sigma^2$, w.p.1. for all $n \geq 1$.

(H5). For each $n \geq 1$, almost surely, the function f_n is convex, differentiable and has gradients uniformly bounded by B on the ball of center 0 and radius D, i.e.

$$\forall \theta \in \mathcal{H}, \forall n \geq 1, \|\theta\| \leq D \Rightarrow \|\nabla f_n(\theta)\| \leq B \tag{6}$$

Note that no function may be strongly convex and Lipstchitz-continuous (i.e. with uniformly bounded gradients) over the entire Hilbert space $\mathcal{H}$. Moreover, if (**H2'**) is satisfied, then we may take $D = \|\theta^*\|$ and $B = LD$.

(H8). The function f attains its global minimum at a certain $\theta^* \in \mathcal{H}$ (which may not be unique).

In the machine learning scenario, this essentially implies that the best predictor is in the function class we consider. In the following theorems, since θ^* is not unique, we only derive a bound on function values. Not assuming strong convexity is essential in practice to make sure that algorithms are robust and adaptative to the hardness of the learning or optimization problem (much like gradient descent is).

3.3. Theorems

The results which interest us the most are those whom use Polyak-Ruppert averaging. It is nevertheless attractive to see the same kind of results without PR averaging, to highlight the usefulness of this procedure.

3.3.1. Stochastic gradient descent

Before starting our first theorem, we introduce the following family of functions $\varphi_\beta : \mathbb{R}_+ \backslash \{0\} \to \mathbb{R}$ given by :

$$\varphi_\beta(t) = \begin{cases} \frac{t^\beta - 1}{\beta} & \text{if } \beta \neq 0 \\ \log t & \text{if } \beta = 0 \end{cases}$$

The function $\beta \mapsto \varphi_\beta(t)$ is continuous for all $t > 0$. Moreover, for $\beta > 0$ we have $\varphi_\beta(t) < \frac{t^\beta}{\beta}$, while for $\beta < 0$ we have $\varphi_\beta(t) < -\frac{1}{\beta}$ (both with asymptotic equality when t is large).

Theorem 1 (Stochastic gradient descent, strong convexity)**.** *Assume* (**H1**,**H2**,**H3**,**H4**)*. Denote $\delta_n = \mathbb{E}[\|\theta_n - \theta^*\|^2]$ where $\theta_n \in \mathcal{H}$ is the n-th iterate of the recursion in Eq.(2) with $\gamma_n = Cn^{-\alpha}$. We have, for $\alpha \in [0, 1]$:*

$$\delta_n \leq \begin{cases} 2\exp(4L^2C^2\varphi_{1-2\alpha}(n))\exp\left(-\frac{\mu C}{4}n^{1-\alpha}\right)\left(\delta_0 + \frac{\sigma^2}{L^2}\right) + \frac{4C\sigma^2}{\mu n^\alpha}, & \text{if } 0 \leq \alpha < 1 \\ \frac{\exp(2L^2C^2)}{n^{\mu C}}\left(\delta_0 + \frac{\sigma^2}{L^2}\right) + 2\sigma^2C^2\frac{\varphi_{\mu C/2-1}(n)}{n^{\mu C/2}} & \text{if } \alpha = 1 \end{cases} \tag{7}$$

The following theorem is demonstrated as follow : we first derive a deterministic recursion, which we analyze with novel tools compared to the non-stochastic case (see details in [1]), obtaining new convergence rates for non-averaged stochastic gradient descent:

Theorem 2 (Stochastic gradient descent, no strong convexity)**.** *Assume* (**H1**,**H2'**,**H4**,**H8**)*. Then for* $\alpha \in \left[\frac{1}{2}, 1\right]$*, we have :*

$$\mathbb{E}[f(\theta_n) - f(\theta^*)] \leq \frac{1}{C}\left(\delta_0 + \frac{\sigma^2}{L^2}\right)\exp(4L^2C^2\varphi_{1-2\alpha}(n))\frac{1+4\sqrt{L^3C^3}}{\min\{\varphi_{1-\alpha}(n), \varphi_{\alpha/2}(n)\}} \quad (8)$$

When $\alpha = \frac{1}{2}$, the bound goes to zero only when $LC < \frac{1}{4}$, at rates which can be arbitrarily slow. For $\alpha \in \left(\frac{1}{2}, \frac{2}{3}\right)$, we get convergence at rate $O(n^{-\alpha/2})$, while for $\alpha \in \left(\frac{2}{3}, 1\right)$, we get a convergence rate of $O(n^{\alpha-1})$. For $\alpha = 1$, the upper bound is of order $O((\log n)^{-1})$, which may be very slow (but still convergent). The rate of convergence changes at $\alpha = \frac{2}{3}$, where we get our best rate $O(n^{-1/3})$, which does not match the minimax rate of $O(n^{-1/2})$ for stochastic approximation in the non-strongly convex case. We conjecture that these rates for stochastic gradient descent without strong convexity assumptions are asymptotically minimax optimal (for stochastic gradient descent, not for stochastic approximation). This result could be the subject of another paper.

If we further assume that we have all gradients bounded by B (that is, we assume $D = \infty$ in (H5)), then we have the following theorem, which allow $\alpha \in \left(\frac{1}{3}, \frac{1}{2}\right)$ with rate $O(n^{-(3\alpha-1)/2})$

Theorem 3 (Stochastic gradient descent, no strong convexity, bounded gradient)**.** *Assume* (**H1**,**H2'**,**H5**,**H8**)*. Then, if* $\gamma_n = Cn^{-\alpha}$*, for* $\alpha \in \left[\frac{1}{3}, 1\right]$*, we have :*

$$\mathbb{E}[f(\theta_n) - f(\theta^*)] \leq \begin{cases} (\delta_0 + B^2C^2\varphi_{1-2\alpha}(n))\frac{1+4\sqrt{LC}}{C\min\{\varphi_{1-\alpha}(n),\varphi_{\alpha/2}(n)\}} & \text{if } \alpha \in \left[\frac{1}{2}, 1\right] \\ \frac{2}{C}\sqrt{\delta_0 + B^2C^2}\frac{1+4\sqrt{LB^2C^3}}{\sqrt{1-2\alpha}\,\varphi_{(3\alpha-1)/2}(n)} & \text{if } \alpha \in \left[\frac{1}{3}, \frac{1}{2}\right] \end{cases} \quad (9)$$

3.3.2. Polyak-Ruppert averaging

We now provide two theorems similar to those above, where we replace θ_n by $\bar{\theta}_n$.

Theorem 4 (averaging, no strong convexity)**.** *Assume* (**H1**,**H2'**,**H4**,**H8**)*. Then, if* $\gamma_n = Cn^{-\alpha}$*, for* $\alpha \in \left[\frac{1}{2}, 1\right]$*, we have :*

$$\mathbb{E}[f(\hat{\theta}_n) - f(\theta^*)] \leq \frac{1}{C}\left(\delta_0 + \frac{\sigma^2}{L^2}\right)\left(1 + (2LC)^{1+\frac{1}{\alpha}}\right)\frac{\exp(2L^2C^2\varphi_{1-2\alpha}(n))}{n^{1-\alpha}} + \frac{\sigma^2 C}{2n}\varphi_{1-\alpha}(n) \quad (10)$$

If $\alpha = \frac{1}{2}$, then we only have convergence under $LC < \frac{1}{4}$ (as in Theorem 1), with potentially slow rate, while for $\alpha > \frac{1}{2}$, we have a rate of $O(n^{-\alpha})$. Here, averaging has allowed the rate to go from $O(\max\{n^{\alpha-1}, n^{-\alpha/2}\})$ to $O(n^{-\alpha})$.

Theorem 5 (averaging, no strong convexity)**.** *Assume* (**H1**,**H5**,**H8**)*. Then, if* $\gamma_n = Cn^{-\alpha}$*, for* $\alpha \in [0, 1]$*, we have :*

$$\mathbb{E}[f(\hat{\theta}_n) - f(\theta^*)] \leq \frac{n^{\alpha-1}}{2C}(\delta_0 + C^2B^2\varphi_{1-2\alpha}(n)) + \frac{B^2}{2n}\varphi_{1-\alpha}(n) \quad (11)$$

With the bounded gradient assumption and in fact without smoothness), we obtain the minimax asympotic rate $O(n^{-1/2})$ up to logarithmic terms for $\alpha = \frac{1}{2}$, and for $\alpha < \frac{1}{2}$, the rate is of order $O(n^{-\alpha})$ while for $\alpha > \frac{1}{2}$ we get $O(n^{\alpha-1})$ Here, averaging has also allowed to increase the range of α, which ensures convergence, to $\alpha \in (0, 1)$.

4. Experiments

We apply the results of the article to the MNIST database, that is with $f(x) = \sum_{i=1}^{n} \max(0, 1 - b_i x^T a_i) + \frac{\lambda}{2}\|x\|^2$. This function verifies the conditions (**H2'**) and (**H3**) with L and μ both equals to λ. We do not use Polyak-Ruppert averaging, therefore we apply Theroem 1 here. We applied the Stochastic Gradient Descent here with a step $\gamma_n = Cn^{-\alpha}$ with $\alpha = 1$ and with $C \in \{0.1\lambda, 0.7\lambda, \lambda, 1.5\lambda, 2\lambda, 4\lambda, 10\lambda\}$. We choose $\lambda = 0.33$ and ran the algorithm 20 times with 100 iterations each time for every values of C. We calcuate the loss for every iteration and thus we estimate $\mathbb{E}[f(x)]$ by its empirical mean over 20 values and we are therefore also able to provide a 95% confidence interval. The results are presented below (note that the confidence intervals are distorted because of the logarithmic scale)

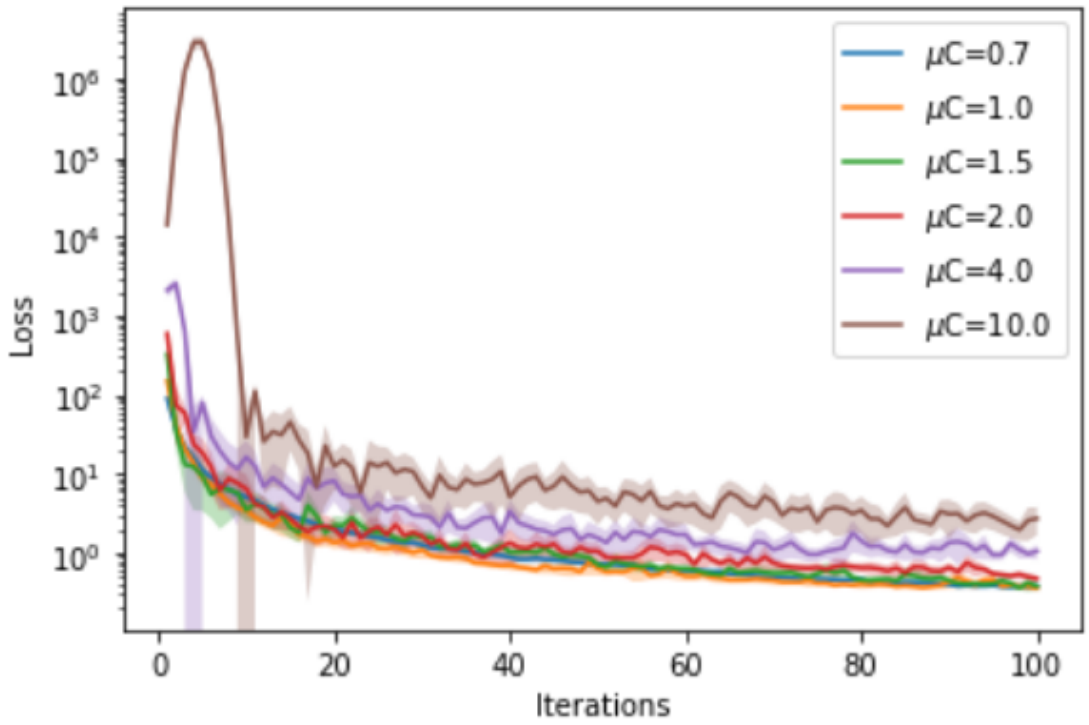

Figure 9: Experiment for different values of μC for $\alpha = 1$

As predicted by the article, we see if we take $\mu C > 2$ (for example $\mu C = 4$ and $\mu C = 10$) there is a catastrophic term which lead to a very high loss for the first iterations, then slowing decaying over time.

5. Conclusion

The paper [1] by Bach and Moulines provides a non-asymptotic analysis for learning rate sequences of a certain form, which fit in both case strong-convexity and no strong-convexity. The results presented are highly usseful in practice.

6. Proof

In this section, we give a proof for theorems 4.

Proof of Theorem 4. We follow the proof of [1], which foolow itself [3] by adapting it to the smooth case. Note first that by convexity of f

$$f(\theta_{k-1}) - f(\theta^*) \leq \langle \nabla f(\theta_{k-1}), \theta_{k-1} - \theta^* \rangle \tag{12}$$

Moreover we have

$$\begin{aligned}\mathbb{E}[\|\theta_k - \theta^*\|^2 \mid \mathcal{F}_{k-1}] &= \mathbb{E}[\|\theta_{k-1} - \theta^* - \gamma_k \nabla f_k(\theta_{k-1})\|^2 \mid \mathcal{F}_{k-1}] \\ &\leq \|\theta_{k-1} - \theta^*\|^2 - 2\gamma_k \langle \nabla f(\theta_{k-1}), \theta_{k-1} - \theta^* \rangle + \gamma_k^2 \mathbb{E}[\|\nabla f_k(\theta)\|^2 \mid \mathcal{F}_{k-1}] \end{aligned} \tag{13}$$

since θ_{k-1} is $\mathcal{F}_{k-1}$-measurable. Next, [4] provide the following inequality (Eq. (2.1.8)) :

$$\frac{1}{L}\|\nabla f_k(\theta_{k-1}) - \nabla f_k(\theta^*)\|^2 \leq \langle \nabla f_k(\theta_{k-1}) - \nabla f_k(\theta^*), \theta_{k-1} - \theta^* \rangle$$

which implies that

$$\|\nabla f_k(\theta_{k-1})\|^2 \leq 2\|\nabla f_k(\theta^*)\|^2 + 2L\langle \nabla f_k(\theta_{k-1}) - \nabla f_k(\theta^*), \theta_{k-1} - \theta^* \rangle$$

Plugging this inequality in Eq.(13), then implies

$$\delta_k \leq \delta_{k-1} - 2\gamma_k(1 - \gamma_k L)\mathbb{E}[\langle \nabla f(\theta_{k-1}), \theta_{k-1} - \theta^* \rangle] + 2\gamma_k^2 \sigma^2$$

where $\delta_k = \mathbb{E}[\|\theta_k - \theta^*\|^2]$, showing that

$$2\gamma_k(1 - \gamma_k L)\mathbb{E}[\langle \nabla f(\theta_{k-1}), \theta_{k-1} - \theta^* \rangle] \leq \delta_{k-1} - \delta_k + 2\gamma_k^2 \sigma^2 \tag{14}$$

Define $n_0 = \inf\left\{k \in \mathbb{N} \mid (1 - \gamma_k L) \geq \frac{1}{2}\right\}$. For any $k \geq n_0$, we have $(1 - \gamma_k L) \leq \frac{1}{2}$ and therefore,

$$\mathbb{E}[\langle \nabla f(\theta_{k-1}), \theta_{k-1} - \theta^* \rangle] \leq gamma_k^{-1}(\delta_{k-1} - \delta_k + 2\gamma_k^2 \sigma^2)$$

Note that, by integrating by parts,

$$\sum_{k=n_0+1}^{n} \gamma_k^{-1}(\delta_{k-1} - \delta_k) = \gamma_{n0+1}^{-1}\delta_{n_0} + \sum_{k=n_0+1}^{n-1} \delta_k(\gamma_{k+1}^{-1} - \gamma_k^{-1}) - \gamma_n^{-1}\delta_n$$

We define $D_n = \exp\left(2L^2 \sum_{k=1}^{n} \gamma_k^2\right)\left(\delta_0 + \frac{\sigma^2}{L^2}\right)$. Then $\delta_n \leq D_n$ and $(D_n)_{n \geq 1}$ is non-decreasing, and we have :

$$\sum_{k=n_0+1}^{n} \gamma_k^{-1}(\delta_{k-1} - \delta_k) \leq D_n \gamma_n^{-1} \tag{15}$$

Combining Eq.(14) and Eq.(15) shows that, for $k \in \{1, \ldots, n_0\}$, under (**H2'**),

$$\|f(\theta_k) - f(\theta^*)\| \leq \left| \int_0^1 \langle \nabla f(\theta^* + t(\theta_k - \theta^*)) - \nabla f(\theta^*), \theta_k - \theta^* \rangle \, \mathrm{d}t \right| \leq \frac{L}{2}\|\theta_k - \theta^*\|^2$$

And finally, using the convexity and the Lipschitz-continuity of f,

$$\begin{aligned}\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] &= \mathbb{E}\left[f\left(\frac{1}{n}\sum_{k=0}^{n}\theta_k\right)\right] - f(\theta^*) \\ &\leq \frac{1}{n}\sum_{k=0}^{n}(f(\theta_k) - f(\theta^*)) \\ &\leq \frac{1}{n}\left(\frac{D_n}{\gamma_n} + \frac{\sigma^2}{2}\sum_{k=n_0+1}^{n}\gamma_k + \frac{L}{2}\sum_{k=1}^{n_0}D_k\right)\end{aligned}$$

Case $\alpha = \frac{1}{2}$

In this case, for $L^2C^2 < \frac{1}{4}$, we have $D_n = \left(\delta_0 + \frac{\sigma^2}{L^2}\right) n^{2L^2C^2}$. This leads to an upper bound of the form

$$\frac{1}{n^{1/2-2L^2C^2}}\frac{1}{C}\left(\delta_0 + \frac{\sigma^2}{L^2}\right) + \frac{C\sigma^2}{n}\left(n^{-1/2} - n_0^{-1/2}\right) + \frac{L}{2n}\left(\delta_0 + \frac{\sigma^2}{L^2}\right)\frac{1}{2L^2C^2+1}n_0^{2L^2C^2+1}$$

leading to, for $n = (2LC)^2$,

$$\frac{1}{n^{1/2-2L^2C^2}}\frac{1}{C}\left(\delta_0 + \frac{\sigma^2}{L^2}\right) + \frac{\sigma^2}{n^{1/2}} + \frac{L}{2}\left(\delta_0 + \frac{\sigma^2}{L^2}\right)\frac{1}{n}\frac{1}{2L^2C^2+1}(4L^2C^2)^{2L^2C^2+1}$$

Case $\alpha \in \left(\frac{1}{2}, 1\right)$

In this case, we have $D_n = \left(\delta_0 + \frac{\sigma^2}{L^2}\right)\exp\left(\frac{2L^2C^2}{2\alpha-1}\right)$, leading to the upper bound

$$\left(\delta_0 + \frac{\sigma^2}{L^2}\right)\exp\left(\frac{2L^2C^2}{2\alpha-1}\right)\frac{1}{C}n^{\alpha-1} + \frac{C\sigma^2}{n}\varphi_{1-\alpha}(n) + \frac{L}{2}\left(\delta_0 + \frac{\sigma^2}{L^2}\right)\exp\left(\frac{2L^2C^2}{2\alpha-1}\right)\frac{(2LC)^{1/\alpha}}{n}$$

Case $\alpha = 1$

In this case, we have $D_n = \left(\delta_0 + \frac{\sigma^2}{L^2}\right)\exp\left(\frac{L^2C^2\pi^2}{6}\right)$, leading to the upper bound

$$\left(\delta_0 + \frac{\sigma^2}{L^2}\right)\exp\left(\frac{L^2C^2\pi^2}{6}\right)\frac{1}{C} + \frac{C\sigma^2}{n}\frac{\ln n}{n} + \frac{L}{2}\left(\delta_0 + \frac{\sigma^2}{L^2}\right)\exp\left(\frac{L^2C^2\pi^2}{6}\right)$$

□

References

[1] F.Bach and E.Moulines, *Non-Asymptotic Analysis of Stochastic Approximation Algorithms for Machine Learning*, 2011, available at this link.

[2] S.Gadat and F.Panloup, *Optimal non-asymptotic analysis of the Ruppert-Polyak averaging stochastic algorithm*, 2012, available at this link.

[3] Y.Nesterov and J.P.Vial, *Confidence level solutions for stochastic programming*, 2000, available at this link.

[4] Y.Nesterov, *Introductory lectures on convex optimization: a basic course*, Kluwic Academic, Publishers, 2004.