FTML practical session 10: 2024/06/21

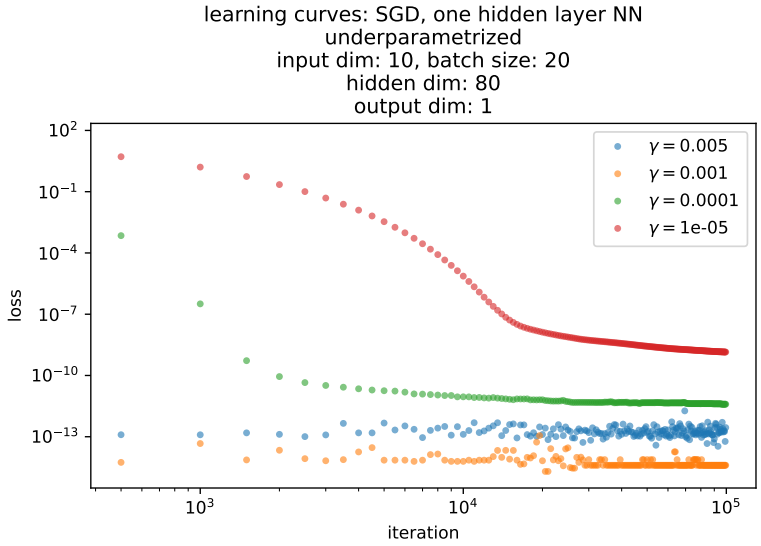


TABLE DES MATIÈRES

1 THE HEAVY-BALL METHOD

1.1 Convergence rates of GD for convex functions

We consider the optimization of a convex function $f : \theta \to f(\theta)$ using a gradient descent (GD). In particular, we consider the **convergence speed** of GD. This speed can be expressed in several manners. For instance, as the distance between the iterate θ_t and a minimizer θ^* (of course assuming that this minimizer exists and is unique), as a function of the iteration number t. It if possible to show the following results for two-times differentiable convex functions :

— if H is invertible ($\mu > 0$), we have a convergence rate in $\exp(-\frac{2t}{\kappa})$.

— if H is not invertible ($\mu = 0$), we have a convergence rate in $\mathcal{O}(\frac{1}{t})$ (probably one of the exercises of the project).

These rates are speed **upper bounds**, meaning that the convergence is at least as fast as those. Also note that these rates of convergence require the use of specific values of the learning rate γ, like for instance $\frac{1}{L}$ (but other values might also be used, depending on the context). We also see that these rates depend on the **condition number** of the Hessian H. If we note μ the smallest eigenvalue of the Hessian H, and L the largest, and if this Hessian is for instance symetric and definite positive, then

$$\kappa = \frac{L}{\mu} \tag{1}$$

However, the condition number might be defined also for general matrices, and even functions.

https://en.wikipedia.org/wiki/Condition_number

1.2 Large condition numbers

Hence, when κ is very large ($>> 1$), the convergence to the optimum might be very slow. Note that matrices with large condition numbers are not rare in large-scale machine learning and scientific computing applications. If the smallest eigen value μ of a given matrix H is very small, or even 0 (which will happen as soon as H is not full rank), κ will be very large as soon as the largest eigenvalue L is not also very small.

1.3 Inertial methods

When κ is large, some methods still exist in order to speed the convergence of gradient descent, such as **Heavy-ball.** This method consists in adding a **momentum term** to the gradient update term, such as the iteration now writes

$$\theta_{t+1} = \theta_t - \gamma \nabla_\theta f(\theta_t) + \beta(\theta_t - \theta_{t-1}) \tag{2}$$

where β and γ are real constants that should be tuned. The update $\theta_{t+1} - \theta_t$ is then a combination of the gradient $\nabla_\theta f(\theta_t)$ and of the previous update $\theta_t - \theta_{t-1}$. The goal of this method it to balance the effet of oscillations in the gradient. The heavy-ball method is called an **inertial method.** When f is a general convex function (not necessary quadratic), some generalizations exist, such as **Nesterov acceleration.** Many of the most famous variations of SGD, like RMSProp and Adam, optionally include such a momentum term.

1.4 Impact on convergence rate for a least squares problem

Assuming $\mu > 0$, in a least squares problem, it is possible to show that the characteristic convergence time with the heavy-ball momentum term is $\sqrt{\kappa}$ instead of κ, if β and γ are tuned well. Formally, with the heavy-ball momentum term, we changed the convergence (upper bound) from $\mathcal{O}(\exp(-\frac{2t}{\kappa}))$ to $\mathcal{O}(\exp(-\frac{2t}{\sqrt{\kappa}}))$. If κ is large, which is the case we are interested in, this can be a significant improvement.

You can try to proove this results following the steps presented in **Heavy_Ball_Exercise.pdf** or read the proof in **Heavy_Ball°solution.pdf**.

1.5 Simulation

In **heavy_ball/**, use the file **heavy_ball.py** to implement the Heavy-ball method and compare the convergence speed to that of GD. You will need to experiment with γ and β, and might obtain results like figures 1 and 2.

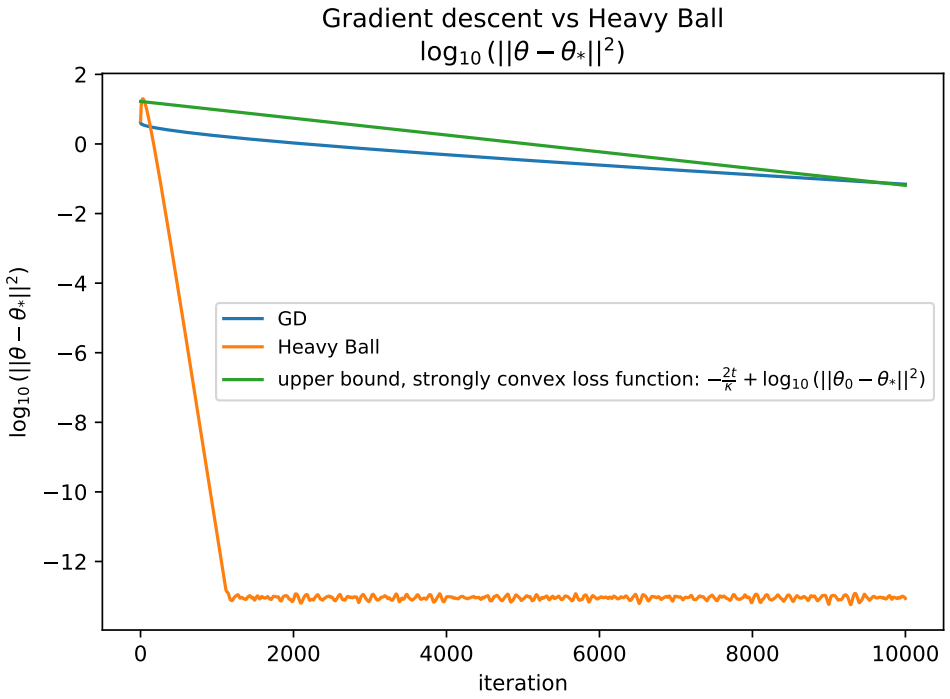


FIGURE 1 – Heavy ball vs GD, semilog scale

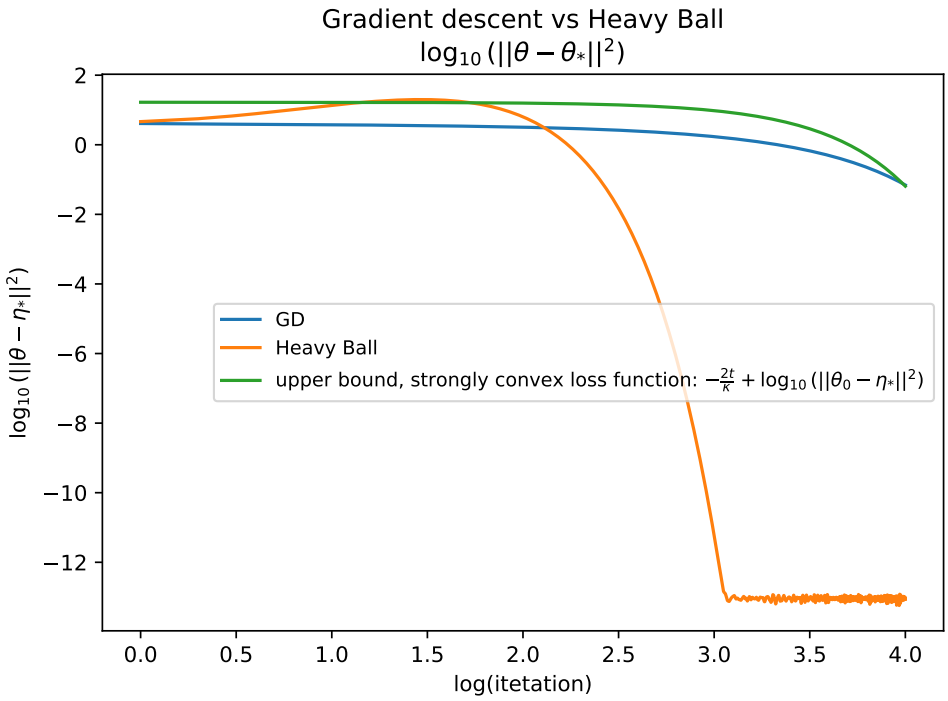


FIGURE 2 – Heavy ball vs GD, logarithmic scale

2 STOCHASTIC AVERAGE GRADIENT (SAG)

SAG is a modern variant of SGD, implemented in some commonly used solvers, for instance scikit's built in GD algorithms. .

— Read the introduction (or more) of the research article placed in the TP16 folder, entitled "Minimizing finite sums with the stochastic average gradient" [Schmidt et al., 2013].

— Optional / suggestion of an exercise that you can explore later : implement a comparison of SAG, SGD and GD on a problem of your choice (for instance a dataset mentioned in the article, but you can also use a simpler dataset.).

It is also interesting to have a look at the **_sag.py** file trom scikit-learn in the **linear_model** module.

`https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/linear_model/_sag.py`

3 ACCELERATION OF SEQUENCES

In this exercice we present two methods that accelerate the convergence of some iterative algorithms to their limit. These methods combine iterates of the algorithms to produce a another sequence in parallel, that sometimes converges faster to the limit. Although rare, methods of this kind sometimes have some applications in machine learning.

3.1 Aitken's Δ^2 process

3.1.1 Presentation

Aitken's process is one of the simplest of these methods. We consider a sequence $(x_k)_{k\in\mathbb{N}} \in \mathbb{R}^{\mathbb{N}}$. The idea is to locally model the sequence as a first-order auto-regressive sequence, which means finding, for each $k \in \mathbb{R}$, two numbers a_k and b_k, such that :

$$\begin{cases} x_{k+1} = a_k x_k + b_k \\ x_{k+2} = a_k x_{k+1} + b_k \end{cases}$$

and then compute the limit of the sequence $(y_i^k)_{i\in\mathbb{N}}$ defined by

$$y_{i+1}^k = a_k y_i^k + b_k \tag{3}$$

we note l_k the limit of $(y_i^k)_{i\in\mathbb{N}}$. This defines another sequence $(l_k)_{k\in\mathbb{N}}$, that might converge faster to the limit of $(x_k)_{k\in\mathbb{N}}$, if this limit exists.

Note that in order to observe an acceleration, this model does not need to be exact, it can also be true only asymptotically.

3.1.2 Equations

Assuming that if iterate is different $x_k \neq x_{k+1}$, solve the linear system defining the locally auto-regressive process for a given k (which means find a_k and b_k)

For a given k, what would be a sufficient condition that ensures that the sequence $(y_i^k)_{i\in}$ has a limit ?

3.1.3 Simulation

We will apply Aitken's method to the Leibniz formula, a method to approximate π as the limit of the following sequence :

$$x_k = 4\sum_{j=0}^{k} \frac{(-1)^j}{2j+1} \tag{4}$$

this is one of the most famous applications of the method.

Is the condition from question 2 verified ?

Run a simulation that applies the method to the sequence $(x_k)_{k\in\mathbb{N}}$. You should observe something like figure 3, i.e. an acceleration of the convergence to the limit.

3.2 Richardson extrapolation (optional)

3.2.1 Presentation

We consider $(x_k)_{k\in\mathbb{N}}$ of points of $\mathbb{R}^d$ and we assume that

$$x_k = x_* + \frac{1}{k}\Delta + \mathcal{O}(\frac{1}{k^2}) \tag{5}$$

With $\Delta \in \mathbb{R}^d$. Hence, $(x_k)_{k\in\mathbb{N}} \to x_*$.

Show that for k even :

$$2x_k - x_{k/2} = x_* + \mathcal{O}(\frac{1}{k^2}) \tag{6}$$

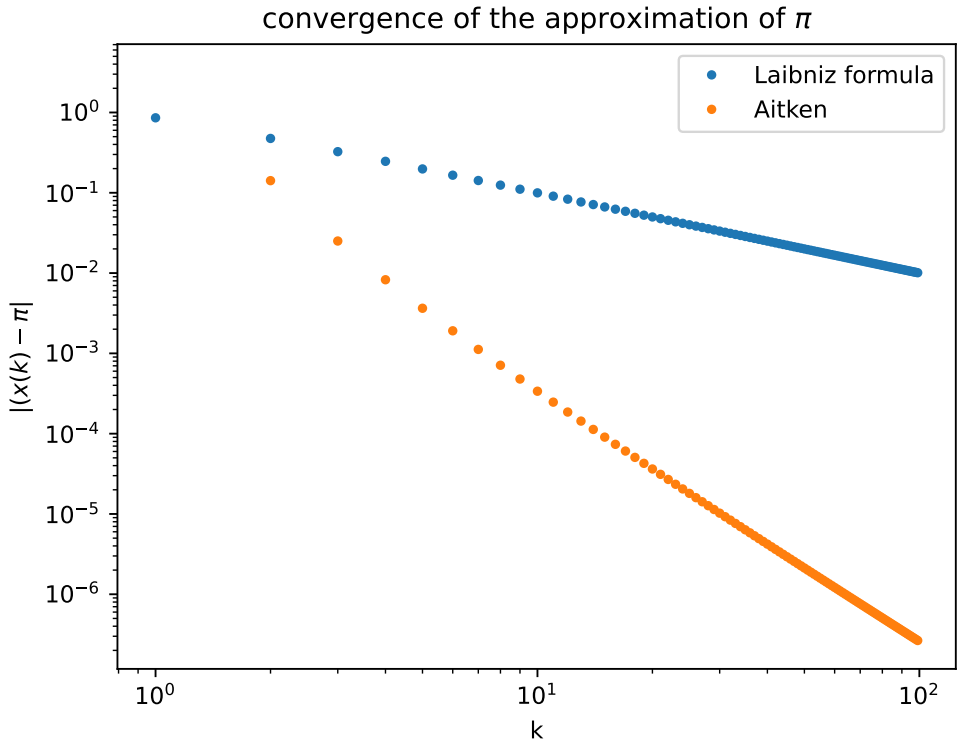


FIGURE 3 – Aitken's Δ^2 process accelerates the convergence to the π.

Hence, the sequence defined by $y_k = 2x_k - x_{k/2}$ might converge faster to the limit x_*. This method is called Richardson extrapolation.

Can Richardson extrapolation have a strong negative impact? what is the worst-case loss of performance?

3.2.2 *Application to logistic regression*

When performing gradient descent, for instance on logistic regression, it is possible to average the obtained iterates $x_j \in \mathbb{R}^d$.

$$z_k = \frac{1}{k} \sum_{j=0}^{k-1} x_j \tag{7}$$

This provides robustness to noise, however the initial conditions are not forgotten very fast. A workaround is **tail-averaging**, which means only taking the last half of the iterates. If k is even, we then define a new sequence t_k.

$$t_k = \frac{1}{k/2} \sum_{j=k/1}^{k-1} x_j \tag{8}$$

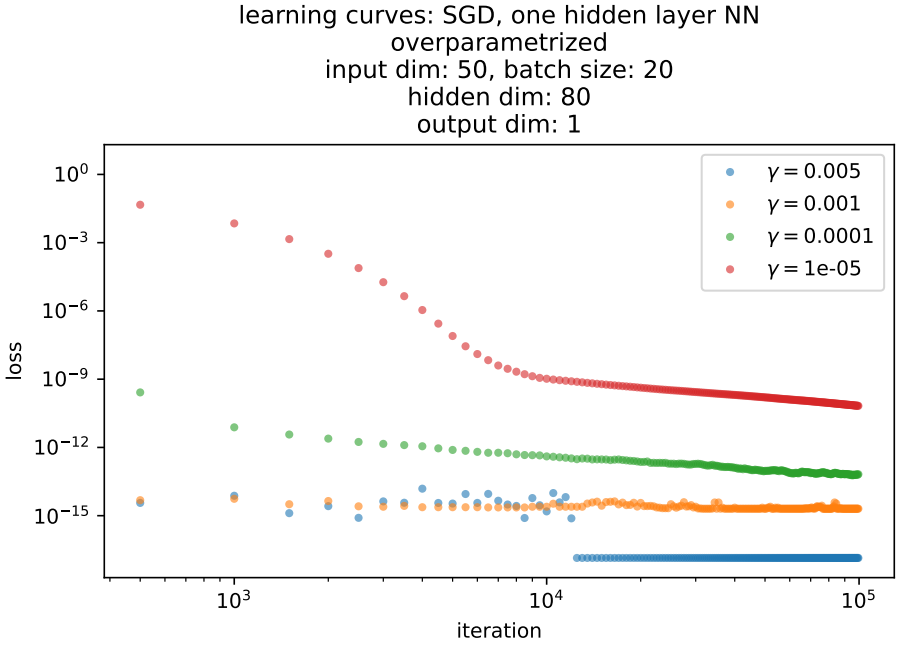
What is the link with Richardson extrapolation?

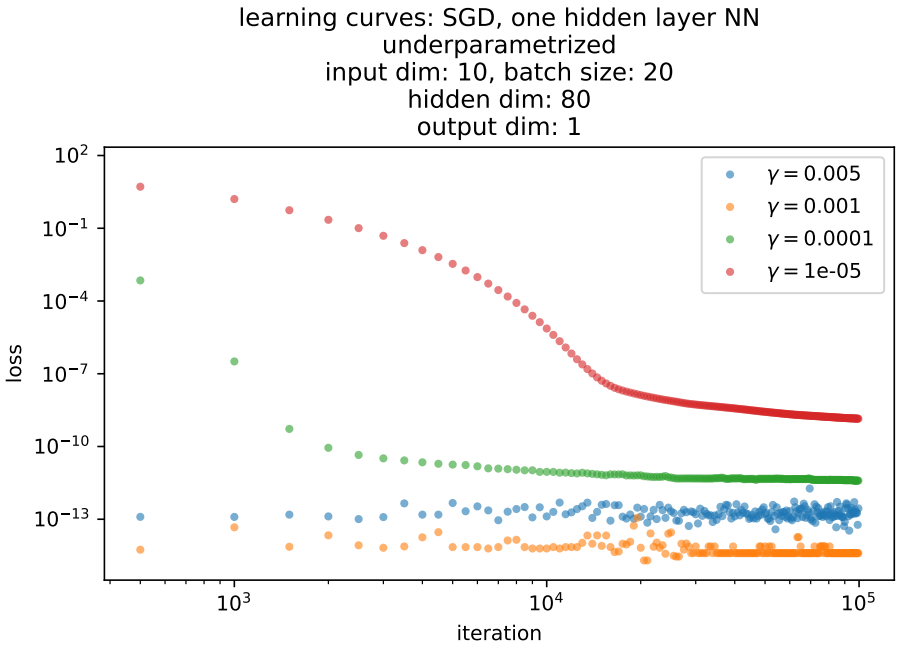
In some contexts, like logistic regression, as explained in [Bach, 2021], it is possible to show that the sequence z verifies 5.

In this blog post, you can find more comments and ressources about these axxeleration methods :

https://francisbach.com/richardson-extrapolation/

4 OVERPARAMETRIZED AND UNDERPARAMETRIZED REGIMES





RÉFÉRENCES

[Bach, 2021] Bach, F. (2021). On the Effectiveness of Richardson Extrapolation in Data Science. SIAM Journal on Mathematics of Data Science, 3(4) :1251–1277.

[Schmidt et al., 2013] Schmidt, M., Le Roux, N., and Bach, F. (2013). Minimizing finite sums with the stochastic average gradient. Mathematical Programming,

162(1-2) :83–112.