FTML practical session 7: 2023/04/21

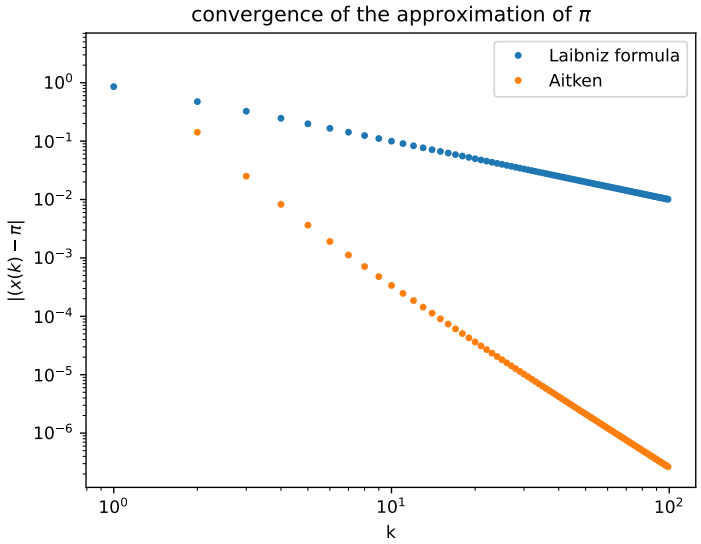

TABLE DES MATIÈRES

INTRODUCTION

In this session we study some more notions related to convergence speed of algorithms that are useful in machine learning, namely acceleration and lower bounds on performance.

1 ACCELERATION OF SEQUENCES

In this briefly we shortly 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.

1.1 Aitken's Δ^2 process

1.1.1 *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

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

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.

1.1.2 *Equations*

Exercice 1 : 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)

Exercice 2 : For a given k, what would be a sufficient condition that ensures that the sequence $(y_i^k)_{i\in}$ has a limit ?

1.1.3 *Simulation*

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

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

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

Exercice 3 : Is the condition from question 2 verified ?

Exercice 4 : Run a simulation that applies the method to the sequence $(x_k)_{k\in\mathbb{N}}$. You should observe something like figure 1, i.e. an acceleration of the convergence to the limit.

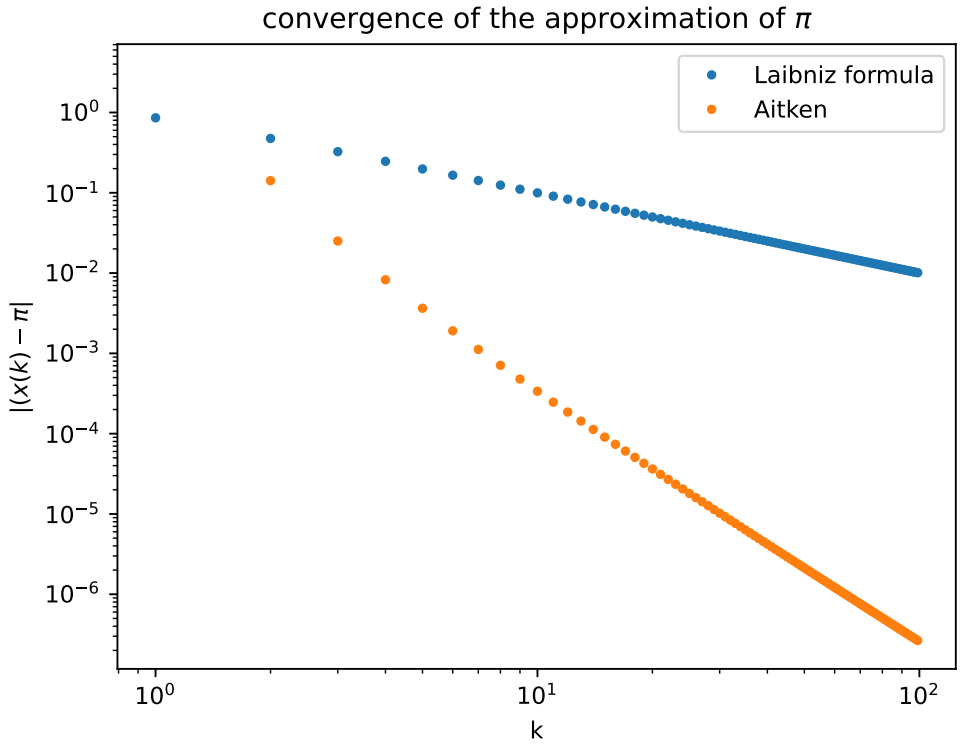

FIGURE 1 – Aitken's Δ^2 process accelerates the convergence to the π.

1.2 Richardson extrapolation (optional)

This section might be skipped during the session.

1.2.1 *Presentation*

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

$$x_k = x_* + \frac{1}{k}\Delta + \mathcal{O}(\frac{1}{k^2}) \tag{3}$$

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

Exercice 5 : Show that for k even :

$$2x_k - x_{k/2} = x_* + \mathcal{O}(\frac{1}{k^2}) \tag{4}$$

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.

Exercice 6 : Can Richardson extrapolation have a strong negative impact ? what is the worst-case loss of performance ?

1.2.2 *Application to logistic regression*

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

$$z_k = \frac{1}{k}\sum_{j=0}^{k-1} x_j \tag{5}$$

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.

$$t_k = \frac{1}{k/2} \sum_{j=k/1}^{k-1} x_j \tag{6}$$

Exercice 7 : What is the link with Richardson extrapolation ?

Exercice 8 : In some contexts, like logistic regression, as explained in [**?**] , it is possible to show that the sequence z verifies 3.

2 THE HEAVY-BALL METHOD

During the class, we have seen that when optimizing a convex function (such a the empirical risk in a least-squares problem or a logistic regression), the convergence might become very slow when the condition number κ of the Hessian is very large. Some methods exist in order to speed it up, such as **Heavy-ball.** This method consists in adding a **momentum term** to the gradient update term, such as the iteration now writes

$$\theta_{t+1} = \theta_t - \gamma \nabla_\theta f(\theta_t) + \beta(\theta_t - \theta_{t-1}) \tag{7}$$

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 might 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.**

2.1 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. 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 great improvement.

Remember that $\kappa = \frac{L}{\mu}$. Often, μ will be very small when n or d is large. For instance, in the case of Ridge regression, we have seen in a previous session that for instance, μ can be of order $\mathcal{O}(\frac{1}{\sqrt{n}})$ (see the computation of the optimal regularisation parameter). Hence, κ may be of order $\sqrt{n}$ or higher.

2.2 Simulation

Exercice 9 : 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 2 and 3.

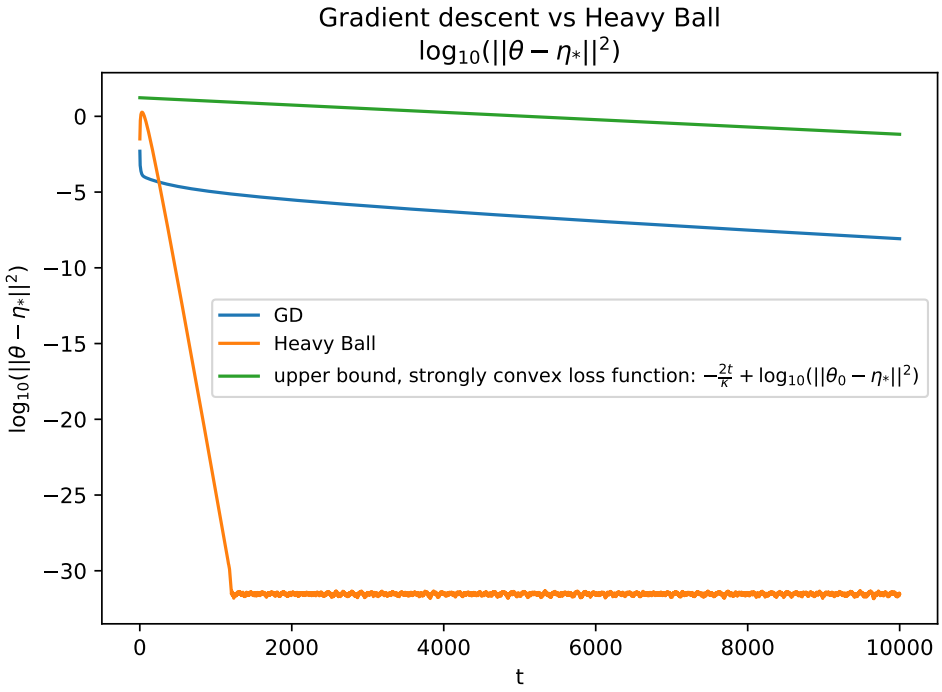

FIGURE 2 – Heavy ball vs GD, semilog scale

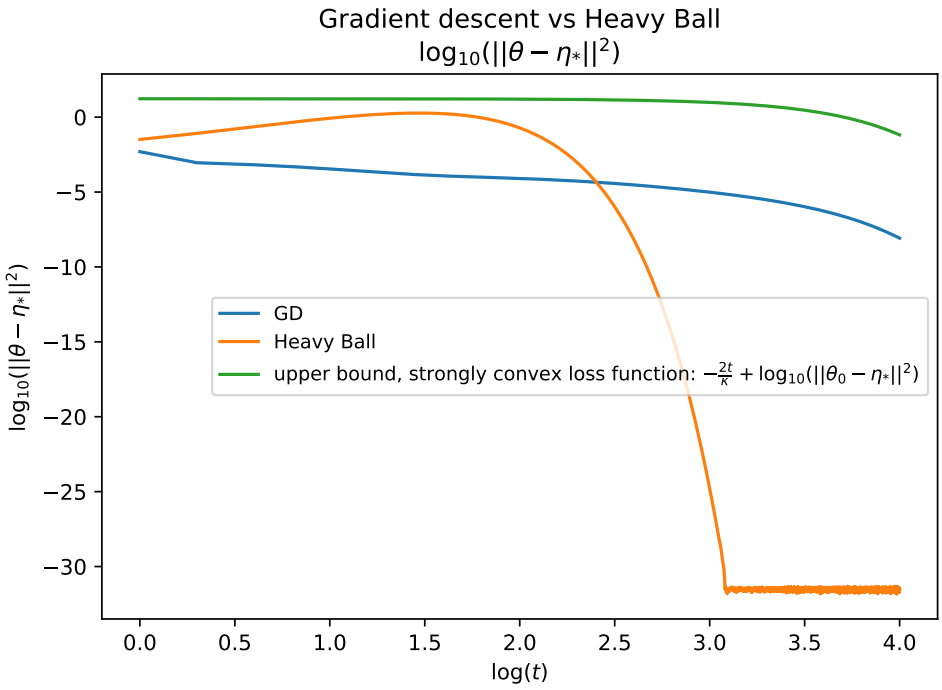

FIGURE 3 – Heavy ball vs GD, logarithmic scale

3 LOWER BOUND ON THE PERFORMANCE OF GRADIENT-BASED ALGORITHMS

In previous sessions, we have discussed several upper bounds on the performance of machine learning methods.

For instance, we have seen that for a strongly convex smooth function f defined on $\mathbb{R}^d$ and with real values, the convergence of gradient descent to the minimizer x^* of f can be upper bounded. More formally, we have seen that if $(x^t)_{t\in\mathbb{N}}$ is the sequence of iterates, then

$$\|x^t - x^*\| = \mathcal{O}(\alpha^t) \tag{8}$$

with $0 \leqslant \alpha < 1$. If f is only convex, and not necessary strongly convex, then we have a weaker upper bound :

$$f(x^t) - f(x^*) = \mathcal{O}(1/t) \tag{9}$$

Having an upper bound guarantees that the algorithm converges **faster** than the bound.

In this session we present the problem of obtaining **lower** bounds on the performance of machine learning methods. This means proving that an algorithm can not be guaranteed to converge too fast to the minimizer of f. We will formally define what this means in section 3.3, as it is not as straightforward as for upper bounds.

There are two kinds of lower bounds : statistical lower bounds and optimization lower bounds, which we will focus on.

Let $\mathcal{F}$ be the space of functions $f : \mathbb{R}^d \to \mathbb{R}$ that

— are convex and differentiable

— and have a minimizer x_f^*

— are L-smooth, with $L \in \mathbb{R}_+$ (the gradient is L-Lipshitz continuous).

3.1 First-order methods

We consider optimization algorithms that are based on linear combinations of local estimations of the gradient of f. These methods are called **first-order methods.** GD is an example of first-order method. Formally, this means that each iterate x^t verifies

$$x^t \in x^0 + \text{span}\{\nabla f(x^0), \dots, \nabla f(x^{t-1})\} \tag{10}$$

"span" means "espace engendré par".

3.2 Nesterov acceleration (AGD)

Nesterov acceleration (also called accelerated gradient descent (AGD)), is another first-order method, slightly different than GD. It is possible to prove that with Nesterov acceleration, the rate of convergence is $\mathcal{O}(1/t^2)$, instead of $\mathcal{O}(1/t)$.

https://blogs.princeton.edu/imabandit/2013/04/01/acceleratedgradientdescent/

3.3 Minimax lower bound

It is possible to show that AGD is optimal among first order methods. But first, we have to define what this means. This optimality is defined as a minmax problem. Indeed, we are interested in finding the algorithm that has the best worst-case performance. It is necessary to restrict algorithms to a relevant class of algorithms $\mathcal{A}$ (here, first order methods), and the functions to a relevant class of functions $\mathcal{F}$ (for instance the set of convex, L-smooth functions on $\mathbb{R}^d$).

For $k \in \mathbb{N}$, we look for an algorithm $a \in \mathcal{A}$ defined by

$$a = \arg\min_{a \in \mathcal{A}} \max_{f \in \mathcal{F}} \left[\|x_a^k - x^*\| \right] \tag{11}$$

where x_a^k is the iterate returned by algorithm a at iteration k.

3.4 Optimality of Nesterov acceleration

If we manage to show that for any algorithm $a \in \mathcal{A}$, there exists a function $f \in \mathcal{F}$, such that

$$f(x^k) - f(x^*) \geqslant \frac{C}{k^2} \|x^0 - x_f^*\|^2 \tag{12}$$

where C is independent on the function, then this will prove that Nesterov acceleration is optimal in $\mathcal{A}$, as this means that for any algorithm in $\mathcal{A}$, there exists a function, for which the iterates produced by a converge slower than $\frac{C}{k^2}$ to its minimizer.

3.5 Hard function

Without loss of generality, we assume that for all algorithms in $\mathcal{A}$, the initialization is $x^0 = 0 \in \mathbb{R}^d$.

We consider $k \leqslant \frac{d-1}{2}$ and f defined by

$$f(x) = \frac{L}{4}\left(\frac{1}{2} x_1^2 + \frac{1}{2} \sum_{i=1}^{2k} (x_i - x_{i+1})^2 + \frac{1}{2} x_{2k+1}^2 - x_1 \right) \tag{13}$$

Exercice 10 : Show that f is convex.

We admit that f is L-smooth, and that the minimizer of f, x^*, is

$$x_i^* : \begin{cases} 1 - \frac{i}{2k+2} \text{ for } 1 \leqslant i \leqslant 2k+1 \\ 0 \text{ for } i \geqslant 2k+2 \end{cases}$$

and that the minimum value is

$$f^* = \frac{L}{8}\left(\frac{1}{2k+2} - 1 \right) \tag{14}$$

We consider another utility function g defined as

$$g(x) = \frac{L}{4}\left(\frac{1}{2} x_1^2 + \frac{1}{2} \sum_{i=1}^{k-1} (x_i - x_{i+1})^2 + \frac{1}{2} x_k^2 - x_1 \right) \tag{15}$$

We also admit that the minimum value of g, noted g^*, is

$$g^* = \frac{L}{8}\left(\frac{1}{k+1} - 1 \right) \tag{16}$$

Exercice 11 : Compute the gradient of f for any point $x \in \mathbb{R}^d$.

Exercice 12 : Show that for all $l \leqslant d$, the components j with $j \geqslant l+1$ are null for the iterate x_l, for any first-order method that produces iterates with gradients of f (as defined in 10). In particular, this is true for $l = k$ and for the iterate k, we have

$$x_{k+1}^k = x_{k+2}^k = \cdots = x_d^k = 0 \tag{17}$$

Exercice 13 : Show that $f(x^k) = g(x^k)$.

Hence, $f(x^k) \geqslant g^*$ and

$$\frac{f(x^k) - f^*}{||x^0 - x^*||^2} \geqslant \frac{g^* - f^*}{||x^0 - f^*||^2} \tag{18}$$

Exercice 14 : Show that

$$||x^*||^2 \leqslant \frac{2k+2}{3} \tag{19}$$

Exercice 15 : Conclude.

3.6 Discussion

By essence, considering the worst-case performance (the hardest function to optimize) is pessimistic. Some functions optimized wit a first-order method by converge faster than $\mathcal{O}(1/t^2)$. We have seen this behavior for a least-squares setting in a previous session.