Tutorial: Last year's exam

M2 IASD Apprentissage

November 16, 2023

Dauphine | PSL
UNIVERSITÉ PARIS

Note: This is the exam given in December 2022. It was available in both an English version and a French version. Students could use any of those languages to write their paper. For sake of brevity, we only provide here the English version.

Foreword

In this exam, we will look at various optimization problems arising from the use of the Huber loss (named after the Swiss statistician Peter J. Huber), a loss function commonly used in robust statistics. The exercises are meant to be independent, yet notations will be passed along from one exercise to another.

- Dimensions of vectors and matrices will always be assumed to be greater than or equal to 1.
- The notation $\|\cdot\|$ will be used for the Euclidean vector norm.
- Given a vector $\boldsymbol{w} \in \mathbb{R}^d$ with $d \geq 1$, the ith coordinate of this vector will be denoted by $[\boldsymbol{w}]_i$.
- For any integer $d \geq 1$, the zero vector in $\mathbb{R}^d$ will be denoted by $\mathbf{0}_{\mathbb{R}^d}$.
- Throughout this exam, we define the *Huber loss* as the function ℓ from $\mathbb{R}$ to $\mathbb{R}$ such that

$$\ell(t) = \begin{cases} \frac{1}{2}t^2 & \text{if } |t| < 1 \\ |t| - \frac{1}{2} & \text{otherwise.} \end{cases} \tag{1}$$

  This function behaves like $t \mapsto \frac{t^2}{2}$ for $|t| < 1$, and looks like $t \mapsto |t|$ when $|t|$ becomes large. Unlike what its expression might suggest, it is continuously differentiable (i.e. $\ell \in \mathcal{C}^1$).

Exercise 1: Huber loss and linear models

We consider a dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Our goal is to find a linear model that predicts every y_i given the corresponding $\boldsymbol{x}_i$ as best as possible. We thus define a family of model functions parameterized by a vector $\boldsymbol{w}$ as follows:

$$\begin{array}{rccl} h_{\boldsymbol{w}}: & \mathbb{R}^d & \to & \mathbb{R} \\ & \boldsymbol{x} & \mapsto & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{w} = \sum_{i=1}^d [\boldsymbol{x}]_i [\boldsymbol{w}]_i. \end{array}$$

Given $h_{\boldsymbol{w}}$, we will consider that the model function correctly predicts y_i from $\boldsymbol{x}_i$ if

$$\ell\left(h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i\right) = \ell\left(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i\right) = 0.$$

The value $\ell\left(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i\right)$ represents the error of the model at $(\boldsymbol{x}_i, y_i)$. Therefore, we are interested in selecting a model (i.e. a vector $\boldsymbol{w} \in \mathbb{R}^d$) such that the sum of the errors is minimized. As a result, we arrive at the following optimization problem:

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{w}) := \frac{1}{n}\sum_{i=1}^n \ell(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i). \tag{2}$$

a) Justify that 0 is a lower bound for problem (2). Is it necessarily the minimum value of (2)?

b) The gradient of f at $\boldsymbol{w} \in \mathbb{R}^d$ is given by

$$\nabla f(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^n \ell'(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i)\boldsymbol{x}_i, \tag{3}$$

with

$$\ell'(t) = \begin{cases} 1 & \text{if} \quad t > 1 \\ t & \text{if} \quad |t| \leq 1 \\ -1 & \text{if} \quad t < -1. \end{cases}$$

i) Write down the gradient descent iteration with a constant stepsize α and using the formula (3).

ii) What happens to this iteration if the current point is a local minimum?

c) A Lipschitz constant for the gradient is given by $L = \frac{1}{n}\sum_{i=1}^n \|\boldsymbol{x}_i\|^2$.

i) How can this constant be used to choose the stepsize?

ii) Provide two additional ways of choosing the stepsize when the value of L is unknown.

d) The function f can be decomposed as $f = \frac{1}{n}\sum_{i=1}^n f_i$, with $f_i(\boldsymbol{w}) = \ell(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i)$. The gradient of f_i at $\boldsymbol{w}$ is

$$\nabla f_i(\boldsymbol{w}) = \ell'(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i)\boldsymbol{x}_i.$$

Write down the stochastic gradient iteration for this problem with a generic stepsize choice.

e) *We suppose that our unit of cost is an access to one $\boldsymbol{x}_i$.* What is the cost of a gradient iteration and that of a stochastic gradient iteration?

f) We now consider a batch stochastic gradient method in which we select a subset of n_b components.

   i) Write the resulting iteration using a constant stepsize.

   ii) If m is the number of processors available for computation, what can be the interest of choosing $n_b = m$?

   iii) Practical situation: suppose that we use several batch sizes and we observe that going from $n_b = 1$ to $n_b = n/4$ constantly gives better results in terms of convergence speed. Suppose that we also see a degradation in the convergence speed for batch values greater than $n/4$. How can this behavior be explained?

g) Using batches of gradients is one way of reducing the variance in the stochastic gradient estimates. Name one other variance reduction technique among those seen in class.

h) Consider an instance of problem (2) for which the components of the stochastic gradient estimates differ by orders of magnitude. Propose (with justification) an advanced stochastic gradient method among those seen in class that could prove efficient given that property.

Exercise 2: Pseudo-Huber loss

The goal of this exercise is to replace the Huber loss by a (smoothed) *pseudo-Huber loss function*, namely:

$$\begin{array}{rccl} p: & \mathbb{R} & \to & \mathbb{R} \\ & t & \mapsto & p(t) := \sqrt{1+t^2} - 1. \end{array} \tag{4}$$

It can be shown that the function p is twice continuously differentiable (whereas ℓ is only once continuously differentiable). Its derivatives are given for every $t \in \mathbb{R}$ by

$$\nabla p(t) = \tfrac{t}{\sqrt{1+t^2}} \quad \text{and} \quad \nabla^2 p(t) = \tfrac{1}{1+t^2}.$$

a) Justify that the function p is convex. What can be said of its local minima ?

b) Show that $\operatorname{argmin}_{t\in\mathbb{R}} p(t) = \{0\}$.

c) Using the same data as that of Exercise 1, consider the problem

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}} \; g(\boldsymbol{w}) := \frac{1}{n}\sum_{i=1}^{n} p(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i). \tag{5}$$

The function g is convex but not strongly convex in general.

   i) What is the convergence rate of gradient descent applied to problem (5)? What quantity does this rate apply to?

   ii) Give the convergence rate of accelerated gradient on problem (5). Is it better or worse than that of gradient descent?

d) Suppose that we apply a stochastic gradient method to problem (5) by exploiting its finite-sum structure, and that the method we apply has a convergence rate (in expectation) in $\mathcal{O}\left(\frac{1}{\sqrt{K}}\right)$ for the same quantity than that considered in question c), with K being the iteration count.

i) In terms of iterations, justify that this rate is worse than that of gradient descent.

ii) Is there another metric in which this rate can be better than that of gradient descent? If so, which metric?

iii) Does the result from the previous question apply when we compare the rates for stochastic gradient and accelerated gradient?

Exercise 3: Reversed Huber loss

In this exercise, we consider the reverse philosophy of the Huber loss, that is, we propose to use a loss function that looks like the absolute value on $[-1, 1]$ and like a quadratic everywhere else. The *reversed Huber loss* is thus defined as:

$$\begin{array}{rccl} r: & \mathbb{R} & \to & \mathbb{R} \\ & t & \mapsto & r(t) := \begin{cases} |t| & \text{if } |t| < 1 \\ \frac{t^2+1}{2} & \text{otherwise.} \end{cases} \end{array} \tag{6}$$

This function is convex but nonsmooth, since it is not differentiable at 0.

As in Exercise 1, we consider linear models $\boldsymbol{x} \mapsto \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$ and a dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$.

a) We first consider the convex, nonsmooth problem:

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}} \frac{1}{n}\sum_{i=1}^{n} r(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i). \tag{7}$$

i) What mathematical tool can we use to design algorithms applicable to problem (7)?

ii) Using this tool, how can the solutions of (7) be characterized?

b) We now study the family of problems:

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{w}) + \lambda\sum_{i=1}^{d} r([\boldsymbol{w}]_i), \tag{8}$$

where f is the objective function of (2), and $\lambda > 0$.

i) How is this type of problem called? What is the purpose of the second term?

ii) Write the generic proximal gradient iteration for this problem.

iii) When is this algorithm worthy of consideration in practice?

Exercise 4: Large-scale reversed Huber loss

In this exercise, we consider an instance of the problem family (8). More precisely, we focus on

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\; f(\boldsymbol{w}) + \gamma\Omega(\boldsymbol{w}), \tag{9}$$

for a fixed value $\gamma \geq 0$, where $f = \frac{1}{n}\sum_{i=1}^n f_i$ is defined as in Exercise 1 and $\Omega(\boldsymbol{w}) := \sum_{i=1}^d r([\boldsymbol{w}]_i)$ with r being the function (6) defined in Exercise 3.

a) Suppose first that the number of parameters d is quite large.

   i) Assuming $\gamma = 0$, write down a block coordinate descent iteration for problem (9).

   ii) How can you combine this iteration with other algorithms seen in class to develop a method based on coordinate updates that can be applied to problem (9) with $\gamma > 0$?

b) We now introduce an auxiliary variable to the problem, which we then rewrite as

$$\begin{array}{ll}\underset{\substack{\boldsymbol{u}\in\mathbb{R}^d\\ \boldsymbol{v}\in\mathbb{R}^d}}{\text{minimize}} & f(\boldsymbol{u}) + \gamma\Omega(\boldsymbol{v}) \\ \text{subject to} & \boldsymbol{u} - \boldsymbol{v} = \boldsymbol{0}_{\mathbb{R}^d}.\end{array} \tag{10}$$

   i) Write down the augmented Lagrangian formula for problem (10).

   ii) Which method is based on the augmented Lagrangian and can exploit the structure of problem (10)? What is the main idea behind exploiting such a structure?

c) Suppose now that the number of data points used in defining f is so large that all f_i are spread across several agents, each of which has only access to its own f_i and maintains its own copy $\boldsymbol{u}^{(i)}$ of $\boldsymbol{u}$. All agents share knowledge of $\boldsymbol{v}$,γ and Ω.

   i) Rewrite problem (10) to model the distributed aspect of the problem as described above.

   ii) How can the method from question b)ii) be adapted to this new setting?

Solutions (English version)

Solutions: Exercise 1

a) The function ℓ is nonnegative on $\mathbb{R}$. Thus, for any $\boldsymbol{w} \in \mathbb{R}^d$,

$$f(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i) \geq \frac{1}{n}\sum_{i=1}^{n} 0 = 0.$$

This shows that the value 0 is a lower bound for problem (**??**). This value is only attained if there exists a $\boldsymbol{w}$ such that $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i = 0$ for every i: this might not always be the case (for instance with $n = 2, d = 1, \boldsymbol{x}_1 = 1, \boldsymbol{x}_2 = -1, y_1 = y_2 = 1$), therefore 0 is not necessarily the minimum value of the problem.

b)

i) At a point $\boldsymbol{w}_k \in \mathbb{R}^d$, the gradient descent iteration with a constant stepsize α is written as follows:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \frac{\alpha}{n}\sum_{i=1}^{n} \ell'(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w}_k - y_i)\boldsymbol{x}_i.$$

ii) If $\boldsymbol{w}_k$ is a local minimum, then $\nabla f(\boldsymbol{w}_k) = 0$, and the gradient descent iteration reduces to $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k$.

c)

i) If a Lipschitz constant L for the gradient is known, an appropriate choice for a constant stepsize is $\alpha = \frac{1}{L}$.

ii) When such a value is unknown, one can use a decreasing stepsize (e.g. $\alpha_k = \frac{1}{k+1}$) or perform a line search at every iteration in order to find an appropriate stepsize value.

d) At $\boldsymbol{w}_k \in \mathbb{R}^d$, the stochastic gradient iteration with generic stepsize α_k consists in two steps. First, an index i_k is drawn at random in $\{1, \ldots, n\}$; secondly, the new iterate is computed using the formula:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f_{i_k}(\boldsymbol{w}_k) = \boldsymbol{w}_k - \alpha_k \ell'(\boldsymbol{x}_{i_k}^{\mathrm{T}}\boldsymbol{w}_k - y_{i_k})\boldsymbol{x}_{i_k}.$$

e) Every gradient descent iteration must access all the data in order to compute the gradient: if our unit of cost is an access to one $\boldsymbol{x}_i$, the cost of a gradient descent iteration is n. As for the stochastic gradient iteration, it only accesses one data point ($\boldsymbol{x}_{i_k}$ where i_k is drawn at random): its cost is thus 1.

f)

i) The batch stochastic gradient iteration at $\boldsymbol{w}_k \in \mathbb{R}^d$ consists in two steps. First, a random index set $S_k \subset \{1, \ldots, n\}$ of size $|S_k| = n_b$ is drawn; then, the following iteration is performed:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \frac{\alpha}{|S_k|}\sum_{i \in S_k} \nabla f_i(\boldsymbol{w}_k),$$

where $\alpha > 0$ is the constant stepsize.

ii) If m processors are available for computation, and the gradients ∇f_i can be computed in parallel, the cost of a batch stochastic gradient can be distributed over these m processors.

iii) If improvement is observed while using a small batch size, this means that the data is sufficiently correlated that considering a subset of it at every iteration is enough to converge. Using more than one data point at every iteration also leads to steps with a lower variance, and this explains why $n_b = n/4$ can give better performance than $n_b = 1$ (classical stochastic gradient iteration). When the batch size gets closer to n, its cost also gets closer to that of a full gradient iteration, and the method is also at risk of suffering from redundancies in the data. This explains why the behavior of the method worsens when $n_b > n/4$.

g) Gradient aggregation is another technique that reduces the variance. *Another valid answer was iterate averaging.*

h) The use of a variant with diagonal scaling is appropriate in this setting, since it uses different stepsizes for every coordinate. As such, it will be less sensitive to differences of magnitude between gradient components.

Solutions: Exercise 2

a) From the formula for the second derivative of p, we observe that

$$\nabla^2 p(t) = \frac{1}{1+t^2} > 0 \ \forall t \in \mathbb{R}.$$

Therefore, the function p has a positive definite Hessian, which means that it is convex. This implies that any local minimum of u is a global minimum.

b) We have $p(t) \geq 0$ for every $t \in \mathbb{R}$, and $p(0) = 0$: this allows us to conclude that 0 is a global minimum of the function. In addition, for any value $t > 0$, we have $p(t) > 0$, showing that 0 is the unique global minimum.

c)

i) After $K \geq 1$ iterations of gradient descent, the convergence rate guarantee is

$$g(\boldsymbol{w}_k) - \min_{\boldsymbol{w} \in \mathbb{R}^d} g(\boldsymbol{w}) \ \leq \ \mathcal{O}\left(\frac{1}{K}\right).$$

ii) The convergence rate of accelerated gradient on this problem is $\mathcal{O}\left(\frac{1}{K^2}\right)$, which is better than that for gradient descent.

d)

i) The sequence $\left\{\frac{1}{\sqrt{K}}\right\}$ converges to 0 more slowly than $\left\{\frac{1}{K}\right\}$, which is the rate of gradient descent. As a result, the rate for stochastic gradient is worse in terms of iterations.

ii) By comparing the convergence rates in terms of epochs rather than iterations, we obtain (for a fixed epoch number $N_E \geq 1$) a rate of $\mathcal{O}\left(\frac{1}{\sqrt{nN_E}}\right)$ for stochastic gradient and a rate of $\mathcal{O}\left(\frac{1}{N_E}\right)$ for gradient descent. The former is better when $n \gg N_E$.

iii) The same observation applies to the rates for stochastic gradient and accelerated gradient, which become $\mathcal{O}\left(\frac{1}{\sqrt{nN_E}}\right)$ and $\mathcal{O}\left(\frac{1}{N_E^2}\right)$, respectively. For stochastic gradient to yield a better convergence rate, one needs $n \gg N_E^2$, which is a stronger requirement.

Solutions: Exercise 3

a)

i) Since $\boldsymbol{w} \mapsto \frac{1}{n}\sum_{i=1}^n r(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i)$ is convex, it is possible to define the subdifferential of v at any point: the elements of the subdifferential, called the subgradients, can be used in lieu of the gradient to construct optimization methods for solving problem (7).

ii) Let $\phi_r : \boldsymbol{w} \mapsto \frac{1}{n}\sum_{i=1}^n r(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i)$. A point $\bar{\boldsymbol{w}} \in \mathbb{R}^d$ is a global minimum of ϕ_r if and only if

$$\mathbf{0} \in \partial\phi_r(\bar{\boldsymbol{w}}),$$

where $\partial\phi_r(\cdot)$ denotes the subdifferential of ϕ_r.

b)

i) Problem (8) is a regularized (or composite) optimization problem. The goal of the second term, that does not depend on data, is to enforce desired properties for the solution.

ii) At a point $\boldsymbol{w}_k$, the generic proximal gradient iteration (with a generic stepsize α_k) for this problem is:

$$\boldsymbol{w}_{k+1} \in \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} \left\{ f(\boldsymbol{w}_k) + \nabla f(\boldsymbol{w}_k)^{\mathrm{T}}(\boldsymbol{w} - \boldsymbol{w}_k) + \frac{1}{2\alpha_k}\|\boldsymbol{w} - \boldsymbol{w}_k\|_2^2 + \lambda \sum_{i=1}^d r([\boldsymbol{w}]_i) \right\}.$$

iii) The proximal gradient algorithm is only interesting when the cost of solving the subproblem is cheaper than that of solving the original problem.

Solutions: Exercise 4

a)

i) Starting from $\boldsymbol{w}_k \in \mathbb{R}^d$, a block coordinate descent iteration with a generic stepsize $\alpha_k > 0$ can be written as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \sum_{j\in\mathcal{B}_k} [\nabla f(\boldsymbol{w}_k)]_j \boldsymbol{e}_j,$$

where $\mathcal{B}_k \subset \{1, \ldots, d\}$ is a block of coordinate indices

ii) Since Ω is separable, we can combine proximal gradient and coordinate descent to perform updates only concerned with the coordinates in the block, leading to the iteration

$$\boldsymbol{w}_{k+1} \in \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^d} \left\{ f(\boldsymbol{w}_k) + \sum_{j\in\mathcal{B}_k} [\nabla f(\boldsymbol{w}_k)]_j \boldsymbol{e}_j^{\mathrm{T}}(\boldsymbol{w} - \boldsymbol{w}_k) + \frac{1}{2\alpha_k}\|\boldsymbol{w} - \boldsymbol{w}_k\|^2 + \sum_{j\in\mathcal{B}_k} r([\boldsymbol{w}]_j) \right\}$$

An answer without a formula could have sufficed.

b)

i) The augmented Lagrangian for problem (10) is

$$\mathcal{L}^a(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z}; \lambda) = f(\boldsymbol{u}) + \gamma\Omega(\boldsymbol{v}) + \boldsymbol{z}^{\mathrm{T}}(\boldsymbol{u} - \boldsymbol{v}) + \frac{\lambda}{2}\|\boldsymbol{u} - \boldsymbol{v}\|^2.$$

ii) The Alternating Direction Method of Multipliers, or ADMM. Its goal consists in exploiting separable structure in the problem to perform possibly cheaper updates consecutively rather than jointly.

c)

i) The problem can be rewritten by adding the copies as variables, so that we obtain

$$\begin{array}{ll} \text{minimize}_{\substack{\boldsymbol{u}^{(1)},\ldots,\boldsymbol{u}^{(n)} \in \mathbb{R}^d \\ \boldsymbol{v} \in \mathbb{R}^d}} & \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{u}^{(i)}) + \gamma\Omega(\boldsymbol{v}) \\ \text{subject to} & \boldsymbol{u}^{(i)} - \boldsymbol{v} = \mathbf{0}_{\mathbb{R}^d} \quad \forall i = 1, \ldots, n. \end{array}$$

ii) To adapt ADMM, one would consider concurrent updates on the variables $\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(n)}$ so as to exploit the separability of the objective further.