M2 MIAGE
& MIAGE IDApp
2025-2026

OPTIMIZATION FOR DATA SCIENCE

January 21, 2026

Today: Exercises (including last year's exam)

Monday: Lab session / Homework, deadline Feb 11
(Notebook online today)

For the exam
- Answers can be in French or English
- 1 sheet of notes allowed (A4, two-sided)

↓

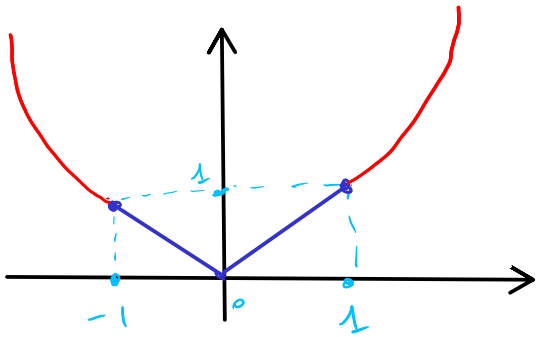
February 2nd, 2 hours (likely 4 exercises)

Exercise 2 (Tutorial 4) : Reversed Huber loss

↳ Last session, we studied the Huber loss

$$l(t) = \begin{cases} \frac{t^2}{2} & \text{if } |t| < 1 \\ |t| - \frac{1}{2} & \text{if } |t| \geq 1 \end{cases}$$

↳ Reversed Huber loss :



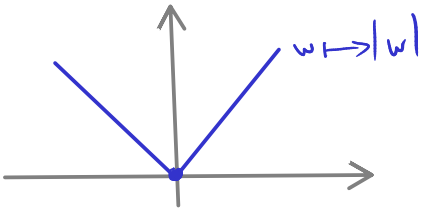
$$r(t) = \begin{cases} |t| & \text{if } |t| < 1 \\ \frac{t^2+1}{2} & \text{if } |t| \geq 1 \end{cases}$$

→ The function r is not C^1 (because it corresponds to the absolute value around 0)

⇒ We say that the function r is nonsmooth

→ For us, a nonsmooth function $f : \mathbb{R}^d \to \mathbb{R}$ is a function that there is at least one point in $\mathbb{R}^d$ at which the gradient of f does not exist

Ex) $w \mapsto |w|$ $d=1$ nonsmooth (no gradient at 0)



NB: This function is convex (and has a single global minimum)

Ex) In dimension d, $w \mapsto \|w\|_1 = \sum_{j=1}^{d} |w_j|$

is nonsmooth.

The gradient does not exist at a point $w \in \mathbb{R}^d$ with at least one zero coordinate $\left(\begin{bmatrix}0\\1\\\vdots\\1\end{bmatrix}, \begin{bmatrix}0\\1\\\vdots\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\\\vdots\end{bmatrix}, \dots\right)$

$\Rightarrow$ Yet $\|\cdot\|_1$ is convex and has a unique minimum at $\begin{bmatrix}0\\\vdots\\0\end{bmatrix}$

↳ To minimize nonsmooth functions, we cannot use gradients!

- Cannot check whether a point is a minimum using the gradient
- Cannot run gradient descent / stochastic gradient

↳ We can use subgradients instead of gradients to minimize nonsmooth convex functions

f C^1 convex	f nonsmooth convex
$w^* \in \operatorname{argmin}_w f(w) \iff \nabla f(w^*) = 0$	$w^* \in \operatorname{argmin}_{w \in \mathbb{R}^d} f(w) \iff 0 \in \partial f(w^*)$
	subdifferential (set of subgradients of f at w^*)
Gradient descent	$\partial f(w^*) \subseteq \mathbb{R}^d$
$w_{k+1} = w_k - \alpha_k \nabla f(w_k)$	Subgradient method
$\alpha_k > 0$	$w_{k+1} = w_k - \alpha_k g_k$
	$\alpha_k > 0$
	$g_k \in \partial f(w_k)$

Ex) $f: w \mapsto |w|$ in dimension 1

$$\partial f(w) = \begin{cases} \{1\} & \text{if } w > 0 \\ \{-1\} & \text{if } w < 0 \\ [-1,1] & \text{if } w = 0 \end{cases}$$

$$0 \in \partial f(w) \iff w = 0 \quad \text{(minimum of } f\text{)}$$

Remark: . Most neural network packages can compute subgradients automatically (automatic differentiation)

$\Rightarrow$ Needed because many architectures use nonsmooth functions, e.g. $\mathrm{ReLU}: t \mapsto \max(t, 0)$

. Methods like SGD, Adam, etc are actually implemented using subgradients and not gradients (and it works!)

Back to the exercise

a) minimize $\frac{1}{n} \sum_{i=1}^{n} r(x_i^T w - y_i)$ over $w \in \mathbb{R}^d$

where $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$ $\forall i = 1..n$ (regression data)

$$r(t) = \begin{cases} |t| & \text{if } |t| < 1 \\ \frac{t^2+1}{2} & \text{if } |t| \geq 1 \end{cases}$$

This objective function is convex but nonsmooth

i) what mathematical tool can be used to solve the problem?

subgradients / subdifferential

ii) Using this tool, how can we characterize solutions of the problem?

Let $g(w) = \frac{1}{n} \sum_{i=1}^{n} r(x_i^T w - y_i)$

Then $w^* \in \underset{w \in \mathbb{R}^d}{\operatorname{argmin}}\, g(w) \iff 0 \in \partial g(w^*)$

b) Consider now

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \quad f(w) + \lambda \sum_{j=1}^{d} r(w_j)$$

↳ does not depend on data

where $f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$, every f_i is C^1 and depends on (x_i, y_i) → depends on data

$\lambda > 0$

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

→ The problem is a regularized problem

(→ The first term is a data-fitting term that represents the task we want to perform)

→ The second term penalizes vectors w / models that do not have a desired structure

ii) Write the proximal gradient iteration for this problem

$$w_{k+1} \in \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ f(w_k) + \nabla f(w_k)^T (w - w_k) + \frac{1}{2\alpha_k} \|w - w_k\|^2 + \lambda \sum_{j=1}^{d} r([w]_j) \right\}$$

where $\alpha_k > 0$

$[w]_j$: j^{th} coordinate of w

iii) When is proximal gradient useful in practice?

When the subproblems that are solved at every iteration are easy to solve (at least easier to solve than the original problem)

Exercise 1 for Tutorial 4

Data: $X \in \mathbb{R}^{n \times d}$ $y \in \mathbb{R}^n$ (Goal: Regression with a linear model)

Problem:

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \quad \frac{1}{2n} \|Xw - y\|^2 + \frac{\lambda_2}{2} \|w\|^2 + \lambda_1 \|w\|_1$$

$\frac{1}{2n} \|Xw - y\|^2$: Data-fitting term (linear regression objective)

$\frac{\lambda_2}{2} \|w\|^2 + \lambda_1 \|w\|_1$: Elastic net regularization (uses two parameters $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$)

a) General goal of regularization? (see question b)i) above)

b) $\lambda_1 = 0$, $\lambda_2 > 0$: what is the goal of the regularization then?

$\frac{\lambda_2}{2}\|w\|^2$: ℓ_2 regularization

$\|w\|^2 = \sum_{j=1}^{d} w_j^2$

$\|w\|_1 = \sum_{j=1}^{d} |w_j|$

→ Reduces over-fitting / sensitivity to the data

→ Penalizes vectors with large ℓ_2 norm

c) Same question than b) with $\lambda_2 = 0$ and $\lambda_1 > 0$

$\lambda_1 \|w\|_1$: ℓ_1/LASSO regularization

→ Penalize vectors with many non-zero coordinates (promote sparse vectors)

→ Perform feature selection (okay here because we use a linear model Xw)

d) Write down the proximal gradient iteration using that

$\phi : w \mapsto \frac{1}{2n}\|Xw - y\|^2$ is C^1

$$w_{k+1} \in \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ \phi(w_k) + \nabla\phi(w_k)^T (w - w_k) + \frac{1}{2\alpha_k}\|w - w_k\|^2 + \frac{\lambda_2}{2}\|w\|^2 + \lambda_1\|w\|_1 \right\}$$

e) When $\lambda_2 = 0$, which algorithm does proximal gradient correspond to?

ISTA

(Unlike general proximal gradient, ISTA has explicitly defined iterates)

f) When $\lambda_1 > 0$ and $\lambda_2 > 0$, the proximal subproblems have no closed-form solutions, and these problems must be solved numerically. Propose an algorithm among those seen in class to solve such problems.

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \; \phi(w_k) + \nabla\phi(w_k)^T(w - w_k) + \frac{1}{2\alpha_k}\|w - w_k\|^2 + \frac{\lambda_2}{2}\|w\|^2 + \lambda_1\|w\|_1$$

→ The objective function is nonsmooth, so we cannot use GD or SG

→ We could use a subgradient method

→ We could use a proximal gradient method

$$\phi(w_k) + \nabla\phi(w_k)^T(w - w_k) + \frac{1}{2\alpha_k}\|w - w_k\|^2 + \frac{\lambda_2}{2}\|w\|^2 + \lambda_1\|w\|_1$$

Smooth part (C^1) — Regularizer

⇒ With this decomposition, can solve the subproblems using ISTA

→ With the other decomposition, you get subproblems that are as difficult as the original subproblems

TUTORIAL 5 (Exam from 2024 - 2025)

Ex 1

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \quad f^{log}(w) = \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + \exp(-y_i x_i^T w)\right)$$

$x_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$ $\forall i = 1..n$

(classification)

f^{log} is C^1 and convex

a) Write the iteration of gradient descent using a constant stepsize

$$w_{k+1} = w_k - \alpha_k \nabla f^{log}(w_k) \qquad \alpha_k = \alpha > 0$$

b) f^{log} is $C^{1,1}_L$ for some $L > 0$. If $\alpha = \frac{1}{L}$, what kind of guarantee do we have at every iteration?

$$f^{log}(w_{k+1}) < f^{log}(w_k) \quad \text{if} \quad w_k \text{ not a minimum}$$

$$\left(f^{log}(w_{k+1}) \leq f^{log}(w_k) - \underbrace{\frac{1}{2L} \|\nabla f^{log}(w_k)\|^2}_{\neq 0 \text{ if } w_k \text{ not a minimum}} \right)$$

$\rightarrow$ the function value decreases at every iteration $\left(f^{log}(w_{k+1}) \leq f^{log}(w_k)\right)$

c) Give another way of choosing the stepsize than using a constant value

→ Decreasing: Choose $\{\alpha_k\}$ a priori so that $\alpha_k \xrightarrow[k\to\infty]{} 0$

$$\left(\text{Ex: } \alpha_k = \frac{1}{\sqrt{k+1}}\right)$$

→ Adaptive: Choose α_k according to w_k and/or values of $f^{\log}(w_k)$ and/or $\nabla f^{\log}(w_k)$

(Ex: Line search)

d) Since $f^{\log}$ is convex, what is the convergence rate of GD on that problem? what quantity does the rate apply to?

After $K \geq 1$ iterations,

$$f^{\log}(w_K) - \min_{w \in \mathbb{R}^n} f^{\log}(w) \leq O\left(\frac{1}{K}\right)$$

$f^{\log}(w_K) - \min_{w \in \mathbb{R}^n} f^{\log}(w)$: Distance to the optimal function value

$O\left(\frac{1}{K}\right)$: Convergence rate of GD in the convex case

e) Assuming now that f is strongly convex, what is the convergence rate and which quantity does it apply?

After $K \geq 1$ iterations,

$$f^{\log}(w_K) - \min_{w \in \mathbb{R}^n} f(w) \leq O(t^K) \quad \text{where } t \in (0,1)$$

(t^K goes to 0 faster than $\frac{1}{K}$)

(If f is $C^{1,1}_{L}$ $L>0$ and μ-strongly convex, $\mu>0$, $t = 1-\frac{\mu}{L}$)

f) Accelerated gradient / Nesterov's method has a better convergence rate than gradient descent on convex problems. What is that rate?

$O\left(\frac{1}{K^2}\right)$ (applies to $f(w_K) - \min_w f(w)$)

($\frac{1}{K^2}$ goes to 0 faster than $\frac{1}{K}$: better rate)

g) Explain the main algorithmic idea behind accelerated gradient

(Accelerated gradient iteration:

$$w_{k+1} = w_k - \alpha_k \nabla f\left(w_k + \beta_{k+1}(w_k - w_{k-1})\right) + \beta_{k+1}(w_k - w_{k-1})$$

$\alpha_k > 0$ $\beta_{k+1} > 0$)

Idea: Combine gradient steps with information from the previous iteration

↓

"momentum step"

b) In terms of algorithm / implementation, what is the difference between accelerated gradient for convex and for strongly convex problems?

The parameter β_{k+1} is set differently for convex and for strongly convex problems

---

Ex 3 (stochastic gradient)

Consider a generic problem of the form

$$(P) \qquad \underset{w \in \mathbb{R}^d}{\text{minimize}} \quad f^{sto}(w) = \frac{1}{n} \sum_{i=1}^{n} f_i^{sto}(w)$$

where every f_i^{sto} is C^1 and depends on the i^{th} data point in a dataset with n elements, $n \gg 100$

n much bigger than 100

a) Justify that the solutions of (P) are the same than the solutions of

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \; \frac{1}{2n} \sum_{i=1}^{n} f_i^{sto}(w)$$

w^* solution of (P) $\iff$ $f^{sto}(w^*) \leq f^{sto}(w) \quad \forall w \in \mathbb{R}^d$

$\frac{1}{2} > 0$ (

$\Leftrightarrow \frac{1}{n}\sum_{i=1}^{n} f_i^{sto}(w^*) \leq \frac{1}{n}\sum_{i=1}^{n} f_i^{sto}(w) \quad \forall w \in \mathbb{R}^d$

$\Leftrightarrow \frac{1}{2n}\sum_{i=1}^{n} f_i^{sto}(w^*) \leq \frac{1}{4n}\sum_{i=1}^{n} f_i^{sto}(w) \quad \forall w \in \mathbb{R}^d$

$\Leftrightarrow w^*$ solution of $\min_w \frac{1}{2n}\sum_{i=1}^{n} f_i^{sto}(w)$

Multiplying the objective by $a > 0$ does not change the set of solutions

b) Write down the stochastic gradient iteration using a constant stepsize

$$w_{k+1} = w_k - \alpha_k \nabla f_{i_k}^{sto}(w_k) \qquad \alpha_k = \alpha > 0$$

i_k index drawn randomly in $\{1, \dots, n\}$

c) Recall the definition of an epoch. How many iterations of SG can be performed with a budget of 1 epoch?

An epoch is a unit of cost corresponding to n accesses to a data point in a dataset with n elements

1 epoch $\equiv$ n accesses to data points

$\equiv$ cost of n iterations of SG

d) Suppose that f_{sto} is nonconvex. Then we can show that the convergence rate of gradient descent is $O\left(\frac{1}{K^{1/4}}\right)$ (in expectation) after K iterations

i) What is the rate of GD on a nonconvex problem? Is it better or worse than that of SG?

$K^{1/2} = \sqrt{K}$

Rate of GD: After $K \geq 1$ iterations $O\left(\frac{1}{K^{1/2}}\right)$

$\Rightarrow$ Better than the rate of SG in terms of powers of K ($\frac{1}{K^{1/2}}$ converges to 0 faster than $\frac{1}{K^{1/4}}$)

(Also better because it is deterministic while the rate of SG is in expected value)

ii) Suppose that we have a budget of $E \geq 1$ epochs. What convergence rate do we obtain for GD?

1 iteration of GD costs 1 epoch

with E epochs, we do E iterations of GD

$\Rightarrow$ Rate $O\left(\frac{1}{E^{1/2}}\right)$

iii) With the same budget E of epochs, what is the rate of SG? Is it better or worse than that of GD?

with E epochs, do nE iterations of SG

$\Rightarrow$ Rate $O\left(\frac{1}{(nE)^{1/4}}\right)$

When ϵ small (< 100) and $n \gg 100$,

$$\frac{1}{(n\epsilon)^{1/4}} << \frac{1}{\epsilon^{1/2}}$$: SG for a better rate

e) Write down the iteration of batch stochastic gradient using a constant stepsize

$$w_{k+1} = w_k - \frac{\alpha_k}{|S_k|} \sum_{i \in S_k} \nabla f_i^{S_k}(w_k)$$

$$\alpha_k = \alpha > 0$$

S_k is a set of indices drawn randomly in $\{1, \dots, n\}$ with or without replacement

f) How does batch SG generalize SG and GD?

$|S_k| = 1 \rightarrow$ (vanilla) SG

$|S_k| = n$
Draw without replacement $\iff S_k = \{1, \dots, n\}$ $\forall k \rightarrow$ GD

g) Suppose that we compare four variants of batch SG using $\{1, \frac{n}{128}, \frac{n}{2}, n\}$ as batch sizes ($|S_k|$)

i) We run each variant 10 times and we observe higher variability for the variant with $|S_k| = 1$. Explain.

It illustrates that using a batch is a way to reduce variance.

i) the method with batch size $\frac{n}{2}$ converges more slowly than that with batch sizes 1 and $\frac{n}{128}$, but eventually reaches a smaller function value. Explain

- $n/2$ seems to be a large batch regime

→ 1 epoch ≡ 2 iterations : slow convergence in terms of epochs

→ Because the batch size is large, behavior close to GD and converge to a small neighborhood of the solution

Bonus question : Advanced SG variants

- If the coordinates of the gradients vary a lot in magnitude, what advanced variant could be appropriate to use?

Variants with diagonal scaling (Adagrad, RMSProp) are appropriate in this setting.

→ Idea : $[w_{k+1}]_j = [w_k]_j - \alpha_k \dfrac{[\nabla f_{i_k}^{stoch}(w_k)]_j}{[v_k]_j}$

v_k : normalization vector, adapts the stepsize to every coordinate

- Adam is the most popular SG variant for deep learning.
Explain the two algorithmic ideas Adam is based upon.

(All using information from all previous iterations)

1) Diagonal scaling: Scale each coordinate differently
2) Momentum/Acceleration: Combine gradient with previous steps