Exam of
Foundational Principles of Machine Learning (FPML)
December 20, 2024 – 3h

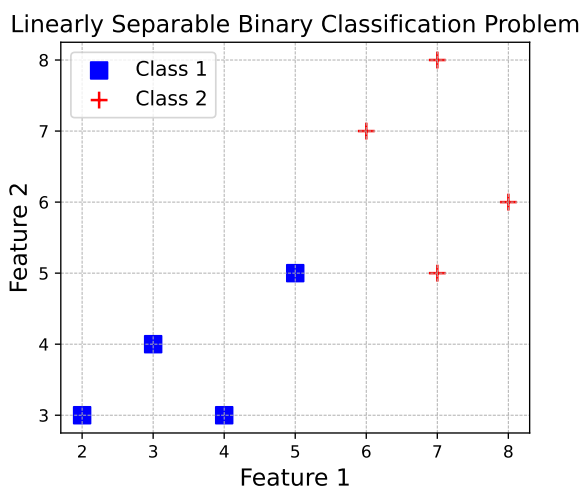
DON'T RETURN THIS SHEET BEFORE YOU ARE ALLOWED TO DO SO!
(you can write your name on the blank papers in the meantime)

General advice:

- **Authorized documents: 6 pages of personal notes.**
- **Exercises 1 and 2 are answered in set of papers, A**, and **Exercises 3 and 4 are answered in another set of papers B**. Differrent people will grade the differents sets.
- Do not hesitate to do the exercises in any order you like: start with the ones you feel are quick to deal with.
- When you are allowed to start, before you start the first exercise, go through the subject quickly. In each exercise, the most difficult question is not necessarily the last, please feel free to skip some questions. Don't hesitate to go and scrap off points where they are easy to take. (vous pouvez "aller grapiller les points")
- The grading points (scale) is indicative, if the exam is too long a correction factor will be applied. So don't panic in front of the length, what you do, do it right! Also, you may notice that points sum up to 24 (8.5+4+5.5+6) instead of 20, so, I will do *something*.
- **French:** Vous êtes autorisés à composer en Français. (Y compris en insérant des mots techniques comme overfitting ou regularization en anglais quand vous ne savez pas la traduction).
- **French:** si certains bouts de l'énoncé ne sont pas clairs, je peux les traduire ! N'hésitez pas à demander si vous n'êtes pas sûrs.
- Calculators not allowed (and useless). No electronic device allowed (cell phone, etc).
- At the end, we will collect your papers. You can leave after you have returned your paper.

DON'T RETURN THIS SHEET BEFORE YOU ARE ALLOWED TO DO SO!
(you can write your name on the copies in the meantime)

1 (Set A) Lecture related questions, all independent (8.5 points)



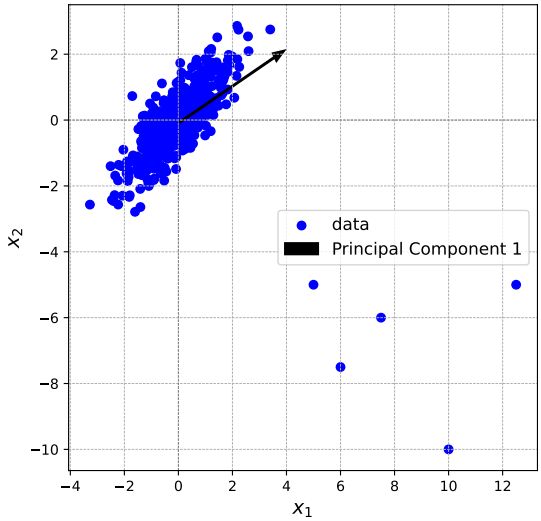


Figure 1: Left: Figure related to question 1.Right: Figure related to question 2.

1. (1.5 pt) In Figure 1 (Left) we show a handful of points of a binary classification problem. Reproduce it on your paper, with a grid showing clearly the cartesian coordinates, esp. in the range $x_1 \in [5, 7], x_2 \in [5, 7]$.
   (a) (0.5 pt) Indicate the linear SVM solution.
   (b) (0.25 pt) Plot two other linear separators between the two classes.
   (c) (0.25 pt) Circle the support vectors.
   (d) (0.5 pt) Hash the region where squares (class 1) points can be added without changing the SVM solution.

2. (1.5 pt) In Figure 1 (Right) we show a number of data points in 2D, with some outliers, and the result of a PCA with 1 component shown.
   (a) (0.5 pt) Why is the arrow showing this 1st component not aligned with the main axis of the blob of points ? (by blob, we mean the group of points that are clustered together in the top-left corner).
   (b) (0.5 pt) Would you say this is a desirable property of PCA, or not ? You can take several lines to really discuss.
   (c) (0.5 pt) And in particular, if you think of PCA only as a way to look at data in low dimension, is this property desirable ?

3. (1 pt) Define what is a Kernel (in the sense of, feature maps and Kernels, as in, the Kernel trick in ML). Especially, try to summarize the key idea about the kernel trick.

4. (2 pt) We recall the Taylor series expansion of $\sin(z)$ up to order 10: $\sin(z) \sim z - \frac{z^3}{3!} + \frac{z^5}{5!} - \frac{z^7}{7!} + \frac{z^9}{9!}$
   (a) (0.25 pt) Compare the feature map $\phi_1(z) = (1, z, \sin(z), \cos(z))$ with a generic polynomial feature map of degree 10, denoted ϕ_2, in terms of expressiveness.
   (b) (0.75 pt) Assuming we know the signal to be of the form $y = \sin(x) + \beta x + \varepsilon$, where $x \in \mathbb{R}$ and ε is noise, which of the two feature maps is likely to fit the data better? Explain your reasoning, keeping in mind that N_{train}, the number of training data points, is finite.
   (c) (0.5 pt) How to interpret this result, in a Bayesian perspective?
   (d) (0.5 pt) Then, what about the case where $y = \sin(x/2) + \beta x + \varepsilon$?

5. (2.5 pt) In Figure 2 we compare 2 choices of the number of components to use in PCA, as pre-processing. After PCA, we apply a simple Perceptron (one layer, as seen in class). We are in a difficult sitation, where the intial input dimension is $D = 2048$. We are varying some hyper-parameter μ. For each choice of the hyper-parameter, we re-sample randomly the training data 10 times, performing a sort of Cross-Validation.
   (a) (0.5 pt) Looking at the fluctuations from run to run, would you say we have a lot or little training data ? And for test data ?

(b) (0.5 pt) For each of the 2 situations shown, what is the optimal choice of the hyper-parameter in terms of compromise between performance and overfitting (or robustness in general) ?

(c) (0.5 pt) Based on this, what is the best choice of number of components to keep ?

(d) (0.5 pt) Independently from considerations of performance, in which case do you trust more the estimation of the test error ?

(e) (0.5 pt) What is the mistake in our methodology of hyper-parameter tuning in this question ?

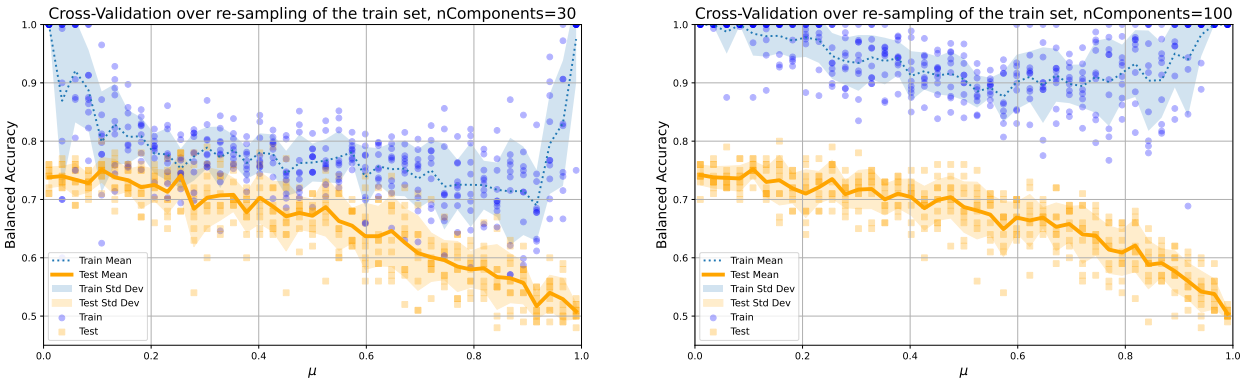


Figure 2: Figure related to question 5 (last question of this exercise).

2 (Set A) Maximum A Posteriori (MAP) (4 pts)

We want to compute the MAP estimation of the parameter λ for a random variable X that is expected to follow an exponential law, $\rho(x) = \lambda e^{-\lambda x}$, where we have an exponential prior for the parameter λ: $\rho(\lambda) = \tau e^{-\lambda \tau}$. We recall that the expectation value for an exponential random variable X is $\mathbb{E}[X] = 1/\lambda$.

We have access to a data set $\widetilde{X} = (x_1, \ldots, x_N)$ of empirical observations. We may refer to the empirical mean as $\overline{x} = \frac{1}{N}\sum_n^N x_n$ and to the empirical variance as $\overline{V} = \overline{\sigma}^2 = \frac{1}{N}\sum_n (x_n - \overline{x})^2$.

The event "I observe that the data is $\widetilde{X}$" can be written $X = \widetilde{X}$.

1. (2.5 pt) Do compute the MAP from the start, i.e. from its definition, recalling the fundamental steps (that are general to all distributions) as well as the computations that apply to this precise case.
2. (0.5 pt) Interpret (comment on) the limit $N \to \infty$.
3. (0.25 pt) Interpret (comment on) the limit $\tau \to 0$.
4. (0.25 pt) Interpret (comment on) the limit $\tau \to \infty$.
5. (0.5 pt) Why would a Gaussian prior be a weird choice for λ ? Think of the range of values the various variables are supposed to be in.

3 (Set B) Squared Hinge Loss (5.5 pts)

We have a dataset of input data $X = (\vec{x}_1, \dots, \vec{x}_N)$ and corresponding binary labels $Y = (y_1 \dots, y_N)$. The data is D dimensional, $\vec{x} \in \mathbb{R}^D$. The labels are encoded like this: $y_n \in \{-1, 1\}$. We now consider the squared hinge loss:

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \ell_{squared}(\hat{y}_n, y_n), \tag{1}$$

$$\ell_{squared}(\hat{y}_n, y_n) = \max(0, 1 - \hat{y}_n y_n)^2, \tag{2}$$

where $\hat{y}_n \in \mathbb{R}$ is the output of the model (for the input $\vec{x}_n$).

We denote $H(z)$ the Heaviside function (step function):

$$H(z) = \begin{cases} 1 & \text{if } z \geq 0, \\ 0 & \text{if } z < 0. \end{cases}$$

Note that it is equivalent to an indicatrix on a boolean condition: $\mathbb{I}_{a>b} = H(a - b)$.

If you feel more comfortable denoting the ground truth labels as y_n^{GT} or t_n, please do so (it will avoid forgetting the hat on $\hat{y}$ and then confusing the model's output and the ground truth labels). You can introduce other notations if you find them useful.

1. (0.5 pt) We will use a linear model, $\hat{y}_n = \vec{w} \cdot \vec{x}_n$. Compute $\vec{\nabla}_{\vec{w}} \hat{y}_n$, detailing the steps (expand the dot product and compute for instance $\frac{\partial}{\partial w_1} \hat{y}$).

2. (0.25 pt) What is the good choice of prediction function, $y_{predicted} = \text{Readout}(\hat{y}) = ?$.

3. (0.5 pt) Draw the function $\ell_{squared}(\hat{y}, y_n)$ as a function of $\hat{y}$ when $y_n = 1$. In particular, pay attention to the values this function takes at the points $\hat{y} = 0, 1, 2$.

4. (0.25 pt) Draw the function $\ell_{squared}(\hat{y}, y_n)$ as a function of $\hat{y}$ when $y_n = -1$. In particular, pay attention to the values this function takes at the points $\hat{y} = -2, -1, 0$.

5. (0.5 pt) What is $\frac{\partial \ell_{squared}}{\partial \hat{y}}(\hat{y}, y_n)$ when $y_n = 1$? You may introduce cases.

6. (0.5 pt) What is $\frac{\partial \ell_{squared}}{\partial \hat{y}}(\hat{y}, y_n)$ when $y_n = -1$? You may introduce cases.

7. **(1 pt)**[1] Taking advantage of the fact that $\forall n, y_n^2 = 1$ (because $(-1)^2 = (1)^2 = 1$); and of the fact that if $y_n = -1$, then $-1 = y_n$ (-1 can be substituted with y_n); and using the Heaviside function H, summarize these two results in a single formula, i.e. write down as explicitly as possible $\frac{\partial}{\partial \hat{y}} \ell_{squared}(\hat{y}, y_n)$ in a concise formula. Erratum: it was $\ell_{squared}$ and not ℓ_{hinge}.

8. **(0.5 pt)**[2] Using this model (We were talking about the model introduced in question 1, but basically this just means, deduce from the last question..), deduce what is $\vec{\nabla}_{\vec{w}} \mathcal{L}$.

9. (0.5 pt) What is the Gradient Descent update for a full epoch, then? Rewrite it using as many matrix operations as possible, in the spirit of vectorization (for leveraging numpy's or GPU's capabilities).

10. (0.5 pt) Do you see some intuitive connection between our squared hinge loss model and SVMs ?

11. (0.5 pt) Intuitively, what is the effect of using a square hinge instead of a hinge (not squared)? Think of the non separable case.

[1] (initially was at (0.5 pt))
[2] (initially was at (1 pt))

4 (Set B) PCA as minimum of reconstruction error (6 pts)

We are going to guide you through steps to re-derive what is the recipe for PCA, seen as a minimization of the (L^2) reconstruction error. Let us introduce some notations.

The input space where the data lives is $\mathbb{R}^D$ and is equipped with the canonical orthonormal basis vectors $(\vec{e}_d)_{d=1...D}$. As for any orthonormal basis, we have $\vec{e}_d \cdot \vec{e}_{d'} = \delta_{dd'} = \begin{cases} 1 & \text{if } d = d', \\ 0 & \text{if } d \neq d' \end{cases}$. The **PCA projection matrix (to be found) is denoted** $P \in \mathbb{R}^{p \times D}$. It is of rank p, and **its supplement**[3] **is denoted** $Q \in \mathbb{R}^{(D-p)\times D}$. Each row (line) of P corresponds to a vector $\vec{u}_i \in \mathbb{R}^D$ with $i = 1, \dots, p$. Similarly for Q, each line is a vector $\vec{u}_j$ with $j = p+1, \dots, D$. The concatenation of P and Q, which may be denoted U, is of rank D, or in other words, the vectors $(\vec{u}_i)_{i=1...D}$ are assumed to form an orthonormal basis of $\mathbb{R}^D$ (similarly to the e_d). Thus, any data point $\vec{x}_n$ can be written in the basis of the u's:

$$\vec{x}_n^{\text{new basis}} = U\vec{x}_n = \sum_{i=1}^{D} (\vec{x}_n \cdot \vec{u}_i)\vec{u}_i$$

This is just a change of basis, no information is lost there.

For simplicity, we will assume the data to be already **centered**: $\forall d, \frac{1}{N}\sum_n^N x_{nd} = 0$.

The **projected version of a data point** $\vec{x}_n$ is denoted $\vec{x}'_n$ and is obtained as:

$$\vec{x}'_n = P\vec{x}_n = \sum_{i=1}^{p} (\vec{x}_n \cdot \vec{u}_i)\vec{u}_i$$

The **reconstructed version of a data point** is $\vec{x}''_n = P^T\vec{x}'_n = P^TP\vec{x}_n$. The **reconstruction error, which is what we choose to minimize**, is defined as:

$$J = \frac{1}{N}\sum_n^N ||\vec{x}_n - \vec{x}''_n||_2^2$$

We provide a few extra tricks that will be useful at some point of the problem.
Note that for any vectors $\vec{a}, \vec{b}$, we have $(\vec{a}\cdot\vec{b})^2 = (\vec{a}\cdot\vec{b})(\vec{a}\cdot\vec{b})^T$, since for any scalar $\lambda \in \mathbb{R}$, $\lambda = \lambda^T$.
The product $(\vec{a}\cdot\vec{b})$ can be thought of as a matrix product for matrices $(1, D), (D, 1)$: $(\vec{a}\cdot\vec{b}) = \vec{a}^T\vec{b}$.
To enforce the constraint $c = 1$ in the minimization of some function $J(c)$, one introduces a lagrange multiplier $\lambda > 0$ and minimizes $J(c) + \lambda(1 - c)$.
In the Covariance matrix, one can use a denominator of $\frac{1}{N}$.

1. (1 pt) Think about how $\vec{x}''_n$ decomposes in the basis of the u's. In the end, its decomposition is very close to that of $\vec{x}'_n$. Write down J using the decomposition in the basis of the u's for both $\vec{x}_n$ and $\vec{x}''_n$.

2. (0.5 pt) Re-write it as a sum involving only Q and the original data points x_n. *(This question is useless for the rest of the exercise, it's a safety check)*

3. (1 pt) Using the form you found in question 1, prove that $J = \frac{1}{N}\sum_n \sum_{j=p+1}^{D} (\vec{x}_n \cdot \vec{u}_j)^2$ Erratum: I had forgotten the sum, $\sum_n$, in the formula. Remember that the u_i are orthonormal, and be especially rigorous (since we are giving you the result to find...).

4. (1 pt) Using the tricks mentionned above, simplfy J further. You have to obtain an expression where the covariance matrix of the data appears (you will denote it C)

5. (0.5 pt) Write down formally the constrained optimization problem that you are trying to solve, and re-write is as an unconstrained optimization problem.

6. (0.5 pt) Show that the extrema of this problem are the solutions of the problem $C\vec{u} = \lambda\vec{u}$. Hint: consider the problem for a generic and single u_j, since they have to be all orthogonal.

7. (0.5 pt) Plugging back the solution into J, simplify J further. You should find $J = \sum_{j=p+1}^{D} \lambda_j$.

   Even if you failed previous questions, try to do the next one:

8. (1 pt) Remember that we wanted to *minimize* J. Which are the eigenvectors we have to put in Q? In P?

[3] To be precise, the union of P and Q spans the whole vector space $\mathbb{R}^D$.