Exam of
Foundational Principles of Machine Learning (FPML)
December 15, 2023 – 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.**
- 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 21 (5+8+5+3) 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 Lecture related questions, all independent (5 points)

1. (1 pt) Explain the purpose of the validation set and of the test set.
   CORRECTION Short answer (we expect a bit more details): The test set allows to simulate the arrival of new, unseen data, i.e. estimate the performance of the model at generalization (to new data). The validation set is a way to simulate the test set, for various combinations of hyper-parameters, selecting the best combination, without compromising the novelty of the test set. We may over-fit the hyper-parameters on the validation set, hence the interest of Cross-Validation and of taking a separate test set, for the final evaluation.

2. (1.5 pt) Explain the typical (expected) relation between overfitting and large weights. Explain the purpose of Ridge regression, the idea behind it, and how/why it works (Ridge is the one with the term $+\lambda||w||_2^2$).
   CORRECTION Typically, overfitting corresponds to a situation with large weights, although you can have somehow large weights and not be overfitted. And of course there are situations where one can overfit without having very large weights. Still, limiting the weights' amplitude is a way to fight overfitting. Thus, adding a penalty $+\lambda||w||_2^2$ allows to limit them, and thus fight overfitting (also restraining expressivity, and thus potentially limiting the fitting power.)

3. (0.5 pt) Let's assume we have finite amound of data and a given model in mind, that we can esily train on the data. What experiment can we do to estimate whether more data would be helpful?
   CORRECTION We can plot the learning curve, i.e. performance (train and validation, typically) as a function of training set size (ideally, keeping the data present in the validation set constant.

4. (1 pt) Define the K-fold cross-validation procedure (the one seen in class). What are its benefits ? (give at least two)
   CORRECTION Cross-Validation (CV) allows to better estimate the error, because if we have e.g. $K = 5$ splits, we can take the average or median value, which is more reliable than a random pick of one of the 5 values. A second advantage is that it allows to compute a confidence interval or uncertainty estimate. When comparing hyperparameter choices, one should plot both the median or average value, AND the error bar associated to our uncertainty (taking the min and max values, or the standard deviation, over K values). This allows to assess whether variations in performance are statistically significant, or just noise from the train sampling.
   Here I did not give points for answers about CV allowing to fine-tune the hyper-parameters, because it can be done with simple trainval split. The point was to say why CV is better than a single train/val split, not re-explain hyper-param tuning.

5. (1 pt) When you have overfitting, what are the things you can do? (assuming we keep the same family of models). Cite as many possible solutions as you know, and each time, quickly explain your choice (1-2 lines per "solution" to overfitting).
   CORRECTION The things we can do to fight overfitting are:

   - *If possible, increase training set size (usually not possible)*
   - *Add some kind of regularization: L2 or L1, prossibly dropout, etc*
   - *Simplify the very architecture of the model: less layers in an MLP, less depth in a decision tree, etc*
   - *Compress the input data, using PCA or an other (un)supervised algorithm*

2 Hinge loss - to be written starting from this one (8 points)

We have a dataset of input data $X = \{\vec{x}_1, \ldots, \vec{x}_N\}$ and corresponding binary labels $Y = \{(y_1 \ldots, y_N\}$. The data is D dimensional, $\vec{x} \in \mathbb{R}^D$. The labels are encoded like this: $y_n \in \{-1, 1\}$. We want to consider the hinge loss:
ERRATUM: I forgot to write $\hat{y}_n$ and wrote $\hat{y}$ instead... sorry.

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

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

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

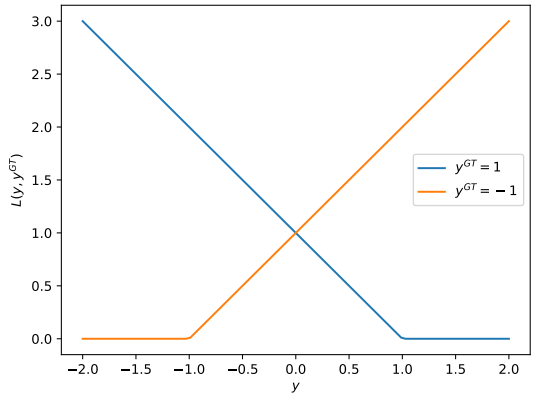
You can introduce other notations if you find them useful.

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).

1. (0.25 pt) Write the prediction function, $y_{predicted}$ = Readout($\hat{y}$) =?. **ERRATUM: to answer this question more easily, it's better to first read questions 6, i.e. know what is the model.**
   CORRECTION The Readout should be the sign() function, that sends $\mathbb{R}$ to $\{-1, +1\}$ based on whether $\hat{y} > 0$ or not.

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

3. (0.25 pt) Draw the function $\ell_{hinge}(\hat{y}, y_n)$ as a function of $\hat{y}$ when $y_n = -1$. In particular, pay attention to the values this function takes at the points $\hat{y} = -2, -1, 0$.
   CORRECTION See the figure:



4. (0.5 pt) What is $\frac{\partial \ell_{hinge}}{\partial \hat{y}}(\hat{y}, y_n)$ when $y_n = 1$? What is $\frac{\partial \ell_{hinge}}{\partial \hat{y}}(\hat{y}, y_n)$ when $y_n = -1$?
   CORRECTION
   For $y_n = 1$, we have two cases. If $1 - \hat{y} y_n > 0$ (looking at the picture, it's more directly seen as $\hat{y} < y_n$), then $\frac{\partial \ell_{hinge}}{\partial \hat{y}}(\hat{y}, y_n = 1) = -1 = -y_n$. Else, the function is 0 and its derivative is also 0.
   For $y_n = -1$, we also have two cases. If $1 - \hat{y} y_n > 0$ (looking at the picture, it's more directly seen as $\hat{y} > y_n$), then $\frac{\partial \ell_{hinge}}{\partial \hat{y}}(\hat{y}, y_n = 1) = +1 = -y_n$. Else, the function is 0 and its derivative is also 0.

5. (0.5 pt) Taking advantage of the fact that $\forall n, y_n^2 = 1$, and using the Heaviside function H, summarize these two results in a single formula, i.e. write down as explicitly as possible $\frac{\partial}{\partial \hat{y}} \ell_{hinge}(\hat{y}, y_n)$.
   CORRECTION For the case $y_n = 1$, we take the condition $\hat{y} < y_n$ and multiply both sides with y_n and obtain $\hat{y} y_n < 1$.
   For the case $y_n = -1$, we take the condition $\hat{y} > y_n$ and multiply both sides with y_n and obtain $\hat{y} y_n < 1$ again.
   The condition can for the derivative being non zero can thus be summarized as $1 - \hat{y} y_n > 0$. When the derivative is non-zero, it's equal to $-y_n$. So, we have: $\frac{\partial \ell_{hinge}}{\partial \hat{y}}(\hat{y}, y_n) = -y_n H(1 - \hat{y} y_n)$ (if the argument of H is less than 0, we obtain 0, and else, obtain $-y_n$).

6. (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 $\frac{\partial}{\partial w_1} \hat{y}$).
CORRECTION This was done in class. $\vec{w} \cdot \vec{x} = (w_1 x_1 + w_2 x_@ + \ldots + w_D x_D$. *The derivative wrt* w_1 *gives* x_1. *We generalize and conclude that* $\vec{\nabla}_{\vec{w}} \hat{y}_n = \vec{x}_n$

7. (1 pt) Using this model, deduce what is $\vec{\nabla}_{\vec{w}} \mathcal{L}$. *This question depends on most of the previous ones, but if you cannot complete it, you can still do a lot in the next questions.*
CORRECTION We just need to apply the chain rule:

$$\vec{\nabla}_{\vec{w}} \mathcal{L} = \nabla_{\vec{w}} \frac{1}{N} \sum_{n=1}^{N} \ell_{hinge}(\hat{y}_n, y_n) \tag{3}$$

$$= \frac{1}{N} \sum_{n=1}^{N} -y_n H(1 - \hat{y}_n y_n) \nabla_{\vec{w}} \hat{y} \tag{4}$$

$$= \frac{1}{N} \sum_{n=1}^{N} -y_n H(1 - \hat{y}_n y_n) \vec{x}_n \tag{5}$$

8. (0.5 pt) What is the Gradient Descent update for a full epoch, then? Re-write it using as many matrix operations as possible, in the spirit of vectorization (for leveraging numpy's or GPU's capabilites).
CORRECTION The GD update is $\vec{w} \mapsto \vec{w} - \eta \vec{\nabla}_{\vec{w}} \mathcal{L}$. *We can use a filter* $F = (1 - \hat{y}y > 0)$ *and write in numpy-style code:*

$$\vec{w} \mapsto \vec{w} + \eta \frac{1}{N} \sum_{n=1}^{N} y_n H(1 - \hat{y}_n y_n) \vec{x}_n \tag{6}$$

$$\mapsto \vec{w} + \eta Y[F] @ X[F] \tag{7}$$

Where @ is the matrix multiplication symbol (which performs the sum over n implictly, here). Thus the pseudo-code can be:

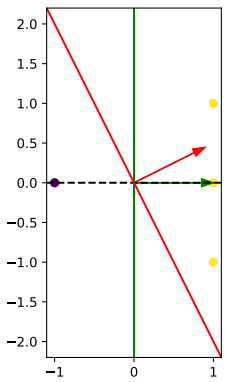
$$F = (1 - \hat{y}y > 0) \tag{8}$$

$$\vec{w} = \vec{w} + \eta Y[F] @ X[F] \tag{9}$$

But a code with more loops is also accepted, as soon as the filtering is correctly done (even with many ifs instead of the elegant condition $1 - \hat{y}_n y_n > 0$*).*

9. *(Independent question)* We have the following points: $X = \left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right)$ with labels $Y = (1, 1, 1, -1)$ and 2 possible planes $\mathcal{P}(\vec{w})$ that separate them: $\vec{w}_a = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \vec{w}_b = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 \\ 1 \end{pmatrix}$

   (a) (1 pt) Draw the 4 points (use different markers for the 2 classes) and the 2 planes in a system of coordinates. Note that the planes have both $b = 0$ implicitly, i.e. are defined only by their orientation (they are orthogonal to the $\vec{w}$ that defines them) and they go through O (the origin of your coordinate system).
   CORRECTION See the figure:



(b) (1 pt) Compute the Losses $\mathcal{L}(X, Y, \vec{w}_a)$, $\mathcal{L}(X, Y, \vec{w}_b)$ for both planes $\vec{w}_a, \vec{w}_b$. Which one is larger ?
CORRECTION We have very easily (from the plots also) that $\mathcal{L}(X, Y, \vec{w}_a) = 0$. It's going to be the smaller one.
For the other one, it's more tedious. We can save time by noticing from the plot that only two yellow points and the blue one will contribute (and the blue one and the yellow on the x-axis contribute the same, being at the same distance from the plane).
We take points one by one to evaluate their contribution. $X_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, X_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, X_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, X_4 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$.
X_1 *contributes for* $\frac{1}{N}(1 - \vec{w}_b \cdot X_1) = \frac{1}{4}(1 - \frac{1}{\sqrt{5}} \begin{pmatrix} 2 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} y_1) = \frac{1}{4}(1 - \frac{2}{\sqrt{5}})$. *Similarly,*
X_2 *contributes for* $\frac{1}{4}(1 - \frac{1}{\sqrt{5}})$
X_3 *contributes for* 0 *since* $\frac{1}{\sqrt{5}} \begin{pmatrix} 2 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{3}{\sqrt{5}} > 1$.
X_4 *contributes for* $\frac{1}{4}(1 - \frac{-2}{\sqrt{5}} y_4) = \frac{1}{4}(1 - \frac{2}{\sqrt{5}})$
Thus, $\mathcal{L}(X, Y, \vec{w}_b) = \frac{1}{4}(3 - \frac{2+1+2}{\sqrt{5}}) = \frac{3-\sqrt{5}}{4}$

(c) (0.5 pt) With or without the previous computation, which plane corresponds to the smaller loss ? Do you think other planes may have a Loss as good as that ? Why ? You can use geometrical arguments.
CORRECTION The plane w_a is the best, no other can be better (simply because 0 is the minimum of this loss, which cannot take negative values). But furthermore, geometrically we can see that any perturbation from w_a will generate a positive loss: as soon as the angle with the vertical line is non zero, we will have one of the two yellow points (with positive or negative x_2 coordinate) that will incur a change in the Loss. So w_a is the only optimal solution.

(d) (0.5 pt) Using the Perceptron Loss $\mathcal{L}_{perceptron} = \frac{1}{N} \sum_{n=1}^{N} max(0, -\vec{w} \cdot \vec{x}_n \ y_n)$, quickly write the result for the perceptron Loss, i.e. give the value of $\mathcal{L}_{perceptron}(X, Y, \vec{w}_a)$ and of $\mathcal{L}_{perceptron}(X, Y, \vec{w}_b)$ Why does the hinge loss seems better than the perceptron one ? *(No computation needed).*
CORRECTION For $\mathcal{L}_{perceptron}$, on both cases we obtain zero, although the first plane seems somehow better. Here it's allowed to do the computation, but simply looking at the figure and noticing that all points are well classified, or checking that all points are well classified by a quick computation of the ditance, for each one of them, is enough to score all the points.

10. (0.25 pt) If you remember the gradient of the classic perceptron update *(the one where the loss is written: $\mathcal{L}_{perceptron} = \frac{1}{N} \sum_{n=1}^{N} max(0, -\vec{w} \cdot \vec{x}_n \ y_n)$)*, what are the similarities and the difference(s) between the gradients of $\mathcal{L}$ and of $\mathcal{L}_{perceptron}$? *(Answering the previous question helps in this one, but it's not required to answer this one).*
CORRECTION It's all the same, except for the condition to appear in the sum. With the hinge loss, some well-classified points appear, if they are too close to the decision boundary.

11. (0.25 pt) Do you see some intuitive connection between our hinge loss model and SVMs ?
CORRECTION Yes !! Basically this is the SVM ! We are asking that points are distant enough from the plane, i.e. asking for a ***margin****!*

12. (0.5 pt) *(Independent question)* We now have the idea that the data comes from the following process:

$$y_n = \vec{w} \cdot \vec{x}_n + \varepsilon_n \tag{10}$$

where ε_n are noise terms, each ε_n is an i.i.d. Gaussian variable: $\rho(\varepsilon) = \mathcal{N}(0, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\varepsilon^2}{2\sigma^2}}$.
We also have a Gaussian prior on each component w_d of the vector $\vec{w}$, $\rho(w_d) = \mathcal{N}(0, \tau) = \frac{1}{\tau\sqrt{2\pi}} e^{-\frac{w_d^2}{2\tau^2}}$.
Can you guess the result of a MAP estimate ? *Note: you cannot and are not asked to provide the solution $\vec{w}_{\text{MAP}}$ explicitly, you should just write the problem that needs to be solved numerically to get $\vec{w}_{\text{MAP}}$, and notice to what it corresponds to.*
CORRECTION It will most likely be an L2-regularization term on w, i.e. the loss will have an additional term $\propto \frac{1}{\lambda} ||\vec{w}||^2$. BONUS: This is a classical computation that was done in class. We quickly sketch it here: $\vec{w}_{MAP} = argmax_{\vec{w}}(logP(X = x, Y = y|\vec{w}) + logP(\vec{w}))$.
Here $P(X = x, Y = y|\vec{w}) \sim \rho(\varepsilon) = \mathcal{N}(0, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\varepsilon^2}{2\sigma^2}}$. Recalling that $\varepsilon_n = \hat{y}_n - y_n$, we get:

$$\vec{w}_{MAP} = argmax_{\vec{w}} \left(\log P(X = x, Y = y|\vec{w}) + \log P(\vec{w}) \right) \tag{11}$$

$$= argmax_{\vec{w}} \left(const + \frac{1}{N} \sum_n \frac{-(\hat{y}_n - y_n)^2}{2\sigma^2} + \prod_d^D \log(\rho(w_d)) \right) \tag{12}$$

$$= argmax_{\vec{w}} \left(-\frac{1}{N} \left(\sum_n (\hat{y}_n - y_n)^2 \right) - \sum_d^D 2\sigma^2 \frac{w_d^2}{2\lambda^2} \right) \tag{13}$$

$$= argmin_{\vec{w}} \left(\frac{1}{N} \left(\sum_n (\hat{y}_n - y_n)^2 \right) + \sum_d^D 2\sigma^2 \frac{w_d^2}{2\lambda^2} \right) \tag{14}$$

$$= argmin_{\vec{w}} \left(MSE + \alpha ||\vec{w}||^2 \right) \tag{15}$$

Which ends up being the usual MSE term + a term in $\frac{\sigma^2}{\lambda^2}||\vec{w}||^2$. That is, the usual L2 regularization. The SVM with slack is indeed the regularized version of our hinge Loss linear model. Congrats, you found a model equivalent to SVM ! (that works also for non-linearly separable data).

3 Maximum A Posteriori (MAP) (5 points)

We have access to a data set $\widetilde{X} = \{x_1, \dots, x_N\}$ of empirical binary ($x_n \in \{0, 1\}$) observations that are assumed to be independent and identically distributed. We may refer to the empirical mean as $\overline{x} = \frac{1}{N}\sum_{n=1}^{N} x_n$.

We model the observations as random variables X_n following a Bernoulli law, $P(x) = p^x(1-p)^{1-x}$ (where x can take the values $\{0, 1\}$). We want to compute the MAP estimate of the parameter p assuming an exponential prior for p (which by definition, is positive), $\rho(p) = \lambda e^{-\lambda p}$, where $\lambda > 0$.
Reminder: The event "I observe the data is $\widetilde{X}$" can be written $X = \widetilde{X}$.
Reminder: $\mathbb{E}_{\rho_\lambda}[p] = \int_0^\infty p\ \lambda e^{-\lambda p}\mathrm{dp} = 1/\lambda$.
Reminder: $\log(a^b) = b\log(a)$ and $\log(u)' = \frac{u'}{u}$ (as e.g. $\log(x)' = \frac{1}{x}$).

1. (0.25 pt) What is the range of possible values for the empirical average $\overline{x} = \frac{1}{N}\sum_n x_n$?
   CORRECTION Since each $x_n \in \{0, 1\}$, $\overline{x} \in [0, 1]$.

2. We want to compute the MAP estimate $\hat{p}_{\mathrm{MAP}}$ from the start, i.e. from the definition, recalling the fundamental steps (that are general to all cases) as well as the computations that apply to this precise case. This can be broken into steps of increasing difficulty:

   (a) (1.5 pt) As a first step, you should do the general reasoning, and when you arrive at concrete computations, you may simplify by injecting $\lambda = 0$ into the formula to recover the MLE result: $\hat{p}_{MLE} = \overline{x}$. If you are lost with MAP things, you can also just do the MLE reasoning from the start to compute $\hat{p}_{MLE}$, and still score some points.
   CORRECTION Let's first make notations more precise. Here we have decided to assume that:
   - for all n, $P(X = x_n|p) = p^{x_n}(1-p)^{1-x_n}$. This is our model of the data, which is already a kind of prior assumption.
   - In addition, we have a prior on p: $\rho(p) = \lambda e^{-\lambda p}$.
   Since the data is i.i.d, $P(X = \widetilde{X}|p) = \prod_n P(x = x_n|p)$. The MAP estimate for p is thus:

$$
\begin{aligned}
p^*_{MAP} &= argmax_p(P(p|X = \widetilde{X})) && (16)\\
&= argmax_p\left(\frac{P(X = \widetilde{X}|p)P(p)}{P(X = \widetilde{X})}\right) && (17)\\
&= argmax_p\Big(\log P(X = \widetilde{X}|p) + \log P(p)\Big) && (18)\\
&= argmax_p\left(\log\left(\prod_n p^{x_n}(1-p)^{1-x_n}\right) + \log\left(\lambda e^{-\lambda p}\right)\right) && (19)\\
&= argmax_p\left(\left(\sum_n \log\left(p^{x_n}(1-p)^{1-x_n}\right)\right) + \log\lambda - \lambda p\right) && (20)
\end{aligned}
$$

   Where between 1st and 2nd line we used Bayes theorem, and between 2nd and third, used the fact that the evidence $P(X = \widetilde{X})$ does not depend on p, i.e. is a constant that is killed in the argmax, and used that log *is monotincally increasing, and thus can be applied as well.*
   The rest is just computation. Calling $\mathcal{L}$ the term inside the argmax, we want to find p such that $\nabla_p\mathcal{L} = 0$.

$$
\begin{aligned}
\nabla_p\mathcal{L} = 0 \Leftrightarrow 0 &= \nabla_p\left(\left(\sum_n \log\left(p^{x_n}(1-p)^{1-x_n}\right)\right) + \log\lambda - \lambda p\right)\\
\Leftrightarrow 0 &= \nabla_p\left(\sum_n \big(x_n\log(p) + (1-x_n)\log(1-p)\big) + \log\lambda - \lambda p\right)\\
\Leftrightarrow 0 &= \sum_n\left(x_n\frac{1}{p} + (1-x_n)\frac{1}{1-p}\right) + 0 - \lambda\\
\Leftrightarrow 0 &= \sum_n\left(x_n(1-p) + (-1)(1-x_n)p\right) - \lambda p(1-p)\\
\Leftrightarrow 0 &= \sum_n (x_n) - N.p - \lambda p(1-p)\\
\Leftrightarrow 0 &= \overline{x} - p - \frac{\lambda}{N}p(1-p)
\end{aligned}
$$

   Where we multiplied both sides with $p(1-p)$ at some point, and later divide by N to let $\overline{x}$ appear.
   We note that inserting $\lambda = 0$, we would obtain $p = \overline{x}$, which is what one expects from the MLE

(and since $\lambda \to 0$ corresponds to a flat prior, the exponential becoming very very flat in this limit, it matches our expectation (i.e. flat prior $\implies$ MAP becomes MLE)). In the general case, we have the equation:

$$0 = \overline{x} - p - \frac{\lambda}{N} p(1-p)$$
$$0 = p^2 \frac{\lambda}{N} - p\left(1 + \frac{\lambda}{N}\right) + \overline{x}$$

***Scoring**: if one computes the MLE correctly, they already score 1.5 points. If they can arrive at the correct MAP equation for p, they score the additional 1 pt of the next question:*

(b) (1 pt) Now, under the general case $\lambda \neq 0$ you should obtain $\hat{p}_{MAP}$. *(Reminder: The solutions (solving for x, assuming $a, b, c \in \mathbb{R}$) of the equation $ax^2 + bx + c = 0$ are $x_+ = \frac{-b+\sqrt{\Delta}}{2a}, x_- = \frac{-b-\sqrt{\Delta}}{2a}$, where $\Delta = b^2 - 4ac$. This can be summarized as $x_\pm = \frac{-b\pm\sqrt{\Delta}}{2a}$.)* Write down the solution $\hat{p}_{MAP}$ exaclty. It does not simplify much, I'm sorry about that.
CORRECTION We identify $a = \frac{\lambda}{N}, b = -\left(1 + \frac{\lambda}{N}\right), c = \overline{x}$ Thus $\Delta = \left(1 + \frac{\lambda}{N}\right)^2 - 4\frac{\lambda}{N}\overline{x}$ and

$$p_\pm = \frac{\left(1 + \frac{\lambda}{N}\right) \pm \sqrt{\left(1 + \frac{\lambda}{N}\right)^2 - 4\frac{\lambda}{N}\overline{x}}}{2\frac{\lambda}{N}}$$

(c) (0.5 pt) Formally, you should have found two solutions. How should we deal with that ? (Hint: what are the values allowed for $\hat{p}$?).
CORRECTION p_{MAP} must be inside $[0, 1]$. Thus, one of the solutions will be < 0 or > 1 and we will be able to exclude it.

(d) (1 pt) From the general formula found in (b), simplify in the limits $N \sim \infty$ and $\lambda \sim 0$. In particular, you may assume that $\frac{\lambda}{N}\overline{x} \ll \left(1 + \frac{\lambda}{N}\right)^2$, and use the development: $\sqrt{1-\varepsilon} \sim 1 - \frac{1}{2}\varepsilon$ (under the assumption $\varepsilon \sim 0$). You should obtain a simple formula where $\hat{p}_{MAP}$ depends on $\overline{x}, \lambda, N$.
CORRECTION When either $N \sim \infty$ or $\lambda \sim 0$, we have $\frac{\lambda}{N} \sim 0$.

$$\begin{aligned} p_\pm &= \frac{\left(1 + \frac{\lambda}{N}\right) \pm \sqrt{\left(1 + \frac{\lambda}{N}\right)^2 - 4\frac{\lambda}{N}\overline{x}}}{2\frac{\lambda}{N}} \\ &\sim \frac{\left(1 + \frac{\lambda}{N}\right) \pm \left(1 + \frac{\lambda}{N}\right)\sqrt{1 - 4\frac{\lambda}{N}\overline{x}\frac{1}{\left(1+\frac{\lambda}{N}\right)^2}}}{2\frac{\lambda}{N}} \\ &\sim \frac{\left(1 + \frac{\lambda}{N}\right) \pm \left(1 + \frac{\lambda}{N}\right)\left(1 - \frac{1}{2}4\frac{\lambda}{N}\overline{x}\frac{1}{\left(1+\frac{\lambda}{N}\right)^2}\right)}{2\frac{\lambda}{N}} \end{aligned}$$

Here in $\pm$ we must keep the minus sign, because the other one would give us a leading term in $\sim \frac{1+1}{2\frac{\lambda}{N}} \sim \infty$.

$$\begin{aligned} p_- &\sim \frac{\left(1 + \frac{\lambda}{N}\right) - \left(1 + \frac{\lambda}{N}\right)\left(1 - \frac{1}{2}4\frac{\lambda}{N}\overline{x}\frac{1}{\left(1+\frac{\lambda}{N}\right)^2}\right)}{2\frac{\lambda}{N}} \\ &\sim \frac{\left(1 + \frac{\lambda}{N}\right) - \left(1 + \frac{\lambda}{N}\right) + \left(\frac{1}{2}4\frac{\lambda}{N}\overline{x}\frac{1}{\left(1+\frac{\lambda}{N}\right)^2}\left(1 + \frac{\lambda}{N}\right)\right)}{2\frac{\lambda}{N}} \\ &\sim \frac{\overline{x}}{1 + \frac{\lambda}{N}} \end{aligned}$$

Which is a simple, rather familiar formula. p_{MAP} is slightly decreased wrt p_{MLE}, because the exponential prior favors small values for p.

(e) (0.25 pt) Without any computation, guess the zero-th order limit when $\lambda \to \infty$.
CORRECTION Without any computation, noticing that the average of an exponential law is $1/\lambda$, if $\lambda \to \infty$ we have a prior that is very peaked in 0. Thus we can already expect that $p_{MAP} \to 0$ in this limit.

(f) (bonus points) From the general formula, simplify in the limit $\lambda \sim \infty$. This is somewhat similar to the previous case. You should obtain a simple formula where $\hat{p}_{MAP}$ depends on $\overline{x}, \lambda, N$.
CORRECTION It's a bit like the previous one, but now we have to keep the other sign, and we obtain $p_{MAP} \sim \frac{N}{\lambda}\left(1 - \overline{x}\frac{N}{\lambda}\right)$. *Which, at first order, is equal to* 0.

3. (0.5) Comment on the meaning of the limits $N \sim \infty$, $\lambda \sim 0$ and $\lambda \sim \infty$.
CORRECTION Usual stuff, seen in previous exams:
$N \sim \infty$ *means lots of sample, prior is crushed by the data and we recover the MLE result.*
$\lambda \sim 0$ *is the flat prior, since the prior becomes uninformative, we recover the MLE result.*
$\lambda \sim \infty$ *is a prior that constrains strongly* p *to be close to* 0, *we obtain* $p_{MAP} \sim 0$.

4 Weighted PCA (3 pts)

We recall the PCA recipe.
We denote $\overline{\vec{x}}$ the empirical average of the data, feature by feature: $\overline{\vec{x}} = \frac{1}{N}\sum_n \vec{x}_n$
We denote C the empirical covariance matrix of the data: $C = \frac{1}{N-1}\sum_n \vec{x}_n \cdot \vec{x}_n^T$, which is thus a $D \times D$ matrix.
Diagonalizing C, we obtain the matrix of its eigenvectors U, that we can truncate (taking only D' columns in it) to obtain P.
The PCA projection formula is then $\Phi_{PCA}(\vec{x}) = (\vec{x} - \overline{\vec{x}})P$, where P is assumed to be a matrix of shape $D \times D'$, where $D' < D$ is the dimension after PCA, and D is the original dimension of each data point.

We want to define a balanced or weighted PCA where each sample is attributed an over-sampling coefficient α_n. For instance, if we have a minority class representing 10% of samples only, we may use $\alpha_n = 1/0.1$ for these samples, and $\alpha_n = 1/(0.9)$ for the other class (that has frequency of 90%). More generally, we may want to boost the importance of some samples in our PCA.

We want to find the exact recipe for weighted PCA that is equivalent to over-sampling by a factor α_n, but without the need to actually over-sample on the computer.

Let's denote $(.)^{(b)}$ the balanced version of something.

1. (1 pt) Write down the over-sampled version of the data average $\overline{\vec{x}}^{(b)}$. Try to write it in terms of the $\vec{x}_n$ and α_n.
   CORRECTION There is not much detail to provide, we can write that the sum becomes $\overline{\vec{x}}^{(b)} = \frac{1}{N}\sum_n \alpha_n \vec{x}_n$. *Intuitively, each sample is counted as many times as* α_n *(assuming that it is integer), it amounts to multiplying* x_n *with* α_n *and counting this n only once.*

2. (1 pt) How should we re-define P in the balanced case ? Define $P^{(b)}$. You may need to introduce $C^{(b)}$, the covariance matrix of the over-sampled data (in which case you need to define it, as you did for $\overline{\vec{x}}^{(b)}$.
   CORRECTION There is not much detail to provide, we can write that the sum becomes $C^{(b)} = \frac{1}{N-1}\sum_n \alpha_n \vec{x}_n \cdot \vec{x}_n^T$., *Intuitively, each term is counted as many times as* α_n *(assuming that it is integer), it amounts to multiplying the basic term* $\vec{x}_n \cdot \vec{x}_n^T$ *with* α_n *and counting this n only once.*

3. (0.5 pt) Can this be re-written as a feature map on the x_n ? Why ? Does it make sense to you ?
   CORRECTION No: in one case we multiply x_n *by* α_n*, but in the other case, we'd need to multiply each* x_n *with* $\sqrt{\alpha_n}$

4. (0.5 pt) Discuss in which sense this balanced or weighted PCA can be though of as an unsupervised method (as PCA) or as a supervised method.
   CORRECTION In a sense, it's still unsupervised because we do not use the labels y_n *directly, we just use some weights* α_n.
   But, in a sense it's now supervised because the α_n *would typically be defined based on* y_n *(or at least the inverse frequency of that class), so it's not a strictly unsupervised method.*