IFT-6390 Fundamentals of Machine Learning
Professor: Ioannis Mitliagkas

Homework 2 - Theoretical part

Devoir 2 - Partie Théorique

- This homework must be done and submitted to Gradescope and can be done in groups of at most 2 students. You are welcome to discuss with students outside of your group but the solution submitted by a group must be its own. Note that we will use Gradescope's plagiarism detection feature. All suspected cases of plagiarism will be recorded and shared with university officials for further handling.
  Ce devoir doit être déposé sur Gradescope et peut être fait en équipes de 2 étudiants maximum. Vous pouvez discuter avec des étudiants d'autres groupes mais les réponses soumises par le groupe doivent être originales. A noter que nous utiliserons l'outil de détection de plagiat de Gradescope. Tous les cas suspectés de plagiat seront enregistrés et transmis à l'Université pour vérification.

- Only one student should submit the homwork and add you should add your group member on the submission page on gradescope
  Seulement une étudiant doit soumettre les solutions et vous devez ajouter votre membre d'équipe sur la page de soumission de gradescope

- You need to submit your solution as a pdf file on Gradescope using the homework titled `(6390: GRAD) Theoretical Homework 2`.
  Vous devez soumettre vos solutions au format pdf sur Gradescope en utilisant le devoir intitulé `(6390: GRAD) Theoretical Homework 2`.

1. **Bias-Variance decomposition** Décomposition biais/variance [2 points]

   Consider the following data generation process: an input point x is drawn from an unknown distribution and the output y is generated using the formula

   $$y = f(x) + \epsilon,$$

where f is an unknown deterministic function and $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$. This process implicitly defines a distribution over inputs and outputs; we denote this distribution by p.

Given an i.i.d. training dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ drawn from p, we can fit the hypothesis h_D that minimizes the empirical risk with the squared error loss function. More formally,

$$h_D = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} (y_i - h(x_i))^2$$

where $\mathcal{H}$ is the set of hypotheses (or function class) in which we look for the best hypothesis/function.

The expected error[1] of h_D on a fixed data point (x', y') is given by $\mathbb{E}[(h_D(x') - y')^2]$. We will show that this error can be decomposed as a function of two meaningful terms:

- The bias, which is the difference between the expected value of hypotheses at x' and the true value $f(x')$. Formally,

$$\textit{bias} = \mathbb{E}[h_D(x')] - f(x')$$

- The variance, which is how far hypotheses learned on different datasets are spread out from their mean $\mathbb{E}[h_D(x')]$. Formally,

$$\textit{variance} = \mathbb{E}[(h_D(x') - \mathbb{E}[h_D(x')])^2]$$

Show that the expected prediction error on (x', y') can be decomposed into a sum of 3 terms: $(\textit{bias})^2$, *variance*, and a *noise* term involving ϵ. You need to justify all the steps in your derivation.

Considérons les données générées de la manière suivante: une donnée x est échantillonnée à partir d'une distribution inconnue, et nous observons la mesure correspondante y générée d'après la formule

$$y = f(x) + \epsilon,$$

où f est une fonction déterministe inconnue et $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$. Ceci définit une distribution sur les données x et mesures y, nous notons cette distribution p.

[1] Here the expectation is over random draws of the training set D of n points from the unknown distribution p. For example (and more formally): $\mathbb{E}[(h_D(x')] = \mathbb{E}_{(x_1,y_1)\sim p} \cdots \mathbb{E}_{(x_n,y_n)\sim p}\, \mathbb{E}[(h_{\{(x_1,y_1),\ldots,(x_n,y_n)\}}(x')]$.

Étant donné un ensemble d'entraînement $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ échantillonné i.i.d. à partir de p, on définit l'hypothèse h_D qui minimise le risque empirique donné par la fonction de coût erreur quadratique. Plus précisément,

$$h_D = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} (y_i - h(x_i))^2$$

où $\mathcal{H}$ est l'ensemble d'hypothèses (ou classe de fonction) dans lequel nous cherchons la meilleure fonction/hypothèse.

L'erreur espérée[2] de h_D sur un point donné (x', y') est notée $\mathbb{E}[(h_D(x') - y')^2]$. Nous allons montrer que cette erreur peut être décomposée en deux termes importants:

- Le biais, qui est la différence entre l'espérance de la valeur donnée par notre hypothèse en un point x' et la vraie valeur donnée par $f(x')$. Plus précisément,

$$\textit{biais} = \mathbb{E}[h_D(x')] - f(x')$$

- La variance, est une mesure de la dispersion des hypothèse apprises sur des ensemble de données différents, autour de la moyenne $\mathbb{E}[h_D(x')]$. Plus précisément,

$$\textit{variance} = \mathbb{E}[(h_D(x') - \mathbb{E}[h_D(x')])^2]$$

Montrez que l'erreur espérée pour un point donné (x', y') peut être décomposée en une somme de 3 termes: $(\textit{biais})^2$, *variance*, et un terme de *bruit* qui implique ϵ. Vous devez justifier toutes les étapes de dérivation.

2. **Feature Maps** Fonctions de transformation des données [8 points]

   In this exercise, you will design feature maps to transform an original dataset into a linearly separable set of points. For the following questions, if your answer is '*yes*', write the expression for the proposed transformation; and if your answer is '*no*', write a brief explanation. You are expected to provide explicit formulas for the feature maps, and these formulas should only use common mathematical operations.

[2] Ici l'espérance porte sur le choix aléatoire d'un ensemble d'entraînement D de n points tirés à partir de la distribution inconnue p. Par exemple (et plus formellement) : $\mathbb{E}[(h_D(x')] = \mathbb{E}_{(x_1,y_1)\sim p} \cdots \mathbb{E}_{(x_n,y_n)\sim p} \mathbb{E}[(h_{\{(x_1,y_1),\ldots,(x_n,y_n)\}}(x')]$.

Dans cet exercice, vous allez concevoir des fonctions de transformation depuis l'espace de traits caractéristiques original vers un espace où les données sont linéairement séparables. Pour les questions suivantes, si vous répondez '*oui*', écrivez l'expression de la transformation correspondante; et si votre réponse est '*non*', ajoutez une courte justification de votre réponse. Vous devez donner les formules explicites des transformations, et ces formules doivent utiliser uniquement des opérations mathématiques simples.

(a) [2 points] Consider the following 1-D dataset (Figure 1). Can you propose a 1-D transformation that will make the points linearly separable? Soit les données 1-D suivantes (Figure 1). Pouvez-vous proposer une transformation 1-D (i.e. vers un espace de dimension 1) qui rend les points linéairement séparables?

Figure 1:

(b) [2 points] Consider the following 2-D dataset (Figure 2). Can you propose a 1-D transformation that will make the data linearly separable? Soit les données 2-D suivantes (Figure 2). Pouvez-vous proposer une transformation 1-D qui rend les points linéairement séparables?

(c) [4 points] Using ideas from the above two datasets, can you suggest a 2-D transformation of the following dataset (as shown in Figure 3) that makes it linearly separable? If '*yes*', also provide the kernel corresponding to the feature map you proposed. En utilisant les idées que vous avez utilisées pour les deux questions précédentes, pouvez-vous proposer une transformation 2-D des données suivantes (Figure 3) qui les rendent linéairement séparables? Si votre réponse est '*oui*', donnez l'expression du noyau qui correspond à la transformation proposée.

3. **Optimization** Optimisation [10 points]

Assume a quadratic objective function of the form:

$$f(x) = \frac{1}{2}x^T Ax + x^T b + a,$$

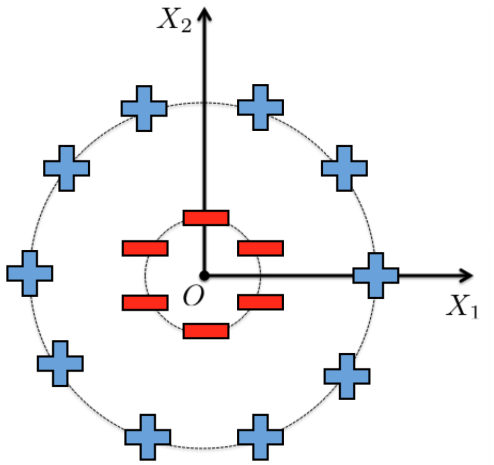

Figure 2:

where $x \in \mathbb{R}^d$, $b \in \mathbb{R}^d$, $a \in \mathbb{R}$ and A is a $d \times d$ symmetric, positive definite matrix. This means that the matrix A admits the eigendecomposition $A = U\Lambda U^T$, where $U \in \mathbb{R}^{d\times d}$ is an orthonormal matrix and $\Lambda \in \mathbb{R}^{d\times d}$ is a diagonal matrix. The i-th column vector of U, denoted by $u_i \in \mathbb{R}^d$, represents the i-th eigenvector of A. The i-th diagonal element of Λ, denoted by $\lambda_i \in \mathbb{R}$, represents the i-th eigenvalue of A. We will assume here that all eigenvalues are unique. Furthermore, without loss of generality, the eigenvectors and eigenvalues in the decomposition can be ordered in such a way that

$$\lambda_1 > \lambda_2 > \ldots > \lambda_d > 0.$$

Soit la fonction objectif suivante:

$$f(x) = \frac{1}{2}x^T Ax + x^T b + a,$$

où $x \in \mathbb{R}^d$, $b \in \mathbb{R}^d$, $a \in \mathbb{R}$ et A est une matrice $d\times d$ symétrique, positive définie. Ceci implique que la matrice A admet la décomposition en valeur propres $A = U\Lambda U^T$, où $U \in \mathbb{R}^{d\times d}$ est une matrice orthogonale et $\Lambda \in \mathbb{R}^{d\times d}$ est une matrice diagonale. La i-ème colonne de U, noté $u_i \in \mathbb{R}^d$, représente le i-ème vecteur propre de A. Le i-ème élément diagonal de Λ, noté $\lambda_i \in \mathbb{R}$, représente la i-ème valeur propre de A. Nous supposons que toutes les valeurs propres sont uniques. De plus,

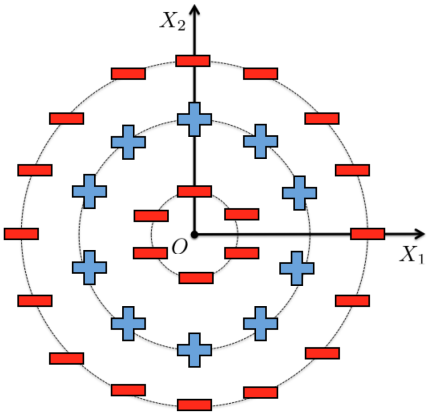

Figure 3:

sans perte de généralité, les vecteurs propres et valeurs propres peuvent être ordonnés de manière à avoir

$$\lambda_1 > \lambda_2 > \ldots > \lambda_d > 0.$$

(a) Find all of the stationary points of $f(x)$ analytically, i.e. through a closed-form expression (Justify). Donnez une expression analytique pour tous les points stationnaires de $f(x)$, en justifiant votre réponse.

(b) Which of those stationary points are minima, maxima and saddle-points? Give a mathematically rigorous explanation why. Lesquels de ces points sont des minima, maxima ou des points-selles? Expliquez pourquoi de manière rigoureuse.

(c) Find the location, x^*, and value, $f(x^*)$, of the global minimum. Trouvez le point x^*, et la valeur correspondante de la fonction objectif $f(x^*)$, au minimum global.

(d) Find the gradient of $f(x)$ at some point x. What are the dimensions of the gradient? Donnez le gradient de $f(x)$ au point x. Quelles sont ses dimensions?

(e) Show how the gradient descent update rule looks like in this case by substituting $f(x)$ with its quadratic form above. Use the following notation: x_0 represents our point at initialization, x_1 represents our point after one step, etc. Donnez l'expression de la

règle de mise à jour de l'algorithme de descente de gradient, en substituant $f(x)$ par son expression. Utilisez la notation suivante: x_0 est le point initial, x_1 est le point obtenu après une itération, etc.

(f) Consider the squared distance from optimum, $d(x_k) = \|x_k - x^*\|_2^2$. Find an exact expression (equality) of $d(x_k)$ that only depends on x_0 (not on other iterates x_i for $i > 0$), the number of iterations, k, as well as the eigenvectors, u_i, and eigenvalues, λ_i of A. Soit la distance à l'optimum (au carré) $d(x_k) = \|x_k - x^*\|_2^2$. Donnez l'expression exacte de $d(x_k)$ en fonction uniquement de x_0 (et pas des autres itérations x_i pour $i > 0$), du nombre d'itérations k, ainsi que des vecteurs propres u_i, et valeurs propres λ_i de A.

(g) Prove that there exist some assumptions on the hyperparameters of the algorithm, under which the sequence $d(x_k)$ converges to 0 as k goes to infinity. What are the exact necessary and sufficient conditions on the hyperparameters in order for $d(x_k)$ to converge to 0? Démontrez que sous certaines hypothèses sur les hyperparamètres de l'algorithme, la suite $d(x_k)$ converge vers 0 lorsque k tend vers l'infini. Quelles sont les conditions nécessaires et suffisantes pour que $d(x_k)$ converge vers 0?

(h) The distance that you computed above is said to converge to 0 at an exponential rate (some other research communities use the term linear rate for the same type of convergence). We often care about the asymptotic rate of convergence, defined for this squared distance as

$$\rho = \exp\left(\lim_{k\to\infty} \frac{1}{2k} \ln d(x_k)\right),$$

where $\ln(\cdot)$ denotes the natural logarithm. Keep in mind that this rate depends on both the objective function, but also on the choice of hyperparameters.

Find an expression of ρ that only depends on the eigenvalues of A and the hyperparameter values. On dit que la distance exprimée ci-dessus converge vers 0 à un taux exponentiel (d'autres communautés scientifiques utilisent plutôt le terme taux de convergence linéaire pour le même type de convergence). Nous nous intéressons souvent au taux de convergence asymptotique, défini dans le cas de l'erreur quadratique par

$$\rho = \exp\left(\lim_{k\to\infty} \frac{1}{2k} \ln d(x_k)\right),$$

où $\ln(\cdot)$ est le logarithme naturel. Gardez bien à l'esprit que ce taux de convergence dépend de la fonction objectif, mais aussi du choix d'hyperparamètres.
Donnez une expression de ρ qui dépend seulement des valeurs propres de A et des valeurs des hyperparamètres.

(i) Prove that, for any choice of hyperparameter values, there exist constants $k_0 \geq 0$ and $C > 0$ such that

$$d(x_k) \leq C\rho^{2k}, \quad \forall k > k_0.$$

Démontrez que pour toute valeur des hyperparamètres, il existe une constante $k_0 \geq 0$ et $C > 0$ tels que

$$d(x_k) \leq C\rho^{2k}, \quad \forall k > k_0.$$

(j) Based on the above, we gather that in order to get fast convergence, we need a small ρ value. Find a value for the hyperparameter(s) of gradient descent that achieves the fastest asymptotic convergence rate possible. En utilisant les questions précédentes, nous pouvons conclure que pour obtenir une convergence rapide, il faut que ρ prenne une petite valeur. Trouvez une valeur des hyperparamètres de l'algorithme de descente de gradient qui permet d'atteindre le meilleur taux de convergence asymptotique possible.

4. **Least Squares Estimator and Ridge Regression** **Estimateur par méthode des moindres carrés et régression ridge** [10 points]

(a) In the problem of linear regression, we are given n observations $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where each input $\mathbf{x}_i$ is a d-dimensional vector. Our goal is to estimate a linear predictor $f(.)$ which predicts y given $\mathbf{x}$ according to the formula

$$f(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\theta}, \tag{1}$$

Let $\mathbf{y} = [y_1, y_2 \ldots y_n]^\top$ be the $n \times 1$ vector of outputs and $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n]^\top$ be the $n \times d$ matrix of inputs. One possible way to estimate the parameter $\boldsymbol{\theta}$ is through minimization of the sum of squares. This is the least squares estimator:

$$\arg\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2. \tag{2}$$

Dans un problème de régression linéaire, on nous donne n observations $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, où chaque donnée d'entrée $\mathbf{x}_i$ est un vecteur à d dimensions. Nous cherchons à estimer un prédicteur linéaire $f(.)$ qui prédit y étant donné $\mathbf{x}$ à partir de la formule

$$f(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\theta}, \tag{3}$$

Nous notons par $\mathbf{y} = [y_1, y_2 \ldots y_n]^\top$ le vecteur $n \times 1$ de sorties et $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n]^\top$ la matrice $n \times d$ des données d'entrée. Une des méthodes pour estimer le paramètre $\boldsymbol{\theta}$ est la minimisation de la somme des erreurs au carré. C'est l'estimateur de moindres carrés:

$$\arg\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2. \tag{4}$$

i. Show that the solution of this minimization problem is given by Montrez que la solution de ce problème de minimisation est donnée par

$$\boldsymbol{\theta}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

ii. When will the matrix $\mathbf{X}^\top \mathbf{X}$ be invertible and when will it be non-invertible? Give your answer in terms of properties of the dataset. Quand la matrice $\mathbf{X}^\top \mathbf{X}$ sera-t-elle inversible, quand sera-t-elle non inversible? Donnez votre réponse en terme de propriétés de l'ensemble de données.

(b) A variation of the least squares estimation problem known as ridge regression considers the following optimization problem:

$$\arg\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_2^2 \tag{5}$$

where $\lambda > 0$ is a regularization parameter. The regularizing term penalizes large components in $\boldsymbol{\theta}$ which causes the optimal $\boldsymbol{\theta}$ to have a smaller norm.
Une des variations de la méthode des moindres carrés, connue sous le nom régression ridge, considère plutôt le problème d'optimisation suivant:

$$\arg\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_2^2 \tag{6}$$

où $\lambda > 0$ est un hyperparamètre de régularisation. Cette régularisation pénalise les composantes trop élevées de $\boldsymbol{\theta}$, ce qui force le $\boldsymbol{\theta}$ optimal à avoir une norme plus petite.

i. Derive the solution of the ridge regression problem. Do we still have to worry about the invertibility of $\mathbf{X}^\top\mathbf{X}$? Donnez la solution du problème de régression ridge. Doit-on toujours faire attention à l'inversibilité de $\mathbf{X}^\top\mathbf{X}$?

ii. Explain why the ridge regression estimator is likely to be more robust to issues of high variance compared with the least squares estimator. Expliquez pourquoi l'estimateur régression ridge sera probablement plus robuste à des problèmes de variance trop élevée que l'estimateur par moindres carrés.

iii. How does the value of λ affect the bias and the variance of the estimator? Quel sera l'effet de λ sur le biais et la variance de l'estimateur?

5. **Leave one out cross-validation** **Validation croisée "leave-one-out"** [10 points]

Let $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training sample drawn i.i.d. from an unknown distribution p. Recall that leave-one-out cross-validation (LOO-CV) on a dataset of size n is the k-fold cross-validation technique we discussed in class for the special case where $k = n - 1$. To estimate the risk (a.k.a. the test error) of a learning algorithm using D, LOO-CV consists in comparing each output y_i with the prediction made by the hypothesis returned by the learning algorithm trained on all the data except the ith sample (x_i, y_i).

Formally, if we denote by $h_{D\setminus i}$ the hypothesis returned by the learning algorithm trained on $D \setminus \{(x_i, y_i)\}$, the leave-one-out error is given by

$$\text{error}_{LOO} = \frac{1}{n}\sum_{i=1}^{n} \ell(h_{D\setminus i}(x_i), y_i)$$

where ℓ is the loss function.

In this exercise, we will investigate some interesting properties of this estimator.

Soit l'ensemble de données $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ échantillonné i.i.d. à partir d'une distribution inconnue p. Nous étudions la validation croisée "leave-one-out", qu'on pourrait traduire par "garder un exemple de côté", par la suite nous utiliserons la notation VCLOO. Pour rappel, la VCLOO sur un ensemble de données de taille n consiste à réaliser k validations croisées dans le cas particulier où $k = n - 1$.

Pour estimer le risque (c'est-à-dire l'erreur de test) d'un algorithme d'apprentissage en utilisant les données D, VCLOO consiste à comparer chaque sortie y_i avec la prédiction effectuée à l'aide du modèle obtenu en entraînant sur toutes les données sauf l'exemple (x_i, y_i).

Plus précisément, si on note $h_{D\setminus i}$ l'hypothèse obtenue par l'algorithme d'apprentissage entraîné sur les données $D \setminus \{(x_i, y_i)\}$, l'erreur leave-one-out est donnée par:

$$\text{erreur}_{LOO} = \frac{1}{n}\sum_{i=1}^{n} \ell(h_{D\setminus i}(x_i), y_i)$$

où ℓ est la fonction de perte.

Dans cet exercice, nous nous intéressons à certaines des propriétés de cet estimateur

Leave-one-out is unbiased **Leave-one-out est non biaisé**

(a) Recall the definition of the risk of a hypothesis h for a regression problem with the mean squared error loss function. Rappelez la définition du risque d'une hypothèse h pour un problème de régression avec la fonction de coût erreur quadratique

(b) Let D' denote a dataset of size $n-1$. Show that

$$\mathop{\mathbb{E}}_{D\sim p}[\text{error}_{LOO}] = \mathop{\mathbb{E}}_{\substack{D'\sim p,\\(x,y)\sim p}}[(y - h_{D'}(x))^2]$$

where the notation $D \sim p$ means that D is drawn i.i.d. from the distribution p and where h_D denotes the hypothesis returned by the learning algorithm trained on D. Explain how this shows that error_{LOO} is an (almost) unbiased estimator of the risk of h_D. En utilisant D' pour dénoter un ensemble de données de taille $n-1$, montrez que

$$\mathop{\mathbb{E}}_{D\sim p}[\text{erreur}_{LOO}] = \mathop{\mathbb{E}}_{\substack{D'\sim p,\\(x,y)\sim p}}[(y - h_{D'}(x))^2]$$

où la notation $D \sim p$ signifie que D est échantillonné i.i.d. à partir de la distribution p et où h_D est l'hypothèse obtenue par l'algorithme d'apprentissage sur les données D. Expliquez en quoi cela montre que erreur_{LOO} est un estimateur (presque) non-biaisé du risque de h_D.

Complexity of leave-one-out **Complexité de leave-one-out**
We will now consider LOO in the context of linear regression where inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are d-dimensional vectors. Similarly to exercise 4, we use $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{y} \in \mathbb{R}^n$ to denote the input matrix and the vector of outputs. Nous étudions maintenant LOO pour la régression linéaire où les données d'entrées $\mathbf{x}_1, \ldots, \mathbf{x}_n$ sont des vecteurs à d dimensions. Comme dans l'exercice 4, nous utilisons $\mathbf{X} \in \mathbb{R}^{n \times d}$ et $\mathbf{y} \in \mathbb{R}^n$ pour représenter la matrice des données d'entrée et le vecteur des sorties correspondantes.

(c) Assuming that the time complexity of inverting a matrix of size $m \times m$ is in $\mathcal{O}(m^3)$, what is the complexity of computing the solution of linear regression on the dataset D? En considérant que la complexité en temps pour inverser une matrice de taille $m \times m$ est en $\mathcal{O}(m^3)$, quelle sera la complexité du calcul de la solution de la régression linéaire sur l'ensemble de données D?

(d) Using $\mathbf{X}_{-i} \in \mathbb{R}^{(n-1) \times d}$ and $\mathbf{y}_{-i} \in \mathbb{R}^{(n-1)}$ to denote the data matrix and output vector obtained by removing the ith row of $\mathbf{X}$ and the ith entry of $\mathbf{y}$, write down a formula of the LOO-CV error for linear regression. What is the complexity of evaluating this formula? En notant $\mathbf{X}_{-i} \in \mathbb{R}^{(n-1) \times d}$ et $\mathbf{y}_{-i} \in \mathbb{R}^{(n-1)}$ la matrice des données d'entrées et le vecteurs des sorties obtenus en supprimant la ligne i de $\mathbf{X}$ et la composante i de $\mathbf{y}$, écrivez l'expression de l'erreur VCLOO pour la régression linéaire. Quelle est la complexité algorithmique du calcul de cette formule?

(e) It turns out that for the special case of linear regression, the leave-one-out error can be computed more efficiently. Show that in the case of linear regression we have

$$\text{error}_{LOO} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \mathbf{w}^{*\top} \mathbf{x}_i}{1 - \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_i} \right)^2$$

where $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ is the solution of linear regression computed on the whole dataset D. What is the complexity of evaluating this formula? Dans le cas particulier de la régression linéaire, l'erreur leave-one-out peut être calculée de manière plus efficace. Montrez que dans le cas de la régression linéaire, on a:

$$\text{erreur}_{LOO} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \mathbf{w}^{*\top} \mathbf{x}_i}{1 - \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_i} \right)^2$$

où $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ est la solution de la régression linéaire calculée sur tout l'ensemble de données D. Quelle est la complexité du calcul de cette expression?

6. **Multivariate Regression** **Régression multivariée** [10 points]

We consider the problem of learning a vector-valued function $f : \mathbb{R}^d \to \mathbb{R}^p$ from input-output training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ where each $\mathbf{x}_i$ is a d-dimensional vector and each $\mathbf{y}_i$ is a p-dimensional vector. We choose our hypothesis class to be the set of linear functions from $\mathbb{R}^d$ to $\mathbb{R}^p$, that is function satisfying $f(\mathbf{x}) = \mathbf{W}^\top \mathbf{x}$ for some $d \times p$ regression matrix $\mathbf{W}$, and we want to minimize the squared error loss function

$$J(\mathbf{W}) = \sum_{i=1}^{n} \|\mathbf{W}^\top \mathbf{x}_i - \mathbf{y}_i\|_2^2 \tag{7}$$

over the training data.

Let $\mathbf{W}^*$ be the minimizer of the empirical risk:

$$\mathbf{W}^* = \arg\min_{\mathbf{W} \in \mathbb{R}^{d \times p}} J(\mathbf{W}).$$

Nous étudions le problème de l'apprentissage d'une fonction vectorielle $f : \mathbb{R}^d \to \mathbb{R}^p$ à partir de données d'entrée/sortie $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ où chacun des $\mathbf{x}_i$ est un vecteur de dimension d et chacun des $\mathbf{y}_i$ est un vecteur de dimension p. Notre classe d'hypothèse est l'ensemble des applications linéaires de $\mathbb{R}^d$ dans $\mathbb{R}^p$, c'est-à-dire les fonctions qui s'écrivent $f(\mathbf{x}) = \mathbf{W}^\top \mathbf{x}$ où $\mathbf{W}$ est une matrice de taille $d \times p$, et nous cherchons à minimiser l'erreur quadratique

$$J(\mathbf{W}) = \sum_{i=1}^{n} \|\mathbf{W}^\top \mathbf{x}_i - \mathbf{y}_i\|_2^2 \tag{8}$$

sur les données d'entraînement.

Notons $\mathbf{W}^*$ le minimiseur du risque empirique

$$\mathbf{W}^* = \arg\min_{\mathbf{W} \in \mathbb{R}^{d \times p}} J(\mathbf{W}).$$

(a) Derive a closed-form solution for $\mathbf{W}^*$ as a function of the data matrices $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{Y} \in \mathbb{R}^{n \times p}$.
(hint: once you have expressed $J(\mathbf{W})$ as a function of $\mathbf{X}$ and

$\mathbf{Y}$, *you may find the matrix cookbook useful to compute gradients w.r.t. to the matrix* $\mathbf{W}$*)* Donnez une expression analytique de $\mathbf{W}^*$ en fonction des matrices $\mathbf{X} \in \mathbb{R}^{n \times d}$ et $\mathbf{Y} \in \mathbb{R}^{n \times p}$.
(suggestion: lorsque vous aurez exprimé $J(\mathbf{W})$ *en fonction de* $\mathbf{X}$ *et* $\mathbf{Y}$*, vous pourrez vous aider du matrix cookbook pour obtenir le gradient par rapport à la matrice* $\mathbf{W}$*)*

(b) Show that solving the problem from the previous question is equivalent to independently solving p independent classical linear regression problems (one for each component of the output vector), and give an example of a multivariate regression task where performing **independent** regressions for each output variables is not the best thing to do. Montrez que le problème de la question précédente peut être résolu de manière équivalente en cherchant la solution de p problèmes classiques de régression linéaire (un pour chaque composante des vecteurs de sortie), et donnez un exemple de tâche de régression multivariée où résoudre **indépendamment** des problèmes de régression ne serait pas la meilleure chose à faire.

(c) The low rank regression algorithm addresses the issue described in the previous question by imposing a low rank constraint on the regression matrix $\mathbf{W}$. Intuitively, the low rank constraint encourages the model to capture linear dependencies in the components of the output vector.

Propose an algorithm to minimize the squared error loss over the training data subject to a low rank constraint on the regression matrix $\mathbf{W}$:

$$\min_{\mathbf{W} \in \mathbb{R}^{d \times p}} J(\mathbf{W}) \qquad \text{s.t. } \operatorname{rank}(\mathbf{W}) \leq R.$$

(hint: There are different ways to do that. You could for example leverage the fact that $\operatorname{rank}(\mathbf{W}) \leq R$ *if and only if there exists* $\mathbf{A} \in \mathbb{R}^{d \times R}$ *and* $\mathbf{B} \in \mathbb{R}^{R \times p}$ *such that* $\mathbf{W} = \mathbf{AB}$*.)*
L'algorithme de régression rang faible corrige le problème décrit dans la question précédente, en imposant la contrainte que la matrice de régression $\mathbf{W}$ soit de rang faible. Intuitivement, la contrainte de rang faible encourage le modèle à prendre en compte les dépendances linéaires entre les composantes des vecteurs de sortie.
Proposez un algorithme qui minimise l'erreur quadratique sur l'ensemble d'entraînement sous une contrainte de rang faible sur

la matrice $\mathbf{W}$:

$$\min_{\mathbf{W} \in \mathbb{R}^{d \times p}} J(\mathbf{W}) \qquad \text{tel que } \operatorname{rank}(\mathbf{W}) \leq R.$$

(suggestion: Il y a plusieurs manières d'aborder ce problème. Vous pourriez par exemple utiliser le fait que $\operatorname{rank}(\mathbf{W}) \leq R$ *si, et seulement si, il existe des matrices* $\mathbf{A} \in \mathbb{R}^{d \times R}$ *et* $\mathbf{B} \in \mathbb{R}^{R \times p}$ *tel que* $\mathbf{W} = \mathbf{AB}$*.)*