IFT-3395/6390 Fundamentals of Machine Learning
Professors: Guillaume Rabusseau / Ioannis Mitliagkas

Homework 1 - Practical component

Devoir 1 - Partie pratique

- This homework must be done and submitted to Gradescope individually. You are welcome to discuss with other students but the solution and code you submit must be your 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 fait et envoyé sur Gradescope individuellement. Vous pouvez discuter avec d'autres étudiants mais les réponses et le code que vous soumettez doivent être les vôtres. 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.

- The practical part should be coded in python (you can use the numpy and matplotlib libraries) and all code will be submitted as a python file to Gradescope. To enable automated code grading you should work off of the template file given in this homework folder. Do not modify the name of the file or any of the function signatures of the template file or the code grading will not work for you. You may, of course, add new functions and any regular python imports.
  La partie pratique doit être codée en python (avec les librairies numpy et matplotlib), et envoyée sur Gradescope sous fichier python. Pour permettre l'évaluation automatique, vous devez travailler directement sur le modèle donné dans le répertoire de ce devoir. Ne modifiez pas le nom du fichier ou aucune des fonctions signatures, sinon l'évaluation automatique ne fonctionnera pas. Vous pouvez bien sur ajouter de nouvelles fonctions et importations python

- Any graphing, charts, or practical report parts should be submitted in a pdf to Gradescope. For the report it is recommended to use a Jupyter notebook, writing math with MathJax and export as pdf.

You may alternatively write your report in LaTeX; LyX; Word. In any case, you should export your report to a pdf file that you will submit. You are of course encouraged to draw inspiration from what was done in lab sessions.
Les figures, courbes et parties pratiques du rapport doivent être envoyées au format pdf sur Gradescope. Pour le rapport il est recommandé d'utiliser un Jupyter notebook, en écrivant les formules mathématiques avec MathJax et en exportant vers pdf. Vous pouvez aussi écrire votre rapport en LaTeX; LyX; Word. Dans tout les cas, exportez votre rapport vers un fichier pdf que vous enverrez. Vous êtes bien sur encouragés à vous inspirer de ce qui a été fait en TP.

You should work off of the template file `solution.py` in the project and fill in the basic numpy functions using numpy and python methods
Vous devez travailler sur le modèle `solution.py` du répertoire et compléter les fonctions basiques suivantes en utilisant numpy et python

Parzen with soft windows (kernels)

Parzen avec fenêtre continue

In this Homework we will use Iris as a toy dataset. It contains 150 samples (one for each row), each with 4 features (the 4 first columns) and one label in {1,2,3} (the 5th column). It is recommended you download it here and then test your code by importing it like this :
Pour ce devoir nous utiliserons le dataset Iris comme exemple. Il contient 150 points (un par rangée), chacun avec 4 attributs (les 4 premières colonnes) et un label dans {1,2,3} (la 5ème colonne). Il est recommandé de le télécharger ici puis de tester votre code en l'important comme ceci :

```
import numpy as np
iris = np.loadtxt('iris.txt')
```

When the answer template in solution.py has "iris" as an argument, you may assume that this argument is the dataset in numpy format. Your function should use this argument to perform computations, not a version of the dataset that you loaded by yourself.
Quand le modèle de réponse dans solution.py contient "iris" comme argument, vous pouvez considérer que cet argument est le jeu de données au

format numpy. Votre fonction doit utiliser cet argument pour faire les calculs, et pas une version du jeu de données que vous auriez chargé vous-même

1. Undergraduates 4 pts Graduates 4 pts
   Question. Write functions that take that dataset as input and return the following statistics:
   Écrivez des fonctions qui prennent le jeu de données en entrée et retournent les statistiques suivantes:

   (a) Q1.feature_means : An array containing the empirical means of each feature, from all examples present in the dataset. Make sure to maintain the original order of features.
   Q1.feature_means : Un tableau contenant pour chaque attribut la moyenne de sa valeur sur l'ensemble des points du jeu de données. Faites attention à maintenir l'ordre des attributs.
   e.g. : Q1.feature_means(iris) = $[\mu_1,\mu_2,\mu_3,\mu_4]$

   (b) Q1.covariance_matrix : A 4×4 matrix that represents the empirical covariance matrix of features on the whole dataset.
   Q1.covariance_matrix : Une matrice 4×4 donnant la matrice de covariance empirique des attributs sur le jeu de données entier

   (c) Q1.feature_means_class_1 : An array containing the empirical means of each feature, but only from examples in class 1. The possible classes in the iris dataset are 1, 2, and 3.
   Q1.feature_means_class_1 : Un tableau contenant pour chaque attribut la moyenne de sa valeur sur l'ensemble des points dont le label est 1. Les classes possibles dans iris sont 1, 2 et 3.
   e.g. : Q1.feature_means_class_1(iris) = $[\mu_1,\mu_2,\mu_3,\mu_4]$

   (d) Q1.covariance_matrix_class_1 : A 4×4 matrix that represents the empirical covariance matrix of features, but only from examples in class 1.
   Q1.covariance_matrix_class_1 : Une matrice 4×4 donnant la matrice de covariance empirique des attributs sur l'ensemble des points dont le label est 1.

2. Undergraduates 1 pts Graduates 1 pts
   Question. Implement Parzen with hard window parameter h. Use the standard Euclidean distance on the original features of the dataset. Your answer should have the following behavior :
   f = HardParzen(h) initiates the algorithm with parameter h

f.train(X, Y) trains the algorithm, where X is a $n \times m$ matrix of n training samples with m features, and Y is an array containing the n labels. The labels are denoted by integers, but the number of classes in Y can vary.
f.compute_predictions(X_test) computes the predicted labels and return them as an array of same size as X_test. X_test is a $k \times m$ matrix of k test samples with same number of features as X. This function is called only after training on (X, Y). If a test sample x has no neighbor within window h, the algorithm should choose a label at random by using **draw_rand_label(x, label_list)**, a function that is provided in the **solution.py** file, where label_list is the list of different classes present in Y, and x is the array of features of the corresponding point.
Implémentez la méthode de Parzen avec une fenêtre discrète de paramètre h. Utilisez la distance euclidienne classique sur les attributs du jeu de données. Votre solution doit avoir le comportement suivant :
f = HardParzen(h) initialise l'instance de l'algorithme avec le paramètre h
f.train(X, Y) entraine l'algorithme, où X est une matrice $n \times m$ de n exemples d'entrainement avec m attributs, et Y est un tableau contenant les n labels. Les labels sont representés par des entiers, mais le nombre de classes dans Y peut varier.
f.compute_predictions(X_test) calcule les labels prédits et les retournent sous forme d'un tableau de même taille que X_test. X_test est une matrice $k \times m$ de k exemples de test avec le même nombre d'attributs que X. Cette fonction n'est appelée qu'après s'être entrainé sur (X, Y). Si un exemple de test x n'a pas de voisin dans la fenêtre de paramètre h, l'algorithme doit choisir un label au hasard en utilisant **draw_rand_label(x, label_list)**, une fonction qui est fournie dans le fichier **solution.py**, avec label_list la liste des différentes classes présentes dans Y, et x le vecteur des attributs du point correspondant.

3. Undergraduates 10 pts Graduates 5 pts
Question. Implement Parzen with a soft window. We will use a radial basis function (RBF) kernel (also known as *Gaussian* kernel) with parameter σ. Use the standard Euclidean distance on the original features of the dataset. Please refer to the slides from the second week for the definition. The structure of your solution should be the same as in the previous question, but you will never need to draw a

label at random with **draw_rand_label(x, label_list)**. The class name is **SoftRBFParzen**.
Implémentez la méthode de Parzen avec une fenêtre continue. Nous utiliserons comme kernel une fonction à base radiale (RBF) (aussi connue comme *Gaussian* kernel) avec paramètre σ. Utilisez la distance Euclidienne sur les attributs du jeu de données. Veuillez consulter les slides de la deuxième semaine pour la définition. La structure de votre solution devrait être la même que pour la question précédente, mais vous n'aurez jamais besoin de choisir un label au hasard en utilisant **draw_rand_label(x, label_list)**. Le nom de la classe est **SoftRBFParzen**

4. Undergraduates 7 pts Graduates 5 pts
   Question. Implement a function **split_dataset** that splits the Iris dataset as follows:

   - A training set consisting of the samples of the dataset with indices which have a remainder of either 0, 1, or 2 when divided by 5
   - A validation set consisting of the samples of the dataset with indices which have a remainder of 3 when divided by 5.
   - A test set consisting of the samples of the dataset with indices which have a remainder of 4 when divided by 5.

   For instance the element of index 14 (in the original dataset) should be part of the test set because the remainder of 14 divided by 5 is 4. Do not use random splitting for this exercise (even though it is generally a very good idea). The function should take as input the dataset and return the three sets as a tuple (train, validation, test), where each element of the tuple is a matrix with 5 columns (the 4 features and the labels, kept in the same order).
   Implémentez une fonction **split_dataset** qui sépare le jeu de données Iris de la façon suivante:

   - Un ensemble d'entrainement composé des points du jeu de données dont l'indice a un reste de 0, 1 ou 2 quand divisé par 5
   - un ensemble de validation composé des points du jeu de données dont l'indice a un reste de 3 quand divisé par 5
   - un ensemble de test composé des points du jeu de données dont l'indice a un reste de 4 quand divisé par 5

Par exemple l'élément dont l'indice est 14 (dans le jeu de données original) doit faire partie de l'ensemble de test, car le reste de 14 divisé par 5 est 4. N'utilisez pas de séparation aléatoire pour cette exercice (bien que ce soit habituellement une très bonne idée). La fonction doit prendre en entrée le jeu de données, et retourner les trois sous-ensembles sous forme d'un tuple (train, validation, test), où chaque élement du tuple est une matrice à 5 colonnes (les 4 attributs et les labels, gardés dans le même ordre).

5. Undergraduates 15 pts Graduates 10 pts
   Question. Implement two functions **ErrorRate.hard_parzen** and **ErrorRate.soft_parzen** that compute the error rate (i.e. the proportion of missclassifications) of the HardParzen and SoftRBFParzen algorithms. The expected behavior is as follows :
   test_error = ErrorRate(x_train, y_train, x_val, y_val) initiates the class and stores the training and validation sets, where x_train and x_val are matrices with 4 feature columns, and y_train and y_val are arrays containing the labels.
   test_error.hard_parzen(h) takes as input the window parameter h and returns as a float the error rate on x_val and y_val of the HardParzen algorithm that has been trained on x_train and y_train.
   test_error.soft_parzen(σ) works just like with Hard Parzen, but with the SoftRBFParzen algorithm.

   Then, include in your report a single plot with two lines:

   (a) Hard Parzen window's classification error on the validation set of iris, when trained on the training set (see question 4) for the following values of h:

   $$h \in \{0.001, 0.01, 0.1, 0.3, 1.0, 3.0, 10.0, 15.0, 20.0\}$$

   (b) RBF Parzen's classification error on the validation set of iris, when trained on the training set (see question 4) for the following values of σ:

   $$\sigma \in \{0.001, 0.01, 0.1, 0.3, 1.0, 3.0, 10.0, 15.0, 20.0\}$$

   The common x-axis will represent either h or σ. Always label your axes and lines in the plot!

Give a detailed discussion of your observations.
Implémentez deux fonctions **ErrorRate.hard_parzen** et **ErrorRate.soft_parzen** qui calculent le taux d'erreur (i.e. la proportion de mauvaises classifications) des algorithms HardParzen et SoftRBGParzen. Le comportement attendu est le suivant :
test_error = ErrorRate(x_train, y_train, x_val, y_val) initialise la classe et sauvegarde les ensembles d'entrainement et de validation, où x_train et x_val sont des matrices d'attributs à 4 colonnes, et y_train et y_val sont des tableaux contenant les labels.
test_error.hard_parzen(h) prends en entrée le paramètre h et retourne sous forme de nombre réel le taux d'erreur sur x_val et y_val de l'algorithm HardParzen qui a été entrainé sur x_train et y_train.
test_error.soft_parzen(σ) fait la même chose mais pour SoftRBF-Parzen.
Ensuite, incluez dans votre rapport un graphe unique avec deux lignes :

(a) l'erreur de classification de Hard Parzen sur l'ensemble de validation de iris, après avoir été enrtrainé sur l'ensemble d'entrainement (voir question 4) pour les valeurs suivantes de h:

$$h \in \{0.001, 0.01, 0.1, 0.3, 1.0, 3.0, 10.0, 15.0, 20.0\}$$

(b) l'erreur de classification de Soft Parzen sur l'ensemble de validation de iris, après avoir été enrtrainé sur l'ensemble d'entrainement (voir question 4) pour les valeurs suivantes de σ:

$$\sigma \in \{0.001, 0.01, 0.1, 0.3, 1.0, 3.0, 10.0, 15.0, 20.0\}$$

L'axe des abscisses représentera à la fois h et σ. Attention à annoter les axes et lignes du graphe!
Donnez une discussion détaillée de vos observations.

6. Undergraduates 6 pts Graduates 5 pts

Question. Implement a function **get_test_errors** that uses the evaluated validation errors from the previous question to select h^* and σ^*, then computes the error rates on the test set. The value h^* is the one (among the proposed set in question 5) that results in the smallest validation error for Parzen with hard window, and σ^* is the parameter (among the proposed set in question 5) that results in the smallest validation error for Parzen with RBF.

The function should take as input the dataset and split it using question 4. The expected output is an array of size 2, the first value

being the error rate on the test set of Hard Parzen with parameter h^* (trained on the training set), and the second value being the error rate on the test set of Soft RBF Parzen with parameter σ^* (trained on the training set).
Implémentez une fonction **get_test_errors** qui utilise les erreurs de classification sur l'ensemble de validation calculées à la question précédente pour sélectionner h^* et σ^*, puis qui calcule le taux d'erreur sur l'ensemble de test. La valeur h^* est celle (parmis celles proposées à la question 5) qui minimise l'erreur de Hard Parzen sur l'ensemble de validation, et σ^* est le paramètre (parmis ceux proposés à la question 5) qui minimise l'erreur de Soft RBF Parzen sur l'ensemble de validation.
La fonction doit prendre en argument le dataset et le séparer en utilisant la question 4. Le résultat attendu est un tableau de taille 2, dont la première valeur est le taux d'erreur sur l'ensemble de test de Hard Parzen avec paramètre h^*, et la deuxième est le taux d'erreur sur l'ensemble de test de Soft RBF Parzen avec paramètre σ^*.

7. Undergraduates 6 pts Graduates 5 pts
   Question. Include in your report a discussion on the running time complexity of these two methods. How does it vary for each method when the hyperparameter h or σ changes? Why ?
   Ajoutez à votre rapport une discussion sur la complexité temporelle (temps de calcul) de ces deux méthodes. Comment cela évolue-t-il pour chaque méthode quand les paramètres h ou σ changent ? Pourquoi ?

8. Undergraduates [bonus] pts Graduates 5 pts
   Question. Implement a random projection (Gaussian sketch) map to be used on the input data:

   Your function **random_projections** should accept as input a feature matrix X of dimension $n \times 4$, as well as a 4×2 matrix A encoding our projection.

   Define $p : x \mapsto \frac{1}{\sqrt{2}} A^T x$ and use this random projection map to reduce the dimension of the inputs (feature vectors of the dataset) from 4 to 2.

   Your function should return the output of the map p when applied to X, in the form of a $n \times 2$ matrix.
   e.g. $random\_projections(X_{n\times 4}, A_{4\times 2}) = X^{proj}_{n\times 2}$

Implémentez une projection aléatoire (Gaussian sketch) pour l'utiliser sur les données d'entrée :
Votre fonction **random_projections** doit accepter en entrée une matrice d'attributs X de dimension $n \times 4$, et une matrice A de dimension 4×2 encodant la projection.
En définissant $p : x \mapsto \frac{1}{\sqrt{2}} x^T A$, utiliser cette projection aléatoire pour réduire la dimension des entrées (attributs des points du jeu de données) de 4 à 2.
Votre fonction doit retourner le résultat de p appliqué à X, sous forme d'une matrice $n \times 2$.
e.g. $random\_projections(X_{n\times 4}, A_{4\times 2}) = X^{proj}_{n\times 2}$

9. Undergraduates [bonus] pts Graduates 10 pts
   Question. Similar to Question 5, compute the validation errors of Hard Parzen classifiers trained on 500 random projections of the training set, for

$$h \in \{0.001, 0.01, 0.1, 0.3, 1.0, 3.0, 10.0, 15.0, 20.0\}$$

   The validation errors should be computed on the projected validation set, using the same matrix A. To obtain random projections, you may draw A as 8 independent variables drawn uniformly from a gaussian distribution of mean 0 and variance 1.
   You can for example store these validation errors in a 500×9 matrix, with a row for each random projection and a column for each value of h.

   Do the same thing for RBF Parzen classifiers, for

$$\sigma \in \{0.001, 0.01, 0.1, 0.3, 1.0, 3.0, 10.0, 15.0, 20.0\}$$

   Plot and include in your report in the same graph the average values of the validation errors (over all random projections) for each value of h and σ, along with error bars of length equal to $0.2\times$ the standard deviations.

   How do your results compare to the previous ones?
   De même que dans la question 5, calculez les erreurs de validation du Hard Parzen entrainé sur 500 projections aléatoires de l'ensemble d'entrainement, pour

$$h \in \{0.001, 0.01, 0.1, 0.3, 1.0, 3.0, 10.0, 15.0, 20.0\}$$

Les erreurs de validation devraient être calculées sur la projection de l'ensemble de validation, en utilisant la même matrice A. Pour obtenir des projections aléatoires, vous pouvez tirer A comme 8 variables indépendantes tirées aléatoirement d'une distribution gaussienne de centre 0 et de variance 1.
Vous pouvez par exemple representer ces erreurs de validation dans une matrice 500×9, avec une rangée par projection aléatoire et une colonne par valeur de h.
Faites la même chose pour Soft RBF Parzen, pour

$$\sigma \in \{0.001, 0.01, 0.1, 0.3, 1.0, 3.0, 10.0, 15.0, 20.0\}$$

Tracez et incluer dans votre rapport sur la même courbe les valeurs moyennes de l'erreur de validation (moyenner sur toutes les projections aléatoires) pour chaque valeur de h et σ, avec des intervals d'erreur (sous forme graphique de barres) de longueur égales à $0.2\times$ la deviation standard.
Comment vos résultats se comparent-ils aux précédents ?