Travaux pratiques de
Machine learning
Cycle pluridisciplinaire d'études supérieures
Université Paris sciences et lettres

Joon Kwon

vendredi 15 mai 2020

Ce TP doit être rédigé et sera évalué. Il doit être rendu sous la forme d'un notebook Jupyter où l'on pourra ajouter des blocs de texte (choisir *Markdown* au lieu de *Code* dans le sélecteur du type de cellule). Il est possible de travailler en binôme. Les noms doivent apparaître dans le document ainsi que dans le nom du fichier de la façon suivante : `TP-NOMDEFAMILLE1_NOMDEFAMILLE2.ipynb`. Il doit être envoyé le jeudi 28 mai 2020 au plus tard sur Microsoft Teams.

On travaille à nouveau avec le jeu de données sur le cancer du sein. On rappelle qu'il s'agit de prédire si une tumeur est maligne (0) ou bénigne (1) à partir de caractéristiques géométriques observées.

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
X = data.data
y = data.target
```

Question 1. — Séparer le jeu de données en un échantillon d'apprentissage (`X_train`,`y_train`) et un échantillon de test (`X_test`,`y_test`).

Nous allons travailler dans ce TP avec des SVM. Nous pouvons les entraîner de la façon suivante :

```
from sklearn.svm import LinearSVC
svm = LinearSVC(C=1).fit(x_train,y_train)
print('Score: ',svm.score(x_test,y_test))
```

où l'argument `C` correspond à l'inverse de l'hyperparamètre de régularisation λ vu dans le cours ; autrement dit plus C est grand, et moins la régularisation est importante.

QUESTION 2. — Sur l'échantillon d'apprentissage, utiliser la validation croisée ainsi que les courbes de validation afin de choisir une valeur pour l'hyperparamètre `C`. Entraîner à nouveau le prédicteur, qu'on appellera `svm`, sur tout l'échantillon d'apprentissage avec la valeur de `C` choisie. Donner son score sur l'échantillon de test. Définir également un `array` `y_test_predict` contenant les prédictions du prédicteur obtenu sur l'échantillon de test.

On peut afficher la matrice de confusion du prédicteur obtenu sur l'échantillon de test comme suit.

```
from sklearn.metrics import confusion_matrix
confusion_matrix_test =
 ↪  confusion_matrix(y_test,y_test_predict)
print(confusion_matrix_test)
```

À partir de la matrice de confusion, nous pouvons construire plusieurs quantités qui vont nous permettre de mesurer la performance des prédicteurs de façon plus fine qu'avec le simple score. Dans problème présent, il s'agit de détecter si une tumeur est maligne (0) ou bénigne (1), et il est donc bien plus grave prédire qu'une tumeur maligne est bénigne, que l'inverse. *On souhaite construire un prédicteur qui détecte 95% des tumeurs malignes, sans toutefois signaler toutes les tumeurs comme malignes.* Nous introduisons à présent plusieurs notions portant sur les prédictions d'un prédicteur sur un échantillon donné, et qui sont définies par rapport à l'étiquette y_*, dite *positive*, c'est-à-dire celle qu'il est le plus important de détecter (ici $y_* = 0$ qui correspond à la tumeur maligne).

— Un *vrai positif* (*true positive*) est une prédiction égale à y_*, et correcte. On note TP le nombre de vrais positifs.
— Un *faux positif* (*false positive*) est une prédiction égale à y_*, et incorrecte. On note FP le nombre de faux positifs.

— Un *vrai négatif* et un *faux négatif* sont définis de façon similaire, et on note TN et FN leurs nombres respectifs.
— La *précision* est la proportion de vrai positifs parmi les prédictions égales à y_*. Autrement dit :

$$\text{précision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

— Le *rappel* (*recall*, ou *true positive rate*) est la proportion de positifs correctement prédits. Autrement dit :

$$\text{rappel} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

— Le *taux de faux positifs* (*false positive rate*) est la proportion de négatifs incorrectement prédits. Autrement dit :

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}}.$$

QUESTION 3. — Calculer la précision et le rappel à partir des coefficients de la matrice de confusion.

`scikit-learn` fournit des fonctions qui permettent de calculer directement la précision et le rappel. Cela nous permet de vérifier les résultats obtenus à la question précédente.

```
from sklearn.metrics import recall_score, precision_score
print('Recall:
 ↪  ',recall_score(y_test,y_test_predict,pos_label=0))
print('Precision:
 ↪  ',precision_score(y_test,y_test_predict,pos_label=0))
```

On notera que l'argument `pos_label=0` permet ici de spécifier que l'étiquette considérée comme *positive* est 0 (tumeur maligne).

QUESTION 4. — Définir une fonction `false_positive_rate`, qui prend trois arguments : `y_true` qui contient les étiquettes réelles d'un échantillon, `y_predict` qui contient les étiquettes prédites par un prédicteur, et `pos_label` qui spécifie l'étiquette considérée comme positive ; et qui renvoie le taux de faux positifs. Calculer le taux de faux positifs donnés sur l'échantillon de test par le prédicteur `svm` obtenu à la question 2.

On souhaite donc construire un prédicteur dont le rappel est d'au moins 0.95. Notons $(\hat{w}, \hat{b}) \in \mathbb{R}^{30} \times \mathbb{R}$ les coefficients du prédicteur construit à la question 2. Le prédicteur lui-même s'écrit, puisque les étiquettes sont ici $\mathcal{Y} = \{0, 1\}$:

$$f_{\hat{w},\hat{b}}(x) = \begin{cases} 1 & \text{si } \langle \hat{w}, x \rangle + \hat{b} > 0 \\ 0 & \text{si } \langle \hat{w}, x \rangle + \hat{b} < 0. \end{cases}$$

Nous allons considérer une famille de prédicteurs modifiés, de la forme :

$$f^{(\tau)}_{\hat{w},\hat{b}}(x) = \begin{cases} 1 & \text{si } \langle \hat{w}, x \rangle + \hat{b} > \tau \\ 0 & \text{si } \langle \hat{w}, x \rangle + \hat{b} < \tau. \end{cases}$$

où $\tau \in \mathbb{R}$ est un *seuil* à choisir. On voit que ces prédicteurs vont avoir tendance, par rapport à $f_{\hat{w},\hat{b}}$, à plus prédire 0 ou plus prédire 1, selon la valeur de τ.

Question 5. — Définir une fonction `modified_predictor` qui prend en argument un `array` contenant les entrées x_i d'un échantillon, ainsi qu'une valeur `tau` (pour τ), et qui renvoie un `array` contenant les prédictions $f^{(\tau)}_{\hat{w},\hat{b}}(x_i)$. On pourra utiliser la fonction `svm.decision_function` contenue dans le prédicteur `svm`, qui prend un argument un `array` contenant les entrées x_i d'un échantillon, et qui renvoie un `array` contenant les valeurs $\langle \hat{w}, x_i \rangle + \hat{b}$.

Pour chaque valeur de τ, le prédicteur correspondant $f^{(\tau)}_{\hat{w},\hat{b}}$ donne sur l'échantillon d'apprentissage un certain rappel et un certain taux de faux positifs. Pour nous aider à choisir parmi cette famille un prédicteur ayant un rappel d'au moins 0.95 et un faible taux de faux positifs, nous allons tracer la *courbe ROC* : il s'agit de tracer les points de coordonnées (rappel, FPR) (calculés sur l'échantillon d'apprentissage) lorsqu'on fait varier τ (notons que τ n'est pas explicitement représenté sur cette courbe).

Question 6. — À quoi ressemble la courbe ROC d'un bon (resp. mauvais) prédicteur ?

Question 7. — Quelles sont les valeurs minimale et maximale pour τ qu'il convient de considérer ? Se donner une grille de 100 valeurs à essayer pour τ. Pour chacune de ces valeurs, calculer sur l'échantillon d'apprentissage le rappel et le taux de faux positifs du prédicteur correspondant $f^{(\tau)}_{\hat{w},\hat{b}}$. Tracer la courbe ROC correspondante.

Question 8. — Parmi les valeurs de τ considérées, déterminer celle donnant sur l'échantillon d'apprentissage un rappel supérieur à 0.95 et le plus faible taux de faux positifs. Pour ce seuil τ, calculer le score, le rappel, la précision et le taux de faux positifs du prédicteur correspondant $f^{(\tau)}_{\hat{w},\hat{b}}$ sur l'échantillon de test.