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

Joon Kwon

vendredi 29 mai 2020

On considère un jeu de données immobilières.

```
import numpy as np

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

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y)
```

Il s'agit de prédire le prix de biens immobiliers en fonction d'informations concernant l'environnement. Il s'agit donc d'un problème de régression. On pourra consulter la description du jeu de données pour davantage d'informations.

Question 1. — Entraîner une régression linéaire aux moindres carrés (en appelant `logreg` le prédicteur) et calculer son erreur de test manuellement (la fonction `logreg.score()` ne correspond pas au risque empirqiue, mais au score R^2 introduit ci-après).

Il est difficile à la seule vue d'un risque empirique (erreur de test) de juger s'il s'agit d'un bon ou d'un mauvais résultat. On définit donc en régression une mesure alternative de la qualité d'un prédicteur appelé *score* R^2 (ou *coefficient de détermination*). Soit $S = (X_i, Y_i)_{i\in[n]}$ un échantillon et f un prédicteur. On note $\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$ et on pose

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - f(X_i))^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(Y_i - f(X_i))^2}{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \bar{Y})^2},$$

où on reconnaît au numérateur le risque empirique $\frac{1}{n}\sum_{i=1}^{n}(Y_i - f(X_i))^2$ et au dénominateur un estimateur classique de la variance $\frac{1}{n}\sum_{i=1}^{n}(Y_i - \bar{Y})^2 \simeq \mathrm{Var}(Y_1)$ (si on suppose que les exemples de l'échantillon sont i.i.d.). Ainsi, le score R^2 peut être approximativement vu comme une transformation simple du risque empirique :

$$R^2 \simeq 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(Y_i - f(X_i))^2}{\mathrm{Var}\, Y_1}.$$

De plus, on voit qu'un prédicteur parfait a un score $R^2 = 1$, tandis qu'un prédicteur stupide qui prédit toujours une valeur proche de la moyenne des Y_i (c'est-à-dire $f(x) \simeq \mathbb{E}[Y] \simeq \bar{Y}$) a un score R^2 proche de 0. Il est implémenté par défaut dans `scikit-learn` pour les prédicteurs de régression.

```
print('Score R²: ',linreg.score(X_test,y_test))
```

Nous allons à présent faire appel à la classe `LassoCV` qui permet d'effectuer une régression linéaire aux moindres carrés avec régularisation LASSO et qui sélectionne automatiquement le paramètre de régularisation par validation croisée.

```
from sklearn.linear_model import LassoCV
```

QUESTION 2. — Utiliser `LassoCV` en spécifiant l'argument `alphas` qui doit être une liste ou un `array` contenant les valeurs à essayer pour l'hyperparamètre de régularisation (on prendra une large grille de valeurs), ainsi que l'argument `cv` qui correspond au paramètre de la validation croisée. On pourra si besoin consulter la documentation. Afficher le score R^2 sur l'échantillon de test du prédicteur obtenu.

De façon similaire, nous pouvons utiliser `RidgeCV` pour effectuer une régression linéaire aux moindres carrés avec régularisation Ridge avec un paramètre de régularisation automatiquement sélectionné par validation croisée.

```
from sklearn.linear_model import RidgeCV
```

Question 3. — Utiliser `RidgeCV` avec une large grille de valeurs pour l'hyperparamètre de régularisation. Afficher le score R^2 sur l'échantillon de test du prédicteur obtenu.

Nous allons à présent considérer la régression Ridge avec des noyaux gaussiens :

$$K_\gamma^{\mathrm{rbf}}(x, x') = e^{-\gamma \|x - x'\|_2^2}, \qquad x, x' \in \mathbb{R}^d.$$

Par exemple :

```
from sklearn.kernel_ridge import KernelRidge
kr = KernelRidge(kernel='rbf',alpha=.001,gamma=.001)
print(kr.fit(X_train,y_train).score(X_test,y_test))
```

où `alpha` correspond au paramètre de régularisation $\lambda > 0$, et `gamma` au paramètre $\gamma > 0$ présent dans le noyau. Étant donné ce choix de noyau, il y a donc deux hyperparamètres à choisir. Lorsqu'on sélectionne les valeurs de plusieurs hyperparamètres simultanément, il est recommandé de limiter la taille de chaque grille de valeurs, mais de réitérer si besoin le procédé avec de nouvelles grilles contenant des valeurs proches de celles précédemment sélectionnées.

Question 4. — Utiliser `GridSearchCV` pour sélectionner des valeurs pour les deux hyperparamètres. Afficher le score du prédicteur obtenu.

On souhaite utiliser à présent un noyau polynomial, de la forme :

$$K_{\gamma,r,\delta}^{\mathrm{poly}}(x, x') = (\gamma \langle x, x' \rangle + r)^\delta, \qquad x, x' \in \mathbb{R}^d.$$

Par exemple :

```
krpoly =
 ↪  KernelRidge(kernel='poly',alpha=0.01,gamma=0.01,degree=3,coef0=1)
print(krpoly.fit(X_train,y_train).score(X_test,y_test))
```

où `degree` et `coef0` correspondent respectivement aux paramètres $\delta \geqslant 1$ (entier) et $r \in \mathbb{R}$ du noyau. Il y a donc, avec `alpha` et `gamma`, quatre hyperparamètres à choisir. Pour alléger les calculs, nous allons ne considérer pour r que la valeur `coef0=1`.

Question 5. — Sélectionner des valeurs pour les trois hyperparamètres restants à l'aide de `GridSearchCV` et afficher le score du prédicteur obtenu.