

Exercice 1

Que faire avec ChatGPT (et ses amis)?

5 points

1. Pouvez-vous écrire un programme Python permettant d'estimer une masse stellaire basée sur le jeu de données PARSEC (propriétés fondamentales des populations stellaires) en utilisant un lasso adaptatif par composants sans utiliser scikitlearn, c'est à dire en résolvant le problème d'optimisation suivant :

$$\arg \min_{\beta \in \mathbf{R}^p} \frac{1}{2} \|y - X\beta\|^2 + \lambda \sum_{j=1}^p w_j |\beta_j|$$

- a) Commentez la la réponse de Chat GPT ci-dessous et si nécessaire, proposez des améliorations au code proposé.

Réponse de ChatGPT en italique :

Où, voici un exemple de programme Python qui peut faire cela :

```
import numpy as np
import pandas as pd

# Chargez le jeu de donn\ees PARSEC
data = pd.read_csv('PARSEC.csv')

# S\electionnez les caract\eristiques suivantes :
# - masse : masse stellaire (M)
# - logTeff : logarithme de la temp\erature effective (K)
# - logg : logarithme de la gravit\e de surface (cgs)
# - [Fe/H] : m\etallicit\e (dex)
X = data[['mass', 'logTeff', 'logg', '[Fe/H]']]

# S\electionnez la variable cible : - masse : masse stellaire (M)
y = data['mass']

# Initialisez les coefficients
alpha = 0.1
w = np.ones(X.shape[1], dtype=float)
coef = np.ones(X.shape[1], dtype=float)

# Effectuez une descente de coordonn\ees pour adapter le mod\ele Lasso
adaptatif par composants
while True:
# Mise \a jour des poids
for j in range(X.shape[1]):
shrink = np.maximum(0., 1-alpha*np.linalg.norm(X[:, j]*coef)**2/np.linalg.
norm(coef))
w[j] = w[j] * shrink

# Mise \a jour des coefficients
for i in range(X.shape[0]):
# Calculez le r\esidu
r = y[i] - np.dot(X[i], coef)

# Calculez le gradient
g = np.dot(X[i].T, r)

# Calculez le facteur de r\etr\ecissement
shrink = np.maximum(0., 1-alpha*np.linalg.norm(g)**2/np.linalg.norm(coef))

# Mettez \a jour les coefficients
coef = shrink * coef - alpha * w[i] * g

# V\erifiez si la convergence est atteinte
diff = np.linalg.norm(coef - coef_old)
coef_old = coef
if diff < 1e-5:
break

# Obtenez la masse stellaire pr\evue
predicted_mass = np.dot(coef, [1.0, 4.0, 4.5, 0.0])

print('Masse stellaire pr\evue :', predicted_mass)
```

Use code with caution.

Ce programme chargera le jeu de données PARSEC, sélectionnera les caractéristiques et la variable cible, initialisera les coefficients pour le modèle Lasso adaptatif par composants, effectuera une descente de coordonnées pour adapter le modèle et fera des prédictions sur les données de test. La masse stellaire prévue sera imprimée sur la console.

Voici un aperçu du code :

La première ligne importe les bibliothèques numpy et pandas.

La deuxième ligne charge le jeu de données PARSEC à partir d'un fichier CSV nommé PARSEC.csv.

Les lignes suivantes sélectionnent les caractéristiques suivantes :

- masse : masse stellaire (M)
- $\log T_{\text{eff}}$: logarithme de la température effective (K)
- $\log g$: logarithme de la gravité de surface (cgs)
- Fe/H : métallicité (dex)

La variable X est créée pour stocker ces caractéristiques.

La ligne suivante sélectionne la variable cible, qui est la masse stellaire (M), et la stocke dans la variable y .

Initialisez les coefficients avec toutes les valeurs définies sur 1.0

Initialisez les poids avec toutes les valeurs définies sur 1.0

Effectuez une descente de coordonnées pour adapter le modèle Lasso adaptatif par composants :

Pour chaque itération :

- Mettez à jour les poids
- Mettez à jour les coefficients
- Vérifiez si la convergence est atteinte

Obtenez la masse stellaire prévue pour une étoile avec une masse de 1,0 masse solaire, une température effective de 4,0 Kelvin, une gravité de surface de 4,5 et une métallicité de 0,0

Imprimez la masse stellaire prévue Fin de la réponse de ChatGPT

Exercice 2

Revue de code

4 points

1. Si vous aviez UNE recommandation à faire à cet autre auteur de ces lignes de code concernant sa méthodologie en apprentissage automatique (machine learning), quelle serait-elle ?

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score

n_samples, n_features, n_classes = 200, 10000, 2
rng = np.random.RandomState(42)
X = rng.standard_normal((n_samples, n_features))
y = rng.choice(n_classes, n_samples)
X_selected = SelectKBest(k=25).fit_transform(X, y)
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, random_state=42)
gbc = GradientBoostingClassifier(random_state=1)
gbc.fit(X_train, y_train)
GradientBoostingClassifier(random_state=1)
y_pred = gbc.predict(X_test)
accuracy_score(y_test, y_pred)
```

2. On a demandé à un data scientist d'estimer les paramètres d'un modèle linéaire à l'aide de la ridge régression sur des données normalisées avec un paramètre $\lambda = 0,1$,

$$\widehat{\beta}_r = \arg \min_{\beta \in \mathbf{R}^p} \frac{1}{2} \|y - X\beta\|^2 + \frac{\lambda}{2} \|\beta\|^2$$

Le code suivant a été produit (par chatGPT). Qu'en pensez vous ?

```

import numpy as np
from sklearn.datasets import make_regression

def ridge_regression(X, y, l):
    """
    Compute the ridge regression coefficients.
    Parameters:
    X (numpy.ndarray): Design matrix of shape (n_samples, n_features).
    y (numpy.ndarray): Target vector of shape (n_samples,).
    l (float): Regularization parameter (lambda).
    Returns:
    numpy.ndarray: Ridge regression coefficients of shape (n_features,).
    """
    n_samples, n_features = X.shape
    # Compute the ridge regression coefficients using the closed-form solution
    #  $w = (X^T X + \lambda I)^{-1} * X^T y$ 
    I = np.eye(n_features)
    w = np.linalg.inv(X.T @ X + l * I) @ X.T @ y
    return w

# Example usage: generate some random data for demonstration
np.random.seed(0)
n_samples = 100
n_features = 3
X, y = make_regression(n_samples=n_samples, n_features=n_features, noise=1)
l = 0.1
# Compute ridge regression coefficients
ridge_coefs = ridge_regression(X, y, l)
print("Ridge regression coefficients:", ridge_coefs)

```

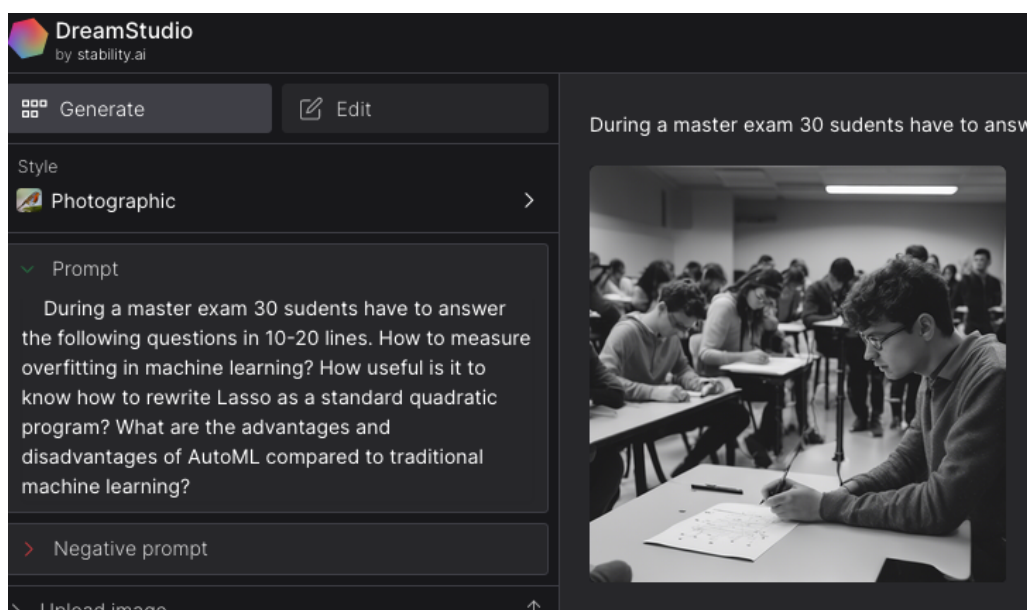
Exercice 3

Êtes-vous meilleur que chatGPT ?

5 points

Comme chatGPT, répondez aux questions suivantes en 10 à 20 lignes.

1. Comment mesurer le sur-apprentissage en apprentissage automatique ?
2. En quoi est-ce utile de savoir réécrire le Lasso comme un programme quadratique standard ?
3. Quels sont les avantages et les inconvénients d'AutoML par rapport au machine learning traditionnel ?
4. Comment Sklearn permet-il de sélectionner les meilleurs algorithmes pour une tâche donnée ?
5. Quels sont les avantages et les inconvénients de l'utilisation d'un modèle de langage génératif (comme Bard ou chatGPT) pour un *data scientist* ?



1. la fused regression est la solution du problème suivant :

$$\widehat{\beta}_f = \arg \min_{\beta \in \mathbf{R}^p} \frac{1}{2} \|y - X\beta\|^2 + \lambda \sum_{j=1}^{p-1} (\beta_j - \beta_{j+1})^2$$

- a) Proposez une matrice D pour que le problème précédent s'écrive :

$$\widehat{\beta}_f = \arg \min_{\beta \in \mathbf{R}^p} \frac{1}{2} \|y - X\beta\|^2 + \lambda \|D\beta\|_2^2$$

- b) Proposez une formule mathématique permettant de calculer $\widehat{\beta}_f$ la solution du problème en fonction de X , y et λ , puis complétez le code suivant pour calculer la solution de la fused regression.

```
import numpy as np
from sklearn.datasets import make_regression

def fused_regression(X, y, l):
    """
    Compute the fused regression coefficients.
    Parameters:
    X (numpy.ndarray): Design matrix of shape (n_samples, n_features).
    y (numpy.ndarray): Target vector of shape (n_samples,).
    l (float): Regularization parameter (lambda).
    Returns:
    numpy.ndarray: fused regression coefficients of shape (n_features,).
    """
    n_samples, n_features = X.shape
    # Compute the fused regression coefficients using the closed-form solution
    D = ...
    beta_f = ...
    return beta_f
```

2. Presque le fused Lasso. L'idée maintenant est de forcer des coefficients β_j d'avoir la même valeur. Pour ce faire on se propose de résoudre le problème suivant :

$$\widehat{\beta}_l = \arg \min_{\beta \in \mathbf{R}^p} \frac{1}{2} \|y - X\beta\|^2 + \lambda \sum_{j=1}^{p-1} |\beta_j - \beta_{j+1}|$$

Soit

$$\widehat{\beta}_l = \arg \min_{\beta \in \mathbf{R}^p} \frac{1}{2} \|y - X\beta\|^2 + \lambda \|D\beta\|_1$$

- a) En quoi cette formulation permet d'obtenir des coefficients β_j ayant la même valeur?
 b) Montrez qu'il existe un $t > 0$ tel que le presque fused Lasso est équivalent à

$$\widehat{\beta}_l = \begin{cases} \arg \min_{\beta \in \mathbf{R}^p} & \frac{1}{2} \|y - X\beta\|^2 \\ \text{avec} & \|D\beta\|_1 \leq t \end{cases}$$

- c) Donner la forme standard de ce programme quadratique

J. R. Statist. Soc. B (2005)
67, Part 1, pp. 91–108

Sparsity and smoothness via the fused lasso

Robert Tibshirani and Michael Saunders,
 Stanford University, USA

Saharon Rosset,
 IBM T. J. Watson Research Center, Yorktown Heights, USA

Ji Zhu
 University of Michigan, Ann Arbor, USA

and Keith Knight
 University of Toronto, Canada