Université Paris 2 Panthéon-Assas
Session 2021–22

Machine Learning

Séance 11

L'objectif de cette session est de (i) rappeler le fonctionnement des arbres de décision et (ii) d'appliquer nos connaissances sur python.

1 Importation, description et visualisation des données

1. Importez la base de données

```
from sklearn.datasets import load_iris
iris = load_iris()
X_base = iris.data[:, 2:] # petal length and width
y_base = iris.target
```

2. Que représente Y ?
3. Que représente X ?

2 Machines à Vecteur de Support

2.1 Deux classes séparables

4. Visualisez les données :

```
X = X_base[:100]
y = y_base[:100]

import matplotlib.pyplot as plt
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1, edgecolor="k")
plt.xlabel("Petal length")
plt.ylabel("Petal width")
plt.xlim(0, 6)
plt.ylim(0, 2)
```

5. Tracez l'hyperplan séparateur

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

svc = svm.SVC(kernel='linear', C=1).fit(X, y)

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                     np.arange(y_min, y_max, 0.02))

Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1)
plt.xlabel('Petal length')
plt.ylabel('Petal width')
plt.xlim(0, 6)
plt.ylim(0, 2)
plt.show()
```

6. Vrai ou faux :
   (a) Dans un espace de p-dimension, un hyperplan est un sous-espace de dimension p+1
   (b) Dans un espace de 3 dimensions, un hyperplan est simplement une droite
   (c) Un hyperplan séparateur permet de classifier les observations selon leur localisation par rapport à l'hyperplan

(d) Dans un espace de 2 dimensions, il existe toujours au moins un hyperplan séparateur

(e) Dans un espace de 3 dimensions, il peut ne pas exister d'hyperplan séparateur

(f) Si les données sont parfaitement séparables avec un hyperplan, alors il existe une infinité d'hyperplans séparateurs

(g) La marge correspond à la distance perpendiculaire entre l'hyperplan et l'observation la plus proche de l'hyperplan

2.2 Deux classes non séparables

7. Visualisez les données

```
X = X_base[50:]
y = y_base[50:]

import matplotlib.pyplot as plt
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1, edgecolor="k")
plt.xlabel("Petal length")
plt.ylabel("Petal width")
```

8. Soft Margin classifier

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

svc1 = svm.SVC(kernel='linear', C=1).fit(X, y)
svc2 = svm.SVC(kernel='linear', C=10).fit(X, y)
svc3 = svm.SVC(kernel='linear', C=100).fit(X, y)
svc4 = svm.SVC(kernel='linear', C=1000).fit(X, y)

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                     np.arange(y_min, y_max, 0.02))

titles = ['C = 1',
          'C = 10',
          'C = 100',
          'C = 1000']

for i, clf in enumerate((svc1, svc2, svc3, svc4)):
    plt.subplot(2, 2, i + 1)
    plt.subplots_adjust(wspace=0.4, hspace=0.4)
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1)
    plt.xlabel('Petal length')
    plt.ylabel('Petal width')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.xticks(())
    plt.yticks(())
    plt.title(titles[i])
plt.show()
```

9. Vrai ou faux :

(a) Dans le cadre d'un problème de classification, on peut toujours utiliser le maximal margin classifier

(b) En présence d'un hyperplan séparateur, le maximal margin classifier produit toujours des meilleures performances que le soft margin classifier

(c) La différence principale entre le maximal margin classifier et le soft margin classifier repose sur la non-linéarité de l'hyperplan séparateur

(d) Le paramètre C détermine le nombre et le degré de violation de la marge

(e) Le paramètre C est toujours compris entre 0 et 1

(f) Plus C est grand, plus le biais sera faible sur la base d'apprentissage

2.3 Trois classes

10. Visualisez les données :

```
X = X_base
y = y_base

import matplotlib.pyplot as plt
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1, edgecolor="k")
plt.xlabel("Petal length")
plt.ylabel("Petal width")
```

11. Tracez les plans séparateurs

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

svc = svm.SVC(kernel='linear', C=1).fit(X, y)

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                     np.arange(y_min, y_max, 0.02))

Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1)
plt.xlabel('Petal length')
plt.ylabel('Petal width')
plt.show()
```

12. Changez le noyau :

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

svc = svm.SVC(kernel='linear', C=1).fit(X, y)
rbf_svc = svm.SVC(kernel='rbf', gamma=0.7, C=1).fit(X, y)
poly_svc = svm.SVC(kernel='poly', degree=3, C=1).fit(X, y)

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                     np.arange(y_min, y_max, 0.02))

titles = ['SVC avec noyau lineaire',
          'SVC avec noyau RBF ',
          'SVC avec noyau polynomial (degre 3)']

for i, clf in enumerate((svc, rbf_svc, poly_svc)):
    plt.subplot(2, 2, i + 1)
    plt.subplots_adjust(wspace=0.4, hspace=0.4)
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1)
    plt.xlabel('Petal length')
    plt.ylabel('Petal width')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.xticks(())
    plt.yticks(())
    plt.title(titles[i])
plt.show()
```

13. Vrai ou faux :

    (a) En général, l'astuce du noyau consiste à transposer les données dans un autre espace de plus petite dimension dans lequel elles sont linéairement séparables

    (b) L'intuition derrière l'astuce du noyau consiste à transposer un problème non-linéaire en problème linéaire

    (c) Dans le cadre des machines à vecteur de support, utiliser un noyau polynomial amène toujours les meilleures performances

    (d) Dans le cadre des machines à vecteur de support, utiliser un noyau gaussien amène toujours les meilleures performances

(e) Si on souhaite déterminer le risque de crise cardiaque en fonction des caractéristiques d'un individu, on peut utiliser une machine à vecteur de support

(f) Les machines à vecteur de support peuvent constituer une alternative à la régression logistique

(g) Les machines à vecteur de support sont souvent utilisées pour des problèmes d'inférence

2.4 Comparaison de plusieurs modèles

14. Divisez la base de données en deux :

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=6)
```

15. Comparaison de la performance des modèles selon la valeur de C :

```
from sklearn import svm

list_C = [1,10,100,1000]
for i in list_C:
    svc = svm.SVC(kernel='linear', C=i).fit(X_train, y_train)
    print(f'% Classification Correct - Lineaire (Base Apprentissage) : {svc.score(X_train, y_train):.3f}')
    print(f'% Classification Correct - Lineaire (Base Test) : {svc.score(X_test, y_test):.3f}')
```

16. Comparaison de la performance des modèles selon le noyau utilisé :

```
from sklearn import svm

svc = svm.SVC(kernel='linear', C=10).fit(X_train, y_train)
rbf_svc = svm.SVC(kernel='rbf', gamma=0.7, C=10).fit(X_train, y_train)
poly_svc = svm.SVC(kernel='poly', degree=3, C=10).fit(X_train, y_train)
print(f'% Classification Correct - Lineaire (Base Test) : {svc.score(X_test, y_test):.3f}')
print(f'% Classification Correct - RBF (Base Test) : {rbf_svc.score(X_test, y_test):.3f}')
print(f'% Classification Correct - Poly (Base Test) : {poly_svc.score(X_test, y_test):.3f}')
```

17. Questions sur la base des résultats obtenues suite aux questions précédentes :

(a) Quelle valeur de C choisiriez-vous ?

(b) Quel noyau choisiriez-vous ?

3 Réseaux de neurones

18. Divisez la base en deux :

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=6)
```

3.1 Couches Cachées

19. Variez le nombre de couches cachées :

```
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier

n_couche = [1,10,20,50]
train_results = []
test_results = []
for i in n_couche:
    rf = MLPClassifier(hidden_layer_sizes=(i), solver = 'lbfgs',
                       random_state=21,)
    rf.fit(X_train, y_train)
    train_pred = rf.predict(X_train)
    test_r = rf.score(X_test,y_test)
    test_results.append(test_r)
    train_r = rf.score(X_train,y_train)
    train_results.append(train_r)
from matplotlib.legend_handler import HandlerLine2D
line1, = plt.plot(n_couche, train_results, label='Erreur d apprentissage')
line2, = plt.plot(n_couche, test_results, label='Erreur de test')
```

```
plt.legend(handler_map={line1: HandlerLine2D(numpoints=2)})
plt.ylabel('Taux de Classification Correct')
plt.xlabel('Nombre de couches cachees')
fig1 = plt.gcf()
fig1.savefig('/Users/maitre/Dropbox/Assas/cours/machine_learning/slides/lecture11/hidden_layer_size.png')
plt.show()
```

3.2 Fonction d'activation

20. Comparez la performance des modèles selon la fonction d'activation :

```
from sklearn.neural_network import MLPClassifier

f_activation = ['identity', 'logistic', 'tanh', 'relu']
train_results = []
test_results = []
for i in f_activation:
    rf = MLPClassifier(hidden_layer_sizes=(1), activation = i, solver = 'lbfgs',
                       random_state=2)
    rf.fit(X_train, y_train)
    print(f'Test : {rf.score(X_test, y_test):.3f}')
    print(f'Train : {rf.score(X_train, y_train):.3f}')
```

3.3 Questions

21. Vrai ou faux :
    (a) Dans le cadre des réseaux de neurones, la performance du modèle augmente nécessairement avec le nombre de couches cachées
    (b) Dans le cadre des réseaux de neurones, la fonction d'activation logistique produit nécessairement des meilleures performances que la fonction linéaire
    (c) Dans le cadre des réseaux de neurones, on initialise les poids aléatoirement
    (d) Dans le cadre des réseaux de neurones, on parle en général de deep learning quand le nombre de couches cachées est inférieur à 3
    (e) Dans le cadre des machines à vecteur de support, on parle en général de deep learning quand on utilise un noyau gaussien
    (f) L'algorithme de rétro-propagation a trois composantes fondamentales : reverse pass, forward pass et step-wise pass