Nom et Prénom : BATTACHE Nassim.

Master 1 OIVM - Optique Image Vision MultiMedia

Apprentissage supervisé

Abstract :

Ce rapport résume l'utilisation des plusieurs algorithmes à noyaux en python (scikit-learn.

Introduction

One Class SVM (OCSVM)

SVM pour la régression

Conclusion

Introduction

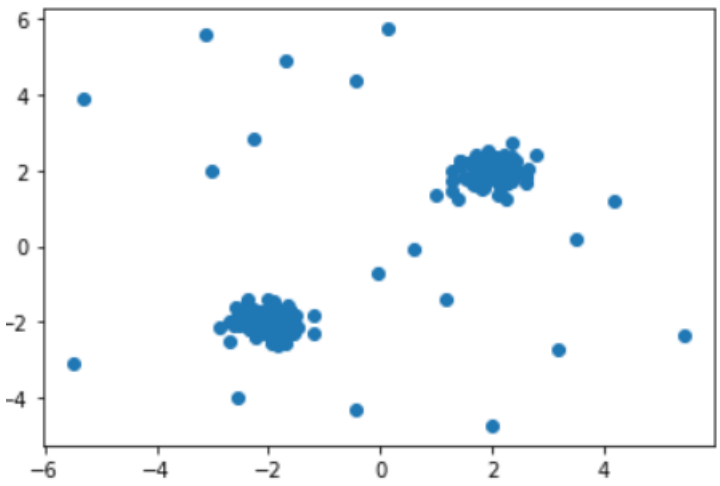
Le SVM à classe unique est un algorithme non supervisé qui apprend une fonction de décision pour la détection de nouveauté : classer les nouvelles données comme similaires ou différentes de l'ensemble d'apprentissage.

Une méthode de classification à classe unique est utilisée pour détecter les valeurs aberrantes et les anomalies dans un ensemble de données. Basé sur l'évaluation des machines à vecteurs de support (SVM), le SVM à classe unique applique une méthode de classification à classe unique pour la détection de nouveauté.

One Class SVM (OCSVM)

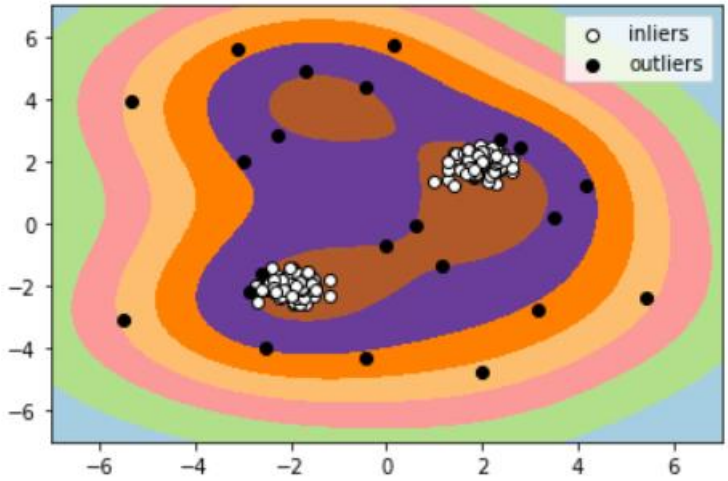
On execute le programme suivant donné en TP on obtiens la figure suivante :

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.svm import OneClassSVM
# On créé deux groupes séparés (échantillons de gaussiennes)
N = 200
data1 = 0.3 * np.random.randn(N // 2, 2) + [2,2]
data2 = 0.3 * np.random.randn(N // 2, 2) - [2,2]
# On créé 10% de données anormales (*outliers*)
outliers = np.random.uniform(size=(N // 10, 2), low=-6, high=6)
# Les données = groupes + anomalies
X = np.concatenate((data1, data2, outliers))
plt.scatter(X[:,0], X[:,1]) and plt.show()
```



On affiche les points et les vecteurs les plus proches du plan de séparation

```
# Afficher les points et les vecteurs les plus proches du plan de séparation
xx, yy = np.meshgrid(np.linspace(-7, 7, 500), np.linspace(-7, 7, 500))
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
y_pred = clf.predict(X)
# Choix du jeu de couleurs
plt.set_cmap(plt.cm.Paired)
# Trace le contour de la fonction de décision
plt.contourf(xx, yy, Z)
# Affiche les points considérés comme "inliers"
plt.scatter(X[y_pred>0,0], X[y_pred>0,1], c='white', edgecolors='k',
label='inliers')
# Affiche les points considérés comme "outliers"
plt.scatter(X[y_pred<=0,0], X[y_pred<=0,1], c='black', label='outliers')
plt.legend()
plt.show()
```



Question

Testez plusieurs valeurs pour le paramètre gamma. Pour quelle valeur le résultat semble meilleur (moins de outliers incorrectement classés) ? En pratique on ne connait pas les outliers, l'utilité des OCSVM est de les détecter. Le paramètre nu doit aussi avoir une bonne valeur pour ne pas sous-estimer (ou sur-estimer) le support de la distribution.

Pour des valeurs differentes de gamma et nu on execute les lignes de commandes suivantes :

```python
# Construction du modèle (noyau RBF) pour plusieurs valeurs gamma
import numpy as np
import collections as cl
gamma=[0.0001,0.001,0.01,0.10,1.00]
nu=[0.1,0.2,0.3,0.4]
print( "le score pour classification par une OneclassSVM à noyau gaussien:" )
#classification par une SVM à noyau gaussien :
for i in range (len (gamma)):
    for j in range(len(nu)):
        clf = svm.OneClassSVM(nu=nu[j], kernel="rbf", gamma=gamma[i])
        clf.fit(X)
        clf_accuracy = clf.score_samples(X)
        clf_threshold = np.quantile(clf_accuracy,0.03)
        print("le seuille est : "+str(clf_threshold))
        outliers = (clf_accuracy<clf_threshold)
        outliers = cl.Counter(outliers == False)
        inliers = (clf_accuracy>=clf_threshold)
        inliers = cl.Counter(inliers == True)
        print ("le score pour une valeur de gamma ="+str(gamma[i])+ " et une valeur nu ="+str(nu[j])+" est : " +str(clf.sco
        print("inliers = {} et outliers = {}".format(inliers[True], outliers[False]))
```

On obtient les résultats suivants :

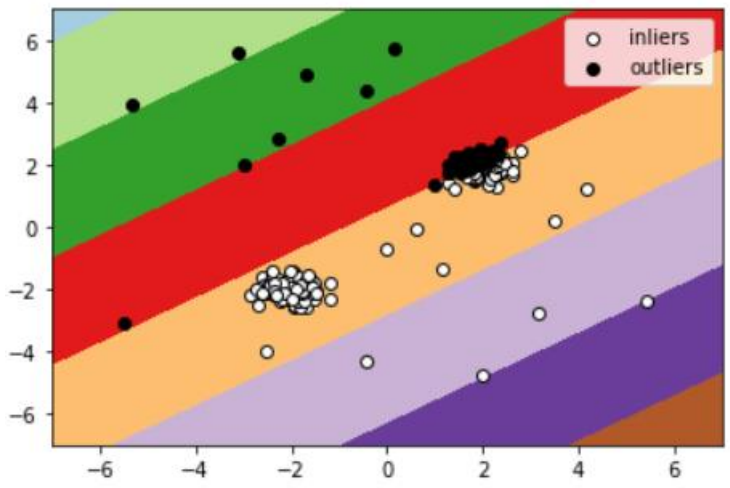
```
le score pour classification par une OneclassSVM à noyau gaussien:
le seuille est : 21.902558705422905
le score pour une valeur de gamma =0.0001 et une valeur nu =0.1 est :
inliers = 213 et outliers = 7
le seuille est : 43.827262272360926
le score pour une valeur de gamma =0.0001 et une valeur nu =0.2 est :
inliers = 213 et outliers = 7
le seuille est : 65.75328927197475
le score pour une valeur de gamma =0.0001 et une valeur nu =0.3 est :
inliers = 213 et outliers = 7
le seuille est : 87.68048346257012
le score pour une valeur de gamma =0.0001 et une valeur nu =0.4 est :
inliers = 213 et outliers = 7
le seuille est : 21.054783730419754
le score pour une valeur de gamma =0.001 et une valeur nu =0.1 est :
inliers = 213 et outliers = 7
le seuille est : 42.31725892616383
le score pour une valeur de gamma =0.001 et une valeur nu =0.2 est :
inliers = 213 et outliers = 7
le seuille est : 63.59589822859319
le score pour une valeur de gamma =0.001 et une valeur nu =0.3 est :
inliers = 213 et outliers = 7
le seuille est : 84.8834067100382
le score pour une valeur de gamma =0.001 et une valeur nu =0.4 est :
inliers = 213 et outliers = 7
le seuille est : 14.901560456825017
le score pour une valeur de gamma =0.01 et une valeur nu =0.1 est :
inliers = 213 et outliers = 7
le seuille est : 30.94016035309538
le score pour une valeur de gamma =0.01 et une valeur nu =0.2 est :
inliers = 213 et outliers = 7
le seuille est : 47.100444657112135
le score pour une valeur de gamma =0.01 et une valeur nu =0.3 est :
inliers = 213 et outliers = 7
le seuille est : 63.292448623227614
le score pour une valeur de gamma =0.01 et une valeur nu =0.4 est :
inliers = 213 et outliers = 7
le seuille est : 4.012084061439124
le score pour une valeur de gamma =0.1 et une valeur nu =0.1 est :
```

```
inliers = 213 et outliers = 7
le seuille est : 6.033246089440288
le score pour une valeur de gamma =0.1 et une valeur nu =0.2 est :
inliers = 213 et outliers = 7
le seuille est : 8.51381953739755
le score pour une valeur de gamma =0.1 et une valeur nu =0.3 est :
inliers = 213 et outliers = 7
le seuille est : 11.100176909126116
le score pour une valeur de gamma =0.1 et une valeur nu =0.4 est :
inliers = 213 et outliers = 7
le seuille est : 1.0807667923839959
le score pour une valeur de gamma =1.0 et une valeur nu =0.1 est :
inliers = 213 et outliers = 7
le seuille est : 1.1066808675636963
le score pour une valeur de gamma =1.0 et une valeur nu =0.2 est :
inliers = 213 et outliers = 7
le seuille est : 1.106682336793638
le score pour une valeur de gamma =1.0 et une valeur nu =0.3 est :
inliers = 213 et outliers = 7
le seuille est : 1.1066829607998407
le score pour une valeur de gamma =1.0 et une valeur nu =0.4 est :
inliers = 213 et outliers = 7
```

Visuellement, une valeur de gamma entre 1e-4 et 5e-2 semble acceptable : les centres des gaussiennes sont identifiées comme inliers tandis que les points plus éloignés sont marqués comme aberrants (outliers), ce qui correspond à la façon dont on a généré les données.

Question

Remplacez le noyau rbf par un noyau linéaire. Quel problème constatez-vous ?



Le noyau linéaire ne permet que de trouver des hyperplans séparateurs, c'est-à-dire des droites (dans notre cas bidimensionnel). On ne peut donc pas séparer les inliers et les outliers de cette façon dans notre cas car la frontière réelle n'est pas linéaire.

SVM pour la régression

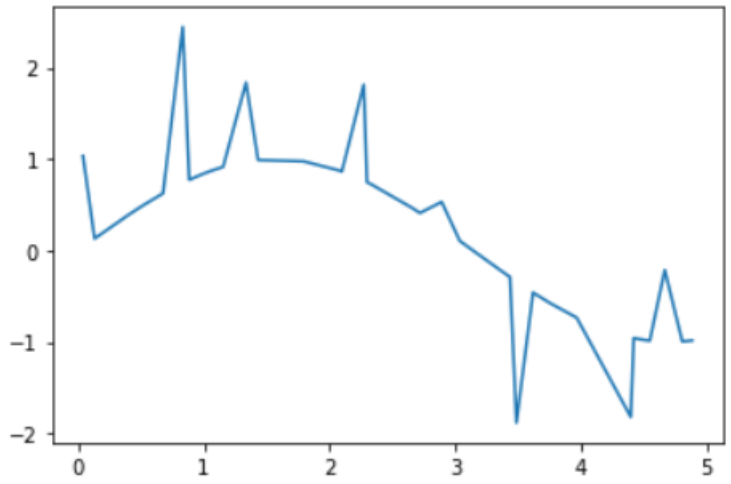
On compile le programme donné , le résultat obtenu est le suivant :

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

```
X = np.sort(5 * np.random.rand(40, 1), axis=0)
y = np.sin(X).ravel()
```

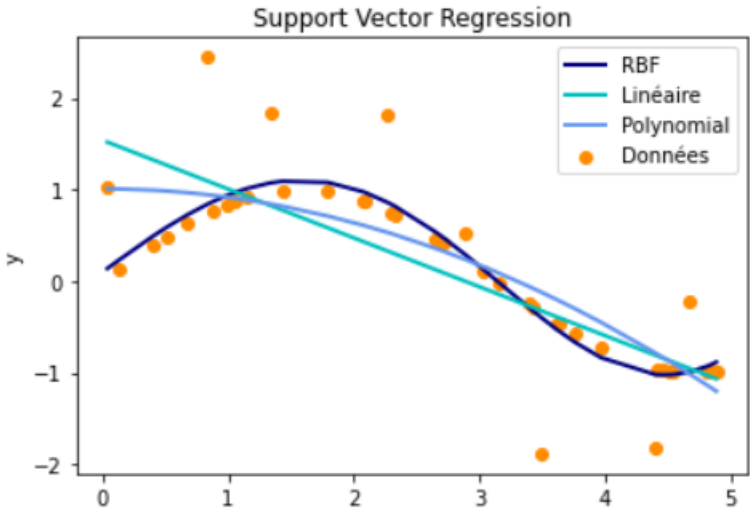
```
y[::5] += 3 * (0.5 - np.random.rand(8))
plt.plot(X, y)
```

```
[<matplotlib.lines.Line2D at 0x1722f7d7460>]
```



```
# Creation des SVM
C = 1e3
svr_rbf = SVR(kernel='rbf', C=C, gamma=0.1)
svr_lin = SVR(kernel='linear', C=C)
svr_poly = SVR(kernel='poly', C=C, degree=2)
# Entraînement des SVM sur les observations bruitées
y_rbf = svr_rbf.fit(X, y).predict(X)
y_lin = svr_lin.fit(X, y).predict(X)
y_poly = svr_poly.fit(X, y).predict(X)
```

```
plt.scatter(X, y, color='darkorange', label='Données')
plt.plot(X, y_rbf, color='navy', lw=2, label='RBF')
plt.plot(X, y_lin, color='c', lw=2, label='Linéaire')
plt.plot(X, y_poly, color='cornflowerblue', lw=2, label='Polynomial')
plt.xlabel('X')
plt.ylabel('y')
plt.title('Support Vector Regression')
plt.legend()
plt.show()
```



Question :

Pourquoi employer une valeur aussi grande pour le parametre C (ici, C = 1000) ?

Une valeur trop petite de C nous donne un sous apprentissage. Pour une valeur trop grande, nous donne l'inverse qui est le sur-apprentissage.

Diabetes dataset

Question

Chargez la base de données Diabetes du module sklearn.datasets et faites une partition aléatoire en partie apprentissage et partie test (70% apprentissage, 30% test). Construisez un modèle de SVM de régression sur cette base et calculez l'erreur quadratique moyenne sur l'ensemble de test. Utilisez GridSearchCV pour déterminer le meilleur noyau à utiliser.

```
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV

param_grid = {
    'C': [0.1, 1.0, 10, 100],
    'gamma':[0.001,0.005,'scale'],
    'kernel':['rbf','linear','poly']
}

search = GridSearchCV(svm.SVR(), param_grid, n_jobs=4, verbose=1, scoring='neg_mean_squared_error')
search.fit(X_train, y_train)
print(search.best_params_)
print(search.score(X_test, y_test))
```

Les meilleurs paramètre sont :

```
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent

{'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
-3088.7746656296613
```

Pour spécifier qu'on veut trouver l'erreur quadratique moyenne on utiliser la ligne de commande suivante : scoring='neg_mean_squared_error'. en plus cette erreur dépend de l'amplitude des données.

Conclusion

Les avantages des SVM :

- Moins efficace sur les jeux de données contenant du bruit et beaucoup d'outliers

Les OCSVM sont des estimateurs de support de densité pour des données multidimensionnelles. L'idée derrière l'implémentation est de trouver l'hyperplan le plus éloigné de l'origine qui sépare les données de l'origine.

- Très efficaces en dimension élevée.
- Ils sont aussi efficaces dans le cas où la dimension de l'espace est plus grande que le nombre d'échantillons d'apprentissage.
- N'utilisent pas tous les échantillons d'apprentissage, mais seulement une partie (les vecteurs de support). En conséquence, ces algorithmes demandent moins de mémoire.
- Sa grande précision de prédiction
- Fonctionne bien sûr de plus petits data sets