Nom et Prénom : BATTACHE Nassim.

Master 1 OIVM - Optique Image Vision MultiMedia

Apprentissage supervisé

Abstract :

Ce rapport résume la suite des SVM linéaires pour l'apprentissage supervisé .

Introduction

Scikit-learn

Jeu de données Digits

Conclusion

Introduction

Les machines à vecteurs de support (SVM : *Support Vector Machines*) sont une classe de méthodes d'apprentissage statistique basées sur le principe de la maximisation de la marge (séparation des classes). Il existe plusieurs formulations (linéaires, versions à noyaux) qui peuvent s'appliquer sur des données séparables (linéairement) mais aussi sur des données non séparables.
Les machines à vecteurs de support ou séparateurs à vaste marge (en anglais support-vector machine, SVM) sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de discrimination et de régression. Les SVM sont une généralisation des classifieurs linéaires.

Une machine à vecteurs de support, traduction littérale pour Support Vector Machine, est un algorithme d'apprentissage automatique supervisé qui peut être utilisé à des fins de classification et de régression. Les SVM sont plus généralement utilisés dans les situations de classification.

Scikit-learn

Question :

Refaites la classification de la base de données iris mais avec un noyau gaussien. Testez l'effet du paramètre d'échelle du noyau (gamma) et du paramètre de régularisation C.

Pour des valeurs differentes de gamma et c et après standarisation des données on execute les lignes de commandes suivantes :

```
gamma=[0.01,0.1,1,10,100]
c=[0.01,0.1,1,10,100]
for i in range (len (gamma)):
    print( "Pour gamma = "+str (gamma[i])+ ":" )
    for j in range (len (c)):
        clf = svm.SVC(C=c[j],kernel='rbf', gamma=gamma[i])
        clf.fit(X_train, y_train)
        clf.score(X_test, y_test)
        print ("le score pour une valeur de c ="+str(c[j])+ " est de : " +str(clf.score(X_test, y_test)))
```

Pour cette exécution on obtient le résultat suivant :

```
Pour gamma = 0.01:
le score pour une valeur de c =0.01 est de : 0.28
le score pour une valeur de c =0.1 est de : 0.28
le score pour une valeur de c =1 est de : 0.8133333333333334
le score pour une valeur de c =10 est de : 0.9466666666666667
le score pour une valeur de c =100 est de : 0.9333333333333333
Pour gamma = 0.1:
le score pour une valeur de c =0.01 est de : 0.28
le score pour une valeur de c =0.1 est de : 0.7333333333333333
le score pour une valeur de c =1 est de : 0.9333333333333333
le score pour une valeur de c =10 est de : 0.9466666666666667
le score pour une valeur de c =100 est de : 0.96
Pour gamma = 1:
le score pour une valeur de c =0.01 est de : 0.28
le score pour une valeur de c =0.1 est de : 0.7466666666666667
le score pour une valeur de c =1 est de : 0.9066666666666666
le score pour une valeur de c =10 est de : 0.92
le score pour une valeur de c =100 est de : 0.92
Pour gamma = 10:
le score pour une valeur de c =0.01 est de : 0.28
le score pour une valeur de c =0.1 est de : 0.28
le score pour une valeur de c =1 est de : 0.76
le score pour une valeur de c =10 est de : 0.8
le score pour une valeur de c =100 est de : 0.8
Pour gamma = 100:
le score pour une valeur de c =0.01 est de : 0.28
le score pour une valeur de c =0.1 est de : 0.28
le score pour une valeur de c =1 est de : 0.32
le score pour une valeur de c =10 est de : 0.32
le score pour une valeur de c =100 est de : 0.32
```

Remarque :

On obtient un meilleur score qui est de 0.96 qui correspond à une valeur de c =100 et gamma = 0.1.

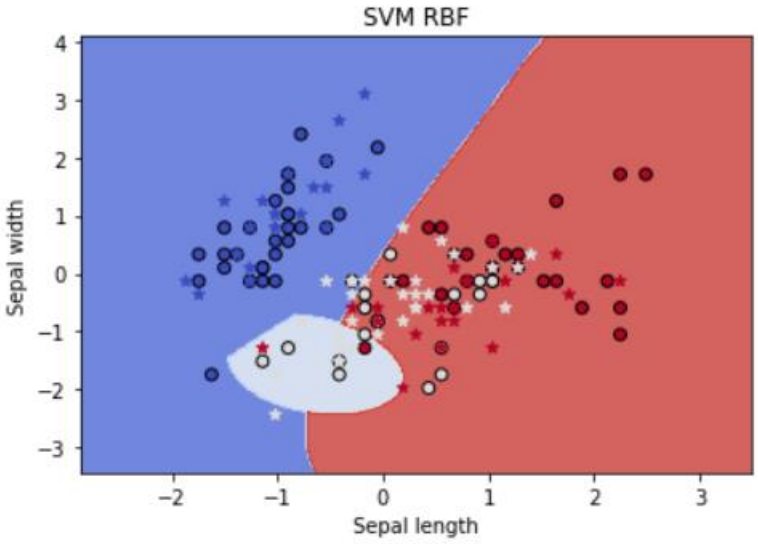
En revanche avec gamma =100 on obtiens les plus faibles scores pour des valeurs de c.

Question :

Que constatez-vous par rapport au TP précédent ?

On exécute le programme donné dans la feuille de TP puis on obtient la figure suivante :

```
Out[19]: Text(0.5, 1.0, 'SVM RBF')
```



L'utilisation du noyau gaussien permet d'obtenir des frontières de décision non linéaires. Le paramètre gamma correspond au rayon d'influence de chaque observation : plus γ est élevé, plus le rayon d'influence de chaque observation est réduit. Les observations plus proches de la frontière ont donc plus de poids et la frontière aura tendance à « coller » aux observations.

Jeu de données Digits

Question

Réalisez une classification par une SVM linéaire et une SVM à noyau gaussien du jeu de données Digits. Comment est choisi le paramètre gamma dans scikit-learn ? Testez différentes valeurs de ce paramètre pour évaluer son influence. En particulier, testez-les paramètres gamma='auto' et gamma='scale'. À quoi correspondent-ils ?

On exécute le code suivant :

```
gamma=[0.01,0.1,1,10,100,'auto','scale']
print( "Pour C = "+str(100)+ ":" )
#classification par une SVM à noyau gaussien :
for i in range (len (gamma)):
        clf = svm.SVC(C=100,kernel='rbf', gamma=gamma[i])
        clf.fit(X_train, y_train)
        clf.score(X_test, y_test)
        print ("le score pour une valeur de gamma ="+str(gamma[i])+ " est de : " +str(clf.score(X_test, y_test)))
#classification par une SVM linéaire
clf = svm.LinearSVC(C=100)
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
print ("le score pour classification par une SVM linéaire et la valeur de c ="+str(100)+ " est de : " +str(clf.score(X_test,
```

On obtient le résultat suivant :

```
le score pour classification par une SVM à noyau gaussien et la valeur de c 100:
le score pour une valeur de gamma =0.01 est de : 0.9805555555555555
le score pour une valeur de gamma =0.1 est de : 0.9388888888888889
le score pour une valeur de gamma =1 est de : 0.08888888888888889
le score pour une valeur de gamma =10 est de : 0.08055555555555556
le score pour une valeur de gamma =100 est de : 0.08055555555555556
le score pour une valeur de gamma =auto est de : 0.975
le score pour une valeur de gamma =scale est de : 0.975
le score pour classification par une SVM linéaire et la valeur de c =100 est de : 0.9333333333333333
```

Pour gamma =’auto’ est une valeur par défaut qui est choisi par 1/ n_features.

Pour gamma =’scale’ la valeur est choisi par 1 / (n_features * X.var())

Question :

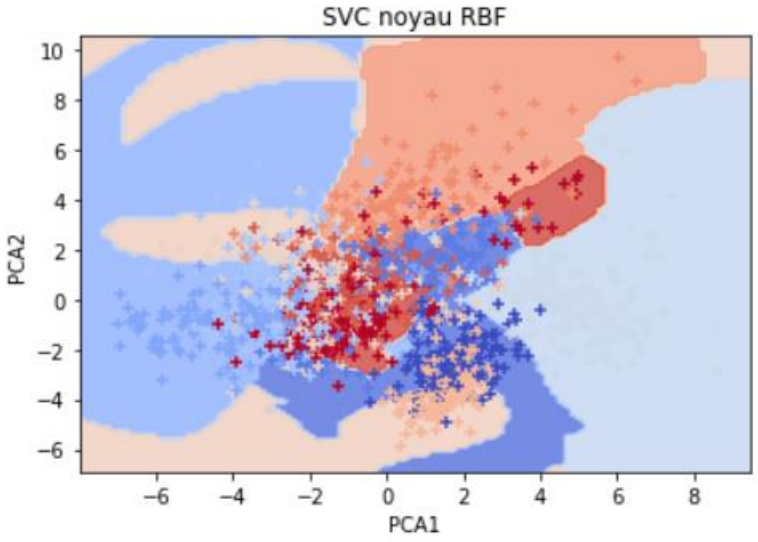
Réalisez une analyse en composante principale (ACP) et gardez les 2 premières composantes principales (voir la documentation Scikit-learn). Ensuite faites une classification avec un noyau gaussien et affichez les points de test ainsi que la surface de décision (reprendre le code du TP SVM linéaire). Comparez avec une SVM linéaire.

On commence par la classification par une SVM à noyau gaussien :

On prend le paramètre de régularisation c= 100 .

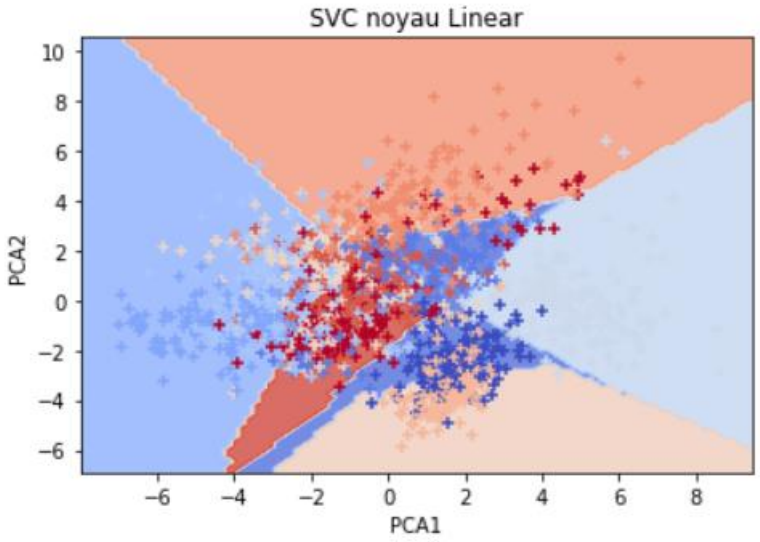
```python
#classification par une SVM à noyau gaussien :
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

# ACP
pca = PCA(n_components=2)
X_s = pca.fit_transform(X_train)


svc = svm.SVC(kernel='rbf', C=100).fit(X_s, y_train)

# Créer la surface de décision
x_min, x_max = X_s[:, 0].min() - 1, X_s[:, 0].max() + 1
y_min, y_max = X_s[:, 1].min() - 1, X_s[:, 1].max() + 1
h = max((x_max - x_min) / 100, (y_max - y_min) / 100)
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
title = 'SVC noyau RBF'

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)
# Afficher aussi les points
plt.scatter(X_s[:, 0], X_s[:, 1], c=y_train, marker='+', cmap=plt.cm.coolwarm)
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.title(title)
plt.show()
```

Voici la figure obtenue :



Pour la partie de la classification par une SVM linéaire il suffit de changer et remplacer kernel='linear'

Avec le même code :



Remarque :

Les surfaces de décisions sont séparées de façon linéaire .

Question

Réalisez une recherche par grille afin de déterminer sur le jeu de données Digits complet (sans l'ACP) : § le meilleur noyau à utiliser, § la meilleure valeur de C, § la meilleure valeur de gamma (ou le degré du polynôme pour un noyau polynomial). § la meilleure valeur de n_components de l'ACP.

Avec GridSearchCV permet de chercher les différents paramètre et combinaisons en fonction du type de de noyaux.

Pour le paramètre n_jobs =4 cela permet de paralléliser les calcule sous 4 cœurs .

On exécute le programme suivant :

```
from sklearn.model_selection import GridSearchCV

param_grid = [
    {'kernel': ['rbf'], 'gamma': ['auto', 'scale', 0.1, 1, 10], 'C': [0.01, 0.1, 1.0, 10, 100]},
    {'kernel': ['poly'], 'degree': [3, 10, 30], 'C': [0.01, 0.1, 1.0, 10, 100]},
    {'kernel': ['linear'], 'C': [0.01, 0.1, 1.0, 10, 100]}
]
clf = GridSearchCV(svm.SVC(), param_grid, cv=3, n_jobs=4, verbose=1)
clf.fit(X_train, y_train)
print(clf.best_params_)
```

```python
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV

pca = PCA()
# En supposant que l'on se concentre sur une SVM RBF
svc = svm.SVC(kernel='rbf', gamma='scale')
pipe = Pipeline(steps=[('pca', pca), ('svm', svc)])

# Syntaxe : nomdustep__nomduparamètre
param_grid = {
    'pca__n_components': [5, 15, 30, 45, 64],
    'svm__C': [0.1, 1.0, 10, 100],
}
search = GridSearchCV(pipe, param_grid, n_jobs=4, verbose=1)
search.fit(X_train, y_train)
print(search.best_params_)
```

On a le résultat suivant :

```
{'pca__n_components': 30, 'svm__C': 10}
```

La meilleure valeur de c =10

La meilleure valeur de nombre de components de l' ACP =30

La meilleure valeur de gamma =

Question

(optionnel) Combien de composantes faut-il garder au minimum dans l'ACP pour classer correctement au moins 97% des images ? À quel facteur de réduction de dimension cela correspond-il ?

```python
pca = PCA(n_components=28)
svc = svm.SVC(kernel='rbf', gamma='scale')
pipe = Pipeline(steps=[('pca', pca), ('svc', svc)])
pipe.fit(X_train, y_train)
print(pipe.score(X_test, y_test))
```

```
0.9722222222222222
```

Dans ce cas on a le nombre de composante est a partir de 28 composant.

Conclusion

Les avantages des SVM :

- 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
- Ils peuvent être plus efficace car ils utilisent un sous-ensemble de points d'entraînement.

Les désavantage et les inconvénients :

- Ne convient pas à des jeux de données plus volumineux, car le temps d'entraînement avec les SVM peut être long
- Moins efficace sur les jeux de données contenant du bruits et beaucoup d'outliers