Scikit Learn

Giảm chiều ma trận đặc trưng, tìm kiếm lưới và lưu mô hình

HK2, Năm học 2019 - 2020

Giảm chiều dữ liệu - Dimensionality Reduction

Dimensionality Reduction (giảm chiều dữ liệu) là một trong những kỹ thuật quan trọng trong Máy học.
Các vector đặc trưng trong các bài toán thực tế có thể có số chiều (cột) lớn. Nếu thực hiện lưu trữ và tính toán trực tiếp trên dữ liệu có số chiều cao này thì sẽ gặp khó khăn về việc lưu trữ và tốc độ tính toán.
Vì vậy giảm số chiều dữ liệu là một bước quan trọng trong nhiều bài toán.
Đây cũng được coi là phương pháp nén dữ liệu.

Một trong những cách có thể sử dụng là Principal Component Analysis (PCA)

Giảm chiều dữ liệu - Dimensionality Reduction

```
from sklearn import datasets
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

wine = datasets.load_wine()
X, y = wine.data, wine.target
X_train_org, X_test_org, y_train, y_test = train_test_split(X, y,
    test_size = 0.3, random_state = 10)
# khoi tao PCA
pca = PCA(n_components=10, random_state=10)
# huan luyen PCA
pca.fit(X_train_org)
# chuyen doi cho tap train va test
X_train = pca.transform(X_train_org)
X_test = pca.transform(X_test_org)
```

Giảm chiều dữ liệu - Dimensionality Reduction

```
model = KNeighborsClassifier()
model.fit(X_train, y_train)
preds = model.predict(X_test)
score = accuracy_score(y_test, preds)
print(score)
```

Tìm tham số tốt cho mô hình

Các mô hình đang sử dụng thường bao gồm nhiều hyperparameters và tùy thuộc vào bài toán cụ thuộc sẽ có các hyperparameters thích hợp.
Để tìm hyperparameters phù hợp cho từng bài toán cụ thể, cách đơn giản là thử nhiều biến khác nhau.

- Random Search: từ những biến cần tìm chọn ngẫu nhiên giá trị cho các hyperparameter để đánh giá độ chính xác của mô hình
- Grid Search: giả dụ có 02 hyperparameter có các giá trị có thể nhận từ 0-9. Grid Search sẽ ghép lần lượt từng giá trị của hyperparam 1 với hyperparam 2 để tính toán độ chính xác của model.

Tìm tham số tốt cho mô hình

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score

irisX, irisY = datasets.load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(
    irisX,irisY,test_size=0.4,
random_state = 1)

parameters = {'n_neighbors':[1,2,3,4,5,6,7,8,9,10]}
knn = KNeighborsClassifier()
clf = GridSearchCV(knn,parameters,scoring='accuracy')
# scoring co the: 'neg_mean_squared_error', 'r2'
grid_result = clf.fit(X_train, y_train)
```

Tìm tham số tốt cho mô hình

```
print(grid_result)
print(grid_result.best_params_)
means = clf.cv_results_['mean_test_score']
print(means)

model = grid_result.best_estimator_

preds = model.predict(X_test)
print(accuracy_score(y_test, preds))
```

Tìm mô hình

```
1  from sklearn import datasets
2  from sklearn.model_selection import train_test_split
3  from sklearn.svm import SVC
4  from sklearn.linear_model import SGDClassifier
5  from sklearn.ensemble import RandomForestClassifier
6  from sklearn.neighbors import KNeighborsClassifier
7  from sklearn.metrics import accuracy_score
8  from sklearn.externals import joblib
9
10 X, y = datasets.load_iris(return_X_y = True)
11 X_train, X_test, y_train, y_test = train_test_split(X
       , y, test_size = 0.4,
12 random_state = 12)
```

Tìm mô hình

```
models = []
models.append(('SupportVectorClassifier', SVC()))
models.append(('SGDClassifier', SGDClassifier()))
models.append(('RandomForest', RandomForestClassifier
    ()))
models.append(('KNN', KNeighborsClassifier()))

results = []
scoring = 'accuracy'
for name, model in models:
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    result = accuracy_score(y_test, pred)
    results.append((name, result))
    joblib.dump(model, name + '.pkl')
```

Tìm mô hình

```
sample = [[3.2 , 5.3 , 4.1 , 2.8] , [2.1 ,3.2 ,1.9 ,0.5]]
for name, result in results:
    model = joblib.load(name+'.pkl')
    pred = model.predict(sample)
    print('Model %s co do chinh xac %f' %(name, result))
    print(pred)
```

Bài 1: Digits

Sử dụng tập dữ liệu `digits` của `sklearn` (`load_digits`), thực hiện các yêu cầu sau

- Kiểm tra có bao nhiêu đặc trưng/thuộc tính đối với tập dữ liệu trên (số cột của ma trận đặc trưng)
- Train và test được chia theo tỷ lệ: 8:2
- Sử dụng `PCA` để giảm chiều dữ liệu xuống 16
- Sử dụng mô hình KNN để đánh giá việc học trên tập dữ liệu và tập dữ liệu đã giảm chiều: việc giảm chiều có tác dụng cải thiện độ chính xác không?

Bài 2: Titanic

Sử dụng titanic từ file *titanic_pre.csv* (file có header nên không cần truyền names cho nó).
Giả sử mô hình SVC có 02 hyparameter: C có giá trị 1 hoặc 10, kernel có giá trị 'linear' hoặc 'rbf'

- Train và test set được chia theo tỷ lệ: 8:2
- Thực hiện chuẩn hóa dữ liệu (chọn 1 trong các cách đã được giới thiệu)
- Sử dụng GridSearchCV để tìm mô hình tốt nhất
- Đánh giá mô hình trên test
- Lưu lại mô hình đó với tên file *svc_titanic.pkl*

Bài 3: Fertility Diagnosis

Sử dụng dữ liệu từ file *fertility_Diagnosis.csv* (file có header nên không cần truyền names cho nó).

- Xác định cần mô hình phân lớp hay hồi quy?
- Chia tập dữ liệu 8:2
- Chuẩn hóa dữ liệu (tùy ý)
- Giả sử có 03 mô hình cần xem xét: KNN, DecisionTree và RandomForest (sử dụng tham số mặc định). In đánh giá của các mô hình trên và cho biết mô hình/những mô hình nào được xem xét chọn