Machine Learning HW6

0556562 陳鴻君

Overview

1. Compare SVM model with different kernels and parameters
2. Project features into two dimension via PCA and compare models
3. Visualization

Compare SVM model with different kernels and parameters

I did this project mainly with Scikit-Learn package in Python.
Below is all the imports.

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from sklearn.decomposition import PCA
5 from sklearn import svm
6 from sklearn.model_selection import cross_val_score
```

Set the directory path and colors of plotted dots.
Load data into numpy arrays.

```
 8 path = '/home/jordan/Documents/ml/hw6/data/'
 9 colors = ['red', 'blue', 'green', 'purple', 'orange']
10
11 def load_data(path):
12     X_train = np.genfromtxt(path+'X_train.csv', delimiter=',')
13     y_train = np.genfromtxt(path+'T_train.csv', delimiter=',')
14     X_test = np.genfromtxt(path+'X_test.csv', delimiter=',')
15     y_test = np.genfromtxt(path+'T_test.csv', delimiter=',')
16     return X_train, y_train, X_test, y_test
```

Set parameter C equal to 5.
Compare 6 different kernels and parameters.

1. Linear kernel
2. Linear SVC
3. RBF kernel with gamma=0.05 and C=5
4. RBF kernel with gamma=0.5 and C=3
5. RBF kernel with sklearn default parameters
6. Polynomial kernel with degree 2

There are two comparisons:

1. 5-fold cross validation via training set
2. Accuracy via testing set

```
29 if __name__ == '__main__':
30     X_train, y_train, X_test, y_test = load_data(path)
31     C = 5
32     #compare kernels and parameters
33     clfs = (svm.SVC(kernel='linear', C=C),
34             svm.LinearSVC(C=C),
35             svm.SVC(kernel='rbf', gamma=0.05, C=C),
36             svm.SVC(kernel='rbf', gamma=0.5, C=3),
37             svm.SVC(kernel='rbf'),
38             svm.SVC(kernel='poly', degree=2))
39     models = (clf.fit(X_train, y_train) for clf in clfs)
40     titles = ('Linear kernel',
41               'LinearSVC (linear kernel)',
42               'RBF kernel (gamma=0.05, C=5)',
43               'RBF kernel (gamma=0.5, C=3)',
44               'RBF kernel (default)',
45               'Polynomial (degree 2)')
46     arr = []
47     for model, title, clf in zip(models, titles, clfs):
48         scores = cross_val_score(clf, X_train, y_train, cv=5)
49         arr.append([title, scores.mean(), model.score(X_test, y_test)])
50     df = pd.DataFrame(arr, columns=["model", "5CV accuracy", "test accuracy"
   ])
51     print df
```

Comparison result:

	model	5CV accuracy	test accuracy
0	Linear kernel	0.9576	0.9500
1	LinearSVC (linear kernel)	0.9394	0.9324
2	RBF kernel (gamma=0.05, C=5)	0.9824	0.9828
3	RBF kernel (gamma=0.5, C=3)	0.4324	0.4376
4	RBF kernel (default)	0.9614	0.9532
5	Polynomial (degree 2)	0.8540	0.8808

Conclusion:
The comparison result shows that RBF kernel with gamma=0.05 and C=5 performs the best.
5-fold cross validation accuracy forwards 98.24% and test accuracy is 98.28%.
I'll try to use RBF kernel to build SVM model after PCA projection.

Project features into two dimension via PCA and compare models

1. Project X_train into two dimension via PCA
2. Transform X_test into this space
3. Compare three pairs of parameters with RBF kernel

```
54     #svm after pca
55     clfs = (svm.SVC(kernel='rbf', gamma=0.05, C=5),
56             svm.SVC(kernel='rbf', gamma=0.5, C=3),
57             svm.SVC(kernel='rbf'))
58     pca = PCA(n_components=2).fit(X_train, y_train)
59     X_pca = pca.transform(X_train)
60     X_test_pca = pca.transform(X_test)
61     models = (clf.fit(X_pca, y_train) for clf in clfs)
62     titles = ('RBF kernel (gamma=0.05, C=5)',
63               'RBF kernel (gamma=0.5, C=3)',
64               'RBF kernel (default)')
65     arr = []
66     print "SVM after pca:"
67     for model, title, clf in zip(models, titles, clfs):
68         scores = cross_val_score(clf, X_pca, y_train, cv=5)
69         arr.append([title, scores.mean(), model.score(X_test_pca, y_test)])
70     df = pd.DataFrame(arr, columns=["model", "5CV accuracy", "test accuracy"
   ])
71     print df
```

The same with previous section, two comparisons are 5CV and test accuracy.
Compared models:
1. RBF with gamma=0.05, C=5
2. RBF with gamma=0.5, C=3
3. RBF default

Result:

```
SVM after pca:
                          model  5CV accuracy  test accuracy
0  RBF kernel (gamma=0.05, C=5)        0.7978         0.7820
1   RBF kernel (gamma=0.5, C=3)        0.7996         0.7884
2          RBF kernel (default)        0.7994         0.7888
```

Conclusion:
After projecting data into 2 dimension via PCA, the accuracy of different parameters with RBF kernel are similar.
5-fold validation accuracy of training data is around 79.9%, and test accuracy is around 78.8%.
Though, there is a discover. The best model in original space performs no longer the best in low dimension space.
Therefore, data researchers or engineers need to tune kernels and parameters case by case. No one is always the best.

Visualization

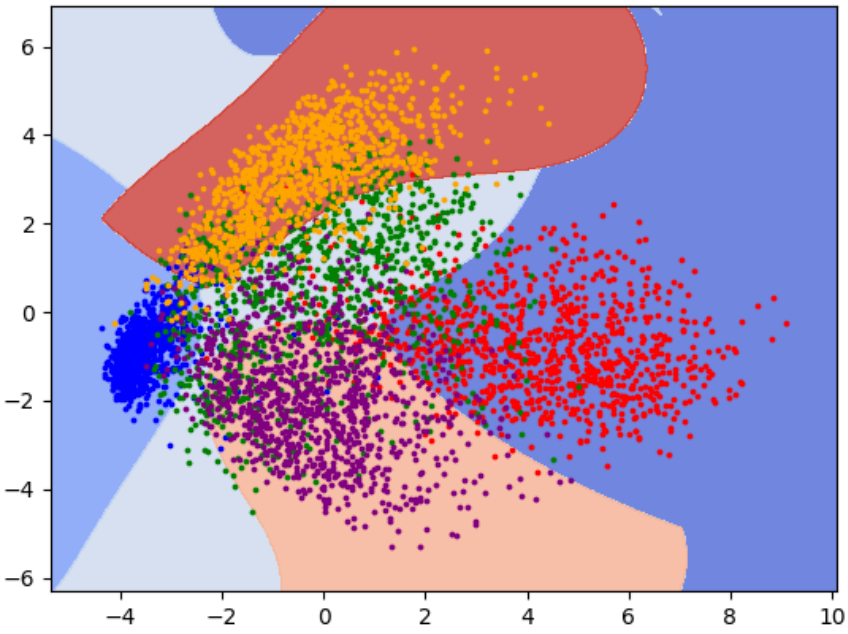
I used Matplotlib for visualization.

“make_meshgrid” is used to create mesh grid.
“plot_decision_boundary” is used to draw contour in mesh grid.

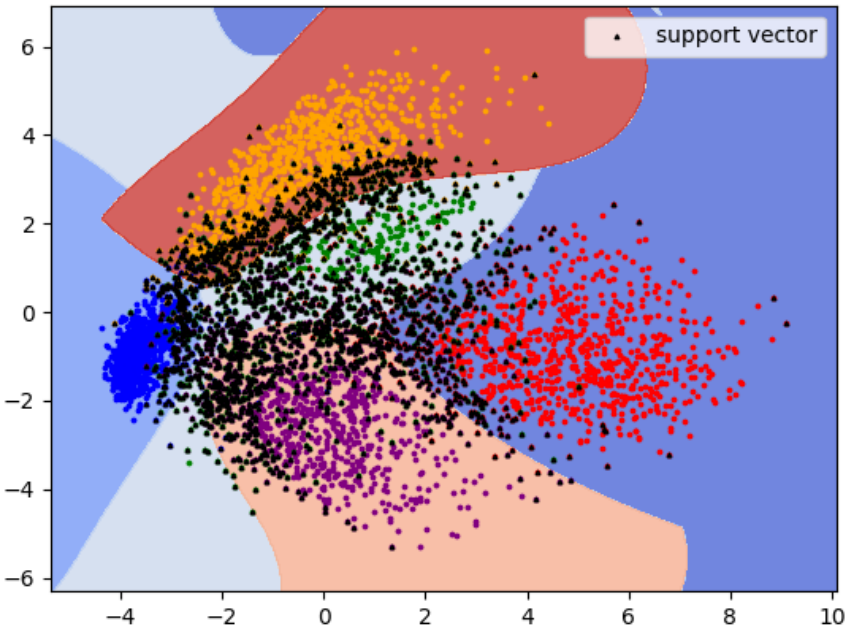
```
18 def make_meshgrid(x, y, h=0.02):
19     x_min, x_max = x.min()-1, x.max()+1
20     y_min, y_max = y.min()-1, y.max()+1
21     xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max,
    h))
22     return xx, yy
23
24 def plot_decision_boundary(ax, clf, xx, yy, **params):
25     Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
26     Z = Z.reshape(xx.shape)
27     return ax.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8, **params)
```

```
72     #plot
73     fig, ax = plt.subplots()
74     xx, yy = make_meshgrid(X_pca[:, 0], X_pca[:, 1])
75     plot_decision_boundary(ax, model, xx, yy)
76     for i in range(len(y_train)):
77         plt.scatter(X_pca[i, 0], X_pca[i, 1], s=3, c=colors[int(y_train[i])-
   1])
78
79
80     plt.savefig('decision_boundary.png')
```

Plot points after PCA projection and corresponding decision boundary.



Plot support vectors.



Conclusion:
In this section, I learned how to visualize decision boundary using mesh grid and contour.
For the visualization result of support vectors plotted, it shows support vectors are surrounding data points, and almost fit decision boundary.