随机森林演示

```
from sklearn.datasets import make_classification

from sklearn.metrics import classification_report

from sklearn.metrics import accuracy_score

from sklearn.cross_validation import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.grid_search import RandomizedSearchCV

from operator import itemgetter

import numpy as np

# 构造一个分类数据集
n_f = 30
inf_f = int(0.6 * n_f)
red_f = int(0.1 * n_f)
rep_f = int(0.1 * n_f)

X,y = make_classification(
    n_samples=500,          # 样本数
    flip_y=0.03,           # 3%改变类别
    n_features=n_f,         # 特征数
    n_informative=inf_f,    # 有信息特征
    n_redundant=red_f,      # 冗余特征
    n_repeated=rep_f,       # 重复特征
    random_state=7)

# 划分训练集测试集
X_train,X_test,y_train,y_test                                    =
train_test_split(X,y,test_size=0.3,random_state=9)

# 构造随机森林
rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train,y_train)

# 分别在训练集测试集上预测
y_train_pred = rf.predict(X_train)
train_score = accuracy_score(y_train,y_train_pred)
```

```
y_test_pred = rf.predict(X_test)
test_score = accuracy_score(y_test,y_test_pred)
print "Train Accuracy = %0.2f" % train_score
print "Test Accuracy = %0.2f" % test_score
Train Accuracy = 1.00
Test Accuracy = 0.79

# 对随机森林进行随机搜索交叉验证
rf = RandomForestClassifier()

n_f = X.shape[1]              # 特征数
sqr_nf = int(np.sqrt(n_f))    # 特征数的平方根

# 构造 20 个随机森林
n_iter = 20

# 每个随机森林中的模型数从 75-200 中随机抽取,纯度度量和最大特征数也随机选取
param = {"n_estimators":np.random.randint(75,200,n_iter),
         "criterion":["gini","entropy"],
         "max_features":[sqr_nf,sqr_nf*2,sqr_nf*3,sqr_nf+10]}

# 构造随机搜索交叉验证:20 个随机森林各进行 5 折交叉验证
grid = RandomizedSearchCV(estimator=rf,
                          param_distributions = param,
                          n_iter=n_iter,
                          cv=5,
                          verbose=1,
                          n_jobs=-1,
                          random_state=77)
grid.fit(X_train,y_train)
Fitting 5 folds for each of 20 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Done  46 tasks    | elapsed:   34.7s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  1.1min finished

# 按评分排序取前 5
scores                                                              =
sorted(grid.grid_scores_,key=itemgetter(1),reverse=True)[:5]

for m,score in enumerate(scores):
    print "M%d,Score = %0.3f" % (m+1,score.mean_validation_score)
    print "Param = {0}".format(score.parameters)

M1,Score = 0.863
Param = {'n_estimators': 199, 'max_features': 5, 'criterion': 'gini'}
```

```
M2,Score = 0.860
Param = {'n_estimators': 143, 'max_features': 5, 'criterion': 'entropy'}
M3,Score = 0.849
Param = {'n_estimators': 123, 'max_features': 5, 'criterion': 'gini'}
M4,Score = 0.849
Param = {'n_estimators': 106, 'max_features': 5, 'criterion': 'gini'}
M5,Score = 0.846
Param = {'n_estimators': 155, 'max_features': 10, 'criterion': 'entropy'}

# 对测试集预测
y_pred = grid.predict(X_test)
print classification_report(y_test,y_pred)
             precision    recall  f1-score   support

          0       0.74      0.91      0.82        69
          1       0.91      0.73      0.81        81

avg / total       0.83      0.81      0.81       150

# 随机森林用于选择特征

# 最佳随机森林模型(即前面排名第一的)
best = grid.best_estimator_

# 特征重要性(用 gini 或 entropy 度量)
f_importance = best.feature_importances_

# 按重要性降序排序，输出前 10 名
i_imp = [(i,imp) for i,imp in enumerate(f_importance)]

i_imp = sorted(i_imp,key=itemgetter(1),reverse=True)[:10]

for imp in i_imp:
    print "Feature %d importance = %0.3f" % (imp[0],imp[1])

Feature 17 importance = 0.103
Feature 16 importance = 0.084
Feature 24 importance = 0.058
Feature 27 importance = 0.049
Feature 10 importance = 0.047
Feature 19 importance = 0.046
Feature 7 importance = 0.043
Feature 3 importance = 0.041
Feature 6 importance = 0.035
```

```
Feature 5 importance = 0.035
```

案例：随机森林识别手写数字

```
from sklearn.datasets import load_digits

digits = load_digits()

print digits.DESCR
```

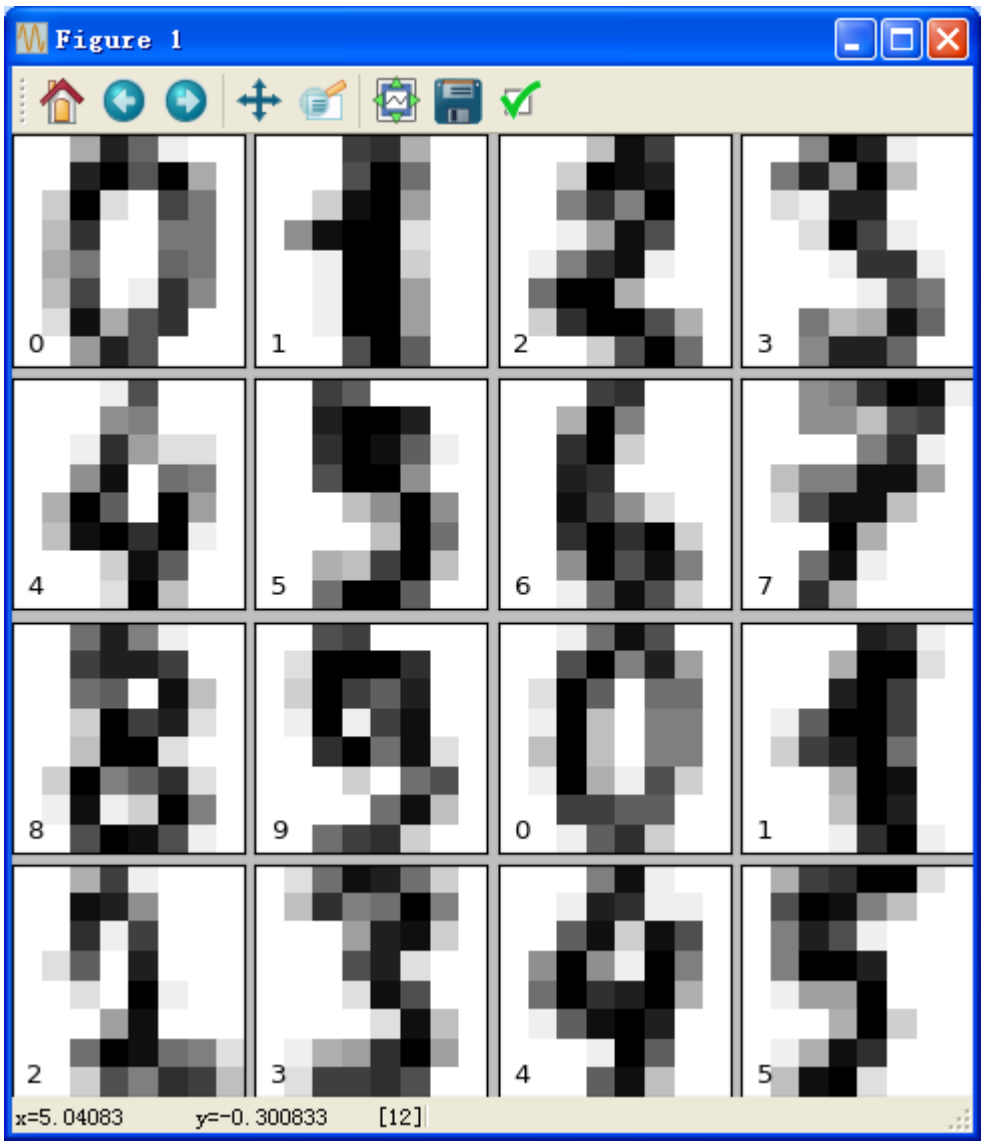
实例数：5620
特征数：64，表示 8x8 图像的 64 个像素，每个像素为 0..16 的整数
类别：10 个

```
# 看看前 16 个数字的图像
fig = plt.figure(figsize=(6,6))

fig.subplots_adjust(left=0,right=1,bottom=0,top=1,hspace=0.05,wspa
ce=0.05)

for i in range(16):
    ax = fig.add_subplot(4,4,i+1,xticks=[],yticks=[])
    ax.imshow(digits.images[i],cmap=plt.cm.binary,
              interpolation='nearest')
    ax.text(0,7,str(digits.target[i]))

plt.show()
```




```
X = digits.data

y = digits.target

X_train,X_test,y_train,y_test                                    =
train_test_split(X,y,random_state=0)

rf = RandomForestClassifier(n_estimators=1000)

rf.fit(X_train,y_train)

y_pred = rf.predict(X_test)

print classification_report(y_pred,y_test)
             precision    recall  f1-score   support
```

```
          0       1.00      0.97      0.99        38
          1       1.00      0.96      0.98        45
          2       0.95      1.00      0.98        42
          3       0.98      0.98      0.98        45
          4       0.97      1.00      0.99        37
          5       0.98      0.98      0.98        48
          6       1.00      1.00      1.00        52
          7       1.00      0.96      0.98        50
          8       0.94      0.98      0.96        46
          9       0.98      0.98      0.98        47

avg / total       0.98      0.98      0.98       450
```

```
# 混淆矩阵
from sklearn.metrics import confusion_matrix

import seaborn as sns

cm = confusion_matrix(y_test,y_pred)

sns.heatmap(cm.T,square=True,annot=True,fmt='d',cbar=False)

plt.xlabel('true label')

plt.ylabel('predicted label')

plt.show()
```

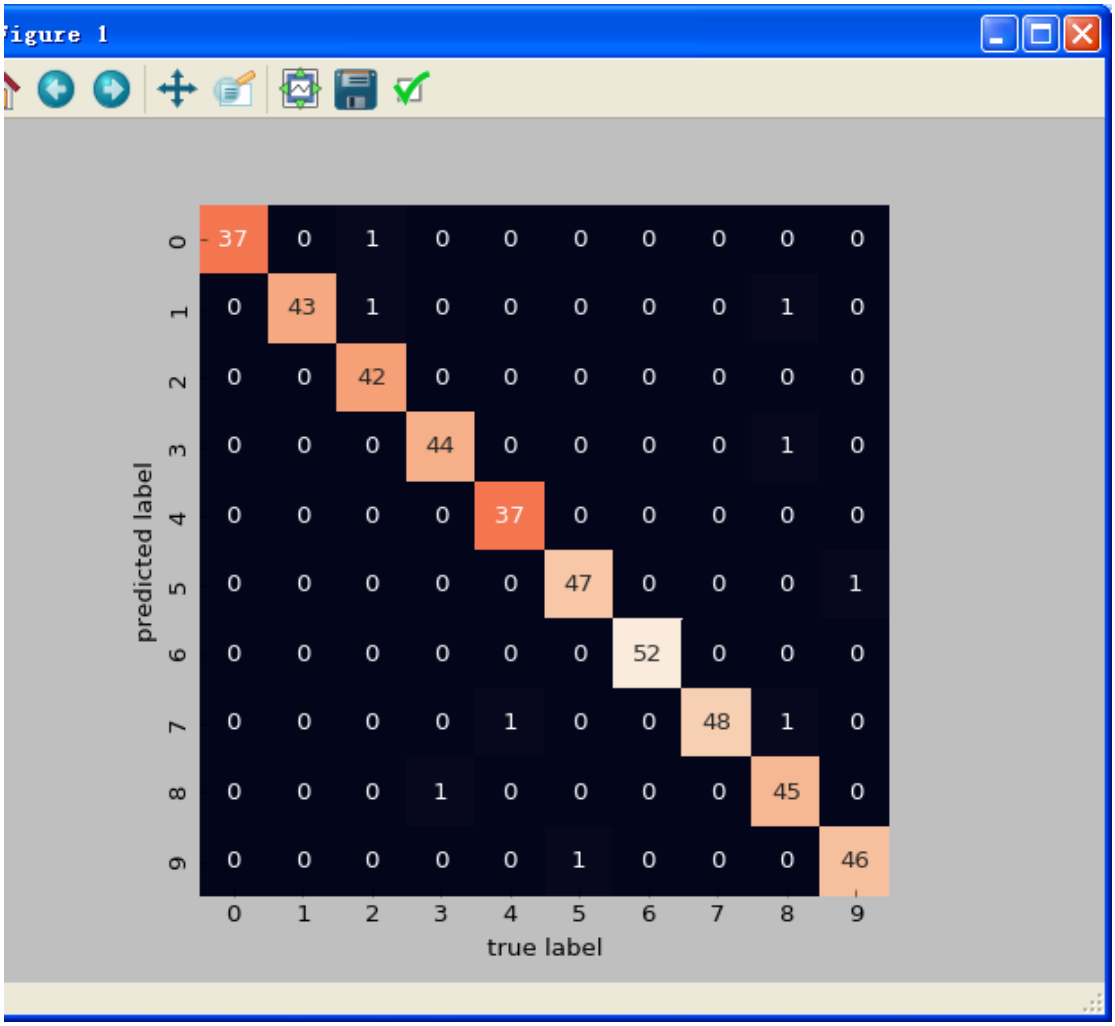


Figure 1
predicted label
true label
0 1 2 3 4 5 6 7 8 9
37 0 1 0 0 0 0 0 0 0
0 43 1 0 0 0 0 0 1 0
0 0 42 0 0 0 0 0 0 0
0 0 0 44 0 0 0 0 1 0
0 0 0 0 37 0 0 0 0 0
0 0 0 0 0 47 0 0 0 1
0 0 0 0 0 0 52 0 0 0
0 0 0 0 1 0 0 48 1 0
0 0 0 1 0 0 0 0 45 0
0 0 0 0 0 1 0 0 0 46