EE559 Homework 8

Jingquan Yan

USC ID: 1071912676

Email: jingquan@usc.edu

EE559 repository: Github

Problem 1 :

```
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
import plotSVMBoundaries


def read_data(data_dir):
    with open(data_dir, 'r') as dataset:
        reader = dataset.readlines()
        data = [rows.split(',') for rows in reader]
        data = [list(map(float, e)) for e in data]
    return np.array(data)


# a
training_data = read_data("HW8_1_csv/train_x.csv")
training_label = read_data("HW8_1_csv/train_y.csv")
training_label = np.hstack(training_label.T)

# c=1
svm_c_1 = SVC(kernel='linear', C=1)
svm_c_1.fit(training_data, training_label)
svm_c_1_predict = svm_c_1.predict(training_data)
svm_c_1_accu = accuracy_score(training_label, svm_c_1_predict)
print('The accuracy when c=1 is: ', svm_c_1_accu)
plotSVMBoundaries.plotSVMBoundaries(training_data, training_label, svm_c_1)

# c=100
svm_c_100 = SVC(kernel='linear', C=100)
svm_c_100.fit(training_data, training_label)
svm_c_100_predict = svm_c_100.predict(training_data)
svm_c_100_accu = accuracy_score(training_label, svm_c_100_predict)
print('The accuracy when c=100 is: ', svm_c_100_accu)
plotSVMBoundaries.plotSVMBoundaries(training_data, training_label, svm_c_100)

# b
support_vector_c_100 = svm_c_100.support_vectors_
plotSVMBoundaries.plotSVMBoundaries(training_data, training_label, svm_c_100,
support_vector_c_100)
w_vector = svm_c_100.coef_
w0_vector = svm_c_100.intercept_
```

```
print("The support vector is: (", w_vector[0][1], w_vector[0][0], w0_vector[0],
")")
print("The decision boundary equation is : ", w_vector[0][0], "x1+", w_vector[0]
[1], "x2+", w0_vector[0], "= 0")

# c
gx = [m[1] * w_vector[0][1] + m[0] * w_vector[0][0] + w0_vector[0] for m in
support_vector_c_100]
print("The g(x) value of the support vectors are: ", gx)

# d
training_data_rbf = read_data("HW8_2_csv/train_x.csv")
training_label_rbf = read_data("HW8_2_csv/train_y.csv")
training_label_rbf = np.hstack(training_label_rbf.T)

# rbf C=50
rbf_c_50 = SVC(kernel='rbf', C=50)
rbf_c_50.fit(training_data_rbf, training_label_rbf)
rbf_c_50_predict = rbf_c_50.predict(training_data_rbf)
rbf_c_50_accu = accuracy_score(training_label_rbf, rbf_c_50_predict)
print('The SVM accuracy when c=50 with rbf is: ', rbf_c_50_accu)
plotSVMBoundaries.plotSVMBoundaries(training_data_rbf, training_label_rbf,
rbf_c_50)

# rbf C=5000
rbf_c_50 = SVC(kernel='rbf', C=5000, gamma='auto')
rbf_c_50.fit(training_data_rbf, training_label_rbf)
rbf_c_50_predict = rbf_c_50.predict(training_data_rbf)
rbf_c_50_accu = accuracy_score(training_label_rbf, rbf_c_50_predict)
print('The SVM accuracy when c=5000 with rbf is: ', rbf_c_50_accu)
plotSVMBoundaries.plotSVMBoundaries(training_data_rbf, training_label_rbf,
rbf_c_50)

# e
# gamma = 10
rbf_c_50 = SVC(kernel='rbf', gamma=10)
rbf_c_50.fit(training_data_rbf, training_label_rbf)
rbf_c_50_predict = rbf_c_50.predict(training_data_rbf)
rbf_c_50_accu = accuracy_score(training_label_rbf, rbf_c_50_predict)
print('The SVM accuracy when c=5000 with rbf is: ', rbf_c_50_accu)
plotSVMBoundaries.plotSVMBoundaries(training_data_rbf, training_label_rbf,
rbf_c_50)

# gamma = 50
rbf_c_50 = SVC(kernel='rbf', gamma=50)
rbf_c_50.fit(training_data_rbf, training_label_rbf)
rbf_c_50_predict = rbf_c_50.predict(training_data_rbf)
rbf_c_50_accu = accuracy_score(training_label_rbf, rbf_c_50_predict)
print('The SVM accuracy when c=5000 with rbf is: ', rbf_c_50_accu)
plotSVMBoundaries.plotSVMBoundaries(training_data_rbf, training_label_rbf,
rbf_c_50)

# gamma = 500
rbf_c_50 = SVC(kernel='rbf', gamma=500)
rbf_c_50.fit(training_data_rbf, training_label_rbf)
rbf_c_50_predict = rbf_c_50.predict(training_data_rbf)
rbf_c_50_accu = accuracy_score(training_label_rbf, rbf_c_50_predict)
print('The SVM accuracy when c=5000 with rbf is: ', rbf_c_50_accu)
```

```
plotSVMBoundaries.plotSVMBoundaries(training_data_rbf, training_label_rbf,
rbf_c_50)
```

Problem 2:

```
import numpy as np
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import plotSVMBoundaries


def read_data(data_dir, start=0, end=1):
    with open(data_dir, 'r') as dataset:
        reader = dataset.readlines()
        data = [rows.split(',')[start:end] for rows in reader]
        data = [list(map(float, e)) for e in data]
    return np.array(data)


training_data = read_data("wine_csv/feature_train.csv", 0, 2)
training_label = read_data("wine_csv/label_train.csv")
training_label = np.hstack(training_label.T)

# a
cross_validation_5 = StratifiedKFold(n_splits=5, shuffle=True)
svm_rbf = SVC(kernel='rbf', gamma=1)
cross_validation_accu = cross_val_score(svm_rbf, training_data, training_label,
cv=cross_validation_5)
print('The accuracy for each fold respectively: ', cross_validation_accu)
print("The mean of cross-validation accuracy is:",
np.mean(cross_validation_accu))

# b
cross_validation_range = StratifiedKFold(n_splits=5, shuffle=True)
set_c = np.logspace(-3, 3, 50)
set_gamma = np.logspace(-3, 3, 50)
acc = np.zeros([50, 50])
dev = np.zeros([50, 50])
for i in range(len(set_c)):
    for j in range(len(set_gamma)):
        svm_rbf = SVC(kernel='rbf', C=set_c[i], gamma=set_gamma[j])
        cross_validation_accu = cross_val_score(svm_rbf, training_data,
training_label, cv=cross_validation_range)
        acc[i][j] = np.mean(cross_validation_accu)
        dev[i][j] = np.std(cross_validation_accu)

imgplot = plt.imshow(acc)
plt.colorbar()
plt.show()
print("The accuracy is the best when C = {} and gamma =
{}".format(set_c[np.where(acc == np.max(acc))[0]],

set_gamma[np.where(acc == np.max(acc))[1]]))
print("The best accuracy and deviation are:", np.max(acc), dev[np.where(acc ==
np.max(acc))])

# c
```

```
set_c = np.logspace(-3, 3, 50)
set_gamma = np.logspace(-3, 3, 50)
pairs = np.zeros([20, 2])
val = np.zeros([20, 2])
for t in range(20):
    acc = np.zeros([50, 50])
    dev = np.zeros([50, 50])
    cross_validation_range = StratifiedKFold(n_splits=5, shuffle=True)
    for i in range(len(set_c)):
        for j in range(len(set_gamma)):
            svm_rbf = SVC(kernel='rbf', C=set_c[i], gamma=set_gamma[j])
            cross_validation_accu = cross_val_score(svm_rbf, training_data,
training_label, cv=cross_validation_range)
            acc[i][j] = np.mean(cross_validation_accu)
            dev[i][j] = np.std(cross_validation_accu)
    print(np.where(acc == np.max(acc)))
    print(np.where(acc == np.max(acc))[0][0], np.where(acc == np.max(acc))[1]
[0])
    pairs[t][0] = set_c[np.where(acc == np.max(acc))[0][0]]
    pairs[t][1] = set_gamma[np.where(acc == np.max(acc))[1][0]]
    val[t][0] = acc[np.where(acc == np.max(acc))[0][0], np.where(acc ==
np.max(acc))[1][0]]
    val[t][1] = dev[np.where(acc == np.max(acc))[0][0], np.where(acc ==
np.max(acc))[1][0]]
print(pairs)
print(val)

# d
# [0.87777778 0.1237281 ] [1.93069773e+01 3.72759372e-01]
optimal_c = 1.93069773e+01
optimal_gamma = 3.72759372e-01
test_data = read_data("wine_csv/feature_test.csv", 0, 2)
test_label = read_data("wine_csv/label_test.csv")
test_label = np.hstack(test_label.T)

test_rbf = SVC(kernel='rbf', gamma=optimal_gamma, C=optimal_c)
test_rbf.fit(test_data, test_label)
test_predict = test_rbf.predict(test_data)
test_accu = accuracy_score(test_label, test_predict)

print("The accuracy of test dataset is:", test_accu)
plotSVMBoundaries.plotSVMBoundaries(test_data, test_label, test_rbf)
```