Samantha Rothman

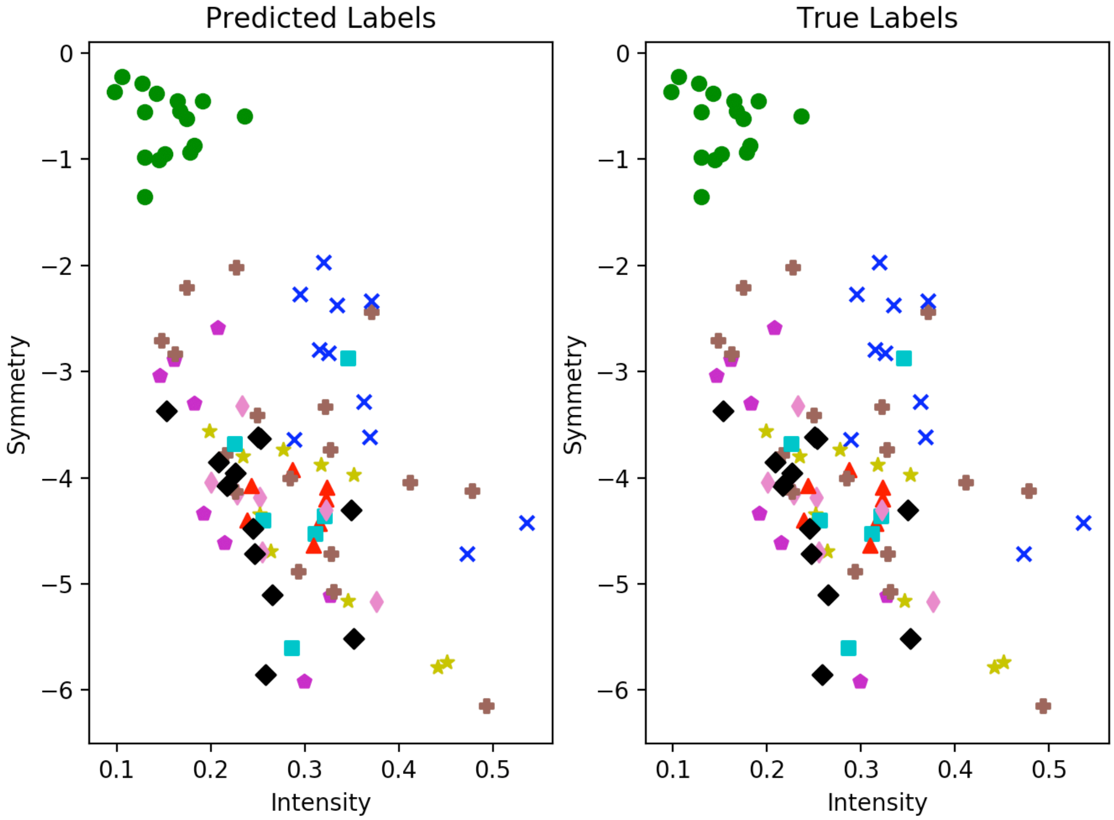
Homework 5

1. Multiclass k-nearest neighbor classifier

   a) When run on the data from the train file given, the cross-validation error for k = 1 was 0.002, k = 11 was 0.01, k = 21 was 0.012, and k = 31was 0.016.

   b) Using all data from the train file as training data, 100 points from the test file as testing data, and optimal k = 1, the error was 0.0.

   c)

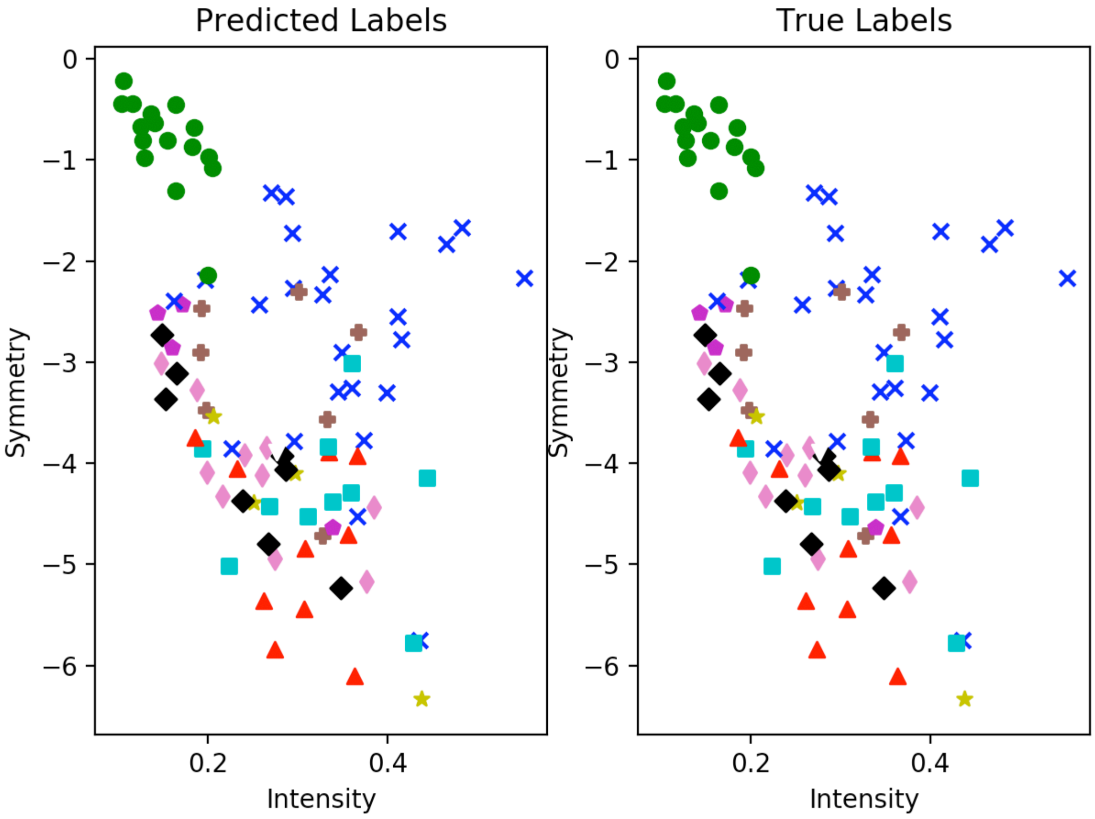


2. Multiclass RBF Classifier

   a) When run on the data from the train file given, the cross-validation error for r = 0.01 was 0.003, r = 0.05 was 0.0, r = 0.1 was 0.001, r = 0.5 was 0.001, and r = 1 was 0.065.

   b) The best value of r is 0.05 because it has the lowest error. Using all data from the train file as training data, 100 points from the test file as testing data, and the best r, the error was 0.0.

c)


Predicted Labels
True Labels
0
−1
−2
−3
−4
−5
−6
Symmetry
0.2
0.4
Intensity

Samantha Rothman

Code for Question 1:

```
import numpy as np
import matplotlib.pyplot as plt


file_test = "/Users/samantharothman/Downloads/features.small.test"
file_train = "/Users/samantharothman/Downloads/features.small.train"

testdata = np.loadtxt(file_test)
traindata = np.loadtxt(file_train)

def nearestneighbors(test, train, k):
    check = []
    for i in test:
        distances = []
        for j in train:
            dist = np.linalg.norm(i-j)
            distances.append((j, dist))
        distances.sort(key=lambda tup: tup[1])
        neighbors = distances[0:k]
        output = []
        for neighbor in neighbors:
            label = neighbor[0][0]
            output.append(label)
        prediction = max(set(output), key=output.count)
        check.append(prediction)
    return check

def train90(data):
    train = np.empty((0,3), float)
    test = np.empty((0,3), float)
    rows = np.random.choice(range(1000), 100, replace=False)
    for num in range(1000):
        arr = np.array([data[num, :]])
        if num in rows:
            test = np.append(test, arr, axis = 0)
        else:
            train = np.append(train, arr, axis= 0)
    return train, test, rows

def error(pred, data, rows):
    count = 0
    miss = 0
    rows.sort()
    for num in rows:
        real = data[num][0]
        val = pred[count]
        if real != val:
            miss += 1
        count += 1
    return miss/(len(rows))

knn = [1, 11, 21, 31]
avg_errors = []
for k in knn:
    errors = []
    for i in range(10):
        train, test, rows = train90(traindata)
        pred = nearestneighbors(test, train, k)
        err = error(pred, traindata, rows)
        errors.append(err)
    avg = np.mean(errors)
    avg_errors.append(avg)
print(avg_errors)

# best k is 1
# test 100 from test data file using all the training data from train file
train2, test2, rows2 = train90(testdata)
totalpred = nearestneighbors(test2, traindata, 1)
totalerr = error(totalpred, testdata, rows2)
print("Error: ", totalerr)

markers = {0.0: "x", 1.0: "o", 2.0: "^", 3.0: "s", 4.0: "p", 5.0: "*",
           6.0: "D", 7.0: "v", 8.0: "P", 9.0: "d"}
colors = {0.0: "b", 1.0: "g", 2.0: "r", 3.0: "c", 4.0: "m", 5.0: "y",
           6.0: "k", 7.0: "w", 8.0: 'tab:brown', 9.0: 'tab:pink'}

fig, (fig1, fig2) = plt.subplots(1, 2)

# predicted labels
for point in test2:
    label = point[0]
    mark = markers[label]
    col = colors[label]
    intense = point[1]
    sym = point[2]
    l = str(label)
    fig1.scatter(intense,sym, marker =mark, color = col)

fig1.set_xlabel('Intensity')
fig1.set_ylabel('Symmetry')
fig1.set_title('Predicted Labels')

# true labels
rows2.sort()
for num in rows2:
    label = testdata[num][0]
    intense = testdata[num][1]
    sym = testdata[num][2]
    mark = markers[label]
    col = colors[label]
    fig2.scatter(intense,sym, marker =mark, color = col)

fig2.set_xlabel('Intensity')
fig2.set_ylabel('Symmetry')
fig2.set_title('True Labels')

plt.show()
```

Samantha Rothman

Code for Question 2:

```
import numpy as np
import matplotlib.pyplot as plt


file_test = "/Users/samantharothman/Downloads/features.small.test"
file_train = "/Users/samantharothman/Downloads/features.small.train"

testdata = np.loadtxt(file_test)
traindata = np.loadtxt(file_train)

def rbf(test, train, r):
    check = []
    for xi in test:
        output = []
        for xj in train:
            dist = (np.linalg.norm(xi-xj))/r
            a = np.exp(-0.5*(dist**2))
            output.append((a, xj[0]))
        num = 0
        denom = 0
        for a in output:
            num += a[0] * a[1]
            denom += a[0]
        g = num/denom
        check.append(round(g))
    return check

def train90(data):
    train = np.empty((0,3), float)
    test = np.empty((0,3), float)
    rows = np.random.choice(range(1000), 100, replace=False)
    for num in range(1000):
        arr = np.array([data[num, :]])
        if num in rows:
            test = np.append(test, arr, axis = 0)
        else:
            train = np.append(train, arr, axis= 0)
    return train, test, rows

def error(pred, data, rows):
    count = 0
    miss = 0
    rows.sort()
    for num in rows:
        real = data[num][0]
        val = pred[count]
        if real != val:
            miss += 1
        count += 1
    return miss/(len(rows))


rval = [0.01, 0.05, 0.1, 0.5, 1]
avg_errors = []
for r in rval:
    errors = []
    for i in range(10):
        train, test, rows = train90(traindata)
        pred = rbf(test, train, r)
        err = error(pred, traindata, rows)
        errors.append(err)
    avg = np.mean(errors)
    avg_errors.append(avg)
print(avg_errors)


# best r = 0.05

# run rbf with 100 test points from testdata as test, and trainingdata as train
train2, test2, rows2 = train90(testdata)
totalpred = rbf(test2, traindata, 0.05)
totalerr = error(totalpred, testdata, rows2)

print("Error: ", totalerr)

markers = {0.0: "x", 1.0: "o", 2.0: "^", 3.0: "s", 4.0: "p", 5.0: "*",
           6.0: "D", 7.0: "v", 8.0: "P", 9.0: "d"}
colors = {0.0: "b", 1.0: "g", 2.0: "r", 3.0: "c", 4.0: "m", 5.0: "y",
           6.0: "k", 7.0: "w",  8.0: 'tab:brown', 9.0: 'tab:pink'}


fig, (fig1, fig2) = plt.subplots(1, 2)

# predicted labels
for point in test2:
    label = point[0]
    mark = markers[label]
    col = colors[label]
    intense = point[1]
    sym = point[2]
    fig1.scatter(intense,sym, marker =mark, color = col)

fig1.set_xlabel('Intensity')
fig1.set_ylabel('Symmetry')
fig1.set_title('Predicted Labels')

# true labels
rows2.sort()
for num in rows2:
    label = testdata[num][0]
    intense = testdata[num][1]
    sym = testdata[num][2]
    mark = markers[label]
    col = colors[label]
    fig2.scatter(intense,sym, marker =mark, color = col)

fig2.set_xlabel('Intensity')
fig2.set_ylabel('Symmetry')
fig2.set_title('True Labels')

plt.show()
```