```
1 | 11111
 2 Wine Dataset (for question 2)
 3 Name: Haolun Cheng
 4 USCID: 1882563827
 5 EE559 HW2
 6 .....
 7
8 import csv
9 import numpy as np
10 import matplotlib.pyplot as plt
11 from plotDecBoundaries import plotDecBoundaries
12
13 clas1xtotal = []
14 clas1ytotal = []
15 clas2xtotal = []
16 clas2ytotal = []
17 clas3xtotal = []
18 clas3vtotal = []
19 allclasstwofeatures = []
20 allclasslabels = []
21 allclassmeans = []
22 feature1 = []
23 feature2 = []
24 | feature3 = []
25 label1 = []
26 \mid label2 = []
27 | label3 = []
28
29 # Open train csv file for training the classifier
30 with open('wine_train.csv', 'r') as train:
31
       training set = csv.reader(train)
32
33
       # Train
34
       for line in training set:
35
           x, y, label = line[0], line[1], line[-1]
36
           allclasstwofeatures.append((float(x), float(y)))
37
           allclasslabels.append(label)
           if label == '1':
38
39
               clas1xtotal.append(float(x))
40
               clas1ytotal.append(float(y))
41
               feature1.append((float(x), float(y)))
42
               label1.append(label)
43
           elif label == '2':
               clas2xtotal.append(float(x))
44
45
               clas2ytotal.append(float(y))
46
               feature2.append((float(x), float(y)))
47
               label2.append(label)
48
           else:
49
               clas3xtotal.append(float(x))
50
               clas3ytotal.append(float(y))
51
               feature3.append((float(x), float(y)))
52
               label3.append(label)
53
54 train.close()
55
56 # Compute mean for each class
57 clas1xmean = np.mean(clas1xtotal)
58 clas1ymean = np.mean(clas1ytotal)
59
```

localhost:4649/?mode=python 1/6

```
60 clas2xmean = np.mean(clas2xtotal)
 61 clas2ymean = np.mean(clas2ytotal)
62
63 clas3xmean = np.mean(clas3xtotal)
 64 clas3ymean = np.mean(clas3ytotal)
65
 66 clas1_mean_point = np.array((clas1xmean, clas1ymean))
 67 clas2_mean_point = np.array((clas2xmean, clas2ymean))
 68 clas3_mean_point = np.array((clas3xmean, clas3ymean))
69
 70 # Calculate the linear equations for each pair of points
 71 # class1 and class2 equation
 72 | slope1 = (clas2ymean - clas1ymean) / (clas2xmean - clas1xmean) # get the
    slope
73 slope12 = -1 / slope1
 74 midPoint1x = (clas2xmean + clas1xmean) / 2
 75 midPoint1y = (clas2ymean + clas1ymean) / 2
 76 intercept12 = midPoint1y - (slope12 * midPoint1x)
 77 clas1sign = (slope12 * clas1xmean + intercept12) - clas1ymean
 78 | q12 1, q12 2 = 0, 0
 79 if clas1sign > 0:
80
        g12_1 = 1
81
        g12_2 = -1
82 elif clas1sign < 0:
 83
        g12_1 = -1
 84
        g12_2 = 1
85
 86 # class1 and class3 equation
 87 slope2 = (clas3ymean - clas1ymean) / (clas3xmean - clas1xmean) # get the
    slope
 88 | slope13 = -1 / slope2
 89 midPoint2x = (clas3xmean + clas1xmean) / 2
90 midPoint2y = (clas3ymean + clas1ymean) / 2
 91 intercept13 = midPoint2y - (slope13 * midPoint2x)
92 clas3sign = (slope13 * clas3xmean + intercept13) - clas3ymean
93 | g13_1, g13_3 = 0, 0
94 if clas3sign > 0:
95
        g13_3 = 1
96
        g13_1 = -1
97 elif clas3sign < 0:
98
        g13_3 = -1
99
        g13_1 = 1
100
101 # class2 and class3 equation
102 slope3 = (clas3ymean - clas2ymean) / (clas3xmean - clas2xmean) # get the
    slope
103 \text{ slope23} = -1 / \text{slope3}
104 \mid midPoint3x = (clas2xmean + clas3xmean) / 2
105 midPoint3y = (clas2ymean + clas3ymean) / 2
106 intercept23 = midPoint3y - (slope23 * midPoint3x)
107 clas2sign = (slope23 * clas2xmean + intercept23) - clas2ymean
108 g23_2, g23_3 = 0, 0
109 if clas2sign > 0:
110
        g23_2 = 1
111
        q23 \ 3 = -1
112 elif clas2sign < 0:
113
        q23 2 = -1
114
        g23_3 = 1
115
116 # Classify data points (training set)
```

localhost:4649/?mode=python 2/6

```
117 countTrainingError = 0
118 totalTrainingPoints = 0
119 with open('wine_train.csv', 'r') as training:
120
        train_set = csv.reader(training)
121
122
        for line in train_set:
123
            count0, count1, count2, count3 = 0, 0, 0
124
            totalTrainingPoints += 1
            x, y, label = line[0], line[1], line[-1]
125
            result12 = (slope12 * float(x) + intercept12) - float(y)
126
127
            result13 = (slope13 * float(x) + intercept13) - float(y)
128
            result23 = (slope23 * float(x) + intercept23) - float(y)
129
130
            # Use the OvO rule
131
            if result12 > 0:
132
                if g12_1 == 1:
133
                     count1 += 1
134
                else:
135
                     count2 += 1
136
            elif result12 < 0:
137
                if g12_1 == -1:
138
                     count1 += 1
139
                else:
140
                    count2 += 1
141
            else:
142
                count0 += 1
143
144
            if result13 > 0:
145
                if q13 1 == 1:
146
                     count1 += 1
147
                else:
148
                    count3 += 1
149
            elif result13 < 0:
150
                if g13_1 == -1:
151
                     count1 += 1
152
                else:
153
                     count3 += 1
154
            else:
155
                count0 += 1
156
157
            if result23 > 0:
158
                if q23 2 == 1:
159
                     count2 += 1
160
                else:
161
                    count3 += 1
            elif result23 < 0:
162
163
                if g23_2 == -1:
164
                     count2 += 1
165
                else:
166
                     count3 += 1
167
            else:
168
                count0 += 1
169
170
            # Classify to class
171
            if count0 != 0:
172
                countTrainingError += count0
173
            else:
174
                if count1 > count2 and count1 > count3:
                     if label != '1':
175
                         countTrainingError += 1
176
```

localhost:4649/?mode=python 3/6

```
177
                if count2 > count1 and count2 > count3:
178
                    if label != '2':
179
                         countTrainingError += 1
                if count3 > count2 and count3 > count1:
180
181
                    if label != '3':
                         countTrainingError += 1
182
183 training close()
184
185 # Fine error rate for training set
186 error_rate = float(countTrainingError) / float(totalTrainingPoints)
187 print("Error rate for the training set: " + str(error_rate))
188
189 # Classify data points (test set)
190 countTestError = 0
191 totalTestPoints = 0
192 with open('wine_test.csv', 'r') as test:
193
        test_set = csv.reader(test)
194
195
        for line in test set:
            count0, count1, count2, count3 = 0, 0, 0
196
197
            totalTestPoints += 1
            x, y, label = line[0], line[1], line[-1]
198
199
            testPoint = np.array((x, y))
200
            result12 = (slope12 * float(x) + intercept12) - float(y)
            result13 = (slope13 * float(x) + intercept13) - float(y)
201
202
            result23 = (slope23 * float(x) + intercept23) - float(y)
203
204
            # Use the OvO rule
205
            if result12 > 0:
206
                if g12_1 == 1:
207
                    count1 += 1
208
                else:
209
                    count2 += 1
210
            elif result12 < 0:
211
                if g12_1 == -1:
212
                    count1 += 1
213
                else:
214
                    count2 += 1
215
            else:
216
                count0 += 1
217
218
            if result13 > 0:
                if g13_1 == 1:
219
220
                    count1 += 1
221
                else:
222
                    count3 += 1
            elif result13 < 0:
223
224
                if g13_1 == -1:
225
                    count1 += 1
226
                else:
227
                    count3 += 1
228
            else:
229
                count0 += 1
230
231
            if result23 > 0:
232
                if q23 2 == 1:
233
                    count2 += 1
234
                else:
235
                    count3 += 1
            elif result23 < 0:
236
```

localhost:4649/?mode=python 4/6

```
2/7/22, 9:53 PM
                                               hw2_question2.py
 237
                  if g23_2 == -1:
 238
                      count2 += 1
 239
                  else:
 240
                      count3 += 1
 241
              else:
 242
                  count0 += 1
 243
 244
              # Classify to class
 245
              if count0 != 0:
 246
                  countTestError += count0
 247
              else:
 248
                  if count1 > count2 and count1 > count3:
                      if label != '1':
 249
 250
                           countTestError += 1
 251
                  if count2 > count1 and count2 > count3:
                      if label != '2':
 252
 253
                           countTestError += 1
 254
                  if count3 > count2 and count3 > count1:
                      if label != '3':
 255
 256
                           countTestError += 1
 257
 258 test.close()
 259
 260 # Fine error rate for test set
 261 error_rate = float(countTestError) / float(totalTestPoints)
 262 print("Error rate for the test set: " + str(error_rate))
 263
 264 # Plot the data points
 265 \times Axis1 = [i[0] \text{ for } i \text{ in feature1}]
 266 \text{ yAxis1} = [i[1] \text{ for } i \text{ in feature1}]
 267 \times Axis2 = [i[0] \text{ for } i \text{ in feature2}]
 268 yAxis2 = [i[1] for i in feature2]
 269 \times Axis3 = [i[0] \text{ for } i \text{ in feature3}]
 270 \text{ yAxis3} = [i[1] \text{ for } i \text{ in feature3}]
 271 plt.plot(xAxis1, yAxis1, 'r.', xAxis2, yAxis2, 'b^', xAxis3, yAxis3, 'g.')
 272 plt.xlabel('Feature1')
 273 plt.ylabel('Feature2')
 274 plt.title('Feature Plot of all Elements')
 275 plt.show()
 276
 277 # Class 1 & 2 decision boundaries and regions variables
 278 classmeans12 = [[clas1xmean, clas1ymean], [clas2xmean, clas2ymean]]
 279 features12 = feature1 + feature2
 280 labels12 = label1 + label2
 281 twofeatures12 = np.array(features12).astype(float)
 282 claslabels12 = np.array(labels12).astype(float)
 283 samplemeans12 = np.array(classmeans12).astype(float)
 284
 285 # Class 1 & 3 decision boundaries and regions variables
 286 classmeans13 = [[clas1xmean, clas1ymean], [clas3xmean, clas3ymean]]
 287 | features13 = feature1 + feature3
 288 labels13 = label1 + label3
 289 twofeatures13 = np.array(features13).astype(float)
 290 claslabels13 = np.array(labels13).astype(float)
 291 samplemeans13 = np.array(classmeans13).astype(float)
 293 # Class 2 & 3 decision boundaries and regions variables
 294 classmeans23 = [[clas2xmean, clas2ymean], [clas3xmean, clas3ymean]]
```

localhost:4649/?mode=python 5/6

295 features23 = feature2 + feature3

296 labels23 = label2 + label3

2/7/22, 9:53 PM hw2_question2.py

```
297 twofeatures23 = np.array(features23).astype(float)
298 claslabels23 = np.array(labels23).astype(float)
299 samplemeans23 = np.array(classmeans23).astype(float)
300
301 #Final decision boundaries and regions variables
302 allclassmeans = [[clas1xmean, clas1ymean], [clas2xmean, clas2ymean],
    [clas3xmean, clas3ymean]]
303 twofeatures = np.array(allclasstwofeatures).astype(float)
304 claslabels = np.array(allclasslabels).astype(float)
305 samplemeans = np.array(allclassmeans).astype(float)
306
307 # Plot the decision boundaries
308 plotDecBoundaries(twofeatures, claslabels, samplemeans12, class1=1, class2=2)
309 plotDecBoundaries(twofeatures, claslabels, samplemeans13, class1=1, class2=3)
310 plotDecBoundaries(twofeatures, claslabels, samplemeans23, class1=2, class2=3)
311 plotDecBoundaries(twofeatures, claslabels, samplemeans)
```

localhost:4649/?mode=python 6/6

```
2 ## EE559 HW1, Prof. Jenkins
 3 ## Created by Arindam Jati
 4 ## Tested in Python 3.6.3, OSX El Capitan, and subsequent versions
 6
 7 import numpy as np
8 import matplotlib.pyplot as plt
9 from scipy.spatial.distance import cdist
10
11 def plotDecBoundaries(training, label_train, sample_mean, class1 = 0, class2 =
  1):
12
13
      #Plot the decision boundaries and data points for minimum distance to
      #class mean classifier
14
15
      # training: traning data
16
17
      # label_train: class lables correspond to training data
      # sample mean: mean vector for each class
18
19
20
      # Total number of classes
21
      nclass = max(np.unique(label train))
22
23
      # Total number of means
      nmean = np.unique(sample mean).shape[0]
24
25
26
      # Set the feature range for ploting
27
      \max x = \text{np.ceil}(\max(\text{training}[:, 0])) + 1
28
      min_x = np.floor(min(training[:, 0])) - 1
      max_y = np.ceil(max(training[:, 1])) + 1
29
      min y = np.floor(min(training[:, 1])) - 1
30
31
      xrange = (min_x, max_x)
32
33
      yrange = (min y, max y)
34
35
      # step size for how finely you want to visualize the decision boundary.
36
      inc = 0.005
37
38
      # generate grid coordinates. this will be the basis of the decision
39
      # boundary visualization.
40
      (x, y) = np.meshgrid(np.arange(xrange[0], xrange[1]+inc/100, inc),
  np.arange(yrange[0], yrange[1]+inc/100, inc))
41
42
      \# size of the (x, y) image, which will also be the size of the
43
      # decision boundary image that is used as the plot background.
44
      image_size = x.shape
45
      xy = np.hstack( (x.reshape(x.shape[0]*x.shape[1], 1, order='F'),
  y.reshape(y.shape[0]*y.shape[1], 1, order='F')) ) # make (x,y) pairs as a
   bunch of row vectors.
46
47
      # distance measure evaluations for each (x,y) pair.
48
      dist_mat = cdist(xy, sample_mean)
49
      pred_label = np.argmin(dist_mat, axis=1)
50
      # reshape the idx (which contains the class label) into an image.
51
52
      decisionmap = pred_label.reshape(image_size, order='F')
53
54
      #show the image, give each coordinate a color according to its class label
```

localhost:4649/?mode=python 1/2

```
55
       plt.imshow(decisionmap, extent=[xrange[0], xrange[1], yrange[0],
  yrange[1]], origin='lower')
56
57
       # plot the class training data.
58
       plt.plot(training[label train == 1, 0], training[label train == 1, 1],
   'rx')
59
       plt.plot(training[label_train == 2, 0],training[label_train == 2, 1],
   'go')
       if nclass == 3:
60
           plt.plot(training[label_train == 3, 0], training[label_train == 3, 1],
61
   'b*')
62
       # include legend for training data
63
       l = plt.legend(('Class 1', 'Class 2', 'Class 3'), loc=2)
64
65
       plt.gca().add artist(l)
66
67
       # plot the class mean vector.
       m1, = plt.plot(sample_mean[0,0], sample_mean[0,1], 'rd', markersize=12,
68
  markerfacecolor='r', markeredgecolor='w')
       m2, = plt.plot(sample_mean[1,0], sample_mean[1,1], 'gd', markersize=12,
69
   markerfacecolor='g', markeredgecolor='w')
       if nmean == 3:
70
71
           m3, = plt.plot(sample_mean[2,0], sample_mean[2,1], 'bd',
  markersize=12, markerfacecolor='b', markeredgecolor='w')
72
       # include legend for class mean vector
73
74
       if nmean == 3:
           l1 = plt.legend([m1,m2,m3],['Class 1 Mean', 'Class 2 Mean', 'Class 3
75
  Mean'], loc=4)
76
       else:
           l1 = plt.legend([m1,m2], ['Class ' + str(class1) + ' Mean', 'Class ' +
77
   str(class2) + ' Mean'], loc=4)
78
       plt.gca().add_artist(l1)
79
80
81
       plt.show()
```

localhost:4649/?mode=python