

```

1 #####
2 # TITLE: ex5 machine learning BIU
3 # WRITER: Raphael Haehnel
4 # DATE: 03/01/2023
5 #####
6
7 import scipy.io
8 from sklearn.model_selection import train_test_split
9 import numpy as np
10 from matplotlib import pyplot as plt
11 from sklearn.metrics import accuracy_score
12 from tqdm import tqdm
13
14
15 def helper_retrieve(data):
16     a1 = np.split(data, 15)
17     a2 = np.array(a1)
18     a31 = a2[:, 0:8, :]
19     a32 = a2[:, 8:11, :]
20
21     train = np.concatenate(a31)
22     test = np.concatenate(a32)
23
24     return train, test
25
26
27 def retrieve_data():
28     mat = scipy.io.loadmat('facesData.mat')
29     faces = mat.get('faces')
30     labels = mat.get('labeled')
31
32     X_train, X_test = helper_retrieve(faces)
33     y_train, y_test = helper_retrieve(labels)
34
35     return X_train, X_test, y_train, y_test
36
37
38 def extract_eigenvectors(X: np.ndarray, mu: np.ndarray):
39     A = np.matmul((X - mu).T, X - mu)
40     w, v = np.linalg.eig(A)
41     return w.real.astype('float64'), v.real.astype('float64')
42
43
44 def sort_eigenvectors(w, v):
45     indexes = np.argsort(np.abs(w))
46
47     v = v[:, indexes]
48     w = w[indexes]
49
50     v = np.flip(v, axis=1)
51     w = np.flip(w, axis=0)
52
53     return w, v
54
55
56 def show_PCA(mu, v):
57     fig = plt.figure(figsize=(10, 2))
58     num = 8
59     # Display the mean
60     fig.add_subplot(1, num, 1)
61     plt.imshow(mu.reshape((32, 32)).T, cmap='gray')
62     plt.title(f"mu")
63     plt.axis('off')
64
65     # Displaying the first 5 eigenvectors
66     for i in range(num-1):
67         fig.add_subplot(1, num, i+2)
68         plt.imshow(v[:, i].reshape((32, 32)).T, cmap='gray')
69         plt.title(f"v{i}")
70         plt.axis('off')
71
72     fig.canvas.manager.set_window_title('show_PCA')
73     plt.show()
74
75
76 def compute_projection(v, x_T, mu, K):
77
78     v_k = v[:, :K]

```

```

79 projection = np.matmul(x_T - mu, v_k)
80 reconstruction = mu + np.matmul(projection, v_k.T)
81
82 return projection, reconstruction
83
84
85 def reconstruct_image(v, mu, x_T):
86
87     # Build the displaying window
88     fig = plt.figure(figsize=(7, 7))
89     rows = 5
90     cols = 5
91
92     for i in range(24):
93
94         # We choose the i first eigenvectors
95         projection, reconstruction = compute_projection(v, x_T, mu, i)
96
97         fig.add_subplot(rows, cols, i+1)
98
99         # Display the projected eigenvector
100        plt.imshow(reconstruction.reshape((32, 32)).T, cmap='gray')
101        plt.axis('off')
102        plt.title(f"K={i}")
103
104    projection, reconstruction = compute_projection(v, x_T, mu, v.shape[1])
105
106    fig.add_subplot(rows, cols, 25)
107
108    # Display the projected eigenvector
109    plt.imshow(reconstruction.reshape((32, 32)).T, cmap='gray')
110    plt.axis('off')
111    plt.title(f"K=1024")
112
113    fig.canvas.manager.set_window_title('reconstruct_image')
114    plt.show()
115
116
117 def mesure_projections(projections_train, projections_test, y_train):
118
119     label_list = []
120
121     for i in range(len(projections_test)):
122
123         # Measure the euclidian distance between one train projection and
124         # all the test projections
125         dist = np.linalg.norm(projections_train - projections_test[i], axis=1)
126
127         # Extract the index from the train set for which the distance is minimal
128         index_train = np.argmin(dist)
129
130         # Find the corresponding label of this train sample
131         label = y_train[index_train]
132
133         label_list.append(label)
134
135     return np.array(label_list)
136
137
138 def show_accuracy_graph(X_train, mu, X_test, y_train, y_test):
139     accuracy_list = []
140     x = list(range(1, 50, 1))
141
142     for i in tqdm(x):
143         K = i
144
145         # Get the projections of all the train set
146         projections_train, _ = compute_projection(v, X_train, mu, K)
147
148         # Get the projections of all the test set
149         projections_test, _ = compute_projection(v, X_test, mu, K)
150
151         # Measure the distance from each projection of the train set
152         # to all the projections of the train set
153         y_pred = mesure_projections(
154             projections_train, projections_test, y_train)
155
156         # Compute the accuracy classification score.
157         accuracy = accuracy_score(y_test, y_pred)

```

```

158
159     accuracy_list.append(accuracy)
160
161 plt.plot(x, accuracy_list)
162 plt.xlabel("K")
163 plt.ylabel("Accuracy")
164 plt.title("Accuracy of the PCA projections")
165 plt.show()
166
167
168 if __name__ == "__main__":
169
170     # Creating training and testing sets
171     X_train, X_test, y_train, y_test = retrieve_data()
172
173     # Find the mean image
174     mu = X_train.mean(0)
175
176     # Find the eigenvectors
177     w, v = extract_eigenvectors(X_train, mu)
178
179     # Sort the eigenvectors
180     w, v = sort_eigenvectors(w, v)
181
182     # Transpose the first image of the dataset
183     x_T = np.reshape(X_train[0], (1, 1024))
184
185     # Show the principal components
186     show_PCA(mu, v)
187
188     # Show the reconstruction of the images
189     reconstruct_image(v, mu, x_T)
190
191     # Show the accuracy graph
192     show_accuracy_graph(X_train, mu, X_test, y_train, y_test)
193
194     K = 16
195
196     # Get the projections of all the train set
197     projections_train, _ = compute_projection(v, X_train, mu, K)
198
199     # Get the projections of all the test set
200     projections_test, _ = compute_projection(v, X_test, mu, K)
201
202     # Measure the distance from each projection of the train set
203     # to all the projections of the train set
204     y_pred = mesure_projections(
205         projections_train, projections_test, y_train)
206
207     # Compute the accuracy classification score.
208     accuracy = accuracy_score(y_test, y_pred)
209
210     print(f"Accuracy: {accuracy*100}%")
211

```