CS498 AML HW5

Yidi Yang (yyang160)
Huiyun Wu (hwu63)

1. Table

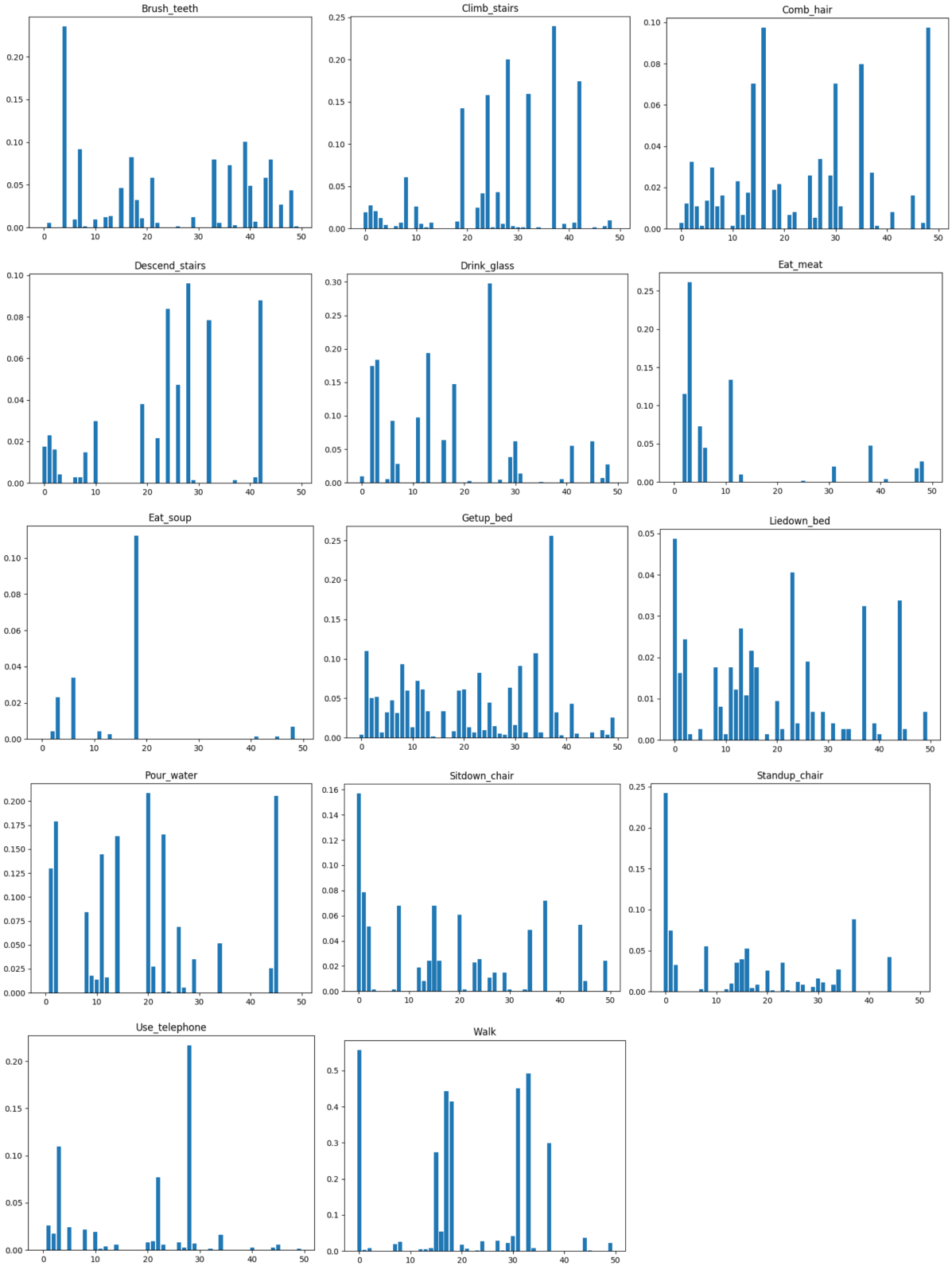
Size of window	Overlap	K-Value	Classifier	Accuracy
32	0% No overlap	50	Random Forest (tree=200,depth =6)	72%
32	0% No overlap	60	Random Forest (tree=200,depth =6)	65.4%
40	0% No overlap	50	Random Forest (tree=200,depth =6)	65%
40	0% No overlap	60	Random Forest (tree=200,depth =6)	65.2%
32	0% No overlap	70	Random Forest (tree=200,depth =6)	63%
40	0% No overlap	70	Random Forest (tree=200,depth =6)	68%

Note1: We use the **standard K-Means** instead of the hierarchical one for all the cases.

Note2: We randomly **split out 100 signals** among all 839 files as the testing data and the remainings are as the training data before applying the vector quantization.

Other findings: We found that relatively more trees and deeper depth give higher accuracies.

2. Histograms. **(K = 50, size of window = 32)**



3. Confusion Matrix.

	Brush Teeth	Climb Stairs	Comb Hair	Descend stairs	Drink Glass	Eat Meat	Eat Soup	Getup Bed	Liedown Bed	Pour Water	Sitdown Chair	Standup Chair	Use Telephone	Walk
Brush Teeth	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Climb Stairs	0	9	0	0	0	0	0	0	0	0	0	0	0	0
Comb Hair	0	0	10	3	0	0	0	0	0	0	0	0	0	1
Descend stairs	0	1	1	8	0	0	0	0	0	0	0	0	0	1
Drink Glass	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Eat Meat	0	1	0	2	0	13	0	0	0	0	0	0	0	0
Eat Soup	0	0	0	0	0	0	2	0	3	0	0	0	0	0
Getup Bed	0	2	0	0	0	0	0	0	3	0	0	0	0	0
Liedown Bed	0	0	0	0	0	0	0	0	9	0	0	0	0	0
Pour Water	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Sitdown Chair	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Standup Chair	0	0	0	2	0	0	0	0	0	0	0	10	0	0
Use Telephone	0	0	0	1	0	0	0	0	1	0	0	2	0	0
Walk	0	0	0	0	0	0	0	0	0	0	0	0	0	13

See code snippets on the next page.

4. Code Snippets.

a. Segmentation of the vector.

```
51  def makeSegments(data, seg_length):
52      segments = []
53      segment_file_no = []
54      for idx, file in enumerate(data):
55          segNumber = len(file) // seg_length
56          temp = file.copy()
57          for i in range(segNumber):
58              segment_file_no.append(idx)
59              segments.append(temp[:seg_length])
60              temp = temp[seg_length:]
61      return segments, segment_file_no
```

```
78  # making segments
79  train_segments, train_seg_file_no = makeSegments(train_data, seg_length)
80  test_segments, test_seg_file_no = makeSegments(test_data, seg_length)
```

b. K-means.

```
84  kmeans = KMeans(n_clusters=K).fit(train_segments)
```

```
103  tkmeans = kmeans.predict(test_segments)
```

c. Generating the histograms.

```
90  # making histogram for each activity
91  for label in range(14):
92      indices = [i for i, x in enumerate(train_labels) if label == int(x)]
93      sum = np.array([0 for col in range(K)])
94
95      for idx in indices:
96          sum += np.array(feature_vectors[idx])
97      avg = sum/len(train_labels)
98      plt.figure(0)
99      plt.bar(range(K), avg)
100     plt.title(act[label])
101     plt.show()
```

d. Classification.

```
108 # classifier
109 clf = RandomForestClassifier(n_estimators=200, max_depth=6)
110 clf.fit(feature_vectors, train_labels)
111 testing_results = clf.predict(t_feature_vectors)
112
113 accuracy = np.sum(testing_results == test_labels) / len(test)
114 error_rate = 1-accuracy
115 print(error_rate, accuracy)
116
117 c_mat = confusion_matrix(test_labels, testing_results)
118 print(c_mat)
```

5. Other relevant code snippets.

The main function of the whole process.

```
68  if __name__ == "__main__":
69      for m in range(5):                                  # run each case 5 times
70          K = 50
71          seg_length = 3 * 32
72          train, test = parse()
73          train_labels = [x[0] for x in train]
74          train_data = np.array([x[1:] for x in train])
75          test_labels = [x[0] for x in test]
76          test_data = np.array([x[1:] for x in test])
77
78          # making segments
79          train_segments, train_seg_file_no = makeSegments(train_data, seg_length)
80          test_segments, test_seg_file_no = makeSegments(test_data, seg_length)
81
82          feature_vectors = [[0 for col in range(K)] for row in range(len(train))]
83          kmeans = KMeans(n_clusters=K).fit(train_segments)
84          feature_vec(feature_vectors, train_segments, train_seg_file_no, kmeans.labels_)
85          # making histogram for each activity
86          for label in range(14):
87              indices = [i for i, x in enumerate(train_labels) if label == int(x)]
88              sum = np.array([0 for col in range(K)])
89
90              for idx in indices:
91                  sum += np.array(feature_vectors[idx])
92              avg = sum/len(train_labels)
93              plt.figure(0)
94              plt.bar(range(K), avg)
95              plt.title(act[label])
96              plt.show()
97          tkmeans = kmeans.predict(test_segments)
98          t_feature_vectors = [[0 for col in range(K)] for row in range(len(test))]
99          feature_vec(t_feature_vectors, test_segments, test_seg_file_no, tkmeans)
100         # classifier
101         clf = RandomForestClassifier(n_estimators=200, max_depth=6)
102         clf.fit(feature_vectors, train_labels)
103         testing_results = clf.predict(t_feature_vectors)
104
105         accuracy = np.sum(testing_results == test_labels) / len(test)
106         error_rate = 1-accuracy
107         print(error_rate, accuracy)
108         c_mat = confusion_matrix(test_labels, testing_results)
109         print(c_mat)
```