CS 498 HW5 Report

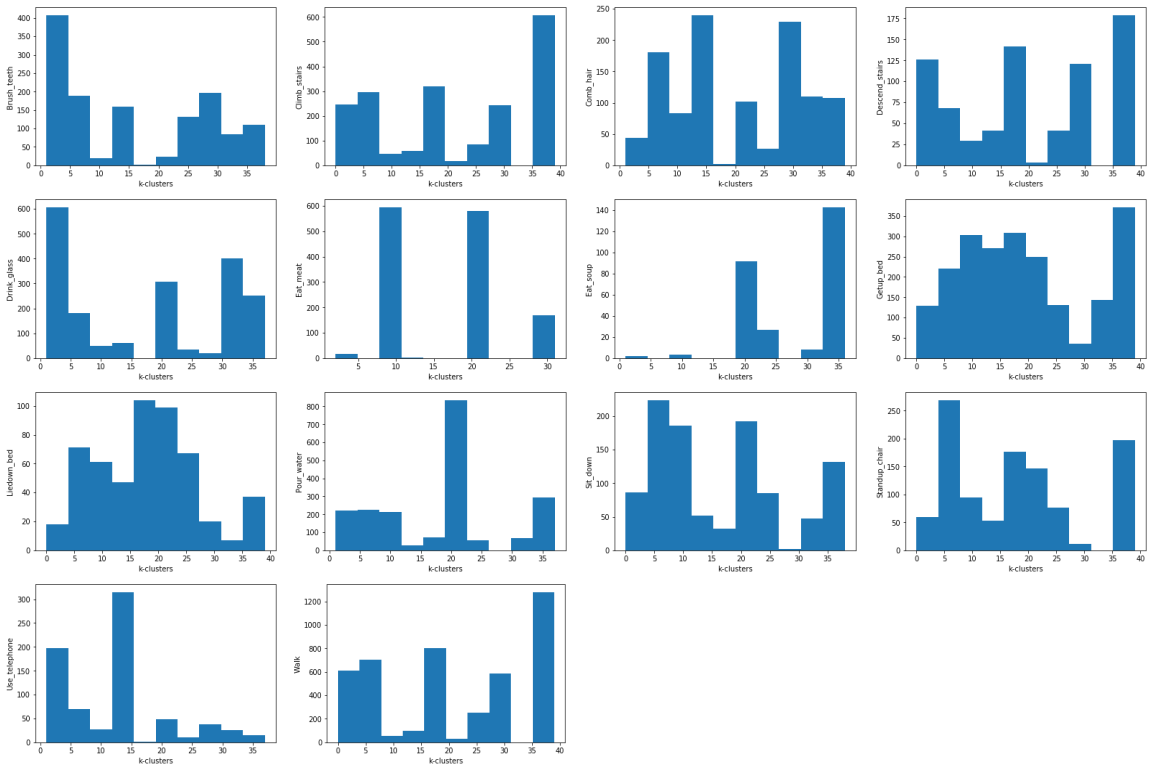
Our group use standard K-means to implement the model. We use two K-values, 40 and 12, and two window lengths, 96 and 72. (all with overlap 48)

The first-row values are window length and the first column values are k-means we used.

Length / K-Clusters	72	96
12	0.6705202312138728	0.7167630057803468
40	0.7687861271676301	0.7745664739884393

The classifier of highest accuracy is trained by data generated by 40 k-clusters, 96 window length, and 48(50% to 96) overlaps.

Histogram plots in alphabetic order:



Confusion matrix. Rows represent the true labels in alphabetic order and columns represent predicted labels in alphabetic order by random forest with 100 estimators and 64 maximum depth.

Label	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	2	0	0	0	0	0	0	0	0	0	1	0	0	0
2	0	19	0	2	0	0	0	0	0	0	0	0	0	0
3	0	0	7	0	0	0	0	0	0	0	0	0	0	0
4	0	4	0	5	0	0	0	0	0	0	0	0	0	0
5	0	0	1	0	19	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0	0
8	0	3	0	0	0	0	0	11	0	0	1	6	0	0
9	0	0	0	0	0	0	0	3	0	0	3	0	0	0
10	0	0	0	0	0	0	0	0	0	20	0	0	0	0
11	0	0	0	0	0	0	0	2	0	1	14	3	0	0
12	0	1	0	0	0	0	0	2	0	3	1	14	0	0
13	0	0	0	0	2	0	0	0	0	0	0	0	1	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	20

1. Segmentation of the vector

```
#Length is the total length of one selection, laplen is the overlap length
def trunk_data(array, length, laplen):
    loc = 0 # starting point of the current segmentation
    list = []
    while((len(array) - loc) >= length):
        small_list = []
        for i in range(length):
             small_list.append(array[loc+i])
        list.append(small_list)
        loc = loc + length - laplen # update starting point
    return list
```

```
#concatenate all data segmentations regardless of files and activities in one list
def superlist(biglist):
    list = []
    for a in range(len(biglist)):
        for i in range(len(biglist[a])):
            list.append(biglist[a][i])
    return list
```

2. K-means

```
super_train = superlist(train) # a big list of all training data for k-means
from sklearn.cluster import KMeans
train_kmeans = KMeans(n_clusters=40).fit(super_train) # fit all training data using 40-cluster
hist = [[] for i in range(14)]
for a in range(14):
    for f in range(len(train_trunks[a])):
        templist = [[0] *40] # cluster distribution for each file
        for i in range(len(train_trunks[a][f])):
            train_trunks[a][f][i] = np.asarray(train_trunks[a][f][i])
            index = int(train_kmeans.predict(train_trunks[a][f][i].reshape(1,-1)))
            templist[0][index] +=1 # add repetition to corresponding cluster
            hist[a].append(index) # update histogram
        templist = np.asarray(templist)
        train_fea.append(templist)
        train_label.append(a)
for a in range(14):
    for f in range(len(test_trunks[a])):
        templist = [[0] *40] # cluster distribution for each file
        for i in range(len(test_trunks[a][f])):
            test_trunks[a][f][i] = np.asarray(test_trunks[a][f][i])
            index = int(train_kmeans.predict(test_trunks[a][f][i].reshape(1,-1)))
            templist[0][index] +=1 # add repetition to corresponding cluster
        templist = np.asarray(templist)
        test_fea.append(templist)
        test_label.append(a)
```

3. Histogram

```
import matplotlib.pyplot as plt
filename = ["Brush_teeth","Climb_stairs","Comb_hair","Descend_stairs","Drink_glass","Eat_meat","Eat_soup","Getup_bed",
           "Liedown_bed","Pour_water","Sit_down","Standup_chair","Use_telephone","Walk"]
for i in range(14):
    plt.hist(hist[i])
    plt.xlabel("k-clusters")
    plt.ylabel(filename[i])
    plt.figure()
#     plt.show()
    plt.savefig(filename[i])
```

4. Classification

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100, max_depth=64, random_state=0)
clf.fit(train_fea, train_label)
clf.score(test_fea, test_label)
```

This is our code of reading file and flattening the data in 1-D sequence.

```
from sklearn.metrics import classification_report, confusion_matrix
test_pred = clf.predict(test_fea)
print(confusion_matrix(test_label,test_pred))
big_list = []
big_test = []
read_files = glob.glob("./HMP_Dataset/Brush_teeth/*.txt")
n_train = int(len(read_files) * 0.8)
train_data = read_files[0:n_train]
test_data = read_files[n_train:]
# with open("result.txt", "w") as outfile:
list_Brush = []
test_Brush = []
for f in train_data:
    list_Brush.append(func(f))
big_list.append(list_Brush)
for f in test_data:
    test_Brush.append(func(f))
big_test.append(test_Brush)
```

This is our code of allocating data into corresponding activities for the sake of implementation convenience. Here, we set window length to be 96 and overlap 48.

```
train_trunks = [[] for i in range(14)] # trunks of training data in all activities
train = []
for a in range(big_list.shape[0]):
    for f in range(big_list[a].shape[0]):
        train_trunks[a].append(trunk_data(big_list[a][f], 96, 48)) # 96 data point segmentation with no overlaps
        train.append(trunk_data(big_list[a][f], 96, 48))
test_trunks = [[] for i in range(14)] # trunks of testing data in all activities
test = []
for a in range(big_test.shape[0]):
    for f in range(big_test[a].shape[0]):
        test_trunks[a].append(trunk_data(big_test[a][f], 96, 48))
        test.append(trunk_data(big_test[a][f], 96, 48))
```