**Comp-6915 machine learning**

**assignment1**

1. **Short description**
2. **collect data: using most part of the data in TF\_sequence.txt and TF\_output as the training data. The rest part as the test data(X\_unseen.txt)**
3. **Enter the data in the program and saved as type of list.**
4. **Use the comparetf() function to calculate the distance between test factor and train factor. The first step to calculate is that we define 15 residues of points in the sequence. The second step is that comparing these points between train sequence factor and test sequence factor. Then if the character in one point is diffenent. The distance will plus one. We will figure out all of distance between all test factors and test factors. And arrange the distance in ascending order.**
5. **User can input a number of k which means that we will fine k nearest neighbors.**
6. **The last step is use the classify() function to find the k nearest neighbors and find out the output of these neighbors. Figure out the average the output numbers of k nearest neighbors like the slides example. The result is the predict of the test data.**

**1.pseudo-code of KNN implementation:****def** inputsequence(filename): #open input file and store the file content as a list。X\_train.txt and X.unseen.txt will use this function  
 **with** open(filename) **as** f:  
 **for e**achline **in** f.readlines():  
 tf\_sq\_sp = tf\_sq.split(**'\t'**) #use tab to split the list  
 tf.append(tf\_sq\_sp) # store in a list named tf  
 **return** tf **def** outputsequence(filename): #open output file and store the file content as a list。Y\_train.txt will use this function to open.  
 **with** open(filename) **as** f:  
 **for** line **in** f.readlines():  
 new\_tf\_sq\_sp = [tf\_sq\_sp.replace(**'"'**, **''**) **for** tf\_sq\_sp **in** tf\_sq\_sp] # delete the “ in the txt  
 tf.append(new\_tf\_sq\_sp) #delete “ in the Y\_train.txt  
 **return** tf **def** classify(k,outputset,nameset,tf\_test\_all): #according to the X\_unseen set to find the distance between each factor in X\_unseen set and each factor in train set. And then ccording to K, we can find the k nearest neighbors. Figure out the average of neighbors' output.  
 **while** w <= (len(nameset)-1): #find form the first unseen factor**while** j <= k - 1: #find k nieghbors  
**while** i <= (len(outputset[0])-1): #find the nearst neighbor name in the Y\_train.txt  
**if** the neighbor name == name in Y\_output:

record the name’s location   
 averageset.append(tf\_test\_all[w][0]) #record the name of a unseen factor  
 **while** z <= (len(outputset) - 1): *#*calculate the distance

**for** x **in** target:  
 sum = sum + float(outputset[z][x])  
 average = sum/len(target)  
 averageset.append(average)#record the average distance  
 averagesetall.append(averageset)#record all the distance in a list  
 **return** averagesetall  
**def** comparetf(tf\_test\_all,tf\_train\_all): # figure out the distance of testset and trainset  
 **while** i<=(len(tf\_test\_all)-1): #traversal all the X\_train  
**while** j<=(len(tf\_train\_all)-1): #traversal all the Y\_unseen  
**for** x **in** [2,4,5,24,30,43,45,46,47]: #set monitoring point  
**if** point in X\_train not the same as point X\_unseen: distance add one  
 put the name and distance together and sotre in a list  
 namedistanceset = []  
 distanceset = sorted(distanceset, key=**lambda** x: x[0])#sort by the distance  
 distancesetall.append(distanceset) #put one test factor distance in the set  
 **return** distancesetall **def** text\_save(filename, data): *#* **save the output as a file**

file = open(filename,**'a'**)  
 **for** i **in** range(len(data)):  
 delete some special simple such as ; , . “…file.write(s)  
 file.close() **if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 *#open files* filename = **"X\_train.txt"** filename1 = **"X\_unseen.txt"** filename2 = **"Y\_train.txt"** filename3 = **"TF\_predict.txt"** #transfer the content of file to list and store in variable  
trainset = inputsequence(filename)  
 unseenset = inputsequence(filename1)  
 outputset = outputsequence(filename2)  
  
 c = comparetf(unseenset,trainset) #find the distance of the neighbors  
k = input(**"please enter k for KNN algorithm: "**)  
 e = classify(int(k),outputset,c,unseenset)colume\_e = [[r[col] **for** r **in** e] **for** col **in** range(len(e[0]))]  
 text\_save(filename3,colume\_e) #save the predict result as a file TF\_predict.txt

**2.the definition of the distance function：**

In this part, we will call X\_unseen and X\_train list and traversal each X\_unseen factors. Pick up one factor in X\_unseen, then compared each monitoring point of TF sequence between X\_unseen factor and X\_train. If the point is not same, the distance will add one. Use one factor in X\_unseen compare with all factor in X\_train and record one set of distances. We will figure out all of factors in X\_unseen like this. According to the K, we can find k nearest neighbors and figure the average of their output.

**3.explaining how we tested our code to make sure our implementation was correct**

To test our code we need to remove 10 of the vectors from TF\_sequence and TF\_output. The vectors in these two file will have the same name. Put 10 factors from TF\_sequence in X\_unseen.txt. Then use our program to predict the output of these 10 TFs. The result will be saved in TF\_predict. We can compare the output numbers between TF\_predict.txt and 10 vectors which we have deletet from TF\_output. If the difference between these numbers is small, that mean our implementation is correct.