Report for Assignment 2

ECE 657A
Group 11
Shala Chen, 20698485
Jinghang Wang, 20676432
Zhao Zhang, 20689699

March 19, 2017

This lab required us to use classification and clustering methods, compare their attributes and results, and analyze different outputs.

Tools we use: MATLAB (Version: 2014b or 2016b)

We used the LIBSVM library for part I.

1 Parameter Selection and Classification (for dataset D)

1.1 Z-score Normalization

The dataset was Z-score normalized using Matlab function zscore.

1.2 KNN Classifier

We used KNN classifier to classify the dataset and iterated k from 1 to 31 increment by 2, then evaluated the results individually in terms of accuracy. The relationship between k and accuracy is displayed in the following plot.

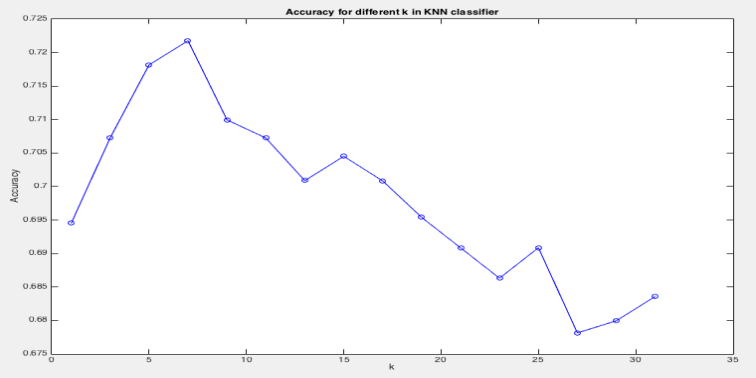

Figure 1: Accuracy of KNN classifier for different K parameter

As it showed in the figure, the accuracy was highest when k = 7.

We think the reason is if the k is too small then the classifier will be easily influenced by outliers, but if k is too large then the border is not accurate enough when two classes blend together. So the best k for this dataset in terms of classification accuracy is 7.

1.3 RBF Kernel SVM

In this section, we first used 5-fold cross validation as in the last section to select training and testing data. Then calculated the accuracy of different RBF kernel SVM based on different parameters **c** and **sigma**, found the best pairs of **c,sigma** and finally plotted the related ROC curve.

To find the best pairs that maximized the accuracy of our RBF kernel SVM, we had to go through all 8 **c** and 8 **sigma**, which meant we had to look over 8*8 = 64 classifiers' accuracy. Besides, we used 5-fold cross validation, thus the process of finding best accuracy classifier went 8*8*5 = 320 times' iteration.

To store the accuracy of different parameter pairs, we constructed a 8*8 empty matrix, each element of the matrix was corresponding to an unique pair of **c, sigma**. In the accuracy calculation process, for each pair we applied 5-fold cross validation, then averaged 5 accuracies and stored the mean value into the matrix.

After calculating all 64 accuracy values, we found the pair with the highest accuracy and reported related **c, sigma**. Since we used 5-fold cross validation on the training set to train the classifiers, the pairs reported might be different every time. However, during our experiment, the best pairs were mostly among (**c = 5, sigma = 0.01**) and (**c = 10, sigma = 0.01**).

We used Matlab functions **svmtrain, predictsvm** which could be accessed from **libSVM**. The **svmtrain** function were used to train each RBF kernel SVM and **predictsvm** were used to calculate the accuracy as well as the classifier working results(predict the class label).

Below is the ROC curve of pair **c = 10, sigma = 0.01**.

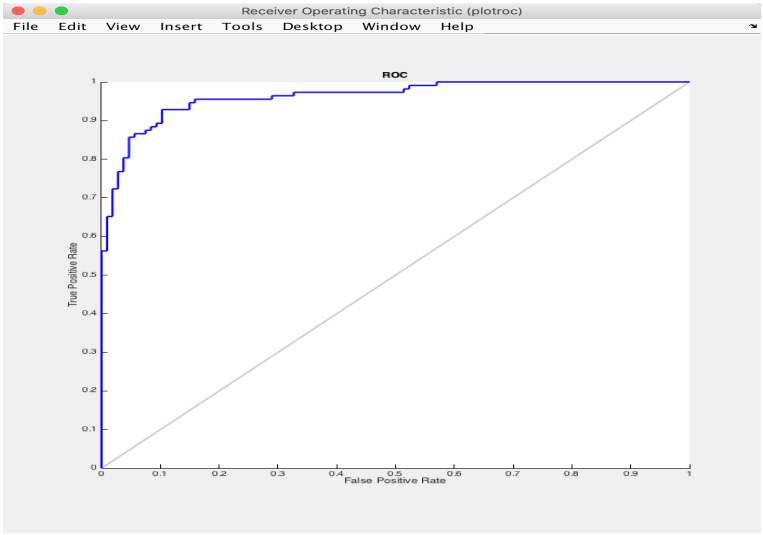

Figure 2: ROC curve of c = 10, sigma = 0.01

1.4 k-NN, SVM, Decision Trees, Random Forests and Neural Networks

In this part, we used the first half as our training samples and the second half as our testing samples. We used five methods to do classification and predict their labels. While doing classification we measured the time for each classifier, including training time and testing time. At last we obtained

results in terms of accuracy, precision, f-measure values and related stand deviation values. The stand deviations were based on repeating 20 times for each classification. Next, I will use bar charts to show the results we got.

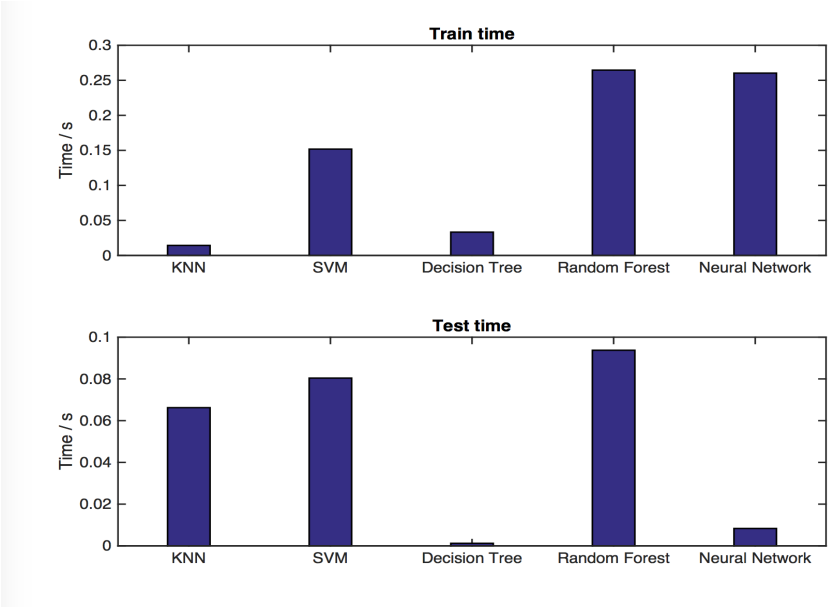

Figure 3: Train time and test time@ KNN, SVM, Decision Tree, Random Forest, Neural Network

Figure 3 showed the training time for these five classifiers. From this figure we can see Decision-Tree and KNN had much less train time. Decision-Tree and Neural-Network had much less test time. So if the training data is too large and we only consider the train time, maybe KNN and Decision-Tree are the best choices to do classification. Also we can see that if we already had a good a Decision-Tree or Neural-Network, we will take less time to do a predicting on new data compared with other methods.

In conclusion, Decision tree had the best performance on training times.

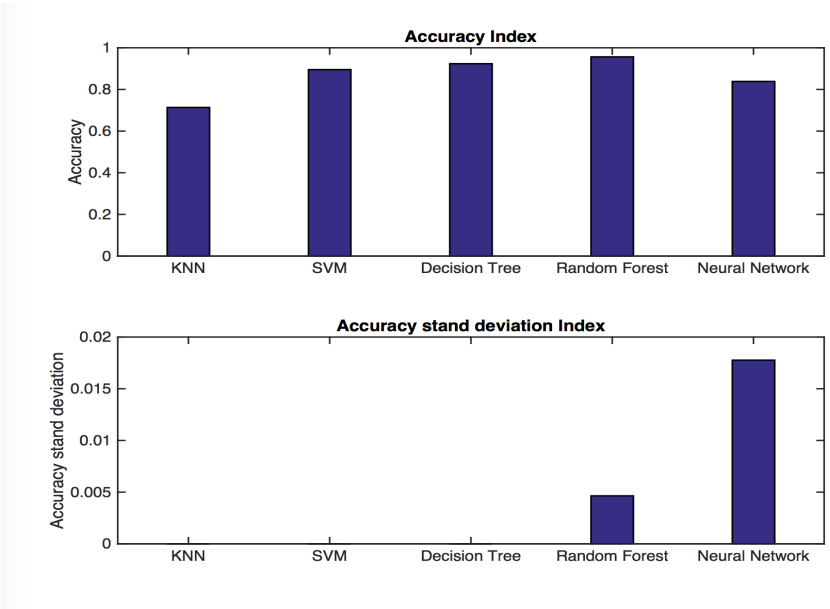

Figure 4: Accuracy index@ KNN, SVM, Decision Tree, Random Forest, Neural Network

Figure 4 shows the accuracies and their stand deviations. From figure 4 we can find out that Random Forest had the highest accuracy and KNN method performed badly. If we want to have a higher accuracy, we can use SVM, Decision Tree and Random Forest. Considering the accuracy stand deviation, KNN, SVM and Decision Tree had the lowest stand deviation: 0. So the first three methods are more stable. Neural Network is the most unstable one among all classifiers. So when we are training classifiers, we must make sure neural network is on its best condition.

Considering both accuracies and standard deviations, SVM and Decision Tree had a better performance for this dataset.

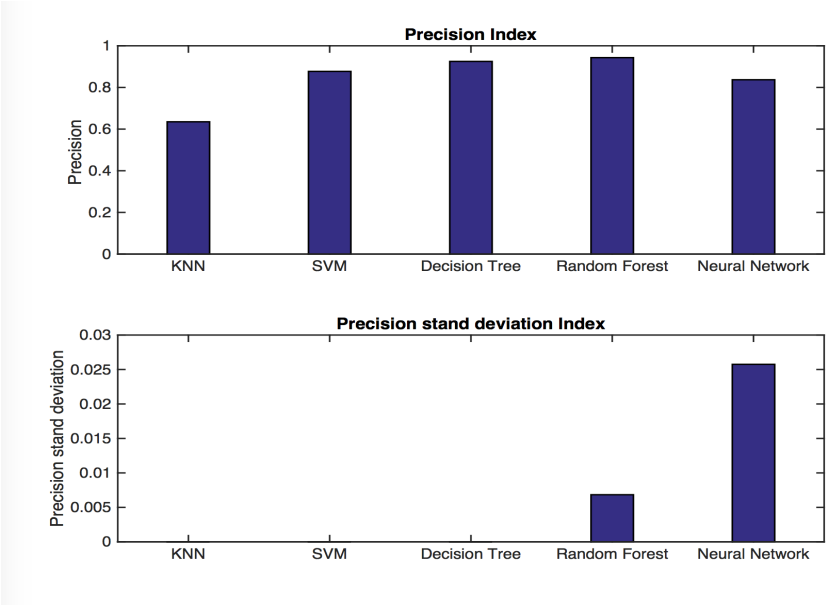

Figure 5: Precision index@ KNN, SVM, Decision Tree, Random Forest, Neural Network

Figure 5 shows classifier precision indexes and their stand deviations. Precisions were calculated as a fraction of pairs correctly put in the same cluster. High precision means that an algorithm returned substantially more relevant results than irrelevant ones. From the figure we could tell that SVM, Decision Tree and Random Forest performed better. More points had been put into the correct cluster. However, random forest had a relatively higher standard deviation. So our conclusion is that SVM and Decision Tree were best classifiers for this dataset.

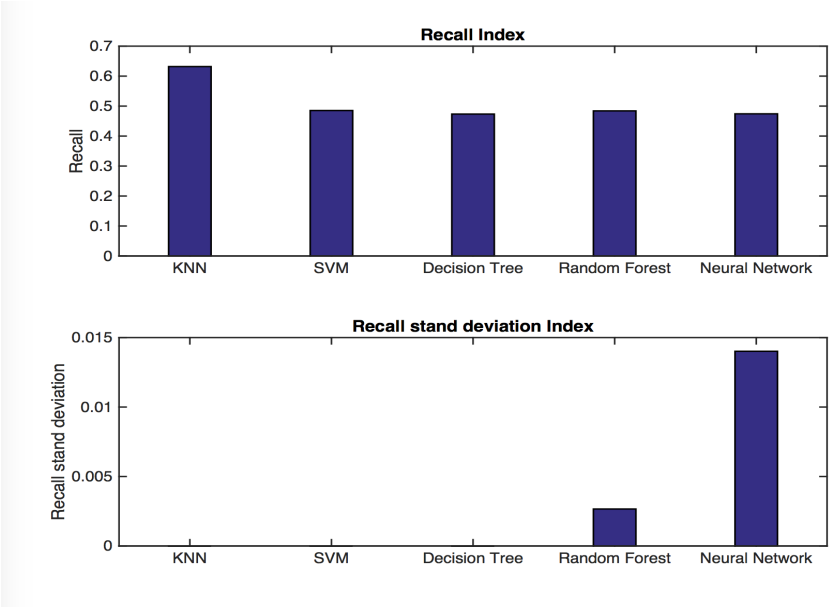

Figure 6: Recall index@ KNN, SVM, Decision Tree, Random Forest, Neural Network

Figure 6 showed classifiers' recall indexes and their standard deviations. Recall means that an algorithm returned most of the relevant result. According to the bar chart, KNN had the highest recall rate with zero standard deviations. The other classifiers had very close recall rates, and only Neural network had a very high standard deviation.

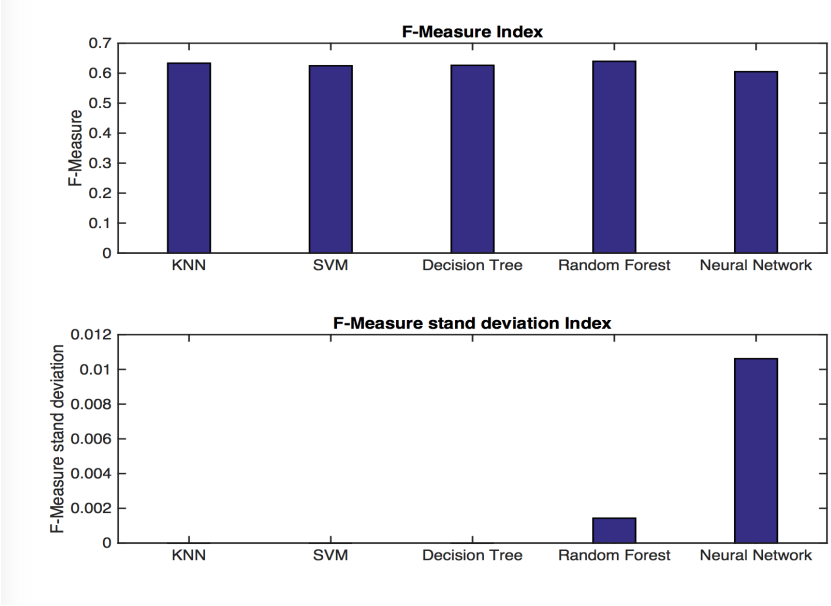

Figure 7: F-measure index@ KNN, SVM, Decision Tree, Random Forest, Neural Network

The last index we used is F-Measure index, which is the harmonic mean of precision and recall. These five methods had close F-Measure values. If we only use F-Measure indexes as our evaluation values, then they had very similar results in this situation. We cannot tell which one is better than others. So we must combine other factors to choose the best cluster.

1.5 Discussion

With all the results generated from above, we can make the following conclusions.

KNN was not the our best choice in general, it was the highest in recall index, however, it was way from the best in terms of accuracy index and precision index.

SVM was among average in training time, accuracy, precision index, recall index and f-measure index. It wasn't too good or too bad in everything, but a very steady choice since the standard deviation was all 0 for all time.

Decision Tree was comparatively a batter choice by all means. It took shorter training time, very high in terms of accuracy, precision index and f-measure index. In the mean while, the classifier was very steady with 0 standard deviations. We think Decision Tree was a very good method to classify this dataset.

Random Forest was also a good method among all classifiers. It has the highest accuracy index, precision index and F measure index. The only thing that kept it from the best method was that it sometimes had a higher standard deviation than the methods we mentioned above. We assume it's due to its randomness. It also took longer time to train the classifier.

Neural Network didn't have a great performance for this dataset. The accuracy index, precision index, recall index and f measure were all about or below average. What's more, it had a very high standard deviation, which means it wasn't very steady while classifying.

Overall, we think Random forest and Decision Tree were the best methods for this dataset.

2 Clustering Analysis (for dataset F)

2.1 Hierarchical clustering using agglomerative algorithms

2.1.1 Three linkage types comparing

For this question, we used agglomerative algorithms and three linkage types: single, complete and ward. We calculated three indexes: separation index, rand index and F-Measure index using following equations:

Separated-Index:

$$SI = \frac{\sum_{i=1}^{k} \sum_{x_j \in C_i} d^2(x_j, m_i)}{n * min_{C_r, C_s \in C} d^2(C_r, C_s)}$$

Rand-Index:

$$RI = \frac{a+d}{\frac{n(n-1)}{2}}$$

F-measure:

$$F(i,j) = \frac{2precision(i,j)recall(i,j)}{precision(i,j) + recall(i,j)}$$

$$F = \sum_{j=1}^{C} \frac{m_j}{n} max_{i \in k} F(i,j)$$

We got the following bar charts:

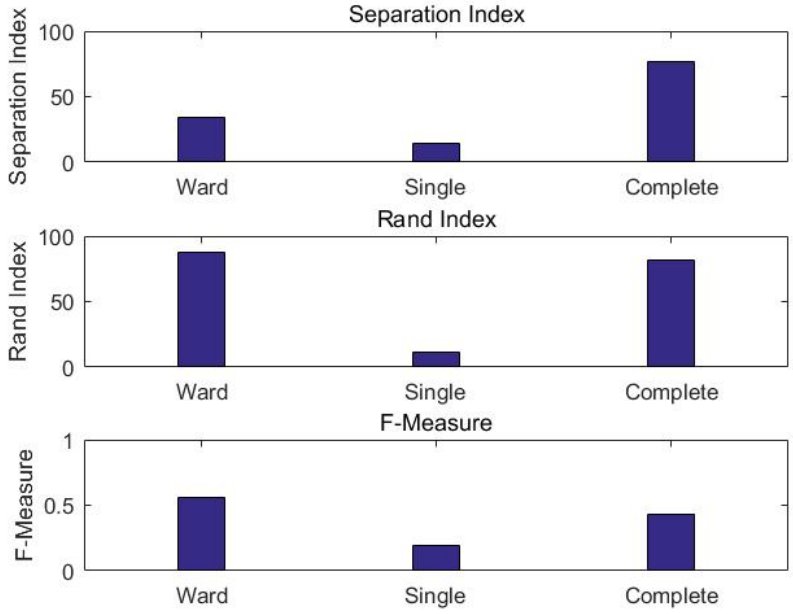

Figure 8: Separation, Rand, F-Measure values for Ward, Single, Complete methods

Firstly, in term of F-Measure index, we can see that Ward is the best method to do clustering. And Single performed badly. Rand index stands for how clustering results fit our real situations. A higher rand index gives us a better cluster results. Ward was also better than the other two methods.

2.1.2 Optimal number for Ward linkage type

We varied K values and calculated separation indexes, then generated then following plot:

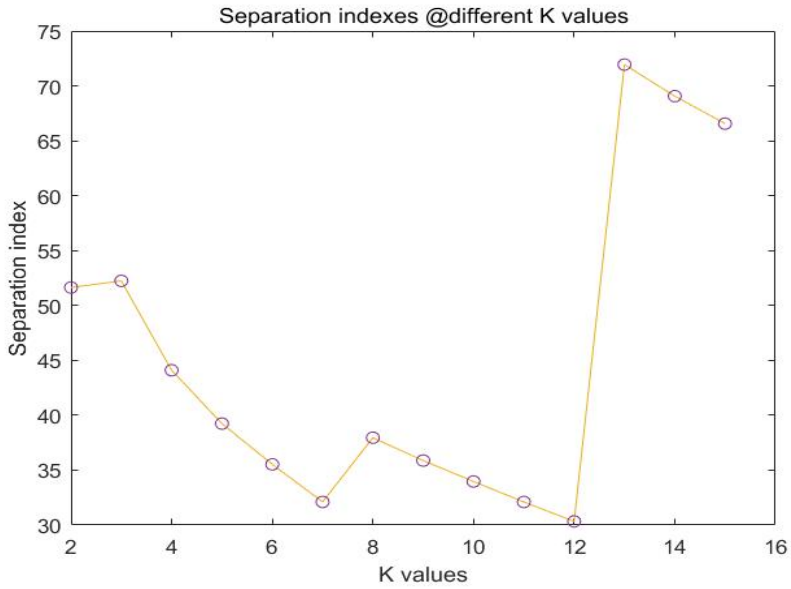

Figure 9: Separation, Rand, F-Measure values for Ward, Single, Complete methods

Figure 9 shows the results we got. Cause a lower separation index means classes are more far

with each other, so in at the point x = 12 we can get the lowest separation index. The optimal number of clusters is 12.

2.2 Cluster the data using k-means algorithm.

We iterated k from 2 to 15 and evaluated the results individually in terms of Separation-Index, Rand-Index, and F-measure

We gathered all the evaluation measures and generated following plots.

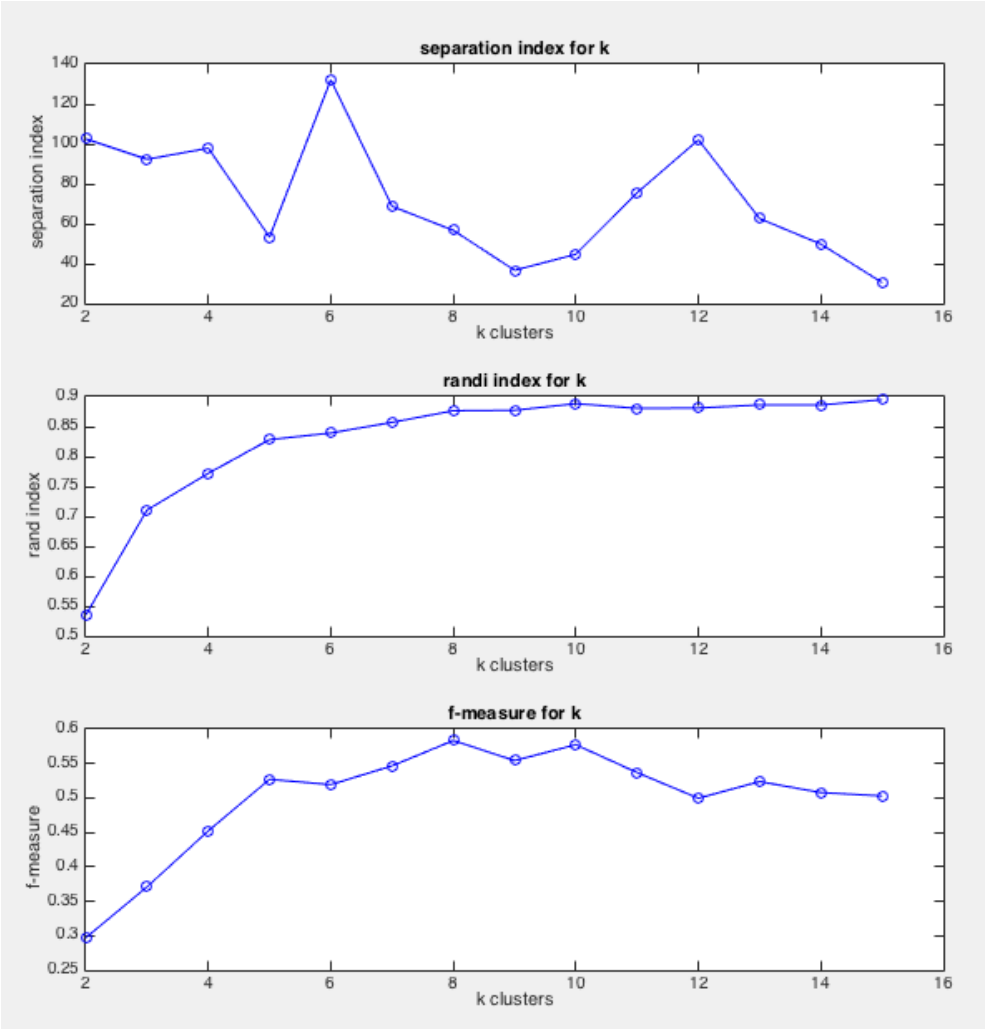

Figure 10: Separation Index, Rand Index, F-measure for K from 2 to 15

From the plots we could see that separation index was really high at k = 6 and k = 12, which means the clusters were not well separated for 6 clusters and 11 clusters. Rand index kept increasing as k got larger. F measure increased for the first half of plot, but decreased a little bit when k got too large. Considering 3 images together, we think the optimal numbers of k should be around 8,9 or 10.

2.3 Fuzzy c-means Algorithm

2.3.1 average cluster membership values and overlaps

In this section, we used thee Fuzzy c-means algorithm to cluster the data in to 10 clusters with a fuzzy parameter m=2.

In hard clustering, each pattern belongs to only one cluster. Fuzzy clustering extends this by allowing each pattern to have a membership in each cluster with different degree. But the result is overlapping clusters, not really partitions. Fuzzy C-means algorithm, which is an extension of k-means, used squared errors to handle membership values.

In order to find the average membership values for digit "1" and "8", we first clustered the data by setting cluster number to 10 and fuzzy parameter m to 2.0. Then found the index number for the two digits from the **gnd**. dataset, extracted corresponding columns from the membership value matrix **U**. Next, calculated the average membership values of the two digits belong to 10 clusters and plot a bar graph to explore their overlapping.

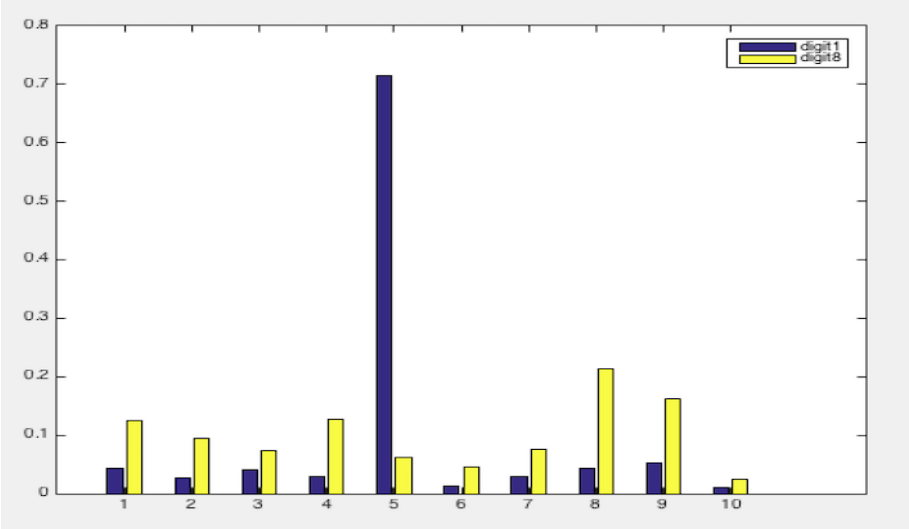

Figure 11: Average membership values for digit 1 and 8

We could see that blue bars denote the membership values for digit "1", which belongs to different clusters, while yellow bars stand for digit "8".

From the graph we can find that the maximum membership value of digit "1" was much greater than that of digit "8", which means the cluster digit "8" belongs to overlapped a lot with other clusters (cluster 9, 4, ...), while digit "1" overlapped much less.

2.3.2 Hard clustering

In this section, we used the hard clustering method which was mentioned in the question by setting the maximum value to be 1 and others to be 0. Then found the cluster number each sample belonged to and let this denote the classification result, then created an array to store these information.

From **2.2** we found that the optimal number of k should be around 8, 9 and 10, so in this section, we decided to choose k = 10 which was same as the cluster number we used in Fuzzy C-Means clustering. And then compare their performance according to Separation Index, Rand Index and F-measure.

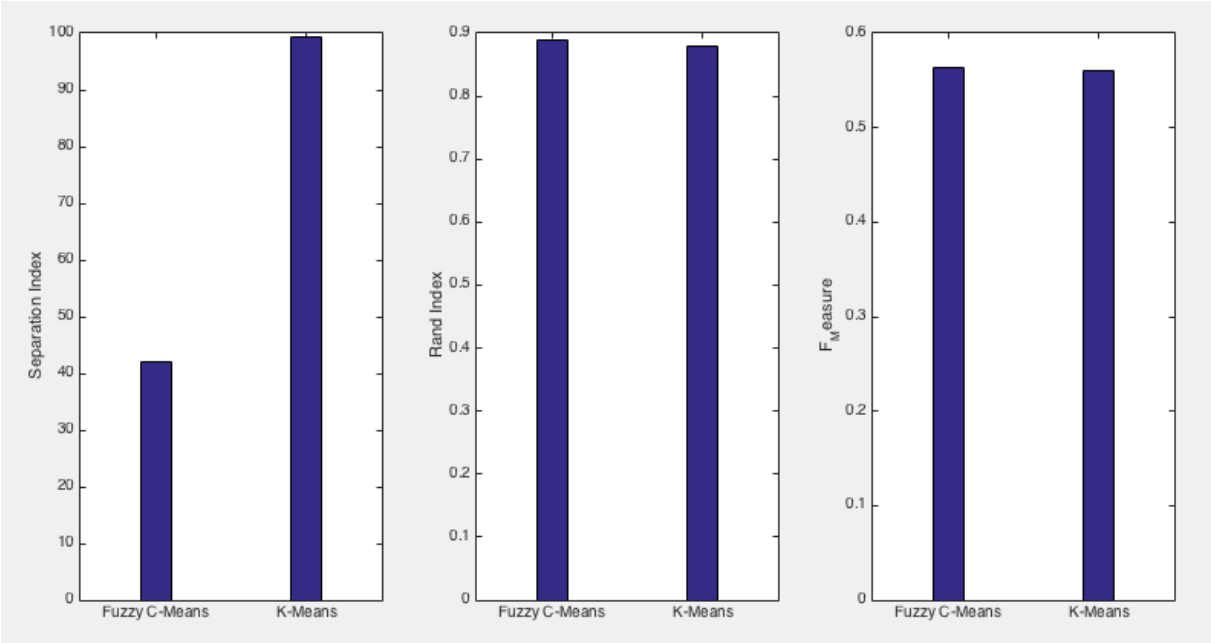

Figure 12: Separation Index, Rand Index, F-measure comparison

From the results comparison above, we found that K-means clustering performed a little bit worse according to F-measure and Rand Index, but from Separation Index, k-means clustering went much better than Fuzzy C-means. So we came out the conclusion that K-means clustering performed better in parallel when k = 10 for this dataset.

References

[1] LibSVM: A Library for Support Vector Machines. (http://www.csie.ntu.edu.tw/ cjlin/libsvm/)