Unsupervised Learning and Dimensionality Reduction

Ting-Yu Kang 903443289 tkang49

Datasets (same as assignment 1)

1. Letter Recognition Dataset (UCI Machine Learning Repository):
   This dataset is generated from black-and-white English characters with 20 different fonts. The images are converted into 16 numerical attributes for machine learning. The goal is to identify the character of each image. (https://archive.ics.uci.edu/ml/datasets/Letter+recognition)

2. Spam Mail (OpenML.org):
   This dataset is collected from real-world emails identified "spam" or "non-spam" by individuals. For each email, there are 57 attributes describing the characteristics of it. The purpose is to identify an email is Spam or not. (https://www.openml.org/d/44)

3. Dataset Comparison

	Letter Recognition	Spam Mail
Instances	20000	4601
Attributes	16	57
Classes	26	2

4. Interesting points
   (1) The two datasets are both classification problems. They are interesting because they are close-related with our real life. To illustrate, letter recognition technique can be used to identify texts, and the Spam Mail identifier can be implemented as a real-purpose spam filter.
   (2) Another reason I choose these datasets is that they are good for comparison. The Letter Recognition dataset has **more classes** than Spambase, while Spam Mail has three-times **more attributes** than Letter Recognition dataset. I am looking forward to new findings in assignment 3.

5. Tools
   (1) In this assignment, I implemented the algorithms with **Scikit-learn** machine learning library and data structures of Pandas and NumPy. The programming language I use is Python.
   (2) The programs can be executed on **Jupyter** platform.

Part 1: Clustering Algorithms

1. Experiment Overview and Methodology
   (1) In part1, I run K-Means algorithm and Expectation Maximization (EM) on the datasets, the goal is to find the best number of clusters (the value of k) for each dataset.
   (2) The datasets are separated into **80%** training set and **20%** testing set.
   (3) To evaluate the clustering performance of different k, I use the following indicators:

Need "ground truth" labels: (relations between clustering result and supervised labels)

Indicator	range	Perfect match score
Adjusted Rand index (ARI)	[-1, 1]	1.0
Mutual Information (MI)	[0, ∞]	∞
Adjusted Mutual Information (AMI)	[0, 1]	1.0
Normalized Mutual Information (NMI)	[0, 1]	1.0
Homogeneity, Completeness, V-measure	[0, 1]	1.0

No need "ground truth" labels: (evaluate the clustering result itself)

Indicator	range	Optimal cluster
Akaike Information Criterion (AIC)	[0, ∞]	0.0
Bayesian Information Criterion (BIC)	[0, ∞]	0.0

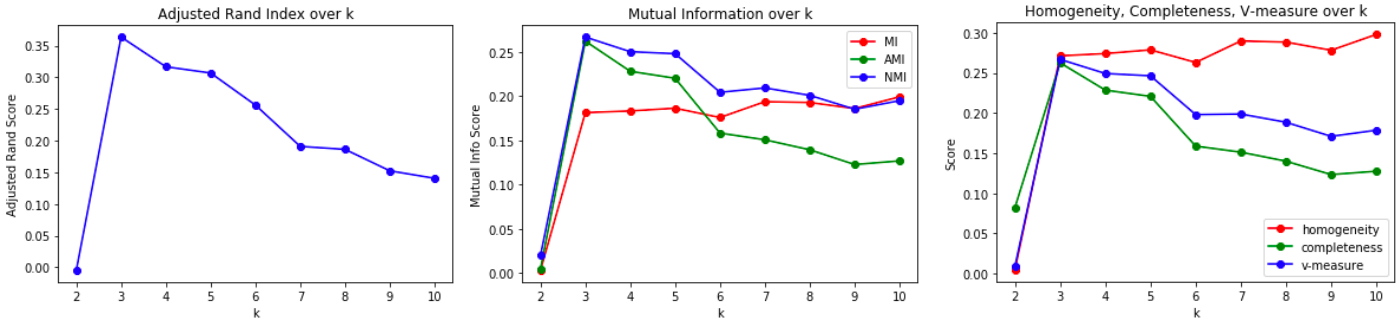
(4) In K-means, I choose k-means++ method to initialize the centroid. K-means++ makes the centroids to be distant from each other. The result will be better than initializing randomly.

(5) In Expectation Maximization, I choose covariance type = 'full' to give the components maximal freedom, and it indeed showed the best.

(6) Before doing clustering, I apply the minmax scaler on training data to unify the scale of all attributes and make clustering results more accurate. The scaler reduces the effects of noise and outliers.

2. Spam Mail Dataset (K-Means):

(1) This dataset has 2 classes, so $k = 2$ is expected to get the highest clustering score. However, when I applied the K-means algorithm on the training set, I found all the indicators showed that $k = 3$ is the best:



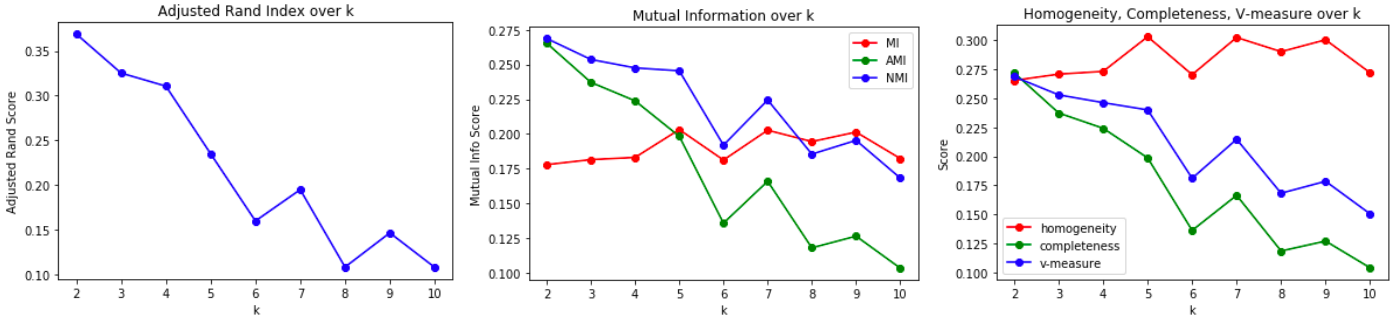
After examining the training set with confusion matrix, I found that when $k = 2$, K-means doesn't separate the data correctly. 99.3% of the points are assigned to cluster 0 (left table). When $k = 3$, the original cluster 0 are separated eventually (right table). It explains why $k = 3$ gets higher score than $k = 2$.

	Cluster 0	Cluster 1
Not Spam	2220	24
Spam	1436	0

	Cluster 0	Cluster 1	Cluster 2
Not Spam	304	1916	24
Spam	1029	407	0

The reason is, the center of cluster 2 is eight times farther than the distance between centers of cluster 0 and 1. Also, cluster2 has very low intra-cluster variance, causing the 24 points to be grouped first when $k = 2$. Therefore, these 24 points are the outliers.

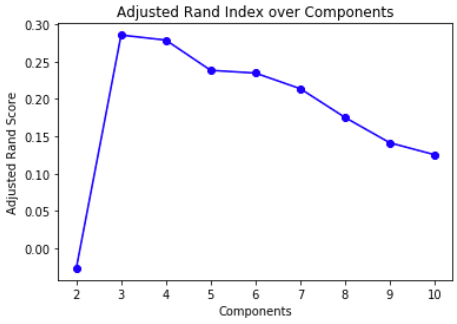
(2) Upon removing these points, the indicators correctly show that $k = 2$ is the best clustering model:

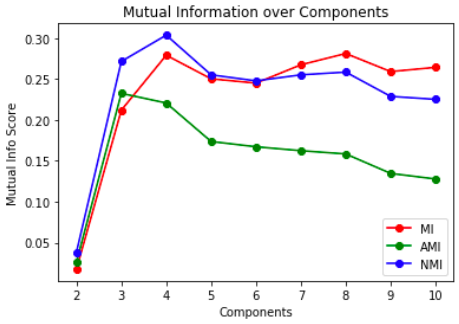


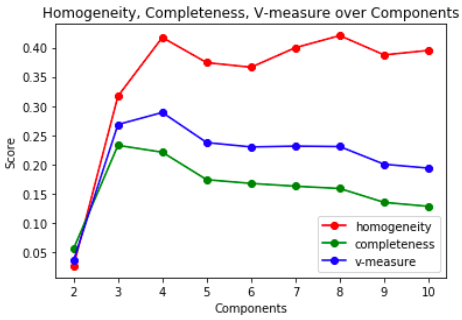
(3) Compared the clustering result ($k = 2$) with the class labels, the accuracy of training set is 80.47% and testing set get the accuracy of 81.76%, showing that the clusters don't overfit the data.

3. Spam Mail Dataset (Expectation Maximization):

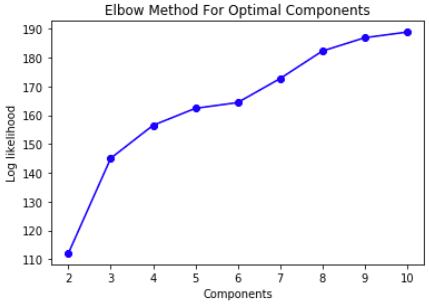
(1) EM is a probability distribution model, in scikit-learn library, it is implemented by Gaussian mixture model (GMM). When I ran it on the dataset, I noticed that all the indicators show that $components = 3\ and\ 4$ are better than 2:

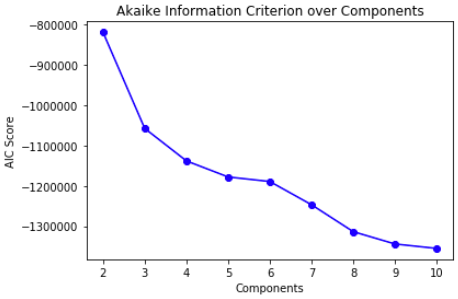


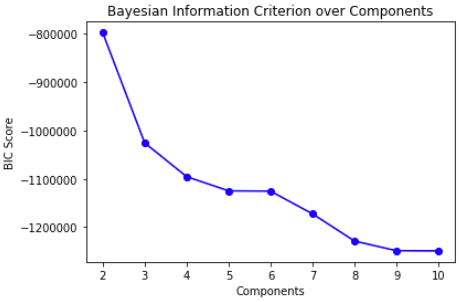




(2) Also, from the log likelihood, AIC and BIC, there are elbows at $components = 3\ and\ 4$:







(3) Therefore, I tried to analyze different components(C) from 2 to 4 using confusion metrics:

	C0	C1
Not Spam	307	1937
Spam	45	1391

	C0	C1	C2
Not Spam	256	1425	563
Spam	33	130	1273

	C0	C1	C2	C3
Not Spam	256	1023	693	272
Spam	33	66	141	1196

When $C = 2$, 3328 points (90%) are in C1 (left table), similar to the K-Means case. But in the other tables, it is showed that it indeed found three and four different components, not outliers.

(4) The reason might because the points of spam mails and not-spam mails are not well-compacted, causing EM to separate them into subgroups. Therefore, I tried to combine the components and compare them with the original labels:

	C0	C1
Not Spam	307	1937
Spam	45	1391
Train Acc.	61%	
Test Acc.	59%	

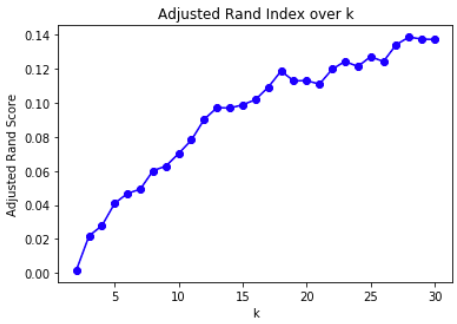
	C0+C1	C2
Not Spam	1681	563
Spam	163	1273
Train Acc.	80.3%	
Test Acc.	80.0%	

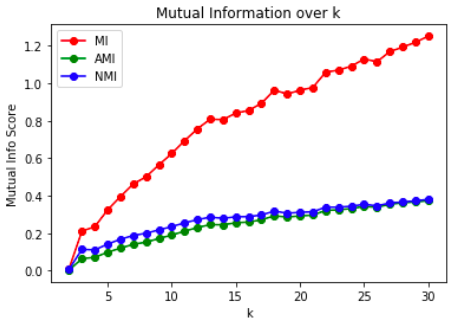
	C0+C1+C2	C3
Not Spam	1972	272
Spam	240	1196
Train Acc.	86.1%	
Test Acc.	85.8%	

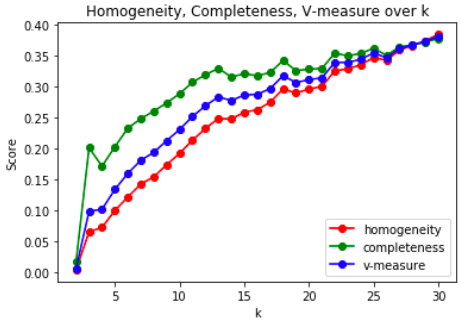
The models of $components = 3\ and\ 4$ get accuracies of 80.3% and 86.1% on training set; 80.0% and 85.8% on testing set, showing they indeed perform better than $components = 2$.

4. Letter Recognition Dataset (K-Means):

(1) This dataset has 26 classes, so I expected the scores of the indicators will go higher as k increases:





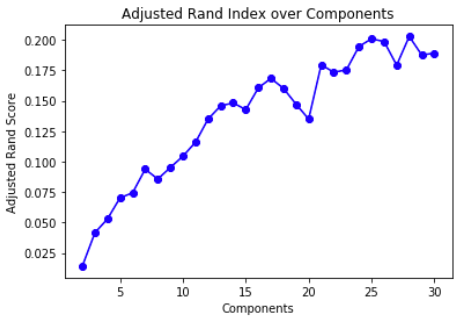


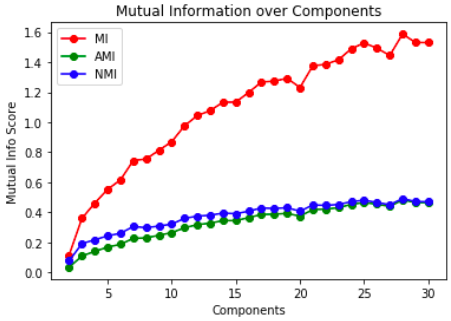
(2) The indicators show that the more clusters the better. The result is consistent with the ground truth (26 clusters) of the dataset. But we should be cautious with overfit if number of clusters keep increasing.

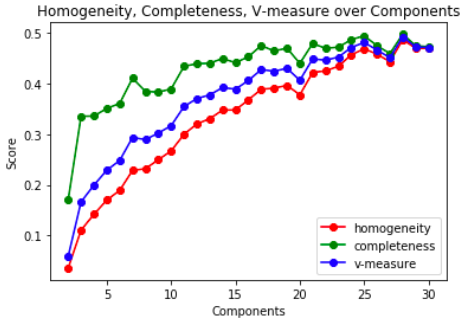
(3) After analyzing the clustering result with original data labels, there are only 30% of data with the same label are clustered in the same component in average. It might because there are characters with different fonts and same-shape confusing the clustering process. For example, Q and O might be clustered in the same group; 𝒁 and 𝓏 might also be grouped in different groups.

5. Letter Recognition Dataset (Expectation Maximization):

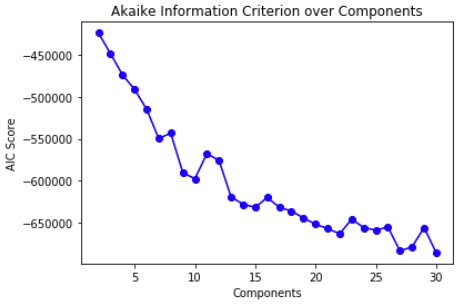
(1) The EM algorithm also shows that the larger number of components, the better separation:

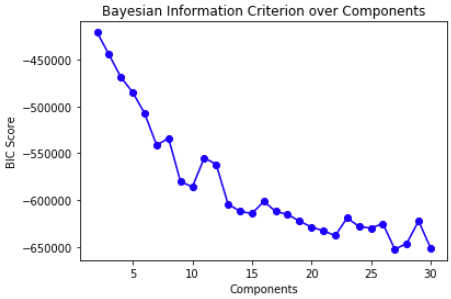






(2) To examine if the model overfits the data, I used AIC and BIC that panelize the model when there are too many components:





The lines go down slower as the number of components increases and be stable at around 25 to 30 components. AIC and BIC show that there are no overfitting issues so far, proving that 25 to 30 components cluster the dataset well.

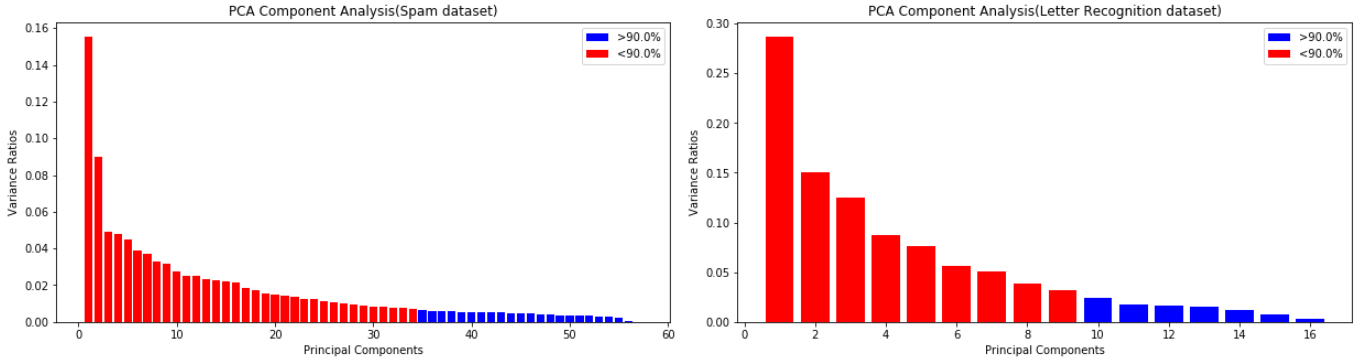
(3) In EM algorithm, there are 42.4% of data with the same label are clustered in the same component in average, better than K-Means but still not good enough. The reason might because the EM utilizes probabilistic model that gives flexibility for hard-classified data, so the accuracy is higher than K-Means.

(4) Although K-Means and EM didn't find the same clusters as the original labels, the interesting point is, they indeed found another way to cluster the data with 25 to 30 components as well.

Part 2: Dimensionality Reduction

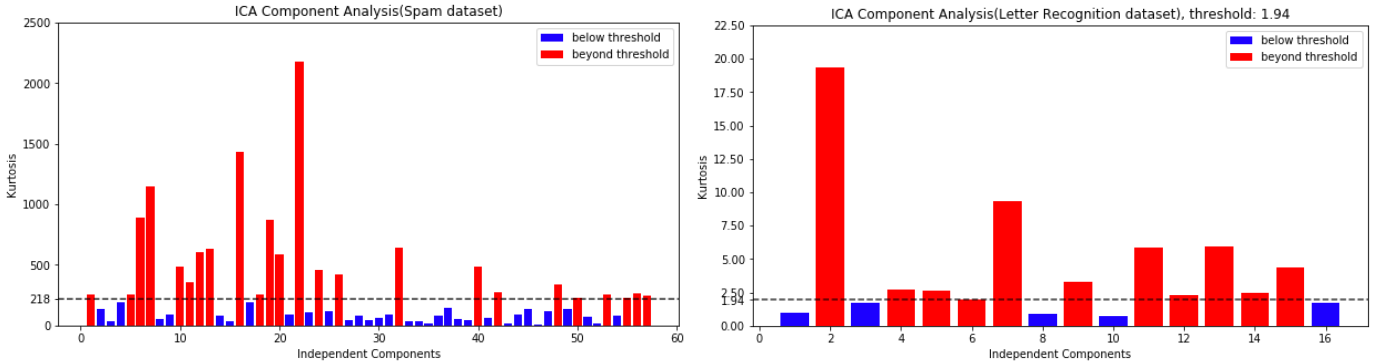
In this part, I applied four dimensionality reduction algorithms on my datasets: Principal Component Analysis (PCA), Independent Component Analysis (ICA), Randomized Projections (RP), and Feature Agglomeration (FA).

1. Principal Component Analysis (PCA)
   (1) PCA finds orthogonal eigenvectors that preserve the maximal amount of variance to be new components.
   (2) First, I conducted PCA on the two datasets and get the variance ratios of each component in descending order. The variance ratio is the amount of variance that each principal component can explain:



   (3) I set an aggregation threshold (90% for example) to determine how many components to keep (red bar) or to drop (blue bar). Noted that the variance ratio is in proportional to the eigenvalues.
   (4) By using 90% as the threshold, 37 (out of 57) components in Spam dataset and 9 (out of 16) components in letter recognition dataset should be kept.

2. Independent Component Analysis (ICA)
   (1) ICA finds the mutual independent components while maximizing mutual information from original dataset. It is mainly used for finding hidden variables of basic elements of an image or voice data.
   (2) Based on the central limit theory (CLT), the linear combination of independent variables tends to form a Gaussian distribution. ICA is the reversion that decomposes it back to independent variables with maximal possible non-Gaussianity (unordered). Kurtosis is a good indicator to measure Gaussianity:



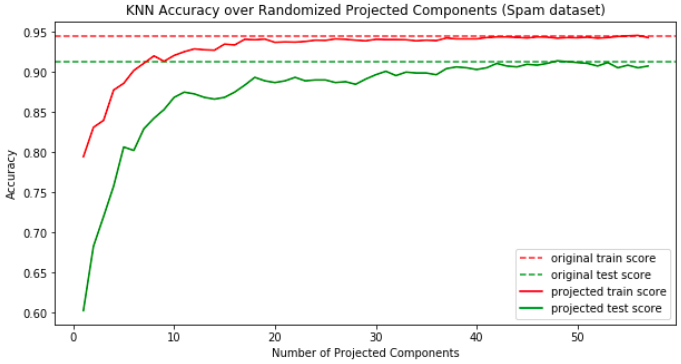
   (3) Each bar is the kurtosis (absolute value) of each independent component found by ICA. The closer to 0, the components are closer to Gaussian distribution, and less likely to be the hidden variable we want.
   (4) Therefore, I set a threshold (10% of the maximal kurtosis for example) and only keep the components above it. In this case, 33 (out of 57) components in Spam dataset and 11 (out of 16) components in letter recognition dataset should be kept.
   (5) Although all of the components are independent with each other, they don't necessarily have semantic meaningful information. In my both datasets, they are digit-based, so it's hard to visualize each component, and I can only rely on kurtosis and try-and-error to find the optimal number of dimensions.

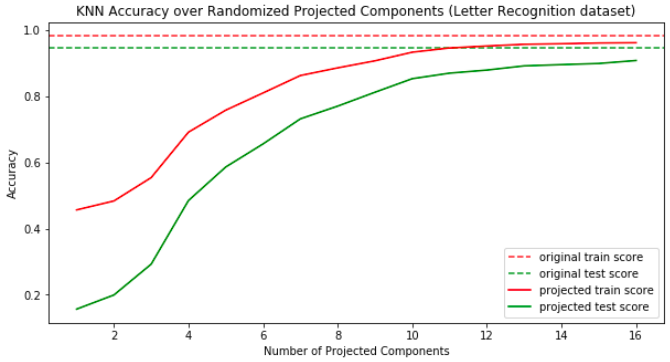
3. Randomized Projections (RP)

(1) Compared to PCA that needs heavy computation on maximizing variance, RP is a simple and efficient dimensionality reduction algorithm that uses a random generated projection matrix to map datasets into lower dimensional spaces.

(2) In practice, I used Gaussian random projection in scikit-learn where the elements of projection matrix are drawn from $N(0, \frac{1}{number\ of\ components})$. To avoid sampling bias, I repeated 10 times for generality.
In fact, after several random restarts, RP's accuracy is surprisingly stable as long as the dimensions are large enough.

(3) I used K nearest neighbor (KNN) to illustrate the performance of RP over different choices of components. Theoretically, RP will preserve the Euclidian distances between data points after projection. Therefore, KNN, for its distance-based nature, is a good algorithm:





(4) From the upper left chart, we can see that only 20 components (35%) are enough for Spam dataset to achieve the accuracy close to the original dataset. From the upper right chart, 10-12 components (62%-75%) are needed for Letter Recognition dataset.

(5) The difference of percentage might because Spam dataset only has 2 classes, but Letter Recognition dataset has 26 classes. Therefore, the latter needs more dimensions to make precise classification.

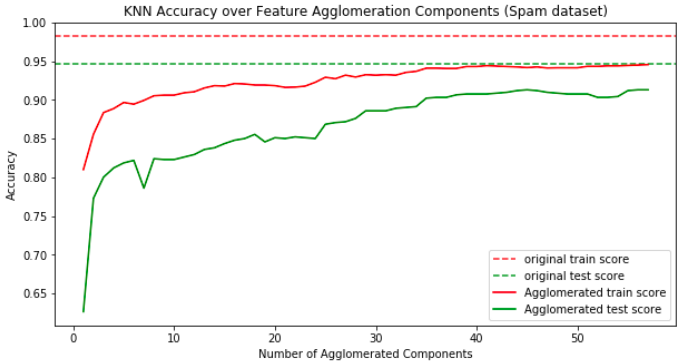
(6) The reconstruction of RP is hard to analyze the performance compared with KNN because of the digit-based datasets. It's also hard to visualized given that the datasets have too many dimensions.

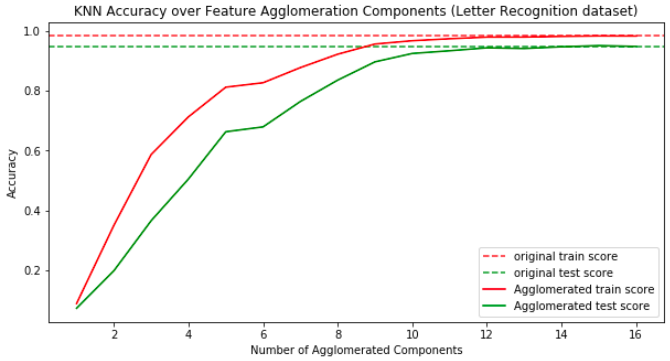
4. Feature Agglomeration (FA)

(1) Agglomeration Clustering (AC) was mentioned in the class right before K-Means. AC creates k cluster in n-k iterations and merges two nearest clusters for each iteration.

(2) FA is very similar to AC, but it aggregates correlated features instead of data points. The mathodology is to transpose the data matrix (swap data row and feature column) and run AC on it. The dimensions are reduced by merging algometric features with arithmetic mean.

(3) I used KNN to illustrate how many components should be kept to maintain acceptable accuracy:





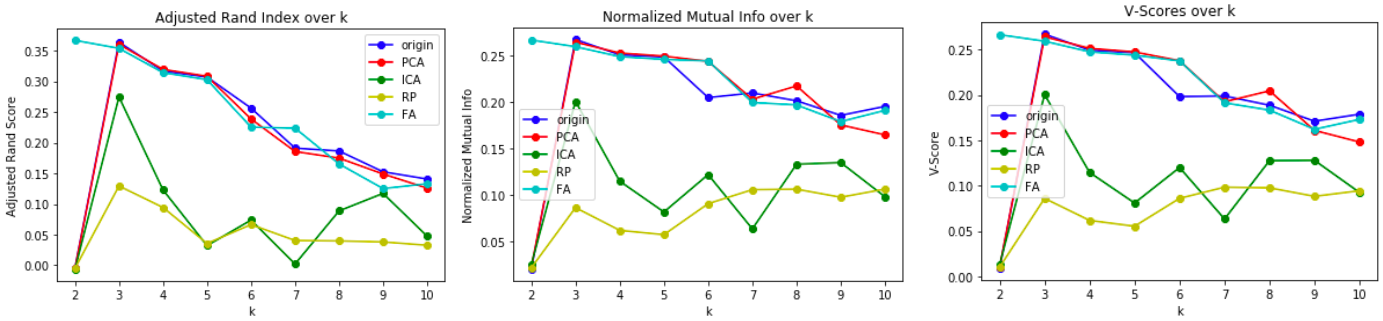
(4) For the Letter Recognition dataset (upper right), FA performs better than RP that the training and testing accuracies are nearly the same as the original ones with at 10-12 components (62%-75%). But for the Spam dataset (upper left), the accuracies are 3% lower.

(5) According to the class, Agglomeration method might not perform well with some special structures (like a dot in a circle). The 57 featured of Spam dataset might be that case and also contain more noise than Letter Recognition dataset, misleading feature merging and causing the loss of accuracy.

Part 3: Clustering on Dimensionality Reduced Dataset

- In this part, I reproduced two clustering algorithms (from part 1) on the dimensionally reduced datasets (from part 2). I used three indicators: Adjusted Rand Index, Normalized Mutual Info, and V-Scores to evaluate the clustering results with original data labels.
- The dimensions for reduction are chosen from part2 where the KNN accuracy is 90% of the original dataset.

1. Spam dataset (K-Means) (Original: 57, PCA: 34, ICA: 25, RP:20, FA:40):



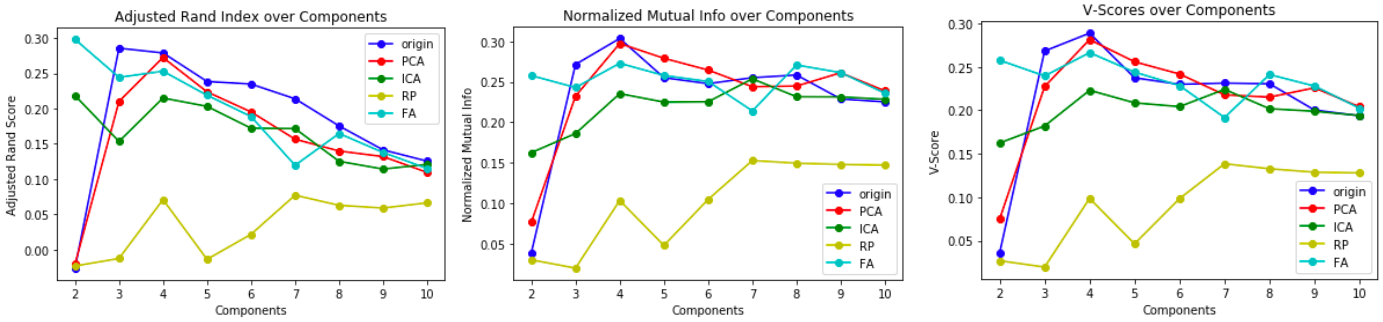
(1) From the indicators, we can see that except FA, all the other reduction algorithms show that $k = 3$ is the best, in consistent with the conclusion from part1.

(2) FA shows that $k = 2$ is the best. It might because during the process of feature aggregation, the effect of outliers (see part1) is diminished, letting the dataset be correctly separated when $k = 2$.

(3) PCA and FA are very close to the original clustering result with only 34(60%) and 40(70%) dimensions, effectively reducing the dimension while keeping maximal information. The same clustering result is because FA only combines features linearly without changing the axis, and PCA tends to increase inter cluster distance by maximizing variance.

(4) RP and ICA make clusters different. RP's random projection and ICA's independent dimensions cause K-Means to separate the datasets in different way. Random centroid initialization of K-Means also affects the clustering result.

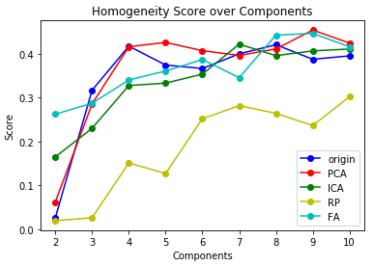
2. Spam dataset (Expectation Maximization) (Original: 57, PCA: 34, ICA: 25, RP:20, FA:45):



(1) From the indicators, we can see that except RP, all the other reduction algorithms show that $k = 3$ or $k = 4$ is the best, in consistent with the conclusion from part 1.

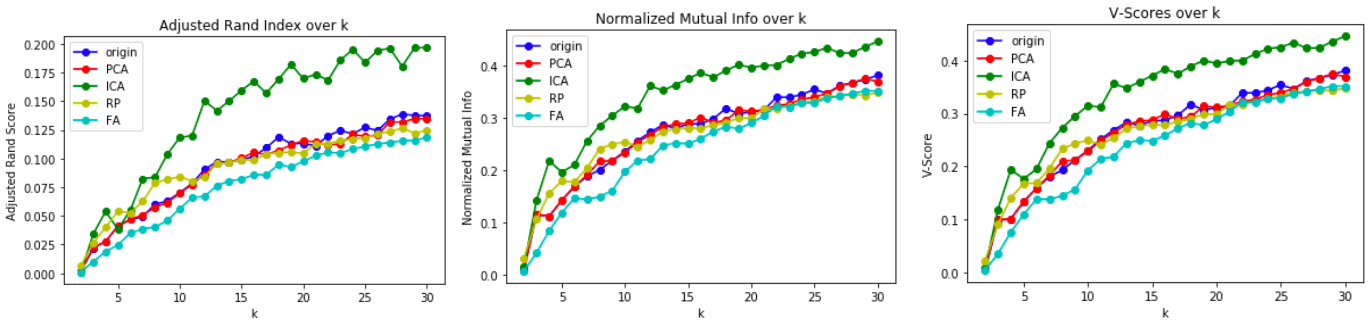
(2) When the dimension of FA was set to 40, the indicators fluctuated a lot, so I increased it to 45, and the result converged. It showed that for different algorithm, we should fine-tune the number of dimensions to be kept so as to preserve enough information for clustering.

(3) RP shows a peak at 4 clusters, the same as others. However, after 7 clusters, the score increases. It indicates that 7-10 clusters might be good as well for it just divide the original data with the same label into subgroups. The homogeneity score (right figure, evaluate if each cluster has a single label) proves my guess as it reaches the maximum at 7 clusters.



(4) In this case, RP and ICA still make clusters different, but ICA is closer to the original clustering result. PCA and FA are close to the original clustering result.

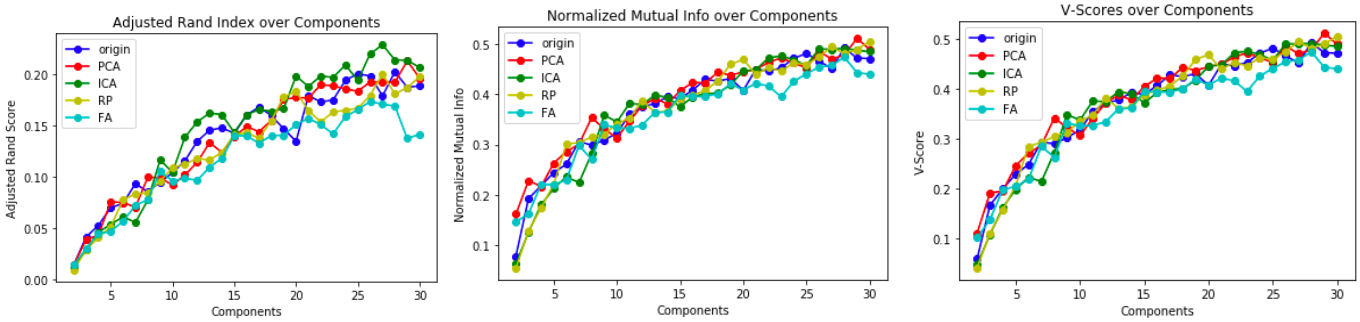
3. Letter recognition dataset (K-Means) (Original: 16, PCA: 9, ICA: 11, RP:12, FA:10):



(1) The clustering of PCA, RP and FA are close to the original clusters. It shows that for letter recognition dataset, the clusters are compact and inter-cluster distances are large. Therefore, the same cluster result can be kept after dimensionally reduction.

(2) The score of ICA is the highest, showing that K-Means might get better clustering result on ICA-reduced dataset. The reason is, this dataset is derived from images of English letters, and ICA is good at extracting independent attributes out of image or voice dataset. It might be the reason why ICA performs the best.

4. Letter recognition dataset (Expectation Maximization)

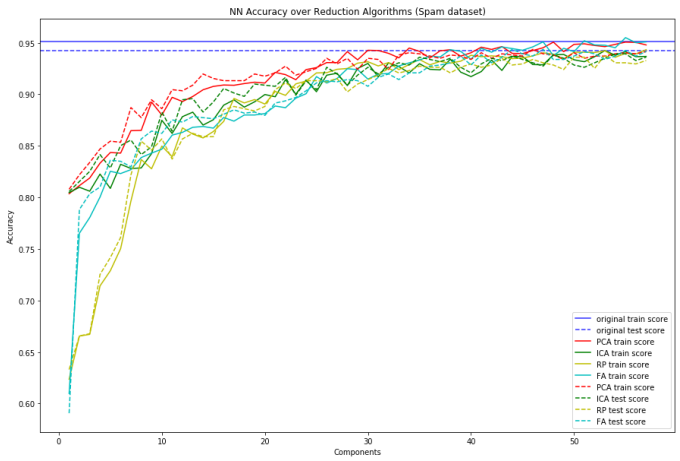


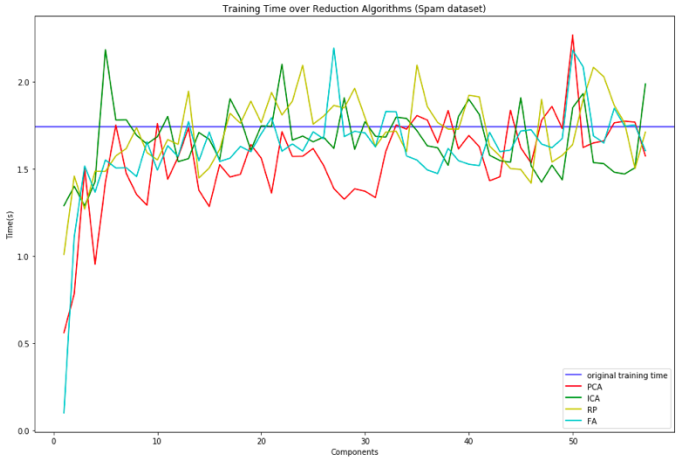
(1) The result of EM is similar as K-Means, but compared to K-Means, the scores of PCA, RP and FA are increased to be the same as ICA. It shows that EM finds better clusters than K-Means. Also, the concave curve shows that 25-30 is the best choice, same as the result of part1.

(2) From the indicators, all reduction algorithms have similar behaviors. They not only show the effectiveness of dimension reduction algorithms, but also tell us that this data set is clean and stable, so that the reduced dimensions can keep most information without distracting by noise.

Part 4: Neural Network on Dimensionality Reduced Dataset

In this part, I chose Spam dataset (2 classes) because it is easier to be analyzed. The neural network (NN) is configured the same as Assignment1 with structure (10, 10) and max iteration 200 to prevent overfit:





(1) The upper left graph shows that all four reduction algorithms perform better as the number of dimensions increases and eventually approach the original NN accuracy.

(2) The training accuracies (solid lines) and testing accuracies (dotted lines) are nearly the same, showing that the reduction algorithms don't overfit the training data.

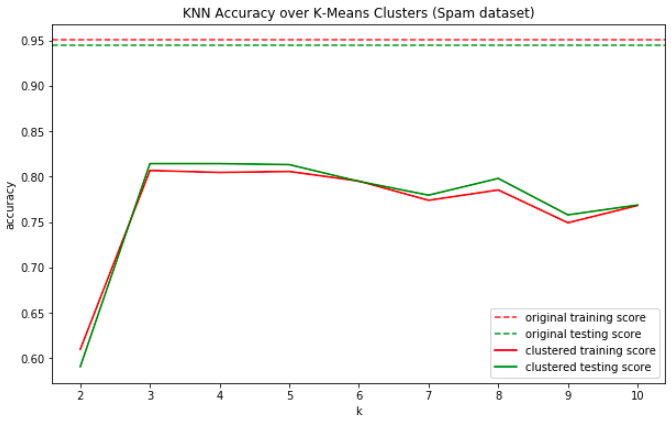
(3) PCA and ICA reaches 85% accuracy with only 10 components, effectively reduced the dimension without losing information too much. It shows that most information of this dataset can be captured by keeping at least 10-20 (17%-34%) of components.

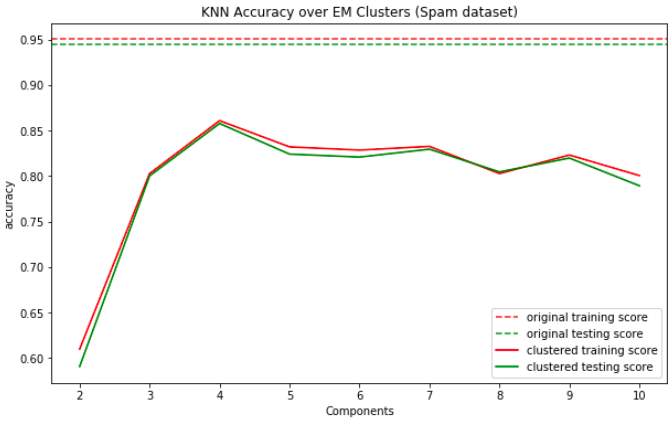
(4) RP and FA don't perform well in the beginning. For RP, it is based on the randomness of projection matrix, and datasets with too few features are easily affected by noises and bias. For FA, it merges features based on the similarity; if we set the dimension too small, it will overly merge the features that are not that similar, causing bad accuracy.

(5) The upper right graph compares the training speed of different reduction algorithms with the baseline NN. We can see after 10 dimensions, the running times fluctuate within 0.5 seconds around the baseline (blue line). The reason is because the structure of NN is (10, 10), so no matter how many dimensions, it will be mapped to 10 perceptrons. Therefore, after 10 dimensions, the running time doesn't change.

Part 5: Neural Network on Clustered Dataset

In this part, I also used Spam dataset and the same neural network (NN) as part 4. Also, I reproduced the clustering algorithms in part1 and make the cluster results as new labels. Below is the accuracy the NN model training on the new labeled datasets:



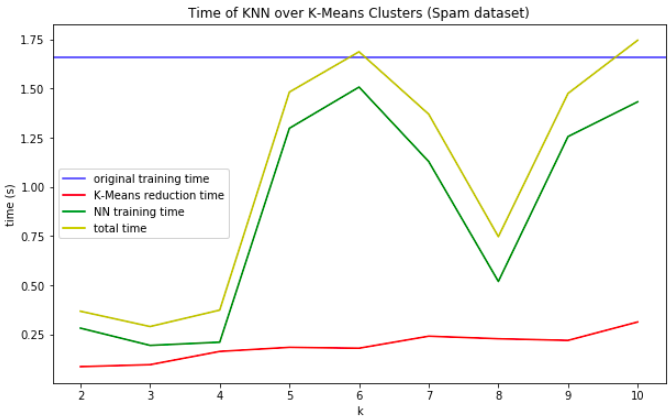


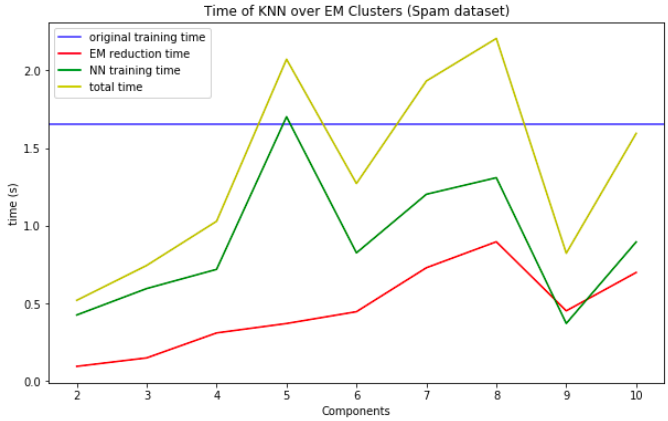
(1) The upper left graph is K-means clustering. It shows that $k = 3$ has the best training and testing accuracy, exactly the same as my analysis in part1. It is because there are 24 outliers that are so far away from the other data points that clustered by K-Means first. $k = 2$ performs bad because 99.3% points are grouped in a single cluster.

(2) When $k = 3$, the correct separation is performed, so the accuracy raises. After that, the accuracy remains because K-means just keeps separating the current clusters into sub-clusters. The testing accuracy shows that it doesn't overfit.

(3) The upper right graph is EM clustering, although when $k = 3$, the accuracy is the same as K-means, it shows that it has 5 % better accuracy at $k = 4$, also in consistent with my analysis in part1. The reason is EM use probability distribution instead of exact fitting like K-means, it can be more flexible when there are some data equally distant with different centroids, therefore, EM has more chance to cluster hard-determined points, so it preforms better.

(4) In total, these two unsupervised algorithms still cannot reach the same performance as supervised algorithms (KNN in this case). It makes sense because they don't have any labels telling the real answer. The only thing they know is the correlation of data points, so there must exist misclassification where some of the data points are not separated well, or there are some outliers.





(5) Compared the running time of clustering + NN training with pure NN, we can see both K-Means and EM take only 25% of normal NN training time when k is 3 or 4 (peak accuracy) after that, the training time surges because more clusters make NN model more complex. Also, it makes sense that EM needs more training time than K-Means because of heavier probability calculations.

(6) In summary, by applying clustering algorithms to reduce dimension, the performance can reach 80%-85% with 4 times training speed than pure NN. Accordingly, the clustering algorithms are very useful for time sensitive but more fault-tolerant tasks.

Reference

[1] Scott Robinson. K-Means Clustering with Scikit-Learn. Retrieved from https://stackabuse.com/k-means-clustering-with-scikit-learn/

[2] Dina Jankovic. K-means Clustering of Wine Data. Retrieved from https://towardsdatascience.com/k-means-clustering-of-wine-data-95bac074baae

[3] Tola Alade. Tutorial: How to determine the optimal number of clusters for k-means clustering. Retrieved from https://blog.cambridgespark.com/how-to-determine-the-optimal-number-of-clusters-for-k-means-clustering-14f27070048f

[4] Gaussian mixture models. Retrieved from https://scikit-learn.org/stable/modules/mixture.html

[5] In Depth: Gaussian Mixture Models. Retrieved from https://jakevdp.github.io/PythonDataScienceHandbook/05.12-gaussian-mixtures.html

[6] Independent Component Analysis. Retrieved from http://users.ics.aalto.fi/whyj/publications/thesis/thesis_node8.html

[7] ICA on Images with Python. Retrieved from http://theautomatic.net/2018/06/23/ica-on-images-with-python/

[8] Measures of Non-Gaussianity. Retrieved from http://fourier.eng.hmc.edu/e161/lectures/ica/node4.html

[9] Experimenting with Random Projections—A Non-maths approach. Retrieved from https://ashokharnal.wordpress.com/tag/random-projections-tutorial/