HW3: Unsupervised Learning and Dimensionality Reduction

Jiun-Yu Lee GTID: 903435223

1 Introduction

1.1 Datasets

The datasets we will use in the following experiment is those two that we have used in assignment 1. Specifically, We will use the following two datasets

- **Credit Card Fraud Detection (CCD):** This dataset contains the real world transaction data that can be either good or fraud, which has an extremely unbalanced data distribution. We downsample the dataset to form another one with approximately 10% of fraud data. The goal is to predict whether a given new transaction is fraud or not.
- **MNIST dataset:** MNIST dataset contains hand-written digit images of data. The reason for choosing this dataset is that it is a high dimension (784) data, and we want to discuss the influence of Curse of Dimensionality. It is also worth to discuss this phenomenon in clustering and dimension reduction. We preprocessed the data to a binary classification problem of 4 and 9 (this two digit is often considered to be similar) so that it is easier to discuss our result.

We will conduct various clustering and dimensionality reduction algorithm on these two datasets and analyze the result in the following section.

1.2 Experiment Environment

Here, we will introduce several constraints in our experiment so that we can analyze our result more precisely:

- All of our implementation of clustering and dimensionality reduction algorithm is based on Sikit Learn, a python open source machine learning library.
- To visualize the clustering result, we will use PCA to project and plot the data in a 2d figure.
- We chose Euclidean distance as the distance/similarity metric for our clustering algorithm. There are several reasons behind it. First, for the algorithm such as KMeans, it is designed to be suitable for Euclidean distance, in the way it calculates the center (centroid) and what it is minimized (sum of square error within a cluster). With other kinds of distance metrics, KMeans will no longer guarantee to converge. Second, the clustering algorithm implementation from sklearn simply does not provide the functionality of using a distance metric other than Euclidean distance. Therefore, in this experiment, we only use Euclidean distance as our similarity metric when doing clustering.
- For the neural network experiment, we will use the same optimal hyperparameter we found in assignment 1 to ensure a fair comparison.

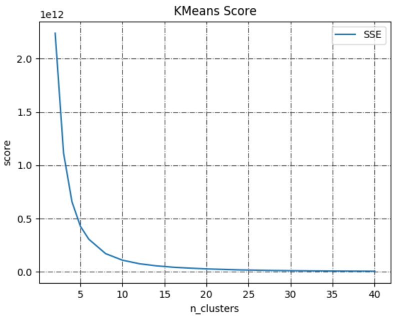
2 Clustering

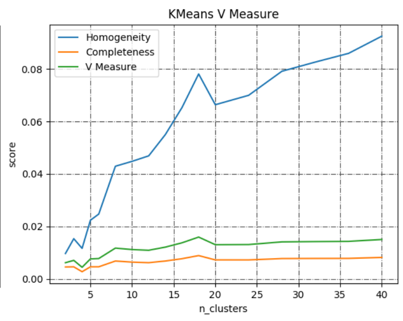
In this section, we will analyze two of the clustering algorithm, KMeans and Expectation Maximization, and their performance on our two datasets. The following metrics/methods are used to evaluate our clustering result:

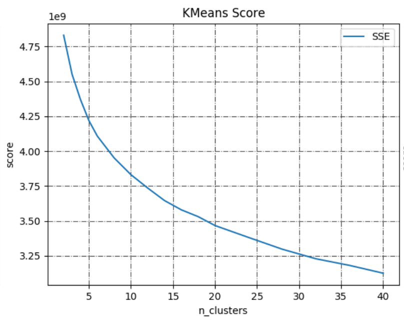
- **Elbow method:** This method is designed to help finding the appropriate number of cluster for a dataset while reducing the chance of overfitting. It suggests that we should choose a number so that adding another cluster doesn't give much better modeling of the data. We use inertia (the sum of square error within clusters) and log-likelihood as a measurement for KMeans and EM, respectively, and plot a graph to determine the "elbow point".
- **V measure:** this is a metric designed to evaluate how well the clusters line up with the ground truth labels. It is a harmonic mean of the following two measures:
  - **homogeneity:** each cluster contains only members of a single class.
  - **completeness:** all members of a given class are assigned to the same cluster.
- **Silhouette Coefficient:** This score aims to evaluate how good a cluster is. A higher score means the cluster is denser and better separated.

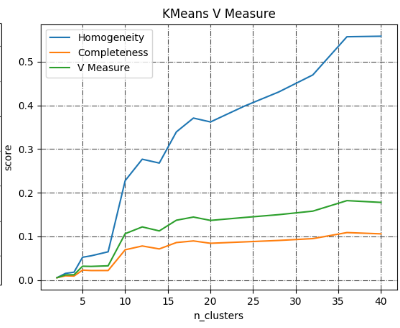
2.1 KMeans

We first discuss one of the famous clustering algorithm - KMeans - and its performance on our two datasets. KMeans try to separate the data into K groups by picking the centroid of each group that minimizes the inertia, or within-cluster sum of square error. The hyperparameter we need the adjust here is the number of clusters (k).





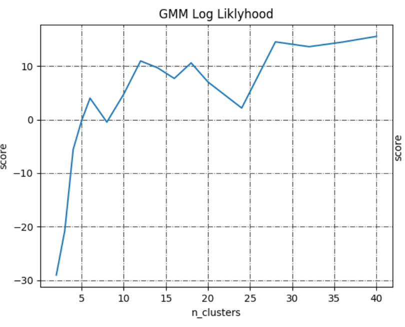


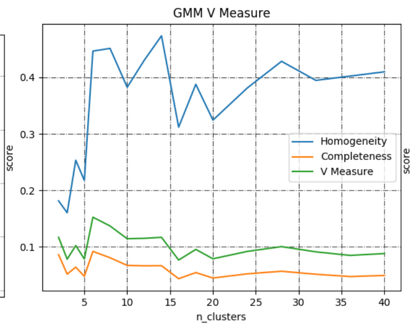


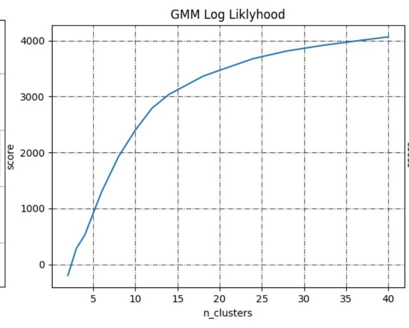
The left two figures are the result for CCD and the right two are for the MNIST. Using the elbow method on the inertia figure, we can determine the appropriate number of k should be 10 for CCD and 20 for MNIST. When we see V measure figure, we will find that both of the clusters are not aligned very well with the ground truth label. This is under expectation because KMeans is trying to find clusters that are "geometrically" dense and well separated. On the other hand, the ground truth label may have lots of overlap between two classes, making KMeans hard to find clusters well-aligned with the ground truth label. In this case, we can see that the KMeans V measure on CCD is far poorer than on MNIST. the reason for this is that the CCD dataset is highly imbalanced. If you see the cluster visualization in the conclusion, you will find that the positive class of CCD is sparse, wide-spread, and overlap with negative class, making KMeans even harder to find well-aligned clusters.

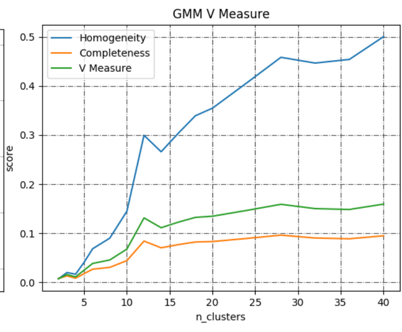
2.2 Expectation Maximization (EM)

In this subsection, we will analysis another clustering algorithm, Gaussian Mixture Model (GMM) with Expectation Maximization. Unlike KMeans, it is a probabilistic model that assumes all the data points are generated from a finite number of Gaussian distributions. Each data point has k probability indicating how likely this point will belong to each cluster. Again, we have to find an appropriate k value for each dataset.





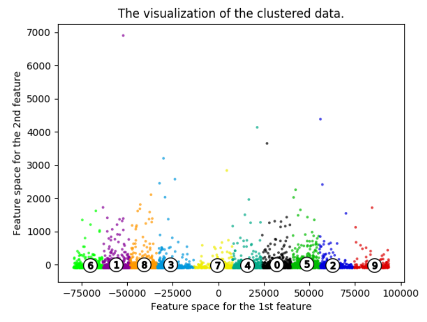
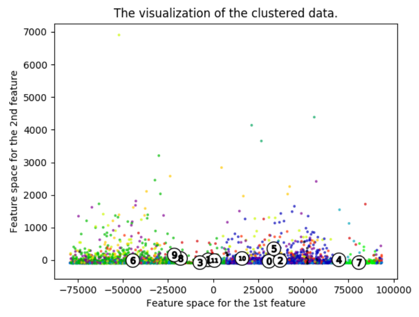
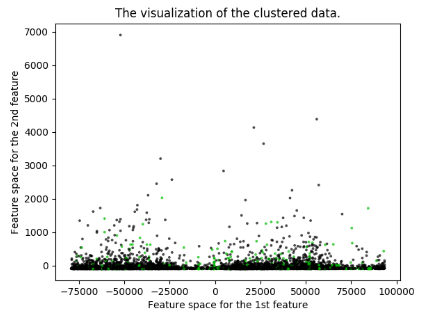
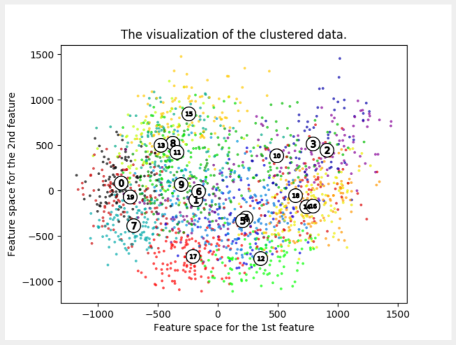
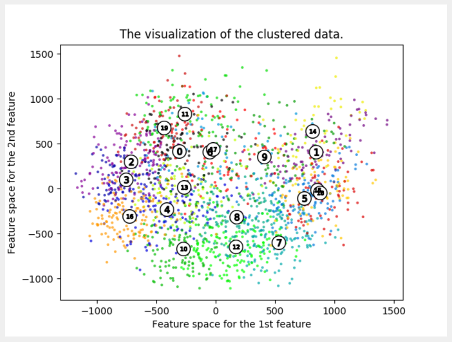
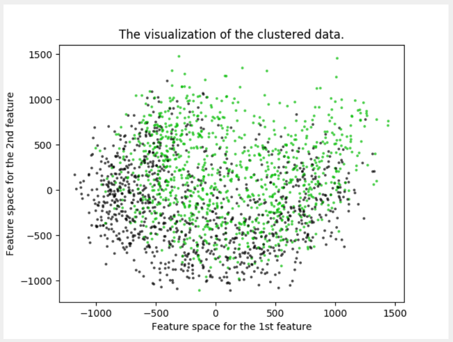




Again, the left two figure is for CCD and the right two is for MNIST. We will find that using the elbow method, we will select a similar k value as we previously did in KMeans, k = 12 for CCD and k = 20 for MNIST. Despite the similarity between the number of clusters, we can still find some differences between V measure score. We can see that, compared with KMeans, the V measure score in GMM increase significantly. Although it still doesn't imply well-aligned clusters, it still gives us an insight that EM can possibly do better than KMeans on lining up with the ground truth label. The reason is that GMM with EM is soft clustering algorithm and can have an arbitrary shape for each gaussian, making it more capable of lining up with a weird distribution such as the distribution of the positive class in CCD, while KMeans is a hard cluster algorithm and only calculate the distance to the centroid. The V measure in MNIST, however, doesn't change much between two algorithms. I will argue that it is due to the high dimensionality of MNIST. Such a high number of dimensions makes the data point so sparse that we didn't benefit too much from the properties of GMM.

2.3 Conclusion

The following table shows the cluster visualization of each algorithm (with the k value we found in previous subsections) and ground truth with respect to each dataset:

	KMeans	**EM**	**Ground truth**
CCD			
MNIST			

Before we start to explain this cluster visualization, we need to first clarify how representative of the visualization for each dataset. For CCD, after we transform the data into two-dimensional space using PCA, we can still preserve a 99% of explained variance ratio. In fact, if you see the PCA experiment in the next section, we can found that only one dimension can still preserve 99%, which is why the data point of CCD is highly skewed at the bottom the figure and variate mainly along the x-direction. Therefore, we can conclude that the visualization of CCD is reliable. As we can see in the KMeans result for CCD, the clusters seem to line up pretty well along the main x-direction. This makes sense because CCD variate mainly in one direction in the original space and therefore the KMeans cluster result also lines up with that direction. However, in the ground truth visualization, we can see that the positive data (green dot) spread throughout the entire x-direction and highly overlap with the negative data. This results in a poor V measure as we mentioned in the previous subsection. In contrast to KMeans, the EM result on CCD has lots of overlap between each cluster. Normally, the clusters produced by EM is considered to be worse than KMeans since it doesn't separate the data well (the Silhouette Coefficient for the result of GMM is near 0 while the coefficient for the result of KMeans is 0.54). However, this overlap seems to make EM better capture the weird distribution of CCD, making its V measure better than KMeans.

For the MNIST dataset, it is relatively hard for us to do visualization due to the high dimensionality. After we conduct PCA to transform the data into two dimensions, it only preserves around 22% of explained variance ratio, which mean, unfortunately, this visualization is not enough to demonstrate the original distribution of the data. Moreover, due to the curse of dimensionality, the data points become so sparse that neither of the two algorithms can find a dense and well separate cluster (the Silhouette Coefficient is 0.054 for KMeans and 0.060 for EM).

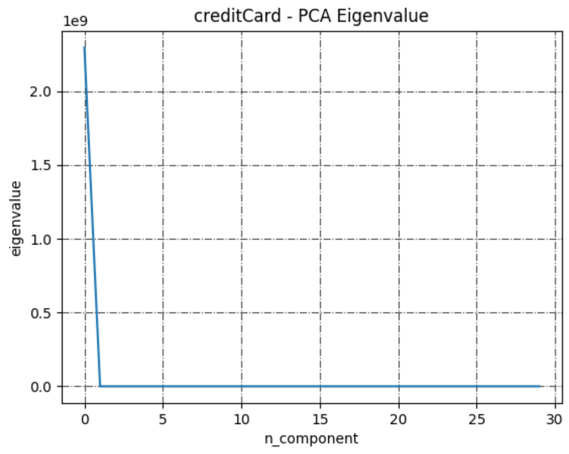
Although the results are not considered to be a very good clustering, there are several ways we can improve the performance. First, the fact that only one direction have high variance may due to the different scale between features. One could do a normalization to alleviate this issue. Also, conducting dimensionality reduction may help to improve the performance, and we will discuss it in the next section.

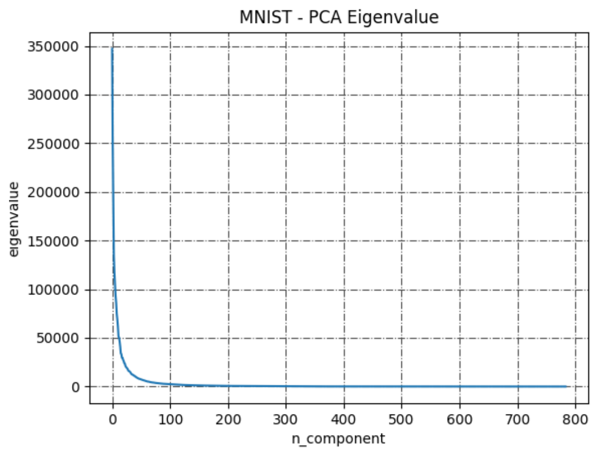
3 Dimensionality Reduction

In this section, we will discuss four famous dimensionality reduction algorithm and how it affects the performance of the clustering algorithm on each dataset.

3.1 Principle Component Analysis (PCA)

We start from Principal Component Analysis (PCA), a feature transformation algorithm that aims to find a linear combination that projects features to new axes that maximize the variance. PCA has one hyperparameter, the number of components to keep. To find an appropriate value for this parameter, we should first look at the eigenvalue/variance distribution in the transformed space.

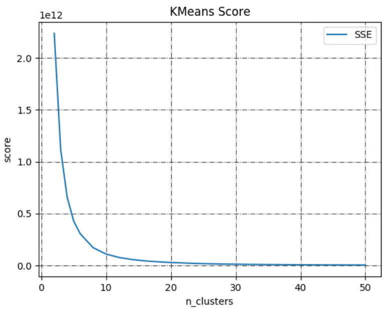


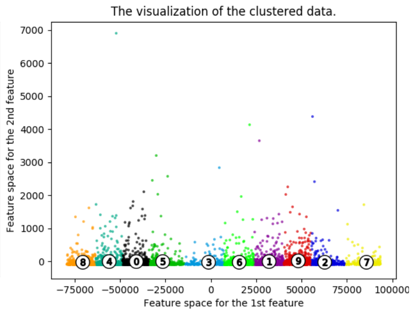


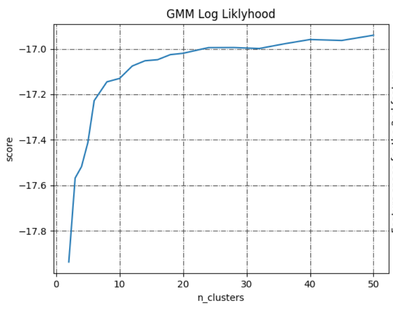
We can immediately see an odd result from CCD's eigenvalue, that is, only the first dimension in the transformed space has high eigenvalue/variance. The eigenvalues for the remaining 29 dimensions are nearly 0. This indicates that CCD data variate nearly only in one direction in the original space. Combined with the CCD ground truth figure we show in the previous section, we will find that the direction with the highest variance isn't necessary to contain sufficient information for us to cluster/classify data. In this case, we can see that the positive class is still completely mixed up with negative class even though we have projected the data to the axes with the highest variance. This result shows us a potential drawback of PCA, that is, the transformed axes doesn't necessary to be more useful. The original space may have other directions that contain the information allows us to separate the data well, but its variance is so small that it is thrown away by PCA.

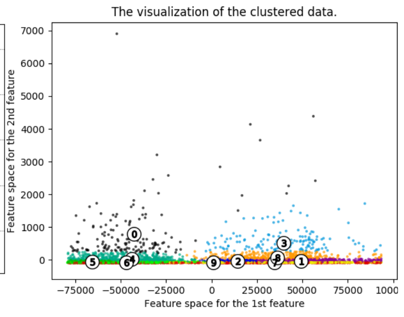
Despite the odd result we found in CCD, the MNIST dataset act more naturally after conducting PCA. The transformed space of MNIST has high variance in the first 100 axes and relatively low variance in the remaining axis. We then make use of the explained_variance_ratio_ provides by sklearn PCA module to determine the appropriate value of n_components. The explained_variance_ratio_, derived from eigenvalues, indicate the percentage of variance explained by each axis. A good transformation should have a high enough cumulative ratio. For MNIST, we choose n_components = 115, which gives us a cumulative ratio of around 95%. For CCD, since using only one dimension is a little bit unreasonable and not good for visualization, we instead pick n_components = 2, which result in a ratio above 99%.

Now we will discuss results when we apply clustering algorithms to the PCA reduced data.

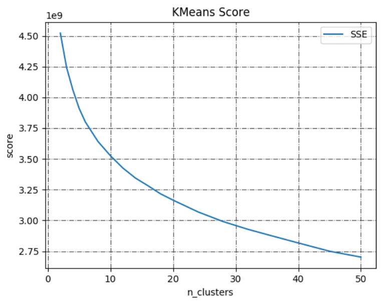


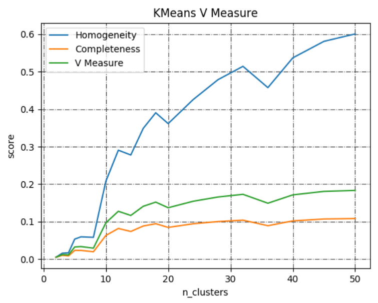


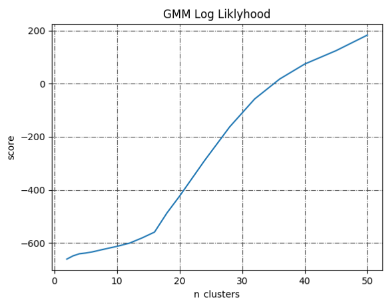


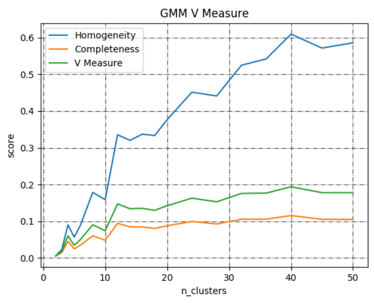


Above figures are the results of CCD. Using the elbow method, we determine the optimal k = 10 for KMeans and k = 12 for EM. We can see that KMeans gives us almost the same cluster result as it did on non-reduced data. This is reasonable because as PCA preserve the only main direction, KMeans will line up with it in both reduced and non-reduced data. For EM, however, we get a completely different result. The cluster is no longer overlap since we now have only two dimensions. This result in a decrease in its V measure (the reason is the same as why KMeans has poor performance on V measure.)





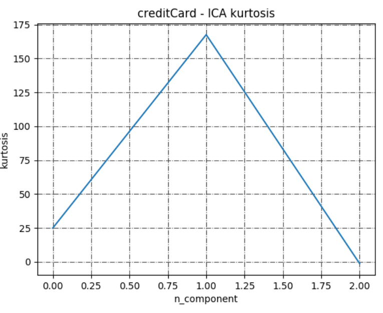


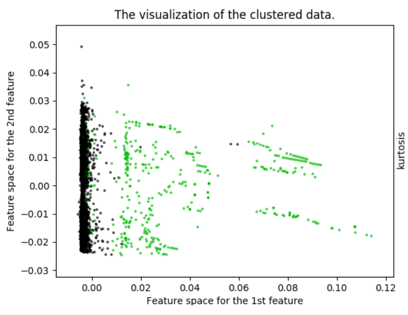


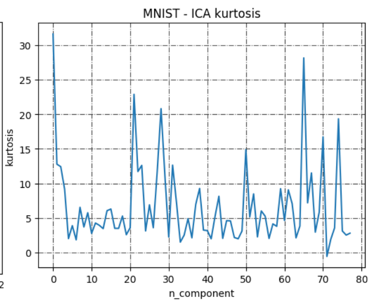
This is the result of MNIST. We pick k = 20 for KMeans. For EM, there is no significant elbow point in the likelihood figure, but we can still tell that the increasing trend starts to mitigate around 30. Therefore, we pick k = 30 for EM. Although the V Measure of both algorithms doesn't change a lot, we do get a slightly better Silhouette Coefficient (0.148 for KMeans and 0.163 for EM), indicating that we get a bettering defined cluster after PCA.

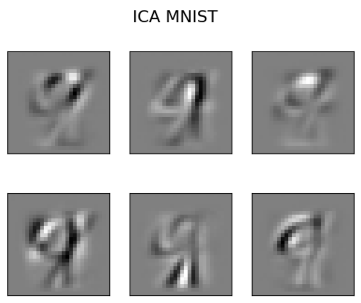
3.2 Independent Component Analysis (ICA)

Independent Component Analysis is similar to PCA, except the fact that it is trying to find the projected axes that are independent of each other while maximizing the mutual information with the original data. To evaluate how successful ICA does, one can determine how non-normal each component in the transformed data is (since each component should be mutually independent), and this can be done by analyzing how far the kurtosis of each component is from a kurtosis of 3 (normal distribution). Here we try several proportion of original dimensions ([0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1]) as n_components and analyze the kurtosis distribution. We end up finding that it is most appropriate to choose n_component = 3 for CCD and n_components = 78 for MNIST since it gives us the most non-normal distribution in the transformed components with fairly low reconstruction error. The following figure shows the result:



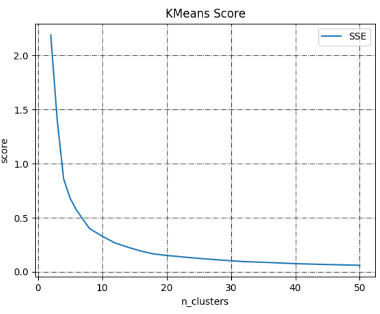


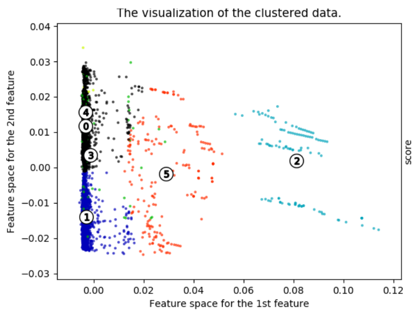


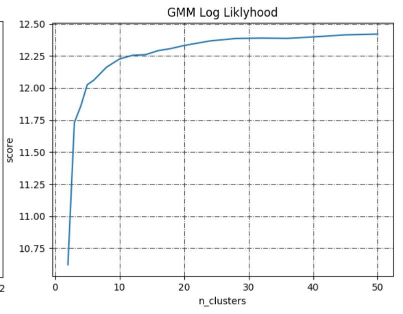


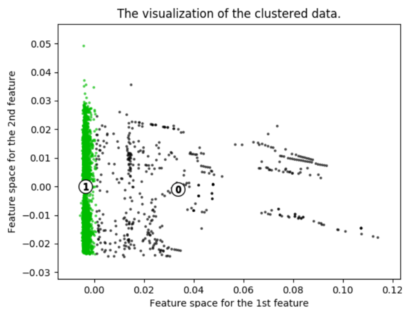
First, we can see that in both of the datasets, the kurtosis figures show that the distribution of the components is fairly non-normal. Furthermore, we can see that ICA does capture some semantic meaning of the data. In the CCD visualization, we can clearly see that the positive class is well separated from negative class. This is a significant improvement compared with the previous results. It also demonstrates the ability that ICA is able to capture the direction that has information for separate pos and neg class, which is an ability that PCA doesn't have. We also visualize the first 6 of ICA generated axes over MNIST, and we can see that it looks just like 4 and 9 digits, though a little blurry. This serves as another evidence that ICA can capture meaningful axes.

After discussing how ICA can capture meaningful component, we now discuss how it could affect our clustering result.

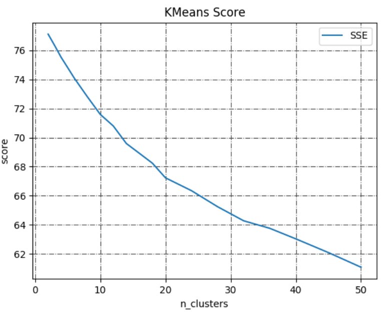


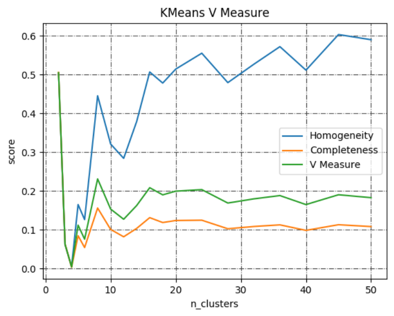


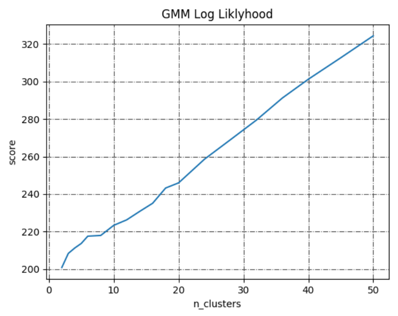


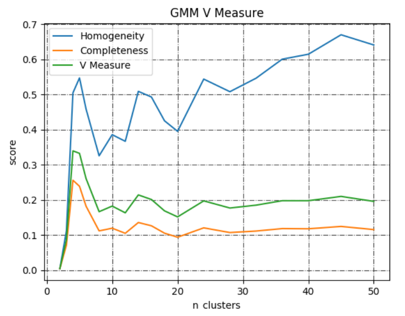


This is the result of CCD. We can determine pretty easy that the k value for KMeans is 6 to 8 using the elbow method, and we pick 6 in order to avoid overfitting. As for EM, though the elbow method suggests that we should pick a value around 6, we find that k = 2 can have a much better clustering result as the visualization shows. As a result, we pick 2. Both of the clusters align significantly better than previous results, especially for EM, which looks almost the same as ground truth label. That is all because ICA has already separated the ground truth well so that both algorithms can easily find clusters that line up with it.







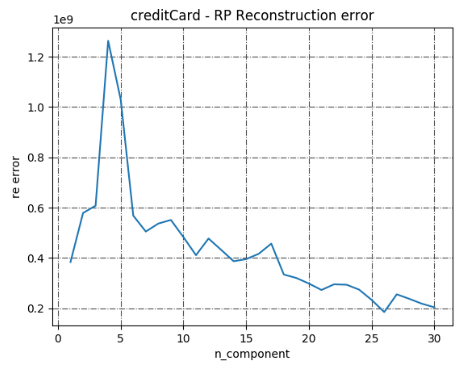


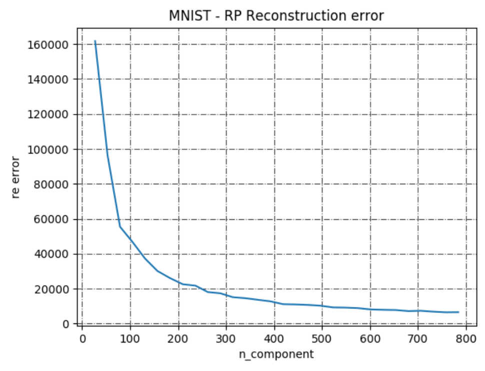
In the result of MNIST, both of the algorithms don't generate a clear elbow point. We then use V measure instead to pick k and find that k = 8 for KMeans and k = 5 for EM has the best V measure. Although we didn't show the visualization of clusters here since the dimensions of data is still too high, we can see there is an improvement in V measure, indicating that we get clustering results that are more closely aligned with ground truth.

3.3 Randomized Projection (RP)

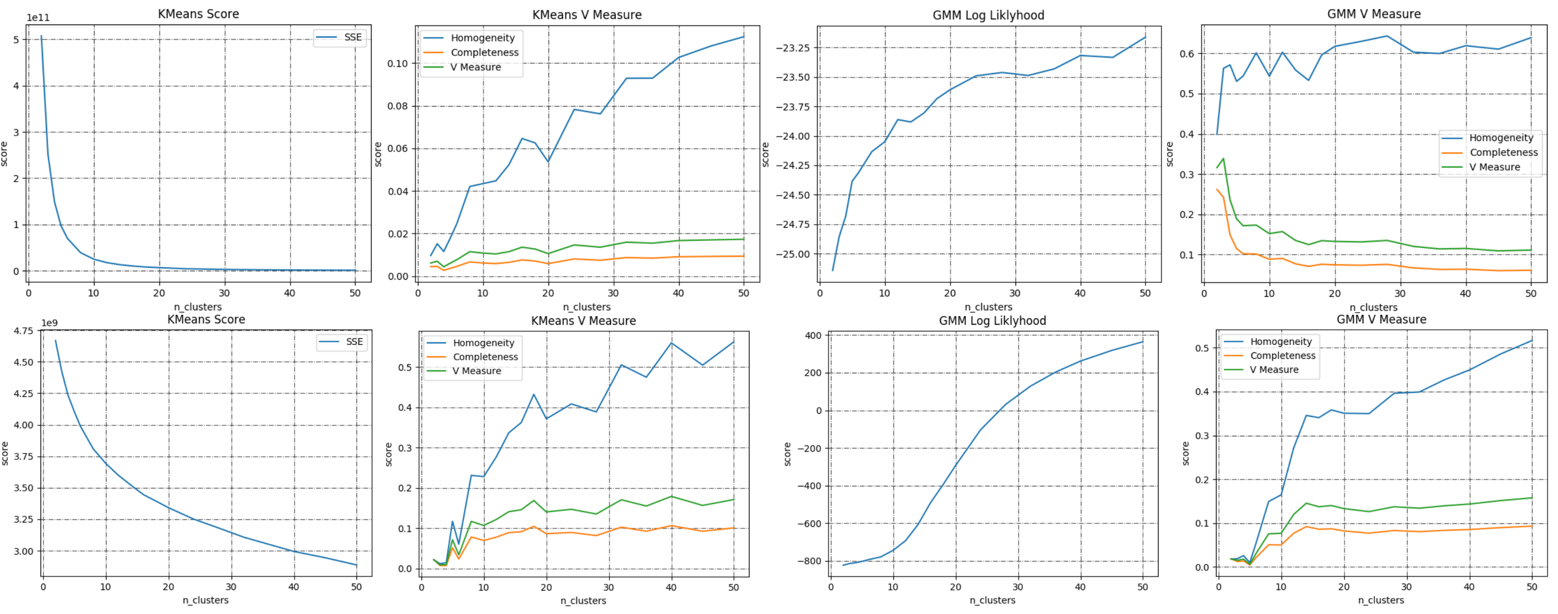
In contrast to PCA and ICA, which aim to find some good projections that either maximize the variance or maximize the mutual information while independent to each other, Randomized Projection (RP) simply project data to some random axes. RP, despite often being doubted at the first glance, turns out to work well in practice generally and is much faster than PCA and ICA.

How to decide whether the transformation generated by RP is good. We don't have a formal definition or criterion such as explained_variance_ratio_ or kurtosis that can help us to judge whether a transformation is good or not. Hence, we use a heuristic method of choosing the number of components that gets us a low enough reconstruction error and also reduce sufficient among of dimension. However, one should know that RP has a high variation between each run. For example, 10 trails of RP on MNIST will have a mean reconstructing error of 34636.38 and std 2960.49. Therefore, we will instead run sklearn RP module 10 times and use the projection with the lowest reconstruction error as our result every time we want to get a randomized projection.





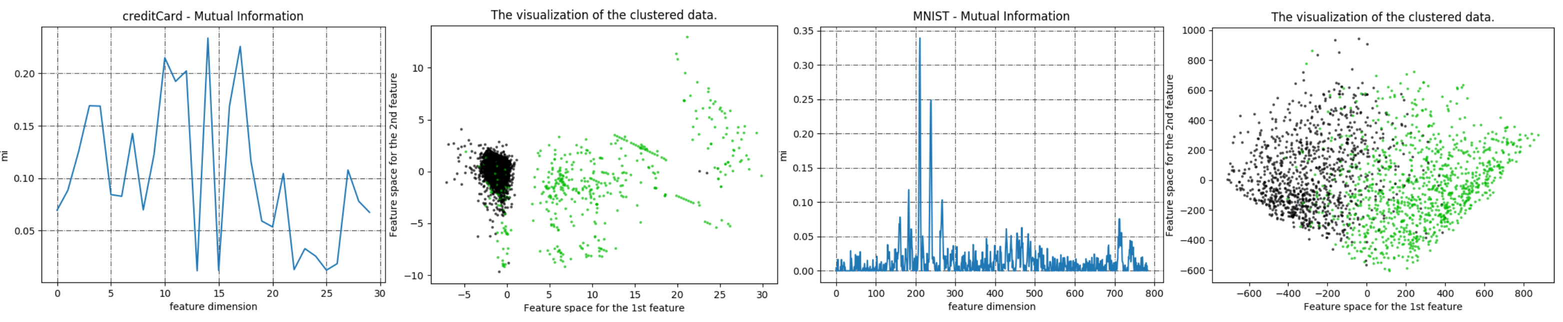
To determine the appropriate n_components for RP, we can start from the rightmost of the curve and move to the left to find a point that reconstruction error starts to increases significantly. We end up finding that n_components = 6 for CCD and n_components = 150 is the points we get. Therefore, we use these two value to reduce our datasets and apply clustering algorithms on the reduced data to see the results.



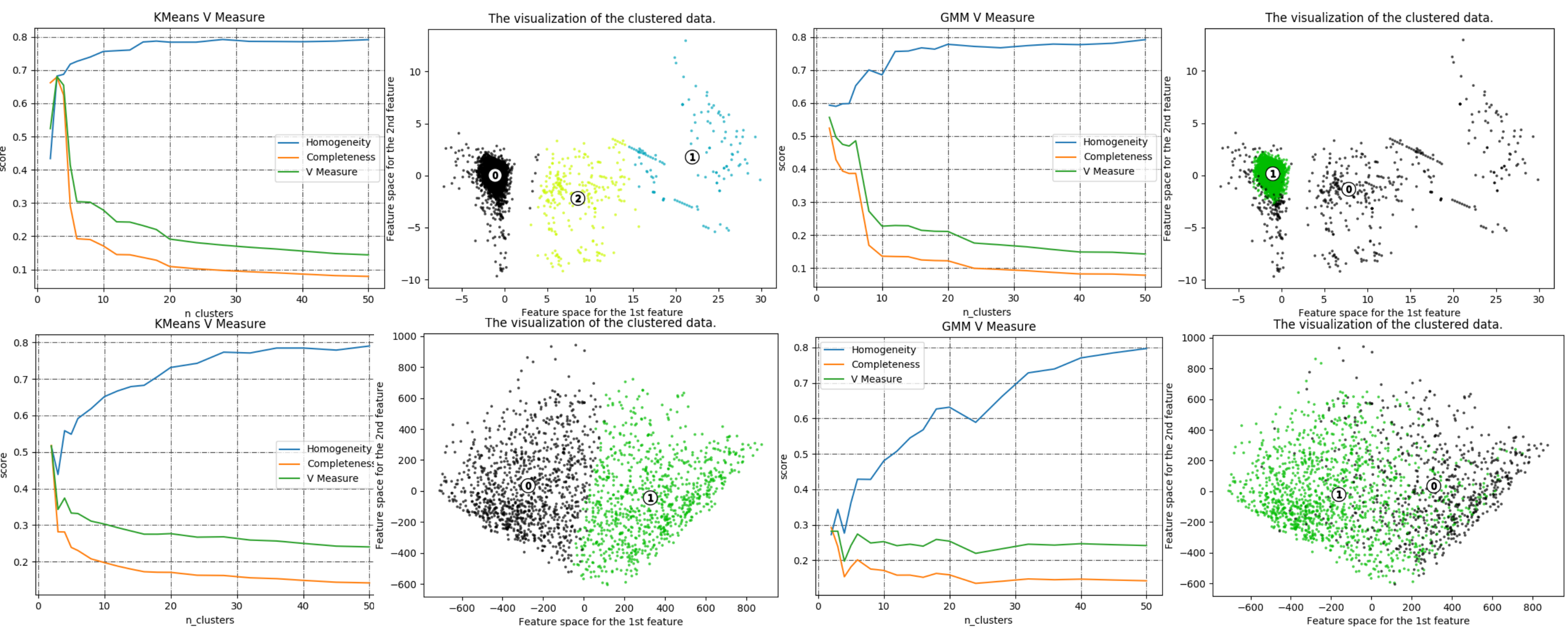
The first row results are for CCD, and the second is for MNIST, with left two figures for KMeans and right two for EM. Using the elbow method on the inertia and log-likelihood, we can again determine the appropriate k value for KMeans and EM. The value we pick is k = 10 for KMeans and k = 20 for EM in CCD, and k = 20 for KMeans and k = 30 for EM in MNIST. These k values and the V measure result is similar to the result of PCA. Unfortunately, since the data reduced by RP didn’t capture a meaningful projection space that can separate the ground truth label well, we didn’t get a good clustering result here. This situation certainly will happen because RP doesn’t care about which component it chooses. It just randomly generates one and does the projection. Therefore, it is almost impossible for it to find a meaningful component. When it comes to dimensionality reduction, it might be a good choice since it is fast and can achieve a similar result as PCA as we show above. However, if we want to retrieve meaningful hidden variable or projection space as ICA does. RP should not be the method we look for.

3.4 Mutual Information (MI)

Unlike PCA, ICA, or RP, which project the entire data to a new feature space, feature selection based on mutual Information will select only a few features from the original space with highest mutual information. Since this and most of the feature selection method will make use of the ground truth label, it is not purely unsupervised learning. However, since both of our dataset s have labels, we want to experiment how this kind of method will affect our clustering result. To determine how many features should be selected, we should first look at the mutual information of each feature with the label.



As we can see in the mutual information figure, only a few features have high mutual information (more clearly in MNIST), indicating that only those features contain predicting information. Here, we choose top 10% of the features from both datasets (3 for CCD and 79 for MNIST) since it turns out to separate high MI features from low MI features well. The visualizations are the ground truth label after applying MI. We can tell that MI has a similar ability to separate two overlapped classes in the original space as ICA. This is reasonable because MI explicitly selects features that have the highest mutual information to the label. As a result, the transformed space can describe the data and separate two classes better.



The first row results are for CCD, and the second is for MNIST, with left two figures for KMeans and right two for EM. Since the elbow plots in this experiment didn't show a clear elbow point, we instead choosing the number of clusters by looking at V Measure, picking a small number of clusters to avoid overfitting while maximizing V measure score. We end up with k = 3 for KMeans and k = 2 for EM in CCD, and k = 2 for both KMeans and EM in MNIST. Similar to ICA, both algorithms find the clusters that are well-aligned with the ground truth label, since the data is already well separated after transformed using MI.

Although MI has a similar ability as ICA in extracting meaningful feature space and separate the ground truth label, it still has shortcomings. The most important one is that we generally will not get the ground truth label when we trying to do unsupervised learning. This makes MI not much feasible in unsupervised learning.

3.5 Conclusion

In conclusion, ICA and MI are both good at choosing a meaningful feature space and separate the underly ground truth label. However, in order to conduct MI, one should be able to obtain the label first, which make it impractical in the field of unsupervised learning. PCA, as being a well-studied dimensionality reduction algorithm for many years, is fast to compute and easy to find the best reduced space for a given dataset, and it can give you the best reconstruction of the data. Nevertheless, as our result in CCD shows, it is not necessary for PCA to find a feature space that is meaningful since the data is extremely dominated by only one direction. Conducting normalization before applying PCA may help to improve the performance since it can make variance more even across each feature dimensions. In this way, the variance of the data will no longer follow only one direction. RP in our experiment is not considered to be a good transformation algorithm for both of the two datasets since it only transforms data into a random projection space and is lack of the ability to find a meaningful feature space that is needed to deal with our datasets.

The following table provides a summary of our finding in previous experiments:

	CCD	MNIST
Non reduced	KMeans k = 10, EM k = 12	KMeans k = 20, EM k = 20
PCA	KMeans k = 10, EM k = 12	KMeans k = 20, EM k = 30
ICA	KMeans k = 6, EM k = 2	KMeans k = 8, EM k = 5
RP	KMeans k = 10, EM k = 20	KMeans k = 20, EM k = 30
MI	KMeans k = 3, EM k = 2	KMeans k = 2, EM k = 2

We also provide the reconstruction error of each of the three feature transformation algorithms on both dataset with the parameter we find in the experiment:

	PCA	ICA	RP
CCD	3.245	1.370	1006119793.96
MNIST	196.982	311.733	31093.81

4 Neural Network

We now move on to analyze the clustering and dimensionality reduction with a neural network. We will conduct our experiment on MNIST dataset.

4.1 Neural Network with dimension reduced data

	training accuracy	testing accuracy	training time (sec)	testing time (sec)
Non reduced	99.56%	97.25%	21.653	0.0023
PCA	99.88%	97.00%	8.487	0.0012
ICA	**50.00%**	**50.00%**	**0.262**	**0.0011**
ICA * 10000	99.12%	93.25%	8.430	0.0012
RP	99.62%	97.75%	11.049	0.0014
MI	99.62%	92.00%	10.615	0.0013

You will notice that NN with ICA reduced data had a very bad result here. After diving into the reduced dataset, we find that all of the other reduced and non-reduced data have the same feature scale around 10^2, while ICA reduced data has only a scale around 10^-2, which cause the optimal NN structure stop working in this dataset. After simply multiplied ICA reduced dataset by 10000, we get a much more reasonable performance as shown in the table.

Overall, applying dimensionality reduction didn't change the accuracy very much. This makes sense because these dimensionality reduction algorithms are designed to preserve the information of the data as much as possible while reducing the feature space, in the same time trying to find a meaningful space and hope it will make data more likely to be separable. As a result, a good dimension reduced data will have an accuracy not much lower than the non-reduced one, but it is not necessary to increase the accuracy. However, as we can see that these algorithms do reduce a significant amount of training time since the input space is dramatically reduced. We will notice that ICA * 10000 and MI has slightly lower testing accuracy. This is because we do a more aggressive reduction (79 dim, while others have 100+.) This may cause them to lose more information when reducing the data compared with other algorithms.

4.2 Neural Network with Clustering Feature

	training accuracy	testing accuracy	V measure	training time (sec)	testing time (sec)
PCA + KMeans	50.00%	50.00%	0.138	0.6196	0.0010
PCA + EM	50.00%	50.00%	0.168	0.5958	0.0010
ICA + KMeans	69.56%	70.25%	0.164	3.2931	0.0010
ICA + EM	86.88%	87.25%	0.384	7.9691	0.0011
RP + KMeans	51.75%	47.50%	0.122	0.7554	0.0011
RP + EM	50.00%	50.00%	0.142	0.6260	0.0011
MI + KMeans	89.00%	89.75%	0.514	3.2809	0.0012
MI + EM	78.12%	76.25%	0.281	2.9561	0.0015

Before we run our neural network on these eight datasets, we need to do scaling like we have done in the previous subsection for ICA or we will get a completely nonsense result. In this experiment, we will instead use the clustering result as one dimension feature dataset to predict the ground truth label. In fact, this can be seen as a kind of measurement of how well the clusters lining up with the label. If the clusters are well-aligned with the label, we should get good accuracy. If not, the neural network will perform poorly since we just feed it a bunch of inconsistent data. We can validate this statement by comparing the result with the V measure of the cluster. As we can see in the table, the training and testing accuracy are almost consistent with V measure, with higher V measure get higher accuracy and vice versa. Moreover, we can also find that this result agrees with our analysis. That is, ICA and MI tend to find good clusters that closely aligned with the ground truth label, while PCA and RP don't. Therefore, it also adds some confidence to our previous experiment and analysis.

5 Reference

Sikit-Learn - Clustering - https://scikit-learn.org/stable/modules/clustering.html