UNSUPERVISED LEARNERS

Wan, Yuet Wing Frank  
ywan33@gatech.edu

***Abstract—***To understand the differences between various unsupervised learning algorithms, the k-means clustering and expectation maximization was tested for clustering, while PCA, ICA, KPCA, and Randomized Projections algorithms were tested for feature selection. These outputs were measured with performance metrics before getting trained on neural networks, before a final round of performance measurements on them. Two data sets were tested with the learning algorithms, before the results were finally compared against.

## Detail of Target Data

The load\_breast\_cancer dataset[[1]](#footnote-1) provided by scikit learn dataset and the openml mnist\_784 dataset ­was chosen as the two datasets to run the experiments on.

The cancer dataset was reused from the supervised learning project as it contains many distinct features for better feature refinement and is fully populated, thus not requiring the need for imputation. As the approach towards data imputation depends on the type of data and requires subject matter experts to weight in, having a fully populated dataset is much more convenient to perform analysis on. One caveat is that the dataset only contains 569 unique rows, which may impact the accuracy of the model. However, with the focus on feature selection rather than training neural networks, this caveat can be overlooked at this point.

The MNIST\_784 dataset was selected as well to increase the number of features and raw data with its 70,000 rows and 784 features, allowing increased effectiveness of dimension reduction. MNIST will be primarily focused on in this project, with the cancer dataset only used as comparison. However, there are 10 target classes instead of the 2 in the cancer dataset from 0-9, which will increase run time. This dataset describes images and has the features cleaned, size-normalized, and centered for clustering algorithms to run on.[[2]](#footnote-2)

To run analysis on the unsupervised clusters, a framework was copied from a similar project that also leveraged dimension reduction techniques before running unsupervised clusters on them.

## K-Means Clustering

K-Means clustering was first used to categorize the datasets into different groups using the clustering package and KMeansClustering library. For the K-Means cluster object, the n\_clusters parameter was set to 2. Due to performance limitations, initializing ten clusters in the unsupervised model proved to be too performance intensive for our local computer to compute in a short amount of time, so the cluster was set to 2. The KMeans algorithm was expected to find its own optimal number of unique classifications labels as it is unsupervised. This would later be graphed out and benchmarked with the true clusters for comparison’s sake.

The ‘init’ variable was set to 'k-means++' which selects initial cluster centers for k-mean clustering in an intelligent method to speed up convergence.[[3]](#footnote-3) Max\_iter was set to 1000 as an upper limit to number of runs for the training iterations, while the random\_state was set to guarantee reproducibility of results.

Afterwards object initialization, KMean’s fit\_predict() method was run on the training features to compute both the cluster centers and predict cluster indexes for each sample. This was selected as the return is stored onto the object cluster and benchmark variable for plotting the graphs later, which stores the true classifications of the data points. Afterwards, the results were printed out and plotted onto a scatterplot.

## K-Means Clustering analysis with PCA and TSNE

However, to plot the graphs out in a visually meaningful way, dimension reductionality needs to first be performed to reduce the dimensions down to 2 variables for the results to be plotted on a 2-dimensional plot. To achieve this, module PCA and TSNE from the library sklearn.decomposition and sklearn.manifold respectively were used to reduce the results into a 2-D array.[[4]](#footnote-4) These unsupervised methods of decomposition allow for retaining as much structure or information of the high-dimensional data in a low-dimensional representation.

A PCA object of a set rando\_state and n\_components of 2 had method fit\_transform() called to reduce the dimensionality of both datasets to 2. The PCA performs linear dimensionality reduction with Singular Value Decomposition of the data to project it to a lower dimensional space, transforming original data to new data by preserving the dataset’s variance using eigenvalues.[[5]](#footnote-5) This focuses on keep distinct points far apart in the resulting data space, but can be impacted by anomalies in the dataset.

Similarly, the T-distributed Stochastic Neighbor Embedding was also used to visualize such high dimensional data into a 2-dimensional plot. This method focuses on keeping similar data points close together using t-distribution to calculate statistically significant[[6]](#footnote-6) data points in the 2-dimenstional space, and as such are not impacted by data outliers.

The fit\_transform results of the PCA and TSNE reduction on the unsupervised training of MNIST data were then stored in a dataframe of columns tsne1, tsne2, pca1, and pca2 for scatterplot as shown below, with the benchmark true clusters stored in column y and the unsupervised clusters in column c.

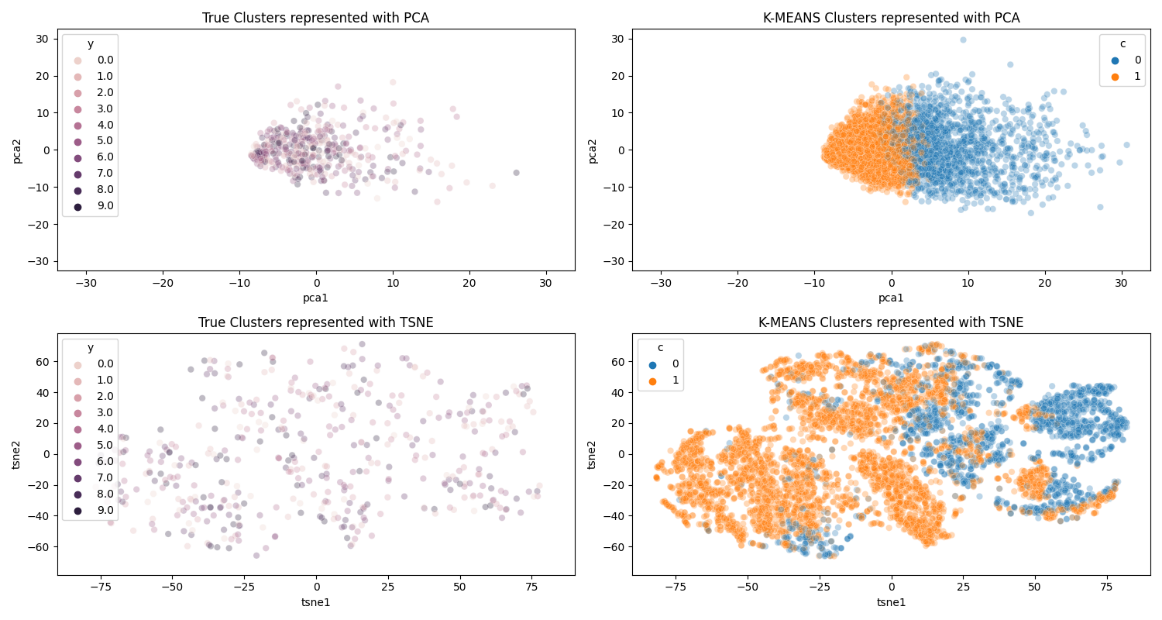


Figure 1 – Analysis on the MNIST dataset

In figure 1, the true clusters and unsupervised clusters are shown in different colors on the left and right subplots respectively. The unsupervised classified outputs on the right in orange and blue. The top and bottom graphs represent the different dimension reductional techniques PCA and TSNE respectively. Due to some technique difficulties in implementing the color coordinations in the subplots, the true cluster plots are not in different colors but rather in varying shades of violet.

Nevertheless, we can observe from top and bottom subplots of figure 1 are similar in shape as both are generated from the same dimension reduction formula, PCA and TSNE respectively. What is noticeably different is how the K-Means unsupervised learning groups the 2 clusters despite the MNIST dataset having 10 distinct labels.

## K-Means Clustering analysis with sklearn measurements

Train on training set with 10 clusters proved to be too computationally expensive and required too much time for a local computer to compute, as there is an exponential number of possibilities and optimizations needed for ten centroids to be calculated in a 569-dimensional space. As a result, no plots with 10 unsupervised clusters where generated, but different scores originating from the python package sklearn.metrics were leveraged to analyze the final results. These metrics are fed with the training column and labels in addition to the fit\_predict() classifications of the unsupervised models from 2 and 10 clusters.

Those metrics include the homogeneity, completeness, V measure, Rand, adjusted mutual info, and silhouette score. The first score reflects the homogeneity if all its clusters contain only data points from members of the same class, measuring the correctness of the point groupings.[[7]](#footnote-7) The completeness\_score reflects if all the data points assigned in the same class are members of the same cluster, while the v\_measure\_score is a balance between homogeneity and completeness.[[8]](#footnote-8) The rand score computes similarity by examining whether a pair of datapoints are in the same of different clusters in the true and predicted clusters, the adjusted mutual info measures the similarity of the labels between the true and calculated datasets, while the silhouette is calculated using the mean intra-cluster distance and the mean nearest-cluster distance.[[9]](#footnote-9)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Clusters | Homogeneity score | Completeness score | V measure score | Rand score | Adjusted mutual info | Silhouette |
| 2 | 0.086 | 0.326 | 0.136 | 0.056 | 0.136 | 0.129 |
| 10 | 0.415 | 0.439 | 0.427 | 0.310 | 0.425 | 0.010 |

Figure 2 – Metrics of performance from 2 and 10 clusters of unsupervised K-Means learning

From figure 2, 10 clusters of K-Means unsupervised learning unsurprisingly perform much better than 2 clusters, as the dataset originally has 10 labels. One interesting note is how the completeness score is similar to both datasets while other metrics vary much more. This is likely because the completeness score only calculates if the given data points in the calculated cluster are in the same cluster as the labels, so decreasing the number of clusters would not affect the score as more data points would be clustered together.

The silhouette is smaller for the 10 clusters compared to the 2 clusters as it is the measure of the distance between a sample and its nearest cluster that the sample is not part of. While a high silhouette may signify that the samples are well separated, we know from these two comparisons that the data sorely needs more clusters as the silhouette decreases ten-fold as the number of clusters increase by five times. Additionally, the low silhouette value in 10 clusters also suggests how the spatially compact the data points are as the data points are on average very close to the centroid of another cluster. This is highlighted by the right-sided plots in figure 1, where all the points are clustered close to each other. However, this may be difficult to confirm as the plots rely on PCA and TSNE reduction, while the measurements do not rely on such techniques.

## K-Means Clustering analysis with dimensionally reduced datasets

One caveat to the above analysis is how inaccurate and computationally expensive it is to find all those clusters with such high degree of dimensionality and freedom. This causes long compute times and bad cluster formations from similar features. To combat this, different dimension reduction techniques including the Independent Components, Principal Components, Kernel Principal Components, and Random Projections were used on the MNIST dataset prior to K-Means clustering calculations again. The Random Projection package was selected from sklearn.random\_projection, while the other 3 were downloaded from the sklearn.decomposition library.

The performance metrics are listed as of below in figure 3.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Clusters and DR performed | Homogeneity score | Completeness score | V-meas score | Rand score | Adjusted mutual info | Silhouette |
| 2 | 0.086 | 0.326 | 0.136 | 0.056 | 0.136 | 0.129 |
| 10 | 0.415 | 0.439 | 0.427 | 0.310 | 0.425 | 0.010 |
| 2 (PCA) | 0.086 | 0.324 | 0.136 | 0.056 | 0.136 | 0.133 |
| 10 (PCA) | 0.423 | 0.447 | 0.435 | 0.317 | 0.433 | 0.015 |
| 2 (ICA) | 0.006 | 0.226 | 0.012 | 0.001 | 0.012 | 0.283 |
| 10 (ICA) | 0.166 | 0.264 | 0.203 | 0.096 | 0.201 | -0.052 |
| 2 (KPCA) | 0.079 | 0.304 | 0.126 | 0.050 | 0.125 | 0.058 |
| 10 (KPCA) | 0.450 | 0.458 | 0.454 | 0.315 | 0.453 | 0.071 |
| 2 (RP) | 0.091 | 0.349 | 0.144 | 0.056 | 0.144 | 0.136 |
| 10 (RP) | 0.420 | 0.447 | 0.433 | 0.320 | 0.432 | 0.003 |

Figure 3 –Performance metrics of unsupervised K-Means learning

From Figure 3, it is generally seen that a higher cluster produces a better performance score, as analyzed earlier. However, it may be more difficult to spot a trend in the dimension reduction techniques and understand which ones perform the best at a quick glance. This was resolved through examining the spread of the PCA and TSNE-reduced plots using the techniques described in figure 1.

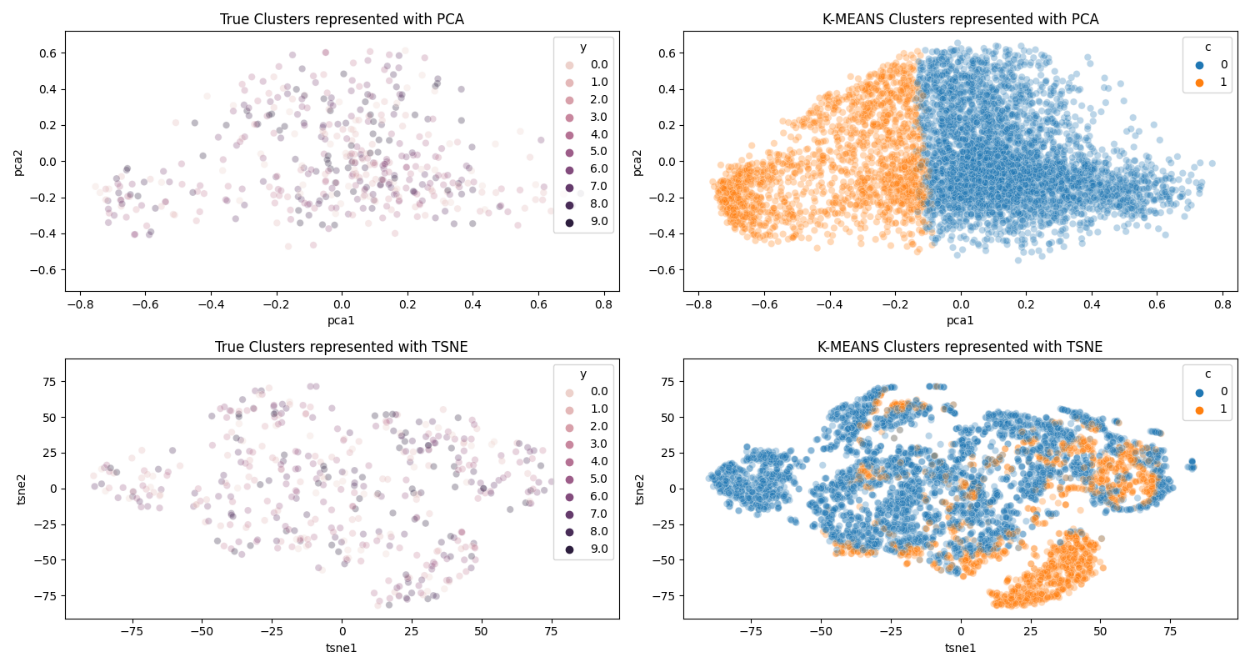
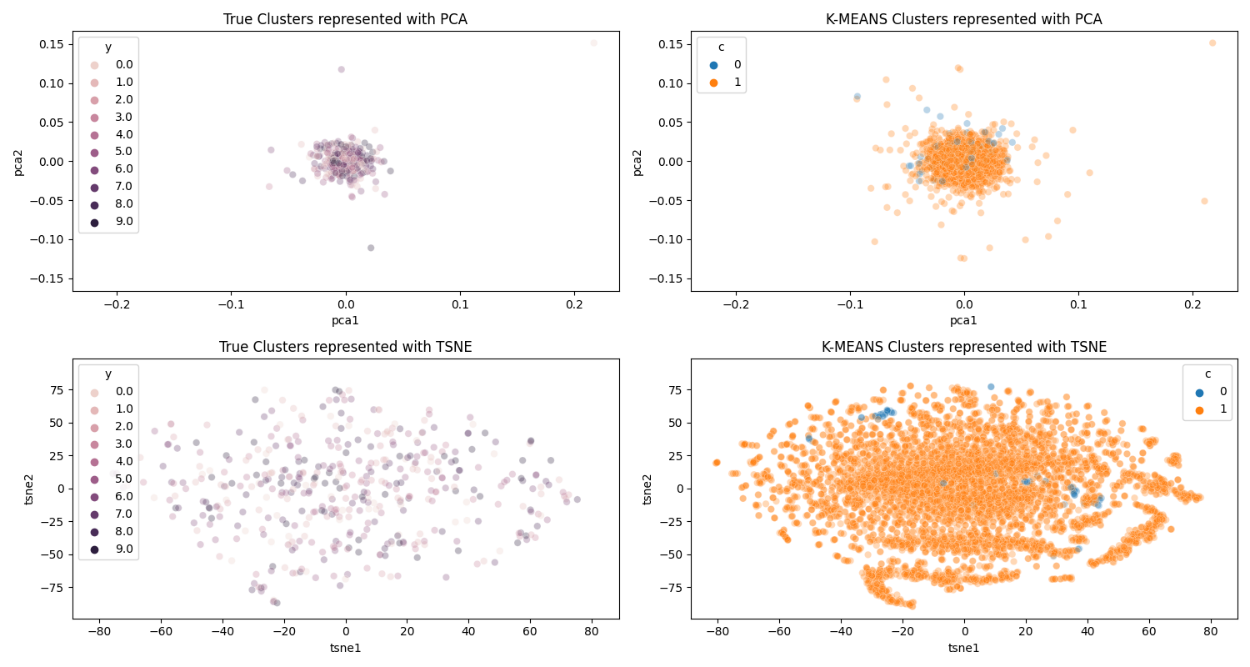


Figure 4 –Performance metrics of unsupervised K-Means learning after ICA reduction (left) and after KPCA reduction (right)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Clusters and DR performed | Homogeneity score | Completeness score | V-meas score | Rand score | Adjusted mutual info | Silhouette |
| 2 (ICA) | 0.006 | 0.226 | 0.012 | 0.001 | 0.012 | 0.283 |
| 10 (ICA) | 0.166 | 0.264 | 0.203 | 0.096 | 0.201 | -0.052 |
| 2 (KPCA) | 0.079 | 0.304 | 0.126 | 0.050 | 0.125 | 0.058 |
| 10 (KPCA) | 0.450 | 0.458 | 0.454 | 0.315 | 0.453 | 0.071 |

Figure 5 –Performance metrics of unsupervised K-Means learning after ICA reduction and after KPCA reduction

The theory is how the dimension reduction of PCA and TSNE on a plot can visually indicate which method of reduction performed the best in reducing the number of similar features in the dataset. From figure 4 it is apparent that the ICA reduction shown on the left is inferior to KPCA reduction shown on the right, as the top left plot shows how ICA reduction is unable to split out the different clusters meaningfully as the groups are intertwined with one another, and the data points are all clustered together. Meanwhile, the KPCA plot on the right is spread out nicely, indicating meaningful selection of distinct features in training, while the different clusters are separately neatly out into two halves.

Looking at figure 5, the performance discrepancy is highly apparent between ICA and KCPA, confirming our earlier suspicion, where the KPCA performance metrics are typically 2-10 times higher than that of the ICA reduction.

ICA performs dimension reduction through decomposing multivariate signals into independent non-Gaussian signals and attempts to locate the single source of signal that causes changes in other features to minimize mutual information through the Fast ISA algorithm.[[10]](#footnote-10)

## 

From the left portion of figure 4, the kurtotic shape can be seen in the circular shape of the plots, with a high density of data points at the center with sparse data points around the circumference of the circle. However, the distributions of the different centroids are not clearly distinguishes and do not capture much meaningful boundaries between the two classes.

Meanwhile, the Kernel PCA in this method utilizes the radial basis function, which projects the data into a higher dimensional space, allowing for better splits and if the graph is circular in shape. This may be a reason why the PCA reduction in the top left of figure 4 from ICA reduction represents a circular shape, while KPCA does not as it can project circular boundaries into a 3rd dimensional space, allowing for better separation.[[11]](#footnote-11) As such, the differences in reduction between the ICA tracking down signals causing variations in other features and KPCA abstracting features to a higher dimension has been highlighted by the performance metrics.

Due to report length limitations, other dimension reduction techniques are not examined.

## Expectation Maximization

The expectation-maximization aims to perform maximum likelihood estimation in the presence of variables and can estimate density of a dataset by searching across probability distributions and their parameters. It does this by first estimating the values for the missing variables, then optimizing the model to maximize parameters of the model, then repeating these two steps until convergence.[[12]](#footnote-12) In effect, this method attempts to find the true origin of a data point, like K-Means clusters, albeit with a different deduction approach.

The EM method was implemented with the sklearn.mixture library’s GaussianMixture module, with the clusters set to 2 for plotting and 10 for performance calculations again, and covariance set to ‘full’ as per default. The approach for fitting and predicting with the K-Means algorithm is replicated here as well, and produces the following graph:

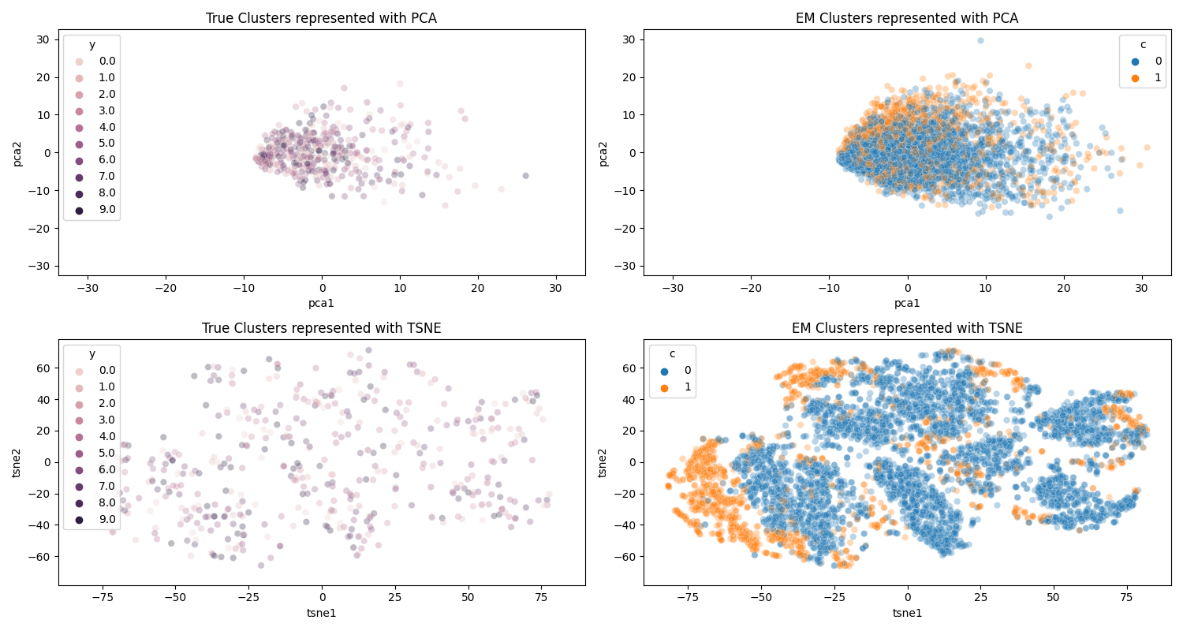


Figure 5 –Performance metrics of unsupervised EM learning

In figure 5, the EM method again does not sure much promise in distinguishing the clusters without prior dimension reduction and with only 2 clusters. The dimension reduction is performed in an identical fashion with the performance metrics and plots documented below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Clusters and DR performed | Homogeneity score | Completeness score | V-meas score | Rand score | Adjusted mutual info | Silhouette |
| 2 | 0.046 | 0.191 | 0.074 | 0.027 | 0.074 | 0.119 |
| 10 | 0.041 | 0.049 | 0.045 | 0.037 | 0.042 | -0.034 |
| 2 (PCA) | 0.034 | 0.154 | 0.056 | 0.016 | 0.055 | 0.209 |
| 10 (PCA) | 0.088 | 0.124 | 0.103 | 0.056 | 0.099 | 0.030 |
| 2 (ICA) | 0.022 | 0.115 | 0.037 | 0.009 | 0.036 | 0.269 |
| 10 (ICA) | 0.069 | 0.100 | 0.082 | 0.041 | 0.078 | 0.010 |
| 2 (KPCA) | 0.076 | 0.267 | 0.118 | 0.058 | 0.118 | 0.024 |
| 10 (KPCA) | 0.129 | 0.145 | 0.137 | 0.091 | 0.134 | 0.009 |
| 2 (RP) | 0.028 | 0.122 | 0.046 | 0.016 | 0.045 | 0.181 |
| 10 (RP) | 0.046 | 0.058 | 0.051 | 0.037 | 0.048 | -0.004 |

Figure 7 –Performance metrics of unsupervised EM learning

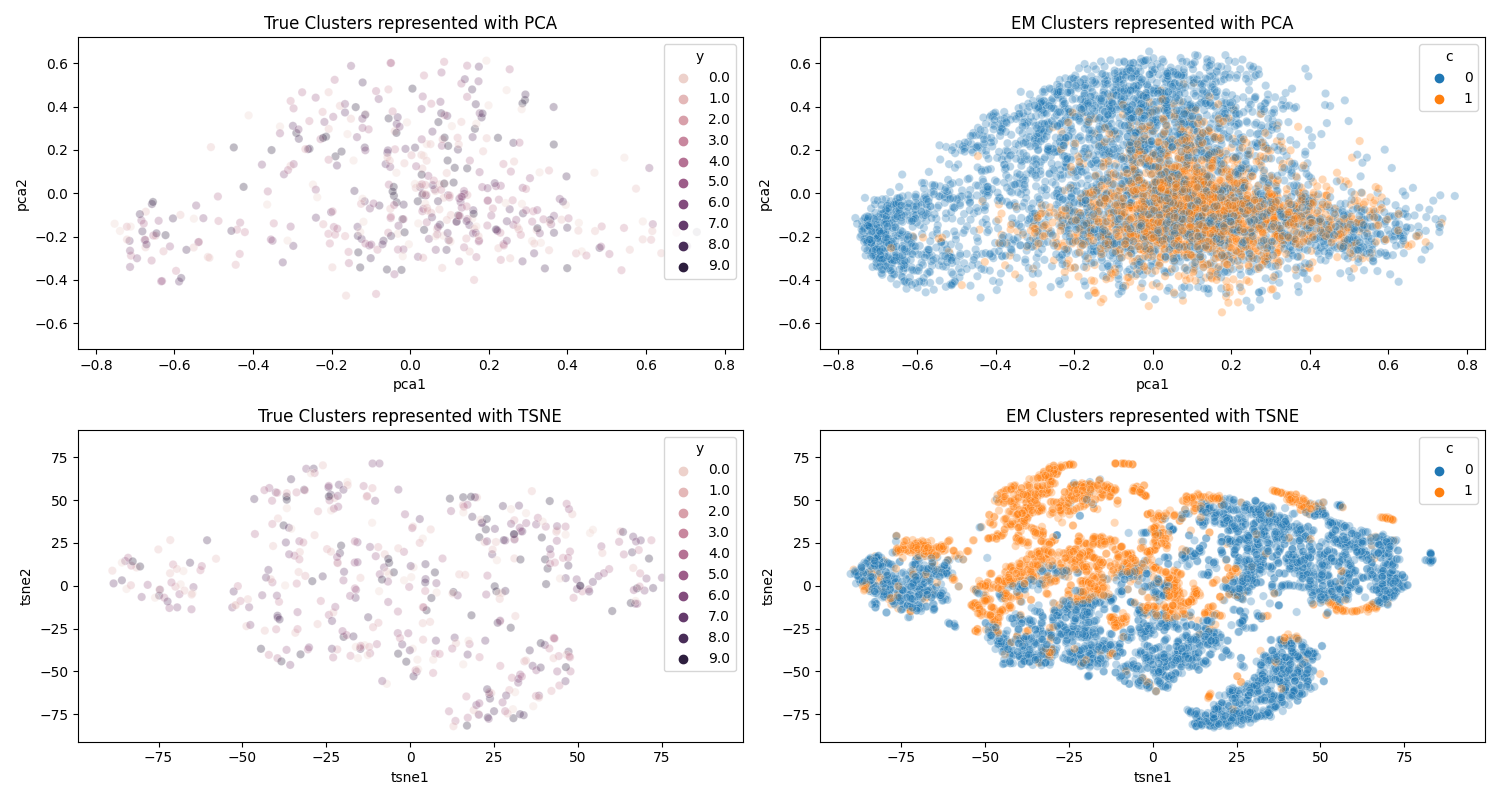
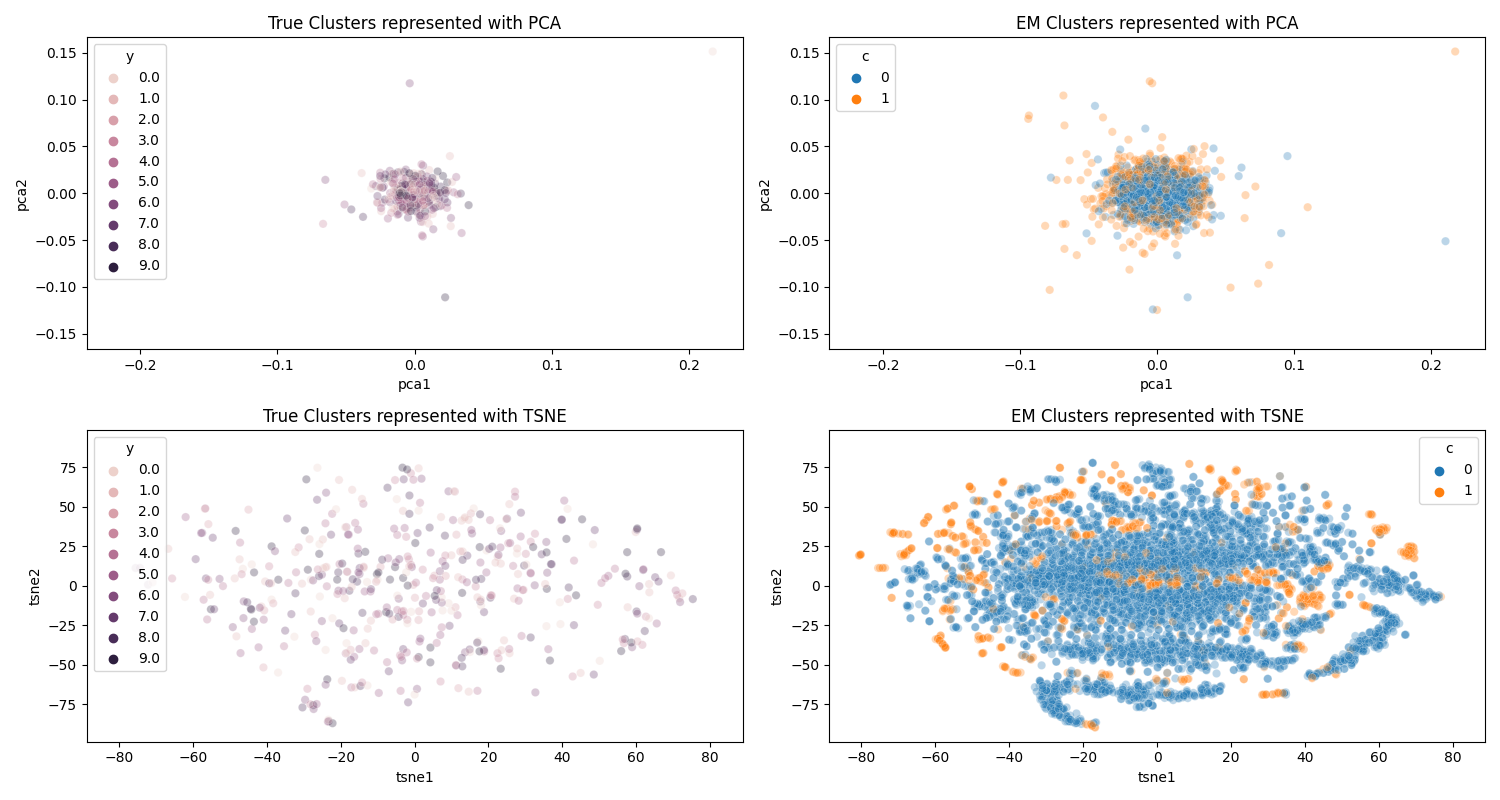


Figure 8 –Performance metrics of unsupervised EM learning after ICA reduction (left) and after KPCA reduction (right)

Once again, figure 8 illustrates the differences between the reduction power of ICA and KPCA. Comparing the red fonts in figure 7 to that of figure 5, we also see that the K-Means cluster outperforms the Expectation-Maximization method, possibly due to the simpleness of locating a centroid and while the EM method needs to calculate all possible center source of signal profile and calculate the likelihood of that being the source signal for that point.

## Neural Network

Afterwards, the neural network was performed on the reduced MNIST dataset. The dataset differed from the original assignment with a size of 784 columns and 70,000 rows. As such, the neural network instead used 150 nodes and 100 on the first and second layer respectively, as the size of the hidden layer should remain roughly half of the input size as a rule of thumb.[[13]](#footnote-13) The learning rate of 0.06 was also selected, as that value was the smallest value to peak in our earlier analysis, allowing for a balance between weight adjusting to feed forward signals and learning fast enough.

To calculate the performance of the neural network, package classification\_report and confusion\_matrix were imported from the library sklearn.metrics. As neural networks are a form of supervised learning, the correct labels inform the network on how to adjust the weights to correctly fit the target label.

The neural network was then run on a variety of transformations of the MNIST dataset, including the base MNIST dataset, dimensionally reduced datasets, and the results of K-Means and EM clustering algorithms. The K-Means and EM also outputs another layer of labelled clusters, which were used to train the neural networks as two additional cases. To save time, the K-Means and EM were clustered for 2 distinct clusters. The metrics were then calculated with the imported packages and the mean results in the table below, where precision is a measurement of true positives over true positives and false positives, recall is a measure of true positive and false negatives, and F1 is a measurement of precision and recall over itself.[[14]](#footnote-14)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Clusters and DR performed | precision | recall | f1-score | support | Fitting time (s) | Prediction time (s) |
| NN | 0.921 | 0.921 | 0.92 | 140 | 32.6824 | 0.0150 |
| PCA + NN | 0.93 | 0.928 | 0.93 | 140 | 14.6389 | 0.0239 |
| ICA + NN | **0.715** | **0.509** | **0.454** | **140** | **17.1124** | **0.0000** |
| KPCA + NN | 0.909 | 0.907 | 0.908 | 140 | 17.0526 | 0.0082 |
| RP+ NN | 0.932 | 0.93 | 0.932 | 140 | 17.4171 | 0.0091 |
| KMEANS+ NN | 0.923 | 0.923 | 0.924 | 140 | 27.2451 | 0.0110 |
| EM + NN | 0.922 | 0.922 | 0.92 | 140 | 24.7034 | 0.0100 |

Figure 9 –Performance metrics of neural networks with different modified datasets

One interesting observation in figure 9 is how the precision, recall, and f1-score of the ICA reduction are significantly lower than the other datasets. This is consistent with the previous analysis on the effect of the ICA reduction, and how it fails to effectively concentrate features while maximizing their variance of effect towards the outcome label, as seen in figure 4 and 8. This effectively lowered the ability of the trained neural network to properly distinguish the labels correctly, severely impacting its final performance.

However, we do see the effect of using dimension reduction on the datasets before training the neural network on it, mostly halving the fitting time. This is expected as the reduction in dataset dimensions reduces the load on the neural network tuning of the weights during feedforward propagation, as less inputs are factored and scored against the output label.

The prediction time remains mostly the same however, hovering around 0.1 seconds. This is expected as well as the neural network will perform the similar number of calculations for each new prediction, though the number of nodes involved may differ as some may not have hit the activation threshold during fitting.

## Conclusion

1. *Sklearn.datasets.load\_breast\_cancer*. scikit. (n.d.). Retrieved February 13, 2022 [↑](#footnote-ref-1)
2. https://www.openml.org/d/554 [↑](#footnote-ref-2)
3. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html [↑](#footnote-ref-3)
4. https://towardsdatascience.com/t-distributed-stochastic-neighbor-embedding-t-sne-bb60ff109561 [↑](#footnote-ref-4)
5. https://blog.clairvoyantsoft.com/eigen-decomposition-and-pca-c50f4ca15501 [↑](#footnote-ref-5)
6. https://en.wikipedia.org/wiki/Statistical\_significance [↑](#footnote-ref-6)
7. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity\_score.html [↑](#footnote-ref-7)
8. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v\_measure\_score.html [↑](#footnote-ref-8)
9. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette\_score.html [↑](#footnote-ref-9)
10. https://www.sciencedirect.com/science/article/abs/pii/S0893608000000265 [↑](#footnote-ref-10)
11. https://ieeexplore.ieee.org/document/9241766 [↑](#footnote-ref-11)
12. https://machinelearningmastery.com/expectation-maximization-em-algorithm [↑](#footnote-ref-12)
13. https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw [↑](#footnote-ref-13)
14. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 [↑](#footnote-ref-14)