CS7641 A3

Unsupervised Learning and Dimensionality Reduction

Wei En Chen

***Abstract* – Analysis of dimensional reduction and clustering algorithms. It is shown in the paper how PCA and ICA transforms the features, and they can be more time consuming than RP. It also shows how KMEANS and GMM differs in clustering. NN retrained with these algorithms show not much improvement in accuracies, but changes how a NN is trained, and further explained how these algorithms work.**

I. INTRODUCTION

In the realm of unsupervised learning, the collected data has no "y" labels, meaning there is no classification or regression. However, we can still analyze the data by grouping instances in likely hyper dimensional spaces, called clustering. It has a wide range of applications such as recommendation system, pattern recognition, and social network analysis.

Another useful algorithm we can do on unlabeled data is to reduce number of features by using dimensionality reduction (DR). Sometimes data with too many features may slow down machine learning algorithms and even decrease prediction accuracy [**Error! Reference source not found.**]. Also the data that suffers from the curse of dimensionality can benefit from DR. DR can also be used for visualization purpose. Visualizing data in hyper dimensions is not easy for humans. With some of the DR tools, data can be projected to 3D or 2D to understand data structure or distribution.

In this assignment, a number of clustering and DR algorithms are applied on the datasets from assignment 1. For the purpose of unsupervised learning, the "y" labels are hidden from the algorithms. As a recap on the datasets, first one is called "credit prediction" which has a mixture of categorical and continuous attributes, with relatively small number of samples compared to attributes. The second dataset is called "eeg eye state". It has only continuous attributes with relatively large number of samples compared to attributes.

II. CLUSTERING

K-MEANS

This is an algorithm that aims to partition samples into k (specified) number of groups. It starts by placing k cluster centres randomly in the data space, and each cluster centre will claim their closest data points. The grouped points are used to recalculate the cluster centres. The algorithm repeats until convergence. The distance metric calculation here is by default Euclidian distance in sklearn [2]. There is no other choice in sklearn, but it can be in different forms and may require some domain knowledge on the data. An important intuition from k-means is that the wider the inter-cluster distance the better, and the smaller the intra-cluster distance the better.

K-means relies on the properties of "distances" and "averages" among data points. With categorical data, these properties may not make much sense. When dealing with dataset 1 (with mixed categorical-continuous attributes), k-prototypes is used instead [3]. It replaces distances with dissimilarity measures on categorical attributes and replaces averages with modes. The rest, continuous attributes, are treated the same in k-means [4,5,6].

EXPECTATION MAXIMIZATION (EM)

Unlike k-means, which groups datapoints with 100% certainty, EM uses probability distributions to conduct clustering. To be more concrete, it iterates between E and M steps, where E is to find the likelihood datapoints belong to certain clusters, and M is to maximize the likelihood [7,8]. It can get stuck in local maxima depending on the starting random values [8], which is typical for clustering. It is important to conduct the test multiple times with different staring values.

One common algorithm under EM is Gaussian Mixture Model (GMM), which will be used in this assignment. Gaussian here means normal distributions are assumed to represent the data. If there is strong domain knowledge on the dataset that the distribution is not Gaussian, then a different algorithm should be used. For dataset 2, it is sufficient to assume Gaussian is the underlying distribution. For dataset 1, which contains a mixture of categorical and continuous attributes, is not a very good fit for GMM. Since there are still some continuous attributes and sklearn does not have many options, GMM is still used for dataset 1.

Both k-means/k-mode and EM have O(nk) complexity per iteration where n is number of samples and k is number of clusters, but there are finite combination of k-means of k^n, while EM has infinite combinations.

III. DIMENSIONALITY REDUCTION

PRINCIPAL COMPONENT ANALYSIS (PCA)

PCA is type of feature extraction method that finds the axis that produces the highest variance among given data, and with the rest of the axes being orthogonal to each other with descending variances. This is similar to calculating principal stresses with eigenvectors and eigenvalues in solid mechanics.

It is important to have zero mean and normalize the data to avoid likelihood of negative impact on the 1st principal component direction and variance [9]. By extension, PCA is not a suitable algorithm on categorical attributes. There are some choices, such as FAMD (Factor Analysis of Mixed Data) [10], apply one hot encoding and then PCA/Sparse PCA. The justification for Sparse PCA is that most of the elements in the one hot encoding matrix are zero.

By using PCA, it is possible to concentrate more relevant features in smaller numbers, therefore, compress the data without losing significant information. PCA can also be used for visualizing hyper dimensional data in low dimensions (2D or 3D) which is easier for humans to process. PCA has a time complexity of O(d^3n) [11], where d is dimensions and n is number of samples.

INDEPENDENT COMPONENT ANALYSIS (ICA)

Unlike PCA, ICA attempts to find statistically independent components from each other. It does not compress the data so the output space is the same as the input space. ICA is defined by minimization of mutual information for inter-clusters, and maximization of non-Gaussianity. They can be analyzed with KL-divergence and kurtosis, respectively [12].

Since ICA does not compress data, it may not be a good algorithm served as dimensionality reduction, if the underlying independent components are close to the original number of attributes. Time complexity of ICA (fastICA) is O(d^2nm) [] where m is number of iterations. This can be faster than PCA if convergence is found in smaller iterations.

RANDOMIZED PROJECTION (RP)

The main idea behind RP is to state that points in high dimensional space may be projected onto a lower dimensional space while preserving important information [13]. In this paper, Gaussian RP (GRP) is used for analysis. Since there is a great deal of randomness in GRP, multiple runs will be used to find average performance metrics. The time complexity of RP is O(dkn) where d is the original dimensions, k is the resulting dimensions, and n is number of samples [14]. This is faster compared to PCA and ICA.

FORWARD SELECTION (KBEST)

For the last dimensionality reduction algorithm, SelectKBest is implemented. This is the only algorithm that peeks at the ground truth "y" label to inform decision what features to keep. Essentially it uses ANOVA to calculate f-score (a measure of similarity of two variances) to inform how much each attribute affect the output labels. The algorithm then selects k best attributes (forward selection) specified by the user. The value of k can be decided by using the scores of attributes, and some measurement of similarity between the original and the reduced feature spaces. This is also the only algorithm that does not transform the feature space. Instead it simply selects the best features according to the variance metric [15,16].

IV. ANALYSES

K-MEANS (OR K-PROTOTYPE)

Dataset 1 has a mixture of categorical and continuous attributes so k-means is replaced with k-prototype which can handle mixture data. To decide on a best cluster k, the algorithm runs multiple times to calculate the mean and std with varying k's, and several metrics are used. The important criterion is the metrics can not use ground truth "y" labels.

Since there are categorical attributes involved, typical metrics to evaluate clusters are thrown out of the door. Left of Figure 1 shows the cost metric of k-prototype result. The cost is defined as the summation of all distances of data points to their cluster centroids. The distance and centroids, as mentioned earlier, are replaced with dissimilarity measures and modes, respectively. The lower the cost the less the disperse the data points are with respect to their clusters. To pick the "best" clustering, we can use the "elbow" method where the cost is low while the number of clusters is low. With this method, 8 or 9 clusters are the best values. In the right figure, the compute (wall clock) time is linearly proportional to the clusters initially and flattens out when clusters > 10. This is due to number of iterations (not included in the report) to convergence flattens out as well.

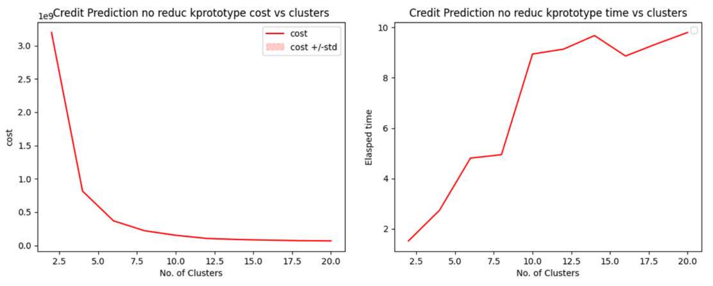

Figure 1 – Metrics on dataset 1 for k-prototype. Left: cost vs clusters. Right: wall clock time vs clusters.

To understand how well the result is, we can compare to the ground truth. Some of the plots are shown in Figure 2. Left is Rand index, a measure of similarity of a pair data points [17]. The higher the closer between predicted and ground truth labels. As we can observe, the highest score is when cluster is equal to 2, which is inline with our binary classification. At clusters = 10, the Rand index is already quite low. On the right figure, a curve of v score is presented. Similar to Rand index, higher score the better [17]. From the curve, previous decision of clusters = 8 is not the worst or the best, and the best is when clusters = 2, or possibly above 20.

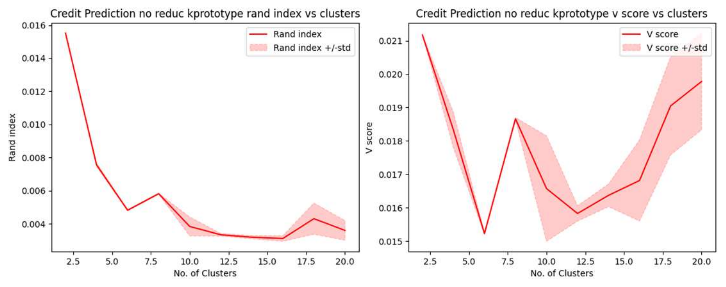

Figure 2 - Metrics involve know true labels on dataset 1 for k-prototype. Left: Rand index vs clusters. Right: V score vs clusters.

With these metrics not quite agreeing each other, I believe a major contribution is the nature of the dataset, which has a lot of categorical attributes, and the number of samples are low compared to the attributes/features.

On the other hand, dataset 2, with continuous attributes only, is analyzed with k-means clustering. The metrics to predict the best clusters are shown in Figure 3. The upper left is the silhouette. The silhouette score of one sample is computed by the following [17]:

$$\frac{b-a}{max(a,b)} \tag{1}$$

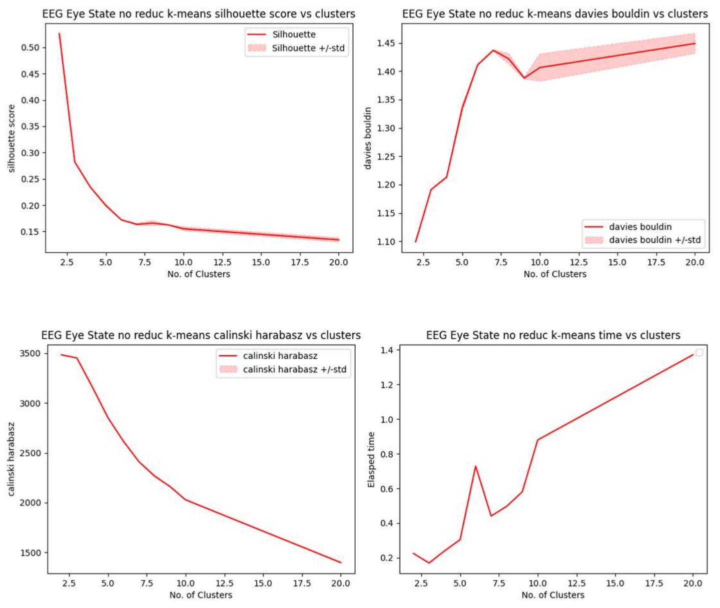

Figure 3 – Metrics on dataset 2 for k-means. TopLeft: silhouette score. TopRight: Davies Bouldin index. BottomLeft: Calinski Harabasz index. BottomRight: wall clock time vs clusters (same x-axis for all of them)

a = mean distance between sample and all other points in the same cluster, b = mean distance between sample and all other points in the closest neighbour cluster. Higher silhouette means the sample is farther away from neighbour clusters and hence better defined clusters. It is shown the best silhouette is when k is 2, which is close to our binary classification of 2.

Davies-Bouldin index is the ratio of cluster diameters and distance of cluster centroids [17]. The lower the value means better defined clusters. From the upper right plot, it agrees with the silhouette that clusters of 2 is the best.

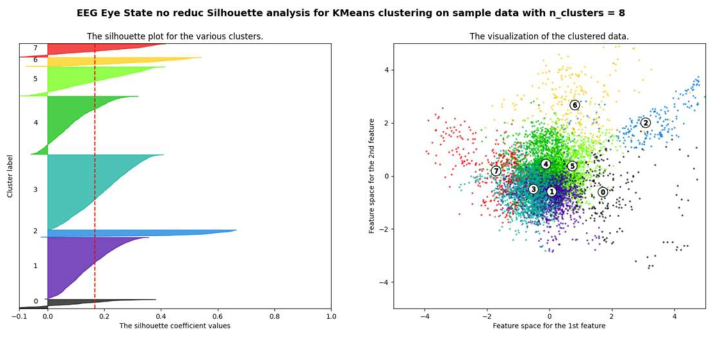

Figure 4 – Left: silhouette analysis on dataset 2 k-means clusters. Right: visualization of dataset 2 k-means clusters.

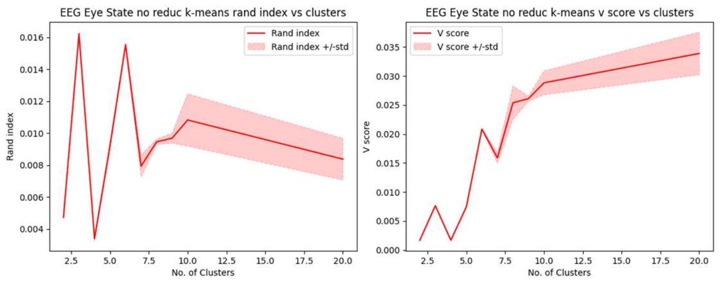

Figure 5 - Metrics involving true labels on dataset 2 for k-means. Left: Rand index vs clusters. Right: V score vs clusters.

Calinski-Harabasz index is defined as the ratio of inter-cluster spread and intra-cluster spread [17]. As mentioned earlier in k-means section with regards to inter/intra cluster distances, the higher the CH ratio the better. In bottom left plot, we see the max value at clusters = 2, and the score is not too bad at clusters = 4.

The last plot in the bottom right is the compute time. The time increases linearly with number of clusters. One big difference here is the time scale in k-prototype is a lot longer (about 10 times slower) to compute. There are a lot more samples in the second dataset, and the features in dataset one (no one hot encoding) are 20 compared to 14 features in dataset two. This means a lot of the calculations go to transforming categorical data into some quantifiable representation.

Clusters = 2 ~ 4 seems to be the best choice. However, it will result in very imbalanced clusters, which is not very helpful at all. Taking a deep look at silhouette analysis and we can pick a cluster value that results in each label occurrence is not too far from others (thicknesses of the colour blocks), while silhouette is high enough, shown in Figure 4 [18]. Cluster = 8 seems to be a good choice, and the right plot shows the data points based on 2 axes.

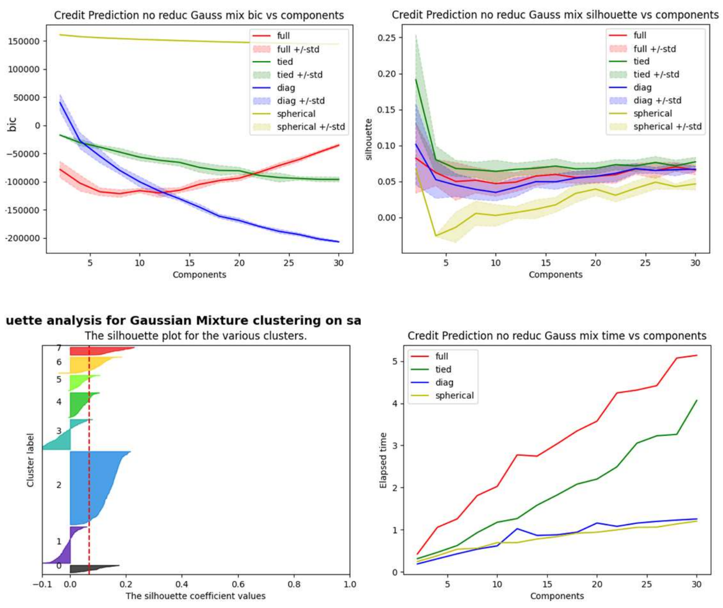

Figure 6 – Metrics on dataset 1 for GaussMix. TopLeft: BIC score vs components. TopRight: silhouette vs components. BottomLeft: silhouette analysis, clusters=8. BottomRight: wall clock time vs components. (components under GaussMix is interchangeably with clusters, and multi-curve plots are with various covariances)

To evaluate the choice of clusters = 8, ground truth labels are involved. Both Rand and V scores are shown in Figure 5. With clusters = 8, Rand index agrees it's not a bad fit, comparing to other clusters. For V score, it says the higher the clusters the better. From the comparison plots, we know the answer is not very cut and dry. At least from a qualitative method, a suitable cluster value should be around low double digits.

EM (GMM)

Dataset 1 with categorical portion is converted with one hot encoding for EM clustering because Gaussian requires "numerical" data.

Figure 6 shows some of the metrics without using ground truth labels. Upper left is Bayesian Information Criterion (BIC), is defined as BIC = -2 * log likelihood + d * log(N) which is an estimate how good the GMM is at clustering [2]. The lower the better because of negative log likelihood. From the full covariance curve, BIC is the lowest around clusters 8 – 12. From Upper right plot, silhouette, the best score is clusters = 2. Bottom left shows the silhouette analysis with clusters = 8. The labeling is very imbalanced with clusters = 2, which does not help. With 8 clusters, the silhouette is not the worst while each cluster has certain members.

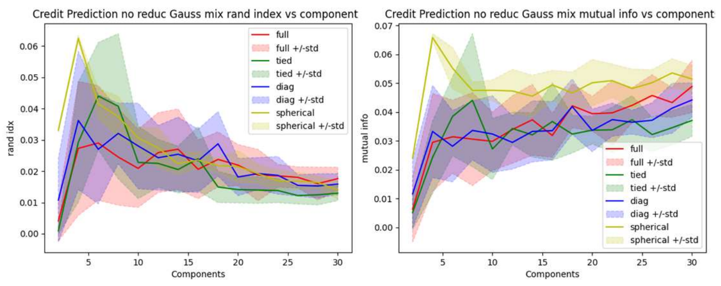

Figure 7 - Metrics involving true labels on dataset 1 with various covariances for GaussMix. Left: Rand index vs components. Right: adj mutual info vs components.

The metrics with ground truth labels in Figure 7 again shows mixed results. Our choice of 8 does not match the two metrics. It shows again here that the dataset is quite difficult to work with.

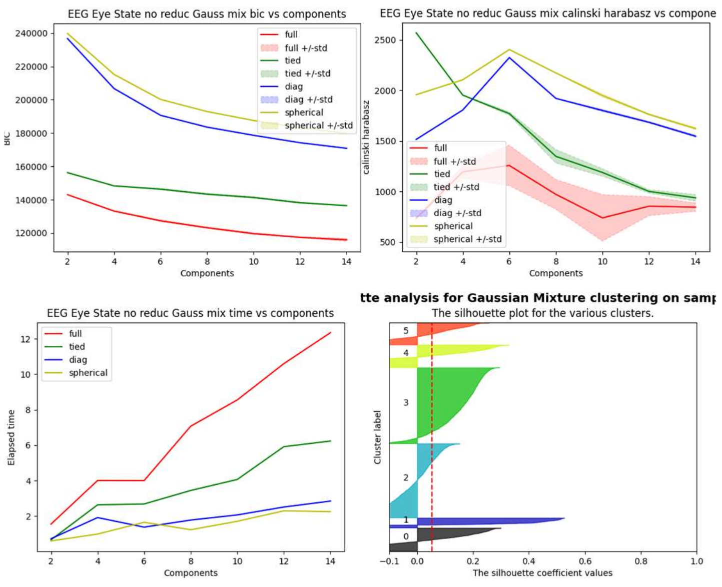

Figure 8 - Metrics on dataset 2 for GaussMix. TopLeft: BIC score vs components. TopRight: Calinski Harabasz vs components. BottomLeft: wall clock time vs components.(with various covariances) BottomRight: silhouette analysis, clusters=6.

Dataset 2 metrics in Figure 8 show if there is potentially good clusters. BIC plot says higher clusters the better, but another metric, CH plot says 6 is good. Silhouette with clusters = 6 is in the bottom right, which is more reasonable compared to other clusters. Compute time in bottom right shows it is proportional to the clusters. Note it is more time consuming than k-means, due to the computational demand on the probabilities. Full covariance is a exponential relationship since the matrix size increases with n^2.

Compare with metrics involving ground truth, shown in Figure 9. At clusters = 6, it does perform okay, which is not the best according to the two metrics.

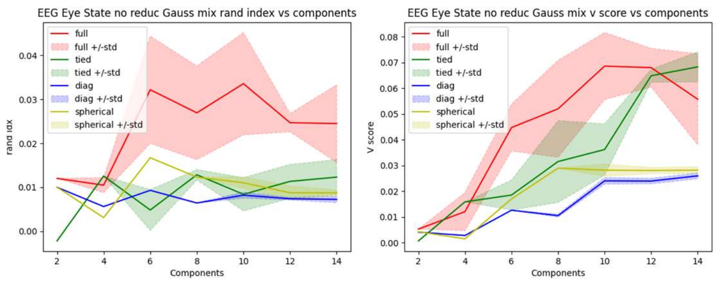

Figure 9 - Metrics involving true labels on dataset 2 with various covariances for GaussMix. Left: Rand index vs components. Right: V score vs components.

PCA

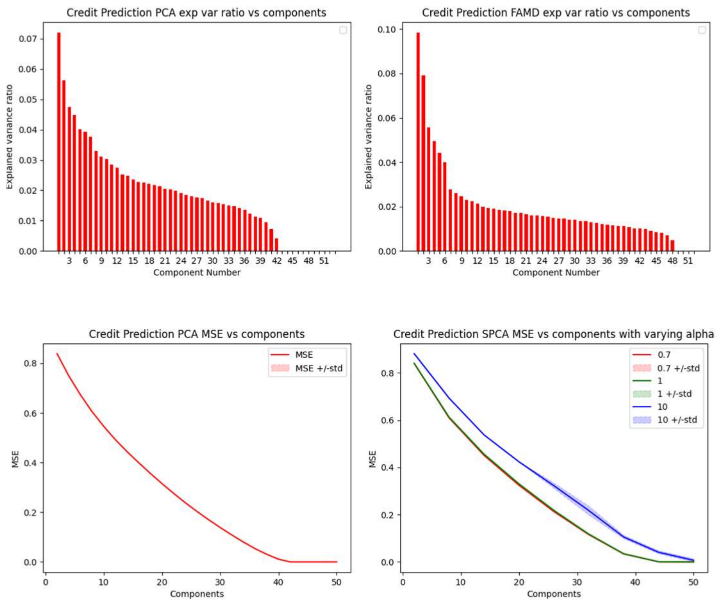

Figure 10 - Metric on dataset 1. TopLeft: PCA explained variance ratio vs components. TopRight: FAMD explained variance ratio vs components. BottomLeft: PCA reconstruction error vs components. BottomRight: SparsePCA reconstruction error with varying alphas vs components. (Here components mean principal components)

For dataset 1, I chose to compare PCA with one hot encoding, Sparse PCA with one hot encoding, and FAMD (which is used for mixed data). From Figure 10, top two plots we can see the explained variance ratios are similar. The difference may be because I substituted some categories with numeric values before one hot encoding. Based on the two plots, we need to pick 40+

components. The compute time plots are not shown, but they are constant time vs components. This is due to the algorithm calculates all components and then show the portions we specify. FAMD takes longer because of all the extra algorithm to convert categorical data.

Bottom left plot shows PCA reconstruction error using MSE. It echoes the explained variance that there are no strong components. Bottom right shows Sparse PCA reconstruction error with different alpha value used as a hyperparameter. It also shows the same trend as regular PCA. From the given results, the best components is 41.

For dataset 2, we can see a very compressed result in Figure 11. The first 3 components are strong in variance and it quickly drops to close to zero. The reconstruction error also supports the same result with a very clear hockey stick curve. If we want to maintain 90% to 95% of the explained variance, components should be 5 to 8, respectively, which is half of the original features.

This clear contrast in results between two datasets is likely due to categorical attributes vs continuous attributes.

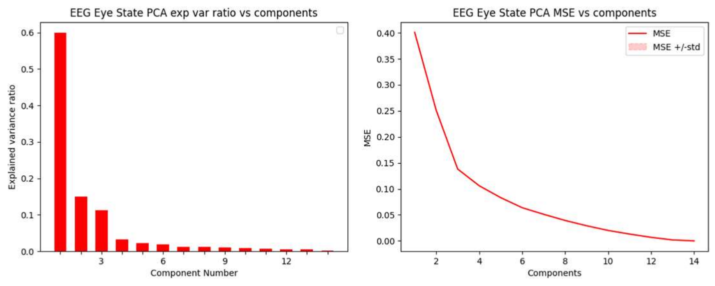

Figure 11 - Metrics on dataset 2 for PCA. Left: explained variance ratio vs components. Right: reconstruction error vs components.

ICA

Kurtosis (skewness) is a good metric to analyze ICA. Kurtosis of zero means a perfect normal distribution. High kurtosis which has thicker tails of the distribution, or more outliers, and is a measure of non-Gaussianity. For dataset 1 (one hot), Figure 12 top left shows the average kurtosis increases and max out around components=38. From the top right plot, reconstruction error, we see the minimum at 40, and sharply goes back to close to 1. This is due to ICA unable to converge with the given max iterations. Coupled with middle left plot, showing compute time swinging wildly. This suggest ICA (FastICA in sklearn [2]) is sensitive to initialization. The multiple runs indicates the wide standard deviations. We can pick the best components=37 to 40.

For dataset 2, we see a bit different story. The average kurtosis maxes out at component=1, in the middle right plot. The reconstruction error in bottom left plot is very similar to PCA. Although ICA does not really necessarily compress the dimensionality, it still does have that effect by observing the reconstruction error. The compute time, in bottom right, indicates there was also problem with convergence.

An interesting observation is when we plot the points on first two axes from PCA and ICA in Figure 13. We can see the variances are high with the first 2 components in PCA plot on the left. For ICA, it is clear that it extracts independent features in the first two components.

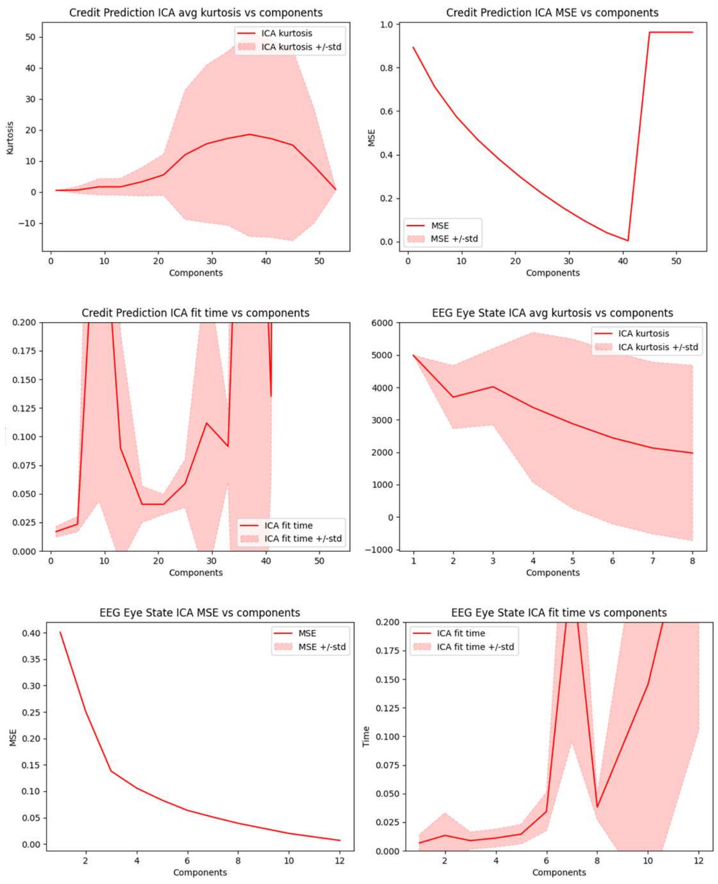

Figure 12 – Metrics for ICA. TopLeft: avg kurtosis on dataset 1. TopRight: reconstruction error on dataset 1. MiddleLeft: wall clock time on dataset 1. MiddleRight: avg kurtosis on dataset 2. BottomLeft: reconstruction error on dataset 2. BottomRight: wall clock time on dataset 2.

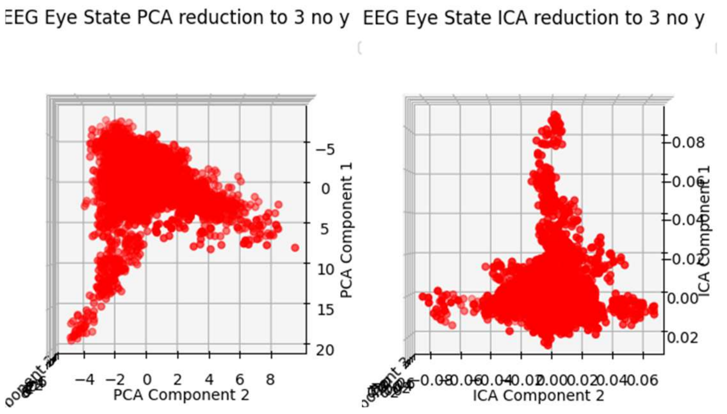

Figure 13 - Data point visualization after DR on dataset 2. Left: PCA. Right: ICA. Both reduced to 3 components and the views display components #1 and 2 in vertical and horizontal directions, respectively.

If we pick the best components based on reconstruction error, it will be 7 to 8 for 10% to 5%, respectively.

RP

A readily available metric for RP is the reconstruction error using MSE. In top left of Figure 14, it shows the construction error of dataset 1 (one hot). It is very typical for RP to get straight lines like this since it is not like PCA and ICA which concentrate more relevant features in fewer components. Same as dataset 2, in the top right plot. It is interesting that dataset 2 shows more variance in error plot than dataset 1. It is likely dataset 1 is a sparse matrix so randomness is limited.

In terms of wall clock time, the two bottom plots indicate it is constant time, which is not true. The number of components is too small to notice the difference. Time should linearly increase with the number of components. This is one reason RP is attractive due to its speed.

To pick a component to reduce, we simply use a threshold on reconstruction error. For dataset 1, components=48 to 51 for 10% to 5% error, respectively. For dataset 2, components=11 to 13 for 20% to 5%, respectively.

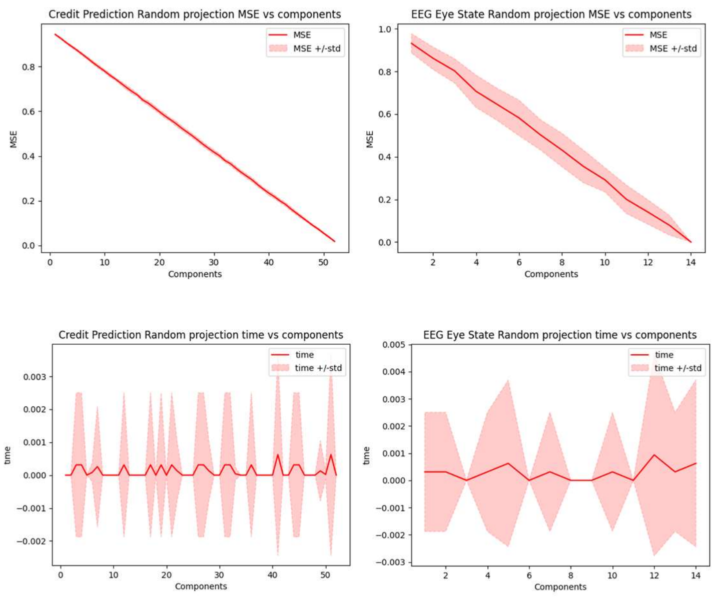

Figure 14 - Metrics for RP. Topleft: reconstruction error on dataset 1. TopRight: reconstruction error on dataset 2. BottomLeft: wall clock time on dataset 1. BottomRight: wall clock time on dataset 2.

KBEST

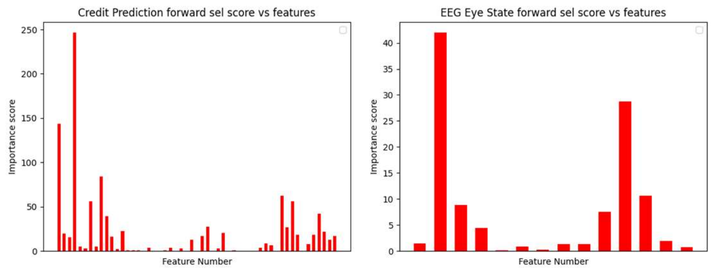

Figure 15 - Metrics for KBEST. Left: selection score vs features on dataset 1. Right: selection score vs features on dataset 2.

As mentioned before, this method uses ground truth labels to assist with component selection. For dataset 1 (one hot), left of Figure 15 shows the score with respect to each feature component. Right plot shows the score for dataset 2. Higher score means it has more impact on the y labels. To pick based on some percentage threshold, I normalized the scores and accumulate until said percentage threshold. In dataset 1, components = 18 to 22 for 90% to 95%, respectively. In dataset 2, components = 8 to 10 for 90% to 95%, respectively. From all four reduction methods, PCA is the most compact, RP is the fastest.

PCA + (K-MEANS AND GMM)

Both datasets are applied with PCA, and then k-means to see if there are changes to the clustering. For dataset 1, the results are noisy so the focus is on dataset 2. From Figure 16, we can compare with previous Figure 4 that when we pick components=5, the silhouette is slightly higher than before, the compute is also slightly faster. As for visualization of the clusters, we can definitely see better clustering with fewer components of 6 (used to be 8).

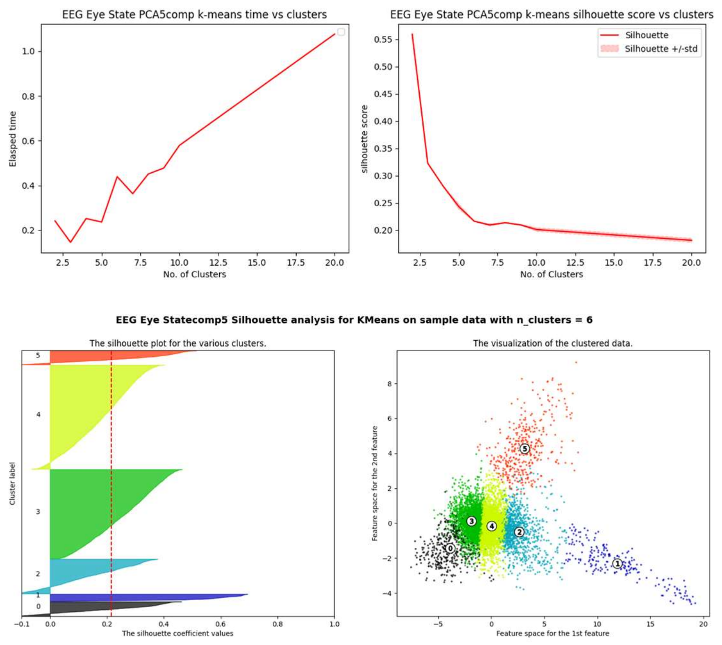

Figure 16 - Metrics on dataset 2 for PCA+KMEANS. TopLeft: wall clock time vs clusters. TopRight: silhouette vs clusters. BottomLeft: silhouette analysis for various clusters. BottomRight: clustering visualization.

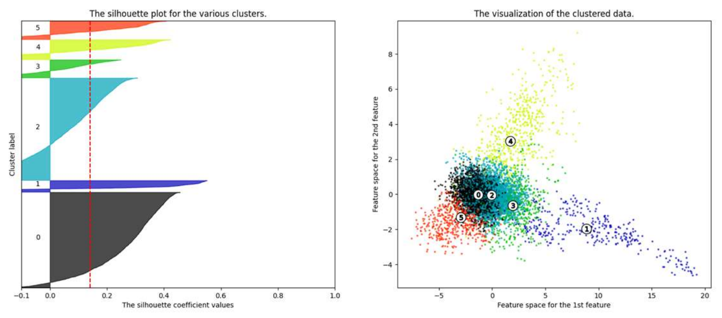

Figure 17 - Metric on dataset 2 for PCA+GMM. Left: silhouette analysis, clusters=6. Right: clustering visualization.

For PCA+GMM, we can also visualize much neater groupings of clusters and with fewer clusters in Figure 17. As a unique feature in GMM cluster, the cluster boundaries are "soft" so there are some mixed labels at the boundaries. The compute time, not shown here, but has decreased dramatically for "full" covariance curve because of the exponential relationship to number of features.

ICA + (K-MEANS AND GMM)

When we run ICA with k-means and GMM on both datasets, we can spot some improvements as well. For dataset 1, it is reduced to 37 features. When comparing the compute time on GMM, shown in left of Figure 18, it is about ~20% faster than in Figure 6. The silhouette score curves also slightly improve, shown in the right, comparing to Figure 6.

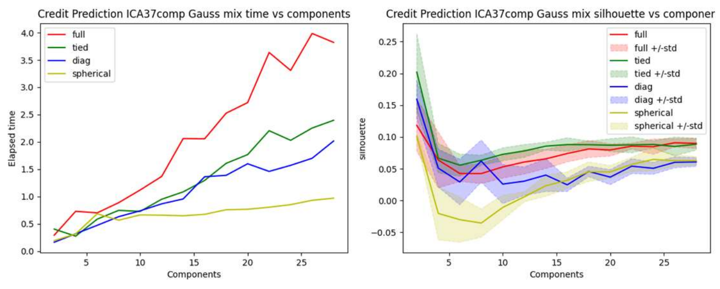

Figure 18 - Metrics on dataset 1 for ICA+GMM. Left: wall clock time vs components. Right: silhouette with various covariances vs components.

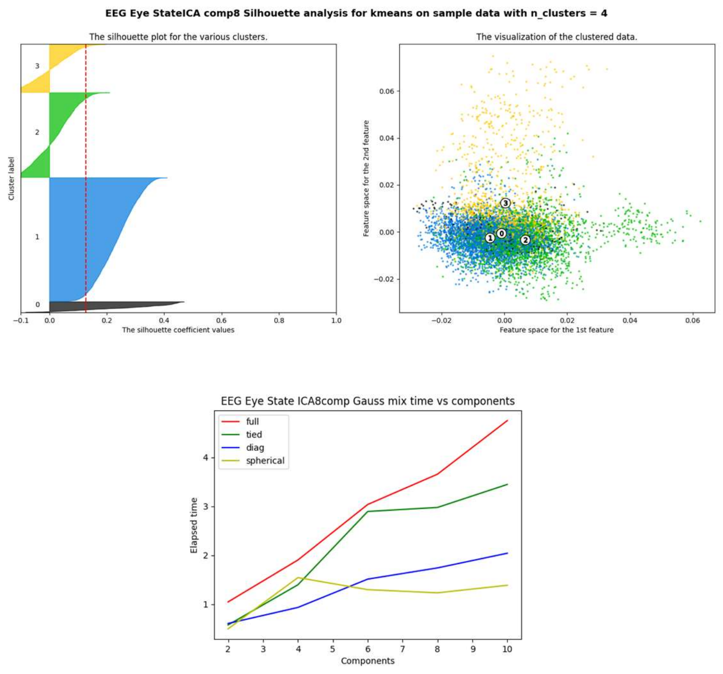

Figure 19 - Metrics on dataset 2. Top: Silhouette and cluster visualization for ICA+KMEANS, clusters=4. Bottom: wall clock time with various covariances for ICA+GMM.

With dataset 2 (features reduced from 14 to 8), when k-means is applied, it is observed that silhouette and the clusters improved, as shown in Figure 19. On the left it shows the silhouette is still relatively high while the clusters are only 4, compared to 8 before. Also on the right, it does not look as well defined as PCA+GMM. It is likely because PCA compresses the information more so plots in 2D would have more visually defined clusters compared to ICA. The bottom plot shows the compute time with GMM. Since features have dropped by ~40%, the compute time for full covariance has dropped by ~60%, comparing to Figure 8. An interesting comparison between PCA and ICA cluster data plots is that the orientations of the data cloud match Figure 13.

RP + (K-MEANS AND GMM)

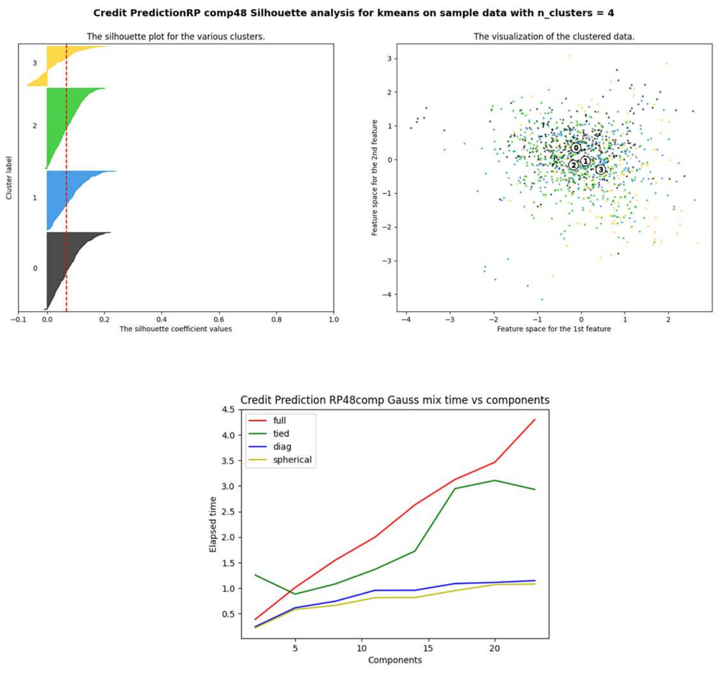

Figure 20 - Metrics on dataset 1. Top: silhouette and cluster visualization for RP+KMEANS, clusters=4. Bottom: wall clock time for RP+GMM.

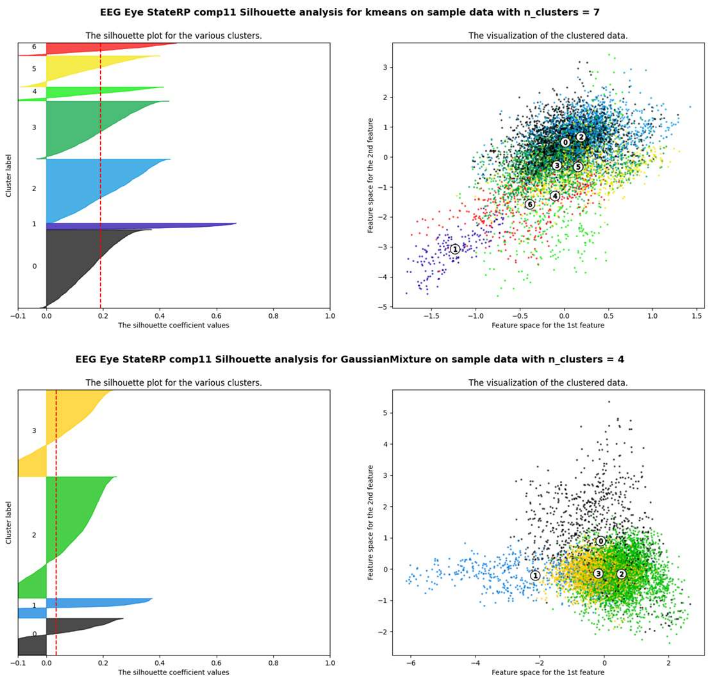

Figure 21 – silhouette and cluster visualization on dataset 2. Top: RP+KMEANS, clusters=7. Bottom: RP+GMM, clusters=4.

RP previously resulted in higher reduced dimensions because it does not compress information like PCA. We end up with dataset

1 components=48, and dataset 2 components=11. Surprisingly, in Figure 20 top plot, dataset 1 result with k-means shows clusters of 4 is not bad at all. The visualization of the clusters on the right isn't very clear likely due to the axes chosen. The bottom plot shows a minor improvement on compute time due to a minor reduction in dimensions.

For dataset 2, we can see again in Figure 21 even with reduction from RP, we can get better silhouette results with visually more defined clusters from either k-means or Gaussian mixture.

KBEST + (K-MEANS AND GMM)

From selected best components of 22 and 9 from datasets 1 and 2, respectively, we then apply k-means and GMM to see what happens. In Figure 22, top two plots show rand index and v score of the k-means method on dataset 1. Both suggest clusters of 6~8 are good choices. Then the middle left plot shows silhouette analysis of clusters=6, which is reasonable. Some clusters are wider than others but it's not bad. Middle right plot shows the silhouette analysis with GMM on dataset 1.

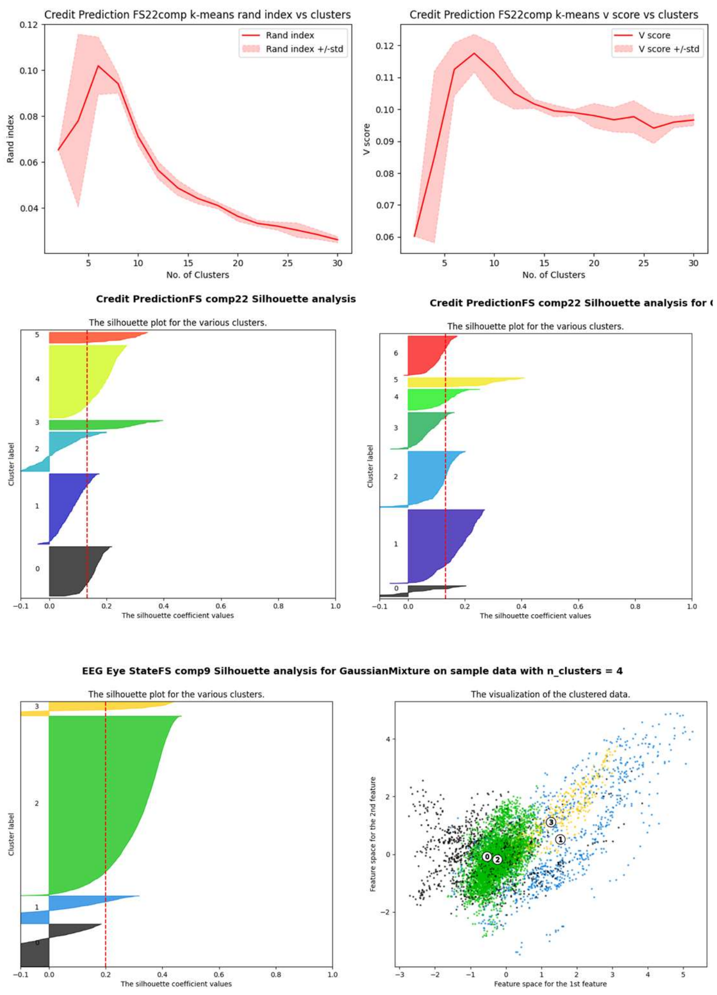

Figure 22 - Top: Rand idx and V score metrics on dataset 1 for KBEST+KMEANS. MiddleLeft: silhouette on dataset 1 for KBEST+KMEANS, clusters=6. MiddleRight: silhouette on dataset 1 for KBEST+GMM, clusters=7. Bottom: silhouette and cluster visualization on dataset 2 for KBEST+GMM, clusters=4.

On dataset 2 with GMM, silhouette analysis and the visualization of clusters are shown in the bottom of Figure 22. Surprisingly, it doesn't seem to do well visually, compared to k-means, which is not included in the report. The silhouette score drops very close to zero when clusters > 4. The clusters are also quite imbalanced. The clusters do seem to overlap more in the visualization. This makes sense because feature selection does not transform features in anyway. If there is no compression or separation of features, it's likely the visualization will not be appeasing. However, it does not suggest that it is not useful in terms of classification improvement, which is discussed in the next section.

V. RETRAIN NN

The last section is to retrain/retune the neural network from assignment 1 using dimensionality reduction methods and clustering algorithms. The selected dataset is number 2, EEG-eye state prediction. During the DR and clustering analysis of this dataset, it was discovered that there were 4 datapoints (out of 14k) had unreasonably high values in some variables even after they were normalized. These datapoints were about 100 to 200σ away from the mean which is statistically impossible with given dataset amount. They were treated as errors and consequently removed from the data for all analyses in this assignment, including retuning NN here. "Default" is designated for the retuned version from assignment 1.

Table 1 summarizes parameters to use with regards to the algorithms from the previous sections, namely number of components for dimensionality reduction, and number of clusters for clustering.

Table 1 - # of clusters (components) used in each algorithm before NN.

	PCA	ICA	RP	KBEST	KMEANS	GMM
# of components or clusters	8,11,14	8,11,14	8,11,14	9,12,14	6,8,10	6,8,10

They are essentially additional hyperparameters to tune, and the previous sections helped gain intuition and knowledge to narrow down the hyperparameter ranges.

First we will look at training wall clock time among the algorithms in Table 2. Note that wall clock time is not only affected by number of epochs, but also learning rate and batch size, which are hyperparameters. Smaller learning rate and batch would consequently increase train time. That being said, they could also imply the model is harder to learn because of some inherent data characteristics. With this knowledge, it is observed here PCA is faster when all components (14) are included compared to 8 & 11. PCA(14) does have tuned higher learning rate and batch, which means the NN model can be optimized to a decent local minimum at a faster pace. ICA takes way longer compared to others while the learning rates are not too far off. RP shares similar story to PCA, and KBEST behaves as what one would expect when more features are added, leading to bigger matrices, and eventually resulting in longer training time. These phenomena can be better explained when learning curves are included.

Table 2 - Training wall clock time for all 6 algorithms.

		Number of components		
		8	11	14
Train Wall Clock Time	Default			11.62
	PCA	10.55	13.19	5.73
	ICA	70.46	39.03	49.92
	RP	11.59	15.32	9.38
	KBEST*	5.01	6.70	11.76
		Number of clusters		
		6	8	10
Train Wall Clock Time	KMEANS	9.79	23.72	11.16
	GMM	11.76	21.85	45.95

Note: * KBEST has slightly different components (9,12,14), but they are close enough for qualitative comparison.

Figure 23 shows PCA learning curves vs epochs. At PCA(8), we have lower bias and high variance. Through PCA(11) to (14), the variance decreases and bias increases. Bias increases as more variables are made available for the model to learn, which makes sense. The overfitting phenomenon can likely be explained by looking at what PCA does. PCA assumes principal components are a linear combo of original features and they are calculated based on the input data only (unsupervised), which means they may not be closely correlated with the output labels. PCA also assumes all principal components are orthogonal. If these don't hold, overfitting will behave in the same way as simple feature selection. With PCA(14), all components ordered by their explained variances are present, which can help the optimizer find the best gradient to descent, and leads to higher learning rate and shorter train time.

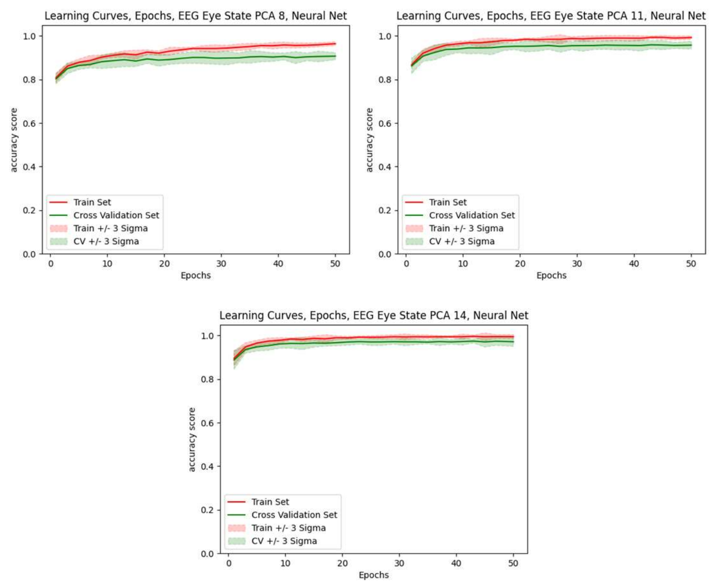

Figure 23 - Learning curves vs epochs for PCA. TopLeft: PCA(8). TopRight: PCA(11). Bottom: PCA(14).

As for ICA, in Figure 24, plots show the bias increases slower vs epochs, and the variance stayed low between train and validation. The slow bias movement explained the training time. For the reason why they take longer is likely when independent features are separated, the optimizer can have more possible directions to conduct a descent step, and also the gradient magnitudes can be low, meaning most independent features can play a role in prediction, but none of them is the critical one. This then also explains the low variance (low overfit). For example, if a NN model takes a photo of a person's face, and it would predict if it's a male or female by only using mouths and eyes. The model can likely make some correct prediction, but it's likely it won't overfit the data with limited independent features.

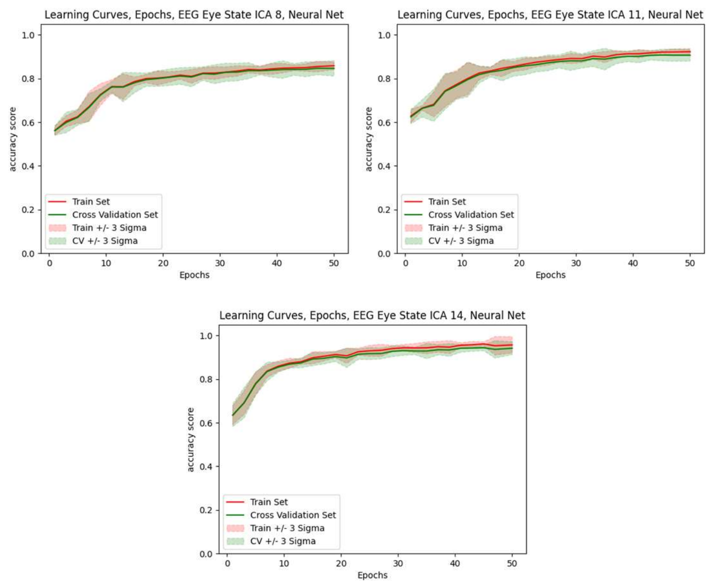

Figure 24 - Learning curves vs epochs for ICA. TopLeft: ICA(8). TopRight: ICA(11). Bottom: ICA(14).

RP randomly projects high dimensional space data into low dimensions with random matrices. It shows similar results in terms of bias-variance and train time behaviours, but since RP does not compress data information, it would generally take the optimizer longer to find a good local minimum. Its only advantage here is it is fast to transform the input data, if time in this step is critically important.

For KMEANS and GMM, it is better to include their train/validation accuracies in the picture for discussion, shown in Table 3. All the training or validation accuracies are not noticeably impacted by the number of clusters. There is a minor decrease in validation accuracies with clusters=10. With clusters=6 & 8, the validation accuracies are very close to the default (retuned). This implies the cluster labels do correlate with the ground truth labels. However, at clusters=10, we started seeing some minor drop in accuracies, meaning the clusters transition into more of a noise instead of signal. For wall clock time, the noticeable trend is time increases as clusters increase for GMM. When digging deeper, the batch size decreases with the increase of clusters. This makes sense as the data contains more noise, it may be more difficult to use the same batch size to conduct gradient descent. By having a smaller batch size, the descent step is noisier and can jump out of undesirable local minima.

Table 4 summarizes the test scores of all dimensional reduction and clustering algorithms. We can see that although PCA with reduced dimensions hurt the test accuracies, PCA at full dimensions increased test accuracy a little. The increase could be simply randomness, but at worst it does not hurt to have PCA with full dimensions because it decreases training time. If we care more about striking a balance between train time and accuracy, KBEST is the way to go. It does not require feature transformation, which means the selection process is fast, and the training time is also quite fast. The test scores under clustering show KMEANS tends to do a little better than GMM. If we recall the cluster visualization plots earlier, KMEANS have clearer defined cluster colour boundaries due to their deterministic nature, while GMM have more blurred, overlapping colour boundaries due to probabilistic nature. Note the "boundaries" here is not meant to say the clusters are more separated by a distance metric, but only to represent the colour boundaries between two clusters. The deterministic nature in KMEANS may have helped in the test accuracies. It can be beneficial to add cluster feature to any of the dimensional reduction algorithms to push the last mile, which is not included in the paper.

Table 3 - Training and validation accuracies for all algorithms.

		Number of components		
		8	11	14
Default	Train Acc			0.9883
	Valid Acc			0.9699
PCA	Train Acc	0.9421	0.9861	0.9898
	Valid Acc	0.9038	0.9568	0.9702
ICA	Train Acc	0.8724	0.9410	0.9610
	Valid Acc	0.8611	0.9260	0.9451
RP	Train Acc	0.9298	0.9796	0.9847
	Valid Acc	0.8897	0.9468	0.9568
KBEST*	Train Acc	0.9571	0.9787	0.9883
	Valid Acc	0.9236	0.9481	0.9699
		Number of clusters		
		6	8	10
KMEANS	Train Acc	0.9878	0.9886	0.9863
	Valid Acc	0.9658	0.9651	0.9649
GMM	Train Acc	0.9891	0.9891	0.9860
	Valid Acc	0.9682	0.9673	0.9639

Note: * KBEST has slightly different components (9,12,14), but they are close enough for qualitative comparison.

Table 4 - Test accuracies for all algorithms.

		Number of components		
		8	11	14
Test Acc	Default			0.9563
	PCA	0.8965	0.9476	0.9736
	ICA	0.8571	0.9089	0.9379
	RP	0.8935	0.9409	0.9553
	KBEST*	0.9196	0.9433	0.9563
		Number of clusters		
		6	8	10
Test Acc	KMEANS	0.9680	0.9700	0.9586
	GMM	0.9693	0.9613	0.9616

Note: * KBEST has slightly different components (9,12,14), but they are close enough for qualitative comparison.

VI. CONCLUSION

It is demonstrated in this assignment the usefulness of dimensionality reduction and clustering algorithms. Each has its own strength and weakness, depending on the need, purpose, and criteria of the ML pipeline. The paper also demonstrates that these algorithms are difficult to interpret, especially on categorical data.

VII. REFERENCES

1. R. Pramoditha, 11 Different Uses of Dimensionality Reduction, URL: https://towardsdatascience.com/11-different-uses-of-dimensionality-reduction-4325d62b4fa6
2. Scikit learn general usage, URL: https://scikit-learn.org/stable/
3. Kmodes general usage, URL: https://pypi.org/project/kmodes/
4. S. Jaiswal, K-ModesClustering, URL: https://medium.com/@shailja.nitp2013/k-modesclustering-ef6d9ef06449
5. H. Bonthu, KModes Clustering Algorithm for Categorical data, URL:https://www.analyticsvidhya.com/blog/2021/06/kmodes-clustering-algorithm-for-categorical-data/
6. S. Kumar, Clustering Algorithm for data with mixed Categorical and Numerical features, URL: https://towardsdatascience.com/clustering-algorithm-for-data-with-mixed-categorical-and-numerical-features-d4e3a48066a0
7. J. Brownlee, A Gentle Introduction to Expectation-Maximization (EM Algorithm), URL: https://machinelearningmastery.com/expectation-maximization-em-algorithm/
8. Wikipedia, Expectation–maximization algorithm, URL: https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm
9. N.B. Gallagher, et al, The Effect of Data Centering on PCA Models, Eigenvector Research Inc. URL: https://eigenvector.com/wp-content/uploads/2020/06/EffectofCenteringonPCA.pdf
10. S. Mahmood, Factor Analysis of Mixed Data, URL: https://towardsdatascience.com/factor-analysis-of-mixed-data-5ad5ce98663c
11. V. Laparra, et al, Iterative Gaussianization: From ICA to Random Rotations, 10.1109/TNN.2011.2106511 IEEE, URL: https://www.uv.es/lapeva/papers/Laparra11.pdf
12. Wikipedia, Independent Component Analysis, URL: https://en.wikipedia.org/wiki/Independent_component_analysis
13. Sklearn, Random Projection, URL: https://scikit-learn.org/stable/modules/random_projection.html
14. Wikipedia, Random Projection, URL: https://en.wikipedia.org/wiki/Random_projection
15. Sklearn, SelectKBest, URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest
16. Sklearn, ANOVA f_classif, URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html#sklearn.feature_selection.f_classif
17. Sklearn, Clustering performance evaluation, URL: https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation
18. Sklearn, Selecting the number of clusters with silhouette analysis on KMeans clustering, URL: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html