MACHINE LEARNING - OMSCS 7641
Spring 2022
Unsupervised Learning and Dimensionality Reduction

Introduction

In this study, I implement the following five scenarios:

1. Run Kmeans and Expectation Minimization (EM) for the car and water datasets used in the homework 1.
2. Run four dimensionality reduction (DR) algorithms; principal component analysis (PCA), independent component analysis (ICA), random component analysis (RCA) and feature agglomeration (FA) on both datasets. RCA is also called as randomized projections (RP).
3. Implement the two clustering methods on the transformed data after four DR's on both datasets. That makes 4*2*2 = 16 combinations.
4. Apply the four DR's to train the car dataset.
5. Use the clusters obtained at step 3 as new features to train the ANN for the car dataset in addition to the original features.

I do critical analysis by comparing and contrasting, for every scenario. Scikit-learn is used for this homework.

Datasets

The first dataset is from the UCI database. It has six input features. Each feature contains 3-4 distinct values such as low, med, high. The output is ordinal, multiclass classification. The input and the output contain only discrete values. The input contains several features of a car such as number of doors or leg room. The output shows whether a car is unacceptable, ok, good or very good. There are 1727 rows.

The second dataset is from Kaggle. It shows whether the water sample is potable or not. There are nine input features. Each feature consists of continuous real values. Some of the features are pH, hardness, sulfate concentration etc. The output is binary (potable or not potable). There are 3276 rows.

Each dataset is first divided into 80% training and 20% testing set with shuffling. Then the training set is further divided into 80% training and 20% validation set with shuffling. The text data is converted into discrete numbers. Training input data is normalized between 0 and 1 by using MinMaxScaler. That normalization is transferred into validation and testing dataset as well. Final scaled training input data is used in DR's and clustering below.

1. Running the Clustering Algorithms on Dataset and Critical Assessment

K-means clustering does clustering by grouping the closest datapoints into a cluster. The cluster center point is calculated by taking the mean. The distance between the data point and the cluster center is recalculated. Then, the data points are assigned to the clusters with the closest centers. This procedure is repeated until the cluster center move below a threshold. This clustering technique is good for well separated datasets with equal co-variance. K-means is also scalable and its computation time is o(knt) where k is the number of clusters, n is the dataset size and t is the number of iterations. It is better to initialize at farther points by using init = k-means++ from scikit. Otherwise, the solution may converge to a local minimum.

Expectation Maximization (EM) assumes gaussian distribution over the data. This method gives a probability for a dataset belonging to a cluster based on its distance to the cluster center with respect to the standard deviation within the cluster. Unlike K-means, this method can represent elliptic shapes as well. One disadvantage of this method is that it not scalable to very large datasets.

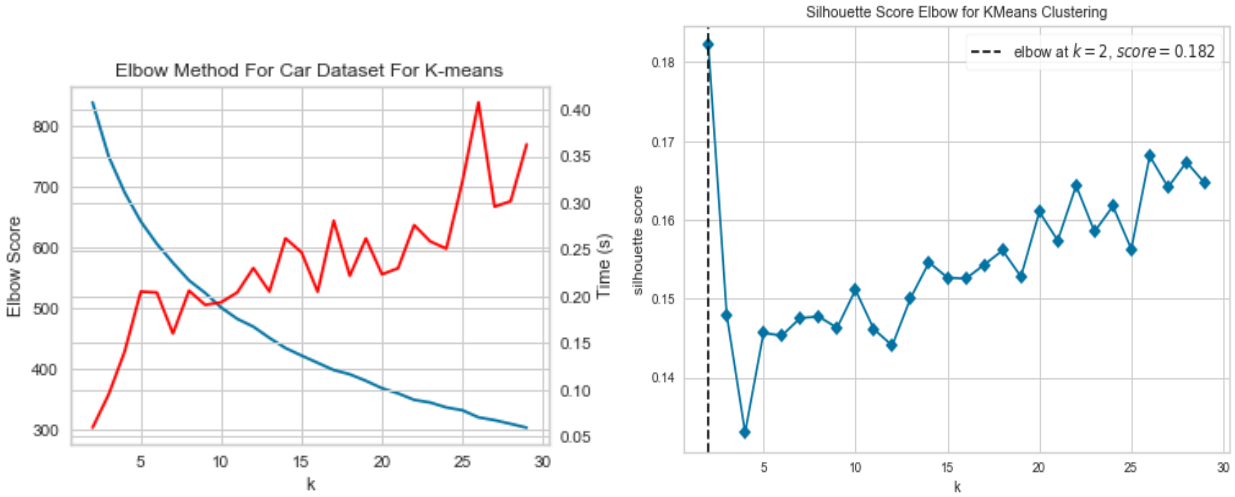

Figure 1. (left). Number of clusters (k) vs Distortion score for the car dataset. Red line shows the computational time. (right) Number of clusters vs Silhouette score for the car dataset.

Car dataset

I try the elbow method and Silhouette method to determine the number of clusters for K-means. inertia_ attribute of k-means outputs the sum of squared distance of each point from their corresponding cluster center. As the number of clusters increases, at first, inertia declines rapidly. However, later, inertia decrease decelerates. This plot of number of clusters vs inertia shows an elbow shape. The elbow is the optimum number of clusters. I used yellowbrick to visualize this plot as shown in Figure 1. Figure 1(left) gives the optimum number of clusters as 10. However, I am not able to see a well-formed elbow shape. Therefore, I also try the Silhouette method. Silhouette method is a more complicated method that looks at more aspects where the average distance between the datapoint and the cluster centers it does not belong to, as well as the average distance between the datapoint and all the other data points in the cluster it belongs to, are considered. For the Silhouette method, the number of clusters that gives the highest Silhouette score is the optimal number of clusters. Figure 1(right) shows that the optimal number of clusters is 2. Figure(1) left also shows the fitting time for K-means which increases linearly as the number of clusters increases because K-means is o(knt) where is the number of clusters.

Water Dataset

K-means is fixed to a random state. Everything is else is default. In order to determine, the number of optimal clusters, I try the elbow method by varying the number of clusters from 2 to 3. Figure 2(right) shows the elbow method for water dataset for K-means. Yellowbrick find the optimal number of clusters as 10. However, I do not see an obvious elbow shape and I try the Silhouette method as well. Figure 3 (left) shows the Silhouette method for the water dataset for K-means. I can see that the Silhouette score is highest at 21 clusters. Figure 3 (right) shows that 2 clusters gives the highest Silhouette score. I would pick the Silhouette method over the Elbow method because the first shows a significant difference as a function of the number of clusters.

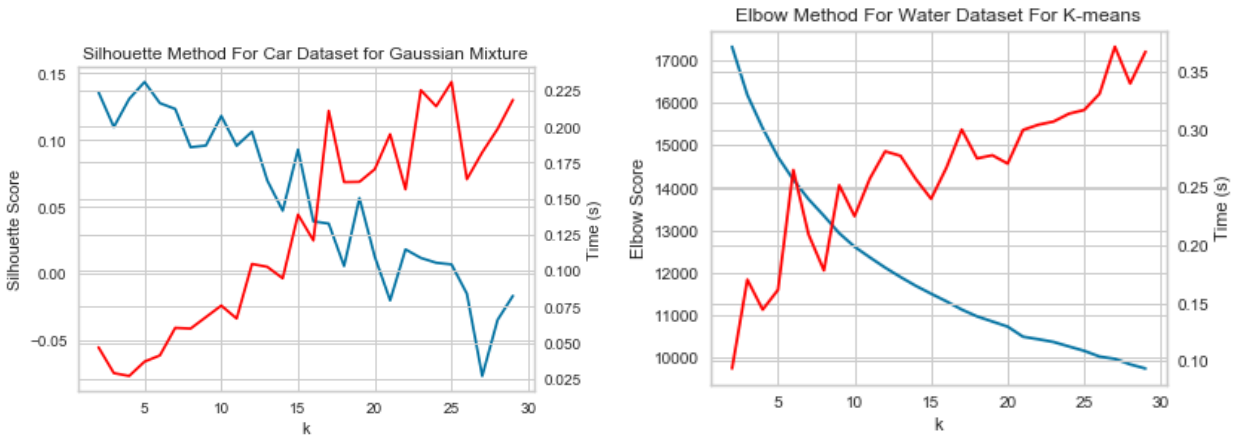

Figure 2. (left) Silhouette method for car data set for Gaussian Mixture Clustering. Red color is computational time. (right) Elbow method for water dataset for K-means clustering. Red color is computational time.

The red lines for both K-means (Figure 2 – right) and GM (Figure 3 – right) show that the computational time increases linearly with the number of clusters. This is same as the car dataset and is expected because the complexity is linear to the number of clusters. Cars training data has 1104 samples and water training data has 2096 data. I had noted that GM is not scalable to large datasets and I can see that water dataset is an order of magnitude slower than car dataset for GM. For K-means, computational time for two datasets is similar.

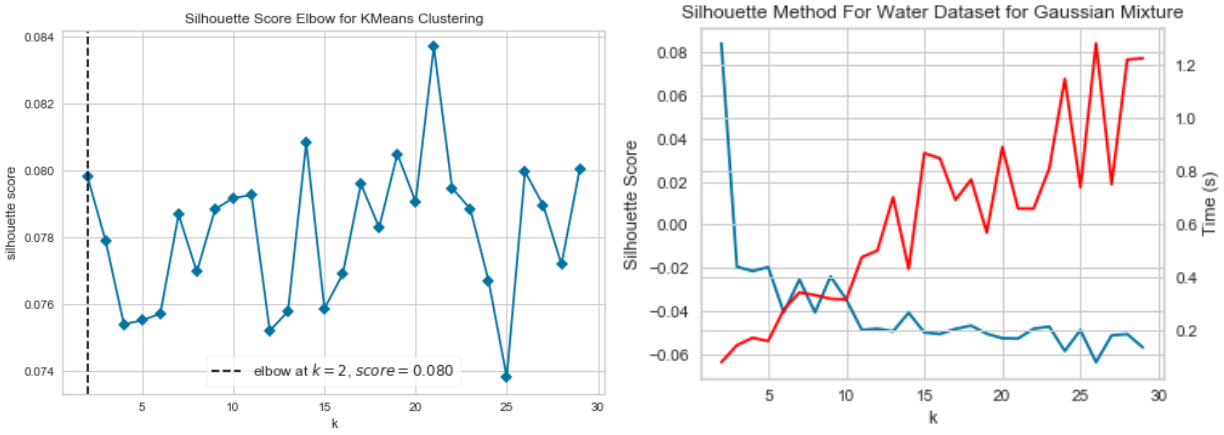

Figure 3. (left) Silhouette method for water dataset for K-means clustering. (right) Silhouette method for water dataset for Gaussian Mixture clustering. The red line shows the computational time.

2. Dimensionality Reduction Algorithms

In this section, I implement four dimensional reduction algorithms (PCA, ICA, RP, FA) to car and water datasets and evaluate what I see.

Principal Component Analysis (PCA)

PCA projects the features of the data onto its principal components by using eigenvalue decomposition. Largest eigenvalues and their corresponding eigenvectors can be used to represent the data. Smaller eig envalues can be ignored to reduce the number of feature by losing very little information. The compone

nts with the largest eigenvalues will also have the largest variance. Below is the explained_variance_rati o_ of the car dataset: [0.17634474, 0.16961097, 0.15875095, 0.14187184, 0.13745751, 0.13159785]. It s eems to me that all the eigenvalues are significant and hence they should all be used. I don't think PCA w ill help for the car dataset. The explained_variance_ratio_ for the water dataset is [0.13176002, 0.13046 058, 0.11935534, 0.11380889, 0.11004361, 0.10634697, 0.10353079, 0.09805662, 0.08663719] by using all its 9 features. I can see that two of the values are less than 0.1 and hence the dimension of the probl em can be used from 9 to 7. It means two of the transformed features are ignored because their varianc es are negligible. From figure 4-1, I can see that the two principal components are able to cluster the car data into 3 whereas in reality there are 4 clusters. The 4th cluster could be visible in another dimension. Figure 4-2 shows that the data is centered in a ball-shape around 0 when plotted on two principal compo nents. This plot does not give info on clustering.

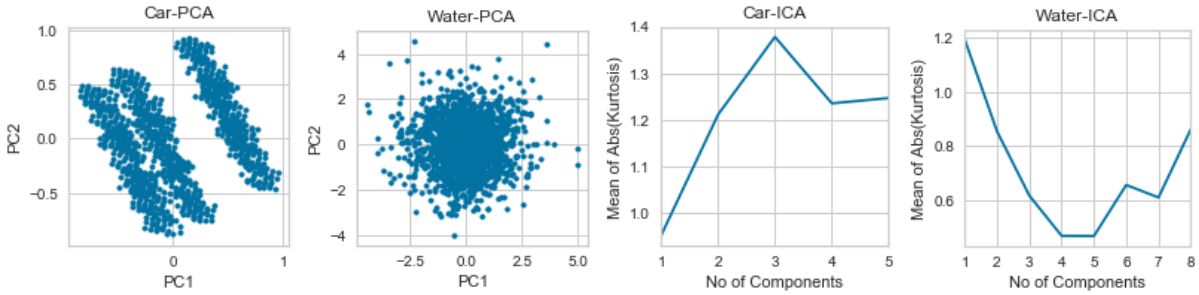

Figure 4: (from left to right) 1. Scatter plot for the two principal components for the car dataset. 2. Scatte r plot for the two principal components for the water dataset. 3. Number of components vs Kurtosis for Car for ICA analysis. 4. Number of components vs Kurtosis for Water for ICA analysis.

Independent Component Analysis (ICA)

ICA analysis creates a mixing matrix that will transform the features into new features where the new features are mutually independent. At the least, that is the goal. This method could be useful if we think that the current features are actually a combination of some other independent features. The original independent features could be more useful in creating a machine learning model. I use the default values for FastICA from sklearn.decomposition with random_state = 1. Figure 3-3 and 3-4 show the number of components vs the mean absolute of kurtosis for the car and water datasets, respectively, after fitting and transforming the input data with ICA. Kurtosis is highest at 3 and 1 components, respective and those can be taken to analyze the data.

Random Component Analysis (RCA)

One big advantage of RCA is that it is fast. It generates the components in random directions. We use the reconstruction error to find the optimum number of components. Reconstruction error is $|X - X^n(W^{-1})^T|$, X is the original input data, X^n is the transformed X, W is the random matrix used for projection, T is transpose. Reconstruction data is equivalent to the sum of the distance from the transformed data points to the selected eigenvectors. The lower this value, the more data points align on the selected components. Therefore, we want to minimize the reconstruction data. Reconstruction error also shows how much the original X data has been distorted. Reconstruction error reduces with the number of components. However, we want to use as few components as possible for efficiency. Therefore, we do an elbow analysis to analyze the trade-off between the number of components and the reconstruction error. Figure 5-1 and 5-2 show the reconstruction error as a function of the number of components for car and water datasets, rescpectively. As soon as one component is removed, the error increases significantly. I do not see an elbow shape and I think this random transformation causes

significant reduction in the representation of the data if the number of components is reduced. Therefore, I would rather use all the components.

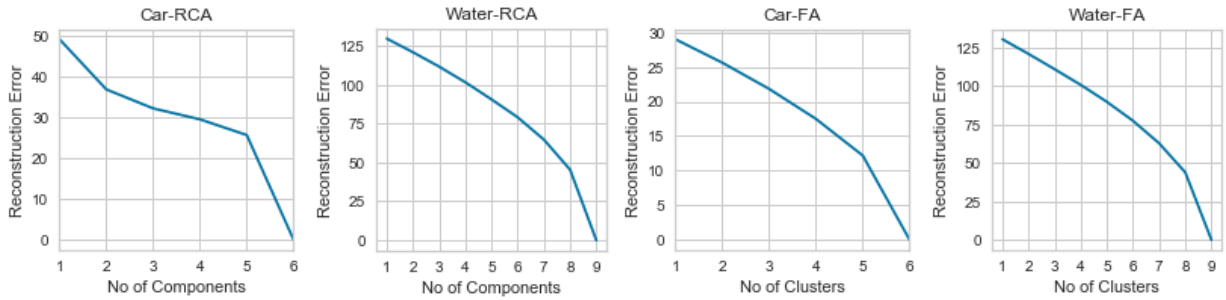

Figure 5 (from left to right): 1. Number of Components vs Reconstruction error for Car data for RCA Analysis. 2. Number of Components vs Reconstruction error for Water data for RCA Analysis. 3. Number of Clusters vs Reconstruction error for Car data for FA Analysis. 4. Number of Clusters vs Reconstruction error for Water data for FA Analysis.

Feature Agglomeration (FA)

Feature agglomeration looks for features that are very similar and combines them in a reduced number of features. FA also uses the reconstruction error that was defined in RCA. I use the default values for sklearn.cluster.FeatureAgglomeration and vary the number of clusters to analyze reconstruction error as a function of cluster. Figures 5-3 and 5-4 show the reconstruction error as a function of number of clusters for car and water datasets, respectively. As soon as one cluster is removed, the error increases significantly and there is no elbow shape. Therefore, it may be the best idea to keep all the features.

3. Dimensionality Reduction + Clustering for 16 Combinations of Datasets, Dimensionality Reduction, and Clustering Methods.

For each scenario, I have two nested loops. First loop is over number of components/clusters for dimensionality reduction. The second loop is over the number of clusters for the clustering method. For the elbow method, I select the candidate with the lowest elbow score. For the Silhouette method, I select the method with the highest silhouette score. For K-means, I look both at the elbow score and the silhouette score. For EM, I look only at the Silhouette score. For each case the following steps are carried out.

loop over i number of components from [1,no of features]:
    Carry out dimensionality reduction and predict X_transformed.
    loop over k clusters between [2,20):
        Carry out clustering.
        Calculate elbow and silhouette score as necessary.
        Store the i, k and elbow score that gives the lowest elbow score.
        Store the I, k and silhouette score that gives the highest silhouette score.
    end loop k
end loop i

Table 1 list the 16 combinations and their optimal cases. DR gives the optimal number of components for dimensionality reduction (PCA, ICA, RCA, FA). Cluster # gives the optimal number of clusters. Elbow and Silhouette scores give the scores for the corresponding optimal cases.

Elbow method looks at the distance between the datapoint and the cluster center. Table 1 shows that the optimal number of DR component that gives the lowest elbow score is always 1 both for the car and water datasets. It means that the points are closer to the cluster center when they are project on one dimension only. I also notice that elbow method always prefers the largest number of clusters. This is because when there are more clusters on a single line, the dataset will be divided into more brackets, and the datapoints will be closer to the corresponding cluster center. For both datasets, random projection (RCA) gives the highest elbow score because random projection does not seek eigenvectors that give the largest variance. It is just another projection on some random directions. I notice that independent component analysis (ICA) gives the lowest elbow score. ICA looks for fewer components to express the features because the original features may be a combination of a smaller feature set. I would expect PCA to give the lowest elbow score as it actually seeks the highest variance. The results also may have been affected by the random_state number.

I use the Silhouette score only for the Kmeans clustering as well as EM. Silhouette does not give a significant difference among different DR methods for a particular dataset. Silhouette method considers not only the points distance to the datapoints in the same cluster, it also looks at the distance between the datapoints and other cluster centers. It is possible that the latter calculation dominates the Silhouette score and hence no difference between the DR methods.

When Kmeans and EM are compared, EM tends to give a higher Silhouette score. I think EM's probabilistic, gaussian distribution approach may be helping evaluation different cases better. Further, unlike Kmeans, EM allows elliptic dataset distributions.

Table 1. Summary for 16 Combinations

Case	Dataset	Method	DR	Cluster #	Elbow Score	DR	Cluster #	Silhouette Score
1	Car	PCA+Kmeans	1	19	0.26	2	2	0.18
2	Car	ICA+Kmeans	1	19	0.004	2	2	0.18
3	Car	RCA+Kmeans	1	19	8.92	3	2	0.115
4	Car	FA+Kmeans	1	19	0.144	2	2	0.18
5	Car	PCA+EM				1	2	0.16
6	Car	ICA+EM				1	2	0.16
7	Car	RCA+EM				6	3	0.14
8	Car	FA+EM				2	2	0.182
9	Water	PCA+Kmeans	1	19	8.03	9	21	0.084
10	Water	ICA+Kmeans	1	19	0.008	8	2	0.08
11	Water	RCA+Kmeans	1	19	210	7	2	0.074
12	Water	FA+Kmeans	1	19	1.38	8	2	0.08
13	Water	PCA+EM				2	2	0.11
14	Water	ICA+EM				2	2	0.1
15	Water	RCA+EM				8	2	0.09
16	Water	FA+EM				9	2	0.08

Silhouette score tends to be lower at number of clusters. This may be because the distance between the data points and the other cluster centers are considered in the Silhouette score equation. Various number of DR components may lead to the best Silhouette score.

4. Use Transformed Input from DR to Train the Car Dataset

There were no insignificant eigenvalues after PCA. I decided to take four principal components. Based on the results of the step 2, I can 3 components for ICA, 2 components for RCA and 5 clusters for FA.

I do a grid search for MLPClassifier by varying number of layers and neurons. early stopping is turned on and the training stops if validation set doesn't show improvement after 100 steps. Maximum number of iterations are 10000, ransom state is 42 and the activation function is tanh. This setup is the same for all the DR+ANN combinations. Each combination tests 225 fits.

Table 2 summarizes the results for the original ANN developed in HW1 as well as the four DR+ANN combinations. None of the dimensionality reduction algorithms help with the F-scores. This is because as we are reducing the dimension, we are losing information as well. Among the DR's, FA gives the highest F-score because it uses the greatest number of components (5). F-scores are the worst for PCA and RCA as they have fewer components/features. As the car dataset only has 6 features, all of them are probably needed and there is no redundant/independent/irrelevant feature. There are no features that can be combined into one or fewer. The number of features cannot be reduced without losing information at another space because eigenvalues are significant for PCA, and reconstruction error increases sharply as soon as a component is reduced for RCA and FA.

Table 2. Summary for DR+ANN Combinations

	Features #	Layer #	Neuron #	Training F-score	Validation F-score	Test F-score	Time (s)
Original	6	4	20	0.9524	0.8242	0.8414	2.35
PCA+ANN	4	3	20	0.3993	0.3840	0.3784	1.25
ICA+ANN	3	3	20	0.3498	0.3612	0.3513	1.16
RCA+ANN	2	3	5	0.3457	0.3455	0.3557	0.66
FA+ANN	5	3	20	0.6088	0.6746	0.5511	1.45

Table 2 also shows that computational time is less when the number of components is reduced after DR. This is expected because as the input matrix gets smaller, forward, and backward calculations are less costly. Another reason for DR+ANN's being fast is fewer number of iterations. Training stops in fewer number of iterations as validation curve very quickly stops making improvements. For example, the original ANN converges in 369 iterations where DR+ANN cases converge in fewer than 270 iterations. RCA+ANN is the fastest as it converges only in about 190 iterations and it only has 2 components/features.

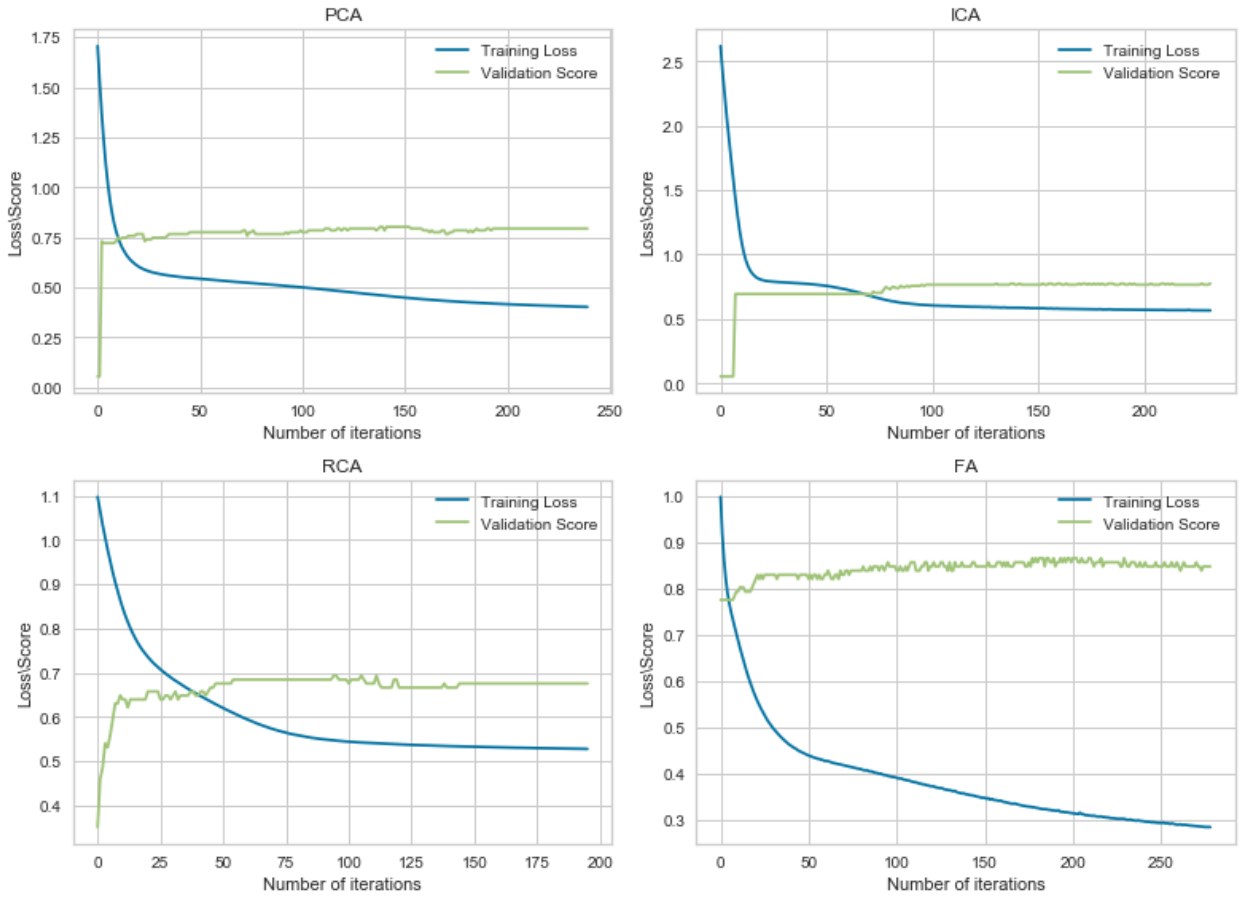

Figure 6. Training Loss and Validation score curves for PCA (Top Right), ICA (Top Left), RCA (Bottom Left), FA (Bottom Right).

5. Using DR+Clusters as Input Features for ANN

In this section I take the first 8 cases listed in Table 1 from Section 3 for the car dataset and use the clusters as input for the ANN in addition to the original features. The selection of dimensions and clusters is based on the Silhouette criterion. The dataset is again the car dataset that has 6 inputs and 1 output with 4 labels.

The following is carried for each of the 8 DR+Cluster combinations:

1. Fit and transform training input by using the selected DR. Carry the transform to validation and testing input as well.
2. Fit and predict the cluster for the transformed training input by using the selected cluster with specified components. By using the existing clustering model, cluster training and validation input as well by doing unsupervised learning. No actual label is used.
3. Concatenate the scaled input data with the cluster data and use as input for MLP.
4. Setup an MLPClassifier with random state 42, activation tanh, with early stopping. Training stops 100 iterations after no improvement in the validation score.
5. By using the model setup in step 4, use GridSearchCV to tune the number of layers, number of neurons and maximum iterations.

6. Use the best model to compute time, f-score for training, validation and testing scores.

Table 3 shows the results of training different combinations. Number of layers and neurons are optimal at 3 and 20, respectively, for each of the 8 cases studied. This is not surprising because already 6 of the 7 features are the original features. When I look at the testing F-score, I can see that PCA+Kmeans, ICA+Kmeans, FA+Kmeans and FA+EM perform better than the original ANN model. Even though the number of features and clusters are only 2 for all these successful cases, the mapping of the clusters introduce new connections to the neural network. The ANN models perform much better by taking advantage of these connections. I can see that PCA+EM and ICA+EM perform the worst and both of these cases only have one component after DR. When it comes to the computational time, all of the combinations are faster than the original dataset even though the original dataset is now smaller because it doesn't have the clusters as input features. However, one of the reasons for this difference may be the number of iterations. I can see that PCA+Kmeans and ICA+Kmeans have similar number of iterations to the original case, and they show similar computational time. Other cases converge in fewer number of iterations.

Table 3. Summary for DR+Clusters+ANN (8 Combinations)

Method	Feature #	Cluster #	Layer #	Neuron #	Training F-score	Validation F-score	Testing F-score	Time (s)
Original			4	20	0.9524	0.8242	0.8414	2.35
PCA+Kmeans	2	2	3	20	0.9445	0.9456	0.8977	2.24
ICA+Kmeans	2	2	3	20	0.9445	0.9456	0.8977	2.38
RCA+Kmeans	3	2	3	20	0.877	0.8823	0.821	1.81
FA+Kmeans	2	2	3	20	0.9223	0.9457	0.9121	1.81
PCA+EM	1	2	3	20	0.8118	0.7358	0.7809	1.14
ICA+EM	1	2	3	20	0.8118	0.7350	0.7809	1.02
RCA+EM	6	3	3	20	0.818	0.8740	0.837	1.06
FA+EM	2	2	3	20	0.9223	0.9457	0.9121	1.77

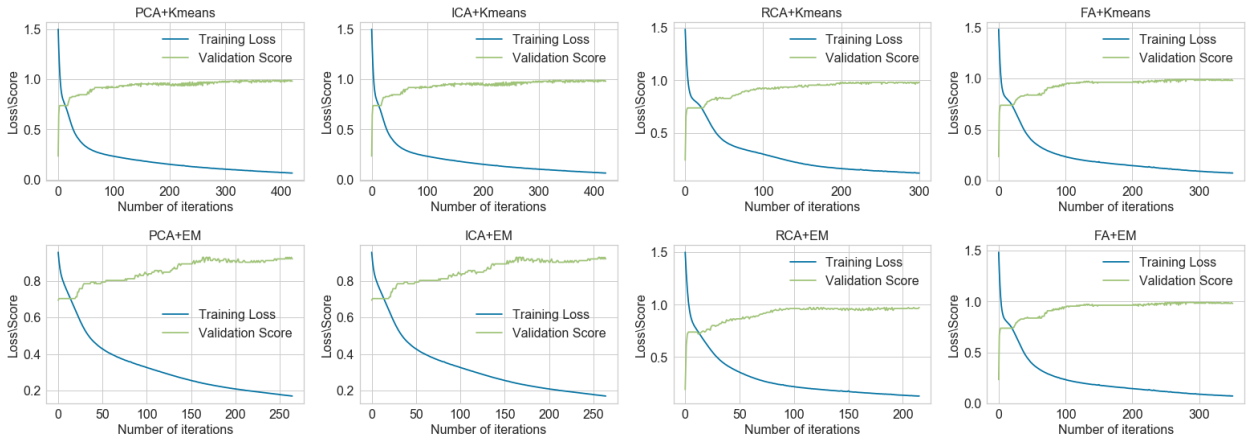

Figure 7. Training loss and Validation score for Kmeans on top and EM on the bottom. From left the right, the DR methods change as 1. PCA, 2. ICA, 3. RCA, 4.FA.