CS 7641 Project 3: Unsupervised Learning

Yaling Wu (ywu342)

Datasets

I used the same datasets as assignment 1.

1. Car Evaluation Data: https://archive.ics.uci.edu/ml/datasets/Car+Evaluation

This data was derived from a hierarchical decision model. It was used for the evaluation of Hierarchical Induction Tool, which was proven to be able to reconstruct the original hierarchical model. So it is particularly useful for testing constructive induction and structure discovery methods. It includes attributes like buying, maintenance prices, doors, persons, size of luggage boot, and safety. The output class indicates acceptability of cars: unacc (70%), acc (22%), good (4%), vgood (3.8%). There are 1728 instances in total.

2. Tic-Tac-Toe Endgame Data: https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame

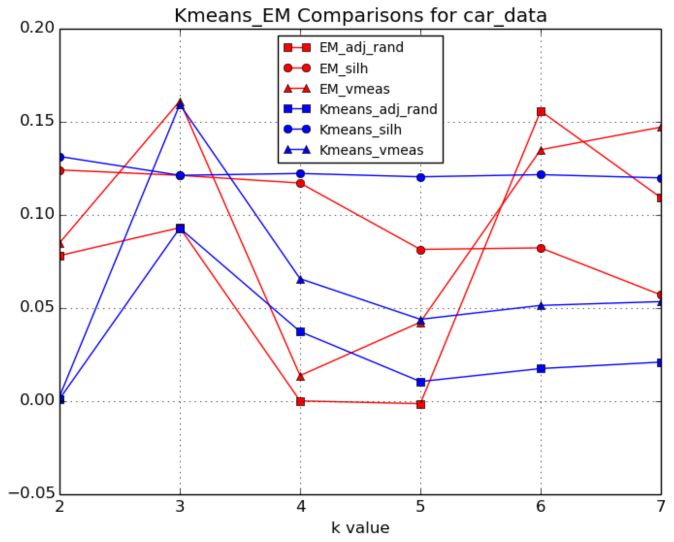
This dataset encodes the complete set of possible board configurations at the end of tic-tac-toe games, assuming "x" goes first. The target concept is "win for x" or positive. There are 958 instances and 9 attributes, each corresponding to one tic-tac-toe square. About 65.3% of the examples are positive.

To prepare for data processing, I randomly selected 70% of car data for training and 80% of Tic-Tac-Toe data for its training, since car data is larger than the latter.
The two datasets are interesting for this assignment because they are of completely different concepts and structures. They have given drastically different results from my experiments, as shown in details below.

Clustering On the Original Data

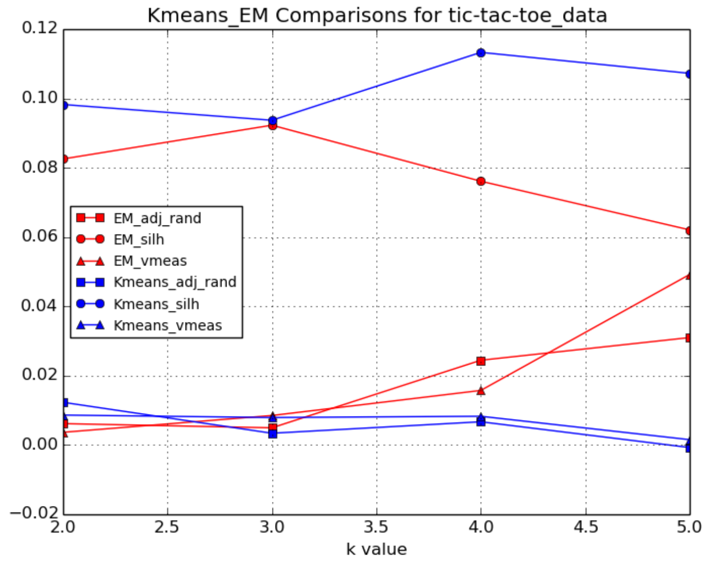
The first set of experiments is to run clustering algorithms K-means and Expectation Maximization (EM) on the original datasets described earlier and see if the output labels line up with their ground truth labels. The corresponding scikit modules I use are KMeans and GaussianMixture. To find a best k, or number of clusters, for each dataset, other parameters of the estimators were fixed and set to default, including the distance function used. To find out which k fits well with the data, I evaluated clustering outputs using silhouette_score (SS), adjusted_rand_score (ARS) and v_measure (VM)_score under the metrics module of sklearn. Among them, silhouette score is calculated using the mean intra-cluster distance and the mean nearest-cluster distance for each sample. The silhouette_score function returns the mean over all samples. Values near 0 indicate overlapping clusters whereas higher silhouette coefficient scores relate to models with better defined clusters (i.e. clusters are denser and well separated). And the adjusted rand index computes a similarity measure between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. A value of 1.0 means the two clusterings of interest are identical. Random label assignments have an ARS score close to 0.0. The V-measure is the harmonic mean between homogeneity and completeness, where homogeneity is maximized when each cluster contains elements of as few different classes as possible and completeness aims to put all elements of each class in single clusters. Thus homogeneity is maximized if a cluster consists of only samples of one class. V-measure is another way to determine whether clustering labels are similar to the ground truth. In all, all three measures are positively related to the correctness of parameter k.

Car Data



From above figure, red indicates EM and blue indicates Kmeans. One shape refers to a kind of metric. In this dataset, there are 4 classes/labels. Silhouette score can be a good indication of how well a clustering is for different k values while ARS and v-measure score indicate how well clustering labels line up with ground truth labels. Based on the results, K-means outperforms EM in the most part. And K-means and EM perform the best when k=3, which is pretty close to the true number of clusters. Three clusters are more likely to be defined in this dataset because the fourth label vgood only spans 3.8% of the instances, which is a very small portion that can be overlooked among other labels in current feature space. For EM, even if v-measure and ARS reach a hill at k=6, its silhouette score implies the opposite (i.e. worse clusters).
To improve the performance, it might matter a lot as to what kind of distance function to use in both algorithms. Euclidean distance may not be a suitable function for this dataset. Also, one other characteristic of kmeans is it treats each feature equally if left unchanged. But in car data, some features may not play a role in determining how acceptable a car is. We may want to use dimensionality reduction to filter out unimportant features which negatively influence clustering results. EM is a pretty similar algorithm as kmeans except it also takes into account the covariance structure of the data and also it does soft clustering. So inclusion of noisy features can still interfere with EM clustering.

Tic-Tac-Toe Data

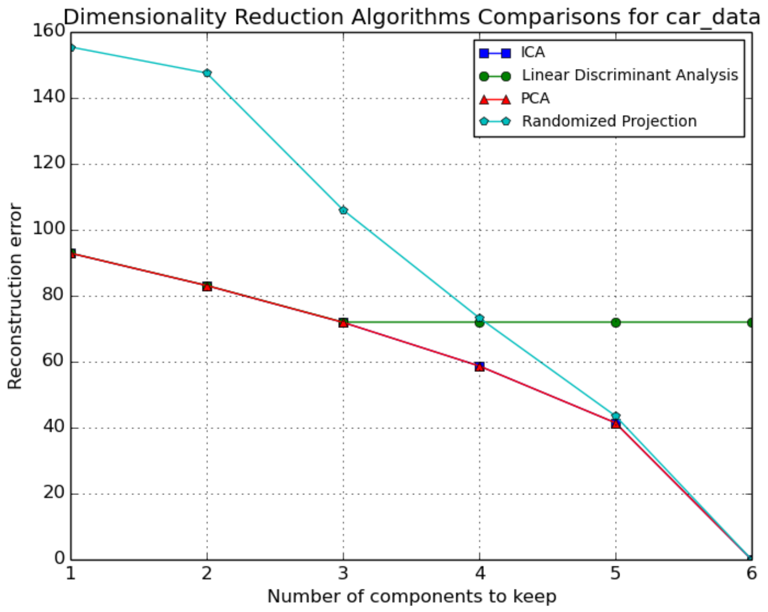


The measures on this dataset are quite different from the first one. Kmeans is better for silhouette but EM is better for other two metrics. ARS and VM come closely together per algorithm. Overall, ARS and VM are very close to 0, which indicates that clustering labels do not line up with ground truth labels very well and low silhouette indicates the clusters are not clearly defined, no matter which algorithm is used. This may be caused by insufficient amount of data available such that some data points may appear sparse and split. EM produces the best outcome when k=5, which is pretty far away from the true number of labels (i.e., 2).
All three metrics are not very high for this dataset. Thus clustering algorithms does a pretty bad job of partitioning data points and labeling original instances to their corresponding true labels. Changing other hyperparameters of each algorithm, besides k, may help in boost the performance, such as the distance function and number of initializations to perform. The initial conditions are important as to how the final clustering will turn out to be like because it can lead to escaping from local optima. Dimensionality reduction can also help to improve clustering because again, in this dataset, some features or positions on the board may not contribute to the final win or lose. It can also be the case that some features should be weighed more in determining clustering. So feature transformation can be employed to alleviate this problem.

Dimensionality Reduction Algorithms

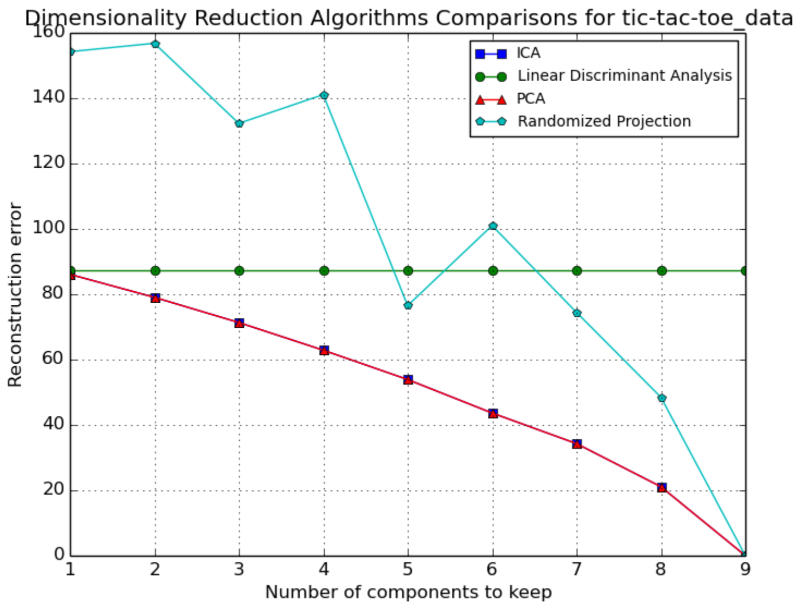
Four algorithms were employed to carry out dimensionality reduction: PCA, ICA, Randomized Projection (RP) and Linear Discriminant Analysis (LDA). Respectively, I used PCA, FastICA, and GaussianRandomProjection in sklearn to implement them. To evaluate on the reduction results, I reconstructed on the reduced feature space and took L2 norm error between reconstructed X and original X. More details about PCA and ICA experiments are demonstrated in the last two subsections.

Car Data



Above graph shows how reconstruction errors change as we throw away less principal components. In general, it makes perfect sense that as more components are kept, less information is lost and thus less error is induced during reconstruction. It is interesting to notice that PCA and ICA curves overlap on the graph. This means they find components with similar distributions during the process. RP induces higher reconstruction errors overall because of its randomness in choosing projections. The transformed X it produces captures correlations stochastically so it is very likely to lose more information from reduction, compared to other methods. The error curve of LDA stops decreasing until n_components goes over 3. That implies LDA can only find 3 component axes that account for the most variance between different classes. And after that LDA has stopped to include any other components even if the request asks for more than 3. So this implies that other axes cannot induce a significant label separation.

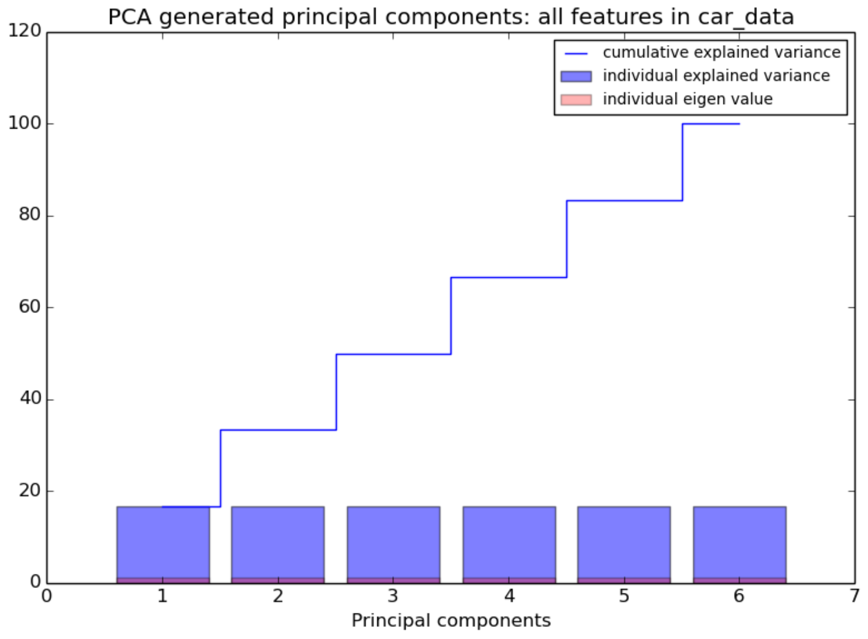
Tic-Tac-Toe Data



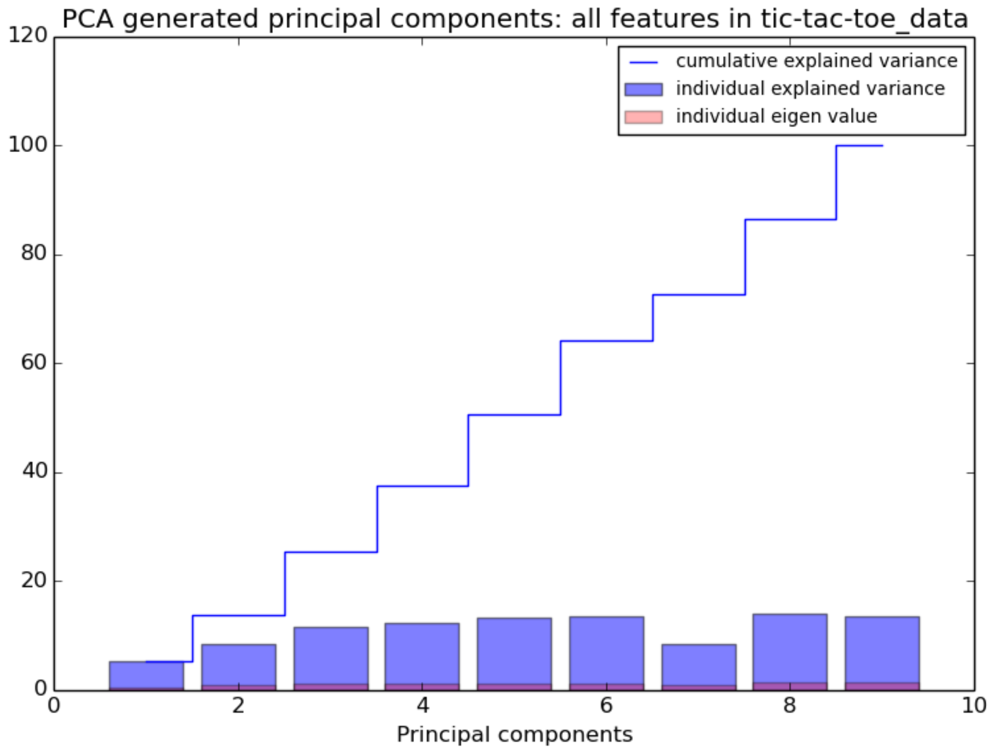
As for the second dataset, everything looks very similar to the previous one. PCA and ICA still reside on the same smooth curve. Randomized Projection curve goes above them for the reason described above. It is worth to note that RP curve reaches a significant valley when n_components=5. This may indicate 5 is the best number. Except this time, LDA error stops decreasing at n_components=1. LDA can find at most 1 discriminant.

PCA

Eigenvalues and explained variance are the two metrics I used in evaluating PCA components. An eigenvalue tells us how much information its corresponding eigenvector/direction bears about the distribution of the data. The higher the eigenvalue is, the more important that component is. We should choose to drop eigenvectors with lower eigenvalues. The explained variance tells us how much information (variance) can be attributed to each of the principal components.



As shown in above graph, all 6 generated components from PCA have a similar eigenvalue and explained variance. This indicates every feature contributes in representing car data equally and so current feature space can be a good one already. It will not make much difference in retaining different principle components. Explained variance bars show each component expresses the same amount of variance in the data. Each time we add a component to the feature space, we add an even amount of information.



As for tic-tac-toe, several components provide significantly more information about distribution and variance of the original data. The 8th component will explain the variance by the most amount (~13.9%), as opposed to the 1st one (~5.2%).

ICA

For ICA, I used kurtosis to evaluate the reduced feature space. Kurtosis measures non-Gaussianity of the data. It is useful in this case because ICA separates sources by maximizing their non-Gaussianity or the independence among them. A kurtosis of 3 indicates a normal distribution.

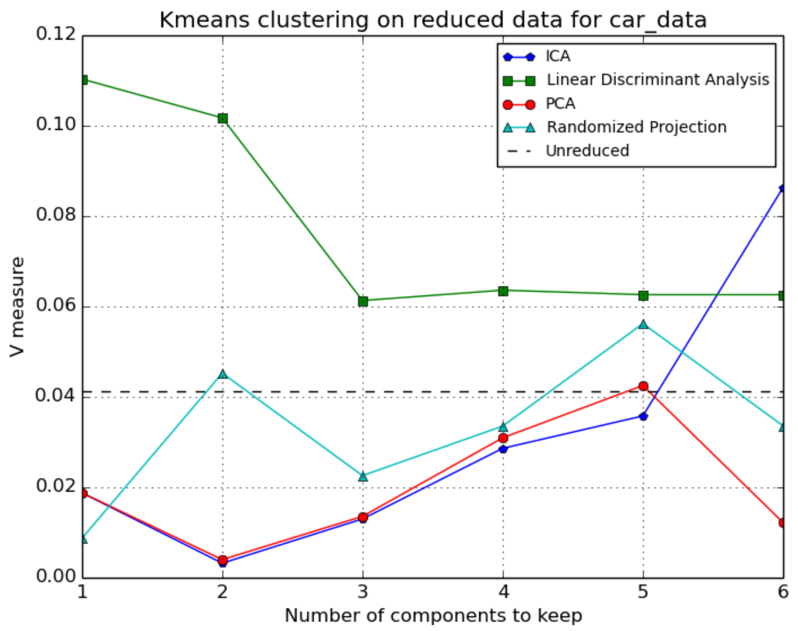
	1	2	3	4	5	6	7	8	9
Car	1.640	1.500	1.640	1.640	1.500	1.499			
Tic-Tac-Toe	1.731	1.727	1.863	2.060	1.730	1.861	1.861	1.729	1.862

Above table shows kurtosis for each of the component per dataset. Interestingly, there is no much variance among components for Car data and there is more of a variance among those for Tic-Tac-Toe data, just like the PCA values shown before. The implications behind this observation are no different. Each component of Car data gives about the same amount of information whereas some components of Tic-Tac-Toe data provide more information than others. Moreover, none of the transformed features displays a Gaussian-like distribution, which indicates they are relatively independent.

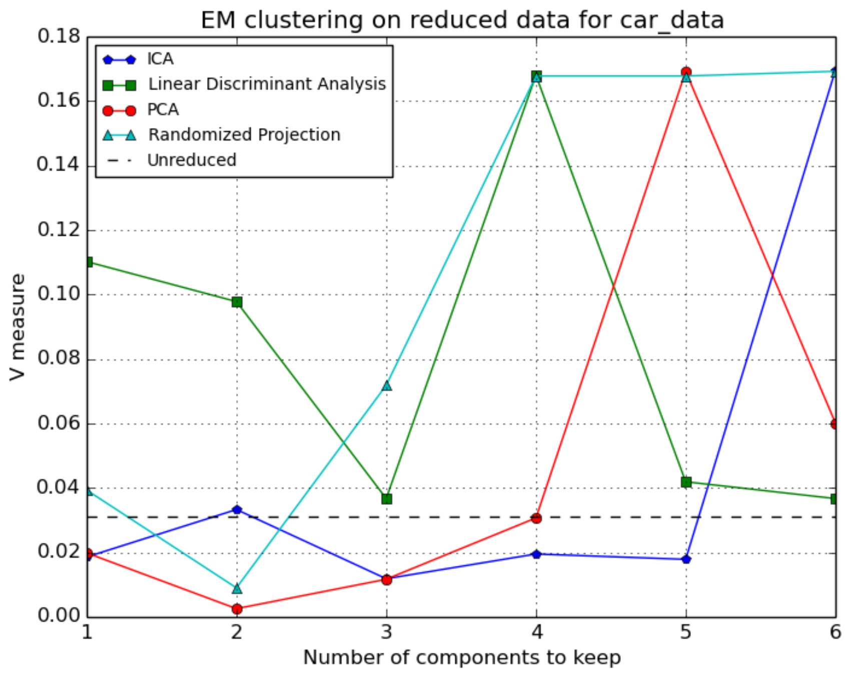
Clustering on Reduced Data Space

In this experiment, I applied Kmeans and EM to dimensionally reduced data and compare the V-measure scores of resulting clusters across different reduction algorithms. For all the v-measure comparison figures, I also included clustering on original data for reference. Also, the number of clusters is fixed to the number of classes available in each dataset.

Car Data

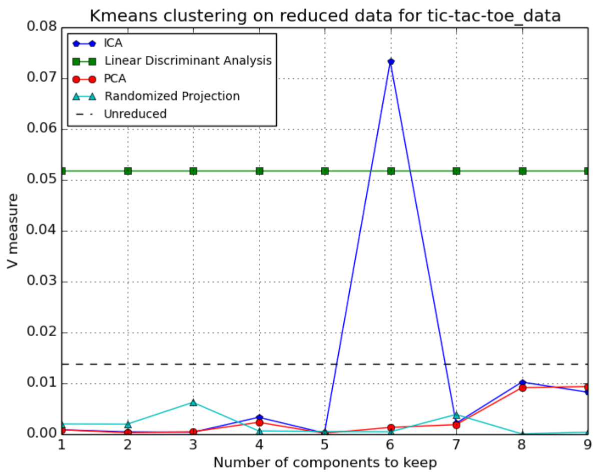


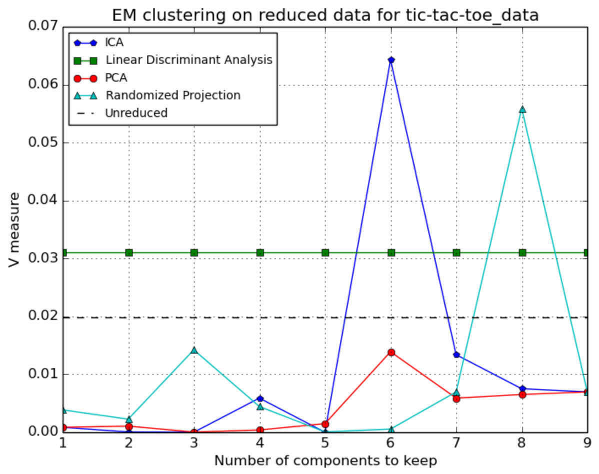
This figure shows how VM score varies for different reduction algorithms across feature space of different sizes. Since unreduced data (black dashed line) has all attributes retained, its clustering quality stays the same. For most of the algorithms, their VM scores tend to increase when they get to keep more information. Similar to results from previous section, PCA and ICA behave the same for the most part. Even though the LDA curve goes down in the beginning, it stays stable after n_component=2, and it is the only algorithm that beats the unreduced clustering. Especially at n_comp=1, it is at its peak. This may imply that one direction that maximizes class separation alone can give the most information about where to cut the lines for classes. Other directions may have interfered the result. The fact that PCA does not perform well in the dataset may be due to the fact that it ignores class labels, unlike LDA. And the maximum class discriminant is different from the component that maximizes the variance.



For EM clustering, not only does it render an overall better score for reduced data, but also most of the transformed spaces perform better than original data. This is the opposite of earlier results of clustering on unreduced data. The reduction algorithms have their peaks at n_comp=4 or 5 or 6. As more information is retained, they are able to leave the right transformed information for probabilistic EM clustering. This means many data points could be residing on the edges where the classes they belong to are ambiguous. Because EM does a soft clustering, with more transformed dimensionality added in, EM is able to assign them more correctly than Kmeans.

Tic-Tac-Toe Data

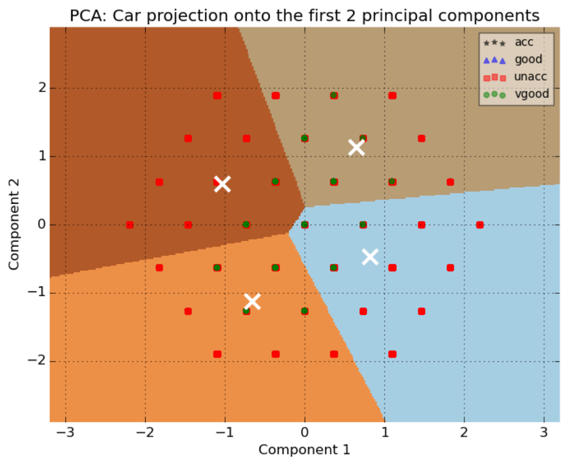


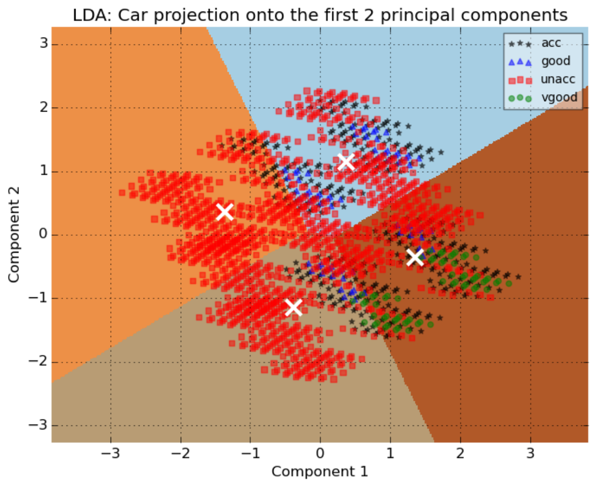


Interestingly, in terms of reduction vs original, Kmeans and EM show different results as last dataset. It is better to use LDA reduction prior to clustering for both Kmeans and EM. ICA has exceeding peaks at n_comp=6 in both cases, specifically. However, whether performing reduction or not does not make much difference. Dimensionality reduction is not very helpful for this dataset as to the two clustering algorithms I used. It can only improve clustering results a tiny in certain circumstances.

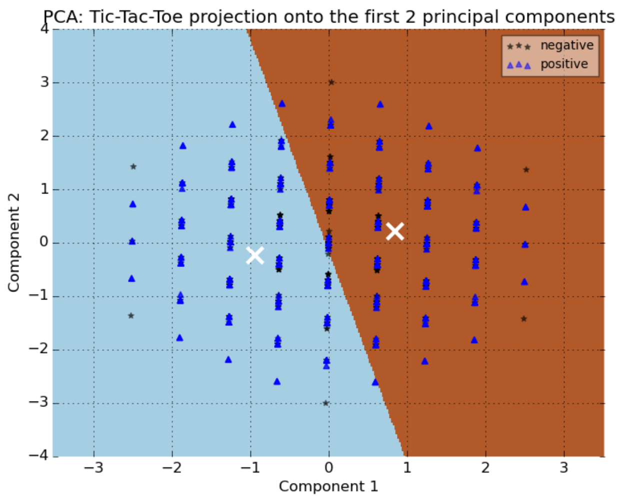
Visualization of Kmeans on Reduced Data

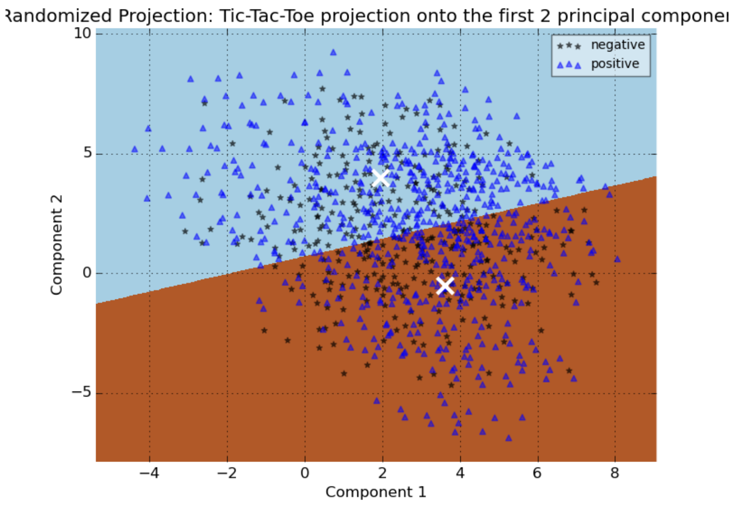
Now that we can reduce feature space to something smaller, to better visualize where data points of different label locate and how clustering has been done on them, I used n_component=2 during the reduction process and plotted the resulting kmeans clusters, where each dot is colored and shaped by its true labeling. The groups labeled by clustering are represented by different-colored area. Centroids are shown as white crosses.





Here shows the effect of two reduction algorithms on Kmeans clustering (EM should be about the same based on above figures). The first two components of PCA clearly do not do a good job of separating out classes. We can see some of “vgood” data points overlap with “unacc” data points and we cannot even see clearly where other two labels are. Each kmeans cluster is a mixture of everything. On contrary, LDA does a much better job of finding the right discriminants that separate labels more. But due to the nature of this dataset, it is very hard to separate out “acc”, “vgood” and some “unacc” on the right side of the graph, which makes the right clusters bomb the v-measure test.

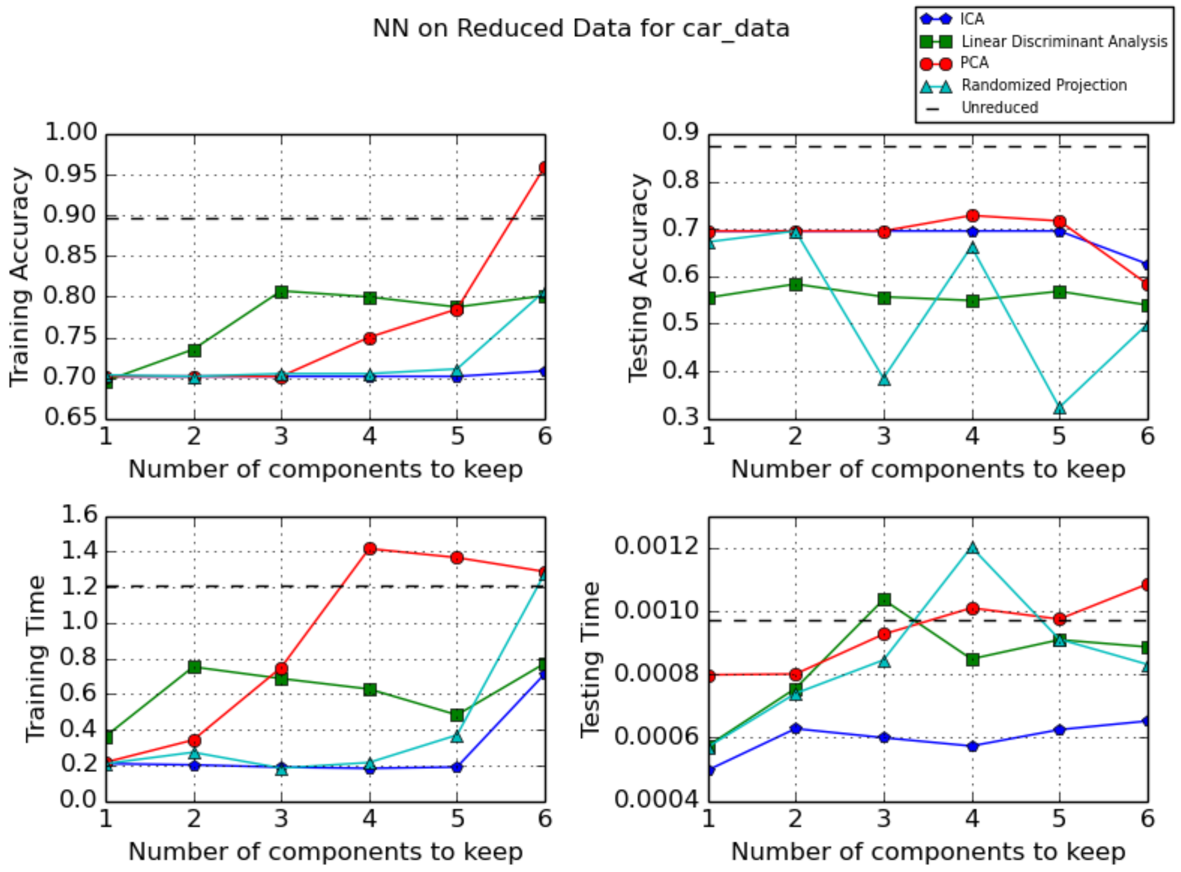




Compared to Car data, Tic-Tac-Toe dataset shows a worse data distribution in regards to how much DR can do to help clustering. Similar to PCA result in Car, labels overlap very well in the 2-feature space. Randomized Projection is a bit better in capturing data variance but can still not find components that well segregate two labels.

Neural Network on Reduced Data Space

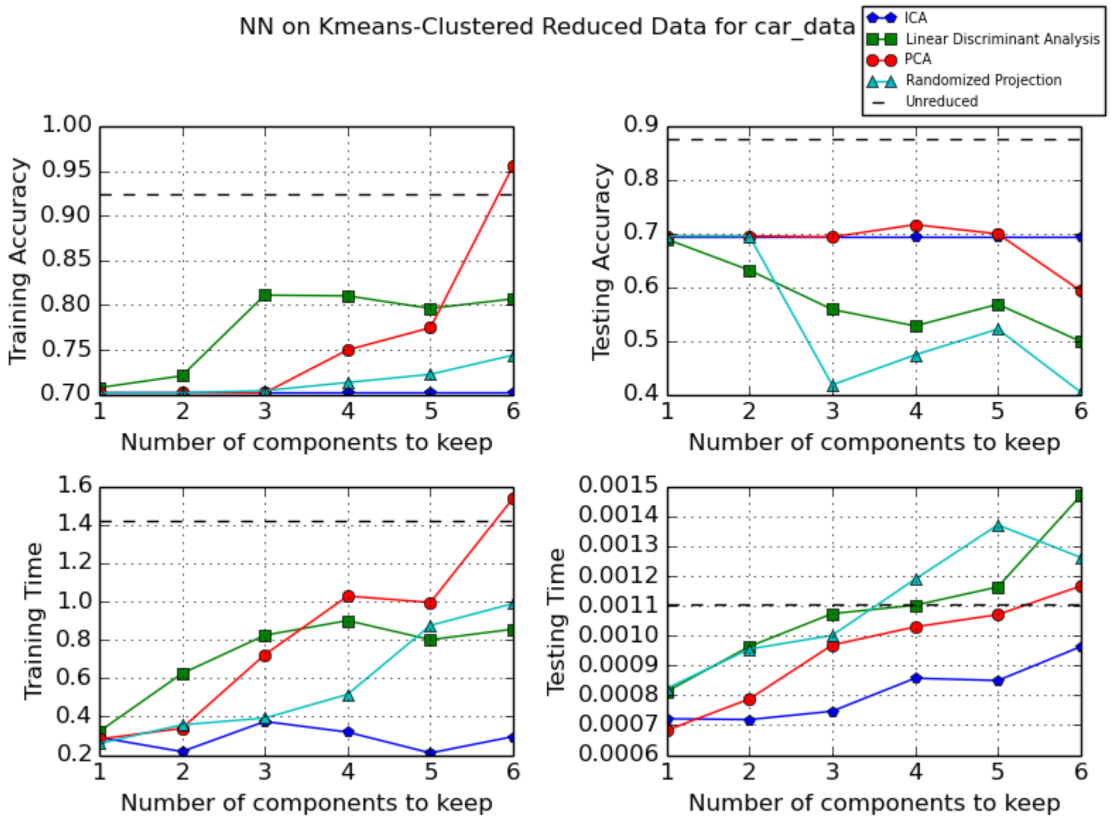
The goal of this set of experiments is to apply Neural Network to the newly projected data to do classification. Again, classification accuracies and running times derived from original data space are also included in the graph for comparison.



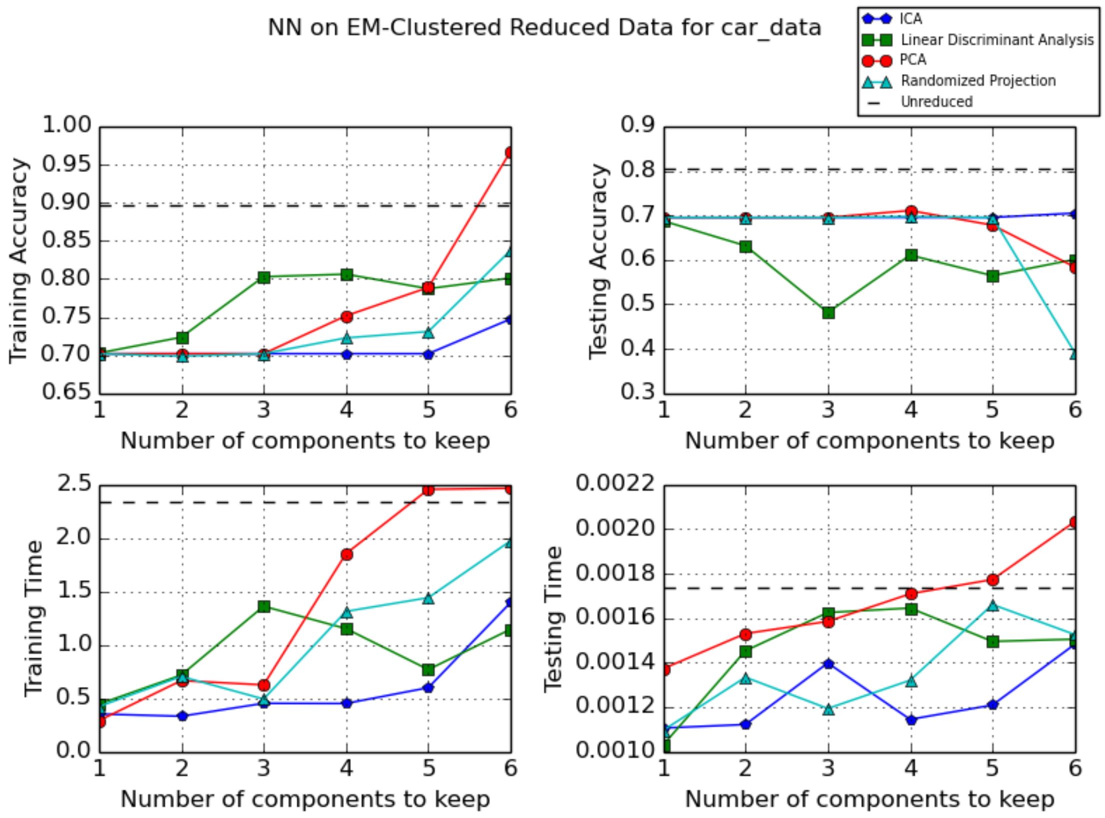
As usual, all training accuracies are higher than testing accuracies. In terms of accuracy, dimensionality reduction does not help much in the improvement of NN. Only when PCA gets to retain all 6 components, is NN able to outperform the case when original data space is used for training. However, when test data is used in prediction, PCA falls a lot back to the level of ICA (around 70%). PCA, LDA and Randomized Projection tend to overfit the training data. On the contrary, ICA curve stays stable at around 70% for both training and testing, even as it increases n_components to keep. Randomized Projection performs the worst. It fluctuates a lot in the testing accuracy plot, this could be due to its randomness property in projections.
It is not surprising to see that training time of NN exceeds testing time of NN by a dramatic amount, no matter how many components are taken out during projection. However, we can see that some reduction in dimensionality can accelerate the training process a lot. All four reduction algorithms successfully decrease training time when 3 and less components are kept in the space. PCA training time quickly goes uphill once more than 3 dimensions are reached. When all 6 dimensions are kept, most algorithms' training and testing times go to the same level as the unreduced. Here ICA still wins the game because it takes the least time in both plots and it does not increase too much until it gets to 6 components kept in training. Therefore, ICA is the best reduction algorithm to use because it retains relatively high accuracies/information and lets NN consume least time. But again, dimensionality reduction worsens performance of NN classification for this dataset. It is inevitable to see information that is important to NN go lost from reduction.

Neural Network on Clustered Reduced Data Space

In the last experiment, the only step that is different from the last one is I added an extra clustering process to the feature space between dimensionality reduction and neural network classification. Then the resulting cluster labels are appended to the reduced space as a new feature. The unreduced feature space also undergoes the same clustering step. Two clustering mechanisms are experimented: Kmeans and EM.



For accuracies, all the curves behave about the same as the fourth experiment, where no clustering is applied. Only the unreduced line in training increases a tiny which can be ignored. So we can infer that clustering labels are not useful at all. It does not provide additional information to help improve the model performance. This makes sense because the new clustering feature is somehow related to other present features. It can be seen as redundant. Especially for this dataset, clustering does not do well on the feature space anyway. As for times, they all increase a bit compared to last experiment because one extra feature is included in the space. Other than that, no much difference is observed.



Interestingly, EM makes everything worse compared to the KMeans version and also the fourth experiment. In terms of accuracies, training accuracy curves lay out in the same way as the previous two. It is the testing accuracy that incurs a difference. The unreduced curve goes down from almost 90% to 80%. The EM clusters actually worsen NN's ability to predict unseen data by a little. Even if EM does a better clustering on reduced feature space (based on previous experiments), its labeling results can interfere with classification. It also takes a bit longer for NN to train and test on the data across all algorithms.

In conclusion, both clustering and dimensionality reduction do not provide help to the Neural Network classifier. Each feature of the car data contributes equally to the data representation (from PCA and ICA analysis), so its present feature should be good enough to do classification.