Assignment 3: Unsupervised Learning

CS 7641 Machine Learning
Spring 2017

Karel Klein Cardeña

Datasets

Pima Indians Diabetes Dataset

This data set is a subset from the database owned by the National Institute of Diabetes and Kidney Diseases. The 768 instances selected from the superset were chosen with the constraints that the patient be a female at least 21 years old and of Pima Indian heritage. Each instance, corresponding to one person, contains eight numeric attributes describing factors such as plasma glucose concentration, body mass index, age, and other quantitative measures of relevance to the presence of diabetes. Each instance contains a binary label: "YES" if the patient has diabetes, and "NO" otherwise.

Early detection of diabetes is crucial in preventing or delaying the onset of serious health complications. An untreated diabetic will face an increased risk of kidney disease, high blood pressure, and stroke -- conditions whose effects may be irreversible and potentially fatal. Certain risk factors for diabetes, especially Type I, are inherited from both parents and thus correlated to the patient's genetic pool. It is thus very relevant to study an ethnic subset of the total population in order to deduce both the group's propensity to develop diabetes as well as to further assess the onset of diabetes as a function of genetics. Correctly forecasting diabetes using only simple health metrics could lead to a dramatic reduction in long-term detriment to a person's health by inciting them to get early treatment and diagnosis. Because the metrics in this study are very easy to obtain, a method of predicting diabetes using these attributes alone would be especially helpful to historically marginalized groups with little access to advanced medical institutions.

Census Income Dataset

This is subset of the data obtained from the UCI Machine Learning Repository, which itself is a subset extracted from the United States' 1994 census database found at www.census.gov. The dataset contains 5000 instances, each corresponding to one individual, each with 12 attributes and one binary class corresponding to income level. Thus, this data attempts to correlate attributes such as age, education, race, sex, and

native country to a discretized income class: “above50” if the individual earns more than $50,000 per year, and “below50” otherwise.

The ability to predict income level based on demographics has proven to be essential for the understanding of consumer behavior and producing target marketing. Income prediction models allow businesses and institutions to use customer attributes to tailor their marketing strategy according to the income class they wish to target. A dental franchise, for example, could use customer demographics to send toothbrush offers based on predicted income. Indeed, it may be futile and even offensive to offer a $400 electric brush deal to an individual who earns $20,000 per year. Accurate prediction of income would thus lead to more a consumer-sensitive marketing strategy.

Implementation

All clustering algorithms were implemented using the scikit learn machine learning library for python. In order to perform the various experiments, a utilities python file was created to house all functions used for reading in data, training models, testing for various k values, and generating graphs with the matplotlib plotting library.

Part one python scripts utilized sklearn.KMeans method, which implements Lloyd’s algorithm, and sklearn.GaussianMixture, which implements the expectation-maximization algorithm for fitting Gaussian mixture models. Once the datasets are applied, the clustering results are analyzed with silhouette and BIC scores, respectively.

Part two scripts implement principal component analysis, independent component analysis, randomized projection, and singular value decomposition. These feature reductions are performed using the sklearn methods PCA, FastICA, GaussianRandomProjection, and TruncatedSVD, respectively. Additionally, modules to analyze the optimal number of components to use were created, whose offspring graphs offer intuitive understanding of the best estimate.

Part three scripts take the take the feature-reduced data and pass it to the k-means and EM clusterers, whereby the analysis methods from part one can again be applied. Part four uses sklearn’s grid search to find the optimal hyperparameters for a neural network. The results are written to .csv files, whereupon the optimal numbers are extracted to run the sklearn.neural_network MLPClassifier. Similarly, Part five scripts implement the same NN classifier on the same data, but adding the clusters as new features. To analyze the effect of data size on accuracy, matplotlib is used to plot learning curves of training and cross-validation sets, along with a 1-standard deviation band to gauge variance.

Clustering Algorithms

The two datasets were applied to the k-means clustering and expectation maximization (EM) algorithms.

K-means Clustering

The k-means clustering algorithm takes a set of n samples, each with m features, and aims to partition the samples into k clusters such that the sum of distances from each point to their respective cluster-center is minimized. The within-cluster sum of squared errors can be interpreted as the measure describing how coherent the clusters are. However, since this inertia measure is non-normalized, Euclidean distances are often inflated in high-dimensional spaces (the curse of dimensionality.) Using dimensionality reduction techniques can often alleviate this. Therefore, experiments were carried out to assess the impact of feature reduction on clustering as well as on neural network performance.

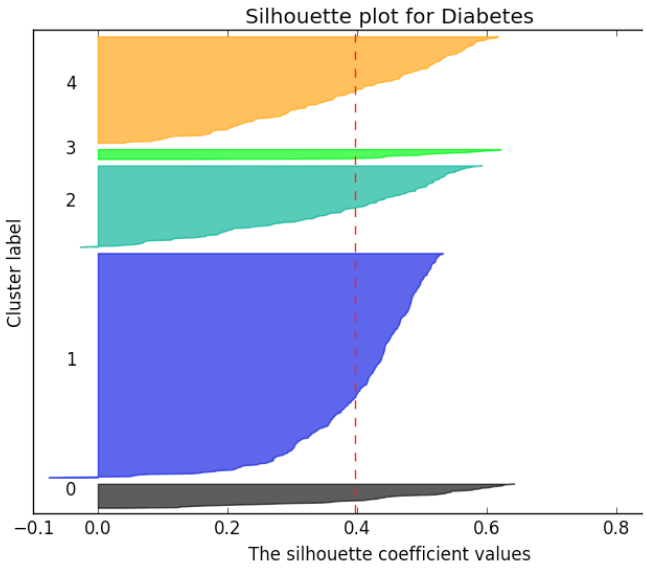
K-means clustering is an iterative algorithm. It first chooses initial centroids, and subsequently assigns samples to their nearest centroid. Next, new centroids are created using the average value of all samples assigned to their respective centroid. This process is repeated until convergence: the point where the difference between two iteration's centroids is less than a given threshold. A disadvantage of this clustering algorithm is that it may converge to a local minima. Therefore, it is recommended to run k-means several times to increase the likelihood of finding the global minimum. Given more time to perform experiments, I would have performed several iterations of k-means in an attempt to reach a global minimum.

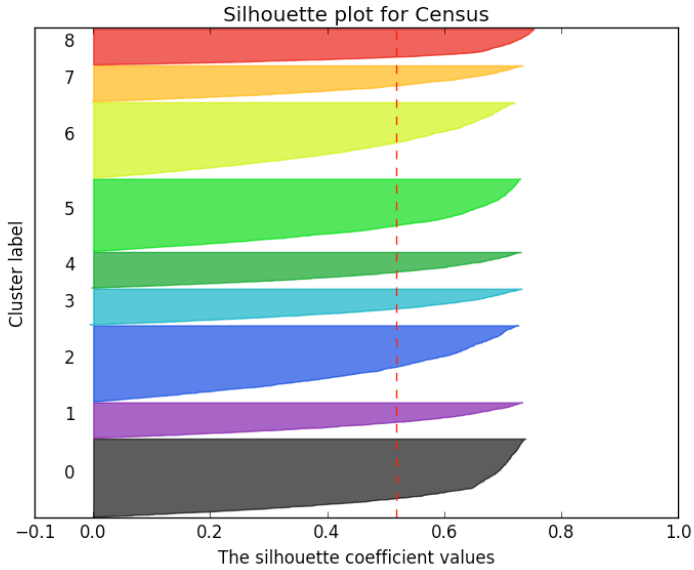
Experiments

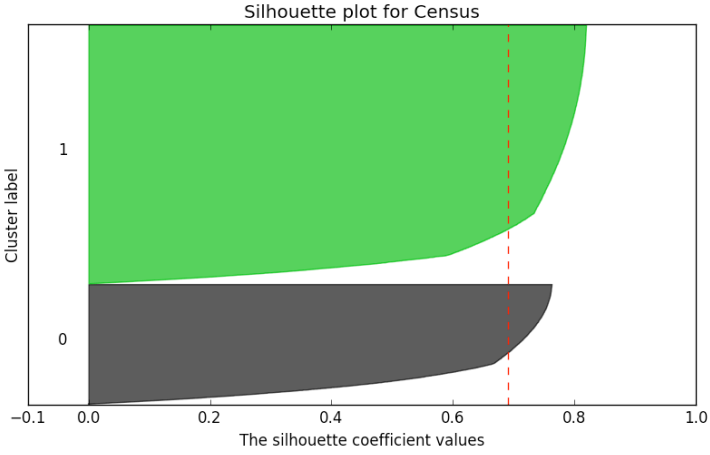
A key challenge in unsupervised learning is to learn the number of classes in a sample space. Hence, electing the appropriate number of clusters is an important decision when running k-means. To analyze the clusters resulting from this algorithm, a silhouette analysis is performed in each dataset.

[Average Silhouette Scores for n Components]

	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10
Diabetes	0.56	0.5104	0.4278	0.3969	0.3290	0.3592	0.3526	0.3386	0.2903
Census	0.69	0.5723	0.5804	0.5670	0.5437	0.5411	0.5213	0.5188	0.5134







A silhouette plot displays a measure of how close each point in a cluster is to points in the neighboring clusters with a value between [-1, 1]. A coefficient of 1 denotes that a sample is far from other clusters, where as a value of 0 indicates that the sample is inside or very close to another cluster. Therefore, the number of clusters that yields the highest average silhouette coefficient would be likely be the optimal number of clusters.

Looking at the Diabetes silhouette plots for k=5 (top left), we see an average coefficient score of about 0.4 and a large variation in the size of the cluster (each 'bar' has size equivalent to the number of samples in the cluster). Reducing k to 3, we notice a higher average silhouette coefficient of 0.51 and 2 clusters containing the grand majority of the samples. This hints at the fact that the third thin cluster is explains little of the variance in the data, an inference that is corroborated by the average silhouette score of 0.57 for

k=2. A similar phenomenon is observed for the Census Income dataset, where k=2 produces the highest average silhouette score of 0.69.

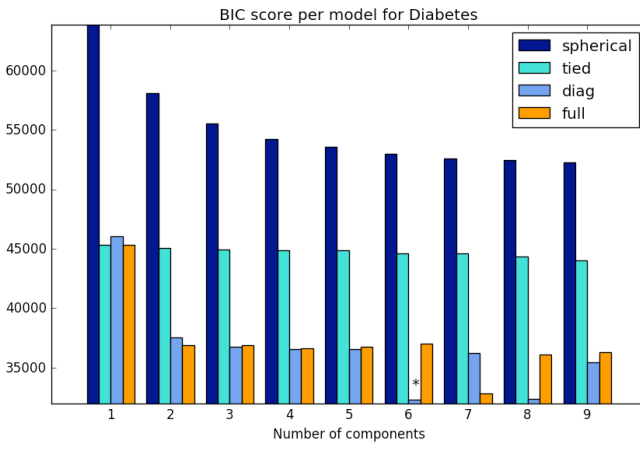
This finding implies that for both datasets, using two clusters maximizes the distance between samples and other clusters. We can tell that there exists overlap between the clusters because the optimal coefficients lie between 0 and 1. Looking at the shapes of the k=3 silhouettes for Diabetes, it can be seen that k-means generates 3 clusters , 2 of which contain about 90% of the data. This may imply that several of the features contribute little information gain-- an observation that was also noted by decision trees in supervised learning. There, we found that factors such as body mass index and number of offspring were closest to the leaves of the tree and therefore the least relevant in terms of information gain. At this stage, I would predict that performing feature reduction could eliminate said factors with little impact to the subsequent clusters.

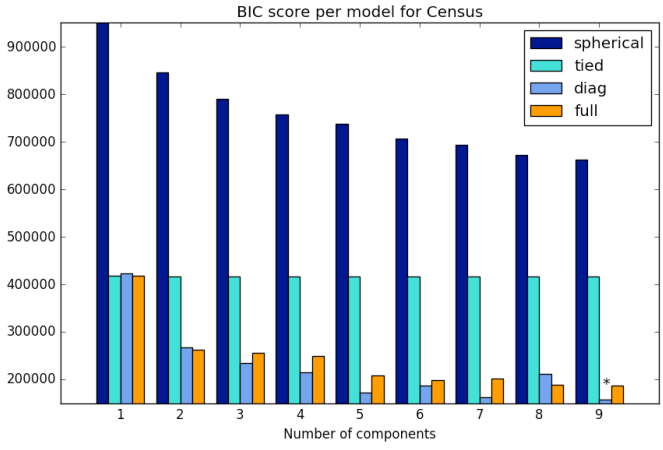
Expectation Maximization

The EM algorithm attempts to find the maximum likelihood hypothesis by seeking the hypothesis that maximizes $E[\ln P(Y|h)]$, the expected value of the log probability of the data given the hypothesis. The first step of the iteration creates a function to estimate this value using the current hypothesis and observed data to estimate the probability distribution over the entire dataset. In step 2, the hypothesis h is replaced by h': the hypothesis that maximizes the expected log-likelihood found in step 1. This process is repeated until convergence.

Similarly to other optimization algorithms such as gradient descent and and k-means, EM may converge to a local maximum. However, if there is a single global maximum, EM is guaranteed to converge to this point. Given more time to conduct experiments, I would run the EM algorithm several times to increase the likelihood of reaching the global maximum.

Experiments



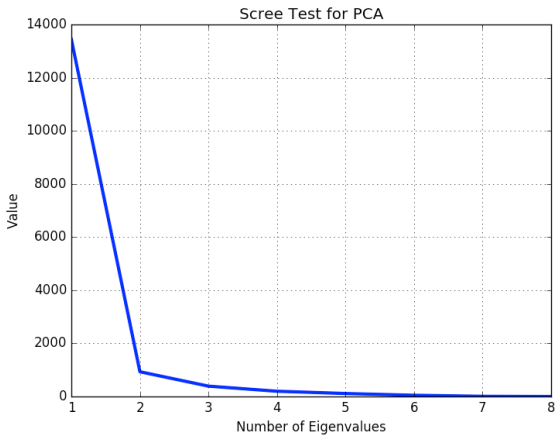


EM model selection analysis was conducted using the Bayesian Information Criterion (BIC). Selecting an optimal number of components (k) in EM is not a straightforward process. It is possible to increase likelihood by adding parameters, though this may come at the cost of overfitting. To attenuate this dilemma, BIC introduces a penalty term for the number of parameters in the model, where a higher number of parameters yields a higher penalty. Using various covariance measures and number of components, the model that produces the lowest BIC score yields the optimal number of components.
These experiments indicate that for the Diabetes data, the optimal number of components to use is 6, while for Census data the optimal number is 9.

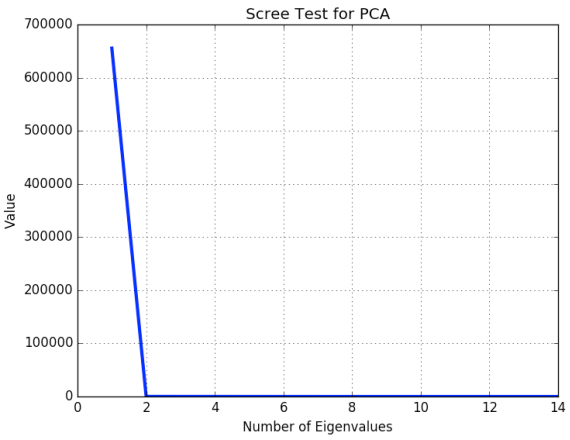
Dimensionality Reduction

The PCA algorithm aims to convert a set of samples consisting of potentially correlated features into a set of linearly uncorrelated features, or principal components. The resulting transformation is such that the principal components are ordered by their variance; the first component has the largest possible variance. The resulting quandary is selecting the number of components to extract, i.e. keeping only the most relevant factors.

Although this decision is somewhat arbitrary, it too depends on having domain knowledge. For the Diabetes data, for example, a feature measuring the height of an individual may be close to irrelevant in determining their diagnosis. Given my lack of knowledge about this particular domain, it is useful to perform a Scree test. This test plots the eigenvalues of each principal component and seeks to find a point along the curve where the decrease of eigenvalues appear to level off.



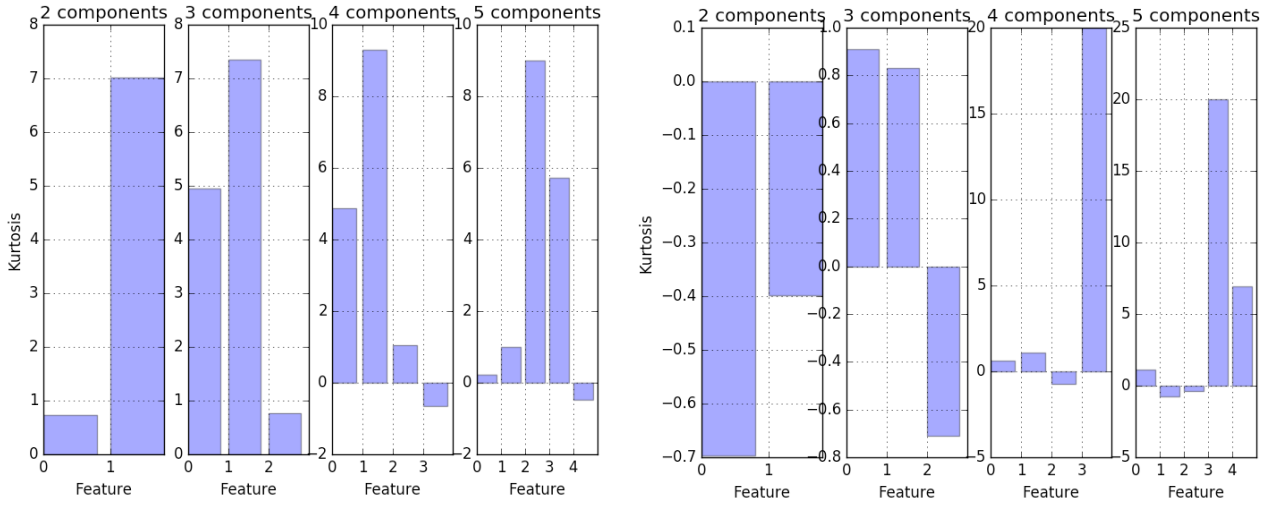
[Diabetes]



[Census]

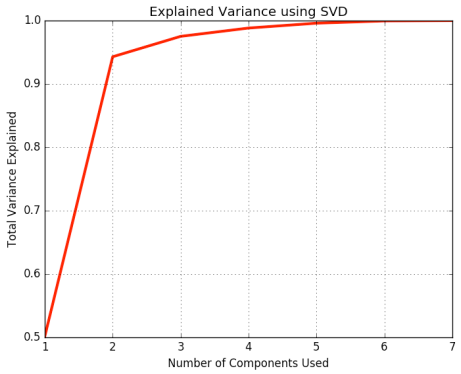
From both datasets, the scree test displays a sharp drop from the first to the second eigenvalue, after which the curve levels off. Therefore, selecting 2 components for each dataset is appropriate for PCA. The results for diabetes data is somewhat surprising; the metrics used as factors were presumably selected by individuals with some degree of domain knowledge. These results perhaps point to the fact that all attributes available were added to the dataset without any filtering process.

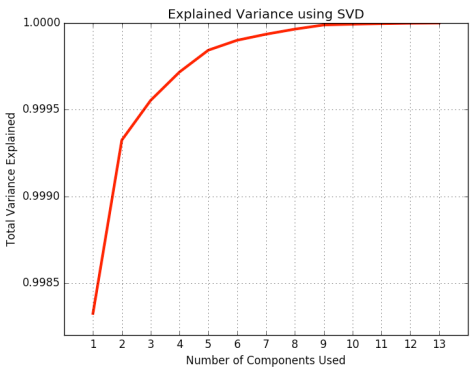
Independent Component Analysis (ICA) attempts to decompose multivariate feature space into independent non-Gaussian features. This is accomplished by assuming that components are statistically independent from each other. To select the appropriate number of independent components, we analyze the kurtosis of features after ICA is performed, and repeat this for various number of components.



The kurtosis for both ICA-reduced datasets seem to vary widely for all number of components used. In the Diabetes set, this may indicate that it is difficult to find any independence in the features, which fits with the prior domain knowledge that many health metrics are interrelated and sometimes influenced by a common source. Similarly in the case of the Census data, demographics are very often correlated and difficult to isolate as independent factors. These findings suggest that for these datasets, ICA does not create a meaningful projection.

In Singular Value Decomposition (SVD), we try to find the optimal number of components by measuring the explained variance for the full range of components.





In the Diabetes set (left), we see a sharp increase in the total variance explained when the number of components reaches two, where 95% of the variance is explained using just the first two components. This result is similar to the findings from PCA. Interestingly for the Census set (right), having just one component explains 99.85% of the data. This may be indicative of SVD's failure to explain the variance neatly described by PCA.

The implementation of Random Projection uses a computationally efficient way of reducing the data's dimensionality by trading a controlled amount of accuracy for faster wall-clock times and smaller model sizes. This feature reduction is applied to the data sets using the full range of possible components, and is repeated 15 times. The optimal number of components, as measured by reconstruction error of 0.674 and 0.708, is again found to be 2 for the respective datasets.

Clustering with Dimensionality Reduction

	Diabetes				Census Income			
	PCA	ICA	RP	SVD	PCA	ICA	RP	SVD
k (KM)	2	2	4	2	2	3	5	3
k (EM)	4	6	6	5	7	7	5	4
Silhouette	0.45	0.23	0.18	0.29	0.42	0.38	0.19	0.33
BIC	7,000	13,000	25,000	23,000	100,00	202,00	255,000	140,000

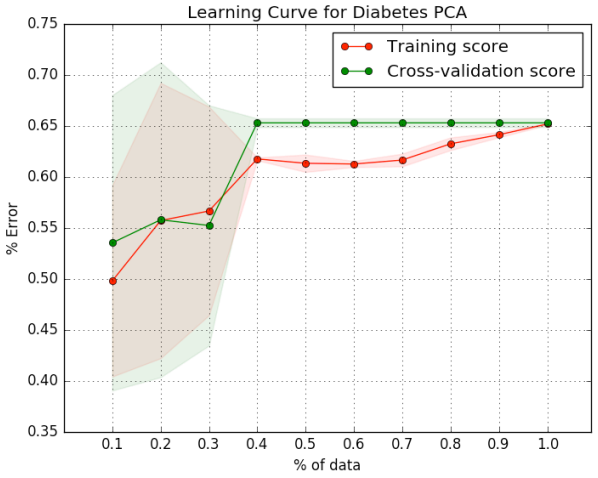
The number of clusters rendered by each feature reduction are very similar to the numbers found with the original untransformed data, especially in the case of k-means. In the case of EM, however, we noticed a drop in the number of clusters for Diabetes using PCA, and a respective BIC score similar to that found in the original clustering

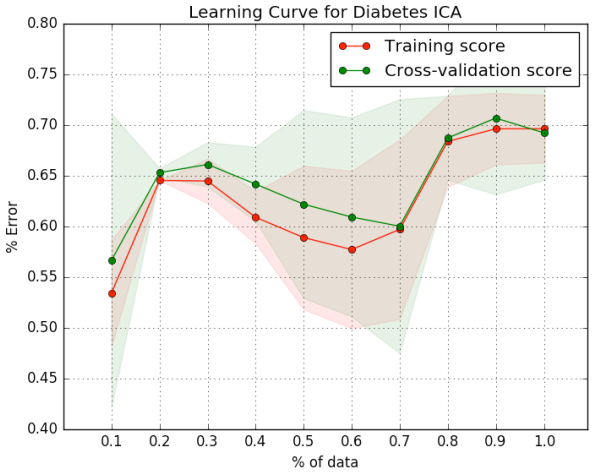
experiments. This suggests that PCA is an effective reduction technique for this data. While it does not earn as low of a BIC score, it reduces total data size and computations time.

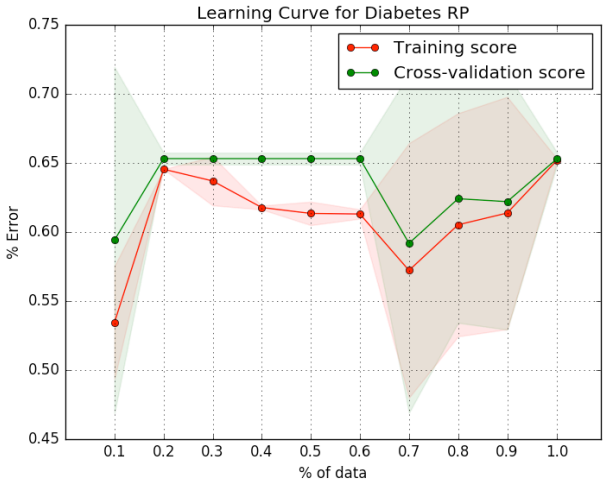
Neural Networks with Projected Data

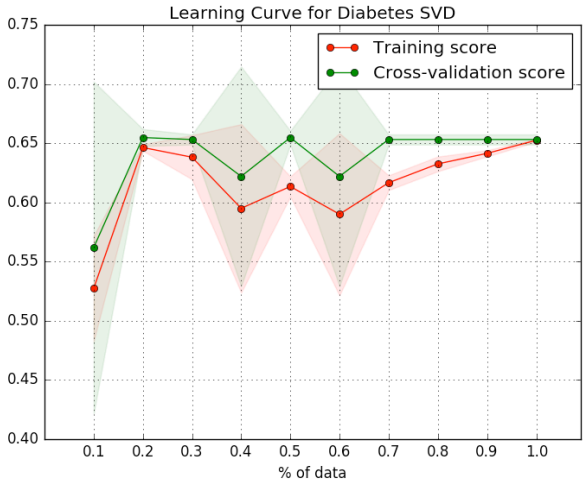
By running our previous NN learner with projected data, we venture to find whether the reduction in features was successful in addressing the curse of dimensionality and still achieving a comparable, or even better, accuracy score. To find the optimal hyperparameters for each NN, a grid search is conducted, resulting in a hidden layer configuration yielding best results. Using these, the NN learner can then be trained with each of the feature-reduced Diabetes datasets.

[Note: y-axes should read "% Accuracy"]
ICA proves to achieve the best performance, allowing the NN to reach an accuracy of 70%, while all other feature reductions led to an accuracy of 65%. Given that the







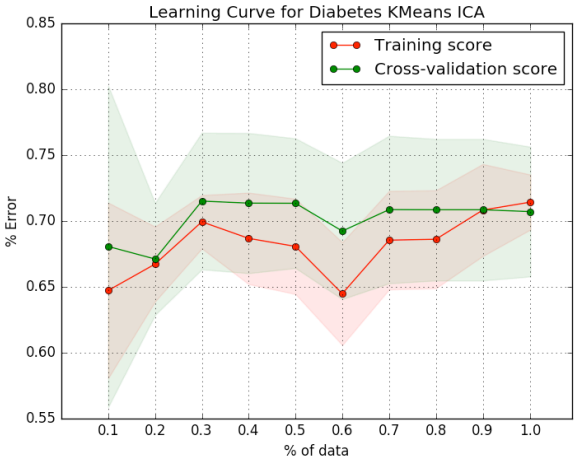


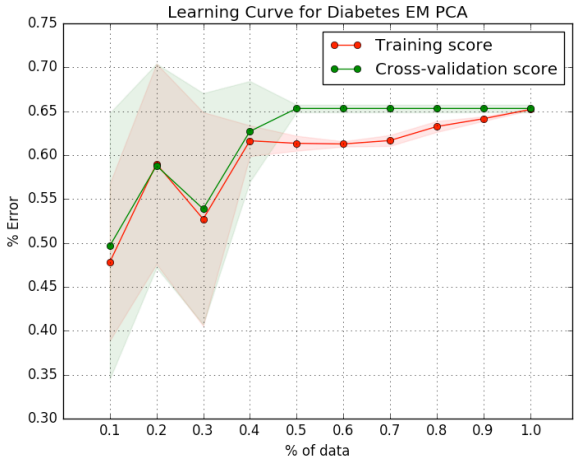
unreduced dataset produced a NN classification accuracy of 74%, it appears as though no features selection algorithm was able to produce such high results. However, it is worth noting that while the original NN learner took 6.9 seconds to run, the ICA-reduced version ran for 6.1 seconds. This difference, while seeming small, represents an 11% reduction, which could be save hours in a big data set.

Neural Network on Dimensionality Reduction with Clusters

Now that we have tested the effect of reducing features, we can test the effect of adding a feature, namely, the cluster labels as a new feature for each sample. If the clusters manage to group samples based on some underlying commonalities not described by any of the existing features, then this method may add information gain to the NN learner and perhaps even allow it to reach a higher accuracy.
[Note: y-axes should read "% Accuracy"]

Neither k-means nor EM were able to reach an accuracy as high as the vanilla original, though k-means using the ICA projection and clusters got close with an accuracy of





70%. Almost all other combinations
managed an accuracy of about 65%. Previous experiments showed that 2 features were responsible for explaining most of the variance in the Diabetes data set. Therefore, it appears that the cluster labels did not provide enough information about the label to make the NN perform better.