Utkarsh Garg
CS 4641 - Machine Learning
Assignment #3
4/15/2015

Unsupervised Learning and Dimensionality Reduction Report

Letter Recognition Dataset:

The dataset:

This dataset contains labeled examples of letters of the english alphabet, followed by a list of 16 attributes. It contains 20,000 instances which are divided into testing and training sets by the code. There are no missing values. The attributes (all integers) used are defined below:

> Horizontal position of box, Vertical position of box, Width of box
> Height of box, Total # on pixels, Mean x of on pixels in box,
> Mean y of on pixels in box, Mean x variance, Mean y variance,
> Mean x y correlation, Mean of x * x * y, Mean of x * y * y,
> mean edge count left to right, Correlation of x-edge with y,
> Mean edge count bottom to top, Correlation of y-edge with x

The number and nature of attributes seems very extensive; the attributes include original readings from the data and then outputs of statistical functions of the same readings.
Note: k-means was very slow to converge on the dataset since it had to form 26 clusters, so the problem was modified to that of a Binary letter data classification, where the two classes are:

> Non-round Alphabets: A,E,F,H,I,J,K,L,M,N,R,T,V,W,X,Y,Z
> Mostly Rounded Alphabets: B,C,D,G,O,P,Q,S,U

This was chosen because the attributes provided are either spatial locations, pixel locations or statistical inferences thereof. This poses an interesting problem because even though the output space has been reduced from 26 to 2, it might stile an extremely hard problem to solve.

Why it is interesting:

Having computers identify text from images is a very established machine learning problem with its complexity arising from the fact that there are tons of different fonts and even worse, many different handwritings that make it hard to identify a discernible pattern. Applying clustering methodologies on this problem is interesting because it gives an insight into how much accuracy basic algorithms can achieve on a legacy problem like this.

The problem is interesting from a machine learning point of view because, one the task is interesting, and two because of the attributes provided. Some attributes are spatial attributes on the box itself, while others are spatial on individual pixels. Using these attributes in the same space can be a challenge, and because no code in this paper tries to deal with the problem, a low accuracy is expected. The data cannot be all treated equally, simply because they do not fit in the same homogenous state space.

Splicing Dataset:

The dataset itself:

This dataset is used to identify two different types of splice junctions in DNA sequences; the exon/intron and the intron/exon junctions which are splices of DNA that are removed in protein formation. The dataset has been modified to eliminate useless information from the file(identification ids of each sequence). It now consists of a label (IE/EI/N), followed by a sequence of 60 characters, representing the DNA splice to be classified. There are 3175 instances, and no missing values.

Why it is interesting:

Because of the vast amount of computational complexity, I have been interested in the applications of machine learning to genomics and sequencing; also machine learning as a field itself has been gaining a lot of traction in this area. It has been interesting to see how these legacy algorithms, not designed specifically for this purpose perform on this dataset. From the perspective of this paper, the dataset has 60 attributes per instance and has only 3 possible classes; this poses an interesting question into the number of extraneous dimensions the dataset has, and how much the various algorithms chose to prune them.

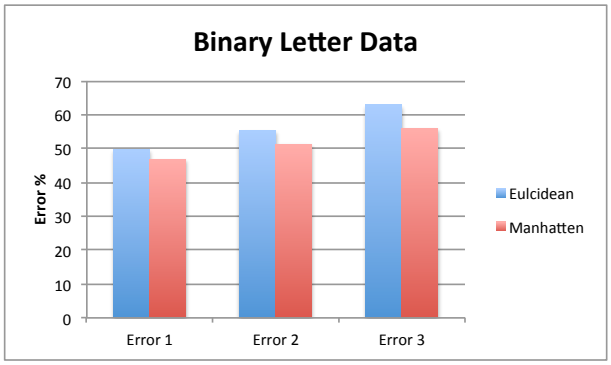
Clustering Algorithms:

k-Means: The k-means clustering algorithm aims to cluster all data points into k-clusters. It does this by initially computing k-clusters and assigning a cluster to each point, computing the center of the clusters (their mean), and then recomputing the clusters based on the new mean; the process is repeated until convergence.

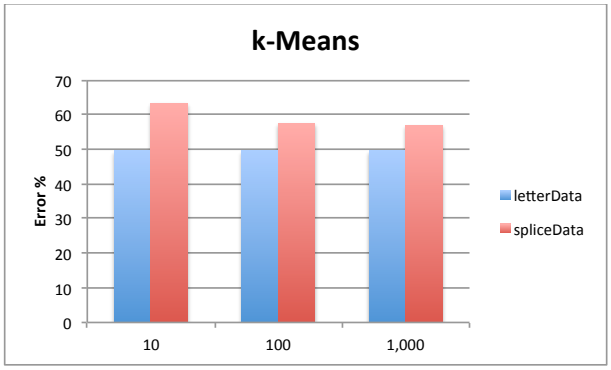
Expectation Maximization (EM): The algorithm calculates the expectation(likelihood of a point belonging to a soft cluster) and then updates the means with weighted values(weighted by probability), accordingly and reiterates. The process continues till convergence.

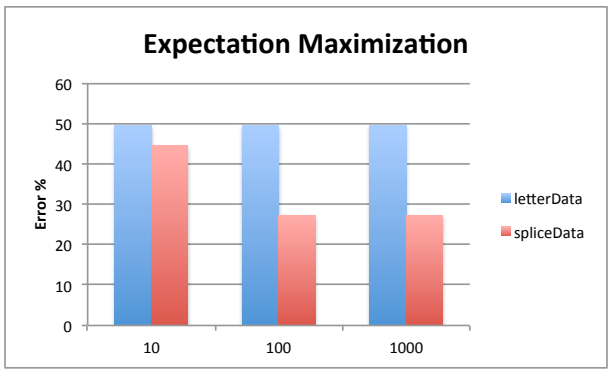
Experiment (k-means): Varying the distance function: The choice of the distance function of a clustering algorithm is integral to its performance. In this experiment, the clustering algorithm was run with Euclidean distance and Manhattan distance as its distance functions, varying all other parameters the equally.

Result: While for the binary letter dataset, the Euclidean distance always performed better than the manhattan distance, there was absolutely no effect on the splice dataset. This is a very clear example of how data-dependent the choice of a distance function is. Since the letter dataset had coordinates and means as attributes, it makes sense to have a Euclidean distance function, since a lot of the attributes are spatial.



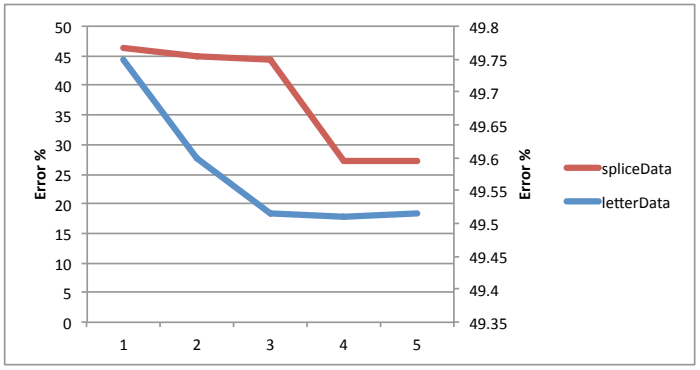
Experiment (k-means and EM): Varying the seed value: Varying the starting point of the algorithm should only effect its performance if it has a tendency to get stuck in a local optima, which both, k-means and EM have. However, given enough iterations we should expect to see a reduction in error as random restarts should average the error out. The seed values used (from left to right) are 10, 100, 1000.





Results: Changing the seed value has absolutely no implication on the accuracy of results for either k-means or EM in case of the letter dataset. The splice dataset however realizes a slight increase in accuracy in k-means and a large increase in EM. This points to the data more than the algorithms; it means that the distribution of data in the letter set is uniform enough for it to not matter, but the splice dataset has an inherent bias, which affects the ability of the algorithms to get out of the local optima.

Experiment (EM): Varying the improvement rate threshold: This threshold is responsible for convergence; when the difference between in the expected value of two successive iterations is less than the threshold, the algorithm has converged, and it stops. The results are displayed from threshold 1.0E-2 to 1.0E-10 in increments of 1.0E-2.

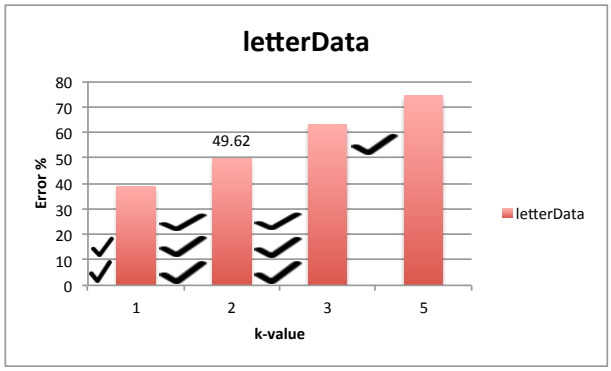


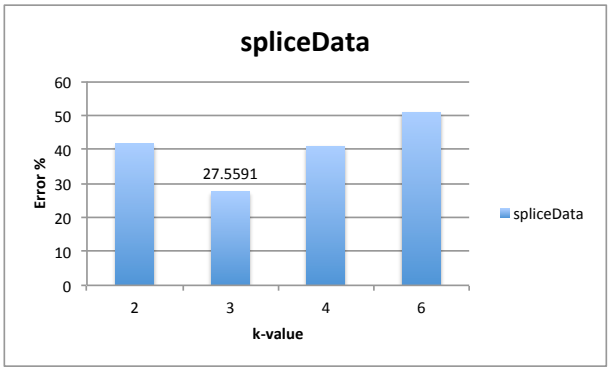
Results: For both datasets, as the threshold value decreases(lesser threshold => more required accuracy), the error percentage decreases until a certain point and then becomes non-decreasing. Both results are as expected and although the degree of variance is different for both datasets, the overall pattern is the same.

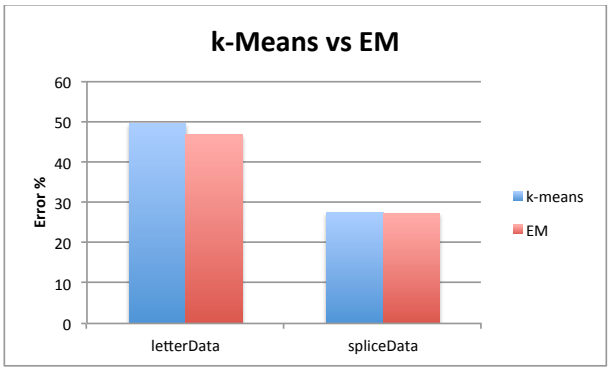
Experiment (k-Means): Varying the number of clusters: In this experiment, the number of clusters (the k-value) was varied. Since both datasets were labeled, the initial k was simply selected to be the number of labels, as that is the number of clusters the algorithm should ideally return. Then, k was set to 1 cluster more than the required clusters and then 3 more. For reference the required k for letterData is 2 and for spliceData is 3. The labeled data in the charts is the required value.

Results: As expected, error everywhere is much more than the error at the optimal number of clusters. There is one anomaly however. When we have only 1 cluster in the letter dataset, the error is lower than the error of the required number of clusters. This is because of the

distribution of the dataset; 61.225% of the data belongs to class A (there are only 2 classes), thus giving 61.225% correctly classified points.







k-means vs EM: As is evident from the graph, both algorithms have a significantly similar performance. Although EM does better than k-means on both datasets, it doesn't do much better; the inference is that the extra iterations performed by EM in this case are infact extraneous iterations and can be done away with. If k-means is allowed only 1 iteration, it does significantly worse than EM, but given enough iterations, its performance matches up to that of EM. This also signifies that k-means in this case is less prone to getting stuck in a local optima (since EM averages this problem out by an increased number of iterations, but that does not have any effect here).

Clustering Insights: K in k-Means was chosen to be equal to the number of classes the dataset had. This k performed expectedly better than all other k except for one anomaly discussed above. The clusters formed for letter data were not very accurate, and that was expected, as discussed in the description of the dataset; the clusters for the splice dataset on the other hand reached an accuracy of up to 73%. Changing the seed value and stance metric had dataset related effects and cannot be generalized from these experiments; varying k in k-means and the threshold in EM had a universal effect though, and can be generalized from this report.

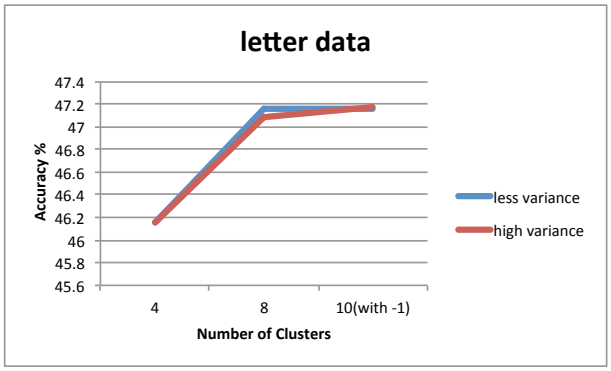
Dimensionality Reduction:

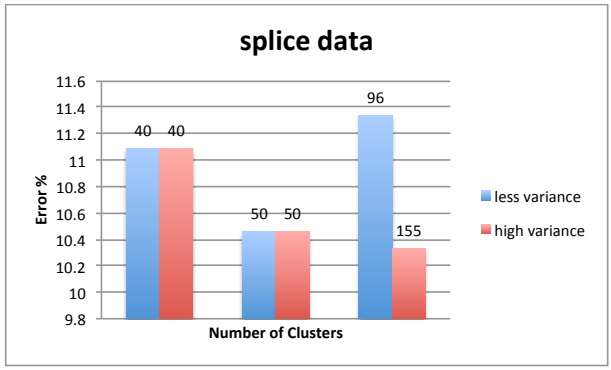
PCA(Principal Components Analysis): PCA is an eigenproblem that maximizes variance; it starts by projecting the data in the direction of maximum variance and subsequently finds mutually orthogonal directions, in decreasing magnitude of variance with respect to the data. Maximizing variance ensures that all variability and uniqueness is accounted for, thus capturing the most amount of information.

Experiment: Two parameters were varied; the variance threshold and the maximum number of clusters.The variance was varied between 0.8 to 0.95 and the number of clusters from the

default value(-1 => allow the algorithm to choose as many as it deems fit) and then values less than the original number of dimensions.

Results: For the letter dataset, the optimal number of dimensions that PCA found was 10 (original dataset has 16). This is a significant reduction of the feature space. Any further reduction however leads to information loss and thus loss of accuracy. For the splice dataset however, PCA does not reduce dimensions inherently. Consider the case with high variance (0.95); PCA ,with default settings (free to choose as many dimensions as it sees fit) actually takes the originally 60 dimensions to a staggering 155 dimensions. The reason for that I feel is that the original dimensions have no inherent correlation amongst themselves, so it seems like PCA has a lot of opportunity to introduce useful correlations. We can see that the accuracy with 155 dimensions is the maximum. However, when forced to reduce the dimensions to down 50, the algorithm performs just a little bit worse. The variance of the splice dataset for 50 attributes varies from 2.900209 to 0.697225. The first 25 eigenvectors have a variance>1 i.e. more than 50% of the new dimensions have a variance >1.





In the letter dataset, the variance ranges from 4.297329 all the way down to 0.419141. Again, 50% of the attributes have variance>1, but of those, 80% have variance 1 < variance < 2.56.

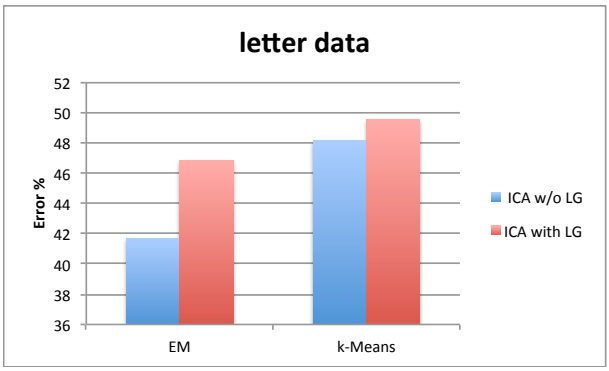
PCA arranges the eigenvectors in decreasing order of variance. This kind of prioritizing can be compared to probabilistic weighting in supervised learning algorithms; both give higher priority to more information gain (variance = 0 => 0 entropy => 0 information gain).

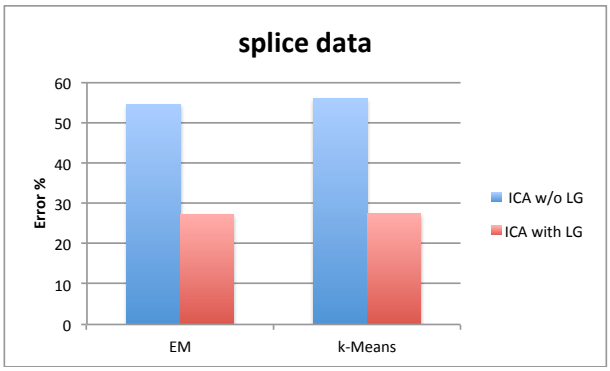
ICA(Independent Components Analysis): ICA tries to make a linear transformation from the current feature space a new feature space with the following two properties; all variables in the new space are mutually independent(there are no redundant dimensions) and all new features can be mapped to hold features (no/minimal loss of information).

Experiment: For both datasets, ICA was run with default settings, and clustering was performed without removing the near gaussian dimensions(LG). Then, the kurtosis of all attributes was calculated, and those with kurtosis~0 i.e. a near gaussian distribution were eliminated and the clustering was run again.

Results: The results for both datasets were very contrasting. While the letter dataset showed an increased accuracy after removing the near gaussian dimensions, in both k-Means and EM, the splice dataset showed a significant decrease. The attributes of the letter datasets are either spatial, pixel coordinates or statistical modifications thereof. This allows ICA to easily separate out the components and find the ones that may not be contributing. On calculating the kurtosis,

only one dimension had a near gaussian distribution and was removed, so there was not much loss of information. For the splice dataset on the other hand, each attribute is an element of he gene sequence; elements are not related to each other. This means that they already have 0 mutual information. Running ICA and then kurtosis gave 23 dimensions (out of 60), that were near gaussian. Removing these dimensions drastically reduced accuracy; this was expected. Since all dimensions were already mutually independent, removing any dimensions only reduces the information gain; it does not keep the information the same and does not lead to information gain, it leads to loss of information. ICA is not meant for problems with already uncorrelated data; it works well for problems like the blind source separation, where it decomposes a linear combination of dimensions to get the underlying, unrelated/independent dimensions. ICA would have worked better on the splice dataset, if the dataset had been modified to include as attributes, not the original attributes, but the attribute*(its place value). This would have introduced a correlation that ICA could potentially work on and probably produce better results.





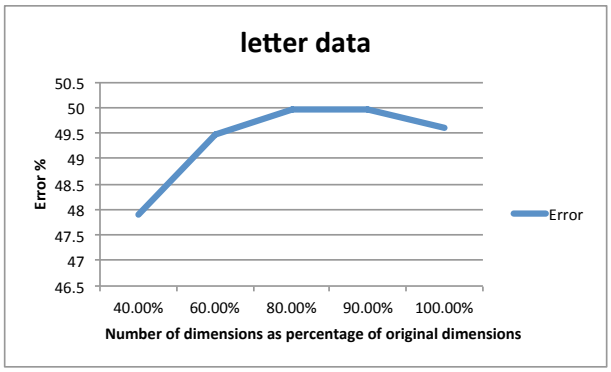
Random Projections (RP): This algorithm is like a random PCA; it projects the data into directions, but it does not have a metric for its choice, the direction is chosen randomly. Surprisingly, in general, random projections still manages to do well because it deals with the curse of dimensionality by eliminating dimensions while at the same time picking up some sort of correlation.

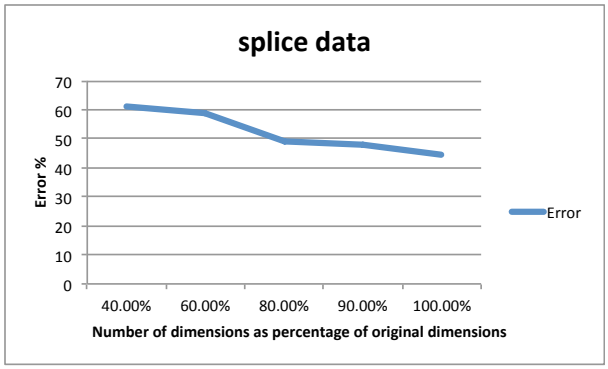
Experiment: Varying the number of final dimensions: In this experiment, the number of projections in the end was varied, as a percentage of the initial number of dimensions. The dimensions were varied from 40%, to 60%, 80% and 90% the number of original dimensions. The clustering was performed using k-means. Note: All results are presented as averages over 5 runs.

Results: Splice data gives very expected results. As discussed above, in its current format, the splice dataset attributes have no correlation to each other, they have 0 mutual information. Thus, reducing the number of dimensions only takes away useful information, making the results more inaccurate. The graph clearly shows decreasing error with increasing dimensions i.e. as data/information is recovered.

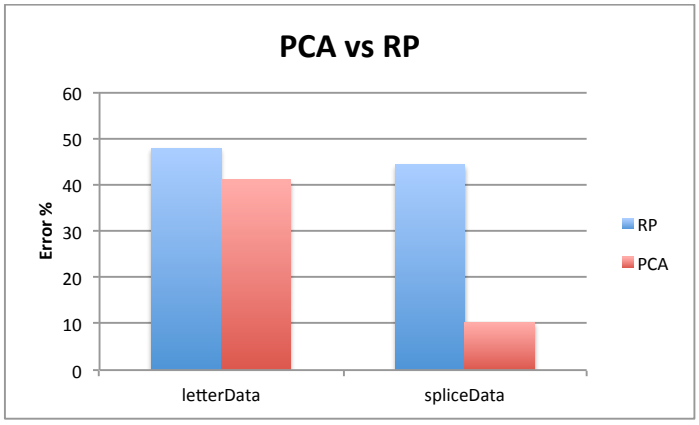
Letter data on the other hand, seems to benefit from the dimensionality reduction that RP gives it. It shows a significant reduction in error when using only 6 dimensions instead of the original 16. As the number of dimensions increase from there, so does the error, until it reaches a 100% and goes back to the original dimensions. This is a good example of how even randomized

projections can capture some correlation between the data, and eliminating seemingly unnecessary dimensions can actually be beneficial.





PCA vs RP: This section analyzes the performance variation in using a computationally complex algorithm such as PCA versus using a randomized approach (RP).



As is evident from the graph, PCA performs better than RP on letterData and significantly better than RP on spliceData. This is to be expected. Though Randomized projections is able to find some sort of correlation between the attributes, by projecting in random directions, the complexity of estimating the optimal dimension in PCA pays off. While PCA maximizes variance, which in turn increases information gain, RP projects randomly, reducing the dimensionality to the required size, thus reducing the amount of data required, while applying no heuristic to do so. Sometimes RP does well, other times it doesn't. On average however, RP performs better than no filter but always worse than PCA.

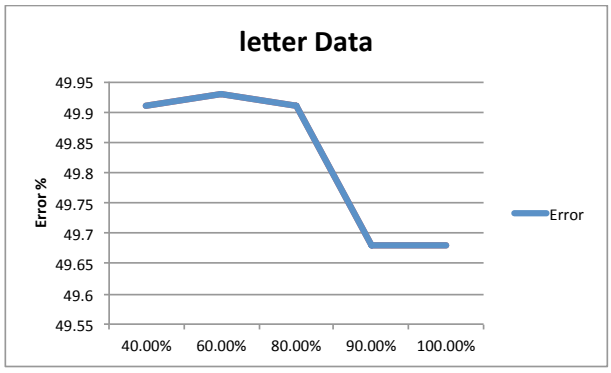
Random Subsets: This filter randomly picks a subset of attributes from the list of features. The size of the subset is to be specified by the user. I expect that randomly picking a small subset should give worse performance on both datasets. Picking a larger(80% / 90%) may on average yield better performance, as it may eliminate the very few attributes that are useless.

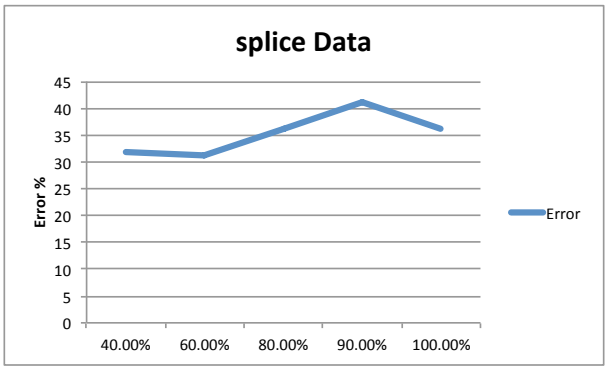
Experiment: Random Subset filter is applied to both datasets and then k-means clustering is performed. The number of attributes is varied as in Random projections (from 40% to 100%).

Results: The results for the letter dataset were as expected, picking a small subset costs information and this information loss less to a lower accuracy. As the size of the subset reaches close to the size of the superset, the accuracy stabilizes.

The splice dataset however returned very unexpected results. It returned better results for smaller subsets and worse for subset sizes close to the superset. As a reminder, there is no

correlation between any of the attributes in the dataset. This result means that some attribute values are more closely correlated to the output label/ correct classification of the dataset. This brings out a very interesting point; although there is no correlation between the attributes themselves, there seems to be some sort of correlation between certain attribute values and the classification/ gene type they belong to. The dataset did not explicitly state any such correlation in DNA splices, but it seems like some parts of the sequence are more important than others.





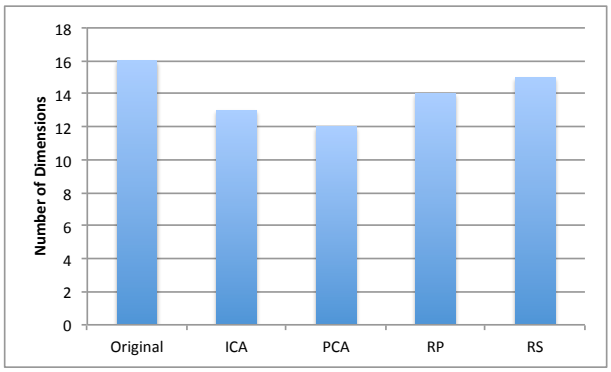
Dimensionality Reduction letter dataset: For the letter dataset, both PCA and ICA performed better than clustering the dataset without any dimensionality reduction. Both random filters performed worse however. The attributes of the letter dataset contained spatial attributes and statistical combinations of these attributes. In a sense, this dataset is a blend of the optimal dataset for both PCA and ICA. PCA tries to create correlations and increasing variance; the attributes are ideal because they are decomposed (basic) and can be combined without losing information. From the perspective of ICA, the statistically combined attributes are ideal. ICA tries to find hidden variables or the underlying attributes by decomposing the provided attributes. As i evident from the rest of the report too, this works well only if the provided attributes are actually linear combinations of underlying attributes, which is the case here. Neither PCA nor ICA perform exceptionally well, and that is simply because the dataset is a combination of the ideal datasets for both; it is not ideal for either one of them, only partially ideal.

Dimensionality Reduction splice dataset: For the splice dataset, only PCA improved performance. ICA, Random Projections and Random Subset all increased error. The reason for this has been discussed throughout the report and is simply that all attributes in this dataset are uncorrelated; thus picking small random subsets and projecting in random directions will do the accuracy no good. For the same reason, ICA does not perform well; ICA decomposes the given attributes to find underlying/hidden variables. In this dataset, the attributes are already in their most basic stage, removing any dimensions (ICA removed 23/60 based on low kurtosis) simply causes information loss. PCA performed well. By default, PCA tended to increase immensely the number of dimensions in the dataset and that can be attributed to the lack of any correlation. Because of the absence of any correlation, there was immense potential for it and PCA was able to exploit it. On the other hand, even when the number of dimensions in PCA are limited to 50, it performs significantly better than the no-filter clustering. This is also because all attributes are in the raw form, making it the perfect dataset for PCA, all linear combinations and complex attributes are made by PCA based on the same heuristic, in order to maximize variance.

Neural Net performance Analysis

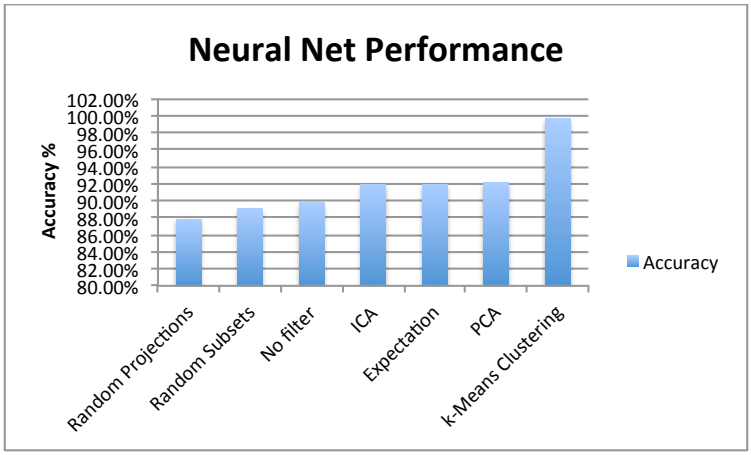
A multi-perceptron classification with 10 fold cross validation, learning rate of 0.3, alpha of 0.2 and 500 epochs was run after filtering the data with each of the dimensionality reduction algorithms discussed above. In addition, a cluster from k-means clustering and EM of the dataset was included as an attribute, while running the neural net. The results are tabulated below.

The neural net was run on the binary letter classification dataset. Both random filters (Random Projections and Random Subsets) performed worse than the neural net without any filter. PCA, ICA, Expectation Maximization and k-means clustering however performed significantly better.



PCA, ICA, RP and RS, all ran faster than the no-filter neural net. ICA and PCA ran much faster as they reduced the number of dimensions to 75 and 70 percent the original number of dimensions. RP and RS ran faster but only by 5-7%. The numbers are based on the dimensions for the optimal performance of the neural net for a particular filter. Rp was at 14/16 and RS at 15/16. Any more reduction using these filters led to a fall in the performance of the neural net.

In terms of clustering algorithms, the k-means cluster as an attribute performed much better than Expectation Maximization cluster in the Neural Network. At each step, expectation maximization updates its model based on the datapoints. After several iterations of EM with 10 iterations of k-means per iteration of EM, EM should relate very closely to a generalized datapoint (since it maximizes on the mean). In contrast, k-means is less correlated to the data, and as seen in the report, has a slightly poorer performance the EM. However, when the clusters are included as an additional attribute, k-means performs significantly better than EM. An explanation for this is overfitting. Because EM is more closely related to the datapoints, the neural net tends to overfit more than it does with k-means. Though both still perform much better than the data without the clusters, k-means performs exceptionally well because it relatively limits overfitting.



Filter	Accuracy
Random Projections	87.82
Random Subsets	89.115
No filter	89.755
ICA	91.945
Expectation Maximization	91.945
PCA	92.125
k-Means Clustering	99.61