Unsupervised Learning and Dimensionality Reduction

In this paper, I am to analyze the strengths and weaknesses of two clustering algorithms: k-means (KM) and Expectation Maximization (EM), 4 dimensionality reduction algorithms: Principal Component Analysis (PCA), Independent Component Analysis (ICA), Random Projections (RP) and Information Gain (IG) Attribute Evaluation. I will then analyze the effect of dimensionality reduction on clustering and classification on Neural Networks (NN). Finally, I will perform clustering on the dimensionally reduced data and use the clusters as new features for classification in NN. I have chosen 2 datasets from the UCI repository for my analysis: Wisconsin Diagnostic Breast Cancer (WDBC) and Seeds.

Wisconsin Diagnostic Breast Cancer (WDBC): I am reusing this dataset from assignment #1. This dataset extracts features from digitized image of a fine needle aspirate of a breast mass to describe the characteristics of the cell nuclei present in the image. The goal is to predict whether the diagnosis is malignant or benign given the mean, standard error and worst values for 10 attributes of the breast mass. Therefore, there are 30 features in total. There are 569 instances, of which 357 are benign and 212 are malignant. This is an interesting dataset as it contains a large number of attributes for a limited number of instances. It is further interesting as the features are statistical values for a group of cell nuclei; thus, I hope to gain some insights about the relationship between this when it comes to diagnosing breast cancer.

Seeds: This dataset describes measurements of geometrical properties of kernels belonging to 3 different varieties of wheat: Kama, Rosa and Canadian. The attributes are all real-valued and they are: area, perimeter, compactness, length of kernel, width of kernel, asymmetry coefficient and length of kernel groove. The goal is to predict the variety of wheat based on these attributes. Each variety has 70 instances; therefore, there are 210 instances in total. I chose this dataset because, unlike WDBC, this is a ternary classification problem. Unlike WDBC, this dataset also has significantly lower number of features. Therefore, it will be interesting to see if dimension reduction is powerful in problems with an already low number of features.

K-Means Clustering

KM is a simple yet powerful clustering technique. It partitions the data into k clusters to minimize the within-cluster sum of squares (WSS). A key property of KM is that it tends to find clusters of comparable spatial extent. We first start by finding the optimal k for the dataset based on 3 methods: (a) Elbow Method: pick k based on the bend (knee) in the WSS v/s k plot i.e. the point at which the gain in variance is marginal. (b) Average Silhouette Method: pick k with highest average silhouette. It is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette value ranges from -1 to +1, and high value indicates that the object is well matches to its own cluster and poorly matched to neighboring clusters. (c) Gap Static Method: it chooses k by using the output of any clustering algorithm (like KM) and comparing the change in within-cluster dispersion with that expected under an appropriate reference null distribution.[1]

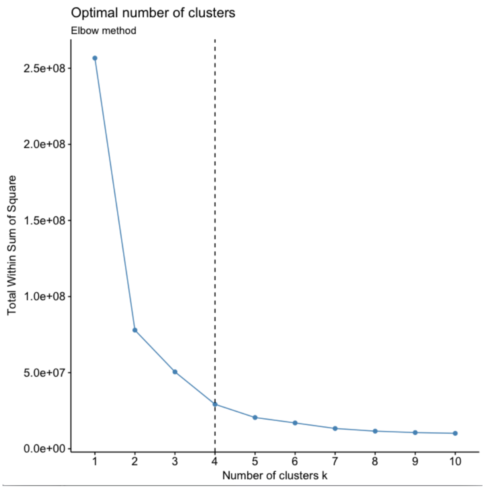


Fig 1: Elbow Method (WDBC)

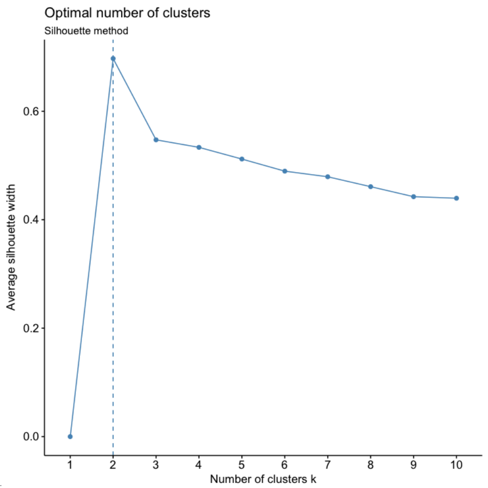


Fig 2: Silhouette Method (WDBC)

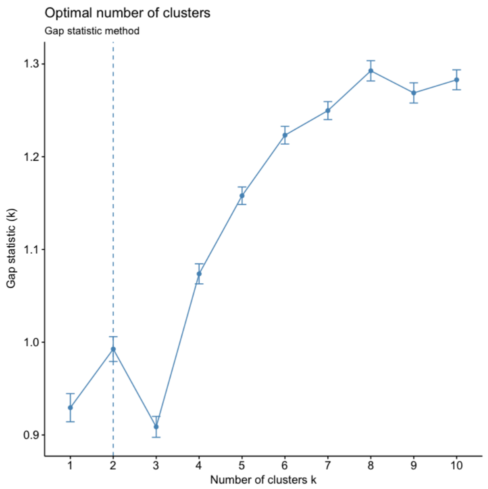


Fig 3: Gap Static Method (WDBC)

As shown in figures 1, 2 and 3, we see that the optimal number of clusters for WDBC is 4, 2 and 2 based on the elbow, silhouette and gap static methods, respectively. Similarly, I obtained the optimal number of clusters to be 4, 2 and 3 for the Seeds dataset by the elbow, silhouette and gap static methods, respectively. Figures 4 and 5 demonstrate the clusters obtained with k = 2, 4 using KM on WDBC. Based on Fig 4 when k = 2 on WDBC, cluster 1 captures malignant instances very well with very benign instances, while cluster 0 captures mostly benign instances with a few malignant instances. In comparison to when k = 4 on WDBC as shown on Fig 5, we observe that k = 2 performs much better. When k = 4, clusters 0 and 1 capture the malignant instances very well and clusters 2 and 3 capture the benign instances quite well. However, cluster 3 also includes a large number of malignant instances. Therefore, by comparing the k values, we conclude that k = 2 lines up naturally in WDBC while k = 4 doesn't line up as well. However, even with k = 2, although cluster 0 mostly signifies benign instances, it has a significant number of malignant instances as well. After analyzing the clusters obtained with respect to each feature, I observed that clusters obtained with certain features like Mean Fractal Dimension, SE Smoothness and SE Symmetry do not line up with the labels naturally i.e. values for each of these features don't differ for benign and malignant instances. For instance, we observe from Fig 6 that Mean Fractal Dimension has no impact on the label. This is an example of a feature that is not useful for classification on WDBC and demonstrates that WDBC could perhaps benefit from dimensionality reduction.

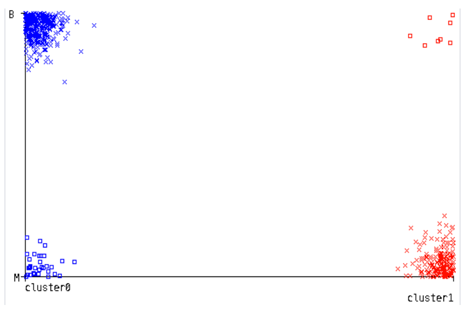


Fig 4: KM with k = 2 (WDBC)

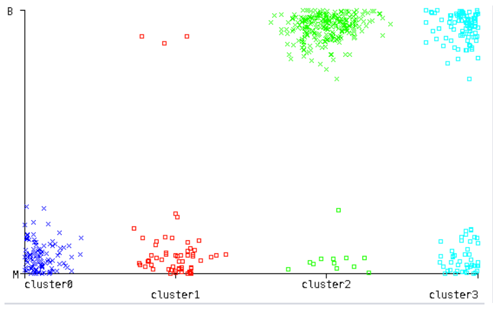


Fig 5: KM with k = 4 (WDBC)

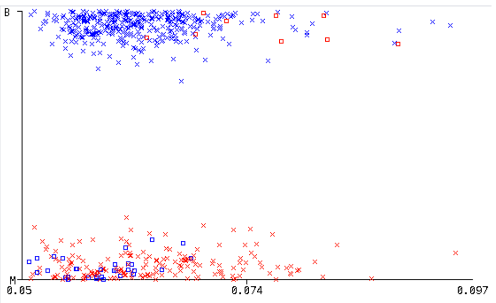


Fig 6: KM: Diagnosis vs. Mean Fractal Dim.

With respect to the Seeds dataset, we obtain the clusters using KM with k = 2, 3 as shown in Fig 7, 8. Of both values of k, we observe that k = 3 lines up most naturally with the labels. Cluster 0 captures Rose very well while cluster 1 captures Canadian very well with a few Kama and cluster 2 captures Kama very well with a few Rose and Canadian instances. Similarly, k = 4 also distributes Seeds reasonably well (not shown here). In contrast, k = 2 doesn't cluster the data as well since cluster 1 contains both Canadian and Kama instances. As we have already seen an example of what a bad feature looks like, we now look at what a good feature looks like. Fig 9 demonstrates that the clusters divide the dataset into the 3 labels based on continuous ranges of Perimeter; thus, this demonstrates that Perimeter is a good feature in Seeds. We see that Canadian has the smallest perimeter while Kama has next largest and Rose has the largest Perimeter of all 3. This is an example of a feature that we'd possibly want to be augmented by reducing the dimensions of Seeds.

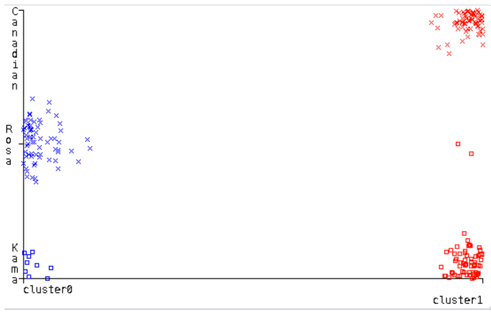


Fig 7: KM with k = 2 (Seeds)

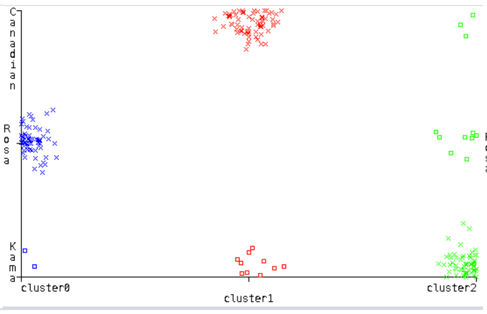


Fig 8: KM with k = 3 (Seeds)

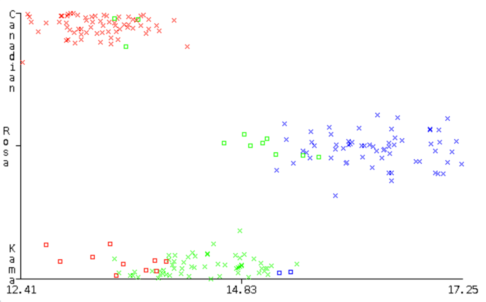


Fig 9: Class vs Perimeter with k = 3 (Seeds)

EM Clustering

The EM algorithm is an unsupervised clustering method based on mixture models. It follows an iterative approach, sub-optimal, which tries to find parameters of the probability distribution that has the maximum likelihood of attributes. For each iteration, we compute the E-step that estimates the probability that each instance belongs to a cluster and then the M-step, which re-estimates the parameter vector of the probability distribution of each class. The algorithm finishes when the distribution parameters converge.[2] In order to find the optimal number of clusters for EM, I calculated the Bayesian Information Criterion (BIC) of the 2 datasets. It is based on the likelihood function.

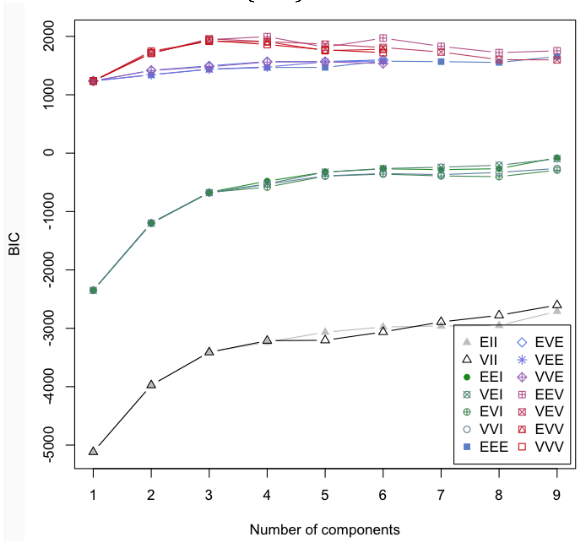


Fig 10: BIC score on Seeds

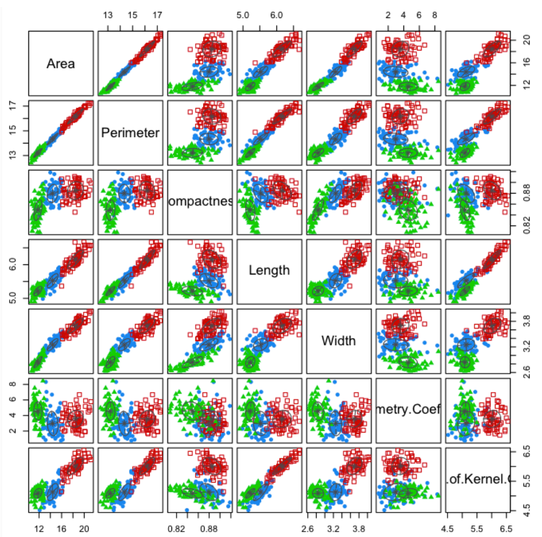


Fig 11: Seeds with 3 clusters (EM)

When fitting models, it is possible to increase the likelihood by adding parameters, but doing so may result in overfitting. BIC attempts to solve this problem by introducing a penalty term for the number of parameters in the model. The BIC obtained for Seeds is as shown in Fig 10.

Although the difference is minor, 4 clusters have the highest BIC score while 3 comes second. However, on plotting both clusters, I concluded that 3 clusters is indeed a better configuration for Seeds. The plot obtained is as shown in Fig 11. The worst clustering is formed between Compactness vs. Asymmetry Coefficient while all other pairs of features cluster quite well. Similarly, I calculated the BIC scores and obtained 2 to be the optimal number of clusters in case of WDBC. Ironically, the log likelihood with k = 2 is 7.14851 and with k = 4 is 10.91986 on WDBC. However, Fig 12, 13 clearly demonstrate that k = 2 forms better clusters than k = 4. Thus, an interesting observation here is that the higher likelihood value for k = 4 is a sign of overfitting in EM.

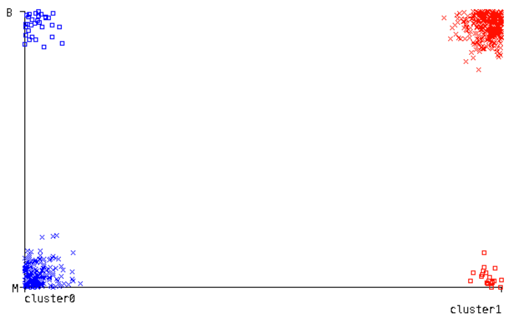


Fig 12: EM with k = 2 (WDBC)

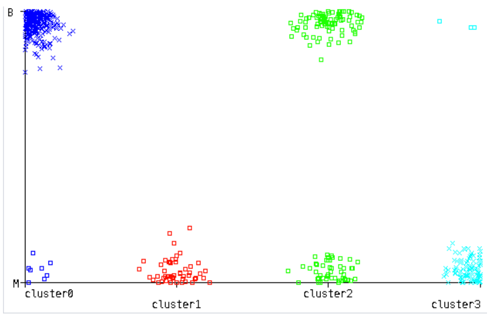


Fig 13: EM with k = 4 (WDBC)

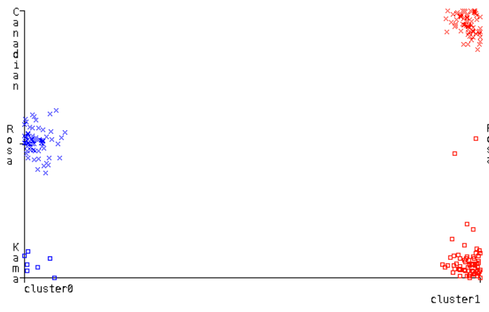


Fig 14: EM with k = 2 (Seeds)

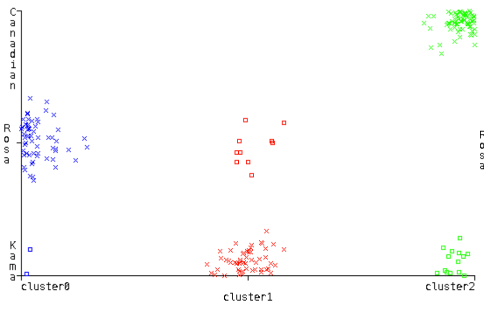


Fig 15: EM with k = 3 (Seeds)

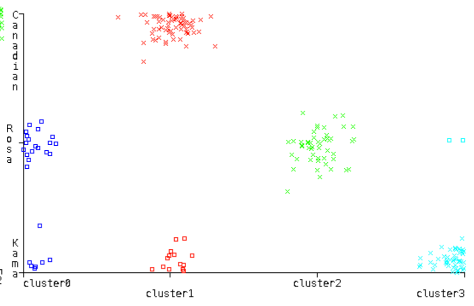


Fig 16: EM with k = 4 (Seeds)

I performed EM on Seeds with k = 2, 3, 4. We obtain log likelihood -2.47728, -1.05943, -0.51732 with k = 2, 3, 4. Here, k = 2 obtained the worst value, which confirms the poor clusters obtained as shown in Fig 14. On the other hand, k = 3 performed the best (Fig 15) although k = 4 came close (Fig 16). As in the case of WDBC, the log likelihood values obtained for each k value have not been fully successful in demonstrating the quality of clustering. This is, again, due to overfitting In EM.

Dimensionality Reduction: PCA

No. of Components	Eigenvalue	
	WDBC	**Seeds**
1	13.2816	5.0312
2	5.69135	1.19757
3	2.81795	0.678
4	1.98064	0.06836
5	1.64873	
6	1.20736	
7	0.67522	
8	0.47662	
9	0.41689	
10	0.35069	
11	0.29392	

Table 1: PCA Eigenvalues

PCA attempts to reduce the total number of dimensions by transforming a number of correlated variables into a number of uncorrelated variables called principal components. The first principal component accounts for as much of the variance in the data as possible, and each succeeding component accounts for as much of the remaining variance. Here, the goal is to look for features that allow us to reconstruct the original features as well as possible. The results of performing PCA on WDBC and Seeds are summarized in Table 1, Fig 17 (Seeds) and Fig 18 (WDBC). The eigenvalues measures the variance in all the variables that is accounted for by that factor. The ratio of eigenvalues is the ratio of explanatory importance of the factors with respect to the variables. In both datasets, we observe that the first 2 components have an overwhelming importance in comparison to the remaining components. In order decide how many components (n) to retain, we look at the variance explained by each n and corresponding eigenvalues. According to the Kaiser criterion, we retain components with eigenvalues ≥ 1. We further visually inspect the cumulative variance explained by each n and choose an optimal value based on the principle of diminishing returns.

For WDBC, we observe that n ≤ 6 have eigenvalues ≥ 1. Fig 17 confirms that n = 6 is a good choice as it captures about 90% variance. Any value of n > 6 suffers from diminishing returns. Therefore, we choose n = 6. On the other hand, n ≤ 2 have eigenvalues ≥ 1 on Seeds. As shown in Fig 18, n = 2 captures a cumulative variance of 89%. While n = 1 captures very little variance ~70%; therefore, we choose n = 2. One way to assess the quality of PCA is to consider the reconstruction error. I did this by reconstructing both datasets using the principal components obtained and taking the mean square difference between the projected data and the original data. I did this experiment using number of components 2 and 6 for WDBC as 2 represents the number of classes (benign and malignant) and 6 is the optimal number of components based on the eigenvalues obtained earlier. Similarly, I obtained the same for Seeds with number of components 2 and 3. The results obtained are shown in Fig 19 and 20. In case of WDBC, we observe that the MSE does not vary significantly between by increasing the number of components to 6 expect in case of Mean Texture, Mean Perimeter, Mean Area, SE Area, Worst Texture, Worst Perimeter and Worst Area.

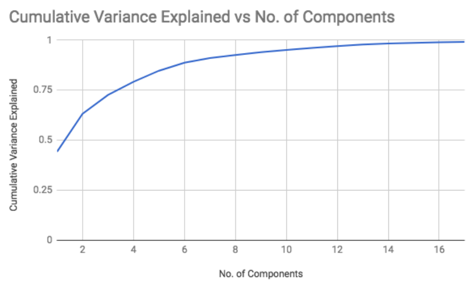


Fig 17: PCA on WDBC (Var. Explained)

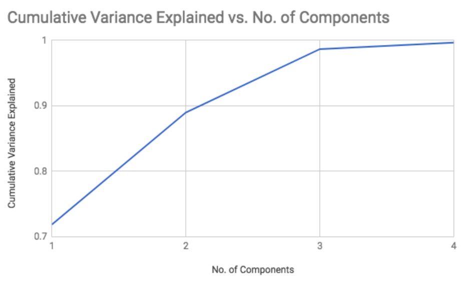


Fig 18: PCA on Seeds (Var. Explained)

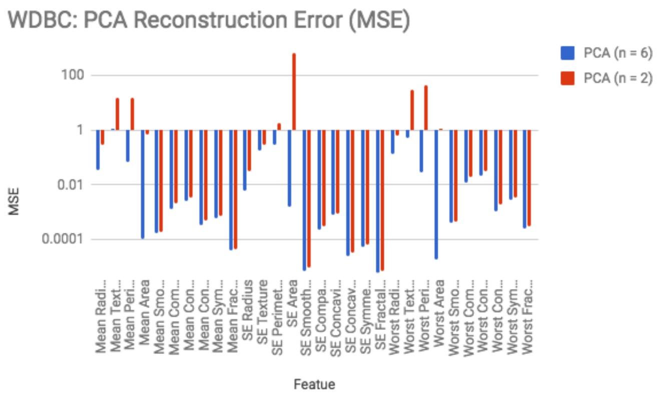


Fig 19: WDBC: PCA Reconstruction Error (MSE)

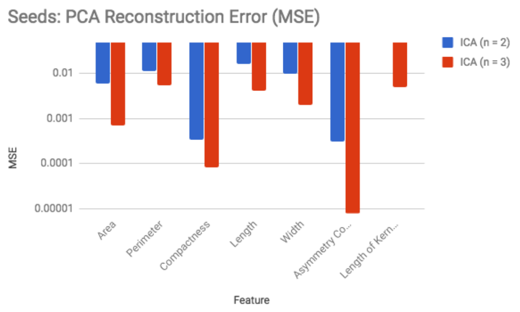


Fig 20: Seeds: Reconstruction Error (MSE)

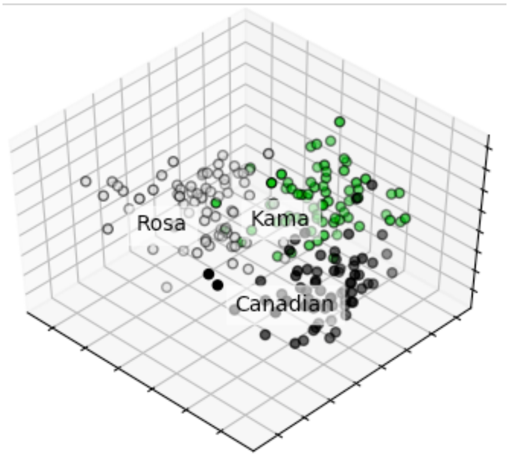


Fig 21: PCA on Seeds with n = 3

This is because of a handful number of outliers in these attributes. However, in general, when compared to the values in the original dataset, choosing 2 components in PCA projects the data sufficiently well. In comparison, the differences in reconstruction error differ by at least one order of magnitude for half the features in case of Seeds: Area, Asymmetry Coefficient and Length of Kernel Groove) for number of components 2 and 3. However, I observed that choosing 3 components reconstructs the data much better than 2 when compared with the original dataset. The projected data after PCA with 3 components in Seeds is obtained as shown in Fig 21 on the left. As seen, PCA projects the Seeds data with 3 components quite well as all 3 classes of wheat are separated into separate regions in a clean manner.

Dimensionality Reduction: ICA

ICA is a statistical technique for decomposing a complex dataset into independent sub-parts. It finds the independent components by maximizing the statistical independence of the estimated components. These independent components provide no mutual information to each other but maximize mutual information as a whole in the dataset; thus, these components can be used to reconstruct the original dataset. The first step of the algorithm is to whiten the data i.e. remove any correlations in the data. The separated signals are then found by an orthogonal transformation of the whitened signals. Unlike PCA, which uses covariance and is good at modeling Gaussian distributions, ICA requires non-Gaussian distributions with mutually statistically independent instances. In order to measure non-Gaussianity, we use Kurtosis and Skewness. Perfectly Gaussian distributions have a kurtosis of 3 and Skewness of 0. Fig 22 represents the log-scale kurtosis and skewness of WDBC distribution. We observe that most features have a kurtosis value higher than 4, implying that most features have heavier tails than a normal distribution. Very few features have kurtosis within the range 3 ± 1: these include Mean Radius (3.828), Mean Texture (3.741), Mean Perimeter (3.953), Mean Smoothness (3.838), Worst Radius (3.925), Worst Texture (3.212), Worst Smoothness (3.503) and Worst Concave Points (2.459) out of the 30 features in the dataset. Similarly, most features also have skewness greater than 1, although 4 of them have skewness ≤ 0.5: Mean Smoothness (0.455), Worst Texture (0.497), Worst Smoothness (0.414) and Mean Perimeter (0.491). Coincidentally, these 4 features also have kurtosis in the range 3 ± 1. Overall, the features in WDBC are sufficiently non-Gaussian to be suitable for ICA. For Seeds (Fig 23), on the other hand, Compactness (2.835) and Asymmetry Coefficient (2.907) have kurtosis closest to 3. On the other hand, Width has a skewness of 0.133, which is close to that of a Gaussian distribution's. Unlike WDBC, features in Seeds have lighter tails than that of a normal distribution.

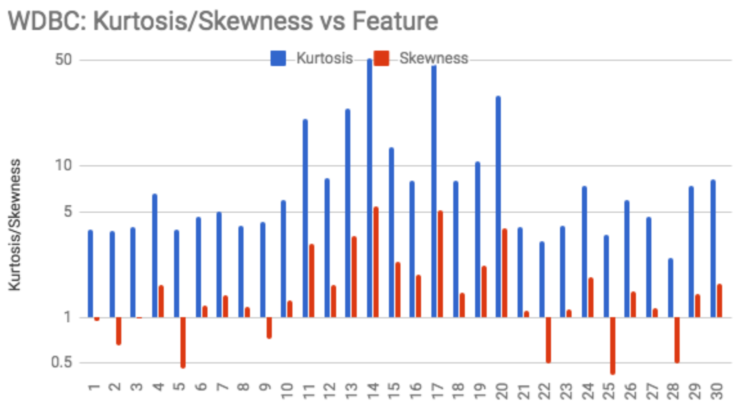


Fig 22: Kurtosis/Skewness vs Feature (WDBC)

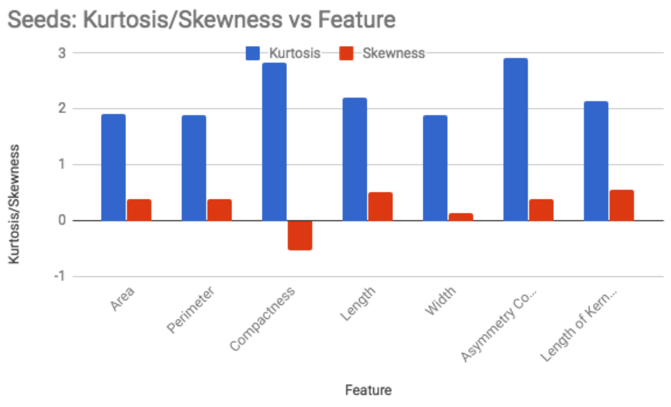


Fig 23: Kurtosis/Skewness vs Feature (Seeds)

An important aspect of ICA is to determine the number of independent components in the dataset. A possible consideration is the number of classes in the dataset i.e. 2 for WDBC and 3 for Seeds. However, we can also use PCA to determine the optimal independent components to be used i.e. 6 for WDBC and 2 for Seeds. I performed ICA on both datasets accordingly.

Dimensionality Reduction: RP

RP is the simplest dimensionality reduction algorithm discussed in this paper. It allows us to substantially reduce dimensionality of a problem while still retaining a significant degree of problem structure. In particular, given n points in Euclidean space, we can project these points down to a random d-dimensional subspace, where $d \ll n$. The reason behind the efficiency of RP is the Johnson-Lindenstrauss Lemma, which states that a small set of points in a high-dimensional space can be embedded into a space of much lower dimension in such a way that distances between the points are nearly preserved. Since RP is random in nature, it is important to consider its performance over a high number of iterations; thus, I ran RP on both datasets over 500 iterations. Unlike PCA and ICA, in which there are strategies to determine the optimal number of components to retain, it is hard to do so on RP and it depends on how well the data is reconstructed with the projected features. Therefore, I ran RP on both datasets with a varying number of components to retain and studied the average MSE between the reconstructed data and original data over each feature. The results obtained are shown in Fig 24 for Seeds. Similarly, I also calculated the average variance in the reconstructed data over each feature with varying number of components as shown in Fig 25. This is an important indicator as RP attempts to reduce the total number of dimensions in the dataset while still preserving its variance.

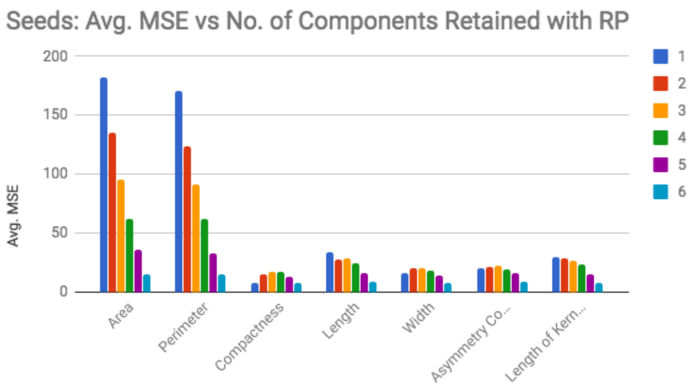


Fig 24: RP: Avg. MSE vs. No. of Components Retained (Seeds)

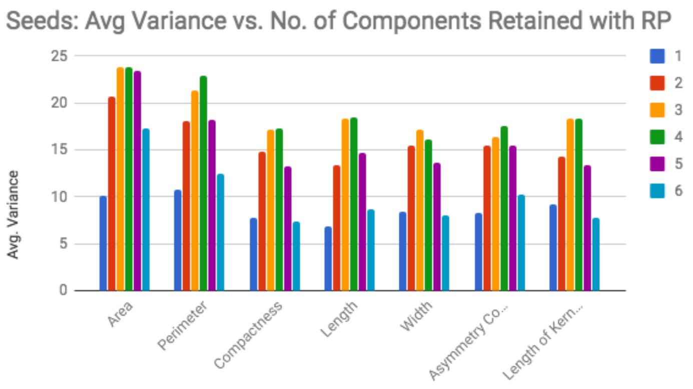


Fig 25: RP: Avg. Variance vs. No. of Components Retained (Seeds)

It is clear that a high number of components result in low MSE for Seeds. This is particularly true for certain features such as Area, Perimeter and Asymmetry Coefficient. In contrast, the average variance in each feature has no particular trend as the number of components increase. However, we obtain the lowest average variance across all features when the number of components is either 1 or 6. Based on the variance in the projected data, it is clear that n = 3, 4 are the best number of components to pick in case of Seeds.

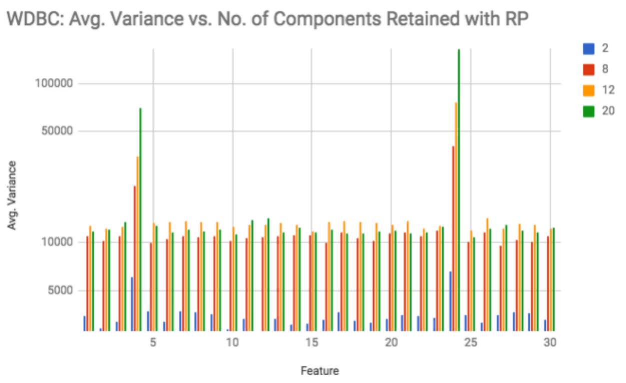


Fig 26: Avg. Variance vs. No. of Components Retained (RP)

RP is especially powerful in high dimensional problems when it is too expensive to compute principal components directly. Given that WDBC has a high number of dimensions, we are likely to benefit the most from RP in case of WDBC. This led me to study the average variances across features with varying number of components with RP on WDBC as shown in Fig 26. As expected, there is generally higher variance with a larger number of components. However, 12 components seems to provide reasonable variance for most features in case of WDBC.

Dimensionality Reduction: IG Attribute Evaluation

The core idea of IG is that we calculate entropy for each attribute in the dataset. A low value of entropy implies that the attribute contributes little information to the problem while a high entropy value maximizes information in the dataset. Those attributes that contribute more information are retained while others that do not add much information are removed. I performed IG on both datasets and obtained the results as shown in Fig 27 and 28.

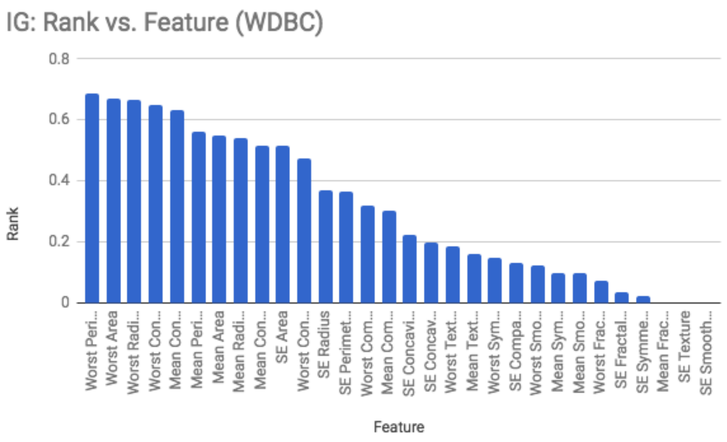


Fig 27: Info Gain: Rank vs. Feature (WDBC)

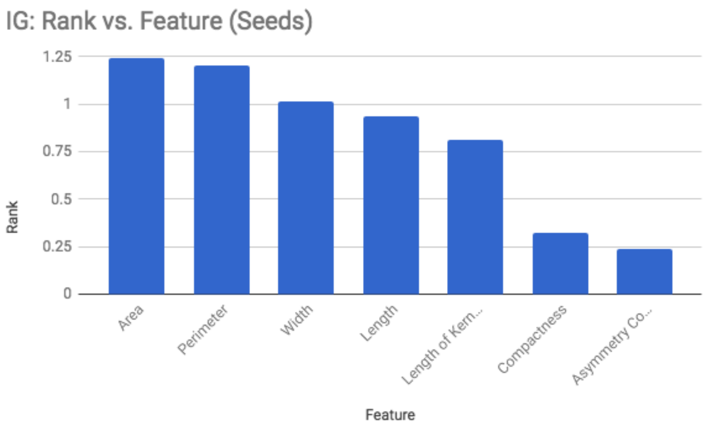


Fig 28: Info Gain: Rank vs. Feature (Seeds)

We observe that there are a considerable number of features in WDBC that contribute little to no information. These include: Mean Fractal Dimension, SE Texture, SE Smoothness, SE Symmetry, SE Fraction Dimension etc. At the same time, certain features seem to contribute most information; mostly, these are worst-type and mean-type features. On the other hand, all features except Compactness and Asymmetry Coefficient seem to contribute a significant amount of information in Seeds.

K-means with Dimensionality Reduction

PCA: I performed KM on the reduced dimensions obtained by performing PCA on WDBC with 2 principal components. The clusters obtained are as shown in Fig 29 a, b. As expected, PCA has significantly reduced the number of dimensions in the problem while yielding very good results in KM clustering. Fig 29a demonstrates that benign instances (light blue) cluster together in a tighter space; thus, it implies that benign instances are more alike while malignant instances are spread across a larger cluster. I then performed KM on the reduced dimensions obtained by PCA on Seeds with 2 principal and the clusters obtained are shown in Fig 30 a, b. From visual inspection, it is clear that KM clusters Seeds much better with PCA than without any dimensionality reduction. We also observe that all 3 clusters are quite equally spread out unlike the case of WDBC.

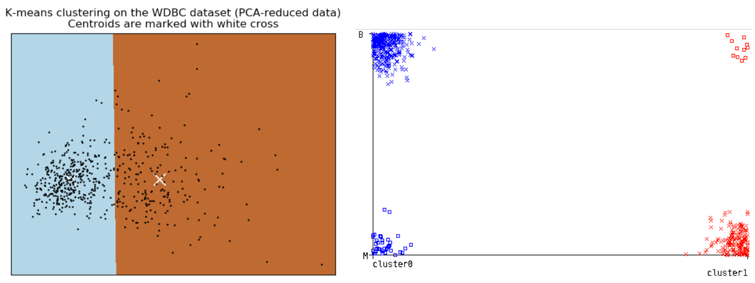


Fig 29 a, b: KM + PCA (n = 2) on WDBC

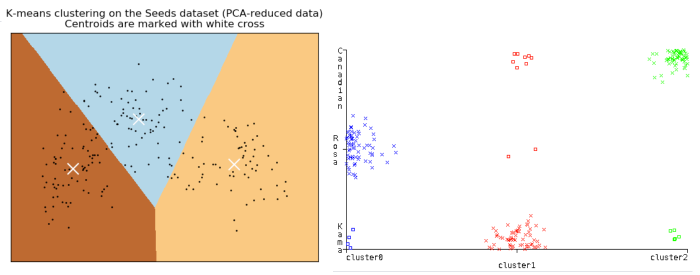


Fig 30 a, b: KM + PCA (n = 2) on Seeds

ICA: Unlike in the case of PCA, KM suffered when tested with ICA-reduced data on both datasets. I ran KM on projections obtained by ICA with a varying range of number of components on both datasets.

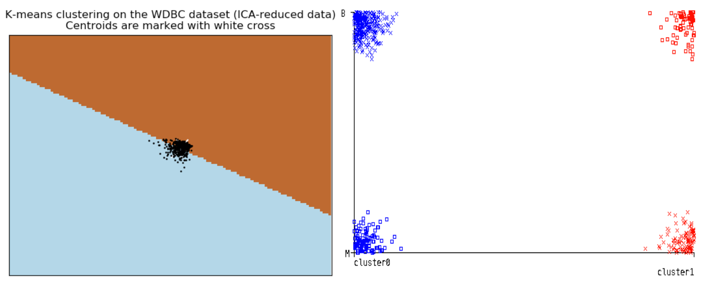


Fig 31 a, b: KM + ICA on WDBC

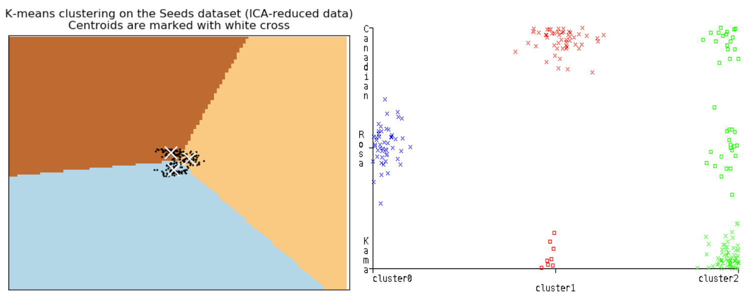


Fig 32 a, b: KM + ICA on Seeds

For WDBC, it was found that 15 independent components yielded the best possible clustering in terms of lining up with the labels while it was 6 independent components in case of Seeds. The clusters obtained for WDBC are as shown in Fig 31 a, b while those for Seeds are shown in Fig 32 a, b. In case of WDBC, it is clearly evident that ICA was unsuccessful in dividing up the instances into their appropriate labels. All instances are tightly projected into a small space as seen in Fig 31a. Both clusters had significant number of instances of both labels; thus, implying poor clustering. As the boundary between the malignant and benign instances is so small, the clusters don't accurately group instances by the expected labels. In case of Seeds, KM was relatively more successful with ICA than WDBC. However, it suffers from a similar problem in that all instances are projected into very tight spaces with very little

<u>RP</u>: I performed KM on the reduced dimensions obtained by performing RP on WDBC and Seeds with varying number of attributes to remain. The clusters obtained are as shown in Fig 33 a, b and Fig 34 a, b. I obtained the best clusters in terms of lining up with the labels with 6 projections on WDBC and 7 projections on Seeds. RP's performance on WDBC goes on to show the strength of RP in that we're able to reduce a 30-feature dataset into just 6 dimensions and obtain relatively good clusters. This proves that a many features in WDBC don't contribute much value to the dataset. In contrast, 7 projections on Seeds isn't really a reduction in dimensions as Seeds originally has 7 features. However, the quality of clusters in Seeds was comparable with number of projections ≥ 4. This demonstrates that certain features are not as critical in Seeds as we can obtain reasonably good clustering with fewer projections.

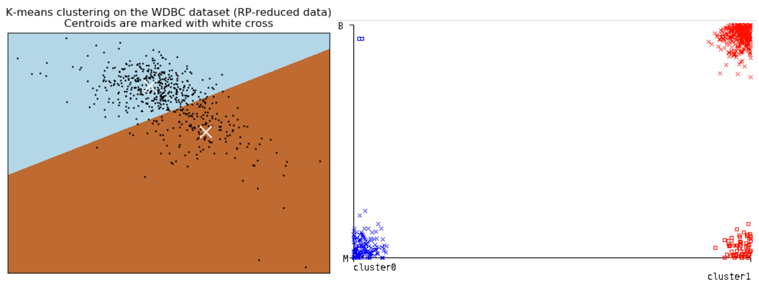


Fig 33 a, b: KM + RP on WDBC

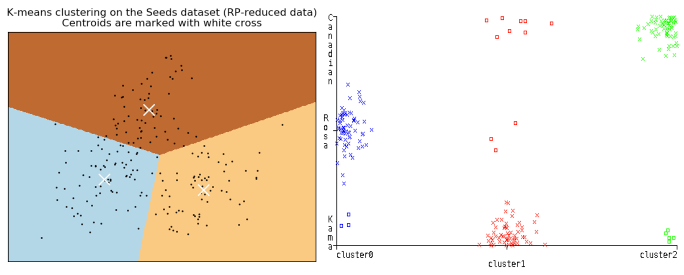


Fig 34 a, b: KM + RP on Seeds

<u>IG</u>: I performed KM on the reduced dimensions obtained by performing IG on WDBC and Seeds. I experimented this by removing several features by its worst ranks obtained through IG as presented earlier. The best cluster obtained on Seeds was with removing the 2 lowest ranked attributes: Asymmetry Coefficient and Compactness. The resulting cluster is shown in Fig 35. On WDBC, removing the 9 worst ranked attributes provided the best cluster (Fig 36a); however, it is worth noting that removing the 25 worst ranked attributes on WDBC provides a cluster almost as good (Fig 36b). This proves that IG has been very successful on WDBC as we are able to get rid of most features yet obtain better clustering.

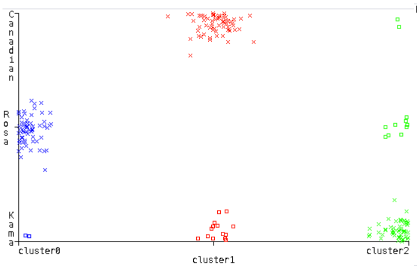


Fig 35 a, b: KM + IG on Seeds

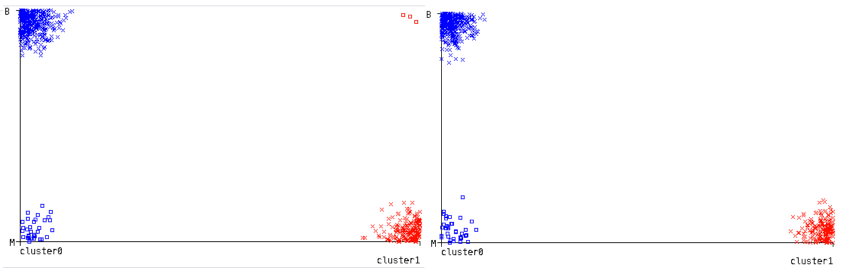


Fig 36 a, b: KM + IG on WDBC

Expectation Maximization with Dimensionality Reduction

<u>PCA</u>: I performed EM on reduced dimensions obtained by applying PCA to WDBC with 2 & 6 principal components as shown in Fig 37 a, b respectively. It is evidently clear that 2 principal components yield a better cluster than 6.

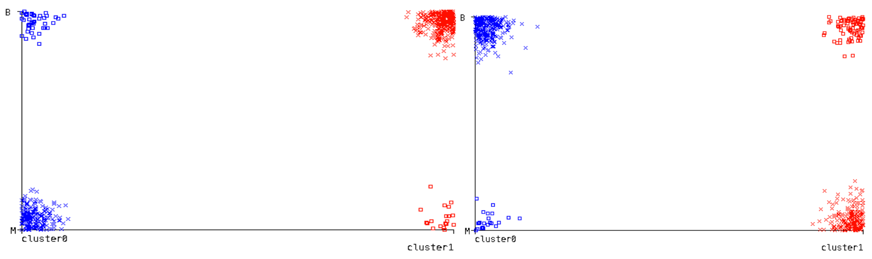


Fig 37 a, b: EM + PCA with 2 and 6 principal components (WDBC)

We further notice that the quality of clustering is relatively poor in comparison to KM with PCA. In fact, the quality of the clusters with or without PCA is quite similar in case of EM for WDBC.

In case of Seeds, I compared the quality of clusters obtained by EM with PCA using 2 and 3 principal components as shown in Fig 38 a, b. We observe that both clusters separate instances into corresponding labels reasonably well.

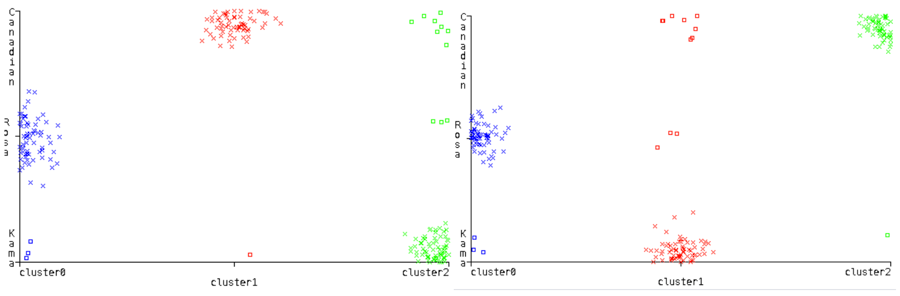
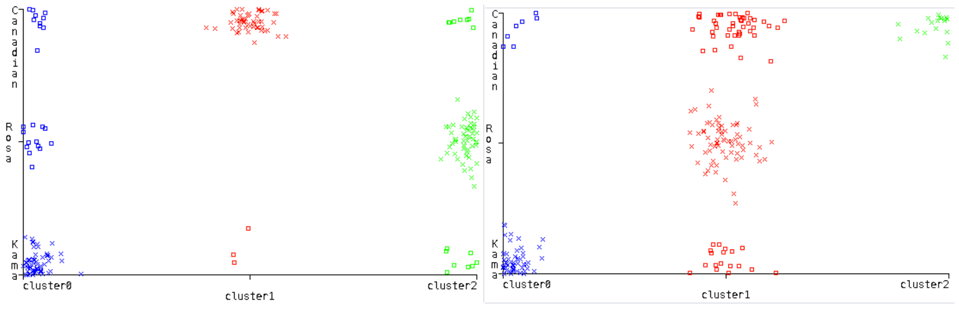


Fig 38 a, b: EM + PCA with 2 and 6 principal components (Seeds)

The quality of clusters obtained with EM and PCA is comparable to that of KM and PCA on Seeds. Given that both clusters are equally good, we choose 2 principal components for Seeds as we prefer fewer dimensions in the dataset.

ICA: I ran EM on projections obtained by ICA with a varying range of number of components on both datasets. In case of Seeds, it was found that 7 independent components yield the best clustering possible while 6 independent components came next as shown in Fig 39 a, b. It is evident that EM was not able to cluster the data well with ICA.



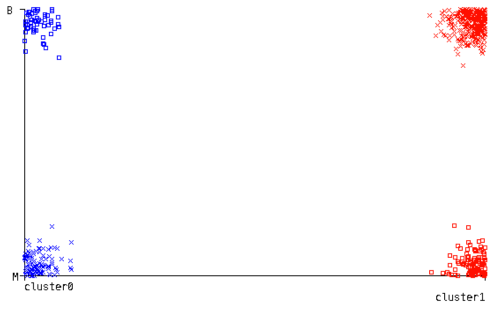


Fig 39 a, b: EM + ICA with 7 and 6 principal components (Seeds) *Fig 40: EM + ICA (n = 15) (WDBC)*

In fact, the quality of clusters obtained by EM without ICA is significantly better as shown earlier in this paper. The result obtained is not surprising as the kurtosis value for most attributes in Seeds were close to that of a Gaussian distribution. However, EM did not yield good clusters with ICA either as shown in Fig 40. This is because of the tight spaces that ICA projects the instances on with a very small margin between the 2 clusters. This leads to poor clustering and misclassifications. Hence, ICA has largely been unsuccessful in with both EM and KM.

RP: I ran EM on projections generated by RP with a varying range of number of projections on both datasets. In case of Seeds, it was found that 7 projections yield the best cluster while choosing 4 projections also results in a reasonably good clustering as shown in Fig 41 a, b. In both cases, most Canadian and Rosa instances associate with 1 specific cluster while Kama belongs to both cluster 1 and 2 in both configurations. On the other hand, we obtain reasonably good clustering with EM on WDBC using 6 projections with RP as shown in Fig 42. The quality of clustering with a higher number of projections (for example, 27) is better; however, the difference is minor. Therefore, RP with EM has largely been successful. It further demonstrates the strength of RP as we are able to vastly reduce the number of dimensions in a large dataset like that of WDBC and obtain good clusters. An important observation here is that RP is very suitable for high-dimensional problems.

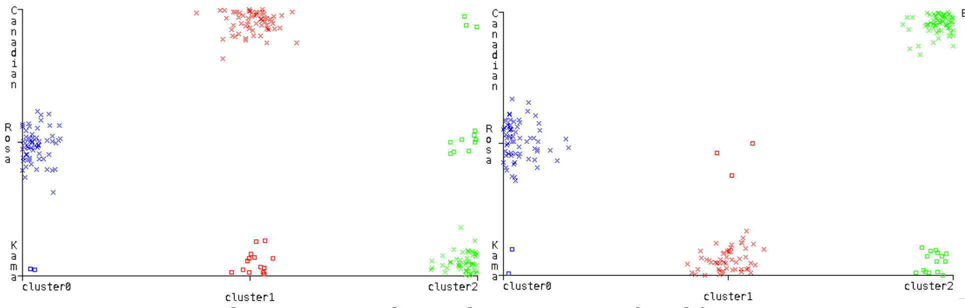


Fig 41 a, b: EM + RP with 4 and 7 projections (Seeds)

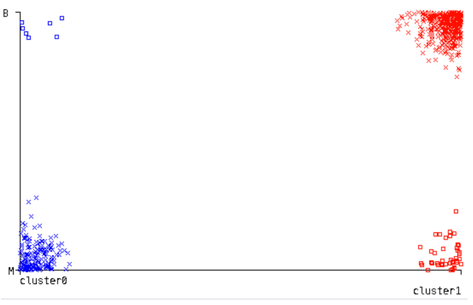


Fig 42: EM + RP w/ 6 projections (WDBC)

IG: Lastly, I ran EM on both datasets with reduced dimensions by eliminating the worst ranked attributes determined by IG. In case of Seeds, the clusters obtained by removing the two worst attributes (Compactness and Asymmetry Coefficient) led to a cluster that separates the data into the 3 labels quite well as shown in Fig 43.

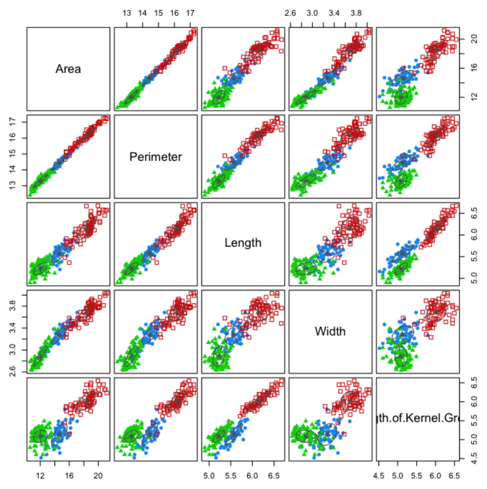


Fig 43: EM + IG (5 features) on Seeds

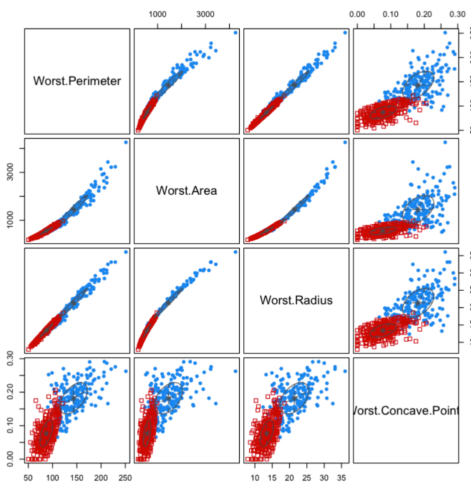


Fig 44: EM + IG (4 features) on WDBC

In case of WDBC, getting rid of all except the best 4 attributes obtained by IG (Worst Perimeter, Worst Area, Worst Radius and Worst Concave Points) led to the best possible cluster as shown in Fig 44. It is evident that the cluster cleanly splits the data by the two labels. This proves the immense advantage of using IG as a dimensionality reduction algorithm as we have successfully reduced 30-feature dataset into just 4 features with good quality clustering.

Effect of Dimensionality Reduction on Neural Network Performance: WDBC

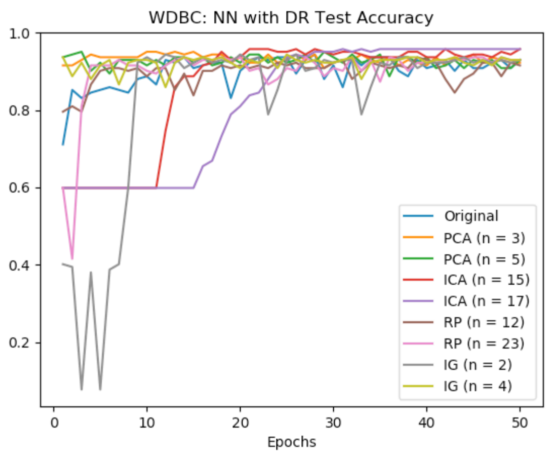


Fig 45: WDBC: NN Test Accuracy with DR

		Train Acc. (%)	Test Acc. (%)	Train Time
No Reduction		92.74	92.96	2.19s
PCA	n = 3	95.08	92.52	2.75s
	n = 5	95.55	92.95	2.91s
ICA	n = 15	98.59	**95.78**	5.42s
	n = 17	98.88	**95.77**	6.32s
RP	n = 12	93.44	91.55	2.19s
	n = 23	92.27	92.96	2.12s
IG	n = 2	88.01	92.96	2.19s
	n = 4	88.06	92.96	2.84s

Table 2: WDBC: NN with DR Statistics

Now that I have applied DR to all clustering algorithms, I would now like to study how DR really is in classification problems such as Neural Networks (NN). As I have already build a NN for WDBC in the first assignment, I will be reusing it. For PCA, I ran the NN with principal components ranging from 1 to 7, while for ICA I ran the NN with independent components ranging from 1 to 29, while for RP I ran the NN with random projections ranging from 1 to 29, as well. Finally, for IG, I ran the NN with an increasingly higher number of features in order of highest ranked features. For each of these, I ran the NN for a total of 50 epochs and used the optimal number of hidden layers for each setting obtained by a Cross Validation Grid Search. Of the best configurations in terms of accuracy for each of the 4 Dimensionality Reduction algorithms, I have plotted the top two performing ones in Fig 45 and presenting the train accuracy, test accuracy and training time for each. It is evident that DR has been largely successful in improving accuracy in WDBC. One that has outperformed the others is ICA, with above 95% test accuracy with both 15 and 17 independent components. Although the other algorithms have not significantly improved test accuracy, it able to achieve the similar levels of accuracy as that without DR; however, this is very beneficial as WDBC has a large number of features; therefore, using DR has helped reduce the size of the problem to a tremendous extent. In terms of training time, RP took the least amount of time while the PCA and IG took about the same time as the original NN without DR.

However, it is surprising that training time with ICA was about 2 to 3 fold higher than that of the other algorithms. This is largely because the number of independent components chosen was large (15 and 17) since I achieved highest test accuracy with these 2 configurations. In summary, ICA has performed the best albeit with a larger training time. However, PCA, RP and IG have also done significantly well given the amount by which it has reduced the dimensions of the problem.

Effect of Dimensionality Reduction and Clustering on Neural Network Performance: WDBC

For this section, I will study the performance of the NN in case of WDBC when applying both clustering algorithms KM and EM along with DR algorithms PCA, ICA, RP and IG. Based on the optimal configurations obtained in the previous section, I ran PCA with 3 and 5 principal components, ICA with 15 and 27 independent components, RP with 12 and 23 random projections and IG with the top 2 and 4 features. For KM and EM, I used 2 clusters as we have seen in the earlier sections that 2 clusters is optimal in case of WDBC. For each of these DR-Clustering combinations, I ran the NN for a total of 50 epochs and used the optimal number of hidden layers for each setting obtained by a Cross Validation Grid Search. The results obtained are summarized in Table 3 given below.

			Train Acc. (%)	Test Acc. (%)	Train Time
No Reduction/Clustering			92.03	92.96	2.84s
PCA	n = 3	KM	93.68	90.85	**2.04s**
	n = 5		95.55	92.25	2.97s
	n = 3	EM	94.15	**93.66**	3.01s
	n = 5		97.66	90.14	3.72s
ICA	n = 15	KM	99.06	95.07	**6.81s**
	n = 17		99.29	95.07	6.12s
	n = 15	EM	82.90	82.39	5.29s
	n = 17		99.06	**95.77**	6.81s
RP	n = 12	KM	92.74	90.85	2.68
	n = 23		92.97	**94.37**	2.73
	n = 12	EM	91.57	90.14	2.34s
	n = 23		87.35	87.32	2.51s
IG	n = 2	KM	88.05	**92.96**	2.72
	n = 4		88.76	92.96	**1.74**
	n = 2	EM	86.42	92.25	2.58s
	n = 4		83.61	88.03	2.38s

Table 3: WDBC: NN with Clustering + DR Statistics

Generally, across all experiments, we see that EM outperforms KM for WDBC in terms of test accuracy. It is certainly true that choosing a good combination of clusters for EM/KM and number of projections for DR algorithms leads to better test accuracy than without clustering or DR. Through the results obtained, we notice the importance of choosing the right configuration as ICA (n=15) with EM only resulted in 82.39% test accuracy while with ICA (n=17) with EM, we obtain the highest test accuracy amongst all experiments conducted ~ 95.77% test accuracy. While KM utilizes a hard assignment partitioning for clustering in that it is certain as to which cluster a point belongs to, EM uses soft a soft assignment in that it assigns probabilities of an instance belonging to a cluster. This helps EM express uncertainty. Another difference between the two is that EM is able to cluster non-linear geometric distributions while KM assumes clusters to be spherical. Given the WDBC clusters discussed earlier in the paper, it is understandable that EM performs better as the data doesn't fit well into spherical clusters.

An interesting observation is that IG has the least amount of training time. This is perhaps due to the extremely low number of dimensions it reduces the dataset to in this problem (only 2 and 4). In contrast, ICA takes the longest amount of time. However, this is because of the large number of independent components used. However, it also provides the highest test accuracy in this experiment, which might be a good reason to justify the higher train time. It is further surprising that RP with KM performed the second best with 12 projections at 94.37% test accuracy. One reason RP performs so well in this problem in comparison to the other DR algorithms is because of the large number of dimensions in the problem. Lastly, I am presenting the learning curves obtained for both cases in Fig 46, 47. In case of EM, it is clear that ICA performed the best on WDBC. In case of KM, RP performed the best. It is noteworthy that ICA performed very well on WDBC both with and without clustering.

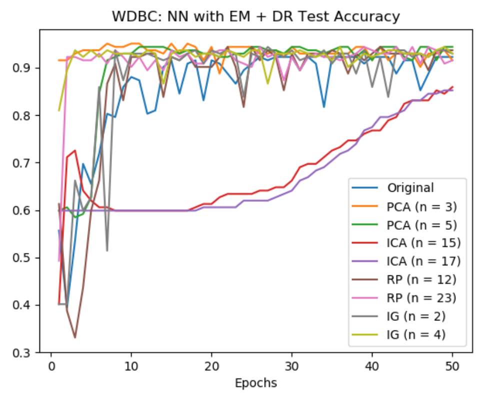


Fig 46: Test Accuracy of NN w/ EM + DR

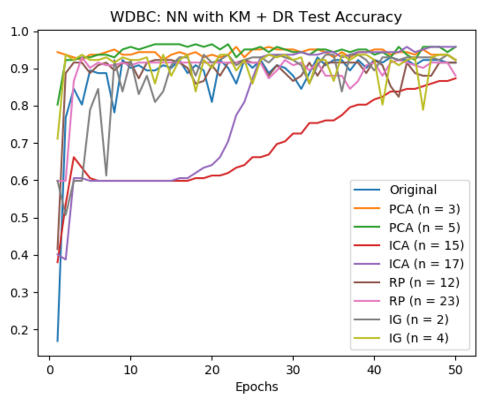


Fig 47: Test Accuracy of NN w/ KM + DR

References

1. https://statweb.stanford.edu/~gwalther/gap
2. https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/Expectation_Maximization_(EM)