Unsupervised Learning and Dimensionality Reduction

Introduction

Unsupervised Learning focuses on learning hidden structure/characteristic of unlabeled data. One task is clustering which group objects in such a way that objects in the same group are more similar in some sense/measurement to each other than to those in other groups. In this analysis, we will contrast two approaches, K-means clustering and expectation maximization. The problem we often faces when performing such task is the curse of dimensionality, and one solution is dimensionality reduction, we will explore four algorithms, including PCA(principal Component Analysis), ICA(independent Component Analysis), RP(random Projection) and CFS(correlation based feature selection implemented in Weka). All those algorithms will be tested on two datasets, one of them (IPO performance dataset) is from our previous assignment. We will apply same neural network learner from assignment 1 to the projected data and show the difference before and after dimensionality reduction to explore the characteristic of these algorithms. All algorithms are available in Weka except ICA which was developed as package by Chris Gearhart and is available on GitHub.

K-Means Clustering

K-Means clustering tries to partition the observations into K clusters while minimizing the within-cluster distance (often Euclidean distance). It is simple yet powerful. It starts by picking K random points as the centroid of each cluster, then calculates the distance of each point to the centroids and assigns the point to the closest centroid accordingly. The next step is to calculate the new means as the new centroid for each cluster, then do the distance calculation and assignment again. The algorithm converges until the assignment no longer changes. There is no guarantee to find the global optimal. Two features of K-means clustering often determine how well it performs on the problem, the first is the choice of distance measure and the second is to pick the appropriate number of clusters K. We will examine both Manhattan distance and the Euclidean distance. As both our datasets have class labels, it is natural to set the k to the number of classes, but that does not always lead the best result (meaning reasonable or not counterintuitively) as we will try a few different K.

Expectation Maximization (EM)

Expectation Maximization (EM) models the cluster using statistical distribution and focusing on finding the parameters of the distribution that maximize likelihood of the attributes. It follows an iterative approach and there are two steps involved in each of iteration. The first step is E-Step (as expectation) to estimate the probability of each point belongs to the cluster, the second step is M-Step(maximization) that re-estimate the parameter vector of probability distribution of each class. It stops when distribution parameters converge or reaches the maximum iteration steps. EM algorithm may converge to local optimal as well depending on the starting values. The Weka implementation of EM can help to decide the number of clusters by cross validation, we will also explore when K is set to number of classes for the datasets.

Dimensionality Reduction Algorithms

Dimensionality Reduction algorithm projects the data in the higher dimensional space to space of fewer dimensions. It helps to avoid the effects of the curse of dimensionality and in many cases to reveal the underlying structure of data. We are going to explore four of such algorithms.

1. Principal Component Analysis (PCA)

Principal Component Analysis is the most widely used subspace projection technique. It finds linear combination of original attributes that are uncorrelated between them and the principal component. The goal is to minimize the re-projection error from compressed data. And the principal components can be ranked by how it preserves the variance. We can pick the top K or set threshold for variance kept.

2. Independent Component Analysis (ICA)

Independent Component Analysis assumes the data can be represented by linear combination of statistically independent variables and try to identify them. Unlike PCA, its goal is to minimize the statistical dependency between projected attributes. The other assumption ICA made is the non- Gaussian distribution of the independent variables. There are two ways to measure independency, one is minimizing mutual information and ICA (using FastICA algorithm) we tested is maximizing non-Gaussianity.

3. Random Projection (RP)

Random Projection basically projects the original high-dimensional data to a lower-dimensional subspace using a random matrix. The motivation for the algorithm is that for a given dataset it may be hard to select the criteria to optimize the projection, and it has been shown to be able to change the shape of clusters to be more spherical, thus easier for clustering algorithm such as EM to work on. The drawback is that the result is highly unstable and randomness could lead us to drastically different clustering results.

4. Correlation based Feature selection(CFS)

The final algorithm we are going to test is correlation based feature selection, which is based on works by M. A. Hall in 1998. It finds locally predicative attributes by iteratively adding attributes that has the highest correlation with the class and at the same time try to keep low inter-correlations between classes. The implementation is available in Weka as well.

Datasets

For this analysis, I choose two datasets, one is from assignment 1 called IPO performance data which tries to predicate mid-term(6-month) performance of IPO stock given its short term performance(1 day, 30 day and 60 day) and who is the underwriter. It is reasonable to think that there are hidden factors that contribute to the stock performance after IPO, but we can only observe the stock price. So it will be a good candidate for dimensionality reduction. The nominal attribute “underwriter” is converted to multiple binary variables, which are linearly dependent, thus posts another challenge to the dimensionality reduction algorithms. It is also good candidate for clustering since it is natural to put IPO stock to different categories.

The other dataset is from UCI machine learning repository called “Yeast” dataset; it tries to find the protein localization site with 8 different kind signal measurements. The number of class in this dataset is 10 and it has very high concentration on four of them (CYT/NUC/MIT/ME3), which help us to explore the impact of choosing different K for clustering. Also it is interesting to see if there are interactions between signals under dimensionality reduction as it is very similar the classic “cocktail party problem”.

Analysis on Clustering Algorithms

When we run our two clustering algorithms on the two datasets, the first choice we need make is to set number of clusters K. Even though the two datasets already have class labels and it seems natural to set K as number of labels, we should have some concerns on our mind; do the current labels represent the data in a more meaningful way? For example for a dataset with 3 labels, we can always combine 2 labels and give it a new label name, in some cases the new labels makes more sense, but in other cases they remove some “good “ information. We will set different K for comparison. The EM implementation in Weka can help us to decide an “optimal” K with cross validation, we will denote that as K*. The original number of classes in the dataset is denoted as K~. Another choice we need make is to pick one distance measure, we will test both Euclidean and Manhattan distance. Once the data is clustered, we compare it to the labels already in the data to measure the accuracy.

		K-Means Clustering				**EM**	
		Euclidean Distance		Manhattan Distance			
Dataset	K Selection	WCSS	error rate	WCSS	error rate	Log Likelihood	error rate
IPO	k~=2	79.7	39.42%	155.07	46.15%	-13.64	39.42%
	k=3	67.46	44.23%	141.39	47.11%	-7.89	44.23%
	k=5	49	59.61%	98.42	56.73%	4.77	61.54%
	k=10	30.48	72.12%	74.42	74.03%	24.45	74.04%
	k*=12	15.32	75.96%	55	75%	25.85	74.04%
Yeast	k=2	129.62	63.61%	705.62	67.52%	8.7	63.75%
	k=3	113.16	62.87%	656.22	64.82%	9.64	61.25%
	k*=5	95.8	59.43%	605.94	67.12%	15.63	66.44%
	k~=10	58.65	61.39%	519.73	64.08%	18.11	64.89%

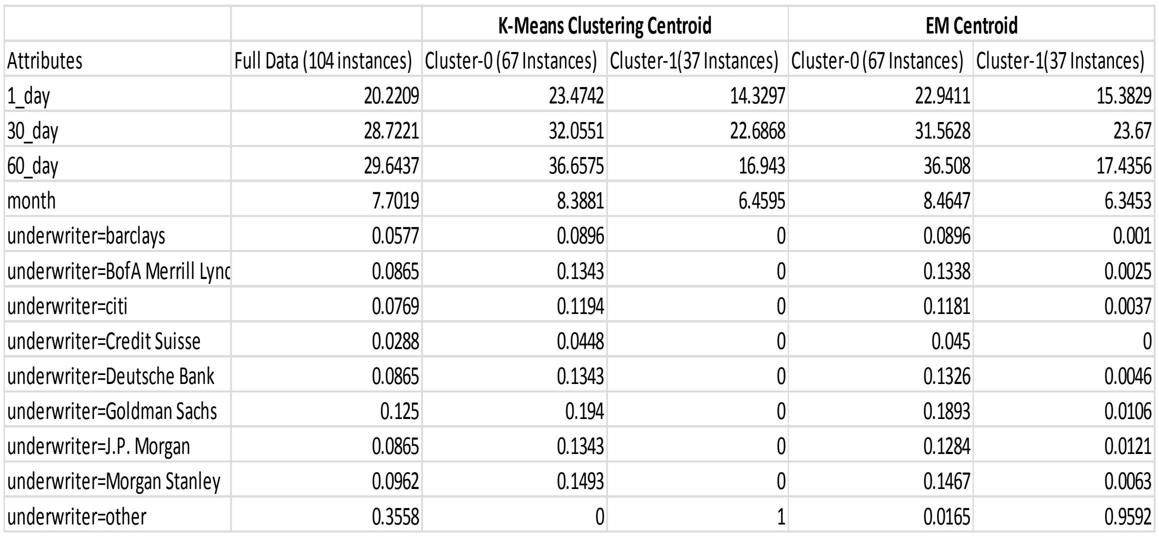
Table 1 Performance of K-means and EM on two datasets

Table 1 shows the result of two datasets on the two algorithms (WCSS is Within Cluster Sum of Squared Error). As we expected, the WCSS and Log likelihood increase as the increase in K will give more descriptive power to the model, but that is not without any cost, we saw increase in error rate as well. In general Euclidean Distance did somewhat better than Manhattan distance but it is really not much a difference, we will stick with Euclidean distance for experiment and analysis in the following sections.

It is not surprising that two algorithms give similar performance as two algorithms both use cluster centers to model data, while K-mean works preferably toward spherical cluster, EM allows clusters take more shapes. It is worth to take a close look at IPO data when K=2. Their classes to cluster matrix is exact the same.

	cluster-0(Assigned T)	cluster-1 (Assigned F)
Class-T	42	16
Class-F	25	21

The final cluster centroid from K-means and EM are shown in table 2. The centroids from the two algorithms are almost identical. The clusters certainly make sense as one have concentration on higher end of performance for 1 day, 30day and 60 day, the other is at lower end but also with some interesting characteristics. The centroid has its 30 day return higher than those of 1 day and 60 day return, which suggesting those are most likely speculation play IPO and people sell the stock to lock their profit after 30 days. And we can expect the return will not be any good if we hold the stock till 6 month. We can conclude the 30 day and 60 day return should be fairly good features to represent the data. We will later check if it is the case when we run the dimensionality reduction algorithms.



		K-Means Clustering Centroid		EM Centroid	
Attributes	Full Data (104 instances)	Cluster-0 (67 Instances)	Cluster-1(37 Instances)	Cluster-0 (67 Instances)	Cluster-1(37 Instances)
1_day	20.2209	23.4742	14.3297	22.9411	15.3829
30_day	28.7221	32.0551	22.6868	31.5628	23.67
60_day	29.6437	36.6575	16.943	36.508	17.4356
month	7.7019	8.3881	6.4595	8.4647	6.3453
underwriter=barclays	0.0577	0.0896	0	0.0896	0.001
underwriter=BofA Merrill Lync	0.0865	0.1343	0	0.1338	0.0025
underwriter=citi	0.0769	0.1194	0	0.1181	0.0037
underwriter=Credit Suisse	0.0288	0.0448	0	0.045	0
underwriter=Deutsche Bank	0.0865	0.1343	0	0.1326	0.0046
underwriter=Goldman Sachs	0.125	0.194	0	0.1893	0.0106
underwriter=J.P. Morgan	0.0865	0.1343	0	0.1284	0.0121
underwriter=Morgan Stanley	0.0962	0.1493	0	0.1467	0.0063
underwriter=other	0.3558	0	1	0.0165	0.9592

Table 2 Final Cluster Centroid from IPO data by K-means and EM

Another observation from the table 2 is that underwriter attribute is not as informative as we think. We can definitely see some positive impact if underwriter is one of eight major players in this field, and just by numbers Goldman Sachs stands out (no surprise here). One thing we need mention is that we convert original underwrite attribute (nominal) to binary attributes for K-means and EM to work on. This projected data to a higher dimensional space and we know that there is linear relationship between those attributes for sure (the sum should be equal to 1). This makes it difficult for both algorithms to discover this linearity and even harder for them to find the role underwriter attribute and its interaction with other attributes. We will later apply dimensionality reduction algorithm and see if and how this issue can be addressed.

The last part of this clustering will focus on the choice of K. We used EM to find the "optimal" K by cross validation. For IPO data, EM finds the optimal value to be 12 which may look counterintuitive at first, but if we follow our discussion just before this part, that can be explained by the linear relationship of 9 underwriter binary attributes. Thus it posts great challenge for EM to find the "optimal" K in this case. If we looked the Yeast data from Table 1, EM finds the "optimal" cluster number to be 5. We know the yeast data has 10 classes, but we also know there are four dominant classes that make up 87.5% of instances. So consider 4 dominant classes plus all other as one class, set K=5 could very likely be an optimal choice. As shown in Table 1, the error rate for K=5 for K-means is indeed the smallest. EM results show K=3 could be another good choice as in Yeast data two dominant classes claim 60% instances. This again highlights the importance to understand the data and choose right K, and the need for dimensionality reduction.

Analysis on Dimensionality Reduction

In this section, we will discuss the experiments in which we apply four dimensionality reduction algorithms on our datasets, and then apply the clustering algorithms on the projected datasets.

Principal Component Analysis (PCA)

For PCA we first try to find out the number of projected attributes to cover different percentage of variance in the original dataset. And there are two matrices we can use for PCA, one is correlation matrices and the other is covariance matrices. Generally the covariance matrix is used when the variable scales are similar and the correlation matrix is used when variables are on different scales since using the correlation matrix standardizes the data.

As shown in Figure 1, we see the number of projected attributes trending down when we relax our requirement for variance coverage. For the same dataset, however, PCA with different matrices used exhibit quite different behaviors. For the Yeast data, we see they are almost in line with each other. A closer look at yeast data shows that the attributes are almost already standardized. As for IPO data, the binary variables for underwriter are in quite different scales than 1-day, 30-day and 60-day return numbers. The variance of those attributes for returns will become dominant and PCA can just throw away some those projected attributes without sacrificing too much for variance coverage. As we see, almost 85% attributes can be removed and we still keep 99% of variance. In fact, we can achieve 99.99% with only five projected attributes (from 13 original attributes)

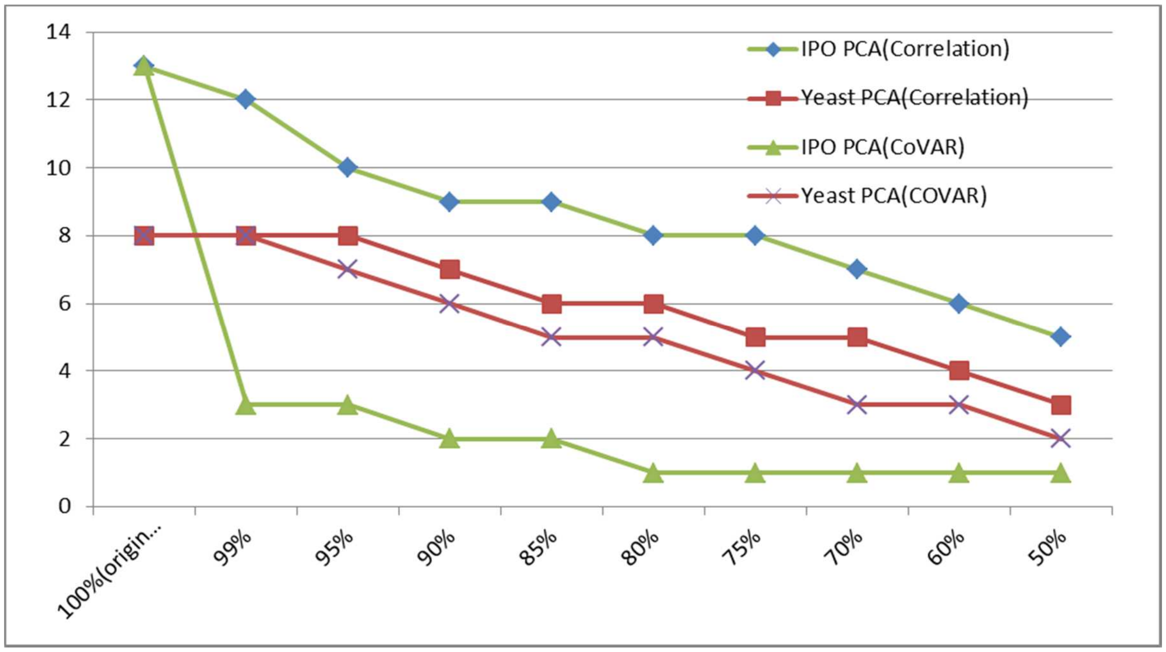


Figure 1 Number of Projected Attributes to cover variance at different Level for PCA

We can further see the effect from statistics of eigenvalue from PCA with those two matrices. With 95% variance covered, the mean and variance of eigenvalue for PCA with correlation matrices are 1.25and 0.46, while for PCA with covariance matrices they are 2240 and 2235.

Independent Component Analysis (ICA)

For ICA, the implementation of Studentfilter package uses FastICA algorithm which maximizes non-Gaussianity. Table 3 shows the Kurtosis measure of the attributes of original data and after we applied ICA; the values have been sorted from largest to smallest.

		attr1	attr2	attr3	attr4	attr5	attr6	attr7	attr8	attr9	attr10	attr11	attr12	attr13
Kurtosis	IPO (before ICA)	31.23512	15.87481	13.07175	8.5458	7.04144	7.04144	7.04144	6.767709	5.840568	3.359348	2.70826	-1.3837	-1.6584759
	IPO (after ICA)	3.255709	2.205161	1.994277	1.1012	0.977296	0.602133	0.509106	0.499677	-0.05582	-0.27143	-0.61357	-0.76978	-0.9660719
Skewness	IPO (before ICA)	0.678387	-0.61139	-1.5243	-2.301	-2.37624	-2.78004	-2.98436	-2.98436	-2.98436	-3.12248	-3.22209	-3.84977	-5.7126797
	IPO (after ICA)	8.868644	7.669831	7.366962	3.9067	3.159891	1.410132	1.108514	0.304506	0.268079	0.175459	-1.25007	-2.53769	-3.0071295
Kurtosis	Yeast (before ICA	105.7387	101.3547	9.501359	7.7778	2.289971	1.60932	0.556111	0.45906					
	yeast (after ICA)	46.41847	15.47955	8.204984	6.7423	4.90965	2.372662	1.138414	0.860068					
Skewness	Yeast (before ICA	10.27688	10.15963	2.413031	1.4448	0.604291	0.416639	-0.221	-1.79164					
	Yeast (after ICA)	5.474068	2.775018	0.349429	0.1232	-0.08079	-0.10781	-1.24146	-1.57833					

Table 3 Kurtosis and Skewness before and after ICA

For IPO data, we can see most distributions from original data are with high kurtosis, and they are for some different reasons. For attributes like 30-day return(kurtosis=15.87), data are highly concentrated in sub 20% range with a few outliers as high as 354%, as we already know that kurtosis can be very sensitive to outliers. Its value may depend on only a few observations in the tails of the distribution (the characteristic is fairly common in finance data). For the binary attributes such as underwriter=Credit Suisse (kurtosis=31.24), the reason is the data represent the specific underwriter is fairly small percentage and that always leads to extreme value of kurtosis. And the Skewness indicates many distributions with flat tails (or outliers). We can observe similar patterns from Yeast data.

Since ICA here tries to find independent components by maximizing non-Gaussianity. Kurtosis is not a robust measure of non-Gaussianity as we know outliers can cause problem. Here we are more interested if Gaussianity can be somewhat "removed" after we apply ICS. As we can see from Table 3, at least it does not introduce more Gaussianity and seems to do little better. For the IPO data, most kurtosis are under 3 which indicating a more flat distribution after ICA. we are not sure it captures the "meaningful" component b/c of the nature of the data(outliers) does not suggest that central limit theorem can apply here, i.e. , there may not be independent variables contributing linearly behind.

Random Projection (RP)

RP(Random Projection) reduces the number of attributes while still preserving some variance like PCA. To make it fair comparison with PCA (and other DR algorithms), we set the number projected attributes to 2 for the IPO data and 7 for Yeast data and run RP five times. The general observations is that RP tends to project to the new attributes with more of Gaussian distribution, data is more centered and could possible mitigate the impact of outliers. RP is computationally cost effective and should produce some fairly good result of multiple runs. We will see if it indeed benefits our clustering algorithm.

Correlation based Feature selection(CFS)

The last dimensionality reduction algorithm we choose is CFS (Correlation based Feature selection) in Weka. According to Weka, it evaluates on subset of attributes and prefers features that are highly correlated with least inter-correlation. For the IPO data, the CFS selected two attributes, 30-day return and 60-day return, which is very intuitive and good selections based on our analysis before. For Yeast data, it selects 7 attributes. We may notice how close those numbers to the PCA's (with Covariance matrices) result when maintaining high enough variance coverage (85%-90%)

Clustering after Dimensionality Reduction

We now should feel more comfortable to set number of projections to 2 for IPO data and 7 for Yeast data given results above. We will use this setting as base for comparing the algorithms when reproducing the clustering experiments with the projected dataset. For RP, we take both average and the best and worst. For ICA however it could not reduce the number of attributes for the two datasets and we will keep the projections to 10 for IPO data and 8 for Yeast data, we also randomly choose 2 new projected attributes for IPO and 7 for Yeast data just to explore. As for number of clusters, it is natural to set k=2 for IPO data and k=5 for Yeast data given our discussion before. And For K-means we will use Euclidean distance.

		K-means							
Dataset	K Selection	No DR	PCA	ICA(full set)	ICA(subset)	RP(average)	RP(best)	RP(worst)	CFS
IPO	k~=2	39.42%	46.15%	47.12%	41.35%	46.73%	45.19%	49.04%	47.12%
	k=3	44.23%	49.04%	59.62%	57.69%	53.65%	51.92%	55.77%	51.92%
	k=5	59.61%	58.65%	64.42%	65.38%	58.27%	52.88%	64.42%	60.58%
	k=10	72.12%	69.23%	73.08%	77.88%	69.81%	52.88%	75.96%	74.04%
	k*=12	75.96%	72.12%	74.04%	77.88%	70.77%	52.88%	80.77%	74.04%
Yeast	k=2	63.61%	62.94%	68.19%	69.88%	65.35%	62.33%	69.81%	63.68%
	k=3	62.87%	63.01%	55.66%	56.94%	63.15%	57.55%	68.13%	63.01%
	k*=5	59.43%	58.49%	52.83%	56.47%	63.14%	59.43%	67.99%	61.79%
	k~=10	61.39%	66.64%	68.60%	64.08%	69.38%	66.31%	73.32%	63.01%
		EM							
		No DR	PCA	ICA(full set)	ICA(subset)	RP(average)	RP(best)	RP(worst)	CFS
IPO	k~=2	39.42%	41.35%	40.38%	45.19%	46.54%	43.27%	49.04%	48.08%
	k=3	44.23%	46.15%	53.85%	53.85%	49.62%	47.12%	52.88%	61.54%
	k=5	61.54%	48.08%	62.50%	62.50%	56.92%	45.19%	65.38%	62.50%
	k=10	74.04%	67.31%	74.04%	76.92%	70.38%	52.88%	80.77%	80.77%
	k*=12	74.04%	80.77%	76.92%	78.85%	73.85%	52.88%	81.73%	79.81%
Yeast	k=2	63.75%	66.64%	68.80%	68.80%	65.13%	62.47%	68.67%	68.40%
	k=3	61.25%	63.68%	62.87%	62.40%	64.47%	59.77%	69.00%	60.92%
	k*=5	66.44%	64.76%	53.30%	61.93%	66.20%	61.93%	68.06%	58.56%
	k~=10	64.89%	71.16%	62.13%	60.11%	71.89%	69.34%	75.40%	60.71%

Table 4 Performance of K-means and EM on datasets after Dimensionality Reduction

Table 4 shows the full results and the first thing we notice is how RP stands out to our surprise given that we only run RP for 5 times. In general, the four dimensionality reduction algorithms do no sacrifice performance very much when comparing to what we see without dimensionality production. K=2 for IPO data and K=5 for Yeast data still can be considered reasonable number of clusters. This actually suggests both the clustering and dimensionality reduction algorithms works as expected for these two data, as the dimensionality reduction algorithms keep the underlying structure intact and both clustering algorithm find similar clusters with and without dimensionality reduction. Certainly dimensionality reduction saves the computation cost greatly. It is worth noting that random projection, as simple and efficient as it is, produces good result fairly consistent, which is not as we originally expected. As suggested from this experiment, some high dimensional space can be randomly projected to low dimensional space without losing too much underlying structure.

Analysis on Neural Network Learner after Dimensionality Reduction

In this final part, instead of feeding the projected data to clustering algorithms, we will treat it as classification problem and test the projected data on neural network learner like the one in assignment 1. We are also interested in results when we treat the clustering algorithm as if they were dimensionality reduction algorithms. To do that, we will just treat the clusters as if they were new features. We will use similar setting of the neural network learner to that in assignment 1-- a simple one layer 3 node neural network implemented as MultiLayerPerceptron in WeKa . We will focus on our IPO data and to make comparison meaningful, we are going to run with dataset whose underwriter attribute converted to multiple binary variables.

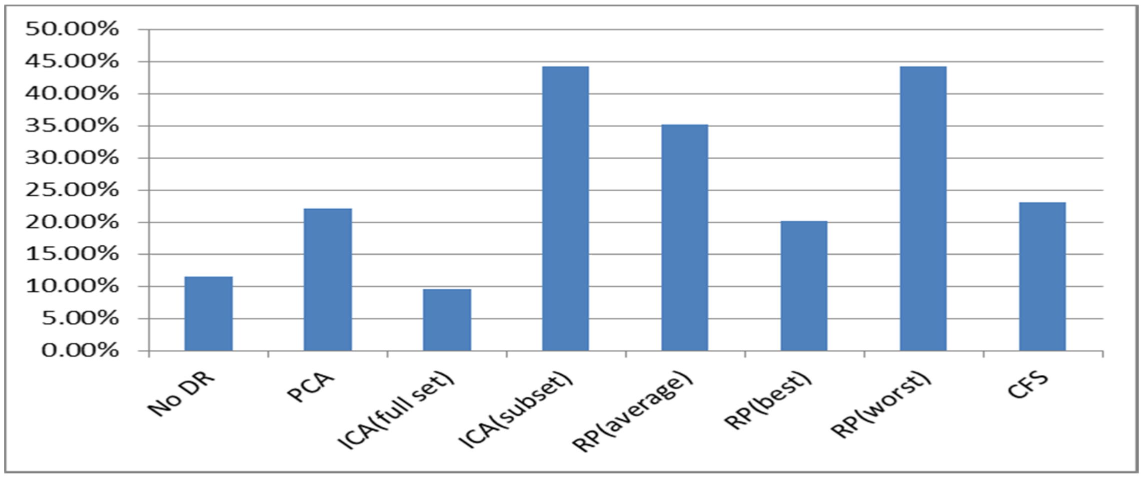


Figure 2 Error rate of Neural Network after Learner with DR algorithms

In general, we are going to lose some useful information when apply dimensionality reduction algorithm when projecting high dimensional space to lower dimensional space. The performance of neural network in most cases gets worse. The three worst ones come from RP(worst) , RP average and ICA(subset) , which should be expected. Once again RP(best) proves that it can match performance with PCA and CFS. ICA (full set) is the best partly due to it preserves same number of attributes of original data(so no actual dimensional reduction here), but it still has its merit in that it has slightly better performance(error rate= 9.62%) than the case without dimensionality reduction(error rate=11.54%).

In the last when we apply the two clustering algorithm (with several K values) as if they were dimensionality reduction algorithms, it produce some interesting result as shown in Table 5.

	ANN with K-Means as DR algorithm	ANN with EM as DR algorithm
k~=2	0%	17.31%
k=3	0%	15.38%
k=5	0.96%	10.58%
k=10	8.65%	13.46%
k*=12	1.92%	13.46%

Table 5 ANN performance on IPO data with K-means and EM as DR algorithm

Both reduce error rate as the clusters as new feature actually present some learned knowledge in assisting neural network. In discussion early we see the clusters learned by K-means and EM are quite close, while the error rate showing here suggests that for the IPO data the small difference cannot be simply ignored.   Even a small movement of the centroid has big impact on how we label the data. ANN with EM exhibits some robustness against the selection of K as EM is based on mapping the statistical distribution(more K given EM more power to learn the “detail”), while ANN with K-means can achieve exceptionally good results, but the performance is not very stable, which suggests K-means has more of a locality characteristic.

Conclusion

In conclusion, clustering algorithms as we discussed can be viewed as one form of dimensionality reduction that project data into one dimensional space while preserving some characteristics/features. Dimensionality reductions can greatly help any learning task by reducing the complexity we have when dealing with high dimensional space. However, to make it most useful and effective, we still need more knowledge of our data to begin with. The more we can tell about our data, whether it is domain knowledge or we learn through any learning algorithm, the better we can choose the suitable dimensionality reduction algorithm and parameters.