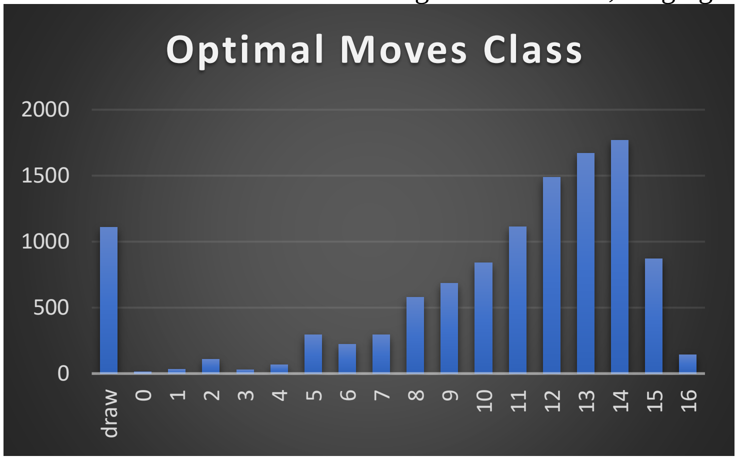
George Maroun CS 4641- Guzdial
Assignment #3

Unsupervised Learning Report

Datasets

Krkopt Chess (King-Rook vs. King):

This is a Chess Endgame Database for White King and Rook against Black King (KRK) - Black-to-move Positions Drawn or Lost in N Moves. The task is to predict the optimal number of moves required for the White player to win the game through checkmate given the relative positions of the two white pieces and one black piece. There are 7 attributes and a total of 11,222 instances. Below is the class attribute distribution for all the endgames in the file, ranging from a draw to 16 moves.



Diabetes:

This is a binary-valued dataset for identifying potential signs of diabetes. This has been used as an adaptive learning routine to forecast the onset of diabetes mellitus. There is a total of 768 instances with 9 attribute including class.

Why are these interesting datasets?

As far as the Krkopt data is concerned, these types of problems have been put to use in industry for the simple reason that a reliable cornerstone of logging our progress in the realm of artificial intelligence is checking algorithms against human beings. Companies like IBM and Intel have poured millions into research for supercomputers that outperform and consistently beat expert humans at strategy games like chess. Consolidating and optimizing searches in chess endgames could allow us to effectuate chess AI systems with more cost efficiency.

The Diabetes dataset is first and foremost blatantly interesting in the world of medicine. There is certainly a lot of medical innovation left to be done, and a huge part of it is going to come with computing power being able to glean through huge archives of data to extract patterns and generalize for further patients. On a small scale, this is what the dataset is doing in that it tracks common health abnormalities associated with diabetes and generates a system for diagnosing, quickly mind you, potential positive-testing individuals. These both do a great job in defining how machine learning will come into play in a practical sense in quotidian life.

**** For space efficiency in this paper, please refer to my assignment #1 on why these data sets are interesting.**

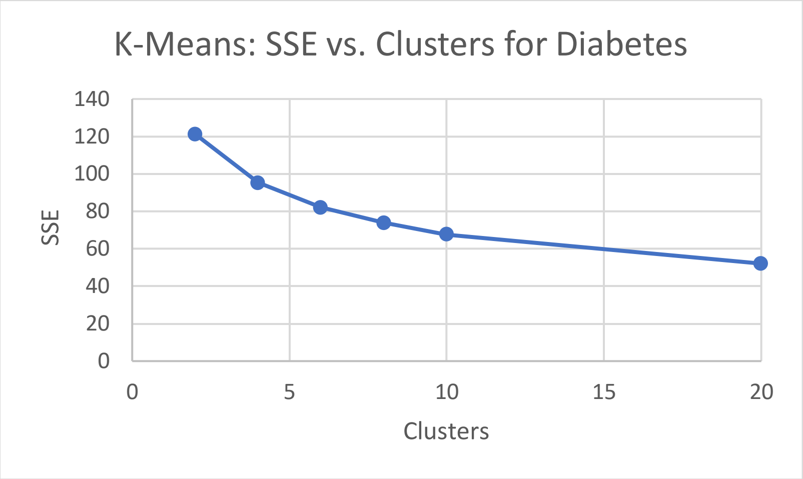
Clustering:

I made use of WEKA's "Classes to Cluster" function, which searches for the strongest class match per cluster K. This might be cause for a less powerful classification for non-binary datasets. K-Means seeks to store K centroids to form clusters of related instances.

k-Means Clustering

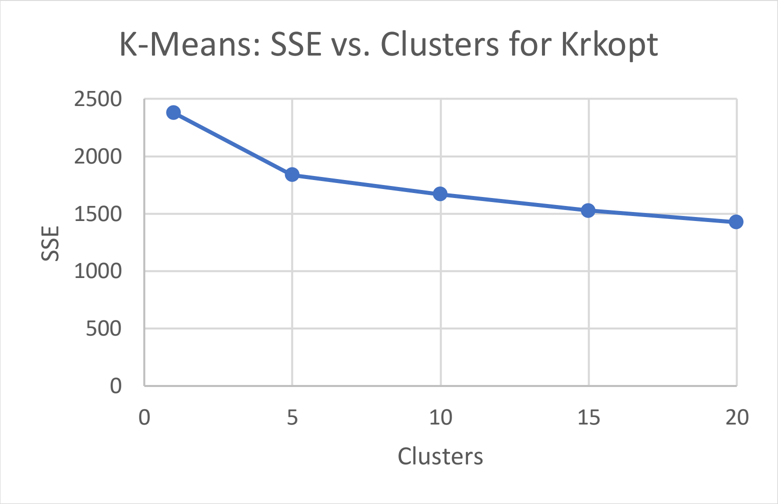
Clusters(K)	Time Elapsed	% Incorrect	#Iterations	SSE
2	0.01	33.2031	7	121.2579
4	0.01	48.0469	31	95.2365
6	0.01	65.7552	10	82.07568
8	0.02	70.0521	26	73.8549
10	0.02	74.6094	23	67.653

20	0.03	82.8125	18	52.0903



For the Diabetes dataset, I used a range of K from 2 – 20. As seen in the graph, the SSE seems to drop off and plateau at the end of the iteration run(~52), which is where I decided to stop the algorithm because the optimal finding would be an area where the SSE is changing incrementally but requires the least amount of cluster groups. I sincerely thought the algorithm would converge earlier on given the ease with which the instances are labeled, and how well the assignments converged in the first assignment, not to mention that this is after all a binary classification. However, it took more to find a better value of K. This is a testament to the potential of hidden players as attributes in the dataset and how they can sway the final classification. There is also a level of danger when we increase K, as shown by the percentage classification that is incorrect (~82%), although this is not the aim with clustering. Training time was not an issue here.

Clusters(K)	Time Elapsed	% Incorrect	#Iterations	SSE
1	0	64.0934	1	2377
5	0	76.6607	4	1837
10	0	80.2513	4	1667
15	0	84.7397	3	1525
20	0	86.535	3	1425

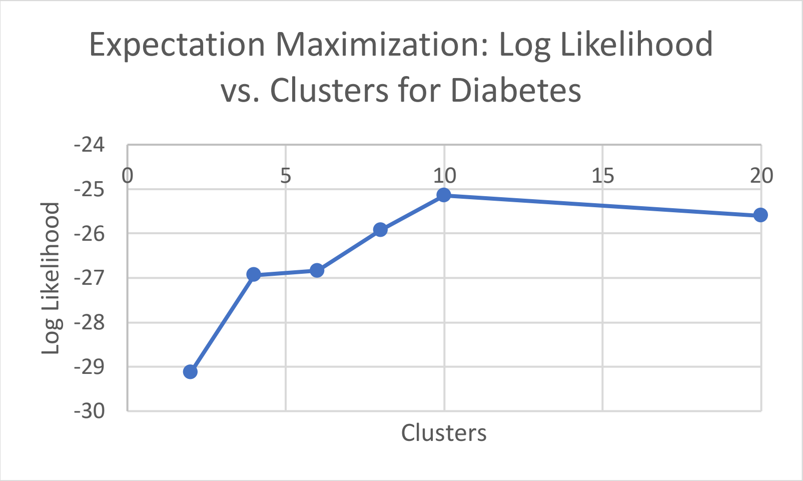


For the Krkopt dataset, I elected nearly the same range 1 – 20, but this time with different intervals since the dataset is more complicated (many more available labels to group instances), and again the algorithm sees a steep decline at the beginning with respect to SSE but proceeds to slow down and converge at a value (albeit 1425 is much higher than the other). I think these datasets are interesting to compare since the value of validity can't be normalized with clustering like it can in Supervised Learning(accuracy). This is highly case by case, and although it might not be better in an objective sense, the algorithm converges nonetheless. Again, we see the level of incorrect classes start to skyrocket, however the tradeoff for the minimization of SSE would suggest that this value is marginally important with regards to K-Means.

Expectation Maximization Clustering

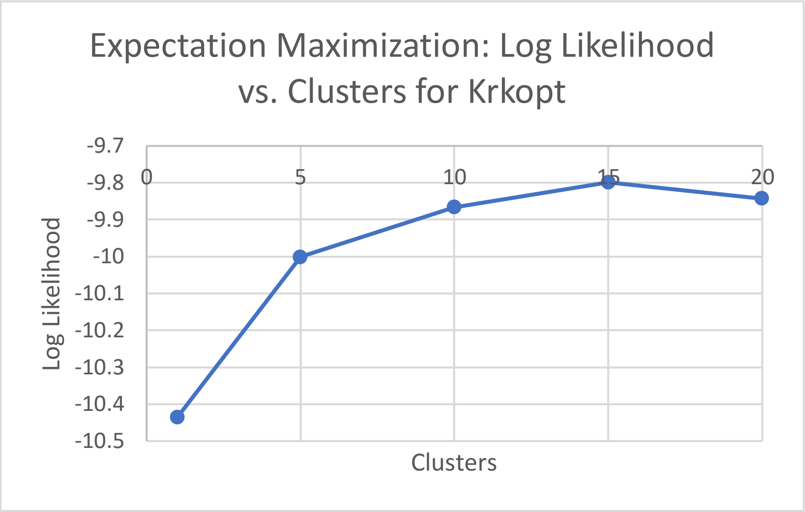
I again made use of the "Classes to Cluster" evaluation for choosing the optimal class for each cluster. Expectation Maximization works by guessing the model's parameters for a probability distribution. The distribution is then tweaked per the data fed into the model, and this process continues until it converges. This works quite well, relatively speaking, with datasets with missing data.

Clusters(K)	Time Elapsed	% Incorrect	#Iterations	Log Likelihood
2	0.07	33.9844	34	-29.1284
4	0.07	45.8333	1	-26.9338
6	0.1	58.724	1	-26.8426
8	0.13	72.7865	1	-25.926
10	0.17	72.7865	1	-25.1447
20	0.3	84.375	1	-25.5997



I kept the range of cluster assignments the same, but this time the metric under consideration is the Log Likelihood, which is an indicator of how likely the data are to be generated by your parameters chosen through the EM Iterations. Since we want this value to be maximized, judging by the graph the peak of this property seems to be at the K=10 mark for this dataset. We're using a log estimator since, as arises, when data get complicated there's too many magnitudes to computationally delineate so instead we take the log. Here, at a value of -25.1447), the specific clustering parameter hints at the tightest group formations able to be generated from the parameters chosen by EM.

Clusters(K)	Time Elapsed	% Incorrect	#Iterations	Log Likelihood
1	0	64.0934	1	-10.43534
5	0.19	71.4542	100	-10.00129
10	0.35	75.763	100	-9.86624
15	0.51	76.3016	100	-9.79887
20	0.31	78.2765	41	-9.84266



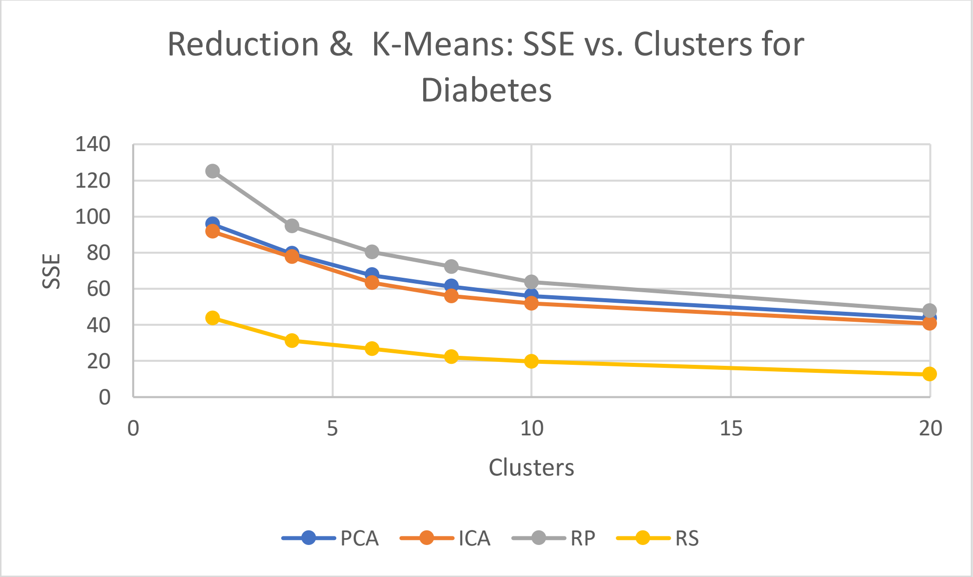
As a preface, for the same reasons mentioned before, we keep the cluster range the same and identify based on the same metric. For the chess data set there is a similar peak in the Log Likelihood Estimator at K=15. Although the neighbors are not far off, when we abstract this on a larger scale the clusters deviate highly in performance from this optimal location and eventually converge to an optimal value, which in this scope, is clearly shown.

I want to note that even as the percent incorrect blows up, this is not necessarily the goal with unsupervised learning, that is, classifying instances. Rather, we are trying to group labels in the k-tightest groups, which is why I argue for an LL estimator metric.

**** To retain consistency, I ran the Clustering Algorithms on a Euclidean Distance similarity measure across instances**

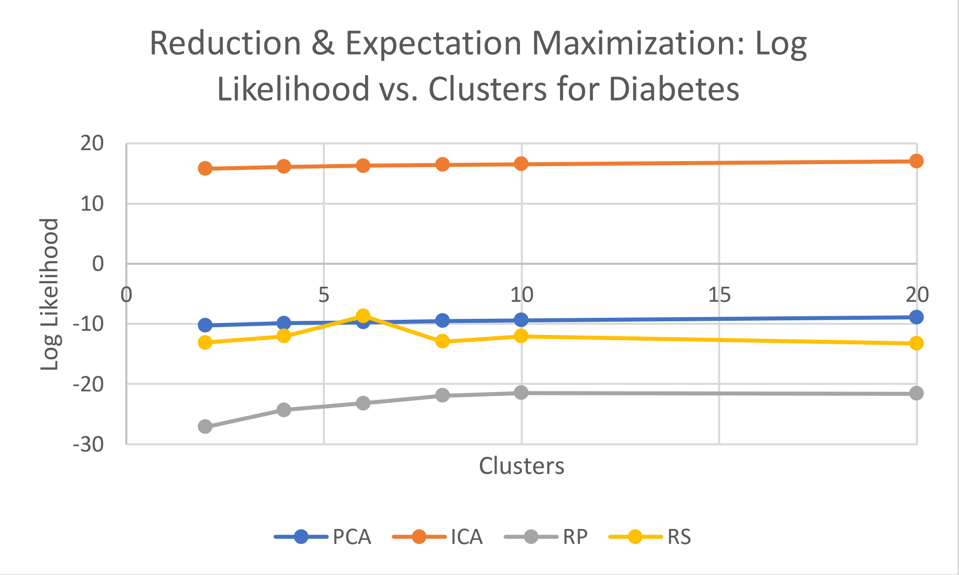
<u>Dimensionality Reduction:</u>

Algorithm	% Incorrect	#Iterations	SSE
PCA	83.724	13	43.4971
ICA	85.6771	21	40.6416
RP	85.2865	26	47.6777
RS	87.8906	36	12.4673

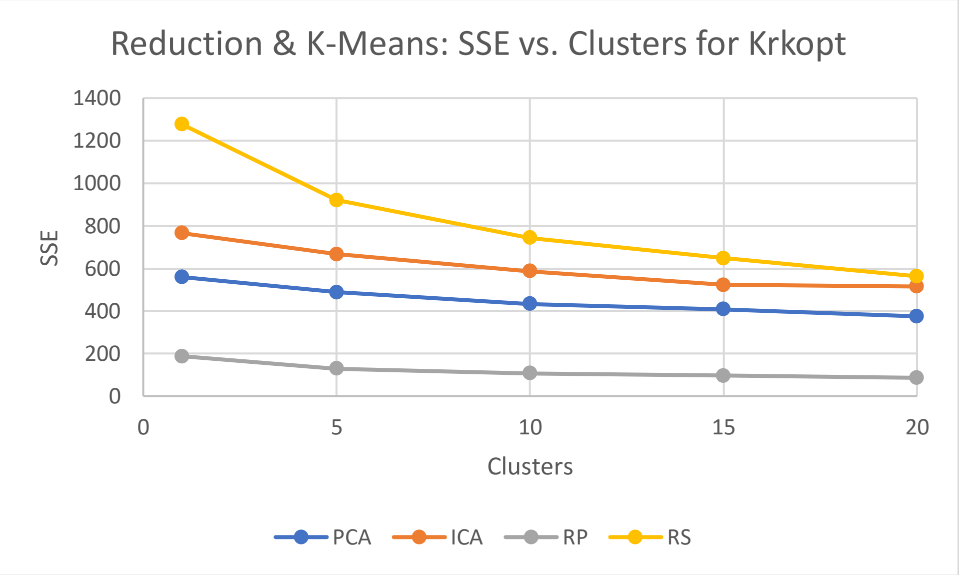


Algorithm	% Incorrect	#Iterations	Log Likelihood
PCA	81.25	100	-8.8955
ICA	84.7656	98	17.0148

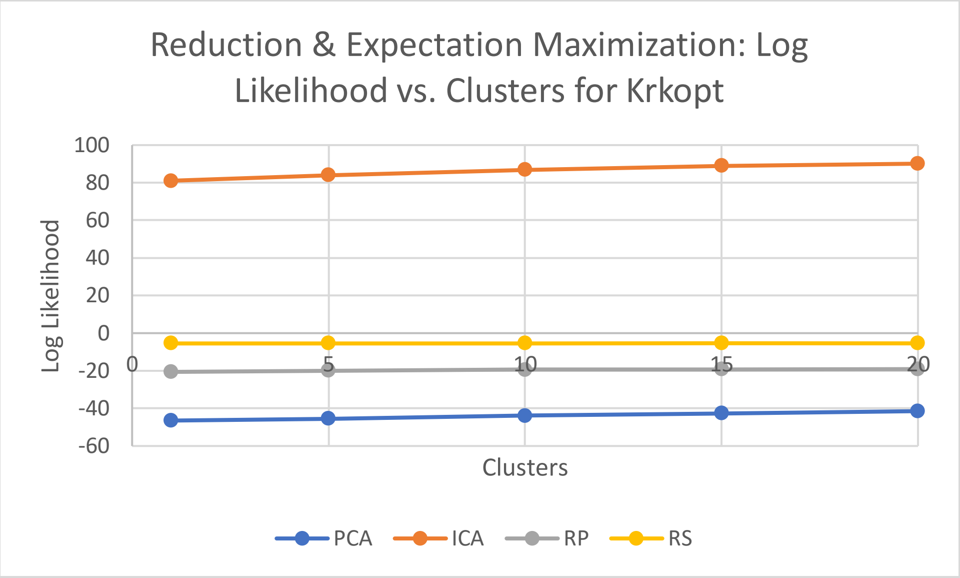
RP	83.4635	1	-21.5555
RS	81.7708	1	-13.2388



Algorithm	% Incorrect	#Iterations	SSE
PCA	85.9964	21	374.7586
ICA	86.7145	15	514.9154
RP	86.3555	19	85.9102
RS	86.535	3	563



Algorithm	% Incorrect	#Iterations	Log Likelihood
PCA	85.0987	38	-41.4654
ICA	85.6373	25	90.1165
RP	85.2783	2	-19.1112
RS	82.5853	1	-5.4374

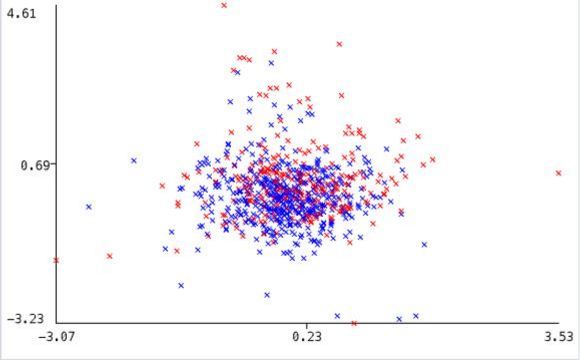


Principle Component Analysis

The goal with PCA is to generate principle components, parameters that are pivotal to understanding trends in the data. By normalizing attributes and finding these combined patterns we can flesh out groupings that tell us more about trends in the data through the component analysis.

For the Diabetes dataset with K-Means, the SSE was minimized even more than the original (~43%), which highlights the notion that combining the patterns does pose a reward even with computation at stake. The group impacts were unforeseeable in a raw data cluster. For the same dataset with EM, the Log Likelihood was once again reduced compared to the original, likely attributed to the same reason. The trends, however, both in incorrect classification and SSE/LL were extremely similar, noting the learning curve that is evident with finding optimal clusters, which remained the same.

For the Krkopt dataset with K-Means, the algorithm saw a massive improvement on the same order of K with respect to SSE. The value was cut in more than half (374), which shows that the triviality of the applied reduction can easily be trumped by the long-run benefit. For the EM clustering, the algorithm performed much worse with LL comparison. It turned out, most likely, that the PCA component assignment did not provide the correlation which it sought out, rather grouping together attributes that are seemingly related but hurt the dataset.



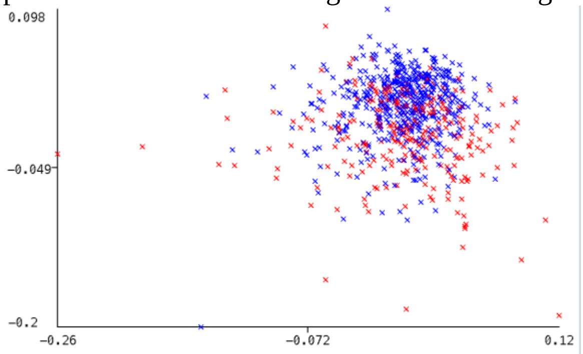
As is clearly shown by the graph, the level of overlap in the PCA algorithm foreshadows the inconsistencies that it might draw. That’s not to say, though, that the new clusters tell less about the data than the old, as was clearly not the case in ¾ PCA comparisons.

Independent Component Analysis

The goal with ICA is to generate independent components, parameters that are equally, in cases, insightful with respect to understanding data trends. These components are found by linearly separating mixtures in the data that tell us something when they are looked at alone. This independent analysis helps to grab the latent variables that make or break a classifier.

Right off the bat we should notice that the kurtosis of the ICA respective to Diabetes is very high and the shape non-Gaussian which already tells us the task of separating independent components might be a very productive one with this data. Looking now at the KM performance, the SSE was minimized more than the original and PCA, which is what expect given the distribution. The relatively simple data set saw an improvement when its attributes were looked at individually. For the EM cluster, the log likelihood blew the other algorithms out of the water, with LL=17.0148. Again, the portion of the data incorrectly classified remained on the same order as did the learning curve of the graphs, which indicates the fact that the convergence isn’t necessarily happening faster.

For the Krkopt dataset with KM, the algorithm performed better than a raw cluster, however did not do as well as the PCA cluster. Although it still did better, the attribute distribution was Gaussian so I would expect the ICA to run poorly, relatively speaking, not to mention the dilemma of a convoluted data set with many labels hiding layers that potentially affect the cluster. When we look at EM for chess, like the diabetes data set, the metric was crushed with ICA comparatively. It would appear that the data are highly likely to be generated by the chosen parameters, which goes against my initial conjecture of a poor performance. Overall the algorithm was strong.



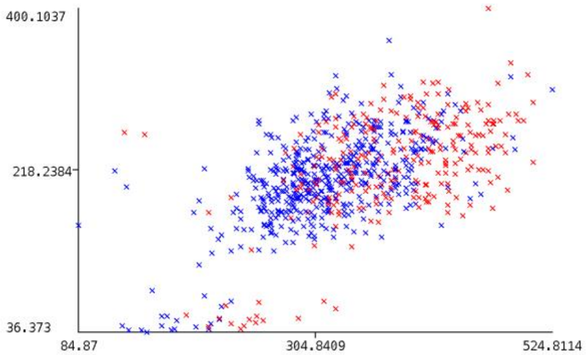
Although the level of overlap is just as strong as PCA in this reduction technique given in the graph, the per-data cluster obviously had a much more powerful presence seeing as for Expectation Max. the data are highly likely to be generated.

Randomized Projections

The goal with RP is to reduce dimensionality of the data using a randomized matrix while preserving the distance variance that PCA exhibits. It’s great at cutting computational complexity, and uses NominalToBinary to convert attributes to numeric before reducing them.

For the Diabetes data set with K-Means, the algorithm clustered only slightly worse in terms of SSE (~47). I expected this to happen given the track record of the other algorithm’s performances and given again the simplicity of diabetes. The trends as we tune clusters follows a very similar path to what we’ve seen. Graphically these algorithms aren’t having an extremely rich effect on the optimal k, it stagnates. For the EM cluster, even though we see improved performance, this is the closest log likelihood (-21) we’ve seen to the long-run average of the original untouched cluster, so it had the smallest effect thus far.

With regards to the Krkopt data set, the K-Means actually performed extremely well. It turned out that the application of the NominalToBinary followed by the KM cluster ended quite fruitfully for the SSE. Not only did it perform much better than the original, but it outperformed (on the order of 3) the past 2 dimensionality reduction algorithms we’ve seen so far. When we look at EM, the Log Likelihood ended up somewhere in the middle, at a point where there is ambiguity as to whether applying the algorithm on top of the cluster is worth the resources lost in order to converge the log value to a maximization. This is an important notion when we consider compounding these algorithms, since the tradeoff of computation for performance become nebulous. Nonetheless, the RP matrix led to a very good output.



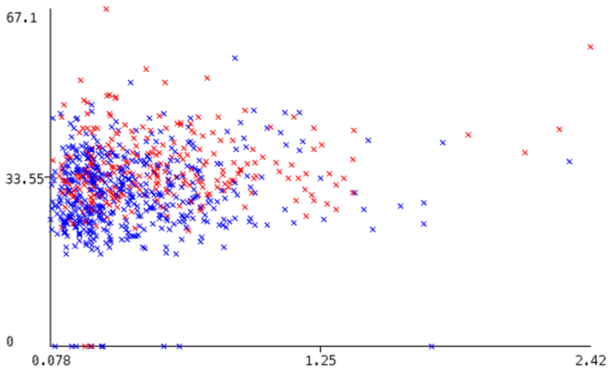
In this graph, we start to see a more apparent cluster separation in this 2-attribute visualization. We can not necessarily, however, make any more generalizations from this seeing as the objective cluster graph is still very weak with this particular reduction algorithm.

Random Subset

The goal with RS is to choose a random number of subsets (either hard number or percentage) to include in the final class output. By removing the rest of the attributes, the idea is to find a subset of informational attributes to cluster on.

For the Diabetes data set with K-Means, even with the touch competition, the SSE is the lowest value that we have seen throughout the experimentation, which is more or less expected in my predisposition given that the attribute space is quite small. It started low and converged faster, which optimizes the use of RS for Diabetes. Looking now at the EM cluster, we've seen the comparisons among original vs. reduction, and this lies about smack dab in the middle. I thought this would also be the lowest since the original reduction worked quite well and the parameter guess would multiply that, not to say it didn't do well either way.

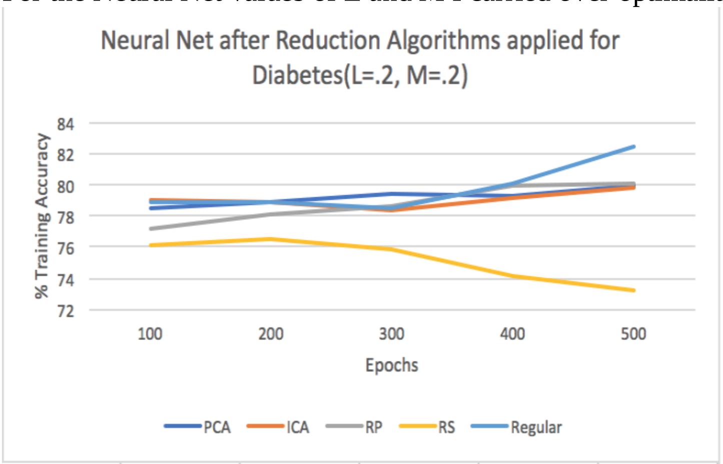
With respect to Krkopt with K-Means, the SSE, albeit better than the original K-means, which is a threshold that we now consider to be trivial, was actually worse in performance than all 3 other reduction algorithms. This might be attributed to the complicated nature of the data set removing attributes that are maybe not the most important, but indeed important in the overall classification. For the EM cluster, the log likelihood for Krkopt does quite well all things considered, which is a weird side by side juxtaposition when we look at trends with inter-cluster relationships among the same reduction algorithms. At this point, we can deem the RS algorithm specific to these data sets strong and a good fit given its low running time and strong output.

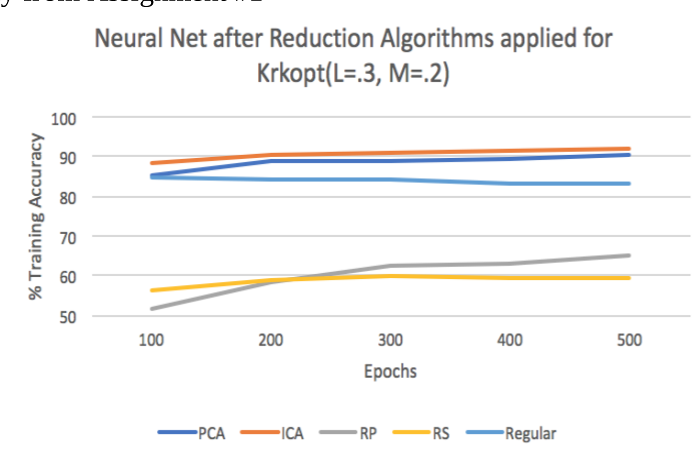


We aren't really seeing much variation with regards to how well each algorithm on the clusters is separating the data, as it seems to be doing about the same. With this in mind, although the algorithms have different approaches and end goals, even with this level of abstraction they weren't THAT much different. Looking at these side by side seemed to be a weak indicator of what might happen, in hindsight.

Holistic Neural Net Comparisons

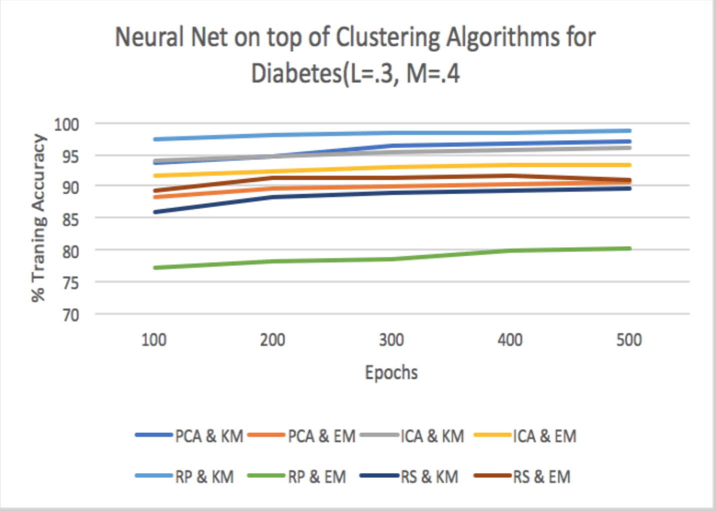
For the Neural Net values of L and M I carried over optimality from Assignment #1

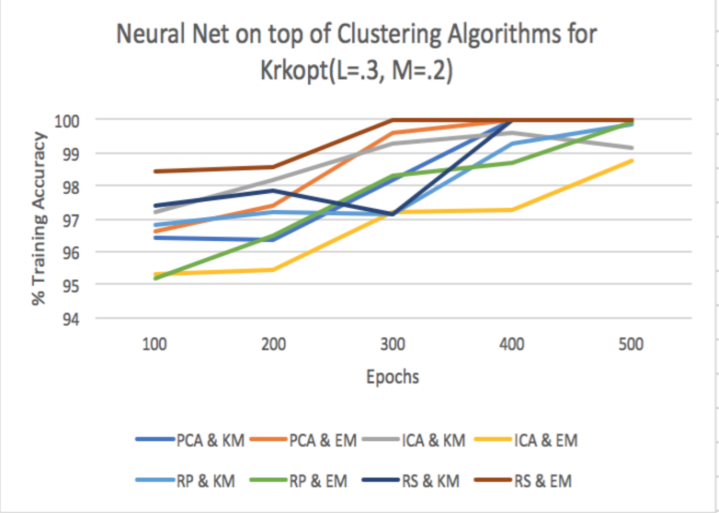




Comparing the post-reduction Neural Net performance against the baseline of raw data for Diabetes, we can see that, even across epochs, all the algorithms are struggling to keep up with raw accuracy, likely since with binary classification one might not want to reduced dimensionality for the simple reason that the original classification was strong and the attributes fell in decent correlation so applying these algorithms turned into overkill. RS never started well and performed WORSE across epochs, describing an error in reducing attributes compounding on itself. The others didn't do horribly, all clumped together, but nevertheless long-run damaged the algorithm performance.

With the Chess data set, a far more complicated classification, we in fact see a separation in performance across reductions, and instead we see ICA and PCA outperforming the original. It turned out digging through the complex attributes and pulling together seemingly unrelated (or the opposite, for ICA) variables provided insight into the inter-instance relationships which led to stronger original inputs for the NN. On the other had, the random projection and random assignment of attributes in the latter 2 algorithms led to an info loss in the description of the data by ignoring pertinent attributes, which is why perhaps they performed so poorly. This is an important observation, and a strong one when we discuss counter-arguments to reducing dimensionality. This is a nice juxtaposition in relationship between data set nature and algorithm benefit across these 2 sets: it's highly case by case.





I then ran the Neural Net on top of the clustering algorithms with the same parameters as before to compare even more differences. For Diabetes, we see a uniform distribution of trends per combination, with some outperforming all the original and latter parts, reaching in some cases 15% more accuracy.

Obviously, however, we need to address the fact that since we are back in supervised learning terms, the neural net clearly over fitted the training data and broke the threshold. What I find astonishing is how the RP algorithm can be so dynamic with clustering (it was the best and worst with KM and EM respectively). The same reduction algorithm in conjunction with different clustering mechanisms can lead to vastly different results. Overall, clustering as a reduction enhanced the training accuracy of the neural net with a hodge-podge of combinations performing all over the place.

Again, with the Chess data set, we see the problem of overfitting manifest itself. Something to take note of quickly is the notion that running these reduction algorithms under the clustering algorithm under then the neural net completely changes the nature of the data, so comparing objective scores is a hard task. All the algorithms were on the uptrend, and together at that, with RS taking precedence. We should always approach these graphs with the remembrance that the attributes are being clustered, removed, or separated at random or with little regard and this leads to sporadic results.

Conclusion

One of the most apparent observations is perhaps the across-the-board strength of dimensionality reduction algorithms for both Diabetes and Krkopt data sets. When we look holistically at these compared to the original standalone clustering algorithms, save a few cases, they perform outright better as a group. That being said, the classification (even though this is not the focus of these algorithms) never improved which I'm still a bit confused about. One strong point to make is that the optimality of clusters(k) never really changed within algorithms although the cross-algorithm performance per K did. Therefore, the cluster vs. metric graphs really took a shape at the beginning and remained very constant throughout running all of these experiments. This provides such a rich understanding moving forward, with regards to how fluid preprocessing can be. I'll be honest, I thought there was a large fundamental fissure in Supervised/Unsupervised Learning before this assignment, but this has given me the insight to come to a clearer understanding that rather than competing in objective performance these two branches of Machine Learning are working in tandem, to hopefully tackle many angles of data and come to a strong classifier. This is something that we can directly see when we look at how well the updated Neural Net feeds did after clustering and reductions took place. These techniques are working in a collaborative manner, which is the biggest take away from me. As an ending note, examining the approach of the reduction algorithms up front I did not see how they could possibly outperform; it seemed whimsical in nature to me. After the data proved me wrong, I took a step back and realized the level of arbitration that we come into the classifiers with to begin with, so the data started to make a whole lot more sense. This was by far the most knowledge-rich assignment for me.