Raphael Gontijo Lopes
903062081

CS 4641
Assignment #3

Unsupervised Learning Report

Datasets Description

Sentiment Labelled Sentences – A dataset with 3,000 examples of online reviews (taken from amazon, imdb and yelp). Each example includes a binary label indicating whether the review is positive or negative. To use this example, we had to create a Bag of Words, and use the presence in a certain sentence as the features of that sentence. To perform well in this classification problem, the algorithm must learn the association that some words have with either positive or negative sentiments.

Mushroom – A binary classification dataset (poisonous/edible) with over 8000 examples. It includes 22 attributes that describe a mushroom instance, like color, shape, etc. To perform well in this dataset, the classifier must be able to distinguish attributes that indicate whether a mushroom is poisonous or not. This makes it perfect for a dimensionality reduction problem.

These datasets are interesting to me mostly because of their real-world applicability. Correctly identifying poisonous mushrooms can be life-saving. Identifying sentences' sentiments can help a review aggregator, for instance, separate praise and criticism. Furthermore, I believe that performing dimensionality reduction on a word vector dataset can be very interesting because of the nature of words: many are synonymous, or represent similar ideas, for instance, positive/negative sentiments.

Algorithm Runs

Clustering Algorithms

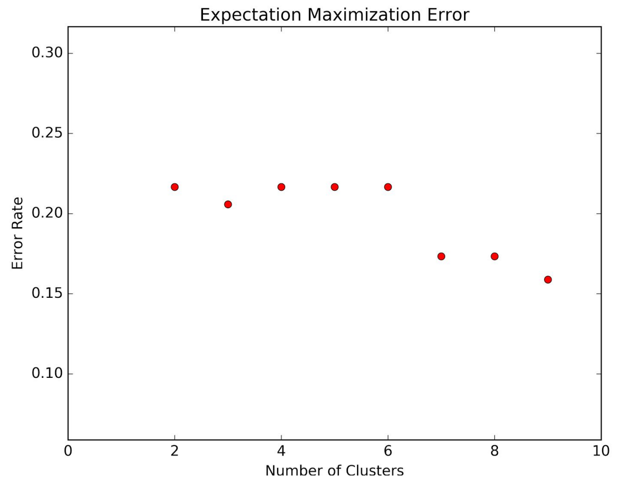
The first couple experiments executed were running the clustering algorithms (k-Means and Expected Maximization) on my datasets. To do this, I had to choose the number of clusters (a hyperparameter). An easy way out of the arbitrarity of this problem is to just use 2 clusters. After all, we have domain knowledge that both our datasets have 2 classes. As we'll see, this method performs perfectly fine. However, in order to achieve higher accuracies, we'd need to modify the distance/similarity measure being used so the data is more easily clustered.

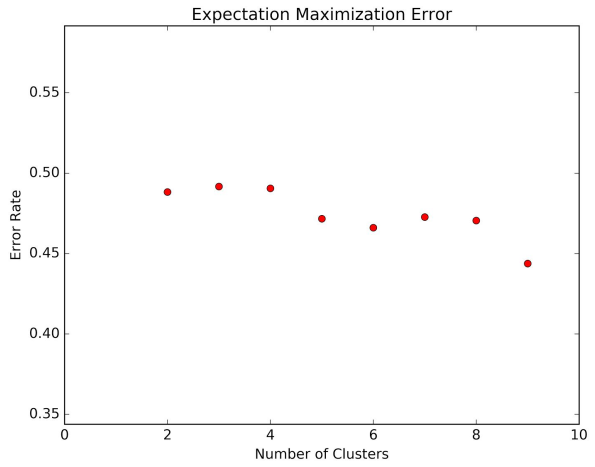
What I've done, instead, is cluster the data points into 2+ clusters. Then, predict the class of each data point in a cluster, and finally "label" the cluster with the class label that is most represented in the cluster. This method demonstrates two advantages over the one above: 1) it allows more specific clustering to capture nuance in the dataset (by simply having more clusters) and 2) it allows us to use a simple vector-distance similarity measure, and still get very good results.

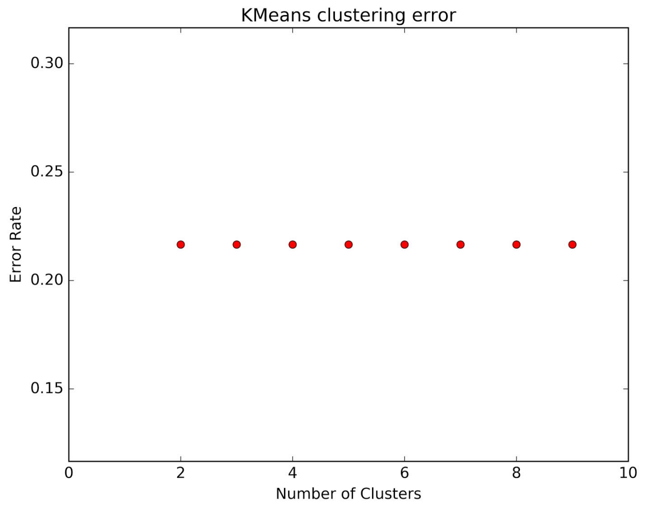
This method also made it possible for me to describe the data with the following plots:

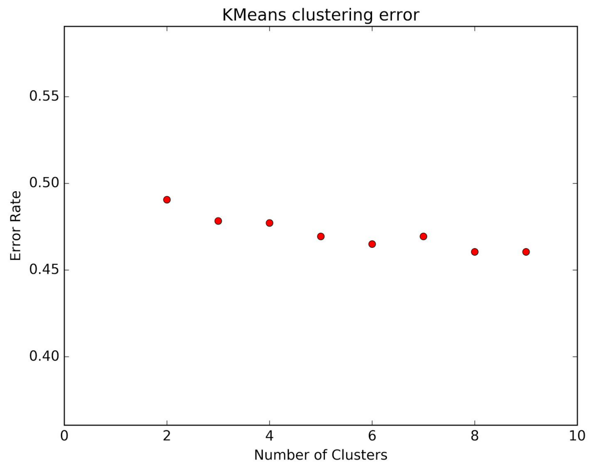
Mushrooms

Sentiment









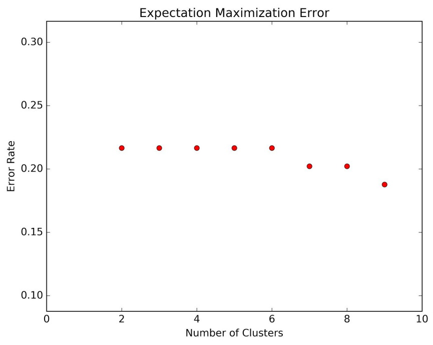
The first noticeable thing from these plots is the fact that the clustering algorithms were capable of capturing interesting insight from the data without being given any knowledge about their classification. This can be seen by the low error rates we achieved, especially in the case of Expected Maximization, with a high number of clusters.

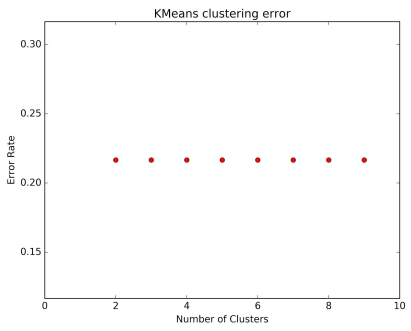
From these plots, it is also clear that the multi-cluster method used can be very helpful in classification problems by the mere fact that it better separates the data with respect to the class labels. The one exception to this was KMeans on the mushroom dataset, which seemed to perform equally well no matter the number of clusters, but I believe that this speaks more about the relatively simple nature of the mushrooms dataset than anything else.

<u>Dimensionality Reduction Algorithms</u>

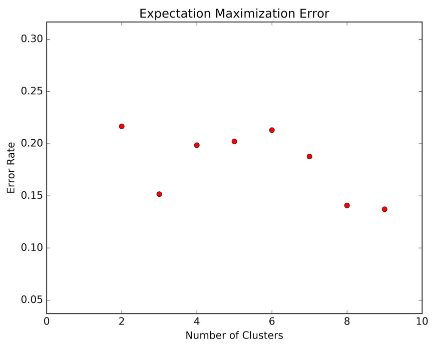
The next experiments run were for applying dimensionality reduction to our dataset. In order to describe our results (and be able to compare them with the results above), I ran the clustering algorithms on the resulting reduced-dimensionality data.

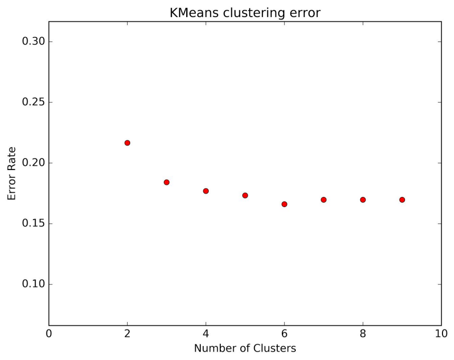
Mushroom + PCA



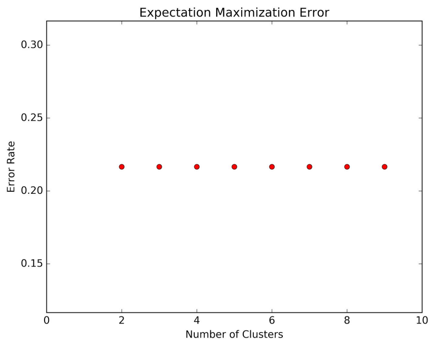


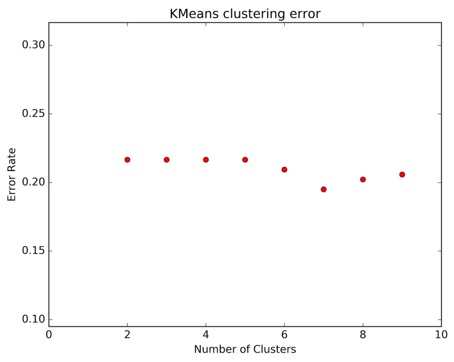
Mushroom + ICA



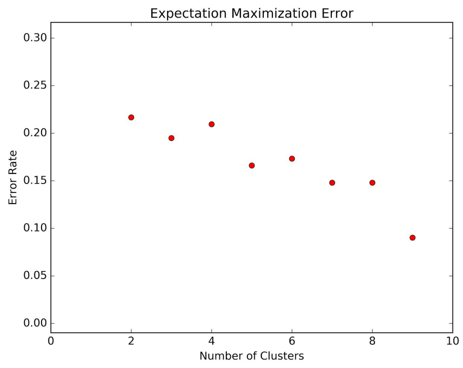


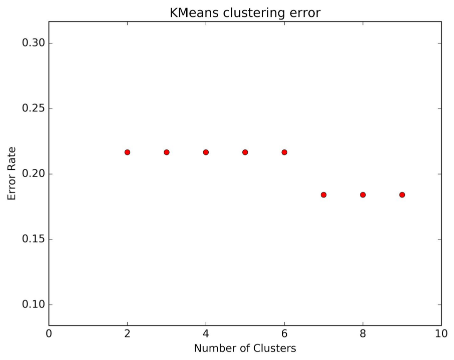
Mushroom + RandProj



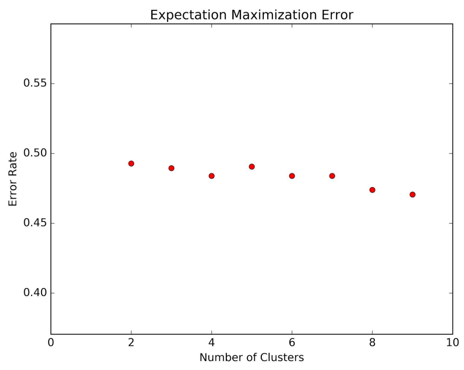


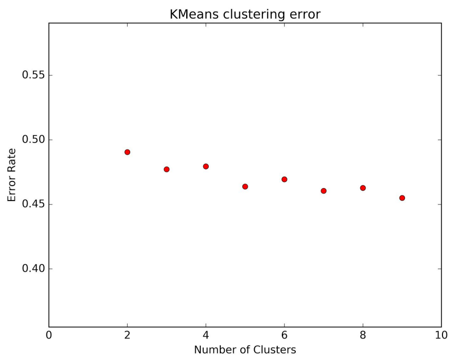
Mushroom + K-Best



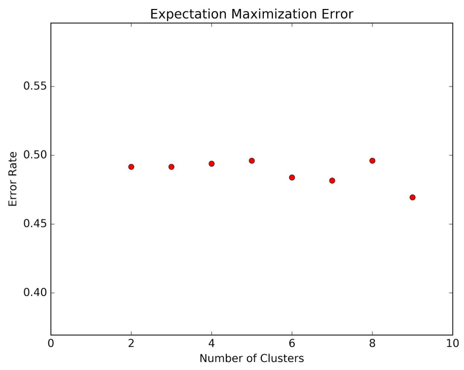


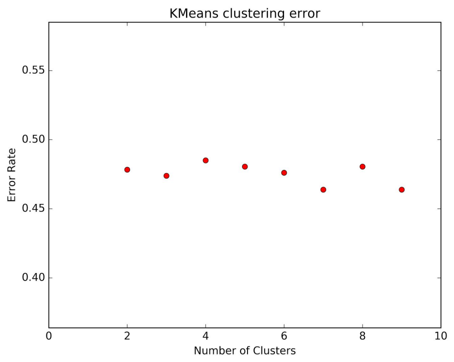
Sentiment + PCA



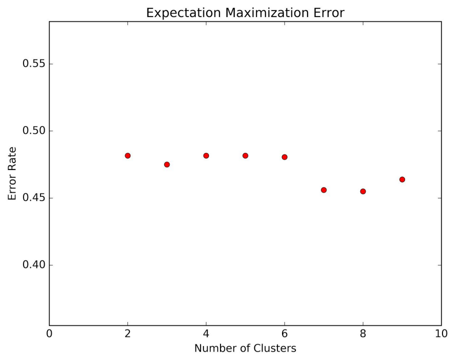


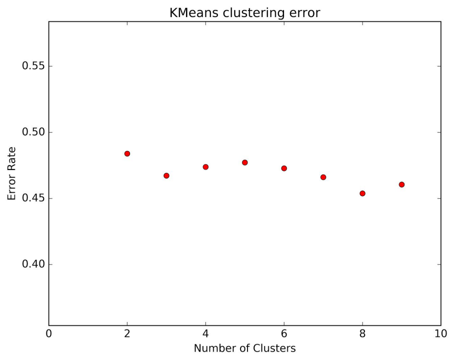
Sentiment + ICA



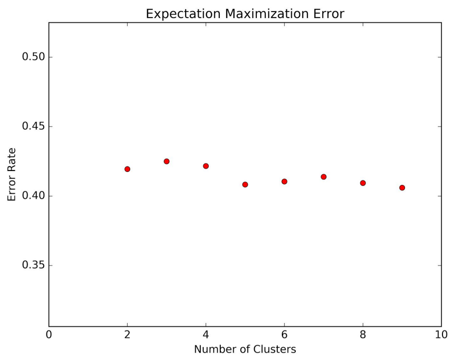


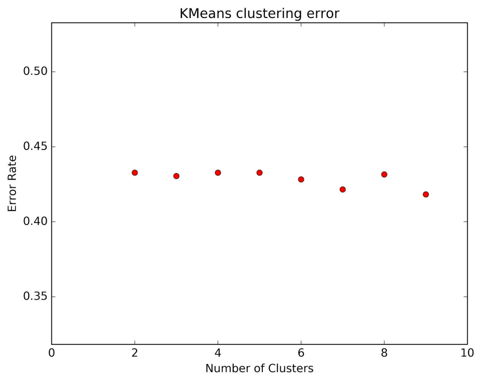
Sentiment + RandProj





Sentiment + KBest





The main noticeable thing about all these graphs is that dimensionality reduction alone does not improve classification by a whole lot. This is expected, since these algorithms are not classifiers. We see that clustering after PCA and Random Projections only slightly improve the cluster's correlation with the truth labels.

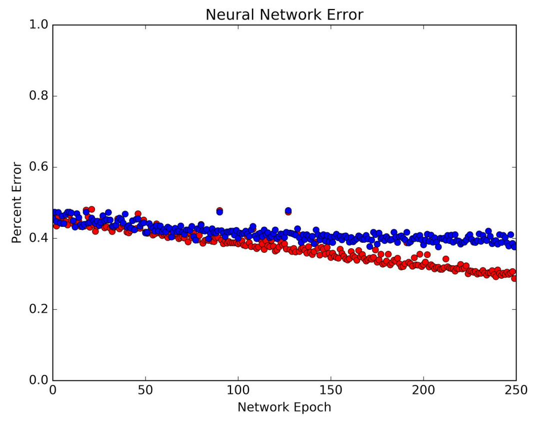
But we do see improvements, especially in the case of KBest, since it is specifically selecting features that seem to be correlated with the classification labels. ICA also performs a bit better, most likely because of the nature of the algorithm: finding independent features that describe the data. Their independence probably helped make the clusters a bit closer to the truth labels.

It was also interesting to see the high error rate in the sentiment dataset, when compared to the mushroom dataset. This is most likely because of the big size of the bag of words used (700 most common words). This number was picked because, as we've seen in previous papers, the bigger the bag of words, the more information our Neural Net has to make inferences, so it tended to perform better. This big number does bring the problem of the curse of dimensionality, but that's why we are using dimensionality reduction algorithms. So it made no sense to go with small bag of words.

It is important to note that the real power of these algorithms will only be revealed when we get to the neural network performance analysis. After all, the idea behind them is tackling the curse of dimensionality and being able to train a classifier more effectively. In fact, those were the next experiments run on the sentiment dataset, the one I used for my first assignment:

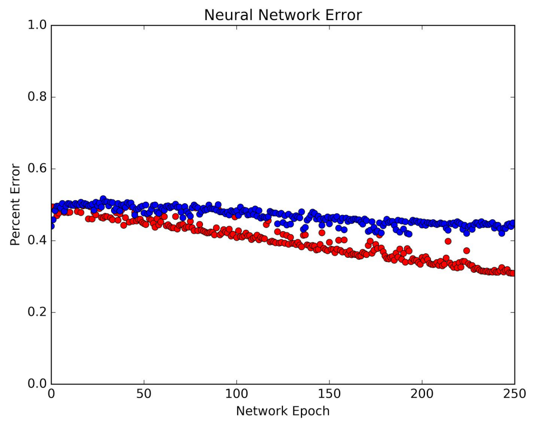
Neural Networks on Sentiment Dataset

We're gonna need to compare our results to something, so I reproduced the graph on the right - the learning curve for a neural network being trained on the sentiment dataset with 700 bag of words. It performs fine, achieving 30% error on the training set, and 40% on the test set after 250 epochs. As we'll see, a big part of the reason why these error rates are so high is the curse of dimensionality. 250 epochs and 3000 examples is not enough to properly learn which of the 700 features are relevant for classification.

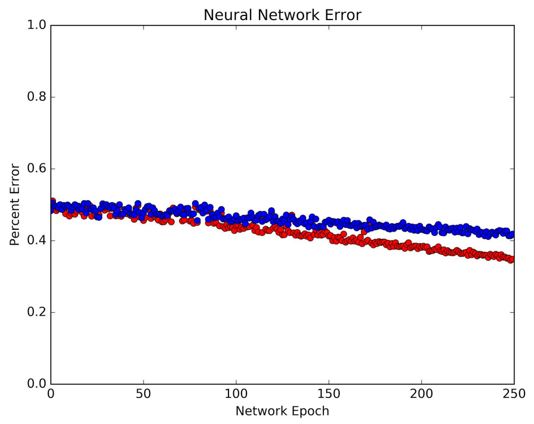


Thus, we train the neural network after applying the clustering algorithms:

Expected Maximization



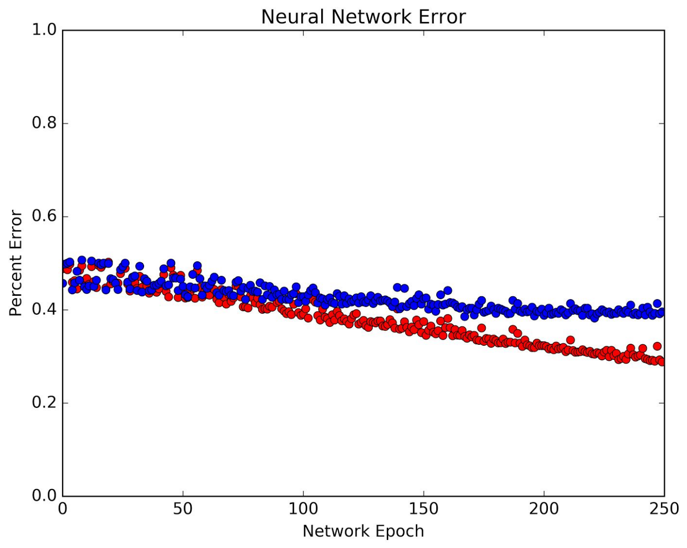
KMeans



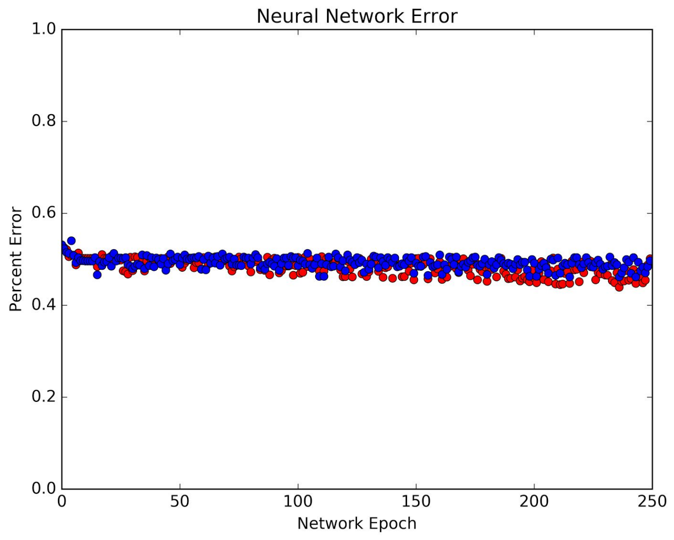
NN+EM was able to classify our training set more accurately than NNs alone. But overall, the differences in error rates don't seem that impressive. In fact, NN+KMeans achieved worse error rates. However, it is important to note the much more expressed downward trend in the new graphs: the neural nets have not converged yet, and with more epochs, they could achieve error rates lower than the initial ones. This contrasts with the first NN, which seems to have stabilized its error curve for the test set.

We also got interesting results when we trained classifiers on the dimensionality-reduced data:

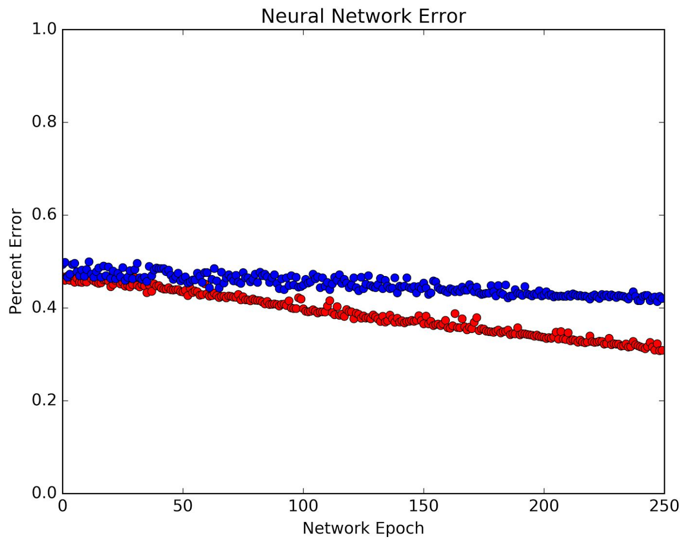
PCA



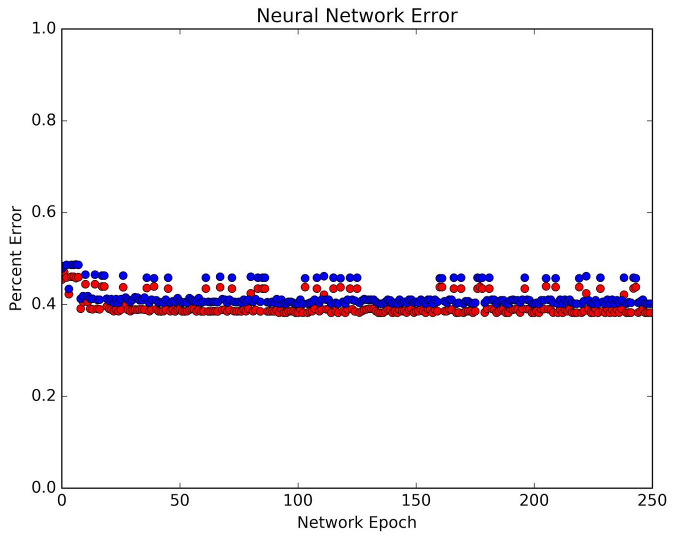
ICA



RandProj



KBest



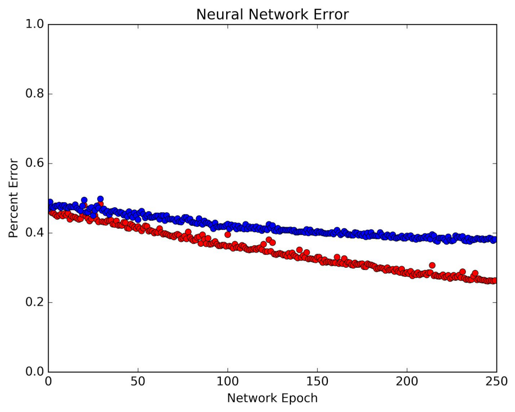
Once again, we get results comparable, and slightly better than NNs without preprocessing, particularly in the case of PCA and RandProj. Both of these classifiers show the big downward trend we saw in NN+EMs and NN+KM.

It is notable, however, that ICA and KBest performed extremely poorly. The reason I can think of to explain this phenomenon is the fact that dimensionality reduction algorithms are prone to finding features that may not be as good for classification as the original features. Maybe they best separate the data from itself, but in the loss of information that ICA and Kbest go through, they ignore attributes that, when combined, prove to be valuable for training a classifier.

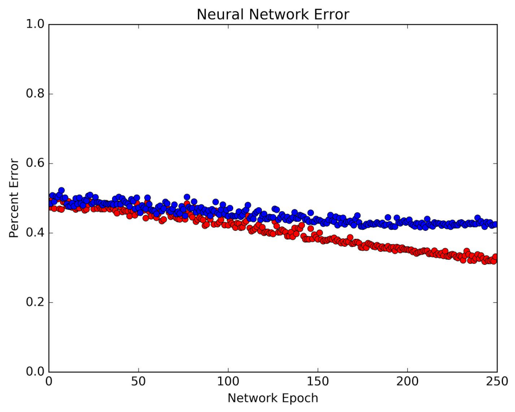
It is always a trade off: selecting features to tackle the curse of dimensionality can lead to selecting suboptimal features for classification. There is no free lunch

In fact, as we'll see, this problem will not go away, even when we try to cluster the dimensionality-reduced data. After all, the preprocessing already lost important features that would have been helpful in classifying. Given more time, I would have re-run these experiments having ICA and KBest pick more attributes, in hope that it will perform better in the classification task.

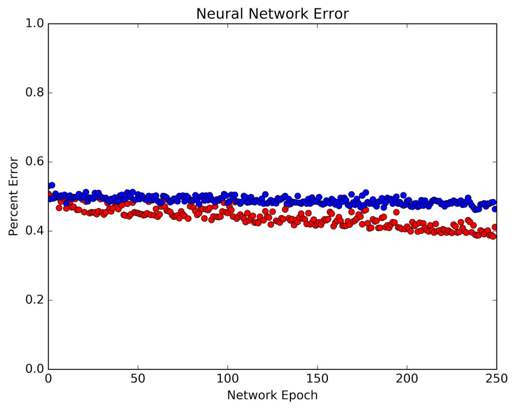
PCA+EM



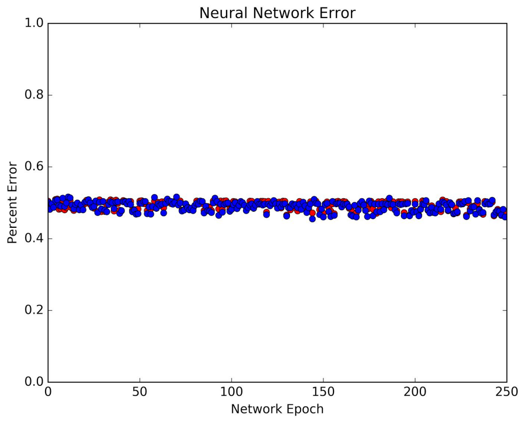
PCA+KM



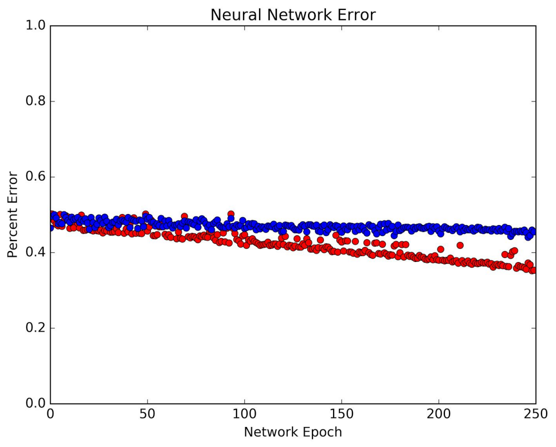
ICA+EM



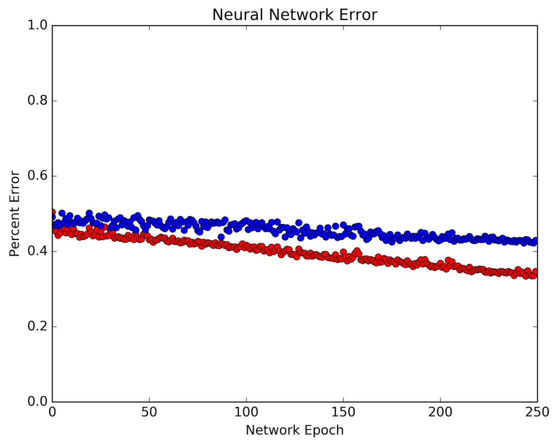
ICA+KM



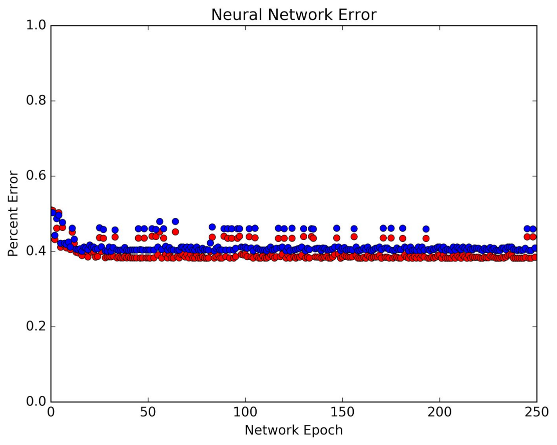
RandProj+EM



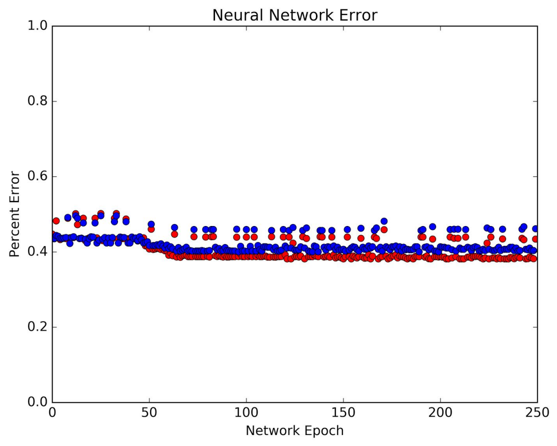
RandProj+KM



KBest+EM



KBest+KM



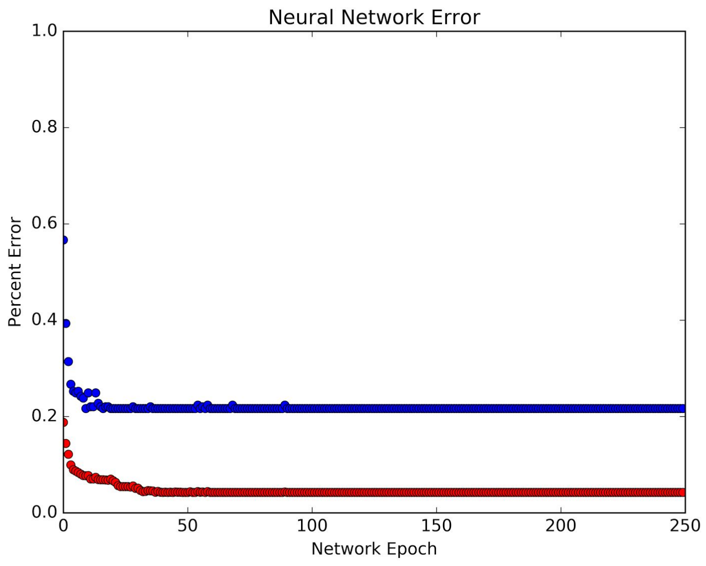
Once again, we see that ICA and Kbest performed below expectations, the reason of which was already explained above. It is interesting to note, however, that KBest was still able to learn and generalize on the bad features, achieving test error comparable to that of NN without preprocessing (40%).

For the dimensionality reduction algorithms that learned good features (PCA, RandProj), we see very good results. The error rates achieved were, in general, slightly better than non-preprocessed NNs. In fact, PCA was able to achieve as low as 25% training error, which is a great improvement on the 30% we had with NNs.

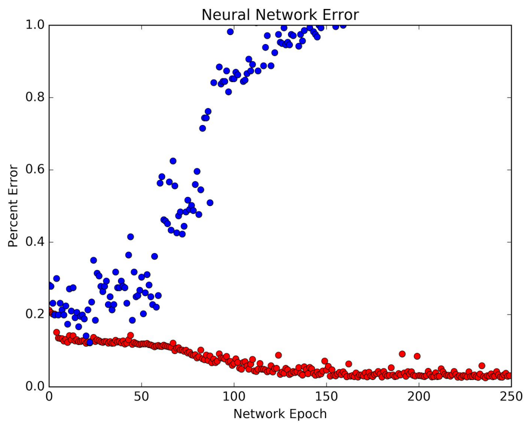
But the main takeaway from these graphs is how much faster the NNs are able to converge on comparably low error rates. This is a direct result of the dimensionality reduction algorithms being able to tackle the curse of dimensionality.

The most interesting results were when I ran this last experiment on the Mushrooms dataset. Dimensionality reduction, in some cases, made it possible to achieve *much* lower error rates on the training set (see KBest+EM achieving 0% training error), but many times that came at the cost of overfitting and not generalizing well.

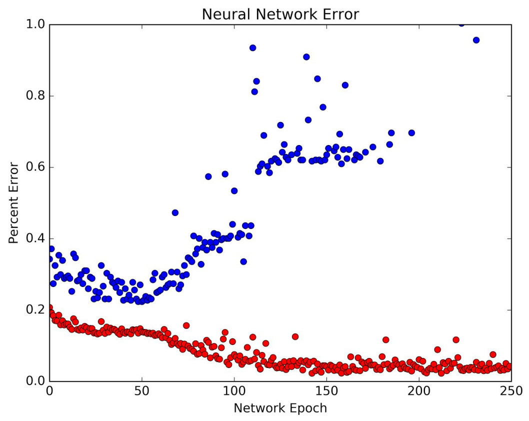
In fact, some examples of overfitting looked very extreme, as it was in the case of ICA.



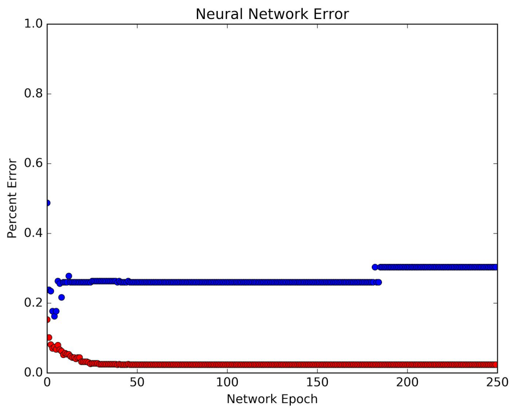
ICA



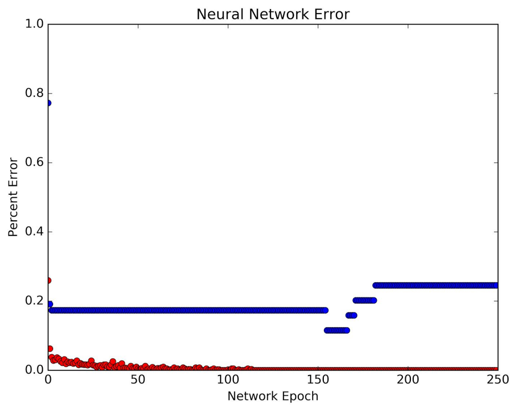
ICA+EM



KBest



KBest+EM



The rest of the plots for the mushroom experiment have been omitted, for brevity's sake.

Conclusion

As we’ve seen in this paper, dimensionality reduction algorithms play an important part in tackling the curse of dimensionality. They can make a classifier much easier to train, require many fewer training examples and iterations, and achieve good, if not better results. However, this has to be done carefully, because by reducing the dimensionality of one’s data, one may be more prone to overfitting it, or ignoring important features for classifications.

Were I to redo these experiments, the main thing I would do differently is time how long it takes for the neural networks to achieve a certain training error. Perhaps the biggest time bottleneck is the number of examples in the training set, but by lowering dimensionality, and thus minimizing its curse, we are sure to converge on small training errors with many fewer iterations (a phenomenon we’ve seen in the experiments in this paper).

Finally, the other thing I would do differently given more time is finding alternative ways of visualizing the data, that doesn’t rely on their classification labels, but rather on the attributes of the data itself.