ekalina3
CS-7641: Spring 2020

Unsupervised Learning & Dimensionality Reduction Writeup

Introduction

In this paper, we will explore unsupervised learning using the techniques of clustering and dimensionality reduction to make sense out of unlabeled data. This builds on the analysis of the two Kaggle datasets analyzed in assignment 1. The first being a mushroom classification dataset with 25 columns of discrete categorical information. The other is a Kiva Crowdfunding dataset consisting of 20 total columns with continuous, categorical and timeseries data. In a preprocessing step, we one-hot encode categorical columns in these datasets resulting in 120 features for the mushroom dataset and 250 features for the Kiva dataset. The mushroom data is interesting because it consists of imbalanced classes and all categorical features, while the Kiva data is interesting because of its varied feature types and sizable feature count after preprocessing. In assignment 1, we examined two classification problems: determining a mushroom's habitat from its characteristics and deciding if a loan is for an agricultural project based on its metadata. In this assignment, we will use unsupervised learning to reveal hidden variables as well as apply dimensionality reduction and rerun the neural network on the Kiva classification task. These investigations are performed using various algorithms provided by scikit-learn and Tensorflow.

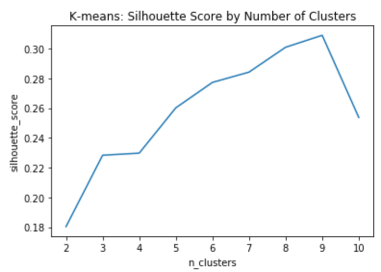
Clustering

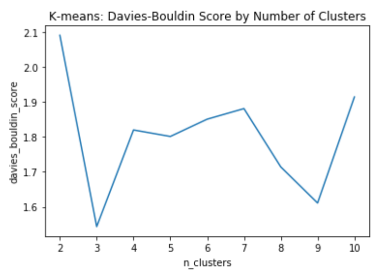
In this section, we'll apply K-means and Expectation-Maximization (EM) clustering to the datasets using sklearn's KMeans and GaussianMixture. To evaluate the performance of these two clustering algorithms, we'll use two distance metrics: the silhouette score and the Davies-Bouldin score. The silhouette score measures the distance between each point from it's cluster's centroid as well as other clusters. It's particularly useful because the scores are normalized between -1 and 1 allowing for comparison across datasets. The Davies-Bouldin score opts to measure the similarity within a cluster and distance from other clusters. According to the scikit-learn documentation, "clusters which are farther apart and less dispersed will result in a better score." (Scikit, 2019) Unlike the silhouette score, the Davies-Bouldin score isn't normalized and cannot be used for comparison across datasets.

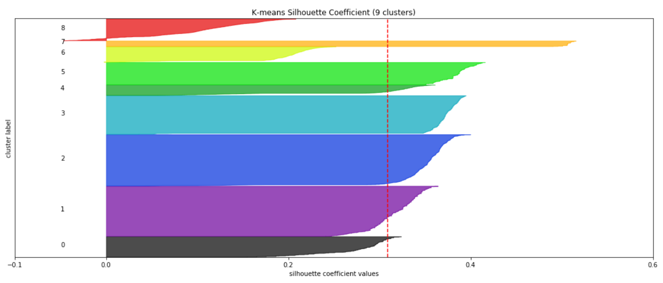
K-means Clustering

K-means is an iterative clustering algorithm that assigns points to distinct clusters. It's an example of hard clustering where a point can belong to only one cluster. It works by looping over all of the clusters and finding the cluster centroid that is closest to a point based on a distance metric.

<u>Mushroom Dataset Plots</u>

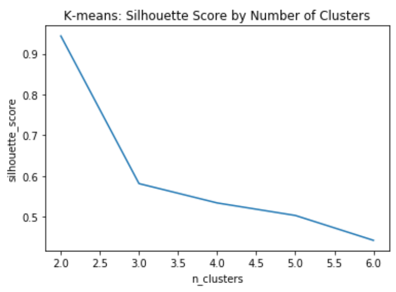


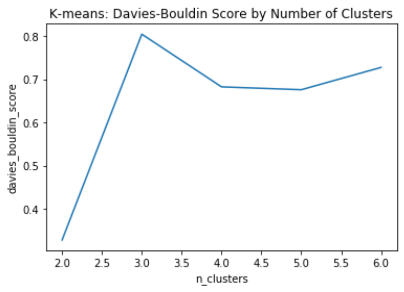


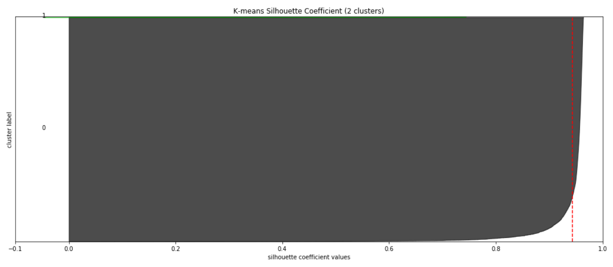


For the mushroom dataset using K-means clustering, the maximum silhouette score is produced by 9 clusters and the maximum Davies-Bouldin score is produced by 2 clusters. While the mean silhouette coefficient (~0.3) is not particularly high, the clusters are relatively well balanced and produce very few negative silhouette values meaning the algorithm is confident in its cluster assignment.

Kiva Dataset Plots





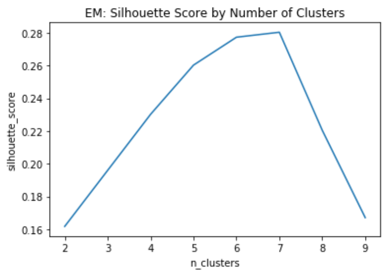


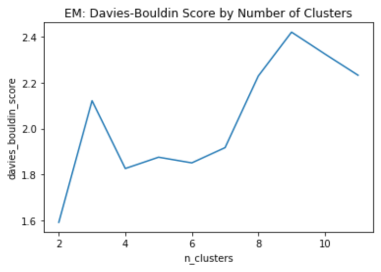
Both the silhouette and Davies-Bouldin scores are closer to agreement on the optimal number of clusters for the Kiva dataset. The Davies-Bouldin score suggests that 2 is the optimal number of clusters while the silhouette score has the optimum at 2. As we can see in the silhouette plot, while the silhouette coefficient values are quite high (~0.95) for the first cluster, the thickness of the grey bar means that almost all of the points are included in the first cluster making it a functionally useless clustering.

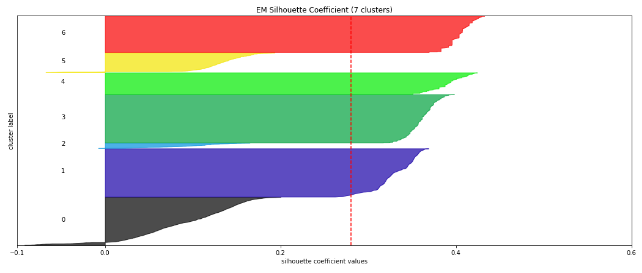
Expectation-Maximization (EM)

While EM is very similar to K-means, it has the added benefit of allowing soft clustering where a point can belong to more than one cluster. EM works by moving between two probabilistic calculations. The first phase is expectation, which partitions the points into soft clusters and the second phase is maximization, which computes the mean of the soft clusters. EM reaches its solution when the points stop moving between clusters.

Mushroom Dataset Plots

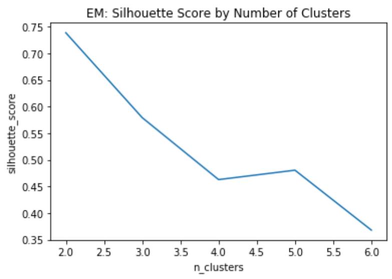


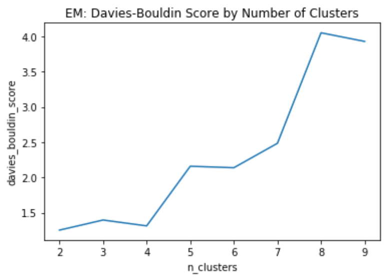


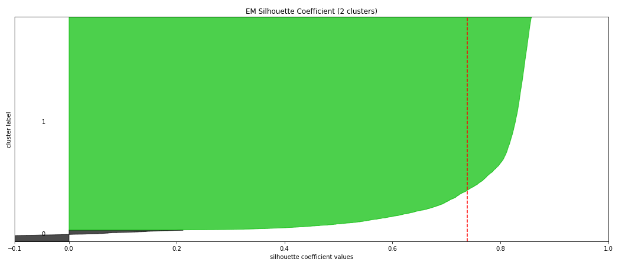


Interestingly the silhouette score above shows that additional clusters improve performance until 7 clusters at which point there is a steep drop off. On the other hand, the Davies-Bouldin scores are slightly more varied, but prefer 8-10 clusters. Something else to note is that several points in clusters 0, 2 and 5 have negative silhouette scores, which suggests that these points might be sorted into the wrong cluster.

Kiva Dataset Plots

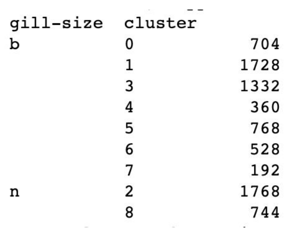






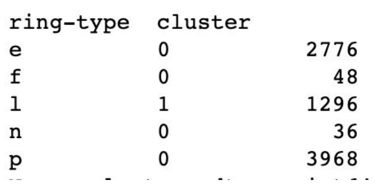
Unlike K-means, the Davies-Bouldin and silhouette scores do not agree on the ideal cluster size for EM, 8 and 2 respectively. The first two plots show that these metrics are effectively inverses of each other. Again we see that the allocation of points within the two silhouette clusters is quite uneven. The majority of the points in the grey cluster have negative silhouette coefficients meaning that they potentially belong in the green cluster making a single cluster and providing no further information. Both the K-means and EM clustering results for the Kiva dataset suggest that this dataset is not an ideal clustering candidate.

K-means (9 clusters)



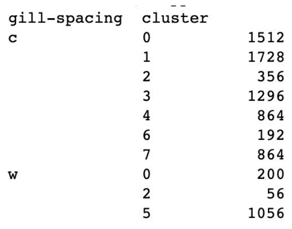
```
gill-size  cluster
b          0            704
           1           1728
           3           1332
           4            360
           5            768
           6            528
           7            192
n          2           1768
           8            744
```

EM (2 clusters)



```
ring-type  cluster
e          0          2776
f          0            48
l          1          1296
n          0            36
p          0          3968
```

EM (8 clusters)



```
gill-spacing  cluster
c             0          1512
              1          1728
              2           356
              3          1296
              4           864
              6           192
              7           864
w             0           200
              2            56
              5          1056
```

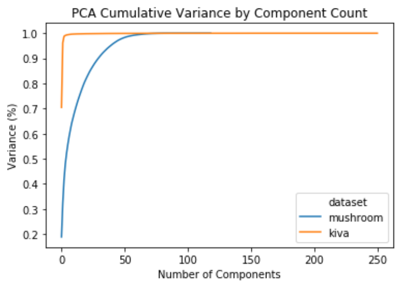
In an attempt to relate these clustering results back to the original features, I looked at how the cluster numbers related back to existing features. While there were some clear correlations between cluster number and existing features in the mushroom dataset (shown above), there was no clear 1:1 connection to the habitat classification problem explored in assignment 1.

Dimensionality Reduction

Dimensionality reduction is a filtering technique that reduces a feature set's dimensionality by distilling the most principal variables. It is important because limiting analysis to the most predictive features helps with the curse of dimensionality. In this section, we'll apply Principal Component Analysis (PCA), Independent Component Analysis (ICA), Randomized Projections (RP) and Linear Discriminant Analysis (LDA) to the two datasets. To quantify the performance of these dimensionality reduction techniques, we'll seek to find the highest variance variables for PCA and LDA and the most kurtotic variables for ICA and RP. Kurtosis is a statistical way of measuring a distribution's Gaussianity with a higher kurtosis value meaning a distribution is less normal. We'll use the mean square kurtosis to prevent negative and positive values from cancelling each other out.

Principal Component Analysis (PCA)

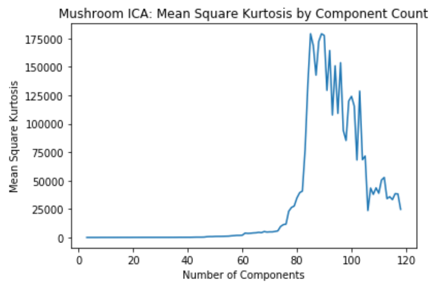
PCA is a form of feature extraction that summarizes data by finding the correlation between variables. It works by calculating the covariance matrix across features, then breaks the matrix down into direction and magnitude. From there, it transforms the data to align more closely with the most important directions. At the end of the run, you can throw away the directions with the least eigenvalue, which will have the lowest variance. We'll implement PCA using sklearn's PCA.



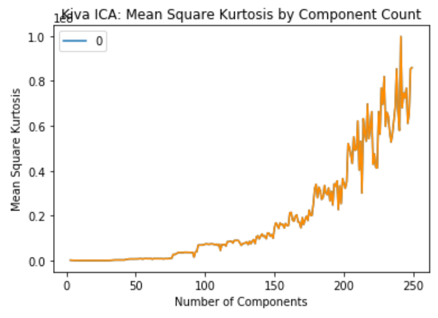
The figure to the left plots the cumulative variance provided as each additional component is added for both datasets. With the Kiva dataset, we see that 90% of the variance can be explained with only 2 components. In contrast, the variance of the mushroom dataset is more spread out with 90% of the variance explained by 31 components.

To determine how well these PCA runs modelled the original data, we ran an inverse transformation on the identified components, then compared them to the original data points to calculate the loss. The results showed that the information loss was minimal on both datasets with 1.3% loss for the mushroom dataset and 0.4% loss for the Kiva dataset.

Independent Component Analysis (ICA)

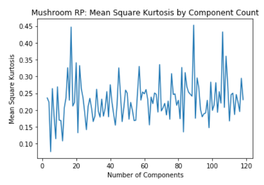


The goal of ICA is to linearly separate mixed sources (or features) in such a way that they are all statistically independent and non-Gaussian. It works by finding a linear transformation of the feature space so that individual features are mutually independent. ICA includes a preprocessing whitening step that makes sure all of the variables are treated equally followed by a series of linear transformations that convert the original features into a new feature space. The new feature space should try to maximize the mutual information with the original feature space so that the original features can be recovered. We'll implement ICA using sklearn's FastICA.

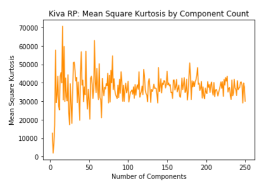


The adjacent plots show how the mean square kurtosis by component count differs for the mushroom and Kiva datasets. For the mushroom dataset, the most kurtotic number of components is 85, which is less than the max number of 120 features. The most kurtotic component count for the Kiva dataset is 241, which is much closer to the total feature count of 250. This suggests that the Kiva dataset is harder to model with fewer components, while the mushroom dataset can be summarized with a smaller feature space of roughly 67% of the information.

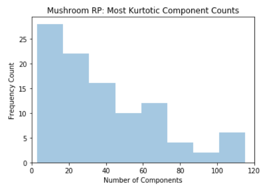
Randomized Projections (RP)

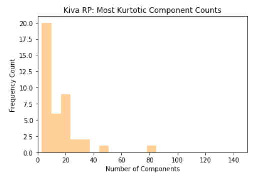


RP projects the original features onto a lower dimensional space using a randomly generated matrix. RP is computationally efficient, helps with the curse of dimensionality and tends to work well when you're reducing features to use on a classification problem. We'll implement RP using sklearn's GaussianRandomProjection.



The kurtosis plots to the left are interesting to me because they show how randomized projection is more volatile than the previously used dimensionality reduction techniques. This is due to the randomness inherent in this algorithm. While RP is much faster than the other techniques, this speedup comes with a possible loss of accuracy due to the fact that RP doesn't require the optimal linear transformation.

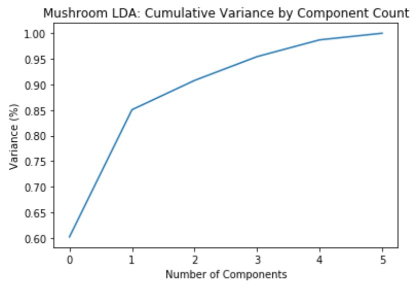




Since no two RP runs are alike, I ran RP several times on each dataset and looked at the frequency counts of the most kurtotic component count. The full distribution of values across runs is shown in the histograms to the left. In general, the distributions skewed to the right with smaller component counts being more kurtotic. For both datasets, 9 was the most frequent, most kurtotic component count.

Linear Discriminant Analysis (LDA)

LDA differs from the previous dimensionality reduction algorithms because it takes into account how the resulting components will be used. Both the features and target labels are fed into LDA with the goal of projecting the data into a lower dimensional space such that the labels are linearly separated. We'll implement LDA using sklearn's LinearDiscriminantAnalysis.

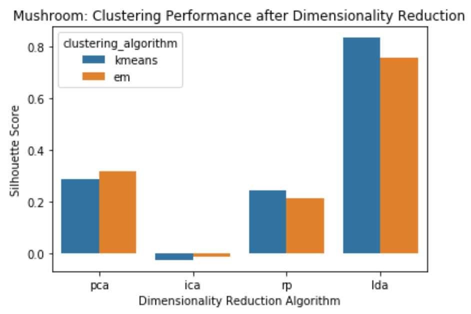


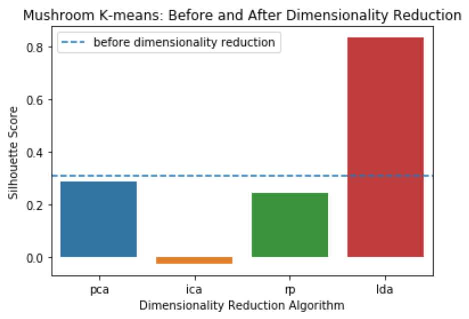
Only 6 LDA components for the mushroom dataset are plotted to the left because LDA reduces the number of dimensions from the original number of features to n_classes - 1. Since the mushroom task is a multiclass classification problem with 7 classes and the Kiva classification problem is binary, then the resulting number of components are 6 and 1. This is a drastic dimensionality reduction with 90% of the variance being described by 3 components for the mushroom dataset and a single component (down from 250 features) for the Kiva dataset.

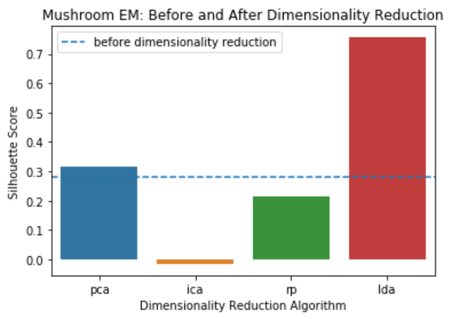
Clustering after Dimensionality Reduction

In this section, we ran each dimensionality reduction algorithm on the two datasets, then applied K-means and EM clustering to the newly reduced feature space. I chose to use the silhouette score as the comparison measure since it's a normalized metric allowing for a consistent comparison. The first plot in each row juxtaposes the performance after dimensionality reduction for K-means and EM, while the latter two plots look at the performance before and after dimensionality reduction for each clustering algorithm separately.

Mushroom Dataset Plots

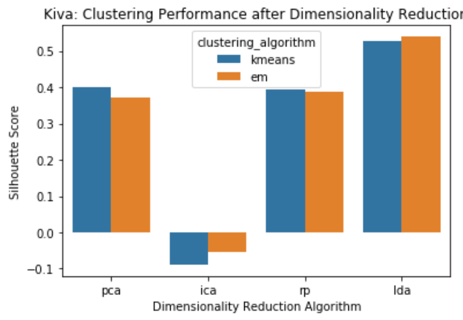


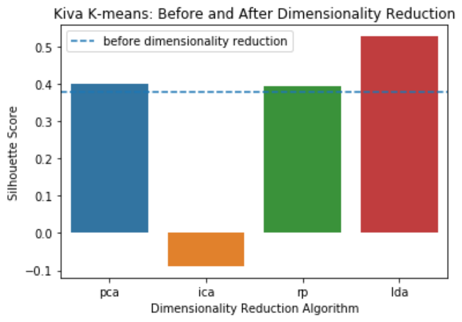


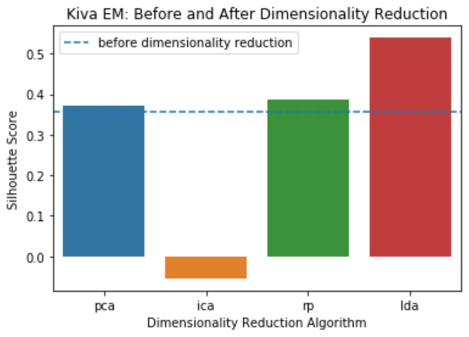


We can see in the plots above that dimensionality reduction produced mixed clustering outcomes. When performing mushroom clustering, PCA and LDA improved the silhouette score for EM and only LDA improved the score for K-means. Something to note is the large magnitude of improvement that came with LDA. The silhouette score without dimensionality reduction was 0.31 for K-means and 0.28 for EM and jumped to 0.83 and 0.76 respectively with a LDA preprocessing step.

Kiva Dataset Plots





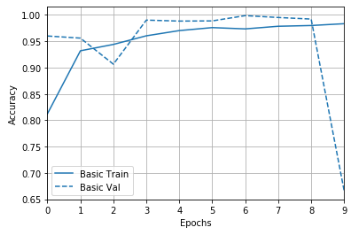
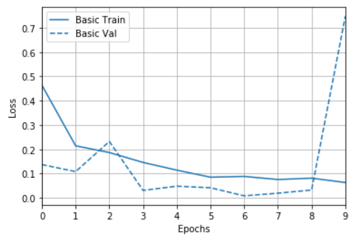


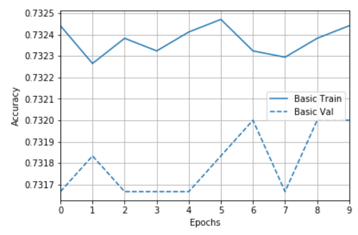
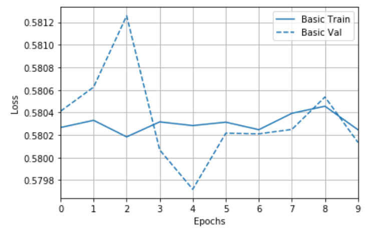
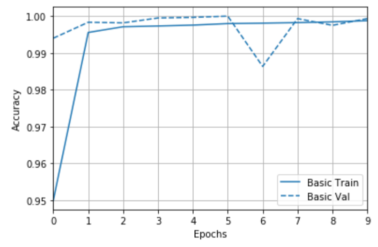
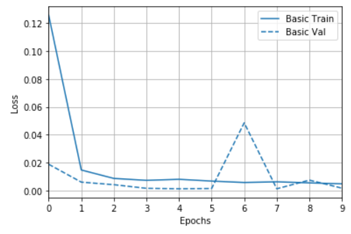
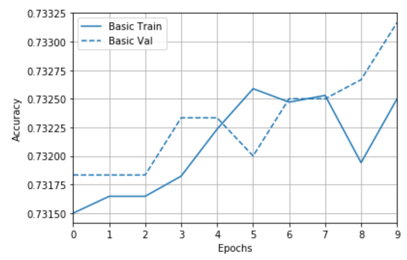
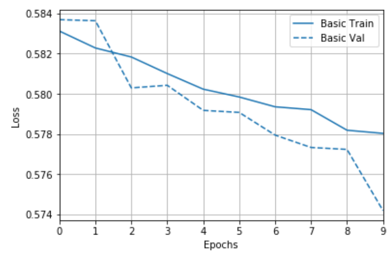
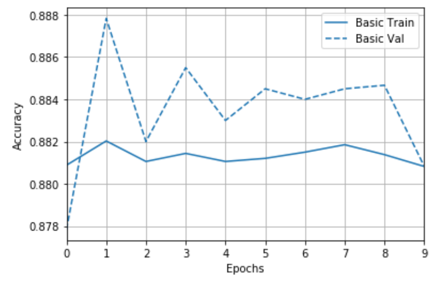
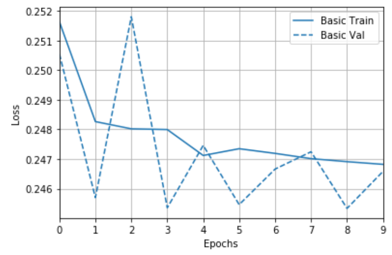
The above plots show that dimensionality reduction was successful in more cases for the Kiva dataset than the mushroom dataset. Dimensionality reduction with PCA, RP and LDA improved the silhouette score for both K-means and EM clustering. This general improvement in clustering could be attributed to the benefit of distilling the large, sparse feature space of the original dataset into a smaller feature set.

Across datasets and clustering algorithms, LDA generated the highest silhouette scores. I believe that LDA performed well because of how it factors in the classification problem when projecting the data into fewer dimensions. Another shared outcome was that ICA performed poorly across the board. In fact, ICA was the only dimensionality reduction algorithm that ended up with a negative silhouette score for the all clusterings.

Neural Network with Dimensionality Reduction

In this section, we applied the four dimensionality reduction algorithms to the Kiva dataset then ran the binary classification neural network from assignment one on the newly reduced feature space. The classification problem asks: Based on a loan's metadata, is the loan for an agricultural project? We then compare the accuracy and the loss outcomes from the original features to the PCA, ICA, RP and LDA generated components. Accuracy is defined as the number of correct classifications divided by the total number of classifications, while the cross-entropy loss function measures the distance between two probability distributions, in this case the true label and the predicted label. Since neural networks are iterative, the learning curve plots below demonstrate the performance of the training and test sets as function of iterations (or training epochs). Something to note is that we only ran each neural network for 10 training epochs due to long runtimes and while additional epochs would be preferable, the small number of epochs allow for a sufficient baseline performance comparison.

Dimensionality Reduction	**Accuracy**	**Loss**
None		

PCA *n_components = 2*		
ICA *n_components = 241*		
RP *n_components = 16*		
LDA *n_components = 1*		

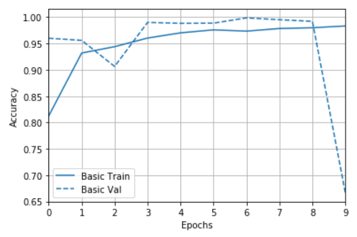
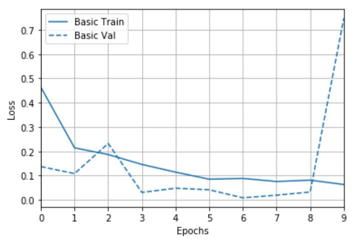
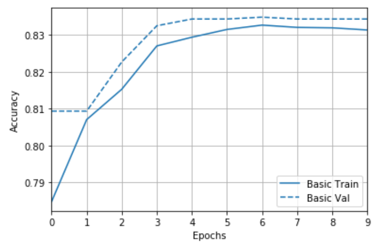
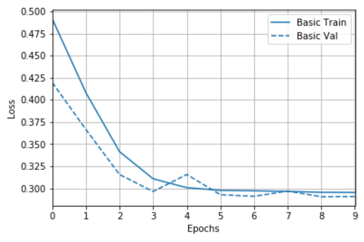
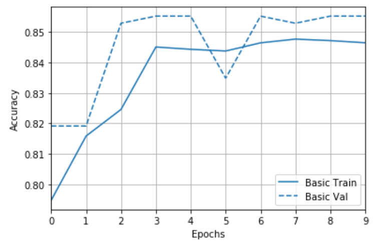
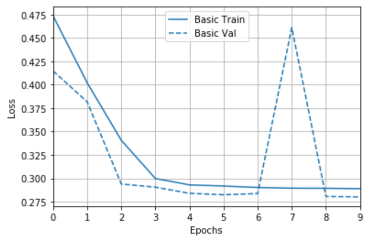
The neural networks performance from assignment 1 (without dimensionality reduction) already performed relatively well with an accuracy hovering around 97% and a loss of roughly 10%. In a reversal from other sections, ICA performs the best out of the four dimensionality reduction algorithms. The accuracy values are almost identical for the original and ICA runs, while ICA's loss values were tenhold better than the original at 1%. These improvements could be attributed to the fact that the ideal ICA number of components (241) is quite high and keeps most of the dimensionality of the original dataset. Neural networks are effective at finding the interplay between variables in complex feature spaces, so the high dimensionality might have helped the overall performance.

The PCA and RP runs had middling performance in comparison. Their accuracy scores were around 73% and their loss scores were about 58%. Both PCA and RP define a projection from a higher dimensional space into a lower dimensional space with the main difference being that PCA looks for the optimal directions while RP picks them randomly. This similarity in algorithm structure explains why their Kiva classification task performance is nearly identical.

Finally, LDA's performance on the Kiva classification task is between that of ICA and PCA/RP with an accuracy of ~88% and a loss of ~25%. What strikes me as interesting here is that LDA far outpaced the other dimensionality reduction algorithms when it came to clustering, but actually hurt the performance of the neural network. Continuing on my previous thought, these dimensionality reduction + neural network results suggest that dimensionality reduction doesn't do all that much for this particular Kiva classification task. In theory, the transformations that take place in dimensionality reduction algorithms could also be performed by the input layers of a neural network. I think dimensionality reduction preprocessing would be more beneficial for an extremely large input space neural network where the number of weights causes overfitting and a need for a huge amount of training data.

Neural Network with Clustering as Dimensionality Reduction

In this final section, we'll apply K-means and EM to the Kiva dataset and then use only the resulting clusters as a single input to the neural network to solve the agricultural loan classification task. This is a form of dimensionality reduction through clustering. The cluster sizes used will be the same optimal sizes (based on the silhouette score) used in previous sections. Again, we'll use the accuracy and cross-entropy loss scores to compare performance to the original neural network.

Clustering Algorithm	**Accuracy**	**Loss**
None		
K-means *n_clusters=9*		
Expectation-Maximization *n_clusters=7*		

Using the cluster numbers as an input feature resulted in a better performance than I expected, but worse than the original neural network. The K-means run with 9 clusters has an accuracy of 83% and a loss of 30% and the EM run with 7 clusters has an accuracy of 85% and a loss of 29%. Before running this experiment, I would have thought that a single input feature into a neural network wouldn't perform well, but these results show that a good portion of the information is encoded into these cluster numbers.

Something else that I found interesting about these results is that they tended to be less volatile than the PCA, ICA, RP and LDA results from the previous section. I don't know for certain what is causing the smoother curve with clustering, but my guess would be that the single clusters provide a more stable distinction between whether a loan is agricultural or not. In comparison, the standard dimensionality reduction algorithms tend to have higher dimensionality and might also be encoding some noise, which lends itself to increased volatility.

Conclusion

The results above illustrate the characteristics of clustering and dimensionality reduction techniques on two very different datasets. The mushroom dataset lent itself more to clustering than the Kiva dataset did. The Kiva results for both K-means and EM split the data points into two extremely imbalanced clusters. For the clustering experiments after dimensionality reduction algorithms were applied, LDA more readily improved the silhouette score of the resulting clusters for all combinations of dataset and clustering algorithm. In this same experiment, ICA hurt the clustering performance producing negative silhouette scores across the board. In addition when it comes to speed, ICA took significantly more time to run than the other filtering techniques. The quickest of all the filtering techniques was RP because, unlike PCA, it doesn't require the optimal linear transformation.

In the neural network runs, the accuracy and loss scores varied greatly depending on the dimensionality reduction technique used. ICA showed matched performance with the original neural network run without dimensionality reduction. PCA, RP and LDA hindered the performance, which I postulate is due to information loss from a less rich feature space. Finally, using the K-means and EM cluster numbers as a single input feature into the neural network degraded the original neural network performance, but did surprisingly well considering the reduction in dimensions from 120 to 1 for the mushroom dataset and 250 to 1 for the Kiva dataset.

Sources

"Is There Any Difference between PCA (Principal Component Analysis) and Random Projection When Preprocessing Data?" *Quora*, www.quora.com/Is-there-any-difference-between-PCA-Principal-component-analysis-and-random-projection-when-preprocessing-data.

Jerom, Samo. "Does Neural Networks Based Classification Need a Dimension Reduction?" *Cross Validated*, 2013, stats.stackexchange.com/questions/67986/does-neural-networks-based-classification-need-a-dimension-reduction.

Kumar, Rishav. "Understanding Principal Component Analysis." *Medium*, Medium, 10 Apr. 2018, medium.com/@aptrishu/understanding-principle-component-analysis-e32be0253ef0.

Ramsey, Parashara. "How Can We Say That a Clustering Quality Measure Is Good?" *ResearchGate*, 3 Mar. 2015, www.researchgate.net/post/How_can_we_say_that_a_clustering_quality_measure_is_good.

"Sklearn.metrics.davies_bouldin_score." *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html.