Randomized Optimization

Assignment 3

CS 7641

N. Ryan Karel (nkarel3)

# Introduction

We will be examining the business classification (ADD CITATION FOR BUSCLASS) and wine quality datasets previously seen in my first assignment for this class (Newman, Hettich, Blake, & Merz, 1998). As a quick reminder, these datasets principally vary in terms of their shapes. The business classification data has many more records and many more features (1,000 to be precise). Each feature here pertains to the presence of a single word in a natural language description of the business. In contrast, the wine dataset is shorter in length and only has twelve features, including values such as sulfate content and pH. The labels here are wine quality. As a reminder, there are 13 distinct classes captured in the business classification data, and only 3 for the wine dataset. Given the drastically differing sizes of the datasets, we should expect substantially different results when performing dimensionality reduction and clustering between them.

# Step 1

We begin with two clustering algorithms: K-Means and the Expectation-Maximization. We will first seek an appropriate number of clusters for each algorithm on each of our datasets in a completely unsupervised way: we do not want to consider the corresponding labels of records or clusters in this first step. After we have committed to a particular number of clusters, we will consider labels to help evaluate the utility of our clusters.

## Determining the number of clusters

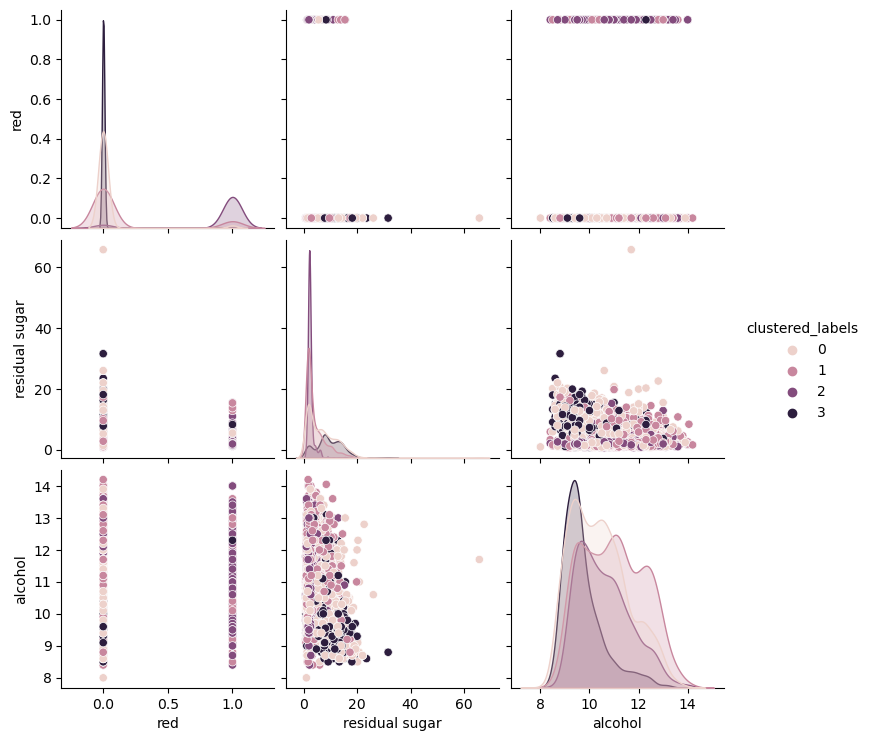
To select the right number of clusters for K-Means without using any labels, we turn to two distinct scores that will simultaneously help us optimize the clustering of like datapoints (reducing intra-cluster variance) and optimize the clustering of unlike datapoints (increasing inter-cluster variance). The metrics that allow us to do this are the Silhouette score (CITATION HERE) and Calinski-Harabasz score (CITIATION HERE).

Higher values of each imply more optimal clusters. For the business classification dataset, both measures were in agreement that the K-Means algorithm should use two clusters. For the wine dataset, the results were a little more mixed. I ultimately decided to use the Calinski-Harabasz recommendation of four clusters and was fairly pleased with the results later when it came to validating the clusters, though that did not factor into my decision at the time to use four. I will admit that after performing the majority of the analysis, I was curious what would happen if I used the three cluster recommendation that seems to be moderately favored by both metrics, but the resulting separations were less intuitive to me. We also reviewed a pairwise plot for the wine dataset, though this was far less interesting for the business classification dataset given the relative meaninglessness of its features.

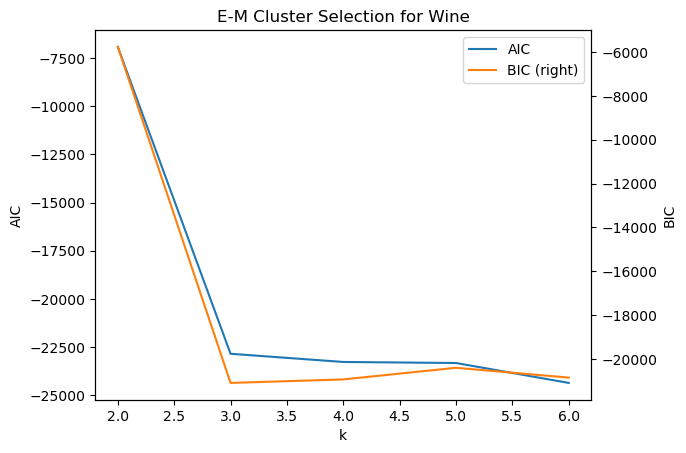
Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated



For the Expectation-Maximization clustering, we used information-theoretic scores like AIC and BIC to select the right number of clusters. Since we are using these information criteria, we seek to minimize these values. There is less agreement between AIC and BIC than the metrics we used for tuning K-Means when it comes to the business classification dataset, though the wine dataset recommendations were fairly consistent between the two.

Chart, line chart

Description automatically generated

For the business classification dataset, we opt to use the two-cluster recommendation (driven by the BIC measure) for two reasons. Firstly, the BIC measure takes into account the dataset size, which is fairly big for the business dataset. The AIC measure does not respond to changes in dataset size. Secondly, selecting two clusters will allow for easier comparison with the two-cluster recommendation for K-Means.

## Cluster evaluation

### K-Means

When reviewing the resulting clusters’ general labelling, we see some encouraging trends on both of these datasets. When tuning the K-Means algorithm for our business classification task, we might have found the optimal number of clusters unsatisfying. Given that there are thirteen possible labels but only two clusters, is some signal slipping through the cracks? When we review which labels are in the two clusters, we find that the K-Means algorithm was finding meaningful separation in the data, just not the one that our labels would dictate. Take a look at the following table, which shows the distribution of labels within each cluster.

|  |  |  |  |
| --- | --- | --- | --- |
| **Industry Group** | **Cluster 0** | **Cluster 1** | **Difference** |
| *Financials* | 0.14 | 0.07 | -0.07 |
| *Information Technology* | 0.13 | 0.06 | -0.07 |
| *Professional Services* | 0.12 | 0.09 | -0.03 |
| *Healthcare* | 0.1 | 0.09 | -0.01 |
| *Commercial Services & Supplies* | 0.09 | 0.09 | 0 |
| *Corporate Services* | 0.09 | 0.09 | 0 |
| *Media, Marketing & Sales* | 0.09 | 0.09 | 0 |
| *Energy & Utilities* | 0.07 | 0.08 | 0.01 |
| *Consumer Discretionary* | 0.03 | 0.04 | 0.01 |
| *Transportation & Logistics* | 0.07 | 0.09 | 0.02 |
| *Materials* | 0.02 | 0.04 | 0.02 |
| *Industrials* | 0.02 | 0.06 | 0.04 |
| *Consumer Staples* | 0.03 | 0.1 | 0.07 |

We see that Cluster 0 seems to include a greater proportion of white collar industries (e.g. Financials, IT) while Cluster 1 seems to hold a greater proportion of blue collar industries (e.g. Industrials, Materials). This was exciting to see, as it provided some proof that despite the absence of labels in the fitting process, the K-Means algorithm was able to split the data up in a way that was understandable to humans.

We see similar results from the wine classification K-Means clusters by reviewing which qualities are present in which clusters. Four clusters may have seemed overkill for a dataset with only three possible labels, but there is still a trend in the more prevalent labels of each cluster.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Wine Quality** | **Cluster 0** | **Cluster 1** | **Cluster 2** | **Cluster 3** |
| *Low* | 0.33 | 0.28 | 0.42 | 0.5 |
| *Medium* | 0.43 | 0.46 | 0.42 | 0.42 |
| *High* | 0.24 | 0.25 | 0.15 | 0.08 |

Although it struggled with the medium quality wine, the clusters are able to find some separation between the low and high quality items.

### Expectation-Maximization

Not unlike K-Means, The E-M algorithm did suggest that 2 clusters were optimal for the business classification data, though it did not produce as interpretable trends in the corresponding labels. The only rationale I can think of to justify this result is the fact that the algorithm is more equipped to deal with ambiguity through its emphasis on probability of belonging to a particular Gaussian process, and hence there are more data points that the algorithm estimates could truly go either way.

|  |  |  |  |
| --- | --- | --- | --- |
| **Industry Group** | **Cluster 0** | **Cluster 1** | **Difference** |
| *Information Technology* | 0.22 | 0.08 | -0.14 |
| *Media, Marketing & Sales* | 0.19 | 0.09 | -0.1 |
| *Energy & Utilities* | 0.09 | 0.08 | -0.01 |
| *Consumer Discretionary* | 0.03 | 0.04 | 0.01 |
| *Materials* | 0.03 | 0.04 | 0.01 |
| *Consumer Staples* | 0.06 | 0.07 | 0.01 |
| *Commercial Services & Supplies* | 0.07 | 0.09 | 0.02 |
| *Corporate Services* | 0.07 | 0.09 | 0.02 |
| *Transportation & Logistics* | 0.07 | 0.09 | 0.02 |
| *Financials* | 0.06 | 0.09 | 0.03 |
| *Industrials* | 0.02 | 0.05 | 0.03 |
| *Healthcare* | 0.05 | 0.1 | 0.05 |
| *Professional Services* | 0.04 | 0.1 | 0.06 |

In contrast, the wine quality clusters show a more pronounced trend than the corresponding results of the K-Means algorithm.

|  |  |  |  |
| --- | --- | --- | --- |
| **Wine Quality** | **Cluster 0** | **Cluster 1** | **Cluster 2** |
| *Low* | 0.62 | 0.47 | 0.27 |
| *Medium* | 0.36 | 0.39 | 0.47 |
| *High* | 0.03 | 0.14 | 0.26 |

This could be due simply to the fact that we are using three clusters instead of four, and hence better matching the distribution of the various wine qualities.

# Step 2

We now shift focus to our dimensionality reduction techniques, including PCA, ICA, Random Projections, and Linear Discriminant Analysis.

## PCA

Given the width of the business classification dataset and the expected correlations between much of its features, I fully expected the PCA algorithm to dramatically reduce the number of columns and still explain a large portion of the variance of the columns. Strangely, this did not happen. Instead, the algorithm still needed to use just over 850 features to capture 95% of the variance within this dataset. A mere 15% reduction was certainly unexpected, which led me to believe that the columns were not nearly as correlated as I had suspected.

Chart

Description automatically generated

This stands in contrast to the PCA algorithms performance on the wine dataset, on which it was able to capture a substantial amount of variance (defined as being over 95%) with only 3 components.

Chart

Description automatically generated

The fact that the algorithm could explain effectively 100% of the variance with only five components caused me to wonder if we could perfectly reconstruct the dataset using just these six components. This did appear to be the case. Why were we able to do this? When we review the individual features for the wine dataset, we do see values that should be heavily correlated, for example acidity measures and pH, residual sugar and alcohol, et cetera.

Chart, line chart

Description automatically generated

In summary, to achieve our requisite explained variance ratio, we could only reduce the business classification dataset by 15% of its columns but an entire 75% for the wine dataset.

## ICA

We took a similar approach when selecting the appropriate number of components for the ICA algorithm, though this time the focus was on maximizing the kurtosis of the resulting columns. Interestingly, more components were helpful for the business classification dataset while fewer were more helpful for the wine dataset. Though as we will see later in Step 5, the optimal values here based on kurtosis did not necessarily lead to the optimal dimensionality reduction for the purpose of better classification predictions.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

## Random Projections

For the randomized projections dimension reduction technique, we are mostly interested in how well we can reconstruct the dataset from its reduced state. Generally speaking, the more randomized projections, the better we are able to do this. This appeared to be the case across both of our datasets.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Given the inherently random process at play here, it is beneficial for us to run the algorithm several times to see if we get similar results. We saw something very similar to the above shapes for the majority of the runs, and the optimal number of components did not differ materially. Even with the larger number of re-runs, it was not time consuming to check this given how time-efficient the random projections algorithm is.

## Linear Discriminant Analysis

The final dimension reduction algorithm was allowed to make use of supervised data in the form of the datasets’ labels. We chose the LDA algorithm as our fourth approach, and the results were encouraging, especially once applied to the supervised training of the neural network after using this technique. As a reminder, the algorithm allows at most one less component than there are classes in our labels, so we could try up to 12 components for business classification and only two for the wine dataset. Not dissimilar to other shapes seen throughout Step 2, the business classification required more components to achieve the 95% cumulative variance threshold. On the other hand, a single component was sufficient for the wine dataset, though two components gave us near 100% capture of variance in the dataset. This correlates to the results we saw on the PCA reduction for the wine dataset, where a few linear combinations were nearly completely sufficient to recreate the dataset.

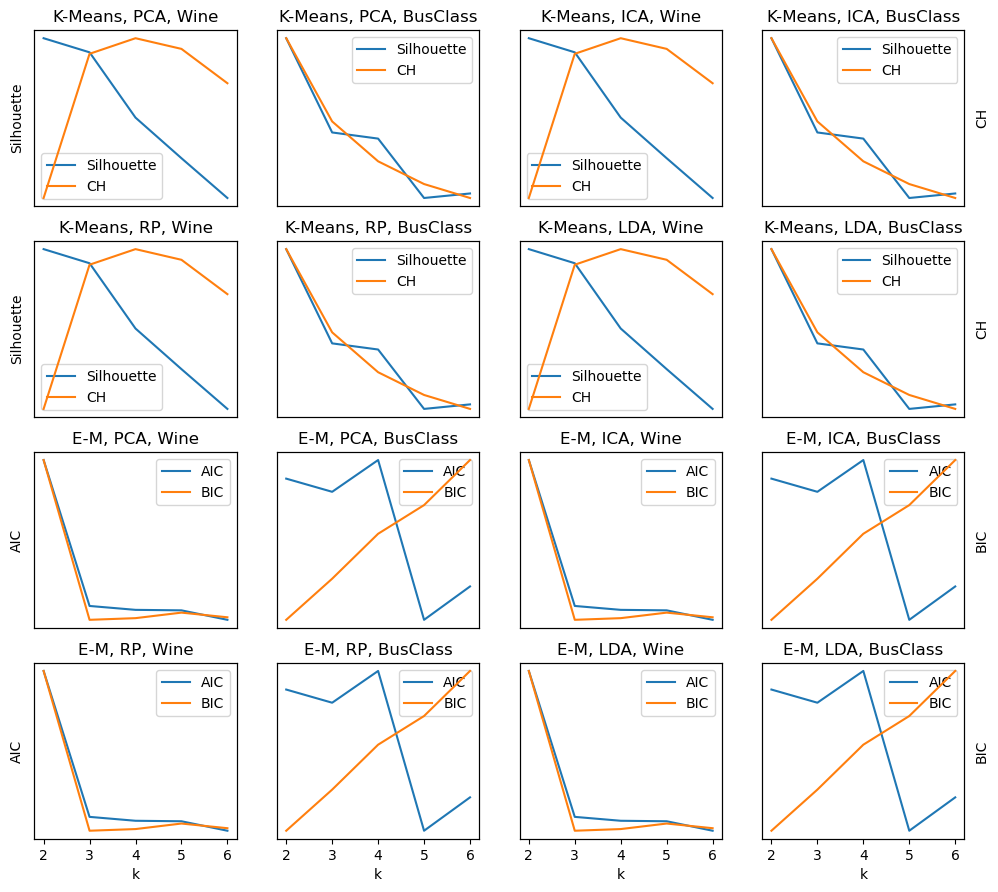
Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

# Step 3

Next we wanted to run both clustering algorithms on both datasets after all four dimensionality reduction algorithms, resulting in 16 different optimization tasks. Having more views on the optimal number of clusters for each of our datasets will help us solidify the case for a particular number of clusters.



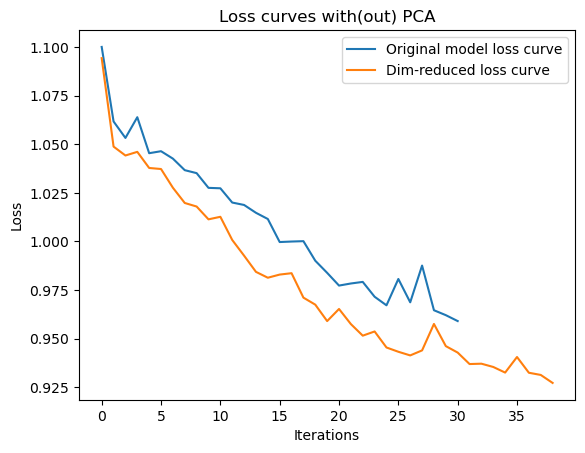
# Step 4

Here we transition to a more practical usage of dimensionality reduction. We will use its output to train neural networks and see how it performs to the original networks trained back in assignment 1. We will focus on the wine dataset for this section. There will be two primary points of comparison: loss curve plots, to see how quickly signal in the training dataset is captured, and ROC AUC OvO (as defined in assignment 1) scores on the same holdout data used previously.

Note that we are using early stopping with 10% of the training data set aside for validation to avoid overfitting and retraining the neural network on each combination of preprocessing steps using a grid search over possible regularization values and hidden layer sizes. After this hyperparameter tuning is performed, the best estimator is fit once again on the data and the loss curves are compared against the original model’s loss curves (retrained for the purposes of this assignment) on non-reduced data.

We see varying degrees of improvement in the supervised learning under the various dimensionality reduction approaches. PCA resulted in faster training signal capture, but did not ultimately increase performance on our holdout set. ICA did not aid training or holdout performance. He we also played with the number of components to use in ICA, deviating from the recommendation found in Step 2. Increasing the number of components up to 8 instead of the recommended 2 improved performance, but did not allow us to beat the original non-reduced model. The model after applying random projections initially found training signal at a similar rate to the original model, but stalled on our early stopping validation data and the holdout data.

The real winner here was applying the model after linear discriminant analysis, which *a priori* I would have assumed would have resulted in overfitting, given that the dimensionality reduction can see the true labels before being fit by the neural network. This fear was given some credence by the fast and effective drop in training loss observed in our loss curve, but not borne out by the performance on the holdout set, which was a decent improvement over the original model’s holdout performance by a little over 5%.

Chart, line chart

Description automatically generated

Chart, line chart

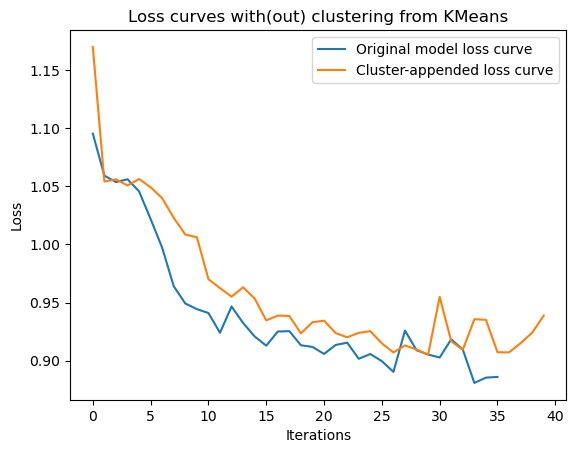
Description automatically generatedChart, line chart

Description automatically generated

# Step 5

Now we take a similar tack but with appending our learned clusters to the original data and then re-running our neural network on the resulting datasets. We use both K-Means and Expectation-Maximization, as in Step 1, to accomplish this. Like in Step 4, we use grid search to retune the models with the new appended variables.

As we can see from the loss curves, the appended clusters do not allow us to capture training data any better than the original model. The performance on the holdout sets is roughly the same, as well, though slightly less.

Chart, line chart

Description automatically generated

# References

Hayes, G. (2019). *mlrose: Machine Learning, Randomized Optimization and SEarch package for Python.* Retrieved from https://github.com/gkhayes/mlrose

Isbell, C., Viola, P., & De Bonet, J. (1997). *MIMIC: Finding Optima by Estimating Probability Densities.* MIT, Cambridge. Retrieved from https://www.cc.gatech.edu/~isbell/papers/isbell-mimic-nips-1997.pdf

Newman, D., Hettich, S., Blake, C., & Merz, C. (1998). *UCI Repository of machine learning databases*. Retrieved from http://www.ics.uci.edu/~mlearn/MLRepository.html

scikit-learn. (2022). *sklearn.neural\_network.MLPClassifier*. Retrieved from scikit-learn.org: https://scikit-learn.org/stable/modules/generated/sklearn.neural\_network.MLPClassifier.html