Project 2 Report

Kumar Gaurav, Shashank Sule, Alexander Wikner

November 2020

Problem 1

K-means clustering

To pick a complete set of ratings, we picked a submatrix of maximal size from the initial matrix with missing entries. To do so we note that this is equivalent to solving for the largest biclique in a bipartite graph. Using the maximal biclique enumeration algorithm described in [1] and choosing the biclique with the greatest number of entries, we obtained the following selection of rows and columns:

```
rows=[0;1;2;3;7;10;12;15;24;28]
columns=[0;1;2;3;4;5;7;8;9;10;12;14;15;17;18;19]
```

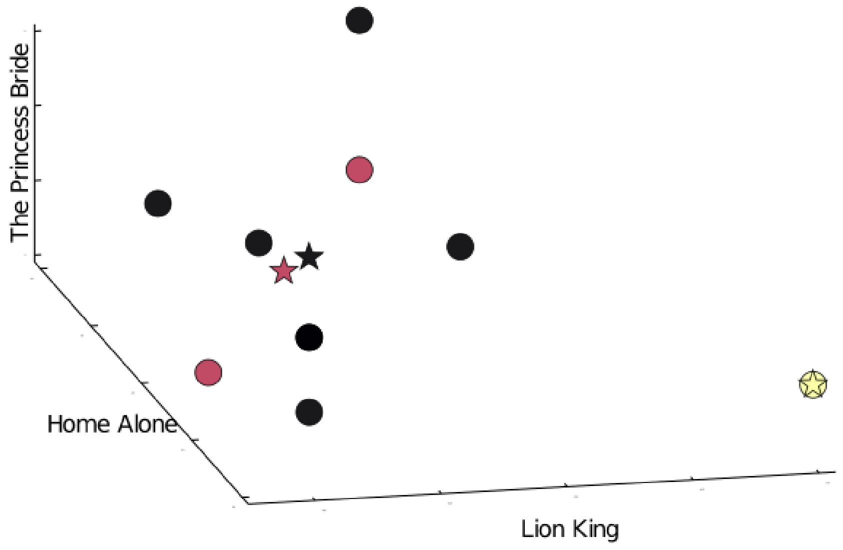

Figure 1: Data from complete submatrix M with 9 users clustered by k-means projected onto the first three features. The clusters centers are denoted by stars and the data by circles.

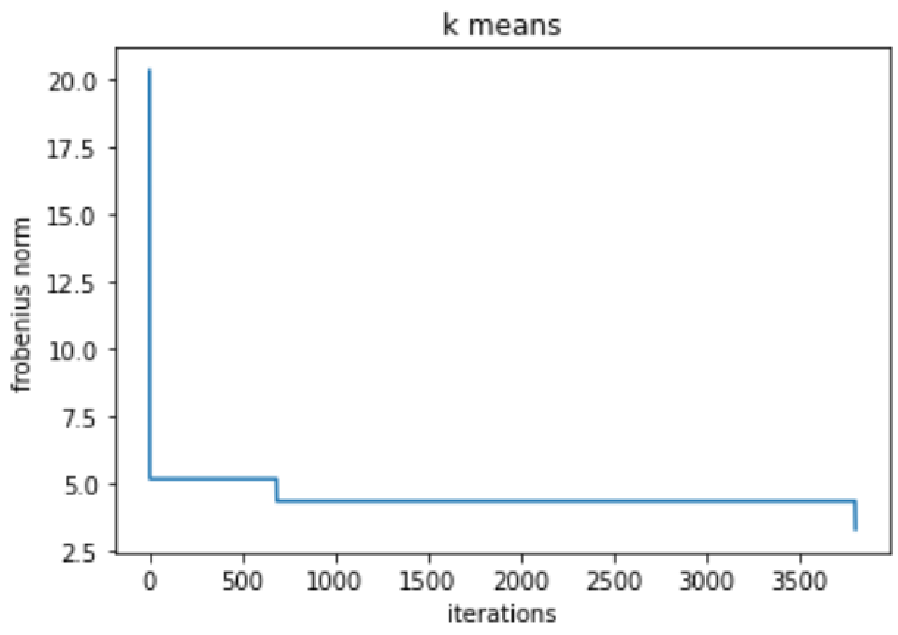

Figure 2: Convergence statistics for the k-means residual $||M - LR||_F$. Here $L_{ij} = 1$ if the ith row of M is contained in the jth cluster and 0 otherwise; R is the set of cluster centers. The minimum is `3.0`

Non-Negative Matrix Factorization

We compute $M = WH$ via Projected Gradient Descent, Lee-Seung, and a combination of the two. In the following data we initialize the algorithms with $W = |U|$ and $H = |\Sigma V^\top|$ where $|X|_{ij} = |X_{ij}|$. Results using other initializations can be found in the code.

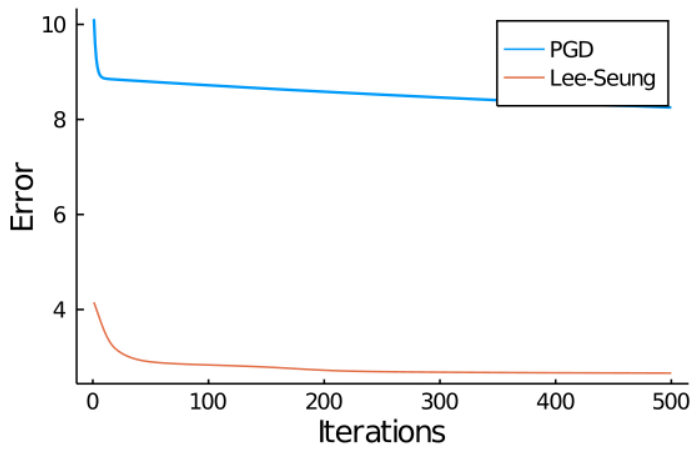

Figure 3: Comparing the first 500 iterations of Projected Gradient Descent (PGD) with $\alpha = 0.01$ and Lee-Seung (LS). Note that PGD has a quick initial decay; LS is more accurate but slower

For the combined approach we used two methods. Method 1 involves computing 5000 iterations of PGD and then 5000 iterations of LS. In Method 2, we adopt the following routine: if PGD gets stuck for a while (say for 100 iterations) then we switch to LS.

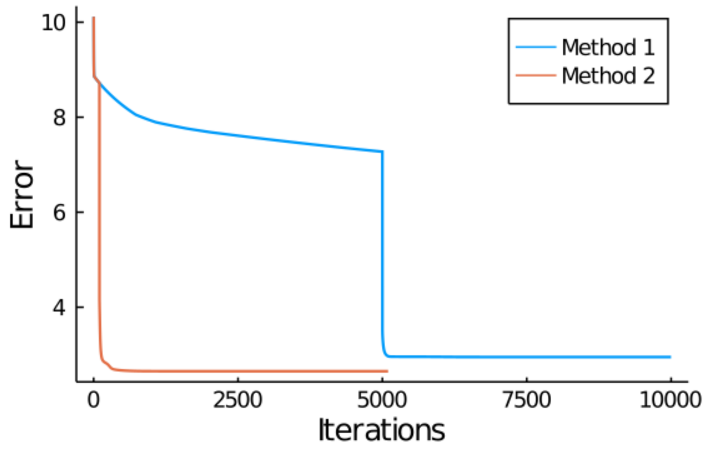

Figure 4: 5000/5000 iterations of PGD and Lee Seung (blue) and best 100/5000 iterations of PGD and Lee Seung (orange). Note that the first subroutine exhibits good progress during its PGD part and collapses rapidly to a minimum when it switches to LS. On the other hand, method 2 is slightly more accurate but stays at the minimum.

Problem 2

For the alternating iterations routine we find that $\lambda = 0.01$ is the best among $\lambda = [0.01, 0.1, 0.05]$. We also found that the Nuclear Norm update typically produces higher rank matrices with lower error, in fewer iterations and in a shorter time than alternating iterations. In addition, when we constrain the value of λ for the nuclear norm trick to be such that the rank of the matrix produced is the same as that of the alternating iterations, we find that the nuclear norm matrix has a lower residual $||M - XY^\top||$ and converges faster than alternating iterations. For example, fixing $\lambda = 10$, a rank-8 approximation using the nuclear norm trick can be computed in 4.4×10^{-3} seconds, while alternating iterations takes 8.9×10^{-3} seconds. Consequently, the alternating iteration method is easier to implement while the Nuclear norm trick method is more efficient.

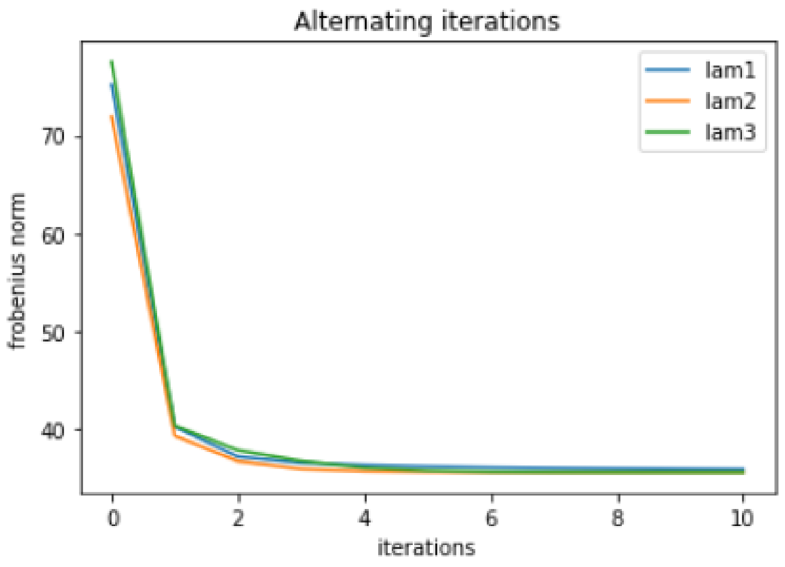

Figure 5: Graph for frobenius norm of residual vs iterations for $\lambda = [0.01, 0.1, 0.05]$

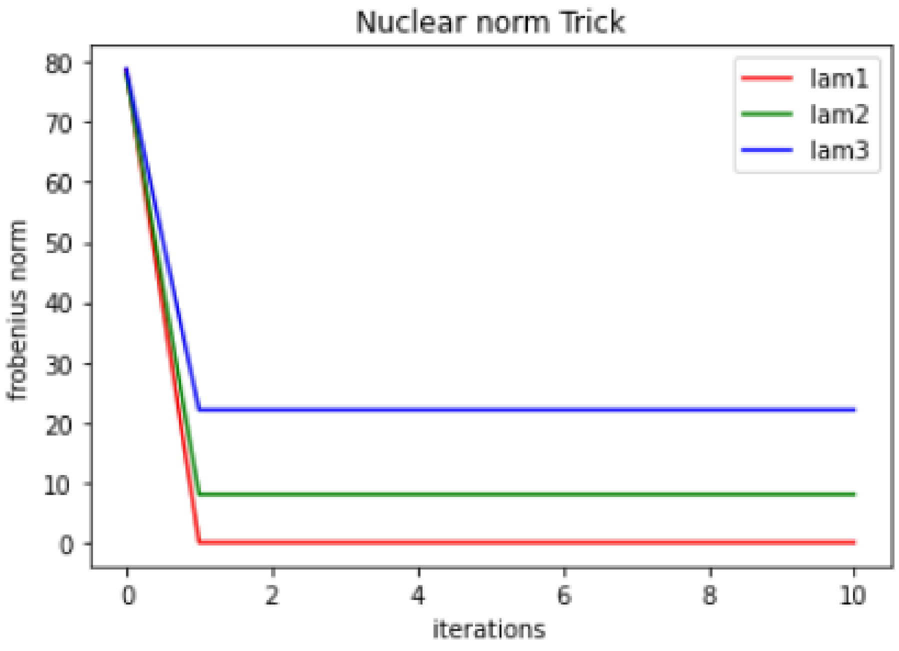

Figure 6: Graph for frobenius norm of residual vs iterations for $\lambda = [1.01, 5, 10]$. The resulting ranks of the matrix M produces by this method (for a maximum possible rank of 20) are, respectively, 20, 16, and 8.

Problem 3

The best value of c that works is 80.

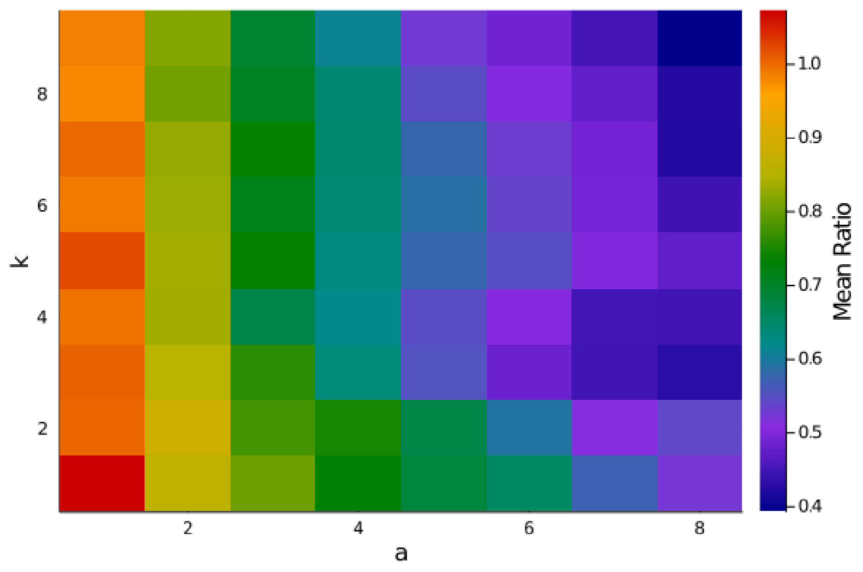

Figure 7: Mean ratio $||M - M_k||/||M - CUR||$ for $c = r = ak$ for $2 \leq k \leq 10$, $1 \leq a \leq 8$. Note that increasing the expected number of columns by adjusting a and keeping the rank k constant does not decrease the mean ratio as appreciably as increasing k while keeping a fixed

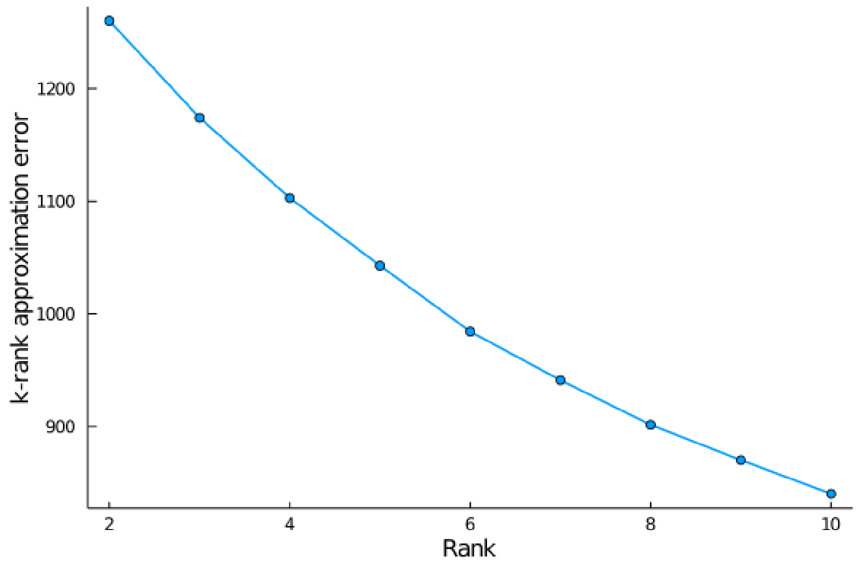

Figure 8: Error $||M - M_k||_F$ for $2 \leq k \leq 10$. Note that $||M - M_k||_F$ is $\sqrt{\sigma_{k+1}^2 + \ldots \sigma_R^2}$, Here R is the rank of M; incidentally, $R = 139$.

Problem 4

The data set used in this problem is a matrix, M, where each entry M_{ij} indicates either the presence of word j in document i (Binary) or the number of times word j appears in document i (Count). The full matrix contains data from 139 documents with 18,446 unique words. Each document is additionally classified as coming from either Indiana or Florida. We calculate the normalized leverage scores of each column, defined as:

$$\mu_j = 1/k \sum_{\xi=1}^{k} (v_j^{\xi})^2. \tag{1}$$

In this equation, v_j^{ξ} is ξ^{th} row of j^{th} column of V_k, the rank k SVD of M. For the following results, we use a rank $k = 10$ SVD. We find the columns of the data matrix corresponding to the following words to have the largest normalized leverage scores:

Binary Feature Data	Count Feature Data
gif	gif
and	shim
the	the
for	bullet
with	limo
contact	and
our	del
evansville	radio
your	los
are	uruguaysc

Table 1: Largest leverage score features (ordered top to bottom greatest to least) for both methods for forming the data matrix.

We note that neither method results in words that are suitable for classifying the documents between Indiana and Florida. Many of the words are articles, which are common in all documents, or other simple, common words (are, gif, with, our). The only word that appears relevant to classification is evansville, a town in Indiana, but this word only appears using one method and appears near the bottom of our leverage score top 10.

Following the direction of Part(b), we use only the 5 columns with the largest leverage scores (again computed with rank $k = 10$) and use Principal Component Analysis (PCA) to compute the first two principal components of the reduced matrix. The results for the count matrix are shown in Fig.(9). We see that there is no clear separation between the documents corresponding either Indiana or Florida, indicating that we need to use some other selection technique.

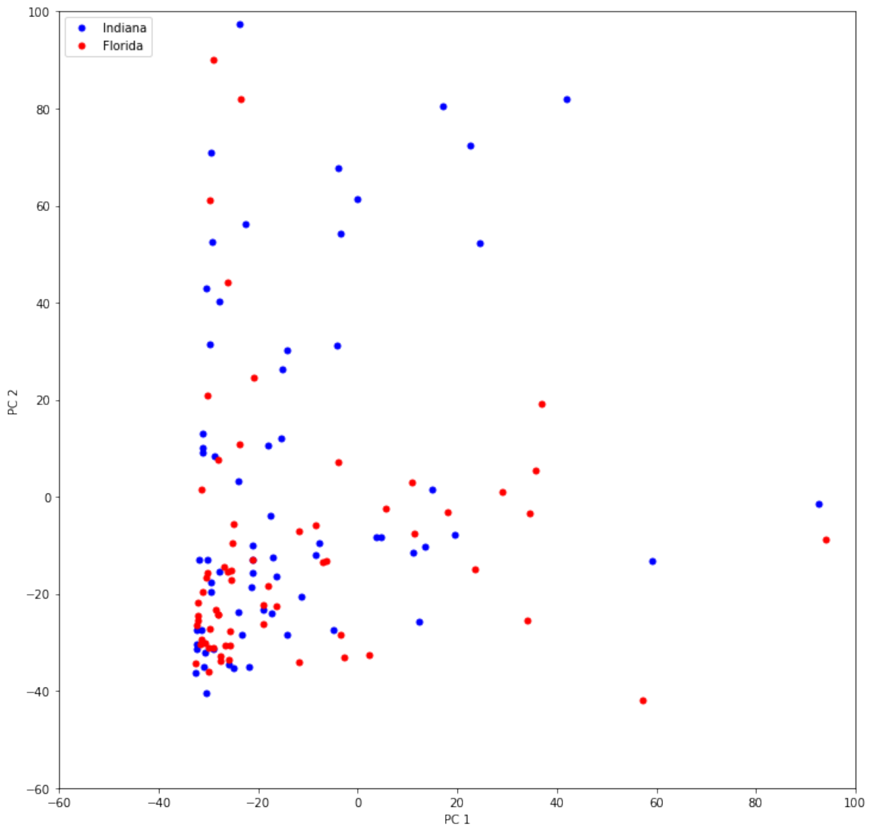

Figure 9: Documents from Indiana or Florida projected onto the first 2 principal components of the reduced matrix using only those 5 word columns corresponding to those with the largest normalized leverage scores. We see that there is no clear separation between documents from these two states.

To obtain a better selection of columns, we choose to select all words that correspond to phone numbers, area codes, and zip codes, as well as words that are not numbers and are longer than 6 letters (since we expect shorter words to be less specific to either state than longer ones, e.g., florida or evansville). After reducing the matrix, we are left with only 10,110 remaining words.

For this reduced data set, we use PCA to compute the first 2 principal components of (a) the full reduced matrix and (b) the matrix where we have only kept 5 columns with the largest normalized leverage scores (computed with $k = 10$). Figure 10 shows the results for each of these matrices using the count word data. We note that while neither results in data sets that are perfectly separable between the two states, both appear to be approximately separable. In addition, only the first principal component appears to be relevant to separating the data in the case of the maximal leverage scores matrix. For the binary data case shown in Fig. 11, we do not see that the full reduced matrix is separable from the first two PC's, but once again the matrix with only the largest leverage score columns is separable along only the first PC.

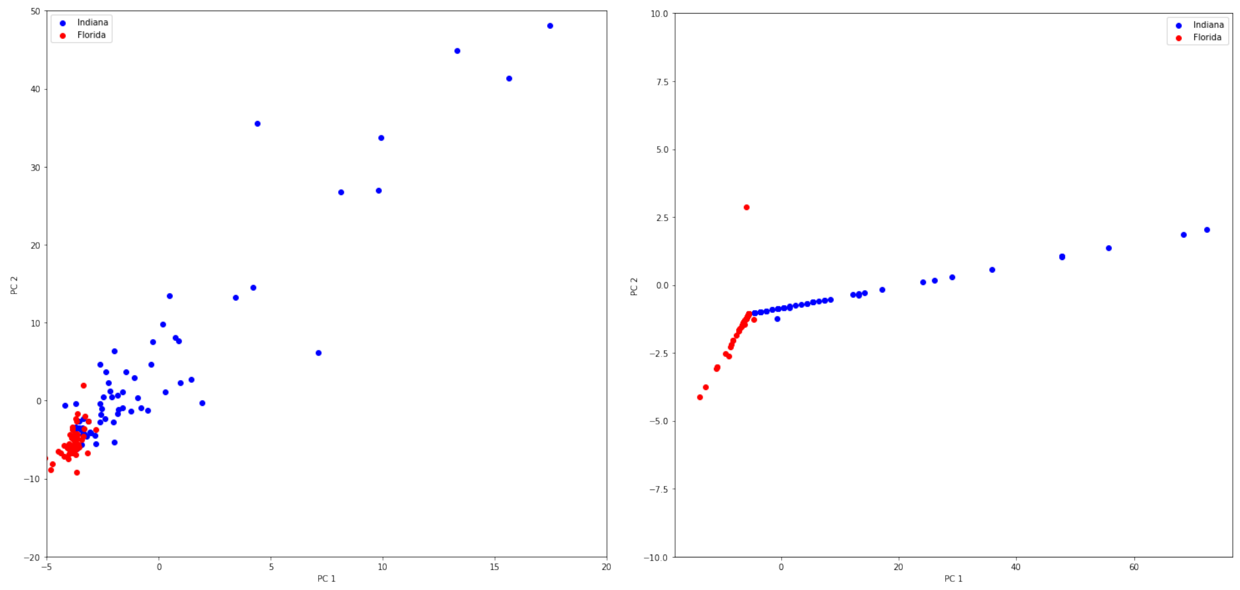

Figure 10: The first 2 principal components of (left) the full reduced matrix from the word count data and (right) the same matrix but where we have only kept 5 columns with the largest normalized leverage scores (computed with $k = 10$). We see that the Florida and Indiana data appears to be approximately separable in both cases.

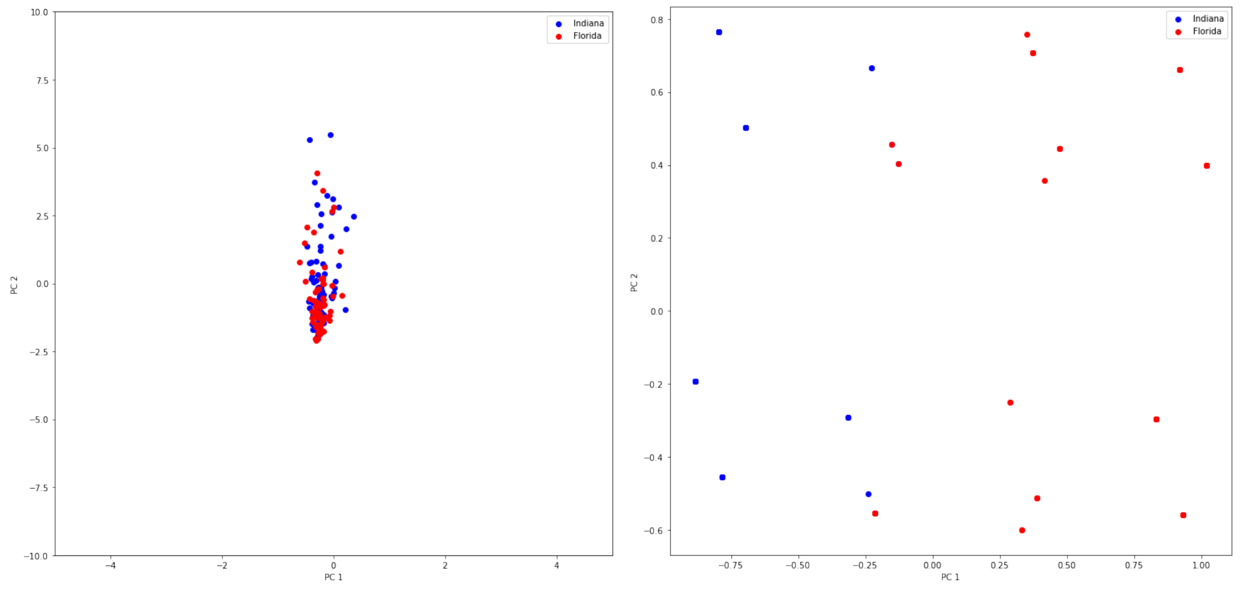

Figure 11: The first 2 principal components of (left) the full reduced matrix from the binary data and (right) the same matrix but where we have only kept 5 columns with the largest normalized leverage scores (computed with $k = 10$). We see that while the data does not appear approximately separable in the full matrix case, the maximal leverage score data does appear to be approximately separable.

In the case of the word count data, we found that the words with the maximal leverage score were, in order from largest to smallest: `clearpixel`, `florida`, `littlepaw`, `evansville`, and `weather`. Florida and Evansville, IN are the places where these documents were gathered from, so it makes sense that these would give the highest leverage for separating the data sets. We found that Big Paw Little Paw Spa is a mobile pet grooming service based near Evansville, IN, again explaining why it would have the most leverage.

For the binary word data, we found that the words with the maximal leverage score were, in order from largest to smallest: `evansville`, `welcome`, `contact`, `indiana`, and `florida`. Once again, geographic indicators are considered the most important, as we would expect.

Code

Our code can be found at `https://github.com/ShashankSule/AMSC808N_Project2` in the folders corresponding to our names.

References

[1] Gabriela Alexe et al. “Consensus algorithms for the generation of all maximal bicliques”. In: *Discrete Applied Mathematics* 145.1 (2004), pp. 11–21.