Dimensional Reduction and Clustering

COSC528 - Fall 2018
Project 2
10/23/18

Rekesh Ali

1 Introduction

Given a list of universities with several attributes including amount of students, various expenses, and degrees offered to name a few, the goal of this project is to be able to 1) reduce the dataset into principal components for more tractable analysis, and 2) cluster the schools into like groups using various subsets of the data. The software needs to be capable of 1) reducing data into components and rebuilding estimations of that data so that it can be 2) clustered and compared at various levels of estimation. Dimensions will be reduced via Principal Component Analysis (PCA), and clustering will be performed using the kmeans and Expectation-Maximization (EM) algorithms. Also included in the EM section is discussion about the effects of feature selection and data standardization on matrix singularity.

2 Data Preparation

The data acquired describes 55 instances split into 65 classes including discrete, unique, and continuous data types. Since the techniques we are using require numerical data, we will discard anything that is a string or a list of discrete values such as the name of the school or whether or not it has a medical school. In addition, features with any missing numbers will be discarded as well because the dataset is not broad enough to merit imputing with averages (and for simplicity). Also, there seems to be enough attributes for a robust analysis that it seems unnecessary to look up these values for imputation.

Upon manual inspection of the data, it appears there are duplicate attributes, either labeled the same name or sometimes even under a different name entirely. This is not good because having duplicates in the dataset will cause the matrix to be singular, which will not be good when doing an inversion or finding the determinant of the covariance. These duplicates are not always the exact same either, so they will be scoured out manually and their indices added to a list of attributes to ignore. Another note from manually inspecting the data is that some of the attributes happen to be linear combinations of the other attributes. This causes the same types of problems, and so they will also have to be added to the list of bad indices. As such, many 'total' attributes such as 'total students' are ignored in favor of features like 'undergraduates' and 'graduates' which will go to diversify our dataset.

To account for multicollinearity that was not seeded out manually, and to reduce the effect of disparate values ranging from hundreds of millions to single digits, the samples for each continuous attribute will, in addition to the raw data analysis, be standardized according to z-normalization. Specifically, a sample's z-score is computed according to **Eq.1**, where μ_j is the sample mean for the attribute and σ_j is the standard deviation.

$$z_{tj} = \frac{x_{tj} - \mu_j}{\sigma_j},\ t = 1 : N_{samples},\ j = 1 : N_{attributes} \tag{1}$$

3 Implementation

3.1 Model

3.1.1 Principal Component Analysis

The PCA is done by performing a singular value decomposition (SVD) on the numerical data matrix. This program uses python's built in svd function in the linear algebra package of numpy. The decomposition is illustrated below, where $\mathbf{U} = [\mathbf{u}_1...\mathbf{u}_t]$ are the left singular column vectors denoting the principal components, Σ is a diagonal matrix containing the singular values which represent the square of the variance, σ_j^2, in each principal direction, and $\mathbf{V}^* = [\mathbf{v}_1...\mathbf{v}_j]^T$ are the right singular row vectors. The product $\sigma_1\mathbf{u}_1$ represents the first principal component weighted by its variance, which is the direction within the dataset that captures the most variance.

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^* \tag{2}$$

The sum of the root of the singular values represents the total variance within the dataset. Luckily, the singular values are ranked by magnitude, so the principal components are in order by variance. Also, it usually does not take too many components to recover a high percentage of the variance, so data can be represented as much lower cost given that there are a low amount of k singular values that capture a high percentage of the variance. Thus the data matrix can thus estimated using the first k principal components, by plugging in the values shown below into **Eq.2** above.

$$\begin{aligned}
\mathbf{X}_k &\approx \mathbf{X} \\
\mathbf{U}_k &= [\mathbf{u}_1...\mathbf{u}_k] \\
\Sigma_k &= diag[\sigma_1^2...\sigma_k^2] \\
\mathbf{V}_k^* &= [\mathbf{v}_1...\mathbf{v}_k]^T
\end{aligned}$$

3.1.2 kmeans

The classification of t instances within $\mathbf{X} = [\mathbf{x}_1...\mathbf{x}_t]^T$ into i classes is fairly simple. To prepare, a $t \times i$ zero matrix denoted by $\mathbf{C}$ is constructed, where a value of 1 for instance t in column i denotes instance t belongs to class i. To keep runs consistent, the classes will be assigned means, $\mathbf{M} = [\mathbf{m}_1...\mathbf{m}_i]^T$, at the first $t = i$ instances corresponding to $\mathbf{X}$. Next the euclidian distance from each instance to the means of each class, $\mathbf{d}_t = [d_{t1}...d_{ti}]$, is calculated and compared, where the smallest distance to a class results in being assigned to that class. After that, the means for each class will be updated now that the classes have been expanded to fit more instances. The algorithm repeats until the instances no longer change classes.

kmeans algorithm:

initialize first t = i instances as cluster focii

repeat:

$$\mathbf{m}_i = \frac{\sum_t C_{ti}\mathbf{x}_t}{\sum_t C_{ti}}$$

$$d_{ti} = \sum_j [\mathbf{x}_{tj} - \mathbf{m}_{ij}]^2$$

$$if\ d_{ti} = min(\mathbf{d}_t),\ C_{ti} = 1$$

until $\mathbf{C}$ converges

As figures of merit, the minimal intercluster distance, maximum intracluster distance, and Dunn index are calculated as well. The intercluster distance is the shortest distance between each class and will be a square matrix of size i. Max intracluster is a measure of how large a cluster is, so it will be an array pertaining to each cluster. The Dunn index is the (min inter)/(max intra). All distances are computed euclidiean.

3.1.3 Expectation-Maximization

Expectation maximization is a little bit more sophisticated, but parallels the kmeans quite well. For starters, instead of classes being boolean, the class of an instance is represented by a likelihood. This range between 0 and 1 is a 'soft' label as opposed to the kmeans boolean 'hard' label. Now, each instance is seen by every cluster, but its effect on that cluster is weighted by its likelihood and unequal priors. The actual assignment will go to the cluster with the highest likelihood of containing an instance. In summary, this method maximizes the probability that a certain instance lies within a specific cluster. The EM algorithm is below, where π_i is the prior likelihood of a class and $\mathbf{S}_i$ is the covariance.

initialize clusters with kmeans

repeat:

$$\pi_i = \frac{\sum_t C_{ti}}{N}$$

$$\mathbf{m}_i = \frac{\sum_t C_{ti}\mathbf{x}_t}{\sum_t C_{ti}}$$

$$\mathbf{S}_i = \frac{\sum_t C_{ti}(\mathbf{x}_t - \mathbf{m}_i)^T(\mathbf{x}_t - \mathbf{m}_i)}{\sum_t C_{ti}}$$

$$C_{ti} = \frac{\pi_i |\mathbf{S}_i|^{-1/2} exp[-1/2(\mathbf{x}_t - \mathbf{m}_i)\mathbf{S}_i^{-1}(\mathbf{x}_t - \mathbf{m}_i)^T]}{\sum_j \pi_j |\mathbf{S}_j|^{-1/2} exp[-1/2(\mathbf{x}_t - \mathbf{m}_j)\mathbf{S}_j^{-1}(\mathbf{x}_t - \mathbf{m}_j)^T]}$$

until $\mathbf{C}$ converges

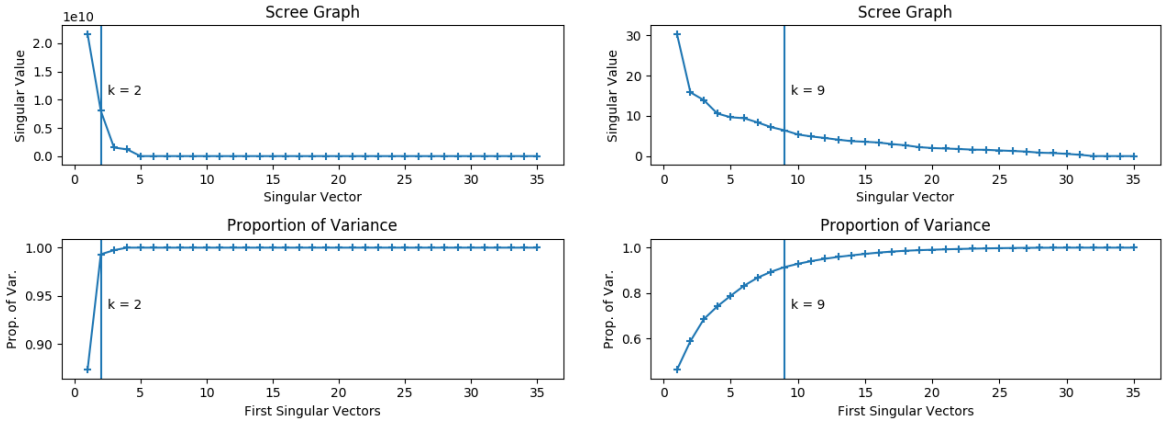
3.2 Program

After constructing the dataset, the program first decomposes the data and finds the k singular vectors that capture 90% of the variance. Plots of the scree and first two components are then generated. Then, for $c = [2...9]$, the full dataset, k reconstructed dataset, and first 2 principal component reconstructed datasets are clustered into c clusters, plotted, and Dunn index reported. Next, EM is performed on the first 2 principal component reconstructed dataset until convergence and it is plotted along with the bivariate gaussian distributions showing the 90% and 95% accuracy bands. This outline is in main.py, data exploration is in explore.py, normalization is in prepare.py, implement.py has SVD/kmeans/EM functions, Dunn indices are reported from output.py, routines such as diagonalizing an array are stored in tools.py, and plots are generated in plot.py. The current build can be found on github at rekeshali/redim-cluster.

4 Results

4.1 Principal Component Analysis

Figure 1 shows the scree graph and proportion of variance covered by the singular values for the normalized and non-normalized dataset. The proportion of variance threshold is set to 90%. As can be seen, it takes many more values for the normalized set, which means the non-normalized set is very heavily influenced by one of its components. This could be due to the disparate value range that occurs in the dataset.

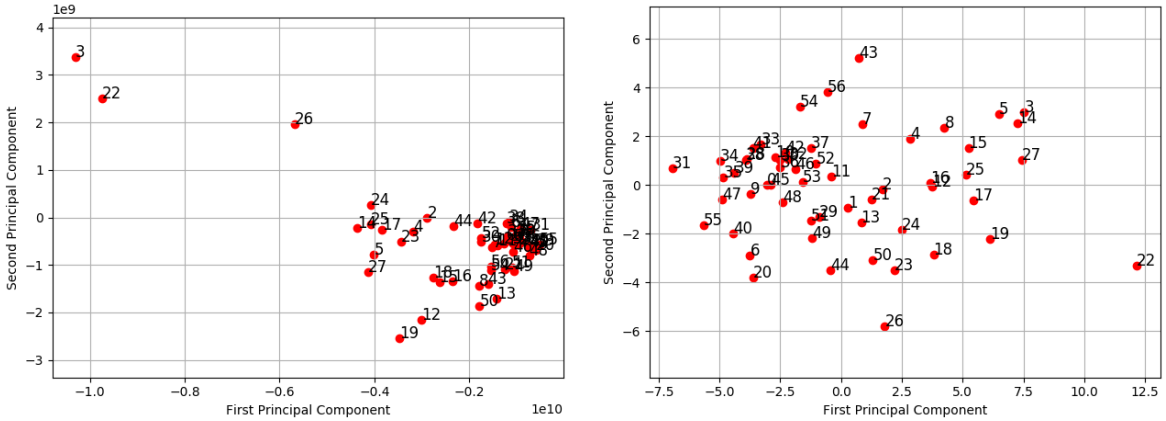


(a) Non-normalized.

(b) Z-normalized.

Figure 1: Scree graph and proportion of variance for non and normalized dataset

Figure 2 shows the same datasets side by side but now illustrates what they look like projected onto their first two principal components. It appears that the non-normalized instances are very heavily concentrated in a local area, which may make them hard to cluster.



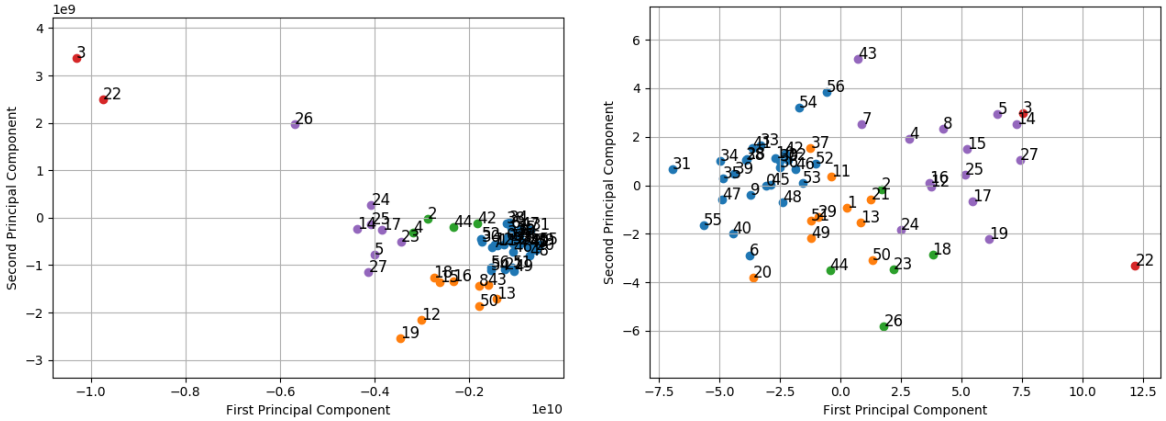
(a) Non-normalized. (b) Z-normalized.

Figure 2: First two principal components, annotated by university numbers

This means there's hardly any variance in the other components, which is telling of the scree graph as well. The normalized data has a moderate spread to it, so it's a bit clearer to observe what instances may be clustered together.

4.2 kmeans

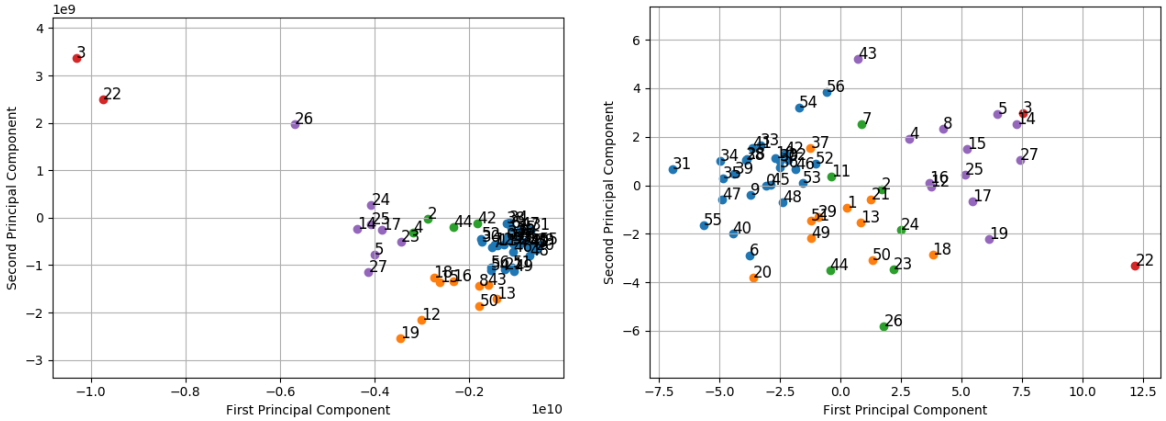
Figure 3 shows the end result of a kmeans with 5 clusters on the full datasets. Even though the datasets principal components are very different, there are some striking similarities between the clusters of the two. Observing the clusters, there are many universities



(a) Non-normalized. (b) Z-normalized.

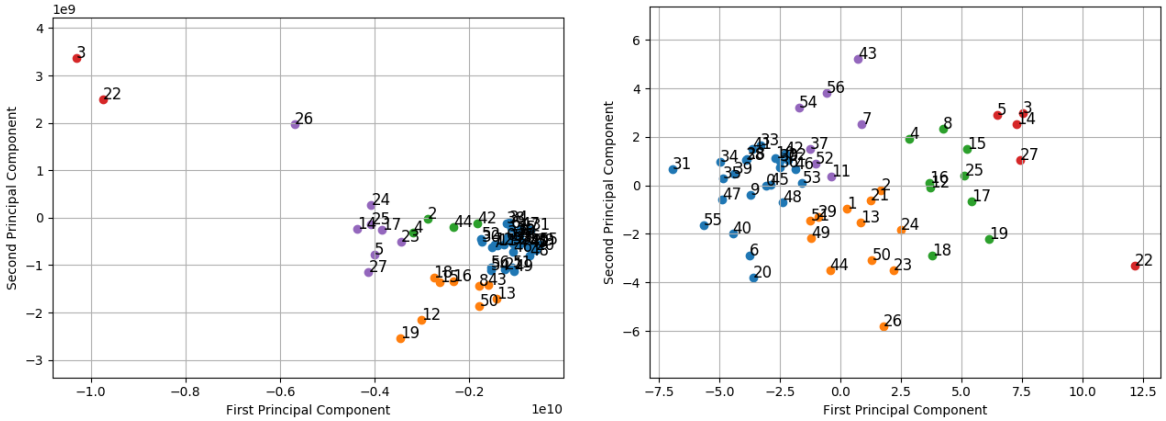
Figure 3: kmeans of the full dataset with 5 clusters.

that end up in clustered together regardless of the normalization. The higher dimensional spaces are perhaps allocated in similar fashions. **Figure 4** has the same amount of clusters, but this time it uses a data matrix reconstructed from the first k PCs as indicated by the scree graphs above. The non-normalized clusters do not change from the previous figure to this one, but there are some slight changes in the normalized graph, such as instances 18 and 24. This makes sense, because according to the scree graphs, the normalized dataset reaches nearly 100% of its variance in just 2 PCs, so the reconstructed matrix is going to be very similar to the full data. The normalized k is sitting at about 90%, so the accuracy of the reconstruction is less, and the results speak to that difference.



(a) Non-normalized. (b) Z-normalized.

Figure 4: kmeans of reconstructed data from the first k PCs with 5 clusters.



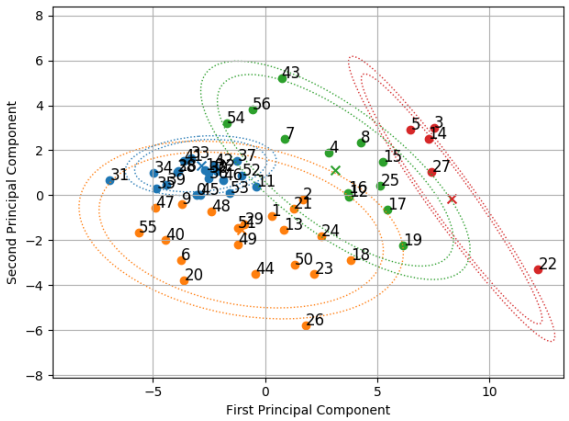
(a) Non-normalized. (b) Z-normalized.

Figure 5: kmeans of reconstructed data from the first 2 PCs with 5 clusters.

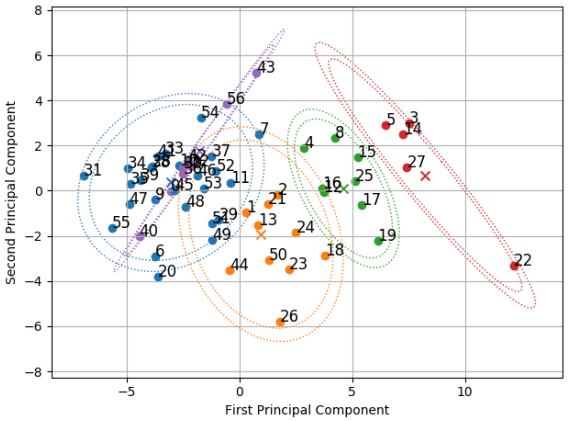
Finally, **Figure 5** shows the kmeans on a reconstruction of the original data using its first 2 PCs. Obviously, the non-normalized clusters do not change, because the previous k turned

out to be 2 as well, so it seems all three clusterings have been exactly the same. The more interesting change occurs within the normalized dataset. There are definitely some changes, and these more drastic than the previous. The clusters now clearly represent a 2D euclidean space, where centers can be visualized by observation and the clusters do not overlap. This just goes to illustrate the effect of the higher dimensionality on the normalized dataset. The previously observed clusters overlapped in the 2D space because they were simply projections from a higher order. This result thus confirms the correctness of the PCA, and its ability to reduce data into simpler bases.

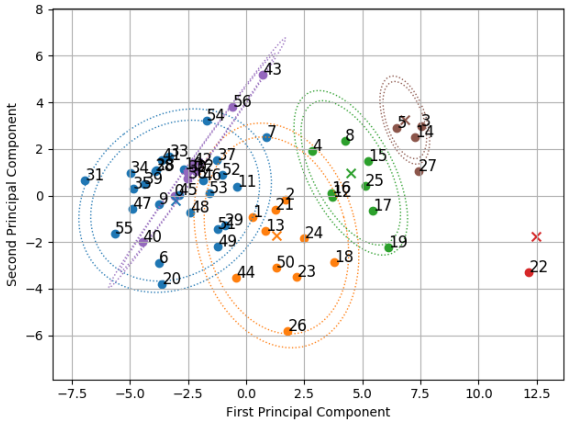
4.3 Expectation-Maximization



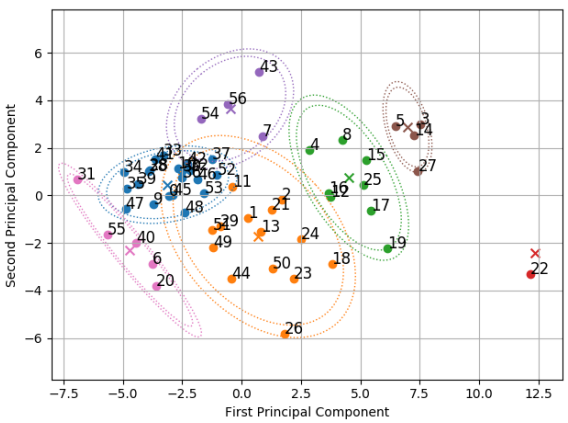
(a) 4 clusters.



(b) 5 clusters.



(c) 6 clusters.



(d) 7 clusters.

Figure 6: EM clustering of reconstructed data from the first 2 PCs of the z-norm data with 90% and 95% confidence bands.

Illustrated in **Figure 6** is the EM clustering result for the normalized data, and sort of an evolution of the clusters as c increases. First off, there is no EM result for the non-normalized data because the covariance matrix, $\mathbf{S}$, always turned out singular which caused the determinant to be 0 thus yielding a divide by 0 error. The normalized data had some kinks as well, but I reduced the singularity of the covariance matrix by adding a very small amount of noise (on the order of 1e-9) to the diagonal. These issues made feature selection really important, because even with the noise, attributes that are linear combinations of others can throw a wrench in the algorithm.

In the 4 cluster result, there is a class nested completely within another cluster, these points probably have a moderate likelihood for either cluster, but edge slightly for one over the other. The effects of the other instances on the bivariate distributions are clearly observable in the 4 cluster graph. If done in the style of kmeans, the red class would have a variance angled more counterclockwise around its center, with its center also looking like it should move up. However, the center is pulled in towards the other points, and angled away from point 27. Again, this is because the EM accounts for all points belonging to a single cluster with varying weights.

The jump from 4 clusters to 5 clusters is very interesting. First, even though some points in the blue class are closer to the orange one, their color is blue because they have a higher likelihood of fitting the blue distribution. The biggest change here seems to be that the orange and green classes spit their load between the blue and purple ones. The purple distribution is quite peculiar, but the points fall very much in line with its distribution. Also notice the red class mean and rotation change due to the decreased likelihood that other points belong to its class, which comes from the fact that there are now more classes to share the points.

From 5 to 6, the red class loses all but one point, and for some reason the gaussian distribution does not show itself, even though it should have one as a result of the pull from other particles, but perhaps it is very small and 22 is an outlier. The reason that happened is because the extra class split red, which had a very high variance, into a smaller class with higher probabilities all around.

Just like what happened to the red class, the purple class decreased its variance from 6 to 7 clusters, thus increasing the probability that it owns its points, by splitting its load. It seems this is a pattern between classes with very high variance in one direction compare to the other. It appears that as more clusters are introduced, the gaussians round themselves out to be circular approaching a limit where each point is in a class all to its own. However, this is not always the case as seen in the creation of the purple and pink classes. The classes, then clearly align themselves with the maximum probability that a point belongs to them.

5 Conclusion

This two part study offered a lot of insight into the spacial characteristics of high dimensional data. PCA can reduce a dataset to varying levels of approximation while saving cost. For example, the 2 PCs result in storing a collection of doubles sized $2 \times t + 2 + 2 \times j$ rather than the much larger $t \times j$, while still getting a very similar clustering job done. The kmeans clustering method is very easy to implement, has fast convergence, and does not require and special handling of the data. The result is not as believable as the EM is however, because it treats all points as being equally likely and minimizes the L2 norm of a point to its mean, thus creating a circular biased boundary. The EM method allows for a variation in the shape of the cluster according to the likelihood that a point belongs to a class. The clusters also take into account the expectation of the entire dataspace with varying priors and maximizes the probability of a point belonging to a cluster, thus producing a holistic and more confident result. The drawback is that it is slightly more difficult to implement, and is dependent on determinants and matrix inversions, so care must be taken into how to prepare the data for effective results.

It turns out that the Univeristy of Tennessee is an instance that stays pretty much in the same cluster the whole time. This is expected because the PCs project it in the denser areas. According to the results, it consistently shares a group with Auburn, Arkansas, Mississippi, Alabama, and Florida State to name a few. No surprise it's surrounded by schools in the same area of the country, with common known similarities ranging from sports and social societies to academia.