Ben Wright Final Report for CS 765

The problem that was examined in this project was the following: What is the behavior of clusters under dimensionality reduction? Essentially, one of the main tasks we can perform on a 2D scatterplot is identifying clusters. But when this data is a transformed version of high-dimensional data, what is the meaning of these clusters? The challenge is to understand whether they represent clusters in the higher dimensional space or not. Another thing that can happen is that the clusters are broken up into distinct regions in lower dimensions, which is less common because generally dimensionality reduction algorithms are based on continuous transformations, but it is something that occurred in this project. So, the idea is to create a tool that helps a user visualize what a specific dimensionality reduction algorithm is doing to their dataset. A secondary focus of the project was to test whether certain transformations of the data performed before dimensionality reduction could help ease the task of distinguishing clusters in lower dimensions – or else make these clusters more distinct. The specific algorithm that was implemented was based on the paper “Visual Cluster Separation Using High-Dimensional Sharpened Dimensionality Reduction” by Youngjoo Kim, Alexandru C. Telea, Scott C. Trager, Jos B. T. M. Roerdink.

There are several ideas behind this project. One idea behind the solution is to use color to represent the clusters in high dimensions on the lower-dimensional scatterplot to visualize the clusters before the dimensionality reduction algorithm makes the data possible to render on the screen. Lots of information is still lost here – for example – the exact distances between points in the original high-dimensional space, but it is basically a way to visualize the nearest neighbors to a point in the high-dimensional space. Another idea is to implement multiple dimensionality reduction algorithms for the same data set to give the user different perspectives on how many clusters to use in the higher dimensional space. Also, to this end, the idea is to let the user specify and vary the number of clusters to use in the high dimensional space, so that when they input their own data, they can deduce visually by scanning the amount of mixing between clusters the appropriate number to use. For example, look at the unsharpened text data with 3 clusters in the final visualization of the project. The data is almost separable by these clusters. On the other hand, examine the IPUMS data with any number of clusters in high dimensions. Although there often appears to be two clusters after dimensionality reduction, there is always significant mixing between the clusters.

The prototype solution that was implemented essentially does this: inputs a data set (several come preloaded, but there is also the option to load any .csv file with all numerical data) and shows the user the behavior of clusters under several popular dimensionality reduction algorithms that were discussed in class, namely, PCA, T-SNE, and UMAP. The clusters are determined by k-means clustering in the high dimensional space. Also, the algorithm discussed in the paper 1. was implemented in Python for this project. (The project creator wrote the program following the C++ implementation given by the authors of the original paper.) Again, this was for two reasons: First, to investigate whether such a data transformation is a useful way of identifying clusters in the lower dimensions on a benchmark dataset. Second, to give the user another tool to identify clusters in their own dataset. The program allows the user to specify a dataset, then select a dimensionality reduction algorithm to perform on it, displaying the results on a 2D scatterplot with points colored by the k-means clusters in high dimensions. An interactive slider allows the user to select the desired number of clusters.

One interesting thing about the visualization is that the implementation of the data sharpening algorithm was programmed basically from scratch in Python, following the original example that was in C++, but there were unique challenges in translating the algorithm such as nuances of memory management and making deep copies of variables and so on. The implementation was all done in Python developed in a Jupyter notebook to make development with the data easier. Unfortunately, because of the Jupyter and matplotlib backends, this made interaction very difficult to implement. The author encourages the reader to investigate how interaction was achieved in the final visualization. Ideally, there would be a prettier animation between changes in the data, but this is the best that could be done for the prototype.

Here are some interesting screenshots of the solution in action. These were already mentioned in a prior paragraph but let us restate them here.

Graphical user interface, text, application

Description automatically generated

This first image is of the sharpened IPUMS data reduced with the T-SNE algorithm, pictured with 2 clusters in the high-dimensional space. Although there appear to be 3 clusters in 2D, the amount of mixing between them may lead a user to suspect that the data is really too mixed in the high-dimensions to group together in a meaningful way. Graphical user interface, application

Description automatically generated

Next is the unsharpened text data with PCA reduction with 3 clusters in the high-dimensional space. Among all the options available by the program, including number of clusters, this minimizing mixing between the clusters and looks like almost separable data, which might lead one to conclude that there are 3 clusters in this dataset. Lastly, here is sharpened MNIST data with 3 clusters and PCA reduction versus sharpened MNIST data with UMAP reduction in 10 clusters. The point of these examples is to show the utility in having multiple algorithms implemented for the same visualization, lest we conclude falsely that there are only 3 groups of data represented in this dataset. Chart

Description automatically generated

Map

Description automatically generated

Taken together, these examples show the utility of this tool for exploring the clustering patterns of a new dataset.

The following paragraphs seek to answer prompts numbers seven and eight. This project sought to explore two major problems, as stated previously. One, how can I explore the clusters of a high-dimensional dataset in low dimensions, and two does data sharpening help with the first problem? On the first problem, several of the key ideas of the project seem like partial successes at least, because take the last example shown above. One idea was to incorporate multiple dimensionality reduction algorithms into the same visualization. This was a success because it prevented us from concluding that the MNIST dataset, which records handwritten digits, represents only 3 groups of information, when it in fact represents 10. Another idea was to allow the user to vary the number of clusters, which was a success because in the case of the text data with PCA reduction, it allows the user to see 3 clusters is an appropriate number. Frankly, however, the data sharpening did not seem to add much value to the visualization, at least in the examples of the 4 built-in datasets. It did not appear to separate the clusters in higher dimensions any better than unsharpened versions of the reduction algorithms, and it did not help identify any new clusters in these examples. On the other hand, this is useful in and of itself to know, that this algorithm is not always the right tool – is not always useful for every dataset. Also, it is apparently the right tool sometimes, so it is still useful to have implemented.

One clear limitation of the prototype is that it only handles numerical data. In theory, it could categorically encode other data, but that was not implemented. Also, there is no way to recover the information lost in the dimensionality reduction from the algorithm about the distances between points in the high dimensional space. An interesting next step would be to copy an idea seen in class today where hovering over a point highlights the k nearest neighbors in the high dimensional space and labels the distances. Lastly, no one visualization of dimensionality reduction can explain the correct number of clusters to use in a low or high dimensional space. One thing that was learned during this project is that in dimensionality reduction, some information is always lost.

References

1. *Visual Cluster Separation Using High-Dimensional Sharpened Dimensionality Reduction*. Youngjoo Kim and Alexandru C. Telea and Scott C. Trager and Jos B. T. M. Roerdink. 2021. https://arxiv.org/abs/2110.00317