EE364B Progress Report

Jonathan Johannemann

May 2019

1 Problem Statement

The problem that we attempt to solve is the one where we have a pair-wise rank ordering over a high dimensional input space X^n and we wish to project points into a 2 or 3 dimensional visualization while maintaining the rank ordering as much as possible. Our approaches to solving this problem come in the form of focusing on maintaining the local structure of the points $x_i \in X^{n \times d}$ by either up-weighting those distances in our objective or by directly tackling the rank ordering via a series of L2 constraints which are predetermined using the original pairwise distance matrix of $X^{n \times d}$.

2 Background

With the growing popularity of nonlinear dimensionality reduction such as t-SNE, UMAP, Auto-encoders, and more, it can become difficult to actually assess the integrity and stability of low dimensional representations of high dimensional data. Challenges in this space include a lack of "ground truth", data sets with intrinsic dimension larger than 2 or 3 dimensions, and lack of true local and global structure. In a recent project, I discovered that optimizing for maintaining the pair-wise rank ordering yields considerably good results compared to existing quality metrics for simple simulated datasets. Due to this recent find, we seek to optimize for this scenario specifically in order to develop a new and unique nonlinear dimensionality reduction algorithm which does a good job of projecting a high dimensional data set to a 2 or 3 dimensional space while maintaining the integrity of the rank relationship between points.

3 Approaches

We explore a few approaches in combination with some of the step size or gradient modifications learned in class in order to search for the top performers based on the evaluation metric we will discuss in the Experiments section. The following approaches are as follows:

3.1 MDS and Sammon Mapping

The two most common visualization methods prior to more recent methods such as t-SNE are Multi-dimensional scaling (MDS) and Sammon Mapping. The Multi-dimensional scaling objective is:

$$Loss = \sum_{i=1}^{n} \sum_{j>i} ||D_{ij} - d_{ij}||^2 \tag{1}$$

where D_{ij} is the Euclidean distance between points x_i and x_j in the original d dimensional space and d_{ij} is the Euclidean distance between points y_i and y_j in the original p dimensional space. The Sammon Mapping merely scales each value by D_{ij} to reduce the impact of larger distances. While MDS is generally not a popular tool for dimensionality reduction in more complex datasets, Sammon mapping is still commonly used.

3.2 L_2 Map

In many cases, dimensionality reduction is conducted in such a way that each data point in $X^{n \times d}$ is being iterated on via some form of gradient descent. While this might be temporarily convenient or simpler to formulate the objective, it creates complications further down the line when new data is introduced. The researcher must run the method all over again which can be computationally intensive or may have to come up with a pseudo-method to project new points into the low dimensional space based on where points ended up during the initial run of the dimensionality reduction algorithm. Instead, we formulate a loss very similar to MDS except we iterate over a map S which projects points from $X^{n \times d} \rightarrow Y^{n \times p}$. This way a researcher can use this map for new data points down the line.

$$Loss = \sum_{i=1}^{n} \sum_{j>i} \Big|\Big| \;\; ||Sx_i - Sy_i||_2 - ||x_i - x_j||_2 \;\; \Big|\Big|^2 \tag{2}$$

3.3 L_2 Map with weighting function f_{w_i}

Inspired by the method of inverse square weighting of residuals for heteroskedastic errors in linear models, we take three different approaches to re-weighting the residuals to focus our algorithm to focus more on maintaining local structure and ideally ultimately rank order of the visualization.

Reweighting by the residuals of MDS One common approach to handling heteroskedastic errors is to weight observations $x_i \in X$ inversely to their error ϵ upon fitting ordinary least squares. Instead, we run multi-dimensional scaling and then take the difference in the pairwise Euclidean distance matrices and use the square inverse as the weights w_i to up-weight local distances in the L_2 map.

Iteratively re-weighting based on residuals To build on the previous method, we begin the first iteration with MDS but then use following iterations with the output from the weighted L_2 map. In the case of heteroskedastic errors, this is generally advised against because it continuously reduces the influence of points with large errors and overly focuses on points with small errors. Since we are more concerned on maintaining the local structure as much as possible in hopes of better maintaining most of the rank structure, we try this as a second approach to tackling this dimensionality reduction problem.

Inverse Probability Weighting via the Empirical CDF Finally, instead of weighting by the residuals, we took one additional approach to focusing on points that maintained local structure. Another approach we posit is to take the empirical CDF of the distribution of distances in the pairwise distance matrix, with the exception of the 0's generated from each point being 0 distance from itself, and inversely weighting by the probabilities generated by the empirical CDF. The resulting probabilities are small for small distances and, as a result, the inverse probabilities are large while larger probabilities are relatively small.

4 Experiments

To explore the above methods, we take a rather simple approach for now. In the Conclusion, we further discuss expanding this analysis to be more exhaustive.

To evaluate each method, we use two data sets. The first data set is the "Cube" data set which is a series of points in 3 dimensional space and are on vertices of a cube. The objective is to project this 3 dimensional object into 2 dimensional space while generating a representation that resembles the original higher dimensional object.

From here, we run each method for 20 different random seeds to avoid poor performance due to poor initialization and evaluate each lower dimensional set of points $Y^{n\times 2}$ using what is referred to below as "Spectral Error". This yields more weight on the most immediately close points and less on points that are much farther away from a given point. The loss is specified as such: [1]

$$\text{Spectral Error} = 1 - \sum_{i=1}^{n-1} \frac{1}{i\cdot(n-1)} \sum_{j=1}^{(n-1)} \sum_{k\neq j} \tilde{K}^i_{jk} * K^i_{jk} \tag{3}$$

where $\tilde{K}^i$ is the KNN graph in input space X with k parameter i and K^i is the KNN graph in output space Y with k parameter i.

A summary table of the results can be found in Table 4 and visualizations of each method's best run can be seen in the Appendix. Code can be found at: https://github.com/JonathanJohann/EE364BFinalProject.

[1] In my current ongoing research, we have found that this meta criterion is reasonably robust across algorithms and data sets for maintaining local structure of the data.

Method	Stepsize	Descent Method	Spectral Error
MDS	-	-	4.931
Sammon	-	-	5.137
Inverse Square	0.1	-	5.137
Inverse Square	1/k	-	4.775
Inverse Square	$1/\sqrt{k}$	-	5.283
Inverse Square	0.1	heavy ball, $\beta = 0.3$	4.970
Inverse Square	0.1	heavy ball, $\beta = 0.7$	4.629
Inverse Square	0.1	adagrad	4.783
Iterative Inverse Square	0.1	-	4.679
Iterative Inverse Square	1/k	-	5.333
Iterative Inverse Square	$\mathbf{1/\sqrt{k}}$	**-**	**5.500**
Iterative Inverse Square	0.1	heavy ball, $\beta = 0.3$	5.200
Iterative Inverse Square	0.1	heavy ball, $\beta = 0.7$	4.150
Iterative Inverse Square	0.1	adagrad	4.825
L_2 Map	0.1	-	5.162
L_2 Map	1/k	-	5.137
L_2 Map	$1/\sqrt{k}$	-	5.162
L_2 Map	0.1	heavy ball, $\beta = 0.3$	5.162
L_2 Map	0.1	heavy ball, $\beta = 0.7$	5.137
L_2 Map	0.1	adagrad	5.304
Inverse Probability	0.1	-	4.608
Inverse Probability	1/k	-	4.887
Inverse Probability	$1/\sqrt{k}$	-	4.950
Inverse Probability	0.1	heavy ball, $\beta = 0.3$	4.533
Inverse Probability	0.1	heavy ball, $\beta = 0.7$	4.595
Inverse Probability	0.1	adagrad	4.512

5 Conclusion

Based on our preliminary results, it appears that the most promising method based on our selected meta criterion is: Iterative Inverse Square Mapping with stepsize $1/\sqrt{k}$.

While we can see that the iterative inverse square weighting does the best for one select setup, we find that the most robust method across the board is L_2 map which has a tendency to perform as well as Sammon Mapping, if not better, irrespective of the descent method or step size schema chosen. The inverse probability approach generally performed the worst with the single iteration inverse square weighting method doing only slightly better on average.

For next steps, our objective is to compare these methods to the most popular method of t-SNE which is considered to do a good job of maintaining local structure in data visualization. Instead of a simple cube, we will run these methods along with t-SNE on larger datasets such as a sub-sample of the MNIST data set or other popular nonlinear data sets of interest in the nonlinear dimensionality reduction space.

6 References

[**1**] Van Der Maaten, Laurens, Eric Postma, and Jaap Van den Herik. "Dimensionality reduction: a comparative." J Mach Learn Res 10.66-71 (2009): 13.

[**2**] Wattenberg, et al., "How to Use t-SNE Effectively", Distill, 2016. http://doi.org/10.23915/distill.00002

[**3**] Chen, Lisha, and Andreas Buja. "Stress functions for nonlinear dimension reduction, proximity analysis, and graph drawing." Journal of Machine Learning Research 14.Apr (2013): 1145-1173.

[**4**] Johnson, William B., Joram Lindenstrauss, and Gideon Schechtman. "Extensions of Lipschitz maps into Banach spaces." Israel Journal of Mathematics 54.2 (1986): 129-138.

7 Appendix

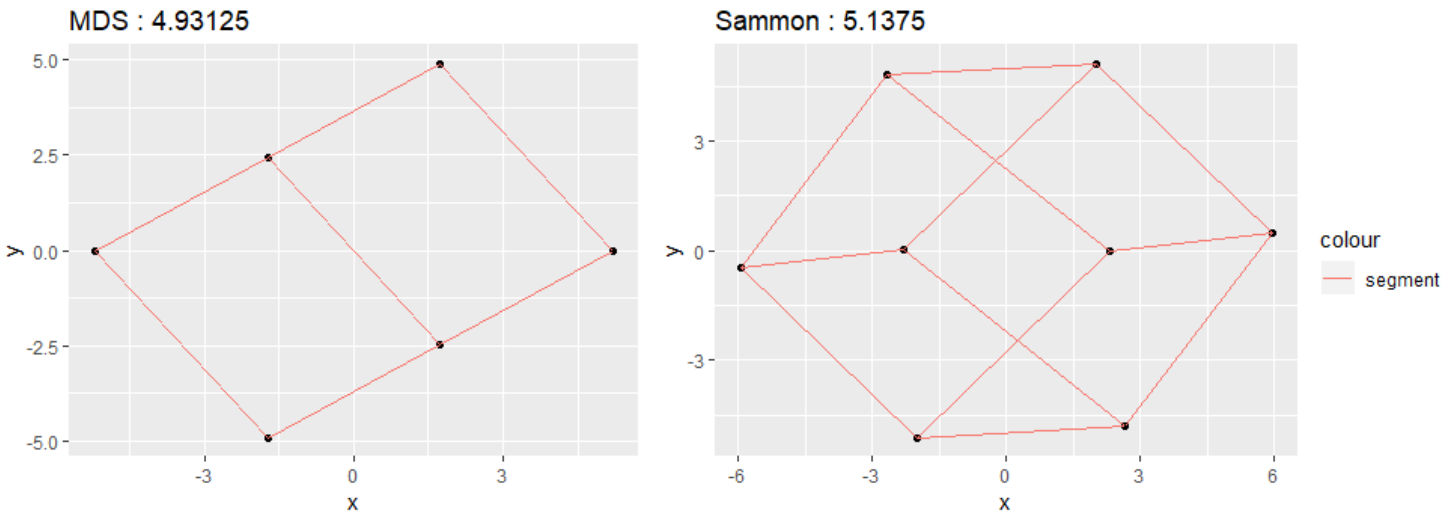

Figure 1: MDS and Sammon Mapping

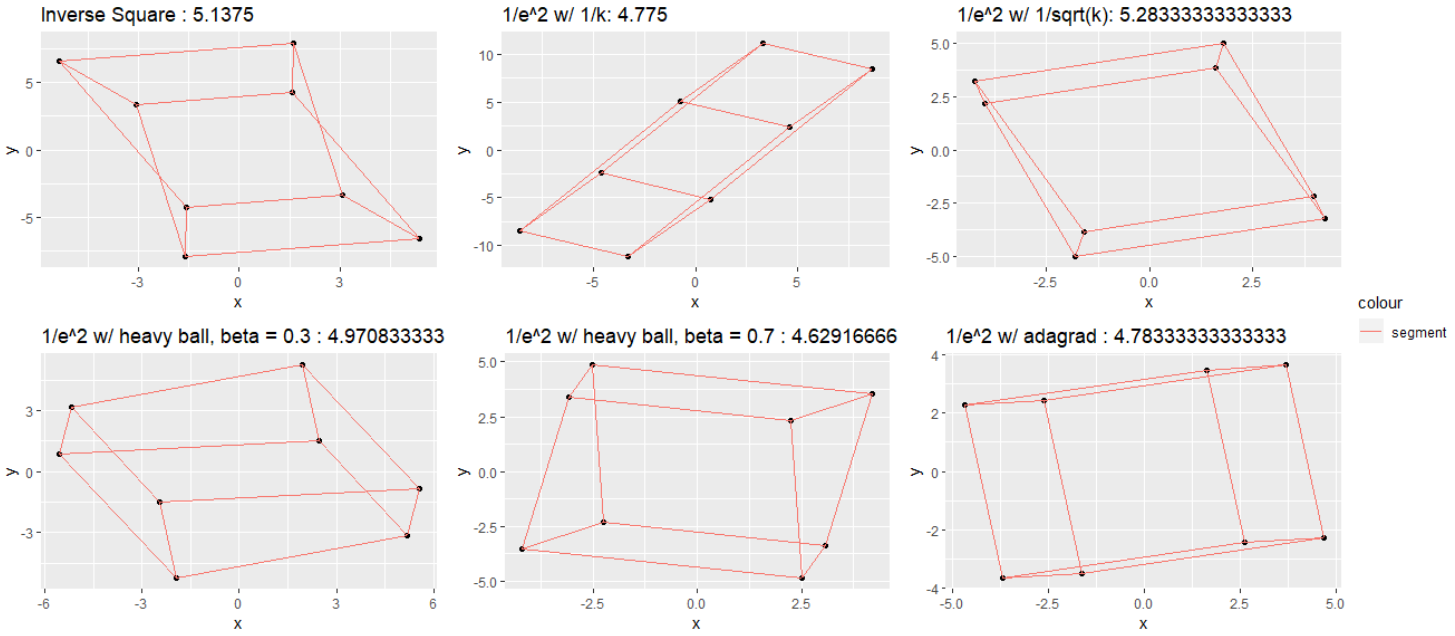

Figure 2: Inverse Square Weighting

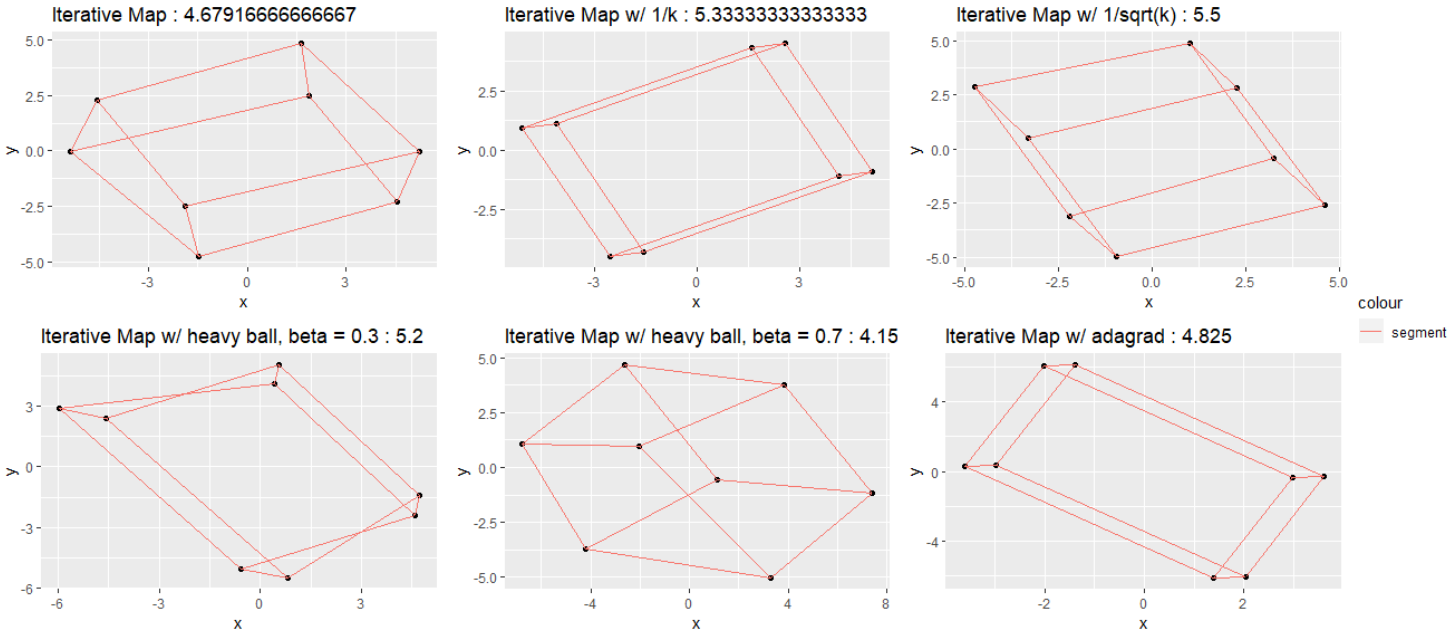

Figure 3: Iterative Inverse Square Weighting

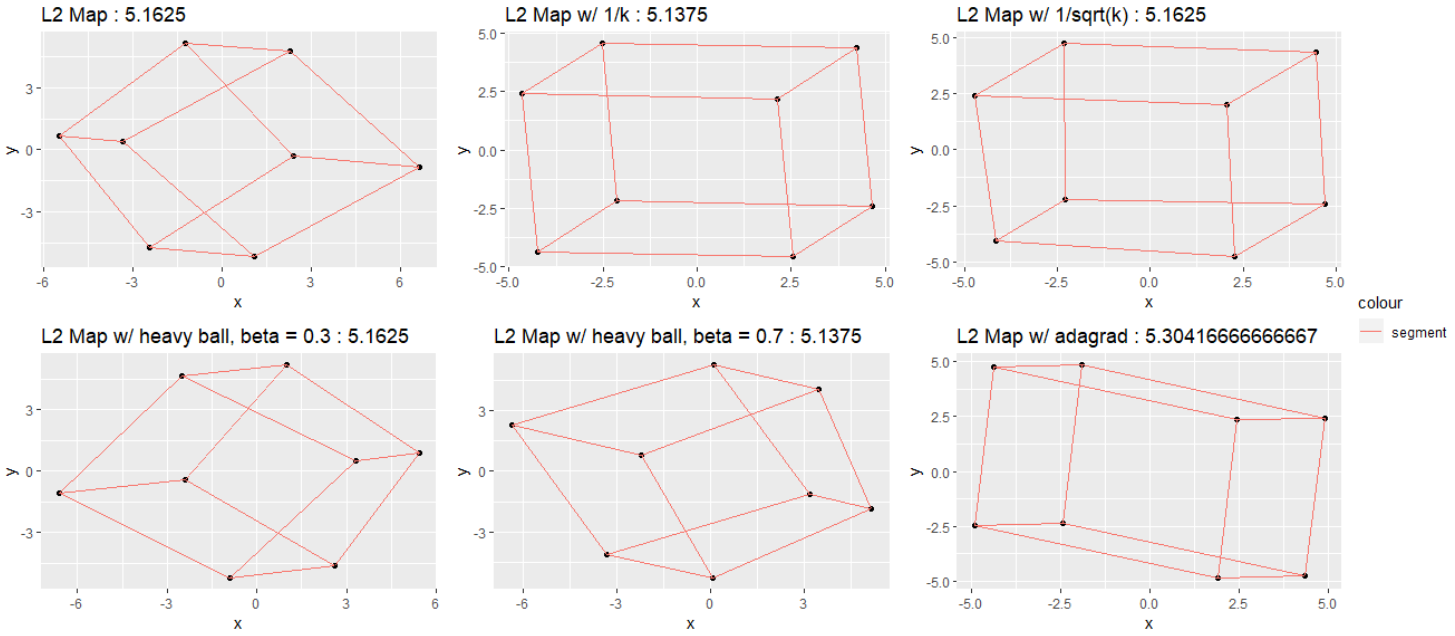

Figure 4: L_2 Map

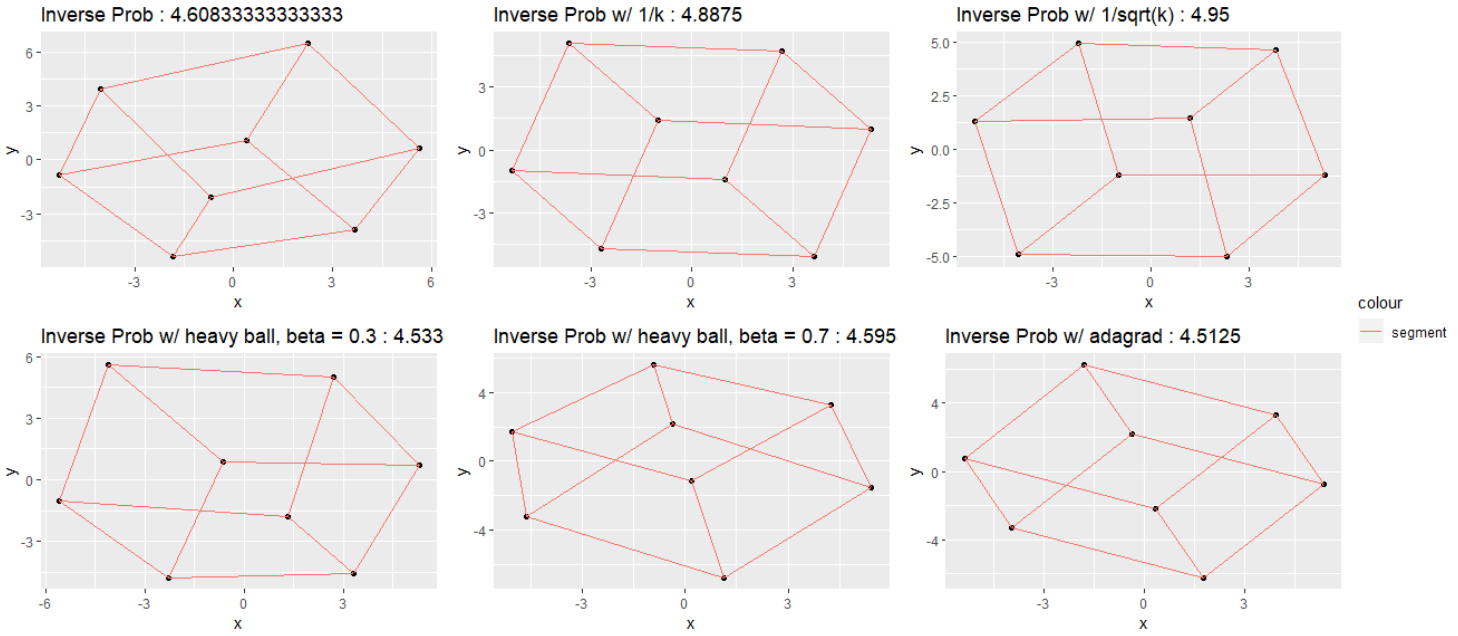

Figure 5: Inverse Probability