Geometric Graphs: Embeddings and Transformations

Author: Robert Bull and Thokozani Tembo

University of Maryland

Date: May 10, 2024

Venue: MATH 420 End-of-semester Project

1. **Introduction**

Our presentation and project revolved around three major steps: estimating an embedding, creating transformation functions, and aligning our estimates with the targets provided. We were given six total datasets for forty nodes: three were ‘Observed data’ files containing partial data on distances between points, and three were ‘Target data' files with exact coordinates for each of the forty vertices.

We implemented several different algorithms to perform each of these steps. Firstly, we used an algorithm to estimate the Gram matrix using partial data. This involved a convex linear program that we solved using CVX, a provided convex programming kit. We then solved the embedding problem to determine the most significant parts of the Gram matrix to get our 3D graph. Secondly, we implemented the solution to the full alignment problem using the parameter interpolation method detailed in the lecture slides. Finally, we calculated the alignment errors between the targets and the transformed estimates to determine the pairs that were closest.

1. **Background/Methods**

The first algorithm we used – the partial data algorithm – was primarily the convex optimization problem described as:

Where G is our resulting Gram matrix. The first condition ensures that it is a positive semi-definite, symmetric matrix. Our second condition ensures that all the rows sum to zero. That is, summing each value in a row would equal 0. The final condition is ensuring that the distance between each given point pair in the Gram matrix minus the distances given to us is less than a certain epsilon value. That is, we create our Gram estimate to follow as closely as possible to the partial data given. Each of these parameters ensures that the estimation we generate is as close as possible to data we already know to be true. We calculated epsilon by starting with an initial value of one and doubling it until we obtained a value that satisfied the semi-definite program above this. Thus, epsilon is very dependent on the initial value we provide at the start of runtime.

The second algorithm we used was the solution to the full alignment problem discussed on the lecture slides. Centered coordinates ( and ) calculated by taking the average of each point’s X Y and Z positions. A new matrix, R, is calculated using the centered coordinates (. Solving the Singular Value Decomposition of R, we get R = and subsequently , , and where are the centers of the estimated set and target set respectively. The equation is the result of the transformation between X and Y. We use a method of interpolation for each of these parameters – hence, parameter space interpolation. Each one is an iteration on previous problems: the Procrustes and classical Procrustes problems were solved using Q – an orthogonal rotation matrix that rotates X to best fit Y – or Q and z – representing both a rotation and a translation vector – respectively. The final algorithm is a solution that combines the results of the classical problem with a scaling factor .

Our final part of the three steps was calculating the error between the target coordinates and the estimated coordinates. For each pair of estimates and targets, we found where was the result of the transformation on each estimate , and the coordinates of the target data.

1. **Results**

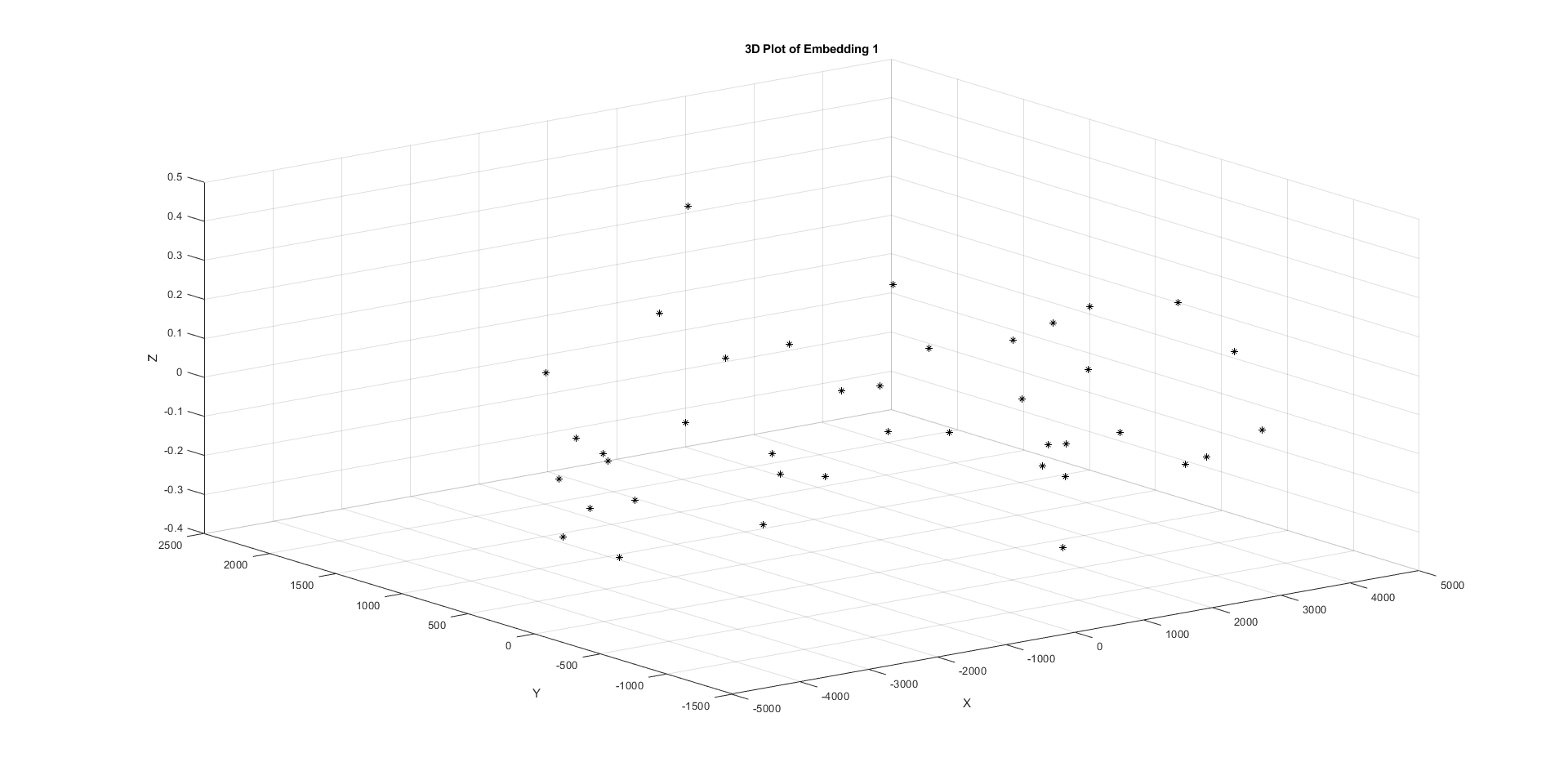
The Gram matrix estimation has several outputs. The first is the primary output of the program.

A close-up of a text

Description automatically generated

Fig 1. The first lines of output in a successful CVX run. The number of constraints (1278) represents the number of values that the CVX constrains the problem to. ‘dim. of sdp var’ is a representation of minimization of the trace of G. This is standard when we’re minimizing a matrix of size 40. The number of socp variable (1238) comes from the parameter. Since the absolute value of a number less than another can be represented as two constraints (greater than the negative and less than the positive), the CVX program splits the constraint into two. Hence, we have double the number of given edges as socp variables. These ‘second-order cone’ variables are how the CVX program defines affine constraints.

The following are the embeddings obtained from the Gram matrix estimation for each observed dataset:

A white background with black dots

Description automatically generated

Fig 2. A 3D and 2D viewing of the embedding for the first observed dataset. This had a final epsilon value of 16.

A grid with lines and dots

Description automatically generated with medium confidence

A white background with black dots

Description automatically generated

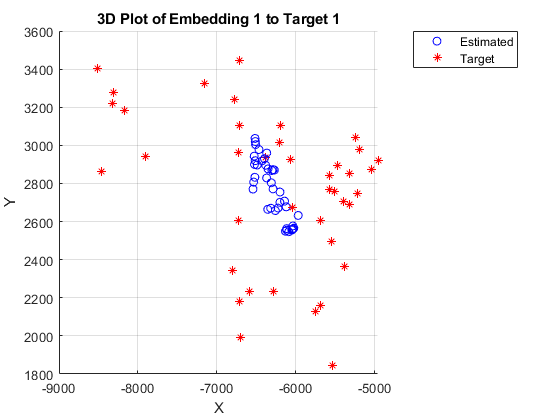
Fig 3. A 3D and 2D viewing of the embedding for the second observed dataset. This had a final epsilon value of 2.

A graph of a graph

Description automatically generated with medium confidenceA white background with black dots

Description automatically generated

Fig 4. A 3D and 2D viewing of the embedding for the third observed dataset. This had a final epsilon value of 2.

 A graph with red dots and numbers

Description automatically generated A graph with red dots

Description automatically generated

Fig 5. Three pictures detailing the results of the transformation corresponding to each target on the first observed dataset.

A graph with red dots

Description automatically generatedA graph with red dots and numbers

Description automatically generatedA graph with red dots and blue dots

Description automatically generated

Fig 6. Three pictures detailing the results of the transformation corresponding to each target on the second observed dataset.

A graph with red dots and blue dots

Description automatically generatedA graph with red dots

Description automatically generatedA graph with red dots and blue dots

Description automatically generated

Fig 7. Three pictures detailing the results of the transformation corresponding to each target on the third observed dataset.

When measuring the error between the transformations and the target sets, we get the following ‘confusion matrix’.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Target 1 | Target 2 | Target 3 |
| Observed Set 1 | 45359570 | 41736965 | 0.177 |
| Observed Set 2 | 0.026 | 40985441 | 55920308 |
| Observed Set 3 | 45647769 | 0.061 | 57307499 |

Fig 8. A table showing the results of for each pair of target and observation data.

Our final entries below are videos detailing the transformations between the lowest error set pairs.



1. **Discussion**

Beginning with the results of the SDP: after running the program, we obtain points that seem particularly like those of the target datasets. While the program seems to have introduced some noise in the z-values, the 2D view of the plots contain many signature features of a single corresponding target. While the results here are somewhat expected, there was an interesting anomaly when calculating initial epsilon values for each of the observed datasets.

To begin, the epsilon values of the graphs above are 16, 2, and 2. These were obtained with an initial epsilon value of 1. If the program could not find a suitable matrix with , it doubled it and moved to 2. However, when starting with alternate epsilon values, I found that the program did not always return a ‘solved’ result with epsilon values greater than 16, 2, and 2. For example, with a starting epsilon value of 0.1, the smallest epsilon value for each observation is determined to be 51.2, 1.6, and 0.2. Notice that 51.2 is far greater than 16. Since the method of implementation for finding the optimal epsilon value is to double the initial condition until a feasible result is found, it follows that there was a check for that had been determined to be ‘infeasible’ by CVX. Even though 16 is a valid value for epsilon, 25.6 did not return a feasible result. While this is certainly an interesting oddity, it did not have a huge practical impact on the overall result of the project. Perhaps the way that CVX calculates possible matrices changed the outcome.

After calculating the different parameters for the transformation functions, the resulting transformed datasets highly correspond with a single target set. This is reflected within the confusion matrix above as well. Approximations that do not correspond to target sets tend to bunch points around a tight center, while correct pairs line up almost exactly. As mentioned previously, it is possible to see this result before any transformation is applied by observing key parts of the initial embeddings. Specifically viewing the upper portion of the first embedding (Observed set 1) shows two ‘prongs’ that correspond exactly with the two ‘prongs’ facing down in the third target dataset. Our findings in the confusion matrix match this as well: the error between the transformed observed set 1 and the target set 3 is a mere 0.177 and is far smaller than the results for the other target sets.