Computing Ordinal Embeddings Quickly and Accurately from Triplet Data

Ari Biswas
aribiswas3@gmail.com

Blake Mason
bmason3@wisc.edu

Abstract

Ordinal embedding represents a set of n items as points in a d-dimensional Euclidean space, and attempts to recover an embedding of the points given ordinal constraints of the form: "item i is closer to item j than item k?". These distance constraints on the embedding, referred to as triplets, define a set of linear constraints on the Gram matrix associated with the set of points. However, optimization over possible Gram matrices requires projections onto the positive semi-definite cone, which is expensive. In our project we analyze the performance of non-convex and projection free methods to efficiently recover triplet embeddings with high accuracy.

1 Introduction

Ordinal embedding, also known as non-metric multi-dimensional scaling, seeks to to learn a d-dimensional embedding of a set of points that maximally agrees with a set of ordinal constraints, such as triplet constraints. The constraints are *ordinal* in that they define an *order* on the distances between the points. For instance consider the following triplet constraint: x_k **is closer to** x_i **than** x_j

As an ordinal constraint:

$$\|x_i - x_k\|_2^2 < \|x_j - x_k\|_2^2 \iff \text{sign}(\|x_i - x_k\|_2^2 - \|x_j - x_k\|_2^2) = -1$$

Given $O(nd\log(n))$ triplets, it can be shown that it is possible to recover the true embedding of points by a maximum likelihood procedure, up to a scaling, translation, and rotation [1].

This is more that an interesting math problem. Ordinal data is commonly gathered when working with humans, because it has been shown that people can more easily and reliably provide ordinal judgements than fine grained numerical information [2]. Another major advantage of ordinal embedding is that it can be used to form classifiers over abstract items [3]. As opposed to traditional, feature based metric learning where we attempt to learn a predictor that weights features based on a set of known labels, classifiers based on ordinal data are agnostic to nonlinearities in the data, correlated features and other problems. Instead, by querying humans about the relationship between items in the set of n items sufficiently many times to estimate the full embedding, it is possible to predict the outcome of other human judgements. For instance, Zappos.com is testing a way of building a recommendation system that uses ordinal information to efficiently search through the thousands of possible shoes in their database to find the one a user would most want. For this to truly be efficient though, it is necessary to be able to quickly estimate the embedding to find the best item.

As early as the 1970s Sheppard and Kruskal proposed an algorithm to recover an embedding, but this method is not robust to noise and is not commonly used in practice. A more well known algorithm, is Generalized Non-metric Multi-Dimensional Scaling (GNMDS) by [4], but the method proposed by [3] is also common. Almost all modern methods recast ordinal embedding as a constraint satisfaction problem over low-rank Euclidean distance matrices. However, it was not until [1] that any sample complexity bounds existed, and a provable algorithm was derived. Little is known about the convergence properties of most of the common methods, such as [3, 4] and little is known about global guarantees. In this paper we focus on the method outlined in [1] that formulates triplet embedding from noisy data as a semidefinite program (SDP). The main contribution of this study is to analyze the performance of proposed methods for accelerating the SDP described in [1] to recover embeddings items more quickly.

2 Notation

In this section we define the notation we will use in the rest of the proposal.

$X \in \mathbb{R}^{n \text{ x } d}$ represents the embedding of n points in d dimensions.

$M = XX^T$ represents the Gram matrix of embedding X.

$q = y_{ijk}$ represents a triplet comparison of the following form : x_k is always closer to x_i than x_j. Index i is called the winner, j the loser and k the head.

$S = \{q | q = y_{ijk}\}$ be set of all triplet constraints.

$s_q = ||x_j - x_k||^2 - ||x_i - x_k||^2$: This is referred to as the "score" of triplet q. If $s_q > 0$ the triplet constraint has been satisfied i.e k was indeed closer to i than j. We ignore cases where i and j are equidistant from k in this project, as they occur with probability 0. If $s < 0$ then the proposed embedding X has violated the triplet constraint.

3 Preliminaries

Before we discus methods to recover point embeddings from triplet constraints, let us begin by discussing two relaxed versions of this problem. First, consider the case where we instead have all $\binom{n}{2}$ distances between n points in the embedding and known the underlying dimension d . Using methods such as [5], it is possible to recover an embedding by back-solving that is accurate up to an orthogonal transformation. Now consider the problem where instead of having access to all $\binom{n}{2}$ distances, we have all $\binom{\binom{n}{2}}{2}$ differences of distances. Until [1], this problem was thought to be unsolvable because it cannot recover uniform scalings of the embedding. Mathematically, this corresponds to the all ones vector being an eigenvector with non zero eigenvalue of the Euclidean distance matrix generated from to the latent embedding. Clearly, the differencing operator cannot recover constants uniformly added or subtracted to every distance, so triplet differences cannot be a full rank set of operators. Because of this, we cannot solve a system of linear equations with the differences of distances to recover the distances themselves and therefore the embedding. One result of [1] was to show that it is still possible to recover the eigenvalue tied to the all ones vector so the distances themselves are still recoverable from their differences with high probability for random embeddings of sufficiently many points.

In the triplets setting, we have far weaker information. Whereas the above case corresponds to knowing the score s_q of each triplet, when working with triplet constraints, we instead know the sign of the score of each triplet. A priori, it is believable that since we can compensate for triplet differences not forming a full rank set of operators, using arguments from the 1-bit matrix completion literature, such as [6], a similar argument will apply in this setting under certain assumptions on the data we gather. In the case of noisy triplets constraints, if we know the monotonic link function corresponding to the noise distribution of triplet responses, we can still recover the pairwise distances between points in the embedding. There are no guarantees however of recovering the true embedding for any algorithm when given noiseless triplets. In fact, the 1-dimensional problem has been proven to be impossible in general via an elegant number theoretic argument in [7]. In the noiseless setting, the problem can be posed as a linear program and solved via a range of methods to recover an embedding that satisfies triplet constraints. Unfortunately, it can easily be shown that satisfying identical triplet constraints is not a sufficient condition for their embeddings to be identical up to a similarity. Moreover, there is no known bound on the difference between embeddings satisfying the same constraints. However, if the number of triplets is large, constraint satisfaction algorithms such as the Stochastic Triplet Embedding method discussed in [1] empirically perform well even on noiseless data. For this reason, we will analyze the performance of different ordinal embedding algorithms on synthetic noisy and noiseless datasets.

4 Data Generation

For the purposes of this experiment, we consider only simulated data with a known noise model. This is necessary, because the method proposed in [1] which we hope to analyze is only globally convergent under the assumption of a known link function that parametrises the distribution. This is similar in concept to maximum likelihood estimation with a generalized linear model where one can use the link function to define a loss over the samples, which are triplets in this setting.

4.1 Without Noise

We begin by randomly generating a set of n points in d dimensions which collectively forms a matrix $X \in \mathbb{R}^{n \text{ x } d}$. Next we geometrically center these points about the origin by multiplying X by $V = I_n - \frac{1}{n}1_{n\times n}$ where I_n is the n by n identity matrix and $1_{n\times n}$ is the n by n matrix of all ones. This centered matrix represents the "ground truth" embedding we wish to recover. While the centering is not fundamentally necessary to recover an embedding, learning embeddings from triplet data is only possible up to a similarity. This restriction will help when comparing learned embeddings to the ground truth embedding. To generate the triplet data, we first draw three integers $i, j, k \in [1, n]$ uniformly at random without replacement. These integers correspond to points in the ground truth embedding. k is referred to as the "head" of the triplet, the point that the other two points will be compared against. We compare the relative distances of points i and j to head k and align the triplet in the form [winner, loser, head], where the 'winner' is i if $\|x_i - x_k\|_2^2 < \|x_j - x_k\|_2^2$ and j otherwise. This process is repeated to generate $O(nd \log n)$ random, possibly repeated triplets.

While there is no theory to guarantee convergence for projected gradient descent on noiseless triplets, the authors of [1] mention that the optimization often is able to recover an embedding that accurately predicts triplets and can be aligned closely with the ground truth embedding by use of Procrustes transform. They cite ordinal rigidity as a potential reason for the behavior. Essentially, as the number of triplets grows large, the feasible region for each point that satisfies all of the constraints (which is possible in the noiseless setting) becomes small, so the total embedding must be close to the ground truth.

4.2 With Noise

The noisy case follows the same overall method as the noiseless case of first computing a ground truth embedding and randomly generating triplets. The added step is in aligning each triplet. Here we compute a score s_q for each triplet. Then for a smooth monotonic link function f, we return the correct alignment of [winner, loser, center] with probability $f(s_q) > 0.5$ and the incorrect alignment of [loser, winner, head] with probability $1 - f(s_q)$. An incorrect triplet falsely indicates that the losing point (which is farther from the head in the ground truth embedding) is nearer than the winner. For this project, we have chosen $f = \frac{1}{1+e^{-s_q}}$ to be our link function for all experiments, but any smooth, monotonic function with range $[0, 1]$ would also be valid.

5 Methods

In the case of the Stochastic Triplet Embedding, we assume that we know the link function related to the noise distribution of triplet responses. This link function naturally induces a loss over triplet constraints which, when minimized, corresponds to the maximum likelihood estimate of the unknown embedding. As an example, if the logistic function is the link function, the loss it induces corresponds to the logistic loss.

Specifically, in this context the link function maps values of the differences of distances, the scores s_q of the triplets, to the probability that the triplet constraint is true for the unknown embedding. Note that the score of a triplet can identically be defined in terms of a linear combination of elements of the Gram matrix.

$$s_q = ||x_j - x_k||^2 - ||x_i - x_k||^2 = M_{jj} - 2M_{jk} + 2M_{ik} - M_{ii}$$

Minimizing the loss induced by the link function with respect to elements of the Gram matrix is a composition of convex functions and therefore convex. As Gram matrices are positive semi-definite (PSD), this minimization is a convex program over the set of $n \times n$ PSD matrices, $\mathbb{S}_n^+$. This problem can be written as

$$\widehat{M} = \text{argmin}_{M \in \mathbb{S}^+} - \sum_{q \in S} \log\Big(1 + exp(s_q)\Big)$$

.

Solutions to this minimization are unique up to a similarity, leading to the guarantee of recovering the ground truth embedding as stated in [1]. While percentage of held out triplet constraints that the learned embedding satisfies (the empirical loss) does not immediately correspond to recovering the true embedding, it is a salient metric commonly reported in the triplet embedding literature, and we also analyze the performance of these algorithms with respect to this notion of loss as well.

In the approach detailed in [1], the above loss function is minimized via projected gradient descent. The constraint $M \succcurlyeq 0$ requires us to project M back onto the PSD cone after each gradient iteration which requires computation of the full eigendecomposition of the matrix, and this is expensive for large values of n. As a final note, we recognize that since we explicitly know the dimension of our true embedding, we likewise know the rank of the Gram matrix M^* we wish to recover. This optimization does not inherently constrain the rank of of the learned $\widehat{M}$ and may instead return a Gram matrix of an embedding in a higher dimension. To mitigate this effect, one can either regularize the nuclear norm of M and instead solve the problem

$$\widehat{M} = \text{argmin}_{M \in \mathbb{S}^+} - \sum_{q \in S} \log\Big(1 + exp(s_q)\Big) + \lambda \|M\|_*$$

or de-bias the recovered Gram matrix $\widehat{M}$ by projecting it onto the subspace spanned by the eigenvectors corresponding to the top d eigenvalues. In this experiment, we have chosen the latter approach. We will compare the performance of projected gradient descent for learning a Gram matrix to other descent methods such as projection free methods, Factored Gradient methods, and an alternate projected method that all seek to minimize the same loss function to learn the underlying embedding.

5.1 Frank-Wolfe

One method that we discussed in class that is immediately applicable in this context is the Frank-Wolfe Algorithm which proceeds as follows:

Let g_k^{FULL} be the full, unconstrained gradient update. Traditionally, in projected gradient descent, given a step length we would descend along this direction and then project back onto the constraint set, $\mathbb{S}^+$ in this case. In Frank-Wolfe, we instead consider first solving the sub-problem:

$$g_k = \arg\min_{y \in \mathbb{S}^+} < y, g_k^{\text{FULL}} >$$

where $< \cdot, \cdot >$ represented the Euclidean inner product.
Next, given parameter γ_k, we update according to: $X_{k+1} = (1 - \gamma_k)X_k + \gamma_k g_k$. Finally, if we assume that our function is β-smooth and set $\gamma_k = \frac{1}{k+1}$ then we can we achieve sub-linear convergence after T iterations of the form

$$f(X^T) - f^* \leq O(1)\frac{\beta R^2}{T+1}$$

where $f^* = \min_{x \in \mathbb{S}^+} f(x)$, the optimum within the constraint set, and R is the radius of the constraint set, defined as $R = \sup_{x,y \in \mathbb{S}^+} \|x - y\|_2$ [8].

For Frank-Wolfe to be a worthwhile approach, the per-iterate optimization must be easy to compute. When optimizing with respect to the spectrahedron, it can be shown that

$$g_k = \arg\min_{y \in \mathbb{S}^+} < y, g_k^{\text{FULL}} >= \mathbf{EV}(-g_k^{\text{FULL}})$$

[9] where $\mathbf{EV}(\mathbf{A})$ represents the largest eigenvector of matrix $\mathbf{A}$. As opposed to computing a full singular value decomposition at each iteration, computing the largest eigenvector is much less expensive, and there are existing software packages to do this effectively by use of power iteration or the Lanczos algorithm. Additionally, [9] provides convergence guarantees that improve on the general sub-linear rate to the case of optimization over the spectrahedron.

5.2 Projection onto manifold of fixed rank PSD matrices

While the optimization discussed in [1] projects onto the PSD cone, which is necessary to maintain convexity, since we know (or at least can assume) the true dimension of the embedding, we likewise know the rank of the Gram matrix, M. If we are willing to drop the constraint of convexity, we can dramatically reduce the cost of projections for low dimensional embeddings. For a d dimenisonal embedding, this can be achieved by using Lanczos method to recover the top only the d eigen pairs, rather than the full eigendecomposition. Although the problem is no longer convex, we are able to experimentally show that this method still converges and avoids local minima. Furthermore, it recovers embeddings with high accuracy one both noisy and noiseless data.

5.3 Burer-Monteiro Factored Gradient

An alternative way to formulate the same is problem in terms of embedding directly as opposed to the Gram matrix. The problem we wish to solve is the following:

$$\text{argmin}_{X \in \mathbb{R}^{nxd}} - \sum_{q \in S} \log\Big(1 + exp(s_q)\Big)$$
$$s_q = \|x_j - x_k\|^2 - \|x_i - x_k\|^2$$

Note our problem is no longer convex and we cannot guarantee that we will reach the global optimum. However if we add the constraint that X is a low rank matrix and our search space is compact and smooth, then from [10] we can show that we will still converge to the global optimum with high probability, assuming appropriate noise levels. [11] provides guarantees of sub linear convergence if start at a specially chosen initial point and use the specific step size in the paper for a loss function that is simpler but very similar to the above loss function. The convergence theory of this loss function is still an open a problem.

6 Experiments

To quantitatively analyze the performance of these algorithms we generated a random embedding using $n = 100$ points in $d = 5$ dimensions. Using these points, we randomly generated noisy and noiseless test and training datasets by the process described in Section 4 with $|S| = 10nd \log n$ triplets for training, satisfying the sample complexity bound stated in [1]. For the noisy data, we also kept an estimate of the best possible logistic and empirical loss in the given dimension which serves as a bound on the achievable accuracy of the methods. This is helpful for detecting instances of over-fitting by recovering a higher-dimensional embedding than the stated problem. All models were evaluated on the same ground truth embedding with identical train and test sets. For the nonconvex methods, we employed multiple random initializations to mitigate the effects of local optima issues. In each case, we kept a running measure of the prediction accuracy of the learned embedding on the training data, the logistic loss of the learned embedding, and the time elapsed per iterate. Furthermore, to qualitatively analyze the accuracy of the learned embeddings, we also computed embeddings on a smaller random dataset with $n = 20$ points in $d = 2$ dimensions, which allowed us to plot the learned embeddings compared to the ground truth to detect trends that were not as easily captured by measures of error purely.

7 Results

7.1 Noiseless Data

In Figure 1 and 2 we present results of the performance of the 4 algorithms on clean data. Factored Gradient converges to the highest accuracy of the 4 methods and in the least time, though it initially diverged with our choice of parameters. By contrast, Frank-Wolfe did not converge on noiseless data. If we calculate the empirical loss per iteration on the training data for the Frank-Wolfe algorithm, it does decrease but does not achieve a high accuracy. This suggests that the learned embedding does satisfy many triplet constraints but does not align closely with the true embedding, as indicated by the logistic loss not decreasing. Finally, both versions of projected gradient descent (Stoachstic Triplet Embedding (STE), which projects onto the PSD cone and rank-D, which projects onto the manifold of rank-d PSD matrices) recover highly accuracte embeddings, though rank-d does so in less time as shown in Figure 2.

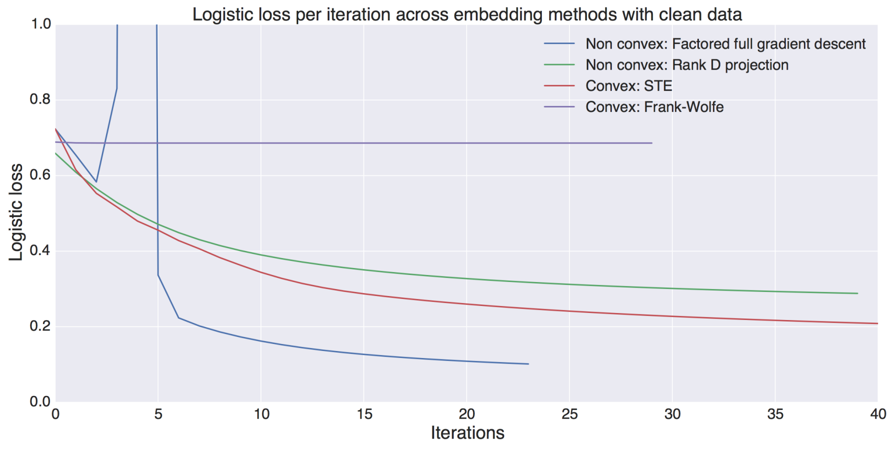

Figure 1: Logistic loss versus iteration curves for different algorithms on noiseless triplet data

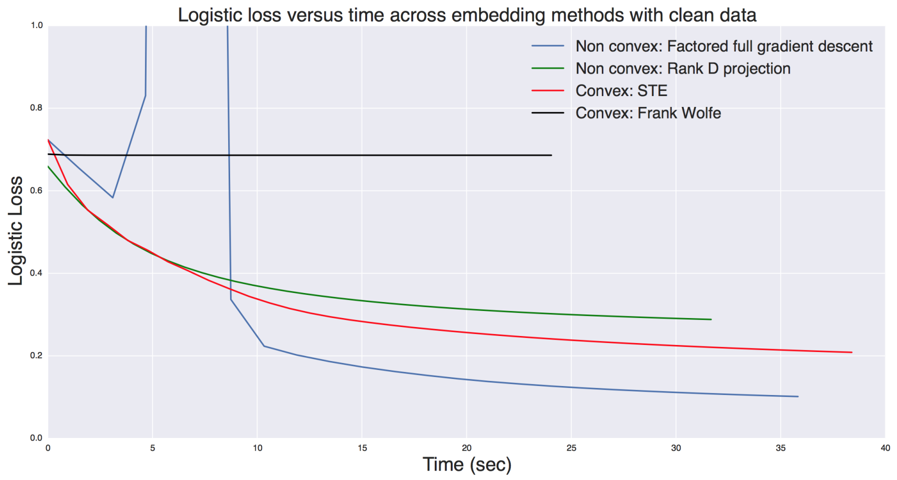

Figure 2: Logistic loss versus time curves for different algorithms on noiseless triplet data

Additionally, we present more detailed accuracy results of the four algorithms in Table 1. For each algorithm, we compute the logistic and empirical loss of the embedding returned by each algorithm on both the training and testing datasets. The Learned Dimension represents the dimension of the output of each embedding algorithm. For Factored Gradient, which optimizes with respect to the embedding matrix X directly is trivially rank 5. For the rank-d projection, the learned matrix is also rank 5 in this problem. For the other two methods, the learned dimension corresponds to the rank of the learned Gram matrix prior to projection. This number is helps to detect instances of over-fitting during training. Average Recovery Error measures the difference between the learned embedding and true embedding. For a learned embedding $\widehat{X}$ and a ground truth embedding X^*, it is calculated as follows:

1. Recover the embedding:$\widehat{M} = USV^T$. $\widehat{X} = (\sum_{i=1}^{d} \sigma_i u_i * v_i^T)[:, 0 : d]$
2. Procrustes Transformation: $\tilde{X} = \min_{\alpha \in \mathbb{R}, \Omega \in \mathbb{R}^{n \times n}} \|X^* - \alpha \Omega \widehat{X}\|_2^2$ subject to $\Omega^T \Omega = I$.
3. Average Recovery Error $= \|\tilde{X} - X^*\|_F$

Factored Gradient Descent exhibited the best performance with noiseless data on all measures of loss. Furthermore, there is a tight correspondence between the training and testing errors, indicating good generalization. By contrast, the STE algorithm with relies on projected gradient descent is learning a high dimensional embedding before projection, and this contributes to its poor generalization performance. Despite this, STE still has the best recovery error performance of all algorithms. Rank-d projection performed similarly to projection onto the PSD cone (STE), as expected. Frank-Wolfe did not perform well, but still satisfied many triplets.

Table 1: Performance of all algorithms on noiseless triplet data.

Algorithm	**Logistic Loss**		**Empirical Loss**		**Average Recovery Error**	**Learned Dimension**
	Training	**Testing**	**Training**	**Testing**		
STE (Projected Gradient Descent)	0.2085	0.3454	0.0304	0.1438	0.005585	63
Frank-Wolfe	0.6861	0.6860	0.3362	0.3372	0.01349	6
Rank-D Projection	0.2769	0.2893	0.08382	0.1013	0.03712	5
Factored Gradient Descent	0.1014	0.1097	0.01279	0.01876	0.05229	5

7.2 Noisy Data

In Figures 3 and 4 we present the results of the 4 algorithms on noisy triplet data. Note here that 0 error on training or testing is no longer possible as the training and testing sets contain falsely labelled and even directly contradictory triplets. In each figure, we also plot a horizontal line giving an estimate of the best possible loss for a d dimensional embedding. It is important to note that for the STE algorithm and its Frank-Wolfe variant, nothing implicitly constrains the rank of the learned Gram matrix (and therefore the dimension of the learned embedding) during training, and we only recover an embedding in the correct dimension via a projection step at the end. Because of this, both algorithms can potentially over-fit the data by recovering an embedding in a higher dimension where certain triplets are no longer contradictory. This behavior can be seen in both figures as the training error of the STE algorithm drops below the achievable error in the specified dimension. On our synthetic datasets where we can actually estimate the best achievable error by knowing the ground truth embedding this can be corrected for, but this is an important consideration for real datasets where the true embedding, and possibly the true embedding dimension is unknown.

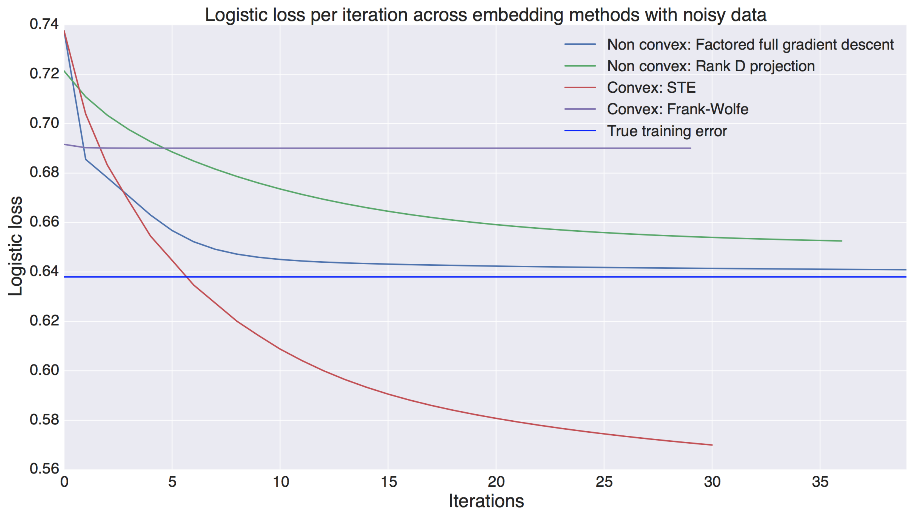

Figure 3: Logistic loss versus iteration curves for different algorithms on noisy triplet data

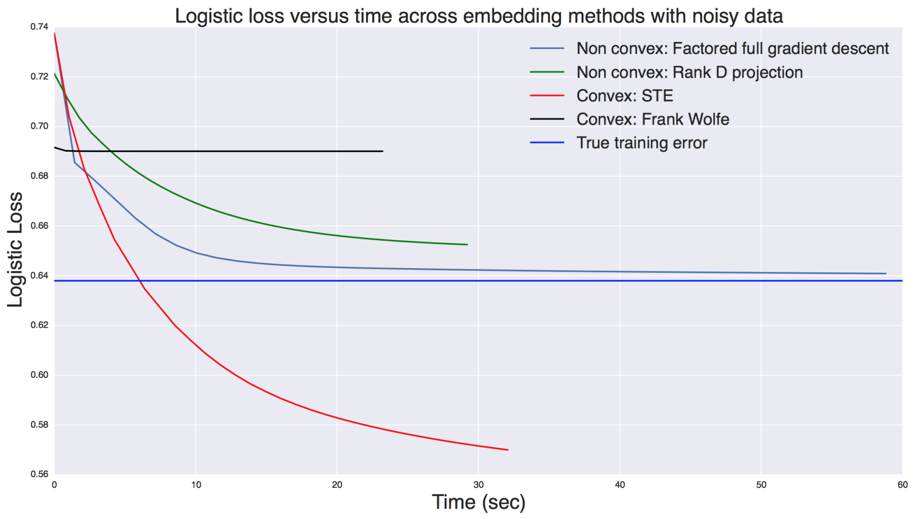

Figure 4: Logistic loss versus time curves for different algorithms on noisy triplet data

In Table 2 we present the results of both algorithms on noisy data. As in the previous case. Factored Gradient showed the greatest performance in terms of empirical and logistic loss on the test set. Note that comparing training error across algorithms no longer makes sense in this context. Because the rank of the Gram matrix and therefore the dimension of the learned embedding, is not explicitly constrained until projection after convergence, it is possible for both Frank-Wolfe and STE to satisfy a greater portion of the triplets by simply learning a higher dimensional embedding where noisy triplets are no longer contradictory. Effectively, this allows these algorithms to fit noise as we know the true rank of the Gram matrix M should be $d = 5$ here. This behavior is especially apparent in STE which, prior to projection, learned a $74-$ dimensional embedding, and this leads to its poor generalization performance. Despite this STE still recovers the embedding to the highest accuracy. As before, Frank-Wolfe performs poorly and rank-d performs similarly to projection onto the PSD cone. Note as well that noise does simplify this problem. For all algorithms except STE, the average recovery error was lower for embedding from noisy data.

Table 2: Performance of all algorithms on noisy triplet data.

Algorithm	**Logistic Loss**		**Empirical Loss**		**Average**	**Learned**
	Training	**Testing**	**Training**	**Testing**	**Recovery Error**	**Dimension**
STE (Projected Gradient Descent)	0.5700	0.6620	0.2898	0.3953	0.008044	74
Frank-Wolfe	0.6901	0.6903	0.4382	0.4452	0.01303	14
Rank-D Projection	0.6525	0.6595	0.3884	0.3983	0.02791	5
Factored Gradient Descent	0.6405	0.6578	0.3687	0.3898	0.02824	5

8 Discussion

8.1 Projected Gradient Descent

The major issue with the method is controlling over-fitting, and were we to redo this experiment, we would also consider this loss with an added nuclear norm penalty to ideally constrain the rank of the learned Gram matrix prior to projection. Doing so will help ameliorate the poor generalization performance. As noted this, algorithm recovered the most accurate embeddings when given both noisy and noiseless data. Though its performance with respect to the logistic and empirical losses was worse than that of Factored Gradient, this result none the less agrees with the intuition that learning the gram matrix in this fashion truly is the maximum likelihood estimate of the ground truth embedding. From a practical standpoint, despite the excellent performance in recovering the embedding, over-fitting is a major issue when using this algorithm. Even if one does not truly care about measures of loss and only wishes to recover the embedding, because this algorithm can over-fit, it is difficult to choose step length and stopping criteria. In this case, since we worked with synthetic data from a known distribution and a ground truth embedding, we could estimate the best error in the chosen dimension and use this information to differentiate between the algorithm converging to the desired embedding and it over-fitting. However, on real data where the true noise distribution or dimension is not exactly known, it can be difficult to tell when this algorithm is converging beyond simple metrics such as the norm of the gradient. A major advantage of this algorithm, however, is that its convergence seems to be robust to the choice of step length. This allowed us to be aggressive with our choice of step length in the early iterations, resulting in good initial convergence.

8.2 Frank-Wolfe

Frank-Wolfe performed poorly in this problem. Not only did it recovery higher dimensional embeddings, the algorithm failed to converge to highly accurate embeddings in reasonable time. We tried many different sequences of step lengths, including $\alpha_k = \frac{2}{k+2}$ which can be shown to guarantee a sublinear rate, but performance did not change greatly. A potential explanation for this poor performance can be observed when plotting the learned embedding for 2-dimensional data. Two behaviors are common. First, if the algorithm is run for many iterations, the points in the learned embedding tend to collapse onto each other. This can actually be corrected for. If we prematurely terminate the optimization, and scale the learned embedding, it is still possible to achieve good recovery of the true embedding, despite poor performance with respect to both measures of loss. The other major issue we observes is that the learned embedding is often approximately 1-dimensional, regardless of the true dimension. his is not always the case, but in this scenario, the algorithm will fail to recover the true embedding to reasonable accuracy. Finally, the spectrum of the gradient matrices tends to be relatively uniform. This is a problem when using methods such as Lanczos' Method to learn the top eigenvector. As the dimension of the true embedding increases, the more uniform the spectrum often is for its top eigenvalues. This can lead to slow convergence of Lanczos' Method which increases the cost of the Frank-Wolfe sub-problem, which can negate the benefit of using Frank-Wolfe as oppose to other projected methods. As an indicator of this difficulty, the original implementation of Lanczos' Method that we employed for our code from SciPy's Sparse library actually failed to recover the leading eigenvector.

8.3 Projection onto manifold of fixed rank PSD matrices

As expected, this method performs similarly to projected gradient descent with the obvious advantage of avoiding over-fitting issues by constraining the rank via projection at each iteration. In the setting we tested, the number of points was not large enough for the lower cost of the projection to show faster convergence than the projection onto the PSD cone. Since the function is non convex and hard to analyse we could not be as aggressive with our step size choices. While the algorithm appeared to be robust to step size choices in the early iterations, as we neared convergence, we recognize that large step sizes may push us away from an optimum or cause the algorithm to converge slowly. This behavior may be fixable by careful selection of a step size at each iteration, but this will increase the computational cost. In a version of this algorithm that we have instead written in optimized C code with an added line search, this method is the fastest by a wide margin for embeddings of large numbers of points. As a final note, we did not observe any serious local optima issues with this algorithm resulting from the non-convexity. It may be the case that this algorithm did converge to a solution that was only locally optimal but this solution was still close to the global optimum. However, this is difficult to disambiguate from incomplete convergence and has little practical effect.

8.4 Burer-Monteiro Factored Gradient

Though it did not recover embeddings to the highest accuracy, Factored Gradient still exhibited the best performance on both measures of loss in this problem. Furthermore, though the convergence was slightly slower for Factored Gradient compared to STE on both noisy and noiseless data, as shown in Figures 2 and 4, Factored Gradient avoids over-fitting issues. Beyond generalizing better, this makes detecting convergence an easier task. When testing on significantly larger embeddings, especially in low dimensions, we expect this method to outperform STE, though selecting step lengths may be more challenging. One important practical consideration for implementing this algorithm to be used on a large data set, is that the gradient computation is slightly more expensive with respect to the embedding matrix X than it is with respect to the Gram matrix M. This was observable in our study but the difference was negligible because the number of points and therefore the number of triplets was small. To speed up the computation of the gradient with respect to the embedding, it is helpful to use the chain rule to compute the outer part of the gradient with respect to the Gram matrix, and then perform updates with respect to the embedding matrix.

Finally, the excellent performance of Factored Gradient on this problem motivates two theoretical questions. First, it is unclear why no local optima issues were observed in our experiments. Though convergence guarantees do exist for the ordinary Factored Gradient setting where the loss is with respect to a the factored matrix directly, as was shown in [12], no similar guarantees exist for the case where the objective is a composition of functions and this inner function is with respect to the factored matrix. Furthermore, Factored Gradient performed especially well compared to the other algorithms on noiseless data. While there are arguments centered around graph rigidity that may explain this, there is yet to be a rigorous theoretical justification.

References

[1] Lalit Jain, Kevin G Jamieson, and Rob Nowak. Finite sample prediction and recovery bounds for ordinal embedding. In *Advances In Neural Information Processing Systems*, pages 2703–2711, 2016.

[2] Neil Stewart, Gordon DA Brown, and Nick Chater. Absolute identification by relative judgment. *Psychological review*, 112(4):881, 2005.

[3] Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Tauman Kalai. Adaptively learning the crowd kernel. *arXiv preprint arXiv:1105.1033*, 2011.

[4] Sameer Agarwal, Josh Wills, Lawrence Cayton, Gert RG Lanckriet, David J Kriegman, and Serge Belongie. Generalized non-metric multidimensional scaling. In *AISTATS*, pages 11–18, 2007.

[5] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[6] Mark A Davenport, Yaniv Plan, Ewout van den Berg, and Mary Wootters. 1-bit matrix completion. *Information and Inference*, 3(3):189–223, 2014.

[7] Jordan S Ellenberg and Dion Gijswijt. On large subsets of fqn with no three-term arithmetic progression. *arXiv preprint arXiv:1605.09223*, 2016.

[8] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML (1)*, pages 427–435, 2013.

[9] Dan Garber. Faster projection-free convex optimization over the spectrahedron. *arXiv preprint arXiv:1605.06203*, 2016.

[10] Nicolas Boumal, Vlad Voroninski, and Afonso Bandeira. The non-convex burer-monteiro approach works on smooth semidefinite programs. In *Advances In Neural Information Processing Systems*, pages 2757–2765, 2016.

[11] Srinadh Bhojanapalli, Anastasios Kyrillidis, and Sujay Sanghavi. Dropping convexity for faster semi-definite optimization. *arXiv preprint*, 2015.

[12] Dohyung Park, Anastasios Kyrillidis, Constantine Caramanis, and Sujay Sanghavi. Finding low-rank solutions to matrix problems, efficiently and provably. *arXiv preprint arXiv:1606.03168*, 2016.