Scalable Ordinal Embedding to Model Text Similarity

PhD Thesis Proposal

Jesse Anderton

March 21, 2017

Abstract

Practitioners of Machine Learning and related fields commonly seek out embeddings of object collections into some Euclidean space. These embeddings are useful for dimensionality reduction, for data visualization, as concrete representations of abstract notions of similarity for similarity search, or as features for some downstream learning task such as web search or sentiment analysis. A wide array of such techniques exist, ranging from traditional (PCA, MDS) to trendy (word2vec, deep learning).

While most existing techniques rely on preserving some type of *exact numeric* data (feature values, or estimates of various statistics), I propose to develop and apply large-scale techniques for embedding and similarity search using purely *ordinal* data (e.g. "object a is more similar to b than to c"). Recent theoretical advances show that ordinal data does not inherently lose information, in the sense that, when carefully applied to an appropriate dataset, there is an embedding satisfying ordinality which is unique up to similarity transforms (scaling, translation, reflection, and rotation). Further, ordinality is often a more natural way to represent the common goal of finding an embedding which preserves some notion of similarity without taking noisy statistical estimates too literally.

The work I propose focuses on three tasks: selecting the minimal ordinal data needed to produce a high-quality embedding, embedding large-scale datasets of high dimensionality, and developing ordinal embeddings that depend on contextual features for, e.g., recommender systems.

1 Introduction

A broad variety of tasks in Machine Learning (ML), Natural Language Processing (NLP), and Information Retrieval (IR) depend on some underlying notion of *similarity* between objects. Supervised learning attempts to assign labels to objects based on a notion of feature similarity, while unsupervised learning attempts to discover hidden patterns of similarity between objects. IR involves searching for documents which are topically similar to a query, and many NLP pipelines rely on similarity between words, documents, translations, paraphrases, and so on.

These various notions of similarity are often made concrete by assigning to each object a position within some metric space. These positions might take the form of a feature representation (which one hopes corresponds somehow to the desired notion of similarity), or can be discovered in the course of learning by optimizing some notion of similarity based on concrete data (e.g. positioning words within a Euclidean space so they are near other words which appear in similar grammatical contexts).

My work develops a set of tools for positioning objects into Euclidean space, with both practical and theoretical contributions. Specifically, I focus on *Ordinal Embedding*, which aims to place objects into a configuration that preserves some set of ordinal constraints (where the order is derived from some notion of similarity, e.g. observed user preferences). The most commonly-used type of ordinal constraint is a triple (a, b, c) of embedded objects, meaning that object a should be placed closer to object b than to object c. To be precise, we deal with a set of constraints of the form

$$\|a - b\| < \|a - c\|,$$

where $\|\cdot\|$ denotes the Euclidean norm within the embedding.

In practice, Ordinal Embedding methods are currently used primarily with crowdsourced comparisons over small object collections. For example, in a typical task one might ask people to compare three facial

expressions by identifying the pair which exhibits more similar emotions. The embedding derived from many such comparisons can then reveal a global structure across the spectrum of emotions. There are also applications to recommendation systems: human assessors might be asked which of two movies is a better suggestion for a consumer who already likes a certain movie. Such an embedding of movies could be used not only for making movie recommendations, but also to seek particular properties of those movies to explain why they are similar. In my work, I propose to extend these methods to more general embedding tasks over similarity functions that derive from some existing dataset rather than from human comparisons. In particular, I aim to show that ordinal embedding is a natural tool for capturing different notions of semantic similarity between text which can supplement existing approaches to estimating text similarity.

I propose to address four specific questions related to this embedding task.

1. (Active learning) What is the minimal set of comparisons needed to fully specify an embedding, and how can we adaptively select such comparisons?
2. (Embedding) Given a set of such comparisons, how can we efficiently find an embedding of a (very large, high-dimensional) dataset which satisfies as many constraints as possible?
3. (Ordinal Geometry) Can we infer some of the geometric structure of the data from a set of comparisons? For example, can we prove whether a Euclidean embedding into a given number of dimensions is possible, or can we identify the nearest neighbors of an arbitrary object?
4. (Text Similarity) How might we apply such an embedding method toward text understanding, similarity search, or other tasks in IR and NLP?

I have already made substantial progress toward answering the first three questions, as I will show in Section 2. My proposed work to complete my thesis amounts to first completing and publishing this preliminary work, and then applying the results toward improvements in large-scale analysis of text similarity for IR and NLP.

1.1 Mathematical Preliminaries and Notation

This section provides a brief primer on the relevant geometry, and introduces the notation we use throughout the document. Refer to Table 1 for a quick reference.

We take as input some sequence of objects $\mathcal{X} = x_1, x_2, \ldots$. These objects could be anything: movies, text documents, photographs of faces, etc. While we conceptually have an infinite sequence, in practice we have only the first n objects, $X^n = \{x_1, \ldots, x_n\}$. Furthermore, we are only provided with the object identifiers $\{1, \ldots, n\} =: [n]$. Although object features may be available, when they are we assume that they are used to answer similarity comparisons and that all relevant feature data is thus made available through ordinal comparisons.

We also have access to some *comparison oracle*, which is a function $cmp(a, b, c) : [n] \times [n] \times [n] \to \{-1, +1\}$ which indicates whether a is more similar to b or to c (i.e. b if $cmp(a, b, c) = -1$, c if $cmp(a, b, c) = +1$). We will sometimes allow the oracle to provide additional answers, such as "unknown," but only when we explicitly say so. In practice, the comparison oracle is typically implemented as a crowdsourcing system in which comparison results are inferred by combining answers from multiple human assessors. Alternatively, comparison outcomes could be determined by some mathematical model run on an existing dataset (e.g., for word similarity, word b might be chosen as more similar to word a if the two words occur in the same grammatical contexts more often). A hybrid model is even possible, e.g. using human assessors only to clarify when the feature representation does not provide a clear comparison, but this possibility has not yet been explored in the literature and we do not consider it.

We assume that comparisons are transitive, that each comparison has a correct answer $cmp^*(a, b, c)$, and that for some (possibly unknown) dimensionality d there is an embedding $Y^* \in \mathbb{R}^{n \times d}$ of X^n which reproduces the correct comparison outcomes in the sense that

$$cmp^*(a, b, c) = -1 \iff \|y_a - y_b\| < \|y_a - y_c\| \text{ and } cmp^*(a, b, c) = +1 \iff \|y_a - y_b\| > \|y_a - y_c\|, \quad (1)$$

where y_i denotes the position of x_i in Y^*. Indeed, it is well known that any ordering of pairwise distances can be embedded within $\mathbb{R}^{n-2}$ [Borg and Groenen, 2005], so this is always the case (even for noisy comparisons).

Table 1: Notation Reference

Name/Symbol	Type/Value	Meaning
$\mathcal{X}$	$\{x_1, x_2, \ldots\}$	The (infinite) sequence of possible objects.
X^n	$\{x_1, \ldots, x_n\}$	The n objects we have access to.
$[n]$	$\{1, \ldots, n\}$	The set of object identifiers.
$cmp(a, b, c)$	$[n] \times [n] \times [n] \to \{-1, +1\}$	A (possibly noisy) comparison oracle.
$cmp^*(a, b, c)$	$[n] \times [n] \times [n] \to \{-1, +1\}$	A noise-free comparison oracle.
Y^*	$\mathbb{R}^{n \times d}$	An embedding consistent with all true comparisons.
d^*	$1 \leq d^* \leq n-2$	The smallest dimensionality such that $Y^* \in \mathbb{R}^{n \times d}$ exists.
$\mathcal{T}$	$\{(a, b, c), \ldots\}$ for $a, b, c \in [n]$	A set of comparison triples.
$\mathcal{Y}_{\mathcal{T},d}$	Subset of $\mathbb{R}^{n \times d}$	The embeddings in $\mathbb{R}^d$ consistent with $\mathcal{T}$ by Eq. 1.
$\delta_{a,b}$	$\|y_a - y_b\|$	Euclidean distance between two points in ("true") Y^*
$\hat{d}_{a,b}$	$\|\hat{y}_a - \hat{y}_b\|$	Euclidean distance between two points in embedding $\hat{Y}$
$r_a(b)$	$0, 1, \ldots, n-1$	Rank of x_b when X^n are sorted by distance from x_a.

It is more interesting to consider whether there is an embedding into some dimensionality $d \ll n$. We will denote by d^* the *smallest* dimensionality such that an embedding $Y^* \in \mathbb{R}^{n \times d^*}$ exists which satisfies Eq. 1.

The correct embedding Y^* is not unique. It is not hard to see that any embedding which rotates, reflects, translates, or scales Y^* is also consistent with the same comparisons. Such transformations are called *similarity transformations*, *isotonic transformations*, or Procrustes transformations. This provides one common way to evaluate how well a proposed embedding reproduces some gold standard embedding. Given a gold standard embedding Y^* and a proposed embedding $\hat{Y}$, one can calculate the Procrustes transformation of $\hat{Y}$ which makes it match Y^* as closely as possible (see [Borg and Groenen, 2005] for details). The evaluation can then measure how far the points in the transformed matrix $\hat{Y}$ are from their correct positions in Y^*.

Even disregarding similarity transformations, Y^* is still not unique. Each point in the embedding can be perturbed slightly without changing its ordering relative to the other points, and so without violating any comparisons. This is particularly true when we're dealing with a small number of objects, or with an incomplete set of comparisons. Let $\mathcal{T} = \{(a, b, c), \ldots\}$ be a set of *comparison triples* (or just "triples"), with $a, b, c \in [n]$ and each triple meaning that object x_a is more similar to object x_b than to object x_c (that is, that $cmp^*(a, b, c) = -1$). For such a set of triples, let $\mathcal{Y}_{\mathcal{T},d}$ be the set of all embeddings into $\mathbb{R}^d$ which are consistent with the triples in $\mathcal{T}$ in the sense defined by Eq. 1. We explore at length how to select a small subset $\mathcal{T}$ of all possible (correct) triples so that any embedding in $\mathcal{Y}_{\mathcal{T},d^*}$ is close to Y^*.

We also introduce a few other conventions. We will often assume that some Y^* is the "true" embedding, especially when comparisons are answered based on some available data rather than from crowdsourcing. We denote by $\delta_{a,b}$ the Euclidean distance between points x_a and x_b in Y^*, and by $\hat{d}_{a,b}$ the points' Euclidean distance in some embedding $\hat{Y}$. We sometimes rank the members of X^n by increasing distance from some "anchor" point x_a, and refer to the rank of an arbitrary point x_b as $r_a(b)$. We define $r_a(a) = 0$; the nearest neighbor of x_a is $b : r_a(b) = 1$, the k-nearest neighbors are $\{b : 0 < r_a(b) \leq k\}$, and so on.

Geometrically speaking, each triple (a, b, c) can be viewed as a constraint on any of the three points.

- x_a must lie in the halfspace containing x_b and defined by the hyperplane orthogonal to the vector halfway from x_c to x_b.
- x_b must lie within the ball centered on x_a and with x_c on its boundary (i.e. with radius δ_{ac}).
- x_c must lie outside the ball centered on x_a and with x_b on its boundary (i.e. with radius δ_{ab}).

1.2 Related Work

The problem of finding an embedding consistent with ordinal data is classically known as Nonmetric Multidimensional Scaling (NMDS). It was first formulated and addressed by Shepard [1962a,b] and Kruskal

[1964a,b]. For a thorough treatment of classical approaches in this field, see Borg and Groenen [2005]. It was developed as a means for testing hypotheses in psychology and social sciences, and the original applications dealt with very small collections of datasets (generally $n < 100$). Typically, similarity data about all pairs of objects were collected from human assessors by various means, and a $n \times n$ *dissimilarity matrix* was constructed whose entries p_{ij} were larger when items i and j were considered less similar by the assessors. It is common to treat the numeric values of the entries p_{ij} as meaningless in themselves, and simply to reconstruct their order in an embedding. This focus on ordinality rather than numeric values is what is meant by "nonmetric."

Distance Geometry is closely related, and is treated extensively by Liberti et al. [2012] and Dattorro [2016]. The primary objective here can be viewed as a matrix completion task: given some of the pairwise distances δ_{ij} between objects, construct a complete distance matrix of minimal rank. Alternatively, one might seek an embedding which recovers the known pairwise distances. It turns out that such a distance matrix of rank r can easily be embedded into $\mathbb{R}^r$, e.g. using the methods of Sippl and Scheraga [1985, 1986]. These works also prove conditions showing when an embedding of a given dimensionality which satisfies the distance matrix exists, and address the important case of reconstructing a rank r distance matrix given exact distances from all objects to just $r+1$ affinely-independent objects. When only noisy distance estimates are available, producing an embedding is still straightforward using (metric) Multidimensional Scaling (MDS).

Ordinal Embedding. Ordinal embedding has been heavily studied, particularly by the metric and kernel learning communities. Metric and kernel learning focus on finding linear transformations of known features to produce a Mahalanobis distance (matrix) which is consistent with ordinal constraints. It differs from our work in that it focuses on transforming existing features, rather than on positioning points entirely from ordinal data. Early approaches employed semidefinite programming [Weinberger et al., 2006, Xing et al., 2003] and/or required eigenvalue decompositions, which are computationally expensive for large datasets. Later approaches focused on minimizing Bregman divergences [Davis et al., 2007, Kulis et al., 2009, Jain et al., 2012], which is guaranteed to find a positive semidefinite (PSD) kernel, or on ignoring semidefiniteness until convergence and projecting the output matrix to the nearest PSD matrix [Chechik et al., 2010]. The latter work focuses on the large-scale setting of image retrieval, exploiting feature sparsity and relaxing the need for semidefiniteness for more efficient optimization.

Ordinal embedding without features has been studied by Agarwal et al. [2007, 2010], who provide a flexible and modular algorithm with proven convergence guarantees. McFee and Lanckriet [2011] consider how to learn a similarity function which is as consistent as possible with multiple sets of features and of ordinal constraints. Another approach, tailored to crowdsourcing and based on explaining disagreement between human assessors, was developed by Tamuz et al. [2011] and extended by Van der Maaten and Weinberger [2012]. These models assume multiple noisy observations of each triple and employ the modeling assumption that disagreement stems entirely from comparing nearly-equal similarities. We explore alternatives to this assumption in Section 2.1.1.

A few more recent models show promise in recovering exact point positions. Local (and Soft) Ordinal Embedding [Terada and von Luxburg, 2014] provide embeddings with guarantees on accurate density recovery. Hashimoto et al. [2015] prove metric recovery results over certain directed graphs, including kNN adjacency graphs, using an approach based on random walks.

Triple Selection. Another topic we explore in some depth is the problem of identifying a minimal subset of ordinal comparisons which will lead embedding algorithms to the best possible solution. As we have mentioned, the original applications of Ordinal Embedding were often studies in psychology and the social sciences. Several of these experiments are described in Borg and Groenen [2005]. Human assessors were sometimes asked triple comparisons (i.e. choosing which of two objects is most similar to a third), and were sometimes given a stack of cards and asked to divide them into a "more similar" group and a "less similar" group. This splitting process was repeated on each group until assessors could no longer meaningfully separate the cards. A partial ordering of pairwise similarities was then inferred from the step at which a given pair of cards was separated.

In more modern crowdsourcing applications, it is common practice to simply select triple comparisons at random. However, Jamieson and Nowak [2011] prove that exact positions cannot be recovered using less than $\Omega(n^3)$ of the $O(n^3)$ possible triples, so this strategy is far from optimal. Indeed, the same paper proves

that at least $\Omega(d^* n \log n)$ comparisons are necessary when they are selected adaptively, and the authors conjecture a matching upper bound. The algorithm they suggest for this is very impractical, however. They propose to iterate through all possible triples; at each question, one would seek an embedding to satisfy all known triples so far, plus one additional triple representing one of the answers to the new question. If embedding exist which satisfying both possible answers, the triple would be sent for assessment. One reason this does not work in practice is that no known embedding routine can reliably find an embedding satisfying *all* triples, even when one exists.

There has been substantial recent progress in proving recovery results for different subsets of triples. The first such result was in Terada and von Luxburg [2014], showing that the correct embedding can be recovered from a kNN adjacency matrix, which implies that $O(nk(n-k)) = O(n^2)$ triples are sufficient. A followup work, Kleindessner and von Luxburg [2014], proved the long-standing conjecture that, under reasonable distributional assumptions, an embedding of a sufficiently large number of objects which preserves the total ordering of pairwise distances between all objects must place all objects to within ϵ of their correct positions, where $\epsilon \to 0$ as $n \to \infty$. We will subsequently refer to this as "recovering the embedding to within small ϵ," or simply as "recovering the embedding." The subsequent paper by Arias-Castro [2015] proved similar recovery results for embeddings preserving smaller subsets of ordinal data. It first proves that preserving all triples suffices to recover the embedding (note that in general the total ordering of pairwise distances is not implied by the complete set of triples), and then it proves (among other results) that preserving a total ordering of each point's k-nearest neighbors, for certain values of k, recovers the embedding. This confirms the results of Terada and von Luxburg [2014] that *local* comparisons suffice for full embedding recovery. This is very important in practice, because it permits full recovery even for datasets where comparison questions become meaningless when objects are too dissimilar. We propose to exploit this fact in Section 3.2.1.

Only a few authors have proposed approaches for selecting triple comparisons beyond random selection. The main paper along this vein is Tamuz et al. [2011], which proposes an active learning strategy which selects the next triple to maximize the expected information gain for their worker response model (discussed above). We have tested this algorithm extensively and with various modifications, and found that it is indeed significantly better than random question selection. However, the best embeddings from the triples it selects do not come particularly close to full embedding recovery, so there is room for improvement. We propose several alternative algorithms in Sections 2.1.2 and 3.1.

Ordinal Geometry. Our final major area of focus is studying the geometry implied by a set of triples, noting that any geometric property implied by the triples must appear in any embedding which preserves those triples. There are only a few results published in this area. Kleindessner and von Luxburg [2015] provide a dimensionality estimation tool based on the k-nearest neighbors adjacency graph, which can be obtained from triple comparisons. Kleindessner and von Luxburg [2016b] employ the lens depth function, which can be calculated from triples, to perform standard Machine Learning tasks such as classification and clustering. Finally, Kleindessner and von Luxburg [2016a] provide kernel functions which can be calculated directly from triples, taking advantage of a recent proof that a valid kernel can be computed from Kendall's tau. The dissimilarity between two objects is calculated based on an estimate of the Kendall's tau value for the rankings of all objects by distance to each of the two objects being compared.

2 Preliminary Work

I have organized the discussion of my preliminary results around the four main questions I proposed to address in the introduction. Proposed extensions of this work are discussed in Section 3.

2.1 Active Learning

The state of the art for selecting triple questions for ordinal embedding is somewhat lacking. In practice, triple questions are often selected at random. To date, there is only one method proposed in the literature which outperforms random triple selection — namely, the Crowd Kernel method of Tamuz et al. [2011]. We have found that this method outperforms random triple selection, but not by a wide margin. In fact, it is currently an open question whether it is possible to do better. Jain et al. [2016] prove an upper bound

on expected loss for a certain class of embedding models which begins to give acceptable bounds when $O(d^* n \log n)$ random triples are selected, and Jamieson and Nowak [2011] have already proven that at least $\Omega(d^* n \log n)$ adaptive comparisons are required in order to fully recover all point positions.

In light of this, one might suppose that there is little to be gained by further studying the triple selection question. However, this conclusion is unsatisfying. First, the expected loss bounds proven by Jain et al. [2016] make no guarantee that similar distances will be well-represented; the class of models they consider supposes that the probability of an incorrect answer increases as the distances $dist(a, b)$ and $dist(a, c)$ approach each other. This is based on a human response model which supposes that very similar distances are hard to precisely compare, and this response model does not apply in all cases for Ordinal Embedding. Second, there are times when arbitrary triple comparisons cannot be answered, as I will discuss below, and in this case one desires a more adaptable approach to triple selection. Third, triple selection algorithms based on an understanding of the underlying geometry are of both theoretical and practical interest: they may reveal properties such as which regions of the embedding are already well-determined by the existing triples or how to develop a better embedding algorithm. Indeed, these theoretical concerns motivate the Ordinal Geometry portion of this document, in Sections 2.3 and 3.3.

2.1.1 User Response Models

This section describes the unpublished work in Anderton et al. [2014]. Our initial interest in Ordinal Embedding was in exploring whether existing methods, particularly those of Tamuz et al. [2011], could be applied to use crowdsourced similarity assessments to embed text fragments from web documents ("nuggets"). We ran a lab study answering triple comparison questions, asking assessors to answer entailment questions such as, "Does nugget b or c contain more information in common with nugget a?" Despite our efforts to preprocess the nuggets so as to include many nuggets with similar information, the most common response by far was that neither b nor c contained anything in common with a. Observing that the nuggets all shared *some* common information, as they were all selected to be on the same topic, we concluded that the problem was that workers could only meaningfully compare nuggets when they are similar enough to have obvious information in common. In short, workers could answer only *local* questions (where all objects are sufficiently similar), and the nuggets in most triples were too different to be answerable.

We decided to update the embedding model to account for this locality requirement. The original model gives the probability that a worker will say that b is more similar to a than is c as

$$\hat{p}^a_{bc} = \frac{\lambda + \delta^2_{ac}}{2\lambda + \delta^2_{ab} + \delta^2_{ac}}, \tag{2}$$

recalling that δ^2_{ab} denotes the squared Euclidean distance between objects a and b in the "correct" embedding Y^*, and where $\lambda \in \mathbb{R}_+$ is a smoothing parameter. Our model introduced a latent distance threshold τ meaning that a given triple is likely to be judged unanswerable when both b and c have distance more than τ away from a. Our model gives the probability that the user would say neither b nor c is similar to a and the probability that the user would prefer b over c as

$$\hat{p}_{neither} = \frac{\mu + \delta^2_{ab}}{\mu + \tau^2 + \delta^2_{ab}} \cdot \frac{\mu + \delta^2_{ac}}{\mu + \tau^2 + \delta^2_{ac}}, \quad \hat{p}^a_{bc} = (1 - \hat{p}_{neither}) \cdot \frac{\lambda + \delta^2_{ac}}{2\lambda + \delta^2_{ab} + \delta^2_{ac}}. \tag{3}$$

with $\mu \in \mathbb{R}_+$ an additional smoothing parameter.

Not knowing how to explain nugget entailment to crowdsourcing workers, we instead built two simpler collections to assess. Each collection contained 100 objects; the first contained popular movies, and the second contained foods from various cuisines around the world. In both cases, workers were asked to recommend either item b or c as a substitute for item a (e.g., "Your friend calls saying they wanted to see Movie a, but the tickets were sold out, and asks whether they would enjoy Movie b or Movie c more."). We compared results on the original model, with users forced to make a decision, to an updated model where users could say "neither b nor c is a good replacement for a." We selected triples using an active learning method similar to the Crowd Kernel method but updated to use our model, and during embedding we estimated the distance threshold d along with the object positions using a quasi-Newton optimization algorithm.

Embedding results for these two datasets appear in Figure 1. We also evaluate the prediction accuracy of our response model on held-out triples in Figure 2. Although we outperformed the Crowd Kernel baseline

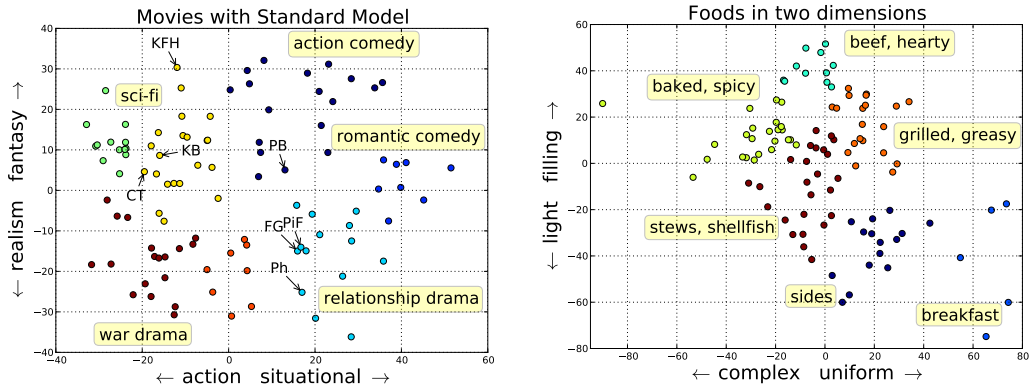

Figure 1: Embeddings into $\mathbb{R}^2$ using our updated model. Axis and cluster labels are speculative. Points are colored based on a hierarchical clustering of the objects.

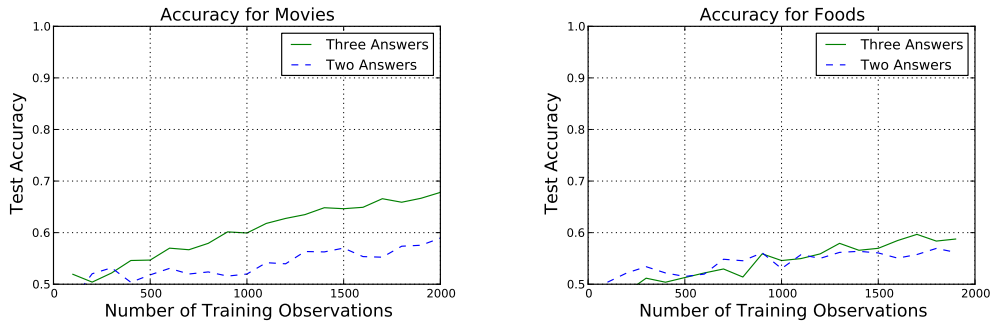

Figure 2: Prediction accuracy of held-out triples based on embeddings. To ensure comparable results, the *Three Answers* model was forced to choose either b or c, so random guessing has an accuracy of 1/2 for either model.

for the movies dataset, the foods dataset did not fare as well. We speculate that this may be because people are very well-versed in the detailed classification of movies; movie genres and sub-genres are widely discussed. Foods, on the other hand, present more ambiguity: is a hamburger more similar to a steak or to a ham sandwich? In this case, it could be that disagreement between users was more significant than the incomparability of triples.

To better handle user disagreement, we developed another modification of our model to account for user preferences. This model introduced a per-user scaling factor for each dimension of the embedding to reflect users' different sensitivities to different aspects of the objects under comparison. For example, a vegetarian would find some foods unacceptable substitutes for others despite being otherwise similar, and movie-goers may differ in the details they consider when comparing films. Our user model employs the same probabilities from Eq. 3, but with a per-user distance $\delta^2_{k,ab}$ reflecting additional parameters for user k's scaling of the embedding dimensions. We also trained a distance threshold parameter τ_k individually for each user. This model has a *General Kernel*, which reflects the embedding with unscaled dimensions shared by all users, and for each user a *Personalized Kernel* with the dimensions scaled with the user's parameters. Learning curves for this updated model are shown in Figure 3. User models helped somewhat, but performance on the foods dataset still does not particularly recommend our method.

To summarize this work, we created a response model which has a built-in capacity to allow users to say "neither b nor c is similar to a." This yielded improvements for our movies dataset, but no clear improvement for the foods dataset. We also added per-user notions of how different objects must be before they seem too dissimilar to compare, and of per-user scaling of each dimension, again yielding improvements for movies but not for foods. We submitted this work for publication to KDD in 2014, but it was not accepted. While we believe there is promise to per-user response models and response models which attempt to select questions

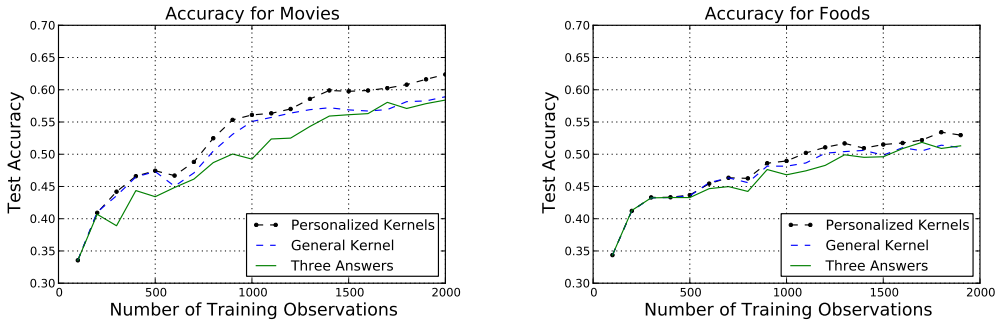

Figure 3: Prediction accuracy of held-out triples using user model. Note that models can choose b, c, or "neither," so random guessing gives an accuracy of $1/3$.

that users can answer, we have not pursued this work any further. Instead, we turned to explore the more basic question of selecting a minimal set of triples in the simpler case when the questions are all answered reliably.

2.1.2 Rank-based Traversal and Efficient Sorting

This section describes the unpublished work in Anderton et al. [2016b]. In this work, we considered the problem of selecting a small set of triples $\mathcal{T} = \{(a, b, c), \ldots\}$ adequate to achieve a high-quality embedding. The lower bound of Jamieson and Nowak [2011] has that $\Omega(d^* n \log n)$ triples are needed to *precisely position* all points, but it is unknown what quality can be achieved with fewer triples. They prove in the same paper that when triples are selected at random, $\Omega(n^3)$ are needed to precisely position all points. It is not clear how this relates to the recent work by Jain et al. [2016] showing that expected model likelihood becomes "small" when only $O(d^* n \log n)$ random triples are collected.

We observe that all triples can be recovered (via transitivity) using $O(n^2 \log n)$ comparisons, simply by treating each point in turn as an anchor x_a and sorting the others by increasing distance to x_a. When $d^* = \Omega(n)$, this is already asymptotically the best we can do. However, we propose a better algorithm for the case when $d^* \ll n$ which we conjecture to recover all triples with just $O(n^2)$ triples collected. One would typically stop early, as a good embedding can be achieved with a much smaller number of triples when collected in the order we specify.

Our algorithm is easy to understand and implement. We strategically choose some *anchor* x_a and sort all the other points by their distances to x_a. We then use the ordinal information learned so far to choose the next anchor and repeat the process. By selecting widely-spread anchors, we rapidly learn about the ordering in all regions and dimensions of the space. For the first $O(d^*)$ anchors, we use $O(n \log n)$ triples per anchor to sort all points by the distance to the anchor. Empirically speaking, after $O(d^*)$ anchors the current embedding gives a good partially-sorted list of all points by embedded distance to the new anchor. We are thus able to adaptively sort using only $O(n)$ triples per anchor on average when $d^* \ll n$. Our testing suggests that the algorithm needs only $O(d^* n \log n)$ adaptive triples to achieve a "good" embedding (matching the lower bound), and linear comparisons per anchor thereafter to improve the embedding. An embedding of triples selected by our algorithm as compared to randomly-selected triples can be seen in Figure 4.

Our Algorithm. We now state our algorithm more precisely. We visit each object in the collection in farthest-rank-first traversal (FRFT) order, as specified in Section 2.3.1. This traversal order requires us to sort the collection for each item we visit. It chooses points which are "as far as possible" from the previously-visited points; its first few points are on or near the convex hull of the collection, and subsequent points tend to be evenly-spread throughout the interior. We improve from $O(n^2 \log n)$ to $O(n^2)$ comparisons by a careful choice of sort algorithm and initial permutation of objects for each sort operation.

We make use of the fact that a nearly-sorted list can be completely sorted using just $O(n)$ comparisons. In order to sort, we first embed the collection using the Soft Ordinal Embedding (SOE) algorithm [Terada and von Luxburg, 2014] on the triples collected so far. We then sort the objects by increasing distance

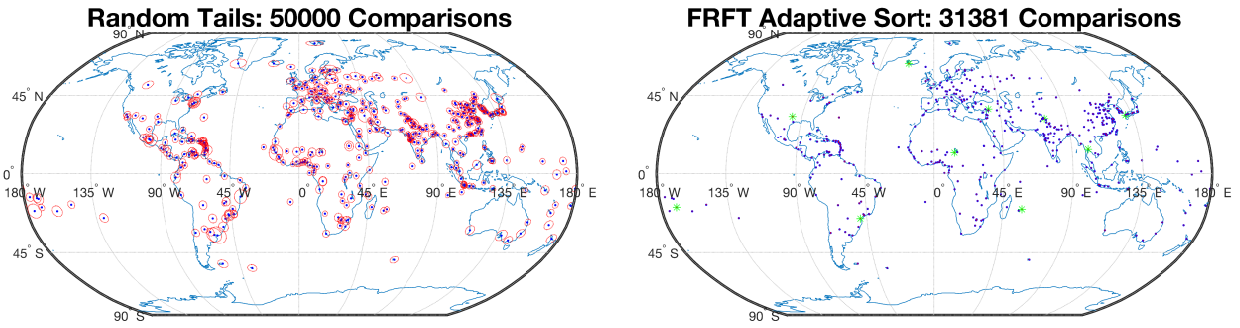

Figure 4: Embedding error comparison for two sets of triples on a 3d dataset with 500 cities of the world. Circle radii show average distance error for a given city, and green asterisks denote anchors used by the FRFT algorithm.

to the new anchor within the embedding, and use that order as our initial permutation for our sort algorithm. After testing many sort algorithms empirically, we found that Splaysort [Moffat et al., 1996] used the fewest comparisons. However, all such adaptive sort algorithms tend to waste comparisons when the initial permutation is random. As a further (constant factor) enhancement, we recommend using Mergesort for the first few rounds, and switching to Splaysort once the initial permutation and the final sorted list are similar enough. We suggest using the `Reg` disorder criterion, defined in Moffat et al. [1996], for this purpose. We show how this criterion drops from anchor to anchor in Figure 5. Splaysort uses a linear number of comparisons once this value is low enough.

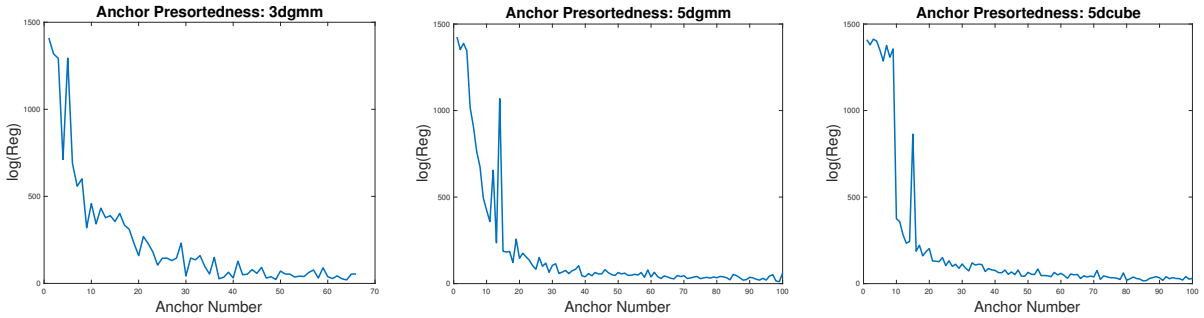

Figure 5: Anchor number versus permutation disorder on synthetic data. The datasets from left to right are a 3D GMM, a 5D GMM, and a 5D cube. The spikes correspond to low-quality embeddings, which can be avoided by repeating the embedding process until a low-loss solution is found.

Evaluation Criteria. For each of our experiments, comparison questions were answered based on true object positions in some configuration we hoped to recover. We evaluated a resulting embedding by comparing it to the true point positions. We used two evaluation measures: one based on recovering pairwise distances between points, and one based on recovering the rankings of the points by distance from each of the objects.

Our first measure, *Distance RMSE* (root mean squared error), is based on the fact that in a perfect embedding all pairwise distances would be scaled by the same constant. That is, there is some scaling constant $s \in \mathbb{R}$ such that for all points $i, j \in [n]$, $\delta_{i,j} \approx s\hat{d}_{i,j}$. We know the exact pairwise distances for our datasets, so we fit an optimal $\hat{s}$ to the embedding distances and report the RMSE of the residuals,

$$drmse(Y^*, \hat{Y}) \equiv \min_{\hat{s}} \left(\frac{1}{n} \sum_{i<j} (\delta_{i,j} - \hat{s}\hat{d}_{i,j})^2 \right)^{1/2} \tag{4}$$

This value measures the average "warping" of the embedding compared to the real positions. Smaller is better, and zero is perfect. However, as the value is not normalized against the "size" of the embedding values are not comparable for different datasets.

Distance RMSE tends to be more affected by errors for larger distances, and we wish to also measure performance on the other end of the spectrum. We are also interested in the ability to predict the ranking of X^n by distance from any point in the embedding. Our second measure achieves both. τ_{AP}, introduced by Yilmaz et al. [2008] and commonly used for Information Retrieval, is a top-heavy rank correlation coefficient similar to Kendall's τ, but which places more weight on correctly ranking the beginning of the list (e.g. at shorter distances). Like Kendall's τ, the perfect ranking has $\tau_{AP} = +1$, a random permutation has τ_{AP} close to zero, and a reverse permutation has $\tau_{AP} = -1$. It can be thought of as the expected value of the following random variable T: select a point $j \in [n]$ uniformly at random, then select another point i uniformly from the set of points ranked before j. Let T be an indicator variable which is 1 when i is ranked before j in the correct ranking and 0 otherwise. We report the mean τ_{AP} value of the rankings of all points in an embedding.

Empirical Results. We ran our algorithm on four datasets with small dimensionality: GMMs in 3D and 5D, a 5D cube, and a dataset of 500 city positions around the world, represented in 3D Euclidean space (taking the planet's center as the origin). Due to time constraints when preparing our paper, we were not able to provide results on larger datasets. This speaks to the larger challenge of efficiently embedding large collections, which we later consider in Sections 2.2 and 3.2.

We compared the following algorithms for question selection. Results are shown in Figure 6.

- *Random Tails* iterates over all points in round-robin fashion, adding a randomly selected triple (a, b, c) : $\delta_{a,b} < \delta_{a,c}$ for each. This simulates the random question method commonly used in practice.
- *Crowd Kernel* is the authors' implementation of the "Crowd Kernel" algorithm [Tamuz et al., 2011]. However, instead of embedding using the authors' algorithm, we embed with SOE for comparable results.
- *FRFT Ranking* visits the points in FRFT order and adds the $n - 2$ triples expressing the correct order of all points by distance to the anchor. This requires prior access to correct rankings, e.g. when comparisons are based on a fixed dataset.
- *FRFT Adaptive Sort* is the algorithm described above. In contrast with FRFT Ranking, it is well-suited to crowdsourcing.
- *Landmarks* visits points ("landmarks") in FRFT order. When each point is visited, $2n$ triples are added to insert the new point into the correct position in the rankings of all other points with respect to the previous landmarks. This requires prior access to correct rankings.
- *kNN* also iterates over all points in round-robin fashion. In the k^{th} iteration, it adds the triple $\delta_{a,b} < \delta_{a,c}$, where the ranks $r_a(b) = k$ and $r_a(c) = k + 1$. Thus, kn triples express the total ordering of each point's k-nearest neighbors. This requires prior access to correct rankings.

We found that the FRFT Ranking algorithm outperforms the others by a wide margin, in terms of converging to nearly-perfect embeddings with small subsets of triples. FRFT Adaptive Sort is close behind.

The performance of the Crowd Kernel algorithm is disappointing on these data. We generally found the algorithm to be quite competitive on other datasets, outperforming Random Tails, as seen in the 3D GMM results. It is not clear what holds the algorithm back for the other datasets.

This method shows promise, particularly with respect to the FRFT order. However, the need to embed the set after sorting for each anchor can greatly slow the algorithm, and the need to sort the entire set for each anchor is likely "wasting" triples when the permutation in the embedding is already nearly correct. Surely many of these triples could be inferred from the triples already collected, if we only knew how. These considerations motivated our subsequent work.

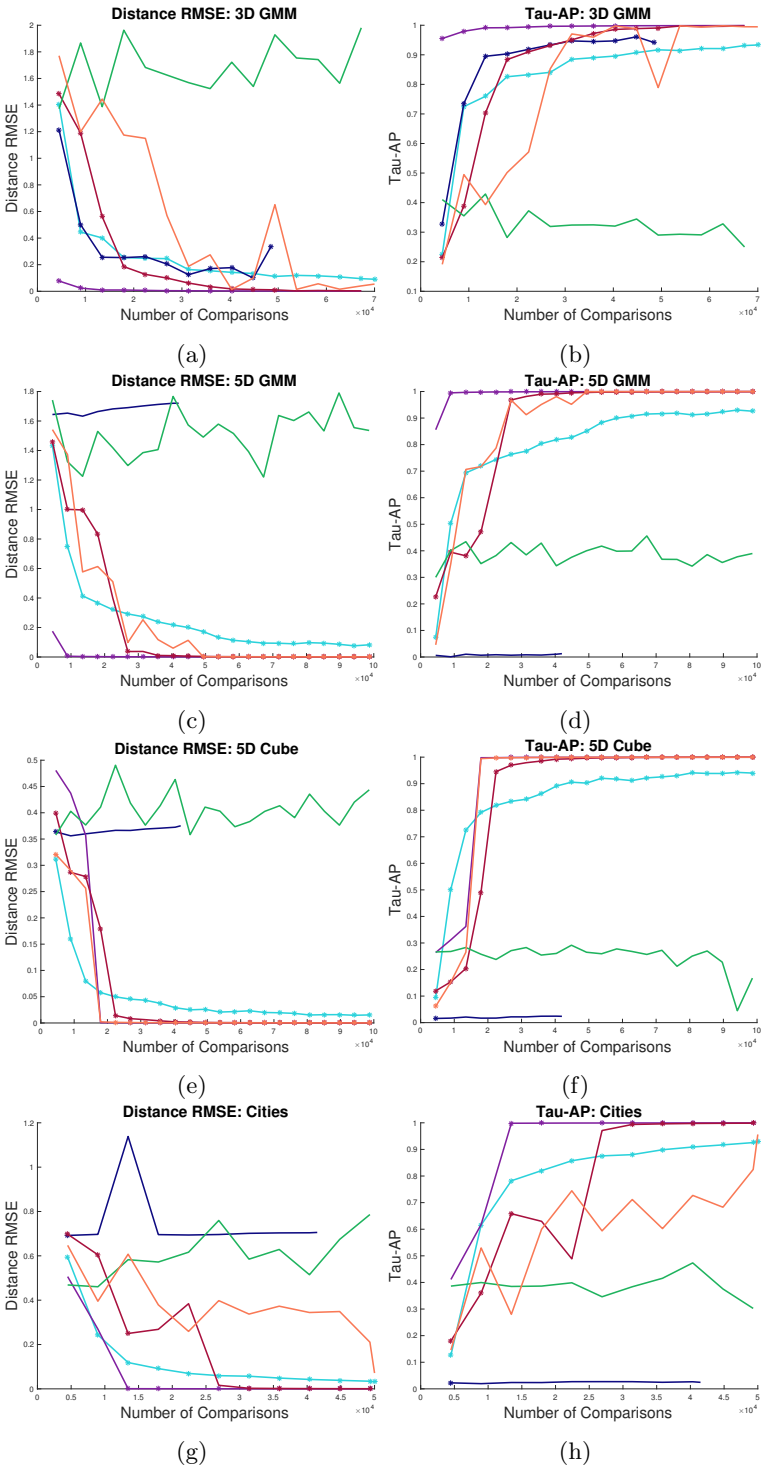

Figure 6: Learning curves for all datasets.

FRFT Ranking and FRFT Adaptive Sort can be embedded successfully more consistently, and achieve lower RMSE and higher τ-AP.

The kNN algorithm is competitive when it can be embedded well. However, it would require more comparisons when triples are not known in advance.

Landmarks can almost never be embedded successfully. It also suffers from an inability to distinguish between points when a small number of landmarks is used.

Random Tails and Crowd Kernel sometimes have an early advantage over FRFT Adaptive Sort, but after an initial period of rapid convergence they slow down. Theory suggests they take $O(n^3)$ triples to converge.

2.2 Embedding

The work in this section continues the task of finding a small subset of comparisons to embed the set. However, our adaptive algorithms for choosing comparisons are based on geometric insights into the correct embeddings and often suggest better embedding methods. There seems to be a natural interaction between comparison selection and embedding algorithms. In this section, we explore embedding methods which are informed by the comparison selection process and thus also approach minimal sets of comparisons.

2.2.1 Direct Embedding through Basis Estimation

This section describes the unpublished work in Anderton et al. [2016a]. We propose an algorithm to embed a set directly from the triples, without optimizing any machine learning objective. These embeddings can be produced very rapidly, using just $O(d^* n \log n)$ triples and $\Theta(n^2 d^{*2})$ total operations. This compares very well to optimization-based embedders, many of which take at least $O(n^2)$ operations *per learning iteration* and which typically require thousands of iterations to converge.

However, our embeddings are of lesser quality than those achieved when an optimization algorithm such as Soft Ordinal Embedding [Terada and von Luxburg, 2014] converges to a global optimum. Our algorithm is of interest for its theoretical contributions, as a means of selecting $O(d^* n \log n)$ triples adaptively which yield good embeddings in practice, and as an embedding algorithm which works even when optimizers fail to find a good local optimum. We find that such optimization failures are typical when the number of parameters $n \times d$ is sufficiently large. Repeated random initialization can help, but the number of attempts required to find a good embedding seems to grow with the number of parameters, so in practice this is the first embedding method which finds a better-than-random embedding for large sets.

General Approach. Before describing our embedding algorithm, it is helpful to explain the exact algorithm which it approximates. Ideally, we would first identify a set of orthogonal line segments which each span the full space to serve as our "axes." These segments need not coincide at the origin; it suffices for each to be orthogonal to all of the others. Having identified an axis for each dimension, we would then embed each of our n objects by finding the closest point on each axis to the object. The coordinate we choose on an axis for an object's embedding is the distance of this closest point from one of the endpoints of the axis. If our line segments were truly orthogonal, if they spanned all dimensions of the latent space, and if we could identify the exact closest point to any object on any line, then this would produce an exact embedding of the objects.

In practice, we have to discover such line segments and distances using only triple comparisons. This data is inherently discrete and imprecise, although the precision improves as the density of X^n increases. To be more precise, when the sequence $\mathcal{X} = \{x_1, x_2, \ldots\}$ are i.i.d. draws from some underlying density over a simply-connected, bounded subset of $\mathbb{R}^{d^*}$, then our algorithm converges to the perfect embedding algorithm just described in the limit as $n \to \infty$.

Our approximation algorithm is described briefly here, and then in detail in the following text. Due to the limitations of ordinal data, we use subsets of X^n in place of line segments. We aim to find a subset whose members are as close to collinear as possible. We accomplish this using a subroutine which finds points which are on or near the convex hull of any subset of X^n. Our embedding algorithm iteratively identifies new axes by selecting pairs of objects near the boundary of the convex hull of the set, $\text{conv}(Y^*)$, and on opposite sides of the set, such that the line passing through both objects is (nearly) orthogonal to the previous axes. We

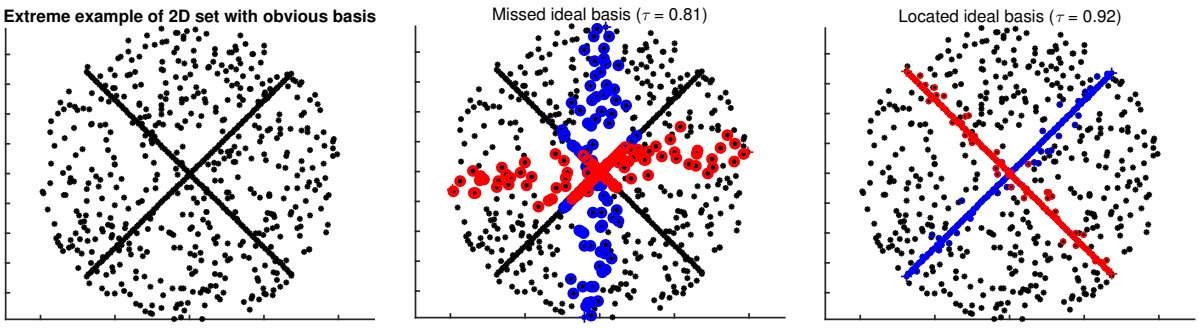

Figure 7: 2D points set X (left) includes two subsets of colinear, dense, evenly-spaced, points that make obvious good axes. We show two possible axis pairs and evaluate the implied embeddings; τ is the mean Kendall's τ between true rankings and basis-estimated rankings. Middle: two axes red and blue found by our algorithm are not the best basis, but reasonable; Right: ideal axis red, blue also includes a few other points since the gap on the collinear points is not small enough; thus $\tau < 1$.

then use our convex hull subroutine to find the nearly-collinear points which constitute points along the axis. We illustrate two possible axis pairs for a 2D set in Figure 7. We assign an embedding coordinate to a point along an axis by selecting one of the points along the axis as the "closest" to the point we are embedding, and using the integer-valued index of this closest point when the points along the axis are sorted by distance from one of the axis endpoints.

Choosing Axis Endpoints. We choose our first two axis endpoints so that they are far from each other and both on the boundary of the convex hull of the set. To find points on this boundary we use the fact that for any points $x, y \in X^n$, if y is the farthest point from x in X^n then all points in X^n are contained in the ball centered on x and with y on its boundary; this implies that y is on the boundary of the convex hull of X^n. We select the first endpoint of our first axis by selecting a point at random and then finding the point furthest from it in X^n. Our second endpoint is the farthest point in X^n from the first endpoint.

We want the endpoints of our additional axes to be as far as possible from the convex hull of all previous axis endpoints. This will help ensure that our endpoints are affinely independent of the previous axes, so each axis will cross some new dimension of the space.

We will explain our algorithm by analogy to the two dimensional case. Suppose we already have two endpoints $p_1, p_2 \in [n]$ which are opposite each other on the convex hull of Y^*, and that all other points in X^n are "between" them, in the sense that for any other $x \in [n]$, both $\delta_{p_1,x} < \delta_{p_1,p_2}$ and $\delta_{p_2,x} < \delta_{p_2,p_1}$ hold. Then any such x is in the *lens* between p_1 and p_2; that is, in the intersection of open balls centered on p_1 and p_2 with radii δ_{p_1,p_2}.

Now suppose that two points p and q both reside in this lens, but that p is farther from both p_1 and p_2 than is q. Then we say that p is "above" q with respect to $\text{conv}(\{p_1, p_2\})$, because its distance to its closest point p' on this convex hull must be greater than the distance from q to *its* closest point q' on the hull. Further, if q is above any point z then p must also be above z; the property is transitive because object rankings by distance are transitive. Also note that this property generalizes naturally to an arbitrary number of dimensions by considering "lenses" formed from the intersection of more than two balls, centered on more than two axis endpoints.

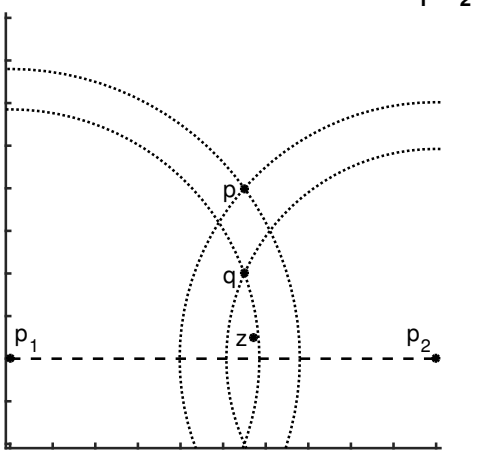

Figure 8: Point p is "above" q because q is found in the intersection of balls centered on p_1 and p_2 and extending to p. This implies that p is farther from the line than q and thus not in $conv(\{p_1, p_2\})$.

Returning to axis endpoint selection, recall that we want to identify a point which is as far as possible from the convex hull of the previous endpoints in some orthogonal direction. We find this point by choosing that point which is "above" the largest number of other points with respect to the previous endpoints. The second endpoint is the point farthest from the first in the set of points within the lens for the existing axis endpoints.

Convex Hull Estimation. In order to find the points close to the line between our new axis endpoints, we need a way to estimate the convex hull of the endpoints. We also rely on convex hull estimation to decide when to stop adding axes, as we will see below. Our convex hull estimate $\widehat{conv}$ relies on the fact that any union of balls which all coincide in some point must contain the convex hull of the ball centers. We state this formally as follows and prove it in Anderton et al. [2016a].

Theorem 1. *Let $\mathcal{B} = \{B(v_1, r_1), \ldots, B(v_k, r_k)\}$ be a set of closed balls in $\mathbb{R}^d$ with centers $v_1, \ldots, v_k$ and radii $r_1, \ldots, r_k$, respectively. If the intersection of all the balls in $\mathcal{B}$ is not empty, then* $\text{conv}(\{v_1, \ldots, v_k\})$ *is a subset of their union.*

Suppose we want to identify points from X^n which lie in the convex hull of a set $P = \{p_1, \ldots, p_k\} \subset [n]$.

We can find candidates for this set by choosing an arbitrary point q and identifying the set

$$C_q(P) \equiv \{x \in [n] : \exists p \in P, r_p[x] \leq r_p[q]\}. \tag{5}$$

Theorem 1 tells us that this will contain $\text{conv}(P) \cap X^n$ as a subset, but it may also contain some additional points. In order to filter out most of these false positives, we take the intersection of $C_q(P)$ across all possible points q as our estimate.

$$\widehat{conv}(P) = \bigcap_{q \in X^n} C_q(P) = \{x \in [n] : \forall q \in [n], \exists p \in P, r_p[x] \leq r_p[q]\} \tag{6}$$

As the intersection of sets which all contain $\text{conv}(P) \cap X^n$, we know $\widehat{conv}(P)$ contains $\text{conv}(P) \cap X^n$. We have additionally proven the following theorem, which says that any false positives in our estimate are close to the boundary of $conv(P)$.

Theorem 2. *Let $\widehat{conv}(A)$ be the estimate of $conv(A)$ for some $A \subseteq X^n \subset \mathbb{R}^d$. If the largest empty ball in $conv(\widehat{conv}(A))$ has radius ϵ, and the maximum distance between any two points in A is m, then for any $c \in \widehat{conv}(A)$ the distance to the closest point $c' \in conv(A)$ is less than $\sqrt{\epsilon(2m+\epsilon)}$. Further, there is no point $x \in X$ such that $r_a[x] < r_a[c]$ for all $a \in A$.*

A trivial corollary of this theorem states that as the density constant $\epsilon \to 0$, our estimate $\widehat{conv}(P) \to \text{conv}(P)$.

When we take points from $\widehat{conv}(\{p_1, p_2\})$ as an axis, it is easy to show that the points along the axis are order-consistent: increasing distance order from one axis endpoint is decreasing distance order from the other endpoint (matching the intuition for points along a line). However, the points may be somewhat distant from the line in an arbitrary direction. Technically speaking, the points are contained in a cylinder centered on the line whose radius is bounded by Theorem 2.

Dimensionality Estimation. Our estimate of the dimensionality of X^n is the number of axes we select, which we denote by $\hat{d}$. We stop adding axes as soon as our next axis endpoint does not appear to be affinely-independent of the previous axis endpoints. Our test for affine independence relies on Carathéodory's Theorem, which states that any point in the convex hull of a set of points in $\mathbb{R}^d$ can be expressed as a convex combination of just $d+1$ or fewer of them. This is discussed in more depth in Section 3.3.1.

We define a set P containing both endpoints for the first axis, plus the first endpoint of each additional axis and our new candidate p. We have $|P| = \hat{d} + 2$. Suppose we have already found d dimensions, and the new axis endpoint is simply not in the convex hull of the previous endpoints. Then any point in $conv(P)$ is also in the convex hull of some set of all but one of the points in P. On the other hand, if $d^* > \hat{d}$ then these extra hulls are the facets of a higher-dimensional simplex, and we expect some of the points in that simplex to be far enough from these exterior facets to not be included in our $\widehat{conv}$ estimates (which may or may not be true, depending on the density of X^n).

We prove that $\hat{d} < d^*$ and that $\hat{d} \to d^*$ as $\epsilon \to 0$ with the following theorem.

Theorem 3. *Let $\mathcal{X} = \{x_1, x_2, \ldots\}$ be an infinite sequence of i.i.d. draws from some smoothly-continuous distribution over a simply connected compact subset $V \subset \mathbb{R}^d$. Also let $X_n = \{x_1, \ldots, x_n\}$ be the first n draws in $\mathcal{X}$, and let $\hat{d}_n$ be the number of axes chosen by our algorithm when the oracle answers consistently with the distances between the points in X_n. Then $\hat{d}_n \leq d$ for all n, and as $n \to \infty, \hat{d} \to d$.*

Table 2: Dimensionality Estimates
(1,000 points, avg. of 100 runs)

True d:	1	2	3	5	8	10	20
Ball	1	2	2.11	3.66	4.22	4.54	5.53
Cube	1	2	2.37	3.74	4.44	4.58	4.78
Gaussian	1	2	2.98	3.91	4.44	4.54	4.52
Sphere	1	1	2	3.09	3.85	4.08	4.93

See Table 2 for dimensionality estimates of various datasets. Since the number of points in each dataset is the same, as the dimensionality increases the density constant ϵ grows (and density decreases). This causes $\widehat{conv}$ to be less precise and leads us to underestimate the dimensionality.

Choosing Embedding Coordinates. Once we have identified all our axes, we use the rankings of X^n for a given pair of axis endpoints to choose the coordinates of each point along that axis. Our algorithm accomplishes this without any additional comparisons.

Ideally, the points along each axis would be evenly-spaced and lie along the line between the axis endpoints. We could then embed any $x \in X$ by simply finding the index of the closest point via binary search, since the members of A_i would be sorted as a bitonic array: the distance to x would descend to a minimum and then ascend. The total comparison cost would be $O(\hat{d}n \log n)$, within the theoretical lower bound.

In practice, we never have such perfect axes. When the points of X^n are in general position, no object will be found in the convex hull of any subset of d^* or fewer other objects and no member of an axis except the endpoints will lie on the line. A binary search will not find the closest point on the axis to x because the points will be not be exactly sorted by distance to x. Further, the closest point on the axis will often be closest simply because it is not found on the line, not because it is near the projection x' of x onto the line (Figure 9).

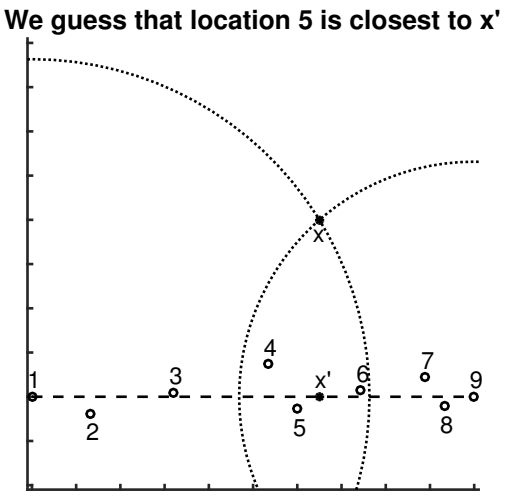

Figure 9: The circled points are axis members. 4 is closest to x. We choose the median point within the lens beneath x (containing 4, 5, and 6) as our guess at the closest point to its projection, x'.

We really want to find the axis point which is closest to x', not closest to x. The projection x' will be in the center of the lens formed from the axis endpoints with x at its apex. The lens will always contain some axis point (otherwise x would be included in the set of axis points). We select as the ordinal coordinate of x along the axis **the median index** for those axis points inside this lens.

While this may not be the axis point closest to x', especially if the density varies greatly along the axis, it costs no additional comparisons to select this point. We have found that the empirical performance on our datasets is comparable to finding the point in the lens closest to x.

While even in dense spaces our algorithm might not find orthogonal axes, if the discovered axes are indeed orthogonal we can guarantee that our embedding recovers the original metric with a precision depending on the density of X and the true dimensionality. We are able to prove the following theorem on the quality of our embeddings. If we fix d and the diameter of X in each dimension, assuming orthogonal axes, this theorem implies that when $n \to \infty$ and thus $\epsilon \to 0$, the correct distances are recovered, up to scaling.

Theorem 4. *If any ball of radius ϵ in $\mathrm{conv}(X^n)$ contains at least 1 and at most k points, and assuming $\hat{d} = d^*$ orthogonal axes are found which extend to the faces of a bounding box for X^n, using linear search in the lens for points' coordinates, then there is a scaling constant $s \in \mathbb{R}$ such that for any two points $x, y \in X^n$,*

- *the coordinate x_i on any axis A_i is bounded by its projection x'_i by $(s/k)x_i - \epsilon \leq x'_i \leq sx_i + \epsilon$, and*
- *the scaled distance estimate $\hat{d} := s \cdot \widehat{dist}(x, y)$ is within $2k\epsilon\sqrt{d^*}$ of the true distance, i.e., $dist(x, y) - 2\epsilon\sqrt{d^*} \leq \hat{d} \leq k(dist(x, y) + 2\epsilon\sqrt{d^*})$.*

Empirical Results. While most of the above discussion has focused on the algorithm as an embedding algorithm, it can also be viewed as an active learning algorithm for selecting triple comparison questions. We evaluate both aspects of our algorithm, and find that the comparisons we select far outperform random triple selection and that our embeddings are of moderate to high quality. When an optimization-based algorithm converges to a point near the global optimum, it finds a much better embedding than ours. However, our method is much faster and is able to find embeddings for datasets where all current optimization methods fail. See our embedding results on a variety of real and synthetic datasets in Table 4.

The *Basis* method embeds as per our discussion above, and *Basis+SOE* embeds with the Soft Ordinal Embedding (SOE) algorithm [Terada and von Luxburg, 2014] using the triples collected by our algorithm. *Extra+SOE* adds comparisons to sort the $2k$ nearest neighbors of each point in our Basis embedding, choosing

Table 4: Embedding Quality. Datasets to the left are synthetic, and to the right are standard (2,000 records from 20newsgroups, the positions of 500 cities, 1,000 records from MNIST digits, and 1,000 records from spambase).
* indicates global optimum was not found; means procedure computationally too expensive

Method	**Dataset**	d	$\hat{d}$	# Cmp.	τ	$k$nn	rmse
Basis	3dgmm	3	3	38K	0.71	0.64	0.77
Basis+SOE	3dgmm	3	3	38K	0.99	0.97	0.02
Extra+SOE	3dgmm	3	3	61K	**0.99**	**0.99**	**0.01**
Rand+SOE	3dgmm	3	3	38K	0.95	0.81	0.11
CK	3dgmm*	3	3	38K	-0.01	0.02	1.79
Basis	5dcube	5	3	39K	0.49	0.40	0.26
Basis+SOE	5dcube	5	6	39K	0.88	0.73	0.05
Extra+SOE	5dcube	5	6	61K	**0.94**	**0.92**	**0.03**
Rand+SOE	5dcube*	5	6	39K	0.61	0.30	0.19
CK	5dcube*	5	5	39K	0.01	0.02	0.34
Basis	5dgmm	5	3	39K	0.68	0.60	0.90
Basis+SOE	5dgmm	5	6	39K	0.94	0.66	0.14
Extra+SOE	5dgmm	5	6	62K	**0.98**	**0.97**	**0.04**
Rand+SOE	5dgmm*	5	6	39K	0.01	0.02	1.77
CK	5dgmm*	5	5	39K	-0.01	0.02	1.57

Method	**Dataset**	d	$\hat{d}$	# Cmp.	τ	$k$nn	rmse
Basis	20news	34K	3	186K	**0.11**	**0.06**	0.53
Basis+SOE	20news*	34K	6	186K	0.01	0.01	**0.34**
Extra+SOE	20news*	34K	6	310K	-0.01	0.01	0.34
Rand+SOE	20news*	34K	3	186K	0.01	0.01	0.44
CK	20news	34K	16	—	—	—	—
Basis	cities	3	2	28K	0.37	0.35	0.60
Basis+SOE	cities	3	4	28K	0.89	0.54	0.13
Extra+SOE	cities	3	4	50K	**0.96**	**0.93**	**0.05**
Rand+SOE	cities*	3	4	28K	0.01	0.02	0.75
CK	cities*	3	3	28K	0.01	0.02	0.67
Basis	digits	784	6	159K	0.52	0.29	3.18
Basis+SOE	digits*	784	12	159K	0.01	0.01	2.48
Extra+SOE	digits*	784	12	211K	0.01	0.01	2.49
Rand+SOE	digits*	784	12	159K	**0.73**	**0.40**	**2.31**
CK	digits	784	10	—	—	—	—
Basis	spam	57	3	85K	0.85	**0.78**	471
Basis+SOE	spam*	57	6	85K	-0.01	0.01	596
Extra+SOE	spam*	57	6	138K	0.01	0.01	596
Rand+SOE	spam	57	3	85K	**0.94**	0.23	**150**
CK	spam	57	10	—	—	—	—

$k = \log_2 n$, and embeds the combined comparison set. We compare against *Rand+SOE*, which embeds from random triples, and *CK*, the Crowd Kernel algorithm of Tamuz et al. [2011].

As a simple test of downstream utility, we trained Gradient Boosting classifiers on our basis embeddings and compared the classification accuracy to classifiers trained on the original feature space. The results can be seen in Table 3. Performance is good even with $\hat{d} \ll d^*$, with the exception of the ill-suited `20newsgroups` (which is at least better than random).

Table 3: Classification Accuracy, 5 folds

		Original			Embedding	
Dataset	d	Train	Test	$\hat{d}$	Train	Test
20news	34K	0.94	0.54	3	0.21	0.08
cities	3	1	0.95	2	0.99	0.90
digits	784	1	0.84	6	0.92	0.71
spam	57	0.99	0.97	3	0.85	0.74

Discussion. When viewed as an active learning method, our algorithm is the first to achieve near-perfect embedding performance using $O(d^* n \log n)$ triples, matching the proven lower bound. As an embedding algorithm, the results appear less impressive. They are promising, however, as the first algorithm to produce embeddings by geometric inference from triples (as opposed to optimization). The geometric tools developed here (convex hull estimate, affine independence tests, etc.) are of theoretical interest, and we propose further exploration of them in Section 3.3.

2.3 Ordinal Geometry

2.3.1 Subset Selection through ϵ-Nets

It is sometimes useful to identify a small subset of X^n which is well-distributed throughout the space and which somehow captures the extent and density of the set "at low resolution." When exact point positions are known, this can be accomplished by building an ϵ-net. An ϵ-net is a subset $N \subset X^n$ having the following two properties:

1. every point in X^n is within distance ϵ of some member of N, and
2. no two members of N are within ϵ of each other.

Such a subset can be identified using a so-called *farthest-first traversal* (FFT) of X^n [Gonzalez, 1985]. We choose a point at random to initialize our net. We then add points one by one until the desired number of points is reached, using the rule that each point must maximize its distance to the closest member already added to the net. More precisely, we choose the point achieving $\max_{x \in [n]} \min_{n \in N} \delta_{x,n}$, with N denoting the set of points already chosen. This max min distance is the value of ϵ when that point is added.

We can produce an approximate ϵ-net using ordinal triples by an algorithm we dub a *farthest-rank-first traversal* (FRFT). We again initialize with a random point, and add points incrementally using a rule to be described momentarily. When we add a point, we sort the remaining members of X^n by increasing distance to the new net member. Instead of choosing a point with the *max min distance to* the previous members, our rule is to choose a point with the *max min rank from* the previous members. That is, we choose a point achieving $\max_{x \in [n]} \min_{n \in N} r_n(x)$, recalling that $r_a(p)$ gives the rank of p when the members of X^n are sorted by increasing distance (or decreasing similarity) from a. Since distance increases monotonically with rank, this tends to produce a good approximation of an ϵ-net. It differs somewhat in that its choices are closer to denser regions and farther from sparser regions. It seems worthwhile to prove results to characterize this behavior.

It is often useful to discard the first (random) point, taking as the first member the point which is farthest from a random point. This guarantees that the first two net members are on $\text{conv}(X^n)$, and the next few members are near the hull boundary (in the sense that there is no point p in X^n having the net member in the convex hull of p and the other previous net members; such a point p would be closer to, or on, the hull of X^n and would achieve a larger minimum rank from the previous net members).

3 Proposed Work

3.1 Active Learning

It is still considered an open question to find an adaptive algorithm which can adaptively select $O(d^* n \log n)$ triples and position all points correctly. This is in spite of the theorem from Jain et al. [2016] showing a convergence result when $O(d^* n \log n)$ randomly-selected triples are observed, for the following reason. The theorem proves a convergence result for a particular class of loss functions in which uncertainty about assessor responses increases as the two distances being compared (δ_{ab} and δ_{ac}) approach each other. It essentially states that uncertainty of an embedding satisfying the observed (noisy) triples will converge to the uncertainty of the true embedding Y^* when some constant multiple of $d^* n \log n$ random triples are observed. This implies that distances which are actually similar will also be similar in the embedding, which means that point positions will be roughly correct. However, the theorem does not say anything about the number of random triples needed to fine-tune the embedding. In particular, the theorem of Jamieson and Nowak [2011] shows that random triple selection cannot precisely position all points with fewer than $\Omega(n^3)$ triples.

I propose to settle this open question by identifying a theory-driven adaptive algorithm with a convergence proof showing that all possible triples can be recovered using just $O(d^* n \log n)$ triples selected by the proposed algorithm, either by applying theoretical considerations to infer the remaining triples or by showing that an embedding satisfying the selected triples will respect all possible triples.

In our work in Anderton et al. [2016a,b], we have already identified several adaptive algorithms which select $O(d^* n \log n)$ triples, meeting the proven lower bound. However, we have not proven convergence results for our algorithms beyond Theorem 3 (Section 2.2.1), which relies on rather strong assumptions.

The simplest algorithm meeting the lower bound simply uses the FRFT algorithm of Section 2.3.1 to select $O(d^*)$ net members ("anchors"). This algorithm sorts all n points by distance to each anchor, so it clearly meets the lower bound. We have often run this algorithm on points with known positions, answering comparison questions with triples reflecting these known distances. Our testing shows that embeddings based on these triples produce very good embeddings on a wide variety of datasets, suggesting that the algorithm is selecting very informative triples. I propose to explore this algorithm theoretically and prove an embedding convergence result on it or on some similar algorithm.

Another promising approach is to adapt the kNN algorithm of Li and Malik [2015, 2017] to use ordinal triples rather than random vector in Euclidean space. This algorithm gives high-quality approximate k-nearest neighbor estimates for all points, using close to $O(d^* n \log n)$ total operations. Given the kNN of all points, we could use the convergence guarantee of Local Ordinal Embedding [Terada and von Luxburg, 2014] to prove an upper bound for an approximate algorithm which converges to the correct embedding in the large sample limit.

3.2 Embedding

Even given the complete set of noise-free triples and the minimum dimensionality d^*, the problem of identifying an embedding into $\mathbb{R}^{d^*}$ to satisfy the triples remains challenging. Many good algorithms exist for the case when $n \times d^*$ is not too large, but in practice these tools are typically limited to, at most, thousands of points in fewer than 10 dimensions. I propose to address this limitation in two ways. First, I will seek out ways to increase the scalability of the known embedding algorithms by applying them to subsets and building global embeddings from the results. Second, I will continue my work on embedding algorithms which supplement or replace purely optimization-based approaches with computational geometry, leveraging the geometric structure of the data whenever possible to lighten the optimization load. I describe two specific approaches in the following sections.

3.2.1 Generalizing from a Subset Embedding

Ordinal Embedding was originally conceived as a way to produce a low-dimensional embedding which preserved the order of dissimilarity values in an arbitrary matrix of pairwise dissimilarities. This is almost identical to the task undertaken by many modern representation learning approaches, such as the GloVe algorithm for word embedding [Pennington et al., 2014]. GloVe seeks an embedding into some Euclidean space which is consistent with a notion of word similarity based on their pointwise mutual information (PMI) with random variables representing various notions of the grammatical context in which those words appear. Two words which appear more often in similar grammatical contexts will have a higher PMI score, and should be placed closer within the embedding. However, current ordinal embedding techniques cannot handle the large numbers of objects and dimensions used for these representation learning tasks.

I propose to explore ordinal embedding algorithms which can handle these large-scale tasks, and to explore their use in large-scale text similarity problems. In this section, I suggest an approach which uses the current optimization approach to ordinal embedding on a small subset of points, and which then uses computational geometry to embed all subsequent points. I will test this approach on various text similarity tasks, such as word embedding and dimensionality reduction. If this approach succeeds, I will move on to more complex semantic matching tasks as described in Section 3.4.

Generalizing from a subset. Our generalization algorithm works as follows. Suppose that X^n consists of n i.i.d. samples drawn from some unknown density over some Euclidean space $\mathbb{R}^d$. We can randomly select a subset X^m for some $m \ll n$, obtaining an i.i.d. subset from the same density. For a simple example, perhaps X^n are $n = 10,000$ points drawn from the uniform ball in $\mathbb{R}^3$, and $m = 500$.

If we can accurately embed the subset X^m, then we can use that embedding to quickly obtain an approximate embedding of the full set X^n as follows. Choose $d+1$ affinely-independent points from X^m ("anchors"), and for each anchor sort the points in X^m by distance. Now suppose we have some new point $x \in X^n$ to embed. We first locate its position in each sorted list using binary search, for $O(d \log m)$ triple comparisons. We then choose a target distance to each anchor, e.g. using the midpoint between the distances to the embedded points preceding and following x in the ordered list for an anchor. Finally, we embed the point using $d+1$ approximate distances to $d+1$ affinely independent anchors. This can be done very efficiently, e.g. using the sphere intersection method of Coope [2000]. The points in X^n are embedded independently, and so can be trivially parallelized.

Early results for this method can be seen in Table 5. In each case, we set $m = 500$ and used the first $d+1$ anchors in FRFT order. The subset X^m was embedded using Soft Ordinal Embedding, which was repeated until a small loss value ($< 10^{-9}$) was achieved. The first four datasets are synthetic, while the remainder are real datasets. We have not been able to embed any dataset in this table with any published method, so the

baseline performance is no better than random. We evaluate as follows. For each of 10 runs, we compute the mean Kendall's τ score when taken across the rankings for all points in the embedding; the median of these 10 scores is reported. Notably, each run completed in less than a minute; many datasets took much less than a minute.

The scores reported on the synthetic datasets are nearly perfect, even though we only claim an approximate embedding. Performance declined for the real datasets, though remaining quite good for all but MNIST Digits. It is not clear yet what leads to the decreased scores here. The dimensionality may be inadequate for an embedding. Alternatively, the distribution could violate our underlying smoothness assumptions, so the subset may not adequately represent the distribution of points. Further study is needed to improve these scores. Nevertheless, these early results look promising.

Table 5: Embedding Accuracy (median of 10 runs). n is the # of points, $m = 500$, d the true dimensionality/# of features, and $\hat{d}$ the embedding dimensionality.

Dataset	n	d	$\hat{d}$	τ
Ball	10K	3	3	0.99
Sphere	10K	3	3	0.99
Swiss Roll	10K	3	3	0.99
GMM	10K	3	3	0.99
Spambase	4.6K	57	3	0.89
Cities	15K	3	3	0.97
20news PCA	2K	10	10	0.96
MNIST Digits PCA	1K	12	12	0.90
MNIST Digits	1K	784	12	0.57

Generalizing from Multiple Subsets. The discussion so far has a key weakness: it implicitly assumes that any object in X^n can be meaningfully placed into order with the random subset X^m. This is often not the case because many triple comparisons can not be meaningfully answered, whether because human assessors find the question meaningless or because no similarity data is available in the training corpus. For example, an embedding of documents might use cosine similarity between document TF-IDF vectors to answer triple comparisons. This will encounter problems as soon as documents with no meaningful vocabulary overlap are compared. In this case, a total ordering of documents is not directly available. As *global* comparisons are not possible, we need to rely on only *local* comparisons.

This problem may need to be addressed in a dataset-dependent way. For example, in the example of cosine similarity above we could potentially use an inverted index (mapping vocabulary words to documents which contain them) to identify subsets of documents with sufficient overlap to be mutually comparable. In a general crowdsourcing context, however, we may need to split the task into a first phase, to identify meaningful subsets, and then compare objects within subsets in a second phase.

However the subsets are obtained, our goal is to treat them individually in the manner discussed above and then to merge the individual subset embeddings into a unified global embedding. Subsets will need to overlap sufficiently in order to be merged; an embedding into d dimensions will presumably need at least $d+1$ points in common between two subsets in order to adequately orient and scale them. Several considerations need to be addressed here.

- How can we efficiently identify our subsets for a given task?
- When we are embedding the full set, how do we efficiently find the subset to which a point belongs? What do we do if there is no such set?
- How will we merge them? We could apply a Procrustes transformation [Borg and Groenen, 2005] to subsets one by one, but accuracy might be better if we apply some "k-way" merge routine which finds a transformation for all k subsets at once while distorting each subset minimally.
- If points are overlapping between subsets, how do we ensure that they are positioned as accurately as possible within each subset? For instance, perhaps we should require our $d + 1$ overlapping points to be part of the X^m subsets so they are positioned by our triple embedding routine. We can then merge all our X^m subsets, and only then proceed to embed the full set of points.

One idea is to build our subsets incrementally. We can iterate over all X^n points in random order, and maintain a list of subsets built so far. For each point, we find out whether there is an existing subset to which it can be added. If not, it becomes the first point in a new subset. As soon as a subset accumulates

m points, we apply our triple embedding routine; subsequent points are embedded into the subset using the sphere intersection approach. When two subsets with overlapping points have both been embedded, we immediately merge them and transform any embedded points into the merged space.

We have recently become aware of the work of Cucuringu and Woodworth [2015a,b], which addresses many but not all of these problems. This work may serve as a partial solution upon which to build.

3.2.2 Embedding through Computational Geometry

We discussed an alternative approach to embedding in Section 2.2.1, using computational geometry rather than optimization. If our exploration of Ordinal Geometry continues to be fruitful, we may be able to develop geometric embedding methods which guarantee exact embedding recovery. This has long been possible for exact distances, and there is reason to believe it also possible for a partial ordering of pairwise distances. Indeed, in a trivial sense this is already possible. That is, a distance matrix can be created from an arbitrary partial ordering of pairwise distances by first storing the integer-valued rank of each pairwise distance in the matrix, and then adding a sufficiently-large constant so that the triangle inequality is preserved without violating the order [Borg and Groenen, 2005]. This matrix will be nearly full-rank, but can be embedded using exact distance embedding methods (such as Sippl and Scheraga [1985]). A much more interesting question is whether the minimum dimensionality d^* can be inferred from the triples, and then whether an embedding into $\mathbb{R}^{d^*}$ can be found. Such a method would presumably be much faster and more reliable than an optimization-based approach.

Our preliminary embedding method, already discussed, relies partially on convex hull estimates between pairs of points to approximate lines, which that method uses as locations along the axes of a Cartesian coordinate system. In fact, it is possible to estimate how far along this line segment each point in the estimated convex hull lies.

Consider a sequence of points $p_1, \ldots, p_k$ lying along a line. Each point between p_1 and p_k can be expressed as a convex combination $\lambda p_1 + (1 - \lambda)p_k$ for some λ between zero and one. Further, there will be some "midpoint" i in the sequence such that all the points $p_1, \ldots, p_i$ are closer to p_1 than to p_k and all the points $p_{i+1}, \ldots, p_k$ are closer to p_k than to p_1. This midpoint can be located using standard ordinal comparisons, e.g. using binary search. We then have that $\lambda > 1/2$ for $p_1, \ldots, p_i$ and $\lambda < 1/2$ for $p_{i+1}, \ldots, p_k$. The process can be repeated recursively on each side to derive tighter intervals for each point's λ value. These intervals will all be correct, and will be larger or smaller depending on the density of the points along the line. When we have points which are only approximately on the line, we can employ a similar method to get approximate intervals. We can then use the distances along these intervals in a similar manner to our subset generalization approach (Section 3.2.1) to find approximate distances for all the points whose distance to p_1 is at most p_k.

We show early results for this method in Table 6. This algorithm uses $d^* + 1$ FRFT anchors, plus the most distance point from each anchor, to define several line segments which (almost) span the set. We use the center of the interval as a distance target, and employ the sphere intersection algorithm of Coope [2000] to embed points at the desired distances. Some care is taken to embed the anchors correctly, and some additional work is needed to transform the various λ values for different line segments of different lengths into consistent distances in the same space. It is not clear from these results whether this method will achieve perfect embeddings with further work; however, it is worth noting that these embeddings are substantially better than random, outperform our method of Section 2.2.1, and also outperform some of the weaker optimization-based approaches.

Table 6: Embedding Accuracy.
n is the # of points, d the true dimensionality/# of features, and $\hat{d}$ the embedding dimensionality.

Dataset	n	d	$\hat{d}$	τ
Cube	1K	5	5	0.72
GMM	1K	3	3	0.77
Spambase	1K	57	3	0.56
Cities	15K	3	3	0.49
20news	2K	34K	10	0.44
20news PCA	2K	10	10	0.68
MNIST Digits PCA	1K	12	12	0.65
MNIST Digits	1K	784	12	0.62

3.3 Ordinal Geometry

My work so far suggests that a wide array of geometric results can be inferred from ordinal triples. Indeed, the work of Kleindessner and von Luxburg [2015, 2016a,b] shows many such results. In this section, I highlight two areas which would be particularly useful in practice: estimating the latent dimensionality of a dataset and efficiently performing k-nearest neighbor (kNN) queries.

3.3.1 Dimensionality Estimation

Recall that we denote by d^* the smallest dimensionality into which our comparisons $\mathcal{T}$ can be embedded without violating any triples. Also recall that for any set of triples, we have $1 \leq d^* \leq n-2$. Can we tell how some estimated dimensionality $\hat{d}$ compares to d^*? That is, can we test whether an embedding into $\mathbb{R}^{\hat{d}}$ is possible without violating any triples?

Our main idea used in Anderton et al. [2016a] is to rely on convex hull estimation. Let $V \subset X^n$ be some m of our points. By Carathéodory's Theorem, any point in the convex hull $\text{conv}(V)$ must be in the convex hull of some subset of V of size d^*+1 or less. From this we can infer two properties.

1. If any point in $\text{conv}(V)$ does not reside in the convex hull of any (strict) subset of V, then we know that $d^* \geq m-1$.

2. If for every subset V of size m all the points in $\text{conv}(V) \cap X^n$ also reside in the convex hull of some subset of V then *perhaps* we have $d^* < m-1$ (or perhaps we have "holes" in X^n). Whether all such sets can be embedded into $m-1$ dimensions without violating any triples is an interesting open question. However, it is always possible that by increasing n we would discover new points that invalidate this dimensionality estimate (by filling the "holes").

It remains to show how to find the convex hull of V using triples. I am aware of two methods: one using balls and one using hyperplanes. Recall that a triple (a, b, c) can be viewed as a geometric constraint on any of the three points.

- x_a must lie in the halfspace containing x_b and defined by the hyperplane orthogonal to the vector halfway from x_c to x_b.
- x_b must lie within the ball centered on x_a and with x_c on its boundary (i.e. with radius δ_{ac}).
- x_c must lie outside the ball centered on x_a and with x_b on its boundary (i.e. with radius δ_{ab}).

Convex Hull Estimation using balls. The first method relies on balls, and in particular on the fact that given any point $p \in \mathbb{R}^{d^*}$, the union of the balls centered on the members of V and with p on their boundary contains $conv(V)$ as a subset (as stated in Theorem 1). The method works as described in Section 2.2.1. We first sort X^n by distance to each member of V. (In practice, for any $x_a \in V$ it suffices to sort only the subset of X^n closer than the most distant other point in V). We will build our estimate from the resulting rankings without using any additional triples. Choose some member of V as our point p, and initialize our estimate to the set of points which are closer to any member of V than is p — this is our union of balls. Now iterate over each point in this estimate, taking each as p in turn, and keep only the intersection of the convex hull estimates for each point. It is not hard to show that the resulting set contains all points in $\text{conv}(V) \cap X^n$, and possibly also some additional points near the boundary of the hull (as defined in Theorem 2).

This method uses $O(d^* n \log n)$ triples and identifies all points in the true convex hull, but since it is built on balls rather than hyperplanes it can also include many extra points near the hull. That is, it has no false negatives (every point in the hull is identified as such) but can have false positives (some points identified as in the hull are not). In particular, all the false positives are near the boundary of the hull.

Convex Hull Estimation using hyperplanes. A more expensive but more precise method uses hyperplanes. It is based on the following fact.

Proposition 1. *If all members of V are closer to some point b than to some point c, then all members of* $\text{conv}(V)$ *are also closer to b than to c.*

Proof. We have the triple (a, b, c) for all members $a \in V$. In other words, all members of V are on the same side of the hyperplane implied by this triple. Therefore, this hyperplane cannot pass through $\text{conv}(V)$, so all members of $\text{conv}(V)$ are on the same side of the hyperplane, implying the triples (a, b, c) for all $a \in \text{conv}(V)$. □

One (naive) way to use this fact is as follows. First, sort the full set X^n by distance to each member of V. Next, calculate an approximate convex hull using the first method. Finally, iterate over every pair (b, c) where all members of V are closer to b than to c and remove points from our estimated hull which are closer to c.

This second method will work much better than the first, but the approach described can surely be refined. For example, since the false positives of our estimate for V from the first method all lie near the boundary, they are all in the convex hull estimate for the subset of V corresponding to the nearest facet of $\text{conv}(V)$. This means that any point which is not in one of these subsets will always be in $\text{conv}(V)$ and need not be tested. In fact, if our aim is only to test a dimensionality estimate then the presence of such a point means that we can answer the dimensionality question immediately and not test *any* point.

Additionally, there is probably some hyperplane which approximates each facet of the hull very well. It seems worth some thought to find a way to identify the pair (b, c) which define the "best" hyperplane for each facet, so we only need one test per facet.

3.3.2 kNN Identification

Identifying the k-nearest neighbors of each member of X^n is of interest for a few reasons. First, sometimes an embedding is wanted only to answer similarity search queries, for which the kNN are the answer. In this case, it isn't necessary to produce an embedding when we can answer the query directly. It is also possible to produce embeddings directly from the kNN, and convergence proofs guaranteeing low embedding error exist for these methods (e.g. Local Ordinal Embedding, Terada and von Luxburg [2014]).

The simplest way to identify the kNN of each point in a set is the brute force approach: simply use a $O(n)$ select algorithm, such as the `SELECT` algorithm of Cormen et al. [2001], on each member of X^n. This will use $O(n^2)$ triples, and when the minimum embedding dimensionality d^* is within a constant factor of n this may be the best we can do (asymptotically). For the remainder of this section, let us assume that d^* is known and that $d^* \ll n$. Can we exploit that fact to do any better?

One promising strategy is to identify points in the "neighborhood" of each point, which will serve as candidate nearest-neighbors. Ideally, we will have $O(k)$ candidates for each point, at least on average. We can then run our select algorithm on these candidates. For example, consider using the FRFT algorithm to select $d^* + 1$ vertices V from which the entire set is ranked, and try to define our neighborhood in terms of those rankings. In most (all?) cases, the first $d^* + 1$ vertices will lie on or near $\text{conv}(X^n)$, and a substantial portion of X^n will lie in $\text{conv}(V)$. This approach will use a total of $O(d^* n \log n + nk)$ triples.

How might we define the neighborhood of a point in a way that can exploit the anchors' rankings of the points? It turns out to be helpful to define some notion of a point lying "between" two other points, in arbitrary dimensionality. We can then say that two points are in each others' neighborhoods when there is no other point "between" them. We would like our definition to have a few useful properties.

1. If some point y is between x and z then we would like it to be closer to z than is x, and closer to x than is z. That is, we want y to be inside the intersection of balls centered on x and z, both with radius δ_{xz}. This makes y a better candidate nearest-neighbor for x than is z.

2. Second, intuitively speaking, we would like some kind of guarantee that we have found all the neighbors in a given "direction," so when we're looking for candidates for x we don't pick up too many extra points in the same direction. This is especially important when some region of the ball containing the true kNN for x has much lower density than the rest of the ball.

3. Finally, we want to find all the points between x and z using only our $d + 1$ rankings for our net members V.

I propose the definition that for three points $x, y, z \in [n]$, point y is "between" points x and z whenever its rank is between the ranks of x and z for all the members of V. Geometrically, this places it in the

intersection of spherical shells (the max-rank ball minus the min-rank ball centered on a given member of V) with x and z on its boundaries, and containing $\text{conv}(\{x, z\})$ as a subset. This definition needs a rigorous mathematical analysis. Some spot checking in low-dimensional spaces shows that a point's nearest neighbor satisfies this property except in extreme cases.

We can use this definition in a few ways. First, we can simply search the candidates in a point's neighborhood for its nearest-neighbor, then remove the selected point and update the neighborhood and repeat until k neighbors are found. Early testing using this method has produced high-precision (> 0.99) kNN estimates for all points in a few synthetic datasets. If $O(k)$ candidates are considered, then the method will use something like $O(k \log k)$ comparisons for each kNN query.

Another approach is to choose some maximum number of candidates to find, such as a constant multiple of k, and then proceed as follows. We first add all points z with no points between x and z. We then add all points z with at most one point between x and z, and then at most two points, and so on until the desired number of candidates is found. The effect is something like a "referral system," in which a point z suggests points y which "it knows" are closer to x than is z. I expect this method to have the most trouble when different "sides" of the ball containing the kNN of x have very different densities. This may add extra candidates on the low density side while not finding enough candidates on the high density side. In any case, once the pool of candidates is finalized we would run the `SELECT` algorithm on all candidates, using just $O(k)$ comparisons for each kNN query.

3.4 Text Similarity

Our initial interest in ordinal embedding was to explore applications to semantic matching of text fragments. If time permits after the success of the preceding tasks, I would like to return to that task.

Similarity Search. Similarity search is an important task for many large-scale corpora used in industry. A common pipeline will aim to speed up similarity searching of, for instance, cosine similarity between TF-IDF document vectors, by first producing a dense, low-dimensional representation of the documents from the sparse similarity scores (by, e.g., LSI), a searchable index of that representation. Such an index is typically some hierarchical data structure that can be navigated to find, near the leaves, a set of points with small mutual distances [Clarkson, 2006, Sankaranarayanan et al., 2007]. Approximation algorithms which are sublinear in the number of points are also available [Muja and Lowe, 2009, Har-Peled et al., 2012]. These algorithms generally guarantee that the identified neighbors are close to the query point, but make mistakes when one of the true kNN is across the boundary of the particular subspaces considered by the algorithm.

There are two possible tasks we could perform here. First, we could seek a better dense representation of an arbitrary similarity function. For example, we could find an ordinal embedding routine which did a better job of preserving the cosine similarity scores than does LSI. This would improve accuracy of the final kNN lists by better preserving the order of neighbors.

Alternatively, we could explore similarity search algorithms which work purely on triples rather than relying on an embedding. Our hope would be to both improve the accuracy (as an embedding would) while also providing a faster similarity search. These algorithms would be a natural extension of the geometric work from Section 3.3.2.

Feature Generation. We could also apply ordinal embedding as producing latent features for some downstream machine learning task. Word embeddings, while not currently built using ordinal embeddings, have proven useful for a variety of Natural Language Processing tasks. It would be interesting to explore embedding text documents or images for Information Retrieval (e.g. as Learning to Rank features), embedding words or text fragments for Natural Language Processing, embedding sentence translation pairs for Machine Translation, and so on.

3.5 Recommendation Systems

A final but significant area of interest is recommendation systems, particularly for music recommendation. After a successful internship with Spotify, I plan to complete a paper based on that work and propose to integrate my main ordinal embedding work with it in a followup paper.

4 Timeline

- **Fall, 2017:** (1) Continue developing my research with Spotify into a paper for SIGIR on using Information Retrieval methods for music recommendation. (2) Develop large-scale ordinal embedding methods, focusing firstly on efficiently finding kNN graphs for large-scale corpora and secondly on tuning active learning and embedding methods for the common case that some triple comparisons cannot be answered from available data. Aim for a paper for AISTATS, ICML, KDD, or the like.
- **Spring 2018:** Unify the work from Fall semester to find large-scale ordinal embeddings based on triples inferred from user interaction with music to improve recommendation performance.

5 Conclusion

I believe that methods in Ordinal Geometry are on the verge of a major breakthrough that will increase its utility for large-scale, high-dimensional datasets. There is accelerating interest both in theoretical aspects and the practical matter of recovering an embedding from a set of triples. I hope that the work described here will help to further advance this field.

References

Arvind Agarwal, Jeff M. Phillips, and Suresh Venkatasubramanian. Universal multi-dimensional scaling. *SIGKDD*, 2010. doi: 10.1145/1835804.1835948.

Sameer Agarwal, Josh Wills, Lawrence Cayton, Gert Lanckriet, David Kriegman, and Serge Belongie. Generalized non-metric multidimensional scaling. *AISTATS*, 2007.

Jesse Anderton, Pavel Metrikov, Virgil Pavlu, and Javed Aslam. Measuring human-perceived similarity in heterogenous collections. *Manuscript*, 2014.

Jesse Anderton, Virgil Pavlu, and Javed Aslam. Revealing the basis: Ordinal embedding through geometry. *Manuscript*, 2016a.

Jesse Anderton, Virgil Pavlu, and Javed Aslam. Triple selection for ordinal embedding. *Manuscript*, 2016b.

Ery Arias-Castro. Some theory for ordinal embedding. *arXiv.org*, January 2015.

Ingwer Borg and Patrick J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, New York, NY, 2005. ISBN 978-0-387-28981-6. doi: 10.1007/0-387-28981-X_20. URL `http://dx.doi.org/10.1007/0-387-28981-X_20`.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *JMLR*, 2010.

K L Clarkson. Nearest-neighbor searching and metric space dimensions. *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, 2006.

I D Coope. Reliable computation of the points of intersection of n spheres in R^n. *ANZIAM Journal*, 42(0): 461–477, December 2000.

Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001. ISBN 0070131511.

Mihai Cucuringu and Joseph Woodworth. Point Localization and Density Estimation from Ordinal kNN graphs using Synchronization. *arXiv.org*, April 2015a.

Mihai Cucuringu and Joseph Woodworth. Ordinal embedding of unweighted kNN graphs via synchronization. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2015b.

Jon Dattorro. *Convex Optimization and Euclidean Distance Geometry*. Meboo Publishing, 2016.

Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. *ICML*, 2007.

Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

S Har-Peled, P Indyk, and R Motwani. Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality. *Theory of computing*, 2012.

T B Hashimoto, Y Sun, and T S Jaakkola. Metric recovery from directed unweighted graphs. *AISTATS*, 2015.

Lalit Jain, Kevin Jamieson, and Robert Nowak. Finite Sample Prediction and Recovery Bounds for Ordinal Embedding. *arXiv.org*, June 2016.

Prateek Jain, Brian Kulis, Jason V. Davis, and Inderjit S. Dhillon. Metric and kernel learning using a linear transformation. *JMLR*, 2012.

Kevin G Jamieson and Robert D Nowak. Low-dimensional embedding using adaptively selected ordinal data. *49th Annual Allerton Conference on Communication, Control, and Computing*, 2011. doi: 10.1109/Allerton.2011.6120287.

M Kleindessner and U von Luxburg. Uniqueness of Ordinal Embedding. *COLT*, 2014.

M Kleindessner and Ulrike von Luxburg. Dimensionality estimation without distances. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, 2015.

Matthäus Kleindessner and Ulrike von Luxburg. Kernel functions based on triplet similarity comparisons. *arXiv.org*, July 2016a.

Matthäus Kleindessner and Ulrike von Luxburg. Lens depth function and k-relative neighborhood graph: versatile tools for ordinal data analysis. *arXiv.org*, February 2016b.

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

J B Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, 1964b.

Brian Kulis, Mátyás A. Sustik, and Inderjit S. Dhillon. Low-rank kernel learning with bregman matrix divergences. *JMLR*, 2009.

Ke Li and Jitendra Malik. Fast k-Nearest Neighbour Search via Dynamic Continuous Indexing. *arXiv.org*, December 2015.

Ke Li and Jitendra Malik. Fast k-Nearest Neighbour Search via Prioritized DCI. *arXiv.org*, March 2017.

Leo Liberti, Carlile Lavor, Nelson Maculan, and Antonio Mucherino. Euclidean distance geometry and applications. *arXiv.org*, May 2012.

Brian McFee and Gert Lanckriet. Learning Multi-modal Similarity. *JMLR*, 12, February 2011.

Alistair Moffat, Gary Eddy, and Ola Petersson. Splaysort: Fast, Versatile, Practical. *Software Practice and Experience*, 26(7):781–797, July 1996.

M Muja and D G Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. *VISAPP (1)*, 2009.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

Jagan Sankaranarayanan, Hanan Samet, and Amitabh Varshney. A fast all nearest neighbor algorithm for applications involving large point-clouds. *Computers and Graphics*, 31(2), April 2007.

Roger N Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. I. *Psychometrika*, 27(2):125–140, 1962a.

Roger N Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. II. *Psychometrika*, 27(3):219–246, 1962b.

M J Sippl and H A Scheraga. Solution of the embedding problem and decomposition of symmetric matrices. *Proceedings of the National Academy of Sciences*, 82(8):2197–2201, April 1985.

M J Sippl and H A Scheraga. Cayley-Menger coordinates. *Proceedings of the National Academy of Sciences*, 83(8):2283–2287, April 1986.

Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Tauman Kalai. Adaptively Learning the Crowd Kernel. *arXiv.org*, May 2011.

Yoshikazu Terada and Ulrike von Luxburg. Local ordinal embedding. *ICML*, 2014.

Laurens Van der Maaten and Kilian Weinberger. Stochastic triplet embedding. *2012 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2012.

Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *NIPS*, 2006.

Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning, with application to clustering with side-information. *NIPS*, 2003.

Emine Yilmaz, Javed Alexander Aslam, and Stephen E Robertson. A new rank correlation coefficient for information retrieval. *SIGIR*, 2008.