Low-Dimensional Embedding using Adaptively Selected Ordinal Data

Kevin G. Jamieson
University of Wisconsin
Madison, WI 53706, USA
kgjamieson@wisc.edu

Robert D. Nowak
University of Wisconsin
Madison, WI 53706, USA
nowak@engr.wisc.edu

***Abstract*—Low-dimensional embedding based on non-metric data (e.g., non-metric multidimensional scaling) is a problem that arises in many applications, especially those involving human subjects. This paper investigates the problem of learning an embedding of n objects into d-dimensional Euclidean space that is consistent with pairwise comparisons of the type "object a is closer to object b than c." While there are $O(n^3)$ such comparisons, experimental studies suggest that relatively few are necessary to uniquely determine the embedding up to the constraints imposed by all possible pairwise comparisons (i.e., the problem is typically over-constrained). This paper is concerned with quantifying the minimum number of pairwise comparisons necessary to uniquely determine an embedding up to all possible comparisons. The comparison constraints stipulate that, with respect to each object, the other objects are ranked relative to their proximity. We prove that at least $\Omega(dn \log n)$ pairwise comparisons are needed to determine the embedding of all n objects. The lower bounds cannot be achieved by using randomly chosen pairwise comparisons. We propose an algorithm that exploits the low-dimensional geometry in order to accurately embed objects based on relatively small number of sequentially selected pairwise comparisons and demonstrate its performance with experiments.**

I. Introduction

This paper studies the problem of learning a low-dimensional embedding from ordinal data. Consider a set of n points $\{x_1, \ldots, x_n\}$ in $\mathbb{R}^d$. The locations of the points are unknown to us, but assume we are given the set of constraints of the form "object x_k is closer to (or further from) x_i than x_j" for all distinct $i, j, k \in \{1, \ldots, n\}$. The goal is to identify an embedding into $\mathbb{R}^d$ consistent with these constraints. This is a classic problem that has been addressed using a technique known as non-metric multidimensional scaling (non-metric MDS).

Here we consider a new variation on this problem. Constraints of the form above are often collected from human subjects. Each constraint is associated with a binary variable, the answer to the *comparison query* "Is object x_k closer to x_i than x_j?" People are better at providing this sort of information as opposed to giving more fine-grained numerical judgments or distances [1]. There are on the order of n^3 constraints. Collecting ordinal data of this type from people is time-consuming and costly, and quickly becomes prohibitive as n grows. So it is of interest to consider whether it is necessary to obtain the complete set of data. Since the points are assumed to exist in $\mathbb{R}^d$, it is reasonable to conjecture that if the embedding dimension is low, i.e., $d \ll n$, then there may be a high degree of redundancy in these constraints. If this conjecture is correct, then it should be possible to identify a *consistent embedding* (i.e., consistent will **all** the constraints) from a small subset of the constraints.

We lower bound the minimum number of constraints needed to determine a consistent embedding by $dn \log n$, far fewer than the total number. We conjecture that this lower bound is tight and propose a sequential procedure for adaptively selecting comparison queries. A comparison query is made if and only if the answer to the query (i.e., the corresponding constraint) is ambiguous given the answers to all previously selected queries. Ambiguity can be tested by solving an optimization problem that is, in general, non-convex but is observed to be well-behaved in practice (see Section IV-D). Analysis of the procedure and numerical experiments support the conjecture that on the order of $dn \log n$ queries/constraints determine the embedding. Furthermore, we show that if queries are selected uniformly at random, then almost all the queries must be requested in order to determine an embedding consistent with all the constraints.

This paper is laid out as follows. Section II reviews related work of non-metric multidimensional scaling and previous advancements made in the active discovery of ordinal information under geometrical constraints. In Section III we discuss the geometrical interpretation of our constraints and use this insight to construct bounds for the query complexity of this problem. In Section IV we propose two algorithms that attempt to determine the embedding by asking as few redundant queries as possible. Finally, in Sections V and VI we present our numerical studies and analyze the results.

II. Related work

Non-metric multidimensional scaling (MDS) was designed to provide researchers with a graphical or spatial representation of the human-perceived relationships between a set of arbitrary objects [2]. In addition to the pairwise comparisons of the form $||x_i - x_j|| < ||x_j - x_k||$ for all triples (i, j, k), non-metric MDS also forces constraints of the type

$$||x_i - x_j|| < ||x_k - x_l|| \tag{1}$$

for all quadruples $(i, j, k, l) \in \{1, \ldots, n\}$. These additional queries make the total number of queries grow like n^4 which is often prohibitively large for even small values of n. Consequently these additional constraints are often omitted in practice [3], [4]. Also note that because our query-ambiguity test alluded to above is a special case of non-metric MDS, it follows that non-metric MDS is also non-convex and these additional queries can make the already difficult optimization even harder. However, these issues are not the only problems; they can also, at times, be difficult to answer accurately because a query "is the distance between objects i and j less than the distance between objects k and l?" requires a comparison of the absolute scales of the dissimilarities instead of simply asking which object is closer to another. This difficulty is our primary reason for considering constraints using just three objects.

While pairwise comparisons using triples of objects are very natural and easy to answer, some research suggests that people can find answering these kind of queries extremely tedious and boring. Presumably, this could lead to erroneous answers after extended sessions of querying a user [5]. Some researchers have suggested that perhaps only a sparse subset of these inequalities are actually required, greatly reducing the load on the human subject [3], [6]. Early work using just a random subset of these kinds of queries by Johnson supports this hypothesis [7]. While researchers in the past have proposed algorithms to find an embedding given a fixed number of answers to queries, we are unaware of any research that attempts to characterize the number of queries that must *necessarily* be made to uniquely determine an embedding. We provide partial answers to this question and propose an algorithm that we conjecture to be optimal in the sense that it asks within a constant factor of the minimum number of necessary queries to uniquely determine an embedding consistent with all the constraints.

Prior to this point, we have assumed that the constraints we are querying for are consistent with an embedding of a known dimension. However, [4] assumes that labels to queries are the result of a consensus from a number of individuals, or a crowd. This perspective allows one to consider the problem from a probabilistic point of view so that one can speak of requesting the comparison that would provide the greatest potential information gain. While [4] presents some results for empirical datasets, few guarantees were made about the quality of the embedding and no guidance was given to how many queries were "enough" or sufficient to achieve an embedding of satisfactory quality.

In previous work we characterized the query complexity of a very related problem that this paper extends [8]. Given a fixed embedding $x_1, \ldots, x_k$ of k objects in $\mathbb{R}^d$ (i.e. the locations of the k objects are known exactly) and just one object placed in an unknown location in the same space, we show that the ranking of the objects relative to their proximity to this new object can be discovered with just $\Theta(d \log k)$ queries on average, depending on the particular placement of the new object. Note that the embeddings we consider in this paper are determined up to an equivalence class by the correct ranking of the objects with respect to each object. This analysis immediately yields a lower bound on the query complexity of finding an embedding.

III. The geometry of an embedding

Consider an embedding of n points in $\mathbb{R}^d$. For any triple (i, j, k), we have that either $||x_i - x_k|| < ||x_k - x_j||$ or its opposite are true (we assume ties are not allowed). We wish to learn an embedding $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ that satisfies all of these constraints. If we concatenate each point $x_i \in \mathbb{R}^d$ into a column vector $x = (x_1^T, \ldots, x_n^T)^T$, we see that an embedding of n points in d dimensions can be represented by a single vector living in $\mathbb{R}^{nd}$. If for every triple (i, j, k) we define the region $r_{i,j,k} = \{x \in \mathbb{R}^{nd} : ||x_i - x_k|| < ||x_k - x_j||\}$, then the query: "is object x_k closer to x_i or x_j" is equivalent to asking if $x \in r_{i,j,k}$. We call this pairwise comparison query a membership query. All possible intersections of these regions (and their complements) partition $\mathbb{R}^{nd}$ into many nd-dimensional regions which we will call nd-cells to distinguish them from the regions of the form of $r_{i,j,k}$. Because every point in an nd-cell agrees with all of the same constraints, we call any two embeddings in the same nd-cell *equivalent*. From this perspective, we see that we are trying to locate a point in one of these nd-cells bounded by surfaces passing through $\mathbb{R}^{nd}$ that are induced by the membership queries $r_{i,j,k}$. Before considering this problem directly, we would like to provide some intuition about the space of embeddings.

A. A lower bound on the query complexity

In this section we state a lower bound on the number of membership queries that are necessary to define an embedding of n objects in d dimensions such that all the constraints are satisfied. Our strategy is to add one object at a time to the embedding and lower bound how quickly the number of embeddings grows.

Recall that we assumed the existence of a fixed embedding of n points in d dimensions that generated the $n\binom{n-1}{2}$ constraints. Suppose we somehow had the exact locations of $k < n$ objects and we would simply like to add the $(k+1)$st point to the embedding. At the very least, we must determine the order of the distances from x_{k+1} to all the other x_i's for $i = 1, \ldots, k$. That is, we must determine some permutation σ of the k indices such that we can write

$$||x_{k+1} - x_{\sigma(1)}|| < ||x_{k+1} - x_{\sigma(2)}|| < \cdots < ||x_{k+1} - x_{\sigma(k)}||. \tag{2}$$

Because the points $x_1, \ldots, x_k$ are embedded in d-dimensions, there are far fewer than $k!$ possibilities for σ. In fact, if all the points are in general position, the number of possibilities for σ is known exactly. Before presenting this result it is instructive to consider the geometric relationship between the rankings of (2) and $\mathbb{R}^d$.

Again, recall that the x_i's for $i = 1, \ldots, k$ are assumed fixed, only x_{k+1} is unknown. To determine the number of possibilities for σ of (2) we notice that every ranking can be uniquely defined by a set of pairwise comparisons:

$||x_{k+1} - x_i|| < ||x_{k+1} - x_j||$ for some set of indices $\{i, j\}$. This pairwise comparison has a geometrical interpretation: it says that object x_{k+1} resides in the halfspace defined by the hyperplane that bisects and lies orthogonal to the line connecting objects x_i and x_j. There is a hyperplane associated with each pair of objects so as a result, the $\binom{k}{2}$ hyperplanes partition the d-dimensional space into many d-dimensional regions we call d-cells. If we orient each of the hyperplanes such that a point in the space either lies above, on or below each hyperplane, then all points in a d-cell are above, on, or below all of the same hyperplanes. While each halfspace is a partial ordering of just two objects, each d-cell is associated with a total ordering over the k objects describing the relative proximity of the $(n+1)$th object to the other objects. Because there is a one-to-one relation between distinct rankings and the number of d-cells, we can use this geometrical interpretation to characterize the number of distinct possibilities of σ of (2). For further discussion see [8] or [9].

Lemma 1. *[9] Let $x_1, \ldots, x_k$ be a fixed embedding of k objects in general position in $\mathbb{R}^d$. Let $C(k, d)$ denote the number of d-cells formed by the arrangement of hyperplanes induced by the $\binom{k}{2}$ pairs of objects. $C(k, d)$ satisfies the recursion*

$$C(k, d) = C(k-1, d) + (k-1)C(k-1, d-1)\ , \qquad (3)$$

where $C(1, d) = 1$ and $C(k, 0) = 1$.

In the hyperplane arrangement induced by the k objects in d dimensions, each hyperplane is intersected by every other and is partitioned into $C(k-1, d-1)$ subsets or $(d-1)$-cells. The recursion, above, arises by considering the addition of one object at a time. Using this lemma in a straightforward fashion, we can show the following corollary (see [8]).

Corollary 1. *There exist positive real numbers c_1 and c_2 such that*

$$c_1 \frac{k^{2d}}{2^d d!} < C(k, d) < c_2 \frac{k^{2d}}{2^d d!}$$

for $k > d + 1$. If $k \leq d + 1$ then $C(k, d) = k!$. For k sufficiently large, $c_1 = 1$ and $c_2 = 2$ suffice.

We are now ready to state a lower-bound on the query complexity of finding an embedding.

Theorem 1. *The number of membership queries $\{x \in r_{i,j,k}\}_{i,j,k \leq n}$ required to determine an embedding of n objects in d dimensions that satisfies all of the constraints is $\Omega(dn \log n)$.*

Proof: Using the help of an oracle, who will not only supply us the answers to membership queries but also additional side information, we will construct an embedding adding one object at a time. We will lower bound the number of bits of information that we will need to collect from the oracle and then use this as a lower bound for the number of queries necessary since a query provides at most one bit of information.

Recall that we assume the existence of a fixed embedding $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ that generated the constraints. We begin by asking the oracle for the exact locations of the first two objects x_1 and x_2. Given the fixed positions of the first two objects, we find which d-cell (a single halfspace in this case) the third object resides in, tell the oracle, and then ask the oracle to provide the exact location of the third object. That is, before getting the exact location of an object from the oracle, we must tell the oracle which d-cell the object is in. After k objects have been embedded this way, we find the d-cell that the $(k+1)$th object resides in, and then tell the oracle who returns the exact location of this object. Because the oracle is providing to us the exact locations of the objects, any queries that are inferred, due to them being unambiguous, are consistent with *any* embedding that satisfies all the $n\binom{n-1}{2}$ constraints; even those that have not been considered by this sequential procedure yet. This subtle point will be considered again in Section IV-B when we do not have access to the exact locations of the objects. There are $C(k, d)$ possible rankings of the k objects that we must discriminate between to tell which d-cell the $(k+1)$th object is in. Therefore, we must provide at least $\Omega(d \log k)$ bits of information to the oracle. The lower bound follows from summing the number of bits to add all of the objects to the embedding sequentially. ■

Based on how the above lower bound was constructed, it is not clear how tight the bound is because the oracle provided the *exact* location of the current object: information which is clearly sufficient but unlikely to be absolutely necessary. However, as alluded to before in Section II, theoretical and empirical evidence suggests that as the number of objects to be embedded grows, the amount of "wiggle" possible in each point of the embedding decreases rapidly to zero [10]. Intuitively, as the number of constraints grows with the number of objects embedded, the embedding acts more and more as if it were constrained with metric constraints. We will revisit this idea in the Section VI.

B. Counting the number of embeddings

Given the lower bound of the last section, it is natural to wonder how tight it is. If we could *upper bound* the number of embeddings and look at the log of this number, this would tell us how many bits it takes to encode an embedding. But even if this number matched the lower bound, this would still not tell us if we could *achieve* the lower bound because the possible membership queries we have at our disposal may not be informative enough. It turns out that to merely upper bound the number of embeddings is a challenge. For the special case when $d = 1$, we will demonstrate an upper bound that matches the lower bound of above. We will then consider the general case for $d > 1$.

Consider the membership query $x \in r_{i,j,k} = \{x \in \mathbb{R}^{nd} : ||x_i - x_k|| < ||x_k - x_j||\}$ for some (i, j, k) triple. This can be simplified to a slightly more interpretable form:

$$(x_i - x_k)^T (2x_j - x_i - x_k) < 0\,. \qquad (4)$$

In $\mathbb{R}^{nd}$ this corresponds to a cone in $2d$ dimensions (in a rotated coordinate space) whose apex is the origin and is constant in the remaining $n-2d$ dimensions. For $d=1$, $x_i \in \mathbb{R}$ for all $i=1,\ldots,n$ so $x \in \mathbb{R}^n$ and (4) degenerates into a very simple geometric relation. If $(x_i-x_k)(2x_j-x_i-x_k) < 0$ then $(x_i-x_k) < 0$ and $(2x_j-x_i-x_k) > 0$, or $(x_i-x_k) > 0$ and $(2x_j-x_i-x_k) < 0$. Because $(x_i-x_k)=0$ and $(2x_j-x_i-x_k)=0$ simply define two hyperplanes, an embedding consistent with all $n\binom{n-1}{2}$ labels is contained within an n-cell bounded by hyperplanes. Because each query induces at most two hyperplanes (actually just 1, on average, because of the queries sharing hyperplanes) it follows that $\mathbb{R}^n$ is partitioned by $O(n^3)$ hyperplanes corresponding to the $n\binom{n-1}{2}$ queries. It is well-known that N hyperplanes partition $\mathbb{R}^p$ into no more than $O(N^p)$ p-dimensional regions [11]. It then follows that for $d=1$ and any n there exists some constant c independent of n such that there are at most cn^{3n} embeddings because each n dimensional region corresponds to at most one embedding.

Unfortunately, for $d > 1$ we have yet to discover a non-trivial upper bound for the number of embeddings. The difficulty arises in the nonlinear nature of the polynomials defining the queries. A common tactic in counting the number of intersections of sets with nonlinear boundaries is to project to a higher dimensional space where the boundaries are linear [12]. Let $\phi : \mathbb{R}^{nd} \rightarrow \mathbb{R}^{n(n+2d+1)/2}$ be a map from $x=(x_1,\ldots,x_n)$ to $(x_1,\ldots,x_n,x_1^Tx_1,\ldots,x_n^Tx_n,x_1^Tx_2,\ldots,x_{n-1}^Tx_n)$. Now each query of the form (4) is a hyperplane in $\mathbb{R}^{n(n+2d+1)/2}$. Using the same technique as above for $d=1$, we can upper bound the total number of embeddings for all $d \geq 1$ by some constant times $n^{3n(n+2d+1)/2}$. Unfortunately, this bound is quite trivial for the following reason: for an embedding to satisfy all the constraints, we would merely need to discover how each of the n objects rank the other $(n-1)$ objects with respect to the distance from themselves. Using binary sort, each ranking of $n-1$ objects can be discovered using $(n-1)\log_2(n-1)$ queries (see Section IV-A.) Because there are n rankings to discover, this implies that it takes at most $n(n-1)\log_2(n-1)$ bits to describe an embedding. Observing that $\log_2(n^{3n(n+2d+1)/2)}) > n(n-1)\log_2(n-1)$, we see that if we use binary sort, we get a *better* bound than if we use this linearizing technique that takes d into account.

This bound is loose because when we linearize the space and count the number of equivalent regions in the projected high dimensional space, this does not take into account the fact that all possible solutions, $x \in \mathbb{R}^{nd}$, actually lie on an nd-dimensional manifold $\{\phi(x) : x \in \mathbb{R}^{nd}\}$ in this high dimensional space. The true number of embeddings is not the total number of these regions in the high-dimensional space, but just those that are intersected by the manifold $\{\phi(x) : x \in \mathbb{R}^{nd}\}$. Fortunately, our experiments and analysis (see Sections V and VI) suggest that the true rate of growth for the number of embeddings is much smaller than that of predicted by binary sort and *does* depend on the dimension d.

C. The inefficiency of randomly selected queries

Before we discuss different adaptive methods of selecting queries, it is natural to wonder if such complicated schemes are really necessary; is it sufficient to simply select queries uniformly at random to find a solution? In this section we show that if membership queries are selected in a random fashion, $\Omega(n^3)$ queries must be requested to uniquely determine an nd-cell and thus, an embedding. In fact, we actually show that to solve a problem using extra side information would require this many queries and because that information could have always been ignored, to solve the problem without the side information is *at least* as hard. Our strategy is to add a single object to the embedding one at a time and show that if there are k objects already embedded, it requires $\Omega(k^2)$ queries to add the $(k+1)$th object.

We assume that queries are selected independently such that after selecting a subset of the queries, they are exchangeable in the sense that we can reorder them any way we like and it does not affect which nd-cell they define. Enumerate the objects so that they are labeled $1,\ldots,n$. Then, order the randomly selected queries such that for any query defined over the triple (i,j,k) in the list, all queries that are ordered before it use objects whose indices are less than or equal to $\max\{i,j,k\}$. In other words, we would like to reorder the selected queries such that it appears as if we are adding one object at a time like we constructed the lower bound of above. Again, suppose we somehow had the exact locations of $k < n$ objects and we would simply like to add the $(k+1)$th point to the embedding. At the very least, we must determine the order of the distances from x_{k+1} to all the other x_i's for $i=1,\ldots,k$ as in (2). Recall from Section III-A that if the k objects are fixed, each possible ranking over the k objects has a one-to-one correspondence with a d-cell that is bounded by hyperplanes corresponding to the queries. If m queries were chosen uniformly at random from the possible $\binom{k}{2}$, the answers to m queries narrows the set of possible rankings to a d-cell in $\mathbb{R}^d$. This d-cell may consist of one or more of the d-cells in the partition induced by all $\binom{k}{2}$ hyperplanes. If it contains more than one of the partition cells, then the underlying ranking is ambiguous.

Lemma 2. *Let $N=\binom{k}{2}$. Suppose m membership queries $\{x \in r_{i,k+1,j}\}_{i,j<k}$ are chosen uniformly at random without replacement from the possible $\binom{k}{2}$. Then for all positive integers $N \geq m \geq d$ the probability that the m queries yield a unique ranking is $\binom{m}{d}/\binom{N}{d} \leq (\frac{em}{N})^d$.*

Proof. No fewer than d hyperplanes bound each d-cell in the partition of $\mathbb{R}^d$ induced by all possible queries. The probability of selecting d specific queries in a random draw of m is equal to

$$\binom{N-d}{m-d}\Big/\binom{N}{m} = \binom{m}{d}\Big/\binom{N}{d} \leq \frac{m^d}{d!}\frac{d^d}{N^d} \leq \left(\frac{m}{N}\right)^d\frac{d^d}{d!} \leq \left(\frac{em}{N}\right)^d. \quad \blacksquare$$

Note that $\binom{m}{d}/\binom{N}{d} < 1/2$ unless $m=\Omega(k^2)$. Therefore, if the queries are randomly chosen, then we will need to ask

almost all queries to guarantee that the inferred ranking over the first k objects is probably correct. The proof of the next theorem is shown by repeated application of the above result using the same line of reasoning as the proof of Theorem 1.

Theorem 2. *Given the existence of an embedding of n objects in d dimensions, if m membership queries $\{x \in r_{i,j,k}\}_{i,j,k \leq n}$ are chosen uniformly at random without replacement from the possible $n\binom{n-1}{2}$, then to uniquely determine the nd-cell and thus an embedding that satisfies all of the constraints with probability greater than $1/2$, $m = \Omega(n^3)$.*

IV. Query selection algorithms

In this section we propose query selection algorithms that attempt to satisfy all of the $n\binom{n-1}{2}$ constraints by only requesting a small subset of them. First, we review classical binary sort in Section IV-A because it is implemented in all of the algorithms and its performance guarantees should be clearly stated. We then propose a sequential algorithm in Section IV-B that adds one object at a time to the embedding and asks for queries only if they cannot be inferred using all the known constraints up to that time. Finally, we present a non-metric (or generalized) version of landmark MDS in Section IV-C that was originally designed to reduce the amount of data collection for metric MDS [13]. Both algorithms assume the existence of a subroutine that, given any set of constraints that are consistent, will output whether there exists an embedding that is consistent with all of the constraints, or not. In addition, we assume that if such an embedding exists, we can request it from the subroutine. After presenting the algorithms that utilize this subroutine, we will consider its implementation in Section IV-D.

A. Binary Sort

Binary sort is a simple, adaptive algorithm that finds an arbitrary ranking over n objects using no more than $n \log_2 n$ pairwise comparisons. Because there are $n! = \Theta(n^n)$ possible rankings, this algorithm is optimal in terms of the number of requested pairwise comparisons if no additional structure about the objects is assumed. The algorithm works as follows: given a ranking of k objects, there are $(k+1)$ positions that the $(k+1)$th object can be put into; and because there is an ordering over the objects, binary search can be used to find the correct position in no more than $\log_2(k+1)$ queries. By induction, no more than $n \log_2 n$ pairwise comparisons are needed to rank n objects.

Consider finding an embedding of n objects in d dimensions. An embedding is only unique up to the constraints $||x_i - x_j|| < ||x_j - x_k||$ for all triples (i, j, k). This is equivalent to having each object rank the other $n-1$ objects relative to their distance away from themselves, like in (2). By the above argument, to find n rankings of $(n-1)$ objects, no more than $n(n-1)\log_2(n-1)$ queries must be requested.

B. A sequential query selection algorithm

Here we introduce an algorithm to find an embedding of n objects in d dimensions that sequentially chooses membership queries in an adaptive way in hopes of drastically reducing the number of requested queries. But first, we consider a naïve approach to point out some potential pitfalls of a sequential algorithm.

Recall the sequential process used in the proof of the lower bound of Theorem 1. We added one object at a time by finding the d-cell the object was located in, and then requested the exact location of the object within that d-cell from the oracle. It is natural to wonder if this *exact* location is really necessary and if picking an *arbitrary* point in the d-cell would suffice. Unfortunately, as Borg illustrates in a non-pathological example of an embedding, this arbitrary placement of objects can potentially close off possibilities for the locations of future objects which would make it impossible to satisfy all the future constraints [2, Chapter 2]. Intuitively, if you are not careful with how you decide the coordinates of the objects, it is very easy to walk yourself into a corner with no escape. What we should take from this example is that we must allow for the objects to have maximum flexibility while obeying the constraints if we would like to guarantee that all the constraints, in the end, are satisfied.

The underlying principle behind our proposed algorithm is very simple and has enjoyed great success in other active learning settings [14], [8]. The sequential algorithm for requesting queries begins by enumerating the objects and considers them one at a time. The algorithm will proceed through the queries using binary sort and request the membership query if only if it cannot be determined using the previously requested constraints. That is, if Q is the set of constraints corresponding to the membership queries we have requested up to the consideration of some new query $\{x \in r_{i,j,k}\}$, we will run our constraint-validating subroutine twice: once with $Q \cup \{x \in r_{i,j,k}\}$ and the second time with $Q \cup \{x \in r^c_{i,j,k}\}$. If the subroutine confirms that both set of constraints lead to valid embeddings, then the query in question $Q \cup \{x \in r_{i,j,k}\}$ is said to be *ambiguous*. Otherwise, if only one of the runs of the subroutine confirms a valid embedding, we can infer what the constraint must be and we do not need to request it from the user. This algorithm is presented in Figure 1. Note that despite what is written in the presentation of the algorithm in Figure 1, binary sort is implemented; it is presented this way for clarity.

Given the full set of $n\binom{n-1}{2}$ constraints from the algorithm, we can then run the subroutine to get the full embedding of the n objects in d dimensions.

C. Landmark non-metric MDS (LNM-MDS)

Here we introduce landmark non-metric MDS (LNM-MDS) which can be thought of as a non-metric or generalized version of landmark *metric* MDS [13]. The basic idea behind landmark-based versions of MDS is that instead of collecting data for all pairs or triples of objects, a small number of objects are designated as landmarks. The objects are embedded using only distances (or comparisons, in this paper) relative to the landmarks. For example, the LNM-MDS proposed here only uses comparisons of the form $||x_i - l|| < ||x_j - l||$ or $||l - x_i|| < ||l' - x_i||$, where x_i and x_j are arbitrary objects,

Sequential query selection algorithm

input: n objects in unknown positions in $\mathbb{R}^d$
initialize: $Q = \emptyset$, enumerate objects $x_1, \ldots, x_n$ in uniformly random order

for $k = \{2, \ldots, n\}$
  for $j = \{1, \ldots, k\}$
    for $i = \{1, \ldots, k\}$
      if $\{x \in r_{i,j,k}\}$ is *ambiguous* using only Q,
        ask if $\{x \in r_{i,j,k}\}$;
      else
        infer if $\{x \in r_{i,j,k}\}$ with Q and add it to Q
output: $n\binom{n-1}{2}$ constraints

Fig. 1: Sequential algorithm for selecting queries. See Section IV-B for the definition of an ambiguous query.

but l and l' must be members of a small set of landmarks. If $d \ll n$ and the number of landmarks is large enough, then the intuition is that using these landmarks may be sufficient to describe the same information as if all information was collected between all objects.

For any integer $L \geq 2$, LNM-MDS chooses L objects uniformly at random from the set of n and requests only the queries between the objects so that each landmark has a complete ranking over the other $n-1$ objects and each non-landmark object has a ranking over the L landmarks. This algorithm is motivated by the idea that if the the dimension d is not too big, perhaps the relative proximities to just a small subset of the objects suffices to define the embedding. While these rankings define $L\binom{n-1}{2} + (n-L)\binom{L}{2}$ total pairwise comparisons, we will use binary sort to acquire these rankings which would mean only about $L(n-1)\log_2(n-1) + (n-L)L\log_2(L)$ will be requested. If the number of landmarks is small, this could be a significant savings in the number of requested queries compared to asking for all n rankings over $n-1$ objects, about $n^2 \log_2 n$. While LNM-MDS does not explicitly take advantage of the low-dimensional nature of the embedding, it may implicitly use it to its advantage because a few landmarks may suffice to define the embedding. One of the drawbacks of this algorithm is that to ensure that all the constraints are satisfied, one must check if all the other queries not asked are unambiguous. If landmarks are added one at a time, this could be very computationally demanding.

D. Constraint validation subroutine

This section describes the constraint validation subroutine that determines whether a query is ambiguous or not. This subroutine is, in essence, an algorithm for non-metric MDS that just uses constraints of triples of objects as input. As described in the beginning of Section III, to check if a set of constraints is valid, we must check if there is non-empty intersection of the sets defined by the membership queries $r_{i,j,k}$.

In general, to find a point in $\mathbb{R}^p$ that lies in the intersection of sets is known as a feasibility program [15]. Unfortunately, it is easy to show that the sets defined by the membership queries, or equivalently the constraints of (4), produce non-convex sets which makes the feasibility program non-convex. This implies that what the constraint validation subroutine converges to could be a local minima (if it converges at all) which may erroneously indicate that a set of constraints do *not* correspond to a valid embedding when they really do. Clearly, this could be disastrous to the algorithm because queries may be indicated as unambiguous when they really are ambiguous. Some algorithms for solving non-metric MDS deal with this non-convexity by allowing d to be variable (in contrast to fixed), possibly as large as n, but penalizing the optimization by adding the trace norm of the inner product matrix of the embedding. This encourages low-dimensional (or approximately low-dimensional) embeddings [3]. Because essentially arbitrary constraints can be obeyed if d is allowed to be n, this sort of approach would not constrain the set of solutions and would indicate that almost all the queries are ambiguous. What this means is that solving the non-convex program is unavoidable and the only thing that can be done is to repeat the optimization multiple times, each with a random initialization. If this is done enough times, we can be relatively confident that its results are trust-worthy. Fortunately, in practice, this optimization problem tends to converge to a solution rather easily if it exists. We will return to this issue in the presentation of our numerical results.

The earliest reference of an algorithm that attempts to do the job of the subroutine is credited to Johnson in 1973 who solves the feasibility problem by penalizing any violated inequalities with a quadratic loss function [7]. In the last few decades there have been enormous advances in optimization and we know that a linear loss function using Lagrange multipliers leads to much quicker convergence [15], [16]. To make the optimization problem converge in a reasonable amount of time for the problem sizes we are considering $(3 \leq n \leq 50)$ many known techniques and tricks for non-convex optimization are necessary [16]. Matlab code for our implementation is available on the first author's website [17].

V. Empirical results

In this section we present empirical results regarding how many queries are requested to embed n objects into d dimensions. We compare standard binary sort of Section IV-A, the sequential algorithm of Section IV-B, and LNM-MDS of Section IV-C. In LNM-MDS, recall from Section IV-C that to check if an existing solution given some number of landmarks satisfies all the constraints, we have to check if any of the queries not requested are ambiguous or not. Because we will be adding one landmark at a time, we will give the algorithm the benefit of the doubt in our simulations and end LNM-MDS as soon as it finds an embedding using the fewest number of landmarks with zero violations of all the constraints (even those that it may not know about yet). Clearly, this is a lower bound on its performance. On the other hand, when either binary sort or the sequential algo-

rithm of Figure 1 finishes, it guarantees that all the constraints are satisfied (under the assumption that the subroutine always returns correct results).

Recall from Section IV-D that the constraint-validating subroutine is solving a non-convex problem. It is possible that the subroutine will converge to a local minima, indicating that there does not exist an embedding consistent with the given constraints when, actually, there does. This behavior could potentially lead to the algorithm believeing that a query is unambiguous when, in reality, it is the opposite case and must be requested. For our simulations, we assumed that if the algorithm failed to converge to a consistent embedding after 3 attempts, then a consistent embedding did not exist. Fortunately, in our studies, with the number of restarts set to 3, in each run we observed no more than about a few of these mistakes out of the total of about 3000 considered for $n = 30$. However, this seemingly disastrous problem is actually not much of a problem at all because in practice, the only queries fed to the constraint validation subroutine are those that were ambiguous when they were considered (we do not need to give the algorithm unambiguous constraints because by definition, their labels were determined by the constraints already in the subroutine, which were ambiguous.) This means that if a query is erroneously indicated as unambiguous, it is not added to the optimization problem and thus does not constrain the solution. Because we expect many queries to be redundant, it is even possible that we will infer the true label of this query with queries requested in the future.

For our experiments, we chose $d = \{1, 2, 3\}$ with $n = \{3, \ldots, 30\}$. Note that because binary sort is implemented in both our sequential algorithm and LNM-MDS, neither algorithm can do worse than binary sort, which requests about $n^2 \log_2 n$ queries, regardless of how large d is. All experiments were repeated just 5 times in the interest of time. In Figure 2 we have plotted the mean and standard deviation of the number of requested queries using error bars for the sequential algorithm of Figure 1 in blue, LNM-MDS in black, and binary sort in red. Clearly, LNM-MDS performs nearly as bad as binary sort (but can never perform as badly because binary sort is implemented for all the rankings in LNM-MDS). LNM-MDS was only run for $d = 1, 2$ because it was clear from just these results that LNM-MDS was not exploiting the fact that $d \ll n$. It is also clear that the sequential algorithm requests significantly fewer membership queries than either binary sort or LNM-MDS.

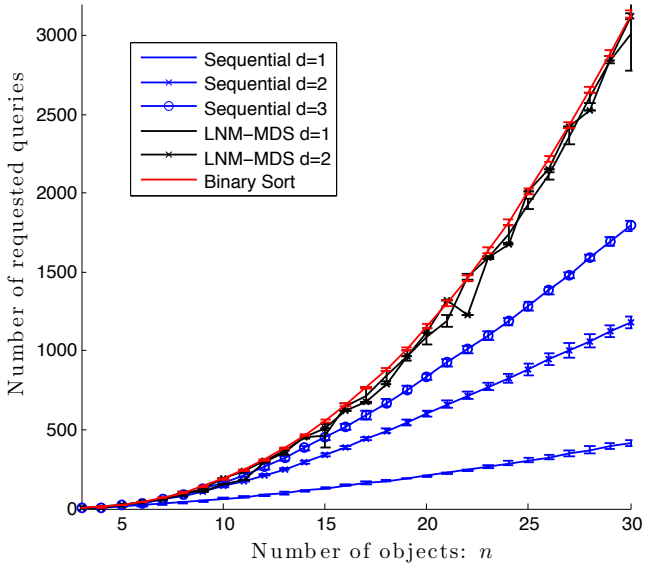

Fig. 2: The mean number of requested membership queries to determine all the constraints of an embedding of n objects in d dimensions using the three algorithms described in Section IV. The standard deviation of the trials are presented using error bars.

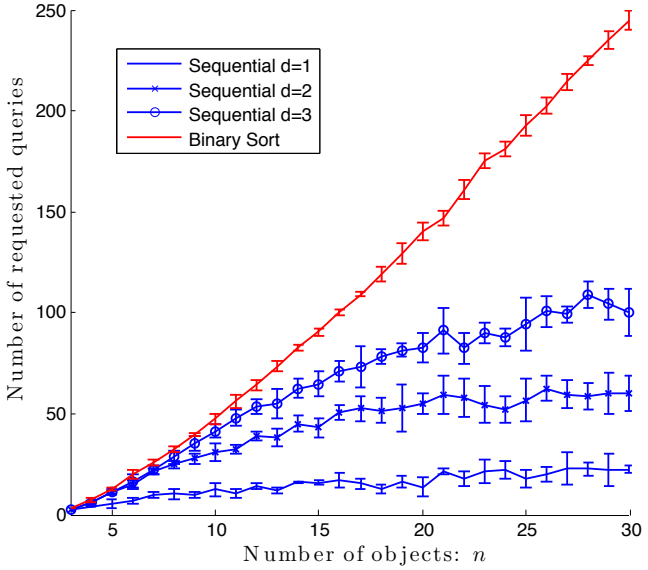

Fig. 3: Given all the constraints between $(n-1)$ objects in d dimensions, the mean number of requested membership queries to determine the all the constraints of n objects in d dimensions. The standard deviation of the trials are presented using error bars.

VI. Discussion

From just Figure 2, for a fixed dimension d, it is unclear how the number of queries grows with n; is it more like $n^2 \log n$ or $n \log n$? It is our conjecture that it grows like the latter. In this section we will analyze the empirical data more closely and also point out some theoretical results that, together, we believe provide strong evidence to support our conjecture.

Consider how many queries are requested when adding just a single object to the embedding. Under the hypothesis

that the number of queries for the sequential algorithm grows like $n \log n$ times some constant depending on the dimension, we should observe that the number of queries required to add just a single object should be no greater than order $\log n$. If the hypothesis is false and the number of queries actually grows faster than this, like $n^2 \log n$, the number of queries requested to add just a single object should grow like $n \log n$. Figure 3 presents the average number of queries required to add just the kth object for $k = 3, \ldots, 30$ and $d = 1, 2, 3$ for the sequential algorithm in blue and for binary sort in red. It is clear that this the quantity associated with the sequential algorithm grows sub-linearly and perhaps even reasonable to conjecture that it grows logarithmically. This behavior can be explained by some previous analyses of non-metric multidimensional scaling and the previous analysis of the ranking problem alluded to earlier.

If we consider an embedding of n objects in d dimensions that satisfies all of the constraints, we know that this embedding lives in some nd-cell and therefore has some amount of flexibility. In related studies, this amount of flexibility is observed to decrease rapidly to zero as n grows. For example, at least qualitatively, the amount of flexibility in an embedding in 2 dimensions has been observed to be negligible for n as small as 10 or 15 using similar constraints to those discussed here [10], [18]. So as $k < n$ becomes very large, adding the $(k+1)$th object becomes more and more like adding an object to a *fixed* embedding of k objects. Recall that the embedding is constrained only so far as forcing each object to rank the other objects with respect to their relative proximity. To add the $(k+1)$th object to the embedding, we must discover how the $(k+1)$th object ranks the other k objects, and how the k objects insert the $(k+1)$th object into their ranking. In previous work, we showed that if the positions of the first k objects are fixed and known, and we have discovered how the $(k+1)$th object has ranked some subset of $j < k$ objects, it requires only about d/j pairwise comparisons, in expectation, to insert the $(j+1)$th object into the ranking [8, Lemma 4]. It follows that to discover how the $(k+1)$th object ranks all k objects, it requires only about $d \log k$ queries. This predicts part of the story, but we still must consider how many queries it requires to insert the $(k+1)$th object in to the rankings of the other $1, \ldots, k$ objects.

As k gets very large, the size of the d-cells corresponding to the possible ways the $(k+1)$th object can rank the first k objects (see Section III) becomes very small, something like on the order k^{-2d}. What this means is that if we first locate the $(k+1)$th object in this tiny cell, with respect to the other objects, *it* looks fixed. This means that to these other objects, it looks as if they are simply adding a fixed object to their ranking which takes only about d/k queries. Using these informal approximations, we should expect that only about $d \log k + k \times d/k \approx d \log k$ queries will be requested to add the $(k+1)$th object. Repeated application of this argument and the observation that embeddings appear more and more fixed as $n \to \infty$, we conjecture with some level of confidence that the algorithm of Section IV-B requests no more than $O(dn \log n)$ queries to uniquely define an embedding of n objects in d dimensions.

VII. Final remarks and extensions

We believe the previous section provided a great deal of support for the conjecture that the number of queries required to embed n objects in d dimensions grows no faster than $O(dn \log n)$. But, of course, this is just a conjecture. Future work will attempt to prove this conjecture. However, theoretical results aside, we have demonstrated an algorithm that requests drastically fewer queries than standard methods due to its exploitation of the known structure of the space. Furthermore, we have made code for this algorithm available on the first author's website [17]. While we have assumed from the start that the objects embed into exactly d dimensions with no violations of the inequalities, this assumption should never be expected to be true in practice, especially when humans providing the query responses. Fortunately, the sequential algorithm described here can easily be made robust to only probably-correct query responses by paying an additional $\log n$ multiplicative factor in the number of requested queries using the techniques developed in [8]. Due to the potential practical benefits of a robust version of the sequential algorithm, we are eager to experiment with human subjects, for that was our driving motivation in the first place.

References

[1] N. Stewart, G.D.A. Brown, and N. Chater. Absolute identification by relative judgment. *Psychological Review*, 112(4):881–911, 2005.

[2] I. Borg and P.J.F. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Verlag, 2005.

[3] S. Agarwal, J. Wills, L. Cayton, G. Lanckriet, D. Kriegman, and S. Belongie. Generalized non-metric multidimensional scaling. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, 2007.

[4] O. Tamuz, C. Liu, S. Belongie, O. Shamir, and A.T. Kalai. Adaptively learning the crowd kernel. *Arxiv preprint arXiv:1105.1033*, 2011.

[5] T.H.A. Bijmolt and M. Wedel. The effects of alternative methods of collecting similarity data for multidimensional scaling* 1. *International Journal of Research in Marketing*, 12(4):363–371, 1995.

[6] B. McFee. Distance metric learning from pairwise proximities.

[7] R.M. Johnson. Pairwise nonmetric multidimensional scaling. *Psychometrika*, 38(1):11–18, 1973.

[8] K. Jamieson and R. Nowak. Active ranking using pairwise comparisons. *Neural Information Processing Systems (NIPS)*, 2011.

[9] C.H. Coombs. A theory of data. *Psychological review*, 67(3):143–159, 1960.

[10] R.N. Shepard. Metric structures in ordinal data. *Journal of Mathematical Psychology*, 3(2):287–315, 1966.

[11] R. C. Buck. Partition of space. *The American Mathematical Monthly*, 50(9):pp. 541–544, 1943.

[12] L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Verlag, 1996.

[13] V. De Silva and J.B. Tenenbaum. Sparse multidimensional scaling using landmark points. *Dept. Math., Stanford University, Stanford, CA, Tech. Rep*, 2004.

[14] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

[15] S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge Univ Pr, 2004.

[16] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer verlag, 1999.

[17] K. Jamieson. Projects. [http://homepages.cae.wisc.edu/~jamieson/me/Projects.html], September 2011.

[18] R. Bissett and B. Schneider. Spatial and conjoint models based on pairwise comparisons of dissimilarities and combined effects: Complete and incomplete designs. *Psychometrika*, 56(4):685–698, 1991.