Universal Multi-Dimensional Scaling[*]

Arvind Agarwal
arvind@cs.utah.edu

Jeff M. Phillips
jeffp@cs.utah.edu

Suresh Venkatasubramanian
suresh@cs.utah.edu

School of Computing
University of Utah
Salt Lake City, Utah, USA

ABSTRACT

In this paper, we propose a unified algorithmic framework for solving many known variants of MDS. Our algorithm is a simple iterative scheme with guaranteed convergence, and is *modular*; by changing the internals of a single subroutine in the algorithm, we can switch cost functions and target spaces easily. In addition to the formal guarantees of convergence, our algorithms are accurate; in most cases, they converge to better quality solutions than existing methods in comparable time. Moreover, they have a small memory footprint and scale effectively for large data sets. We expect that this framework will be useful for a number of MDS variants that have not yet been studied.

Our framework extends to embedding high-dimensional points lying on a sphere to points on a lower dimensional sphere, preserving geodesic distances. As a complement to this result, we also extend the Johnson-Lindenstrauss Lemma to this spherical setting, by showing that projecting to a random $O((1/\varepsilon^2) \log n)$-dimensional sphere causes only an ε-distortion in the geodesic distances.

Categories and Subject Descriptors

H.2.8 [**Database applications**]: Data mining; F.2.2 [**Non-numerical algorithms and problems**]: Geometrical algorithms

Keywords

Multi-dimensional scaling, dimensionality reduction.

1. INTRODUCTION

Multidimensional scaling (MDS) [26, 11, 3] is a widely used method for embedding a general distance matrix into a low dimensional Euclidean space, used both as a preprocessing step for many problems, as well as a visualization tool in its own right. MDS has been studied and used in psychology since the

---

1930s [39, 36, 25] to help visualize and analyze data sets where the only input is a distance matrix. More recently MDS has become a standard dimensionality reduction and embedding technique to manage the complexity of dealing with large high dimensional data sets [9, 10, 34, 7].

In general, the problem of embedding an arbitrary distance matrix into a fixed dimensional Euclidean space with minimum error is nonconvex (because of the dimensionality constraint). Thus, in addition to the standard formulation [13], many variants of MDS have been proposed, based on changing the underlying error function [39, 9]. There are also applications where the target space, rather than being a Euclidean space, is a manifold (e.g. a low dimensional sphere), and various heuristics for MDS in this setting have also been proposed [14, 7].

Each such variant is typically addressed by a different heuristic, including majorization, the singular value decomposition, semidefinite programming, subgradient methods, and standard Lagrange-multipler-based methods (in both primal and dual settings). Some of these heuristics are efficient, and others are not; in general, every new variant of MDS seems to require different ideas for efficient heuristics.

1.1 Our Work

In this paper, we present a unified algorithmic framework for solving many variants of MDS. Our approach is based on an iterative local improvement method, and can be summarized as follows: "Pick a point and move it so that the cost function is locally optimal. Repeat this process until convergence." The improvement step reduces to a well-studied and efficient family of iterative minimization techniques, where the specific algorithm depends on the variant of MDS.

A central result of this paper is a single general convergence result for all variants of MDS that we examine. This single result is a direct consequence of the way in which we break down the general problem into an iterative algorithm combined with a point-wise optimization scheme. Our approach is generic, efficient, and simple. The high level framework can be written in 10-12 lines of MATLAB code, with individual function-specific subroutines needing only a few more lines each. Further, our approach compares well with the best methods for all the variants of MDS. In each case our method is consistently either the best performer or is close to the best, regardless of the data profile or cost function used, while other approaches have much more variable performance. A useful feature of our method is that it is parameter-free, requiring no tuning parameters or Lagrange multipliers in order to perform at its best. Finally, our method has a small memory footprint, allowing it to scale well for large data sets.

Spherical MDS.

An important application of our approach is the problem of performing *spherical* MDS. Spherical MDS is the problem of embedding a matrix of distances onto a (low-dimensional) sphere. Spherical MDS has applications in texture mapping and image analysis [7], and is a generalization of the spherical *dimensionality reduction* problem, where the goal is to map points from a high dimensional sphere onto a low-dimensional sphere. This latter problem is closely related to dimensionality reduction for finite dimensional distributions. A well-known isometric embedding takes a distribution represented as a point on the d-dimensional simplex to the d-dimensional sphere while preserving the Hellinger distance between distributions. A spherical dimensionality reduction result is an important step to representing high dimensional distributions in a lower-dimensional space of distributions, and will have considerable impact in domains that represent data natively as histograms or distributions, such as in document processing [32, 21, 2], image analysis [28, 12] and speech recognition [19].

Our above framework applies directly to this setting, where for the local improvement step we adapt a technique first developed by Karcher for finding geodesic means on a manifold. In addition, we prove a Johnson-Lindenstrauss-type result for the sphere; namely, that n points lying on a d-dimensional sphere can be embedded on a $O((1/\varepsilon^2)\log n)$-dimensional sphere while approximately preserving the geodesic distances between pairs of points, that is, no distance changes by more than a relative $(1+\varepsilon)$-factor. This latter result can be seen as complementary to the local improvement scheme; the formal embedding result guarantees the error while being forced to use $\log n$ dimensions, while the local improvement strategy generates a mapping into any k dimensional hypersphere but provides no formal guarantees on the error.

Summary of contributions.

The main contributions of this paper can be summarized as follows:

- In Section 4 we present our iterative framework, illustrate how it is applied to specific MDS variants and prove a convergence result.

- In Section 5 we present a comprehensive experimental study that compares our approach to the prior best known methods for different MDS variants.

- In Section 6 we prove a formal dimensionality reduction result that embeds a set of n points on a high-dimensional sphere into a sphere of dimension $O(\log n/\epsilon^2)$ while preserving all distances to within relative error of $(1+\epsilon)$ for any $\epsilon > 0$.

2. BACKGROUND AND EXISTING METHODS

Multidimensional scaling is a *family* of methods for embedding a distance matrix into a low-dimensional Euclidean space. There is a general taxonomy of MDS methods [11]; in this paper we will focus primarily the metric and generalized MDS problems.

The traditional formulation of MDS [26] assumes that the distance matrix D arises from points in some d-dimensional Euclidean space. Under this assumption, a simple transformation takes D to a matrix of *similarities* S, where $s_{ij} = \langle x_i, x_j \rangle$.

These similarities also arise from many psychology data sets directly [36, 39]. The problem then reduces to finding a set of points X in k-dimensional space such that XX^T approximates S. This can be done optimally using the top k singular values and vectors from the singular value decomposition of S.

A more general approach called SMACOF that drops the Euclidean assumption uses a technique known as stress majorization [30, 13, 14]. It has been adapted to many other MDS variants as well including restrictions of data to lie on quadratic surfaces and specifically spheres [14].

Since the sum-of-squares error metric is sensitive to outliers, Cayton and Dasgupta [9] proposed a robust variant based on an ℓ_1 error metric. They separate the rank and cost constraints, solving the latter using either semidefinite programming or a subgradient heuristic, followed by a singular value decomposition to enforce the rank constraints.

Many techniques have been proposed for performing spherical MDS. Among them are majorization methods ([33] and SMACOF-Q [14]), a multiresolution approach due to Elad, Keller and Kimmel [16] and an approach based on computing the classical MDS and renormalizing [34].

Embeddings that guarantee bounded error.

A complementary line of work in dimensionality reduction fixes an error bound for *every* pair of distances (rather than computing an average error), and asks for the minimum dimension a data set can be embedded in while maintaining this error. The Johnson-Lindenstrauss Lemma [22] states that any collection of n points in a Euclidean space can be embedded in a $O((1/\varepsilon^2)\log n)$ dimensional Euclidean space that preserves all distances within a relative error of ε. If the points instead define an abstract metric space, then the best possible result is an embedding into $O(\log n)$-dimensional Euclidean space that preserves distances up to a factor of $O(\log n)$. An exhaustive survey of the different methods for dimensionality reduction is beyond the scope of this paper - the reader is directed to the survey by Indyk and Matousek for more information [20].

The Johnson-Lindenstrauss lemma can be extended to data lying on manifolds. Any manifold M with "linearization dimension" k (a measure of its complexity) can be embedded into a $O((1/\varepsilon^2)k\log(kn))$ dimensional space so that all pairwise *Euclidean* distances between points on M are distorted by at most a relative $(1+\varepsilon)$-factor [1, 35, 29]. A k-dimensional sphere has linearization dimension $O(k)$, so this bound applies directly for preserving the chordal (i.e Euclidean) distance between points on a sphere. The geodesic distance between points on a sphere can be interpreted as the angle between the points in radians, and a result by Magen [29] show that $O((1/\varepsilon^2)\log n)$ dimensions preserve angles to within a relative factor of $1+\sqrt{\varepsilon}$ (which is weaker than our result preserving the geodesic distance to within a relative factor of $1+\varepsilon$).

3. DEFINITIONS

Let $D = (d_{ij})$ be an $n \times n$ matrix representing distances between all pairs of points in a set $Y = \{y_1, \ldots y_n\}$. In general, we assume that D is symmetric (i.e $d_{ij} = d_{ji}$), although our method does not formally require this. The multidimensional scaling problem takes as input Y, D and k, and asks for a mapping $\mu : Y \to X$ from Y to a set of points X in a k-dimensional space T such that the difference between the original and resulting distances is minimized.

There are many different ways to measure the difference between the sets of distances, and these can be captured by the

following general function:

$$C(X, D) = \sum_i \sum_j \text{Err}(f(x_i, x_j) - d_{ij})$$

where Err measures the discrepancy between the source and target distances, and f denotes the function that measures distance in the target space.

- $T = \mathbb{R}^k$, $\text{Err}(\delta) = \delta^2$, $f(x, x') = \|x - x'\|_2$: This is a general form of the MDS problem, which we refer to as fMDS.

- $T = \mathbb{R}^k$, $\text{Err}(\delta) = |\delta|$, $f(x, x') = \|x - x'\|_2$: This is a *robust* variant of MDS called rMDS, first suggested by Cayton and Dasgupta [9].

- $T = S^k$, $\text{Err}(\delta) = |\delta|$ or δ^2, $f(x, x')$ is either chordal(c) or geodesic distance (g) on S^k. We refer to this family of problems as {c,g}-{1,2}-sMDS.

It will be convenient to split the expression into component terms. We define

$$C_i(X, D, x_i) = \sum_j \text{Err}(f(x_i, x_j) - d_{ij})$$

which allows us to write $C(X, D) = \sum_i C_i(X, D, x_i)$.

Notes.

The actual measure studied by Cayton and Dasgupta[9] is not rMDS. It is a variant which takes the absolute difference of the *squared* distance matrices. We call this measure r^2MDS. Also, classical MDS does not appear in this list since it tries to minimize the error between similarities rather than distances. We refer to this measure as cMDS.

4. ALGORITHM

We now present our algorithm PLACECENTER(X, D) that finds a mapping $Y \to X$ minimizing $C(X, D)$. For now we assume that we are given an initial embedding $X_1 \in \mathbb{R}^k$ to seed our algorithm. Our experiments indicate the SVD-based approach [39] is almost always the optimal way to seed the algorithm, and we use it unless specifically indicated otherwise.

Algorithm 1 PLACECENTER (D)

Run any MDS strategy to obtain initial seed X.
repeat
  $\epsilon \leftarrow C(X, D)$.
  for $i = 1$ to n **do**
    $x_i \leftarrow$ PLACE$_i(X, D)$. {this updates $x_i \in X$}
  end for
until $(\epsilon - C(X, D) < t)$ {for a fixed threshold t}
Return X.

The algorithm operates by employing a technique from the block-relaxation class of heuristics. The cost function can be expressed as a sum of costs for each point x_i, and so in each step of the inner loop we find the best placement for x_i while keeping all other points fixed, using the algorithm PLACE$_i(X, D)$. A key insight driving our approach is that PLACE$_i(X, D)$ can be implemented either iteratively or exactly for a wide class of distance functions. The process terminates when over all i, invoking PLACE$_i(X, D)$ does not reduce the cost $C(X, D)$ by more than a threshold t. The algorithm takes $O(n^2)$ for each iteration, since PLACE$_i(X, D)$ will take $O(n)$ time and computing $C(X, D)$ takes $O(n^2)$ time.

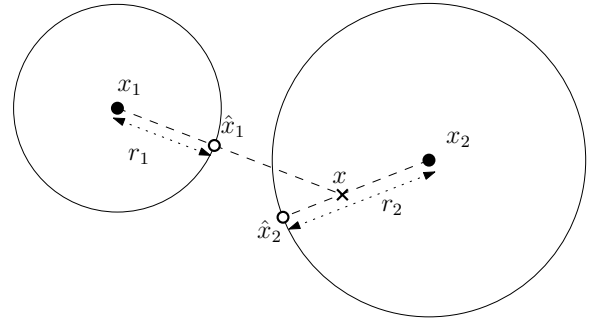**Figure 1: A geometric interpretation of the error term.**

4.1 A Geometric Perspective on PLACE$_i(X, D)$

The routine PLACE$_i(X, D)$ is the heart of our algorithm. This routine finds the optimal placement of a fixed point x_i with respect to the cost function $C_i(X, D, x_i) = \sum_j \text{Err}(f(x_i, x_j) - d_{ij})$. Set $r_j = d_{ij}$. Then the optimal placement of x_i is given by the point x^* minimizing the function

$$g(x) = \sum_j \text{Err}(f(x, x_j) - r_j).$$

Note that the terms $f(x, x_i)$ and $r_i = d_{ii}$ are zero, so we can ignore their presence in the summation for ease of notation.

There is a natural geometric interpretation of $g(x)$, illustrated in Figure 1. Consider a sphere around the point x_j of radius r_j. Let $\hat{x}_j$ be the point on this sphere that intersects the ray from x_j towards x. Then the distance $f(x, \hat{x}_j) = |f(x, x_j) - r_j|$. Thus, we can rewrite $g(x)$ as

$$g(x) = \sum_j \text{Err}(f(x, \hat{x}_j)).$$

This function is well-known in combinatorial optimization as the min-sum problem. For $\text{Err}(\delta) = \delta^2$, $g(x)$ finds the point minimizing the sum-of-squared distances from a collection of fixed points (the 1-mean), which is the centroid $x^* = \frac{1}{n} \sum_j \hat{x}_j$. For $\text{Err}(\delta) = |\delta|$, $g(x)$ finds the 1-median: the point minimizing the sum of distances from a collection of fixed points. Although there is no closed form expression for the 1-median, there are numerous algorithms for solving this problem both exactly [38] and approximately [4]. Methods that converge to the global optimum exist for any $\text{Err}(\delta) = |\delta|^p, p \le 2$; it is known that if p is sufficiently larger than 2, then convergent methods may not exist [6].

While $g(x)$ can be minimized optimally for error functions Err of interest, the location of the points $\hat{x}_j$ depends on the location of the solution x^*, which is itself unknown! This motivates an alternating optimization procedure, where the current iterate x is used to compute $\hat{x}_j$, and then these $\hat{x}_j$ are used as input to the min-sum problem to solve for the next value of x.

4.2 Implementing RECENTER

Up to this point, the description of PLACECENTER and PLACE has been generic, requiring no specification of Err and f. In fact, all the domain-specificity of the method appears in RECENTER, which solves the min-sum problem. We now demonstrate how different implementations of RECENTER allow us to solve the different variants of MDS discussed above.

Algorithm 2 $\text{PLACE}_i(X, D)$

repeat
   $\epsilon \leftarrow g(x_i)$
   for $j = 1$ **to** n **do**
      $\hat{x}_j \leftarrow$ intersection of sphere of radius r_j around x_j with ray from x_j towards x_i.
   end for
   $x_i \leftarrow \text{RECENTER}(\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n\})$.
until $(\epsilon - g(x_i) < t)$ {for a fixed threshold t}
Return x_i.

4.2.1 The original MDS: fMDS

Recall from Section 3 that the fMDS problem is defined by $\text{Err}(\delta) = \delta^2$ and $f(x, x') = \|x - x'\|_2$. Thus, $g(x) = \sum_j \|x - \hat{x}_j\|^2$. As mentioned earlier, the minimum of this function is attained at $x^* = (1/n) \sum_j \hat{x}_j$. Thus, $\text{RECENTER}(\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n\})$ merely outputs $(1/n) \sum_j \hat{x}_j$, and takes $O(n)$ time per invocation.

4.2.2 Robust MDS: rMDS

The robust MDS problem rMDS is defined by $\text{Err}(\delta) = |\delta|$ and $f(x, x') = \|x - x'\|_2$. Minimizing the resulting function $g(x)$ yields the famous Fermat-Weber problem, or the 1-median problem as it is commonly known. An exact iterative algorithm for solving this problem was given by Weiszfeld [38], and works as follows. At each step of PLACE_i the value x_i is updated by

$$x_i \leftarrow \sum_j \frac{\hat{x}_j}{\|x_i - \hat{x}_j\|} \Big/ \sum_j \frac{1}{\|x_i - \hat{x}_j\|}.$$

This algorithm is guaranteed to converge to the optimal solution [27, 31], and in most settings converges quadratically [24].

Other Norms and Distances.

If $\text{Err}(\delta) = |\delta|^p, 1 < p < 2$, then an iterative algorithm along the same lines as the Weiszfeld algorithm can be used to minimize $g(x)$ optimally [6]. In practice, this is the most interesting range of values for p. It is also known that for p sufficiently larger than 2, this iterative scheme may not converge.

We also can tune PLACECENTER to the $r^2\text{MDS}$ problem (using squared distances) by setting $r_j = d_{ij}^2$. Although the convergence proofs (below) do not hold in this case, the algorithm works well in practice.

4.2.3 Spherical MDS

Spherical MDS poses special challenges for the implementation of RECENTER. Firstly, it is no longer obvious what the definition of $\hat{x}_j$ should be, since the "spheres" surrounding points must also lie on the sphere. Secondly, consider the case where $\text{Err}(\delta) = \delta^2$, and $f(x, x')$ is given by geodesic distance on the sphere. Unlike in the case of $\mathbb{R}^k$, we no longer can solve for the minimizer of $g(x)$ by computing the centroid of the given points, because this centroid will not in general lie on the sphere, and even computing the centroid followed by a projection onto the sphere will not guarantee optimality.

The first problem can be solved easily. Rather than draw spheres around each x_j, we draw *geodesic spheres*, which are the set of points at a fixed *geodesic* distance from x_j. On the sphere, this set of points can be easily described as the intersection of an appropriately chosen halfplane with the sphere. Next, instead of computing the intersection of this geodesic sphere with

the ray from x_j towards the current estimate of x_i, we compute the intersection with a *geodesic ray* from x_j towards x_i.

The second problem can be addressed by prior work on computing min-sums on manifolds. Karcher [23] proposed an iterative scheme for the geodesic sum-of-squares problem that always converges as long as the points do not span the entire sphere. His work extends (for the same functions Err, f) to points defined on more general Riemannian manifolds satisfying certain technical conditions. It runs in $O(n)$ time per iteration.

For the robust case ($\text{Err}(\delta) = |\delta|$), the Karcher scheme no longer works. For this case, we make use of a Weiszfeld-like adaption [18] that again works on general Riemannian manifolds, and on the sphere in particular. Like the Weiszfeld scheme, this approach takes $O(n)$ time per iteration.

4.3 Convergence Proofs

Here we prove that each step of PLACECENTER converges as long as the recursively called procedures reduce the relevant cost functions. Convergence is defined with respect to a cost function κ, so that an algorithm converges if at each step κ decreases until the algorithm terminates.

Theorem 4.1. *If each call to $\tilde{x}_i \leftarrow \text{PLACE}_i(X, D)$ decreases the cost $C_i(X, D, x_i)$, then $\text{PLACECENTER}(D)$ converges with respect to $C(\cdot, D)$.*

Proof. Let $\tilde{X} \leftarrow \text{PLACE}_i(X, D)$ result from running an iteration of $\text{PLACE}_i(X, D)$. Let $\tilde{X} = \{x_1, \ldots, x_{i-1}, \tilde{x}_i, x_{i+1}, \ldots, x_n\}$. Then we can argue

$$
\begin{aligned}
&C(X, D) - C(\tilde{X}, D) \\
&= 2 \sum_{j=1}^{n} \text{Err}(f(x_i, x_j) - d_{i,j}) - 2 \sum_{j=1}^{n} \text{Err}(f(\tilde{x}_i, x_j) - d_{i,j}) \\
&= 2 C_i(X, D, x_i) - 2 C_i(\tilde{X}, D, \tilde{x}_i) > 0.
\end{aligned}
$$

The last line follows because X and $\tilde{X}$ only differ at x_i versus $\tilde{x}_i$, and by assumption on $\text{PLACE}_i(X, D)$, this sub-cost function must otherwise decrease. $\square$

Theorem 4.2. *If each call $x_i \leftarrow \text{RECENTER}(\hat{X})$ reduces $\sum_{j=1}^{n} f(x_i, \hat{x}_j)^p$, then $\text{PLACE}_i(X, D, x_i)$ converges with respect to $C_i(X, D, \cdot)$.*

Proof. First we can rewrite

$$
\begin{aligned}
C_i(X, D, x_i) &= \sum_{j=1}^{n} \text{Err}(f(x_i, x_j) - d_{i,j}) \\
&= \sum_{j=1}^{n} \text{Err}((f(x_i, \hat{x}_j) + d_{i,j}) - d_{i,j}) \\
&= \sum_{j=1}^{n} \text{Err}(f(x_i, \hat{x}_j)).
\end{aligned}
$$

Since $\text{Err}(f(x_i, \hat{x}_j))$ measures the distance to the sphere $\circ_j$. Then choosing x_i' to minimize (or decrease) $\sum_{j=1}^{n} \text{Err}(f(x_i', \hat{x}_j))$, must decrease the sum of distances to each point $\hat{x}_j$ on each sphere $\circ_j$. Now let $\hat{x}_j'$ be the closest point to x_i' on $\circ_j$. Hence $\text{Err}(f(x_i', \hat{x}_j')) \leq \text{Err}(f(x_i', \hat{x}_j))$ and thus

$$
\begin{aligned}
C_i(X, D, x_i') &= \sum_{j=1}^{n} \text{Err}(f(x_i', \hat{x}_j')) \leq \sum_{j=1}^{n} \text{Err}(f(x_i', \hat{x}_j)) \\
&\leq \sum_{j=1}^{n} \text{Err}(f(x_i, \hat{x}_j)) = C_i(X, D, x_i)
\end{aligned}
$$

where equality only holds if $x_i = x'_i$, in which case the algorithm terminates. □

4.4 Working Space Usage

PLACECENTER(D) takes an $n \times n$ distance matrix as input, but each invocation of PLACE$_i$(X, D) only operates on a single point. This means that although the input complexity is $O(n^2)$, the working memory footprint of the algorithm is only $O(n)$. This is a significant advantage of PLACECENTER(D) over many existing MDS methods that require the entire matrix D to be stored in memory. In Section 5 we will see that this small memory footprint enables us to run PLACECENTER(D) for values of n well beyond the point where other methods start to fail.

5. EXPERIMENTS

In this section we evaluate the performance of PLACECENTER (PC). Since PC generalizes to many different cost functions, we compare it with the best known algorithm for each cost function, if one exists. For the fMDS problem the leading algorithm is SMACOF [14]; for the r^2MDS problem the leading algorithm is by Cayton and Dasgupta (CD) [9]. We know of no previous scalable algorithm designed for rMDS. We note that the Cayton-Dasgupta algorithm REE does not exactly solve the r^2MDS problem. Instead, it takes a non-Euclidean distance matrix and finds a Euclidean distance matrix that minimizes the error without any rank restrictions. Thus, as suggested by the authors [9], to properly compare the algorithms, we let CD refer to running REE and then projecting the result to a k-dimensional subspace using the SVD technique [39] (our plots show this projection after each step). With regards to each of these Euclidean measures we compare our algorithm with SMACOF and CD. We also compare with the popular SVD-based method [39], which solves the related cMDS problem based on similarities, by seeding all three iterative techniques with the results of the closed-form SVD-based solution.

Then we consider the family of spherical MDS problems {c,g}-{1,2}-sMDS. We compare against a version of SMACOF-Q [14] that is designed for data restricted to a low dimensional sphere, specifically for the c-2-SMDS measure. We compare this algorithm to ours under the c-2-SMDS measure (for a fair comparison with SMACOF-Q) and under the g-1-SMDS measure which is the most robust to noise.

The subsections that follow focus on individual cost measures. We then discuss the overall behavior of our algorithm in Section 5.6.

Data sets, code, and setup.

Test inputs for the algorithms are generated as follows. We start with input consisting of a random point set with $n = 300$ points in $\mathbb{R}^d$ for $d = 200$, with the target space $T = \mathbb{R}^k$ with $k = 10$. Many data sets in practice have much larger parameters n and d, but we limit ourselves to this range for most of the experiments because for larger values CD becomes prohibitively slow, and both SMACOF and CD run into memory problems. In Section 5.5 we explore the performance of our algorithm on larger data sets (up to 50,000 points). The data is generated to first lie on a k-dimensional subspace, and then (full-dimensional) Poisson noise is applied to all points up to a magnitude of 30% of the variation in any dimension. Finally, we construct the Euclidean distance matrix D which is provided as input to the algorithms.

These data sets are Euclidean, but "close" to k-dimensional. To examine the behavior of the algorithms on distance matri-
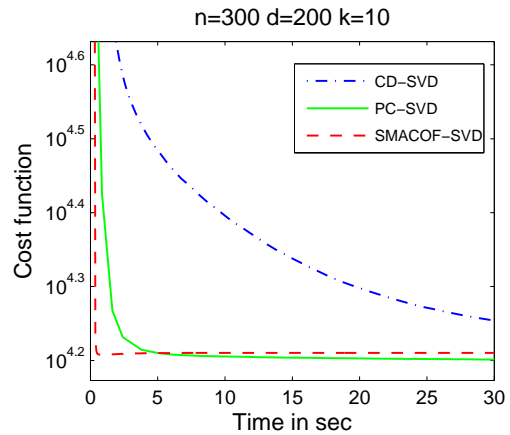

Figure 2: rMDS: A typical behavior of the PC, CD and SMACOF for rMDS problem.

ces that are non-Euclidean, we generate data as before in a k-dimensional subspace and generate the resulting distance matrix D. Then we perturb a fraction of the elements of D (rather than perturbing the points) with Poisson noise. The fraction perturbed varies in the set (2%, 10%, 30%, 90%).

All algorithms were implemented in MATLAB. For SMACOF, we used the implementation provided by Bronstein [8], and built our own implementation of SMACOF-Q around it. For all other algorithms, we used our own implementation[1]. In all cases, we compare performance in terms of the error function Err as a function of clock time.

5.1 The rMDS Problem

Figure 2 shows the cost function Err associated with rMDS plotted with respect to runtime. PLACECENTER always reaches the best local minimum, partially because only PLACECENTER can be adjusted for the rMDS problem. We also observe that the runtime is comparable to SMACOF and much faster than CD in order to get to the same Err value. Although SMACOF initially reaches a smaller cost that PC, it later converges to a larger cost because it optimizes a different cost function (fMDS).

We repeat this experiment in Figure 3 for different values of k (equal to {2, 20, 50, 150}) to analyze the performance as a function of k. Note that PC performs even better for lower k in relation to CD. This is likely as a result of CD's reliance on the SVD technique to reduce the dimension. At smaller k, the SVD technique has a tougher job to do, and optimizes the wrong metric. Also for $k = 150$ note that CD oscillates in its cost; this is again because the REE part finds a nearby Euclidean distance matrix which may be inherently very high dimensional and the SVD projection is very susceptible to changes in this matrix for such large k. We observe that SMACOF is the fastest method to reach a low cost, but does not converge to the lowest cost value. The reason it achieves a cost close to that of PC is that for this type of data the rMDS and fMDS cost functions are fairly similar.

In Figure 4 we evaluate the effect of changing the amount of noise added to the input distance matrix D, as described above. We consider two variants of the CD algorithm, one where it is seeded with an SVD-based seed (marked CD+SVD) and one

---

[1]All of our code may be found at `http://www.cs.utah.edu/~suresh/papers/smds/smds.html`.

Figure 3: rMDS: Variation with $k = 2, 20, 50, 150$.

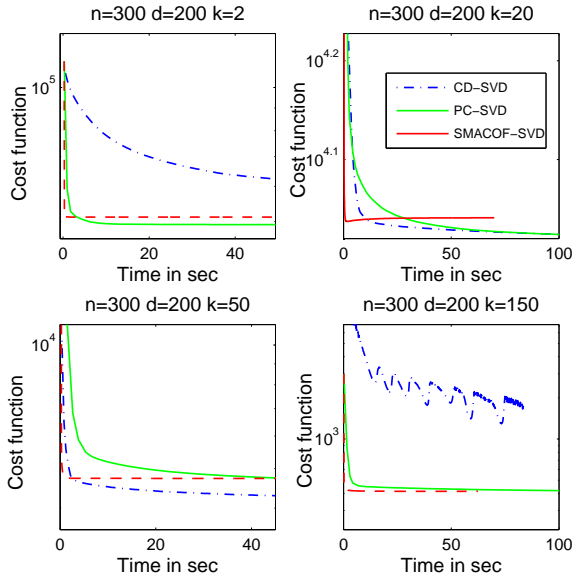Figure 4: rMDS: Variation with noise$= 2, 10, 30, 90$.

Figure 5: fMDS: Variation with $k = 2, 20, 50, 150$

where it is seeded with a random projection to a k-dimensional subspace (marked CD+rand). In both cases the plots show the results of the REE algorithm after SVD-type projections back to a k-dimensional space.

The CD+SVD technique consistently behaves poorly and does not improve with further iterations. This probably is because the REE component finds the closest Euclidean distance matrix which may correspond to points in a much high dimensional space, after which it is difficult for the SVD to help. The CD+rand approach does much better, likely because the random projection initializes the procedure in a reasonably low dimensional space so REE can find a relatively low dimension Euclidean distance matrix that is nearby. SMACOF is again the fastest algorithm, but with more noise, the difference between fMDS and rMDS is larger, and thus SMACOF converges to a configuration with much higher cost than PC. We reiterate that PC consistently converges to the lowest cost solution among the different methods, and consistently is either the fastest or is comparable to the fastest algorithm. We will see this trend repeated with other cost measures as well.

5.2 The fMDS Problem

We next evaluate the algorithms PC, SMACOF, and CD under the fMDS distance measure. The results are very similar to the rMDS case except now both SMACOF and PC optimizing the correct distance measure and converge to the same local minimum. SMACOF is still slightly faster that PC, but since they both run very fast, the difference is of the order of less than a second even in the very worst part of the cost/time tradeoff curve shown in Figure 5. Note that CD performs poorly under this cost function here except when $k = 50$. For smaller values of k, the SVD step does not optimize the correct distance and for larger k the REE part is likely finding an inherently very high dimensional Euclidean distance matrix, making the SVD projection very noisy.

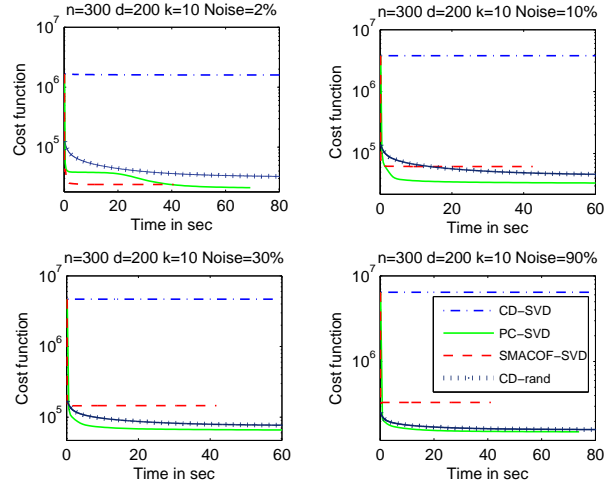For the fMDS measure, SMACOF and PC perform very similarly under different levels of noise, both converging to similar
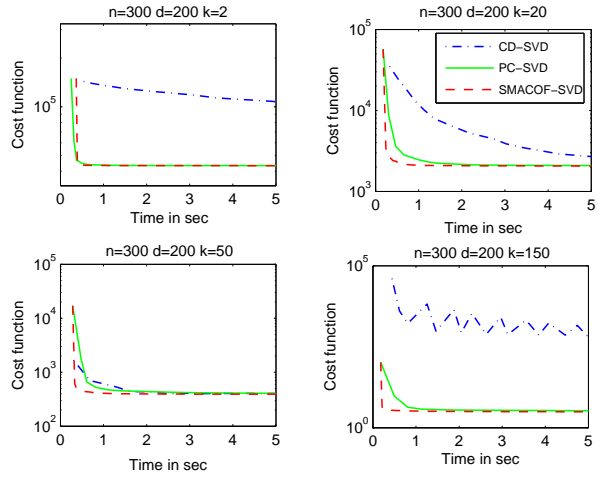
cost functions with SMACOF running a bit faster, as seen in Figure 6. CD consistently runs slower and converges to a higher cost solution.

5.3 The r^2MDS Problem

In this setting we would expect CD to perform consistently as well as PC because both minimize the same cost function. However, this is not always the case because CD requires the SVD step to generate a point set in $\mathbb{R}^k$. As seen in Figure 7 this becomes a problem when k is small ($k = 2, 10$). For medium values of k, CD converges slightly faster than PC and sometimes to a slightly lower cost solution, but again for large k ($= 150$), the REE part has trouble handling the amount of error and the solution cost oscillates. SMACOF is again consistently the fastest to converge, but unless k is very large (i.e. $k = 150$) then it converges to a significantly worse solution because the fMDS and r^2MDS error functions are different.
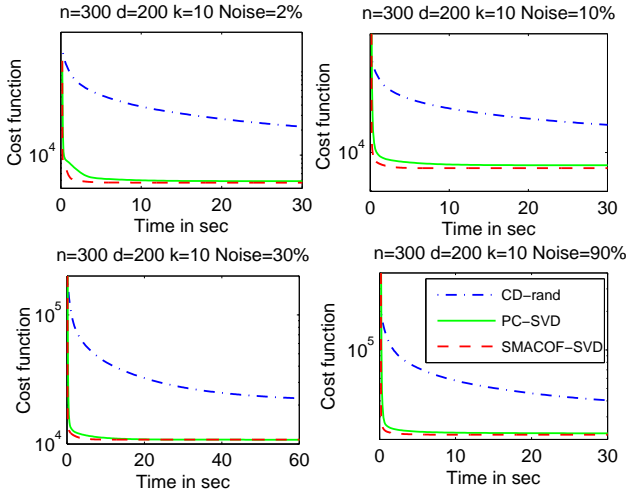
5.4 The Spherical MDS Problem

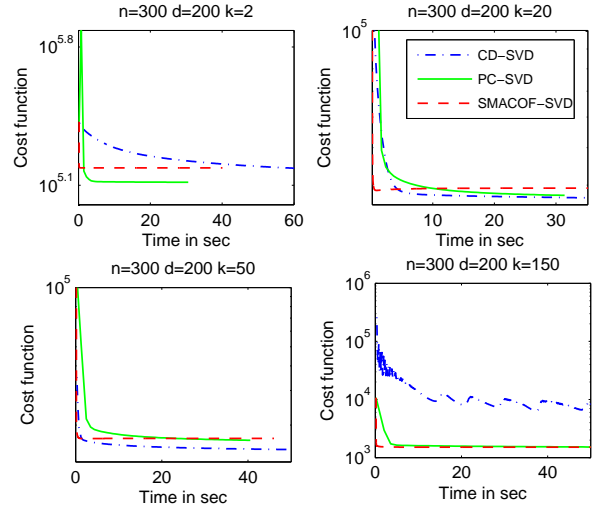**Figure 6:** fMDS: Variation with noise = $2, 10, 30, 90$.

Figure 7: r^2MDS: Variation with $k = 2, 20, 50, 150$

For the spherical MDS problem we compare PC against SMACOF-Q, an adaptation of SMACOF to restrict data points to a low-dimensional sphere, and a technique of Elad, Keller and Kimmel [16]. It turns out that the Elad *et.al.* approach consistently performs poorly compared to both other techniques, and so we do not display it in our reported results. SMACOF-Q basically runs SMACOF on the original data set, but also adds one additional point p_0 at the center of the sphere. The distance $d_{0,i}$ between any other point p_i and p_0 is set to be 1 thus encouraging all other points to be on a sphere, and this constraint is controlled by a weight factor κ, a larger κ implying a stronger emphasis on satisfying this constraint. Since the solution produced via this procedure may not lie on the sphere, we normalize all points to the sphere after each step for a fair comparison.

Here we compare PC against SMACOF-Q in the g-1-SMDS (Figure 8) and the c-2-SMDS (Figure 9) problem. For g-1-SMDS, PC does not converge as quickly as SMACOF-Q with small κ, but it reaches a better cost value. However, when SMACOF-Q is run with a larger κ, then PC runs faster and reaches nearly the same cost value. For our input data, the solution has similar g-1-MDS and c-1-MDS cost. When we compare SMACOF-Q with PC under c-2-MDS (Figure 9) then for an optimal choice of κ in SMACOF-Q, both PC and SMACOF-Q perform very similarly, converging to the same cost function and in about the same time. But for larger choices of κ SMACOF-Q does much worse than PC.

In both cases, it is possible to find a value of κ that allows SMACOF-Q to match PC. However, this value is different for different settings, and varies from input to input. The key observation here is that since PC is *parameter-free*, it can be run regardless of the choice of input or cost function, and consistently performs well.

5.5 Processing Large Data Sets

As mentioned in Section 4, the memory footprint of PC is linear in the number of points. We ran PC for fMDS and compared it to SMACOF and CD (Figure 10). Both SMACOF and CD fail to run after $n = 5000$ because they run out of memory, while the performance of PC scales fairly smoothly even up to $50,0000$ points. Before $n = 5000$, SMACOF performs quite well, but the performance of CD starts deteriorating rapidly.

We avoid storing the full $O(n^2)$-sized distance matrix D, by recomputing each distance $d_{i,j}$ as needed. Thus we only store the original point set, which has size $O(nd)$. This approach works for any dataset where distances $d_{i,j}$ can be quickly recomputed, not just Euclidean data in $\mathbb{R}^d$ for moderate d. Alternatively, we could have read distances from disk for the point currently being processed instead of recomputing them on the fly. Note, we also seed all algorithms with a random projection instead of cMDS since cMDS also has a memory bottleneck of around $n = 5000$.

These preliminary results indicate that our method can be effective on larger data sets. In ongoing work, we are comparing PC to scalable MDS methods like FastMap[17], Metric Map[37], Landmark MDS[15] and Pivot-MDS[5] that sacrifice quality (by reducing the number of "pivot" or "landmark" points supplied to the MDS routine) for speed. Preliminary experiments indicate that our method is comparable to these approaches in speed, while delivering significantly better quality. We note that these methods are limited in general to cMDS, unlike PC.

5.6 Summary of Results

In summary, here are the main conclusions that can be drawn from this experimental study. Firstly, PC is consistently among the top performing methods, regardless of the choice of cost function, the nature of the input, or the level of noise in the problem. Occasionally, other methods will converge faster, but will not in general return a better quality answer, and different methods have much more variable behavior with changing inputs and noise levels.

6. A JL LEMMA FOR SPHERICAL DATA

In this section we present a Johnson-Lindenstrauss-style bound for mapping data from a high dimensional sphere to a low-dimensional sphere while preserving the distances to within a multiplicative error of $(1 + \epsilon)$.

Consider a set $Y \subset \mathbb{S}^d \subset \mathbb{R}^{d+1}$ of n points, defining a distance matrix D where the element $d_{i,j}$ represents the geodesic distance between y_i and y_j on $\mathbb{S}^k$. We seek an embedding of Y into $\mathbb{S}^d$ that preserves pairwise distances as much as possible. For a set $Y \in \mathbb{S}^d$ and a projection $\pi(Y) = X \subset \mathbb{S}^k$ we say the X has γ-
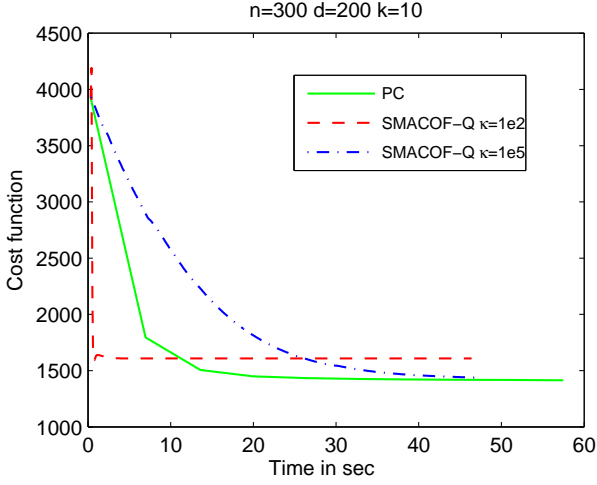
Figure 8: **g-1-SMDS: Comparing PC with SMACOF-Q for different values of penalty parameter** κ.

Figure 9: **c-2-SMDS: Comparing PC with SMACOF-Q for different values of penalty parameter** κ

distortion from Y if these exists a constant c such that for all $x_i, x_j \in X$

$$(1 - \gamma)f(y_i, y_j) \leq cf(x_i, x_j) \leq (1 + \gamma)f(y_i, y_j).$$

For a subspace $H = \mathbb{R}^k$, let $\pi_H(Y)$ be the projection of $Y \in \mathbb{R}^d$ onto H and then scaled by d/k. For $X \in \mathbb{R}^k$, let $S(X)$ be the projection to $\mathbb{S}^{k-1}$, that is for all $x \in X$, the corresponding point in $S(X)$ is $x/\|x\|$.

When $f(y_i, y_j) = \|y_i - y_j\|$, and $Y \in \mathbb{R}^d$, then the Johnson-Lindenstrauss (JL) Lemma [22] says that if $H \subset \mathbb{R}^d$ is a random k-dimensional linear subspace with $k = O((1/\varepsilon^2) \log(n/\delta))$, then $X = \pi_H(Y)$ has ε-distortion from Y with probability at least $1 - \delta$.

We now present the main result of this section. We note that recent results [1] have shown similar results for point on a variety of manifolds (including spheres) where projections preserve Euclidean distances. We reiterate that our results extend this to geodesic distances on spheres which can be seen as angle $\angle_{x,y}$ between the vectors to points $x, y \in \mathbb{S}^k$. Another recent result [29] shows that $k = O((1/\varepsilon^2) \log(n/\delta))$ dimensions preserves $\sqrt{\varepsilon}$-distortion in angles, which is weaker than the following result.

Theorem 6.1. *Let $Y \subset \mathbb{S}^d \subset \mathbb{R}^{d+1}$, and let $H = \mathbb{R}^{k+1}$ be a random subspace of $\mathbb{R}^d$ with $k = O((1/\varepsilon^2) \log(n/\delta))$ with $\varepsilon \in (0, 1/4)$. Let $f(y_i, y_j)$ measure the geodesic distance on $\mathbb{S}^d$ (or $\mathbb{S}^k$ as appropriate). Then $S(\pi_H(Y))$ has ε-distortion from Y with probability at least $1 - \delta$.*

This implies that if we project n data points that lie on any high-dimensional sphere to a low-dimensional sphere $\mathbb{S}^k$ with $k \sim \log n$, then the pairwise distances are each individually preserved. Before we proceed with the proof, we require a key technical lemma.

Lemma 6.1. *For $\varepsilon \in [0, 0.5]$ and $x \in [0, 0.7]$,*

(1) $\sin((1 - 2\varepsilon)x) \leq (1 - \varepsilon) \sin(x)$*, and*

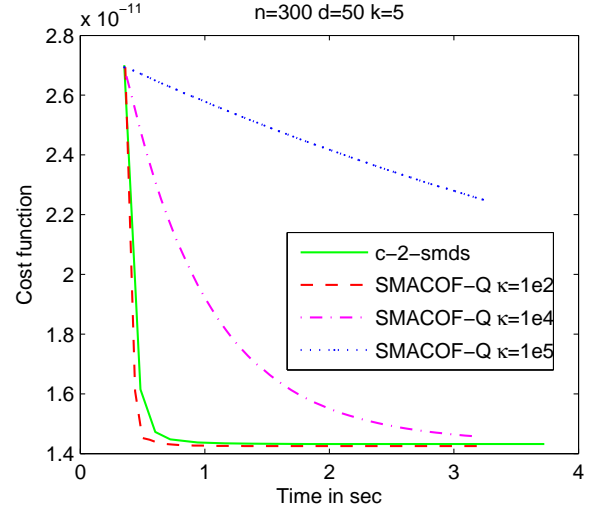*(2)* $\sin((1 + 2\varepsilon)x) \geq (1 + \varepsilon) \sin(x)$.*

Proof. Let $g_\varepsilon(x) = (1 - \varepsilon) \sin x - \sin((1 - 2\varepsilon)x)$. We will show that for $x \in [0, 1]$ and $\varepsilon \in [0, 0.5]$, $g_\varepsilon(x)$ is concave. This implies that it achieves its minimum value at the boundary. Now $g_\varepsilon(0) = 0$ for all ε, and it can be easily shown that $g_\varepsilon(0.7) \geq 0$ for $\varepsilon \in [0, 0.5]$. This will therefore imply that $g_\varepsilon(x) \geq 0$ in the specified range.

It remains to show that $g_\varepsilon(x)$ is concave in $[0, 0.7]$.

$$\begin{aligned} g_\varepsilon''(x) &= (1 - 2\varepsilon)^2 \sin((1 - 2\varepsilon)x) - (1 - \varepsilon) \sin x \\ &\leq (1 - \varepsilon)(\sin((1 - 2\varepsilon)x) - \sin x) \end{aligned}$$

which is always negative for $\varepsilon \in [0, 0.5]$ and since $\sin x$ is increasing in the range $[0, 0.7]$.

This proves the first part of the lemma. For the second part, observe that $h_\varepsilon(x) = \sin((1 + 2\varepsilon)x) - (1 + \varepsilon) \sin(x)$ can be rewritten as $h_\varepsilon(x) = g_{-\varepsilon}(-x)$. The rest of the argument follows along the same lines, by showing that $h_\varepsilon(x)$ is concave in the desired range using that $h_\varepsilon''(x) = g_{-\varepsilon}''(-x)$.

While the upper bound of 0.7 on x is not tight, it is close. The actual bound (evaluated by direct calculation) is slightly over 0.72. $\square$

Proof of Theorem 6.1. Let $X = \pi_H(Y)$. We consider two cases, (*Short Case*) when $\|y_i - y_j\| \leq 1/2$ and (*Long Case*) when $\|y_i - y_j\| \in (1/2, 2]$.

Short Case: First consider points $y_i, y_j \in \mathbb{S}^d$ such that $\|y_i - y_j\| \leq 1/2$. Note that $\|y_i - y_j\| = 2 \sin(\angle_{y_i, y_j}/2)$, since $\|y_i\| = \|y_j\| = 1$. By JL, we know that there exists a constant c such that

$$(1 - \varepsilon/8)\|y_i - y_j\| \leq c\|x_i - x_j\| \leq (1 + \varepsilon/8)\|y_i - y_j\|.$$

We need to compare the angle $\angle_{x_i, x_j}$ with that of $\angle_{y_i, y_j}$. The largest $\angle_{x_i, x_j}$ can be is when $c\|x_i\| = c\|x_j\| = (1 - \varepsilon/8)$ is as small as possible, and so $\|cx_i - cx_j\| = (1 + \varepsilon/8)\|y_i - y_j\|$ is as
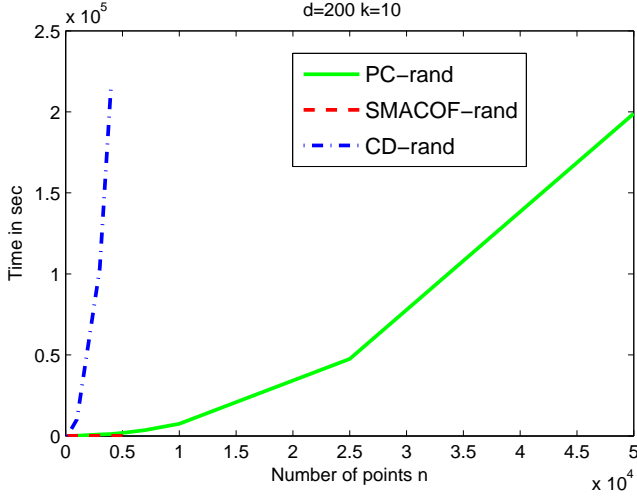
Figure 10: **The behavior of PC, SMACOF and CD for large values of** n**. The curves for CD and SMACOF terminate around** $n = 5000$ **because of memory limitations.**
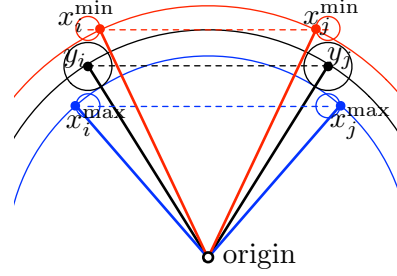

Figure 11: Illustration of the bounds on $\angle_{x_i,x_j}$ when $||y_i - y_j|| \leq 1/2$. The angle $\angle_{x_i^{\max},x_j^{\max}}$ is the largest when $||x_i^{\max}|| = ||x_j^{\max}||$ is as small as possible (lies on inner circle) and $||x_i^{\max} - x_j^{\max}||$ is as large as possible (on the outer edges of the disks of diameter $\varepsilon/8$ shifted down from dashed line of length $||y_i - y_j||$. Bounds for $x_i^{\min}$ and $x_j^{\min}$ are derived symmetrically.

large as possible. See Figure 11. In this case, we have

$$(||cx_i|| + ||cx_j||)\sin(\angle_{x_i,x_j}/2) \leq ||cx_i - cx_j||$$
$$2(1 - \varepsilon/8)\sin(\angle_{x_i,x_j}/2) \leq (1 + \varepsilon/8)||y_i - y_j||$$
$$2(1 - \varepsilon/8)\sin(\angle_{x_i,x_j}/2) \leq (1 + \varepsilon/8)2\sin(\angle_{y_i,y_j}/2)$$
$$\sin(\angle_{x_i,x_j}/2) \leq \frac{1 + \varepsilon/8}{1 - \varepsilon/8}\sin(\angle_{y_i,y_j}/2),$$

which for $\varepsilon < 4$ implies

$$\sin(\angle_{x_i,x_j}/2) \leq (1 + \varepsilon/2)\sin(\angle_{y_i,y_j}/2).$$

Similarly, we can show when $\angle_{x_i,x_j}$ is as small as possible (when $c||x_i|| = c||x_j|| = (1 + \varepsilon)$ and $||cx_i - cx_j|| = (1 - \varepsilon)||y_i - y_j||$), then

$$(1 - \varepsilon/2)\sin(\angle_{y_i,y_j}/2) \leq \sin(\angle_{x_i,x_j}/2).$$

We can also show (via Lemma 6.1) that since $||y_i - y_j|| \leq 1/2$ implies $\angle_{y_i,y_j} < 0.7$ we have

$$\sin((1 - \varepsilon)\angle_{y_i,y_j}) \leq (1 - \varepsilon/2)\sin(\angle_{y_i,y_j})$$

and

$$(1 + \varepsilon/2)\sin(\angle_{y_i,y_j}) \leq \sin((1 + \varepsilon)\angle_{y_i,y_j}).$$

Thus, we have

$$\sin((1 - \varepsilon)\angle_{y_i,y_j}/2) \leq \sin(\angle_{x_i,x_j}/2) \leq \sin((1 + \varepsilon)\angle_{y_i,y_j}/2)$$
$$(1 - \varepsilon)\angle_{y_i,y_j}/2 \leq \angle_{x_i,x_j}/2 \leq (1 + \varepsilon)\angle_{y_i,y_j}/2$$
$$(1 - \varepsilon)\angle_{y_i,y_j} \leq \angle_{x_i,x_j} \leq (1 + \varepsilon)\angle_{y_i,y_j}.$$

Long Case: For $||y_i - y_j|| \in (1/2, 2]$, we consider 6 additional points $y_{i,j}^{(h)} \in \mathbb{S}^{d+1}$ (for $h \in [1 : 6]$) equally spaced between y_i and y_j on the shortest great circle connecting them. Let $\hat{Y}$ be the set Y plus all added points $\{y_{i,j}^{(h)}\}_{h=[1:6]}$. Note that $|\hat{Y}| = O(n^2)$, so by JL we have that

$$(1 - \varepsilon/8)||y_i - \hat{y}_{i,j}|| \leq c||x_i - \hat{x}_{i,j}|| \leq (1 + \varepsilon/8)||y_i - \hat{y}_{i,j}||.$$

For notational convenience let $y_i = y_{i,j}^{(0)}$ and $y_j = y_{i,j}^{(7)}$. Since for $||y_i - y_j|| \in (1/2, 2]$ then $||y_{i,j}^{(h)} - y_{i,j}^{(h+1)}|| \leq 1/2$, for $h \in [0 : 6]$. This follows since the geodesic length of the great circular arc through y_i and y_j is at most π, and $\pi/7 < 1/2$. Then the chordal distance for each pair $||y_{i,j}^{(h)} - y_{i,j}^{(h+1)}||$ is upper bounded by the geodesic distance. Furthermore, by invoking the short case, for any pair

$$(1 - \varepsilon)\angle_{y_{i,j}^{(h)},y_{i,j}^{(h+1)}} \leq \angle_{x_{i,j}^{(h)},x_{i,j}^{(h+1)}} \leq (1 + \varepsilon)\angle_{y_{i,j}^{(h)},y_{i,j}^{(h)}}.$$

Then since projections preserve coplanarity (specifically, the points 0 and $y_{i,j}^{(h)}$ for $h \in [0 : 7]$ are coplanar, hence 0 and $x_{i,j}^{(h)}$ for $h \in [0 : 7]$ are coplanar), we can add together the bounds on angles which all lie on a single great circle.

$$(1 - \varepsilon)\angle_{y_i,y_j} \leq (1 - \varepsilon)\sum_{h=0}^{6}\angle_{y_{i,j}^{(h)},y_{i,j}^{(h+1)}}$$
$$\leq \sum_{h=0}^{6}\angle_{x_{i,j}^{(h)},x_{i,j}^{(h+1)}}$$
$$\leq (1 + \varepsilon)\sum_{h=0}^{6}\angle_{y_{i,j}^{(h)},y_{i,j}^{(h+1)}}$$
$$\leq \min\{\pi, (1 + \varepsilon)\angle_{y_i,y_j}\}$$

and thus by $\angle_{x_i,x_j} = \sum_{h=0}^{6}\angle_{x_{i,j}^{(h)},x_{i,j}^{(h+1)}}$ implies

$$(1 - \varepsilon)\angle_{y_i,y_j} \leq \angle_{x_i,x_j} \leq \min\{\pi, (1 + \varepsilon)\angle_{y_i,y_j}\}. \qquad \square$$

7. REFERENCES

[1] P. K. Agarwal, S. Har-Peled, and H. Yu. On embeddings of moving points in Euclidean space. In *Proceedings 23rd Symposium on Computational Geometry*, 2007.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[3] I. Borg and P. J. F. Groenen. *Modern Multidimensional Scaling*. Springer, 2005.

[4] P. Bose, A. Maheshwari, and P. Morin. Fast approximations for sums of distances, clustering and the Fermat–Weber problem. *Comput. Geom. Theory Appl.*, 24(3):135–146, 2003.

[5] U. Brandes and C. Pich. Eigensolver methods for progressive multidimensional scaling of large data. In *Graph Drawing*, pages 42–53. Springer, 2006.

[6] J. Brimberg and R. F. Love. Global convergence of a generalized iterative procedure for the minisum location problem with ℓ_p distances. *Operations Research*, 41(6):1153–1163, 1993.

[7] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer, 2008.

[8] M. Bronstein. Accelerated MDS. http://tosca.cs.technion.ac.il/book/resources_sw.html.

[9] L. Cayton and S. Dasgupta. Robust Euclidean embedding. In *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*, pages 169–176, 2006.

[10] L. Chen and A. Buja. Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the Americal Statistical Association*, 104:209–219, 2009.

[11] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling, Second Edition*. Chapman & Hall/CRC, September 2000.

[12] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[13] J. de Leeuw. Applications of convex analysis to multidimensional scaling. In J. Barra, F. Brodeau, G. Romier, and B. Van Custem, editors, *Recent Developments in Statistics*, pages 133–146. North Holland Publishing Company, 1977.

[14] J. de Leeuw and P. Mair. Multidimensional scaling using majorization: SMACOF in R. Technical Report 537, UCLA Statistics Preprints Series, 2009.

[15] V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS*, pages 705–712. MIT Press, 2002.

[16] A. E. Elad, Y. Keller, and R. Kimmel. Texture mapping via spherical multi-dimensional scaling. In R. Kimmel, N. A. Sochen, and J. Weickert, editors, *Scale-Space*, volume 3459 of *Lecture Notes in Computer Science*, pages 443–455. Springer, 2005.

[17] C. Faloutsos and K.-I. Lin. Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 163–174, New York, NY, USA, 1995. ACM.

[18] P. T. Fletcher, S. Venkatasubramanian, and S. Joshi. The Geometric Median on Riemannian Manifolds with Application to Robust Atlas Estimation. *Neuroimage (invited to special issue)*, 45(1):S143–S152, March 2009.

[19] R. Gray, A. Buzo, A. Gray Jr, and Y. Matsuyama. Distortion measures for speech processing. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):367–376, Aug 1980.

[20] P. Indyk and J. Matousek. Low-distortion embeddings of finite metric spaces. In *Handbook of Discrete and Computational Geometry*, pages 177–196. CRC Press, 2004.

[21] T. Joachims. *Learning to Classify Text Using Support Vector Machines – Methods, Theory, and Algorithms*. Kluwer/Springer, 2002.

[22] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[23] H. Karcher. Riemannian center of mass and mollifier smoothing. *Comm. on Pure and Appl. Math.*, 30:509–541, 1977.

[24] I. N. Katz. Local convergence in Fermat's problem. *Mathematical Programming*, 6:89–104, 1974.

[25] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[26] J. B. Kruskal and M. Wish. Multidimensional scaling. In E. M. Uslander, editor, *Quantitative Applications in the Social Sciences*, volume 11. Sage Publications, 1978.

[27] H. W. Kuhn. A note on Fermat's problem. *Mathematical Programming*, 4:98–107, 1973.

[28] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2), 2004.

[29] A. Magen. Dimensionality reductions that preserve volumes and distance to affine spaces, and their algorithmic applications. In *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, Lecture Notes In Computer Science; Vol. 2483, 2002.

[30] A. W. Marshall and I. Olkin. *Inequalities: Theory of Majorization and Its Applications*. Academic Press, 1979.

[31] L. M. Ostresh. On the convergence of a class of iterative methods for solving the Weber location problem. *Operations Research*, 26:597–609, 1978.

[32] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190, 1993.

[33] R. Pietersz and P. J. F. Groenen. Rank reduction of correlation matrices by majorization. Technical Report 519086, SSRN, 2004.

[34] R. Pless and I. Simon. Embedding images in non-flat spaces. In *Proc. of the International Conference on Imaging Science, Systems, and Technology*, 2002.

[35] T. Sarlós. Improved approximation algorithms for large matrices via random projections. In *Proceedings 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006.

[36] W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419, 1952.

[37] J. T.-L. Wang, X. Wang, K.-I. Lin, D. Shasha, B. A. Shapiro, and K. Zhang. Evaluating a class of distance-mapping algorithms for data mining and clustering. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 307–311, New York, NY, USA, 1999. ACM.

[38] E. Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Math. J.*, 43:355–386, 1937.

[39] G. Young and A. S. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3:19–22, 1938.