Distance Metric Learning from Pairwise Proximities

Brian McFee
University of California, San Diego
bmcfee@cs.ucsd.edu

Abstract

We compare techniques for embedding a data set into Euclidean space under different notions of proximity constraints.

1 Introduction

The problem of learning a similarity measure between objects arises in many fields and applications. We are given a set of objects and a set of *proximity* constraints over the set, and we must produce a function which mimics the given proximities. Proximity constraints may take on various forms, such as quantitative distance or similarity between two objects, a binary label {*similar*, *not similar*}, or something more elaborate.

To simplify the task, we restrict attention to the class of similarity functions corresponding to distance in *metric spaces*. In the machine learning literature, this problem is commonly referred to as *distance metric learning*. In particular, this paper will focus on algorithms to embed data into the familiar Euclidean space, i.e. $\mathbb{R}^d$ equipped with the ℓ_2-norm.

1.1 Examples

To motivate the discussion, we present a few problems which exemplify different notions of proximity.

Example 1.1. Geographic visualization. Suppose that we want to draw a map containing n major cities in the United States but the geographic coordinates of the cities are unknown. Instead, we have been supplied with a matrix of target distances between every pair of cities. The goal is to find a corresponding arrangement of n points in $\mathbb{R}^2$ such that that for any two cities, the Euclidean distance between their embeddings is as close as possible to the provided target distance.

Example 1.2. Object recognition. Suppose now that we are building a visual object recognition system. It may not be clear what the correct notion of "distance" between two images is, but we can work with qualitative information. For example, we could present a human with two images and ask if both contain the same object (similar) or not (dissimilar). The goal in this case is to find an embedding of images into Euclidean space such that pairs labeled *similar* are mapped near one-another, and pairs labeled *dissimilar* are mapped far apart from each-other. Note that the quality of the resulting embedding here depends heavily on the criteria used to determine similarity after embedding.

Example 1.3. Acoustic similarity. Finally, suppose that we want to build a music recommendation system. We have a set of songs and we wish to discover a similarity structure among them. For this problem, similarity itself is a subjective concept, so humans may not reliably produce quantitative similarity scores, or even similar/not-similar labels. However, humans may be more likely to agree on whether a pair $\{i, j\}$ is more similar than another pair $\{k, \ell\}$. The goal now is to embed each song into Euclidean space such that for all measured proximities (i, j, k, ℓ), the distance between i and j is smaller than the distance between k and ℓ.

1.2 Preliminaries

Before we proceed further, we must first establish some notation and mathematical background.

Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ denote a set of n arbitrary objects. In some settings, $x_i \in \mathcal{X}$ may have high-dimensional, real vector representations, which will be explicitly indicated by $\mathcal{X} \subset \mathbb{R}^D$. Let $g : \mathcal{X} \to \mathbb{R}^d$ represent the function which maps objects into d-dimensional Euclidean space. When g is a linear projection, it will be denoted by the matrix $G \in \mathbb{R}^{d \times D}$.

A function $\delta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ is a *metric* if it satisfies the following properties for all $x, y, z \in \mathcal{X}$:

- uniqueness: $(\delta(x, y) = 0) \Leftrightarrow (x = y)$.
- symmetry: $\delta(x, y) = \delta(y, x)$,
- triangle inequality: $\delta(x, y) + \delta(y, z) \geq \delta(x, z)$,

We will be specifically interested in the Euclidean metric composed with g:

$$\begin{aligned}\delta(x, y) &= \|g(x) - g(y)\| \\ &= \sqrt{(g(x) - g(y))^\mathsf{T}(g(x) - g(y))}.\end{aligned}$$

A symmetric matrix $A \in \mathbb{R}^{n \times n}$ has a spectral decomposition, denoted $A = V \Lambda V^\mathsf{T}$, where Λ is a diagonal matrix consisting of the eigenvalues λ_i of A and the columns of V are the corresponding eigenvectors v_i. We will assume that eigenvalues (and corresponding eigenvectors) are sorted in descending order: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. A symmetric matrix A is *positive semi-definite* (PSD), denoted $A \succeq 0$, if each of its eigenvalues is non-negative.

For a square matrix A, let $\mathrm{Tr}(A)$ denote the trace: $\mathrm{Tr}(A) = \sum_i A_{ii}$. For a matrix $B \in \mathbb{R}^{m \times n}$, let $\|B\| = \sqrt{\mathrm{Tr}\,(B^\mathsf{T} B)}$ denote the Frobenius norm of B. Finally, for a vector x and a PSD matrix M, let $\|x\|_M = \sqrt{x^\mathsf{T} M x}$.

1.3 Outline

The rest of this paper is organized as follows. Section 2 presents an overview of multi-dimensional scaling. Section 3 presents algorithms which embed data subject to discrete proximity constraints. Section 4 discusses some recent advances in partial-ordering-constrained embedding. Finally, we conclude in Section 5 and discuss directions for future work.

2 Multi-dimensional scaling

Imagine an experiment where a human is presented with pairs of objects, and asked to choose a number between 1 and 10 indicating the degree of dissimilarity between the objects (1 being very similar and 10 being very dissimilar). Assuming that an object is always perfectly self-similar, the outcome of this experiment can be expressed as a symmetric matrix Δ of non-negative entries, with zeros on the diagonal. Δ satisfies the first two properties of a metric but not necessarily the triangle inequality.

MDS algorithms take as input a matrix Δ as described above and produce an embedding of the data set which approximately reproduces Δ by Euclidean distance. MDS has been applied to a wide variety of problems in many disciplines, ranging from computer science to psychology [8]. While thorough review of MDS applications lies well outside the scope of this paper, we will briefly examine the basic setup and techniques. For a substantial treatment of MDS techniques and applications, see Borg and Groenen [8].

MDS algorithms come in two basic varieties: metric and non-metric. Metric MDS algorithms seek an embedding which quantitatively preserves the given dissimilarities. Non-metric MDS is concerned only with preserving the rank-order of given dissimilarities, under the assumption that the quantitative information may be unreliable or irrelevant. As a special case of metric MDS, we will first examine classical MDS, originally due to Torgerson [38] and Gower [21].

2.1 Classical MDS

The classical MDS algorithm was designed for the special case where a dissimilarity matrix Δ arises from squared Euclidean distances between points in a set $\mathcal{X}$: $\Delta_{ij} = \|x_i - x_j\|^2$ for some unknown vectors x_i, x_j. The key observation behind the algorithm is that a Euclidean distance matrix Δ gives rise to a PSD matrix

$$A_\Delta = -\frac{1}{2} H_n \Delta H_n, \tag{1}$$

where H_n is the $n \times n$ centering matrix (see Appendix A for a proof of this correspondence) [8]. Since $A_\Delta \succeq 0$, it can be factored by spectral decomposition to recover a

Algorithm 2.1 Classical multi-dimensional scaling.

ClassicalMDS (Δ, d)

1. $A_\Delta \leftarrow -\frac{1}{2} H_n \Delta H_n$.
2. Compute the spectral decomposition $A_\Delta = V \Lambda V^\mathsf{T}$.
3. Let Λ^+ be diagonal with $\Lambda^+_{ii} = \max(0, \Lambda_{ii})$.
4. Return the first d columns of $X = V(\Lambda^+)^{1/2}$.

matrix X:

$$A_\Delta = V \Lambda V^\mathsf{T} = (V \Lambda^{1/2})(V \Lambda^{1/2})^\mathsf{T} = X X^\mathsf{T},$$

where each row X_i contains the embedding $g(x_i) = X_i$. The recovered configuration is centered around the origin, and exactly generates Δ. However, for real-world data, Δ will not generally derive from Euclidean distances. It follows from Theorem A.1 that A_Δ will not be PSD, so an exact configuration cannot by recovered by this method.

In order to recover a Euclidean embedding from a non-Euclidean Δ, A_Δ is projected onto the cone of PSD matrices by first computing its spectral decomposition and then replacing all negative eigenvalues with 0. This is equivalent to solving the minimization

$$\min_{A \succeq 0} \|A - A_\Delta\|. \tag{2}$$

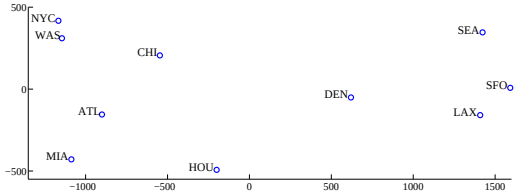
Factoring the A which achieves this minimum yields a configuration of the points which in some sense approximates the original distances Δ.

Many applications, especially visualization, require a low-dimensional representation of $\mathcal{X}$. Since the embedding is recovered by spectral decomposition, simply truncating X to the first d columns provides the orthogonal projection of X onto $\mathbb{R}^d$. This is equivalent to performing principal components analysis (PCA) on the recovered X. The full classical MDS procedure is summarized in Algorithm 2.1. Figure 1 illustrates the results of the classical MDS algorithm for the U.S. cities data set of example 1.1.

It has been observed that, because the centering transformation (1) was defined only for Euclidean distance matrices, classical MDS is not equivalent to directly optimizing over distance matrices, i.e. (1) and (2) do not preserve optimality of X [10]. Furthermore, the dimensionality reduction step may collapse the very distances that we were attempting to preserve, so the d-dimensional embedding may be sub-optimal for the original problem.

	ATL	CHI	DEN	HOU	LAX	MIA	NYC	SFO	SEA	WAS
ATL	0	587	1212	701	1936	604	748	2139	2182	543
CHI	587	0	920	940	1745	1188	713	1858	1737	597
DEN	1212	920	0	879	831	1726	1631	949	1021	1494
HOU	701	940	879	0	1374	968	1420	1645	1891	1220
LAX	1936	1745	831	1374	0	2339	2451	347	959	2300
MIA	604	1188	1726	968	2339	0	1092	2594	2734	923
NYC	748	713	1631	1420	2451	1092	0	2571	2408	205
SFO	2139	1858	949	1645	347	2594	2571	0	678	2442
SEA	2182	1737	1021	1891	959	2734	2408	678	0	2329
WAS	543	597	1494	1220	2300	923	205	2442	2329	0

(a) The dissimilarity matrix Δ for ten U.S. cities.



(b) The resulting 2-dimensional embedding after classical MDS.

Figure 1: An example of classical MDS on the U.S. cities data set of example 1.1.

2.2 Metric and non-metric MDS

Metric MDS generalizes classical MDS setting by allowing alternative cost functions and optimizing directly over the choice of embedding. This freedom allows the practitioner to specify the target dimension d and supply weights $w_{ij} \geq 0$ to dissimilarity measurements, shifting the focus of the optimization to certain pairs of interest. By setting weights to 0, the algorithm can operate with incomplete measurements, an important feature for large data sets where measuring all-pairs distances becomes infeasible. Moreover, the cost function may be augmented in any number of ways to incorporate available side-information, something not easily achieved in the classical framework. However, this increased flexibility often complicates the problem by introducing local optima to the objective function.

Metric MDS algorithms are typically formulated to minimize the *stress*of the embedding, derived from the

sum-squared-error of distances:

$$\sigma(X) = \sum_{i<j} w_{ij}(\Delta_{ij} - \|X_i - X_j\|)^2. \qquad (3)$$

When the dimensionality is fixed, stress is not convex in X, making it susceptible to local minima and difficult to optimize. Numerous techniques exist to find locally optimal solutions, primarily based on either gradient descent or majorization, where a convex upper-bound is minimized as a surrogate for stress (see [8] for a detailed treatment of stress majorization). Glunt et al. developed an alternating projection algorithm to find the closest Euclidean distance matrix D to a given Δ (in the squared-error sense), but it cannot constrain the rank of the solution because the embedding is recovered by performing classical MDS on D [19].

Generalizing the problem further, we could consider replacing Δ_{ij} in (3) by some transformation $\Delta_{ij} \mapsto f(i,j)$. In some cases, a suitable f may be constructed from domain knowledge. Going back to Example 1.1, the between-city distance measurements are not Euclidean distances in $\mathbb{R}^2$; they are geodesic distances along a spherical surface in $\mathbb{R}^3$. If this were known a priori, we could construct an f which corrects the measured dissimilarities to more closely resemble Euclidean distances.

Non-metric MDS algorithms exploit this idea by automatically constructing f to be non-decreasing with the rank order of the original dissimilarities Δ, thereby ignoring the quantitative information. More concretely, f should be chosen to map pairs of objects into a set of distances, while preserving order:

$$f(i,j) \leq f(k,\ell) \Leftrightarrow \Delta_{ij} \leq \Delta_{k\ell}.$$

Kruskal devised an algorithm to simultaneously optimize both X and f by isotonic regression [24]. The isotonic regression works by first fixing an embedding X and then sorting the distances $\|X_i - X_j\|$ in ascending order of dissimilarity Δ_{ij}. This permuted list of distances is then broken into maximal contiguous blocks, where the (weighted) average distance for one block cannot exceed that of the next. Finally, $f(i,j)$ maps to the average distance over the block containing (i,j) (see Figure 2). For a fixed choice of X, this construction of f provably minimizes $\sigma(X)$ [13].

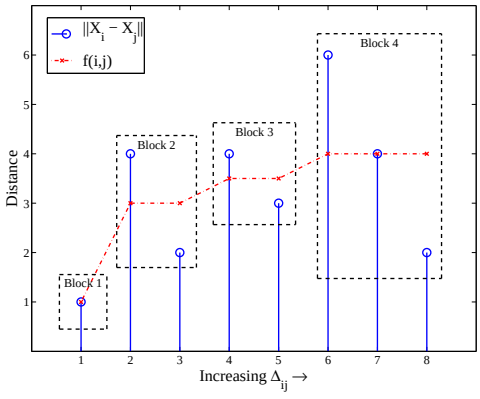


Figure 2: An example of isotonic regression of f over an embedding X and dissimilarities Δ. Each point on the x-axis corresponds to a pair (i,j).

The combined optimization is carried out by randomly initializing X, and then alternately solving for f and updating X by gradient descent. The algorithm halts upon achieving a local minimum. Kruskal's non-metric MDS procedure is summarized in Algorithm 2.2.

Non-metric MDS preserves qualitative information, but it requires a total ordering of all $\binom{n}{2}$ pairs from $\mathcal{X}$, typically provided by quantitative measurements in Δ. In many problem domains, obtaining this much quantitative information may not be feasible. It may be more natural to directly support qualitative measurements as input, eliminating the need for (temporary) quantitative measurements. This extension is discussed in Section 4.

2.3 Modern MDS extensions

Due to its relative simplicity, classical MDS has been the starting point for most recent developments in MDS algorithms. We list here a brief overview of some modern work on MDS algorithms.

Laub and Müller demonstrated that data obtained from human subjects exhibits useful information in the negative part of A_Δ's spectrum [25]. Classical MDS discards this information (step 3 of Algorithm 2.1). Rather than eliminating negative eigenvalues, the *constant-shift embedding* algorithm produces a PSD matrix by applying the rule $\lambda_i \mapsto \lambda_i - \lambda_n$, which can be interpreted as correcting

Algorithm 2.2 Kruskal's non-metric MDS algorithm. d is the desired dimensionality, and α is a step-size for the gradient descent. $\sigma(X)$ is the stress function (3), augmented by $\Delta_{ij} \mapsto f(i,j)$.

NMDS(Δ, d, α)

1. Initialize $X \in \mathbb{R}^{n \times d}$ randomly and center X at the origin.
2. Repeat until convergence of X:
   (a) Compute f by isotonic regression on (X, Δ).
   (b) Update $X \leftarrow X - \alpha(\nabla\sigma)_X$ and re-center.
3. Return X.

Δ for violations of the triangle inequality [32]. For large $|\lambda_n|$, the matrix may have much higher rank than the result of classical MDS, and it may not be suitable for visualization. The resulting matrix is not the closest to A_Δ in the least-squares sense, but it does preserve certain qualitative structures in the dissimilarities, which are discussed further in Section 4.

Cayton and Dasgupta modified the stress function (3) of metric MDS to penalize the absolute error of the embedding (the ℓ_1 cost) instead of the usual sum-squared error [10]. This modification allows the problem to be formulated as a semidefinite program [9]. This choice of cost function makes the algorithm less sensitive to outliers in the dissimilarity matrix, so unlike classical MDS, a small number of corrupted entries of Δ do not corrupt the global structure of the embedding. Moreover, they show that achieving a low-cost embedding into $\mathbb{R}^1$ is NP-Hard for a general class of cost functions, including absolute error (ℓ_1) and stress (ℓ_2).

Manifold learning is closely related to MDS. The Isomap algorithm uses classical MDS as a sub-routine to embed points based on approximate geodesic distances along a manifold [37]. Maximum variance unfolding (MVU) operates directly on a given distance matrix, and can be viewed as a weighted MDS algorithm which seeks to maximize the sum of embedded distances, while preserving local neighborhood distances exactly (i.e. isometrically) [43]. Larger distances are interpreted as inaccurate and thus unconstrained in the optimization problem. The sum over all distances is maximized (subject to local isometry constraints), resulting in a flattening of the data into a low-dimensional space. Although the strict local isometry constraints do not apply for the general dissimilarities under consideration here, it is straightforward to relax the constraints by least-squares approximation (see e.g. [44]).

For large data sets, the spectral decomposition step of classical MDS can be prohibitively expensive. Several algorithms have been proposed to approximate classical MDS on large data sets, including FastMap, MetricMap, and Landmark MDS [16, 41, 14]. These three algorithms were later unified by showing that they all apply the Nyström method to approximate the spectral decomposition, albeit in slightly different ways [45, 31]. For Landmark MDS — the most recent and conceptually simplest of the three — the approximation works by performing classical MDS on a small, square sub-matrix of Δ (the distances between *landmarks*) and mapping each remaining point by a linear combination of the embedded landmarks.

Finally, Bengio et al. constructed an out-of-sample extension to classical MDS [5]. Their extension applies the Nyström method by interpreting the spectral decomposition of A_Δ as an empirical estimate of the eigenfunctions of an operator whose kernel function h generates A_Δ: $h(i,j) = (A_\Delta)_{ij}$. Using the eigenvectors and eigenvalues of A_Δ, an approximation to h can be recovered as a linear combination of the eigenfunctions. An unseen point x can then be inserted into an existing embedding, provided the distances from x to all other points in $\mathcal{X}$.

2.4 Metric embeddings

The theoretical computer science community has devoted a great deal of attention to metric embeddings, i.e. the case where Δ arises from a (not necessarily Euclidean) metric space. Data derived from human measurements rarely meets this criterion, so the algorithms developed for metric embedding are not generally applicable. Instead, the metric embedding literature focuses on theoretical results and developing approximation algorithms for combinatorial problems. For a basic introduction to this body of work, see chapter 15 of the book by Matoušek [29] and the survey by Indyk [23].

Although algorithms developed for metric embedding cannot be applied directly for non-metric Δ, certain neg-

ative results can apply, including lower bounds on target dimensionality and distortion of distances (see e.g. [26] or [28]).

3 Discrete proximity

In many problem domains, there may not exist a well-defined, quantitative notion of distance or similarity between two objects. It is therefore not reasonable to expect humans to reliably produce numerical similarity scores. This often arises in situations where distance is used as a surrogate for more coarse notions of similarity, e.g. class membership. With this in mind, we can revise our experimental procedure. Instead of asking for similarity score, we can simply request a binary *similar/dissimilar* label for each pair of items, resulting in a set of discrete proximity constraints.

Approaches to dealing with discrete proximities fall roughly into two categories. The first approach treats a similarity constraint $\{i, j\}$ as evidence that i and j belong to the same class or cluster. The second approach treats proximities as indicating that i and j should map to the same local neighborhood. The fundamental difference between these two notions lies in the assumption of transitivity of the similarity relation. For this reason, we will refer to algorithms based on the first notion as *equivalence class methods*.

3.1 Equivalence class methods

Equivalence class methods often assume that $\mathcal{X}$ should form clusters under the learned metric, and that the given proximity constraints reflect this desired clustering. It is therefore not surprising that many of these algorithms have been engineered specifically for use with (or adapted from) k-means clustering [27, 36, 7].

3.1.1 Relevant Components Analysis

One of the first algorithms to incorporate distance metric learning with clustering was Relevant Components Analysis (RCA) [36, 3]. RCA was developed for a setting where the entire data set has been corrupted by some global, high-variance additive noise. For example, in a face recognition task, lighting conditions may account for the majority of the variance but contribute little useful information for class discrimination. If the variance of the noise process significantly exceeds within-class variances, it may confuse standard dimensionality-reduction algorithms such as PCA [4]. By using subsets of the data known to belong to the same class, the noise can be estimated and subsequently corrected. Note that this is similar in spirit to linear discriminant analysis, but differs in that RCA does not require prior knowledge of the true class labels.

Formally, the RCA algorithm is given a set $\mathcal{X} \subset \mathbb{R}^D$ and a set $\mathcal{C}_+ = \{c_1, c_2, \ldots c_m : c_i \subseteq \mathcal{X}\}$ of pairwise-disjoint *chunklets*. Each chunklet c_i is assumed to lie entirely within some (unknown) class, and several may belong to the same class. The global noise process is assumed to be normally distributed as $\mathcal{N}(0, S_N)$. Each cluster is also assumed to be normally distributed with small variance relative to S_N. The goal is to learn a linear projection $G \in \mathbb{R}^{d \times D}$ which compresses distances along the directions of irrelevant variance, i.e. the principal directions of S_N.

If true class labels of the data were available, S_N could be directly estimated from the within-class scatter:

$$S_N \approx S_w = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{k} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^\mathsf{T},$$

where C_i and μ_i denote the i^{th} class and its mean. However, since class labels are unknown, we approximate S_w by using chunklets:

$$S_w \approx M = \left(\sum_{c_i \in \mathcal{C}} |c_i| \right)^{-1} \sum_{c_i \in \mathcal{C}} \sum_{x \in c_i} (x - \mu_i')(x - \mu_i')^\mathsf{T}, \tag{4}$$

where c_i and μ_i' now denote the i^{th} chunklet and its mean. Note that when each $\mathcal{C}_+$ partitions $\mathcal{X}$, the discrepancy between S_w and M only depends on how well each μ_i' approximates that of its containing class. Therefore, a more densely-sampled $\mathcal{C}_+$ leads to better approximations of S_w and S_N.

Because $M \succeq 0$, it can be factored: $M = V \Lambda V^\mathsf{T}$. Defining $G = M^{-\frac{1}{2}} = \Lambda^{-\frac{1}{2}} V^\mathsf{T}$ yields a projection $x \mapsto Gx$ which effectively re-scales the data so that the noise contributes equally in all directions[1] (i.e. G is a

[1] Inverting Λ may lead to numerical instability if M is (effectively)

Algorithm 3.1 Relevant Components Analysis (RCA) [36].

RCA($\mathcal{X}, \mathcal{C}_+, d$)

1. Compute M by (4).
2. Compute the spectral decomposition $M = V\Lambda V^\mathsf{T}$.
3. Let Λ_d denote the $d \times d$ upper-left block of Λ, and let V_d denote the first d columns of V.
4. Return $G = \Lambda_d^{-1/2} V_d^\mathsf{T}$.

whitening transformation). The complete RCA algorithm is given in Algorithm 3.1.

The practical applications of RCA are limited for two reasons. First, classes which are separated along directions parallel to the noise may be collapsed onto eachother. Second, the assumption that S_N dominates the variance of each cluster may be too restrictive for many applications. Furthermore, RCA cannot make use of dissimilarity constraints. Incorporating dissimilarity information may yield a more flexible algorithm.

3.1.2 Metric Pairwise-constrained k-Means

Proximity constraints can be used by RCA as a preprocessing step for clustering applications, but the constraints would be ignored by any off-the-shelf clustering algorithm. Directly integrating proximity constraint knowledge with a clustering algorithm may yield better results. This approach was first taken by Wagstaff et al., where the update step of the k-Means algorithm was modified so that cluster assignments strictly adhere to given *must-link* and *cannot-link* constraints [40]. Their greedy cluster-assignment rule was fundamentally flawed, owing to the fact that the constraints may not even be satisfiable and it is NP-Hard to minimize constraint violations in such cases [2]. More flexible algorithms have been subsequently developed, many of which include metric learning [7, 17, 47].

Bilenko et al. extend the constrained k-Means algorithm in two significant ways [7]. First, rather than giving up when a satisfying assignment cannot be found, the cost function is modified to penalize violated constraints. This does not ensure a globally optimal solution, but it is certainly better than aborting the algorithm. The second major change from [40] is that the algorithm fits a local distance metric for each cluster.

Formalizing the Metric Pairwise-constrained k-Means algorithm (MPCk-Means), the input is a set $\mathcal{X} \subset \mathbb{R}^D$, sets $\mathcal{C}_+, \mathcal{C}_- \subseteq \mathcal{X}^2$ containing similar and dissimilar pairs, and a positive integer k denoting the number of desired clusters. For each cluster $i \in 1 \ldots k$, the algorithm will output a mean μ_i and PSD matrix M_i defining the cluster's metric: $\|x - y\|_{M_i}$.

Let $N(x)$ denote the cluster to which x is assigned. We first define a cost function f_+ for violations of $\{i, j\} \in \mathcal{C}_+$:

$$f_+(i, j) = \frac{1}{2}\left(\|i - j\|^2_{M_{N(i)}} + \|i - j\|^2_{M_{N(j)}}\right),$$

that is, the average of the squared distances under each offending cluster's metric.

Next, we define a cost function f_- for violations of $\{i, j\} \in \mathcal{C}_-$:

$$f_-(i, j) = \max_{(a,b)\in\mathcal{X}^2} \|a - b\|^2_{M_{N(i)}} - \|i - j\|^2_{M_{N(i)}},$$

Intuitively, this penalty is proportional to the discrepancy between the distance from i to j and the diameter of $\mathcal{X}$ according to the offending cluster's metric.

Finally, the cluster-assignment component of the cost is defined as:

$$f(\mathcal{X}) = \sum_{x\in\mathcal{X}} \left(\|x - \mu_{N(x)}\|^2_{M_{N(x)}} - \log\det M_{N(x)}\right),$$

where the second term in the summation arises due to the normalization constants in the related mixture of Gaussians problem. Here, $-\log\det M_i$ can be interpreted as a approximating a low-rank penalty, thus preventing M_i from collapsing points to achieve a low-cost embedding. Putting it all together, we obtain the complete cost func-

rank-deficient. The authors resolve this by computing G from only the top d eigenvalues/vectors of M [36].

tion:

$$\begin{aligned}\text{Cost}(\mathcal{X}, \mu_{1\ldots k}, M_{1\ldots k}) = f(\mathcal{X}) \\ + \sum_{\{i,j\}\in\mathcal{C}_+} \mathbb{1}_{N(i)\neq N(j)} f_+(i,j) \\ + \sum_{\{i,j\}\in\mathcal{C}_-} \mathbb{1}_{N(i)=N(j)} f_-(i,j). \quad (5)\end{aligned}$$

The resulting algorithm strongly resembles the standard formulation of learning a mixture of Gaussians via expectation-maximization (E-M) [15]. The key difference here lies in the distinction between hard and soft clustering. In the mixture of Gaussians case, each data point generates a penalty derived from a weighted sum of its likelihood under each mixture component. However, in the hard clustering problem, each point is assigned to exactly one cluster and that alone defines the point's contribution to the cost. This distinction is crucial: the decision to assign a point to a given cluster impacts the global cost function both directly, through its distance from the mean, and indirectly, through the constraint violation penalties.

Since the cost function is not convex, the authors designed a randomized, greedy algorithm to achieve a local optimum, summarized in Algorithm 3.2[2]. The randomization step removes any bias due to the initial ordering of $\mathcal{X}$. However, like any E-M algorithm, it may still suffer from local optima.

Since Algorithm 3.2 adapts a metric to each cluster, it should automatically compensate for global noise, such as in the setup of RCA. This link can be made more explicit by constraining the algorithm to learn a single distance metric for all clusters. (This comparison to RCA is supported by the experimental results of [7].) Furthermore, MPCk-Means does not require proximity measurements, but it can make use of both similarities and dissimilarities if they are available. In this sense, MPCk-Means generalizes RCA (with k-Means) for clustering applications. However, the algorithm is significantly more complex than either RCA or k-Means. Because it constructs several (high-dimensional) embeddings, it does not immediately lend itself to visualization applications.

[2]The cost function at step 2b is based on the partial calculation of constraint penalties generated by those points which have already been assigned in the current iteration.

Algorithm 3.2 The Metric Pairwise-constrained k-Means algorithm [7].

MPCk-Means $(\mathcal{X}, \mathcal{C}_+, \mathcal{C}_-, k)$

1. Compute the transitive closure of $\mathcal{C}_+$ and use it to initialize cluster centers. Initialize each $M_i \leftarrow I_D$.
2. Repeat until convergence:
   (a) Randomly permute $\mathcal{X}$.
   (b) For each $x \in \mathcal{X}$, assign x to the cluster which minimizes the cost (5).
   (c) Update each μ_i according to the new point assignments.
   (d) Update each M_i: solve $\dfrac{\partial}{\partial M_i}\text{Cost} = 0$.
3. Return $\mu_{1\ldots k}, M_{1\ldots k}$.

3.2 Neighbor-preserving methods

In some settings, it is unreasonable to assume that the data will form coherent clusters, but we may still wish to embed the data for visualization, classification, or neighbor search applications. Neighbor relations are not generally transitive, so it does not make sense to apply the equivalence class methods of Section 3.1. An algorithm developed in the neighbor-preserving framework may still perform well in clustering applications, but the underlying assumptions are less restrictive.

3.2.1 A convex optimization approach

Xing et al. developed a metric learning algorithm which directly optimizes distances subject to the given constraint set [46]. The algorithm is given $\mathcal{X} \subset \mathbb{R}^D$, and proximity constraints $\mathcal{C}_+, \mathcal{C}_- \subseteq \mathcal{X}^2$. The goal is to learn a global linear projection G such that the average similar-point distance does not exceed a fixed constant and the average dissimilar-point distance is maximized. If we let $M = G^\mathsf{T}G$, this can be formulated as a convex optimization problem, given in Algorithm 3.3.

Note that the similarity constraint is expressed in terms of squared distance (linear in M), but the objective is not. This was specifically engineered to avoid a trivial rank-1

Algorithm 3.3 The distance metric learning procedure of Xing et al. [46]

$$\begin{aligned} \max_{M} \quad & f(M) = \sum_{\mathcal{C}_-} \|i-j\|_M \\ \text{s.t.} \quad & \sum_{\mathcal{C}_+} \|i-j\|_M^2 \leq 1 \\ & M \succeq 0. \end{aligned}$$

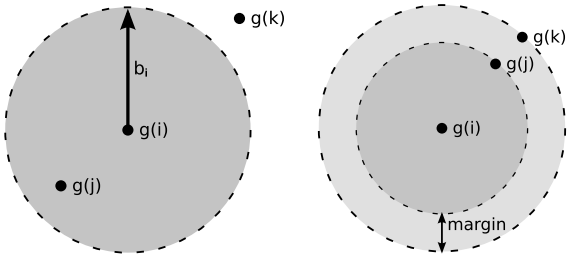
LearnMetric$(\mathcal{X}, \mathcal{C}_+, \mathcal{C}_-)$

1. Initialize M.
2. Repeat until convergence:
   (a) Repeat until convergence:
      i. $M \leftarrow \text{argmin}_{M'} \|M' - M\|^2$ s.t. $\sum_{\mathcal{C}_+} \|i-j\|_{M'}^2 \leq 1$
      ii. $M \leftarrow \text{argmin}_{M'} \|M' - M\|^2$ s.t. $M' \succeq 0$
   (b) $M \leftarrow M + \alpha(\nabla f)_M$
3. Compute the spectral decomposition $M = V\Lambda V^\mathsf{T}$.
4. Return $G = \Lambda^{\frac{1}{2}} V^\mathsf{T}$.

solution, but it also leads to an simple optimization procedure.

To efficiently optimize the objective function, the authors implemented an alternating projection algorithm. This begins with M randomly initialized, alternately projecting M onto the feasible region governed by the two constraints, and then updating M by gradient ascent on the objective function (see Algorithm 3.3). Since the objective function is convex, this algorithm is guaranteed to converge to the correct solution.

The similarity constraint in Algorithm 3.3 only requires that similar points lie (on average) within a fixed radius $\left(|\mathcal{C}_+|^{-1/2}\right)$ of each-other. The constraint does *not* force dissimilar points to exceed this radius: it simply makes a best-effort attempt by maximizing the objective function. The learned metric, therefore, may not effectively distinguish the similar pairs from the dissimilar. In effect, the algorithm as formulated above may be too conservative. By sacrificing a small number of constraints, a better solution may become achievable.



(a) The ball constraint of Globerson and Roweis [18] and Shalev-Shwartz et al. [35]

(b) The large margin constraint of Weinberger et al. [42].

Figure 3: Two different notions of a neighborhood margin constraint: $\{i,j\} \in \mathcal{C}_+$, $\{i,k\} \in \mathcal{C}_-$. The shaded region represents the neighborhood of point i.

3.2.2 Margin methods

In nearest-neighbor search and classification applications, it is crucial that a learned metric map dissimilar pairs *significantly* farther away than similar pairs. Several margin-based methods have been developed to explicitly address this problem [39, 35, 42, 18].

With an eye toward visualization applications, Globerson and Roweis formulated a semi-definite program (SDP) for embedding with margin constraints [18]. The key intuition behind their algorithm is that for each $i \in \mathcal{X}$, all points similar to i should map to a tight neighborhood around i and all dissimilar points should map far away from that neighborhood.

Formally, the algorithm is given a set $\mathcal{X}$ and constraints $\mathcal{C}_+, \mathcal{C}_- \subseteq \mathcal{X}^2$. The goal is to learn a mapping g such that for each $i \in \mathcal{X}$, all similar points $j : \{i,j\} \in \mathcal{C}_+$ are constrained to lie within a ball of radius $b_i \geq 0$, and all dissimilar points $k : \{i,k\} \in \mathcal{C}_-$ are embedded with distance at least b_i (see Figure 3a).

To construct the SDP, let Y be a matrix where the i^{th} row contains the embedded coordinates of the i^{th} element of $\mathcal{X}$ (i.e. $Y_i = g(x_i)$). Defining $A = YY^\mathsf{T}$, we can express squared Euclidean distance between embedded points i, j as a linear function of A:

$$\begin{aligned} \|g(i) - g(j)\|^2 &= \|Y_i - Y_j\|^2 \\ &= (Y_i - Y_j)(Y_i - Y_j)^\mathsf{T} \\ &= A_{ii} + A_{jj} - 2A_{ij}. \end{aligned}$$

As in MDS, the embedding is centered at the origin to account for shift-invariance in the distance function. Unlike MDS, no sense of scale is provided to the algorithm (e.g. from quantitative distances). Since any scaling of $\beta Y, \beta > 0$ will satisfy the proximity constraints as well as Y, the embedding is constrained by the sum of the norms of embedded points, yielding the constraints:

$$\sum_i A_{ii} = \mathrm{Tr}(A) \leq 1 \text{ and } \sum_{ij} A_{ij} = 0.$$

Finally, since the constraint sets may not be satisfiable, we introduce slack parameters $\xi_{ij} \geq 0$ for each margin constraint and add a hinge-loss penalty to the objective. A trade-off parameter γ allows the user to tune the contribution to the total cost due to constraint violations. The final program is given in Algorithm 3.4.

The algorithm above was designed for the case where $\mathcal{X}$ is a general set. However, if $\mathcal{X} \subset \mathbb{R}^D$, the problem can be reformulated to learn a linear projection G of the original points. Let $X \in \mathbb{R}^{n \times D}$ be the matrix where row i is the i^{th} element of $\mathcal{X}$. By choosing an appropriate regularization function of G — i.e. one that is monotonic in $\|G\|$, such as $\|G\|^2$ — we can apply the *representer theorem* and see that the optimal solution can be expressed as a linear combination of points in $\mathcal{X}$: $G = WX$ [33].

This new formulation (K-PSDE) outputs a projection matrix $W \in \mathbb{R}^{d \times D}$ and can therefore map previously unseen (out-of-sample) points into the learned metric space. Since both the optimization procedure and the learned mapping function G only depend on the inner-products between points (and not their vectorial representation), we can apply the *kernel trick* and replace the inner-product matrix with one calculated from a kernel function [30]. This allows the algorithm to construct a mapping from high-dimensional (perhaps infinite) space, to a low-dimensional Euclidean space. The derivation of K-PSDE is given in Appendix B. Note that unlike the Nyström approximation methods described in Section 2.3, K-PSDE only requires evaluation of the kernel function on the new point and no additional proximity constraints. The out-of-sample extension is therefore automatic.

Algorithms using similar intuitions have been designed for classification applications. Shalev-Shwartz et al. developed an online metric learning algorithm where the goal is to predict whether a pair of vectors $\{i, j\} \in \mathcal{X}^2$ should be classified as *similar* or *dissimilar*, effectively reducing the metric learning to a binary classification problem [35]. Their algorithm uses a single radius b as a threshold on learned distance to make a prediction, but otherwise the geometric intuition is quite similar to that of Algorithm 3.4. Weinberger et al. formulate a similar margin constraint, but the margin is instead forced between the *difference* of similar- and dissimilar-pair distances (see Figure 3b) [42]. This type of constraint attempts to push dissimilar points to a safe distance away from the neighborhood. When $\mathcal{C}_+$ is constructed to contain only similarly labeled pairs, the learned metric strengthens the performance of nearest-neighbor classification.

Algorithm 3.4 The Pairwise Semi-definite Embedding (PSDE) program of Globerson and Roweis [18]. s_{ij} is $+1$ if $\{i, j\} \in \mathcal{C}_+$ and -1 if $\{i, j\} \in \mathcal{C}_-$, and $\mathcal{C}_\pm = \mathcal{C}_+ \cup \mathcal{C}_-$.

PSDE($\mathcal{X}, \mathcal{C}_+, \mathcal{C}_-, \gamma$)

$$\begin{aligned}
\min_{A,b,\xi} \quad & \sum_{\mathcal{C}_\pm} s_{ij} \left(A_{ii} + A_{jj} - 2A_{ij}\right) && + \gamma \sum_{\mathcal{C}_\pm} \xi_{ij} \\
\text{s.t.} \quad & & & \\
\forall_{\{i,j\} \in \mathcal{C}_\pm} \quad & s_{ij} \left(A_{ii} + A_{jj} - 2A_{ij}\right) && \leq s_{ij} b_i + \xi_{ij} \\
\forall_{i \in \mathcal{X}} \quad & b_i && \geq 0 \\
\forall_{\{i,j\} \in \mathcal{C}_\pm} \quad & \xi_{ij} && \geq 0 \\
& \mathrm{Tr}(A) && \leq 1 \\
& \sum_{i,j} A_{ij} && = 0 \\
& A && \succeq 0
\end{aligned}$$

3.3 Related methods

Some other recent developments in the discrete proximity setting include the works of Goldberger et al. [20], Yang et al. [47], and Cook et al. [11]. These methods take a probabilistic approach, learning distance metrics which maximize the likelihood of choosing similar points under a stochastic neighbor-selection rule. Hadsell et al. approach the problem by constructing a convolutional network to learn the embedding [22]. The objective functions in these algorithms are non-convex and difficult to optimize, and often include several free parameters which must be manually tuned by the practitioner.

4 Partial ordering

For some types of data, especially subjective media such as music, an objective notion of similarity may not be well-defined. Recall that the discrete proximity setting arose because humans may not reliably produce quantitative measurements of similarity. Similarly, two people may have different internal thresholds for what constitutes a "similar" pair. Revisiting Example 1.3, it may be possible to obtain a robust set of proximity measurements by asking if a pair of objects $\{i, j\} \subset \mathcal{X}^2$ is more similar than a pair $\{k, \ell\}$, generating the constraint set:

$$\mathcal{C} = \{(i, j, k, \ell) : \|i - j\| < \|k - \ell\|\}.$$

Barring inconsistencies, $\mathcal{C}$ can be represented as a directed acyclic graph (DAG) over pairs of items, perfectly capturing the notion of a *partial ordering*. Even though this notation allows for inconsistencies, any derived embedding must satisfy ordering structure, so we will refer to this as the partial ordering framework.

The partial ordering framework is closely related to non-metric MDS, described in Section 2.2. Recall that non-metric MDS seeks an embedding which preserves the rank ordering of dissimilarities Δ_{ij}. Since Δ_{ij} are real-valued, there is an implicit assumption of total ordering, i.e. every pair of items is comparable. The present framework relaxes the requirements to a (not necessarily complete) partial ordering of similarities. This removes the dependence on any (albeit temporary) quantitative measurements, thereby simplifying the evaluation task for humans.

In fact, the partial ordering framework strictly generalizes non-metric MDS. Agarwal et al. prove that the constant-shift embedding procedure described in Section 2.3 perfectly preserves the ordering defined by any dissimilarity matrix Δ [1]. Clearly, any Δ can be translated into an equivalent constraint set by taking all $O(n^2)$ adjacent pairs from the sorted list of dissimilarities. The generalization is strict in the sense that $\mathcal{C}$ can encode constraints that cannot be represented by any real-valued dissimilarity matrix Δ. However, such a $\mathcal{C}$ cannot be a partial ordering, since any partial order can be reduced to the non-metric case (see Appendix C). Still, the flexibility of this representation allows us to identify and explicitly address any structural violations.

The partial ordering framework also generalizes discrete proximities. Given $\mathcal{C}_+$ and $\mathcal{C}_-$, we can construct a partial ordering constraint set $\mathcal{C}$ from the Cartesian product $\mathcal{C}_+ \times \mathcal{C}_-$, yielding

$$\mathcal{C} = \{(i, j, k, \ell) : (i, j) \in \mathcal{C}_+, (k, \ell) \in \mathcal{C}_-\}.$$

Assuming no pair belongs to both $\mathcal{C}_+$ and $\mathcal{C}_-$, this reduction yields a bipartite DAG. Note that in the absence of dissimilarity constraints, additional assumptions must be made for the constraints to carry any semantic meaning.

Since the partial ordering framework has only recently emerged, relatively few algorithms have been designed within this setting. We present two of them here.

4.1 A support-vector algorithm

Schultz and Joachims constructed a metric learning algorithm for the present framework, motivated by the following text-mining application [34]. A user queries a search engine and is presented with a ranked list of documents. Any two documents (i, j) which the user decides to open are assumed to be more similar than a pair consisting of one opened and one un-opened document, (i, k). It may not be the case that (i, k) are objectively dissimilar, since they were both ranked highly in the search results. The goal is to learn a linear projection, given some features of the documents $\mathcal{X} \subset \mathbb{R}^D$ and a set of proximity constraints $\mathcal{C} = \{(i, j, i, k)\}$.

To simplify the task, the metric is restricted to have the form $\|x - y\|_{YWY^\intercal}$, where Y is a fixed linear projection and W is diagonal with non-negative entries. (This is equivalent to learning a linear projection $G = W^{1/2}Y^\mathsf{T}$.) If $Y = I_D$, this corresponds to learning a weighting $\sqrt{W_{ii}}$ over the features of $\mathcal{X}$. It is also possible to learn a distance metric in a feature space induced by some mapping $\phi(\cdot)$. By defining the column $Y_i = \phi(x_i)$, the distance $\|\phi(x) - \phi(x)\|_{YWY^\intercal}$ can be expressed as

$$\begin{aligned}\|\phi(x) - \phi(y)\|_{YWY^\intercal} &= \left\|Y^\mathsf{T}(\phi(x) - \phi(y))\right\|_W \\ &= \sum_{i=1}^{n} W_{ii} \left(h(x, x_i) - h(y, x_i)\right)^2\end{aligned}$$

where $h(\cdot, \cdot)$ is the kernel induced by $\phi(\cdot)$, with Gram matrix $H = Y^\mathsf{T}Y$ over $\mathcal{X}$. From this, we can see that W defines a weighting of the training examples $x_i \in \mathcal{X}$, and

Algorithm 4.1 The quadratic program of Schultz and Joachims [34].

SVEmbed$(Z, L, \mathcal{C}, \gamma)$

$$\begin{array}{lll} \min\limits_{w,\xi} & w^\mathsf{T} L w + \gamma \sum\limits_{\mathcal{C}} \xi_{ijk} & \\ \text{s. t.} & & \\ \forall_{(i,j,i,k)\in\mathcal{C}} & w^\mathsf{T}(Z(i,k) - Z(i,j)) & \geq 1 - \xi_{ijk} \\ \forall_{(i,j,i,k)\in\mathcal{C}} & \xi_{ijk} & \geq 0 \\ \forall_{i\in 1\ldots n} & w_i & \geq 0 \end{array}$$

the examples with large weight can be interpreted as landmarks.

For each proximity constraint $(i, j, i, k) \in \mathcal{C}$, we constrain the embedding by forcing a unit margin between the (squared) distances:

$$\|Y^\mathsf{T}(\phi(x_i) - \phi(x_k))\|_W^2 \geq 1 + \|Y^\mathsf{T}(\phi(x_i) - \phi(x_j))\|_W^2. \quad (6)$$

If we then define $Z(i, j)$ as the vector with p^{th} coordinate

$$Z(i,j)_p = (H_{ip} - H_{jp})^2,$$

and let w be the vector of diagonal entries of W, the margin constraint (6) can be expressed as

$$w^\mathsf{T}(Z(i,k) - Z(i,j)) \geq 1,$$

which is linear in w. (Slack variables are incorporated to compensate for unsatisfiable constraints.)

To achieve a low-rank embedding, the authors follow the work of Tsang and Kwok and penalize the ℓ_2-norm[3] of the eigenvalues of YWY^T (equivalently, penalizing $\|YWY^\mathsf{T}\|^2$) [39]. Since W is diagonal, we can define a matrix $L = H * H \succeq 0$, where $*$ denotes element-wise multiplication, so that $\|YWY^\mathsf{T}\|^2 = w^\mathsf{T} L w$. This objective is quadratic in w and the constraints are linear, so the final result is a quadratic program, summarized as Algorithm 4.1.

The program listed above can be solved efficiently, but this formulation relies on W being diagonal. In some applications, we may desire more flexibility in the learned metric. It would be straightforward revise the algorithm to optimize over M (or non-diagonal $W \succeq 0$ for the kernelized version) as well, thereby optimizing over linear combinations of features. However, such a modification would result in a semidefinite program, instead of the comparatively simpler quadratic program.

[3]The choice of ℓ_2 regularization in Algorithm 4.1 seems sub-optimal. Since $YWY^\mathsf{T} \succeq 0$, using the trace would correspond to ℓ_1 regularization, provide a tighter approximation to a rank penalty, and yield a simple linear program. However, using ℓ_2 yields a program that more closely resembles the standard soft-margin SVM formulation [12].

4.2 Generalized non-metric MDS

Agarwal et al. developed an embedding algorithm for partial order constraints for $\mathcal{X} \nsubseteq \mathbb{R}^D$, generalizing non-metric MDS [1]. They formulate the problem as a semidefinite program over a matrix $A = XX^\mathsf{T} \succeq 0$, which can then be factored by spectral decomposition to recover the embedded coordinates.

By now, the ingredients of the algorithm are all familiar. We retain the margin constraint (6) and center the embedding around the origin as in Algorithm 3.4. However, instead of explicitly bounding the scale of the embedding, we minimize the sum of (squared) norms of the embedded points, which can be expressed simply as $\mathrm{Tr}(A)$. Moreover, the trace provides a convex approximation to the rank, so the algorithm tends to prefer low-dimensional solutions. Introducing slack variables for the constraints yields the final SDP, Algorithm 4.2.

The algorithm, as presented above, is much more flexible than the original non-metric MDS formulation (Algorithm 2.2). Because it does not rely on a fixed total ordering of the distances, it is free to re-order distances (subject to $\mathcal{C}$) in a way that directly optimizes the objective. Furthermore, introducing slack to the constraints allows the algorithm to operate in the presence of noisy constraints — i.e. $\mathcal{C}$ that is not a partial ordering — or sacrifice certain constraints to achieve a better embedding; these enhancements were not readily achievable with Algorithm 2.2.

If we want to extend the embedding for unseen points, we might imagine applying the two techniques discussed in Sections 2.3 and 3.2. Because A is not a simple function of $\mathcal{X}$ (unlike classical MDS), there is no clear way to derive an appropriate kernel function for use with the Nyström method. However, the method used to extend Algorithm 3.4 (Appendix B) can be applied quite easily to Algorithm 4.2. At present, this extension has not been tested.

Solving the problem above requires optimizing a function of $O(n^4)$ variables over $O(n^4)$ constraints, which

Algorithm 4.2 Generalized non-metric MDS [1].

GNMDS$(\mathcal{X}, \mathcal{C}, \gamma)$

$$
\begin{array}{lll}
\min\limits_{A,\xi} & \operatorname{Tr}(A) + \gamma \sum\limits_{\mathcal{C}} \xi_{ijk\ell} & \\
\text{s. t.} & & \\
\forall_{(i,j,k,\ell)\in\mathcal{C}} & (A_{kk} - 2A_{k\ell} + A_{\ell\ell}) & \\
& \quad - (A_{ii} - 2A_{ij} + A_{jj}) & \geq 1 - \xi_{ijk\ell} \\
\forall_{(i,j,k,\ell)\in\mathcal{C}} & \xi_{ijk\ell} & \geq 0 \\
& \sum\limits_{i,j} A_{ij} & = 0 \\
& A & \succeq 0
\end{array}
$$

is simply not feasible with general purpose solvers on moderately large data sets. Algorithm 4.1, by comparison, only requires a quadratic optimization over a much smaller set of variables and can therefore scale up to large data sets more easily. To improve the scalability of Algorithm 4.2, the authors implemented an alternating projection algorithm, similar to the one described in Algorithm 3.3. They demonstrate the results of this algorithm on a set of 55 objects and 15000 human-generated constraints, which, although not a huge data set, is certainly bigger than general purpose SDP solvers can currently support for this program.

4.3 Obtaining constraints

The partial ordering framework provides a rich language for describing constraints and effectively generalizes both non-metric MDS and discrete proximities in certain contexts. However, there are practical issues which must be resolved before implementing a partial ordering method. The most pressing issue is the acquisition of suitable constraints over a data set.

The alternative frameworks surveyed in Sections 2 and 3 have comparatively long histories and numerous data sets have accumulated over time. It is also relatively straightforward to generate constraint sets in the discrete proximity setting, by taking either similarly labeled pairs or nearest neighbors. Moreover, both MDS and discrete proximities only require measurements between pairs of points, so the constraints can be completely specified with $O(n^2)$ measurements.

Partial ordering, on the other hand, is relatively new, and data sets are few and far between. The reduction from discrete proximities to partial ordering given in the introduction to this section serves mainly as a conceptual device and would not likely produce a sufficiently structured constraint set for practical applications. (An empirical study of the loss incurred by this reduction on real data sets would be quite interesting.) Furthermore, partial ordering constraints are given over *pairs of pairs*, requiring $O(n^4)$ measurements ($O(n^3)$ if the pairs overlap) for a complete specification.

If partial ordering constraints are generated by humans, the natural experimental setup is the *triadic comparison* model: are $\{i, j\}$ more similar than $\{i, k\}$? Unfortunately, human subjects consider triadic comparisons to be the most boring and fatiguing type of experiment, suggesting that the quality of constraints generated in this fashion may decay as the experiment progresses [6]. Factoring in the volume of measurements required for a well-structured constraint set only exacerbates the problem. However, it may be possible to exploit metric properties (i.e. the triangle inequality) and the transitive nature of partial orderings to obtain a relatively sparse constraint set with little appreciable loss of quality.

5 Conclusion

In this survey, we examined three distinct proximity constraint frameworks for distance metric learning, and examined several algorithms developed within each framework.

The correct choice of proximity framework depends heavily on the problem at hand. For applications that only require the preservation of qualitative information (e.g. preserving nearest neighbors), partial ordering is clearly the most general, but least-developed framework. Applications involving clustering or classification may fit most naturally with discrete proximities, but additional modeling assumptions (e.g. Gaussian clusters or ball constraints) are often required. Finally, MDS methods should only be applied when the quantitative dissimilarity matrix is reasonably close to Euclidean.

5.1 Future work

There are several possible directions for future work in this field. The following represent just a few ideas that may prove to be especially useful in applications involving subjective data.

One potentially useful extension to algorithms 3.4, 4.1, and 4.2 (after out-of-sample extension) would be to allow not just one kernel function, but a mixture of kernels. Revisiting the song similarity example (1.3), one may imagine several potential features which contribute to similarity, e.g. rhythm, timbre, pitch, or semantic information, but the relative weights of each type of feature are unknown and may vary from song to song. It is straightforward to see how this can be incorporated to Algorithm 4.1: simply extend the Z vectors in blocks according to each kernel. A similar approach could work for the SDP-based algorithms 3.4 and 4.2, but the increase in computational overhead would be much more severe. Still, such an extension could be extremely valuable for visualizing feature-rich multimedia data sets.

Another direction for future work is to develop an efficient method for gathering partial order constraints. As mentioned in Section 4.3, obtaining constraints in the partial order framework can be significantly more expensive than in other frameworks. The naïve method serially presents humans with a randomly chosen triadic comparison. However, assuming that there is some true underlying partial order structure to the constraint set we're trying to obtain, there may exist a more intelligent sampling algorithm which can adapt to this structure as it is uncovered and avoids querying for constraints that could be deduced from present knowledge (i.e. those which are implied by the partial order assumption and metric properties). Not every edge in the underlying DAG must be discovered, only a sufficient subset to define all ancestor-descendent relations. Designing an efficient adaptive sampling algorithm for this problem could be of great practical significance, as well as an interesting theoretical challenge.

Finally, in certain applications, it may be advantageous to relax the symmetry requirement of proximity measurements. This arises in concatenative synthesis applications, where a proximity (i, j) may relate to how well j follows i in a sequence (e.g. a musical playlist). In these situations, the assumption of symmetry may not be justified. At present, it is not clear how to effectively incorporate this relaxation while still retaining the mathematical conveniences afforded by Euclidean (or general metric) embedding.

Acknowledgements

The author would like to thank his research exam committee: Sanjoy Dasgupta, Lawrence Saul, Dean Tullsen, and Gert Lanckriet.

References

[1] Sameer Agarwal, Joshua Wills, Lawrence Cayton, Gert Lanckriet, David Kriegman, and Serge Belongie. Generalized non-metric multi-dimensional scaling. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, 2007.

[2] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56(1–3):89–113, July 2004.

[3] Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall. Learning distance functions using equivalence relations. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[4] Peter N. Belhumeur, João P. Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projections. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(7):711–720, July 1997.

[5] Yoshua Bengio, Jean-François Paiement, Pascal Vincent, Olivier Delalleau, Nicolas Le Roux, and Marie Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. 2004.

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

[7] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

[8] Ingwer Borg and Patrick J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer-Verlag, second edition, 2005.

[9] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[10] Lawrence Cayton and Sanjoy Dasgupta. Robust euclidean embedding. In *Proceedings of the Twenty-third International Conference on Machine Learning*, 2006.

[11] James Cook, Ilya Sutskever, Andriy Mnih, and Geoff Hinton. Visualizing pairwise similarity with a mixture of maps. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, 2007.

[12] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[13] Trevor F. Cox and Michael A.A. Cox. *Multidimensional Scaling*. Chapman and Hall, 1994.

[14] Vin de Silva and Joshua B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 705–712, Cambridge, MA, 2003. MIT Press.

[15] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, 2001.

[16] Christos Faloutsos and King-Ip Lin. Fastmap: a fast algorithm for indexing, data-mining and visualization. In *Proceedings of ACM SIGMOD*, pages 163–174, 1995.

[17] Amir Globerson and Sam Roweis. Metric learning by collapsing classes. In Yair Weiss, Bernhard Schölkopf, and John Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 451–458, Cambridge, MA, 2006. MIT Press.

[18] Amir Globerson and Sam Roweis. Visualizing pairwise similarity via semidefinite embedding. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, 2007.

[19] W. Glunt, T.L. Hayden, S. Hong, and J. Wells. An alternating projection algorithm for computing the nearest euclidean distance matrix. *SIAM Journal on Matrix Analysis and Applications*, 11(4):589–600, October 1990.

[20] Jacob Goldberger, Sam Roweis, Geoffrey Hinton, and Ruslan Salakhutdinov. Neighborhood components analysis. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 513–520, Cambridge, MA, 2005. MIT Press.

[21] J.C. Gower. Some distance properties of latent root and vector methods in multivariate analysis. *Biometrika*, 53:325–338, 1966.

[22] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer Vision and Pattern Recognition*, 2006.

[23] Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Annual Symposium on Foundations of Computer Science*, volume 42, pages 10–35, 2001.

[24] J.B. Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2), June 1964.

[25] Julian Laub and Klaus-Robert Müller. Feature discovery in non-metric pairwise data. *Journal of Machine Learning Research*, 5(2):801–818, 2004.

[26] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:215–245, 1995.

[27] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[28] Jiří Matoušek. Bi-lipschitz embeddings into low-dimensional euclidean spaces. *Commentationes Mathematicae Universitatis Carolinae*, 31(3):589–600, 1990.

[29] Jiří Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag, 2002.

[30] Klaus-Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, March 2001.

[31] John C. Platt. Fastmap, metricmap, and landmark mds are all nyström algorithms. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 261–268, 2005.

[32] Volker Roth, Julian Laub, Joachim M. Buhmann, and Klaus-Robert Müller. Going metric: denoising pairwise data. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 809–816, Cambridge, MA, 2003. MIT Press.

[33] Bernhard Schölkopf, Ralf Herbrich, Alex J. Smola, and Robert Williamson. A generalized representer theorem. In *Proceedings of the 14th Annual Conference on Computational Learning Theory*, pages 416–426, 2001.

[34] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

[35] Shai Shalev-Shwartz, Yoram Singer, and Andrew Y. Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

[36] Noam Shental, Tomer Hertz, Daphna Weinshall, and Misha Pavel. Adjustment learning and relevant components analysis. In *European Conference on Computer Vision*, 2002.

[37] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2322, December 2000.

[38] W.S. Torgerson. Multidimensional scaling: 1. theory and method. *Psychometrika*, 17:401–419, 1952.

[39] Ivor W. Tsang and James T. Kwok. Distance metric learning with kernels. In *International Conference on Artificial Neural Networks*, 2003.

[40] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schroedl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

[41] Jason T.L. Wang, Xiong Wang, King-Ip Lin, Dennis Shasha, Bruce A. Shapiro, and Kaizhong Zhang. Evaluating a class of distance-mapping algorithms for data mining and clustering. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 307–311, 1999.

[42] Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. In Yair Weiss, Bernhard Schölkopf, and John Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 451–458, Cambridge, MA, 2006. MIT Press.

[43] Kilian Q. Weinberger, Fei Sha, and Lawrence K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the Twenty-first International Conference on Machine Learning*, pages 839–846, 2004.

[44] Kilian Q. Weinberger, Fei Sha, Qihui Zhu, and Lawrence K. Saul. Graph laplacian regularization for large-scale semidefinite programming. In *Advances in Neural Information Processing Systems*,

number 19, pages 1489–1496, Cambridge, MA, 2007. MIT Press.

[45] Christopher K.I. Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. pages 682–688, 2001.

[46] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning, with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 505–512, Cambridge, MA, 2003. MIT Press.

[47] Liu Yang, Rong Jin, Rahul Sukthankar, and Yi Liu. An efficient algorithm for local distance metric learning. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, pages 543–548, 2006.

Appendix

A Correspondence of EDM and PSD

The correspondence between the set of (squared) Euclidean distance matrices ($\mathbb{EDM}$) and PSD matrices is well known. The proof here is adapted from [9] and [10].

Theorem A.1. *A distance matrix $\Delta \in \mathbb{EDM}$ if and only if $-\frac{1}{2}H_n\Delta H_n \succeq 0$.*

Proof. ($\Rightarrow$) Let $\Delta \in \mathbb{EDM}$. Let $X \in \mathbb{R}^{n\times d}$ denote a configuration of the points, centered around the origin, which generates Δ. Let X_i denote the i^{th} row of X and let y be the vector of squared norms: $y_i = X_i X_i^\mathsf{T}$. Then

$$\Delta = y\mathbf{1}^\mathsf{T} + \mathbf{1}y^\mathsf{T} - 2XX^\mathsf{T}.$$

Now, multiply both sides of the equation on the left and right by the centering matrix $H_n = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\mathsf{T}$:

$$\begin{aligned} H_n\Delta H_n &= H_n y\mathbf{1}^\mathsf{T}H_n + H_n\mathbf{1}y^\mathsf{T}H_n - 2H_nXX^\mathsf{T}H_n \\ &= -2H_nXX^\mathsf{T}H_n^\mathsf{T}. \end{aligned}$$

The second equality follows from symmetry of H_n and the fact that $(\mathbf{1}^\mathsf{T}H_n)^\mathsf{T} = H_n\mathbf{1} = \mathbf{0}$. By hypothesis, X is already centered, so $H_nX = X$. This yields the desired result:

$$-\frac{1}{2}H_n\Delta H_n = XX^\mathsf{T} \succeq 0.$$

($\Leftarrow$) Let Δ be a dissimilarity matrix such that $-\frac{1}{2}H_n\Delta H_n = A \succeq 0$, with $A = XX^\mathsf{T}$. We can express each A_{ij} as

$$A_{ij} = \Delta_{ij} - \frac{1}{n}\Delta_i^\mathsf{T}\mathbf{1} - \frac{1}{n}\mathbf{1}^\mathsf{T}\Delta_j + \frac{1}{n^2}\mathbf{1}^\mathsf{T}\Delta\mathbf{1}.$$

Then, plugging back into $\|X_i - X_j\|^2 = A_{ii} + A_{jj} - 2A_{ij}$ yields Δ_{ij}, so $\Delta \in \mathbb{EDM}$. □

B Derivation of Kernel Pairwise Semi-definite Embedding (K-PSDE)

This derivation is adapted directly from [18]. Recall that we are learning a matrix A corresponding to a linear projection G: $A_{ij} = i^\mathsf{T}G^\mathsf{T}Gj$, where $G = WX$. Then $A_{ij} = i^\mathsf{T}X^\mathsf{T}W^\mathsf{T}WXj$. Since X is fixed, we will optimize over $Q = W^\mathsf{T}W$ with the constraint $Q \succeq 0$.

Let $K = XX^\mathsf{T}$, i.e. the Gram matrix of $\mathcal{X}$. Then the regularization term $\|G\|^2$ can be expressed in terms of Q:

$$\begin{aligned} \|G\|^2 = \operatorname{Tr}(GG^\mathsf{T}) &= \operatorname{Tr}(WXX^\mathsf{T}W^\mathsf{T}) \\ &= \operatorname{Tr}(W^\mathsf{T}WXX^\mathsf{T}) = \operatorname{Tr}(QK). \end{aligned}$$

Similarly, we can replace all A_{ij} in Algorithm 3.4 by

$$A_{ij} = i^\mathsf{T}G^\mathsf{T}Gj = i^\mathsf{T}X^\mathsf{T}QXj = K_i^\mathsf{T}QK_j,$$

which is still linear in Q.

Introducing a trade-off parameter $\beta > 0$ and adding $\beta\operatorname{Tr}(QK)$ to the objective gives the final K-PSDE program. A low-dimensional projection can be recovered by spectral decomposition of $G^\mathsf{T}G$. When K is constructed from a kernel function, this cannot be done explicitly, but it can be approximated by the kernel PCA method used in [17].

C Partial order reduces to non-metric MDS

Lemma C.1. *Let $\mathcal{C}$ be any partial ordering over pairs of objects from $\mathcal{X}$. Then there exists a dissimilarity matrix Δ such that $(i, j, k, \ell) \in \mathcal{C} \rightarrow \Delta_{ij} < \Delta_{k\ell}$.*

Proof. Since $\mathcal{C}$ is a partial ordering, it has a DAG representation $\mathcal{G} = (V, E)$. Let

$$F \subseteq \{f : V \rightarrow \{1, 2, \ldots, |V|\}\}$$

denote the set of all topological orderings of $\mathcal{G}$. Now, we can define Δ as follows:

$$\Delta_{ij} = \begin{cases} 0 & i = j \\ \min\limits_{f \in F} f\{i, j\} & \{i, j\} \in V \\ |V| + 1 & \{i, j\} \notin V. \end{cases}$$

Clearly, Δ satisfies the definition of a dissimilarity matrix. Now, for any $(i, j, k, \ell) \in \mathcal{C}$, there exists a path from $\{i, j\}$ to $\{k, \ell\}$ in $\mathcal{G}$. Let f be the ordering which minimizes Δ_{ij} and let f' minimize $\Delta_{k\ell}$. Then,

$$\Delta_{ij} = f\{i, j\} \leq f'\{i, j\} < f'\{k, \ell\} = \Delta_{k\ell}. \qquad \square$$

Theorem C.2. *Every partial ordering over dissimilarities between n objects can be embedded into $\mathbb{R}^{n-1}$.*

Proof. Combining Lemma C.1 with the constant-shift embedding result from [1] shows that any partial ordering $\mathcal{C}$ over pairs of n objects can be embedded into $\mathbb{R}^{n-1}$ such that all order constraints are satisfied by Euclidean distance. $\square$