7 EMBEDDING AND PARAMETERIZATION

In the previous chapter, we explored the *forward problem* of computing distances between points given a representation of a geometric object. Now, we consider the *inverse problem* of going from distances to a geometric representation.

While this description might appear abstract, this problem appears across the computational literature, from graph theory to data science to mesh processing. Variations of this problem appear under many names in many disciplines:

- Dimensionality reduction: One motivation for using tools like principal component analysis (PCA) for dimensionality reduction is that we suspect data to embed well in a lower-dimensional space; dimensionality reduction arguably is successful when the geometry of high-dimensional data is preserved well when we project onto the (unknown) lower-dimensional space.

- Manifold learning: A common schematic in machine learning is to think of a cloud of data points as cutting out a relatively low-dimensional—but curved—submanifold of the large-dimensional space in which they are collected. Implicitly or explicitly discovering the manifold on which the data lives can help design sensible learning procedures that propagate labels along the manifold instead of diffusing into extrinsic space.

- Data visualization: The majority (arguably the whole) of display technology—from computer screens to 3D printers—is limited to two or three dimensions. Visualization tools intended to help data scientists seek patterns in unexplored datasets must embed as much information as possible from different types of data into $\mathbb{R}^2$ or $\mathbb{R}^3$.

- Representation learning: A popular topic in deep learning, representation learning is built on the observation that the space in which your data is embedded is not necessarily the best space for describing its structure. For example, distances between photographs measured in pixel color space is less meaningful for many computer vision applications than distances that take into account image content. In representation learning, the task is to learn a map from data points into a space that makes their relevant relationships explicit. Recent results in this discipline are surprising: One well-known paper even computes an embedding of (a piece of) the English language dictionary into $\mathbb{R}^n$ that preserves semantic structure CITE.

- Parameterization: Applications like 3D modeling and medical imaging store data *over* a wide variety of shapes. As a classic example, 3D surfaces in graphics and computer-aided design are often accompanied by dense textures storing color or reflectance per point at a density higher than the number of vertices on a mesh. In these cases, it may be useful to parameterize a shape to a canonical domain on which we can accelerate information storage. In surface texturing, a common practice is to map surfaces into pieces of the plane so that we can store textures in 2D photograph format, capable of relatively high information density. In these applications, we seek a parameterization into the image plane that distorts surface geometry as little as possible.

7.1 METRIC SPACES

There are many notions of *distance*, some of which we have encountered in previous chapters. Perhaps the most familiar example is distance in Euclidean space $\mathbb{R}^n$, $d(\mathbf{x}, \mathbf{y}) := \|x - y\|_2$. In the previous chapter, we defined distances between points on a submanifold of $\mathbb{R}^n$, using the arc

lengths of curves as a proxy for distances. But many other notions of distances exist. For example, as mentioned in § REF, the distance between nodes on a graph is well-defined so long as the edge weights are positive. And sometimes our notions of distance are somewhat fuzzy: In the English dictionary, we might think of words like *red*, *green*, and *blue* as somehow closer to one another than *red* and *banana*. Note there still is some distance-like triangle inequality in our final example: We might think of *yellow* as roughly at a midpoint between *banana* and *red*, since it is both a color and a reasonable descriptor for *banana*.

A key object of study in mathematics attempts to formalize what it means to be a distance. Enshrining some of the mathematical intuition built from the examples above, we have the following definition:

Definition 7.1 (Metric space)**.** *A* metric space *is a pair* (M, d) *where* M *is a set and* $d(\cdot,\cdot) : M \times M \to \mathbb{R}$ *is a function satisfying the following axioms for all* $x, y, z \in M$*:*

$$\begin{aligned} d(x,y) &\geq 0 && \textit{Positivity} \\ d(x,y) = 0 &\iff x = y && \textit{Indiscernibility} \\ d(x,y) &= d(y,x) && \textit{Symmetry} \\ d(x,z) &\leq d(x,y) + d(y,z) && \textit{Triangle inequality.} \end{aligned}$$

This definition is quite flexible and can be used to interpret the same set more than one way. For instance, suppose M is a surface embedded in $\mathbb{R}^3$. There are two interpretations of M as a metric space that are *not* the same: M could be equipped with the extrinsic metric $d(\mathbf{x}, \mathbf{y}) := \|x - y\|_2$ from ambient space $\mathbb{R}^3$, or M could be equipped with the metric $d(\mathbf{x}, \mathbf{y})$ that gives the geodesic distance from $\mathbf{x}$ to $\mathbf{y}$ along M.

Metric spaces need not even be finite dimensional:

Example 7.1 (C^∞ as a metric space)**.** *The set of infinitely-differentiable functions* $C^\infty([0,1])$ *on the unit interval* $[0,1]$ *is a metric space when equipped with the distance*

$$d(f,g)^2 := \int_0^1 (f(t) - g(t))^2 \, dt.$$

7.2 EMBEDDING A METRIC SPACE

As we can see above, the notion of a metric space is incredibly broad and encapsulates a huge set of example space of assorted sizes and/or dimensions. Some metric spaces, like subsets of $\mathbb{R}^2$, are easily visualized and understood, while others, like high-dimensional manifolds of data, are less easily digested. In the latter case, it can be beneficial to try to squeeze a complicated metric space into one we are more used to processing, while preserving as much structure as possible. This process is often called *embedding*.

Suppose (M, d) and (M', d') are two metric spaces. We can think of an embedding as a map $\phi : M \to M'$. In this case, the strongest condition we can put onto ϕ is that it completely preserves metric space structure:

Definition 7.2 (Isometry)**.** *A map* $\phi : M \to M'$ *between metric spaces* (M, d) *and* (M', d') *is an* isometry *if it preserves pairwise distances. That is, for all* $x, y \in M$ *we have*

$$d(x,y) = d'(\phi(x), \phi(y)).$$

If there exists a bijective isometry between two metric spaces, we call those spaces isometric*.*

Example 7.2 (Isometry). *Suppose $G = (V, E)$ is the graph with three vertices connected together into a triangle with edge lengths equal to 1. This is a* discrete *metric space when equipped with the shortest-path metric along graph edges. This space embeds isometrically into $\mathbb{R}^2$ by taking*

$$\begin{aligned}\phi(v_1) &= (0, 0)\\ \phi(v_2) &= (0, 1)\\ \phi(v_3) &= (1/2, \sqrt{3}/2).\end{aligned}$$

The condition that one space embeds isometrically into another is strong, and a priori there is no reason to expect an isometry to exist given an arbitrary pair of spaces (M, d) and (M', d'). Considerable mathematical effort goes into characterizing the *embedding capacity* of assorted metric spaces (M, d), that is, the likelihood that another metric space (M', d') embeds into (M, d) isometrically.

Disappointingly, there are plenty of useful spaces that do not embed isometrically into our favorite space computationally, Euclidean space $\mathbb{R}^n$. Our counterexample consists of a four-point metric space CITE:

Example 7.3 (Embedding into $\mathbb{R}^n$). *Consider the four-element space $M = \{a, b, c, d\}$ with equipped with the distance metric d defined via:*

$$\begin{aligned}d(a, b) = d(a, c) = d(b, c) &= 2\\ d(a, d) = d(b, d) &= 1\\ d(c, d) &= 1.5.\end{aligned}$$

Since this space is finite, we can check manually that (M, d) satisfies the criteria in Definition 7.1 to be a metric space.

We cannot, however, embed this space into any $\mathbb{R}^n$. To see this, notice that (a, b, c) would form an equilateral triangle of edge length 2, from the first set of conditions on d. From the second, the point b must be on the midpoint from a to b. As shown in Figure REF, in $\mathbb{R}^n$ this would constrain $d(c, d)$ to be the height of the triangle, which is not 1.5.

Different spaces have more embedding capacity than $\mathbb{R}^n$ for any n. Following CITE, as an example in the opposite direction consider the space $\ell_\infty(\mathbb{R}^n)$, defined to be $\mathbb{R}^n$ equipped with the norm

$$\|\mathbf{x}\|_\infty := \max_k |\mathbf{x}_k|.$$

We can show the following:

Proposition 7.1. *Every finite space embeds isometrically into $\ell_\infty(\mathbb{R}^n)$ for some n.*

Proof. Take a finite metric space (M, d) where $M = \{a_1, a_2, \ldots, a_n\}$. Define a map $\phi : M \to \mathbb{R}^n$ as follows:

$$\phi(a_k) := (d(a_1, a_k), d(a_2, a_k), \ldots, d(a_n, a_k)).$$

To check that this map is an isometry, we need to verify that distances are preserved:

$$\|\phi(a_i) - \phi(a_j)\|_\infty = \max_k |d(a_k, a_i) - d(a_k, a_j)|.$$

We will bound this quantity above and below by $d(a_i, a_j)$. Taking $k = i$ in the max above shows $\|\phi(a_i) - \phi(a_j)\|_\infty \geq d(a_i, a_j)$, since $d(a_i, a_i) = 0$. To show the reverse, we can apply the triangle inequality:

$$\begin{aligned}d(a_i, a_j) + d(a_j, a_k) \geq d(a_i, a_k) &\implies d(a_k, a_i) - d(a_k, a_j) \leq d(a_i, a_j)\\ d(a_i, a_j) + d(a_i, a_k) \geq d(a_j, a_k) &\implies d(a_k, a_j) - d(a_k, a_i) \leq d(a_i, a_j)\\ &\implies |d(a_k, a_i) - d(a_k, a_j)| \leq d(a_i, a_j)\end{aligned}$$

Hence,

$$\begin{aligned}\max_k |d(a_k, a_i) - d(a_k, a_j)| &\leq \max_k d(a_i, a_j) \\ &= d(a_i, a_j),\end{aligned}$$

as needed. □

The proof above even extends to infinite-sized metric spaces so long as we suitably extend the definition of ℓ_∞. This proof in some sense shows that $\ell_\infty(\mathbb{R}^n)$ has better embedding capacity than $\mathbb{R}^n$ equipped with the standard Euclidean metric. But before we begin replacing all our applications of Euclidean space with ℓ_∞, we should note the limited utility of Proposition 7.1: Once we embed a space into ℓ_∞, it is not clear that it is any easier to process or visualize, and the dimensionality of the embedding equals the number of points in the space.

7.3 APPROXIMATE EMBEDDING

Given the negative results above, it appears we should relax the isometry condition when embedding spaces into one another and instead seek an *approximate* isometry. This leads to a key topic that will appear and reappear in our discussion of different geometry problems: Constructing a measure of *distortion* of a map ϕ, in this case of deviation from isometry.

A few standard distortion measures appear in the embedding literature. Given a map $\phi : M \to M'$, some standard measures of distortion are CITE:

$$\begin{aligned}\text{expansion}(\phi) &:= \sup_{x,y \in M} \frac{d'(\phi(x), \phi(y))}{d(x,y)} \\ \text{contraction}(\phi) &:= \sup_{x,y \in M} \frac{d(x,y)}{d'(\phi(x), \phi(y))} \\ \text{distortion}(\phi) &:= \text{expansion}(\phi) \cdot \text{contraction}(\phi).\end{aligned} \tag{7.1}$$

An isometry has distortion equal to 1. As distortion increases, the map ϕ stretches or shrinks the two spaces more and more.

A classical result in the theory of metric embedding into Euclidean space indicates that we can do much better than the negative examples for isometric embedding if we allow for some wiggle room in the form of distortion. Define $\ell_p(\mathbb{R}^n)$ to be the space $\mathbb{R}^n$ equipped with the p-norm $\|\mathbf{x}\|_p := (\sum_i |x_i|^p)^{1/p}$. A classical result in the theory of embedding shows that *any* finite metric space embeds well into $\ell_p(\mathbb{R}^n)$ for remarkably small n [4]:

Proposition 7.2 (Bourgain's Theorem). *Suppose (M, d) is a metric space consisting of n points, that is, $|M| = n$. Then, for $p \geq 1$, M embeds into $\ell_p(\mathbb{R}^m)$ with $O(\log n)$ distortion, where $m = O(\log^2 n)$.*

Matousek improved the distortion bound to $\log n/p$ [14].

From a practical perspective, the embedding result in Proposition 7.2 is far more practical than the isometric embedding provided in Proposition 7.1, for many applications. In particular, if we are willing to pay in the form of distortion, we can use a dimensionality that is *logarithmic* in the number of points in M. In contrast, the construction in Proposition 7.1 requires as many dimensions as there are points in M, limiting the algorithmic utility of that result.

Behind Proposition 7.2 is a simple algorithm that explicitly constructs the embedding. Following CITE, define the *Fréchet embedding* as follows:

Definition 7.3 (Fréchet embedding). *Suppose (M, d) is a metric space that $S_1, \ldots, S_r \subseteq M$. We define the* Fréchet embedding *of M with respect to $\{S_1, \ldots, S_r\}$ to be the map $\phi : M \to \mathbb{R}^r$ given by*

$$\phi(x) := (d(x, S_1), d(x, S_2), \ldots, d(x, S_r)), \tag{7.2}$$

where $d(x, S) := \min_{y \in S} d(x, y)$.

The construction in the proof of Proposition 7.1 could be considered a special case of Fréchet embedding where $S_i = \{a_i\}$ for $i \in \{1, \ldots, n\}$.

An easily-implemented probabilistic algorithm justifies Proposition 7.2. Construct a number of subsets $S_{ij} \subseteq M$ by sampling each node in M independently with probability 2^{-j}. Then, the embedding achieving $O(\log n)$ distortion is the Fréchet embedding of M with respect to S_{ij}, where $j \in \{1, \ldots, \lceil \log n \rceil\}$ and $i \in \{1, \ldots, 576\lceil \log n \rceil\}$. One surprising aspect of this construction is that there is no p dependence: The same embedding is effective for all $p \geq 1$. Formally justifying that this algorithm achieves the proper distortion is outside of the scope of our discussion since it involves probabilistic techniques we have not introduced, but given a few commonly-applied probabilistic inequalities the proof is technical but fairly elementary; the weakest version of the proof shows that the probabilistic algorithm has nonzero probability of succeeding.

7.4 MULTIDIMENSIONAL SCALING

Bourgain's Theorem is a constructive, approximate result for embedding. Although the guarantee of limited distortion provided by Bourgain's construction is attractive, in many applications we want better control over the dimensionality of the target space, even if this means we cannot perform as well at embedding theoretically. Rather than employing randomized, constructive algorithms, in this case it might make sense to *optimize* for the best possible embedding given a fixed target space and measure of distortion; these algorithms are more numerical than combinatorial in nature.

A class of algorithms known as multidimensional scaling (MDS) techniques aims to embed a finite metric space into $\mathbb{R}^m$ for some fixed m. The n-point metric space is given as input in the form of a matrix $D_0 \in \mathbb{R}_+^{n \times n}$. Put differently, we wish to find a set of embedding points $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^m$ so that $\|\mathbf{x}_i - \mathbf{x}_j\|_2 \approx D_{0ij}$. The objective function used to evaluate the quality of the set of $\mathbf{x}_i$'s determines the particular variant of MDS.

7.4.1 *Classical MDS*

Suppose we go in the (much easier) reverse direction, computing a distance matrix from a set of points $\mathbf{x}_i \in \mathbb{R}^m$. Algebraically, it is more convenient to work with the *squared* distance matrix $P_{ij} := \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$ rather than non-squared distances D. For convenience, we will also define a matrix $X \in \mathbb{R}^{m \times n}$ whose columns are the $\mathbf{x}_i$'s.

Define the *Gram matrix* of the $\mathbf{x}_i$'s as follows:

Definition 7.4 (Gram matrix)**.** *Given a collection of points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^m$, the* Gram matrix *associated to the $\mathbf{x}_i$'s is the matrix $G \in \mathbb{R}_+^{n \times n}$ whose entries are given by $G_{ij} := \mathbf{x}_i \cdot \mathbf{x}_j$. If $X \in \mathbb{R}^{m \times n}$ is the matrix whose columns are the $\mathbf{x}_i$'s, then $G = X^\top X$.*

A simple relationship relates G to P. Expanding the square:

$$P_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = \|\mathbf{x}_i\|_2^2 + \|\mathbf{x}_j\|_2^2 - 2\mathbf{x}_i \cdot \mathbf{x}_j = G_{ii} + G_{jj} - 2G_{ij}.$$

Hence, we can write

$$P = -2G + \operatorname{diag}(G)\mathbf{1}^\top + \mathbf{1}\operatorname{diag}(G)^\top, \tag{7.3}$$

where $\operatorname{diag}(G) \in \mathbb{R}^n$ denotes the vector composed of diagonal elements of the matrix G and $\mathbf{1} \in \mathbb{R}^n$ denotes the vector of all ones.

Now we turn our discussion backward and try to estimate the columns of X—and hence the embedding of our data—from the pairwise squared distance matrix P. We will first consider the case where $P_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$ exactly for a set of $\mathbf{x}_i$'s that we have somehow lost, and our job is to recover the $\mathbf{x}_i$'s; we then relax to the approximate case.

Unfortunately, given G or P, it is usually impossible to recover X. Take any orthogonal matrix $R \in \mathrm{O}(m)$. Then, the Gram matrix of the rotated point set RX is

$$G' = X^\top R^\top RX = X^\top X = G.$$

Similarly, shifting the $\mathbf{x}_i$'s uniformly by any constant vector $\mathbf{v}$–or applying R—does not affect P. Instead, we satisfy ourselves with computing any set of points that is *isometric* to the original set of $\mathbf{x}_i$'s.

By construction, the Gram matrix G is positive semidefinite with rank $r = \min\{n, m\}$. Hence, it admits an eigendecomposition $G = Q\Lambda Q^\top$, where $Q \in \mathbb{R}^{n\times r}$ has orthonormal columns and $\Lambda \in \mathbb{R}^{r\times r}_+$ is a diagonal matrix of eigenvalues. Define $\overline{X} := \sqrt{\Lambda}Q^\top$; then, $G = \overline{X}^\top \overline{X}$. Reversing (7.3) shows that the pairwise squared distances between the columns of $\overline{X}$ agree with P. That is, given G we can find an isometric embedding of our data as the columns of $\overline{X}$.

Define the *centering matrix*

$$J := I_{n\times n} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top.$$

By construction, J projects onto the subspace of $\mathbb{R}^n$ orthogonal to $\mathbf{1}$. If we assume our $\mathbf{x}_i$'s sum to $\mathbf{0}$ (a safe assumption since shifting a point set by a constant does not affect pairwise distances), then by (7.3) we have

$$G = -\frac{1}{2}J^\top PJ. \tag{7.4}$$

Motivated by this formula, the classical MDS algorithm CITE for embedding a pairwise distance matrix D_0 into $\mathbb{R}^m$ proceeds as follows:

1. Squared distance computation: Compute the squared distance matrix P with entries $P_{ij} := D^2_{0ij}$.
2. Double centering: Compute the centered Gram matrix $G := -\frac{1}{2}J^\top PJ$.
3. Eigendecomposition: Factor $G = Q\Lambda Q^\top$ where Λ contains the largest m eigenvalues of G.
4. Embedding: Take the final embedding to be the columns of $\overline{X} := \sqrt{\Lambda}Q^\top$.

By our argument above, when D_0 truly contains pairwise distances between points in $\mathbb{R}^m$, G will have rank $\leq m$; in this case, classical MDS is *guaranteed* to recover an embedding that reflects the pairwise distances exactly.

Recovering pairwise distances from an exact distance matrix is hardly an interesting problem. Classical MDS can be applied without change, however, when D_0 encodes any finite metric space. While there are fewer guarantees in this case, the use of the m largest eigenvalues guarantees that we have recovered the best rank-m approximation of G in the Frobenius norm CITE.

7.4.2 *Landmark MDS*

The assumption that MDS takes as input an $n \times n$ matrix of pairwise distances is somewhat limiting: It requires $O(n^2)$ storage and implies we know all possible relationships between the entities in our metric space. A small extension of MDS known as *landmark MDS* addresses this issue, by assuming instead that we have distances from a few landmark points to all others in the space we wish to embed CITE.

Before introducing landmarks, we derive a useful formula from MDS. Continuing in the notation of the previous section, recall $G = \overline{X}^\top \overline{X}$ and that $\overline{X} = \sqrt{\Lambda}Q^\top$. Since we chose only to keep the top eigenvalues of G, we have $Q^\top Q = I_{m\times m}$ (but possibly $QQ^\top \neq I_{n\times n}$); furthermore, assuming we remove the zero eigenvalue thanks to double-centering, we know $Q^\top \mathbf{1} = \mathbf{0}$.

Plugging these identities into (7.3) shows

$$\begin{aligned}\overline{X}P &= -2\Lambda^{3/2}Q^\top + \sqrt{\Lambda}Q^\top \text{diag}(Q\Lambda Q^\top)\mathbf{1}^\top \\ &= -2\Lambda\overline{X} + \overline{X}\text{diag}(Q\Lambda Q^\top)\mathbf{1}^\top \\ \implies \overline{X} &= \Lambda^{-1}\overline{X}(P - \text{diag}(Q\Lambda Q^\top)\mathbf{1}^\top).\end{aligned}$$

Reading this expression column-by-column, if we take $\overline{\mathbf{x}}_i$ to be the embedding of point i we have

$$\overline{\mathbf{x}}_i = \frac{1}{2}\Lambda^{-1}\overline{X}(\mathbf{p}_i - \mathbf{g}), \tag{7.5}$$

where $\mathbf{g} := \text{diag}(\overline{X}^\top\overline{X})$ and $\mathbf{p}_i$ is the i-th column of P, that is, the squared distances from $\mathbf{x}_i$ to each of the other points.

Landmark MDS simply extrapolates (7.5) to new points. Suppose we do MDS to embed *exclusively* the landmark points to obtain $\overline{X}$ and the other matrices above. Then, given a new point we wish to embed, we compute its squared distances to the landmarks in a vector $\mathbf{p}$, and plug this unseen $\mathbf{p}$ into (7.5). This landmark-based embedding has the property that if $\mathbf{p} = \mathbf{p}_i$, then the embedding agrees with $\overline{x}_i$, while reasonably extrapolating linearly to points nearby.

7.4.3 *SMACOF and Variants*

Classical MDS is reasonable if we expect our input distance matrix already to be approximately Euclidean. When G does not admit a rank-m eigenfactorization exactly, however, it is unclear the extent to which classical MDS recovers a reasonable embedding into $\mathbb{R}^m$. To alleviate this issue, other MDS algorithms approach the embedding problem *variationally*, defining the embedding as the minimizer of some optimization problem that directly measures distortion. As an example algorithm here we consider "scaling by majorizing a complicated function," or SMACOF for short [CITE]. This algorithm searches for a local optimum of a nonconvex problem to find an embedding, directly reducing a measure of distortion from the input distance matrix D_0.

In particular, SMACOF considers the following objective function:

$$f(X) := \sum_{ij}(D_{0ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2)^2. \tag{7.6}$$

The function $f(\cdot)$ takes as input an embedding encoded in the columns of X and outputs a least-squares objective measuring how well X conforms with the prescribed distances in D_0; this particular choice of objectives is sometimes known as the (squared) *stress* function [CITE]. Inside of the square is a vector norm, making $f(\cdot)$ nonconvex.

Expanding the square in (7.6) yields three terms:

$$\begin{aligned}\sum_{ij} D_{0ij}^2 &= \text{const.} \\ \sum_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 &= \text{tr}(XVX^\top), \text{ where } V = 2nJ \\ -2\sum_{ij} D_{0ij}\|\mathbf{x}_i - \mathbf{x}_j\|_2 &= -2\sum_{ij,\mathbf{x}_i\neq\mathbf{x}_j}\frac{D_{0ij}}{\|\mathbf{x}_i - \mathbf{x}_j\|_2}\cdot\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \\ &= -2\text{tr}(XB(X)X^\top), \text{ where } B_{ij}(X) := \begin{cases} -\frac{D_{0ij}}{\|\mathbf{x}_i - \mathbf{x}_j\|_2} & \text{if } \mathbf{x}_i \neq \mathbf{x}_j \text{ and } i \neq j \\ 0 & \text{if } \mathbf{x}_i = \mathbf{x}_j \text{ and } i \neq j \\ -\sum_{j\neq i} B_{ij} & \text{if } i = j. \end{cases}\end{aligned}$$

The fraction in the definition of $B(X)$ replaces the norm $\|\mathbf{x}_i - \mathbf{x}_j\|_2$ with the squared norm $\|\mathbf{x}_i - \mathbf{x}_j\|_2^2$, which is amenable to easier numerical methods for least-squares optimization.

Define a function

$$\tau(X, Z) := \text{const.} + \text{tr}(XVX^\top) - 2\text{tr}(XB(Z)Z^\top). \tag{7.7}$$

By the expressions above, we have $f(X) = \tau(X, X)$. Furthermore, we can prove that $\tau(\cdot, \cdot)$ provides a *majorizer* for f:

Proposition 7.3. *For matrices $Z \in \mathbb{R}^{m\times n}$, we have $\tau(X, X) \leq \tau(X, Z)$ with equality exactly when $X \propto Z$.*

Proof. This proposition follows from the Cauchy–Schwarz inequality REF. In particular, a direct consequence of this inequality is

$$(\mathbf{x}_i - \mathbf{x}_j) \cdot (\mathbf{z}_i - \mathbf{z}_j) \leq \|\mathbf{x}_i - \mathbf{x}_j\|_2 \|\mathbf{z}_i - \mathbf{z}_j\|_2,$$

with equality when $\mathbf{x}_i - \mathbf{x}_j = \mathbf{z}_i - \mathbf{z}_j$. Rearranging this inequality gives a *lower* bound for the distance between two points:

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2 \geq (\mathbf{x}_i - \mathbf{x}_j) \cdot \frac{\mathbf{z}_i - \mathbf{z}_j}{\|\mathbf{z}_i - \mathbf{z}_j\|_2}.$$

Now, we expand part of the value of τ:

$$\begin{aligned}
-2\text{tr}(XB(X)X^\top) &= -2\sum_{ij} D_{0ij}\|\mathbf{x}_i - \mathbf{x}_j\|_2 \text{ by definition} \\
&\leq -2\sum_{ij} D_{0ij}(\mathbf{x}_i - \mathbf{x}_j) \cdot \frac{\mathbf{z}_i - \mathbf{z}_j}{\|\mathbf{z}_i - \mathbf{z}_j\|_2} \text{ by the inequality above} \\
&= -2\sum_{ij} \frac{D_{0ij}}{\|\mathbf{z}_i - \mathbf{z}_j\|_2}(\mathbf{x}_i - \mathbf{x}_j) \cdot (\mathbf{z}_i - \mathbf{z}_j) \text{ after rearranging} \\
&= -2\text{tr}(XB(Z)Z^\top).
\end{aligned}$$

Plugging into (7.7) shows $\tau(X, X) \leq \tau(X, Z)$. The equality case follows directly from equality for Cauchy–Schwarz. □

A function τ with the property in the statement of Propostion 7.3 enables a simple optimization algorithm known as the Majorization-Minimization (MM) algorithm CITE:

$$X^{k+1} \leftarrow \arg\min_X \tau(X, X^k). \tag{7.8}$$

We can find an explicit expression for X^{k+1} in terms of X^k since our majorizer leads to a least-squares problem. In particular, to solve the minimization over X we write

$$0 = \nabla_X \tau(X, X^k) = 2XV - 2X^k B(X^k).$$

Solving this expression for $X = X^{k+1}$ gives an explicit iteration for optimizing SMACOF:

$$X^{k+1} \leftarrow \frac{1}{2n} X^k B(X^k) \left(I_{n\times n} - \frac{\mathbf{1}\mathbf{1}^\top}{n} \right). \tag{7.9}$$

Exercise 7.5. verifies that this is the solution to the linear system.

A key property verifies that our iteration above improves the SMACOF objective value in every step:

Proposition 7.4. *For X^k defined in (7.9), we have $f(X^{k+1}) \leq f(X^k)$ for all k.*

Proof. The proof is a standard one for MM algorithms. We have:

$$\begin{aligned} f(X^{k+1}) &= \tau(X^{k+1}, X^{k+1}) \text{ by (7.7)} \\ &\leq \tau(X^{k+1}, X^k) \text{ by Proposition 7.3} \\ &\leq \tau(X^k, X^k) \text{ by (7.8)} \\ &= f(X^k) \text{ by (7.7)} \end{aligned}$$

□

This proposition shows that the objective *value* for SMACOF converges under iteration (7.9). More challenging proofs verify cases in which the *iterates* converge CITE, which is a more difficult property to verify.

Example 7.4 (Shape-from-operator). *JS: fill this in some day*

Sammon CITE introduced a slightly modified version of the stress function

$$\bar{f}(X) := \sum_{ij} \frac{(D_{0ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2)^2}{D_{0ij}}. \tag{7.10}$$

This objective function measures relative changes in distances, and hence is more sensitive to preserving short distances than the stress function (7.6). Our MM algorithm above is easily extended to this objective function, as explored in problem 7.7..

Example 7.5 (Comparing SMACOF to Sammon mapping). *JS: add a plot*

7.5 INTRINSIC-TO-EXTRINSIC EMBEDDING

JS: Add text distinguishing terms "intrinsic" and "extrinsic"

So far, the algorithms we have considered assume as input that we are given a metric space as input, e.g. in the form of a matrix of pairwise distances. This is not always the most practical way to express a metric space, since it take $O(n^2)$ storage and requires knowledge of a complete set of relationships between the different elements in the space. As an alternative, a wide class of algorithms takes as input a Euclidean embedding of a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^m$, where m is extremely large. The task is to embed these points into a lower-dimensional space, while preserving their key relationships.

One possibility is that our data points are sampled from a *submanifold* of $\mathbb{R}^m$ whose dimensionality is far less than m. In this case, we have a hope of embedding our data into a lower-dimensional space. This hope is not simply optimistic but actually comes from a few theoretical results in modern differential topology. For instance, the Whitney embedding theorem states that we can embed any smooth m-dimensional manifold into $\mathbb{R}^{2m}$ CITE:

Theorem 7.1 (Whitney embedding theorem). *Any smooth, real k-dimensional manifold maps smoothly into $\mathbb{R}^{2k}$.*

This theorem does not guarantee an isometry, just that we can embed the manifold without self-intersections or singularities. The dimensionality $2k$ is a tight bound: k-dimensional real projective space does not embed into $\mathbb{R}^{2k-1}$. If we care about obtaining an isometric map, a second famous result comes to the rescue CITE:

Theorem 7.2 (Nash–Kuiper embedding theorem, simplified). *Any k-dimensional Riemannian manifold admits an isometric, differentiable embedding into $\mathbb{R}^{2k}$.*

Submanifolds of $\mathbb{R}^m$ are a special case of Riemannian manifolds, covering the definitions we introduced in § REF.

There are plenty of situations in which we observe manifold structures in data in practice. For instance, consider the following examples:

Example 7.6 (S^1 in data). *Suppose we collect a video sequence of a carousel moving in circles, as illustrated in Figure* REF*. If each frame of the video sequence is an image of size $r \times c$, then we can think of the frames as data points in $\mathbb{R}^{rc}$ containing one grayscale value per pixel. This is an extremely high-dimensional space! But thanks to the circular motion of the carousel, the data points lie roughly on a one-dimensional submanifold of $\mathbb{R}^{rc}$ isomorphic to the circle S^1. The embedding theorems above show that we can embed that circle into $\mathbb{R}^2$ without any distortion of geodesic distance.*

Example 7.7 (Protein folding). *JS: Proteins fold with a fixed set of intrinsic degrees of freedom*

The proofs of the theorems above are extremely involved and not well-suited to designing algorithms that cope with noisy data, uncertain intrinsic dimensionality, and other confounding factors. A set of algorithms, however, attempts to find embeddings of data given local relationships that resemble the manifold structures needed for the theorems above to apply. These popular techniques may not come with guarantees linking them to the theory above but can be effective tools embedding in a way that reveals the intrinsic structure in data that may originally be embedded in a high-dimensional space.

7.5.1 *Isomap*

The Isomap algorithm, introduced in CITE, attempts to find an embedding that *realizes* intrinsic distance as extrinsic distance. That is, if we think of a submanifold M as a set of points equipped with the geodesic distance metric $d_g : M \times M \to \mathbb{R}_+$, Isomap attempts to find a map $\phi : M \to \mathbb{R}^p$ from M into a subset of $\mathbb{R}^p$ so that Euclidean distances in the $\mathbb{R}^p$ embedding resemble geodesic distances along M.

As illustrated in Figure REF, the task for Isomap is fairly rigid. In particular, if M is a k-dimensional submanifold of $\mathbb{R}^m$, the image $\phi(M)$ in Isomap is considered as a region in $\mathbb{R}^p$ rather than as a k-dimensional submanifold of $\mathbb{R}^p$. That is, we equip $\phi(M)$ with the Euclidean metric rather than geodesic distances.

The Isomap algorithm, illustrated in Figure REF, is straightforward given the techniques we have already derived:

1. As input, Isomap takes a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^m$ as well as a dimension p into which the points should be embedded and an integer k.

2. A k-nearest neighbor graph is constructed on the $\mathbf{x}_i$'s, whose edge lengths are chosen to be Euclidean distances in $\mathbb{R}^m$.

3. We compute a pairwise distance matrix $D \in \mathbb{R}^{n \times n}$ between all the vertices in the graph, e.g. using n instances of Dijkstra's algorithm on graphs outlined in §6.4.1 or using the (easily-implemented) Floyd–Warshall all-pairs-shortest-path algorithm CITE. D is intended to approximate pairwise *geodesic* distance between the $\mathbf{x}_i$'s.

4. We use classical MDS to embed D into $\mathbb{R}^p$, as outlined in §7.4.1.

Note that D is subject to all the drawbacks of k-nearest neighbor distances for approximating geodesics outlined in § REF; given the approximate nature of Isomap, however, this is likely not the largest source of error or distortion in the technique.

Figure REF shows an example of Isomap in action. JS: say some qualitative things The parameter k also has a critical effect on the behavior of this algorithm: When k is too large, "short circuit" errors add spurious edges to the graph that make the distances in D too small, while small k can disconnect the graph.

The landmark MDS algorithm from §7.4.2 is directly relevant to a landmark-based variant of Isomap CITE. In this version, we construct the same graph as in classical Isomap but only compute distances from a few landmark points to the remainder of the dataset; we then employ landmark

MDS instead of classical MDS. Shortest-path computation on a graph can be expensive to carry out and store, making this variant valuable in practice.

7.5.2 *Locally-Linear Embedding (LLE)*

As mentioned in the previous section, Isomap is rigid in that it attempts to capture the distance between *every* pair of points when embedding a pointset into $\mathbb{R}^p$. As a point of contrast, the *locally-linear embedding* (LLE) algorithm attempts to compute an embedding that only preserves local structure.

Before providing technical details, Figure REF shows a classic example comparing Isomap to LLE. Here, a punctured planar shape is unrolled to $\mathbb{R}^2$. From the perspective of geodesic distance, the top and bottom of the hole are farther apart than they would be had that hole been filled; this causes Isomap to pull apart the two sides of the puncture. LLE is less (but not negligibly) affected by this incomplete data, treating the hole as a less significant feature.

The basic idea of LLE is illustrated in Figure REF: We express a point as a weighted average of its neighbors, and then we find an embedding that preserves this weighted average structure. These two steps are explained in detail below:

ANALYSIS STEP. As with the other embedding methods above, LLE takes as input a point set $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^m$. In the analysis step, our goal is to approximate how each point $\mathbf{x}_i$ as a weighted average of nearby points, giving some notion of a local neighborhood to each $\mathbf{x}_i$.

Concretely, the analysis step computes a matrix $W \in \mathbb{R}^{n \times n}$ that satisfies a few criteria:

$$\begin{array}{rll} \mathbf{x}_i & \approx \sum_j W_i^j \mathbf{x}_j & \text{Barycentric approximation} \\ W_i^i & = 0 \ \forall i & \text{No self weight} \\ \sum_j W_i^j & = 1 & \text{Weighted average} \\ W_i^j & = 0 \text{ when } \|\mathbf{x}_i - \mathbf{x}_j\|_2 \gg 0 & \text{Local} \end{array}$$

To fulfill the criteria above efficiently, the matrix W is computed one row at a time. For row i, the collection of k nearest neighbors for $\mathbf{x}_i$ in the point cloud is collected. For convenience we will denote those neighbors $\mathbf{n}_1, \ldots, \mathbf{n}_k \in \mathbb{R}^m$ with entries $\omega^1, \ldots, \omega^k$ in W; so, we will take $W_i^j = \omega^\ell$ if $\mathbf{x}_j = \mathbf{n}^\ell$. We will set W_i^j corresponding to points outside those k nearest neighbors to zero; this achieves the locality property.

To determine the ω^j's, we solve the following problem

$$\begin{array}{ll} \min_{\omega^1, \ldots, \omega^k} & \left\| \mathbf{x}_i - \sum_j \omega^j \mathbf{n}_j \right\|_2^2 \\ \text{subject to} & \sum_j \omega^j = 1. \end{array}$$

Write the ω^j's in a column vector $\boldsymbol{\omega}$ and take N to be the matrix whose columns are the $\mathbf{n}_j$'s. Then, we can write the problem as

$$\begin{array}{ll} \min_{\omega^1, \ldots, \omega^k} & \boldsymbol{\omega}^\top N^\top N \boldsymbol{\omega} - 2\mathbf{x}_i^\top N \boldsymbol{\omega} \\ \text{subject to} & \mathbf{1}^\top \boldsymbol{\omega} = 1, \end{array}$$

where $\mathbf{1}$ denotes the vector of all ones. Introducing a Lagrange multiplier λ for the constraint, we can recover $\boldsymbol{\omega}$ by solving the linear system:

$$\begin{pmatrix} N^\top N & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\omega} \\ \lambda \end{pmatrix} = \begin{pmatrix} N^\top \mathbf{x}_i \\ 1 \end{pmatrix}. \tag{7.11}$$

EMBEDDING STEP. From the analysis step, we now have a matrix W so that $X \approx XW^\top$, where the rows of W are computed using the technique above and $X \in \mathbb{R}^{m\times n}$ is the matrix whose columns are the $\mathbf{x}_i$'s. The next step of LLE embeds the columns of X into $\mathbb{R}^p$ for some $p < m$ by preserving this relationship.

Suppose $Y \in \mathbb{R}^{p\times n}$ is the matrix containing the embeddings of the $\mathbf{x}_i$'s. Then, we compute Y using the following problem

$$\begin{array}{ll} \min_Y & \|Y - YW^\top\|_{\text{Fro}}^2 \\ \text{subject to} & YY^\top = I_{p\times p} \\ & Y\mathbf{1} = \mathbf{0}. \end{array} \tag{7.12}$$

The objective here tries to preserve the relationship $X \approx XW^\top$ in the lower-dimensional embedding encoded by Y. This relationship, however, is scale-independent: For example, taking $Y = 0$ would preserve it exactly! Hence, we add the constraint $YY^\top = I_{p\times p}$ to yield a nontrivial constraint. The second constraint removes non-uniqueness due to shifting the columns of Y by a constant vector.

We can rewrite the objective function as follows:

$$\begin{aligned} \|Y - YW^\top\|_{\text{Fro}}^2 &= \|Y(I_{n\times n} - W^\top)\|_{\text{Fro}}^2 \\ &= \text{tr}(Y(I_{n\times n} - W^\top)(I_{n\times n} - W^\top)^\top Y^\top) \text{ by definition of the Frobenius norm} \\ &= \text{tr}(Y^\top YM) \text{ after defining } M := (I_{n\times n} - W)^\top (I_{n\times n} - W) \end{aligned}$$

Hence, we can write our problem in a different form:

$$\begin{array}{ll} \min_Y & \text{tr}(YMY^\top) \\ \text{subject to} & YY^\top = I_{p\times p} \\ & Y\mathbf{1} = \mathbf{0}. \end{array} \tag{7.13}$$

After applying symmetry of M and the "no self weight" condition, we find that the rows of Y are exactly the p eigenvectors of M corresponding to the p smallest nonzero eigenvalues of M; exercise 7.8. checks this in detail.

Hence, the final step of LLE is to compute eigenvectors of the matrix M, which comprise the desired embedding. As discussed above, Figure REF shows LLE applied to a sample problem and compares to Isomap.

7.5.3 *Diffusion Maps*

From a high level, the LLE algorithm uses as an embedding the eigenvectors of a matrix M measuring the relationship between a point and its neighborhood. A number of techniques use a similar approach, constructing matrices that measure affinities or differences between objects in a metric space and using their eigenvectors as embeddings.

One key approach known as *diffusion maps* CITE embeds based on (generalized) eigenvectors corresponding to eigenvalues of a kernel matrix

$$K_{ij} := e^{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/\varepsilon}.$$

Once again, K measures some notion of affinity between points, in this case via a Gaussian in extrinsic distances. This easily-implemented method simply uses the eigenvectors of $D^{-1}K$ for the embedding, where D is a diagonal matrix of row sums of K JS: check me.

We will revisit diffusion maps in § REF, once we have discussed the Laplacian and its role in the heat equation, to see that it is a natural way to embed based on a Brownian motion interpretation of distance.

7.6 EUCLIDEAN TO EUCLIDEAN: PCA AND THE JOHNSON–LINDENSTRAUSS LEMMA

JS: not covered in 6.838 beyond a slide or two, so will skip writing for now; use phrase "concentration of measure" in here somewhere, dimensionality reduction

7.7 REPRESENTATION LEARNING

JS: Nothing too interesting to say here, but basically all the objective functions above can be rolled into deep networks and other learning pipelines

7.8 MORE EXOTIC EMBEDDINGS

JS: tree/snowflake embeddings? Bartal's theorem? sketching? topological considerations? mapper algorithm? orbifolds?

7.9 t-SNE, UMAP, AND VARIANTS

JS: Relationship-based embedding

7.10 OTHER PROBLEMS INVOLVING METRICS

We have covered a few animals in the huge zoo of potential embedding techniques. Each algorithm has different theoretical properties, but largely they are straightforward to implement and try. As with many topics in geometry, probably the easiest way to get a feel for these methods is simply to give them a try: Choosing an effective embedding is as much an art as a science, and the best choice largely depends on the application or end user.

Embedding is one of many possible problems involving metric spaces. We conclude here by mentioning a few additional interesting directions, many of which involve noisy or incomplete data typical in learning and statistical settings.

7.10.1 *Metric Nearness*

Embedding in some sense attempts to extract a complete set of relationship between elements of a metric space: Once they are all placed into the same space, a distance can be measured between any pair while satisfying axioms like the triangle inequality. But in doing so, we force the structure of the embedding space onto our data, even if the data does not embed perfectly into that space.

Returning to Definition 7.1 suggests an alternative approach. Take $D \in \mathbb{R}_+^{n\times n}$ to be a matrix of pairwise distances between elements of a metric space consisting of n elements. Then, for all $i, j, k \in \{1, \dots, n\}$ we have

$$\begin{aligned} D_{ij} &\geq 0 && \text{Positivity} \\ D_{ii} &= 0 && \text{Partial indiscernibility} \\ D_{ij} &= D_{ji} && \text{Symmetry} \\ D_{ik} &\leq D_{ij} + D_{jk} && \text{Triangle inequality.} \end{aligned} \tag{7.14}$$

Any matrix that satisfies these four criteria *nearly* defines a discrete metric space on n elements, the only missing component being $D_{ij} \neq 0$ if $i \neq j$—the reverse part of the indiscernibility property.

A key observation about (7.14) is that these conditions are *convex* in the entries of D. That is, if we take $\mathcal{D}$ to be the set of matrices $D \in \mathbb{R}^{n\times n}$ satisfying the expressions in (7.14), this set is convex. This enables us to carry out optimization problems for metric matrices $D \in \mathcal{D}$ with global optimality.

As one example, the *metric nearness* problem proposed in [CITE] attempts to find the closest approximation D of a matrix D_0 with the constraint that D satisfies the metric axioms (7.14):

$$\min_{D \in \mathcal{D}} \|D - D_0\|_{\text{Fro}}. \tag{7.15}$$

While any convex optimization procedure will suffice to solve this least-squares problem, the algorithm derived in that paper and shown in Figure [REF] is particularly simple: Keep choosing ijk triplets in the current estimate of D and repair them if they do not satisfy the triangle inequality.

7.10.2 *Euclidean Matrix Completion*

A related problem to metric nearness is *Euclidean matrix completion*, in which the distance matrix D is assumed to encode pairwise distances specifically in a Euclidean space $\mathbb{R}^m$ rather than a general metric space. In this case, we can leverage (7.3), developed in our discussion of classical MDS, which shows that we can compute pairwise squared distances from a Gram matrix G.

This formula shows that any pairwise squared-distance matrix P can be computed directly from the Gram matrix G of inner products. In fact, we can prove something stronger. Take *any* semidefinite matrix $G \succeq 0$ and apply (7.3) to compute $P(G)$. Then, this matrix $P(G)$ is actually a Euclidean distance matrix. One way to show this is to apply Cholesky factorization to write $G = X^\top X$ for some matrix X; the columns of X are an embedding whose inner products are in G.

The set of semidefinite matrices $\{G \in \mathbb{R}^{n \times n} : G \succeq 0\}$ is convex and enforceable in standard convex optimization algorithms, making G a reasonable proxy for computing pairwise Euclidean distances. As an example, suppose D_0 encodes noisy Euclidean distances between a sparse set of pairs of points; we define the matrix $H \in \{0,1\}^{n \times n}$ so that $H_{ij} = 1$ if D_{0ij} is meaningful and $H_{ij} = 0$ if D_{0ij} is missing. Then, [CITE] proposes completing and denoising D_0 by solving the following convex problem:

$$\begin{aligned} \min_G \quad & \|H \circ (P(G) - P_0)\|_{\text{Fro}}^2 \\ \text{subject to} \quad & G \succeq 0. \end{aligned} \tag{7.16}$$

Here, P_0 is the matrix whose entries are squares of the entries in D_0, and $\circ$ denotes the Hadamard—or entrywise—product, effectively removing entries in the objective term where D_{0ij} is unknown. The eigenvectors of G can be used to find an embedding into $\mathbb{R}^n$ of the data.

A related technique is *semidefinite embedding*, also known as *maximum variance unfolding* [CITE]. In this setting, we are given some triplets (i, j, D_{0ij}) specifying hard constraints on distances between some pairs of points. We once again aim to compute a Gram matrix G, but this time our goal is to maximize the variance of the embedding. In particular, suppose G implies an embedding $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^m$. The variance of this set is:

$$\begin{aligned} \text{Var}(Y) &:= \frac{1}{2n} \sum_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 \\ &= \frac{1}{2n} \sum_{ij} (\|\mathbf{y}_i\|_2^2 + \|\mathbf{y}_j\|_2^2 - 2\mathbf{y}_i \cdot \mathbf{y}_j) \text{ by expanding the square} \\ &= \sum_i \|\mathbf{y}_i\|_2^2 - \frac{1}{n} \left(\sum_i \mathbf{y}_i \right) \cdot \left(\sum_j \mathbf{y}_j \right) \text{ accounting for repeated terms} \\ &= \sum_i \|\mathbf{y}_i\|_2^2 \text{ if we assume } \sum_i \mathbf{y}_i = \mathbf{0} \\ &= \sum_i G_{ii} \text{ for Gram matrix } G_{ij} := \mathbf{y}_i \cdot \mathbf{y}_j \\ &= \text{tr}(G). \end{aligned}$$

Hence, semidefinite embedding solves the following problem:

$$\begin{aligned} \max_G \quad & \text{tr}(G) \\ \text{subject to} \quad & G \succeq 0 \\ & G_{ii} + G_{jj} - G_{ij} - G_{ji} = D_{0ij}^2 \; \forall (i, j, D_{0ij}) \text{ triplets} \\ & G\mathbf{1} = \mathbf{0}. \end{aligned} \tag{7.17}$$

The second constraint is equivalent to constraining the points to have $\mathbf{0}$ as a center of mass. Again Cholesky factorization or eigenvalue decomposition can be used to obtain an embedding. This is a *semidefinite program*, meaning the problem has a linear objective and constraints optimized over the set of semidefinite matrices; interestingly, we have eliminated any explicit reference to the original embedding variables $\mathbf{y}_i$ in our formulation, enabling this more elegant way to pose the problem.

Maximum variance unfolding has a curious geometric property due to the variance objective function. We provide some intuition in Figure REF; the theory of semidefinite programming fills in details. Suppose we have a three-point space and constrain only two distances, effectively defining a hinged structure about the shared vertex. If we maximize the variance, we three points unfold to a line: A one-dimensional structure! Generically speaking, the rank of G computing using this technique is often relatively low, providing a means of low-dimensional embedding of data by discarding eigenvectors corresponding to zero eigenvalues. This idea is related to notions of *tensegrity* in semidefinite programming CITE.

Example 7.8 (Maximum variance unfolding of a graph). *JS: add knn to the mix; plot the eigenvalues; show we can just keep the top few*

7.10.3 *Theoretical Challenges*

A number of interesting theoretical challenges are worth considering in the space of embedding problems. These span the breadth from computer science theory to the mathematical structure of assorted spaces.

We have introduced a variety of embedding techniques, some of which are numerical in nature. We have not, however, assessed the *complexity* or *conditioning* of the embedding problem, reflected e.g. in the number of iterations needed for an embedding algorithm to converge and/or the stability of the computed results. In some cases, these are fairly easy to assess: For instance, classical MDS is an eigenvalue problem that can be carried out in polynomial time CITE, with well-understood conditioning CITE. On the other hand, CITE shows that optimizing the following L_1 variant of the SMACOF objective function is NP-hard, even when the $\mathbf{x}_i$'s are in $\mathbb{R}^1$:

$$f(X) := \sum_{ij} |D_{0ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2|. \tag{7.18}$$

In particular, minimizing this objective can be reduced to a variant of 3SAT. This shows that we are unlikely to extract the global optimum to the embedding problem, although the techniques we have introduced often work well in practice. Other embedding problems, e.g. checking if two spaces are isometric, boil down to other computationally intractable problems like graph isomorphism.

Other challenges stem from choosing the appropriate embedding method, assorted parameters, or even whether embedding is appropriate at all. The well-known theoretical work CITE examines how to "test the manifold hypothesis," i.e. to check whether data is sampled from a lower-dimensional manifold to begin with. In practice, estimating the proper intrinsic dimensionality of an embedding requires some experimentation and is far from an exact science.

JS: applications to add as examples to this chapter: algorithmic runtime, compression, visualization, sampling

7.11 EXERCISES

7.1. Show that the real numbers $\mathbb{R}$ is a metric space when equipped with the distance metric $d(s,t) := |\log(y/x)|$.

7.2. Show that a map ϕ with distortion (7.1) equal to 1 is an isometry.

7.3. **JS: lemma 3.1 of**

7.4. **JS: some special case of Bourgain theorem**

7.5. Verify equation 7.9.

7.6. **JS: Sammon mapping instead of SMACOF**

7.7. **JS: MM algorithm for** (7.10)

7.8. **JS: Verify eigenvalue problem in for** (7.13)**; maybe also LLE from pairwise distance**