4

Dimensionality Reduction, Manifold Learning, and Metric Geometry

> A map is not the territory it represents, but, if correct, it has a similar structure to the territory, which accounts for its usefulness.
>
> *Alfred Korzybski*

Although topological data analysis is new, the idea of studying data by analyzing shape is classical. The original forms of this kind of analysis (regression, principal components analysis (PCA), and multidimensional scaling (MDS)) make the assumption that the data lies on a linear subspace in $\mathbb{R}^n$. In contrast, TDA makes minimal assumptions about the underlying metric measure space generating the data. On the one hand, this means that we can apply TDA to data sets where we have no reason to expect linear structure. On the other hand, strong geometric assumptions have many benefits. For example, assuming that the data lies on a k-dimensional subspace of $\mathbb{R}^n$ characterizes the problem as searching for a linear transformation $\theta\colon \mathbb{R}^n \to \mathbb{R}^k$ such that $\{\theta(x_i)\}$ retains something about the structure of $\{x_i\}$. Assuming linearity

1. provides coordinates for describing the data and predicting where new data points might lie,
2. allows the application of standard statistical inference methods, and
3. makes it straightforward to perform dimensionality reduction by constraining the value of k. For example, even if we believe that the data lies on a plane of dimension $\ell > 3$, it can be useful to project into $\mathbb{R}^2$ or $\mathbb{R}^3$ for visualization purposes.

Linear models are arguably the most frequently used tools in applied mathematics; however, the assumption of linearity is often unreasonable. As a result, there has been a lot of recent work generalizing these methods to algorithms that operate under the assumption that the data has been sampled from a compact manifold $M \subseteq \mathbb{R}^n$ of *much lower dimension* than n. These algorithms, loosely

referred to as *dimensionality reduction* or *manifold learning*, then seek to infer a parameterized representation of the data in terms of a coordinate system for a manifold. It is interesting to point out that in many biological applications we do not expect the data to lie on a manifold. For instance, the intrinsic dimension of transcriptomic data is related to active transcription programs (see Chapter 7). The number of these programs, relative to the intrinsic dimension, is not expected to be constant.

Although the manifold assumption is usually unrealistic for genomic data, dimensionality reduction has been successfully applied in various ways to analyze real biological data. For example, most applications of clustering in genomic analysis use dimensionality reduction as a preprocessing step (e.g., the frequent application of t-SNE), which is becoming standard in single cell analysis, see Chapter 7. More interesting from our point of view is the fact that some of the most successful genomic applications of Mapper have used coordinates from PCA as filter functions.

In the first part of this chapter we give a rapid overview of manifold learning and dimensionality reduction, starting with the classical techniques and moving on to recent generalizations. There is a vast literature on this subject, and we cannot hope to do more than give a flavor of these techniques. Our goal is to convey the central ideas underlying these approaches to analyzing data. Roughly speaking, the basic strategy of most manifold learning techniques is to take the k-nearest neighbors of a point x and use the vectors specified by the line segments from x to its neighbors as an approximation for the tangent plane at x. Global optimization then sews these local approximations together to produce a low-dimensional representation of the data. In a sense that can be made precise, the efficacy of these approaches depends on the fact that the Laplace-Beltrami operator on the manifold (which describes heat flow) can be approximated from finite samples by a certain *graph Laplacian* matrix.

In Figure 4.1, we indicate the results of different manifold learning representations on data that lies on a plane in $\mathbb{R}^3$; all of them recover coordinates for the plane. In Figure 4.2, we show a plane that has been rolled up – although the plane is flat, the embedding is twisted and so cross-cutting connections are potentially a problem (recall Section 2.2). Here, there is a noticeable difference in performance between classical techniques that assume linearity and manifold learning algorithms that do not.

In contrast to these cases, we will explore our running example of nested circles (which are not linear at all), and also consider nested arcs. In this context, manifold learning algorithms do a much worse job at recovering meaningful parametrizations. These simple experiments highlight the ways that topological data analysis

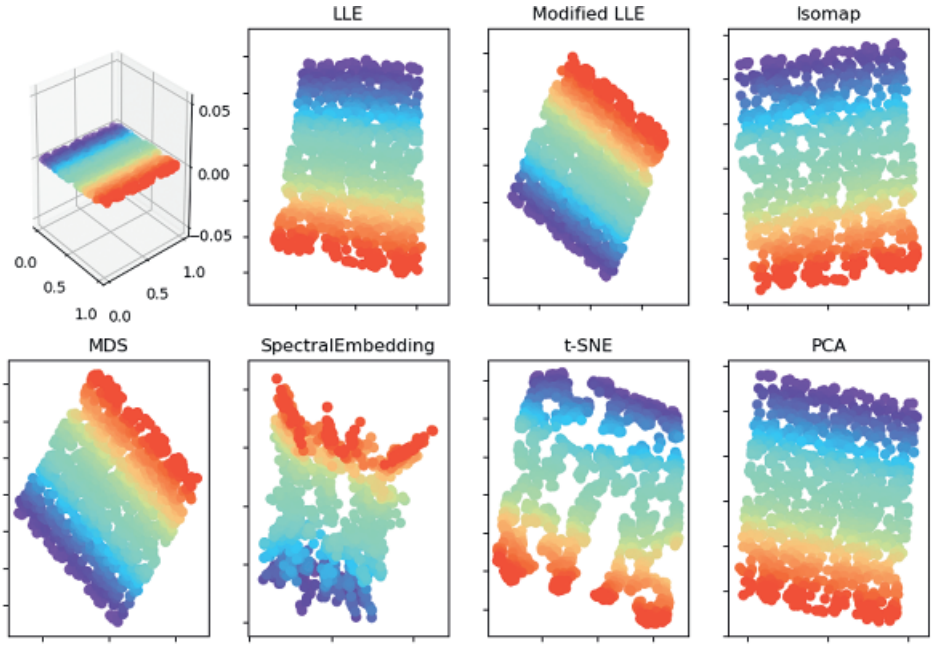

Figure 4.1 When the data lies on a plane in $\mathbb{R}^3$, all algorithms successfully recover a representation of the original data.

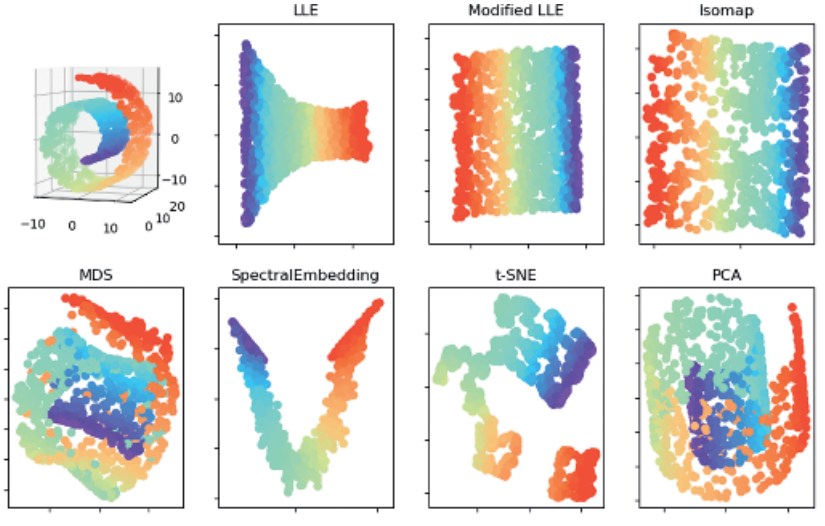

Figure 4.2 When the data lies on a rolled-up sheet, classical algorithms like PCA and MDS perform very poorly, whereas manifold learning techniques successfully capture the intrinsic shape of the data.

can be useful, even in situations where the data does lie on a low-dimensional manifold. Of course, in general, we do not necessarily expect such a hypothesis to hold.

However, in genomics there is an even more specialized geometric assumption that is frequently warranted. When working with data generated by evolutionary processes, it is standard to assume that the data can be organized into a phylogenetic tree. A phylogenetic tree is typically represented as a metric tree; recall from Example 1.2.4 that this is a graph with no cycles and weighted edges, where the metric is computed as the sum of the weights along the shortest path between two points.

In the second part of this chapter we give an overview of mathematical frameworks for dealing with phylogenetic trees. Again, this is a vast area of research with many excellent books; Felsenstein's text is a classic exposition [172]. We begin by giving a quick treatment of how to infer a phylogenetic tree from genomic data presented as a finite metric space; see Appendix C for a more detailed review. We then explain celebrated work of Billera, Holmes, and Vogtmann [55] that shows that phylogenetic trees can themselves be organized into a metric space; we will later see in Chapter 5 that the associated metric geometry (see Section 4.7.3) supports clinically significant analysis.

4.1 A Quick Refresher on Eigenvectors and Eigenvalues

Almost all of the dimensionality reduction techniques we will describe in this section involve computation of the eigenvectors of a matrix formed from the data points. Although we have assumed that the reader has familiarity with basic linear algebra, in this section we briefly review the relevant definitions. In the following, we always work with real vector spaces.

Definition 4.1.1. Let A be an $n \times n$ matrix. An *eigenvector* v for A with *eigenvalue* λ is a non-zero vector $v \in \mathbb{R}^n$ such that

$$Av = \lambda v.$$

The first key observation is that for symmetric matrices A (i.e., matrices such that $A = A^T$), eigenvectors with distinct eigenvalues are orthogonal.

Proposition 4.1.2. *Let A be a symmetric $n \times n$ matrix and let $v_1, v_2 \in \mathbb{R}^n$ be eigenvectors with distinct eigenvalues $\lambda_1 \neq \lambda_2$. Then v_1 is perpendicular to v_2.*

This suggests that we can think of eigenvectors for different eigenvalues as giving a preferred alternative set of coordinates for $\mathbb{R}^n$ which are adapted to the linear transformation represented by A. We do not always have enough eigenvectors to form a basis for all of $\mathbb{R}^n$, however.

Proposition 4.1.3. *An $n \times n$ matrix A has at most n distinct eigenvalues and at most n linearly independent eigenvectors. When there are exactly n independent eigenvectors, they form a basis.*

It is standard to sort the eigenvectors by the size of the associated eigenvalues; when we talk about the "top k" eigenvectors, we mean those with the k largest eigenvalues.

4.2 Background on PCA and MDS

A classical example of dimensionality reduction is principal component analysis (PCA). The idea here is, given a set of points $\{x_1, x_2, \ldots, x_m\}$ in $\mathbb{R}^n$ as data, to find an "optimal" linear projection $\theta\colon \mathbb{R}^n \to \mathbb{R}^k$, for $k < n$. Here is an outline of the algorithm.

1. We normalize to center the data and define
$$\widetilde{x_i} = x_i - \mu, \qquad \text{where} \qquad \mu = \frac{1}{n}\sum_i x_i.$$
2. We then form the covariance matrix
$$C = \frac{1}{n}\sum_i \tilde{x}_i \tilde{x}_i^T.$$
3. We compute the top k eigenvectors $\{v_1, \ldots, v_k\}$ of C to use as our basis.
4. These eigenvectors span a hyperplane (subspace) of $\mathbb{R}^n$ that is isomorphic to $\mathbb{R}^k$; the projection $\theta\colon \mathbb{R}^n \to \mathbb{R}^k$ of the data is precisely the orthogonal projection onto this plane followed by a choice of identification of the plane with $\mathbb{R}^k$.
5. We can also regard θ as producing vectors in $\mathbb{R}^n$; adding back μ yields approximations $y_i = \theta(\tilde{x}_i) + \mu$ of each x_i.

This process chooses the basis which maximizes the variance captured by the representation; the eigenvector v_1 with the largest eigenvalue is the single direction which captures the maximal amount of information about the variance in the points, the plane spanned by $\{v_1, v_2\}$ is the plane with the most variance, and so forth. Interestingly, we can also characterize the output of PCA as the projection that minimizes the error function
$$E = \sum_{i=1}^{m} \partial_{\mathbb{R}^n}(x_i, y_i)^2.$$
That is, PCA produces the points $\{y_i\}$ which minimize the reconstruction error among all projections onto a k-dimensional subspace.

In fact, another classical approach to dimensionality reduction is to take minimization of E as a point of departure. Metric multidimensional scaling (metric MDS), takes as input a finite metric space (X, ∂_X) and computes an optimal embedding of X into a Euclidean space $\mathbb{R}^k$. Here the optimality criterion is to preserve the original metric data as much as possible, i.e., to minimize an analogue of E. Specifically, in MDS we search for a map $\theta\colon X \to \mathbb{R}^k$ that minimizes

$$\mathcal{E} = \sum_{x_i, x_j \in X} \left(\partial_X(x_i, x_j) - \partial_{\mathbb{R}^k}(\theta(x_i), \theta(x_j))\right)^2.$$

We do this as follows.

1. Let D denote the matrix with entries $D_{ij} = \partial_X(x_i, x_j)$.
2. Set
$$H = I - \frac{1}{n} ee^T \qquad \text{and} \qquad Z = -\frac{1}{2} HDH,$$
where as usual I denotes the identity matrix and e is the vector with all entries 1. (This step centers the results; since the minimizing embedding is not unique as distances are preserved by translation, we need to impose such a constraint to get a specific output.)
3. The embedding that minimizes $\mathcal{E}$ is then given by finding the eigenvectors $\{v_j\}$ of Z. Specifically, the embedding $\theta(x_i) \in \mathbb{R}^k$ is specified by normalizing so that $\|v_j\|^2 = \lambda_j$, making a matrix with the eigenvectors $\{v_j\}$ as columns, and taking the ith row.

When the metric space (X, ∂_X) arises as a subspace of $\mathbb{R}^n$, then it turns out that PCA and metric MDS coincide.

Theorem 4.2.1. *Given $\{x_1, x_2, \ldots, x_\ell\} \subset \mathbb{R}^n$ and $k < n$, the results of metric MDS and PCA embedding $\{x_i\}$ into $\mathbb{R}^k$ are isometric.*

However, metric MDS has the advantage that it can be applied to arbitrary metric spaces, i.e., metric spaces that are not subspaces of $\mathbb{R}^n$. Moreover, posing the problem as minimizing the embedding error function $\mathcal{E}$ allows us to consider variants which minimize different error functions. For example, work on "antigenic maps" describing genomic and phenotypic variability in the flu virus uses an MDS variant [467]. Of course, changing $\mathcal{E}$ can result in substantially more difficult optimization problems.

These procedures are very widely used in data analysis because they are in general easy to compute and (especially when k is chosen to be 2 or 3) result in convenient visualizations of the embedded data. However, these algorithms can be very unstable in response to perturbations of the data (although there is a growing literature on robust variants of MDS and PCA, e.g., [89]), especially when noise

processes vary in different directions (e.g., see [411]). Under assumptions that the signal is low rank, variants known collectively as sparse PCA do a good job at recovering a sparse basis to describe the signal [552]. Sparsity of the data can also result in serious distortions; this is a particular problem in single-cell expression data. For a more extensive discussion of this problem and its relation to random matrix theory, see [15]. Another issue is that the optimal choice of k is a priori unknown. In practice, one often looks for an "eigenvalue gap," i.e., a natural splitting of the eigenvalues into a group of "large" eigenvalues and then a collection of much smaller eigenvalues. However, this procedure requires a threshold for deciding where the gap is, and is in general more of an art than a science.

A more serious issue from our perspective is the fact that when the data cannot be isometrically embedded as a Euclidean subspace of $\mathbb{R}^n$, PCA and MDS simply do not work particularly well to capture the intrinsic geometric structure. In Figure 4.3, we see that for a single curved ribbon in $\mathbb{R}^3$, PCA captures the intrinsic geometry with some distortion. But in Figures 4.4 and 4.5, for more complicated geometric objects (the union of two ribbons and a sphere in $\mathbb{R}^3$, respectively), PCA does not recover the intrinsic coordinates along the circle but rather just embeds a flattening of the circle in Euclidean space.

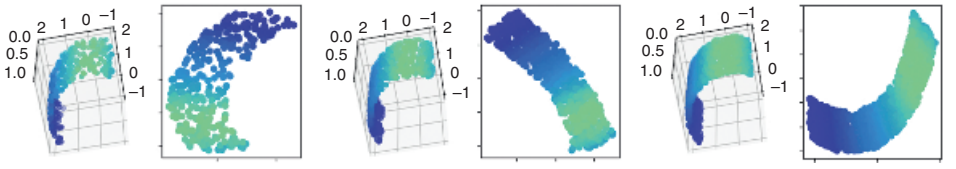

Figure 4.3 When the data lies on a single curved ribbon, the embedding into $\mathbb{R}^2$ exhibits distortion arising from the curvature.

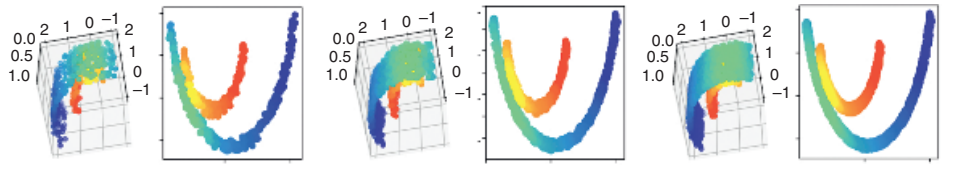

Figure 4.4 When the data lies on nested ribbons, the embedding is further distorted by the proximity of the two components.

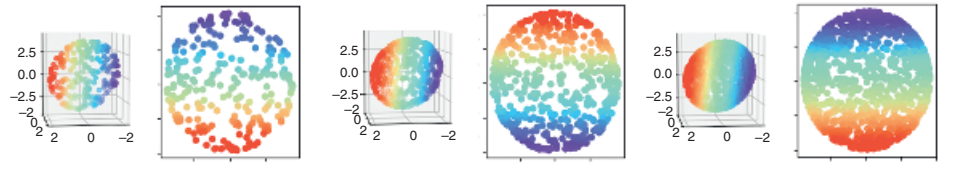

Figure 4.5 When the data lies on the standard sphere S^2, the embedding flattens the sphere and distorts the distances along an arbitrary axis.

Remark 4.2.2. When using the eigenvectors from PCA to describe the data, one possible source of difficulty in interpretation arises from the fact that the relevant linear combinations might include negative terms. In many applications, subtraction of basis vectors does not make sense. For example, many genomics applications of dimensionality reduction are interpretable only for positive combinations of terms. In this context, an algorithm called "non-negative matrix factorization" (NMF) is often used. In contrast to PCA, NMF is an iterative optimization procedure. See [319] for a classic rigorous discussion.

4.3 Manifold Learning

Suppose that we are given data points $\{x_1, \ldots, x_m\} \subseteq \mathbb{R}^n$, but we no longer assume that they admit a nearly isometric embedding as a hyperplane (i.e., an affine linear subspace). As we explained in Section 2.2, even when the points $\{x_i\}$ are produced by sampling from some embedding $\gamma\colon M \to \mathbb{R}^n$ of a compact Riemannian manifold M, the distance $\partial_{\mathbb{R}^n}(\gamma(x_i), \gamma(x_j))$ may not be very representative of the *intrinsic distance* $\partial_M(x_i, x_j)$. For example, as Figure 4.2 indicates, even when the manifold in question is homeomorphic to a plane, PCA and MDS can perform very poorly.

Consider the case of a line segment $\gamma\colon [0, 1] \to \mathbb{R}^2$ which is very twisted. Clearly, the distance along the curve $\gamma([0, 1])$ is poorly approximated by the Euclidean distance, especially near kinks. However, when γ is sufficiently smooth, there exists a feature scale at which Euclidean distances and intrinsic distances agree up to small error. In the work described in Section 2.2, this observation was leveraged to justify an algorithm for recovering the homology (and in fact homotopy type) of M. Here, we are interested in recovering coordinates on the manifold. This is a meaningful and potentially subtle question even in the case where M is contractible, and in fact most manifold learning algorithms focus on the case where M is contractible but the embedding $\gamma\colon M \to \mathbb{R}^n$ is twisted.

Manifold learning approaches ultimately rely on the fact that in favorable cases the manifold structure can be reconstructed by considering the "short distances" as reliable indicators of the intrinsic distance and ignoring the "long distances." One way to express this idea is to hypothesize that the basis determined by the k-nearest neighbors of a point z give a good approximation of the tangent plane to M at z.

4.3.1 Isomap

An early and prominent manifold learning algorithm is Isomap, which simply applies MDS to an empirical approximation of the intrinsic metric [495]. The

procedure works as follows. We assume we are given data points $\{x_1, \ldots, x_n\} \in \mathbb{R}^n$. We fix a scale parameter ϵ and a target dimension parameter k.

1. Form the weighted graph G with
   - vertices the points $\{x_i\}$, and
   - edges (i, j) with weight $w_{ij} = \partial_{\mathbb{R}^n}(x_i, x_j)$ when $\partial_{\mathbb{R}^n}(x_i, x_j) \leq \epsilon$.
2. We now form a new metric space X' with points $\{x_1, \ldots, x_n\}$ but distance given by the graph metric on G. Recall from Example 1.2.4 that this means that the distance between two vertices is the length of the shortest path in the graph. The graph metric can be efficiently computed, for example via Dijkstra's algorithm (e.g., see [125, 24.3]).
3. Finally, we use MDS to embed this new metric space into $\mathbb{R}^k$ as above, producing points $y_i = \theta(x_i)$.

When the points $\{x_i\}$ are sampled from a convex subset $M \subseteq \mathbb{R}^m$ embedded isometrically into $\mathbb{R}^n$, $k \geq m$, and ϵ is in the right range, Isomap can recover almost exactly the coordinates for M. (Here recall that a subset A of $\mathbb{R}^n$ is convex if for $x_1, x_2 \in A$, the line between x_1 and x_2 is entirely contained in A. In particular, this implies that A is contractible.) The recovery guarantees follow from the fact that for sufficiently dense sampling from a Riemannian manifold and suitable ϵ, the graph metric computed in the second step of the procedure approximates the underlying distance [51].

In Figure 4.6, we see that for a single curved ribbon in $\mathbb{R}^3$, Isomap does recover the intrinsic distances, with a small amount of distortion. But in Figures 4.7 and 4.8,

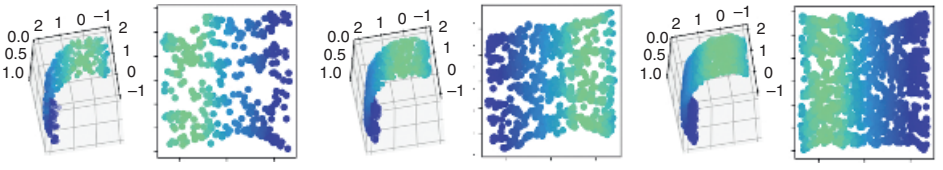

Figure 4.6 When the data lies on a single curved ribbon, Isomap does a good job of recovering the intrinsic coordinates.

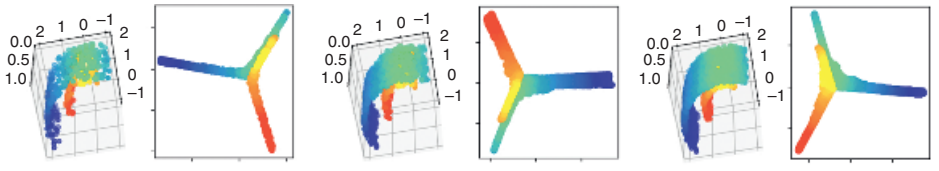

Figure 4.7 When the data lies on nested ribbons, Isomap collapses the two components to single lines.

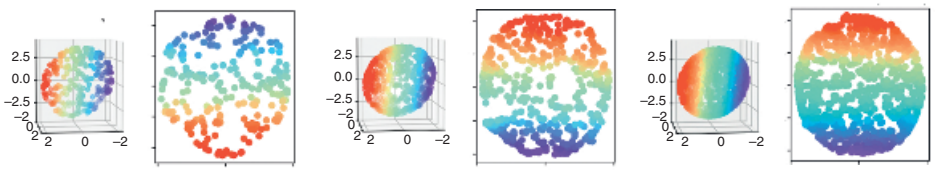

Figure 4.8 When the data lies on the standard sphere S^2 in $\mathbb{R}^3$, Isomap is not able to recover the intrinsic distances and embeds a flattening of the sphere in $\mathbb{R}^2$.

for more complicated geometric objects (the union of two ribbons and a sphere in $\mathbb{R}^3$, respectively), Isomap again fails to recover the intrinsic coordinates of the data. These examples illustrate some of the problems with Isomap.

1. An intrinsic issue is that MDS presumes that M can be isometrically embedded in Euclidean space; if it is not, the procedure seriously distorts the coordinates [335]. As a consequence, M must be flat in the sense of having zero curvature. Moreover, Isomap performs poorly on non-convex but contractible subspaces of Euclidean space, e.g., a space in the shape of the letter "Y."
2. Given new data points, the Isomap embedding has to be recomputed; there is no way to adapt an existing embedding.
3. A further issue in practice is that the algorithm is not robust to outliers and is very sensitive to differences in density or the precise value of ϵ. (For example, see [373] for discussion of these points.)
4. Finally, efficiency can also be a problem, especially for large numbers of samples. These issues arise both from the substantial costs of computing the graph metric and from the size of the resulting MDS problem. Some efforts to approach this by subsampling have been studied, e.g., see [460] for a sparse version of Isomap.

4.3.2 Local Linear Embedding (LLE)

A closely related approach is the *local linear embedding* (LLE) algorithm [439]. Once again, we assume we have data points $\{x_1, \ldots, x_n\} \subset \mathbb{R}^n$ and we fix a target dimension parameter k and a neighborhood size K.

1. For each point x_i, we solve for weights w_{ij} which minimize the expression

$$\mathcal{E}(x_i) = \sum_i \left(x_i - \sum_j w_{ij} x_j \right)^2 ,$$

subject to the constraints

$$\begin{cases} w_{ij} = 0 & x_j \text{ not a } K\text{-nearest neighbor of } x_i \\ \sum_j w_{ij} = 1. \end{cases}$$

Roughly speaking, we are solving for weights that optimally reconstruct each point x_i from its K-nearest neighbors. The weights can efficiently computed via least squares.

2. Embedding points $\{y_i = \theta(x_i)\} \subseteq \mathbb{R}^k$ are computed so that

$$\mathcal{E} = \sum_i \left(y_i - \sum_j w_{ij} y_j \right)^2$$

is minimized. This problem can be solved by computing the top k eigenvectors of the matrix corresponding to the associated quadratic form, subject to some nondegeneracy constraints.

Broadly speaking, LLE has fairly similar qualitative properties as Isomap; this is illustrated in Figures 4.9, 4.10, and 4.11. In practice, it turns out to work somewhat better than Isomap on samples of non-convex contractible subsets $M \subseteq \mathbb{R}^k$ (e.g., regions with dents in them), and also has the advantage that the eigenvector problem involves a matrix that is always sparse, and hence it can be run on substantially larger data sets than Isomap.

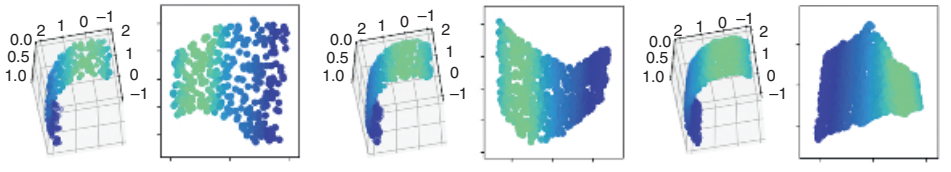

Figure 4.9 When the data lies on a curved ribbon, LLE does a good job of recovering the intrinsic coordinates and unfolding the ribbon.

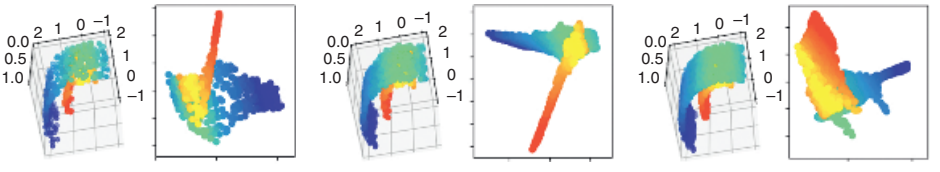

Figure 4.10 When the data lies on nested ribbons, LLE does a better job than PCA or Isomap but still engages in serious distortion of the intrinsic metric.

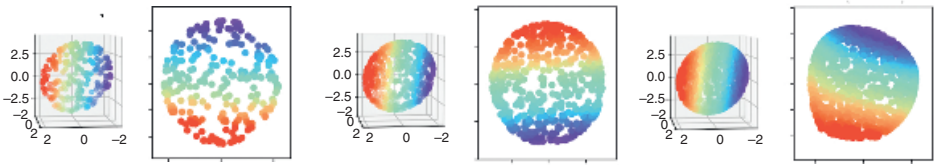

Figure 4.11 When the data lies on the standard S^2 in $\mathbb{R}^3$, LLE does not do a good job of capturing the intrinsic distances and simply embeds a flattening of the sphere.

4.3.3 Laplacian Eigenmaps

Isomap and LLE implicitly use the tangent plane of a manifold to perform local reconstruction of points. A more explicit use of the manifold structure is to try to exploit the existence of the Laplace-Beltrami operator, a map from functions on M to functions on M which is computed as the divergence of the gradient; on $\mathbb{R}^n$, this takes the classical form

$$\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}.$$

The first technique to take this approach is the *Laplacian eigenmaps* algorithm due to Belkin and Niyogi [45].

Once again, we assume we are given data points $\{x_1, \ldots, x_k\} \subseteq \mathbb{R}^n$ and we form a neighborhood graph that captures the "small" distances between points. Precisely, we fix a width parameter σ and proceed as follows.

1. Form the weighted graph G with
   - vertices in bijection with the points $\{x_i\}$, and
   - edges (i, j) with weight $w_{ij} = e^{-\frac{\partial(x_i, x_j)^2}{\sigma}}$ when $\partial(x_i, x_j) \leq \epsilon$.
2. We let D denote the diagonal matrix specified by $D_{ii} = \sum_j w_{ij}$ and define the *graph Laplacian* as $L = D - W$, where W is the matrix of edge weights from G.
3. We solve $Lf = \lambda Df$ for the top k eigenvectors, which determine the embedding; we form the matrix with columns these eigenvectors, and the rows are the embedded points $\{y_i\}$. This procedure can be viewed as finding a solution to the optimization problem of determining $\{y_i\}$ that minimize

$$\mathcal{E} = \sum_{i,j} (y_i - y_j)^2 W_{ij},$$

   i.e., finding an embedding that penalizes nearby points x_i and x_j being sent to distant points y_i and y_j.

Here the basic technical underpinning is one of the fundamental insights of spectral graph theory, namely that the graph Laplacian we describe above shares many interesting properties with the Laplacian of a manifold [116]. Moreover, as a basic consistency check, when the points $\{x_i\}$ are sampled from a compact Riemannian manifold, as the number of points increase and σ decreases, the graph Laplacian converges in a precise sense to the Laplace-Beltrami operator on the manifold [46].

As with Isomap and LLE, Laplacian eigenmaps is expected to work best on convex subsets of $\mathbb{R}^n$; like LLE, the eigenvector problems involved tend to be sparse and so Laplacian eigenmaps can handle comparatively larger data sets. However, as Figures 4.12, 4.13, and 4.14 indicate, Laplacian eigenmaps has distinctly worse performance than either Isomap or LLE. We discuss the method due to its historical importance and conceptual clarity.

Remark 4.3.1. A refinement of the Laplacian eigenmaps algorithm, called Hessian eigenmaps [146], uses a discretized version of the Hessian matrix of second

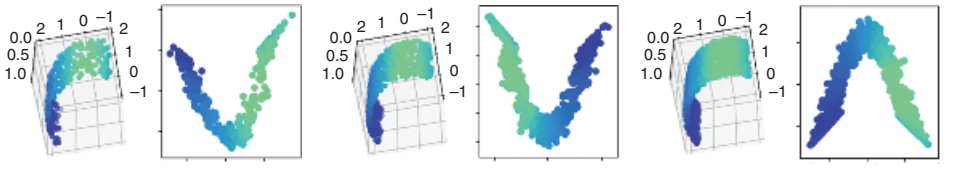

Figure 4.12 When the data lies on a single curved ribbon, Laplacian eigenmaps does not unfold the ribbon properly.

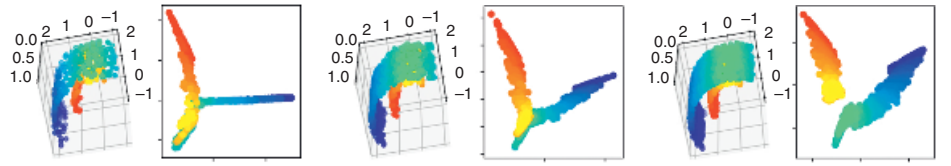

Figure 4.13 When the data lies on nested ribbons, Laplacian eigenmaps does in fact manage to recover some aspects of the relationship between the ribbons, although each one is compressed and distorted.

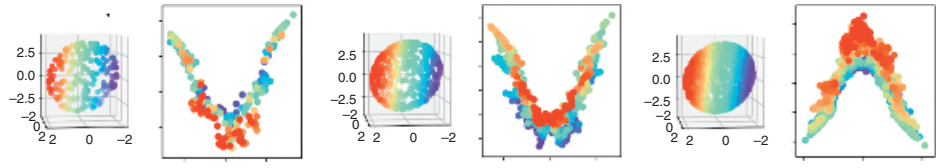

Figure 4.14 When the data lies on the standard S^2 in $\mathbb{R}^3$, Laplacian eigenmaps does not do a good job of capturing the intrinsic distances.

partial derivatives. The advantage of using the Hessian is improved theoretical guarantees: the Hessian eigenmaps algorithm can be shown to be asymptotically correct for arbitrary connected subsets of $\mathbb{R}^n$. Although in idealized situations Hessian eigenmaps outperforms other manifold learning algorithms, in practice it does not work well – estimating second derivatives is well known to be numerically unstable.

4.3.4 Manifold Learning and Kernel Methods

There is a basic resemblance between all of the manifold learning techniques described in the preceding subsections; at a high level, it appears that they are relying on similar geometric ideas. It turns out that this connection can be made precise using a standard body of techniques for handling nonlinearity in data analysis, *kernel methods*.

The basic idea is to choose a nonlinear embedding Ψ of the points $\{x_i\}$ into an infinite-dimensional inner-product space H, and use the inner product and norm on H to analyze the points. The idea is that Ψ will *unfold* the data so that linear techniques applied to $\Psi(x_i\}$ will reveal nonlinear structure in $\{x_i\}$. For example, *kernel PCA* performs PCA on the embedded points and is frequently applied in conjunction with clustering algorithms.

Although it is easy to see that for a suitable nonlinear embedding, such techniques would be very effective, a number of questions about how to implement this procedure arise.

1. It is not clear how easy it will be to produce a suitable map Ψ without possessing a priori knowledge of the data, and
2. it is not clear that working directly in the infinite-dimensional space H is algorithmically tractable.

The key insight that makes kernel methods effective is the observation that a wide variety of algorithms (including clustering and PCA) can be computed without explicit knowledge of Ψ or H provided one has access to a *kernel* K, which is a map

$$K\colon X \times X \to \mathbb{R}$$

such that

$$K(x_i, x_j) = \langle \Psi(x_i), \Psi(x_j) \rangle_H,$$

where the expression on the right denotes the inner product in H. This formulation reduces the question to producing a kernel K which is algorithmically tractable

and also encodes geometric information about X. The construction of such kernels turns out to be much more tractable than producing Ψ.

Example 4.3.2.

1. A standard kernel is the *radial basis function* or Gaussian kernel

$$K(x_i, x_j) = e^{-\frac{\partial_X(x_i,x_j)^2}{2\sigma^2}}.$$

   This kernel is standard in classification and clustering applications.
2. On the set of trees (acyclic connected graphs), an interesting kernel is the *subtree kernel* which is defined as

$$K(t_i, t_j) = \#\{\text{isomorphic subtrees of } t_i \text{ and } t_j\}.$$

   This kernel is frequently used in analysis of linguistic and phylogenetic data. (See [121] for an early paper on kernel methods in natural language processing.)
3. On the set of strings, the (k, m) mismatch kernel is defined as

$$K(w_i, w_j) = \#\{\text{substrings } s_1, s_2 \text{ of } w_i \text{ and } w_j \text{ such that } |s_1| = |s_2| = k \text{ and } s_1, s_2 \text{ agree up to } m \text{ mismatches}\}.$$

   This kernel has seen notable applications to protein matching [322].

It now turns out that many manifold learning approaches can be interpreted as kernel PCA; notably, we can represent Isomap, LLE, and Laplacian eigenmaps in this fashion [225]. For example, for a data set $\{x_i\}$, Isomap is (up to scaling) identical to kernel PCA for the kernel

$$K(x_i, x_j) = \left(-\frac{1}{2}(I - ee^T)D_2(I - ee^T)\right)_{ij},$$

where D_2 denotes the matrix of squared distances, and e denotes the vector with all entries 1.

4.3.5 Discrete Harmonic Analysis

As we have seen, many manifold learning algorithms essentially involve an approximation to the Laplace-Beltrami operator on a manifold via the graph Laplacian. That is, one way to think about the underlying mathematics of manifold learning is in terms of discrete approximations of heat flow on the underlying manifold. In classical physics, the Laplace operator Δ arises in the heat equation, which in its simplest form can be written

$$\frac{\partial f}{\partial u} = \Delta f.$$

The heat equation describes how the geometry of a manifold interacts with local temperature information to control the diffusion of heat over time.

Elaborating on this perspective, work of Jones, Coifman, Maggioni and collaborators has produced a large body of work on *discrete harmonic analysis* in terms of the heat kernel, the function that describes the infinitesimal flow. Two basic observations in the mathematical setting are that:

- the heat flow on the manifold describes the geometry of the manifold, and
- harmonic analysis (a generalization of Fourier analysis) in terms of the basis of powers of the heat kernel gives a good description of functions on the manifold.

The idea of discrete harmonic analysis is to analyze data using discretized approximations to the heat flow. Given the data represented as a finite metric space (X, ∂_X), we fix a rapidly decaying function $K(x_i, x_j)\colon X \times X \to \mathbb{R}$. For example, a standard similarity measure is given by

$$K(x_i, x_j) = e^{-\frac{\partial_X(x_i, x_j)}{\sigma}},$$

where σ is a width parameter. We now proceed as follows.

1. Form the weighted graph G with vertex set in bijection with the points $x \in X$ and an edge (x_i, x_j) of weight $w_{ij} = K(x_i, x_j)$ provided that $K(x_i, x_j) > 0$.
2. Writing $D_{ii} = \sum_j w_{ij}$, define the (normalized) graph Laplacian to be

$$\mathcal{L} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}},$$

where $W_{ij} = w_{ij}$. (Notice that there is a slight difference from the graph Laplacian used in Section 4.3.3.)

The discrete version of the heat flow is given by the random walk on the graph, sometimes referred to as the diffusion walk in this situation; this is a Markov process on the vertices where the vertex at time t is selected from the neighbors of the vertex at time $t - 1$ according to the edge weights. Precisely, the probability of moving to vertex k from vertex i is

$$p_{ki} = \frac{w_{ki}}{\sum_j w_{kj}}.$$

The random walk on the graph G above has transition probabilities determined by $D^{-1}W$. It is sometimes useful to make this walk symmetric (i.e., to ensure that $p_{ki} = p_{ik}$); in this case, the transition probabilities for the symmetrized walk are given by

$$T = I - \mathcal{L} = D^{-\frac{1}{2}}WD^{-\frac{1}{2}}.$$

Roughly speaking, the diffusion walk on the graph describes how a point mass at a given point in the graph spreads out over time as it diffuses according to the edge weights. In the limit as the number of points increases and σ decreases, the diffusion walk converges to the actual heat flow on the manifold described by the heat equation.

The key observation is now that harmonic analysis of T gives rise to geometric descriptions of the data set. To explain, the eigenfunctions of the powers T^t determine a metric on X; this is the so-called "diffusion distance" at scale t. Specifically, if we denote the eigenvectors of T (regarded as an L^2 operator) by $\{\Psi_k\}$ and the associated eigenvalues $\{\lambda_k\}$, the diffusion distance at scale t is given by

$$D_t(x, y)^2 = \sum_{k \geq 0} \lambda_k^{2t} (\Psi_k(x) - \Psi_k(y))^2.$$

Since T and its powers describe an ergodic Markov process, the eigenvalues λ_i satisfy $|\lambda_0| = 1$ and

$$|\lambda_0| \geq |\lambda_1| \geq |\lambda_2| \geq \dots.$$

Truncating the expression for D_t by removing eigenfunctions corresponding to eigenvalues smaller than some threshold ϵ provides a tractable approximation. Roughly speaking, the diffusion distance between two points is a measure of how connected the points are; i.e., it reflects the probability of moving from x to y in the diffusion walk.

Moreover, we can use eigenvectors of T to embed the data $\{x_i\}$ in $\mathbb{R}^k$ so as to optimally preserve the diffusion distance; just as in the manifold learning algorithms described above, we put the scaled eigenvectors $\lambda_i^t \Psi_i$ as the columns of a matrix and take the rows to compute the embedding. This embedding is such that the Euclidean distance between the embedded points is close to the diffusion distance on G.

Furthermore, wavelet bases for functions on the data can be constructed using powers of T; this gives a geometric basis for representing functions. The diffusion process can also be used to smooth data before applying machine learning algorithms. See [119] for the original paper; more generally, Maggioni's research group has an extensive bibliography.

4.3.6 Other Manifold Learning Techniques

We have chosen to highlight manifold learning techniques that are conceptually significant and of historical importance; however, this has subsequently become a very active area of research. There are now many other techniques, each with slightly different virtues, insights, and limitations. As a few examples, some interesting methods include the following.

1. *Local tangent space alignment* [519], which uses the nearest neighbors of a point to estimate the tangent space locally and then performs a global optimization to align these compatibly.
2. *Maximum variance unfolding* [532], which forms a neighborhood graph that maximizes distance between far-away points by solving a semidefinite programming (SDP) problem and then computes eigenvectors, and uses these as the basis for an embedding.
3. *Manifold charting* [69], which solves for local neighborhood coordinate patches for each point and then sews them together using a global optimization process.

4.3.7 Manifolds of Differing Dimension

An obvious extension of the setup for manifold learning is the case where the data is generated from the union of manifolds of differing dimension. This situation is the simplest case of the general problem of "stratified space learning" (see Example 1.11.9). Of course, ad hoc adaptations of manifold learning techniques could be used: for example, cluster points by some estimate of local dimensionality (so that points in a cluster come from a subset of roughly constant dimension) and then apply manifold learning techniques to each cluster separately.

However, it is reasonable to expect that more systematic approaches would be superior. So far, there have been two main settings studied.

1. The data is assumed to lie on the union of hyperplanes of different dimensions; i.e., $M = \cup_i \mathbb{R}^{n_i}$, where each $\mathbb{R}^{n_i}$ is presented as embedded in an ambient space $\mathbb{R}^m$ via a linear map $\gamma_i \colon \mathbb{R}^{n_i} \to \mathbb{R}^m$ [481].
2. The data is assumed to lie on a metric graph. Recall from Example 1.11.9 that these are stratified spaces with a zero dimensional stratum for the vertices and a one dimensional stratum for the edges. We discuss the special case of trees further below in Section 4.7.1. See for example [106] for an approach to general graphs using Reeb graphs.

(Although note that [48] studied the problem of clustering points into different strata, using estimates of local homology.)

4.4 Neighbor Embedding Algorithms

In this section, we discuss a different approach to dimensionality reduction, *stochastic neighbor embedding* (SNE) [242] and its more popular descendant t-distributed stochastic neighbor embedding (t-SNE) [336]. One issue with many of the manifold learning algorithms we have discussed so far is that they do not work well when the data points are of non-uniform density. The stochastic

neighbor embedding algorithms address this by constructing a similarity measure between points that reflects the local density around each point. They are much more explicitly probabilistic (and less geometric) in their design.

4.4.1 Stochastic neighbor Embedding (SNE)

Let $\{x_1, \dots, x_m\} \subset \mathbb{R}^n$ denote the data and $\{y_1, \dots, y_m\} \subset \mathbb{R}^k$ denote a candidate set of corresponding image points, for $k \leq n$. Then we define

$$p_{j|i} = \frac{e^{-\frac{\partial_{\mathbb{R}^n}(x_i, x_j)^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{\partial_{\mathbb{R}^n}(x_i, x_k)^2}{2\sigma_i^2}}}$$

and

$$q_{j|i} = \frac{e^{-\partial_{\mathbb{R}^n}(y_i, y_j)^2}}{\sum_{k \neq i} e^{-\partial_{\mathbb{R}^n}(y_i, y_k)^2}},$$

where the variances σ_i are obtained by an optimization process we will describe shortly and in the second equation we are fixing all of the variances to be identically $\frac{\sqrt{2}}{2}$. We set $p_{i|i} = q_{i|i} = 0$.

The idea behind SNE [242] is that good image points $\{y_1, \dots, y_n\}$ have the property that the difference between $p_{j|i}$ and $q_{j|i}$ is minimized, in the sense that we minimize the summed Kullback-Leibler divergences via the cost function

$$C = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}.$$

(See Remark 3.2.32 for discussion of the Kullback-Leibler divergence as a dissimilarity measure on probability distributions.) Notice that ensuring these local distributions are similar means that SNE is sensitive to variation in density; the density around a point is explicitly represented in the cost function.

Remark 4.4.1. Recall that the Kullback-Leibler divergence is a dissimilarity measure but not a metric: it is not symmetric. In the context of SNE, this asymmetry is a regarded as a feature – it serves to enforce a preference for preserving local distances.

In practice, the SNE algorithm proceeds by solving for minimizing points $\{y_i\}$ via a gradient descent procedure (often with diminishing amounts of noise added as a form of simulated annealing) in order to find a good local minimum for C. Convergence is often slow and depends critically on good choices of the variances σ_i, which we now explain how to obtain. In principle, differing choices of σ_i

amount to enforcing variable numbers of neighbors used to do the local estimation of the coordinates; this is expressed here via the use of the "perplexity," which is computed as

$$P = 2^{-\sum_j p_{j|i} \log_2 p_{j|i}}.$$

Roughly speaking, P controls the number of effective neighbors that are used; recall that this is a loose estimate of the local dimension. The desired perplexity (typically in the interval $[10, 100]$) is a parameter, and we solve for values σ_i which achieve the perplexity. The output of the algorithm can be fairly sensitive to the choice of perplexity value.

4.4.2 t*-Distributed Stochastic Neighbor Embedding (*t*-SNE)*

In practice, a refinement of stochastic neighbor embedding is commonly used. One of the problems with the original SNE procedure is that the gradient descent optimization procedure is slow and it can be difficult to get it to converge. Another problem afflicts the standard application of SNE to visualization. Specifically, in order to use SNE to visualize data, one solves for embedded points $\{y_i\}$ in $\mathbb{R}^2$ or $\mathbb{R}^3$. When there are even a moderately large number of points, the cost function can cause compression of the points so that they all lie very close to the center of mass, which makes the visualization hard to use. (This compression also defeats clustering algorithms.)

To resolve these problems, van der Maaten and Hinton [336] proposed the variant algorithm *t*-SNE. This is quite similar to SNE, with the following modifications.

1. We symmetrize $p_{j|i}$ as follows:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2}.$$

2. We define a symmetrized variant of $q_{j|i}$ as follows:

$$q_{ij} = \frac{\left(1 + \partial_{\mathbb{R}^m}(y_i, y_j)^2\right)^{-1}}{\sum_{k \neq \ell} \left(1 + \partial_{\mathbb{R}^m}(y_k, y_\ell)^2\right)^{-1}}.$$

   In the original definition, the $q_{j|i}$ was defined using a Gaussian; this expression replaces that with the Student *t*-distribution with one degree of freedom (i.e., a Cauchy distribution), which has more weight in the tails.
3. Finally, the cost function is replaced with the alternative expression

$$C = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

Once again, the cost function is optimized via a gradient descent procedure. These modifications have a number of interesting consequences.

1. The use of a heavier tailed distribution in the embedding space means that outliers have less impact on the overall results and the compression effects around the center of mass are alleviated to some degree (although there are still upper bounds on the number of points that can reasonably be embedded before "clumping" occurs).
2. The adjusted formula for q_{ij} also has the effect of substantially improving the efficiency and quality of the gradient descent procedure.

Figures 4.15, 4.16, and 4.17 show that the t-SNE procedure can produce very reasonable embeddings recovering local geometry; in particular, t-SNE arguably performs best of the methods we have examined on the nested ribbons.

However, caution is required when interpreting the results of t-SNE. In contrast to other dimensionality reduction methods, t-SNE does not directly depend on a discretization of the Laplace-Beltrami operator or approximation of the local tangent

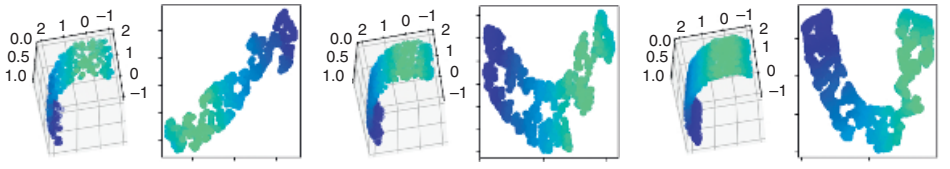

Figure 4.15 Especially at lower densities, t-SNE unfolds the ribbon to recover its intrinsic coordinates.

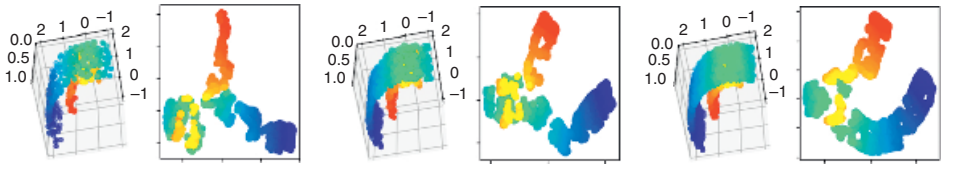

Figure 4.16 When the data lies on the nested arcs, t-SNE actually does a reasonable job at recovering the intrinsic coordinates.

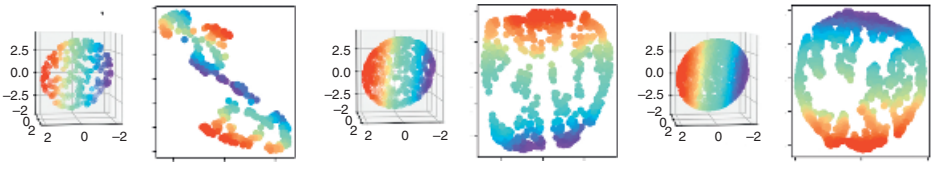

Figure 4.17 When the data lies on the standard S^2 in $\mathbb{R}^3$, t-SNE does not do a good job of capturing the intrinsic distances and instead flattens the sphere.

planes. Put another way, it is more sensitive to different properties of the underlying geometric object than other manifold learning methods. On the one hand, this flexibility can be a real asset when working with data that does satisfy the manifold hypothesis. On the other hand, the geometric properties of $\{y_i\}$ can be difficult to relate to the geometric properties of $\{x_i\}$. For example, when clustering points after computing the t-SNE embedding in $\mathbb{R}^2$, inter-cluster distances do not reflect global properties faithfully, and relative sizes of clusters are usually meaningless. Moreover, there are no geometric theoretical guarantees about the ideal behavior of t-SNE. We now turn to discuss an application highlighting best practice in using t-SNE.

4.4.3 Reliable Use of t-*SNE*

A common usage pattern for t-SNE is to project into $\mathbb{R}^2$ and then apply a standard clustering method. This application of the algorithm has had some impressive successes, as a method which adjusts for local density can reveal clusterings which would not be evident using methods which impose a global constraint. A celebrated application is the viSNE procedure [13], a tailored use of t-SNE, which has been used for visualizing and classifying single-cell expression data, notably to distinguish healthy and cancerous bone marrow samples. We highlight the protocol here as it provides an exemplary case study of how to robustly apply dimensionality reduction.

The basic approach of viSNE applies t-SNE to embed gene expression data collected from single-cell bone marrow samples, regarded as vectors in $\mathbb{R}^n$ with the correlation metric, into $\mathbb{R}^2$ and then performs clustering on the embedded data. The overall conclusion is that in this embedding, healthy bone marrow cells are close together across samples and quite far from cancerous samples. This conclusion was carefully validated.

1. The stability of each clustering was tested via standard cross-validation; some data points were removed and deviation in the clusters was measured.
2. In order to handle the limits on the total numbers of points for embeddings in $\mathbb{R}^2$, the algorithm was run repeatedly on subsamples from the data. The clusters were compared to ensure that the analysis was robust to this subsampling.
3. To demonstrate more global stability, samples from different normal patients were compared (and observed to be extremely similar in terms of the resulting clusterings).
4. To ensure that the results were not artifacts of experimental procedure, different experimental methods for obtaining the expression data were compared by contrasting the resulting clusters.

The procedures outlined above give very good confidence that the results of the viSNE procedure are capturing real geometric information about the expression data of bone marrow. In general, this example is a good model for an analytical protocol for topological data analysis.

4.5 Mapper and Manifold Learning

In principle, one could use the coordinate charts provided by manifold learning algorithms for all sorts of geometric inference about the data. In practice, dimensionality reduction procedures are most commonly used as a preprocessing step before applying some kind of clustering algorithm. From the perspective of TDA, a very interesting extension of this approach is to use the output of manifold learning algorithms as filters for Mapper.

Recall from Example 2.8.3 that using PCA coordinates of expression data as a filter function for Mapper captured cell differentiation trajectories [431]. A reasonable question to ask is what Mapper adds over standard manifold learning; to answer this, we can directly compare the output of Mapper to the output of various manifold learning procedures. In Figure 4.18, we represent both the output of Mapper and the raw outputs of PCA, MDS, and t-SNE for the differentiation process.

The results are informative.

1. Mapper is only very slightly better than MDS and t-SNE for estimating a cell's position along the differentiation trajectory.
2. However, the graphical representation of Mapper contains additional informative structure; the loops and flares in the resulting Mapper graph are biologically relevant.

In general, we expect that this kind of fusion of topological data analysis and dimensionality reduction will provide a useful technique for describing the structure of genomic data.

4.6 Dimensionality Estimation

Recall that the basic operating assumption in dimensionality reduction is that the data points $\{x_i\} \subseteq \mathbb{R}^n$ lie on a geometric object that has much lower intrinsic dimension k than the ambient space. As such, a natural problem to consider is whether we can directly determine the dimension of $\{x_i\}$ without actually computing a description of the lower dimensional object.

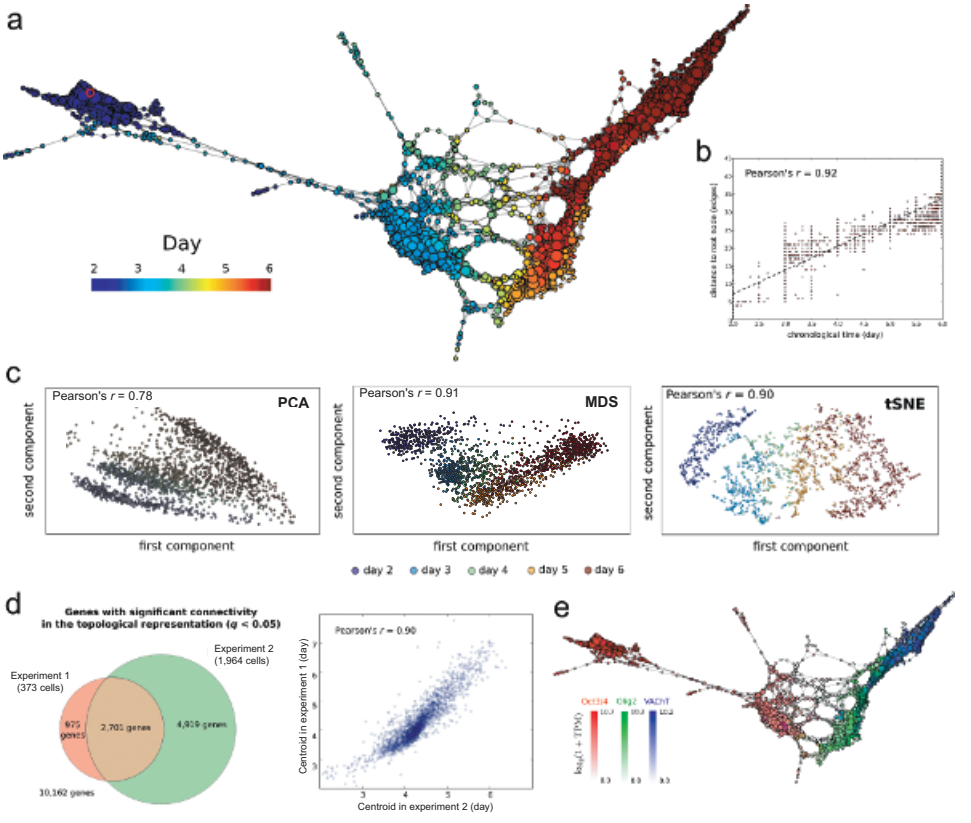

Figure 4.18 The differentiation timeline can be extracted from various manifold learning techniques. Source: [431]. From Abbas H. Rizvi et al., *Nature Biotechnology* 35, 551-560 (270). © 2017 Nature. Reprinted with Permission from Springer Nature.

There are several reasons why we might want to have efficient approaches to this problem. Although the very idea of intrinsic dimension presupposes the data has enough geometric structure to give rise to a notion of dimension, there are many classes of objects that have a good definition of dimension which are not manifolds (e.g., fractals). Moreover, estimating the intrinsic dimension can provide a sense of how good low-dimensional summaries can be; for example, projecting a data set with intrinsic dimension 4 into $\mathbb{R}^2$ will typically result in much less distortion than projecting a data set with intrinsic dimension 50 into $\mathbb{R}^2$. Finally, understanding the intrinsic dimension gives us a sense of the number of sample points required to accurately estimate geometric features of the underlying object.

There are a number of ways to try to directly estimate the dimension of the points $\{x_i\}$; almost all of them consider the rate of growth of the number of points within an ϵ-ball as ϵ increases. For example, the correlation dimension [205] of a set of points $\{x_1, \ldots, x_n\}$ is computed by considering the curve produced by plotting

$$(x, y) = \left(\log \epsilon, \frac{2}{n(n-1)}\theta_\epsilon\right),$$

where θ_ϵ is the count of the number of pairs (x_i, x_j) such that $\partial(x_i, x_j) \leq \epsilon$, and performing a regression to estimate the slope of this curve. The correlation dimension has proven useful in handling estimation of dimension for geometric objects like fractals that are not well described by more classical measures of dimension.

When working under the manifold learning assumptions, i.e., that the data is given as points $\{x_i\} \subset \mathbb{R}^n$ sampled from a k-dimensional manifold, a more geometric version of this idea can be applied. The basic observation is that when points are sampled from a density ρ in $\mathbb{R}^k$, the number of points expected in a ball of radius ϵ centered around z is approximately $\rho(z)$ times the volume of the ball. Therefore, empirical estimates of the rate of growth of the count of sample points in Euclidean balls of expanding radius can be used to estimate dimension. A very clean form of this approach is given by the maximum-likelihood estimator of Levina and Bickel [327], which assumes a Poisson distribution for the data and is given at the point x by the formula

$$\left(\frac{1}{N(R, x)} \sum_{j=1}^{N(R,x)} \log \frac{R}{T_j(x)}\right)^{-1},$$

where $N(R, x)$ is the number of points in the ball of radius R around x and $T_j(x)$ is the distance from x to the jth point in this ball. To compute a global estimate, we can average the likelihood estimators.

Remark 4.6.1. MacKay and Ghahramani observe that the estimator above has substantial bias (even for low dimensions) which can be corrected by a slightly different approach to combining the points, namely again using maximum likelihood estimation. This amounts to computing the following:

$$m^{-1} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{N(R,x_i)} \log \frac{R}{T_j(x_i)}}{\sum_{i=1}^{n} N(R, x_i)}.$$

Notice that this is very similar to the original approach; here we are just averaging the inverses. Numerical experiments suggest this correction reduces both bias and variance.

Another interesting approach in the manifold setting, due to Little et al. [332], combines local PCA estimates of the dimension at various scales. The idea is around each data point z to choose k-nearest neighbors and perform PCA on the vectors determined by the pairs of z and a nearest neighbor to obtain a local estimation of the tangent plane. More precisely, we perform the following algorithm.

1. For each point z, we compute the eigenvalues $\lambda_1(r), \ldots, \lambda_K(r)$ of the covariance matrix
$$C = \frac{1}{n} \sum_i x_i x_i^T,$$
restricted to the data points $x_i \in B_r(z)$, as r varies. (We assume that the data has been centered.)
2. Ideally, we will see the magnitudes of the eigenvalues cluster into two groups, i.e., there will be a substantial eigenvalue gap over a broad range of values of r. We regard the smallest ones as noise. We then restrict attention to the region of the eigenvalue curves (i.e., plots of the values of the kth eigenvalue as a function of r) where the rate of growth of the noise eigenvalues is flat.
3. Of the remaining eigenvalues, we separate those having linear growth as a function of r from those having quadratic growth via a regression. The linear growth eigenvalues are regarded as corresponding to eigenvectors in the direction of the local tangent plane, whereas the quadratic growth eigenvalues are coming from eigenvectors in the direction of the local curvature of the manifold. The number of linear growth eigenvalues provides a local dimension estimate at z.
4. Finally, we average the dimension estimates over all points z.

4.7 Metric Trees and Spaces of Phylogenetic Trees

In this section we explain an approach to the analysis of phylogenetic trees based on endowing sets of trees with geometric structure. The importance of phylogenetic tree structures in biological sciences cannot be overstated; their use begins with Darwin's proposal of the tree as a metaphor for the process of species generation through branching of ancestral lineages [133]. Since then, tree structures have become ubiquitous in biology for describing evolutionary relations: notably, clonal evolution events that start from asexual reproduction of a single organism (primordial clone), which mutates and differentiates into a large progeny (see the left panel of Figure 4.19) [293]. Examples of these processes include single gene phylogeny in non-recombinant viruses, bacteria that are not involved in horizontal gene transfer events, and metazoon development from a single germ cell.

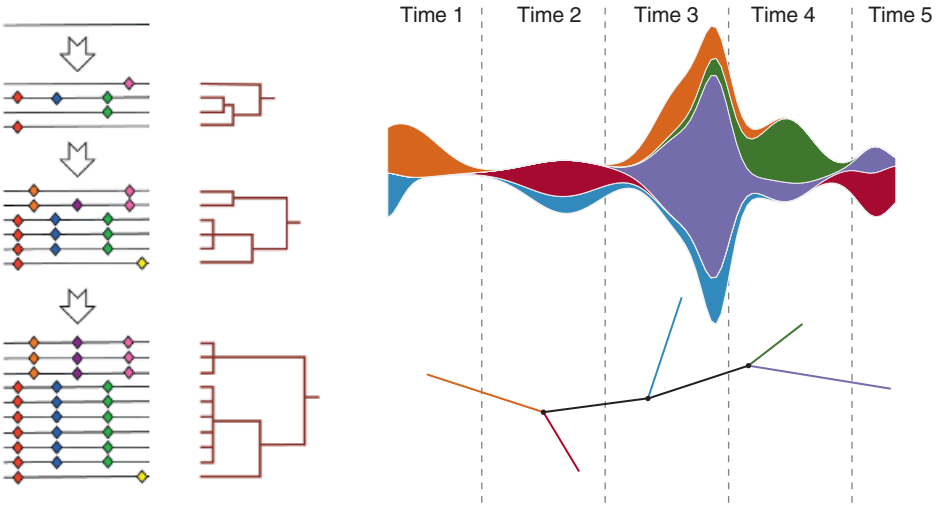

Figure 4.19 Clonal evolution of an asexually reproducing organism. Left: Through acquiring mutations and differentiation, the primordial clonal organism gives rise to a large heterogeneous population, whose evolution can be described with tree-like structures; here horizontal lines represent organisms, and symbols on the line represent mutations. Right: Longitudinal sampling of a clonal population permits the construction of phylogenetic trees that describe its evolutionary history. Here, subpopulations are represented by different colors; subsampling of a particular subpopulation is illustrated by the color of the branch in the tree, one of the many trees that can be reconstructed from this population. Source: [545]. From Zairis et al., Genomic data analysis in tree spaces, arXiv: 1607.07503 [q-bio.GN].

There are a number of basic mathematical and algorithmic questions that arise in this context.

1. Given genomic data, how can we fit a "best" phylogenetic tree to this data that optimally encodes the evolutionary relationships in the data?
2. Given two trees, how can we assess quantitatively how different they are? Given a collection of trees, can we compute a "summary" or average tree?
3. More generally, how can we describe probability distributions on trees? (For example, more sophisticated output of algorithms to answer the first question might provide a distribution of trees.)

There is a tremendous body of work on the first question; in the first part of this section, we focus on a purely metric method (neighbor-joining) that takes as input a finite metric space and produces a corresponding metric tree. We give a rapid but more comprehensive treatment of tree inference algorithms in Appendix C.

In the remainder of the section, we assume that our raw data has been turned into phylogenetic trees, and describe an approach to the second and third questions based on using the *metric geometry* (Section 4.7.3) associated to a specific metric on the set of phylogenetic trees.

4.7.1 Inferring Trees from Metric Data

One way to formulate the problem of inferring a phylogenetic tree structure from genomic data is to regard the data as a finite metric space (X, ∂_X) and postulate that the metric ∂_X is a *tree metric*, i.e., the points correspond to the leaves of a tree and the distance corresponds to the length of the shortest path in the graph. (Recall the discussion of graph metrics from Example 1.2.4.)

Definition 4.7.1. A *phylogenetic tree* with m leaves is a weighted, connected graph with no circuits, having m distinguished vertices of degree 1 labeled $\{1, \ldots, m\}$ (referred to as *leaves*), and all other vertices of degree ≥ 3.

We refer to edges that terminate in leaves as *external* edges and the remaining edges as *internal*. We will use the term tree metric to refer to the metric induced on the leaves from the graph metric of the phylogenetic tree.

Of course, not every metric arises from a tree metric. Specifically, given a metric space (X, ∂_X), the metric ∂_X is a tree metric if and only if it satisfies the *four point condition* [82].

Lemma 4.7.2. *A metric space (X, ∂_X) is isometric to a tree metric space if and only if for any $u, v, w, x \in X$, two of the three sums*

$$\partial_X(u, v) + \partial_X(w, x), \qquad \partial_X(u, w) + \partial_X(v, x), \qquad \partial_X(u, x) + \partial_X(v, w)$$

are equal and greater than the third.

But although this can be used as a test, it does not provide an algorithm for producing a tree. A good solution to the tree inference problem then ideally has the following properties.

1. When ∂_X really is the metric corresponding to a tree metric, the algorithm recovers a tree such that the associated metric is isometric to the input.
2. When ∂_X is "close" to a tree metric in a suitable sense, the algorithm recovers a tree T such that the associated tree metric is close to ∂_X.

An influential method to do this is *neighbor-joining* [442], which recursively constructs the output tree by selecting a pair of points, joining them as leaves coming out from an internal vertex, and then repeating the process with the new vertex

regarded as a leaf and the joined points removed, until all of the points are part of the tree. More precisely, the algorithm works as follows. We assume we are given as input a finite metric space (X, ∂_X) such that $|X| = n$.

1. We initialize the output tree T to have a vertex for each point of X, each connected to a central root and with no other edges.
2. Calculate the function
$$Q(x_i, x_j) = \partial_X(x_i, x_j) - \frac{1}{n-2}\left(\sum_{k\neq i} \partial_X(x_i, x_k) + \sum_{k\neq j} \partial_X(x_j, x_k)\right).$$
3. Find the points x_i and x_j that minimize $Q(x_i, x_j)$.
4. Define a new point z, and form T' from T by adding edges from x_i and x_j to z, deleting the edges from x_i and x_j to the root, and connecting z to the root. Define the edge weights as
$$w_{x_i,z} = \frac{1}{2}\left(\partial_X(x_i, x_j) + \frac{1}{n-2}\left(\sum_{k\neq i} \partial_X(x_i, x_k) - \sum_{k\neq j} \partial_X(x_j, x_k)\right)\right)$$
and
$$w_{x_j,z} = \frac{1}{2}\left(\partial_X(x_i, x_j) + \frac{1}{n-2}\left(\sum_{k\neq j} \partial_X(x_j, x_k) - \sum_{k\neq i} \partial_X(x_i, x_k)\right)\right).$$
5. Form the discrete metric space $X' = X - \{x_i, x_j\} \cup \{z\}$, with
$$\partial_{X'}(x_k, z) = \frac{1}{2}(\partial_X(x_i, x_k) + \partial_X(x_j, x_k) - \partial_X(x_i, x_j)).$$
6. If X' consists only of $\{z\}$, terminate and return T'. Otherwise, return to step 2 with T' and X' in place of T and X.

First, the algorithm is sound, in the sense that when the metric ∂_X actually is a tree metric, the neighbor-joining algorithm recovers the tree. More interestingly, it is fairly robust to noise; notice that neighbor-joining does not really require a metric space as input, as the triangle inequality is never used. We have the following consistency result [23].

Theorem 4.7.3. *Let (X, ∂_X) be a tree metric space and $D\colon X \times X \to \mathbb{R}$ a function satisfying*
$$|D(x, y) - \partial_X(x, y)| \leq \frac{1}{2} \min_{\substack{x_1,x_2\in X \\ x_1\neq x_2}} \partial_X(x_1, x_2)$$
for all $x, y \in X$. Then neighbor-joining applied to D returns a metric space isometric to (X, ∂_X).

Neighbor-joining turns out to work surprisingly well in practice; a theoretical justification for this is given in [351]. Nonetheless, there are no global guarantees about the behavior of the procedure for metric spaces far from trees; for instance, in some cases neighbor-joining can produce negative edge lengths or exhibit other perverse behavior.

Remark 4.7.4. A natural question to ask is how to determine whether a metric space is far from being tree-like. One measure of the divergence from being a tree metric is given by Gromov's *δ-hyperbolicity*, which is a relaxation of the four-point condition.

Persistent homology also gives an interesting approach to detecting whether a metric space is a tree: metric trees are contractible and should have no homology. Therefore computing PH_k for any $k > 0$ yields information about divergence from being tree-like. We discuss applications of this idea to population genetics in Section 5.2.

4.7.2 The Billera-Holmes-Vogtmann Metric Spaces of Phylogenetic Trees

For many applications, it would be very desirable to have a metric on the set of phylogenetic trees. A distance function would permit quantitative comparisons. It would also allow one to apply clustering algorithms to collections of trees (e.g., produced from samples from distinct patients). A metric would also provide some of the foundations for dealing with probability distributions on phylogenetic trees as well as summary statistics.

Billera-Holmes-Vogtmann (BHV) constructed a metric on weighted phylogenetic trees using the tools of metric geometry [55]. They defined a metric space BHV_m of isometry classes of rooted phylogenetic trees with m labeled leaves where the non-zero weights are on the internal branches. The space BHV_m is constructed by gluing together (recall Section 1.5) $(2m-3)!!$ positive orthants $\mathbb{R}^n_{\geq 0}$ of dimension $n = m - 2$, where

$$\mathbb{R}^n_{\geq 0} = \{(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n \mid x_1 \geq 0, x_2 \geq 0, \ldots, x_n \geq 0\};$$

each orthant corresponds to a particular tree shape, with the coordinates specifying the lengths of the internal edges. A point in the interior of an orthant represents a binary tree; if any of the coordinates are 0, the tree is obtained from a binary tree by collapsing the internal edges with length 0. We glue orthants together such that a (non-binary) tree is on the boundary between two orthants when it can be obtained by collapsing edges from either tree geometry. Put another way, orthants corresponding to two tree topologies are adjacent when they are connected by a

rotation, i.e., one topology can be generated from the other by collapsing an edge to length 0 and then expanding out another edge from the incident vertex.

The metric on BHV_m is induced from the standard Euclidean distance on each of the orthants.

1. For two trees t_1 and t_2 which are both in a given orthant, the distance $d_{\mathrm{BHV}_m}(t_1, t_2)$ is defined to be the Euclidean distance between the points specified by the weights on the edges.
2. For two trees which are in different orthants, there exist (many) paths connecting them which consist of a finite number of straight lines in each orthant. The length of such a path is the sum of the lengths of these lines, and the distance $d_{\mathrm{BHV}_m}(t_1, t_2)$ is then the minimum length over all such paths.

For many points, the shortest path goes through the "cone point," the star tree in which all internal edges are zero. See Figure 4.20 for a picture of tree space.

As explained in [55, §4.2], efficiently computing the metric on BHV_m is a non-trivial problem. However, there exists a polynomial-time algorithm based on successive approximation of geodesic paths [394].

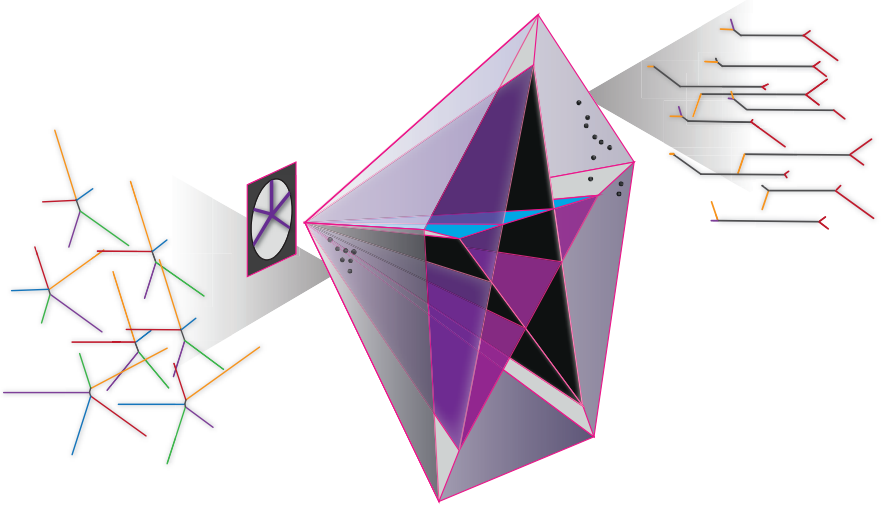

Figure 4.20 Moduli space of phylogenetic trees describing clonal evolution. Collections of trees can be mapped onto a geometric space, forming a point cloud. Trees with the same topology will live in the same orthant, and crossing into an adjacent orthant corresponds to a tree rotation (collapsing an edge to 0 and expanding out a new edge). Points closer to the vertex of the cone have relatively little internal branch length, while points near the base of the cone have little weight in the external branches. Source: [545]. From Zairis et al., Genomic data analysis in tree spaces, arXiv: 1607.07503 [q-bio.GN].

The main result of Billera, Holmes, and Vogtmann is that the length metric on BHV_n endows this space with a (global) CAT(0) structure (see Definition 4.7.9 below). The fact that BHV_n is a CAT(0) space means that points are connected by unique geodesics (which realize the distance between them) and there are unique centroids. As a consequence, it is reasonable to consider geometric inference in this setting. Moreover, BHV_n is clearly a complete metric space and is separable, which means that it contains a countable dense subset; any tree can be approximated by a sequence of trees in the same orthant that have rational edge lengths. That is, BHV_n is a Polish space and so as discussed in Section 3.3 is a reasonable space on which to apply the standard machinery of probability theory (see [247, 248] for work in this direction). In some applications it is also useful to consider a *projectivized* variant of the tree space where the internal edges are constrained to have lengths that sum to 1. We denote this subspace of Σ_n by $\mathbb{P}\Sigma_n$ and refer to it as the projective tree space.

Remark 4.7.5. The space of phylogenetic trees turns out to appear in various other contexts in mathematics; for instance, it is closely related to the moduli space of algebraic curves [141]. Perhaps more relevantly, it appears in the context of Diaconis and Sturmfels' algebraic statistics [145] as a tropical Grassmannian [479] (see also [395]).

4.7.3 Metric Geometry

Although metric spaces often arise in contexts in which there is no evident notion of geometry, it turns out that under very mild hypotheses a metric space (X, ∂_X) can be endowed with structures analogous to those arising on Riemannian manifolds. See [73, 83] for a comprehensive treatment of metric geometry. The basic approach to this involves the notion of *length* of a path in a metric space.

Definition 4.7.6. Let (X, ∂_X) be a metric space. Let $I \subset \mathbb{R}$ be an interval $[a, b]$. The *length* of a path $\gamma\colon I \to X$ is

$$L(\gamma) = \sup \sum_{i=1}^{n} \partial_X(\gamma(x_i), \gamma(x_{i-1})),$$

where the sup is taken over all collections $a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b$.

We can define a potential distance function on (X, ∂_X) by defining the distance between x and y to be

$$\inf_{\gamma} L(\gamma),$$

where the infimum is taken over all $\gamma\colon [0,1] \to X$ such that $\gamma(0) = x$ and $\gamma(1) = y$. A metric space is a *length space* if this distance agrees with the metric. When the infimum can be achieved, we have the notion of a geodesic metric space.

Definition 4.7.7. A metric space (M, ∂_M) is a *geodesic metric space* if any two points x and y can be joined by a path with length precisely $\partial_M(x, y)$.

Any Riemannian manifold is a geodesic metric space. But more generally, a good notion of *curvature* makes sense in any geodesic metric space [11]. The idea is that the curvature of a space can be detected by considering the behavior of the area of triangles, and triangles can be defined in any geodesic metric spaces. Specifically, given points p, q, r, we have the triangle $T = [p, q, r]$ with edges the paths that realize the distances $\partial_M(p, q)$, $\partial_M(p, r)$, and $\partial_M(q, r)$. The connection between curvature and area of triangles is revealed by the observation that given side lengths $(\ell_1, \ell_2, \ell_3) \subset \mathbb{R}^3$, a triangle with these side lengths on the surface of the Earth is "fatter" than the corresponding triangle on a Euclidean plane. To be precise, we consider the distance from a vertex of the triangle to a point p on the opposite side – in a fat triangle, this distance will be larger than in the corresponding Euclidean triangle. (Thin triangles are defined analogously.) See Figure 4.21 for examples of thin and fat triangles.

Given a triangle $T = [p, q, r]$ in (M, ∂_M), we can find a corresponding triangle $\tilde{T}$ in Euclidean space with the same edge lengths. Given a point z on the edge $[p, q]$, a comparison point in $\tilde{T}$ is a point $\tilde{z}$ on the corresponding edge $[\tilde{p}, \tilde{q}]$ such that $\partial_{\mathbb{R}^2}(\tilde{z}, \tilde{p}) = \partial_M(p, z)$.

Definition 4.7.8. Let (M, ∂_M) be a metric space. We say that a triangle T in M satisfies the CAT(0) inequality if for every pair of points x and y in T and comparison points $\tilde{x}$ and $\tilde{y}$ on $\tilde{T}$, we have $\partial_M(x, y) \leq \partial_{\mathbb{R}^2}(\tilde{x}, \tilde{y})$.

Definition 4.7.9. If every triangle in M satisfies the CAT(0) inequality then we say that M is a *CAT*(0) *space*.

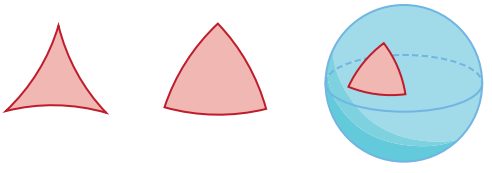

Figure 4.21 Thin triangles have angles that add up to less than 180 degrees; fat triangles have angles that add up to more. We can detect the curvature of the Earth by observing that big triangles are fat.

More generally, let M_κ denote a complete and simply connected two dimensional Riemannian manifold with curvature κ; the classification results discussed above show that there is a unique such manifold up to homeomorphism. The diameter of M_κ will be denoted D_κ. A D_κ-geodesic metric space is one in which all pairs of points p and q such that $\partial_M(p, q) < D_\kappa$ are connected by a geodesic.

Definition 4.7.10. A D_κ-geodesic metric space M is a *CAT*(κ) *space* if every triangle in M with perimeter $\leq 2D_\kappa$ satisfies the inequality above for the corresponding comparison triangle in M_κ.

Clearly, if $\kappa' \leq \kappa$, any CAT(κ') space is also CAT(κ). More importantly, this notion coincides with standard ideas about curvature in geometric examples: An n-dimensional Riemannian manifold M that is sufficiently smooth has sectional curvature $\leq \kappa$ if and only if M is CAT(κ). For instance, Euclidean spaces are CAT(0), unit spheres are CAT(1), and hyperbolic spaces are CAT(-1).

As described, CAT(κ) is a global condition.

Definition 4.7.11. A metric space (X, ∂_X) is *locally CAT*(κ) if for every x there exists a radius r_x such that $B_{r_x}(x) \subseteq X$ is CAT(κ).

Example 4.7.12. For example, the flat torus (formed by taking the box $[0, 1] \times [0, 1]$ and gluing together the edges $\{0\} \times [0, 1]$ to $\{1\} \times [0, 1]$ to make a cylinder and the edges $[0, 1] \times \{0\}$ to $[0, 1] \times \{1\}$ to make a torus) is locally CAT(0) but not globally CAT(0).

Theorem 4.7.13 (Cartan-Hadamard)**.** *A simply connected metric space that is locally CAT*(0) *is also globally CAT*(0)*.*

A remarkably productive observation of Gromov is that many geometric properties of Riemannian manifolds are shared by CAT(κ) spaces. In particular, CAT(κ) spaces with $\kappa \leq 0$ (referred to as *non-positively curved metric spaces*):

1. admit unique shortest paths joining each pair of points x and y,
2. have the property that all balls $B_\epsilon(x)$ are convex and contractible for all x and $\epsilon \geq 0$, and
3. have stable midpoints of shortest paths.

It is in general very difficult to determine for an arbitrary metric space whether it is CAT(κ) for any given κ. Even for finite simplicial complexes where the metric is induced from the Euclidean metric on each face, this problem does not have a general solution.

4.8 Summary

- A standard approach in data analysis is to search for low-dimensional structure in high-dimensional data points using the geometry encoded in the interpoint distances.
- Principal component analysis (PCA) takes a finite set of points in $\mathbb{R}^n$ (with the assumption that these points admit an isometric embedding as a plane) and seeks to find an optimal linear projection of the data into $\mathbb{R}^k$ for $k < n$.
- Metric dimensionality scaling (MDS) is another classical method which determines an optimal embedding of a finite metric space (X, ∂_X) into a Euclidean space. MDS is similar to PCA, but can be applied to arbitrary metric spaces.
- Isomap and local linear embedding (LLE) are two related algorithms that apply MDS to empirical approximations of the intrinsic metric of a manifold. Isomap and LLE differ slightly in their procedures. LLE is slightly more successful in practice on non-convex contractible subsets of Euclidean space.
- Almost all manifold learning algorithms depend on approximating the local tangent structure of the manifold from the data, typically by studying the spectrum of the graph Laplacian. Heat flow provides a conceptual framework for describing these approximations.
- Neighbor embedding algorithms like stochastic neighbor embedding (SNE) make different geometric assumptions and are effective in working with data points of non-uniform density. A descendant of SNE, t-distributed stochastic neighbor embedding (t-SNE), is an extremely popular choice in applications.
- As synthetic examples illustrate, classical dimensionality reduction and manifold learning techniques work best under restrictive hypotheses about the geometry of the data.
- The coordinates provided by manifold learning algorithms can be used as filters for Mapper. This is an interesting avenue for combining TDA and geometric dimensionality to provide a more flexible description of the underlying structure of the data.
- Metric geometry is the study of geometric structures on metric spaces that are similar to those that arise in Riemannian geometry.
- The space of phylogenetic trees may be endowed with geometric structure. This structure provides the foundations for dealing with probability distributions and summary statistics.

4.9 Suggestions for Further Reading

There is of course a tremendous literature on PCA and many different variants of MDS; we particularly recommend Hastie, Tibshirani, and Friedman's classic

text [233] for a wonderful exposition in the context of classification and learning. The area of manifold learning and the problem of working with non-Euclidean manifolds embedded in Euclidean space is substantially more recent. A nice survey of manifold learning techniques (discussed in a broader machine learning context) is available in [50].