IAN: Iterated Adaptive Neighborhoods for Manifold Learning and Dimensionality Estimation

Luciano Dyballa
luciano.dyballa@yale.edu
Department of Computer Science, Yale University, New Haven, CT 06511, U.S.A.

Steven W. Zucker
steven.zucker@yale.edu
Departments of Computer Science and of Biomedical Engineering, Yale University, New Haven, CT 06511, U.S.A.

Invoking the manifold assumption in machine learning requires knowledge of the manifold's geometry and dimension, and theory dictates how many samples are required. However, in most applications, the data are limited, sampling may not be uniform, and the manifold's properties are unknown; this implies that neighborhoods must adapt to the local structure. We introduce an algorithm for inferring adaptive neighborhoods for data given by a similarity kernel. Starting with a locally conservative neighborhood (Gabriel) graph, we sparsify it iteratively according to a weighted counterpart. In each step, a linear program yields minimal neighborhoods globally, and a volumetric statistic reveals neighbor outliers likely to violate manifold geometry. We apply our adaptive neighborhoods to nonlinear dimensionality reduction, geodesic computation, and dimension estimation. A comparison against standard algorithms using, for example, *k*-nearest neighbors, demonstrates the usefulness of our approach.

1 Introduction

A starting point for many algorithms in data science—from clustering to manifold inference—is knowing the neighbor relationships among data points. Clustering, for example, often begins with a "k-nearest neighbor graph," while manifold inference involves a kernel, that is, a measure of similarity between data points. In the first case, the neighborhoods are local and discrete; in the second, they are global and continuous, with concentration of influence controlled by the kernel bandwidth, or scale. Such neighbor relationships are fundamental to defining a topology. Moreover, dimensionality may be estimated based on the rate of change in the density of points within a ball, that is, within a neighborhood, with respect to its radius. It is helpful when the number of data points is large, a requirement that grows with dimensionality; asymptotic analysis is often favored by theoreticians.

In practice, we rarely have enough data points to satisfy asymptotic bounds. Nor are we given the precise number of neighbors, k, that each point should have. We often make the manifold assumption—that the data points are drawn randomly from a (or near a) manifold—but rarely try to assess the basic properties of the manifold assumed by theorists: its dimensionality, sampling density, curvature, medial axis, or reach (defined in the next section). All of these could influence k.

Instead, we rely on different visualization algorithms, such as Isomap, diffusion maps, t-SNE, and many others, to find a pleasing organization of the data. This is dangerous, of course, because these algorithms have free parameters. In particular, and central to this article, most require specifying the number of neighbors, k (or its equivalent): changing k or other parameters changes the result. Unless one knows the answer, one is caught in a conundrum: imposing a prior belief amounts to "fixing" the solution (examples of changing k are shown later in the article).

This gap between theory and practice shows up from the start. If the manifold is not pure (i.e., if it consists of a union of manifolds of possibly different dimensionality), then there may be no global k that suffices; furthermore, the manifold may have a boundary. Even if it is pure and without boundary, the temptation to choose k large is common. But this can incorrectly fill in the open space around curved manifolds ("folding" or "short-circuiting"), linking distant points that should not be neighbors. On the other hand, choosing k small can induce holes and break connectivity. Such phenomena are illustrated in Figure 1. As we shall demonstrate, sampling issues and manifold geometry interact in causing these. Moreover, in real data sets the appropriate number of neighbors may differ from point to point. This final issue is a principal motivation for this article.

We present an algorithm to estimate an effective neighborhood—the immediate neighbors, or scale of a similarity kernel—around each point. We seek to identify those nearest neighbors that are "correct" in the sense that they support dimensionality and volume estimates, and manifold inference in general, without covering holes or filling in concavities. It is inspired by the philosophical position that views discrete and continuous mathematics as "two sides of the same," as argued by Lovász (2010), and iterates between them.

Our algorithm builds from a conservative initial estimate of neighbors (based on a discrete construct, the Gabriel graph) toward a refined one, based on continuous estimates from a multiscale gaussian kernel. The discrete and continuous volume estimates must be consistent, however, and this provides the glue for our iteration. Since not all of the initial putative neighbors may actually be closest neighbors, neighbors that violate the volume relationship are pruned, and the process repeats until the two

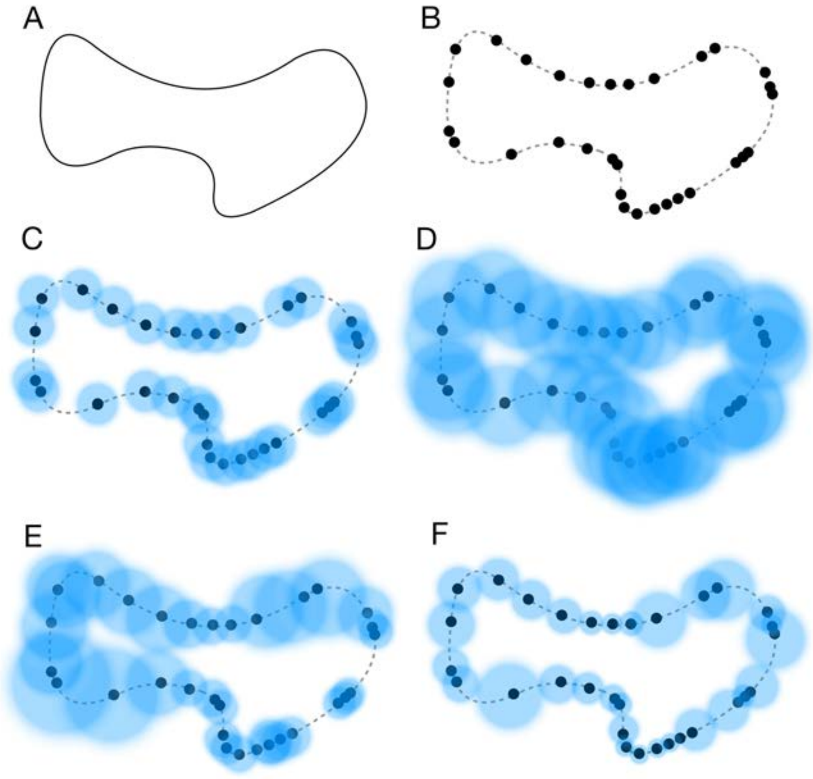


Figure 1: Inferring the geometry of manifolds requires neighborhoods around each given data point. Setting the correct scale for these neighborhoods, shown as balls, is fundamental. (A) Example of a one-dimensional manifold, $\mathcal{M}$. (B) Collection of points sampled from an unknown distribution over $\mathcal{M}$. Their pairwise distances are the only available data; properties of $\mathcal{M}$ are not given a priori. (C) Using a global kernel scale: if it is too small, the manifold will appear disconnected, artificially producing clusters. Notice how some balls do not touch. (D) If it is too big, the manifold may collapse, giving rise to incorrect geometry/topology. Notice how the balls overlap (covering dimension). (E) The use of local scales based on a global number of nearest neighbors (in this example, $k = 2$) is still susceptible to the problems above. (F) Our approach computes locally adaptive neighborhood sizes, resulting in scales that conform to the local geometry and sampling.

perspectives agree. Our algorithm thus can be considered an iterative graph sparsification.

Technically, it involves two different graphs: a discrete one, which links only putative nearest neighbors (pairs of points defining the diameter of an otherwise empty ball), and a weighted one, structured by a multiscale

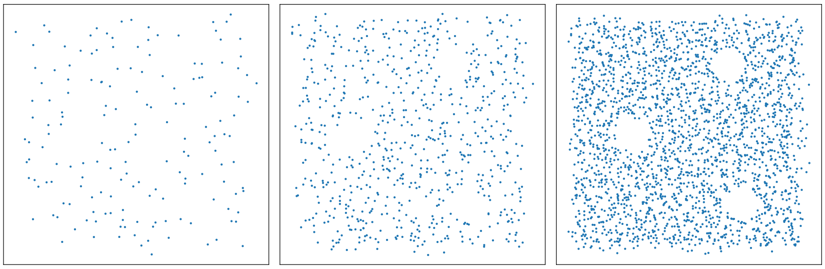


Figure 2: Sampling a Swiss cheese. The available data constrain manifold complexity. Viewed from left to right, as the number of sample points increases, the apparent manifold goes from a plane to a plane with holes. The central panel shows the sampling density for which actual holes in the manifold become roughly distinguishable from holes due to sampling. The results of our algorithm on these examples are shown in Figure 23.

gaussian kernel, whose individual scales must cover the neighborhood given in the discrete graph. Keeping the two graphs consistent is another way to think about our iteration. Each resulting graph can be applied to many different algorithms for data visualization, dimensionality reduction, and manifold inference.

Our approach to the problem is in the spirit of exploratory data analysis; it works with the available data. This provides another view regarding the interaction between sampling and geometry: one can only do as well as the available data allow (see Figure 2). The situation is analogous to that in learning theory, where there is a trade-off between the accuracy of the learner and the coarseness of the hypothesis class over which she is learning (Alon, Ben-David, Cesa-Bianchi, & Haussler, 1997). Here, the space of manifolds over which inferences are made is dictated by the available samples.

An overview of the article is as follows. In the next section, we review the background in some detail, covering both the zoo of similarity kernels that exist, plus several relevant notions, such as the reach of a manifold, that are well studied in the theory literature. The discussion is organized to emphasize the centrality of scale, or neighborhood, in all of the references. In section 3, we provide an overview of our algorithm. It includes a brief sketch of both graphs we work with, plus the connection back to manifolds. Pseudocode for the algorithm is given in algorithm 1, which also includes pointers to where each of its steps is developed.

We then expand on the algorithm. In section 3.3, we study the Gabriel graph and putative neighbors. Two features are emphasized: scale-free neighborhoods and the relationship between node degree and local dimensionality. A structural criterion is revealed, showing how putative edges between neighbors fill "volumes" that block others from being

Algorithm 1: Iterated Adaptive Neighborhoods Kernel.

```
1:  procedure IANKERNEL($D$)                                        ▷ Input: distance matrix, $D$
2:      $G^{(0)} \leftarrow$ GABRIELGRAPH($D$)                      ▷ Compute initial $G$ (sec. 3.3)
3:      repeat Iteration
4:          $\boldsymbol{\sigma}^{(t)}, \leftarrow$ OPTIMIZESCALES($G^{(t)}, D$)        ▷ Update scales $\boldsymbol{\sigma}$ (sec. 3.4)
5:          $\mathcal{G}^{(t)} \leftarrow$ MULTISCALEKERNEL($D, \boldsymbol{\sigma}^{(t)}$)      ▷ Weighted graph (eq. 3.1)
6:          $\delta', C \leftarrow$ COMPUTEVOLUMERATIOS($G^{(t)}, \boldsymbol{\sigma}^{(t)}$)   ▷ Statistic $\delta'$ (sec. 3.5.1)
7:          $G^{(t+1)} \leftarrow$ SPARSIFY($G^{(t)}, \delta'$)          ▷ Update $G$ (sec. 3.5.2)
8:      until no further change in $G$
9:      return $G^\star, \mathcal{G}^\star, \boldsymbol{\sigma}^\star$      ▷ Output: final graphs and optimal scales
10: end procedure
```

neighbors. This graph serves as an initialization. It is then refined iteratively in several steps. First, continuous kernel scales are computed based on the discrete, putative neighbors. A linear program relaxation bridges local scales to a global cover, in which each node's weighted degree is comparable to the number of its neighbors. In other words, each neighborhood radius should not cover too many outside points. If it does, then it indicates that the neighborhood itself should be refined. That is, some putative scales are likely wrong, in the sense that their neighborhood contains an extreme outlier. This leads directly to a volumetric statistic (see section 3.5.1), and to a pruning technique for sparsifying edges from the discrete graph. The process iterates until there are no more outliers.

In section 4, we evaluate the results for estimating manifold low-dimensional embeddings, geodesics, and local intrinsic dimensionality. Comparisons against popular algorithms, such as UMAP and t-SNE, illustrate the power of the approach. In the end, we demonstrate that it is possible to infer data-driven local neighborhoods that remain consistent with geometric and topological properties of manifolds.

Code for our algorithm will be available at github.com/dyballa/IAN.

2 Background

Manifold learning is a vast area of machine learning where high-dimensional data are analyzed based on the assumption that they were sampled from a low-dimensional manifold, $\mathcal{M}$ (Fefferman, Mitter, & Narayanan, 2016), in which case geodesic distances over $\mathcal{M}$ provide a better description of the relationships between data points than Euclidean distances in ambient space (Belkin & Niyogi, 2004). The manifold assumption finds applications in nonlinear dimensionality reduction (van der Maaten, Postma, & van den Herik, 2009), denoising (Hein & Maier, 2006), interpolation (Bregler & Omohundro, 1994), dimensionality estimation (Camastra & Staiano, 2016), computational geometry (Crane, Weischedel, & Wardetzky, 2013), and more.

Since $\mathcal{M}$ locally resembles Euclidean space, it is standard to define a similarity kernel to define (possibly weighted) neighborhoods around each point in terms of other points. This naturally leads to a graph having data points as nodes and similarity values as edge weights. Then, by computing the graph Laplacian, one can apply a variety of methods from spectral graph theory (see, e.g., Spielman, 2012). Formal analysis involves the limit as the number of data points grows large; the practical success of such methods depends on how well graph neighborhoods capture the topology and geometry of $\mathcal{M}$.

We here review the many approaches to specifying a similarity kernel or a local neighborhood. Let $\mathcal{M}$ be a d-dimensional manifold in ambient space $\mathbb{R}^n$. When only pairwise distances are known, an intuitive approach is to define the neighbors of a point, $\boldsymbol{x}_i$, as those within a certain distance threshold, or, equivalently, inside an n-dimensional ball around $\boldsymbol{x}_i$. A kernel function assumes the role of this ball by assigning values to neighboring points as a function (discrete or continuous) of how close they are to $\boldsymbol{x}_i$. The question becomes, What kernel size should be used for each point?

2.1 Similarity Kernels. Consider a set of points $\mathcal{X} \in \mathbb{R}^n$. Typically, a symmetric, positive semidefinite similarity kernel (Schölkopf & Smola, 2002) is chosen to determine weighted connections between data points based on the ambient Euclidean distances between them. For each pair of data points $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^n$, it returns a number between 0 and 1, which determines how close, or strongly connected, they are. This effectively defines a neighborhood around each point.

2.1.1 Discrete Kernels. Possibly the simplest choice for a kernel is the ε-neighborhood (Belkin & Niyogi, 2003):

$$K_{ij}(\varepsilon) = \begin{cases} 1, & \text{if } \|\boldsymbol{x}_i - \boldsymbol{x}_j\| < \varepsilon \\ 0, & \text{otherwise,} \end{cases} \tag{2.1}$$

where $\|\cdot\|$ is typically the Euclidean norm in $\mathbb{R}^n$. This results in discrete-like neighborhoods whose sizes may be quite sensitive to the choice of ε, so implicit is the assumption that sampling is approximately uniform.

Instead of defining a neighborhood radius, a more common approach is to specify the number of neighboring points, k. Letting $\mathcal{N}_k(\boldsymbol{x}_i)$ be the set containing the k points closest to $\boldsymbol{x}_i$ in $\mathbb{R}^n$ (not including $\boldsymbol{x}_i$[1]), a k-nearest neighbors kernel can be defined as

[1]Throughout, when referring to a point's set of k-nearest neighbors, we shall not include the point itself (unless otherwise stated) and further assume that no two points are identical.

$$K_{ij}(k) = \begin{cases} 1, & \text{if } \boldsymbol{x}_j \in \mathcal{N}_k(\boldsymbol{x}_i) \\ 0, & \text{otherwise,} \end{cases} \tag{2.2}$$

which is commonly symmetrized by making $K_{ij}(k) = 1$ if $\boldsymbol{x}_j \in \mathcal{N}_k(\boldsymbol{x}_i) \vee \boldsymbol{x}_i \in \mathcal{N}_k(\boldsymbol{x}_j)$.

2.1.2 Continuous Kernels: Global Scale. In order to have the kernel values decrease with increasing distance between data points, a gaussian kernel is commonly used:

$$K_{ij}(\sigma) = \exp\left(\frac{-\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{\sigma^2}\right). \tag{2.3}$$

This gives a continuous similarity scale from 1 (when $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are identical) down to some predetermined cutoff below which the kernel is considered to be zero (meaning no connection in the data graph). Such a threshold is typically chosen to be a very small value, often at the limit of numerical precision, and is often required to ensure compactness of the kernel.

One would like the parameter σ to be just large enough to be able to capture local manifold patches. There are several heuristics for finding such a scale: the median of all pairwise distances in $\mathcal{X}$ (or another percentile), the mean (or median) of the distances to each point's kth nearest neighbor (Lafon, 2004), or a scalar multiple of the maximal distance from a point to its nearest neighbor in the data (Keller, Coifman, Lafon, & Zucker, 2009). Also common is to choose a scale so that each data point is sufficiently connected to at least one other point (Lafon, Keller, & Coifman, 2006).

A different approach is based on inspection of the curve given by the sum of pairwise kernel values. When the double-sum $\sum_{i,j} K_{ij}(\sigma)$ is plotted against σ using a log-log scale, the slope

$$\frac{\mathrm{d}\log\sum_{i,j} K_{ij}(\sigma)}{\mathrm{d}\log\sigma} \tag{2.4}$$

is proportional to the intrinsic dimensionality of the data (Coifman, Shkolnisky, Sigworth, & Singer, 2008). A global scale is then chosen from within a linear region of such curve. Haghverdi, Buettner, and Theis (2015) proposes a similar procedure that considers instead the curve given by the weighted average of the degrees $Z_i(\sigma) = \sum_j K_{ij}$ of each data point $\boldsymbol{x}_i$, after taking the logarithm:

$$\left\langle \log Z_i(\sigma) \right\rangle = \frac{\sum_i \log Z_i(\sigma) \cdot (1/Z_i(\sigma))}{\sum_i (1/Z_i(\sigma))}. \tag{2.5}$$

The use of the inverse of each point's degree as weights is intended to compensate for density heterogeneities. The choice of σ is then made precise by choosing the argmax of the slope of $\langle \log Z_i(\sigma) \rangle$ plotted against $\log \sigma$, which in many cases should occur near the center of the linear region of equation 2.4. One complication occurring in both approaches, however, is that more than one linear section (and, equivalently, more than one local maximum of the slope) may exist, requiring that additional criteria be defined to make the choice of σ truly automated.

2.1.3 Continuous Kernels: Multiscale. A more localized strategy is to use a multiscale kernel, where each point has an individual scale, or bandwidth. Instead of a single, global scale, there are now N parameters. The advantage is that if the scale selection is adequate, the kernel may capture the characteristics of more complex data sets and manifolds that have nonuniform sampling and geometry.

In the self-tuning method (Zelnik-Manor & Perona, 2004), local scales are used in a gaussian kernel by replacing the global scale σ, from equation 2.3, by $\sqrt{\sigma_i \sigma_j}$, where σ_i and σ_j are the scales assigned to $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, respectively. This results in the symmetric kernel:

$$K_{ij}(\sigma_i, \sigma_j) = \exp\left(\frac{-\left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|^2}{\sigma_i \sigma_j} \right). \tag{2.6}$$

Each σ_i is set as the distance to the kth nearest neighbor of $\boldsymbol{x}_i$; authors recommend $k = 7$ (Zelnik-Manor & Perona, 2004; Mishne & Cohen, 2012).

In Berry, Giannakis, and Harlim (2015) and Berry and Harlim (2016), a variable bandwidth kernel is proposed that combines the use of local bandwidths with a global scale parameter, ϵ. The kernel then takes the form

$$K_\varepsilon\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \exp\left(\frac{-\left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|^2}{4\epsilon (q_\epsilon(\boldsymbol{x}_i) q_\epsilon(\boldsymbol{x}_j))^\beta} \right), \tag{2.7}$$

where q_ϵ is a local density function and β an additional (nonpositive) parameter. An initial estimate for the local bandwidth around each point $\boldsymbol{x}_i$ is set as the square root of the mean squared distance to the k-nearest neighbors of $\boldsymbol{x}_i$, with $k = 8$. Finally, ϵ is automatically tuned as the argmax of equation 2.4; however, the authors do not consider cases in which more than one local maximum may exist.

Other methods also adopt individual bandwidth parameters but use asymmetric kernels that are symmetrized a posteriori. In the t-SNE algorithm (van der Maaten & Hinton, 2008), the single-scale gaussian kernel

$$K_{ij}(\sigma_i) = \exp\left(\frac{-\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\sigma_i^2}\right) \tag{2.8}$$

gives a measure of affinity, or similarity, between pairs of points. It is then normalized as

$$p_{j|i}(\sigma_i) = \frac{K_{ij}(\sigma_i)}{\sum_{k\neq i} K_{ik}(\sigma_i)} \tag{2.9}$$

to yield transition probabilities, and finally symmetrized as

$$p_{ij}(\sigma_i, \sigma_j) = \frac{1}{2N}\left(p_{j|i}(\sigma_i) + p_{i|j}(\sigma_j)\right). \tag{2.10}$$

Each σ_i is fit to $\boldsymbol{x}_i$ so that the distribution of $p_{j|i}, \forall j$ attains entropy H_i such that its perplexity, 2^{H_i} (a real-valued number representing the "effective number of neighbors"), approximates some prespecified value, k. The authors recommend a value for k between 5 and 50.

In the UMAP algorithm (McInnes, Healy, Saul, & Grossberger, 2018), an exponential kernel is used instead of the typical gaussian. Using a prespecified neighborhood size, k, let $\mathcal{N}_k(i)$ be the set of k-nearest neighbors of $\boldsymbol{x}_i$. With ρ_i as the distance to the nearest neighbor of $\boldsymbol{x}_i$, the kernel has the form

$$K_{ij}(\sigma_i) = \exp\left(\frac{-\max\{0, \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\| - \rho_i\}}{\sigma_i}\right), \; j \in \mathcal{N}_k(i) \tag{2.11}$$

and is symmetrized as

$$U_{ij}(\sigma_i, \sigma_j) = K_{ij}(\sigma_i) + K_{ji}(\sigma_j) - K_{ij}(\sigma_i)K_{ji}(\sigma_j). \tag{2.12}$$

It can be seen as a hybrid between continuous and discrete, since U_{il} is set to zero for any point $\boldsymbol{x}_l$ not in $\mathcal{N}_k(i)$. Each σ_i is fit to $\boldsymbol{x}_i$ so that $\sum_j K_{ij}(\sigma_i)$ approximates $\log_2 k$ (loosely analogous to the perplexity approach from t-SNE).

2.1.4 Adaptive Neighborhood Size Methods. Other methods attempt to automatically determine optimal neighborhoods. Most of these are based on determining an optimal k for a k-nearest neighbors (k-NN) graph; this can be done either globally or by selecting a local neighborhood size k_i around each point $\boldsymbol{x}_i$, known as adaptive neighborhood selection (van der Maaten et al., 2009).

Some approaches optimize a global k based on its performance in a specific embedding algorithm. For instance, the method from Samko, Marshall, and Rosin (2006) is tailored to Isomap (Tenenbaum, de Silva, & Langford, 2000), while others (Kouropteva, Okun, & Pietikäinen, 2002; Álvarez-Meza, Valencia-Aguirre, Daza-Santacoloma, & Castellanos-Domínguez, 2011) apply to LLE (Roweis & Saul, 2000). In Álvarez-Meza et al. (2011), a local method is additionally proposed that produces a nearest-neighbor graph with variable k_i, under the assumption that the manifold is connected.

Others are based on first estimating the local tangent space around each point, then setting k_i to include as neighbors those points that are close to it. Such methods (Wang, Zhang, & Zha, 2004; Mekuz & Tsotsos, 2006) typically work with positional information for the tangent space computation (usually via SVD).

Also available are methods that are not based on the nearest-neighbors concept. In computational geometry, the idea of refining an initial estimate of connectivity from a simplicial mesh has been used before, usually specific to the case when $d = 2$ and $n = 3$, surfaces in 3-D space (Amenta, Bern, & Kamvysselis, 1998; Amenta & Bern, 1999; Bernardini, Mittleman, Rushmeier, Silva, & Taubin, 1999; Belkin, Sun, & Wang, 2008). Other approaches extend this idea to arbitrary dimension (Belkin, Sun, & Wang, 2009; Boissonnat, Guibas, & Oudot, 2009) but still require knowledge of d. Most of the algorithms in this class use point clouds as input, so they can exploit positional information to decide on the appropriate neighborhood/connectivity.

Among the myriad ways of estimating neighborhoods, there is little agreement on which is most successful (see Lindenbaum, Salhov, Yeredor, & Averbuch, 2020, for a review). Before proceeding to our algorithm, then, it is helpful to first understand what makes this such a hard problem. How can it fail, and what requirements must it fulfill in order to properly capture the topology and geometry of $\mathcal{M}$? This brings us to the geometry of manifolds.

2.2 Reach and the Geometry of Manifolds. The neighborhoods implied by a kernel should agree with $\mathcal{M}$, or at least approximate a tubular neighborhood of it. As exemplified in Figure 1, if neighborhoods are too small, the implied manifold may become disconnected (i.e., falsely divided into disjoint submanifolds or clusters; Samko et al., 2006); if too large, they may cause $\mathcal{M}$ to self-intersect, collapsing bottlenecks or curved regions, or cause "smoothing" or "folding." Such shortcomings are well known in the manifold inference literature; while the former case typically occurs due to nonuniform sampling, the latter is mainly caused by an incompatibility between the sampling rate and the reach of $\mathcal{M}$ (Federer, 1959; Thäle, 2008). We now expand on these points.

Letting the medial axis of $\mathcal{M}$ be the set of points in $\mathbb{R}^n$ with at least two closest points in $\mathcal{M}$, the reach, τ, can be defined as the minimum distance from $\mathcal{M}$ to its medial axis. Locally, it is constrained by the minimal radius of curvature (i.e., maximal curvature of a geodesic through $\mathcal{M}$); globally, it is

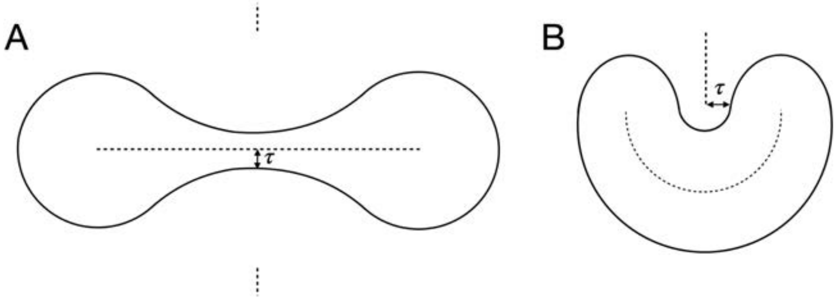


Figure 3: The reach, τ, is a measure of the shape of a manifold. (A) A 1-dimensional manifold $\mathcal{M}$ with a bottleneck; the reach (double arrow) is the smallest distance between $\mathcal{M}$ and its medial axis (dashed curves). (B) A highly curved manifold; now the reach indicates the high curvature region.

constrained by the presence of bottlenecks (see Figure 3). The reach encodes essential geometric properties of $\mathcal{M}$, and has been widely used in the manifold learning community (Amenta & Bern, 1999; Belkin et al., 2008; Niyogi, Smale, & Weinberger, 2008, 2011; Boissonnat et al., 2009; Genovese, Perone-Pacifico, Verdinelli, & Wasserman, 2012; Little, Maggioni, & Rosasco, 2017; Fefferman, Ivanov, Kurylev, Lassas, & Narayanan, 2018; Aamari et al., 2019; Boissonnat, Lieutier, & Wintraecken, 2019). It approximates the size of the largest ball in ambient $\mathbb{R}^n$ such that points in $\mathcal{M}$ can be seen as lying in Euclidean space $\mathbb{R}^d$ (Block, Jia, Polyanskiy, & Rakhlin, 2021). A related concept, the local feature size of a point $x_i \in \mathcal{M}$, is the smallest distance between x_i and the medial axis of $\mathcal{M}$, so τ can be seen as the infimum of the local feature size anywhere on $\mathcal{M}$ (Belkin et al., 2009).

When τ is positive, it provides a measure of the "local distortion" (Block et al., 2021); the larger it is, the easier inference becomes. Some authors (e.g., Narayanan & Mitter, 2010; Fefferman et al., 2016) assume large reach in order to test the manifold hypothesis and find bounds on the required sample size. In Block et al. (2021), the reach is used when establishing bounds on the quality of an intrinsic dimensionality estimation based on k-nearest neighbors.

Obtaining a good representation of $\mathcal{M}$ therefore requires consideration of its reach. In terms of our problem of finding an appropriate kernel, this effectively means that no neighborhood radius should cross the medial axis of $\mathcal{M}$.

Sampling is a further complication and essentially what makes this a hard problem: when it is nonuniform and sparse (common in real-life data sets), it is not always clear whether the space between points constitutes an undersampled piece of $\mathcal{M}$, a hole, or a gap between disjoint submanifolds (cf. Figure 2). The latter two conditions, of course, relate to reach. Narayanan and Mitter (2010) prove that the number of required samples depends polynomially on curvature, exponentially on intrinsic dimension, and linearly on intrinsic volume. Aspects of our algorithm address each of these during the iteration process.

In all such cases, choosing a globally fixed radius is likely to be problematic. While defining neighborhood size based on a fixed number k of neighbors can be helpful to deal with nonuniform density (since the neighborhood radius adapts to the local pairwise distances), it is bound to violate the reach if k is too large. It will also be a problem when the intrinsic dimensionality is not constant throughout $\mathcal{M}$, as higher dimensions require exponentially more neighbors.

Mekuz and Tsotsos (2006) point out the lack of a principled way for setting this parameter, which in practice is often tuned empirically based on prior knowledge of the desired output. As put by Wang et al. (2004), the effectiveness of manifold learning algorithms depends on how nearby neighborhoods overlap and on the interplay between the curvature of the manifold and sampling density.

In terms relevant to this article, the neighborhood radius should be smaller than the local feature size but large enough to account for sampling variability and local dimensionality. We propose an iterative approach to developing a kernel, so that it can adapt appropriately to the neighborhood characteristics around each point.

3 The Algorithm

We here overview our algorithm for finding the neighborhood scale around each point in a manner that makes it globally consistent as a covering of the data points. As is common in manifold learning, we start with a pairwise distance matrix, not the points themselves. The first step is to build a graph in which each datum is connected to an appropriate neighborhood containing other data points. This data graph defines a topology; we refer to it as the *neighborhood graph*.

As we reviewed above, in the discrete case, one might choose k-nearest neighbors, while in the continuous kernel case, there is a bandwidth parameter that effectively defines a "ball of influence" around each point. *Scale* is the radius of such a ball—a level set of the kernel function that essentially contains those neighbors whose weights are nontrivial. Our goal, then, is to find those scales—or neighborhoods—that support nonlinear dimensionality reduction, geodesic estimation, and, in general, manifold inference from the given pairwise distances. We do not have sampling guarantees so will develop a statistic to check whether reach and curvature constraints might be violated.

3.1 Subtleties of Scale.

Since scale may not be constant across the data set, we argue that it should be the first property to be inferred from the data. We start by imposing the manifold assumption, but from an empirical

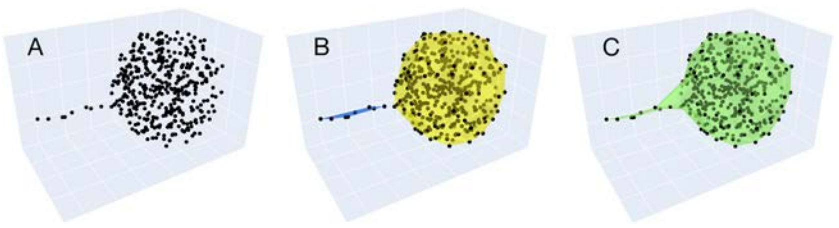


Figure 4: The manifold subtleties of complex data sets. (A) Sampled data from a nonlinear system that includes two regimes. (B) It may be the case that the data in each regime define separate manifolds, shown by color. After sampling their union, however, the evidence for the separation is absent. (C) Or the data may be drawn from a single, connected manifold whose geometric properties change rapidly. In both cases, the intrinsic dimensionality appears different in the spike versus the ball. Colored meshes indicate underlying manifolds.

perspective. Unlike most theoretical studies, we do not assume the manifold is pure (i.e., that it has constant dimension). In a simple case, the data may be drawn from a union of different manifolds whose dimensions are not known a priori; such data sets have been considered infrequently, although exceptions exist (e.g., Haro, Randall, & Sapiro, 2008; Little et al., 2017).

Second, we do not know the sampling rate, or density. Rather, we build it up, conservatively, with putative nearest neighbors to each data point, by imposing a necessary (but not sufficient) condition. These putative neighbors will be refined, as the algorithm iterates, to achieve sufficiency. While the manifold assumption does imply the existence of local neighborhoods, their size may vary over the data set; we require that the sampling be nearly constant over each of them. In effect, the density of points must be determined locally while respecting the global manifold geometry.

We illustrate the complexity of this situation in Figure 4. Shown is a data sphere with an apparent spike emerging from it. On one hand, such complex data sets could derive from two unrelated systems, which only appear to connect through their embeddings. On the other hand, the data could derive from a nonlinear system that includes two regimes, one responsible for the spherical data and the other for the spike. To handle the first situation, we must allow data sets to consist of unions of manifolds. This suggests the interpretation in Figure 4B, where the separation is obscured by sampling. Since manifolds with boundary and high curvature are also possible, the situation in Figure 4C arises. There is an apparent change in intrinsic dimension due to the small reach in the spike and the large boundary curvature. Because the (3D) spike is so narrow, sampling suggests it is 1-dimensional, while the bulk of the points derive from a 3D manifold.

We submit that such situations occur in real data sets and, since the data are fixed, we cannot appeal to knowing the sampling density or the manifold dimensions and reach. Instead, we address the interplay between manifold reach and sampling density pragmatically. Along the spike, the data appear to be 1D; in the ball, 3D. We seek a neighborhood graph that supports these inferences, so "most" points enjoy a neighborhood that agrees with their apparent dimension. At the join (or high-curvature neck), it is unclear. Moving from the spike to the ball suggests that dimension should be increasing; from the ball to the spike, it should be decreasing. For the neighborhood graph, most points along the spike should see about 2 neighbors, and most points in the ball should see about 2^3 neighbors; the problematic points should see something intermediate. Such results will be shown to follow from our algorithm.

We claim that either of the alternatives is worse; one should not impose an apparent dimensionality (or connectivity in the neighborhood graph) globally. If small numbers of neighbors (appropriate for the spike) are enforced on the ball, then holes are likely to be introduced. Or if too many neighbors are enforced on the spike, it will collapse on itself. Both change the topology drastically (these situations are illustrated later, in Figures 24 and 25).

3.2 Overview of the Algorithm.

Let the data set, $\mathcal{X}$, be a sampling of a (possibly nonpure) manifold $\mathcal{M} = \cup_\alpha \mathcal{M}_\alpha$, with the dimension of each component $\mathcal{M}_\alpha$ denoted by d_α. It consists of N points in ambient space $\mathbb{R}^n$, where $n \geq d_\alpha, \forall \alpha$. The manifold may have a boundary, and the number of components is not known a priori.

We work with two graphs: the first unweighted, and the second with edge weights given by a kernel. Our strategy is to begin with a conservative estimate of the unweighted graph and extend it to a global weighted graph that suggests an estimated manifold covering. The validity of this extension is evaluated by a measure of volume in both graphs; an iterative algorithm is used to infer individual local scales for each point $\boldsymbol{x}_i$. Before presenting the algorithm, we introduce the two graphs.

Let the unweighted graph be $G = (V, E)$, with $|V| = N$ and adjacency matrix A with entries a_{ij}, where to each point $\boldsymbol{x}_i \in \mathcal{X}$ is associated a node $i \in V$. We denote its initial estimate by $G^{(0)}$; successive refinements are indicated as $G^{(t)}$ until convergence ($G^\star$).

Since we seek a scale for each data point, we work with a multiscale gaussian similarity kernel, defined as in section 2.1:

$$K_{ij} = \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma_i \sigma_j}\right). \tag{3.1}$$

The kernel value K_{ij} is therefore symmetric and equivalent to that of a traditional gaussian kernel (see equation 2.3), except using the geometric mean of σ_i and σ_j as its scale. Notice, in particular, how the scales and the kernel value are coupled: setting the scale incorrectly could make distant points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ appear close in similarity.

Given a set of individual point scales σ_i (sometimes collected into the vector $\boldsymbol{\sigma} \in \mathbb{R}^N$), we define a second, weighted graph $\mathcal{G} = (V, \mathcal{E}, W)$ as the complete graph on all pairs of data points in $\mathcal{X}$. Its weighted adjacency matrix, W, has entries $w_{ij} = K_{ij}$.

While the unweighted graph will be related to nearest neighbors and computational geometry, the weighted graph will be related to spectral methods on manifold inference. In particular, we expect the Laplacian of $\mathcal{G}$ to approximate the Laplace-Beltrami operator on $\mathcal{M}$, subject to the number of data points and their sampling.

The algorithm is initialized by computing a coarse estimate of G. As described in section 3.3, this is achieved by exploiting the geometry of medial balls between pairs of points to produce a Gabriel graph (Gabriel & Sokal, 1969; Matula & Sokal, 1980). A Gabriel graph is that in which there is an edge between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ if and only if they are the only two closest points to the midpoint of the line segment joining them. The main advantages of using a Gabriel graph as a starting point are that (1) it is scale invariant, so a prespecified ε-neighborhood (equation 2.1) is not required; (2) there is no global constant k (it can vary); and (3) neighbors are not limited to the closest neighbors in ambient space. Thus, it allows for connections to "jump across" sampling gaps while keeping the data graph sparse.

However, as described in section 2.2, obtaining a good inference of $\mathcal{M}$ amounts to finding reasonable estimates of its reach and local feature size. For that to occur, no edge segment ℓ_{ij} between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ should cross a medial axis of $\mathcal{M}$. As the examples that follow will show, there are several cases in which the Gabriel graph will violate this. Therefore, additional steps are necessary to refine it. The Gabriel graph provides a necessary condition (all the correct connections are present, but possibly others as well); our refinement moves toward sufficiency.

In order to estimate $\mathcal{G}$—the weighted counterpart of G—we will use the weights that are obtained by applying a continuous kernel (see equation 3.1) over the points in $\mathcal{X}$. Such a kernel requires scales, or bandwidths, $\boldsymbol{\sigma}$ that must be estimated from G. These will be obtained from an optimization procedure that finds the smallest such scales ensuring that all discrete edges have a minimum kernel value as weight. At this point, a weighted graph $\mathcal{G}$ can be obtained from $\boldsymbol{\sigma}$.

It is now helpful to articulate the geometry more carefully. Figure 5 depicts how the discrete connectivity relates to the manifold geometry. In particular, for a real data set, the few closest points surrounding $\boldsymbol{x}_i$ are the best candidates for "nearest" neighbors—this is all that can be asserted locally. Let $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ be the projections of two neighbors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ onto $\mathcal{M}$, respectively. Then any point along the geodesic between $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ should be closer to no sampled point other than $\boldsymbol{x}_i$ or $\boldsymbol{x}_j$. By further assuming $\boldsymbol{x}_i \in \mathcal{M}$, $\forall i$ or at

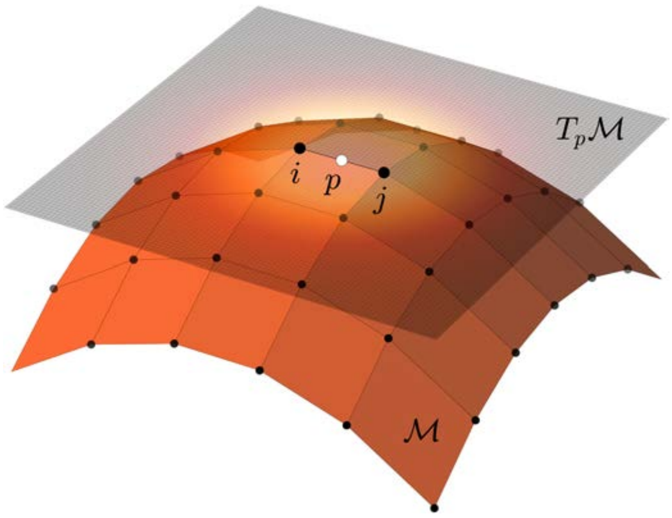


Figure 5: Relating the discrete neighborhood graph to manifold geometry. Nearby sampled points (i and j) on a patch of manifold $\mathcal{M}$ lie in (or near) the tangent plane $T_p\mathcal{M}$ to the midpoint (p). Line segments (edges) between neighboring points lift, via the exponential map, to geodesics in $\mathcal{M}$. The continuous kernel extends this discrete relationship to the full tangent plane. The values of the kernel centered at p are shown as shading, extending in every direction in $T_p\mathcal{M}$. Our algorithm shall enforce this relationship (i.e., the consistency between discrete edges and large kernel values).

least that $\|\boldsymbol{x}_i - \mathcal{M}\|_{\mathbb{R}^n} < \varepsilon, \forall i$ and small ε, then $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\mathbb{R}^n}$ approximates the geodesic when the curvature between $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ is small. Equivalently, the line segment ℓ_{ij} between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ lies on the tangent space $T_p\mathcal{M}$, where p is the midpoint between $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ (see Figure 5). The existence of a geodesic follows from identifying the tangent plane that includes the points with the exponential map of the manifold around them.

Such an "edge-centric" approach connects differential geometry to the underlying graph. This is illustrated in Figure 5, where the kernel values are shown as shading in the tangent plane. Notice how $\boldsymbol{x}_i$ and its neighbor $\boldsymbol{x}_j$ both fall under the bright kernel values; they are very similar (in this measure) to each other. Stated in geometric terms, we assume that the neighbors lie within the injectivity radius around p. In fact, we will show (in Figure 14) that the value of a multiscale kernel between two data points is equivalent to that of a rescaled, single-scale kernel centered at the midpoint between those two points.

The optimized scales can be used to evaluate the current approximation and identify the edges in G that are "too expensive," that is, are likely to violate the local feature size. We proceed by computing successive refinements of both G and $\boldsymbol{\sigma}$ in an iterative manner until no further change is observed. We then return the final version of the discrete and weighted graphs (denoted by $G^\star$ and $\mathcal{G}^\star$, respectively).

One can view the computation of $\mathcal{G}$ as a relaxation of the discrete connectivity in G. In fact, as we shall see in section 3.5, a relaxation statistic, δ_i', will be used to prune discrete edges that produce a poor approximation. More specifically, when a node i with degree, deg(i), in G has δ_i' close to 1, it means i has retained approximately the same degree in $\mathcal{G}$, only continuously spread as a gaussian around it.

Each of these steps is listed in algorithm 1 and will be described in detail. We begin with the discrete connectivity rule (Gabriel graph); then the scale optimization is developed, followed by the edge-pruning step. Figure 6 illustrates the results of our algorithm on data sets for which the Gabriel graph alone cannot infer a good approximation of the manifold connectivity.

3.3 Neighbors in a Gabriel Graph. We begin by defining a set of putative neighboring points of $\boldsymbol{x}_i$ (denoted by $\mathcal{N}(i)$) that uses the connectivity rule found in a Gabriel graph (Gabriel & Sokal, 1969; Matula & Sokal, 1980). It directly incorporates the observation that closest neighbors should have no points "between" them.

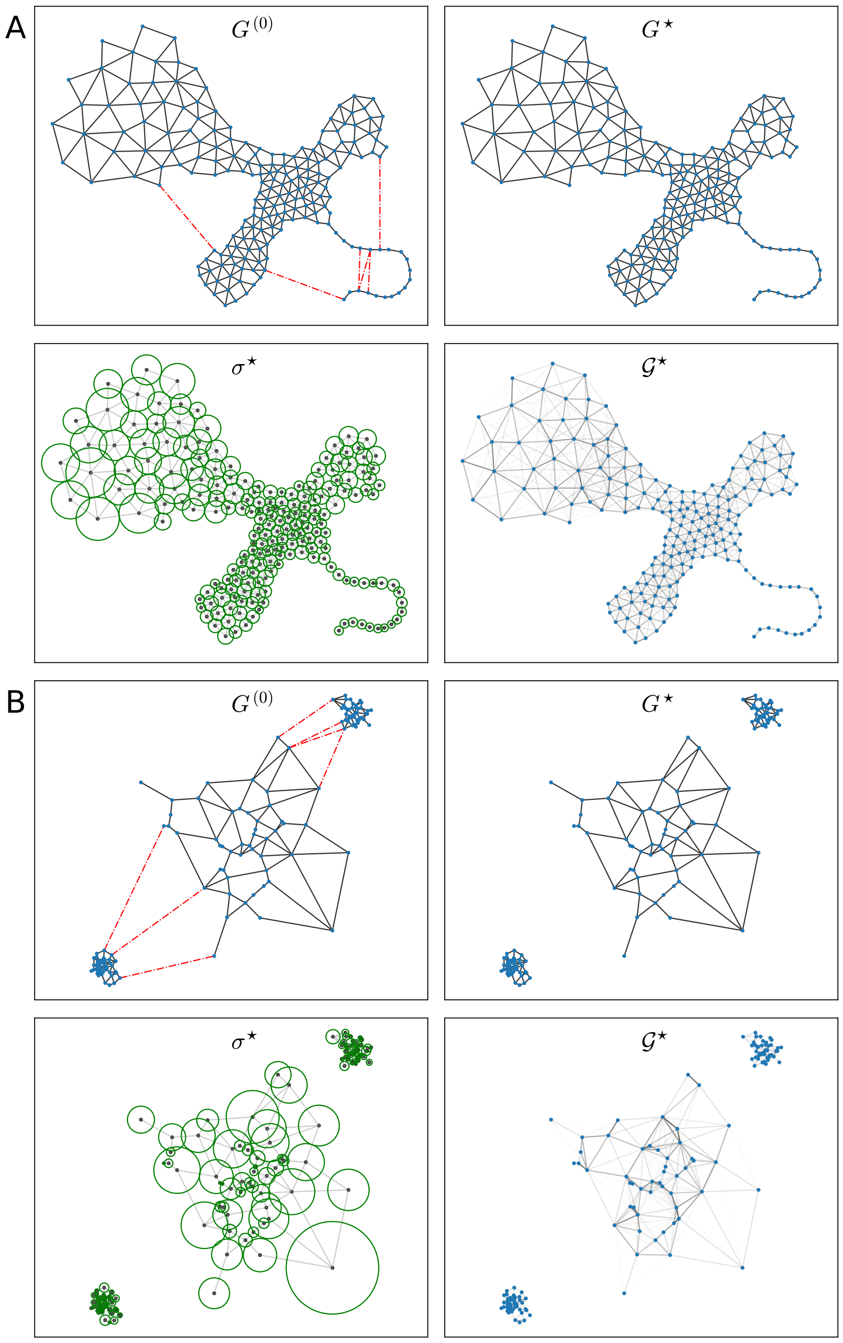
Remark 1. Two points, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, are Gabriel-nearest neighbors to each other if and only if they both touch the same closed ball, $\mathcal{B}_{ij}$, that is empty except for $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Note that $\mathcal{B}_{ij}$ is therefore a medial ball—a ball whose center point is a medial axis (with respect to the set of sampled points). Thus, this connectivity criterion can be restated as creating an edge for all those medial balls, and only those, touching exclusively two points (to be clear, if a third point touches $\mathcal{B}_{ij}$, no edge shall be formed between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$). Hence, to each edge e_{ij} is associated a medial ball $\mathcal{B}_{ij}$ centered at the midpoint between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with radius $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|/2$ (see Figure 7). This is furthermore equivalent to the following alternative definitions:

Remark 2. Points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are Gabriel-nearest neighbors if and only if any point along the line segment $\ell_{ij} = \overline{\boldsymbol{x}_i\boldsymbol{x}_j}$ in $\mathbb{R}^n$ has either $\boldsymbol{x}_i$ or $\boldsymbol{x}_j$ (or both) as its only closest point(s).

Remark 3. In terms of the Voronoi diagram (Fortune, 1995) of $\mathcal{X}$ (with the cell around $\boldsymbol{x}_i$ denoted by V_i), $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are neighbors when ℓ_{ij} crosses a single Voronoi hyperplane H_{ij} (namely, that between the cells V_i and V_j), and the midpoint between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is in H_{ij}.

As a concrete example (refer to Figure 7), consider two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ at a distance r_{ij} from each other, with midpoint p. Assume the region in the manifold between them is uniformly sampled. Now consider the ball


A
$G^{(0)}$
$G^\star$
$\sigma^\star$
$\mathcal{G}^\star$
B
$G^{(0)}$
$G^\star$
$\sigma^\star$
$\mathcal{G}^\star$

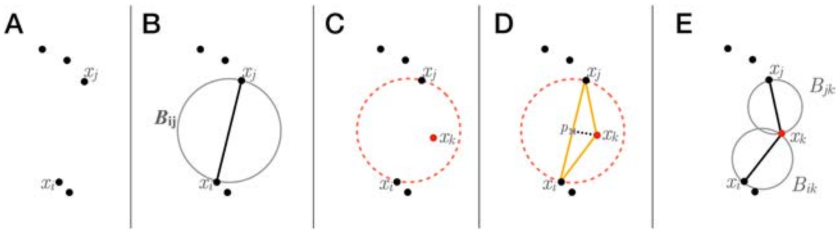


Figure 7: Connecting "nearest neighbors." (A) A set of data points in space. (B) An edge can be formed between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ because there is no other point in the interior of the ball $\mathcal{B}_{ij}$ centered halfway between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. (C) Here, because of the presence of a third point $\boldsymbol{x}_j$ inside $\mathcal{B}_{ij}$, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ cannot be neighbors. (D) Even in the absence of the original data point coordinates (i.e., given only the distances between all pairs of points), Apollonius's formula can be used to determine the length of the segment p–$\boldsymbol{x}_k$, where p is the center of $\mathcal{B}_{ij}$. Namely, p–$\boldsymbol{x}_k$ is a median of the depicted triangle. Here, because the length of the median is less than the radius of $\mathcal{B}_{ij}$, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ cannot be neighbors. (E) Edges are drawn connecting points $\boldsymbol{x}_i$ to $\boldsymbol{x}_k$ and $\boldsymbol{x}_k$ to $\boldsymbol{x}_j$ because both $\mathcal{B}_{ik}$ and $\mathcal{B}_{jk}$ are empty except for those pairs of points, respectively.

centered at p with radius $r_{ij}/2$, therefore touching $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. If there are no points in its interior, we say $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are nearest neighbors. Conversely, if it contains other points in its interior, under our assumption of uniform density, this means that there is at least one other point $\boldsymbol{x}_k$ "between" $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. So we say that $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are *not* nearest neighbors, in the sense that connecting $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ directly would be "crossing over" $\boldsymbol{x}_k$; this implies that an edge e_{ij} in the resulting graph would be a poor approximation to a geodesic in $\mathcal{M}$ (i.e., if $\mathcal{M}$ is "locally uniformly sampled," the segment ℓ_{ij} would be passing outside $\mathcal{M}$). Note that even when the input to the algorithm is solely a distance matrix (i.e., with no position information), this connectivity criterion may still be evaluated by considering the triangle $\boldsymbol{x}_i$–$\boldsymbol{x}_j$–$\boldsymbol{x}_k$ and using Apollonius's theorem to compute the length of the median from $\boldsymbol{x}_k$ to $\boldsymbol{p}$ (see Figure 7D).

Figure 6: Steps of algorithm 1 on toy data sets. (A) Data set with several challenges: nonuniform density, nonuniform dimension, and high curvature. After pruning six edges (dashed red lines) from the original Gabriel graph, $G^{(0)}$, the algorithm converges, inferring reasonable discrete neighborhoods ($G^{\star}$); the optimal scales $\boldsymbol{\sigma}^{\star}$ produce a weighted graph $\mathcal{G}^{\star}$ whose connectivity closely approximates that of $G^{\star}$. (B) Data set with three gaussian clusters of nonuniform density. The Gabriel graph approximation, $G^{(0)}$, naively connects all clusters using multiple edges. After convergence, the clusters become disconnected in $G^{\star}$, and its weighted version follows this by assigning negligible weights (due to $\boldsymbol{\sigma}^{\star}$) between points in different clusters.

The Gabriel graph is a subgraph of the Delaunay graph (Dyer, Zhang, & Möller, 2009) and enjoys a number of key properties (Matula & Sokal, 1980). We emphasize that (1) they are scale invariant, that is, there is no prespecified threshold on the diameter of medial balls that can form connections; (2) the guarantee that Gabriel graphs connect points to their true nearest neighbors when $\mathcal{M}$ is uniformly sampled as a grid (shown in Figure 9); and (3), Gabriel graphs provide a locally adapted neighborhood size k_i, for each point $\boldsymbol{x}_i$, based on the local geometry. Crucially, they do not require an initial guess of the number of neighbors, of the intrinsic dimensionality, or of a maximum neighborhood radius.

Nevertheless, the neighborhoods given by the Gabriel graph are not sufficent. We now expand on a few of their properties—that will be useful in motivating the rest of the algorithm.

3.3.1 Closing Triangles. Here we show that the edges created using the above connectivity rule can only form acute triangles in $\mathbb{R}^n$. Let three points $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, $\boldsymbol{x}_k$ be such that $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ are connected, as well as $\boldsymbol{x}_j$ and $\boldsymbol{x}_k$. The rule says that $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ shall be connected only if $\boldsymbol{x}_k$ is outside the closed ball $\mathcal{B}_{ij}$ of radius $R = r_{ij}/2$ centered halfway between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ (where r_{ij} stands for the Euclidean distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$). Using Apollonius's formula for the squared distance m^2 between $\boldsymbol{x}_k$ and the midpoint between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, we obtain

$$m^2 = \frac{1}{4}(2r_{ik}^2 + 2r_{jk}^2 - r_{ij}^2). \tag{3.2}$$

Then, $\boldsymbol{x}_k$ is in $\mathcal{B}_{ij}$ if and only if $m^2 \leq R^2$, so

$$\begin{aligned} \frac{1}{4}(2r_{ik}^2 + 2r_{jk}^2 - r_{ij}^2) \leq R^2 = \left(\frac{r_{ij}}{2}\right)^2 \\ r_{ik}^2 + r_{jk}^2 \leq r_{ij}^2. \end{aligned} \tag{3.3}$$

Notice that equality will hold when $\boldsymbol{x}_i$–$\boldsymbol{x}_j$–$\boldsymbol{x}_k$ is a right triangle. Therefore:

Remark 4. A triangle will be formed by edges in a Gabriel graph only when it is acute (see Figure 8A).

3.3.2 Maximum Curvature. The above result leads to a bound on the maximum principal curvature that is allowed locally on $\mathcal{M}$ such that the Gabriel graph correctly approximates it (i.e., without closing a triangle). Assume $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, and $\boldsymbol{x}_k$ are points in a smooth manifold $\mathcal{M}$ as in Figure 8B, up to the level that the sampling defines. If we assume that the curvature, κ, is locally constant, then the geodesic from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$ passing through $\boldsymbol{x}_k$ is an arc of a circle $\mathcal{C}$. Therefore, the segments ℓ_{ik} and ℓ_{kj} approximate geodesics on

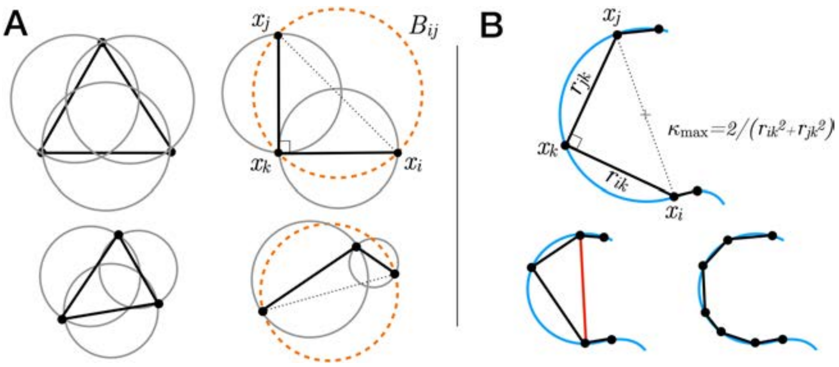


Figure 8: Implications of the connectivity rule in a Gabriel graph. (A) Closing triangles from edges: three points will be mutual neighbors if and only if they form an acute triangle (left). If the angle between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ at $\boldsymbol{x}_k$ is at least $\pi/2$, all three points will lie in $\mathcal{B}_{ij}$, so no edge is created (right). (B) The maximum principal curvature in $\mathcal{M}$ (shown in blue) that can be reasonably approximated by the resulting graph geodesic (path) is constrained by the sampling interval. The limiting case occurs when three points form a right triangle (top, see equation 3.4). When sampling is too sparse (bottom left), a triangle may be formed, in this case preventing the graph from adequately capturing the manifold's geometry. As sampling frequency increases (bottom right), higher curvatures can be better approximated.

$\mathcal{M}$ but not ℓ_{ij} (which would cause "folding"). Hence, values of curvature that can be correctly inferred are those that do not create an edge between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ (i.e., those for which the ball $\mathcal{B}_{ij}$ is nonempty). In this case, from equation 3.3, the maximum such curvature, $\kappa^{\max}$, occurs when $\boldsymbol{x}_i$, $\boldsymbol{x}_k$, and $\boldsymbol{x}_j$ form a right triangle in space (as any larger value would cause this triangle to be acute, connecting $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$). Then, from Thales's theorem, the diameter D of $\mathcal{C}$ would equal that of the hypotenuse ℓ_{ij}, so

$$\kappa^{\max} = \frac{1}{D/2} = \frac{2}{D} = \frac{2}{\sqrt{r_{ik}^2 + r_{jk}^2}}. \tag{3.4}$$

A special case to consider is when $\mathcal{M}$ is uniformly sampled with constant interval T over arc length. Then the arc length s between i and j is $2T$, but since $r_{ij} = D$, s covers half the circle and we have $2T = \pi D/2$. Equation 3.4 then becomes

$$\kappa^{\max}(T) = \frac{\pi}{2T}. \tag{3.5}$$

Remark 5. Equations 3.4 and 3.5 define the maximum geodesic curvature in $\mathcal{M}$ that can be adequately inferred from a Gabriel graph. As a consequence, the reach is lower-bounded by $1/\kappa^{\max}$.

3.3.3 Degree Distribution in Gabriel Graphs. We now study the above connectivity rule starting with flat, uniformly sampled manifolds (i.e., "regular grids") to illustrate how Gabriel graphs naturally adapt to their geometry and dimensionality. As shown in Figure 9A, in such ideal cases, the degree of an interior node in the Gabriel graph agrees with the true number of (literal) nearest neighbors: two for collinear points, four for a square grid, and six for a triangular grid.

Node degree appears to grow with dimension as 2^d, except for the triangular grid (which, in some sense, looks too "nongeneric"). Adding noise (gaussian, with standard deviation equal to half the spacing between neighboring points) supports this conjecture, as the degree then approaches 2^d regardless of the original grid structure. This holds in higher dimensions as well, for both normal and uniform sampling at random (see Figures 9B, 9C, and 12).

Remark 6. The expected number of neighbors in a Gabriel graph approximately follows a distribution centered at 2^d (where d is the intrinsic dimension of the data) for a variety of sampling strategies (see Figure 9C).

Importantly, because Gabriel graphs are inherently scale invariant, this degree distribution is largely independent of sampling density.

How to explain such remarkable regularity despite the randomness of sampling? A complementary geometric view of the Gabriel graph connectivity rule is illuminating: each edge between data points implies an "occluding hyperplane" that blocks other points from becoming neighbors (see Figure 10). For example, when $d = 1$, two points necessarily occlude any additional connections and every nonboundary point must have two neighbors. Now, using the diagrams in Figure 11 as reference, we find that when $d = 2$, on average about 4 points are sufficient to occlude a point $\boldsymbol{x}_i$ from all sides. For $d = 3$ this number is doubled again, and the expected number of neighbors becomes about 8, revealing the trend. Every additional dimension adds a new coordinate axis along which the previous constraints are duplicated, roughly doubling the average number of directions available from which neighbors can connect. Once 2^d balls are "attached" to $\boldsymbol{x}_i$, the remaining space is greatly reduced, and so is the probability of drawing a sample point from inside the region $\mathcal{H}$ enclosed by the hyperplanes.

When the neighbors are regularly spread around $\boldsymbol{x}_i$, by construction this region $\mathcal{H}$ is equivalent to a d-dimensional orthoplex[2] (or cross-polytope). A

[2] An orthoplex is a line segment in 1D, a square in 2D, a regular octahedron in 3D, a 16-cell in 4D, and so on.

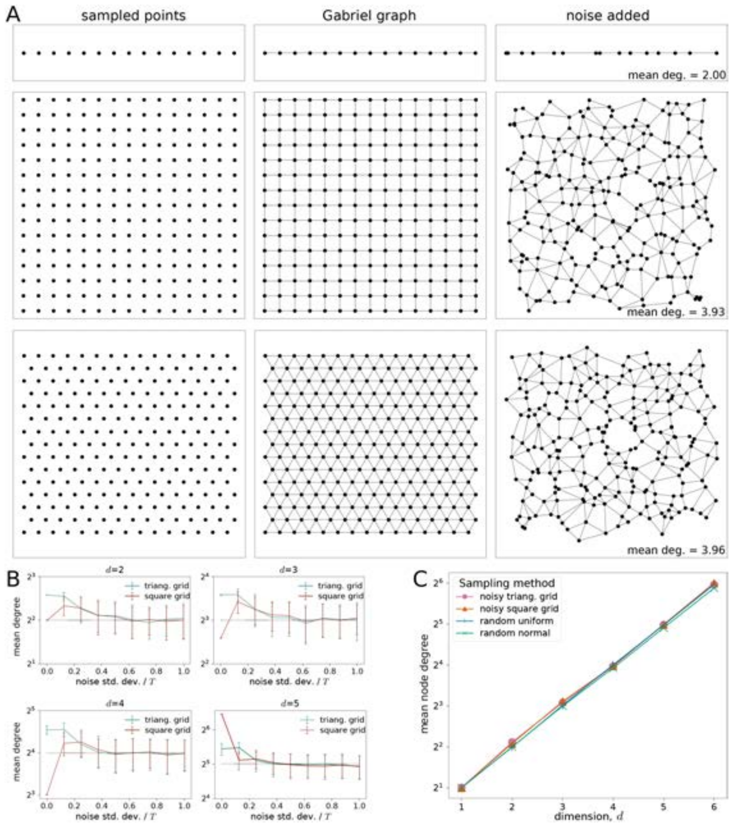


Figure 9: Regularity of node degree distribution in Gabriel graphs with random sampling. (A) Node degree in graphs computed from regular grids (constant sampling interval, T) and their jittered versions (gaussian noise with std. dev. $0.5T$). (Top) A sequence of collinear points (left) produces a one-dimensional grid (center). Addition of noise (right) does not change the mean degree (constant 2 for interior points). (Middle) A square grid (left) results in a quadrilateral mesh with constant degree 4 in its interior. Although addition of noise considerably scrambles the points, the mean degree is roughly unchanged. (Bottom) Points arranged as a triangular grid (left) result in a triangular mesh where every interior node has degree 6. Its noisy version looks similar to a noisy square grid, with mean degree also approximating 4. (B) Degree distribution for interior points of d-dimensional triangular and square grids after addition of gaussian noise. Moderate amounts of noise are sufficient to make the mean degree become approximately 2^d. Error bars indicate standard deviation; dotted lines show constant 2^n values for reference. (C) Mean degree of d-dimensional manifolds sampled using different strategies: uniformly at random, normally at random, and as jittered versions of regular triangular and square grids (as in panel A, added gaussian noise with std. dev. $0.5T$). Remarkably, mean degree grows approximately as 2^n regardless of the sampling strategy.

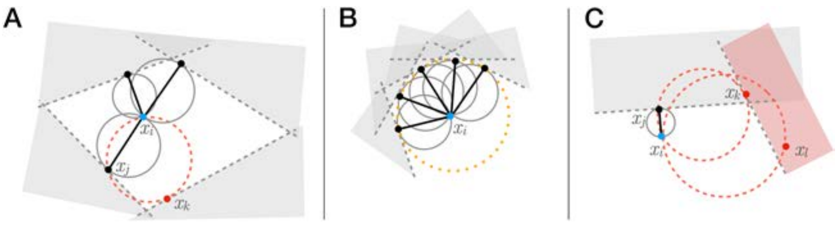


Figure 10: (A) A central point $\boldsymbol{x}_i$ (in blue) and its neighbors (in black). Every neighbor $\boldsymbol{x}_j$ of $\boldsymbol{x}_i$ will "occlude" the entire area behind a hyperplane tangent to $\mathcal{B}_{ij}$ at $\boldsymbol{x}_j$ (dashed lines). That is, no point inside the occluded areas (shaded region) can form a connection with $\boldsymbol{x}_i$. Here, the dashed ball does not form a connection between $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ because $\boldsymbol{x}_j$ lies exactly on its boundary; despite this, $\boldsymbol{x}_k$ still contributes with an occluding hyperplane, preventing farther points from connecting to $\boldsymbol{x}_i$. (B) In principle, there is no limit to the number of neighbors a point in ambient space $\mathbb{R}^n$ may have (when $n \geq 2$); for example, any number of points lying exactly on a hypersphere around $\boldsymbol{x}_i$ (dotted curve, in orange) will not occlude one another. Sets of nodes with connectivity such as this are termed "wheels" in graph theory, and the more points they contain, the less likely they are to occur in real data sets. In this example, any appreciable variability in the distance from $\boldsymbol{x}_i$ to its neighbors would cause one (or several) of them to become occluded. (C) Points inside occluded areas can also contribute with additional occluding hyperplanes. Here, although $\boldsymbol{x}_k$ lies inside the region occluded by $\boldsymbol{x}_j$ (and therefore cannot form a connection with $\boldsymbol{x}_i$), it produces further occlusion behind a hyperplane of its own (region shaded in red). So $\boldsymbol{x}_l$ cannot connect to $\boldsymbol{x}_i$ either due to the presence of $\boldsymbol{x}_k$ (even though it is not occluded by $\boldsymbol{x}_j$).

d-orthoplex has 2^d facets (or (d-1)-faces), and is one of the three finite, regular, convex polytopes that exist in dimension higher than 4 (the other two being hypercubes and simplices). Naturally, when sampling is not uniform, we should find irregular orthoplexes instead.

While this geometric construction supports our empirical results and implies they should hold in higher dimensions, it also suggests the following:

Remark 7. Our experiments on the growth in dimension of randomly sampled points agree with a model in which Gabriel neighbors lie approximately in the facets of an orthoplex.

We later use the additional observation that the dual polytope (a d-hypercube) of an orthoplex is obtained by placing a vertex (i.e., a neighbor) in each of its 2^d facets.

The Gabriel graph enjoys many attractive properties and provides the starting point for our algorithm. The above arguments show how the space is largely filled by "Gabriel balls" within the manifold, but such balls may also fill space across holes and bottlenecks; curvature must be dealt with.

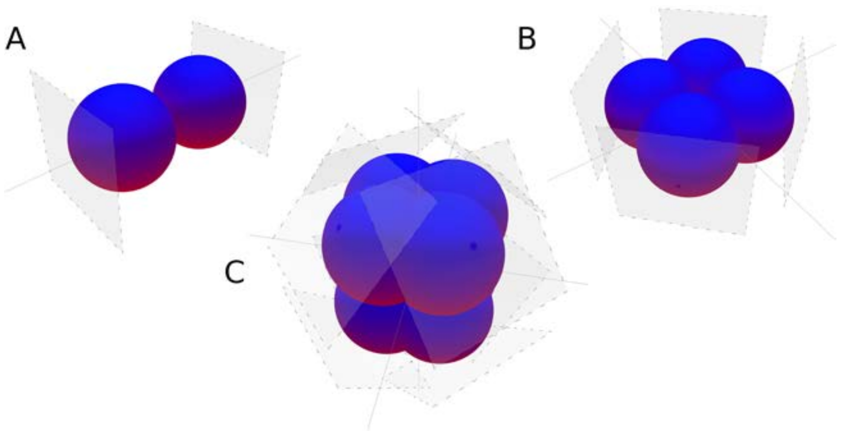


Figure 11: Occlusion hyperplanes (shown in gray) due to neighbors in dimensions 1, 2, and 3 (A–C, respectively); compare with Figure 10. Every additional dimension adds a new coordinate axis along which the previous constraints are duplicated, roughly doubling the average number of directions available from which neighbors can connect. Once 2^d Gabriel balls are "attached" to $\boldsymbol{x}_i$, the remaining space is greatly reduced, and so is the probability of drawing a sample point from inside the region enclosed by the hyperplanes.

Examples were given in Figure 6, where we showed that Gabriel connections can arise incorrectly and must be removed. To do so, one must "look" in every direction (of the tangent plane) and past immediate neighbors. For this, we now develop the weighted graph counterpart to the Gabriel graph, exploiting the kernel to extend local information globally. This begins to connect the graph construction more directly to manifold properties.

3.4 Multiscale Optimization.

We now begin to develop the iteration in algorithm 1, given the initial Gabriel neighborhood graph, $G^{(0)}$. Assuming (temporarily) that this gives correct local neighborhoods, what should the corresponding scales be for a gaussian kernel? In effect this is an extension of G into a weighted counterpart, $\mathcal{G}$. From Figure 5, this weighted graph is also a type of approximation of (aspects of) the continuous manifold. Because density is not necessarily uniform, different points might have different neighborhood radii, so a multiscale gaussian similarity kernel (see equation 3.1) is used. Each point $\boldsymbol{x}_i$ has its own associated scale, σ_i. To develop the computation of such scales, we now move into the continuous domain and exploit the geometric notion of a cover.

3.4.1 Covering Criterion.

A criterion for separability between two gaussians has been developed in the mixture-of-gaussians literature (Dasgupta,

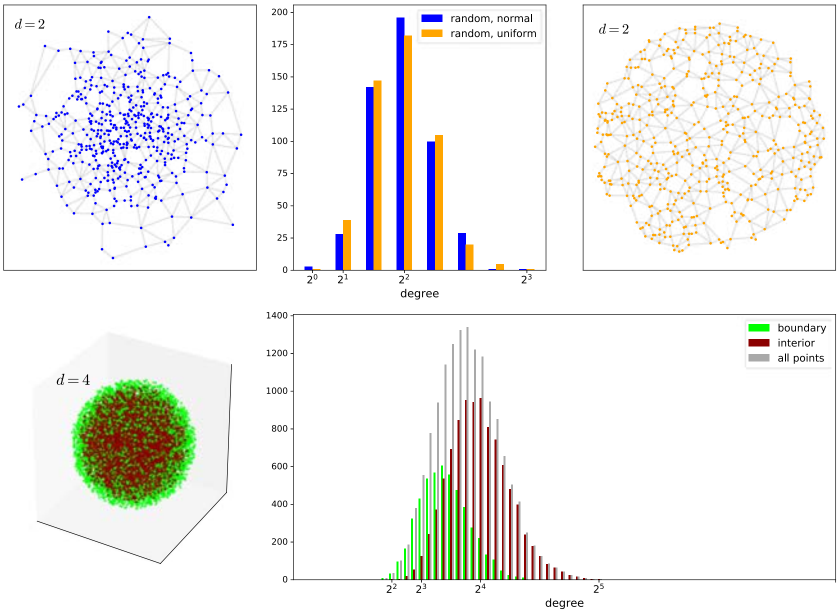


Figure 12: Distribution of node degree in the Gabriel graph of data sets with different sampling strategies and dimensionalities. (Top) Points sampled normally (blue) or uniformly (orange) at random from a two-dimensional ball result in similar degree distributions centered at 2^2. (Bottom) In higher dimensions, interior points continue to follow this pattern. On the left, a four-dimensional unit ball sampled uniformly at random is shown projected onto $\mathbb{R}^3$, with boundary points labeled as those with vector norm > 0.9 (edges omitted for clarity). It produces a Gabriel graph where interior points have degree distribution centered at $\sim 2^4$, and the mean degree of boundary points is close to 2^3.

1999; Vempala & Wang, 2004; Arora & Kannan, 2005): two spherical gaussians, i and j, can be distinguished (in the sense of solving a classification problem) with reasonable probability when they have a separation of at least

$$\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\| > C \max\{\sigma_i, \sigma_j\}, \tag{3.6}$$

at which the overlap in their probability mass is a constant fraction (Vempala & Wang, 2004).

We flip this around by using a different but related construction: consider gaussians now centered at the midpoints (i.e., not on data points) to indicate whether nearby points should be connected, not separated (Figure 5 illustrates this construction directly). Furthermore, because we use a multiscale kernel (see equation 3.1), the (nonnormalized) gaussian density

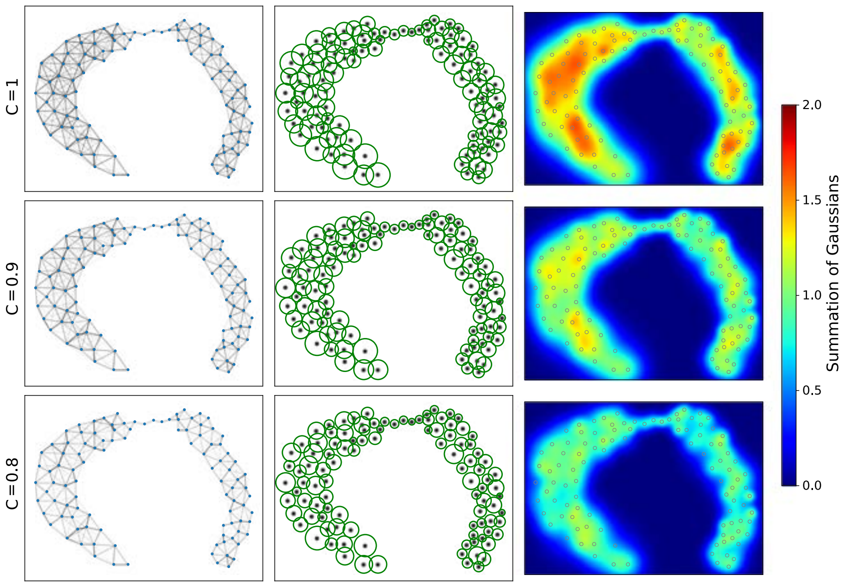


Figure 13: Effect of hyperparameter C from equation 3.7 on the resulting weighted graph (left), optimal scales (middle), and manifold approximation (right, shown as the resulting summation over the gaussian kernels around each point using their individual scales). For $C = 1$ (top), the scales overlap too much, and as a result, the gaussian summation (right) is highly nonuniform. For $C = 0.8$ (bottom), the scales are not sufficiently large to properly cover the underlying manifold, resulting in holes (right). When $C = 0.9$, there is a good compromise between covering and keeping a uniform density, so the gaussian summation approximates a partition of unity (summing to about 1 everywhere) when the scales correctly conform to the local sampling characteristics. Our approach will allow us to tune C based on a relaxation statistic, δ_i'.

becomes a function of $\sqrt{\sigma_i\sigma_j}$. Hence, we obtain a criterion for what we term *C-connectivity*:

Definition 1. *Two neighbors i and j in the discrete graph $G = (E, V)$ are C-connected by the multiscale kernel when the geometric mean of their individual scales is at least the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ scaled by a positive constant, C:*

$$C\|\boldsymbol{x}_i - \boldsymbol{x}_j\| \leq \sqrt{\sigma_i\sigma_j}. \tag{3.7}$$

The constant C plays a role in normalizing for unknown density; it will be developed in section 3.5.2. For now, we illustrate its role in the connection from graphs to manifolds. Figure 13 shows the graph over a set of data points and the local scales obtained (by the algorithm below) for different values of C. Choosing C too large yields scales (and therefore gaussians) that are too large, that is, their overlap has peaks. Choosing it too small yields scales that introduce holes. When it is chosen correctly, the gaussians form a covering of the manifold that approximates a partition of unity. Such partitions of unity are used in differential geometry to extend local information (in our case, the scales) to global information (a covering of the manifold).

By choosing appropriate scales (i.e., scales that meet our criterion for all edges in E), we also ensure a covering of the edges in the following sense: the value of the multiscale kernel K_{ij} between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is identical to that of a kernel recentered at the midpoint $\boldsymbol{p} \equiv (\boldsymbol{x}_i + \boldsymbol{x}_j)/2$ and rescaled using half the geometric mean of σ_i and σ_j as its scale, $\sigma_{\boldsymbol{p}}$:

$$K_{ij} = \exp \frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma_i \sigma_j} = \exp \frac{-\|(\boldsymbol{x}_i - \boldsymbol{x}_j)/2\|^2}{\sigma_i \sigma_j / 2^2} = \exp \frac{-\|(\boldsymbol{p} - \boldsymbol{x}_i)\|^2}{\sigma_{\boldsymbol{p}}^2}, \tag{3.8}$$

with $\sigma_{\boldsymbol{p}} \equiv \sqrt{\sigma_i \sigma_j}/2$ (see Figure 14).

Remark 8. We say a *C-covering* is attained when every pair $(i, j) \in E$ is *C-connected* (see equation 3.7). Additionally, when the spacing between neighboring points is approximately uniform locally, the pointwise summation over all gaussian kernel bumps given by the individual scales provides an (unnormalized) partition of unity of $\mathcal{M}$.

We now use the covering constraints to solve for the set of scales, $\boldsymbol{\sigma}$. It is desirable that the scales be small (respecting the reach) while at the same time maintaining the connectivity in $\mathcal{G}$ close to that of G. Thus, one idea is to find scales such that the sum of edge weights in $\mathcal{G}$ incident to a node i from its neighbors in G approximates the degree of i in G, for all i, while at the same time ensuring a C-covering. This, however, amounts to a nonconvex problem in which the cost function involves a summation of multiscale kernel values. We are unable to solve this efficiently. Instead, we find the smallest individual scales such that our covering criterion is satisfied for all edges (a "minimal covering") and later address the quality of the relaxation by using a statistical pruning (edge sparsification). This can be transformed into a convex, linear program with linear constraints by which all scales can be solved for simultaneously, as we show next. (We also present, in the appendix, a greedy approach to this optimization that may be convenient when dealing with very large data sets.)

3.4.2 Linear Program Relaxation. To achieve a minimal covering, one might minimize $\sum_i \sigma_i$ (or, equivalently, the 1-norm of the vector $\boldsymbol{\sigma}$, since

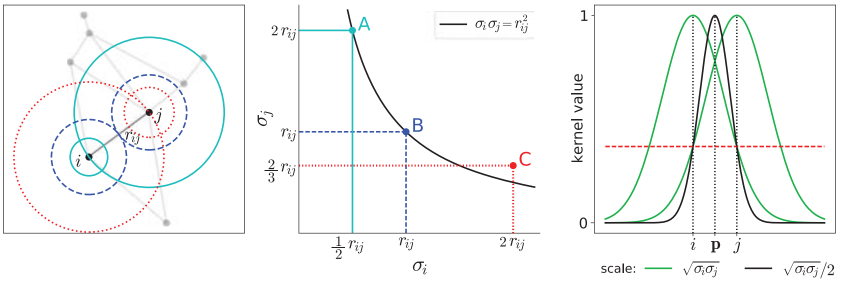


Figure 14: Covering constraint for the multiscale kernel of equation 3.1. (Left) A graph G with two nodes i and j at a distance r_{ij} from each other in $\mathbb{R}^n$. Since they are connected, their assigned individual scales σ_i and σ_j must satisfy $\sigma_i \sigma_j \geq r_{ij}^2$, the covering constraint (here we assume $C = 1$). (Center) All feasible pairs (σ_i, σ_j) lie inside the region above a positive hyperbola, three of which are indicated as colored points; pairs A and B satisfy exactly, while C satisfies in excess. Each is also depicted as a pair of circles on the left plot using the same color code, each one centered at its corresponding node (radii are set to half the scale, for clarity). Although pairs A and B differ in their ratio σ_i/σ_j, both result in the same multiscale kernel value for the edge (i, j), since the product $\sigma_i \sigma_j$ is the same; pair C yields a slightly higher value. This illustrates the freedom that might exist in choosing an optimal combination of scales for all nodes (i.e., a covering). (Right) Multiscale kernel values, K_{ij}, centered at either i or j, shown in green, are symmetric (with scale $\sqrt{\sigma_i \sigma_j}$). The horizontal axis represents position over the line in $\mathbb{R}^n$ passing through i and j. A kernel centered at the midpoint $\boldsymbol{p}$ between i and j using half the scale (black curve) attains the same value as K_{ij} at i and j. The dashed red line indicates the common value between the three kernels.

scales are positive) subject to the covering constraint.[3] This suggests the following:

Optimization Problem

$$
\begin{aligned}
\min_{\boldsymbol{\sigma}} \quad & \mathbf{1}^{\mathsf{T}} \boldsymbol{\sigma} \\
\text{s.t.} \quad & (i, j) \text{ is } C\text{-connected}, \ \forall\, (i, j) \in E \\
& \sigma_i \text{ is bounded}, \forall\, i \in V,
\end{aligned}
\tag{3.9}
$$

[3] Another possibility is to use a weighted sum $\sum_i \nu_i \sigma_i$ while keeping the same constraints, thus still guaranteeing a covering. The weights ν_i add a bias to how the length of an edge is split between its two incident nodes (by balancing their individual scales). One interesting option is to set $\nu_i = r_i^{\text{non}} / r_i^{\text{FN}}$, the ratio between the distance to the nearest nonneighboring point, r_i^{non}, and the farthest neighbor, r_i^{FN}.

where $\boldsymbol{\sigma}$ is the vector of individual scales, σ_i, and $\mathbf{1}$ is the all-ones vector. Now it remains to represent the C-covering requirement by a set of constraints.

Looking in detail at C-connectedness (equation 3.7) as a function of σ_i and σ_j, observe that it represents a region delimited by a single-branched hyperbola (since the distance and scales are positive),

$$\sigma_i\sigma_j \geq (Cr_{ij})^2, \quad \sigma_i > 0, \sigma_j > 0, \tag{3.10}$$

where $r_{ij} \equiv \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\mathbb{R}^n}$. Each σ_i is naturally bounded above by the distance to i's farthest neighbor, r_i^{FN}:

$$\sigma_i \leq r_i^{\mathrm{FN}}, \tag{3.11}$$

beyond which all neighbors are satisfied,[4] so further increasing either scale would make the weights to nonneighbors larger than strictly necessary (thereby hurting the kernel graph relaxation). These bounds, combined, specify a bounding box for each edge that must necessarily be crossed (or at least touched) by the hyperbola, since $r_{ij} > 0$.

Due to the hyperbolas, this amounts to a non-linear, non-convex set of constraints. However, we can convexify the feasible set by considering, for each edge (i, j), the line(s) passing through the hyperbola's vertex (the point at which $\sigma_i = \sigma_j = Cr_{ij}$) and the points where the hyperbola intersects the bounding box. The four possibilities are shown in Figure 15. The feasible region for each edge therefore is bounded by a convex envelope given by such line(s) and those defined by the upper bounds to σ_i and to σ_j. Such envelopes for all edges, combined, define the boundaries of a convex polytope. Note that this convexification is conservative in the sense that only the objective is relaxed—the feasible scales are always at least as large as required by the original nonconvex problem; therefore, our covering requirement is not relaxed. (Because of the presence of a later pruning stage in the algorithm, it is better to overconnect here than to inadvertently disconnect nodes that should otherwise be connected.)

Letting $m \leq 2|E|$ be the total number of linear constraints obtained as above and N the number of nodes in G, we define the $m \times N$ matrix $\boldsymbol{\Lambda}$ and the $m \times 1$ vector $\boldsymbol{b}$. Now, for each edge, e_{ij}, let its two possible constraints be expressed as

$$\sigma_j \geq \alpha_{ij}^{(1)}\sigma_i + \beta_{ij}^{(1)}, \tag{3.12}$$

$$\sigma_j \geq \alpha_{ij}^{(2)}\sigma_i + \beta_{ij}^{(2)}, \tag{3.13}$$

[4]That is assuming $C \leq 1$ (a natural choice). If for some reason one needs to allow $C > 1$, then the upper bounds must be scaled by C in order to ensure feasibility.

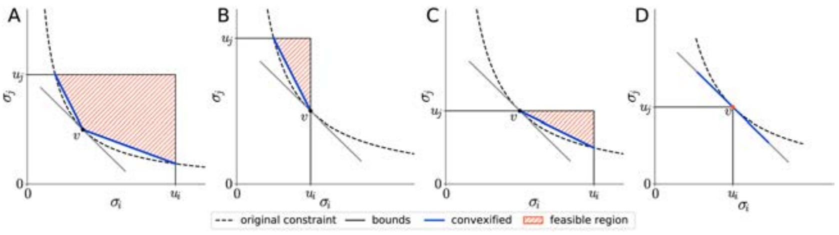


Figure 15: Examples of constraints introduced by an edge, e_{ij}, in G. The C-connectivity rule, that is, the hyperbola given by $\sigma_i\sigma_j = (C\|\boldsymbol{x}_i - \boldsymbol{x}_j\|)^2$ (dashed curve), when convexified, may give rise to one or two linear constraints, depending on whether the hyperbola's vertex v (point where $\sigma_i = \sigma_j$) intersects the bounding box given by the lines $\sigma_i = 0, \sigma_j = 0, \sigma_i = u_i$, and $\sigma_j = u_j$, where u_i and u_j denote upper bounds. The hatched area (in orange) shows the feasible region using convexified constraints; the tangent line at v is shown in gray. When v is interior to the bounding box (A), two secants (in blue) define the feasible region (namely, the lines passing through v and the points where the hyperbola intersects the lines $\sigma_i = u_i$ and $\sigma_j = u_j$); when either $v = u_i$ (B) or $v = u_j$ (C), only one secant is necessary; when v coincides with both u_i and u_j (D) (which may occur if C is set to 1), again only one inequality is necessary, namely the tangent line at v.

with α_{ij} and β_{ij} denoting, respectively, the slope and intercept of the corresponding line(s) forming its convex envelope. Rearranging, we obtain $\alpha_{ij}\sigma_i - \sigma_j \leq -\beta_{ij}$ for each line, which is encoded as a row in $\boldsymbol{\Lambda}$ with values α_{ij} and -1 at columns i and j, respectively (with zeros everywhere else), and an entry in $\boldsymbol{b}$ with value $-\beta_{ij}$:

$$
\begin{array}{c}
 \\
 \\
\vdots \\
e_{ij}^{(1)} \\
e_{ij}^{(2)} \\
\vdots \\

\end{array}
\begin{array}{c}
\boldsymbol{\Lambda} \\
\begin{array}{ccccccccc} & \cdots & i & & \cdots & & j & \cdots & \end{array} \\
\left[\begin{array}{ccccccccc}
 & \vdots & & \vdots & & \vdots & & \vdots & \\
0 & \cdots & \alpha_{ij}^{(1)} & \cdots & 0 & \cdots & -1 & \cdots & 0 \\
0 & \cdots & \alpha_{ij}^{(2)} & \cdots & 0 & \cdots & -1 & \cdots & 0 \\
 & \vdots & & \vdots & & \vdots & & \vdots &
\end{array}\right] \\
m \times N
\end{array}
\begin{array}{c} \\ \\ \boldsymbol{\sigma} \\ \\ N \times 1 \end{array}
\begin{array}{c} \\ \\ \leq \\ \\ \end{array}
\begin{array}{c}
\boldsymbol{b} \\
\\
\left[\begin{array}{c} \vdots \\ -\beta_{ij}^{(1)} \\ -\beta_{ij}^{(2)} \\ \vdots \end{array}\right] \\
m \times 1
\end{array}.
$$

Remark 9. The convex envelope defining the constraints can be expressed by the linear inqualities,

$$\begin{aligned} \boldsymbol{\Lambda}\boldsymbol{\sigma} &\leq \boldsymbol{b}, \\ \mathbf{0} < \boldsymbol{\sigma} &\leq \boldsymbol{r}^{\mathrm{FN}}, \end{aligned} \tag{3.14}$$

where $\boldsymbol{r}^{\mathrm{FN}}$ is the vector of distances to each node's farthest neighbor.

Hence the problem now amounts to a convex, linear program (LP) with linear constraints:

Optimization Problem: LP Relaxation

$$\begin{aligned} \min_{\boldsymbol{\sigma}} \quad & \mathbf{1}^{\mathsf{T}}\boldsymbol{\sigma} \\ \text{s.t.} \quad & \boldsymbol{\Lambda}\boldsymbol{\sigma} \leq \boldsymbol{b} \\ & \mathbf{0} < \boldsymbol{\sigma} \leq \boldsymbol{r}^{\mathrm{FN}}, \end{aligned} \tag{3.15}$$

which can be readily solved by a variety of methods (see, e.g., Boyd & Vandenberghe, 2004). Figures 19, 21, and 20 show the results of running this optimization on different examples.

3.5 Sparsification. Summarizing what we have seen so far, the Gabriel graph provides an initial estimate of connectivity, while the LP optimization provides minimal scales for a continuous kernel to cover those connections. However, since the initial estimate of the discrete graph might contain incorrect connections, its resulting optimal scales might also be inadequate. An example of this can be seen in Figure 19: initially, two pairs of nodes are connected across the central gap since a Gabriel ball exists between them. This will require very large scales to "cover" these edges. Furthermore, the Gabriel graph is based on a local connectivity rule; however, as illustrated in Figure 16, decisions about connecting nodes across a gap should not be local. We here address both of these issues by introducing a global statistic based on how frequently such a gap occurs in the data. In terms of algorithm 1, we are now at steps 6 and 7.

3.5.1 Volume Ratio. Because incorrect connections can be given by Gabriel balls lying in the free space between parts of a manifold (i.e., across the medial axis), it is tempting to simply prune the longest connections. Note, however, that the size of a scale by itself is not necessarily important. In both examples shown in Figure 6, the nonuniform density causes scale sizes to vary considerably, and even the largest ones are appropriate, that is, they are still consistent with the distances to neighboring points.

Conversely (and importantly), a scale that is excessively large will likely cover "too many" points. That is, it will cover neighbors in excess of the number of discrete neighbors of its corresponding node in G. We quantify this notion by observing that an individual scale, σ_i, should produce kernel values whose sum is comparable to the discrete degree, $\deg(i)$, of node i

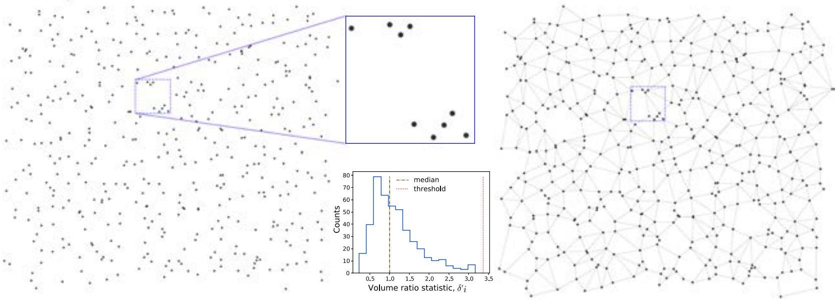


Figure 16: Local versus global assessment of neighborhoods. (Left) The points inside the cropped window appear to form two well-defined clusters when looked at up close (local estimation). However, when considered in the context of the full data set (global estimation), the apparent gap between the top and bottom groups "disappears," well within the range of gaps observed throughout the data. More precisely, it does not significantly deviate from the average sampling interval. (Right) The converged graph G indeed connects the two groups by edges, and the distribution of volume ratios, δ'_i (lower inset), confirms that all edges are reasonable.

in G. As will be shown, after proper normalization, this also means σ_i shall relate to a local volume element around $\boldsymbol{x}_i$, or the inverse of the local density. Since each connection in G can be seen as having unit weight, a gaussian kernel around $\boldsymbol{x}_i$ with scale σ_i should distribute that same amount, $\deg(i)$, only continuously over ambient space.

We start our derivation with a definition:

Definition 2. *Let $w_{ij}^{(\sigma_i)}$ be the gaussian kernel value between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ using scale $\sigma_i^{(t)}$ at iteration t. A (nonisolated) node's* volume ratio *at iteration t, denoted by $\delta_i^{(t)}$, is defined as*

$$\delta_i^{(t)} \equiv \frac{\sum_{j\in V} w_{ij}^{(\sigma_i^{(t)})}}{\sum_{j\in V} a_{ij}}, \tag{3.16}$$

the ratio between node i's weighted degree due to $\sigma_i^{(t)}$ and its discrete degree in $G^{(t)}$ (henceforth, we suppress the iteration dependency (t) to simplify notation).

An individual-scale gaussian kernel is needed to correctly assess the impact of σ_i on the relaxation from the perspective of i alone. The multiscale kernel here might artificially increase the weighted degree of i when other nodes (even nonneighbors of i!) have incorrect scales. (Nevertheless, as discussed below, a corresponding ratio using the actual weights in $\mathcal{G}$ may eventually be used for convergence purposes.)

Now, using a mean-value integral (as in Coifman et al., 2008), the numerator approximates the volume under the continuous gaussian kernel over $\mathcal{M}$ and can be further approximated by

$$\sum_j w_{ij}^{(\sigma_i)} \approx \frac{N}{\mathrm{vol}(\mathcal{M})} \int_{\mathcal{M}} \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma_i^2}\right) d\boldsymbol{x}_j \tag{3.17}$$

when $\mathcal{M}$ has uniform density and low curvature. In practice, the kernel will have compact support due to numerical precision (i.e., its values become effectively zero for sufficiently large distances), so by defining the volume element $dV_i \equiv \mathrm{vol}(\mathcal{N}(\boldsymbol{x}_i))/|\mathcal{N}(\boldsymbol{x}_i)|$ of a neighborhood $\mathcal{N}(\boldsymbol{x}_i) \in \mathcal{M}$ around $\boldsymbol{x}_i$, we may rewrite equation 3.17 as

$$\sum_j w_{ij}^{(\sigma_i)} dV_i \approx \int_{\mathcal{M}} \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma_i^2}\right) d\boldsymbol{x}_j \tag{3.18}$$

when the sampling is approximately uniform around $\boldsymbol{x}_i$. By further assuming that σ_i is small and that $\mathcal{M}$ can be well approximated locally by its tangent space $\mathbb{R}^d$, then

$$\int_{\mathcal{M}} \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma_i^2}\right) d\boldsymbol{x}_j \approx \int_{\mathbb{R}^d} \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma_i^2}\right) d\boldsymbol{x}_j = (\sqrt{\pi}\sigma_i)^d, \tag{3.19}$$

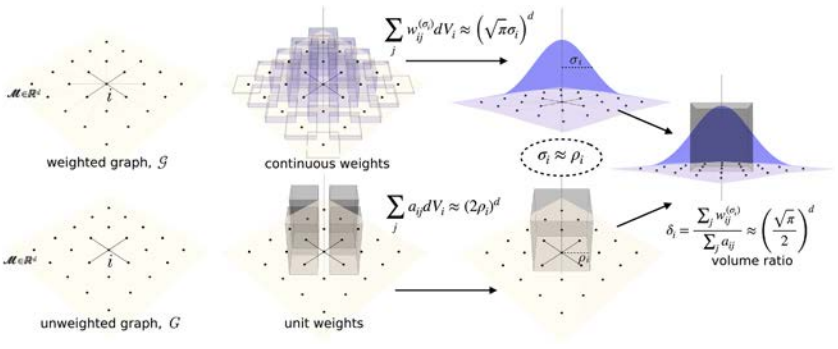
so

$$\sum_j w_{ij}^{(\sigma_i)} dV_i \approx (\sqrt{\pi}\sigma_i)^d, \tag{3.20}$$

as shown in Figure 17.

An analogous derivation for the discrete degree summation is as follows. First, note that the edge weight in this case is a constant (unity); it remains to determine its support over $\mathcal{M}$. From section 3.3.3, we know that, for simple manifolds with random sampling, the node degree $\deg(i)$ in a Gabriel graph is approximately 2^{d_i} within a region of constant intrinsic dimensionality, where d_i denotes the local intrinsic dimension around $\boldsymbol{x}_i$ (possibly different around other points in $\mathcal{X}$).[5] In more general manifolds, we expect the converged graph $G^\star$ instead to approach such a property. This means $\sum_j a_{ij} \approx 2^{d_i}$ will approximate the volume of a hyperrectangle (or box) of unit height and having a d_i-dimensional hypercube of side 2 as its

[5]We abuse notation, therefore, when we say "d-dimensional manifold" or "$\mathcal{M} \in \mathbb{R}^d$."



$\sum_j w_{ij}^{(\sigma_i)} dV_i \approx \left(\sqrt{\pi}\sigma_i\right)^d$
σ_i
$\mathcal{M} \in \mathbb{R}^d$
i
weighted graph, $\mathcal{G}$
continuous weights
$\sigma_i \approx \rho_i$
$\sum_j a_{ij} dV_i \approx (2\rho_i)^d$
$\delta_i = \frac{\sum_j w_{ij}^{(\sigma_i)}}{\sum_j a_{ij}} \approx \left(\frac{\sqrt{\pi}}{2}\right)^d$
volume ratio
ρ_i
unweighted graph, G
unit weights

Figure 17: Computing the volume ratio between continuous and discrete degrees of a node i with neighboring points sampled uniformly over a d-dimensional manifold $\mathcal{M}$. (Top row) Using a gaussian kernel, the weighted degree of i (sum of kernel values $\sum_j w_{ij}^{(\sigma_i)}$) in $\mathcal{G}$ approximates the volume of a gaussian with scale σ_i (equation 3.20). (Bottom row) The number of edges adjacent to i in G (sum of unit weights) approximates the volume of a box with unit height and a hypercube of side $2\rho_i$ as its base, where ρ_i is the radius of a local volume element of $\mathcal{M}$ around $\boldsymbol{x}_i$ (see equation 3.21). (Right) When the scale σ_i is compatible with ρ_i, the volume ratio, δ_i, is expected to be approximately $(\sqrt{\pi}/2)^d$, and therefore is a scale-invariant quantity.

base.[6] So, by defining ρ_i as the radius of the local volume element dV_i (such that $\rho_i = \sqrt[d_i]{dV_i}$), we may write:

$$\sum_j a_{ij} dV_i \approx \int_{-\rho_i}^{\rho_i} \cdots \int_{-\rho_i}^{\rho_i} 1 dx_{j_1} \ldots dx_{j_d} = (2\rho_i)^{d_i}, \tag{3.21}$$

as illustrated in Figure 17. Hence, ρ_i is a kind of "neighborhood radius" of $\boldsymbol{x}_i$.

From equations 3.20 and 3.21, equation 3.16 becomes

$$\frac{\sum_j w_{ij}^{(\sigma_i)}}{\sum_j a_{ij}} = \frac{\sum_j w_{ij}^{(\sigma_i)}\, dV_i}{\sum_j a_{ij}\ dV_i} \approx \left(\frac{\sqrt{\pi}\sigma_i}{2\rho_i}\right)^{d_i}, \tag{3.22}$$

representing the ratio between the volume of a gaussian with scale σ_i and that of a box of side $2\rho_i$ and height 1 (cf. Figure 17). As the algorithm approaches convergence, we expect $\sigma_i \approx \rho_i$ (scales are compatible with

[6]This agrees with our observation (in section 3.3.3) that the unoccluded region around $\boldsymbol{x}_i$ is similar to a d_i-orthoplex: by placing a vertex (i.e., a neighbor) in each of its 2^{d_i} facets, we obtain a d_i-hypercube, the dual polytope of an orthoplex.

neighborhood radius) and deg(i) should approach, on average, the empirically observed value of 2^{d_i} (meaning that the number of neighbors in G is compatible with dimensionality of $\mathcal{M}$). This results in

$$\frac{\sum_j w_{ij}^{(\sigma_i)}}{\sum_j a_{ij}} \approx \left(\frac{\sqrt{\pi}}{2}\right)^{d_i}. \tag{3.23}$$

Finally, we can estimate d_i as

$$\tilde{d_i} \equiv \log_2 \sum_j a_{ij}, \tag{3.24}$$

based on the empirical degree distribution of $G^{(t)}$. From this, we can compute a normalized volume ratio, $\delta_i'^{(t)}$, dividing $\delta_i^{(t)}$ by the value from equation 3.23:

Definition 3. *A node's* normalized volume ratio *is computed as*

$$\delta_i'^{(t)} \equiv \frac{\sum_j w_{ij}^{(\sigma_i)}}{\sum_j a_{ij}} \left(\frac{2}{\sqrt{\pi}}\right)^{\tilde{d_i}}. \tag{3.25}$$

Nodes whose degree deviate from exactly 2^{d_i} will, likewise, under- or overestimate the local dimension, so reasonable volume estimates are still obtained regardless. However, in order to avoid a dimension less than 1 for connected nodes, in practice when deg(i) $= 1$, we replace $\sum_j a_{ij}$ with $\max\{2, \sum_j a_{ij}\}$.

Thus, we expect $\delta_i' \approx 1$ for points obeying $\sigma_i \approx \rho_i$ and $\tilde{d_i} \approx d_i$. Crucially, points for which these conditions are not met (those having "wrong" neighbors in the original Gabriel graph, $G^{(0)}$) will depart from this by having $\delta_i' \gg 1$. In the next section, we use this fact to guide a sparsification of edges in $G^{(0)}$ based on δ_i'.

Interestingly, δ_i' can also be interpreted as measuring how well the scale σ_i fits the local volume element dV_i (or, equivalently, how it counteracts the local sampling density, $1/dV_i$). Since $dV_i = \rho_i^{d_i}$ (from the definition of ρ_i), we may rewrite equation 3.22 as

$$\frac{\sum_j w_{ij}^{(\sigma_i)}}{\sum_j a_{ij}} \approx \frac{(\sqrt{\pi}\sigma_i)^{d_i}}{2^{d_i} dV_i}. \tag{3.26}$$

Summarizing the above, when $\tilde{d_i} \approx d_i$ and $\sigma_i \approx \rho_i$ we have:

Remark 10. A node's normalized volume ratio may alternatively be expressed as

$$\delta_i'^{(t)} \equiv \frac{\sum_j w_{ij}^{(\sigma_i)}}{\sum_j a_{ij}} \left(\frac{2}{\sqrt{\pi}}\right)^{\tilde{d}_i} \approx \frac{(\sqrt{\pi}\sigma_i)^{d_i}}{2^{d_i} dV_i} \left(\frac{2}{\sqrt{\pi}}\right)^{\tilde{d}_i} \approx \frac{\sigma_i^{d_i}}{dV_i}. \tag{3.27}$$

Therefore, δ_i' can be thought of as the product between kernel scale and local density. When σ_i is optimal, it should be approximately equal to the inverse of the local density, so $\delta_i' \approx 1$.

3.5.2 Uniformity of Sampling and Edge Pruning. Since $\delta_i'^{(t)}$ is evaluated for every node $\boldsymbol{x}_i$, we can collect it across nodes and view it as a statistic. This has two consequences: (1) it can be used to enforce consistency in sampling, and (2) outliers in this statistic are likely candidates for edge pruning. We address consistency of sampling first.

We have several times stated that sampling is required to be locally uniform, although its rate may change over the manifold. Examples of this were shown in, for example, Figure 12, where the sampling was denser in the center of the gaussian distribution than in the periphery. This example differs from the regular grids in which all nearest neighbors had exactly the same distance. Putting this together, we have:

Remark 11. Locally uniform sampling: Let node i have k_i neighbors in $G^{(t)}$. Among these, let r_i^{FN} denote the distance from $\boldsymbol{x}_i$ to its farthest neighbor, and r_i^{NN} that to its nearest neighbor. When $r_i^{\mathrm{FN}} \approx r_i^{\mathrm{NN}}$ for all i, we say the sampling is locally uniform.

This is useful because a departure from the assumption that sampling is locally uniform will cause δ_i' to be on average greater than 1 throughout the data set. To see this, when sampling is not uniform, we have $r_i^{\mathrm{FN}} > r_i^{\mathrm{NN}}$. Now, since σ_i is optimized to cover all of i's neighbors, it will have in most cases the same order of magnitude as r_i^{FN} (minus some possible slack due to the multiscale interaction). Therefore, the higher the variability in the neighbors' distances, the larger the difference between r_i^{FN} and r_i^{NN} will be, making σ_i, in turn, be larger than the distance to most neighbors of i. Ultimately, this will increase $\sum_j w_{ij}^{(\sigma_i)}$ beyond what we would have in a uniform-sampling scenario (in which $r_i^{\mathrm{FN}} \approx r_i^{\mathrm{NN}}$).

When data are acquired using a global sampling strategy, this variability in the neighbors' distances should be roughly constant throughout the data set (rather than the distances). So we use the scalar parameter, C, from equation 3.7 to correct for this "bias" and bring the median of the distribution of $\delta_i'^{(t)}$ (denoted as $\langle \delta_i'^{(t)} \rangle$) close to 1.[7]

Remark 12. Let the tuned $C^{\star(t)}$ be that which causes $\langle \delta_i'^{(t)} \rangle$ to be closest to 1.

[7]Although the mean typically gives smoother tuning curves, the median is more robust. This matters because of the possible outlying δ_i' values.

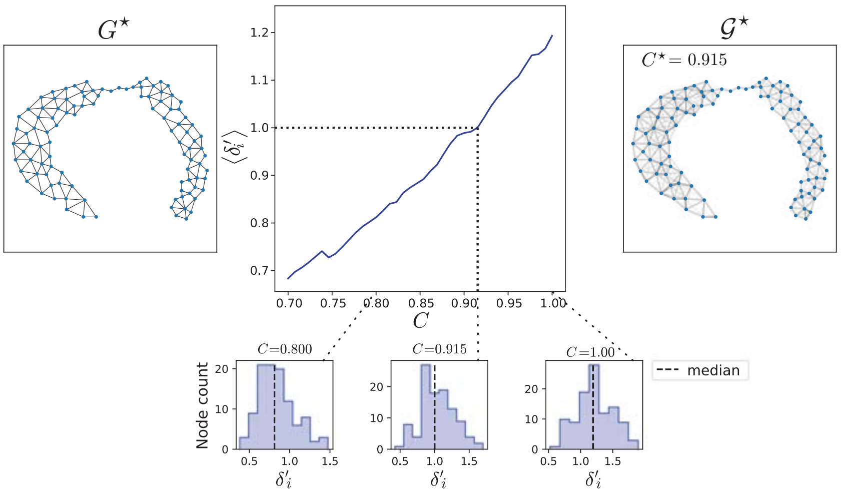


Figure 18: Tuning the hyperparameter C based on the median of the distribution of normalized volume ratios, $\langle \delta_i' \rangle$. (Left) Converged unweighted graph $G^\star$ obtained for the data set from Figure 19. (Center) After computing $\langle \delta_i' \rangle$ for a range of values of $C \leq 1$, the optimal $C^\star$ is that resulting in a $\langle \delta_i' \rangle$ closest to 1. Histograms below show the distribution of δ_i' for different values of C, including $C^\star = 0.915$. (Right) The resulting weighted graph $\mathcal{G}^\star$ after C-tuning typically exhibits a more uniform connectivity throughout (see Figure 13).

Typically, $C^{\star(t)} < 1$, which, in the scale optimization procedure, means that the covering constraints (equation 3.10) are being relaxed using the distribution of δ_i' as a guide (see Figure 18). Note that although the tuning of C is not necessary for finding candidates for sparsification, it attributes a quantitative meaning to the value of δ_i', so any $\delta_i' \gg 1$ is guaranteed to indicate the need for edge pruning. Such tuning should be performed at $t = 0$ and repeated as needed over the iterations whenever $\langle \delta_i'^{(t)} \rangle$ deviates too much from unity (which may happen after several edges have been pruned). Most commonly, we find $0.5 < C^{\star(t)} < 1$.

Thus, we have a data-driven way of finding an appropriate value for C. Because it is a global constant applied to all connection constraints, it shifts the distribution of δ_i' to have a median around 1 without changing its general shape.

This leads us to the second use of our statistic: any node whose normalized volume ratio is much greater than the median of the population should be identified as an outlier. Such nodes will have a neighbor considerably farther than its other neighbors (relative to the median variability of such neighboring distances throughout the data) and are candidates for the sparsification step.

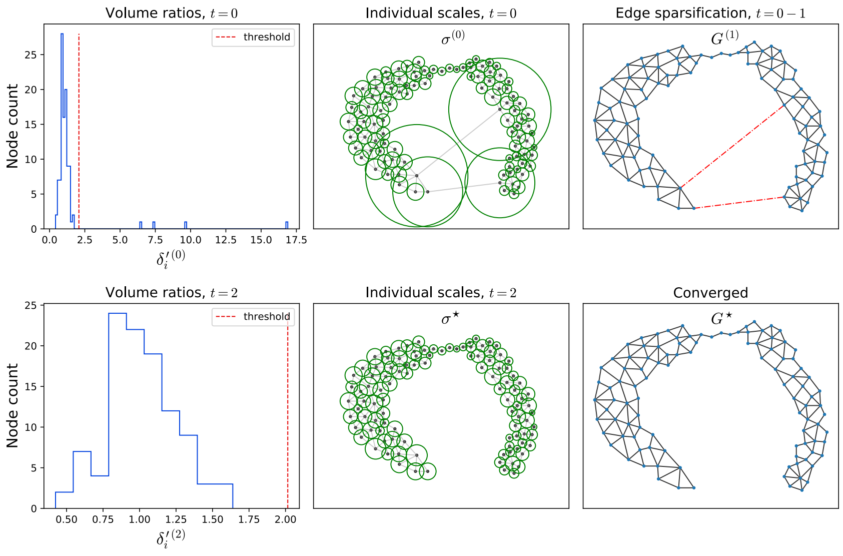


Figure 19: Optimal scales and associated normalized volume ratios, δ_i', at iterations 0 and 2 of algorithm 1 on the horseshoe data set (see Figure 13). (Top row) The δ_i' statistic has a median around 1 and several outliers. These are caused by the long edges and huge scales (middle). (Right column) $G^{(t)}$ after iteration 1, with edges deleted shown in red (top) and after iteration 2 (bottom).

Remark 13. Nodes that are robust outliers according to the δ_i' statistic have an overly distant neighbor (relative to the other neighbors for that node) and hence are likely to be in violation of reach or other geometric constraints. These relatively distant neighbors are candidates for having an edge pruned.

Given the distribution of normalized volume ratios, statistical models can be used to define a threshold for identifying outliers (see Figures 19 to 22). It is likely that data sets with a large number of problematic connections will exhibit a distribution with a heavy tail or that looks like a mixture of two distributions (see the example in Figure 22), so using the distribution's quartiles may give a more robust result. One option that seems to work particularly well is to use estimates of the sample mean and standard deviation from the quartiles, as in Wan, Wang, Liu, and Tong (2014) (throughout, we make use of the C3 method derived there, setting the δ_i' threshold to 4.5 standard deviations above the mean thus estimated). Still, we found that results are typically quite invariant to this particular choice, especially in real-life data sets. Finally, we note that our algorithm can be run interactively, so the user can analyze the histogram of the distribution after each iteration to judge whether the choice of threshold is reasonable and thus be confident in the results.

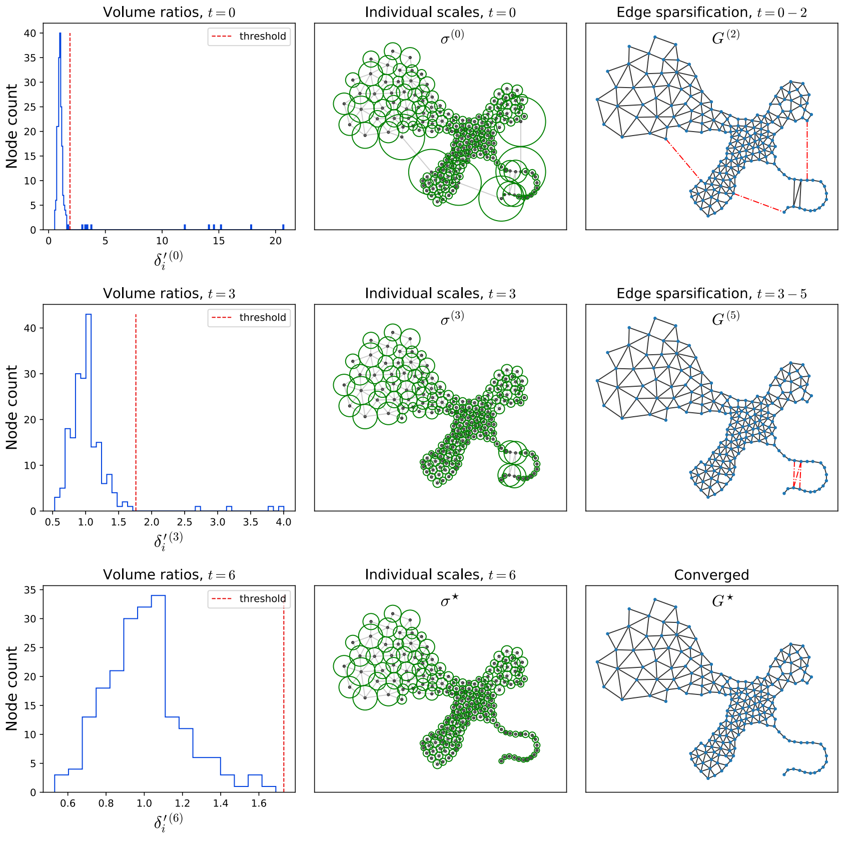


Figure 20: Optimal scales and associated normalized volume ratios, δ_i', at different iterations of algorithm 1 on the data set from Figure 6. The distribution of δ_i' (left) indicates the connections that are least likely to represent reasonable geodesics over the underlying manifold. The right column shows $G^{(t)}$ after iterations 2, 5, and 6 (deleted edges in red).

Nodes with δ_i' above the threshold should have their connection to their farthest neighbor deleted. Ideally, only one such connection is pruned after each iteration; however, should that become impractical with large data sets, a compromise is to limit the pruning, at each iteration, to a single edge from each node that is above the threshold (giving the chance for its δ_i' value to be updated before the next pruning).

3.5.3 Convergence. The algorithm converges at iteration t when no point i has an outlying $\delta_i'^{(t)}$ (i.e., greater than a statistical threshold). This implies that no edges will be pruned, so $G^{(t+1)} = G^{(t)}$, and therefore no further changes can occur to either $\boldsymbol{\sigma}^{(t)}$ or $\mathcal{G}^{(t)}$. Note that convergence is guaranteed:

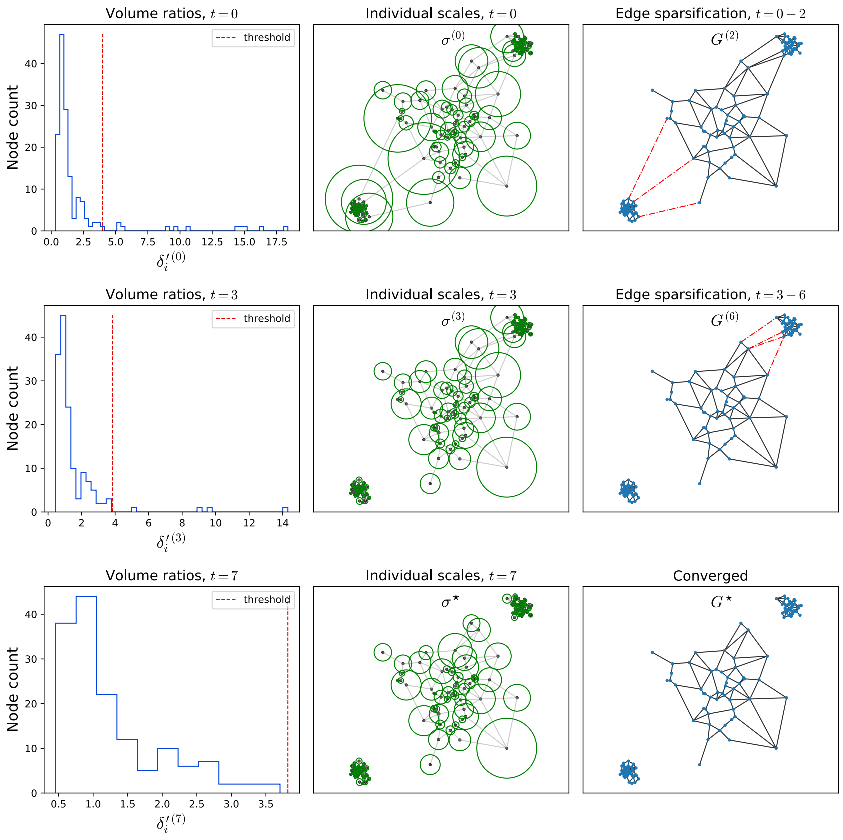


Figure 21: Optimal scales and associated normalized volume ratio distributions at different iterations of algorithm 1 on the clustered data set from Figure 6. Pruned edges (in red) are precisely those connecting the three clusters together.

since at every iteration t an edge is removed, the algorithm must necessarily reach a certain t at which all outliers (if there were any to begin with) have been pruned.

If one is solely interested in obtaining $\mathcal{G}^{\star}$ (i.e., not interested in $G^{\star}$), an alternative convergence condition may be adopted that looks at the distribution of the (normalized) multiscale volume ratio, $\delta_{i_{\mathrm{MS}}}'^{(t)}$,

$$\delta_{i_{\mathrm{MS}}}'^{(t)} \equiv \frac{\sum_j w_{ij}}{\sum_j a_{ij}} \left(\frac{2}{\sqrt{\pi}}\right)^{\tilde{d}_i}, \tag{3.28}$$

analogous to equation 3.25 but using the weights from $\mathcal{G}^{(t)}$ directly. Since the multiscale kernel takes into account the interaction of individual scales,

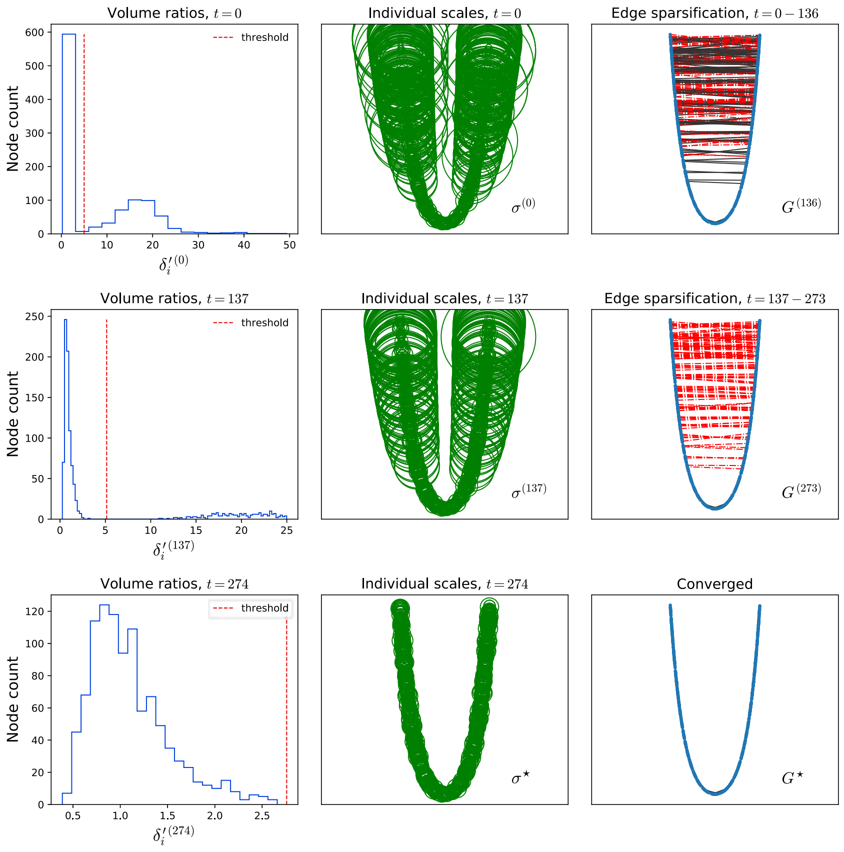


Figure 22: Optimal scales and associated normalized volume ratios, δ_i', after each iteration of the algorithm on the data set from Figure 28 (here, seen from a lateral view). The number of initial connections in $G^{(0)}$ (Gabriel graph) is very large, so the initial distribution of δ_i' shows two modes. However, ratios in the right-side peak are very high and are therefore easily identified as outliers. The algorithm converges soon after all edges crossing the gap are eliminated.

the distribution of $\delta_{i_{\mathrm{MS}}}'$ will be typically tighter than that of δ_i' (i.e., some of the excessively large scales might be compensated by small neighboring scales). Therefore, one may wish to allow for an earlier convergence when there are no remaining outliers in the distribution of $\delta_{i_{\mathrm{MS}}}'$.

Finally, in applications where it is required that $G^{\star}$ be connected, pruning can simply be stopped before disconnection. Naturally, $\mathcal{G}^{\star}$ is always connected up to machine precision or some numerical tolerance.

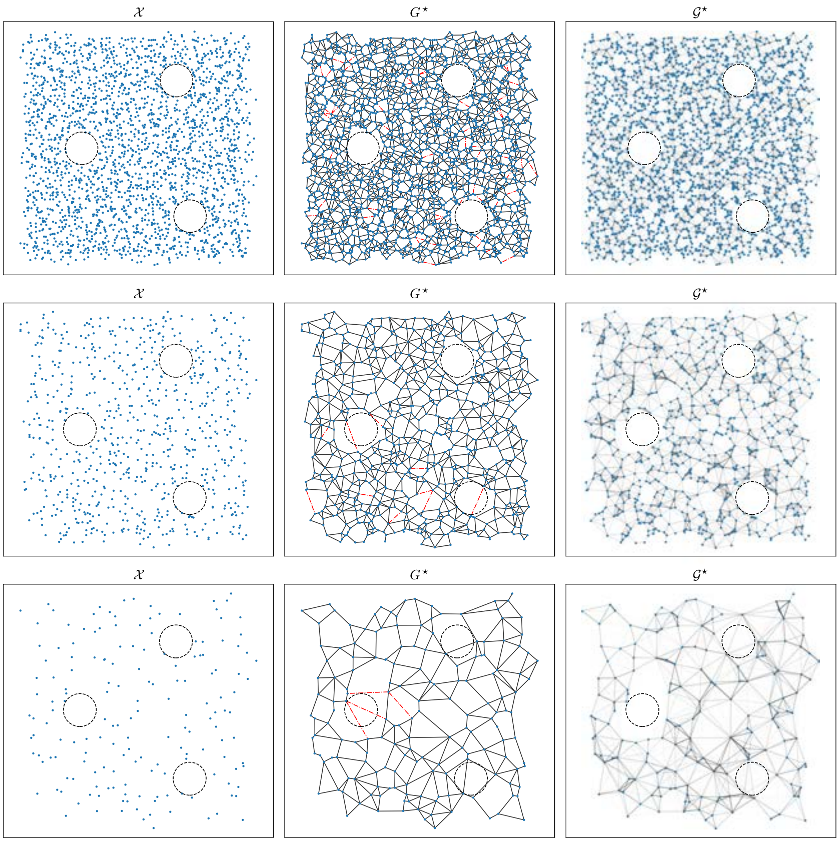


Figure 23: Sampled Swiss cheese results (cf. Figure 2). The original sampled points (true holes outlined) are shown, together with the converged graphs. In the sparse case (bottom), sampling is close to locally uniform so not all holes are correctly inferred. As sampling gets denser (top two rows), no holes are violated.

In closing this section, we return to one of our introductory examples and show, in Figure 23, the resulting graphs for the sampling Swiss cheese patterns (from Figure 2). When sampling is too sparse (bottom), there is only so much that can be inferred, and not all holes are free of edges after convergence. As sampling gets denser, however, the algorithm correctly identifies that edges across holes should be pruned (middle). When it is very dense (top), even the initial Gabriel graph is able to correctly infer the true holes.

3.6 Comparison with Other Kernel Methods. We now compare the data graphs obtained using our iterated adaptive neighborhoods (IAN)

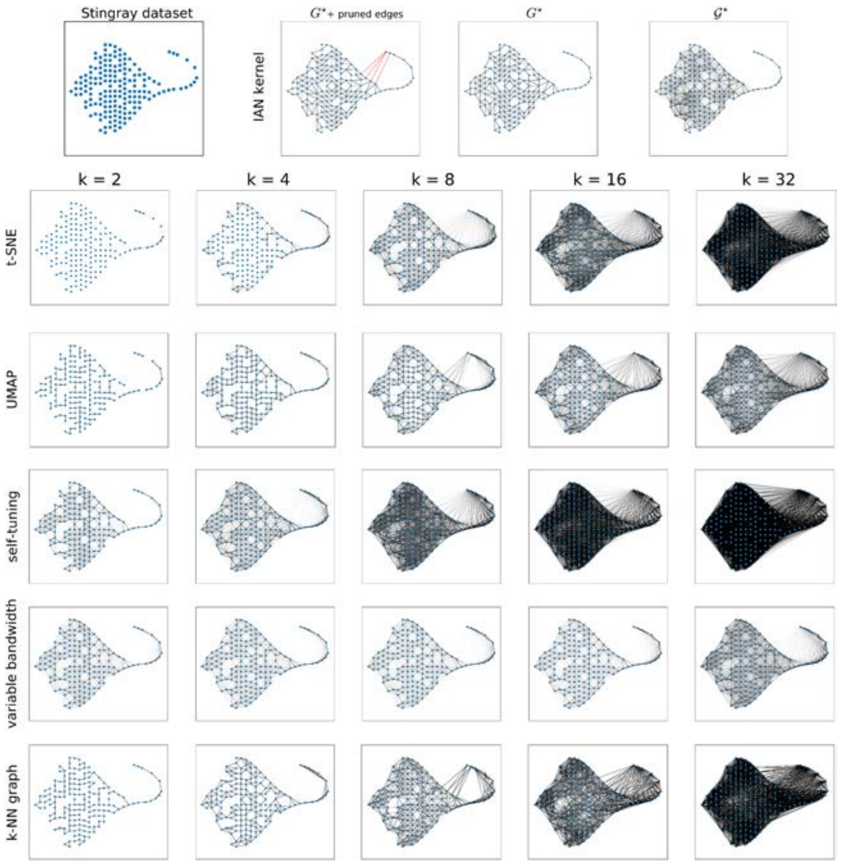


Figure 24: (Top) The stingray data set and the converged graphs, $G^{\star}$ and $\mathcal{G}^{\star}$; pruned edges are shown in red. (Bottom) Other algorithms produce qualitatively different graphs depending on the neighborhood size parameter, k. All graphs shown are weighted (using a continuous kernel) except for the k-nearest neighbors graph (bottom row). Edge weights are visualized as the intensity of the line segments (each w_{ij} is divided by the kernel value when r_{ij} equals the scale, for a fair comparison across algorithms).

with those from other popular manifold learning methods. In Figure 24, a synthetic "stingray" data set exhibits a transition of apparent dimension from 2 (body) to 1 (tail), a variation of the scenario explored in Figure 4. Points were uniformly sampled, with 20% deleted at random.

Our converged, unweighted graph, $G^{\star}$ (see the top row in Figure 24), can be compared with the traditional k-nearest neighbors graph (bottom row), used in a variety of methods, including Isomap (Tenenbaum et al., 2000).

In the latter, when $k = 2$, the tail exhibits perfect connectivity, but the body is too sparse. If $k = 4$, the body is more properly connected, but the tail becomes overly connected, and "folding" or "short circuits" start to appear. Finally, for $k \geq 8$, the connectivity is inappropriate as the tip of the tail connects directly to the body. In contrast, $G^\star$ manages to retain a minimally connected tail while covering the body almost everywhere, creating appropriate edges across many of the sampling gaps (compare with the holes that remain in the k-NN graph with $k = 4$, some of which are present even when $k = 8$).

Our weighted graph, $\mathcal{G}^\star$, can be compared against methods that use a gaussian-like kernel and where each point has an individual scale. Some of these methods were described in section 2.1: t-SNE (van der Maaten & Hinton, 2008), UMAP (McInnes et al., 2018), self-tuning (Zelnik-Manor & Perona, 2004), and variable bandwidth (Berry et al., 2015; Berry & Harlim, 2016); their resulting connectivity can be visualized in Figure 24, where edges have intensity proportional to their weight.

In Figure 25, we visualize the individual scales resultant from each of these methods. Each σ_i is represented, around each point i, as the level set corresponding to a (single-scale) kernel value of 0.75. At the top, we see that the scales found by our kernel seem to nicely conform to the space between each point and its neighbors. Especially illuminating is what happens along the tail, where scales either "expand" or "shrink" so as to minimally cover the spaces between neighboring points; this illustrates what our scale optimization achieves. Among the other methods, with few exceptions, the scales seem to cover either too much (collapsing the tail on itself) or too little (leaving holes in the body).

The weighted graphs in Figure 24 reveal the result of the interaction between these individual scales (namely, the edge weights). Our $\mathcal{G}^\star$ (top right) manages to cover almost the entire body with edges, while keeping the tail minimally connected—in fact, resembling the unweighted version in $G^\star$, and therefore respecting the original curvature and reach. Other methods, in contrast, have a hard time achieving both things with a global value for k. In t-SNE, the scales over the body are much smaller when $k \leq 4$, so its weighted graph looks too sparse; for $k \geq 8$, the scales over the tail become too large, and therefore strong edges appear, connecting it to the body. In UMAP, the scales do not grow as much with increasing k, but at $k = 4$, the body in the weighted graph is still too sparse, while for $k \geq 8$, the tail is strongly connected to the body. With the self-tuning, scales seem to grow faster with k, while with variable bandwidth, this growth is somewhat counteracted by the action of their global scale, ϵ (see equation 2.7). In fact, the graph that most resembles our own $\mathcal{G}^\star$ is the one using the variable bandwidth kernel with $k = 2$, the main difference being that the big sampling gap near the tip of the body is poorly connected, while in our case, it is slightly overly connected (due to connections in $G^\star$ crossing that gap).

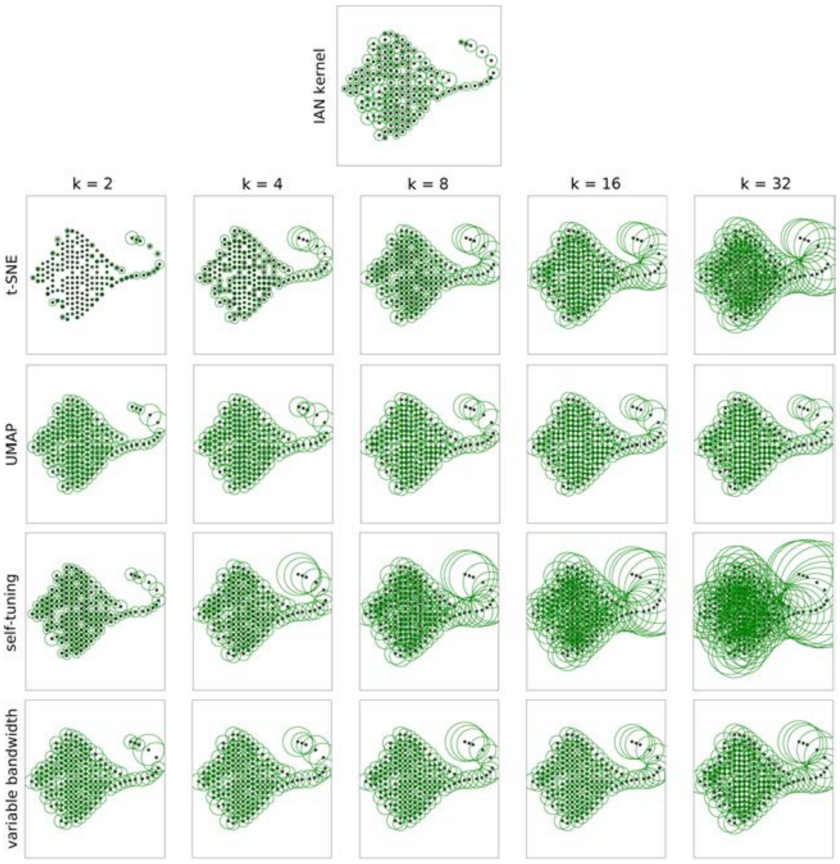


Figure 25: Individual scales obtained using our algorithm (top) compared to other methods (bottom table), as represented by their level sets for a (single-scale) kernel value of 0.75.

4 Applications

We now provide examples of application of our kernel to three different manifold learning tasks: dimensionality reduction, by means of a nonlinear embedding algorithm; geodesic estimation, which typically finds application in computational geometry, vision, and graphics; and local intrinsic dimensionality estimation.

4.1 Low-Dimensional Embeddings. Dimensionality reduction is now ubiquitous in visualization of high-dimensional data. Several methods exist (Saul, Weinberger, Sha, Ham, & Lee, 2006; Lee & Verleysen, 2007;

Goldberg & Ritov, 2009; van der Maaten et al., 2009), and most of the nonlinear methods are manifold based (Roweis & Saul, 2000; Tenenbaum et al., 2000; Roweis, Saul, & Hinton, 2001; Hinton & Roweis, 2002; Belkin & Niyogi, 2003; Donoho & Grimes, 2003; Zhang & Zha, 2004; Coifman & Lafon, 2006; Weinberger & Saul, 2006; van der Maaten & Hinton, 2008; Tang, Liu, Zhang, & Mei, 2016; McInnes et al., 2018; Moon et al., 2019). Given a collection of points in high-dimensional space sampled from a low-dimensional manifold $\mathcal{M}$, the goal is to find a good parameterization for the data in terms of intrinsic coordinates over $\mathcal{M}$, which in turn can be used to produce a low-dimensional embedding.

In surveying the literature, it is common to find a heuristic, or a range of values, suggested for choosing the neighborhood size (see section 2.1), but rarely do we see examples of the sensitivity of the results to that choice. In this section, we ran a few of the most popular methods using a wide range of values for the kernel scale parameter, k, and compared their results to those using our own kernel.

We have limited our comparison to some of the embedding methods that use a neighborhood kernel and for which pairwise information is sufficient as input (i.e., do not require positional information): diffusion maps (Coifman et al., 2005; Coifman & Lafon, 2006), Isomap (Tenenbaum et al., 2000), t-SNE (van der Maaten & Hinton, 2008), and UMAP (McInnes et al., 2018). As shown in Figures 26 to 28, results can vary qualitatively depending on the choice of k. Five values were tested for each data set, spanning a wide range of scales and different geometries. Next, we summarize each of these methods and their results.

4.1.1 Diffusion Maps with Self-Tuning Kernel. Diffusion maps are based on the spectral properties of the random walk matrix (normalized graph Laplacian) over the weighted data graph; integration over all paths in the graph makes diffusion distances, in principle, more robust to "short-circuiting" than graph geodesics. For better comparison with IAN, instead of the standard single-scale gaussian kernel, we use the self-tuning approach of Zelnik-Manor and Perona (2004) from equation 2.6. Our kernel was applied to diffusion maps by directly using $\mathcal{G}^\star$ as a similarity matrix (weighted adjacency matrix). We use the diffusion map parameters $\alpha = 1$ and $t = 1$ (cf. Coifman & Lafon, 2006).

With the stingray data set (see Figure 26), we see that the fully extended tail at $k = 2$ becomes progressively more folded and compressed as k increases. The body appears contracted at $k = 2$ but expands with larger k. Using our own $\mathcal{G}^\star$, although we obtain excellent embeddings of both body and tail (right-most column), they are represented by separate sets of coordinates (two for the body, and a third for the tail), which happens due to the change in dimensionality.

Applying self-tuning to the spiral data set (see Figure 27), only $k = 2$ and $k = 4$ were able to prevent folding. The bent plane (see Figure 28) was more

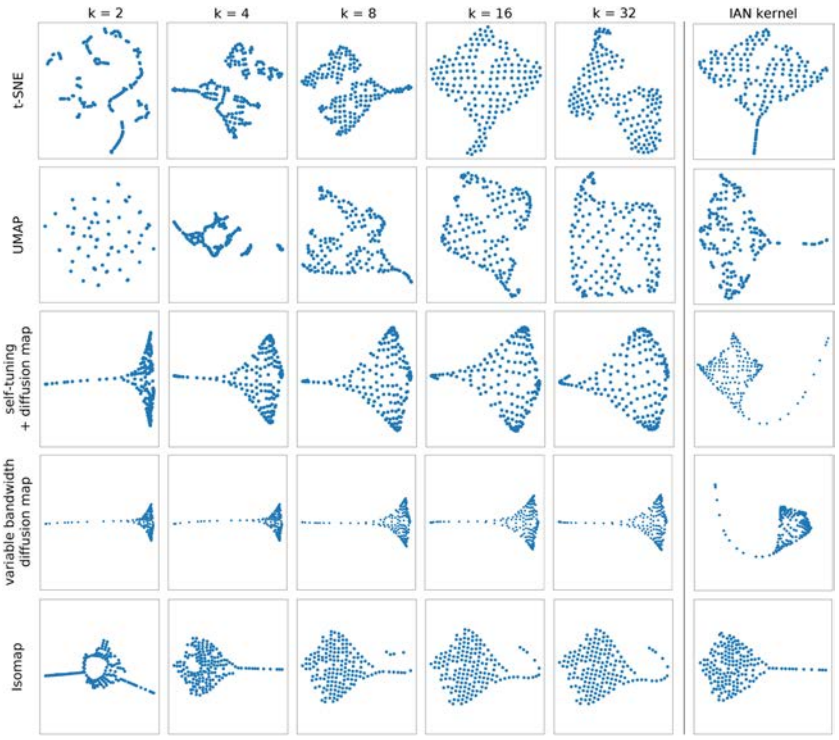


Figure 26: Running different embedding algorithms on the stingray data set (see Figure 24). Different choices of the neighborhood size, k, may produce qualitatively different results, depending on the algorithm. Running those same algorithms using the IAN kernel (right) typically gives a reasonable result. Refer to the main text for details.

tolerant, with good results for all k except 64, for which the plane remained folded. When using IAN, a good parameterization was obtained for both data sets.

4.1.2 Variable Bandwidth Diffusion Embedding. We also tested a variant of diffusion maps using the variable bandwidth kernel of Berry et al. (2015), in which a distinct type of multiscale kernel is proposed, along with a specific normalization of the weighted graph Laplacian. Because it computes an other global scale, ϵ, based on the individual scales, in order to apply our algorithm to this method we replaced the density estimates, q_ϵ (see equation 2.7), with the inverse of our optimal scales. We used $\alpha = 0$ and $\beta = -1/2$, as recommended in Berry and Harlim (2016); eigenvectors were scaled by the square root of the inverse of their respective eigenvalues

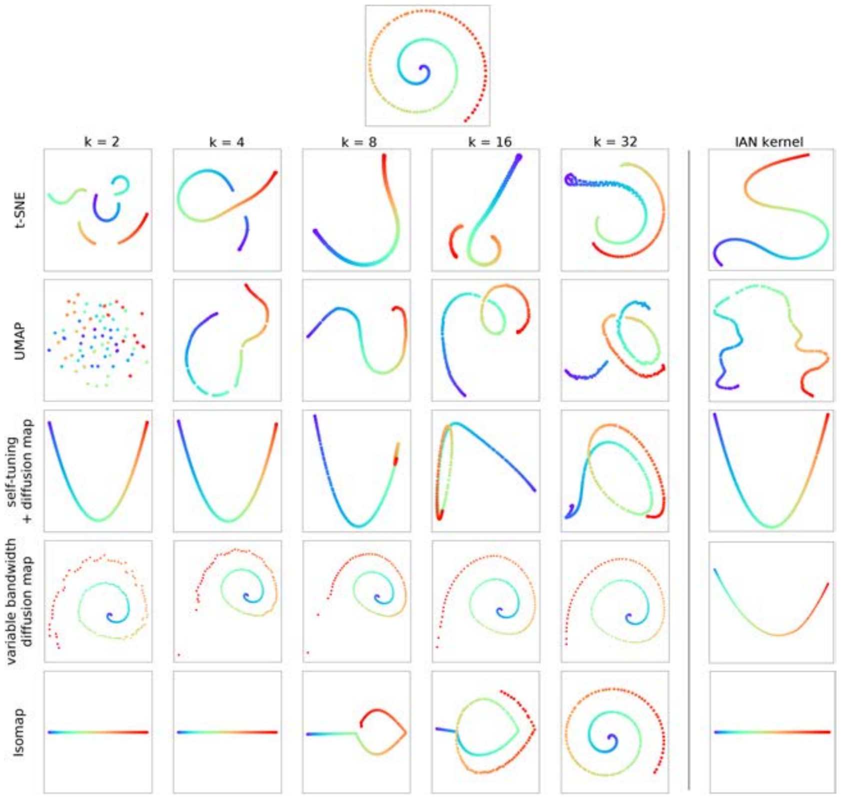


Figure 27: Running different embedding algorithms on the spiral data set (top), in which points are sampled from a unit-speed parameterized Archimedean spiral. Different choices of the neighborhood size, k, may produce qualitatively different results, depending on the algorithm. Running those same algorithms using the IAN kernel (right) typically gives a reasonable result. Refer to the main text for details.

(Saerens, Fouss, Yen, & Dupont, 2004; Noé, Banisch, & Clementi, 2016), following the implementation in Banisch, Thiede, & Trstanova (2017).

This method produced good embeddings for the stingray, especially for $k = 8$ (see Figure 26). For the spiral (see Figure 27), using $k \leq 8$ caused some points to drift apart, and although it returned basically the original curve when $k = 16$ or 32, a spectral algorithm such as this is expected to "unroll" the spiral, finding a good (1D) parameterization of it. The same happened with the bent plane (see Figure 28), which could not be embedded into two coordinates for any choice of k. Using our scales, however, the algorithm managed to find appropriate parameterizations for all three data sets.

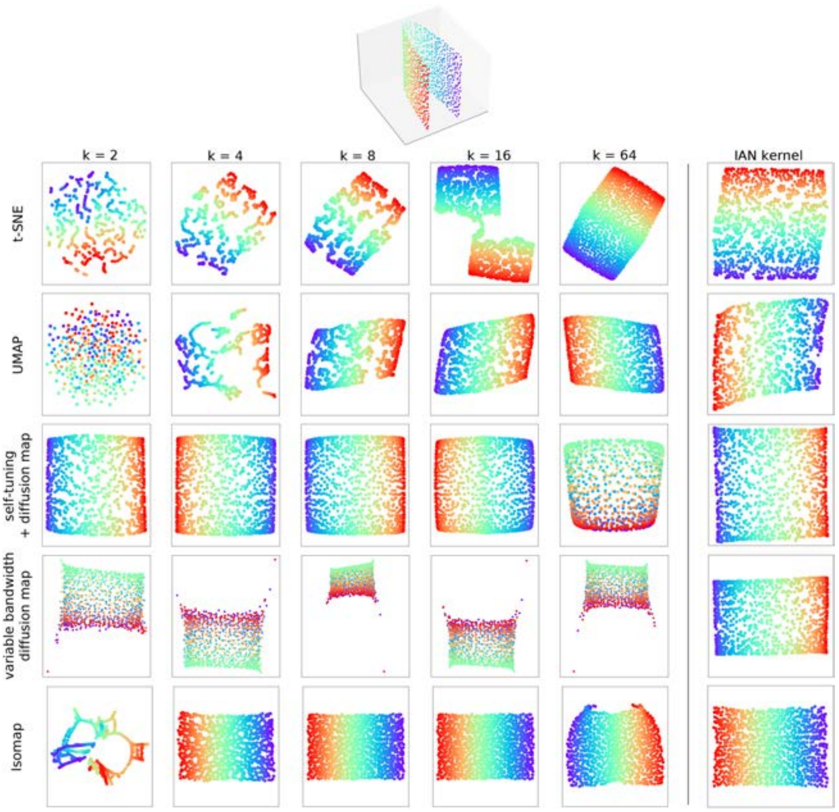


Figure 28: Running different embedding algorithms on the bent plane data set (top), generated by extending a unit-speed parameterized catenary curve into two dimensions. Different choices of the neighborhood size, k, may produce qualitatively different results, depending on the algorithm. Running those same algorithms using the IAN kernel (right) typically gives a reasonable result. Refer to the main text for details.

4.1.3 Isomap. Isomap applies classical multidimensional scaling (MDS) to geodesic distances computed as shortest paths over a k-nearest neighbors graph (see equation 2.2). Because the graph is unweighted, this method is particularly sensitive to the choice of k. Our kernel was applied to Isomap by directly replacing the k-NN graph with $G^\star$.

With the stingray (see Figure 26), Isomap produced a good embedding with $k = 4$. The result with $k = 2$ was completely wrong (an additional tail appears), and with $k = 8$, the tip of the tail was disconnected. With $k = 16$ and $k = 32$, it essentially returned the original data, without any dimensionality reduction. Our $G^\star$ improved on the result of $k = 4$ by making the points in the body more uniformly spread.

The spiral (see Figure 27) was properly embedded (1-dimensional) only when $k \leq 4$. With the bent plane (see Figure 28), good results were obtained for k between 4 and 16, but $k = 2$ produced 1-dimensional curves, and $k = 64$ did not completely unfold it. Our $G^\star$ produced the correct mapping in either case.

4.1.4 t-SNE and UMAP. t-SNE and UMAP are related methods that have gained popularity in recent years (Becht et al., 2019). Both compute similarities between data points using individual scales based on $\log_2 k$ (section 2.1) and adopt a secondary kernel for computing similarities between embedded points: t-SNE uses a Student t-distribution (Cauchy kernel), while UMAP uses an nonnormalized variant requiring a hyperparameter, `min_dist`. In t-SNE, embedding coordinates are initialized at random, while UMAP adopts the strategy of refining an initial spectral embedding. Both then optimize their embeddings by running gradient descent on an information-theoretic cost function between similarities in input space versus embedded space: t-SNE minimizes the KL-divergence; UMAP uses a variant of cross-entropy.

Alternative initializations are typically used with t-SNE (e.g., PCA) to improve results (Kobak & Berens, 2019; Linderman, Rachh, Hoskins, Steinerberger, & Kluger, 2019; Kobak & Linderman, 2021); in our experiments, for better comparison with UMAP, we used a spectral embedding initialization computed from its own symmetrized similarity matrix (see equation 2.10). The IAN kernel was applied to t-SNE by replacing the individual scales (see equation 2.8) with those in $\boldsymbol{\sigma}^\star$; with UMAP, because a different kernel function is used, we directly replaced the weighted graph (with adjacencies given by U_{ij} in equation 2.12) with $\mathcal{G}^\star$.

We executed t-SNE assigning the various k values to the perplexity parameter, leaving the remaining parameters to their defaults in the scikit-learn implementation (Pedregosa et al., 2011). We used the Barnes-Hut method (van der Maaten, 2014) for the cylinder data set and the "exact" method for all others. In UMAP, the `n_neighbors` parameter was set to k, with remaining parameters using default values (in particular, `min_dist` $=$ 0.1). Because of the stochastic nature of both algorithms (even when using a fixed initialization), different runs will produce slightly different results. Therefore, in order to avoid cherry-picking, both algorithms were executed a single time, using the same random seed.

Results for the stingray (see Figure 26) were quite analogous between the two algorithms: both produced artificial clustering for $k \leq 8$, while for $k \geq 16$, the tail began to fuse with the body. The gaps in sampling within the body were accentuated by both algorithms, even at $k = 32$, where we see a big hole in the UMAP embedding; in t-SNE, it almost breaks into two pieces (despite the large neighborhood size). This example is illustrative of how much an embedding algorithm based on attractive versus repulsive forces can end up exaggerating nonuniform sampling.

The spiral (see Figure 27) was disconnected by t-SNE for all values of k except 8. UMAP produced reasonable results for k between 4 and 8; however, for $k = 2$, a multitude of clusters was obtained, and when $k \geq 16$, the curve twisted over itself. Using our kernel (right column) produced a connected, non-self-intersecting curve. Neither algorithm was capable of returning a good arc-length parameterization of the spiral, however.

With the bent plane (see Figure 28), although both algorithms succeeded in unfolding it, t-SNE was only able to produce a fully two-dimensional plane (with no gaps) when setting $k = 32$ (not shown) or 64, while UMAP required $k \geq 16$. Both gave reasonable results using our kernel.

4.1.5 A Higher Dimensional Example. Because all of the examples above have $d \leq 2$, we also tested our kernel when applied to a higher-dimensional manifold, namely, a 5-dimensional cylinder ($\mathbb{R}^1 \times S^4$) with radius 1 and length 3, sampled uniformly at random ($N = 8403$, ambient space $\mathbb{R}^6$). Here we used a pure, connected manifold with no bottlenecks and low curvature in order to simplify interpretation.

Figure 29 shows two-dimensional embeddings obtained by applying our kernel to different embedding algorithms. Although all correctly produced an oblong, various degrees of mixing of the original color labels were observed, which can be used to qualitatively indicate the quality of the embedding schemes. A quantitative assessment was computed as the rank correlation coefficient, or Kendall's tau (Kendall, 1948; Knight, 1966) between the ranking (positional order) of each point along the main axis in the original versus embedded spaces.

Both t-SNE and UMAP produced similar or better results when using the IAN kernel (we set $k = 27$ based on the mean degree found in $G^\star$, compatible with $d = 5$; results were robust to this particular choice). Despite their current popularity (e.g., Wattenberg, Viégas, & Johnson, 2016; Arora, Hu, & Kothari, 2018; Chan, Rao, Huang, & Canny, 2018; Dimitriadis, Neto, & Kampff, 2018; Becht et al., 2019; Kobak & Berens, 2019; Fujiwara, Ida, Kanai, Kumagai, & Ueda, 2021; Kobak & Linderman, 2021; Wang, Huang, Rudin, & Shaposhnik, 2021), they produced considerably jittered outputs, however, implying that the original neighborhoods were not preserved. This appears to be caused by an attempt to reproduce the spherical shape of the cylinder's base along the main axis, so different "slices" ended up projected on top of one another. However, UMAP produced jittered results even when set to return six components (as in the original space) instead of two.

Diffusion maps using IAN resulted in little mixing except near the boundaries, so neighborhoods were better preserved. Running it with either self-tuning or variable-bandwidth kernels using $k = 27$ gave comparable results; Isomap also produced excellent results, with tau=0.98 (not shown).

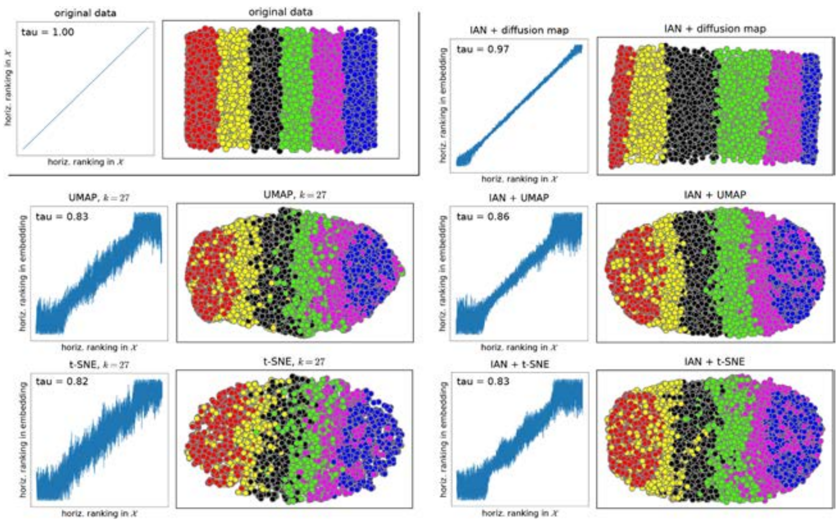


Figure 29: Performance of different embedding algorithms on a 5-dimensional cylinder ($\mathbb{R}^1 \times S^4$) sampled uniformly at random ($N = 8403$, ambient space $\mathbb{R}^6$). (Top left) Original data, $\mathcal{X}$, projected onto first two coordinates (points colored according to their position along the cylinder's long axis). Other plots show embeddings using different kernels and/or algorithms. The resulting degree of mixing of the original color labels indicates the quality of the embedding. A quantitative assessment (plots to the left of each embedding) was computed as the rank correlation coefficient, tau (see the main text), between the ranking (positional order) of each point along the horizontal axis in the original versus embedded spaces (a value closer to 1 indicates fewer exchanges in the original order). Use of the IAN kernel produced similar or better results with both t-SNE and UMAP ($k = 27$ was set based on the mean degree of $G^\star$, compatible with $d = 5$). Diffusion maps resulted in very little mixing except near the boundaries.

4.2 Geodesic Computation. Using the unweighted graph, $G^\star$, one may immediately compute graph geodesics (shortest paths using distances in ambient space as edge lengths) as an estimate of the geodesics over $\mathcal{M}$. The latter are likely to be underestimated by the former when sampling is sparse (Bernstein, De Silva, Langford, & Tenenbaum, 2000), even when the graph connectivity is correct, for example, due to curvature (cf. section 3.3.2). It seems a good idea, then, to incorporate the continuous kernel values present in its weighted counterpart, $\mathcal{G}^\star$, as a means to possibly improve geodesic estimation.

We propose to use the heat method for geodesic computation of Crane et al. (2013). It consists of solving the Poisson equation to find a function, ϕ, whose gradient follows a unit vector field, $\boldsymbol{X}$, pointing along geodesics; $\boldsymbol{X}$ can be obtained by normalizing the temperature gradient, $\nabla \boldsymbol{u}$, due to a

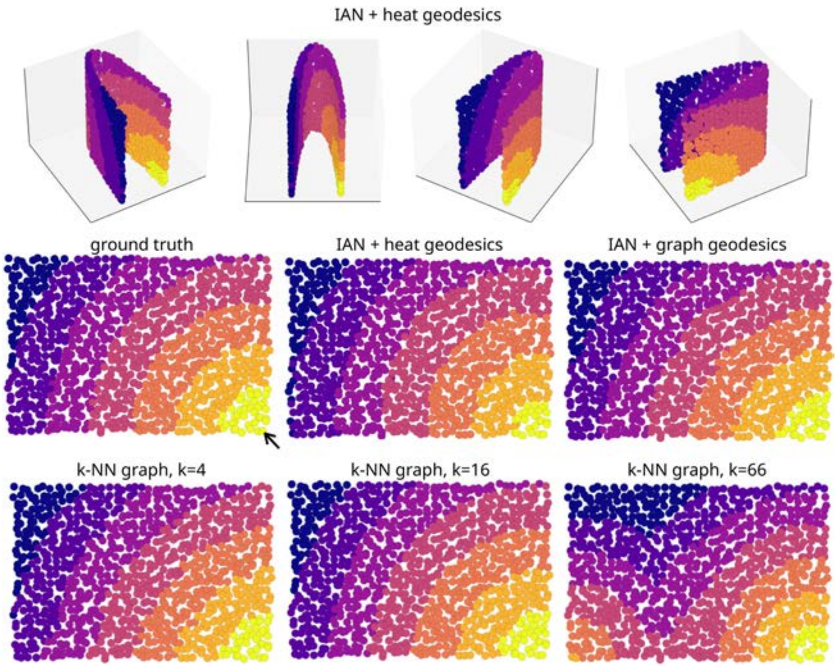


Figure 30: Geodesic estimation for the bent plane from Figure 28; yellow points are closer to the source (marked with an arrow in the ground truth plot). (Top) Different views of the data in 3D, with points colored according to the heat geodesics computed from $\mathcal{G}^\star$. (Middle) Geodesics displayed on an unbent version of the data set: heat geodesics approximate well the true geodesics over $\mathcal{M}$, and graph geodesics computed from $G^\star$ follow closely. (Bottom) Graph geodesics computed from k-NN graphs using different choices of k. Choosing $k = 16$ gives near-perfect results, but $k = 4$ shows distortions, and $k = 66$ misses completely.

diffusion process in which heat, $\boldsymbol{u}$, is allowed to diffuse for a short time. Although this method is tailored to applications where positional information and dimensionality are known (in particular, surfaces in $\mathbb{R}^3$), here we apply it to $\mathcal{G}^\star$, since discrete versions of the operators used (Laplacian, gradient, and divergence) can be readily defined on a weighted graph (see Desquesnes, Elmoataz, & Lézoray, 2013).

Despite using pairwise information only, our method produces reasonable estimates, as shown in Figures 30 and 31. To understand why, notice that IAN indirectly solves for a weighted graph for which a random walk starting at node i has a higher probability of reaching a node in its discrete neighborhood, $\mathcal{N}(i)$, than any other nonneighboring node. Given that random walks are closely related to diffusion over a graph, one should expect

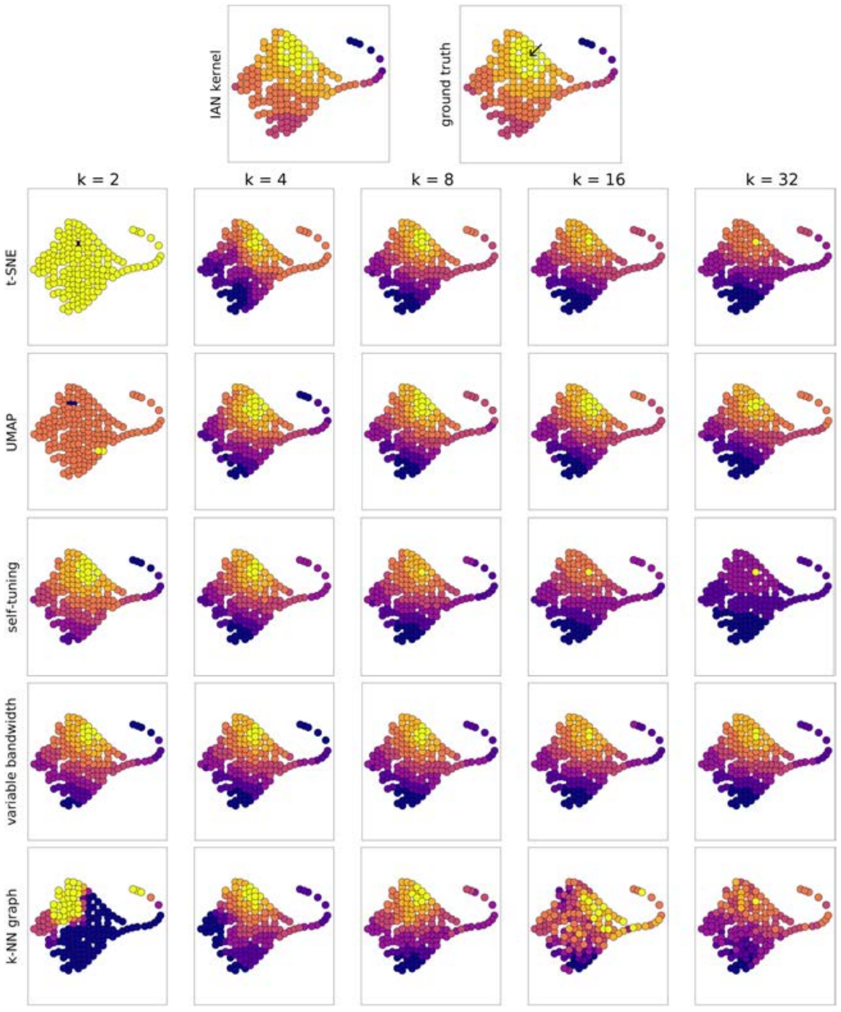


Figure 31: Geodesics estimated using the heat method applied to $\mathcal{G}^{\star}$ are close to the ground truth (top). Other kernels yield suboptimal results for most choices of k (bottom); in particular, notice how the tip of the tail is usually inferred to be closer than it should (due to its being directly connected to the body in the underlying graph; see Figure 24). Yellow points are closer to the source (marked with an arrow in the ground truth plot).

$\mathcal{G}^{\star}$ to be able to provide reasonable information about how a diffusion process propagates over $\mathcal{M}$. In other words, the Laplacian obtained from $\mathcal{G}^{\star}$ should be a good approximation of a continuous operator over $\mathcal{M}$. This is empirically confirmed by our results.

In Figure 30, heat geodesics computed from $\mathcal{G}^{\star}$ for the bent plane data set approximate well the true geodesics over $\mathcal{M}$, and graph geodesics obtained from $G^{\star}$ follow closely. Comparison with those from a naive k-NN graph illustrates that the choice of k is critical (compare with the bottom row of Figure 28).

In Figure 31, we compare the results using weighted graphs from various kernels on the stingray data set; interestingly, heat geodesics computed from $\mathcal{G}^{\star}$ hold reasonably well even when facing a continuous change in dimensionality. (The diffusion time parameter used by the heat method was optimized for each data set.)

4.3 Local Dimensionality Estimation. Intrinsic dimensionality (ID) estimation is tightly associated with dimensionality reduction tasks, especially in manifold learning, where knowledge of d can help, among others, to determine the appropriate number of embedding dimensions. Informally, ID may be seen as the minimum number of parameters required to accurately describe the data. In the context of manifold learning, it is typically equivalent to the topological dimension of $\mathcal{M}$ (e.g., a general space curve has dimensionality 1 since it requires a single parameter, arc length).

There are many different ways to estimate it (Camastra, 2003; Camastra & Staiano, 2016); global approaches are typically divided into two. The first group is based on some variant of PCA (e.g., Fukunaga & Olsen, 1971; Little et al., 2017) and uses the number of significant eigenvalues to infer dimensionality; these may be applied globally or by combining local estimates. The second group of methods, termed geometric (or fractal, when a noninteger ID is computed), exploits the geometric relationships in the data, such as neighboring distances. Some are based on estimating packing numbers (Kégl, 2002) or on distances to nearest neighbors (Trunk, 1976; Pettis, Bailey, Jain, & Dubes, 1979; Verveer & Duin, 1995; Costa, Girotra, & Hero, 2005; Facco, d'Errico, Rodriguez, & Laio, 2017; Block et al., 2021).

Among the most popular are the correlation dimension methods (Camastra & Vinciarelli, 2002; Grassberger & Procaccia, 2004; Hein & Audibert, 2005), a variant of which has been specifically applied in the context of determining an appropriate kernel width for manifold learning (see Coifman et al., 2008; Berry et al., 2015; Haghverdi et al., 2015). The dimension is computed as the slope of a log-log plot of the number of neighboring points versus neighborhood radius (see section 2.1). A recent variation is Kleindessner and Luxburg (2015); others cover the difficult case of high ID (Camastra & Vinciarelli, 2002; Rozza, Lombardi, Ceruti, Casiraghi, & Campadelli, 2012).

In our scenario, since we do not assume a pure manifold (section 3.1), we focus on local (i.e., pointwise) ID estimation approaches, namely, those in which dimension is estimated within a neighborhood around each data point (e.g., Farahmand, Szepesvári, & Audibert, 2007; He, Ding, Jiang, Li, & Hu, 2014). This notion can be formalized as the local Hausdorff dimension

(Young, 1982; Camastra & Staiano, 2016), and a global estimate is typically found by averaging over local values.

A popular approach is the maximum likelihood estimator (MLE) of Levina and Bickel (2004), which computes local dimension based on k-nearest neighbors:

$$\hat{m}_k(\boldsymbol{x}_i) = \left(\frac{1}{k-1} \sum_{j=1}^{k} \log \frac{T_k(\boldsymbol{x}_i)}{T_j(\boldsymbol{x}_i)} \right), \tag{4.1}$$

where $T_j(\boldsymbol{x}_i)$ denotes the distance between $\boldsymbol{x}_i$ and its jth nearest neighbor. We use this method in our experiments, in which we compute a final $m_k(\boldsymbol{x}_i)$ by averaging $\hat{m}_k(\boldsymbol{x}_i)$ over i's neighbors in order to reduce the variance of the local estimates (in the original, this is done over all data points).

Notice that our kernel can be readily used with this method by simply replacing the k-NN graph with $G^\star$, therefore summing over nodes in the neighborhood $\mathcal{N}(i)$ instead of over the k nearest. Additionally, we propose a correlation dimension-based method that allows for local estimates. We describe it next, then compare its results with those from the MLE method.

4.3.1 Algorithm: Neighborhood Correlation Dimension. Our proposed method is adapted from the approach from Hein and Audibert (2005; also used in Coifman et al., 2008; Haghverdi et al., 2015; Berry et al., 2015), where an estimate of correlation dimension is obtained using a general kernel. It consists of computing a curve, $Z(\sigma)$, over all pairwise kernel values (e.g., a gaussian) at different values of the scale parameter σ:

$$Z = \sum_{i=1}^{N} \sum_{j=1}^{N} \exp \frac{-\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\sigma^2}. \tag{4.2}$$

As in Coifman et al. (2008) (and analogous to equations 3.17 to 3.19), by assuming that for small values of σ the manifold $\mathcal{M}$ looks locally like its tangent space, $\mathbb{R}^d$, we have

$$Z \approx \frac{N^2(\sqrt{2\pi}\sigma)^d}{\mathrm{vol}^2(\mathcal{M})}, \tag{4.3}$$

which, after taking the logarithm, yields

$$\log Z \approx d \log \sigma + \log \frac{N^2(2\pi)^{d/2}}{\mathrm{vol}(\mathcal{M})}, \tag{4.4}$$

so the slope of $\log Z \times \log \sigma$ can be used to estimate the global dimensionality of the manifold, d. To do so, one typically looks for a region where this slope is most stable (i.e., the curve is approximately linear). Automated ways of finding the slope of such a region are by linear regression of the middle portion of the curve (Hein & Audibert, 2005) or by taking a point of maximum of $Z'(\sigma$; Berry et al., 2015; Haghverdi et al., 2015).

However, because we assume that intrinsic dimension may vary over $\mathcal{M}$, global averages cannot work in general. Moreover, nonuniform density, curvature, or multiple connected components may all create multiple peaks for $Z'(\sigma)$, so inspection of the log-log plot cannot be automated.

Therefore, we modify this approach to use individual $Z_i(\sigma)$ curves for each data point $\boldsymbol{x}_i$. To keep the summation local, points are restricted to those in the neighborhood of i in $G^\star$. Here, it is advantageous to work with an extended neighborhood (e.g., by also including neighbors-of-neighbors) due to the theoretical limit to the value of the dimension d that can be accurately estimated given a set of N points (Eckmann & Ruelle, 1992), namely, $d < 2\log_{10} N$. In fact, if N is large compared to d, even additional hops away from i may be considered. Because such extension is done by following edges in $G^\star$ (as opposed to naively expanding a ball in $\mathbb{R}^n$), we may thus obtain a larger (approximately tubular) neighborhood around $\boldsymbol{x}_i$ without ever leaving the manifold. We denote such a neighborhood $\mathcal{N}'(i)$, as opposed to the immediate neighborhood $\mathcal{N}(i)$; throughout this section, both will include the node i itself.

Our algorithm involves the following steps:

1. For each data point $\boldsymbol{x}_i$ and its extended neighborhood, $\mathcal{N}'(i)$, define Z_i as

$$Z_i(\sigma) = \sum_{j \in |\mathcal{N}'(i)|} \exp \frac{-\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\sigma^2}. \tag{4.5}$$

2. Analogous to equation 4.4, by taking the logarithm, we have that the slope of the $\log Z_i \times \log \sigma$ curve, that is,

$$Z_i'(\sigma) \overset{\text{def}}{=} \frac{\mathrm{d}\log Z_i}{\mathrm{d}\log \sigma}, \tag{4.6}$$

is an estimate of d_i, the dimension around $\boldsymbol{x}_i$, as a function of σ. Computationally, it is desirable to use the closed-form expression for accuracy:

$$Z_i'(\sigma) = \frac{\sum_{j=1}^{|\mathcal{N}'(i)|} \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2 \exp \frac{-\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\sigma^2}}{\sigma^2 \sum_{j=1}^{|\mathcal{N}'(i)|} \exp \frac{-\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\sigma^2}}. \tag{4.7}$$

3. A region of stability of Z'_i (i.e., a local maximum) is then an estimate of the dimension around $\boldsymbol{x}_i$.

A local maximum ("peak") in $Z'_i(\sigma)$ can be interpreted as follows: as a ball around $\boldsymbol{x}_i$ is expanded, the rate at which neighbors are seen has stopped increasing and must decrease with larger σ, since no additional neighbors can be found after the ball encompasses all points in $\mathcal{N}'(i)$. Underlying is the assumption that $\mathcal{N}'(i)$ is sufficiently representative of the manifold around $\boldsymbol{x}_i$. If neighbors are approximately uniformly distributed and dimensionality is constant within it, then Z'_i should remain constant over some appreciable range of σ, whence the notion of "stability."

Even though we work with a subset of $\mathcal{X}$, there may still be multiple maxima in Z'_i, for example, when the neighbors of $\boldsymbol{x}_i$ are far from uniformly distributed around it. So operationally, we use the global maximum of Z'_i, as this takes into account the information given by the majority of neighboring points. Now, because $Z_i \to 1$ as $\sigma \to 0$, and $Z_i \to N$ as $\sigma \to \infty$, the slope of $\log Z_i$ must approach 0 at both extremes; thus, the global maximum of Z'_i must also be a relative one (a "peak").

We now proceed to avoid boundary effects by *recentering neighborhoods*. The boundary, $\partial\mathcal{M}$, of a d-dimensional manifold (when present) has dimensionality $d-1$ (Lee, 2010). The correlation integral approach often fails for these; it typically returns $d/2$ for points in $\partial\mathcal{M}$, since they have roughly half the number of neighbors compared to interior points. For the same reason, it tends to also underestimate d for points near the boundary. Since we work locally over a graph, we can regularize the computation by moving the focus to a more central, nearby point (thus regularizing over sampling artifacts as well):

4. Letting $\mathcal{N}(i)$ be the set of adjacent nodes to i in $G^\star$ and including i itself, define $\bar{\imath}$ as the node $j \in \mathcal{N}(i)$ with the smallest median squared distance to all points in the extended neighborhood $\mathcal{N}'(i)$:

$$\bar{\imath} = \operatorname{argmin}_{j\in\mathcal{N}(i)} \text{median}\left\{\|\boldsymbol{x}_j - \boldsymbol{x}_l\|^2), \forall l \in \mathcal{N}'(i)\right\}. \tag{4.8}$$

Thus, $\bar{\imath}$ is, in effect, the most central node in i's immediate neighborhood.[8]

5. Use $\bar{\imath}$ as the point from which kernel values are computed for $Z_i(\sigma)$ by replacing $\boldsymbol{x}_i$ with $\boldsymbol{x}_{\bar{\imath}}$ in equation 4.5, thereby shifting the center of estimation of d_i. This assumes that the dimension does not change abruptly across neighboring points. Denote the resulting estimate by $\hat{d}_i$.

[8] Since we know $G^\star$, graph-theoretical quantities such as shortest-path betweenness centrality (Freeman, 1977; Brandes, 2001) may also be used here.

6. As with the MLE method (see section 4.3), we may obtain a smoother estimate, $\hat{d}'_i$, by averaging over immediate neighbors in $\mathcal{N}(i)$:

$$\hat{d}'_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j\in\mathcal{N}(i)} \hat{d}_j. \tag{4.9}$$

Finally, recall from section 3.5 that we also obtain a degree-based estimate, $\tilde{d}_i$, when computing volume ratios (see equation 3.24); we can use this information to further improve our results. A final estimate, $d^\star_i$, is then obtained as follows:

7. To avoid overestimating the true dimension, compute an average $\tilde{d}'_i$ over $\mathcal{N}(i)$ as

$$\tilde{d}'_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j\in\mathcal{N}(i)} \lfloor \tilde{d}_j \rfloor = \frac{1}{|\mathcal{N}(i)|} \sum_{j\in\mathcal{N}(i)} \lfloor \log_2 \deg(j) \rfloor. \tag{4.10}$$

8. Compute the optimal estimate, $d^\star_i$, as

$$d^\star_i = \max\left\{\hat{d}'_i, \tilde{d}'_i\right\}. \tag{4.11}$$

Application of this technique and comparison with other methods are given next.

4.3.2 Experimental Results. Results of applying our neighborhood correlation dimension (NCD) algorithm compared to Levina and Bickel's MLE estimator (see equation 4.1) are shown in Figures 32 to 34. For NCD, we compared results using IAN against those from k-NN graphs using various values of k (a range was chosen that included the best results for each algorithm). The IAN kernel was applied by using the discrete neighborhoods of $G^\star$, recentered using neighbors-of-neighbors at most three hops away from i (see equation 4.8).

Using IAN, we obtained near-optimal results for the stingray and the bent plane. For the 5-dimensional cylinder, the dimension was underestimated (mean 4.6). Methods based on correlation dimension are known to underestimate the true d when the sample size is not sufficiently large (Camastra & Staiano, 2016). In these cases, the method of Camastra and Vinciarelli (2002) can be applied a posteriori to improve results.

For the MLE method, using large values of k tended to improve results, but only when dimension was constant (as in the bent plane and cylinder data sets). For the stingray, however, no value of k gave correct results: small values of k increased the dimension estimates due to a bias, and large values tended to produce a uniform value throughout (thus giving better estimates only when d is constant). We found that computing the neighborhood averages using the correction of MacKay and Ghahramani (2005),

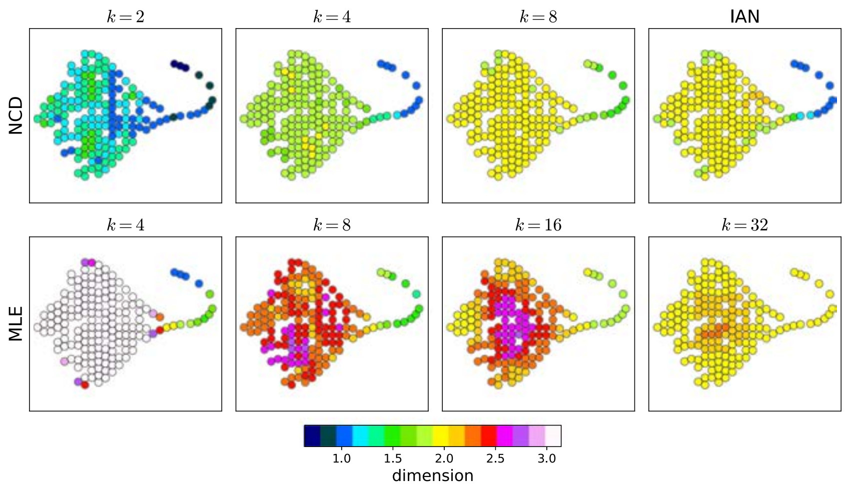


Figure 32: Estimation of local intrinsic dimension on the stingray data set. The top row shows results for our neighborhood correlation dimension (NCD) algorithm using k-NN graphs with various k and using adaptive neighborhoods from $G^\star$ (IAN). The bottom row shows results using Levina and Bickel's MLE estimator, which was sensitive to the choice of k: using a small-value grossly overestimated the dimension over the body, and a large k ignored the geometry of the tail. NCD using IAN gave the best results, estimating dimension 2 for the body and 1 for the tail, with intermediate values for the transition tail-body and the boundary.

averaging the inverse of the estimators to reduce bias when k is small, gave slightly better results. (We did not use the final smoothing procedure, which involves choosing two additional neighborhood-size parameters, k_1 and k_2.)

Finally, we confirmed these observations by testing two additional data sets with non-uniform dimensionality (see Figure 35). Again, while our algorithm achieved good results locally, there was no single value of k that allowed MLE to find appropriate local estimates everywhere.

5 Summary and Conclusion

In theory, applying the manifold assumption requires prior knowledge about the manifold: its geometry, topology, and how it was sampled. In practice, however, these manifold properties are rarely known. Instead, one typically imposes an assumption about the manifold's dimension, d, which in turn suggests that $k = 2^d$ nearest neighbors should suffice. This is how many—most!—of the data graphs underlying manifold inference and nonlinear dimensionality reduction are built. Since it is difficult to know

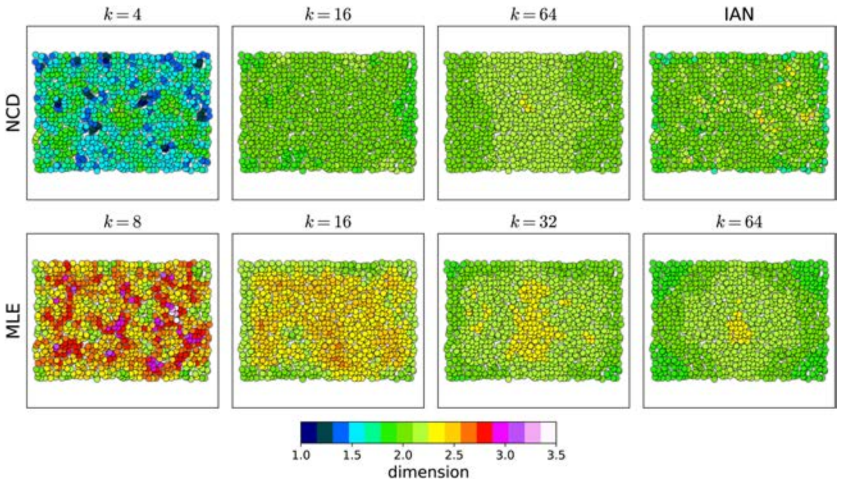


Figure 33: Estimation of local intrinsic dimension on the bent plane data set. As with the stingray (see Figure 32), results are sensitive to the choice of k, but here, a wider range of values work due to the constant dimension. For NCD, results with IAN are comparable to those using the best k-NN graph ($k = 16$). With MLE, larger k improved results (comparable to those using NCD).

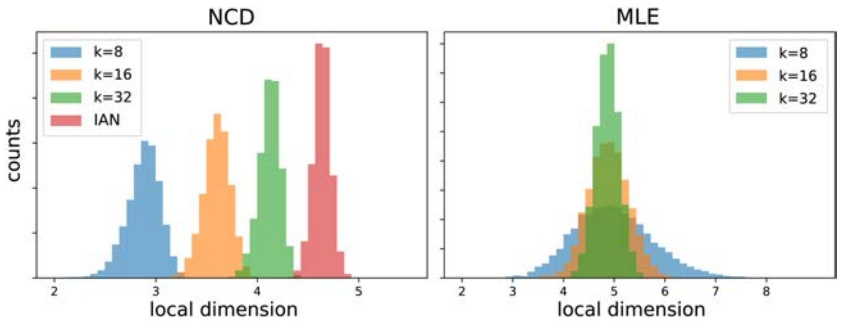


Figure 34: Estimation of local intrinsic dimension for the 5-D cylinder data set of Figure 29. With NCD, results using IAN underestimated the true dimensionality (mean 4.63), but are still better than using a k-NN graph with arbitrary k. With MLE, larger values of k gave tighter distributions centered near the correct value (mean 4.86 for $k = 32$).

whether this assumption about dimension is accurate, it is common practice to test a few values of k and choose among the results.

Apart from the subjective nature of this choice, there are more general problems. Manifolds may not have a fixed dimension, they may be curved

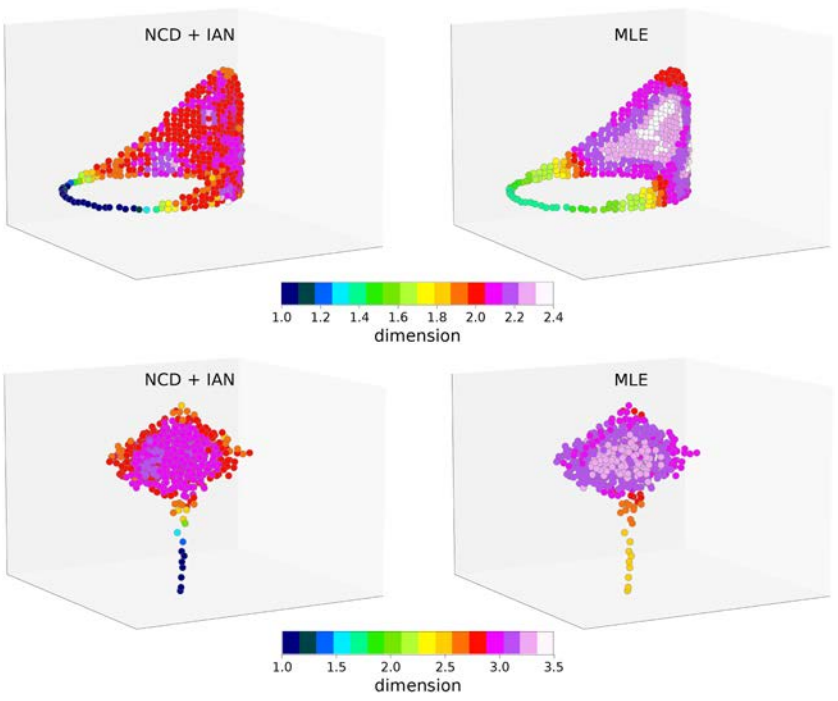


Figure 35: Dimensionality estimation for two data sets with nonuniform local d_i: a "tiara" (top row), where dimension varies smoothly from 2 to 1, and a "spinning top" (bottom row, middle cross-section shown), where dimension reduces from 3 to 1 as one moves from the bulky part toward the tip. Using the optimal k for MLE could not produce good results for the entire data set (here, $k = 32$ for both data sets). The NCD method, in contrast, was able to correctly adapt to the local geometry by taking advantage of the data graph produced by the IAN kernel.

or with boundary, and sampling may vary. The intrinsic dimension may vary across the data, and so should the number of neighbors. In such cases, finding a compromise k may be far from ideal. We suggest a different approach: that one should build the nearest-neighbor graph, and hence the graph-Laplacian approximation, in as data-driven a manner as possible, while being aware of the manifold properties.

Our algorithm of iterated adaptive neighborhoods (IAN) starts with a conservative assumption: that nearest neighbors should have no "nearer" neighbors between them. We then alternate between a discrete and a continuous view of neighborhood graphs and use a volumetric statistic to check for outliers. A linear program keeps the scales minimal while providing a global cover. This optimization is convex, so results are deterministic; other approaches, such as t-SNE and UMAP, are stochastic, so depend critically on the initialization.

Our kernel has been applied successfully to a variety of data sets, and compared against some of the most popular algorithms available. In all cases, our performance dominates. Furthermore, IAN can be incorporated directly into many embedding algorithms, including diffusion maps, Isomap, UMAP, and t-SNE, improving their results. Most of these algorithms involve several free parameters; we have none other than the robust requirement for an outlier.

Other popular embedding algorithms, such as LLE (Roweis & Saul, 2000), approximate the tangent space over a local neighborhood around each point. Although not explored here, using $G^{\star}$ to automatically provide such neighborhoods is straightforward (analogous to what was done in section 4.3 to estimate the local dimensionality). Applications to clustering need to be explored.

Our weighted graph has also been applied to geodesic estimation, achieving comparable results to those obtained from graph geodesics. In contrast, the graphs obtained from other similarity kernels produced less than optimal results.

Our unweighted graph has found application in local dimensionality estimation. Our proposed algorithm, neighborhood correlation dimension (NCD), takes advantage of the adaptive connectivity of our graph to improve results based on correlation dimension, namely, by restricting the correlation integral to an approximately tubular neighborhood around $\boldsymbol{x}_i$ in $\mathcal{M}$. As a result, we obtained accurate estimates of the local dimension in data sets where it is not uniform.

Several theoretical bounds are implied throughout this article; these need to be proved. Multiscale kernels, such as those from equations 2.6 and 2.7, are known to approximate Laplacian operators asymptotically (Ting, Huang, & Jordan, 2010; Berry & Harlim, 2016). Using our application examples as evidence, we conjecture that our version also results in good approximations.

In conclusion, understanding the interplay between manifold geometry, topology, and sampling lies at the heart of many data science applications. We have taken a first step to illustrate how discrete relates to continuous, how local estimates relate to global ones, and how uncertainties in data gathering relate to both. Applying data science in a way that leads to rigorous, scientifically appropriate conclusions must take all of these into account.

Appendix: Greedy Splitting

As an alternative to the optimization from section 3.4 (which can be expensive when the number of edges in G is very large, mainly due to large dimensionality), we have developed a greedy approach in which scales that

"C-cover" each edge e_{ij} are assigned in decreasing order of length, r_{ij} (the Euclidean distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in $\mathbb{R}^n$). We call this algorithm *greedy splitting*.

Starting with the edge e_{ij} with largest r_{ij}, set $\sigma_i = \sigma_j = Cr_{ij}$, with $C \leq 1$, thereby satisfying $\sigma_i\sigma_j \geq (Cr_{ij})^2$ with equality: we say Cr_{ij} is evenly "split" between σ_i and σ_j. Moreover, since $r_i^{\text{FN}} = r_j^{\text{FN}}$, we know the constraints $\sigma_i \leq r_i^{\text{FN}}$ and $\sigma_j \leq r_j^{\text{FN}}$ are also satisfied.

Continue with the edge e_{ij} that has the next largest length, r_{ij}. Here we are met with three possible cases in which a (re)assignment of scales is needed:

1. If neither of the nodes has been assigned a scale yet, evenly split the distance between σ_i and σ_j, as above.
2. If one of the nodes does not have a scale yet (without loss of generality, let that node be j), set σ_j' to the minimum scale that ensures $\sigma_i\sigma_j' \geq (Cr_{ij})^2$, that is, $\sigma_j' = (Cr_{ij})^2/\sigma_i$;
3. If both nodes have previously been assigned a scale but e_{ij} is not C-covered by the current values of σ_i and σ_j, then set the quotient $a = \frac{Cr_{ij}}{\sqrt{\sigma_i\sigma_j}}$ and update both scales: $\sigma_i' = a\sigma_i$ and $\sigma_j' = a\sigma_j$, thereby evenly splitting the quotient between the two nodes.

After cases 2 and 3, the updated scales might need to be "rebalanced" in order to meet the constraints $\sigma_i' \leq r_i^{\text{FN}}$ and $\sigma_j' \leq r_j^{\text{FN}}$. Without loss of generality, let $\sigma_i' > r_i^{\text{FN}}$. Then, we set $\sigma_i'' = r_i^{\text{FN}}$ and $\sigma_j'' = \sigma_j' \frac{\sigma_i'}{\sigma_i''}$. Only one of the two scales may exceed its upper bound: in case 2, this is trivially true since only the newly assigned scale may be greater than Cr_{ij}; in case 3, since both σ_i and σ_j have been previously assigned, we have $\sigma_i \leq r_i^{\text{FN}}$ and $\sigma_j \leq r_j^{\text{FN}}$, as well as $r_{ij} \leq r_i^{\text{FN}}$ and $r_{ij} \leq r_j^{\text{FN}}$, so therefore it must be the case that $r_i^{\text{FN}} r_j^{\text{FN}} \geq r_{ij}^2 = \sigma_i'\sigma_j'$. Note that as a corollary, both scales must meet their respective constraints after being rebalanced as above.

The above is repeated until all edges have been visited. By covering the largest edges first, we assign the largest, most constrained scales first, allowing for the later, less constrained scales, to be as small as possible. Because in most cases this tends to evenly split the scaled edge lengths Cr_{ij} between σ_i and σ_j, the algorithm produces reasonable (but usually suboptimal) results when compared to the linear program of section 3.4.2.

Acknowledgments

This research was supported by NIH grant EY031059, NSF CRCNS grant 1822598, and the Swartz Foundation. This project derived from problems in manifold inference involving neuroscience data. We thank G. Field and M. Stryker for motivating discussions.

References

Aamari, E., Kim, J., Chazal, F., Michel, B., Rinaldo, A., & Wasserman, L. (2019). Estimating the reach of a manifold. *Electronic Journal of Statistics*, *13*(1), 1359–1399. 10.1214/19-EJS1551

Alon, N., Ben-David, S., Cesa-Bianchi, N., & Haussler, D. (1997). Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of the ACM*, *44*(4), 615–631. 10.1145/263867.263927

Álvarez-Meza, A., Valencia-Aguirre, J., Daza-Santacoloma, G., & Castellanos-Domínguez, G. (2011). Global and local choice of the number of nearest neighbors in locally linear embedding. *Pattern Recognition Letters*, *32*(16), 2171–2177.

Amenta, N., & Bern, M. (1999). Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, *22*(4), 48–504. 10.1007/PL00009475

Amenta, N., Bern, M., & Kamvysselis, M. (1998). A new Voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 415–421).

Arora, S., Hu, W., & Kothari, P. K. (2018). An analysis of the t-SNE algorithm for data visualization. In *Conference on Learning Theory* (pp. 1455–1462).

Arora, S., & Kannan, R. (2005). Learning mixtures of separated nonspherical gaussians. *Annals of Applied Probability*, *15*(1A), 69–92. 10.1214/105051604000000512

Banisch, R., Thiede, E. H., & Trstanova, Z. (2017). pydiffmap. https://github.com/DiffusionMapsAcademics/pyDiffMap.

Becht, E., McInnes, L., Healy, J., Dutertre, C.-A., Kwok, I. W., Ng, L. G., . . . Newell, E. W. (2019). Dimensionality reduction for visualizing single-cell data using UMAP. *Nature Biotechnology*, *37*(1), 38–44. 10.1038/nbt.4314

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, *15*(6), 1373–1396. 10.1162/089976603321780317

Belkin, M., & Niyogi, P. (2004). Semi-supervised learning on Riemannian manifolds. *Machine Learning*, *56*(1), 209–239. 10.1023/B:MACH.0000033120.25363.1e

Belkin, M., Sun, J., & Wang, Y. (2008). Discrete Laplace operator on meshed surfaces. In *Proceedings of the 24th Annual Symposium on Computational Geometry* (pp. 278–287).

Belkin, M., Sun, J., & Wang, Y. (2009). Constructing Laplace operator from point clouds in R^d. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 1031–1040).

Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., & Taubin, G. (1999). The ballpivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, *5*(4), 349–359. 10.1109/2945.817351

Bernstein, M., De Silva, V., Langford, J., & Tenenbaum, J. (2000). Graph approximations to geodesics on embedded manifolds (Technical Report).

Berry, T., Giannakis, D., & Harlim, J. (2015). Nonparameteric forecasting of low-dimensional dynamical systems. *Physical Review E*, *91*(3), 032915. 10.1103/PhysRevE.91.032915

Berry, T., & Harlim, J. (2016). Variable bandwidth diffusion kernels. *Applied and Computational Harmonic Analysis*, *40*(1), 68–96. 10.1016/j.acha.2015.01.001

Block, A., Jia, Z., Polyanskiy, Y., & Rakhlin, A. (2021). Intrinsic dimension estimation. *Journal of Machine Learning Research*, *22*, 1–30.

Boissonnat, J.-D., Guibas, L. J., & Oudot, S. Y. (2009). Manifold reconstruction in arbitrary dimensions using witness complexes. *Discrete and Computational Geometry*, *42*(1), 37–70. 10.1007/s00454-009-9175-1

Boissonnat, J.-D., Lieutier, A., & Wintraecken, M. (2019). The reach, metric distortion, geodesic convexity and the variation of tangent spaces. *Journal of Applied and Computational Topology*, *3*(1), 29–58. 10.1007/s41468-019-00029-8

Boyd, S. P., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, *25*(2), 163–177. 10.1080/0022250X.2001.9990249

Bregler, C., & Omohundro, S. (1994). Nonlinear image interpolation using manifold learning. In G. Tesauro, D. Toretsky, & T. Leen (Eds.), *Advances in neural information processing systems*, *7*. Curran.

Camastra, F. (2003). Data dimensionality estimation methods: A survey. *Pattern Recognition*, *36*(12), 2945–2954. 10.1016/S0031-3203(03)00176-6

Camastra, F., & Staiano, A. (2016). Intrinsic dimension estimation: Advances and open problems. *Information Sciences*, *328*, 26–41. 10.1016/j.ins.2015.08.029

Camastra, F., & Vinciarelli, A. (2002). Estimating the intrinsic dimension of data with a fractal-based method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(10), 1404–1407. 10.1109/TPAMI.2002.1039212

Chan, D. M., Rao, R., Huang, F., & Canny, J. F. (2018). t-SNE-CUDA: GPU-accelerated t-SNE and its applications to modern data. In *Proceedings of the 30th International Symposium on Computer Architecture and High Performance Computing* (pp. 330–338).

Coifman, R. R., & Lafon, S. (2006). Diffusion maps. *Applied and Computational Harmonic Analysis*, *21*(1), 5–30. 10.1016/j.acha.2006.04.006

Coifman, R. R., Lafon, S., Lee, A. B., Maggioni, M., Nadler, B., Warner, F., & Zucker, S. W. (2005). Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. In *Proceedings of the National Academy of Sciences*, *102*(21), 7426–7431. 10.1073/pnas.0500334102

Coifman, R. R., Shkolnisky, Y., Sigworth, F. J., & Singer, A. (2008). Graph Laplacian tomography from unknown random projections. *IEEE Transactions on Image Processing*, *17*(10), 1891–1899. 10.1109/TIP.2008.2002305

Costa, J. A., Girotra, A., & Hero, A. (2005). Estimating local intrinsic dimension with k-nearest neighbor graphs. In *Proceedings of the IEEE/SP 13th Workshop on Statistical Signal Processing* (pp. 417–422).

Crane, K., Weischedel, C., & Wardetzky, M. (2013). Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics*, *32*(5), 1–11.

Dasgupta, S. (1999). Learning mixtures of gaussians. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science* (pp. 634–644).

Desquesnes, X., Elmoataz, A., & Lézoray, O. (2013). Eikonal equation adaptation on weighted graphs: Fast geometric diffusion process for local and non-local image and data processing. *Journal of Mathematical Imaging and Vision*, *46*(2), 238–257. 10.1007/s10851-012-0380-9

Dimitriadis, G., Neto, J. P., & Kampff, A. R. (2018). t-SNE visualization of large-scale neural recordings. *Neural Computation*, *30*(7), 1750–1774. 10.1162/neco_a_01097

Donoho, D. L., & Grimes, C. (2003). Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. In *Proceedings of the National Academy of Sciences*, *100*(10), 5591–5596. 10.1073/pnas.1031596100

Dyer, R., Zhang, H., & Möller, T. (2009). Gabriel meshes and Delaunay edge flips. In *Proceedings of the 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling* (pp. 295–300).

Eckmann, J.-P., & Ruelle, D. (1992). Fundamental limitations for estimating dimensions and Lyapunov exponents in dynamical systems. *Physica D: Nonlinear Phenomena*, *56*(2–3), 185–187. 10.1016/0167-2789(92)90023-G

Facco, E., d'Errico, M., Rodriguez, A., & Laio, A. (2017). Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific Reports*, *7*(1), 1–8. 10.1038/s41598-017-11873-y

Farahmand, A. M., Szepesvári, C., & Audibert, J.-Y. (2007). Manifold-adaptive dimension estimation. In *Proceedings of the 24th International Conference on Machine Learning* (pp. 265–272).

Federer, H. (1959). Curvature measures. *Transactions of the American Mathematical Society*, *93*(3), 418–491. 10.1090/S0002-9947-1959-0110078-1

Fefferman, C., Ivanov, S., Kurylev, Y., Lassas, M., & Narayanan, H. (2018). Fitting a putative manifold to noisy data. In *Proceedings of the 31st Conference on Learning Theory*, vol. *75* (pp. 688–720).

Fefferman, C., Mitter, S., & Narayanan, H. (2016). Testing the manifold hypothesis. *Journal of the American Mathematical Society*, *29*(4), 983–1049. 10.1090/jams/852

Fortune, S. (1995). Voronoi diagrams and Delaunay triangulations. *Computing in Euclidean Geometry* (pp. 225–265).

Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, *40*, 35–41. 10.2307/3033543

Fujiwara, Y., Ida, Y., Kanai, S., Kumagai, A., & Ueda, N. (2021). Fast similarity computation for t-SNE. In *Proceedings of the 2021 IEEE 37th International Conference on Data Engineering* (pp. 1691–1702).

Fukunaga, K., & Olsen, D. R. (1971). An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, *100*(2), 176–183. 10.1109/T-C.1971.223208

Gabriel, K. R., & Sokal, R. R. (1969). A new statistical approach to geographic variation analysis. *Systematic Zoology*, *18*(3), 259–278. 10.2307/2412323

Genovese, C., Perone-Pacifico, M., Verdinelli, I., & Wasserman, L. (2012). Minimax manifold estimation. *Journal of Machine Learning Research*, *13*(43), 1263–1291.

Goldberg, Y., & Ritov, Y. (2009). Local procrustes for manifold embedding: A measure of embedding quality and embedding algorithms. *Machine Learning*, *77*(1), 1–25. 10.1007/s10994-009-5107-9

Grassberger, P., & Procaccia, I. (2004). Measuring the strangeness of strange attractors. In B. E. Hunt (Ed.), *The theory of chaotic attractors* (pp. 170–189). Springer.

Haghverdi, L., Buettner, F., & Theis, F. J. (2015). Diffusion maps for high-dimensional single-cell analysis of differentiation data. *Bioinformatics*, *31*(18), 2989–2998. 10.1093/bioinformatics/btv325

Haro, G., Randall, G., & Sapiro, G. (2008). Translated Poisson mixture model for stratification learning. *International Journal of Computer Vision*, *80*(3), 358–374. 10.1007/s11263-008-0144-6

He, J., Ding, L., Jiang, L., Li, Z., & Hu, Q. (2014). Intrinsic dimensionality estimation based on manifold assumption. *Journal of Visual Communication and Image Representation*, *25*(5), 740–747. 10.1016/j.jvcir.2014.01.006

Hein, M., & Audibert, J.-Y. (2005). Intrinsic dimensionality estimation of submanifolds in R^d. In *Proceedings of the 22nd International Conference on Machine Learning* (pp. 289–296).

Hein, M., & Maier, M. (2006). Manifold denoising. In B. Schölkopf, J. Platt, & T. Hoffman (Eds.), *Advances in neural information processing systems*, *19*. Curran.

Hinton, G. E., & Roweis, S. (2002). Stochastic neighbor embedding. In S. Becker, S. Thrun, & K. Overmayer (Eds.), *Advances in neural information processing systems*, *15*. Curran.

Kégl, B. (2002). Intrinsic dimension estimation using packing numbers. In S. Becker, S. Thrun, & K. Overmayer (Eds.), *Advances in neural information processing systems*, *15*. Curran.

Keller, Y., Coifman, R. R., Lafon, S., & Zucker, S. W. (2009). Audio-visual group recognition using diffusion maps. *IEEE Transactions on Signal Processing*, *58*(1), 403–413. 10.1109/TSP.2009.2030861

Kendall, M. G. (1948). *Rank correlation methods*. Griffin.

Kleindessner, M., & Luxburg, U. (2015). Dimensionality estimation without distances. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, vol. 38 (pp. 471–479).

Knight, W. R. (1966). A computer method for calculating Kendall's tau with ungrouped data. *Journal of the American Statistical Association*, *61*(314), 436–439. 10.1080/01621459.1966.10480879

Kobak, D., & Berens, P. (2019). The art of using t-SNE for single-cell transcriptomics. *Nature Communications*, *10*(1), 1–14. 10.1038/s41467-019-13056-x

Kobak, D., & Linderman, G. C. (2021). Initialization is critical for preserving global data structure in both t-SNE and UMAP. *Nature Biotechnology*, *39*(2), 156–157. 10.1038/s41587-020-00809-z

Kouropteva, O., Okun, O., & Pietikäinen, M. (2002). Selection of the optimal parameter value for the locally linear embedding algorithm. *FSKD*, *2*, 359–363.

Lafon, S. (2004). *Diffusion maps and geometric harmonics*. PhD thesis, Yale University.

Lafon, S., Keller, Y., & Coifman, R. R. (2006). Data fusion and multicue data matching by diffusion maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(11), 1784–1797. 10.1109/TPAMI.2006.223

Lee, J. (2010). *Introduction to topological manifolds*. Springer Science & Business Media.

Lee, J. A., & Verleysen, M. (2007). *Nonlinear dimensionality reduction*, vol. 1. Springer. 10.1007/978-0-387-39351-3

Levina, E., & Bickel, P. (2004). Maximum likelihood estimation of intrinsic dimension. In L. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems*, *17*. Curran.

Lindenbaum, O., Salhov, M., Yeredor, A., & Averbuch, A. (2020). Gaussian bandwidth selection for manifold learning and classification. *Data Mining and Knowledge Discovery*, *34*(6), 1676–1712. 10.1007/s10618-020-00692-x

Linderman, G. C., Rachh, M., Hoskins, J. G., Steinerberger, S., & Kluger, Y. (2019). Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. *Nature Methods*, *16*(3), 243–245. 10.1038/s41592-018-0308-4

Little, A. V., Maggioni, M., & Rosasco, L. (2017). Multiscale geometric methods for data sets I: Multiscale SVD, noise and curvature. *Applied and Computational Harmonic Analysis*, *43*(3), 504–567. 10.1016/j.acha.2015.09.009

Lovász, L. (2010). Discrete and continuous: Two sides of the same? In N. Alon, J. Bourgain, A. V. Connes, M. Gromov, & V. Milman (Eds.), *Visions in mathematics* (pp. 359–382). Springer.

MacKay, D. J., & Ghahramani, Z. (2005). Comments on "maximum likelihood estimation of intrinsic dimension" by E. Levina and P. Bickel. Inference Group website, Cavendish Laboratory, Cambridge University.

Matula, D. W., & Sokal, R. R. (1980). Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane. *Geographical Analysis*, *12*(3), 205–222. 10.1111/j.1538-4632.1980.tb00031.x

McInnes, L., Healy, J., Saul, N., & Grossberger, L. (2018). UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, *3*(29), 861. 10.21105/joss.00861

Mekuz, N., & Tsotsos, J. K. (2006). Parameterless isomap with adaptive neighborhood selection. In *Proceedings of the Joint Pattern Recognition Symposium* (pp. 364–373).

Mishne, G., & Cohen, I. (2012). Multiscale anomaly detection using diffusion maps. *IEEE Journal of Selected Topics in Signal Processing*, *7*(1), 111–123. 10.1109/JSTSP.2012.2232279

Moon, K. R., van Dijk, D., Wang, Z., Gigante, S., Burkhardt, D. B., Chen, W. S., . . . Krishnaswamy, S. (2019). Visualizing structure and transitions in high-dimensional biological data. *Nature Biotechnology*, *37*(12), 1482–1492. 10.1038/s41587-019-0336-3

Narayanan, H., & Mitter, S. (2010). Sample complexity of testing the manifold hypothesis. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, & A. Culotta (Eds.), *Advances in neural information processing systems*, *23*. Curran.

Niyogi, P., Smale, S., & Weinberger, S. (2008). Finding the homology of submanifolds with high confidence from random samples. *Discrete and Computational Geometry*, *39*(1), 419–441. 10.1007/s00454-008-9053-2

Niyogi, P., Smale, S., & Weinberger, S. (2011). A topological view of unsupervised learning from noisy data. *SIAM Journal on Computing*, *40*(3), 646–663. 10.1137/090762932

Noé, F., Banisch, R., & Clementi, C. (2016). Commute maps: Separating slowly mixing molecular configurations for kinetic modeling. *Journal of Chemical Theory and Computation*, *12*(11), 5620–5630.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pettis, K. W., Bailey, T. A., Jain, A. K., & Dubes, R. C. (1979). An intrinsic dimensionality estimator from near-neighbor information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *1*(1), 25–37. 10.1109/TPAMI.1979.4766873

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326. 10.1126/science.290.5500.2323

Roweis, S., Saul, L., & Hinton, G. E. (2001). Global coordination of local linear models. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems*, *14*. Curran.

Rozza, A., Lombardi, G., Ceruti, C., Casiraghi, E., & Campadelli, P. (2012). Novel high intrinsic dimensionality estimators. *Machine Learning*, *89*(1), 37–65. 10.1007/s10994-012-5294-7

Saerens, M., Fouss, F., Yen, L., & Dupont, P. (2004). The principal components analysis of a graph, and its relationships to spectral clustering. In *Proceedings of the European Conference on Machine Learning* (pp. 371–383).

Samko, O., Marshall, A. D., & Rosin, P. L. (2006). Selection of the optimal parameter value for the isomap algorithm. *Pattern Recognition Letters*, *27*(9), 968–979. 10.1016/j.patrec.2005.11.017

Saul, L. K., Weinberger, K. Q., Sha, F., Ham, J., & Lee, D. D. (2006). Spectral methods for dimensionality reduction. In O. Chapelle, B. Schölkopf, & A. Zien (Eds.), *Semisupervised learning*. MIT Press.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press.

Spielman, D. (2012). Spectral graph theory. In U. Neumann & O. Schenck (Eds.), *Combinatorial scientific computing*. Taylor & Francis.

Tang, J., Liu, J., Zhang, M., & Mei, Q. (2016). Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web* (pp. 287–297).

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323. 10.1126/science.290.5500.2319

Thäle, C. (2008). 50 years sets with positive reach—a survey. *Surveys in Mathematics and Its Applications*, *3*, 123–165.

Ting, D., Huang, L., & Jordan, M. I. (2010). An analysis of the convergence of graph Laplacians. In *Proceedings of the 27th International Conference on International Conference on Machine Learning* (pp. 1079–1086).

Trunk, G. V. (1976). Statistical estimation of the intrinsic dimensionality of a noisy signal collection. *IEEE Transactions on Computers*, *100*(2), 165–171. 10.1109/TC.1976.5009231

van der Maaten, L. (2014). Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, *15*(1), 3221–3245.

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(11).

van der Maaten, L., Postma, E., & van den Herik, J. (2009). Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, *10*(66–71), 13.

Vempala, S., & Wang, G. (2004). A spectral algorithm for learning mixture models. *Journal of Computer and System Sciences*, *68*(4), 841–860. 10.1016/j.jcss.2003.11.008

Verveer, P. J., & Duin, R. P. W. (1995). An evaluation of intrinsic dimensionality estimators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *17*(1), 81–86. 10.1109/34.368147

Wan, X., Wang, W., Liu, J., & Tong, T. (2014). Estimating the sample mean and standard deviation from the sample size, median, range and/or interquartile range. *BMC Medical Research Methodology*, *14*(1), 1–13. 10.1186/1471-2288-14-135

Wang, J., Zhang, Z., & Zha, H. (2004). Adaptive manifold learning. In L. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems*, *17*. Curran.

Wang, Y., Huang, H., Rudin, C., & Shaposhnik, Y. (2021). Understanding how dimension reduction tools work: An empirical approach to deciphering t-SNE, UMAP, TriMap, and PaCMAP for data visualization. *Journal of Machine Learning Research*, *22*(201), 1–73.

Wattenberg, M., Viégas, F., & Johnson, I. (2016). How to use t-SNE effectively. *Distill*, *1*(10), e2.

Weinberger, K. Q., & Saul, L. K. (2006). Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, *70*(1), 77–90. 10.1007/s11263-005-4939-z

Young, L.-S. (1982). Dimension, entropy and Lyapunov exponents. *Ergodic Theory and Dynamical Systems*, *2*(1), 109–124. 10.1017/S0143385700009615

Zelnik-Manor, L., & Perona, P. (2004). Self-tuning spectral clustering. In *Proceedings of the 17th International Conference on Neural Information Processing System* (pp. 1601–1608).

Zhang, Z., & Zha, H. (2004). Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM Journal on Scientific Computing*, *26*(1), 313–338. 10.1137/S1064827502419154

Received June 27, 2022; accepted October 25, 2022.