Nonlinear Dimensionality Reduction with UMAP

MA4270 Computational Assignment

Hu Hanyang

2024-11-11

1 Introduction and Preliminaries

How do we tell stories from high-dimensional data in which our intuition hardly works?

This project studies UMAP [13], an unsupervised learning algorithm designed to uncover low-dimensional structures within high-dimensional data, i.e. (nonlinear) dimensionality reduction. In this section, we first describe the curse of dimensionality (CoD) for Euclidean distance-based machine learning methods, which motivates dimensionality reduction. Subsequently, we outline the notion of manifold learning. Furthermore, we discuss how to leverage insights from tangent spaces of differentiable manifolds to estimate the intrinsic dimension of the low-dimensional manifold structure.

1.1 Curse of Dimensionality in the Euclidean Space

Kernel methods in machine learning such as kernel SVM [6], Gaussian processes [19], and kernel density estimation [26] commonly employ stationary kernels: for any pair of data points $\mathbf{x}, \mathbf{y} \in \Omega$ in the domain Ω, $k(\mathbf{x}, \mathbf{y})$ only depends on the displacement $\mathbf{x} - \mathbf{y}$, independent of exact positions of $\mathbf{x}$ or $\mathbf{y}$. Hence, we can write $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$.

One of the most popular choices is the radial basis function (RBF) kernel which resembles the probability density function of a multivariate Gaussian distribution:

$$k(\mathbf{x} - \mathbf{x}') = \exp\left\{-\frac{1}{2}\sum_{i=1}^{d}\frac{(\mathbf{x}_i - \mathbf{x}_i')^2}{l_i^2}\right\}$$

where $l \in \mathbb{R}^d$ is the lengthscale hyperparameter that determines the importance of each dimension.

This stationary assumption has several benefits. Firstly, estimating hyperparameters for the stationary kernel is data-efficient because we only need data from a small part of the domain to reasonably infer its global behavior [4]. In addition, it also admits efficient kernel approximation methods such as random Fourier features [18]. However, when the kernel function value decays as the Euclidean distance between points increases (e.g. the RBF kernel), it suffers the well-known issue referred to as the curse of dimensionality (CoD): points uniformly distributed in a D-dimensional hypercube tend to get far away from each other as the dimension D increases [1]. Specifically, we could obtain the following result:

Proposition 1.1. Let X, Y be independent random variables uniformly distributed in $[0,1]^D$, then the expectation $\mathbb{E}[\|X - Y\|_2]$ is $\Theta(\sqrt{D})$.

Proof. We can bound the expectation $\mathbb{E}[\|X - Y\|_2]$ as follows:

$$(\mathbb{E}[\|X - Y\|_2])^2 \leq \mathbb{E}[\|X - Y\|_2^2] = \sum_{i=1}^{D}\mathbb{E}[(X_i - Y_i)^2] = \frac{1}{6}D$$

since X_i, Y_i are i.i.d. uniform on $[0,1]$ for $i = 1, \ldots, D$. Meantime, we have

$$\mathbb{E}[\|X - Y\|_2] \geq (\sqrt{D})^{-1}\mathbb{E}[\|X - Y\|_1] = (\sqrt{D})^{-1}\sum_{i=1}^{D}\mathbb{E}[|X_i - Y_i|] = (\sqrt{D})^{-1} \cdot \frac{1}{3}D = \frac{1}{3}\sqrt{D}$$

since $\|X - Y\|_2 \geq (\sqrt{D})^{-1}\|X - Y\|_1$. □

Remark. We note that the CoD could refer to various phenomena for large D, and we only mentioned a specific situation (the "vastness" of high-dimensional Euclidean space) as a motivating example.

Therefore, for methods based on the RBF kernel (or similar kernels, e.g. the Matérn kernel), data points tend to be considered less similar as dimensionality increases, which requires a much larger magnitude of data to convey effective reasoning, increasing the complexity of the problem. Based on this observation, a simple yet effective approach to overcome this CoD is to (implicitly or explicitly) scale the distance between data points by a factor of $(\sqrt{D})^{-1}$. An interesting example can be found in a recent paper by Hvarfner et al. [9] on high-dimensional Bayesian optimization.

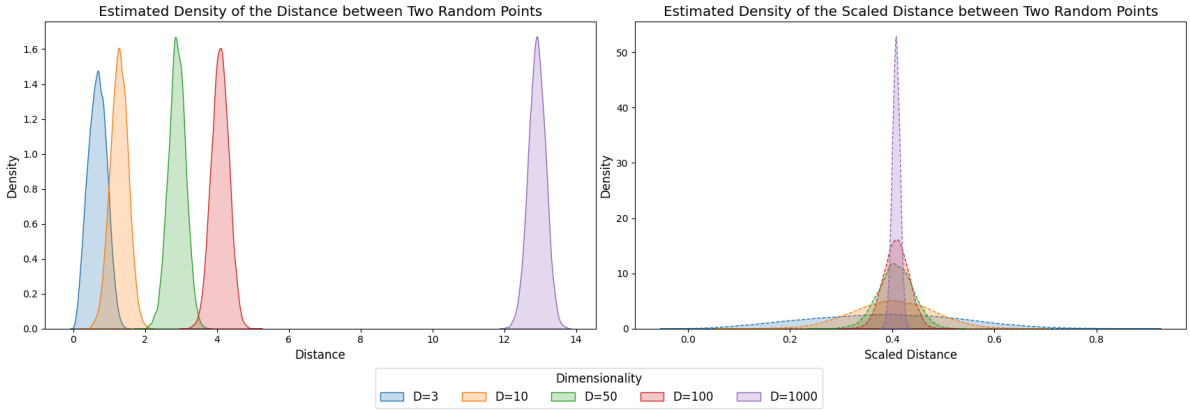


Figure 1: Estimated density (via kernel density estimation [26]) of the Euclidean distance before and after scaling by $(\sqrt{D})^{-1}$ between two points sampled from Uniform$([0,1]^D)$, for $D \in \{3, 10, 50, 100, 1000\}$. By Hoeffding's inequality, the distribution of the scaled distance concentrates around $\sqrt{1/6} \approx 0.41$. Furthermore, the variance of the unnormalized distance seems to converge to a constant.

However, the distribution of the scaled distance in high-dimensional space still differs from that in the low-dimensional case in terms of variance. From the Hoeffding's inequality, we have

$$\mathbb{P}\left[\left|\left(\frac{\|X-Y\|_2}{\sqrt{D}}\right)^2 - \frac{1}{6}\right| > \epsilon\right] = \mathbb{P}\left[\frac{1}{D}\left|\|X-Y\|_2^2 - \mathbb{E}[\|X-Y\|_2^2]\right| > \epsilon\right] \leq 2\exp\left(-2D\epsilon^2\right).$$

hence the scaled distance of points uniformly distributed in $[0,1]^D$ concentrates around $\sqrt{1/6} \approx 0.41$ for large D, as shown in Figure 1.

On the other hand, notice that $\mathbb{E}[(X_i - Y_i)^2] = 1/6 > 0$ and $\mathrm{Var}((X_i - Y_i)^2) = 7/180$ for $i = 1, \ldots, D$, hence we have the limit of the variance

$$\mathrm{Var}(\|X-Y\|_2) \to \frac{\mathrm{Var}((X_i - Y_i)^2)}{4\mathbb{E}[(X_i - Y_i)^2]} = \frac{7}{120} \qquad \text{as} \qquad D \to \infty$$

by the central limit theorem (see the discussion in [22]). This could be observed in Figure 1 as well. Therefore, scaling the distance by $(\sqrt{D})^{-1}$ results in the variance converging to zero, as stated below.

Proposition 1.2. Let X, Y be independent random variables uniformly distributed in $[0,1]^D$, then

$$\mathrm{Var}((\sqrt{D})^{-1}\|X-Y\|_2) \to 0 \qquad \text{as} \qquad D \to \infty.$$

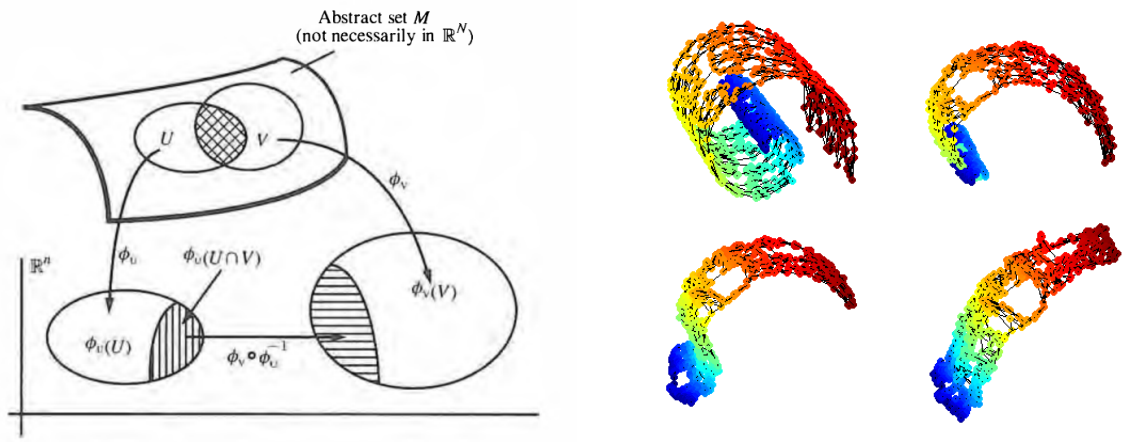
The above observations suggest that linearly scaling down distances between data points by a factor of $(\sqrt{D})^{-1}$ would cause the distances between any pair of points to become nearly identical. This is problematic for stationary kernels (or more general machine learning methods based on the Euclidean distance): when querying at a point, almost all data points are considered nearly equally important. Consequently, the Euclidean distance (with or without linear scaling) might not be suitable for modeling unstructured high-dimensional data. However, when we have prior knowledge that the data is structured, we could employ certain methods to identify the underlying low-dimensional structure and alleviate the CoD. For example, manifold learning techniques [5, 13, 20] could reduce the dimensionality when data lies on some low-dimensional manifold.

1.2 Manifolds for Dimensionality Reduction

The term "manifold" often comes up in discussions about dimensionality reduction [5, 13, 20]. The concept is essentially generalizing the idea of curves and surfaces: near each point, a manifold locally looks like a flat space, resembling many real-world scenarios. Consider a "Swiss roll", i.e. a 2-dimensional manifold (as it is a surface) that is rolled up in 3-dimensional space. Manifold learning methods work by "unrolling" it, obtaining a map from $\mathbb{R}^3$ to $\mathbb{R}^2$, hence reducing the dimension by 1. See Figure 2(b).

In general, if we assume our data are distributed on an n-dimensional manifold lying in the m-dimensional input space ($n \ll m$), then we might be able to use some methods to identify this manifold and map them into $\mathbb{R}^d$ (where $n \leq d \ll m$), hence reducing the dimensionality. It is important to note that the underlying manifold is not necessarily able to embed in $\mathbb{R}^n$; rather, we can only assert that it locally "looks like" an open set in $\mathbb{R}^n$ (see Figure 2(a)). For instance, we cannot embed a sphere in $\mathbb{R}^2$ although it is a 2-dimensional manifold. However, from results in differential geometry (e.g. the Nash embedding theorem [15]), these manifolds can be embedded in $\mathbb{R}^d$ isometrically (i.e., preserving distances) where d is only moderately larger than n.

Practically, since we are only given a finite number of data points, a common technique to identify and represent the manifold is to build a graph of which data points are the vertices [3, 5, 13, 20], then some notion of distance (based on weighted edges) between vertices reveals the geometry of the approximated manifold. In particular, we may consider the neighborhood graph which draws edges between K-nearest neighbors (KNN), capturing the topological properties. Intuitively, a larger K makes the KNN graph carry more global information, and a smaller K makes it carry more local information. Given the constructed graph, we can use force-directed graph drawing [12] to lay the vertices in a low-dimensional space whilst preserving certain useful properties of the graph.



(a) (b)

Figure 2: (a) An Illustration of a real manifold (`https://ncatlab.org/nlab/show/manifold`); (b) Identify a manifold structure (i.e. a Swiss roll) from discrete data via graphs, and gradually unroll it to obtain 2-dimensional embeddings (`https://www.numerical-tours.com/matlab/shapes_7_isomap/`).

1.3 Tangent Spaces and the Embedding Dimension

How do we find the intrinsic dimension n, and hence determine the embedding dimension d?

Unfortunately, many popular dimension reduction methods such as VAE [10] or UMAP [13] need to assume that n is known *a priori* or try out with different values (then validate to find the best option). On the other hand, spectral embedding methods such as diffusion maps [5] could use a cut-off of the large eigenvalues of the $|\mathcal{D}| \times |\mathcal{D}|$ kernel matrix to directly determine the embedding dimension, due to spectrum decay. However, the $O(|\mathcal{D}|^3)$ eigendecomposition typically does not scale with large $|\mathcal{D}|$.

We consider the fact that the tangent space at any point of an n-dimensional manifold is an n-dimensional vector space.[1] This is intuitive: the tangent line of a smooth curve (a 1-dimensional manifold) is 1-dimensional, the tangent plane of a smooth surface (a 2-dimensional manifold) is 2-dimensional, etc. Consequently, we could estimate the dimension of the manifold's tangent spaces and take it as our estimate of the intrinsic dimension n. Notice that the former should be easier due to its linearity and locality, and hence could usually be approximated by conducting PCA on nearest neighbors, for example in local tangent space alignment [27].

[1] It might be a nice (and not too difficult) exercise to prove this statement once all definitions are cleared by showing that the Jacobians of charts are linear isomorphisms. Interested readers may refer to Chapter 1 in [25].

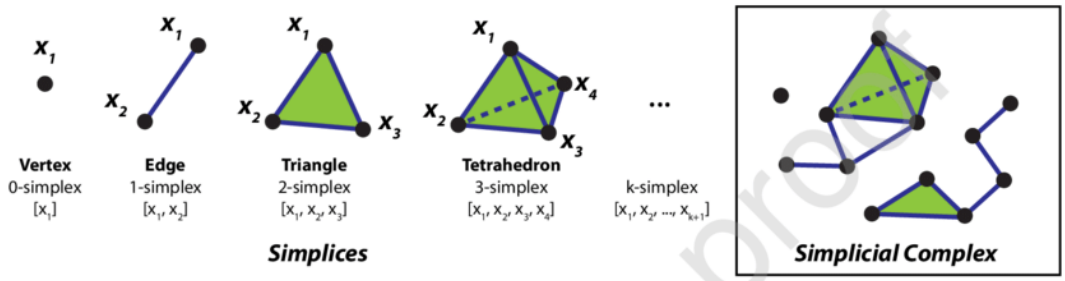


Figure 3: Illustrations of simplices (`https://www.researchgate.net/publication/339744199_Topological_portraits_of_multiscale_coordination_dynamics`). 0-D simplices are points, 1-D simplices are line segments, 2-D simplices are triangles, 3-D simplices are tetrahedrons, etc.

In particular, for an arbitrary data point $p \in \mathcal{D}$, take its K-nearest neighbors $q_1, \ldots, q_K \in \mathcal{D}$, then $q_i - p$ approximates the tangent vectors in the tangent space $T_p(M)$ of M at p, where M is the underlying manifold. Assume that K is large enough so that $\{q_i - p\}_{i=1}^K$ spans $T_p(M)$, we can form a $K \times K$ covariance matrix and automatically determine the number of principal components (by applying methods like probabilistic PCA [21] that is capable of performing model selection). On another note, we could choose multiple points $p_1, \ldots, p_L$, and summarize the results to obtain a more robust estimation.

Once the intrinsic dimension n is estimated, we might heuristically choose the embedding dimension d to be slightly larger than n (e.g. using insights from the Nash embedding theorem [15]). Choosing $d < n$ is also possible and can be particularly useful if we want to visualize the data in 2-D or 3-D space. However, it is important to note that this setting may result in failures to preserve the global structure.

2 Uniform Manifold Approximation and Projection

In this section, we summarize how the algorithm of interest - UMAP [13] - constructs graphs to approximate the underlying manifold; and how data points in the graph can be projected in low-dimensional space using neural networks. This approach is referred to as the parametric UMAP [20]. We are more interested in parametric UMAP instead of its original version because it is more flexible, allowing efficient projection of new points and reconstruction from the embedding space to the original space (similar to autoencoders [2, 10]), which could be useful in many contexts, say Bayesian optimization (where many unlabeled design points are available to identify the underlying structure, but evaluating each design point is expensive). Specifically, (parametric) UMAP makes the following assumptions, which are leveraged to identify the underlying manifold structure using graphs:

1. The data are uniformly distributed on a (Riemannian) manifold, which means that any ball of fixed volume must contain about the same number of data points. Conversely, a ball centered at a data point containing exactly the K-nearest neighbors should be of a fixed volume. This requires a principled way to warp the notion of distance by assigning weights to edges.

2. The underlying manifold is locally connected, meaning that connecting vertices that are nearest neighbors is reasonable (since there should be no isolated points).

3. The primary goal of the machine learning algorithm is to preserve the topological structure.

2.1 Approximate Manifolds via Fuzzy Simplicial Sets

UMAP approximates a manifold by fuzzy simplicial sets, which turns out to be constructing a graph by cleverly assigning weights to edges:

1. Simplices are just the generalization of triangles in arbitrary dimensions (see Figure 3 for an illustration), and a simplicial set can be understood as a collection of simplices that are put together to capture the geometry of a manifold. For computational purposes, UMAP only considers 0-D and 1-D simplices, hence the simplicial set here is just a graph as aforementioned.

2. The fuzziness refers to the uncertainty of the membership of each edge in the graph, considering the noise and outliers. For example, although all data points are adjacent to their K-nearest neighbors in the constructed graph, the grades of membership of edges incident to a vertex that is far from all its neighbors should be assigned low values, indicating that it is likely to be an outlier.

Specifically, consider the data set $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \subset \mathcal{X}$ and a metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$. Given the hyperparameter K, we construct the K-nearest neighbor graph $G = (V, E)$ where $V = \mathcal{D}$.

The weight of each edge in E should reflect its grade of membership, the specific weight assignment is outlined as follows: For each $\mathbf{x}_i \in \mathcal{D}$, let $\{\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_K}\}$ be the set of its K-nearest neighbors. Define ρ_i and σ_i such that

$$\begin{cases} \rho_i = \min\{d(\mathbf{x}_i, \mathbf{x}_{i_j}) \,|\, 1 \leq j \leq K, d(\mathbf{x}_i, d(\mathbf{x}_i, \mathbf{x}_{i_j}) > 0)\}, \\ \sum_{j=1}^{K} \exp\left(\dfrac{-\max(0, d(\mathbf{x}_i, \mathbf{x}_{i_j}) - \rho_i)}{\sigma_i}\right) = \log_2(K) \end{cases}$$

and assign the weight to $(\mathbf{x}_i, \mathbf{x}_{i_j})$ for $1 \leq j \leq K$ according to

$$w((\mathbf{x}_i, \mathbf{x}_{i_j})) = \exp\left(\frac{-\max(0, d(\mathbf{x}_i, \mathbf{x}_{i_j}) - \rho_i)}{\sigma_i}\right) \in (0, 1].$$

We can observe that the definition of ρ_i ensures $w((\mathbf{x}_i, \mathbf{x}_{i_j})) = 1$ for at least one $j \in \{1, \dots, K\}$, reflecting the assumption of local connectivity and overcoming the "vastness" of high-dimensional Euclidean space discussed in Section 1.1. The definition of σ_i normalizes the distance, reflecting the assumption that data are uniformly distributed on the manifold. The choice of RHS (i.e. $\log_2(K)$) is rather empirical.[2]

Furthermore, notice that the directed edges $(\mathbf{x}_i, \mathbf{x}_{i_j})$ and $(\mathbf{x}_{i_j}, \mathbf{x}_i)$ might be of different weights. If we interpret these weights as uncertainties in the membership of each directed edge, then the uncertainty of the corresponding undirected edge $(\mathbf{x}_i, \mathbf{x}_{i_j})$ could be

$$\tilde{w}((\mathbf{x}_i, \mathbf{x}_{i_j})) := w((\mathbf{x}_i, \mathbf{x}_{i_j})) + w((\mathbf{x}_{i_j}, \mathbf{x}_i)) - w((\mathbf{x}_i, \mathbf{x}_{i_j})) \times w((\mathbf{x}_{i_j}, \mathbf{x}_i))$$

which resembles the identity $P(A \cup B) = P(A) + P(B) - P(A \cap B)$, i.e. the probability that the undirected edge is in the graph equals the probability that at least one of the directed edge is in the graph.

Remark. This standard approach in UMAP might not preserve outliers directly (although detecting outliers in low-dimensional space might still be easier), one alternative is to consider "intersections" instead of "unions": $\tilde{w}((\mathbf{x}_i, \mathbf{x}_{i_j})) := w((\mathbf{x}_i, \mathbf{x}_{i_j})) \times w((\mathbf{x}_{i_j}, \mathbf{x}_i))$, ensuring that outliers stay disconnected.[3]

2.2 Projection using Force-Directed Graph Drawing

Given the constructed graph, UMAP uses a force-directed graph layout algorithm to embed the data points $\{\mathbf{x}_i\} \subset \mathbb{R}^m$ to $\{\mathbf{z}_i\} \subset \mathbb{R}^d$ in low-dimensional space, which is equivalent to minimizing a cross-entropy loss [13, 20]. Specifically, let the weight of an edge between two embeddings $\mathbf{z}_i, \mathbf{z}_j$ be

$$q_{ij} = \Phi(\mathbf{z}_i, \mathbf{z}_j) = (1 + a\|\mathbf{z}_i - \mathbf{z}_j\|^{2b})^{-1}$$

where a, b are hyperparameters controlling how tightly UMAP is allowed to pack points together. Let $p_{ij} = \tilde{w}((\mathbf{x}_i, \mathbf{x}_j))$ (put $p_{ij} = 0$ if the two vertices are not adjacent). The following loss is minimized

$$\mathcal{L} = \sum_{i \neq j} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) + (1 - p_{ij}) \log\left(\frac{1 - p_{ij}}{1 - q_{ij}}\right)$$

which is the sum of KL-divergences between Bernoulli(p_{ij}) and Bernoulli(q_{ij}), i.e. the probability distributions on whether the edges $(\mathbf{x}_i, \mathbf{x}_j)$ and $(\mathbf{z}_i, \mathbf{z}_j)$ should exist in the graph, respectively. Unlike t-SNE [23], it does not require normalization of q_{ij} over all edges.

Minimizing $\mathcal{L}$ via gradient descent could be interpreted as applying attractive and repulsive forces between embedding points $\mathbf{z}_i$ and $\mathbf{z}_j$, since $\log(1/q_{ij})$ decreases (resp. $\log(1/(1 - q_{ij}))$ increases) when $s = \|\mathbf{z}_i - \mathbf{z}_j\|$ decreases. Furthermore, minimizing $p_{ij} \log(p_{ij}/q_{ij})$ requires q_{ij} to be large whenever p_{ij} is large; and minimizing $(1 - p_{ij}) \log((1 - p_{ij})/(1 - q_{ij}))$ requires q_{ij} to be small when p_{ij} is small.[4] In

[2] See this discussion: `https://github.com/lmcinnes/umap/discussions/920`.

[3] See this tutorial in the official document of UMAP: `https://umap-learn.readthedocs.io/en/latest/outliers.html`.

[4] See this blog post for a more general discussion: `https://dibyaghosh.com/blog/probability/kldivergence.html`.

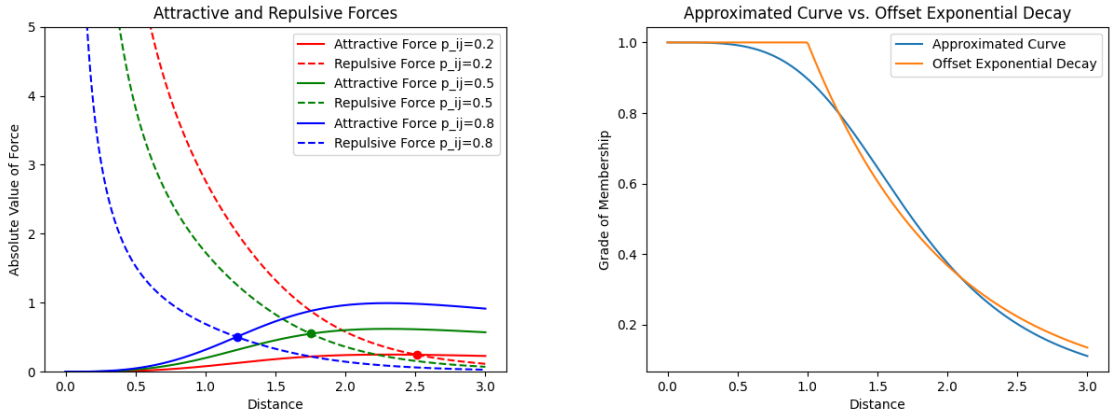
particular, by taking partial derivatives over $s = \|\mathbf{z}_i - \mathbf{z}_j\|$, we have

$$\mathcal{F}_{\text{attractive}} = \frac{\partial}{\partial s}\left[p_{ij}\log\left(\frac{p_{ij}}{q_{ij}}\right)\right] = \frac{2abp_{ij}}{(a+s^{-2b})s}$$
$$\mathcal{F}_{\text{repulsive}} = \frac{\partial}{\partial s}\left[(1-p_{ij})\log\left(\frac{1-p_{ij}}{1-q_{ij}}\right)\right] = -\frac{2b(1-p_{ij})}{(1+as^{2b})s}$$

hence the force is balanced (i.e. $|\mathcal{F}_{\text{attractive}}| = |\mathcal{F}_{\text{repulsive}}|$) at

$$s^* = \sqrt[2b]{\frac{1-p_{ij}}{ap_{ij}}}$$

which has negative derivative over $p_{ij} \in (0,1)$, i.e. as p_{ij} increases, s^* decreases, and vice versa. Hence the loss function $\mathcal{L}$ encourages positive correlations between the distances of data points in the input space and their corresponding distances in the embedding space. In addition, the attractive force $\mathcal{F}_{\text{attractive}} \to 0$ as $s \to \infty$, which may suggest that suitable initializations (e.g. eigenmaps [3, 13]) or regularizations (e.g. Pearson correlation [20]) are required to preserve the global structure or to facilitate faster and more stable convergence. A matters arising article [11] reports on empirical justifications of this statement.



(a) (b)

Figure 4: (a) Illustrations of absolute values of the attractive and repulsive forces between $\mathbf{z}_i$ and $\mathbf{z}_j$. The attractive force is weak (resp. the repulsive force is strong) when $s = \|\mathbf{z}_i - \mathbf{z}_j\|$ is small. However, the attractive forces also diminish when s is very large. In general, the higher p_{ij} is, the smaller s is required to balance the force. (b) a and b are estimated to fit the offset exponential decay Ψ using a differentiable curve Φ, where min_dist = scale = 1.

Remark. The hyperparameters a and b are not easily interpretable. However, they are estimated to fit an offset exponential decay (similar to the weights between data points in the original space defined in Section 2.1) related to two other hyperparameters - the effective minimum distance between embedding points `min_dist` and the scaling factor `spread` (corresponding to ρ_i and σ_i in Section 2.1):

$$\tilde{q}_{ij} = \Psi(\mathbf{z}_i, \mathbf{z}_j) = \exp\left(-\frac{\max(0, \|\mathbf{z}_i - \mathbf{z}_j\| - \text{min\_dist})}{\text{spread}}\right)$$

The benefit of using the formulation of Φ instead of Ψ is that it is differentiable. See Figure 4(b).

The original UMAP [13] uses stochastic gradient descent with learning rate η to minimize the loss function $\mathcal{L}$ aforementioned. In particular, the initialization is based on spectral embeddings similar to eigenmaps [3] to achieve faster and more stable convergence. In each epoch, each edge $(\mathbf{x}_i, \mathbf{x}_j) \in E$ is sampled based on its probabilistic weight p_{ij}, then the embedding $\mathbf{z}_i$ is attracted to $\mathbf{z}_j$ by updating $\mathbf{z}_i \leftarrow \mathbf{z}_i + \eta\nabla(\log(\Phi))(\mathbf{z}_i, \mathbf{z}_j)$. Furthermore, T embedding points $\{z_{k_1}, \ldots, z_{k_T}\}$ are sampled randomly as negative samples [14], and $\mathbf{z}_i$ is repulsed from them by updating $\mathbf{z}_i \leftarrow \mathbf{z}_i + \eta\sum_{t=1}^{T}\nabla(\log(1-\Phi))(\mathbf{z}_i, \mathbf{z}_{k_t})$.[5]

[5]Notice that the numerators p_{ij} and $1-p_{ij}$ of the fractions appeared in $\mathcal{L}$ does not matter for the optimization problem.

Clearly, the sum of attractive forces for edges sampled with probability p_{ij} is an unbiased estimator of the gradient of $\sum_{i \neq j} p_{ij} \log(p_{ij}/q_{ij})$. One question is, how does negative sampling approximate the gradient of $\sum_{i \neq j}(1-p_{ij}) \log((1-p_{ij})/(1-q_{ij}))$? Intuitively, for large $|\mathcal{D}|$, the majority of the data point $\mathbf{x}_j \in \mathcal{D}$ are not adjacent to $\mathbf{x}_i$ due to the sparsity of the constructed graph, i.e. $p_{ij} = 0$. Therefore, randomly sampling T points and assuming them to be negative samples *might* make a reasonable estimate. However, we should notice that it may change the weight of repulsive forces drastically, depending on the negative sampling rate T and the data set size $|\mathcal{D}|$.[6]

In the case of parametric UMAP [20], the same loss function is optimized over the neural network parameters. However, the edges are sampled in batches of fixed size B, with the probability of sampling each edge being proportional to its weight. Negative sampling is performed by having multiple copies of the batch of edges (i.e. paired vertices) and randomly shuffling the vertices on one side to form negative pairs. To capture the global structure, parametric UMAP uses the *sample* Pearson correlation:

$$C_{\text{Pearson}} = \frac{\sum_{i=1}^{B \times B}(d_X^{(i)} - \overline{d_X})(d_Z^{(i)} - \overline{d_Z})}{\sqrt{\sum_{i=1}^{B \times B}(d_X^{(i)} - \overline{d_X})^2}\sqrt{\sum_{i=1}^{B \times B}(d_Z^{(i)} - \overline{d_Z})^2}}$$

where d_X, d_Z are flattened $B \times B$ matrices[7] representing pairwise distances within batches of original and embedded data points, respectively. The training objective is to minimize the regularized loss function $\mathcal{L} - \alpha \cdot C_{\text{Pearson}}$ where $\alpha \geq 0$ is a hyperparameter. Intuitively, the distances between embedded data points should be more positively correlated with the corresponding distances in the original space.

On another note, parametric UMAP may converge slower than the original UMAP [20]. Intuitively, learning an embedding map is a more complex task than directly adjusting embeddings obtained from eigenmaps. We can think of it as a trade-off between flexibility and convergence speed.

3 Implementation and Applications

In this section, we discuss the details of our implementation of parametric UMAP using the PyTorch library [16], then apply it to some synthetic and real-world datasets.

To compute the nearest neighbor graph, we follow the same practice in the implementation of the original UMAP [13] by using the PyNNDescent library which implements the NN-Descent algorithm [8], returning an approximate result of KNN with a reported empirical complexity of $O(|\mathcal{D}|^{1.14})$.

To compute σ_i for each data point $\mathbf{x}_i \in \mathcal{D}$ (see Section 2.1), we use the binary search algorithm, which may require us to find an upper bound. Consider

$$\tilde{\sigma}_i := \max\{d(\mathbf{x}_i, \mathbf{x}_{i_j}) \,|\, 1 \leq j \leq k, d(\mathbf{x}_i, d(\mathbf{x}_i, \mathbf{x}_{i_j}) > 0)\} - \rho_i$$

so that

$$\exp\left(\frac{-\max(0, d(\mathbf{x}_i, \mathbf{x}_{i_j}) - \rho_i)}{\tilde{\sigma}_i}\right) \geq \exp(-1)$$

for $j = 1, \ldots, K$. We have $K \exp(-1) \geq \log_2(K)$ for $K \geq 9$, suitable for the default choice of $K = 15$. For smaller K, we may multiply a constant factor $C > 1$ on $\tilde{\sigma}_i$ so that $K \exp(-1/C) \geq \log_2(K)$. This upper bound is very loose, but it should be sufficient in practice.

To represent the constructed graph more efficiently, we use the compressed row sparse matrix implemented in the SciPy library [24] since there are only K non-zero entries in each row of the KNN graph's weighted adjacency matrix A. Moreover, the weighted adjacency matrix B for the fuzzy simplicial set (see Section 2.1) is computed by

$$B = A + A^\top - A \circ A^\top$$

where $\circ$ is the Hadamard (or element-wise) product. Notice that B is symmetric.

We use the edges (corresponding to the non-zero entries in the sparse symmetric matrix B) to form a PyTorch-style dataset. For each iteration, we use the `torch.utils.data.WeightedRandomSampler` to sample a batch of edges based on their weights. The negative samples are obtained by concatenating multiple copies of the batch and shuffling vertices on one side randomly.

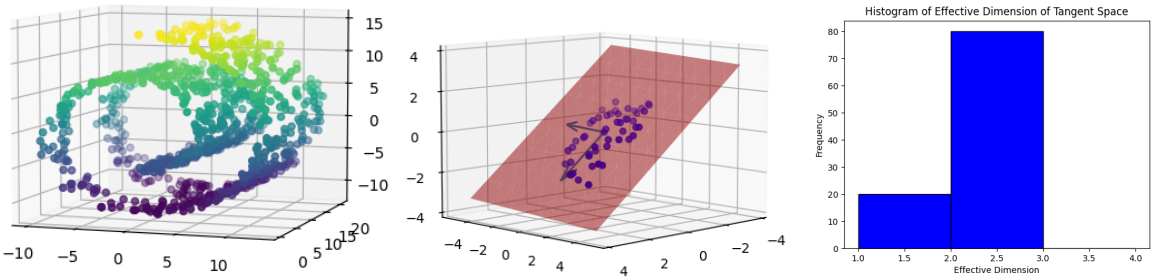
[6] An interesting discussion is presented in [7]; however, it is beyond the scope of this project.

[7] We choose the first vertex in each edge (as pair of vertices) so that there are B vertices to compute pairwise distances.

We run probabilistic PCA [21] implemented in the Scikit-Learn library [17] on the nearest neighbors of data points to estimate the dimension of the tangent space, and hence the intrinsic dimension of the underlying manifold (see Section 1.3).

3.1 Swiss Roll

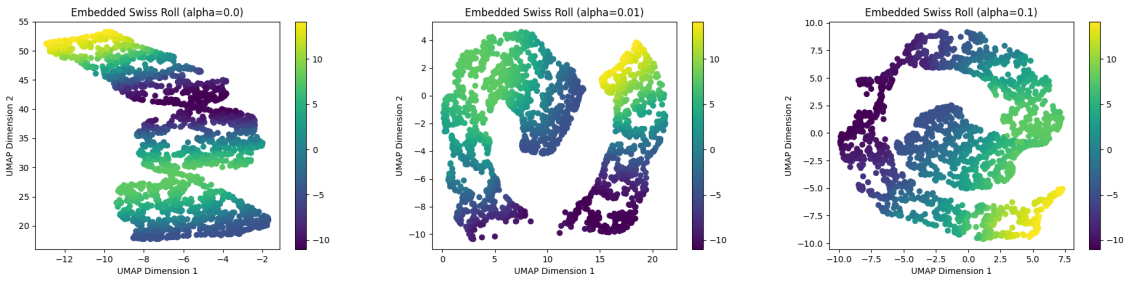
Figure 5 demonstrates the application of probabilistic PCA to estimate the intrinsic dimension. As expected, the tangent spaces at 80% of the randomly selected points are estimated to be 2-dimensional.



(a) (b) (c)

Figure 5: (a) The synthetic dataset for a Swiss roll (with a hole). (b) The tangent space approximated from a neighborhood of a data point. (c) The histogram of the dimension estimated by probabilistic PCA over 100 randomly chosen data points, the Swiss roll is identified as a 2-dimensional manifold.

For simplicity, we generate 2000 synthetic data points on a Swiss roll without holes and noises (contrary to Figure 5(a)) and run 1000 epochs training to see how our implementation of the parametric UMAP unfolds the Swiss roll with different weights put on the Pearson correlation term C_{Pearson}. As shown in Figure 6, setting higher weight values α for C_{Pearson} allows the algorithm to better preserve the global structure in the embedding. For smaller weight α, the algorithm tends to focus more on unfolding the Swiss roll to a rectangle.



(a) (b) (c)

Figure 6: The embedded Swiss roll dataset with $\alpha \in \{0, 0.01, 0.1\}$ where α is the weight put on C_{pearson}.

3.2 Three Concentric Semicircles

In this example, we demonstrate the advantage of (parametric) UMAP as a preprocessing step for other learning algorithms, such as clustering, even without dimensionality reduction. We generate a synthetic dataset of three concentric semicircles that violates the assumptions of the K-means algorithm (i.e., each cluster could be represented by a centroid) and train the model on two of the semicircles (red and blue) only. Notably, we observe that it generalizes to the green semicircle to some extent (though not perfectly), highlighting the flexibility of parametric UMAP as it can directly embed new data points without modifying the model parameters (i.e., without retraining). See Figure 7.

Why does applying (parametric) UMAP to this dataset improve K-means clustering? From a topological point of view, the three semicircles form distinct (path) connected components, meaning that there exists no path between points from different semicircles, hence only repulsive forces act between these points. When these components are repelled far apart, it is easier for the K-means algorithm to identify distinct groups and assign points to the correct cluster.

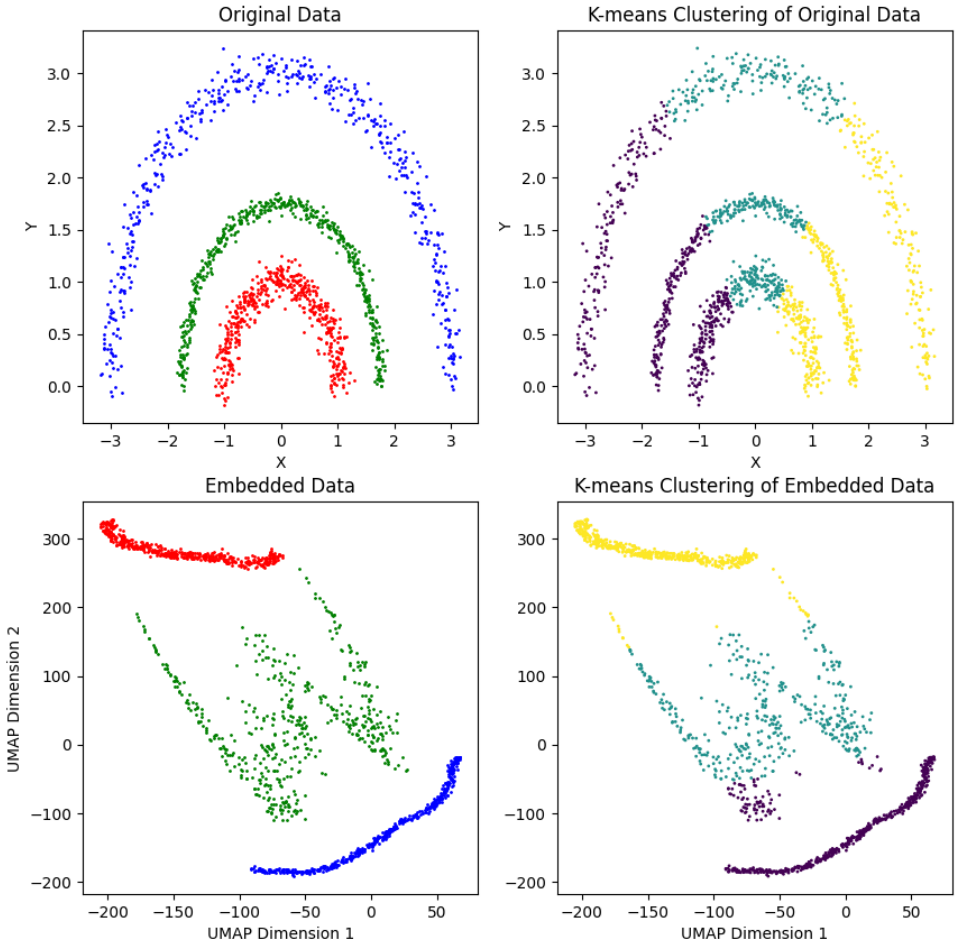


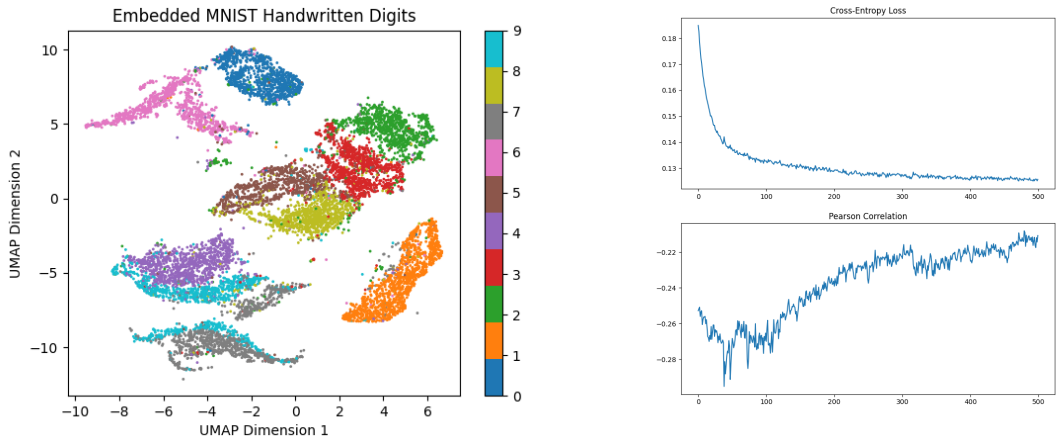
Figure 7: Clustering of three semicircles using K-means, before and after parametric UMAP embedding (the green data points are not included in the training set). UMAP improves cluster separation by repulsing the semicircles away in the embedding space (as they form different connected components), resulting in a more meaningful clustering result. **Note.** In this experiment, we set a larger negative sampling rate T (increased from the default $T = 5$ to $T = 200$) to encourage stronger repulsion.

3.3 MNIST Handwritten Digits

For the MNIST handwritten digits dataset, we train the parametric UMAP model to embed in 2-dimensional space for visualization. To reduce the computation requirement, we only select 10000 images from the dataset. The result after training for 500 epochs (with zero weight on the Pearson correlation term) is shown in Figure 8.

We would also like to apply the tangent space approach to estimate the intrinsic dimension of this real-world dataset. However, computing probabilistic PCA requires fitting PCAs for all possible dimensions, which is computationally intractable in very high-dimensional settings. Alternatively, we could randomly sample 10 dimensions from the entire list of $784 = 28 \times 28$ possible dimensions and estimate the effective

dimension as the one with the highest score.[8] Moreover, we may further enhance the efficiency by assuming that the intrinsic dimension should be less than 300, so that fewer neighboring points (e.g. $500 > 300$) need to be found. The result is computed in nearly half an hour, see Figure 9.



(a) (b)

Figure 8: Embeddings and loss curves (the cross-entropy loss and regularization term $-C_{\text{Pearson}}$, notice the negative sign) after training for 500 epochs. Although our model has not fully converged yet (indicated by the loss curve), it has captured some meaningful patterns. We hypothesize that the category groups $\{4, 7, 9\}$ and $\{2, 3, 5, 8\}$ are not fully separated in this stage because their members share more visual similarities in handwriting. In addition, the *negative* Pearson correlation term $-C_{\text{Pearson}}$ starts to increase after initially decreasing, as the convergence of the cross-entropy loss begins to slow down.

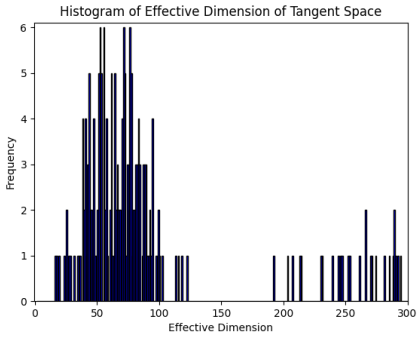


Figure 9: The histogram of effective dimensions of tangent spaces estimated on 200 randomly chosen points, assuming that the intrinsic dimension is at most 300.

4 Conclusion

This report explored the UMAP algorithm as a nonlinear dimensionality reduction method, with a particular focus on its parametric implementation. Through experiments on synthetic and real-world datasets, we demonstrated UMAP's effectiveness in revealing meaningful structures, validating its potential for visualization and data preprocessing in high-dimensional settings. Furthermore, we use insights from the tangent spaces of differentiable manifolds to estimate the intrinsic dimension of the underlying manifold structure of the data, which might help in heuristically determining the embedding dimension.

[8] This approach is heuristic and lacks formal justification, but it may still exhibit some patterns as shown in Figure 9.

References

[1] R. S. Anderssen, R. P. Brent, D. J. Daley, and P. A. P. Moran. Concerning $\int_0^1 \cdots \int_0^1 (x_1^2 + \cdots + x_k^2)^{1/2} dx_1 \cdots dx_k$ and a taylor series method. *SIAM Journal on Applied Mathematics*, 30(1):22–30, 1976.

[2] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2021.

[3] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[4] Felix Berkenkamp, Angela P. Schoellig, and Andreas Krause. No-regret bayesian optimization with unknown hyperparameters. *Journal of Machine Learning Research*, 20(50):1–24, 2019.

[5] Ronald R. Coifman and Stéphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, July 2006.

[6] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[7] Sebastian Damrich and Fred A Hamprecht. On umap's true loss function. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 5798–5809. Curran Associates, Inc., 2021.

[8] Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, page 577–586, New York, NY, USA, 2011. Association for Computing Machinery.

[9] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla bayesian optimization performs great in high dimensions, 2024.

[10] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[11] Dmitry Kobak and George C. Linderman. Initialization is critical for preserving global data structure in both t-sne and umap. *Nature Biotechnology*, 39(2):156–157, February 2021.

[12] Stephen G. Kobourov. Spring embedders and force directed graph drawing algorithms, 2012.

[13] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013.

[15] John Nash. C^1 isometric imbeddings. *The Annals of Mathematics*, 60(3):383, November 1954.

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[17] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python, 2018.

[18] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[19] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005.

[20] Tim Sainburg, Leland McInnes, and Timothy Q Gentner. Parametric umap embeddings for representation and semi-supervised learning, 2021.

[21] Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61(3):611–622, September 1999.

[22] S. Catterall (`https://stats.stackexchange.com/users/86998/s-catterall`). Central limit theorem for square roots of sums of i.i.d. random variables. Cross Validated. URL: `https://stats.stackexchange.com/q/242165` (version: 2016-10-25).

[23] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[24] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, Aditya Vijaykumar, Alessandro Pietro Bardelli, Alex Rothberg, Andreas Hilboll, Andreas Kloeckner, Anthony Scopatz, Antony Lee, Ariel Rokem, C. Nathan Woods, Chad Fulton, Charles Masson, Christian Häggström, Clark Fitzgerald, David A. Nicholson, David R. Hagen, Dmitrii V. Pasechnik, Emanuele Olivetti, Eric Martin, Eric Wieser, Fabrice Silva, Felix Lenders, Florian Wilhelm, G. Young, Gavin A. Price, Gert-Ludwig Ingold, Gregory E. Allen, Gregory R. Lee, Hervé Audren, Irvin Probst, Jörg P. Dietrich, Jacob Silterra, James T Webber, Janko Slavič, Joel Nothman, Johannes Buchner, Johannes Kulick, Johannes L. Schönberger, José Vinícius de Miranda Cardoso, Joscha Reimer, Joseph Harrington, Juan Luis Cano Rodríguez, Juan Nunez-Iglesias, Justin Kuczynski, Kevin Tritz, Martin Thoma, Matthew Newville, Matthias Kümmerer, Maximilian Bolingbroke, Michael Tartre, Mikhail Pak, Nathaniel J. Smith, Nikolai Nowaczyk, Nikolay Shebanov, Oleksandr Pavlyk, Per A. Brodtkorb, Perry Lee, Robert T. McGibbon, Roman Feldbauer, Sam Lewis, Sam Tygier, Scott Sievert, Sebastiano Vigna, Stefan Peterson, Surhud More, Tadeusz Pudlik, Takuya Oshima, Thomas J. Pingel, Thomas P. Robitaille, Thomas Spura, Thouis R. Jones, Tim Cera, Tim Leslie, Tiziano Zito, Tom Krauss, Utkarsh Upadhyay, Yaroslav O. Halchenko, and Yoshiki Vázquez-Baeza. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, February 2020.

[25] Raymond O. Wells. *Differential Analysis on Complex Manifolds*. Springer New York, December 2007.

[26] Weglarczyk, Stanisław. Kernel density estimation and its application. *ITM Web Conf.*, 23:00037, 2018.

[27] Zhenyue Zhang and Hongyuan Zha. Principal manifolds and nonlinear dimension reduction via local tangent space alignment, 2002.