Uniform Manifold Approximation and Projection (UMAP) and its Variants: Tutorial and Survey

Benyamin Ghojogh   BGHOJOGH@UWATERLOO.CA

Department of Electrical and Computer Engineering,
Machine Learning Laboratory, University of Waterloo, Waterloo, ON, Canada

Ali Ghodsi   ALI.GHODSI@UWATERLOO.CA

Department of Statistics and Actuarial Science & David R. Cheriton School of Computer Science,
Data Analytics Laboratory, University of Waterloo, Waterloo, ON, Canada

Fakhri Karray   KARRAY@UWATERLOO.CA

Department of Electrical and Computer Engineering,
Centre for Pattern Analysis and Machine Intelligence, University of Waterloo, Waterloo, ON, Canada

Mark Crowley   MCROWLEY@UWATERLOO.CA

Department of Electrical and Computer Engineering,
Machine Learning Laboratory, University of Waterloo, Waterloo, ON, Canada

arXiv:2109.02508v1 [cs.HC] 25 Aug 2021

Abstract

Uniform Manifold Approximation and Projection (UMAP) is one of the state-of-the-art methods for dimensionality reduction and data visualization. This is a tutorial and survey paper on UMAP and its variants. We start with UMAP algorithm where we explain probabilities of neighborhood in the input and embedding spaces, optimization of cost function, training algorithm, derivation of gradients, and supervised and semi-supervised embedding by UMAP. Then, we introduce the theory behind UMAP by algebraic topology and category theory. Then, we introduce UMAP as a neighbor embedding method and compare it with t-SNE and LargeVis algorithms. We discuss negative sampling and repulsive forces in UMAP's cost function. DensMAP is then explained for density-preserving embedding. We then introduce parametric UMAP for embedding by deep learning and progressive UMAP for streaming and out-of-sample data embedding.

1. Introduction

Dimensionality reduction and manifold learning can be used for feature extraction and data visualization. Dimensionality reduction methods can be divided into three categories which are spectral methods, probabilistic methods, and neural network-based methods (Ghojogh, 2021). Some of the probabilistic methods are neighbor embedding methods where the probabilities of neighborhoods are used in which attractive and repulsive forces are utilized for neighbor and non-neighbor points, respectively. Some of the well-known neighbor embedding methods are Student's t-distributed Stochastic Neighbor Embedding (t-SNE) (van der Maaten & Hinton, 2008; Ghojogh et al., 2020a), LargeVis (Tang et al., 2016), and Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018). Interestingly, both t-SNE and UMAP are state-of-the-art methods for data visualization. The reason behind the name of UMAP is that it assumes and approximates that the data points are uniformly distributed on an underlying manifold. The term "projection" in the name of algorithm is because it sort of projects, or embeds, data onto a subspace for dimensionality reduction.

The theory behind UMAP is based on algebraic topology and category theory. The main idea of UMAP is constructing fuzzy topological representations for both high-dimensional data and low-dimensional embedding of data and changing the embedding so that its fuzzy topological representation becomes similar to that of the high-dimensional data. UMAP has been widely used for DNA

and single-cell data visulization and feature extraction (Becht et al., 2019; Dorrity et al., 2020). It is noteworthy that t-SNE has also been used for single-cell applications (Kobak & Berens, 2019). Some other applications of UMAP are visualizing deep features (Carter et al., 2019), art (Vermeulen et al., 2021), and visualizing BERT features in natural language processing (Coenen et al., 2019; Levine et al., 2019). This paper is a tutorial and survey paper on UMAP and its variants.

The remainder of this paper is organized as follows. We explain the details of UMAP algorithm in Section 2 and explain the category theory and algebraic topology behind it in Section 3. Explaining UMAP as neighbor embedding and comparison with t-SNE and LargeVis are explained in Section 4. Then, we discuss the repulsive forces, negative sampling, and effective cost function of UMAP in Section 5. DensMAP, parametric UMAP, and progressive UMAP are introduced in Sections 6, 7, and 8, respectively. Finally, Section 9 concludes the paper.

Required Background for the Reader

This paper assumes that the reader has general knowledge of calculus, probability, linear algebra, and basics of optimization. The required background on algebraic topology and category theory are explained in the paper.

2. UMAP

2.1. Data Graph in the Input Space

Consider a training dataset $X = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n}$ where n is the sample size and d is the dimensionality. We construct a k-Nearest Neighbors (kNN) graph for this dataset. It has been empirically observed that UMAP requires fewer number of neighbors than t-SNE (Sainburg et al., 2020). Its default value is $k = 15$. We denote the j-th neighbor of x_i by $x_{i,j}$. Let $\mathcal{N}_i$ denote the set of neighbor points for the point x_i, i.e., $\mathcal{N}_i := \{x_{i,1}, \ldots, x_{i,k}\}$. We treat neighbor relationship between points stochastically. Inspired by SNE (Hinton & Roweis, 2003) and t-SNE (van der Maaten & Hinton, 2008; Ghojogh et al., 2020a), we use the Gaussian or Radial Basis Function (RBF) kernel for the measure of similarity between points in the input space. The probability that a point x_i has the point x_j as its neighbor can be computed by the similarity of these points:

$$p_{j|i} := \begin{cases} \exp\left(-\frac{\|x_i - x_j\|_2 - \rho_i}{\sigma_i}\right) & \text{if } x_j \in \mathcal{N}_i \\ 0 & \text{Otherwise,} \end{cases} \quad (1)$$

where $\|.\|_2$ denotes the ℓ_2 norm. The ρ_i is the distance from x_i to its nearest neighbor:

$$\rho_i := \min\{\|x_i - x_{i,j}\|_2 \mid 1 \leq j \leq k\}. \quad (2)$$

The σ_i is the scale parameter which is calculated such that the total similarity of point x_i to its k nearest neighbors is normalized. By binary search, we find σ_i to satisfy:

$$\sum_{j=1}^{k} \exp\left(-\frac{\|x_i - x_{i,j}\|_2 - \rho_i}{\sigma_i}\right) = \log_2(k). \quad (3)$$

Note that t-SNE (van der Maaten & Hinton, 2008) has a similar search for its scale using entropy as perplexity. These searches make the neighborhoods of various points behave similarly because the scale for a point in a dense region of dataset becomes small while the scale of a point in a sparse region of data becomes large. In other words, UMAP and t-SNE both assume (or approximate) that points are uniformly distributed on an underlying low-dimensional manifold. This approximation is also included in the name of UMAP.

Eq. (1) is a directional similarity measure. To have a symmetric measure with respect to i and j, we symmetrize it as:

$$\mathbb{R} \ni p_{ij} := p_{j|i} + p_{i|j} - p_{j|i}\, p_{i|j}. \quad (4)$$

This is a symmetric measure of similarity between points x_i and x_j in the input space.

2.2. Data Graph in the Embedding Space

Let the embeddings of points be $Y = [y_1, \ldots, y_n] \in \mathbb{R}^{p \times n}$ where p is the dimensionality of embedding space and is smaller than input dimensionality, i.e., $p \ll d$. Note that y_i is the embedding corresponding to x_i. In the embedding space, the probability that a point y_i has the point y_j as its neighbor can be computed by the similarity of these points:

$$\mathbb{R} \ni q_{ij} := (1 + a\, \|y_i - y_j\|_2^{2b})^{-1}, \quad (5)$$

which is symmetric with respect to i and j. The variables $a > 0$ and $b > 0$ are hyperparameters determined by the user. By default, we have $a \approx 1.929$ and $b \approx 0.7915$ (McInnes et al., 2018), although it has been empirically seen that setting $a = b = 1$ does not qualitatively impact the results (Böhm et al., 2020).

2.3. Optimization Cost Function

UMAP aims to make the data graph in the low-dimensional embedding space similar to the data graph in the high-dimensional embedding space. In other words, we treat Eqs. (4) and (5) as probability distributions and minimize the difference of these distributions to make similarities of points in the embedding space similar to similarities of points in the input space. A measure for the difference of these similarities of graphs is the fuzzy cross-entropy defined as:

$$c_1 := \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} \left(p_{ij} \ln\left(\frac{p_{ij}}{q_{ij}}\right) + (1 - p_{ij}) \ln\left(\frac{1 - p_{ij}}{1 - q_{ij}}\right) \right), \quad (6)$$

where $\ln(.)$ is the natural logarithm. The definition of this cross-entropy is in the field of fuzzy category theory which we will explain in Section 3 (see Eq. (17)).

The first term in Eq. (6) is the *attractive force* which attracts the embeddings of neighbor points toward each other. This term should only appear when $p_{ij} \neq 0$ which means either x_j is a neighbor of x_i, or x_i is a neighbor of x_j, or both (see Eq. (4)). The second term in Eq. (6) is the *repulsive force* which repulses the embeddings of non-neighbor points away from each other. As the number of all permutations of non-neighbor points is very large, computation of the second term is non-tractable in big data. Inspired by Word2Vec (Mikolov et al., 2013) and LargeVis (Tang et al., 2016), UMAP uses *negative sampling* where, for every point x_i, m points are sampled randomly from the training dataset and treat them as non-negative (negative) points for x_i. As the dataset is usually large, i.e. $m \ll n$, the sampled points will be actual negative points with high probability. The summation over the second term in Eq. (6) is computed only over these negative samples rather than *all* negative points.

UMAP changes the data graph in the embedding space to make it similar to the data graph in the input space. Eq. (6) is the cost function which is minimized in UMAP where the optimization variables are $\{y_i\}_{i=1}^n$:

$$\min_{\{\boldsymbol{y}_i\}_{i=1}^n} c_1 := \min_{\{\boldsymbol{y}_i\}_{i=1}^n} \sum_{i=1}^n \sum_{j=1, j \neq i}^n \Big(p_{ij} \ln(p_{ij}) - p_{ij} \ln(q_{ij})$$
$$+ (1 - p_{ij}) \ln(1 - p_{ij}) - (1 - p_{ij}) \ln(1 - q_{ij}) \Big)$$
$$= \min_{\{\boldsymbol{y}_i\}_{i=1}^n} - \sum_{i=1}^n \sum_{j=1, j \neq i}^n \Big(p_{ij} \ln(q_{ij}) + (1 - p_{ij}) \ln(1 - q_{ij}) \Big)$$

According to Eqs. (1), (4), and (5), in contrast to q_{ij}, the p_{ij} is independent of the optimization variables $\{y_i\}_{i=1}^n$. Hence, we can drop the constant terms to revise the cost function:

$$c_2 := - \sum_{i=1}^n \sum_{j=1, j \neq i}^n \Big(p_{ij} \ln(q_{ij}) + (1 - p_{ij}) \ln(1 - q_{ij}) \Big), \tag{7}$$

which should be minimized. Two important terms in this cost function are:

$$c_{i,j}^a := -\ln(q_{ij}), \tag{8}$$
$$c_{i,j}^r := -\ln(1 - q_{ij}), \tag{9}$$

and we can write:

$$c_2 := \sum_{i=1}^n \sum_{j=1, j \neq i}^n \Big(p_{ij}\, c_{i,j}^a + (1 - p_{ij})\, c_{i,j}^r \Big) \tag{10}$$

$$\overset{(a)}{=} 2 \sum_{i=1}^n \sum_{j=i+1}^n \Big(p_{ij}\, c_{i,j}^a + (1 - p_{ij})\, c_{i,j}^r \Big), \tag{11}$$

Algorithm 1

1   **Input**: Training data $\{x_i\}_{i=1}^n$
2   Construct kNN graph
3   Initialize $\{y_i\}_{i=1}^n$ by Laplacian eigenmap
4   Calculate p_{ij} and q_{ij} for $\forall i, j \in \{1, \ldots, n\}$ by Eqs. (4) and (5)
5   $\eta \leftarrow 1, \nu \leftarrow 0$
6   **while** *not converged* **do**
7     $\nu \leftarrow \nu + 1$ // epoch index
8   **for** *i from* 1 *to* n **do**
9   **for** *j from* 1 *to* n **do**
10        $u \sim U(0, 1)$
11   **if** $u \leq p_{ij}$ **then**
12          $y_i \leftarrow y_i - \eta \frac{\partial c_{i,j}^a}{\partial y_i}$
13          $y_j \leftarrow y_j - \eta \frac{\partial c_{i,j}^a}{\partial y_j}$
14   **for** *m iterations* **do**
15            $l \sim U\{1, \ldots, n\}$
16            $y_i \leftarrow y_i - \eta \frac{\partial c_{i,l}^r}{\partial y_i}$
17   // The next line does not exist in original UMAP:
18            $y_l \leftarrow y_l - \eta \frac{\partial c_{i,l}^r}{\partial y_l}$
19     $\eta \leftarrow 1 - \frac{\nu}{\nu_{\max}}$
20   **Return** $\{y_i\}_{i=1}^n$

Algorithm 1: UMAP algorithm

where (a) is because $p_{ij} = p_{ji}$, $c_{i,j}^a = c_{j,i}^a$, and $c_{i,j}^r = c_{j,i}^r$ are symmetric.

The Eqs. (8) and (9) are the attractive and repulsive forces in Eq. (7), respectively. The attractive force attracts the neighbor points toward each other in the embedding space while the repulsive force pushes the non-neighbor points (i.e., points with low probability of being neighbors) away from each other in the embedding space. According to Eq. (10), $c_{i,j}^a$ and $c_{i,j}^r$ occur with probability p_{ij} and $(1 - p_{ij})$, respectively. For every point, we call it the *anchor* point and we call its neighbor and non-neighbor points, with large and small p_{ij}, as the *positive* and *negative* points, respectively.

2.4. The Training Algorithm of UMAP

The procedure of optimization in UMAP is shown in Algorithm 1. As this algorithm shows, a kNN graph is constructed from the training data $\{x_i\}_{i=1}^n$. UMAP uses Laplacian eigenmap (Belkin & Niyogi, 2001; Ghojogh et al., 2021), also called spectral embedding, for initializing the embeddings of points denoted by $\{y_i\}_{i=1}^n$. Using Eqs. (4) and (5), p_{ij} and q_{ij} are calculated for all points. Stochastic Gradient Descent (SGD) is used for optimization where optimization is performed iteratively. In

every iteration (epoch), we iterate over points twice with indices i and j where the i-th point is called the *anchor*. For every pair of points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, we update their embeddings $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with probability p_{ij} (recall Eq. (7)). If p_{ij} is large, it means that the points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are probably neighbors (in this case, the j-th point is called the *positive* point) and their embeddings are highly likely to be updated to become close in the embedding space based on the attractive force. For implementing it, we can sample a uniform value from the continuous uniform distribution $U(0,1)$ and if that is less than p_{ij}, we update the embeddings. We update the embeddings $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ by gradients $\partial c_{i,j}^a/\partial \boldsymbol{y}_i$ and $\partial c_{i,j}^a/\partial \boldsymbol{y}_j$, respectively, where η is the learning rate.

For repulsive forces, we use negative sampling as was explained before. If m denotes the size of negative sample, we sample m indices from the discrete uniform distribution $U\{1,\ldots,n\}$. These are the indices of points which are considered as *negative* samples $\{\boldsymbol{y}_l\}$ where $|\{\boldsymbol{y}_l\}| = m$. As the size of dataset is usually large enough to satisfy $n \gg m$, these negative points are probably valid because many of the points are non-neighbors of the considered anchor. In negative sampling, the original UMAP (McInnes et al., 2018) updates only the embedding of anchor $\boldsymbol{y}_i$ by gradient of the repulsive force $\partial c_{i,j}^a/\partial \boldsymbol{y}_i$. One can additionally update the embedding of negative point $\boldsymbol{y}_l$ by gradient of the repulsive force $\partial c_{i,j}^a/\partial \boldsymbol{y}_l$ (Damrich & Hamprecht, 2021). The mentioned gradients are computed in the following lemmas.

Lemma 1 ((McInnes et al., 2018)). *The gradients of attractive and repulsive cost functions in UMAP are:*

$$\frac{\partial c_{i,j}^a}{\partial \boldsymbol{y}_i} = \frac{2ab\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2(b-1)}}{(1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})}(\boldsymbol{y}_i - \boldsymbol{y}_j), \qquad (12)$$

$$\frac{\partial c_{i,j}^r}{\partial \boldsymbol{y}_i} = \frac{-2b}{(\varepsilon + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2)(1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})}(\boldsymbol{y}_i - \boldsymbol{y}_j), \qquad (13)$$

where ε is a small positive number, e.g. $\varepsilon = 0.001$, for stability to prevent division by zero when $\boldsymbol{y}_i \approx \boldsymbol{y}_j$. Likewise, we have:

$$\frac{\partial c_{i,j}^a}{\partial \boldsymbol{y}_j} = \frac{2ab\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2(b-1)}}{(1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})}(\boldsymbol{y}_j - \boldsymbol{y}_i),$$

$$\frac{\partial c_{i,j}^r}{\partial \boldsymbol{y}_j} = \frac{-2b}{(\varepsilon + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2)(1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})}(\boldsymbol{y}_j - \boldsymbol{y}_i).$$

Proof. For the first equation, we have:

$$\frac{\partial c_{i,j}^a}{\partial \boldsymbol{y}_i} = \frac{\partial c_{i,j}^a}{\partial q_{ij}} \times \frac{\partial q_{ij}}{\partial \boldsymbol{y}_i} = \frac{-1}{q_{ij}} \times \Big(\frac{-1}{(1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})^2}$$

$$\times 2ab(\boldsymbol{y}_i - \boldsymbol{y}_j) \times \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2(b-1)} \Big)$$

$$\overset{(5)}{=} \frac{2ab\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2(b-1)}}{(1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})}(\boldsymbol{y}_i - \boldsymbol{y}_j). \qquad (14)$$

For the second equation, we have:

$$\frac{\partial c_{i,j}^r}{\partial \boldsymbol{y}_i} =$$

$$\frac{\partial c_{i,j}^r}{\partial q_{ij}} \times \frac{\partial q_{ij}}{\partial \boldsymbol{y}_i} = \frac{1}{1 - q_{ij}} \times \Big(\frac{-1}{(1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})^2}$$

$$\times 2ab(\boldsymbol{y}_i - \boldsymbol{y}_j) \times \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2(b-1)} \Big)$$

$$= \frac{-2ab\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2(b-1)}}{(1 - q_{ij})(1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})^2}(\boldsymbol{y}_i - \boldsymbol{y}_j). \qquad (15)$$

The term in the numerator can be simplified as:

$$-2ab\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2(b-1)} = -2b(a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{-2}$$

$$\overset{(5)}{=} -2b(q_{ij}^{-1} - 1)\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{-2}.$$

The term in the denominator can be simplified as:

$$(1 - q_{ij})(1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})^2 \overset{(5)}{=} (1 - q_{ij})q_{ij}^{-2}$$

$$= q_{ij}^{-2} - q_{ij}^{-1} = q_{ij}^{-1}(q_{ij}^{-1} - 1).$$

Hence, Eq. (15) can be simplified as:

$$\frac{\partial c_{i,j}^r}{\partial \boldsymbol{y}_i} = \frac{-2b(q_{ij}^{-1} - 1)\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{-2}}{q_{ij}^{-1}(q_{ij}^{-1} - 1)}(\boldsymbol{y}_i - \boldsymbol{y}_j)$$

$$= \frac{-2b}{\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2\, q_{ij}^{-1}}(\boldsymbol{y}_i - \boldsymbol{y}_j)$$

$$\overset{(5)}{=} \frac{-2b}{\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2\, (1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})}(\boldsymbol{y}_i - \boldsymbol{y}_j).$$

If we add ε for stability to the squared distance in the denominator, the equation is obtained. Q.E.D. $\square$

2.5. Supervised and Semi-supervised Embedding

The explained UMAP algorithm is unsupervised. We can have supervised and semi-supervised embedding by UMAP (Sainburg et al., 2020). For supervised UMAP, we can use UMAP cost function, Eq. (7), regularized by a classification cost function such as cross-entropy or triplet loss. In semi-supervised case, some part of dataset has labels and some part does not. We can iteratively alternate between UMAP's cost function, Eq. (7), and a classification cost function. In this way, the embeddings are updated by UMAP and fine-tuned by class labels and this procedure is repeated iteratively until convergence of embedding.

3. Justifying UMAP's Cost Function by Algebraic Topology and Category Theory

UMAP is a neighbor embedding method where the probability of neighbors for every point is used for optimization of embedding. However, its cost function, Eq. (6), can

be justified by algebraic topology (May, 1992; Friedman, 2012) and category theory (Mac Lane, 2013; Riehl, 2017). Specifically, its theory heavily uses the fuzzy category theory (Spivak, 2012). In this section, we briefly introduce the theory behind UMAP. The reader can skip this section if they do not wish to know the theory behind UMAP's cost function.

First, we introduce some concepts which will be used for theory of UMAP:

- A *simplex* is a generalization of triangle to arbitrary dimensions.

- A *simplicial complex* is a set of points, line segments, triangles, and their d-dimensional counterparts (May, 1992).

- A *fuzzy set* is a mapping $\mu : \mathcal{A} \rightarrow [0, 1]$ from carrier set $\mathcal{A}$ where the mapping is called the membership function (Zadeh, 1965). We can denote a fuzzy set by $(\mathcal{A}, \mu)$.

- In category theory, a *category* is a collection of objects which are linked by arrows. For example, objects can be sets and the arrows can be functions between sets (Mac Lane, 2013). A *morphism* is a mapping from a mathematical structure to another structure without changing type (e.g., morphisms are functions in set theory). A *functor* is defined as a mapping between categories. *Adjunction* is a relation between two functors. The two functors having an adjunction are *adjoint* functors, one of which is the *left adjoint* and the other is the *right adjoint*.

- A *topology* is a geometrical object which is still preserved by continuous deformations such as stretching and twisting but without tearing and making or closing holes. A *topological space* is a set of topologies whose operations are continuous deformations.

- We define the category $\boldsymbol{\Delta}$ whose objects are finite order sets $[n] = \{1, \dots, n\}$ with order-preserving maps as its morphisms (McInnes et al., 2018, Definition 1). A simplicial set is a functor from $\boldsymbol{\Delta}$ to the category of sets (McInnes et al., 2018, Definition 2).

- Consider a category of fuzzy sets, denoted by **Fuzz** (McInnes et al., 2018, Definition 4). The category of fuzzy simplicial sets, denoted by **sFuzz**, is the category of objects with functors from $\boldsymbol{\Delta}$ to **Fuzz** and natural transformations as its morphisms (McInnes et al., 2018, Definition 5).

- An extended-pseudo-metric space is a set $\mathcal{X}$ and a mapping $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_{\geq 0} \cup \{\infty\}$ where for $x, y \in \mathcal{X}$, we have $d(x, y) \geq 0$ and $x = y \implies d(x, y) = 0$ and $d(x, y) = d(y, x)$ and $d(x, z) \leq d(x, y) + d(y, z)$

(McInnes et al., 2018, Definition 6). The mapping d can be seen as a distance metric or pseudo-metric.

- Let **Fin-sFuzz** be the sub-category of bounded fuzzy simplicial sets. Let **FinEPMet** be the sub-category of finite extended-pseudo-metric spaces.

Theorem 1 ((McInnes et al., 2018, Theorem 1)). *The functors **FinReal**: **Fin-sFuzz** $\rightarrow$ **FinEPMet** and **FinSing**: **FinEPMet** $\rightarrow$ **Fin-sFuzz** form an adjunction with **FinReal** and **FinSing** as the left and right adjoints.*

Proof. Proof is available in (McInnes et al., 2018, Appendix B). $\square$

The above theorem shows that we can convert an extended-pseudo-metric space to a fuzzy simplicial set and vice versa. In other words, we have a fuzzy simplicial representation of data space. Hence, we have the following corollary.

Corollary 1 (Fuzzy Topological Representation (McInnes et al., 2018, Definition 9)). *Consider a dataset $\mathcal{X} := \{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^n$ lying on an underlying manifold $\mathcal{M}$. Let $\{(\mathcal{X}, d_i)\}_{i=1}^n$ be a family of extended-pseudo-metric spaces with common carrier set $\mathcal{X}$ such that:*

$$d_i(\boldsymbol{x}_j, \boldsymbol{x}_l) := \begin{cases} d_{\mathcal{M}}(\boldsymbol{x}_j, \boldsymbol{x}_l) - \rho_i & \text{if } i = j \text{ or } i = l, \\ \infty & Otherwise, \end{cases}$$
(16)

where $d_{\mathcal{M}}(., .)$ is the geodesic (shortest) distance on manifold and ρ_i is the distance to the nearest neighbor of $\boldsymbol{x}_i$ (see Eq. (2)). The fuzzy topological representation of dataset $\mathcal{X}$ is:

$$\bigcup_{i=1}^n \boldsymbol{FinSing}((\mathcal{X}, d_i)),$$

where $\bigcup$ is the fuzzy set union.

UMAP creates a fuzzy topological representation for the high-dimensional dataset. Then, it initializes a low-dimensional embedding of dataset and creates a fuzzy topological representation for the low-dimensional embedding of dataset. Then, it tries to modify the low-dimensional embedding of dataset in a way that the fuzzy topological representation of embedding becomes similar to the fuzzy topological representation of high-dimensional data. A measure of difference of two fuzzy topological representations $(\mathcal{A}, \mu_1)$ and $(\mathcal{A}, \mu_2)$ is their cross-entropy defined as (McInnes et al., 2018):

$$c\big((\mathcal{A}, \mu_1), (\mathcal{A}, \mu_2)\big) :=$$
$$\sum_{a \in \mathcal{A}} \left(\mu_1(a) \ln\left(\frac{\mu_1(a)}{\mu_2(a)}\right) + (1 - \mu_1(a)) \ln\left(\frac{1 - \mu_1(a)}{1 - \mu_2(a)}\right) \right).$$
(17)

UMAP minimizes this cross-entropy by changing the embedding iteratively. Hence, it uses cross-entropy as its cost function, i.e., Eq. (6).

4. Neighbor Embedding: Comparison with t-SNE and LargeVis

UMAP has a connection with t-SNE (van der Maaten & Hinton, 2008; Ghojogh et al., 2020a) and LargeVis (Tang et al., 2016). This connection is explained in (McInnes et al., 2018, Appendix C). In fact, all UMAP, t-SNE, and LargeVis are neighbor embedding methods in which attractive and repulsive forces are used (Böhm et al., 2020). As was explained before, for every point considered as anchor, attractive forces are used for pushing neighbor (also called positive) points to anchor and repulsive forces are used for pulling non-neighbor (also called negative) points away from the anchor points. Note that the ideas of triplet and contrastive losses as well as Fisher discriminant analysis are the same (Ghojogh et al., 2020b). An empirical comparison of UMAP and t-SNE is also available in (Repke & Krestel, 2021).

– Comparison of probabilities: In t-SNE, the probabilities in the input and embedding spaces are (Ghojogh et al., 2020a):

$$p_{j|i} := \frac{\exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2}{\sigma_i}\right)}{\sum_{k=1, k \neq i}^n \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_k\|_2}{\sigma_i}\right)}, \qquad (18)$$

$$p_{ij} := \frac{p_{j|i} + p_{i|j}}{2n}, \qquad (19)$$

$$q_{ij} := \frac{(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2)^{-1}}{\sum_{k=1}^n \sum_{l=1, l \neq k}^n (1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l\|_2^2)^{-1}}, \qquad (20)$$

where $p_{i|i} = 0, \forall i$. The $p_{j|i}$ probabilities can be computed for kNN graph where $p_{j|i}$ is set to zero for non-neighbor points in the kNN graph. LargeVis uses the same p_{ij} probabilities as t-SNE but approximates the kNN to compute it very fast and become more efficient. In LargeVis, the probability in the embedding space is:

$$q_{ij} := (1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2)^{-1}. \qquad (21)$$

Comparing Eqs. (1) and (18) shows that UMAP, t-SNE, and LargeVis all use Gaussian or RBF kernel for probabilities in the input space. Comparing Eqs. (4) and (19) shows that UMAP and t-SNE/LargeVis use different approaches for symmetrizing the probabilities in the input space. Comparing Eqs. (5), (20), and (21) shows that, in contrast to t-SNE, UMAP and LargeVis do not normalize the probabilities in the embedding space by all pairs of points. This advantage makes UMAP much faster than t-SNE and also makes it more suitable for mini-batch optimization in deep learning (this will be explained more in Section 7). Comparing Eqs. (5), (20), and (21) also shows that UMAP, t-

SNE, and LargeVis all use Cauchy distribution for probabilities in the embedding space. In fact if we set $a = b = 1$ in Eq. (5), it is exactly the same as Eq. (20) up to the scale of normalization.

– Comparison of cost functions: The cost function in t-SNE, to be minimized, is the KL-divergence (Kullback & Leibler, 1951) of the probabilities in the input and embedding spaces:

$$\begin{aligned} c_4 &:= \sum_{i=1}^n \sum_{j=1, j \neq i}^n p_{ij} \ln\left(\frac{p_{ij}}{q_{ij}}\right) \\ &= \sum_{i=1}^n \sum_{j=1, j \neq i}^n \left(p_{ij} \ln(p_{ij}) - p_{ij} \ln(q_{ij})\right), \end{aligned} \qquad (22)$$

where Eqs. (19) and (20) are used. The cost function of LargeVis, to be minimized, is a negative likelihood function stated below:

$$c_5 := -\sum_{i=1}^n \sum_{j=1, j \neq i}^n \left(p_{ij} \ln(q_{ij}) + \lambda \ln(1 - q_{ij})\right), \qquad (23)$$

where λ is the regularization parameter and Eqs. (19) and (21) are used. Comparing Eqs. (7), (22), and (23) shows that UMAP, t-SNE, and LargeVis have similar, but not exactly equal, cost functions. The first term in all these cost functions is responsible for the attractive forces and the second term is for the repulsive forces; hence, they can all be considered as neighbor embedding methods (Böhm et al., 2020).

5. Discussion on Repulsive Forces and Negative Sampling in the UMAP's Cost Function

5.1. UMAP's Emphasis on Repulsive Forces

The gradient of UMAP's cost function, Eq. (10), in epoch ν is denoted by $(\partial c_2 / \partial \boldsymbol{y}_i)|_\nu$ and it can be stated as (Damrich & Hamprecht, 2021):

$$\left.\frac{\partial c_2}{\partial \boldsymbol{y}_i}\right|_\nu = \sum_{j=1}^n \left(\mathbb{I}_{ij}^\nu \frac{\partial c_{i,j}^a}{\partial \boldsymbol{y}_i} + \mathbb{I}_{ji}^\nu \frac{\partial c_{j,i}^a}{\partial \boldsymbol{y}_i} + \mathbb{I}_{ij}^\nu \sum_{l=1}^n \mathbb{I}_{ijl}^\nu \frac{\partial c_{i,l}^r}{\partial \boldsymbol{y}_i}\right), \qquad (24)$$

where $c_{i,j}^a$ and $c_{i,j}^r$ are defined in Eqs. (8) and (8), respectively, and $\mathbb{I}_{ij}^\nu$ is a binary random variable which is one if the points $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ are randomly selected in epoch ν and otherwise it is zero. Also, $\mathbb{I}_{ijl}^\nu$ is a binary random variable which is one if the point $\boldsymbol{y}_l$ is one of the negative samples which is randomly sampled for the pair $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ in epoch ν and otherwise it is zero. Recall that the original UMAP does not update the embedding of negative sample itself so we do not have a term for that update in this gradient.

Eq. (24) is only for one epoch. The expectation of Eq. (24) over all epochs is (Damrich & Hamprecht, 2021):

$$\mathbb{E}\Big[\frac{\partial c_2}{\partial \boldsymbol{y}_i}\Big|_\nu\Big]$$

$$= \mathbb{E}\Big[\sum_{j=1}^n \Big(\mathbb{I}_{ij}^\nu \frac{\partial c_{i,j}^a}{\partial \boldsymbol{y}_i} + \mathbb{I}_{ji}^\nu \frac{\partial c_{j,i}^a}{\partial \boldsymbol{y}_i} + \mathbb{I}_{ij}^\nu \sum_{l=1}^n \mathbb{I}_{ijl}^\nu \frac{\partial c_{i,l}^r}{\partial \boldsymbol{y}_i}\Big)\Big]$$

$$\stackrel{(a)}{=} \sum_{j=1}^n \Big(\mathbb{E}[\mathbb{I}_{ij}^\nu]\frac{\partial c_{i,j}^a}{\partial \boldsymbol{y}_i} + \mathbb{E}[\mathbb{I}_{ji}^\nu]\frac{\partial c_{j,i}^a}{\partial \boldsymbol{y}_i}\Big)$$
$$+ \sum_{j=1}^n \sum_{l=1}^n \mathbb{E}[\mathbb{I}_{ij}^\nu \mathbb{I}_{ijl}^\nu]\frac{\partial c_{i,l}^r}{\partial \boldsymbol{y}_i}$$

$$\stackrel{(b)}{=} \sum_{j=1}^n \Big(p_{ij}\frac{\partial c_{i,j}^a}{\partial \boldsymbol{y}_i} + p_{ji}\frac{\partial c_{j,i}^a}{\partial \boldsymbol{y}_i}\Big) + \sum_{j=1}^n \sum_{l=1}^n p_{ij}\frac{m}{n}\frac{\partial c_{i,l}^r}{\partial \boldsymbol{y}_i}$$

$$\stackrel{(c)}{=} \sum_{j=1}^n \Big(p_{ij}\frac{\partial c_{i,j}^a}{\partial \boldsymbol{y}_i} + p_{ji}\frac{\partial c_{j,i}^a}{\partial \boldsymbol{y}_i}\Big) + \frac{m}{n}\underbrace{\sum_{j=1}^n p_{ij}}_{= d_i}\sum_{l=1}^n \frac{\partial c_{i,l}^r}{\partial \boldsymbol{y}_i}$$

$$\stackrel{(d)}{=} \sum_{j=1}^n \Big(p_{ij}\frac{\partial c_{i,j}^a}{\partial \boldsymbol{y}_i} + p_{ji}\frac{\partial c_{j,i}^a}{\partial \boldsymbol{y}_i}\Big) + \frac{d_i m}{n}\sum_{j=1}^n \frac{\partial c_{i,j}^r}{\partial \boldsymbol{y}_i}$$

$$\stackrel{(e)}{=} 2\sum_{j=1}^n \Big(p_{ij}\frac{\partial c_{i,j}^a}{\partial \boldsymbol{y}_i} + \frac{d_i m}{2n}\frac{\partial c_{i,j}^r}{\partial \boldsymbol{y}_i}\Big), \qquad (25)$$

where (a) is because expectation is a linear operator, (b) is because $\mathbb{E}[\mathbb{I}_{ij}^\nu] = p_{ij}$ and $\mathbb{E}[\mathbb{I}_{ij}^\nu \mathbb{I}_{ijl}^\nu] = \mathbb{E}[\mathbb{I}_{ijl}^\nu \,|\, \mathbb{I}_{ij}^\nu] \times \mathbb{E}[\mathbb{I}_{ij}^\nu] = \frac{m}{n}\times p_{ij}$, (c) is because we define the degree of the i-th point (node) in the kNN graph as $d_i := \sum_{j=1}^n p_{ij}$, (d) is because we change the dummy variable l to j in the last summation, and (e) is because $p_{ij} = p_{ji}$ and $c_{i,j}^a = c_{j,i}^a$ are symmetric.

On the other hand, according to Eq. (11), the gradient of UMAP's cost function, Eq. (7), can be stated as (Damrich & Hamprecht, 2021):

$$\frac{\partial c_2}{\partial \boldsymbol{y}_i} = 2\sum_{j=1}^n \Big(p_{ij}\frac{\partial c_{i,j}^a}{\partial \boldsymbol{y}_i} + (1-p_{ij})\frac{\partial c_{i,j}^r}{\partial \boldsymbol{y}_i}\Big). \qquad (26)$$

Eq. (26) is the gradient of the original UMAP's loss function while Eq. (25) is the expected gradient of UMAP's loss function. Comparing Eqs. (25) and (26) shows that the original UMAP puts more emphasis on negative samples (or repulsive forces) compared to the expected UMAP's loss function because for a negative sample we have $1-p_{ij} \approx 1$ while $d_i m/n \approx 0$ because $m \ll n$. Therefore, UMAP is mistakenly putting more emphasis on negative sampling (or repulsive forces) than required (Damrich & Hamprecht, 2021). This has also been empirically investigated in (Böhm et al., 2020) that negative sampling (or repulsive forces) in UMAP has more weight than required.

5.2. UMAP's Effective Cost Function

The original UMAP does not update the embedding of negative samples themselves. If we also update them, i.e. we perform line 18 in Algorithm 1, the gradient of UMAP's cost function, Eq. (10), in epoch ν can be stated as (Damrich & Hamprecht, 2021):

$$\frac{\partial c_2}{\partial \boldsymbol{y}_i}\Big|_\nu = \sum_{j=1}^n \Big(\mathbb{I}_{ij}^\nu \frac{\partial c_{i,j}^a}{\partial \boldsymbol{y}_i} + \mathbb{I}_{ji}^\nu \frac{\partial c_{j,i}^a}{\partial \boldsymbol{y}_i}$$
$$+ \mathbb{I}_{ij}^\nu \sum_{l=1}^n \mathbb{I}_{ijl}^\nu \frac{\partial c_{i,l}^r}{\partial \boldsymbol{y}_i} + \sum_{k=1}^n \mathbb{I}_{jk}^\nu \mathbb{I}_{jki}^\nu \frac{\partial c_{j,i}^r}{\partial \boldsymbol{y}_i}\Big). \quad (27)$$

If we do reverse engineering to find the cost function from its gradient, the cost function at epoch ν becomes:

$$c_2\big|_\nu := \sum_{i=1}^n \sum_{j=1,j\neq i}^n \Big(\mathbb{I}_{ij}^\nu c_{i,j}^a + \sum_{l=1}^n \mathbb{I}_{ij}^\nu \mathbb{I}_{ijl}^\nu c_{i,l}^r\Big). \quad (28)$$

This is the cost at one epoch. The expectation of this cost over all epochs is (Damrich & Hamprecht, 2021):

$$c_2 = \mathbb{E}[c_2\big|_\nu]$$

$$= \sum_{i=1}^n \sum_{j=1,j\neq i}^n \Big(\mathbb{E}[\mathbb{I}_{ij}^\nu]\,c_{i,j}^a + \sum_{l=1}^n \mathbb{E}[\mathbb{I}_{ij}^\nu \mathbb{I}_{ijl}^\nu]\,c_{i,l}^r\Big)$$

$$= \sum_{i=1}^n \sum_{j=1,j\neq i}^n \Big(\mathbb{E}[\mathbb{I}_{ij}^\nu]\,c_{i,j}^a\Big)$$
$$+ \sum_{i=1}^n \sum_{j=1,j\neq i}^n \sum_{l=1}^n \Big(\mathbb{E}[\mathbb{I}_{ij}^\nu \mathbb{I}_{ijl}^\nu]\,c_{i,l}^r\Big)$$

$$\stackrel{(a)}{=} \sum_{i=1}^n \sum_{j=1,j\neq i}^n \Big(p_{ij}\,c_{i,j}^a\Big) + \sum_{i=1}^n \sum_{j=1,j\neq i}^n \sum_{l=1}^n \Big(\frac{m}{n}p_{ij}\,c_{i,l}^r\Big)$$

$$= \sum_{i=1}^n \sum_{j=1,j\neq i}^n \Big(p_{ij}\,c_{i,j}^a\Big)$$
$$+ \frac{m}{n}\Big(\underbrace{\sum_{i=1}^n \sum_{j=1}^n p_{ij}}_{=d_i} + \underbrace{\sum_{j=1}^n \sum_{i=1}^n p_{ji}}_{=d_j}\Big)\sum_{l=1}^n c_{i,l}^r$$

$$\stackrel{(b)}{=} \sum_{i=1}^n \sum_{j=1,j\neq i}^n \Big(p_{ij}\,c_{i,j}^a\Big) + \sum_{i=1}^n \sum_{j=i+1}^n \frac{(d_i+d_j)m}{n}c_{i,j}^r$$

$$\stackrel{(c)}{=} 2\sum_{i=1}^n \sum_{j=i+1}^n \Big(p_{ij}\,c_{i,j}^a + \frac{(d_i+d_j)m}{2n}c_{i,j}^r\Big), \quad (29)$$

where (a) is because $\mathbb{E}[\mathbb{I}_{ij}^\nu] = p_{ij}$ and $\mathbb{E}[\mathbb{I}_{ij}^\nu \mathbb{I}_{ijl}^\nu] = \mathbb{E}[\mathbb{I}_{ijl}^\nu \,|\, \mathbb{I}_{ij}^\nu] \times \mathbb{E}[\mathbb{I}_{ij}^\nu] = \frac{m}{n} \times p_{ij}$, (b) is because we change the dummy variable l to j in the last summation, and (c) is because of symmetry of terms in the fist summations with respect to i and j. Eq. (29) can be considered as UMAP's effective cost function (Damrich & Hamprecht, 2021) because it also updates the embedding of negative samples by line 18 in Algorithm 1.

Comparing UMAP's cost, Eq. (11), with UMAP's effective cost, Eq. (29), shows that the weight of negative sample (or repulsive forces) should be $\frac{(d_i+d_j)m}{2n}$ rather than $(1 - p_{ij})$ if we also update the embeddings of negative samples. As we have $m \ll n$ and p_{ij} is small for negative samples, this weight is much less than the weight in the original UMAP.

6. DensMAP for Density-Preserving Embedding

As was explained before, UMAP uses a binary search for the scale of each point, σ_i, to satisfy Eq. (3), so as t-SNE (van der Maaten & Hinton, 2008) which has a similar search for its scale using entropy as perplexity. The search makes the neighborhoods of various points behave similarly so UMAP and t-SNE both assume that points are uniformly distributed on an underlying low-dimensional manifold. Hence, UMAP ignores the density of data around every point by canceling the effect of density with binary search for scales of points. DensMAP (Narayan et al., 2021) regularizes the cost function of UMAP to take into account and add back the information of density around each point. It is shown empirically that this consideration of density information results in better embedding (Narayan et al., 2021) although we will have more computation for calculation of the regularization term.

If the neighbors of a point are very close to it, that region is dense for that point. Therefore, a measure of local density can be the local radius defined as the expected (average) distances from neighbors. We denote the local densities in the input and embedding spaces by:

$$R_p(\boldsymbol{x}_i) := \mathbb{E}_{j \sim p}\big[\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2\big] = \frac{\sum_{j=1}^n p_{ij}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{\sum_{j=1}^n \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}, \tag{30}$$

$$R_q(\boldsymbol{y}_i) := \mathbb{E}_{j \sim q}\big[\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2\big] = \frac{\sum_{j=1}^n q_{ij}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2}{\sum_{j=1}^n \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2}. \tag{31}$$

As the volume of points is proportional to the powers of radius (e.g., notice that the volume of three dimensional sphere is proportional to radius to the power three), the relation of local densities in the input and embedding spaces can be:

$$R_q(\boldsymbol{y}_i) = \alpha\big(R_p(\boldsymbol{x}_i)\big)^\beta \implies r_q^i = \beta\, r_p^i + \gamma, \tag{32}$$

where $r_q^i := \ln(R_q(\boldsymbol{y}_i))$, $r_p^i := \ln(R_p(\boldsymbol{y}_i))$, and $\gamma := \ln(\alpha)$. Therefore, the relation of logarithms of the local densities should be affine dependence. A measure of linear (or affine) dependence is correlation so we use the correlation of logarithms of local densities:

$$\mathrm{Corr}(r_q, r_p) := \frac{\mathrm{Cov}(r_q, r_p)}{\sqrt{\mathrm{Var}(r_q)\mathrm{Var}(r_p)}}, \tag{33}$$

where the covariance and variance of densities are:

$$\mathrm{Cov}(r_q, r_p) := \frac{1}{n-1}\sum_{i=1}^n \Big[(r_q^i - \mu_q)(r_p^i - \mu_p)\Big]$$

$$\mathrm{Var}(r_q) := \frac{1}{n-1}\sum_{i=1}^n (r_q^i - \mu_q)^2,$$

where $\mu_q := (1/n)\sum_{j=1}^n r_q^j$, $\mu_p := (1/n)\sum_{j=1}^n r_p^j$, and $\mathrm{Var}(r_p)$ is defined similarly. The cost function of densMAP, to be minimized, is the UMAP's cost function regularized by maximization of the correlation of local densities (Narayan et al., 2021):

$$c_6 := c_2 - \lambda\,\mathrm{Corr}(r_q, r_p), \tag{34}$$

where λ is the regularization parameter which weights the correlation compared to the UMAP's original cost. The gradient of c_2 is the gradient of original UMAP discussed before. The gradient of the correlation term is (Narayan et al., 2021):

$$\frac{\partial c_6}{\partial \boldsymbol{y}_i} = \sum_{i=1}^n \sum_{j=1,j\neq i}^n \frac{\partial \mathrm{Corr}(r_q, r_p)}{\partial d_{ij}^2}(y_i - y_j), \tag{35}$$

where $d_{ij}^2 := \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2$ and:

$$\frac{\partial \mathrm{Corr}(r_q, r_p)}{\partial d_{ij}^2} = \frac{1}{(n-1)\mathrm{Var}(r_q)^{(3/2)}} \times$$
$$\Bigg[\mathrm{Var}(r_q)\big(r_p^i \frac{\partial r_q^i}{\partial d_{ij}^2} + r_p^j \frac{\partial r_q^j}{\partial d_{ij}^2}\big)$$
$$- \mathrm{Cov}(r_q, r_p)\big((r_q^i - \mu_q)\frac{\partial r_q^i}{\partial d_{ij}^2} + (r_q^j - \mu_q)\frac{\partial r_q^j}{\partial d_{ij}^2}\big)\Bigg],$$

and:

$$\frac{\partial r_q^i}{\partial d_{ij}^2} = (1 + ad_{ij}^{2b})^{-2}\Big[abd_{ij}^{2(b-1)} + e^{-r_q^i}(1 + a(1-b)d_{ij}^2)\Big].$$

Proofs of these derivatives are available in (Narayan et al., 2021, Supplementary Note 2). As in UMAP, DensMAP uses stochastic gradient descent for optimization. Except for the cost function, the algorithm of DensMAP is the same as UMAP.

7. Parametric UMAP for Embedding by Deep Learning

Dimensionality reduction algorithms can have their parametric version in which the cost function of the algorithm is used as the loss function of a neural network and the parameters (i.e., weights) of network are trained by backpropagating the error of loss function. Inspired by parametric t-SNE (Van Der Maaten, 2009), we can have parametric UMAP (Sainburg et al., 2020). Parametric UMAP uses

---

1 **Input**: Training data $\{\boldsymbol{x}_i\}_{i=1}^n$, learning rate η
2 Construct kNN graph
3 Initialize weights θ of network randomly
4 Initialize $\{\boldsymbol{y}_i\}_{i=1}^n \leftarrow \{f_\theta(\boldsymbol{x}_i)\}_{i=1}^n$
5 Calculate p_{ij} and q_{ij} for $\forall i, j \in \{1, \dots, n\}$ by
    Eqs. (4) and (5)
6 // make batches:
7 **for** s *from* 1 *to* $\lfloor n/b \rfloor$ **do**
8 $\mathcal{B}_s \leftarrow \{\boldsymbol{x}_{(s-1)b+1}, \dots, \boldsymbol{x}_{sb}\}$
9   // We denote $\mathcal{B}_s = \{\boldsymbol{x}_1^{(s)}, \dots, \boldsymbol{x}_b^{(s)}\}$
10 $\nu \leftarrow 0$
11 **while** *not converged* **do**
12 $\nu \leftarrow \nu + 1$ // epoch index
13   // optimize over every mini-batch:
14   **for** s *from* 1 *to* $\lfloor n/b \rfloor$ **do**
15 $\{\boldsymbol{y}_i^{(s)}\}_{i=1}^b \leftarrow \{f_\theta(\boldsymbol{x}_i^{(s)})\}_{i=1}^b$
16 $c_s^a \leftarrow 0, c_s^r \leftarrow 0$
17   **for** i *from* 1 *to* b **do**
18   **for** j *from* 1 *to* b **do**
19 $u \sim U(0,1)$
20   **if** $u \le p_{ij}$ **then**
21 $q_{ij} =$
               $(1 + a\|\boldsymbol{y}_i^{(s)} - \boldsymbol{y}_j^{(s)}\|_2^{2b})^{-1}$
22 $c_s^a = c_s^a + (-\ln(q_{ij}))$
23   **for** m *iterations* **do**
24 $l \sim U\{1, \dots, b\}$
25 $q_{il} =$
                $(1 + a\|\boldsymbol{y}_i^{(s)} - \boldsymbol{y}_l^{(s)}\|_2^{2b})^{-1}$
26 $c_s^r = c_s^r + (-\ln(1 - q_{il}))$
27 $\theta \leftarrow$ backpropagate with loss $(c_s^a + c_s^r)$
28 **Return** $\{\boldsymbol{y}_i\}_{i=1}^n \leftarrow \{f_\theta(\boldsymbol{x}_i)\}_{i=1}^n$

Algorithm 2: Parametric UMAP algorithm

UMAP's cost function, Eq. (6), as the loss function of a neural network for deep learning. It optimizes this cost in mini-batches rather than on the whole dataset; therefore, it can be used for embedding large datasets. Also, because of nonlinearity of neural networks, the parametric UMAP can handle highly nonlinear data better than UMAP. Note that the learnable parameters in UMAP are the embedding of points but the learnable parameters in the parametric UMAP are the weights of neural network so that the embedding is obtained by network. An advantage of parametric UMAP over parametric t-SNE is that the probability of UMAP in the embedding space, Eq. (5), does not have a normalization factor; hence, there is no need to normalize over the whole dataset. This makes UMAP easy to be used in deep learning.

The algorithm of parametric UMAP is shown in Algorithm 2. As this algorithm shows, we make mini-batches of data points where $\mathcal{B}_s$ denotes the s-th batch. In every epoch, we iterate over mini-batches and for every mini-batch, we iterate twice over the points of batch to have anchors and positive points. We optimize the cost function of UMAP over the batch and not the entire data. We denote the attractive and repulsive cost functions by c_s^a and c_s^r, respectively. According to Eq. (10), the loss function of neural network in parametric UMAP should be $(c_s^a + c_s^r)$. Backpropagating this loss function trains the parameters θ of the neural network denoted by $f_\theta(.)$. After training, the embeddings are obtained as $\{f_\theta(\boldsymbol{x}_i)\}_{i=1}^n$. Note that as was discussed in Section 2.5, one can combine the UMAP's cost function with cross-entropy loss or triplet loss to have supervised or semi-supervised embedding. Moreover, combining UMAP's cost function with reconstruction loss can train an autoencoder for embedding in its middle code layer.

8. Progressive UMAP for Streaming and Out-of-sample Data

The original UMAP does not support out-of-sample (test) data embedding. Progressive UMAP (Ko et al., 2020) can embed out-of-sample data. It can also be used for embedding streaming (online) data. Although UMAP is generally faster than t-SNE, it takes some noticeable time to embed big data. Progressive UMAP can be used to embed some portion of data and then complete the embedding by embedding the rest of data as streaming data.

The algorithm of progressive UMAP is shown in Algorithm 3. It uses PANENE (Jo et al., 2018) for constructing a streaming kNN graph. PANENE uses randomized kd-trees (Muja & Lowe, 2009) for approximating and updating kNN graph. When a new batch of data, $\boldsymbol{X}_{\text{new}}$, is arrived, some of data points are affected in kNN graph because their neighbors are changed or they become new neighbors of some points or they are no longer neighbors of some points. We denote the affected points by $\boldsymbol{X}_{\text{updated}}$. If $\boldsymbol{X}_{\text{new}}$ is the first batch of data, we embed data by Laplacian eigenmap, exactly as the original UMAP does. If the coming batch is not the first batch, we find the nearest neighbor of every new point among the previously accumulated data. Then, we set the initial embedding of every new point as the embedding of its nearest neighbor point added with some Gaussian noise. As in the original UMAP, for every new or updated point, we calculate or update ρ_i and σ_i by Eqs. (2) and (3). For every new point $\boldsymbol{x}_i$, we also calculate p_{ij} and q_{ij} by Eqs. (4) and (5), where j indexes all existing and new points. Then, for every new or updated point, we update the embedding of point by gradient descent where m negative samples are used as in the original UMAP. The paper of progressive UMAP has not mentioned updating the

1 **Input**: New batch of training data
$X_{\text{new}} = \{x_i\}_{i=1}^n$

2 $X_{\text{new}}, X_{\text{updated}} \leftarrow$ Update kNN graph by PANENE method

3 **if** *it is initial batch* **then**

4 Initialize $\{y_i\}_{i=1}^n$ by Laplacian eigenmap

5 **else**

6 **for** *each new point x_i in X_{new}* **do**

7 Find nearest neighbor to previously accumulated data

8 Initialize y_i to the embedding of the nearest neighbor point plus Gaussian noise

9 **for** *each new or updated point x_i in X_{new} or $X_{updated}$* **do**

10 Calculate/update ρ_i and σ_i by Eqs. (2) and (3)

11 Calculate p_{ij} and q_{ij} for $\forall j$ by Eqs. (4) and (5)

12 **while** *not converged* **do**

13 **for** *each new or updated point x_i in X_{new} or $X_{updated}$* **do**

14 **for** *j from 1 to n* **do**

15 $u \sim U(0,1)$

16 **if** $u \leq p_{ij}$ **then**

17 $y_i \leftarrow y_i - \eta \frac{\partial c_{i,j}^a}{\partial y_i}$

18 **for** *m iterations* **do**

19 $l \sim U\{1, \ldots, n\}$

20 $y_i \leftarrow y_i - \eta \frac{\partial c_{i,l}^r}{\partial y_i}$

21 **Return** embedding $\{y_i\}$ for the new and updated points

Algorithm 3: Progressive UMAP algorithm

embedding of neighbor (positive) points which we have in the original UMAP.

9. Conclusion

This was a tutorial paper on UMAP and its variants. We explained the details of UMAP and derived its gradients. We explained the theory of UMAP's cost function by algebraic topology and fuzzy category theory. We compared UMAP with t-SNE and LargeVis, discussed the repulsive forces and negative sampling, and introduced DensMAP, parametric UMAP, and progressive UMAP. For brevity, some other algorithms such as using genetic programming with UMAP (Schofield & Lensen, 2021) was not explained in this paper.

References

Becht, Etienne, McInnes, Leland, Healy, John, Dutertre, Charles-Antoine, Kwok, Immanuel WH, Ng, Lai Guan, Ginhoux, Florent, and Newell, Evan W. Dimensionality reduction for visualizing single-cell data using UMAP. *Nature biotechnology*, 37(1):38–44, 2019.

Belkin, Mikhail and Niyogi, Partha. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, volume 14, pp. 585–591, 2001.

Böhm, Jan Niklas, Berens, Philipp, and Kobak, Dmitry. A unifying perspective on neighbor embeddings along the attraction-repulsion spectrum. *arXiv preprint arXiv:2007.08902*, 2020.

Carter, Shan, Armstrong, Zan, Schubert, Ludwig, Johnson, Ian, and Olah, Chris. Activation atlas. *Distill*, 4(3):e15, 2019.

Coenen, Andy, Reif, Emily, Yuan, Ann, Kim, Been, Pearce, Adam, Viégas, Fernanda, and Wattenberg, Martin. Visualizing and measuring the geometry of BERT. *arXiv preprint arXiv:1906.02715*, 2019.

Damrich, Sebastian and Hamprecht, Fred A. On UMAP's true loss function. *arXiv preprint arXiv:2103.14608*, 2021.

Dorrity, Michael W, Saunders, Lauren M, Queitsch, Christine, Fields, Stanley, and Trapnell, Cole. Dimensionality reduction by UMAP to visualize physical and genetic interactions. *Nature communications*, 11(1):1–6, 2020.

Friedman, Greg. Survey article: an elementary illustrated introduction to simplicial sets. *The Rocky Mountain Journal of Mathematics*, pp. 353–423, 2012.

Ghojogh, Benyamin. *Data Reduction Algorithms in Machine Learning and Data Science*. PhD thesis, University of Waterloo, 2021.

Ghojogh, Benyamin, Ghodsi, Ali, Karray, Fakhri, and Crowley, Mark. Stochastic neighbor embedding with Gaussian and Student-t distributions: Tutorial and survey. *arXiv preprint arXiv:2009.10301*, 2020a.

Ghojogh, Benyamin, Sikaroudi, Milad, Shafiei, Sobhan, Tizhoosh, Hamid R, Karray, Fakhri, and Crowley, Mark. Fisher discriminant triplet and contrastive losses for training Siamese networks. In *2020 international joint conference on neural networks (IJCNN)*, pp. 1–7. IEEE, 2020b.

Ghojogh, Benyamin, Ghodsi, Ali, Karray, Fakhri, and Crowley, Mark. Laplacian-based dimensionality reduction including spectral clustering, Laplacian eigenmap, locality preserving projection, graph embedding,

and diffusion map: Tutorial and survey. *arXiv preprint arXiv:2106.02154*, 2021.

Hinton, Geoffrey E and Roweis, Sam T. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pp. 857–864, 2003.

Jo, Jaemin, Seo, Jinwook, and Fekete, Jean-Daniel. PANENE: A progressive algorithm for indexing and querying approximate k-nearest neighbors. *IEEE transactions on visualization and computer graphics*, 26(2): 1347–1360, 2018.

Ko, Hyung-Kwon, Jo, Jaemin, and Seo, Jinwook. Progressive uniform manifold approximation and projection. In *EuroVis (Short Papers)*, pp. 133–137, 2020.

Kobak, Dmitry and Berens, Philipp. The art of using t-SNE for single-cell transcriptomics. *Nature communications*, 10(1):1–14, 2019.

Kullback, Solomon and Leibler, Richard A. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Levine, Yoav, Lenz, Barak, Dagan, Or, Ram, Ori, Padnos, Dan, Sharir, Or, Shalev-Shwartz, Shai, Shashua, Amnon, and Shoham, Yoav. Sensebert: Driving some sense into BERT. *arXiv preprint arXiv:1908.05646*, 2019.

Mac Lane, Saunders. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 2013.

May, J Peter. *Simplicial objects in algebraic topology*, volume 11. University of Chicago Press, 1992.

McInnes, Leland, Healy, John, and Melville, James. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Muja, Marius and Lowe, David G. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

Narayan, Ashwin, Berger, Bonnie, and Cho, Hyunghoon. Density-preserving data visualization unveils dynamic patterns of single-cell transcriptomic variability. *Nature Biotechnology*, 39:765–774, 2021.

Repke, Tim and Krestel, Ralf. Robust visualisation of dynamic text collections: Measuring and comparing dimensionality reduction algorithms. In *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*, pp. 255–259. ACM, 2021.

Riehl, Emily. *Category theory in context*. Courier Dover Publications, 2017.

Sainburg, Tim, McInnes, Leland, and Gentner, Timothy Q. Parametric UMAP: learning embeddings with deep neural networks for representation and semi-supervised learning. *arXiv preprint arXiv:2009.12981*, 2020.

Schofield, Finn and Lensen, Andrew. Using genetic programming to find functional mappings for UMAP embeddings. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC*, 2021.

Spivak, David I. Metric realization of fuzzy simplicial sets. Technical report, Self published notes, 2012.

Tang, Jian, Liu, Jingzhou, Zhang, Ming, and Mei, Qiaozhu. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th international conference on world wide web*, pp. 287–297, 2016.

Van Der Maaten, Laurens. Learning a parametric embedding by preserving local structure. In *Artificial Intelligence and Statistics*, pp. 384–391. PMLR, 2009.

van der Maaten, Laurens and Hinton, Geoffrey. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Vermeulen, Marc, Smith, Kate, Eremin, Katherine, Rayner, Georgina, and Walton, Marc. Application of uniform manifold approximation and projection (UMAP) in spectral imaging of artworks. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 252: 119547, 2021.

Zadeh, Lofti A. Fuzzy sets. *Information and Control*, 8(3): 338–353, 1965.