# UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction

Leland McInnes
Tutte Institute for Mathematics and Computing
leland.mcinnes@gmail.com

John Healy
Tutte Institute for Mathematics and Computing
jchealy@gmail.com

James Melville
jlmelville@gmail.com

December 7, 2018

**Abstract**

UMAP (Uniform Manifold Approximation and Projection) is a novel manifold learning technique for dimension reduction. UMAP is constructed from a theoretical framework based in Riemannian geometry and algebraic topology. The result is a practical scalable algorithm that applies to real world data. The UMAP algorithm is competitive with t-SNE for visualization quality, and arguably preserves more of the global structure with superior run time performance. Furthermore, UMAP has no computational restrictions on embedding dimension, making it viable as a general purpose dimension reduction technique for machine learning.

## 1 Introduction

Dimension reduction seeks to produce a low dimensional representation of high dimensional data that preserves relevant structure (relevance often being application dependent). Dimension reduction is an important problem in data science for both visualization, and a potential pre-processing step for machine learning.

Dimension reduction plays an important role in data science, being a fundamental technique in both visualisation and as pre-processing for machine learning. Dimension reduction techniques are being applied in a broadening range of fields and on ever increasing sizes of datasets. It is thus desirable to have an algorithm that is both scalable to massive data and able to cope with the diversity of data available. Dimension reduction algorithms tend to fall into two categories; those that seek to preserve the distance structure within the data and those that favor the preservation of local distances over global distance. Algorithms such as PCA [22], MDS [23], and Sammon mapping [41] fall into the former category while t-SNE [50, 49], Isomap [47], LargeVis [45], Laplacian eigenmaps [5, 6] and diffusion maps [14] all fall into the latter category.

In this paper we introduce a novel manifold learning technique for dimension reduction. We provide a sound mathematical theory grounding the technique and a practical scalable algorithm that applies to real world data. UMAP (Uniform Manifold Approximation and Projection) builds upon mathematical foundations related to the work of Belkin and Niyogi on Laplacian eigenmaps. We seek to address the issue of uniform data distributions on manifolds through a combination of Riemannian geometry and the work of David Spivak [43] in category theoretic approaches to geometric realization of fuzzy simplicial sets. t-SNE is the current state-of-the-art for dimension reduction for visualization. Our algorithm is competitive with t-SNE for visualization quality and arguably preserves more of the global structure with superior run time performance. Furthermore, UMAP's topological foundations allow it to scale to significantly larger data set sizes than are feasible for t-SNE. Finally, UMAP has no computational restrictions on embedding dimension, making it viable as a general purpose dimension reduction technique for machine learning.

Based upon preliminary releases of a software implementation, UMAP has already found widespread use in the fields of bioinformatics [4, 15, 37, 2, 36, 13], materials science [27, 19], and machine learning [8, 20, 17, 38] among others.

This paper is laid out as follows. In Section 2 we describe the theory underlying the algorithm. Section 2 is necessary to understand both the theory underlying why UMAP works and the motivation for the choices that where made in developing the algorithm. A reader without a background (or interest) in topological data analysis, category theory or the theoretical underpinnings of UMAP should skip over this section and proceed directly to Section 3.

That being said, we feel that strong theory and mathematically justified algorithmic decisions are of particular importance in the field of unsupervised learning. This is, at least partially, due to plethora of proposed objective functions

within the area.

In Section 3 we provide a more computation description of UMAP. Section 3 should provide readers less familiar with topological data analysis with a better foundation for understanding the theory described in Section 2. Appendix C contrasts UMAP against the more familiar algorithms t-SNE and LargeVis, describing all these algorithms in similar language. This section should assist readers already familiar with those techniques to quickly gain an understanding of the UMAP algorithm if not its theoretical underpinnings.

In Section 4 we discuss implementing the UMAP algorithm. This includes a more detailed algorithmic description, and discussion of the hyper-parameters involved and their practical effects.

In Section 5 we provide practical results on real world datasets as well as scaling experiments to demonstrate the algorithm's performance in real world scenarios as compared with other dimension reduction algorithms.

In Section 6 we discuss relative weakenesses of the algorithm, and applications for which UMAP may not be the best choice.

Finally, in Section 7 we detail an number of potential extensions of UMAP that are made possible by its construction upon solid mathematical foundations. These avenues for further development include semi-supervised learning, metric learning and heterogeneous data embedding.

## 2 Theoretical Foundations for UMAP

The theoretical foundations for UMAP are largely based in manifold theory and topological data analysis. Much of the theory is most easily explained in the language of topology and category theory. Readers may consult [31], [40] and [32] for background. Readers more interested in practical computational aspects of the algorithm, and not necessarily the theoretical motivation for the computations involved, may wish to skip this section.

At a high level, UMAP uses local manifold approximations and patches together their local fuzzy simplicial set representations to construct a topological representation of the high dimensional data. Given some low dimensional representation of the data, a similar process can be used to construct an equivalent topological representation. UMAP then optimizes the layout of the data representation in the low dimensional space, to minimize the cross-entropy between the two topological representations.

The construction of fuzzy topological representations can be broken down

into two problems: approximating a manifold on which the data is assumed to lie; and constructing a fuzzy simplicial set representation of the approximated manifold. In explaining the algorithm we will first discuss the method of approximating the manifold for the source data. Next we will discuss how to construct a fuzzy simplicial set structure from the manifold approximation. Finally, We will discuss the construction of the fuzzy simplicial set associated to a low dimensional representation (where the manifold is simply $\mathbb{R}^d$), and how to optimize the representation with respect to our objective function.

## 2.1 Uniform distribution of data on a manifold and geodesic approximation

The first step of our algorithm is to approximate the manifold we assume the data lies on. The manifold may be known apriori (as simply $\mathbb{R}^n$) or may need to be inferred from the data. Suppose the manifold is not known in advance and we wish to approximate geodesic distance on it. Let the input data be $X = \{X_1, \ldots, X_N\}$. As in the work of Belkin and Niyogi on Laplacian eigenmaps [5, 6], for theoretical reasons it is beneficial to assume the data is uniformly distributed on the manifold. In practice, real world data is rarely so nicely behaved. However, if we assume that the manifold has a Riemannian metric not inherited from the ambient space, we can find a metric such that the data is approximately uniformly distributed with regard to that metric.

Formally, let $\mathcal{M}$ be the manifold we assume the data to lie on, and let $g$ be the Riemannian metric on $\mathcal{M}$. Thus, for each point $p \in \mathcal{M}$ we have $g_p$, an inner product on the tangent space $T_p\mathcal{M}$.

**Lemma 1.** *Let $(\mathcal{M}, g)$ be a Riemannian manifold in an ambient $\mathbb{R}^n$, and let $p \in M$ be a point. If $g$ is locally constant about $p$ in an open neighbourhood $U$ such that $g$ is a constant diagonal matrix in ambient coordinates, then in a ball $B \subseteq U$ centered at $p$ with volume $\frac{\pi^{n/2}}{\Gamma(n/2+1)}$ with respect to $g$, the geodesic distance from $p$ to any point $q \in B$ is $\frac{1}{r} d_{\mathbb{R}^n}(p, q)$, where $r$ is the radius of the ball in the ambient space and $d_{\mathbb{R}^n}$ is the existing metric on the ambient space.*

See Appendix A of the supplementary materials for a proof of Lemma 1.

If we assume the data to be uniformly distributed on $\mathcal{M}$ (with respect to $g$) then any ball of fixed volume should contain approximately the same number of points of $X$ regardless of where on the manifold it is centered. Conversely, a ball centered at $X_i$ that contains exactly the $k$-nearest-neighbors of $X_i$ should have

fixed volume regardless of the choice of $X_i \in X$. Under Lemma 1 it follows that we can approximate geodesic distance from $X_i$ to its neighbors by normalising distances with respect to the distance to the $k^{\text{th}}$ nearest neighbor of $X_i$.

In essence, by creating a custom distance for each $X_i$, we can ensure the validity of the assumption of uniform distribution on the manifold assumption. The cost is that we now have an independent notion of distance for each and every $X_i$, and these notions of distance may not be compatible. We have a family of discrete metric spaces (one for each $X_i$) that we wish to merge into a consistent global structure. This can be done in a natural way by converting the metric spaces into fuzzy simplicial sets.

## 2.2 Fuzzy topological representation

We will convert to fuzzy topological representations as a means to merge the incompatible local views of the data. The topological structure of choice is that of simplicial sets. For more details on simplicial sets we refer the reader to [21], [32], [39], or [18]. Our approach draws heavily upon the work of Michael Barr [3] and David Spivak in [43], and many of the definitions and theorems below are drawn or adapted from those sources.

To start we will review the definitions for simplicial sets. Simplicial sets provide a combinatorial approach to the study of topological spaces. They are related to the simpler notion of simplicial complexes – which construct topological spaces by gluing together simple building blocks called simplices – but are more general. Simplicial sets are most easily defined purely abstractly in the language of category theory.

**Definition 1.** *The category $\mathbf{\Delta}$ has as objects the finite order sets $[n] = \{1, \dots, n\}$, with morphims given by (non-strictly) order-preserving maps.*

**Definition 2.** *A* simplicial set *is a functor from $\mathbf{\Delta}^{op}$ to* **Sets***, the category of sets.*

Given a simplicial set $X : \mathbf{\Delta}^{\text{op}} \to \mathbf{Sets}$, it is common to denote the set $X([n])$ as $X_n$ and refer to the elements of the set as the $n$-simplices of $X$. The simplest possible examples of simplicial sets are the *standard simplices* $\Delta^n$, defined as the representable functors $\hom_{\mathbf{\Delta}}(\cdot, [n])$. It follows from the Yoneda lemma that there is a natural correspondence between $n$-simplices of $X$ and morphisms $\Delta^n \to X$ in the category of simplicial sets, and it is often helpful to think in these terms. Thus for each $x \in X_n$ we have a corresponding morphism $x : \Delta^n \to X$. By the

density theorem and employing a minor abuse of notation we then have

$$\operatorname*{colim}_{x \in X_n} \Delta^n \cong X$$

There is a standard covariant functor $|\cdot| : \mathbf{\Delta} \to \mathbf{Top}$ mapping from the category $\mathbf{\Delta}$ to the category of topological spaces that sends $[n]$ to the standard $n$-simplex $|\Delta^n| \subset \mathbb{R}^{n+1}$ defined as

$$|\Delta^n| \triangleq \left\{ (t_0, \ldots, t_n) \in \mathbb{R}^{n+1} \mid \sum_{i=0}^{n} t_i = 1, t_i \geq 0 \right\}$$

with the standard subspace topology. If $X : \mathbf{\Delta}^{\text{op}} \to \mathbf{Sets}$ is a simplicial set then we can construct the realization of $X$ (denoted $|X|$) as the colimit

$$|X| = \operatorname*{colim}_{x \in X_n} |\Delta^n|$$

and thus associate a topological space with a given simplicial set. Conversely given a topological space $Y$ we can construct an associated simplicial set $S(Y)$, called the singular set of $Y$, by defining

$$S(Y) : [n] \mapsto \hom_{\mathbf{Top}}(|\Delta^n|, Y).$$

It is a standard result of classical homotopy theory that the realization functor and singular set functors form an adjunction, and provide the standard means of translating between topological spaces and simplicial sets. Our goal will be to adapt these powerful classical results to the case of finite metric spaces.

We draw significant inspiration from Spivak, specifically [43], where he extends the classical theory of singular sets and topological realization to fuzzy singular sets and metric realization. To develop this theory here we will first outline a categorical presentation of fuzzy sets, due to [3], that will make extending classical simplicial sets to fuzzy simplicial sets most natural.

Classically a fuzzy set [55] is defined in terms of a carrier set $A$ and a map $\mu : A \to [0, 1]$ called the membership function. One is to interpret the value $\mu(x)$ for $x \in A$ to be the *membership strength* of $x$ to the set $A$. Thus membership of a set is no longer a bi-valent true or false property as in classical set theory, but a fuzzy property taking values in the unit interval. We wish to formalize this in terms of category theory.

Let $I$ be the unit interval $(0, 1] \subseteq \mathbb{R}$ with topology given by intervals of the form $[0, a)$ for $a \in (0, 1]$. The category of open sets (with morphisms given by inclusions) can be imbued with a Grothendieck topology in the natural way for any poset category.

**Definition 3.** *A presheaf $\mathscr{P}$ on $I$ is a functor from $I^{op}$ to* **Sets**. *A* fuzzy set *is a presheaf on $I$ such that all maps $\mathscr{P}(a \leq b)$ are injections.*

Presheaves on $I$ form a category with morphisms given by natural transformations. We can thus form a category of fuzzy sets by simply restricting to the sub-category of presheaves that are fuzzy sets. We note that such presheaves are trivially sheaves under the Grothendieck topology on $I$. As one might expect, limits (including products) of such sheaves are well defined, but care must be taken to define colimits (and coproducts) of sheaves. To link to the classical approach to fuzzy sets one can think of a section $\mathscr{P}([0, a))$ as the set of all elements with membership strength at least $a$. We can now define the category of fuzzy sets.

**Definition 4.** *The category* **Fuzz** *of* fuzzy sets *is the full subcategory of sheaves on $I$ spanned by fuzzy sets.*

With this categorical presentation in hand, defining fuzzy simplicial sets is simply a matter of considering presheaves of $\mathbf{\Delta}$ valued in the category of fuzzy sets rather than the category of sets.

**Definition 5.** *The category of* fuzzy simplicial sets **sFuzz** *is the category with objects given by functors from $\mathbf{\Delta}^{op}$ to* **Fuzz**, *and morphisms given by natural transformations.*

Alternatively, a fuzzy simplicial set can be viewed as a sheaf over $\mathbf{\Delta} \times I$, where $\mathbf{\Delta}$ is given the trivial topology and $\mathbf{\Delta} \times I$ has the product topology. We will use $\Delta^n_{<a}$ to denote the sheaf given by the representable functor of the object $([n], [0, a))$. The importance of this fuzzy (sheafified) version of simplicial sets is their relationship to metric spaces. We begin by considering the larger category of extended-pseudo-metric spaces.

**Definition 6.** *An* extended-pseudo-metric space $(X, d)$ *is a set $X$ and a map $d : X \times X \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ such that*

1. $d(x, y) \geqslant 0$, *and* $x = y$ *implies* $d(x, y) = 0$;
2. $d(x, y) = d(y, x)$; *and*
3. $d(x, z) \leqslant d(x, y) + d(y, z)$ *or* $d(x, z) = \infty$.

*The category of extended-pseudo-metric spaces* **EPMet** *has as objects extended-pseudo-metric spaces and non-expansive maps as morphisms. We denote the subcategory of finite extended-pseudo-metric spaces* **FinEPMet**.

The choice of non-expansive maps in Definition 6 is due to Spivak, but we note that it closely mirrors the work of Carlsson and Memoli in [12] on topological methods for clustering as applied to finite metric spaces. This choice is significant since pure isometries are too strict and do not provide large enough Hom-sets.

In [43] Spivak constructs a pair of adjoint functors, Real and Sing between the categories **sFuzz** and **EPMet**. These functors are the natural extension of the classical realization and singular set functors from algebraic topology. The functor Real is defined in terms of standard fuzzy simplices $\Delta^n_{<a}$ as

$$\mathsf{Real}(\Delta^n_{<a}) \triangleq \left\{ (t_0, \ldots, t_n) \in \mathbb{R}^{n+1} \mid \sum_{i=0}^{n} t_i = -\log(a), t_i \geq 0 \right\}$$

similarly to the classical realization functor $|\cdot|$. The metric on $\mathsf{Real}(\Delta^n_{<a})$ is simply inherited from $\mathbb{R}^{n+1}$. A morphism $\Delta^n_{<a} \to \Delta^m_{<b}$ exists only if $a \leq b$, and is determined by a $\mathbf{\Delta}$ morphism $\sigma : [n] \to [m]$. The action of Real on such a morphism is given by the map

$$(x_0, x_1, \ldots, x_n) \mapsto \frac{\log(b)}{\log(a)} \left( \sum_{i_0 \in \sigma^{-1}(0)} x_{i_0}, \sum_{i_0 \in \sigma^{-1}(1)} x_{i_0}, \ldots, \sum_{i_0 \in \sigma^{-1}(m)} x_{i_0} \right).$$

Such a map is clearly non-expansive since $0 \leq a \leq b \leq 1$ implies that $\log(b)/\log(a) \leq 1$.

We then extend this to a general simplicial set $X$ via colimits, defining

$$\mathsf{Real}(X) \triangleq \operatorname*{colim}_{\Delta^n_{<a} \to X} \mathsf{Real}(\Delta^n_{<a}).$$

Since the functor Real preserves colimits, it follows that there exists a right adjoint functor. Again, analogously to the classical case, we find the right adjoint, denoted Sing, is defined for an extended pseudo metric space $Y$ in terms of its action on the category $\mathbf{\Delta} \times I$:

$$\mathsf{Sing}(Y) : ([n], [0, a)) \mapsto \hom_{\mathbf{EPMet}}(\mathsf{Real}(\Delta^n_{<a}), Y).$$

For our case we are only interested in finite metric spaces. To correspond with this we consider the subcategory of bounded fuzzy simplicial sets **Fin-sFuzz**. We therefore use the analogous adjoint pair FinReal and FinSing. Formally we define the finite fuzzy realization functor as follows:

**Definition 7.** *Define the functor* $\mathsf{FinReal} : \mathbf{Fin\text{-}sFuzz} \to \mathbf{FinEPMet}$ *by setting*

$$\mathsf{FinReal}(\Delta^n_{<a}) \triangleq (\{x_1, x_2, \ldots, x_n\}, d_a),$$

*where*

$$d_a(x_i, x_j) = \begin{cases} -\log(a) & \text{if } i \neq j, \\ 0 & \text{otherwise} \end{cases}.$$

*and then defining*

$$\mathsf{FinReal}(X) \triangleq \operatorname*{colim}_{\Delta^n_{<a} \to X} \mathsf{FinReal}(\Delta^n_{<a}).$$

Similar to Spivak's construction, the action of $\mathsf{FinReal}$ on a map $\Delta^n_{<a} \to \Delta^m_{<b}$, where $a \leq b$ defined by $\sigma : \Delta^n \to \Delta^m$, is given by

$$(\{x_1, x_2, \ldots, x_n\}, d_a) \mapsto (\{x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)}\}, d_b),$$

which is a non-expansive map since $a \leq b$ implies $d_a \geq d_b$.

Since $\mathsf{FinReal}$ preserves colimits it admits a right adjoint, the fuzzy singular set functor $\mathsf{FinSing}$. We can then define the (finite) fuzzy singular set functor in terms of the action of its image on $\mathbf{\Delta} \times I$, analogously to $\mathsf{Sing}$.

**Definition 8.** *Define the functor* $\mathsf{FinSing} : \mathbf{FinEPMet} \to \mathbf{Fin\text{-}sFuzz}$ *by*

$$\mathsf{FinSing}(Y) : ([n], [0, a)) \mapsto \hom_{\mathbf{FinEPMet}}(\mathsf{FinReal}(\Delta^n_{<a}), Y).$$

We then have the following theorem.

**Theorem 1.** *The functors* $\mathsf{FinReal} : \mathbf{Fin\text{-}sFuzz} \to \mathbf{FinEPMet}$ *and* $\mathsf{FinSing} : \mathbf{FinEPMet} \to \mathbf{Fin\text{-}sFuzz}$ *form an adjunction with* $\mathsf{FinReal}$ *the left adjoint and* $\mathsf{FinSing}$ *the right adjoint.*

The proof of this is by construction. Appendix B provides a full proof of the theorem.

With the necessary theoretical background in place, the means to handle the family of incompatible metric spaces described above becomes clear. Each metric space in the family can be translated into a fuzzy simplicial set via the fuzzy singular set functor, distilling the topological information while still retaining metric information in the fuzzy structure. Ironing out the incompatibilities of the resulting family of fuzzy simplicial sets can be done by simply taking a (fuzzy)

union across the entire family. The result is a single fuzzy simplicial set which captures the relevant topological and underlying metric structure of the manifold $\mathcal{M}$.

It should be noted, however, that the fuzzy singular set functor applies to extended-pseudo-metric spaces, which are a relaxation of traditional metric spaces. The results of Lemma 1 only provide accurate approximations of geodesic distance local to $X_i$ for distances measured from $X_i$ – the geodesic distances between other pairs of points within the neighborhood of $X_i$ are not well defined. In deference to this lack of information we define distances between $X_j$ and $X_k$ in the extended-pseudo metric space local to $X_i$ (where $i \neq j$ and $i \neq k$) to be infinite (local neighborhoods of $X_j$ and $X_k$ will provide suitable approximations).

For real data it is safe to assume that the manifold $\mathcal{M}$ is locally connected. In practice this can be realized by measuring distance in the extended-pseudo-metric space local to $X_i$ as geodesic distance *beyond* the nearest neighbor of $X_i$. Since this sets the distance to the nearest neighbor to be equal to 0 this is only possible in the more relaxed setting of extended-pseudo-metric spaces. It ensures, however, that each 0-simplex is the face of some 1-simplex with fuzzy membership strength 1, meaning that the resulting topological structure derived from the manifold is locally connected. We note that this has a similar practical effect to the truncated similarity approach of Lee and Verleysen [26], but derives naturally from the assumption of local connectivity of the manifold.

Combining all of the above we can define the fuzzy topological representation of a dataset.

**Definition 9.** *Let $X = \{X_1, \ldots, X_N\}$ be a dataset in $\mathbb{R}^n$. Let $\{(X, d_i)\}_{i=1\ldots N}$ be a family of extended-pseudo-metric spaces with common carrier set $X$ such that*

$$d_i(X_j, X_k) = \begin{cases} d_{\mathcal{M}}(X_j, X_k) - \rho & \text{if } i = j \text{ or } i = k, \\ \infty & \text{otherwise}, \end{cases}$$

*where $\rho$ is the distance to the nearest neighbor of $X_i$ and $d_{\mathcal{M}}$ is geodesic distance on the manifold $\mathcal{M}$, either known apriori, or approximated as per Lemma 1.*

*The fuzzy topological representation of $X$ is*

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

The (fuzzy set) union provides the means to merge together the different metric spaces. This provides a single fuzzy simplicial set as the global representation of the manifold formed by patching together the many local representations.

Given the ability to construct such topological structures, either from a known manifold, or by learning the metric structure of the manifold, we can perform dimension reduction by simply finding low dimensional representations that closely match the topological structure of the source data. We now consider the task of finding such a low dimensional representation.

## 2.3 Optimizing a low dimensional representation

Let $Y = \{Y_1, \ldots, Y_N\} \subseteq \mathbb{R}^d$ be a low dimensional ($d \ll n$) representation of $X$ such that $Y_i$ represents the source data point $X_i$. In contrast to the source data where we want to estimate a manifold on which the data is uniformly distributed, we know the manifold for $Y$ is $\mathbb{R}^d$ itself. Therefore we know the manifold and manifold metric apriori, and can compute the fuzzy topological representation directly. Of note, we still want to incorporate the distance to the nearest neighbor as per the local connectedness requirement. This can be achieved by supplying a parameter that defines the expected distance between nearest neighbors in the embedded space.

Given fuzzy simplicial set representations of $X$ and $Y$, a means of comparison is required. If we consider only the 1-skeleton of the fuzzy simplicial sets we can describe each as a fuzzy graph, or, more specifically, a fuzzy set of edges. To compare two fuzzy sets we will make use of fuzzy set cross entropy. For these purposes we will revert to classical fuzzy set notation. That is, a fuzzy set is given by a reference set $A$ and a membership strength function $\mu : A \to [0, 1]$. Comparable fuzzy sets have the same reference set. Given a sheaf representation $\mathscr{P}$ we can translate to classical fuzzy sets by setting $A = \bigcup_{a \in (0,1]} \mathscr{P}([0, a))$ and $\mu(x) = \sup\{a \in (0, 1] \mid x \in \mathscr{P}([0, a))\}$.

**Definition 10.** *The cross entropy $C$ of two fuzzy sets $(A, \mu)$ and $(A, \nu)$ is defined as*

$$C((A, \mu), (A, \nu)) \triangleq \sum_{a \in A} \left( \mu(a) \log \left( \frac{\mu(a)}{\nu(a)} \right) + (1 - \mu(a)) \log \left( \frac{1 - \mu(a)}{1 - \nu(a)} \right) \right).$$

Similar to t-SNE we can optimize the embedding $Y$ with respect to fuzzy set cross entropy $C$ by using stochastic gradient descent. However, this requires a differentiable fuzzy singular set functor. If the expected minimum distance between points is zero the fuzzy singular set functor is differentiable for these purposes, however for any non-zero value we need to make a differentiable approximation (chosen from a suitable family of differentiable functions).

This completes the algorithm: by using manifold approximation and patching together local fuzzy simplicial set representations we construct a topological representation of the high dimensional data. We then optimize the layout of data in a low dimensional space to minimize the error between the two topological representations.

We note that in this case we restricted attention to comparisons of the 1-skeleton of the fuzzy simplicial sets. One can extend this to $\ell$-skeleta by defining a cost function $C_\ell$ as

$$C_\ell(X, Y) = \sum_{i=1}^{\ell} \lambda_i C(X_i, Y_i),$$

where $X_i$ denotes the fuzzy set of $i$-simplices of $X$ and the $\lambda_i$ are suitably chosen real valued weights. While such an approach will capture the overall topological structure more accurately, it comes at non-negligible computational cost due to the increasingly large numbers of higher dimensional simplices. For this reason current implementations restrict to the 1-skeleton at this time.

# 3 A Computational View of UMAP

To understand what computations the UMAP algorithm is actually making from a practical point of view, a less theoretical and more computational description may be helpful for the reader. This description of the algorithm lacks the motivation for a number of the choices made. For that motivation please see Section 2.

The theoretical description of the algorithm works in terms of fuzzy simplicial sets. Computationally this is only tractable for the one skeleton which can ultimately be described as a weighted graph. This means that, from a practical computational perspective, UMAP can ultimately be described in terms of, construction of, and operations on weighted graphs. In particular this situates UMAP in the class of k-neighbour based graph learning algorithms such as Laplacian Eigenmaps, Isomap and t-SNE.

As with other k-neighbour graph based algorithms, UMAP can be described in two phases. In the first phase a particular weighted k-neighbour graph is constructed. In the second phase a low dimensional layout of this graph is computed. The differences between all algorithms in this class amount to specific details in how the graph is constructed and the layout is computed. The theoretical basis

for UMAP as described in Section 2 provides novel approaches to both of these phases.

Finally, since t-SNE is not usually described as a graph based algorithm, a direct comparison of UMAP with t-SNE, using the similarity/probability notation commonly used to express the equations of t-SNE, is given in the Appendix, section C.

## 3.1 Graph Construction

The first phase of UMAP can be thought of as the construction of a weighted k-neighbour graph. Let $X = \{x_1, \ldots, x_N\}$ be the input dataset, with a metric (or dissimilarity measure) $d : X \times X \to \mathbb{R}_{\geq 0}$. Given an input hyper-parameter $k$, for each $x_i$ we compute the set $\{x_{i_1}, \ldots, x_{i_k}\}$ of the $k$ nearest neighbors of $x_i$ under the metric $d$. This computation can be performed via any nearest neighbour or approximate nearest neighbour search algorithm. For the purposes of our UMAP implemenation we prefer to use the nearest neighbor descent algorithm of [16].

For each $x_i$ we will define $\rho_i$ and $\sigma_i$. Let

$$\rho_i = \min\{d(x_i, x_{i_j}) \mid 1 \leq j \leq k, d(x_i, x_{i_j}) > 0\},$$

and set $\sigma_i$ to be the value such that

$$\sum_{j=1}^{k} \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right) = \log_2(k).$$

We can now define a weighted directed graph $\bar{G} = (V, E, w)$. The vertices $V$ of $\bar{G}$ are simply the set $X$. We can then form the set of directed edges $E = \{(x_i, x_{i_j}) \mid 1 \leq j \leq k, 1 \leq i \leq N\}$, and define the weight function $w$ by setting

$$w((x_i, x_{i_j})) = \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right).$$

Let $A$ be the weighted adjacency matrix of $\bar{G}$, and consider the symmetric matrix

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

where $\circ$ is the Hadamard (or pointwise) product. The UMAP graph $G$ is then an undirected weighted graph whose adjacency matrix is given by $B$.

### 3.2 Graph Layout

In practice UMAP uses a force directed graph layout algorithm in low dimensional space. A force directed graph layout utilizes of a set of attractive forces applied along edges and a set of repulsive forces applied among vertices. Any force directed layout algorithm requires a description of both the attractive and repulsive forces. The algorithm proceeds by iteratively applying attractive and repulsive forces at each edge or vertex. Convergence is guaranteed by slowly decreasing the attractive and repulsive forces in a similar fashion to that used in simulated annealing.

In UMAP the attractive force between two vertices $i$ and $j$ at coordinates $\mathbf{y_i}$ and $\mathbf{y_j}$ respectively, is determined by:

$$\frac{-2ab\|\mathbf{y_i} - \mathbf{y_j}\|_2^{2(b-1)}}{1 + \|\mathbf{y_i} - \mathbf{y_j}\|_2^2} w((x_i, x_j))\,(\mathbf{y_i} - \mathbf{y_j})$$

where $a$ and $b$ are hyper-parameters.

Repulsive forces are computed via sampling due to computational constraints. Thus, whenever an attractive force is applied to an edge, one of that edge's vertices is repulsed by a sampling of other vertices. The repulsive force is given by

$$\frac{b}{(\epsilon + \|\mathbf{y_i} - \mathbf{y_j}\|_2^2)\,(1 + \|\mathbf{y_i} - \mathbf{y_j}\|_2^2)}\,(1 - w((x_i, x_j)))\,(\mathbf{y_i} - \mathbf{y_j})\,.$$

$\epsilon$ is a small number to prevent division by zero (0.001 in the current implementation).

The algorithm can be initialized randomly but in practice, since the symmetric Laplacian of the graph $G$ is a discrete approximation of the Laplace-Beltrami operator of the manifold, we can use a spectral layout to initialize the embedding. This provides both faster convergence and greater stability within the algorithm.

## 4 Implementation and Hyper-parameters

Having completed a theoretical description of the approach, we now turn our attention to the practical realization of this theory. We begin by providing a more detailed description of the algorithm as implemented, and then discuss a

few implementation specific details. We conclude this section with a discussion of the hyper-parameters for the algorithm and their practical effects.

## 4.1 Algorithm description

In overview the UMAP algorithm is relatively straightforward (see Algorithm 1). When performing a fuzzy union over local fuzzy simplicial sets we have found it most effective to work with the probabilistic t-conorm (as one would expect if treating membership strengths as a probability that the simplex exists). The individual functions for constructing the local fuzzy simplicial sets, determining the spectral embedding, and optimizing the embedding with regard to fuzzy set cross entropy, are described in more detail below.

**Algorithm 1** UMAP algorithm

**function** UMAP($X$, $n$, $d$, min-dist, n-epochs)
  **for all** $x \in X$ **do**
    fs-set$[x] \leftarrow$ LocalFuzzySimplicialSet($X$, $x$, $n$)
  top-rep $\leftarrow \bigcup_{x \in X}$ fs-set$[x]$  ▷ We recommend the probabilistic t-conorm
  $Y \leftarrow$ SpectralEmbedding(top-rep, $d$)
  $Y \leftarrow$ OptimizeEmbedding(top-rep, $Y$, min-dist, n-epochs)
  **return** $Y$

Algorithm 2 describes the construction of local fuzzy simplicial sets. To represent fuzzy simplicial sets we work with the fuzzy set images of $[0]$ and $[1]$ (i.e. the 1-skeleton), which we denote as fs-set$_0$ and fs-set$_1$. One can work with higher order simplices as well, but the current implementation does not. We can construct the fuzzy simplicial set local to a given point $x$ by finding the $n$ nearest neighbors, generating the appropriate normalised distance on the manifold, and then converting the finite metric space to a simplicial set via the functor FinSing, which translates into exponential of the negative distance in this case.

Rather than directly using the distance to the $n^{\text{th}}$ nearest neighbor as the normalization, we use a smoothed version of knn-distance that fixes the cardinality of the fuzzy set of 1-simplices to a fixed value. We selected $\log_2(n)$ for this purpose based on empirical experiments. This is described briefly in Algorithm 3.

Spectral embedding is performed by considering the 1-skeleton of the global fuzzy topological representation as a weighted graph and using standard spectral

**Algorithm 2** Constructing a local fuzzy simplicial set

**function** LocalFuzzySimplicialSet($X$, $x$, $n$)
knn, knn-dists $\leftarrow$ ApproxNearestNeighbors($X$, $x$, $n$)
$\rho \leftarrow$ knn-dists[1] ▷ Distance to nearest neighbor
$\sigma \leftarrow$ SmoothKNNDist(knn-dists, $n$, $\rho$) ▷ Smooth approximator to knn-distance
fs-set$_0 \leftarrow X$
fs-set$_1 \leftarrow \{([x, y], 0) \mid y \in X\}$
**for all** $y \in$ knn **do**
$d_{x,y} \leftarrow \max\{0, \text{dist}(x, y) - \rho\}/\sigma$
fs-set$_1 \leftarrow$ fs-set$_1 \cup ([x, y], \exp(-d_{x,y}))$
**return** fs-set

**Algorithm 3** Compute the normalizing factor for distances $\sigma$

**function** SmoothKNNDist(knn-dists, $n$, $\rho$)
Binary search for $\sigma$ such that $\sum_{i=1}^{n} \exp(-(\text{knn-dists}_i - \rho)/\sigma) = \log_2(n)$
**return** $\sigma$

methods on the symmetric normalized Laplacian. This process is described in Algorithm 4.

**Algorithm 4** Spectral embedding for initialization

**function** SpectralEmbedding(top-rep, $d$)
$A \leftarrow$ 1-skeleton of top-rep expressed as a weighted adjacency matrix
$D \leftarrow$ degree matrix for the graph $A$
$L \leftarrow D^{1/2}(D - A)D^{1/2}$
evec $\leftarrow$ Eigenvectors of $L$ (sorted)
$Y \leftarrow$ evec$[1..d + 1]$ ▷ 0-base indexing assumed
**return** $Y$

The final major component of UMAP is the optimization of the embedding through minimization of the fuzzy set cross entropy. Recall that fuzzy set cross

entropy, with respect given membership functions $\mu$ and $\nu$, is given by

$$\begin{aligned} C((A,\mu),(A,\nu)) &= \sum_{a\in A} \mu(a)\log\left(\frac{\mu(a)}{\nu(a)}\right) + (1-\mu(a))\log\left(\frac{1-\mu(a)}{1-\nu(a)}\right) \\ &= \sum_{a\in A} \left(\mu(a)\log(\mu(a)) + (1-\mu(a))\log(1-\mu(a))\right) \\ &\qquad - \sum_{a\in A} \left(\mu(a)\log(\nu(a)) + (1-\mu(a))\log(1-\nu(a))\right). \end{aligned} \tag{1}$$

The first sum depends only on $\mu$ which takes fixed values during the optimization, thus the minimization of cross entropy depends only on the second sum, so we seek to minimize

$$-\sum_{a\in A} \left(\mu(a)\log(\nu(a)) + (1-\mu(a))\log(1-\nu(a))\right).$$

Following both [45] and [33], we take a sampling based approach to the optimization. We sample 1-simplices with probability $\mu(a)$ and update according to the value of $\nu(a)$, which handles the term $\mu(a)\log(\nu(a))$. The term $(1-\mu(a))\log(1-\nu(a))$ requires negative sampling – rather than computing this over all potential simplices we randomly sample potential 1-simplices and assume them to be a negative example (i.e. with membership strength 0) and update according to the value of $1-\nu(a)$. In contrast to [45] the above formulation provides a vertex sampling distribution of

$$P(x_i) = \frac{\sum_{\{a\in A | d_0(a)=x_i\}} 1-\mu(a)}{\sum_{\{b\in A | d_0(b)\neq x_i\}} 1-\mu(b)}$$

for negative samples, which can be reasonably approximated by a uniform distribution for sufficiently large data sets.

It therefore only remains to find a differentiable approximation to $\nu(a)$ for a given 1-simplex $a$ so that gradient descent cane be applied for optimization. This is done as follows:

**Definition 11.** *Define $\Phi : \mathbb{R}^d \times \mathbb{R}^d \to [0,1]$, a smooth approximation of the membership strength of a 1-simplex between two points in $\mathbb{R}^d$, as*

$$\Phi(\mathbf{x},\mathbf{y}) = \left(1 + a(\|\mathbf{x}-\mathbf{y}\|_2^2)^b\right)^{-1},$$

*where $a$ and $b$ are chosen by non-linear least squares fitting of $\Psi : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$ where*

$$\Psi(\mathbf{x}, \mathbf{y}) \begin{cases} 1 & \textit{if } \|\mathbf{x} - \mathbf{y}\|_2 \leq \textit{min-dist} \\ \exp(-(\|\mathbf{x} - \mathbf{y}\|_2 - \textit{min-dist})) & \textit{otherwise} \end{cases}.$$

The optimization process is now executed by stochastic gradient descent as given by Algorithm 5.

**Algorithm 5** Optimizing the embedding

**function** OPTIMIZEEMBEDDING(top-rep, $Y$, min-dist, n-epochs)
  $\alpha \leftarrow 1.0$
  Fit $\Phi$ from $\Psi$ defined by min-dist
  **for** $e \leftarrow 1, \ldots,$ n-epochs **do**
    **for all** $([a, b], p) \in$ top-rep$_1$ **do**
      **if** RANDOM( ) $\leq p$ **then** ▷ Sample simplex with probability $p$
        $y_a \leftarrow y_a + \alpha \cdot \nabla(\log(\Phi))(y_a, y_b)$
        **for** $i \leftarrow 1, \ldots,$ n-neg-samples **do**
          $c \leftarrow$ random sample from Y
          $y_a \leftarrow y_a + \alpha \cdot \nabla(\log(1 - \Phi))(y_a, y_c)$
    $\alpha \leftarrow 1.0 - e/$n-epochs
  **return** $Y$

This completes the UMAP algorithm.

## 4.2 Implementation

Practical implementation of this algorithm requires (approximate) $k$-nearest-neighbor calculation and efficient optimization via stochastic gradient descent.

Efficient approximate $k$-nearest-neighbor computation can be achieved via the Nearest-Neighbor-Descent algorithm of [16]. The error intrinsic in a dimension reduction technique means that such approximation is more than adequate for these purposes. While no theoretical complexity bounds have been established for Nearest-Neighbor-Descent the authors of the original paper report an empirical complexity of $O(N^{1.14})$. A further benefit of Nearest-Neighbor-Descent is its generality; it works with any valid dissimilarity measure, and is efficient even for high dimensional data.

In optimizing the embedding under the provided objective function, we follow work of [45]; making use of probabilistic edge sampling and negative sampling [33]. This provides a very efficient approximate stochastic gradient descent algorithm since there is no normalization requirement. Furthermore, since the normalized Laplacian of the fuzzy graph representation of the input data is a discrete approximation of the Laplace-Betrami operator of the manifold [**?**, see]]belkin2002laplacian, belkin2003laplacian, we can provide a suitable initialization for stochastic gradient descent by using the eigenvectors of the normalized Laplacian. The amount of optimization work required will scale with the number of edges in the fuzzy graph (assuming a fixed negative sampling rate), resulting in a complexity of $O(kN)$.

Combining these techniques results in highly efficient embeddings, which we will discuss in Section 5. The overall complexity is bounded by the approximate nearest neighbor search complexity and, as mentioned above, is empirically approximately $O(N^{1.14})$. A reference implementation can be found at `https://github.com/lmcinnes/umap`, and an R implementation can be found at `https://github.com/jlmelville/uwot`.

While our reference implementation is single core for simplicity it should be noted that both Nearest-Neighbor-Descent and SGD can be parallelised. A parallel multi-core implementation of UMAP is therefore achievable.

## 4.3 Hyper-parameters

As described in Algorithm 1, the UMAP algorithm takes four hyper-parameters:

1. $n$, the number of neighbors to consider when approximating the local metric;
2. $d$, the target embedding dimension;
3. min-dist, the desired separation between close points in the embedding space; and
4. n-epochs, the number of training epochs to use when optimizing the low dimensional representation.

The effects of the parameters $d$ and n-epochs are largely self-evident, and will not be discussed in further detail here. In contrast the effects of the number of neighbors $n$ and of min-dist are less clear.

One can interpret the number of neighbors $n$ as the local scale at which to approximate the manifold as roughly flat, with the manifold estimation averaging over the $n$ neighbors. Manifold features that occur at a smaller scale than within the $n$ nearest-neighbors of points will be lost, while large scale manifold features that cannot be seen by patching together locally flat charts at the scale of $n$ nearest-neighbors may not be well detected. Thus $n$ represents some degree of trade-off between fine grained and large scale manifold features — smaller values will ensure detailed manifold structure is accurately captured (at a loss of the "big picture" view of the manifold), while larger values will capture large scale manifold structures, but at a loss of fine detail structure which will get averaged out in the local approximations. With smaller $n$ values the manifold tends to be broken into many small connected components (care needs to be taken with the spectral embedding for initialization in such cases).

In contrast min-dist is a hyperparameter directly affecting the output, as it controls the fuzzy simplicial set construction from the low dimensional representation. It acts in lieu of the distance to the nearest neighbor used to ensure local connectivity. In essence this determines how closely points can be packed together in the low dimensional representation. Low values on min-dist will result in potentially densely packed regions, but will likely more faithfully represent the manifold structure. Increasing the value of min-dist will force the embedding to spread points out more, assisting visualization (and avoiding potential overplotting issues). We view min-dist as an essentially aesthetic parameter, governing the appearance of the embedding, and thus is more important when using UMAP for visualization.

In Figure 1 we provide examples of the effects of varying the hyperparameters for a toy dataset. The data is uniform random samples from a 3-dimensional color-cube, allowing for easy visualization of the original 3-dimensional coordinates in the embedding space by using the corresponding RGB colour. Since the data fills a 3-dimensional cube there is no local manifold structure, and hence for such data we expect larger $n$ values to be more useful. Low values will interpret the noise from random sampling as fine scale manifold structure, producing potentially spurious structure[1].

[1]See the discussion of the constellation effect in Section 6

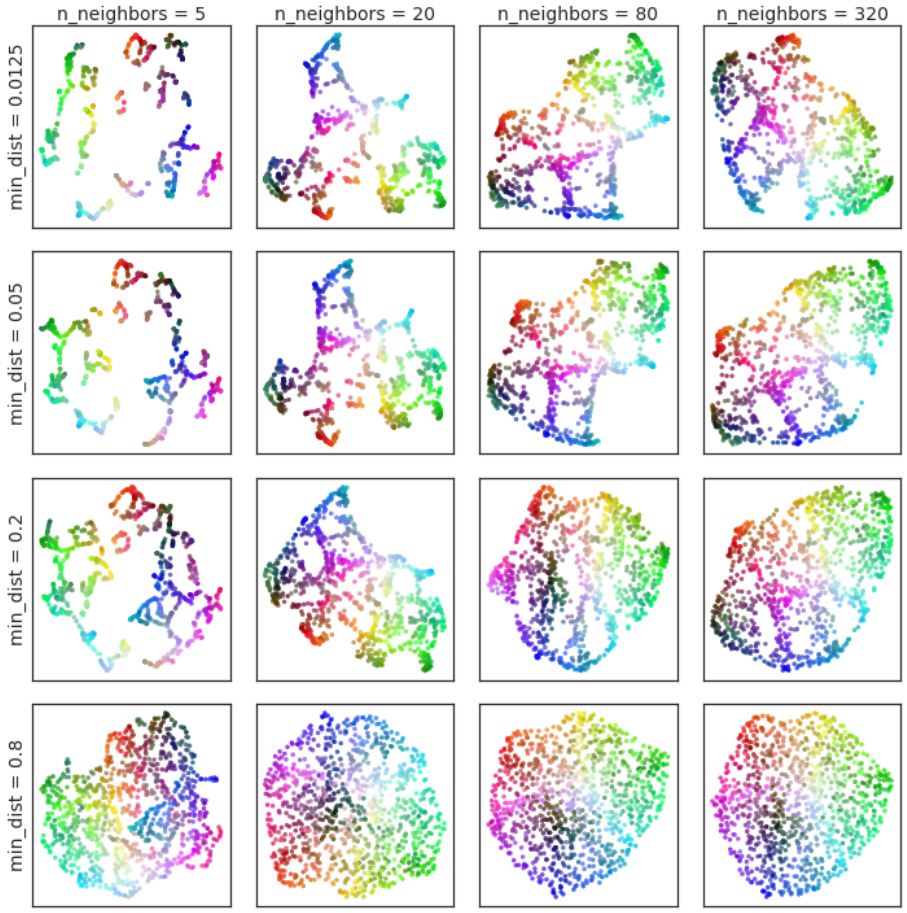



Figure 1: Variation of UMAP hyperparameters $n$ and min-dist result in different embeddings. The data is uniform random samples from a 3-dimensional color-cube, allowing for easy visualization of the original 3-dimensional coordinates in the embedding space by using the corresponding RGB colour. Low values of $n$ spuriously interpret structure from the random sampling noise – see Section 6 for further discussion of this phenomena.

# 5 Practical Efficacy

While the strong mathematical foundations of UMAP were the motivation for its development, the algorithm must ultimately be judged by its practical efficacy. In this section we examine the fidelity and performance of low dimensional embeddings of multiple diverse real world data sets under UMAP. The following datasets were considered:

**Pen digits** [1, 10] is a set of 1797 grayscale images of digits entered using a digitiser tablet. Each image is an 8x8 image which we treat as a single 64 dimensional vector, assumed to be in Euclidean vector space.

**COIL 20** [34] is a set of 1440 greyscale images consisting of 20 objects under 72 different rotations spanning 360 degrees. Each image is a 128x128 image which we treat as a single 16384 dimensional vector for the purposes of computing distance between images.

**COIL 100** [35] is a set of 7200 colour images consisting of 100 objects under 72 different rotations spanning 360 degrees. Each image consists of 3 128x128 intensity matrices (one for each color channel). We treat this as a single 49152 dimensional vector for the purposes of computing distance between images.

**Mouse scRNA-seq** [11] is profiled gene expression data for 20,921 cells from an adult mouse. Each sample consists of a vector of 26,774 measurements.

**Statlog (Shuttle)** [28] is a NASA dataset consisting of various data associated to the positions of radiators in the space shuttle, including a timestamp. The dataset has 58000 points in a 9 dimensional feature space.

**MNIST** [25] is a dataset of 28x28 pixel grayscale images of handwritten digits. There are 10 digit classes (0 through 9) and 70000 total images. This is treated as 70000 different 784 dimensional vectors.

**F-MNIST** [53] or Fashion MNIST is a dataset of 28x28 pixel grayscale images of fashion items (clothing, footwear and bags). There are 10 classes and 70000 total images. As with MNIST this is treated as 70000 different 784 dimensional vectors.

**Flow cytometry** [42, 9] is a dataset of flow cytometry measurements of CDT4 cells comprised of 1,000,000 samples, each with 17 measurements.

**GoogleNews word vectors** [33] is a dataset of 3 million words and phrases derived from a sample of Google News documents and embedded into a 300 dimensional space via word2vec.

For all the datasets except GoogleNews we use Euclidean distance between vectors. For GoogleNews, as per [33], we use cosine distance (or angular distance

in t-SNE which does support non-metric distances, in contrast to UMAP).

## 5.1 Qualitative Comparison of Multiple Algorithms

We compare a number of algorithms–UMAP, t-SNE [51, 49], LargeVis [45], Laplacian Eigenmaps [6], and Principal Component Analysis [22]–on the COIL20 [34], MNIST [25], Fashion-MNIST [53], and GoogleNews [33] datasets. The Isomap algorithm was also tested, but failed to complete in any reasonable time for any of the datasets larger than COIL20.

The Multicore t-SNE package [48] was used for t-SNE. The reference implementation [44] was used for LarveVis. The scikit-learn [10] implementations were used for Laplacian Eigenmaps and PCA. Where possible we attempted to tune parameters for each algorithm to give good embeddings.

Historically t-SNE and LargeVis have offered a dramatic improvement in finding and preserving local structure in the data. This can be seen qualitatively by comparing their embeddings to those generated by Laplacian Eigenmaps and PCA in 2. We claim that the quality of embeddings produced by UMAP is comparable to t-SNE when reducing to two or three dimensions. For example, Figure 2 shows both UMAP and t-SNE embeddings of the COIL20, MNIST, Fashion MNIST, and Google News datasets. While the precise embeddings are different, UMAP distinguishes the same structures as t-SNE and LargeVis.

It can be argued that UMAP has captured more of the global and topological structure of the datasets than t-SNE [4]. More of the loops in the COIL20 dataset are kept intact, including the intertwined loops. Similarly the global relationships among different digits in the MNIST digits dataset are more clearly captured with 1 (red) and 0 (dark red) at far corners of the embedding space, and 4,7,9 (yellow, sea-green, and violet) and 3,5,8 (orange, chartreuse, and blue) separated as distinct clumps of similar digits. In the Fashion MNIST dataset the distinction between clothing (dark red, yellow, orange, vermilion) and footwear (chartreuse, sea-green, and violet) is made more clear. Finally, while both t-SNE and UMAP capture groups of similar word vectors, the UMAP embedding arguably evidences a clearer global structure among the various word clusters.

## 5.2 Embedding Stability

Since UMAP makes use of both stochastic approximate nearest neighbor search, and stochastic gradient descent with negative sampling for optimization, the

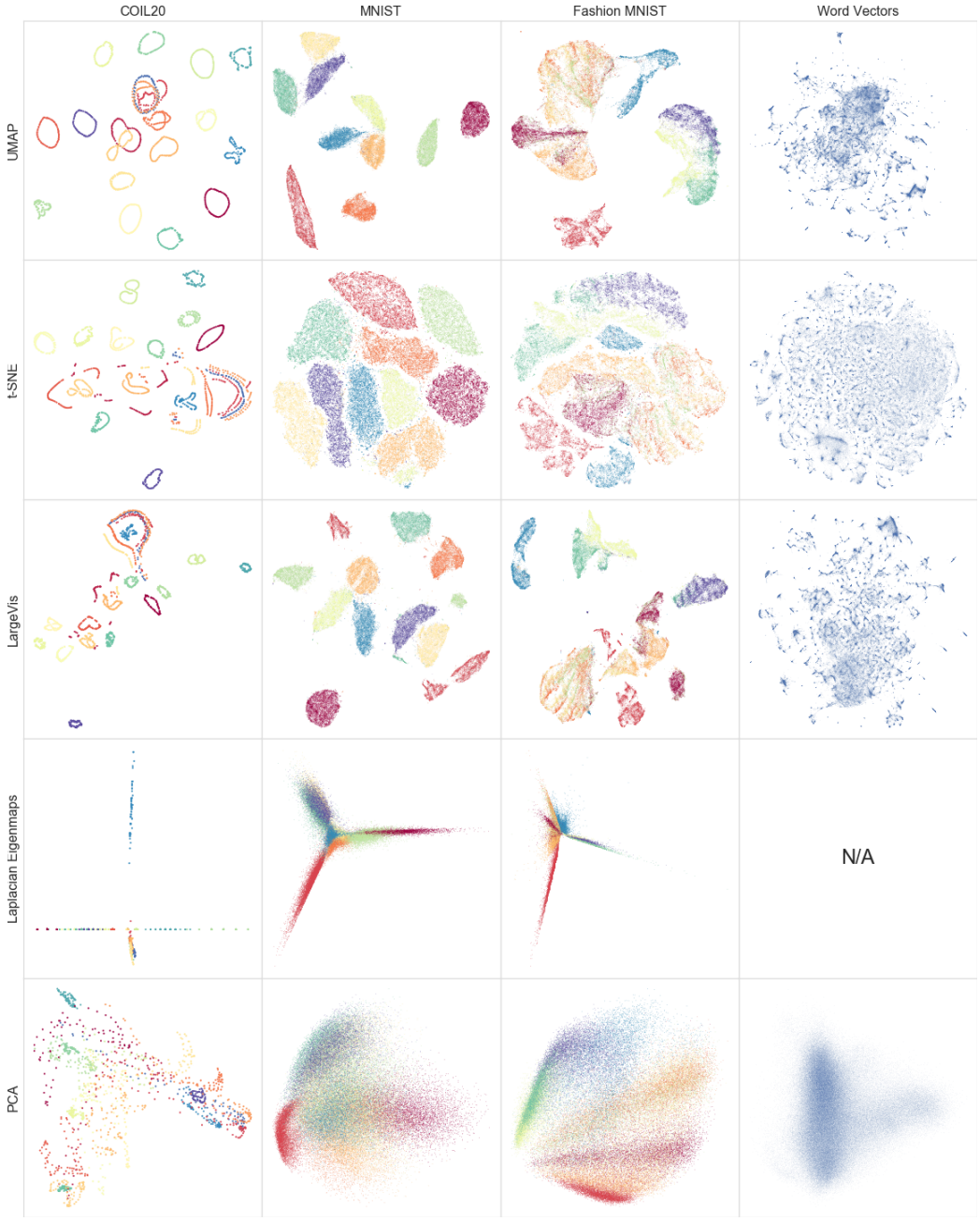



Figure 2: A comparison of several dimension reduction algorithms. We note that UMAP successfully reflects much of the large scale global structure that is well represented by Laplacian Eigenmaps and PCA (particularly for MNIST and Fashion-MNIST), while also preserving the local fine structure similar to t-SNE and LargeVis.

resulting embedding is necessarily different from run to run, and under subsampling of the data. This is potentially a concern for a variety of uses cases, so establishing some measure of how stable UMAP embeddings are, particularly under sub-sampling, is of interest. In this subsection we compare the stability under subsampling of UMAP, LargeVis and t-SNE (the three stochastic dimension reduction techniques considered).

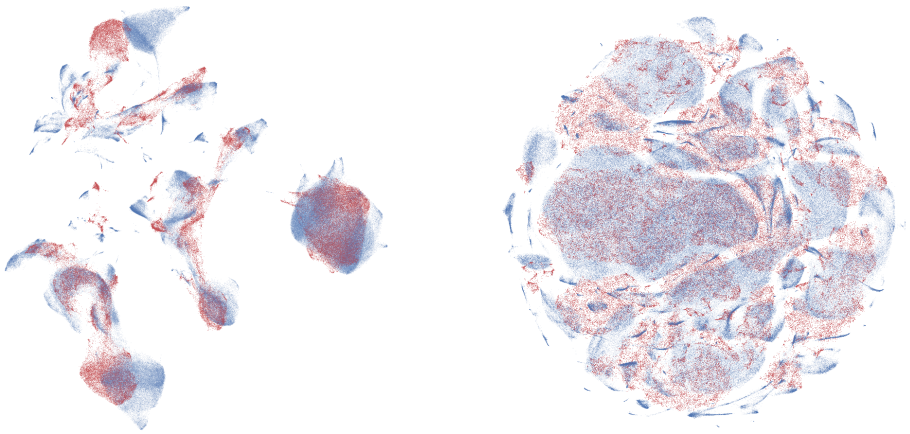
To measure the stability of an embedding we make use of the normalized Procrustes distance to measure the distance between two potentially comparable distributions. Given two datasets $X = \{x_1, \ldots, x_N\}$ and $Y = \{y_1, \ldots, y_N\}$ such that $x_i$ corresponds to $y_i$, we can define the Procustes distance between the datasets $d_P(X, Y)$ in the following manner. Determine $Y' = \{y_1', \ldots, y_N'\}$ the optimal translation, uniform scaling, and rotation of $Y$ that minimizes the squared error $\sum_{i=1}^{N}(x_i - y_i')^2$, and define

$$d_P(X, Y) = \sqrt{\sum_{i=1}^{N}(x_i - y_i')^2}.$$

Since any measure that makes use of distances in the embedding space is potentially sensitive to the extent or scale of the embedding, we normalize the data before computing the Procrustes distance by dividing by the average norm of the embedded dataset. In Figure 3 we visualize the results of using Procrustes alignment of embedding of sub-samples for both UMAP and t-SNE, demonstrating how Procrustes distance can measure the stability of the overall structure of the embedding.

Given a measure of distance between different embeddings we can examine stability under sub-sampling by considering the normalized Procrustes distance between the embedding of a sub-sample, and the corresponding sub-sample of an embedding of the full dataset. As the size of the sub-sample increases the average distance per point between the sub-sampled embeddings should decrease, potentially toward some asymptote of maximal agreement under repeated runs. Ideally this asymptotic value would be zero error, but for stochastic embeddings such as UMAP and t-SNE this is not achievable.

We performed an empirical comparison of algorithms with respect to stability using the Flow Cytometry dataset due its large size, interesting structure, and low ambient dimensionality (aiding runtime performance for t-SNE). We note that for a dataset this large we found it necessary to increase the default `n_iter` value for t-SNE from 1000 to 1500 to ensure better convergence. While this had



(a) UMAP (b) t-SNE

Figure 3: Procrustes based alignment of a 10% subsample (red) against the full dataset (blue) for the flow cytometry dataset for both UMAP and t-SNE.

an impact on the runtime, it significantly improved the Procrustes distance results by providing more stable and consistent embeddings. Figure 4 provides a comparison between UMAP and t-SNE, demonstrating that UMAP has significantly more stable results than t-SNE. In particular, after sub-sampling on 5% of the million data points, the per point error for UMAP was already below any value achieved by t-SNE.

## 5.3 Computational Performance Comparisons

Benchmarks against the real world datasets were performed on a Macbook Pro with a 3.1 GHz Intel Core i7 and 8GB of RAM for Table 1, and on a server with Intel Xeon E5-2697v4 processors and 512GB of RAM for the large scale benchmarking in Subsections 5.3.1, 5.3.2, and 5.3.3.

For t-SNE we chose MulticoreTSNE [48], which we believe to be the fastest extant implementation of Barnes-Hut t-SNE at this time, even when run in single core mode. It should be noted that MulticoreTSNE is a heavily optimized implementation written in C++ based on Van der Maaten's `bhtsne` [49] code.

As a fast alternative approach to t-SNE we also consider the FIt-SNE algorithm [30]. We used the reference implementation [29], which, like Multi-

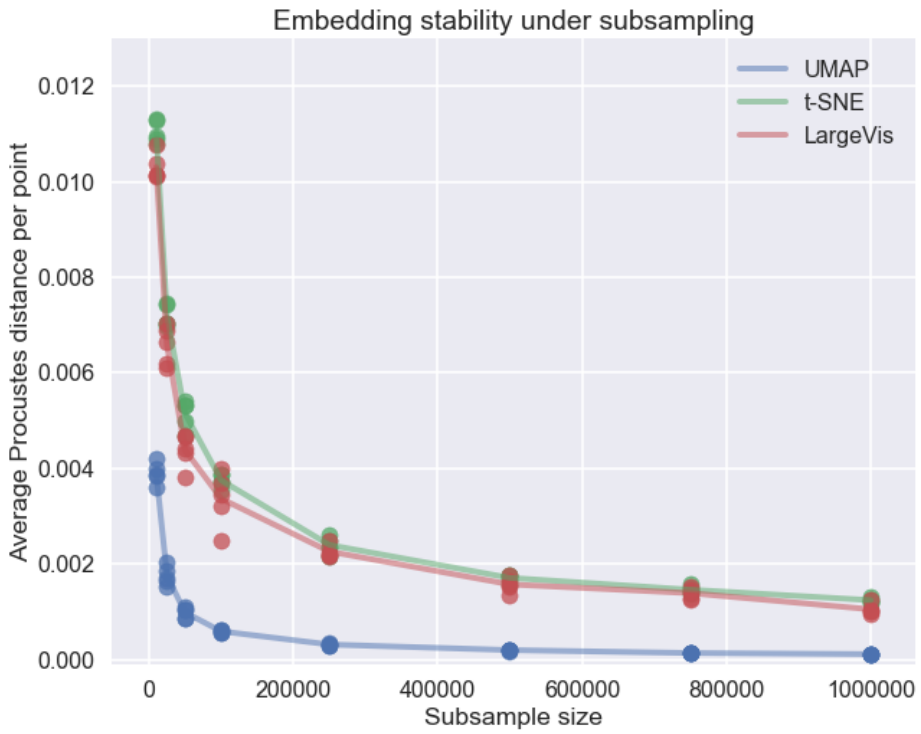



Figure 4: Comparison of average Procustes distance per point for t-SNE, LargeVis and UMAP over a variety of sizes of subsamples from the full Flow Cytometry dataset. UMAP sub-sample embeddings are very close to the full embedding even for subsamples of 5% of the full dataset, outperforming the results of t-SNE and LargeVis even when they use the full Flow Cytometry dataset.

coreTNSE is an optimized C++ implementation. We also note that FIt-SNE makes use of multiple cores.

LargeVis [45] was benchmarked using the reference implementation [44]. It was run with default parameters including use of 8 threads on the 4-core machine. The only exceptions were small datasets where we explicitly set the `-samples` parameter to `n_samples`/100 as per the recommended values in the documentation of the reference implementation.

The Isomap [46] and Laplacian Eigenmaps [6] implementations in scikit-learn [10] were used. We suspect the Laplacian eigenmaps implementation may not be well optimized for large datasets but did not find a better performing implementation that provided comparable quality results. Isomap failed to complete for the Shuttle, Fashion-MNIST, MNIST and GoogleNews datasets, while Laplacian Eigenmaps failed to run for the GoogleNews dataset.

To allow a broader range of algorithms to run some of the datasets where subsampled or had their dimension reduced by PCA. The Flow Cytometry dataset was benchmarked on a 10% sample and the GoogleNews was subsampled down to 200,000 data points. Finally, the Mouse scRNA dataset was reduced to 1,000 dimensions via PCA.

Timing were performed for the COIL20 [34], COIL100 [35], Shuttle [28], MNIST [25], Fashion-MNIST [53], and GoogleNews [33] datasets. Results can be seen in Table 1. UMAP consistently performs faster than any of the other algorithms aside from on the very small Pendigits dataset, where Laplacian Eigenmaps and Isomap have a small edge.

### 5.3.1 Scaling with Embedding Dimension

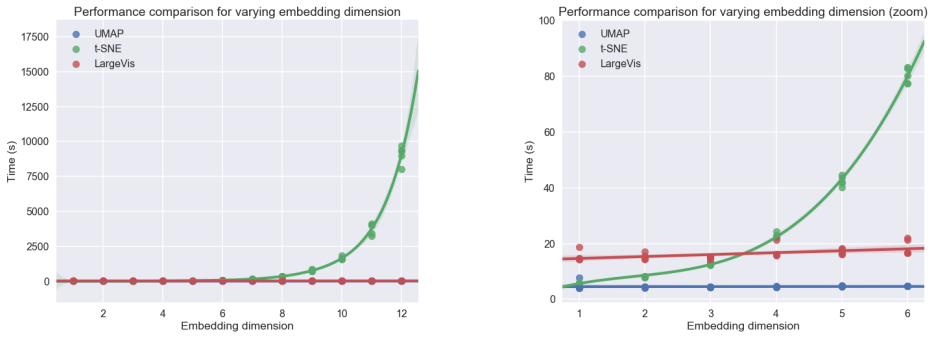
UMAP is significantly more performant than t-SNE[2] when embedding into dimensions larger than 2. This is particularly important when the intention is to use the low dimensional representation for further machine learning tasks such as clustering or anomaly detection rather than merely for visualization. The computation performance of UMAP is far more efficient than t-SNE, even for very small embedding dimensions of 6 or 8 (see Figure 5). This is largely due to the fact that UMAP does not require global normalisation. This allows the algorithm to work without the need for space trees —such as the quad-trees and oct-trees that t-SNE uses [49]—. Such space trees scale exponentially in dimension, resulting in t-SNE's relatively poor scaling with respect to embedding dimension.

[2]Comparisons were performed against MulticoreTSNE as the current implementation of FIt-SNE does not support embedding into any dimension larger than 2.

| | UMAP | FIt-SNE | t-SNE | LargeVis | Eigenmaps | Isomap |
|---|---|---|---|---|---|---|
| **Pen Digits** (1797x64) | 9s | 48s | 17s | 20s | **2s** | **2s** |
| **COIL20** (1440x16384) | **12s** | 75s | 22s | 82s | 47s | 58s |
| **COIL100** (7200x49152) | **85s** | 2681s | 810s | 3197s | 3268s | 3210s |
| **scRNA** (21086x1000) | **28s** | 131s | 258s | 377s | 470s | 923s |
| **Shuttle** (58000x9) | **94s** | 108s | 714s | 615s | 133s | – |
| **MNIST** (70000x784) | **87s** | 292s | 1450s | 1298s | 40709s | – |
| **F-MNIST** (70000x784) | **65s** | 278s | 934s | 1173s | 6356s | – |
| **Flow** (100000x17) | **102s** | 164s | 1135s | 1127s | 30654s | – |
| **Google News** (200000x300) | **361s** | 652s | 16906s | 5392s | – | – |

Table 1: Runtime of several dimension reduction algorithms on various datasets. To allow a broader range of algorithms to run some of the datasets where subsampled or had their dimension reduced by PCA. The Flow Cytometry dataset was benchmarked on a 10% sample and the GoogleNews was subsampled down to 200,000 data points. Finally, the Mouse scRNA dataset was reduced to 1,000 dimensions via PCA. The fastest runtime for each dataset has been bolded.

By contrast, we see that UMAP consistently scales well in embedding dimension, making the algorithm practical for a wider range of applications beyond visualization.




(a) A comparison of run time for UMAP, t-SNE and LargeVis with respect to embedding dimension on the Pen digits dataset. We see that t-SNE scales worse than exponentially while UMAP and LargeVis scale linearly with a slope so slight to be undetectable at this scale.

(b) Detail of scaling for embedding dimension of six or less. Wessentially e can see that UMAP and LargeVis are essentially flat. In practice they appear to scale linearly, but the slope is essentially undetectable at this scale.

Figure 5: Scaling performance with respect to embedding dimension of UMAP, t-SNE and LargeVis on the Pen digits dataset.

### 5.3.2 Scaling with Ambient Dimension

Through a combination of the local-connectivity constraint and the approximate nearest neighbor search, UMAP can perform effective dimension reduction even for very high dimensional data (see Figure 9 for an example of UMAP operating directly on 1.8 million dimensional data). This stands in contrast to many other manifold learning techniques, including t-SNE and LargeVis, for which it is generally recommended to reduce the dimension with PCA before applying these techniques (see [50] for example).

To compare runtime performance scaling with respect to the ambient dimension of the data we chose to use the Mouse scRNA dataset, which is high dimensional, but is also amenable to the use of PCA to reduce the dimension of the data

as a pre-processing step without losing too much of the important structure[3]. We compare the performance of UMAP, FIt-SNE, MulticoreTSNE, and LargeVis on PCA reductions of the Mouse scRNA dataset to varying dimensionalities, and on the original dataset, in Figure 6.

While all the implementations tested show a significant increase in runtime with increasing dimension, UMAP is dramatically more efficient for large ambient dimensions, easily scaling to run on the original unreduced dataset. The ability to run manifold learning on raw source data, rather than dimension reduced data that may have lost important manifold structure in the pre-processing, is a significant advantage.

Since UMAP scales well with ambient dimension the python implementation also supports input in sparse matrix format, allowing scaling to extremely high dimensional data, such as the integer data shown in Figures 9 and 10.

### 5.3.3 Scaling with the Number of Samples

For dataset size performance comparisons we chose to compare UMAP with FIt-SNE [30], a version of t-SNE that uses approximate nearest neighbor search and a Fourier interpolation optimisation approach; MulticoreTSNE [48], which we believe to be the fastest extant implementation of Barnes-Hut t-SNE; and LargeVis [45]. It should be noted that FIt-SNE, MulticoreTSNE, and LargeVis are all heavily optimized implementations written in C++. In contrast our UMAP implementation was written in Python — making use of the numba [24] library for performance. MulticoreTSNE and LargeVis were run in single threaded mode to make fair comparisons to our single threaded UMAP implementation.

We benchmarked all four implementations using subsamples of the GoogleNews dataset. The results can be seen in Figure 7. This demonstrates that UMAP has superior scaling performance in comparison to Barnes-Hut t-SNE, even when Barnes-Hut t-SNE is given multiple cores. Asymptotic scaling of UMAP is comparable to that of FIt-SNE (and LargeVis). On this dataset UMAP demonstrated somewhat faster absolute performance compared to FIt-SNE, and was dramatically faster than LargeVis.

The UMAP embedding of the full GoogleNews dataset of 3 million word vectors, as seen in Figure 8, was completed in around 200 minutes, as compared with several days required for MulticoreTSNE, even using multiple cores.

To scale even further we were inspired by the work of John Williamson on

[3] In contrast to COIL100, on which PCA destroys much of the manifold structure

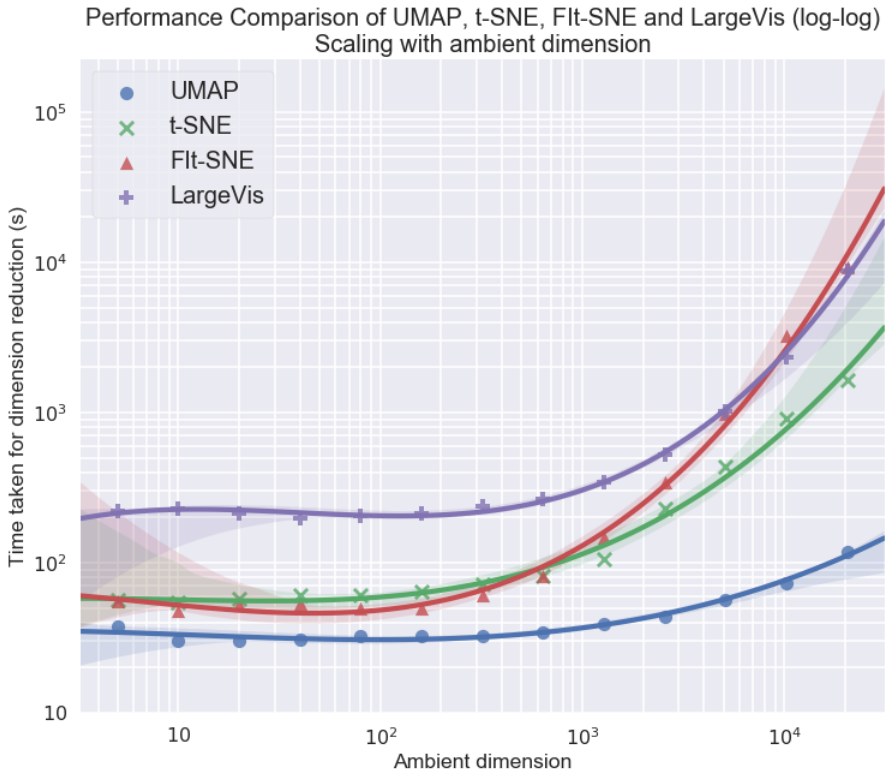



Figure 6: Runtime performance scaling of UMAP, t-SNE, FIt-SNE and Largevis with respect to the ambient dimension of the data. As the ambient dimension increases beyond a few thousand dimensions the computational cost of t-SNE, FIt-SNE, and LargeVis all increase dramatically, while UMAP continues to perform well into the tens-of-thousands of dimensions.

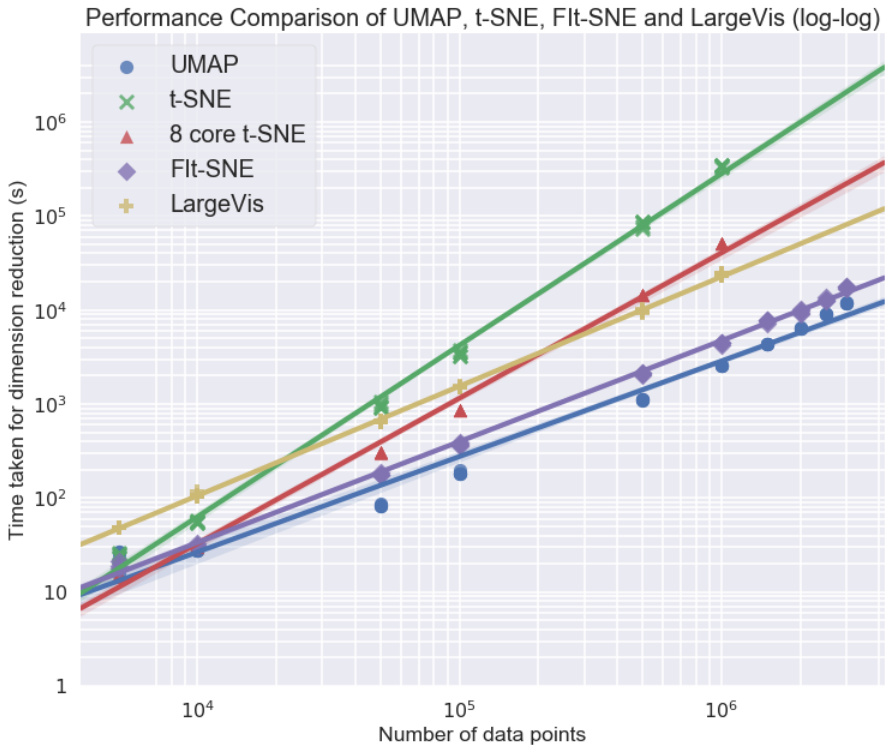



Figure 7: Runtime performance scaling of t-SNE and UMAP on various sized subsamples of the full Google News dataset. The lower t-SNE line is the wall clock runtime for Multicore t-SNE using 8 cores.

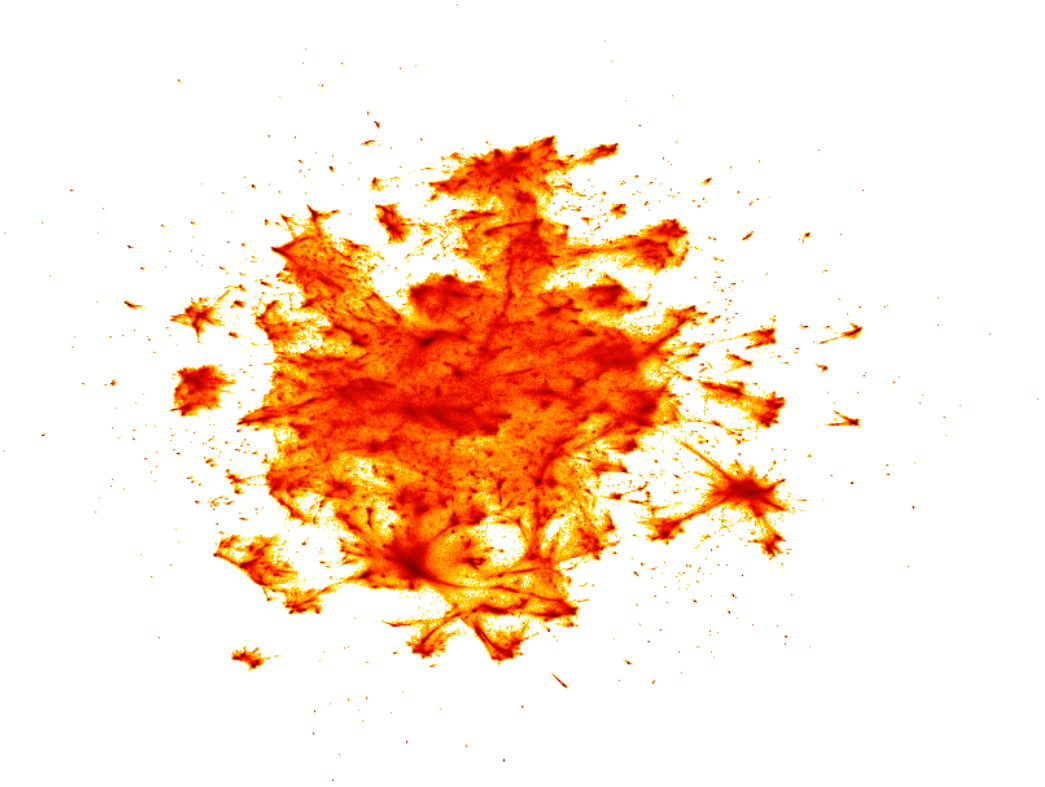


Figure 8: Visualization of the full 3 million word vectors from the GoogleNews dataset as embedded by UMAP.

embedding integers [52], as represented by (sparse) binary vectors of their prime divisibility. This allows the generation of arbitrarily large, extremely high dimension datasets that still have meaningful structure to be explored. In Figures 9 and 10 we show an embedding of 30,000,000 data samples from an ambient space of approximately 1.8 million dimensions. This computation took approximately 2 weeks on a large memory SMP. Note that despite the high ambient dimension, and vast amount of data, UMAP is still able to find and display interesting structure. In Figure 11 we show local regions of the embedding, demonstrating the fine detail structure that was captured.

# 6 Weaknesses

While we believe UMAP to be a very effective algorithm for both visualization and dimension reduction, most algorithms must make trade-offs and UMAP is no exception. In this section we will briefly discuss those areas or use cases where UMAP is less effective, and suggest potential alternatives.

For a number of uses cases the interpretability of the reduced dimension results is of critical importance. Similarly to most non-linear dimension reduction techniques (including t-SNE and Isomap), UMAP lacks the strong interpretability of Principal Component Analysis (PCA) and related techniques such a Non-Negative Matrix Factorization (NMF). In particular the dimensions of the UMAP embedding space have no specific meaning, unlike PCA where the dimensions are the directions of greatest variance in the source data. Furthermore, since UMAP is based on the distance between observations rather than the source features, it does not have an equivalent of factor loadings that linear techniques such as PCA, or Factor Analysis can provide. If strong interpretability is critical we therefore recommend linear techniques such as PCA and NMF.

One of the core assumptions of UMAP is that there exists manifold structure in the data. Because of this UMAP can tend to find manifold structure within the noise of a dataset – similar to the way the human mind finds structured constellations among the stars. As more data is sampled the amount of structure evident from noise will tend to decrease and UMAP becomes more robust, however care must be taken with small sample sizes of noisy data, or data with only large scale manifold structure. Detecting when a spurious embedding has occurred is a topic of further research.

UMAP is derived from the axiom that local distance is of more importance than long range distances (similar to techniques like t-SNE and LargeVis). UMAP

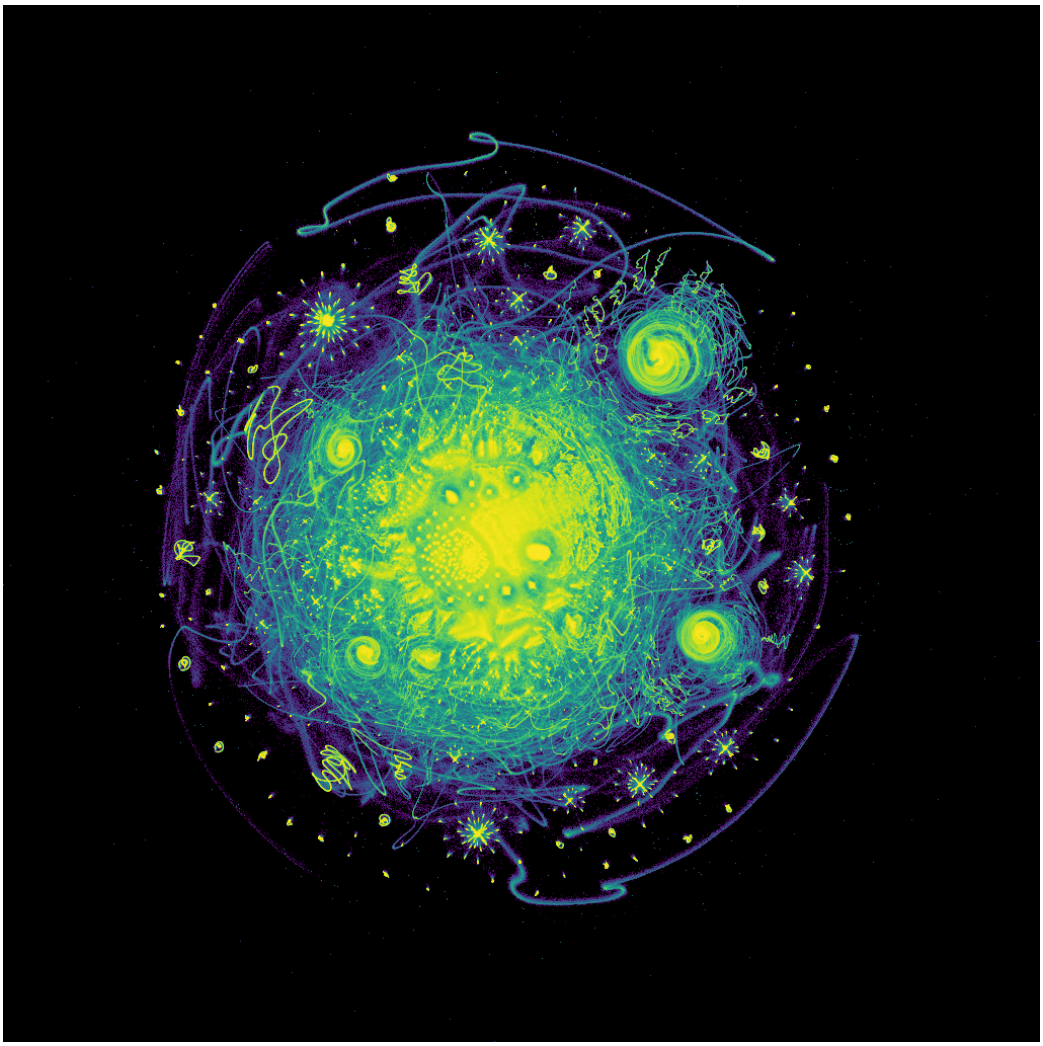


Figure 9: Visualization of 30,000,000 integers as represented by binary vectors of prime divisibility, colored by density of points.

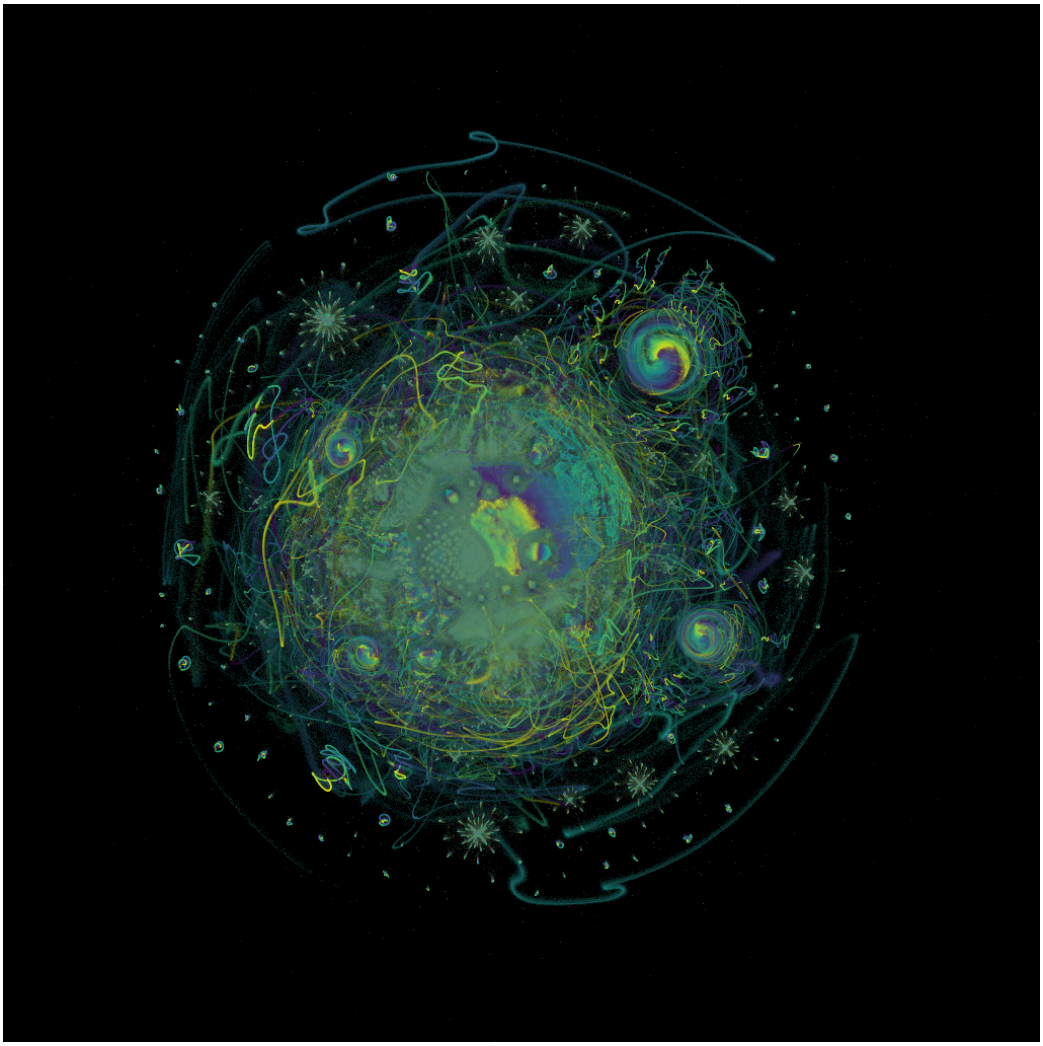
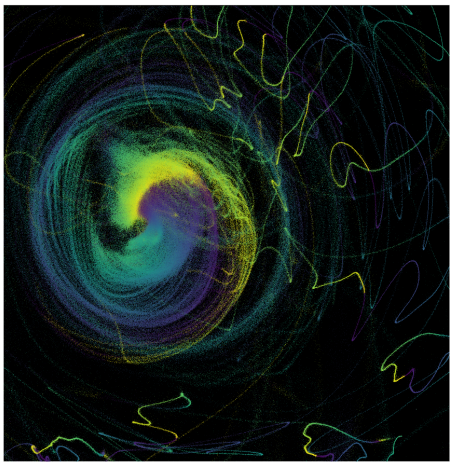
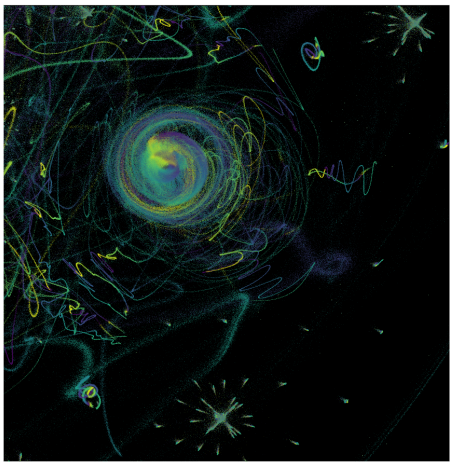


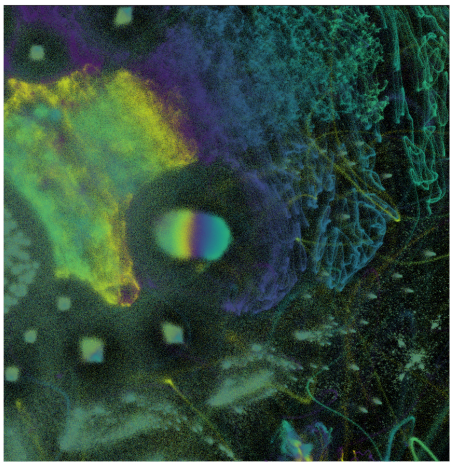
Figure 10: Visualization of 30,000,000 integers as represented by binary vectors of prime divisibility, colored by integer value of the point (larger values are green or yellow, smaller values are blue or purple).



(a) Upper right spiral



(b) Lower right spiral and starbursts



(c) Central cloud

Figure 11: Zooming in on various regions of the integer embedding reveals further layers of fine structure have been preserved.

therefore concerns itself primarily with accurately representing local structure. While we believe that UMAP can capture more global structure than these other techniques, it remains true that if global structure is of primary interest then UMAP may not be the best choice for dimension reduction.

Finally, to improve the computational efficiency of the algorithm a number of approximations are made. This can have an impact on the results of UMAP for small (less than 500 samples) dataset sizes. In particular the use of approximate nearest neighbor algorithms, and the negative sampling used in optimization, can result in suboptimal embeddings. For this reason we encourage users to take care with particularly small datasets. A slower but exact implementation of UMAP for small datasets is a future project.

# 7 Future Work

Having established both relevant mathematical theory and a concrete implementation, there still remains significant scope for future developments of UMAP.

Making use of the fuzzy simplicial set representation of data UMAP can potentially be extended to support (semi-)supervised dimension reduction, and dimension reduction for datasets with heterogeneous data types. Each data type (or prediction variables in the supervised case) can be seen as an alternative view of the underlying structure, each with a different associated metric – for example categorical data may use Jaccard or Dice distance, while ordinal data might use Manhattan distance. Each view and metric can be used to independently generate fuzzy simplicial sets, which can then be intersected together to create a single fuzzy simplicial set for embedding. Extending UMAP to work with mixed data types would vastly increase the range of datasets to which it can be applied. Use cases for (semi-)supervised dimension reduction include semi-supervised clustering, and interactive labelling tools.

The computational framework established for UMAP allows for the potential development of techniques to add new unseen data points into an existing embedding, and to generate high dimensional representations of arbitrary points in the embedded space. Furthermore, the combination of supervision and the addition of new samples to an existing embedding provides avenues for metric learning. The addition of new samples to an existing embedding would allow UMAP to be used as a feature engineering tool as part of a general machine learning pipeline for either clustering or classification tasks. Pulling points back to the original high dimensional space from the embedded space would poten-

tially allow UMAP to be used as a generative model similar to some use cases for autoencoders. Finally, there are many use cases for metric learning; see [54] or [7] for example.

There also remains significant scope to develop techniques to both detect and mitigate against potentially spurious embeddings, particularly for small data cases. The addition of such techniques would make UMAP far more robust as a tool for exploratory data analysis, a common use case when reducing to two dimensions for visualization purposes.

Experimental versions of some of this work are already available in the referenced implementations.

# 8 Conclusions

We have developed a general purpose dimension reduction technique that is grounded in strong mathematical foundations. The algorithm implementing this technique is demonstrably faster than t-SNE and provides better scaling. This allows us to generate high quality embeddings of larger data sets than had previously been attainable. The use and effectiveness of UMAP in various scientific fields demonstrates the strength of the algorithm.

**Acknowledgements** The authors would like to thank Colin Weir, Rick Jardine, Brendan Fong, David Spivak and Dmitry Kobak for discussion and useful commentary on various drafts of this paper.

# References

[1] E Alpaydin and Fevzi Alimoglu. Pen-based recognition of handwritten digits data set. university of california, irvine. *Machine Learning Repository. Irvine: University of California*, 4(2), 1998.

[2] Frederik Otzen Bagger, Savvas Kinalis, and Nicolas Rapin. Bloodspot: a database of healthy and malignant haematopoiesis updated with purified and single cell mrna sequencing profiles. *Nucleic Acids Research*, 2018.

[3] Michael Barr. Fuzzy set theory and topos theory. *Canad. Math. Bull*, 29(4):501–508, 1986.

[4] Etienne Becht, Charles-Antoine Dutertre, Immanuel W.H. Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W Newell. Evaluation of umap as an alternative to t-sne for single-cell data. *bioRxiv*, 2018.

[5] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

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

[7] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

[8] Kenneth Blomqvist, Samuel Kaski, and Markus Heinonen. Deep convolutional gaussian processes. *arXiv preprint arXiv:1810.03052*, 2018.

[9] Tess Brodie, Elena Brenna, and Federica Sallusto. Omip-018: Chemokine receptor expression on human t helper cells. *Cytometry Part A*, 83(6):530–532, 2013.

[10] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[11] John N Campbell, Evan Z Macosko, Henning Fenselau, Tune H Pers, Anna Lyubetskaya, Danielle Tenen, Melissa Goldman, Anne MJ Verstegen, Jon M Resch, Steven A McCarroll, et al. A molecular census of arcuate hypothalamus and median eminence cell types. *Nature neuroscience*, 20(3):484, 2017.

[12] Gunnar Carlsson and Facundo Mémoli. Classifying clustering schemes. *Foundations of Computational Mathematics*, 13(2):221–252, 2013.

[13] Brian Clark, Genevieve Stein-O'Brien, Fion Shiau, Gabrielle Cannon, Emily Davis, Thomas Sherman, Fatemeh Rajaii, Rebecca James-Esposito, Richard

Gronostajski, Elana Fertig, et al. Comprehensive analysis of retinal development at single cell resolution identifies nfi factors as essential for mitotic exit and specification of late-born cells. *bioRxiv*, page 378950, 2018.

[14] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[15] Alex Diaz-Papkovich, Luke Anderson-Trocme, and Simon Gravel. Revealing multi-scale population structure in large cohorts. *bioRxiv*, page 423632, 2018.

[16] 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, pages 577–586, New York, NY, USA, 2011. ACM.

[17] Carlos Escolano, Marta R Costa-jussà, and José AR Fonollosa. (self-attentive) autoencoder-based universal language representation for machine translation. *arXiv preprint arXiv:1810.06351*, 2018.

[18] Greg Friedman et al. Survey article: an elementary illustrated introduction to simplicial sets. *Rocky Mountain Journal of Mathematics*, 42(2):353–423, 2012.

[19] Lukas Fuhrimann, Vahid Moosavi, Patrick Ole Ohlbrock, and Pierluigi Dacunto. Data-driven design: Exploring new structural forms using machine learning and graphic statics. *arXiv preprint arXiv:1809.08660*, 2018.

[20] Benoit Gaujac, Ilya Feige, and David Barber. Gaussian mixture models with wasserstein distance. *arXiv preprint arXiv:1806.04465*, 2018.

[21] Paul G Goerss and John F Jardine. *Simplicial homotopy theory*. Springer Science & Business Media, 2009.

[22] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[23] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, Mar 1964.

[24] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, pages 7:1–7:6, New York, NY, USA, 2015. ACM.

[25] Yann Lecun and Corinna Cortes. The MNIST database of handwritten digits.

[26] John A Lee and Michel Verleysen. Shift-invariant similarities circumvent distance concentration in stochastic neighbor embedding and variants. *Procedia Computer Science*, 4:538–547, 2011.

[27] Xin Li, Ondrej E Dyck, Mark P Oxley, Andrew R Lupini, Leland McInnes, John Healy, Stephen Jesse, and Sergei V Kalinin. Manifold learning of four-dimensional scanning transmission electron microscopy. *arXiv preprint arXiv:1811.00080*, 2018.

[28] M. Lichman. UCI machine learning repository, 2013.

[29] George Linderman. Fit-sne. `https://github.com/KlugerLab/FIt-SNE`, 2018.

[30] George C Linderman, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, and Yuval Kluger. Efficient algorithms for t-distributed stochastic neighborhood embedding. *arXiv preprint arXiv:1712.09005*, 2017.

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

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

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

[34] Sameer A. Nene, Shree K. Nayar, and Hiroshi Murase. Columbia object image library (coil-20. Technical report, 1996.

[35] Sameer A. Nene, Shree K. Nayar, and Hiroshi Murase. object image library (coil-100. Technical report, 1996.

[36] Karolyn A Oetjen, Katherine E Lindblad, Meghali Goswami, Gege Gui, Pradeep K Dagur, Catherine Lai, Laura W Dillon, J Philip McCoy, and Christopher S Hourigan. Human bone marrow assessment by single cell rna sequencing, mass cytometry and flow cytometry. *bioRxiv*, 2018.

[37] Jong-Eun Park, Krzysztof Polanski, Kerstin Meyer, and Sarah A Teichmann. Fast batch alignment of single cell transcriptomes unifies multiple mouse cell atlases into an integrated landscape. *bioRxiv*, page 397042, 2018.

[38] Jose Daniel Gallego Posada. Simplicial autoencoders. 2018.

[39] Emily Riehl. A leisurely introduction to simplicial sets. *Unpublished expository article available online at http://www. math. harvard. edu/˜ eriehl*, 2011.

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

[41] John W Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 100(5):401–409, 1969.

[42] Josef Spidlen, Karin Breuer, Chad Rosenberg, Nikesh Kotecha, and Ryan R Brinkman. Flowrepository: A resource of annotated flow cytometry datasets associated with peer-reviewed publications. *Cytometry Part A*, 81(9):727–731, 2012.

[43] David I Spivak. Metric realization of fuzzy simplicial sets. *Self published notes*, 2012.

[44] Jian Tang. Largevis. `https://github.com/lferry007/LargeVis`, 2016.

[45] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web*, pages 287–297. International World Wide Web Conferences Steering Committee, 2016.

[46] Joshua B. Tenenbaum. Mapping a manifold of perceptual observations. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 682–688. MIT Press, 1998.

[47] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[48] Dmitry Ulyanov. Multicore-tsne. `https://github.com/DmitryUlyanov/Multicore-TSNE`, 2016.

[49] Laurens van der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245, 2014.

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

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

[52] John Williamson. What do numbers look like? `https://johnhw.github.io/umap_primes/index.md.html`, 2018.

[53] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[54] Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, 2(2):4, 2006.

[55] Lofti A Zadeh. Information and control. *Fuzzy sets*, 8(3):338–353, 1965.

# A Proof of Lemma 1

**Lemma 1.** *Let $(\mathcal{M}, g)$ be a Riemannian manifold in an ambient $\mathbb{R}^n$, and let $p \in M$ be a point. If $g$ is locally constant about $p$ in an open neighbourhood $U$ such that $g$ is a constant diagonal matrix in ambient coordinates, then in a ball $B \subseteq U$ centered at $p$ with volume $\frac{\pi^{n/2}}{\Gamma(n/2+1)}$ with respect to $g$, the geodesic distance from $p$ to any point $q \in B$ is $\frac{1}{r} d_{\mathbb{R}^n}(p, q)$, where $r$ is the radius of the ball in the ambient space and $d_{\mathbb{R}^n}$ is the existing metric on the ambient space.*

*Proof.* Let $x^1, \ldots, x^n$ be the coordinate system for the ambient space. A ball $B$ in $\mathcal{M}$ under Riemannian metric $g$ has volume given by

$$\int_B \sqrt{\det(g)} dx^1 \wedge \cdots \wedge dx^n.$$

If $B$ is contained in $U$, then $g$ is constant in $B$ and hence $\sqrt{\det(g)}$ is constant and can be brought outside the integral. Thus, the volume of $B$ is

$$\sqrt{det(g)} \int_B dx^1 \wedge ... \wedge dx^n = \sqrt{det(g)} \frac{\pi^{n/2} r^n}{\Gamma(n/2+1)},$$

where $r$ is the radius of the ball in the ambient $\mathbb{R}^n$. If we fix the volume of the ball to be $\frac{\pi^{n/2}}{\Gamma(n/2+1)}$ we arrive at the requirement that

$$\det(g) = \frac{1}{r^{2n}}.$$

Now, since $g$ is assumed to be diagonal with constant entries we can solve for $g$ itself as

$$g_{ij} = \begin{cases} \frac{1}{r^2} & \text{if } i = j, \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

The geodesic distance on $\mathcal{M}$ under $g$ from $p$ to $q$ (where $p, q \in B$) is defined as

$$\inf_{c \in C} \int_a^b \sqrt{g(\dot{c}(t), \dot{c}(t))} dt,$$

where $C$ is the class of smooth curves $c$ on $\mathcal{M}$ such that $c(a) = p$ and $c(b) = q$, and $\dot{c}$ denotes the first derivative of $c$ on $\mathcal{M}$. Given that $g$ is as defined in (2) we see that this can be simplified to

$$\begin{aligned} &\frac{1}{r} \inf_{c \in C} \int_a^b \langle \sqrt{\dot{c}(t), \dot{c}(t)\rangle} dt \\ =&\frac{1}{r} \inf_{c \in C} \int_a^b \langle \|\dot{c}(t), \dot{c}(t)\| dt \\ =&\frac{1}{r} d_{\mathbb{R}^n}(p, q). \end{aligned} \tag{3}$$

□

# B Proof that FinReal and FinSing are adjoint

**Theorem 2.** *The functors* FinReal : **Fin-sFuzz** $\to$ **FinEPMet** *and* FinSing : **FinEPMet** $\to$ **Fin-sFuzz** *form an adjunction with* FinReal *the left adjoint and* FinSing *the right adjoint.*

*Proof.* The adjunction is evident by construction, but can be made more explicit as follows. Define a functor $F : \mathbf{\Delta} \times I \to \mathbf{FinEPMet}$ by

$$F([n], [0, a)) = (\{x_1, x_2, \ldots, x_n\}, d_a),$$

where

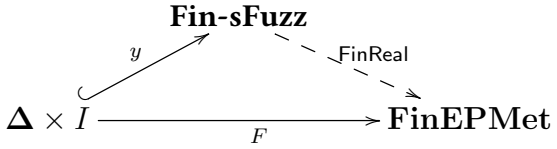
$$d_a(x_i, x_j) = \begin{cases} -\log(a) & \text{if } i \neq j, \\ 0 & \text{otherwise} \end{cases}.$$

Now FinSing can be defined in terms of $F$ as

$$\mathsf{FinSing}(Y) : ([n], [0, a)) \mapsto \hom_{\mathbf{FinEPMet}}(F([n], [0, a)), Y).$$

where the face maps $d_i$ are given by pre-composition with $Fd^i$, and similarly for degeneracy maps, at any given value of $a$. Furthermore post-composition with $F$ level-wise for each $a$ defines maps of fuzzy simplicial sets making FinSing a functor.

We now construct FinReal as the left Kan extension of $F$ along the Yoneda embedding:




Explicitly this results in a definition of FinReal at a fuzzy simplicial set $X$ as a colimit:

$$\mathsf{FinReal}(X) = \operatorname*{colim}_{y([n],[0,a)) \to X} F([n]).$$

Further, it follows from the Yoneda lemma that $\mathsf{FinReal}(\Delta^n_{<a}) \cong F([n], [0, a))$, and hence this definition as a left Kan extension agrees with Definition 7, and the definition of FinSing above agrees with that of Definition 8. To see that FinReal and FinSing are adjoint we note that

$$\begin{aligned} \hom_{\mathbf{Fin\text{-}sFuzz}}(\Delta^n_{<a}, \mathsf{FinSing}(Y)) &\cong \mathsf{FinSing}(Y)^n_{<a} \\ &= \hom_{\mathbf{FinEPMet}}(F([n], [0, a)), Y) \\ &\cong \hom_{\mathbf{FinEPMet}}(\mathsf{FinReal}(\Delta^n_{<a}), Y)). \end{aligned} \tag{4}$$

The first isomorphism follows from the Yoneda lemma, the equality is by construction, and the final isomorphism follows by another application of the Yoneda lemma. Since every simplicial set can be canonically expressed as a colimit of

standard simplices and FinReal commutes with colimits (as it was defined via a colimit formula), it follows that FinReal is completely determined by its image on standard simplices. As a result the isomorphism of equation (4) extends to the required isomorphism demonstrating the adjunction. □

# C From t-SNE to UMAP

As an aid to implementation of UMAP and to illuminate the algorithmic similarities with t-SNE and LargeVis, here we review the main equations used in those methods, and then present the equivalent UMAP expressions in a notation which may be more familiar to users of those other methods.

In what follows we are concerned with defining similarities between two objects $i$ and $j$ in the high dimensional input space $X$ and low dimensional embedded space $Y$. These are normalized and symmetrized in various ways. In a typical implementation, these pair-wise quantities are stored and manipulated as (potentially sparse) matrices. Quantities with the subscript $ij$ are symmetric, i.e. $v_{ij} = v_{ji}$. Extending the conditional probability notation used in t-SNE, $j \mid i$ indicates an asymmetric similarity, i.e. $v_{j|i} \neq v_{i|j}$.

t-SNE defines input probabilities in three stages. First, for each pair of points, $i$ and $j$, in $X$, a pair-wise similarity, $v_{ij}$, is calculated, Gaussian with respect to the Euclidean distance between $x_i$ and $x_j$:

$$v_{j|i} = \exp(-\left\|x_i - x_j\right\|_2^2 / 2\sigma_i^2) \tag{5}$$

where $\sigma_i^2$ is the variance of the Gaussian.

Second, the similarities are converted into $N$ conditional probability distributions by normalization:

$$p_{j|i} = \frac{v_{j|i}}{\sum_{k \neq i} v_{k|i}} \tag{6}$$

$\sigma_i$ is chosen by searching for a value such that the perplexity of the probability distribution $p_{\cdot|i}$ matches a user-specified value.

Third, these probability distributions are symmetrized and then further normalized over the entire matrix of values to give:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{7}$$

Similarities between pairs of points in the output space $Y$ are defined using a Student t-distribution with one degree of freedom on the squared Euclidean distance:

$$w_{ij} = \left(1 + \|y_i - y_j\|_2^2\right)^{-1} \tag{8}$$

followed by the matrix-wise normalization, to form $q_{ij}$:

$$q_{ij} = \frac{w_{ij}}{\sum_{k \neq l} w_{kl}} \tag{9}$$

The t-SNE cost is the Kullback-Leibler divergence between the two probability distributions:

$$C_{t-SNE} = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{10}$$

this can be expanded into constant and non-constant contributions:

$$C_{t-SNE} = \sum_{i \neq j} p_{ij} \log p_{ij} - p_{ij} \log q_{ij} \tag{11}$$

Because both $p_{ij}$ and $q_{ij}$ require calculations over all pairs of points, improving the efficiency of t-SNE algorithms has involved separate strategies for approximating these quantities. Similarities in the high dimensions are effectively zero outside of the nearest neighbors of each point due to the calibration of the $p_{j|i}$ values to reproduce a desired perplexity. Therefore an approximation used in Barnes-Hut t-SNE is to only calculate $v_{j|i}$ for $n$ nearest neighbors of $i$, where $n$ is a multiple of the user-selected perplexity and to assume $v_{j|i} = 0$ for all other $j$. Because the low dimensional coordinates change with each iteration, a different approach is used to approximate $q_{ij}$. In Barnes-Hut t-SNE and related methods this usually involves grouping together points whose contributions can be approximated as a single point.

LargeVis uses a similar approach to Barnes-Hut t-SNE when approximating $p_{ij}$, but further improves efficiency by only requiring approximate nearest neighbors for each point. For the low dimensional coordinates, it abandons normalization of $w_{ij}$ entirely. Rather than use the Kullback-Leibler divergence, it optimizes a likelihood function, and hence is maximized, not minimized:

$$C_{LV} = \sum_{i \neq j} p_{ij} \log w_{ij} + \gamma \sum_{i \neq j} \log \left(1 - w_{ij}\right) \tag{12}$$

$p_{ij}$ and $w_{ij}$ are defined as in Barnes-Hut t-SNE (apart from the use of approximate nearest neighbors for $p_{ij}$) and $\gamma$ is a user-chosen positive constant which weights the strength of the the repulsive contributions (second term) relative to the attractive contribution (first term). Note also that the first term resembles the optimizable part of the Kullback-Leibler divergence but using $w_{ij}$ instead of $q_{ij}$. Abandoning calculation of $q_{ij}$ is a crucial change, because the LargeVis cost function is amenable to optimization via stochastic gradient descent.

Ignoring specific definitions of $v_{ij}$ and $w_{ij}$, the UMAP cost function, the cross entropy, is:

$$C_{UMAP} = \sum_{i \neq j} v_{ij} \log \left( \frac{v_{ij}}{w_{ij}} \right) + (1 - v_{ij}) \log \left( \frac{1 - v_{ij}}{1 - w_{ij}} \right) \tag{13}$$

Like the Kullback-Leibler divergence, this can be arranged into two constant contributions (those containing $v_{ij}$ only) and two optimizable contributions (containing $w_{ij}$):

$$\begin{aligned} C_{UMAP} = \sum_{i \neq j} & v_{ij} \log v_{ij} + (1 - v_{ij}) \log (1 - v_{ij}) \\ & - v_{ij} \log w_{ij} - (1 - v_{ij}) \log (1 - w_{ij}) \end{aligned} \tag{14}$$

Ignoring the two constant terms, the UMAP cost function has a very similar form to that of LargeVis, but without a $\gamma$ term to weight the repulsive component of the cost function, and without requiring matrix-wise normalization in the high dimensional space. The cost function for UMAP can therefore be optimized (in this case, minimized) with stochastic gradient descent in the same way as LargeVis.

Although the above discussion places UMAP in the same family of methods as t-SNE and LargeVis, it does not use the same definitions for $v_{ij}$ and $w_{ij}$. Using the notation established above, we now provide the equivalent expressions for the UMAP similarities. In the high dimensional space, the similarities $v_{j|i}$ are the local fuzzy simplicial set memberships, based on the smooth nearest neighbors distances:

$$v_{j|i} = \exp[(-d(x_i, x_j) - \rho_i)/\sigma_i] \tag{15}$$

As with LargeVis, $v_{j|i}$ is calculated only for $n$ approximate nearest neighbors and $v_{j|i} = 0$ for all other $j$. $d(x_i, x_j)$ is the distance between $x_i$ and $x_j$, which UMAP does not require to be Euclidean. $\rho_i$ is the distance to the nearest neighbor

of $i$. $\sigma_i$ is the normalizing factor, which is chosen by Algorithm 3 and plays a similar role to the perplexity-based calibration of $\sigma_i$ in t-SNE. Calculation of $v_{j|i}$ with Equation 15 corresponds to Algorithm 2.

Symmetrization is carried out by fuzzy set union using the probabilistic t-conorm and can be expressed as:

$$v_{ij} = \left(v_{j|i} + v_{i|j}\right) - v_{j|i}v_{i|j} \tag{16}$$

Equation 16 corresponds to forming top-rep in Algorithm 1. Unlike t-SNE, further normalization is not carried out.

The low dimensional similarities are given by:

$$w_{ij} = \left(1 + a \left\|y_i - y_j\right\|_2^{2b}\right)^{-1} \tag{17}$$

where $a$ and $b$ are user-defined positive values. The procedure for finding them is given in Definition 11. Use of this procedure with the default values in the UMAP implementation results in $a \approx 1.929$ and $b \approx 0.7915$. Setting $a = 1$ and $b = 1$ results in the Student t-distribution used in t-SNE.