A Comprehensive Comparison of tSNE and UMAP

January 19, 2022

Abstract

Dimensionality reduction algorithms are used across scientific disciplines to visualize and understand data. Among them, tSNE and UMAP have become two of the most popular due to their quick runtimes and intuitive results. However, despite their ubiquity and similarities there has not been a comprehensive comparison of the two approaches. In this work, we describe a general framework for comparing dimensionality reduction algorithms and identify simple methods for obtaining each algorithm from the other. We disprove several claims regarding the source of the algorithms' efficacy and clarify misconceptions regarding components of each algorithm. Inspired by this, we create a new dimensionality reduction algorithm, Uniform UMAP, by combining the best characteristics of tSNE and UMAP. Uniform UMAP can recreate the results of both tSNE & UMAP and, when executed in a single thread, performs the optimizations an order of magnitude faster than multi-threaded UMAP and several orders of magnitude faster than multi-threaded tSNE on even modestly sized datasets. We additionally highlight that Uniform UMAP is easier to parallelize than either of the previous approaches and provide a simple GPU implementation for yet another order of magnitude speedup. Lastly, we provide theoretical insights for future dimensionality reduction improvements.

1 Introduction

Dimensionality reduction (DR) algorithms are invaluable for qualitatively inspecting high-dimensional data. As such, they are consistently used across the sciences with famous applications in _____, _____, and _____. A DR algorithm receives an input dataset $X \in \mathbb{R}^{N \times D}$ of N points in D dimensional space and attempts to find a faithful embedding into a lower dimensional space $Y \in \mathbb{R}^{N \times d}$, where $d << D$. The term *faithful* is defined as keeping similar points similar and dissimilar points dissimilar, where similarity is commonly measured by distance metrics or kernels in the corresponding spaces.

Most DR methods fall into one of the following two categories:

Linear and Manifold Methods frame the DR assignment as an eigendecomposition problem. The most famous example, Principal component analysis (PCA) projects the data onto a lower-dimensional space defined by the leading eigenvectors of the data's covariance matrix. Laplacian eigenmaps [1] instead define the nearest neighbor graph of the high-dimensional dataset and perform the embedding using the graph's spectral decomposition. In a sort of mixture of these two, Isomap [6] finds the distances along the nearest neighbor graph between each pair of points before using multi-dimensional scaling to low-dimensional points with similar distances. These methods prioritize long-distance relationships over short-distance relationships due to relying on the global-scale leading eigenvalues. Said otherwise, their projections interpret faithfulness as keeping dissimilar points dissimilar.

Gradient Descent Methods instead define objective functions without clear closed-form solutions and resort to gradient descent to embed suitable low-dimensional points. This is achieved by defining kernels on the pairwise distances between points in each space. Traditionally, the high-dimensional distance kernels are Gaussian whereas the low-dimensional ones are negative-degree polynomials. We then employ gradient descent to minimize a loss function between these respective high-dimensional and low-dimensional kernel relationships.

In this paper, we focus on the tSNE [4] and UMAP [5] gradient descent methods. These have become the standard DR techniques in recent years, each getting cited more than 1000 times per year on average. Despite both their ubiquity and similarity, there has not been a thorough analysis of the implementation choices between them. This paper attempts to address this gap in the following manner. In section 2, we present a comprehensive discussion of the differences between tSNE and UMAP, highlighting which ones depend on design choices and which ones are generalizable. Section 3 defines our methods for analyzing the quantitative and qualitative performance of a DR algorithm and shows which parts of each algorithm are necessary for obtaining its unique results. Finally in section 2.2, we introduce an algorithm that combines the best of both tSNE and UMAP while being an order of magnitude more efficient than either. Importantly, our method can recreate both of the originals while also allowing flexibility to obtain results in between tSNE's and UMAP's outputs. We introduce methods for effective parallelization and discuss our CPU- and GPU-based implementations.

2 Comparison of tSNE and UMAP

We begin by formally introducing the tSNE and UMAP DR algorithms. Let $X \in \mathbb{R}^{N \times D}$ be the high dimensional dataset of N points and let $Y \in \mathbb{R}^{N \times d}$ be a previously initialized set of N points in lower-dimensional space such that $d << D$.

We now respectively establish Gaussian and student-t kernels on the high- and low-dimensional distances to represent the likelihood that x_i and y_i choose x_j and y_j as their respective nearest neighbors. These kernels are defined by

$$p_{j|i}^{tsne}(x_i, x_j) = \frac{\exp(-d(x_i, x_j)^2/2\sigma_i^2)}{\sum_{k \neq l} \exp(-d(x_k, x_l)^2/2\sigma_k^2)} \tag{1}$$

$$q_{ij}^{tsne}(y_i, y_j) = \frac{(1 + ||y_i - y_j||_2^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||_2^2)^{-1}} \tag{2}$$

$$p_{j|i}^{umap}(x_i, x_j) = \frac{\exp(-d(x_i, x_j)^2 + \rho_i)}{\tau_i} \tag{3}$$

$$q_{ij}^{umap}(y_i, y_j) = \left(1 + a(||y_i - y_j||_2^2)^b\right)^{-1} \tag{4}$$

where $d(x_i, x_j)$ is the high-dimensional distance function, σ and τ are point-specific variance scalars, $\rho_i = \min_{j \neq i} d(x_i, x_j)$, and a and b are constants. In practice, we can assume that $2\sigma_i^2$ is functionally equivalent to τ_i, and we will use τ when referring to this kernel variance term. The high-dimensional kernels are symmetrized by applying symmetrization functions. Without loss of generality, let $p_{ij} = S(p_{j|i}, p_{i|j})$ for some symmetrization function S.

We posit that the primary difference between tSNE and UMAP is the change in normalization. tSNE normalizes the high- and low-dimensional kernels by all of the pairwise relations whereas UMAP allows the Gaussian and Student-t kernels to remain untouched[1]. This implies a different probabilistic interpretation between the two algorithms. In the case of tSNE, the weights are normalized across the entire matrix of pairwise relations, giving us a probability distribution across the entire dataset. In the case of UMAP, our pairwise kernels remain unnormalized, implying a Bernoulli distribution along each edge that can be interpreted as the likelihood of that edge existing (effectively independently from the other edges).

Each algorithm then applies gradient descent with respect to the KL-divergence(s) between these probability distributions. We refer to the high- and low-dimensional probability distributions as $P = (p_{ij})_{i,j<N}$ and $Q = (q_{ij})_{i,j<N}$ respectively. This gives us the loss functions

$$\mathcal{L}_{tsne} = \mathrm{KL}(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{5}$$

$$\mathcal{L}_{umap} = \sum_{i \neq j} \left[p_{ij} \log \frac{p_{ij}}{q_{ij}} + (1 - p_{ij}) \log \frac{1 - p_{ij}}{1 - q_{ij}} \right] \tag{6}$$

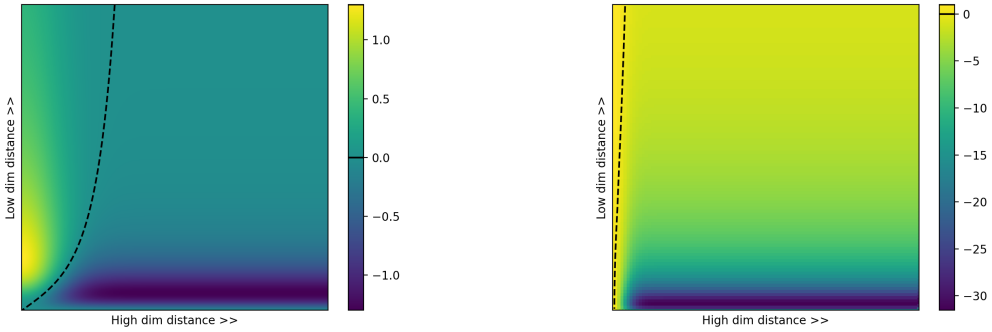
In essence, tSNE minimizes the KL divergence of the low- and high-dimensional pairwise distance matrices under their respective kernels, whereas UMAP sums over each edge's KL divergence in terms of the Bernoulli distribution.

2.1 Intuition

We would like to provide a probabilistic interpretation of these two normalizations. In the normalized case we have a probability distribution over all distinct pairwise relationships $p(x_i, x_j)$ and $q(y_i, y_j)$ for $i \neq j$; while in the unnormalized case we have a probability distribution over each individual distinct pairwise relationship. We can look at these through the lens of a fully-connected graphs in which each edge e_{ij} is the kernel value $p(x_i, x_j)$ or $q(y_i, y_j)$. Under this lens, the probabilities in the normalized case are akin to asking "when we pick a random edge, what is the probability that we pick e_{ij}?". Alternatively, the unnormalized variant asks "for the specific edge e_{ij}, what is the probability that it exists?"

In practice, both tSNE and UMAP optimize this KL divergence by separately calculating attractive and repulsive forces. It is, however, unnecessary to calculate each such attraction and repulsion as, for example, dissimilar points in the high-dimensional space impose negligible attraction in the low-dimensional space. To simplify the complexity, both approaches establish a nearest neighbor graph [7] with edges $|\mathcal{E}|$ among the high-dimensional points where points x_i and x_j share an edge if either x_i is x_j's nearest neighbor or x_j is x_i's nearest neighbor. Then both algorithms simply perform attractions along these edges while performing repulsions evenly across the rest of the points. In this sense, we can interpret the gradient descent problem as a set of springs between every pair of points in Y where the spring constants are determined by the Gaussian and Student-t kernels.

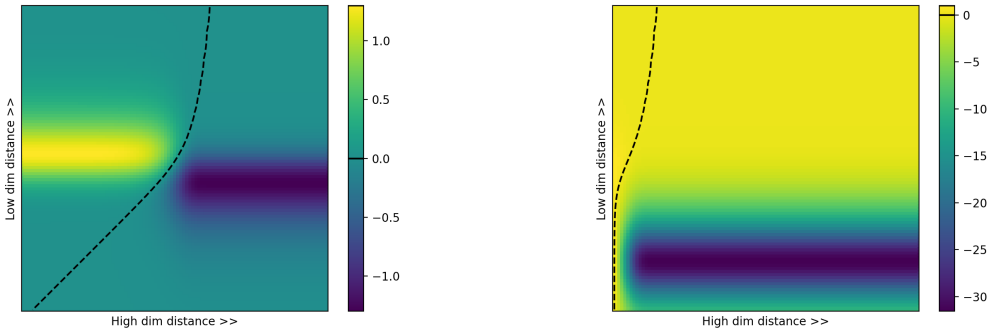
[1] We note that the tSNE papers give the impression that high-dimensional normalization occurs across rows, while in reality the code performs normalization across the entire pairwise kernel matrix.



(a) tSNE gradients

(b) UMAP gradients

Figure 1: A comparison of tSNE and UMAP gradients for linearly growing distances



(a) tSNE gradients

(b) UMAP gradients

Figure 2: A comparison of tSNE and UMAP gradients for exponentially growing distances

2.2 Gradients

The gradient of the KL divergences becomes substantially different due to the differing normalizations. In tSNE, the gradient can be written as

$$\frac{\partial \mathcal{L}_{tsne}}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) q_{ij} Z (y_i - y_j) \tag{7}$$

where $Z = \sum_{k \neq l} (1 + ||y_k - y_l||_2^2)^{-1}$ is the normalization factor for the low-dimensional kernel. This is often represented as an attractive and repulsive force with

$$\begin{aligned} \frac{\partial \mathcal{L}_{tsne}}{\partial y_i} &= 4(F_{attr}^{tsne} + F_{rep}^{tsne}) = \\ &= 4 \left[\sum_{j \neq i} p_{ij} q_{ij} Z (y_i - y_j) - \sum_{j \neq i} q_{ij}^2 Z (y_i - y_j) \right] \end{aligned}$$

UMAP also describes separating its gradient into attractive and repulsive terms, with

$$F_{attr}^{umap} = \frac{-2ab||y_i - y_j||_2^{2(b-1)}}{1 + ||y_i - y_j||_2^2} p_{ij} (y_i - y_j) \tag{8}$$

$$F_{rep}^{umap} = \frac{2b}{(\epsilon + ||y_i - y_j||_2^2)(1 + a||y_i - y_j||_2^{2b})} \cdot \tag{9}$$

$$\cdot (1 - p_{ij})(y_i - y_j) \tag{10}$$

Note that the UMAP repulsive force is inversely quadratic with respect to the low-dimensional distance. This accounts for UMAP's significantly stronger repulsions seen in figures 1 and 2.

2.3 Implementation differences

Given the above descriptions of the algorithms, we now describe specific implementation differences between the two. Section 3 will discuss the practical effect that each of these has on the results.

- tSNE collects all of the attractive and repulsive forces before applying momentum gradient descent across every point simultaneously. In contrast, UMAP updates the position of every point directly upon calculating its attractive and repulsive forces.
  - Accordingly, UMAP applies attractive and repulsive forces iteratively by calculating k random repulsive forces for every attractive force along the nearest neighbor edges.
  - In contrast, tSNE similarly calculates attractions along each edge but estimates each point's repulsions across the entire dataset with Barnes-Hut trees.
- tSNE's gradient descent has an additional scalar that amplifies the forces if a point moves in the same direction at time $t+1$ as it did at time t and dampens the forces otherwise.
- UMAP finds approximate nearest neighbors using nearest-neighbor descent whereas tSNE takes the time to exactly identify nearest neighbor relationships.
- UMAP's high dimensional kernel on points x_i and x_j subtracts the minimum distance $\rho_i = \min_{k \neq i} d(x_i, x_k)$. The theoretical justification for this is discussed at length in the UMAP paper.
- tSNE symmetrizes the high-dimensional kernels with $S_{tsne}(p_{j|i}, p_{i|j}) = (p_{j|i} + p_{i|j})/2$ while UMAP uses a probabilistic symmetrization with $S_{umap}(p_{j|i}, p_{i|j}) = p_{j|i} + p_{i|j} - p_{j|i} \cdot p_{i|j}$
- tSNE performs random initialization whereas UMAP initializes with a Laplacian eigenmap projection.
- UMAP applies the attractive force between y_i and y_j to both y_i's and y_j's positions whereas tSNE applies the attractive force to only y_i. We refer to these options as *symmetric* and *asymmetric* attraction.

2.4 Uniform UMAP

We must make a few modifications to either UMAP's or tSNE's optimization methodology in order to perform a fully controlled comparison between the two. The principle obstacle is that UMAP performs live gradient updates on each point (which precludes the use of momentum gradient descent) while tSNE collects the gradients across the entire dataset before applying them simultaneously. This complicates comparing the two as tSNE normalizes repulsive forces by the sum over all kernels Z, therefore making it impossible to implement tSNE within UMAP's original framework. Specifically, UMAP would require access to all the repulsions for normalization but applies the repulsions before all of them have been collected.

UMAP's original algorithm approximates the p_{ij} and $(1 - p_{ij})$ scalars in the attractive and repulsive forces by selectively applying attractions and repulsions per epoch. Namely, the attractive and repulsive forces are applied on any given epoch with likelihood proportional to p_{ij} or $(1 - p_{ij})$. Thus if a force was applied on epoch i it will not necessarily be performed on epoch $i+1$. Note that this choice was not necessitated by theoretical derivations and can be modified without sacrificing the theoretical soundness of the UMAP algorithm.

We would instead like to implement UMAP such that we uniformly calculate one repulsion for every attraction in order to neatly collect all of forces before normalizing and performing gradient descent. To do this, note that the nearest-neighbor edge weights p_{ij} are available to us during optimization. In order to approximate the $(1-p_{ij})$ scalars, however, we hypothesize that they will effectively average out when applied to random pairs of points over many repulsions. We therefore replace the sampling scheme by doing one repulsion for every attraction and scaling the repulsive force by $(1 - \bar{p}_{ij})$, where $\bar{p}_{ij}$ is the mean high-dimensional kernel value. We experimentally validate this decision in table ?? and show that this achieves identical results to UMAP across datasets and metrics.

We name this modified UMAP implementation *Uniform UMAP* and note that it has several structural advantages to the original method.

- Uniform UMAP removes the need for calculating multiple repulsions during the sampling schema, cutting down on the number of force calculations by $n - 1$, where n is the number of repulsions performed when the repulsive forces are sampled.
- We can now easily collect gradients across the entire dataset, allowing us to apply techniques like momentum gradient descent for quicker convergence.
- The repulsions no longer depend on the epoch or edge weights, allowing us to effectively distribute force computations
- When correctly modified to implement tSNE, Uniform UMAP removes tSNE's dependency on the computationally costly Barnes-Hut trees, obtaining a very significant speedup over other tSNE implementations.

With this change, we can use Uniform UMAP to obtain UMAP's results using full-dataset gradient descent as is done in tSNE.

3 Embedding Results

We now show that, subject to the change in 2.2, one can directly implement both tSNE and UMAP through either framework. We first show that Uniform UMAP can successfully recreate UMAP's outputs across a variety of datasets. Then, we go through the elements in 2.1 to identify what effect each choice has on the tSNE and Uniform UMAP embeddings. Using this, we identify the changes necessary to produce tSNE's outputs within the UMAP framework and vice versa.

3.1 Metrics

To avoid relying on qualitative observation, we introduce several metrics that we will be using to describe both embedding quality and the differences between tSNE and UMAP.

Want one metric that is "quality of embedding" and one that captures "amount of space between clusters". For the latter, it should be enough to get distances of all points between two clusters and take the 1% cut. Basically, something to give you the distance between the shells of the two clusters.

3.2 UMAP and Uniform UMAP Equivalency

3.3 Analysis of Implementation Choices

3.4 Achieving tSNE within UMAP

The main obstacle to overcome is undoing the normalization differences between tSNE and UMAP. In fact, tSNE can be achieved within the UMAP implementation through the following modifications:

1. Normalize the high- and low-dimensional kernels as they are normalized in tSNE
2. Perform tSNE's gradient descent as described in 2.2
3. Apply asymmetric attraction forces

Due to using tSNE's normalization, we also require the corresponding modification to the KL divergence. As such, we obtain gradients

$$F_{attr} = 4\sum_{j\neq i} p_{ij}q_{ij}Z(y_i - y_j)$$
$$F_{rep} = 4\sum_{j\neq i} q_{ij}^2 Z(y_i - y_j)$$

where

$$q_{ij} = \frac{(1 + a||y_i - y_j||_2^{2b})^{-1}}{\sum_{k\neq l}(1 + a||y_i - y_j||_2^{2b})^{-1}}$$
$$p_{j|i} = \frac{\exp(-(d(x_i, x_j)^2 - \rho_i)/\tau_i)}{\sum_{k\neq l}\exp((-d(x_k, x_l)^2 + \rho_i)/\tau_k)}$$

and $Z = \sum_{k\neq l}(1 + a||y_i - y_j||_2^{2b})^{-1}$. Notice that this is essentially the tSNE kernels with UMAP's ρ_i, a, and b scalars. We show later that these can be removed in practice.

3.5 Achieving UMAP within tSNE

This direction turns out to be significantly easier. Performing the Barnes-Hut algorithm for tSNE with a Laplacian eigenmap initialization and symmetric attraction is sufficient to obtain UMAP consistently.

3.6 Claims and misconceptions

We would like to discuss several claims regarding these algorithms' choices and use-cases. Firstly, we'd like to note that the tSNE paper mentions that the high-dimensional probabilities $p_{j|i}$ are normalized by the sum $\sum_{k\neq i}\exp\left(||y_i - y_k||^2\right)/\tau_i$. To the authors of this paper and other people whom we've spoken to, this implies that the normalization occurs across rows of

the pairwise probability matrix. Instead, tSNE implementations normalize $p_{j|i}$ across the entire pairwise probability matrix, giving us symmetric normalizations in the high- and low-dimensional spaces (a necessary condition for the KL divergence).

Additionally, there are several choices made in UMAP that do not provide practical changes. Namely, we find that the choice of normalization is the single necessary condition for switching between tSNE and UMAP. This puts the practical necessity of multiple components of UMAP's algorithm into question. The first of these is the pseudo-distance metric $\tilde{d}_i(x_j, x_k) = ||x_j - x_k||_2^2 - \rho_i$. Its inclusion theoretically justifies that the nearest-neighbor graph captures the topology of the high-dimensional dataset. Despite the sound theoretical derivations we find across datasets, metrics, and contexts that subtracting the minimum distance has a negligible effect on the resulting embeddings. We also refute the assertion that tSNE's lack of a mathematical framework prohibits it from being extended; specific examples include embedding unseen points given an existing projection, utilizing categorical variables, and consolidating differing metrics. Our implementation of Uniform UMAP is able to, under tSNE's normalization and optimization schema, perform all of these tasks in the same manner as UMAP while obtaining results in line with tSNE's. Lastly, there have been claims made that tSNE with the Laplacian eigenmap initialization recreates UMAP's outputs [3] or that the initialization is primarily responsible for the quality of the resulting embedding [2]. While this may be true for certain datasets, we do not observe it to be the case universally.

4 Left to Show

This is a scratchwork section for identifying remaining topics of study

- Paper organization
  - We should have more discussion around the fact that Uniform UMAP (when recreating UMAP with it) still follows all of UMAP's theory, and only changes implementation choices that are arbitrary with respect to that theory. Although implementing tSNE within the framework requires a modification of the normalization, it does not substantially modify the algorithm beyond that.
- tSNE and UMAP
  - Should we show that the additional KL sum in UMAP grad directly accounts for the change in normalization?
- Uniform UMAP
  - When can we replace k applications of an attractive force with a single application of a stronger attractive force
    - Probably when it's normalized since the repulsions are managable
  - Stochastic gradient descent with batches rather than doing the entire dataset at once?
  - GPU implementation and speed tests for Uniform UMAP?
- General
  - Why do the PCA plots look the way that they do?

References

[1] Mikhail Belkin and Partha Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation". In: *Neural computation* 15.6 (2003), pp. 1373–1396.

[2] Dmitry Kobak and George C Linderman. "Initialization is critical for preserving global data structure in both t-SNE and UMAP". In: *Nature biotechnology* 39.2 (2021), pp. 156–157.

[3] Dmitry Kobak and George C Linderman. "UMAP does not preserve global structure any better than t-SNE when using the same initialization". In: *bioRxiv* (2019).

[4] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[5] Leland McInnes, John Healy, and James Melville. "Umap: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426* (2018).

[6] Joshua B Tenenbaum, Vin De Silva, and John C Langford. "A global geometric framework for nonlinear dimensionality reduction". In: *science* 290.5500 (2000), pp. 2319–2323.

[7] Laurens Van Der Maaten. "Accelerating t-SNE using tree-based algorithms". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 3221–3245.