A Comprehensive Comparison of tSNE and UMAP

December 27, 2021

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. Despite their ubiquity and similarities, however, 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 identifies a single line of code that is responsible for switching between tSNE and UMAP. We disprove several claims to explain their efficacy and clarify misconceptions regarding components of each algorithm. Inspired by this, we create a new dimensionality reduction algorithm by combining the best characteristics of tSNE and UMAP while running an order of magnitude faster than UMAP on CPU. Lastly, we provide GPU code that further accelerates our algorithm and provide theoretical insight for future dimensionality reduction approaches.

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 go-to DR technique 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. The rest of this paper is structured as follows. In section 2, we present a comprehensive discussion of the differences between tSNE and UMAP, highlighting which ones are structural and which ones are arbitrary. Sections 3 and 5 define our methods for analyzing the quantitative and qualitative performance of a DR algorithm and show how to obtain tSNE in UMAP's framework or UMAP in tSNE's framework. Finally in section 4, 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 establish Gaussian (Student-t) kernels on the high (low) dimensional distances to represent the likelihood that x_i (y_i) chooses x_j (y_j) as its nearest neighbor. These kernels

are defined by

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

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

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

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

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.

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.

We can rearrange terms in $\mathcal{L}_{umap}$ to obtain

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

Notice that the first sum is identical to the sum in $\mathcal{L}_{tsne}$ up to normalization. Thus, the difference in normalization is directly accounted for by the additional sum in $\mathcal{L}_{umap}$.

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 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 leaving the repulsions to evenly sample the remaining 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.

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

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

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

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

$$F^{umap}_{rep} = \frac{2b}{(\epsilon + ||y_i - y_j||_2^2)(1+a||y_i - y_j||_2^{2b})}\cdot \tag{9}$$
$$\cdot(1-p_{ij})(y_i - y_j) \tag{10}$$

However, we note that these are actually the gradients of the two terms in the summand of $\mathcal{L}_{umap}$, and each has its own attractive and repulsive component within it. To avoid confusion, we will continue to refer to these as attractive and repulsive forces to stay consistent with the terminology used by the original authors.

2.1 Implementation differences

Given the above descriptions of the algorithms, we now describe specific implementation differences between the two. Section 5 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. This affects how each algorithm performs gradient clipping, where UMAP clips each gradient individually while tSNE clips the sum of the gradients across the entire epoch.

[1] We note that the tSNE papers sometimes misrepresent the normalization as occurring across rows, while in reality the code performs normalization across the entire pairwise kernel matrix.

  - 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.

- 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*.

2.2 Claims and misconceptions

We would like to highlight several misconceptions and claims regarding these algorithms' performances. Firstly, we'd like to note that the tSNE paper mentions that the high-dimensional probabilities $p_{j|i}$ are normalized by the sum $\sum_{k \neq i} \exp\left(||y_i - y_j||^2\right)/\tau_i$. 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 claims regarding UMAP that we intend to shine light on in future sections. Specifically, we investigate which changes in 2.1 are necessary for simultaneously preserving local and global relationships. In the results section, we show that the choice of symmetric or asymmetric attraction is sufficient to switch between tSNE and UMAP. This puts into question several claims regarding UMAP. The first claim we discuss is that the pseudo-distance metric $\tilde{d}_i(x_j, x_k) = ||x_j - x_k||_2^2 - \rho_i$ is responsible for the balance between local and global relationships. Indeed, we find across datasets, metrics, and contexts that subtracting the minimum distance has a negligible effect on the resulting embeddings. Another assertion is 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. However, our implementation of Uniform UMAP is able to, under tSNE's normalization and optimization schema, perform all of these tasks in the same way as UMAP. We also point out that UMAP's described attractive and repulsive forces actually each contain their own attractive and repulsive forces. Lastly, there is a discussion 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.

3 Theoretical Analysis

3.1 A framework for dimensionality reduction

We identify a general framework for studying DR algorithms that is based on the following observation: each DR algorithm attempts to match kernel on high-dimensional distances with kernels on low-dimensional distances. This means that there are four defining characteristics to each algorithm – the high-dimensional kernel, the low-dimensional kernel, the similarity metric, and the optimization method. In the cases of tSNE and UMAP, these are formally defined as exponential Gaussian kernels in the high-dimensional space and Student-t kernels in the low dimensional space. Both then use the KL-divergence and gradient descent to optimize the embeddings.

4 Uniform UMAP

We propose a modified implementation of UMAP that we call Uniform UMAP. The principle difference consists in the gradient descent techniques. Each epoch of UMAP loops through the nearest neighbor edges and performs one attraction along with n repulsions. These gradient updates are performed live within the loop, modifying the positions of each point as it is processed.

Our first modification changes the gradient application process. Rather than updating positions within the loop, we can instead collect all of the attractive and repulsive forces and apply them simultaneously at the end of the epoch. This in practice removes the need for calculating multiple repulsions for every attraction, cutting down on the number of force calculations by $n - 1$, where n is the number of repulsions performed for every attraction. This is complicated, however, by the fact that UMAP applies the $(1 - p_{ij})$ scaling in 10 by sampling the edge proportionally to the weight. Due to uniformly optimizing all of the edges, we cannot apply this sampling strategy. Instead, we assume that the randomly chosen pairwise repulsions will average out and simply use the average weight $\bar{p}_{ij}$ as a substitute when calculating the UMAP repulsive forces. We theoretically substantiate this decision and also experimentally validate that it does not impart a noticeable change in any of our test cases.

The second major modification has to do with the iterative nature of the gradient descent epochs. Consider the following common situation: assume during epoch t that we apply an attraction along the edge (y_i, y_j) and then a repulsion along (y_i, y_k). Then, in epoch $t + 1$ we once again attract along

(y_i, y_j) and repel along (y_i, y_l). Due to randomness, many individual repulsive forces will negligibly affect the embedding. Thus the two attractive forces, originally separated by a repulsion, can be approximated by performing the attraction (y_i, y_j) at time t twice, followed by applying the two repulsive forces in tandem. Generalizing this logic, we note that applying a k-times stronger attractive force with k random repulsions approximates the iterative nature of UMAP and tSNE, with approximations worsening as k grows. Furthermore, if we intend to apply an attractive force k times then we can calculate apriori how much the force will diminish with each application. Thus, we can be more sophisticated than simply scaling the force linearly by the number of repeated applications. *It remains to identify when this approximation is appropriate and to what extent it is correct.*

5 Experimental Results

We show that, subject to a few approximations, one can directly implement both tSNE and UMAP through the Uniform UMAP framework. We first show that Uniform UMAP can successfully recreate UMAP's outputs across a variety of datasets. Then, we identify the changes necessary to produce tSNE's outputs. Finally, we note that implementing tSNE with UMAP's framework implies that one can implement UMAP within tSNE's framework and experimentally validate that this is indeed possible.

5.1 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. 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

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

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.

5.2 Embedding analysis

We now show multiple plots from applying multiple dimensionality reduction algorithms to several datasets. The plots are as follows:

1. Change in sort index - If we sort the high-dimensional distances and the low-dimensional distances, how different are the sort indices? The x-axis goes through all the high-dimensional distances in order; the y-axis is the relative change for that high-dimensional sort index

2. High dimensional distance vs. low dimensional distance - These are simply the low-dimensional distances plotted vs. the high dimensional distances. The high dimensional distances have been sorted from least to greatest. Note that PCA is a straight line (which is what we expected). Also, note that the large high-dim distances are represented by larger low-dim distances in UMAP than in tSNE

3. Sorted distances - This is just the y-values of the previous plot. Basically, when we sort the high-dimensional distances, what do the low dimensional ones look like?

4. Nearest neighbor overlap - For every point, what percent of its [k, k + 10] nearest neighbors are preserved? Note that immediate local distances are preserved in tSNE and UMAP, after which there is a giant portion where the middle distances aren't relevant.

5. Relative error - This is the relative error for high-dimensional distances. We normalize both the high-dim and low-dim distances to $[0, 1]$, since the magnitudes are arbitrary in our setting. Thus, we do $|(d^h(x_i, x_j) - d^l(y_i - y_j))|/d^h(x_i, x_j)$. We replace the 0's in the high dimensional distances with 1's to avoid division errors.

6 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 directly accounts for the change in normalization?
  - Why does adding symmetric attraction make tSNE exactly replicate UMAP?
- Uniform UMAP

  - When can we replace k applications of an attractive force with a single application of a stronger attractive force
  - Stochastic gradient descent with batches rather than doing the entire dataset at once?
  - Why does Uniform UMAP not need muliple repulsions for each attraction?
  - GPU implementation and speed tests for Uniform UMAP?
  - Why does numba implementation get different results from cython implementation?
- 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.