A Comprehensive Comparison of tSNE and UMAP

November 29, 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. This work identifies a general framework for comparing dimensionality reduction algorithms and analyzes tSNE and UMAP through the developed lens. We resolve misconceptions regarding components of each algorithm while highlighting theoretical and practical similarities between them. In addition, we provide code that substantially accelerates both algorithms and provide theoretical insight for future dimensionality reduction approaches.

1 Dimensionality Reduction Algorithms

We begin by formally introducing the tSNE and UMAP dimensionality reduction algorithms. Let $X \in \mathbb{R}^D$ be the high dimensional dataset of N points and let $Y \in \mathbb{R}^d$ be a randomly initialized set of N points in some lower-dimensional space such that $d << D$. We establish a nearest neighbor graph 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 <u>or</u> x_j is x_i's nearest neighbor.

We now establish Gaussian and Student-t kernels on the respective high- and low-dimensional distances to represent the likelihood that x_i chooses x_j as its nearest neighbor. These kernels are defined by

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

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

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

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

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 primary difference between the two is the change in normalization. tSNE normalizes the high- and low-dimensional relationships by all of the pairwise distances whereas UMAP allows the Gaussian and Student-t kernels to remain untouched. We note that while the tSNE paper implies that high-dimensional normalizations occur across rows of the pairwise distance matrix, the code performs normalization across the entire pairwise distance matrix. The high-dimensional kernels are symmetrized by applying symmetrization functions. Without loss of generality, let $p_{ij} = S(p_{j|i}, p_{i|j})$.

Each algorithm then applies gradient descent with respect to the KL-divergence(s) between probability distributions. In the case of tSNE, the probability distributions are defined by the matrices $P = (p_{ij})_{i,j<N}$ and $Q = (q_{ij})_{i,j<N}$. However, UMAP's lack of normalization instead implies a single Bernoulli distribution for each edge. Thus, UMAP actually minimizes the sum of KL divergences between every edge in high- and low-dimensional space.

This gives us the loss functions

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

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

where the operand inside $\mathcal{L}_{umap}$'s sum is the KL divergence of a Bernoulli distribution. We point out that usage of the KL divergence necessitates symmetrical normalization between the high- and low-dimensional spaces, further supporting tSNE's implementation of the normalization over the asymmetrical normalization discussed in the paper.

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. Thus, 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 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

$$\frac{\partial \mathcal{L}_{tsne}}{\partial y_i} = 4(F_{attr}^{tsne} + F_{rep}^{tsne}) =$$

$$= 4 \left[\sum_{j \neq i} p_{ij} q_{ij} Z (y_i - y_j) - \sum_{j \neq i} q_{ij}^2 Z (y_i - y_j) \right]$$

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

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

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

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

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 Theoretical Results

2.1 General framework

We identify a general framework for studying dimensionality reduction algorithms that is based on the following observation: each dimensionality reduction 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.

2.2 Implementation differences

Despite using similar methods, there are several important differences in the implementation of these algorithms.

- Similar to the kernel normalization, tSNE normalizes the repulsive forces by the sum of distances from every point to every other point in the dataset. In contrast, UMAP only applies repulsive forces with respect to a constant number of points. In practice, this means that the repulsive forces in UMAP are done per-point while the repulsive forces in tSNE are applied as an average across all of the points.

- 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 their attractive and repulsive forces. This requires UMAP to apply n repulsions for every attractive force. This also 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.

- 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 $\rho_i = \min_{k \neq i} d(x_i, x_k)$. There is a lot of work done to justify this choice in both the papers and presentations.

- 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.

- tSNE applies the attractive force between y_i and y_j to only y_i's position whereas UMAP applies the attractive force to both y_i and y_j

2.3 Consolidating tSNE and UMAP

We show that one can obtain tSNE directly through the UMAP framework subject to a few approximations. The main obstacle to overcome is to undo the normalization differences between tSNE and UMAP. In the naive implementation, running UMAP's code with tSNE's normalization, kernels, initialization, and batch gradient descent does not give a satisfactory result.

2.4 Uniform UMAP

We propose a modified implementation of UMAP that we call Uniform UMAP. The principle difference consists in the gradient descent methodology. 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. However, assuming that the projection dimensionality d is sufficiently low, many of these gradient operations will either stack or cancel, as the number of points N is significantly larger than the dimensionality d.

The first modification changes the gradient application methodology. Rather than updating positions within the for-loop we can instead collect all of the attractive and repulsive forces and apply them simultaneously at the end of the training. This in practice removes the need for calculating multiple repulsions for every attraction, cutting down on the number of gradients 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.

Thus, we implement several approximation heuristics that facilitate faster convergence for UMAP. For example, the following situation inevitably occurs throughout the training process: 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 then repel along (y_i, y_l). Due to randomness, many individual repulsive forces are negligible. Thus, the previously described gradient updates will oftentimes be approximately equal to performing the attraction (y_i, y_j) at time t twice and applying the two repulsive forces in tandem. Furthermore, if we apply the attractive forces 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.

3 Experimental Results

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.

4 Left to Show

This is a scratchwork section for identifying remaining topics of study

- tSNE and UMAP
  - It would be good to show that the additional KL sum in UMAP directly accounts for the change in normalization
  - Is it possible to get tSNE directly from UMAP's code?
  - Why does tSNE not work in UMAP's code??
- Uniform UMAP
  - When can we replace k applications of an attractive force with a single application of a stronger attractive force
  - Why does Uniform UMAP not need muliple repulsions for each attraction?
  - GPU implementation of Uniform UMAP?
- General
  - Why do the PCA plots look the way that they do?