On UMAP's True Loss Function

Anonymous Author(s)
Affiliation
Address
email

Abstract

UMAP has supplanted t-SNE as state-of-the-art for visualizing high-dimensional datasets in many disciplines, but the reason for its success is not well understood. In this work, we investigate UMAP's sampling based optimization scheme in detail. We derive UMAP's effective loss function in closed form and find that it differs from the published one. As a consequence, we show that UMAP does not aim to reproduce its theoretically motivated high-dimensional UMAP similarities. Instead, it tries to reproduce similarities that only encode the shared k nearest neighbor graph, thereby challenging the previous understanding of UMAP's effectiveness. Alternatively, we consider the implicit balancing of attraction and repulsion due to the negative sampling to be key to UMAP's success. We corroborate our theoretical findings on toy and single cell RNA sequencing data.

1 Introduction

Today's most prominent methods for non-parametric, non-linear dimension reduction are t-Distributed Stochastic Neighbor Embedding (t-SNE) [17, 16] and Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) [11]. The heart of UMAP is claimed to be its sophisticated method for extracting the high-dimensional similarities, motivated in the language of algebraic topology and category theory. However, the reason for UMAP's excellent visualizations is not immediately obvious from this approach. In particular, UMAP's eponymous uniformity assumption is arguably difficult to defend for the wide range of datasets on which UMAP performs well. Therefore, it is not well understood what exactly about UMAP is responsible for its great visualizations.

Both t-SNE and UMAP have to overcome the computational obstacle of considering the quadratic number of interactions between all pairs of points. The breakthrough for t-SNE came with a Barnes-Hut approximation [16]. Instead, UMAP employs a sampling based approach to avoid a quadratic number of repulsive interactions. Other than [3] little attention has been paid to this sampling based optimization scheme. In this work, we fill this gap and analyze UMAP's optimization method in detail. In particular, we derive the effective, closed form loss function which is truly minimized by UMAP's optimization scheme. While UMAP's use of negative sampling was intended to avoid quadratic complexity, we find, surprisingly, that the resulting effective loss function differs significantly from UMAP's purported loss function. The weight of the loss function's repulsive term is drastically reduced. As a consequence, UMAP is not actually geared towards reproducing the clever high-dimensional similarities. In fact, we show that most information beyond the shared kNN graph connectivity is essentially ignored as UMAP actually approximates a binarized version of the high-dimensional similarities. These theoretical findings underpin some empirical observations in [3] and demonstrate that the gist of UMAP is not in its high-dimensional similarities. This resolves the disconnect between UMAP's uniformity assumption and its success on datasets of varying density.

From a user's perspective it is important to gain an intuition for deciding which features of a visualization can be attributed to the data and which ones are more likely artifacts of the visualization method.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

With our analysis, we can explain UMAP's tendency to produce crisp, locally one-dimensional substructures as a side effect of its optimization.

Without the motivation of reproducing sophisticated high-dimensional similarities in embedding space, it seems unclear why UMAP performs well. We propose an alternative explanation for UMAP's success: The sampling based optimization scheme balances the attractive and repulsive loss terms despite the sparse high-dimensional attraction. Consequently, UMAP can leverage the connectivity information of the shared kNN graph via gradient descent effectively.

2 Related work

For most of the past decade t-SNE [17, 16] was considered state-of-the-art for non-linear dimension reduction. In the last years UMAP [11] at least ties with t-SNE. In both methods, points are embedded so as to reproduce high-dimensional similarities. Different from t-SNE, the high-dimensional similarities are sparse for UMAP and do not need to be normalized over the entire dataset. Additionally, t-SNE adapts the local scale of high-dimensional similarities by achieving a predefined perplexity, while UMAP uses its uniformity assumption. The low-dimensional similarity functions also differ. Recently, Böhm et al. [3] placed both UMAP and t-SNE on a spectrum of dimension reduction methods that mainly differ in the amount of repulsion employed. They argue that UMAP uses less repulsion than t-SNE. A parametric version of UMAP was proposed in [15].

UMAP's success, in particular in the biological community [1, 14], sparked interest in understanding UMAP more deeply. The original paper [11] motivates the choice of the high-dimensional similarities using concepts from algebraic topology and category theory and thus justifies UMAP's transition from local similarities $\mu_{i\to j}$ to global similarities μ_{ij}. The authors find that while the algorithm focuses on reproducing the local similarity pattern similar to t-SNE, it achieves better global results than t-SNE. In contrast, Kobak and Linderman [8] attribute the better global properties of UMAP visualizations to the more informative initialization and show that t-SNE manages to capture more global structure if initialized in a similar way.

Narayan et al. [12] observe that UMAP's uniformity assumption leads to visualizations in which denser regions are more spread out while sparser regions get overly contracted. They propose an additional loss term that aims to reproduce the local density around each point and thus spaces sparser regions out. We provide an additional explanation for overly contracted visualizations: UMAP does not reproduce the high-dimensional similarities but exaggerates the attractive forces over the repulsive ones, which can result in overly crisp visualizations, see Figures 1b and 2a.

Our work aligns with Böhm et al. [3]. The authors conjecture that the sampling based optimization procedure of UMAP prevents the minimization of the supposed loss function, thus not reproducing the high-dimensional similarities in embedding space. They substantiate this hypothesis by qualitatively estimating the relative size of attractive and repulsive forces. In addition, they implement a Barnes-Hut approximation to the loss function (6) and find that it yields a diverged embedding. We analyze UMAP's sampling procedure in depth and compute UMAP's true loss function in closed form and contrast it against the supposed loss in Section 5. Based on this analytic effective loss function, we can further explain Böhm et al. [3]'s empirical finding that the specific high-dimensional similarities provide little gain over the binary weights of a shared kNN graph,[1] see Section 6. Finally, our theoretical framework leads us to a new tentative explanation for UMAP's success in Section 7.

3 Background on UMAP

The key idea of UMAP [11] is to compute pairwise similarities in high-dimensional space which inform the optimization of the low-dimensional embedding. Let $x_1, ..., x_n \in \mathbb{R}^D$ be high-dimensional, mutually distinct data points for which low-dimensional embeddings $e_1, ..., e_n \in \mathbb{R}^d$ shall be found, where $d \ll D$, often $d = 2$ or 3. First, UMAP extracts high-dimensional similarities between the data points. To do so, the k nearest neighbor (kNN) graph is computed. Let $i_1, ..., i_k$ denote the indices of x_i's k nearest neighbors in increasing order of distance to x_i. Then, using its uniformity assumption, UMAP fits a local notion of similarity for each data point i by selecting a scale σ_i such

[1] The shared k nearest neighbor graph contains an edge ij if i is among j's k nearest neighbors or vice versa.

that the total similarity of each point to its k nearest neighbors is normalized, i.e. find σ_i such that

$$\sum_{\kappa=1}^{k} \exp\left(-(d(x_i, x_{i_\kappa}) - d(x_i, x_{i_1}))/\sigma_i\right) = \log_2(k). \tag{1}$$

This defines the directed high-dimensional similarities

$$\mu_{i\to j} = \begin{cases} \exp\left(-(d(x_i, x_j) - d(x_i, x_{i_1}))/\sigma_i\right) & \text{for } j \in \{i_1, \dots, i_k\} \\ 0 \text{ else.} \end{cases} \tag{2}$$

Finally, these are symmetrized to obtain undirected high-dimensional similarities or input similarities between items i and j

$$\mu_{ij} = \mu_{i\to j} + \mu_{j\to i} - \mu_{i\to j}\mu_{j\to i} \in [0, 1]. \tag{3}$$

While each node has exactly k non-zero directed similarities $\mu_{i\to j}$ to other nodes which sum to $\log_2(k)$, this does not hold exactly after symmetrization. Nevertheless, typically the μ_{ij} are highly sparse, each node has positive similarity to about k other nodes and the degree of each node $d_i = \sum_{j=1}^{n} \mu_{ij}$ is approximately constant and close to $\log_2(k)$, see Figures 5 and 6 in Appendix B. For convenience of notation, we set $\mu_{ii} = 0$ and define the total similarity as $\mu_{\text{tot}} = \frac{1}{2}\sum_{i=1}^{n} d_i$.

Distance in embedding space is transformed to low-dimensional similarity by a smooth approximation to the high-dimensional similarity function, $\phi(d; a, b) = (1 + ad^{2b})^{-1}$, using the same slack and scale for all points. The shape defining parameters a, b are essentially hyperparameters of UMAP. We will overload notation and write

$$\nu_{ij} = \phi(||e_i - e_j||) = \phi(e_i, e_j) \tag{4}$$

for the low-dimensional or embedding similarities and usually suppress their dependence on a and b.

Supposedly, UMAP approximately optimizes the following objective function with respect to the embeddings $e_1, \dots, e_n$:

$$\mathcal{L}(\{e_i\}|\{\mu_{ij}\}) = -2 \sum_{1\le i<j\le n} \mu_{ij} \log(\nu_{ij}) \qquad + (1 - \mu_{ij}) \log(1 - \nu_{ij}) \tag{5}$$

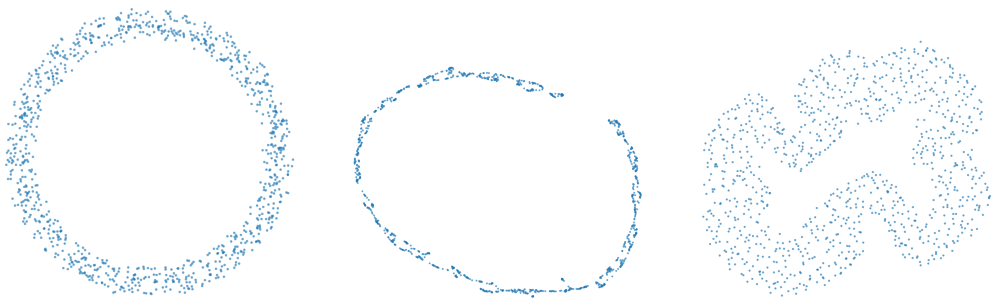
$$= -2 \sum_{1\le i<j\le n} \mu_{ij} \underbrace{\log(\phi(e_i, e_j))}_{-\mathcal{L}^a_{ij}} + (1 - \mu_{ij}) \underbrace{\log(1 - \phi(e_i, e_j))}_{-\mathcal{L}^r_{ij}}. \tag{6}$$

While the high-dimensional similarities μ_{ij} are symmetric, UMAP's implementation does consider their direction during optimization. For this reason, our losses in equations (5) and (6) differ by a factor of 2 from the one given in [11]. Viewed through the lens of a force-directed model, the derivative of the first term in each summand of the loss function, $-\partial\mathcal{L}^a_{ij}/\partial e_i$, captures the attraction of e_i to e_j due to the high-dimensional similarity μ_{ij} and the derivative of the second term, $-\partial\mathcal{L}^r_{ij}/\partial e_i$, represents the repulsion that e_j exerts on e_i due to a lack of similarity in high dimension, $1 - \mu_{ij}$. Alternatively, the loss can be seen as the sum of binary cross entropy losses for each pairwise similarity. Thus, it is minimized if the low-dimensional similarities ν_{ij} exactly match their high-dimensional counterparts μ_{ij}, that is, if UMAP manages to perfectly reproduce the high-dimensional similarities in low-dimensional space.

UMAP uses a sampling based stochastic gradient descent to optimize the low-dimensional embedding typically starting from a Laplacian Eigenmap layout [2, 8]. Our main contribution is to show that the sampling based optimization in fact leads to a different objective function, so that UMAP does not reproduce the high-dimensional similarities in low-dimensional space, see Sections 4 to 6.

4 UMAP does not reproduce high-dimensional similarities

UMAP produces scientifically useful visualizations for several domains and is fairly robust to its hyperparameters. Since any visualization of intrinsically high-dimensional data must be somehow unfaithful, it is not straightforward to check the visualization quality other than by its downstream use. Some quantitative measures exist such as the Pearson correlation between high- and low-dimensional distances used e.g. in [8, 1]. We follow a different route to show unexpected properties of UMAP. Consider the toy example of applying UMAP to data that is already low-dimensional, such that no



(a) Original data (b) UMAP (c) UMAP from dense similarities

Figure 1: UMAP does not preserve the data even when no dimension reduction is required. **1a:** Original data consisting of 1000 points sampled uniformly from a ring in 2D. **1b:** Result of UMAP after 10000 epochs, initialized with the original data. The circular shape is visible but the ring width is nearly completely contracted. **1c:** Result of UMAP after 10000 epochs for dense input similarities computed from the original data with ϕ, initialized at the original embedding. No change from the initialization would be optimal in this setting. Instead the output has spurious curves and larger width. Additional figures with the default number of epochs and initialization can be found in Figure 11 in the Appendix.

reduction in dimension is required: $D = d = 2$. Ideally, the data would be preserved in this situation. One might also expect that UMAP achieves its aim of perfectly reproducing the input similarities. Surprisingly, neither of these expectations is met. In Figure 1, we depict 2D UMAP visualizations of a two-dimensional uniform ring dataset. To ease UMAP's task, we initialized the embedding with the original data, hoping that UMAP's optimization would just deem this layout to be optimal. We also used a longer run time to ensure convergence of UMAP's optimization procedure. All other hyperparameters were kept at their default values. The results with all hyperparameters at their defaults are qualitatively similar and shown in Figure 11 in the Appendix. UMAP manages to capture the ring shape of the data, but changes its appearance significantly. UMAP contracts the width of the ring nearly to a line, see Figure 1b, a phenomenon also observed on many real-world datasets, see Figure 2a. Whether this exaggeration of the ring shape is useful depends on the use case. Note that this finding goes beyond Narayan et al. [12]'s observation that over-contraction happens in regions of low density since our toy dataset is sampled uniformly from a circular ring.

As described in Section 3, UMAP employs different methods in input and embedding space to transform distances to similarities. In particular, in input space similarities are zero for all but the closest neighbors, while in embedding space they are computed with the heavy-tailed function ϕ. To test whether this prevents the reproduction of the input data, we also computed the dense similarities on the original data with ϕ and used these as input similarities for the embedding in Figure 1c. Since we also initialize with the original dataset, a global optimum of the objective function (6) for this choice of input similarity, one would expect no change by UMAP's optimization scheme. However, we observe that in this setting UMAP produces spurious curves and increases the width of the ring.

Böhm et al. [3] implemented a Barnes-Hut approximation of UMAP's objective function (6), which produced a diverged embedding. Inspired by this finding, we compute the loss values according to equation (5) for various input and embedding similarities μ_{ij} and ν_{ij} in our toy example, see Table 1. Consider the row with the usual input similarities ($\mu_{ij} = \mu(\{x_1, \dots, x_n\})$). In a completely diverged embedding, all self similarities are one and all others zero ($\nu_{ij} = \mathbb{1}(i == j)$). We find that the loss for such an embedding is lower than for the optimized UMAP embedding. This is in accordance with Böhm et al. [3]'s Barnes-Hut experiment and shows that UMAP does not optimize its supposed objective function (6) as a diverged embedding is approximately feasible in two dimensions. This discrepancy is not just due to the fact that input and embedding similarities are computed differently: The second row of Table 1 contains loss values for the setting in which we use the dense similarities as input similarities, as in Figure 1c. We initialize the embedding at the optimal loss value ($\nu_{ij} = \phi(\{x_1, \dots, x_n\}) = \mu_{ij}$), but UMAP's optimization moves away from this layout and towards an embedding with higher loss ($\nu_{ij} = \phi(\{e_1, \dots, e_n\})$) although we always compute similarity in the same manner. Clearly, UMAP's optimization yields unexpected results.

Table 1: UMAP loss value for various combinations of input and embedding similarities, μ_{ij}, ν_{ij}, of the toy example in Figure 1. The loss for the UMAP embedding (middle column) is always higher than for another two-dimensional layout (**bold**). Hence, UMAP does not minimize its purported loss. Results are averaged over 7 runs and one standard deviation is given, see also Appendix E.1.

	Embedding similarities ν_{ij}		
Input similarities μ_{ij}	$\mathbb{1}(i == j)$ (diverged layout)	$\phi(\{e_1, \dots, e_n\})$ (UMAP result)	$\phi(\{x_1, \dots, x_n\})$ (input layout)
$\mu(\{x_1, \dots, x_n\})$	$\mathbf{62959 \pm 82}$	70235 ± 1301	136329 ± 721
$\phi(\{x_1, \dots, x_n\})$	902757 ± 2788	331666 ± 1308	$\mathbf{224584 \pm 8104}$

5 UMAP's sampling strategy and effective loss function

UMAP uses a sampling based approach to optimize its loss function, in order to reduce complexity. A simplified version of the sampling procedure can be found in Algorithm 1. Briefly put, an edge ij is sampled according to its high-dimensional similarity and the embeddings e_i and e_j of the incident nodes are pulled towards each other. For each such sampled edge ij, the algorithm next samples m negative samples s uniformly from all nodes and the embedding of i is repelled from that of each negative sample. Note that the embeddings of the negative samples are not repelled from that of i, see commented line 10. So there are three types of gradient applied to an embedding e_i during an epoch:

1. e_i is pulled towards e_j when edge ij is sampled, see line 5
2. e_i is pulled towards e_j when edge ji is sampled, see line 6
3. e_i is pushed away from embeddings of negative samples when some edge ij is sampled, see line 9.

The full update of embedding e_i during epoch t is, according to UMAP's implementation, given by

$$g_i^t = \sum_{j=1}^{n} X_{ij}^t \cdot \frac{\partial \mathcal{L}_{ij}^a}{\partial e_i} + X_{ji}^t \cdot \frac{\partial \mathcal{L}_{ji}^a}{\partial e_i} + X_{ij}^t \cdot \sum_{s=1}^{n} Y_{ij,s}^t \cdot \frac{\partial \mathcal{L}_{is}^r}{\partial e_i}, \tag{7}$$

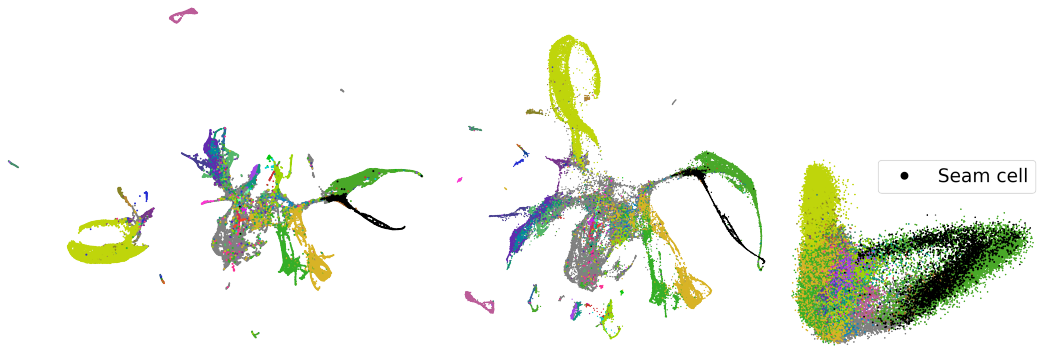
where X_{ab}^t is the binary random variable indicating whether edge ab was sampled in epoch t and $Y_{ab,s}^t$ is the random variable for the number of times s was sampled as negative sample for edge ab in epoch t if ab was sampled in epoch t and zero otherwise. By construction, $\mathbb{E}(X_{ab}^t) = \mu_{ab}$ and $\mathbb{E}(Y_{ab,s}^t | X_{ab}^t = 1) = m/n$. Taking the expectation over the random events in an epoch, we obtain the expected update of UMAP's optimization procedure, see Appendix C for more details:

$$\mathbb{E}\left(g_i^t\right) = 2 \sum_{j=1}^{n} \mu_{ij} \cdot \frac{\partial \mathcal{L}_{ij}^a}{\partial e_i} + \frac{d_i m}{2n} \cdot \frac{\partial \mathcal{L}_{ij}^r}{\partial e_i}. \tag{8}$$

Comparing the above closed formula for the expectation of the UMAP updates of the low-dimensional embeddings, to the gradient of UMAP's loss function (6)

$$\frac{\partial \mathcal{L}}{\partial e_i} = 2 \sum_{j=1}^{n} \mu_{ij} \cdot \frac{\partial \mathcal{L}_{ij}^a}{\partial e_i} + (1 - \mu_{ij}) \cdot \frac{\partial \mathcal{L}_{ij}^r}{\partial e_i}, \tag{9}$$

we find that the sampling procedure yields the correct weight for the attractive term in expectation, as designed. However, as noticed by Böhm et al. [3], the negative sampling changes the weight for the repulsive term significantly. Our closed formula helps to make their qualitative arguments precise: Instead of $1 - \mu_{ij}$, we have a term $\frac{d_i m}{2n}$, which depends on the hyperparameter m or `negative_sample_rate`. Contrary to the intention of McInnes et al. [11], the repulsive weights are not uniform but vary with the degree of each point d_i, which, however, is typically close to $\log_2(k)$, see Appendix B. More practically, since the non-zero high-dimensional similarities are sparse, $1 - \mu_{ij}$ is equal to 1 for most ij. In contrast, the expected repulsive weight is typically small for large datasets as the numerator $d_i m$ is of the order of $\log_2(k)m$ independent of the dataset size n but the denominator scales with n.



(a) UMAP (b) Inverted weights UMAP (c) PCA

Figure 2: UMAP on C. elegans data from [14, 12]. **2a:** UMAP visualization with the hyperparameters of [12]. Several parts of the embedding appear locally one-dimensional, for instance the seam cells. **2b:** Same as 2a but with inverted positive high-dimensional similarities. The result is qualitatively similar, if not better. **2c:** Two dimensional PCA of the dataset. Highlighted seam cells clearly have two dimensional variance in the PCA plot, but are overly contracted to nearly a line in the UMAP plots 2a and 2b. Full legend with all cell types and further information can be found in Figure 14. Figure best viewed in color.

Another effect of the negative sampling is that in general the expected update (8) does not correspond to any loss function, see Appendix D. We remedy this by additionally pushing the embedding of a negative sample i away from the embedding e_j, whenever i was sampled as negative sample to some edge jk, see line 10 in Algorithm 1. This yields the following gradient at epoch t

$$\tilde{g}_i^t = \sum_{j=1}^{n} \left(X_{ij}^t \cdot \frac{\partial \mathcal{L}_{ij}^a}{\partial e_i} + X_{ji}^t \cdot \frac{\partial \mathcal{L}_{ji}^a}{\partial e_i} + X_{ij}^t \cdot \sum_{s=1}^{n} Y_{ij,s}^t \cdot \frac{\partial \mathcal{L}_{is}^r}{\partial e_i} + \sum_{k=1}^{n} X_{jk}^t Y_{jk,i}^t \cdot \frac{\partial \mathcal{L}_{ji}^r}{\partial e_i} \right), \tag{10}$$

corresponding to a loss in epoch t of

$$\tilde{\mathcal{L}}^t = \sum_{i,j=1}^{n} \left(X_{ij}^t \cdot \mathcal{L}_{ij}^a + \sum_{s=1}^{n} X_{ij}^t Y_{ij,s}^t \cdot \mathcal{L}_{is}^r \right). \tag{11}$$

Using the symmetry of μ_{ij}, $\mathcal{L}_{ij}^a$ and $\mathcal{L}_{ij}^r$ in i and j, we compute the effective loss

$$\tilde{\mathcal{L}} = \mathbb{E}(\tilde{\mathcal{L}}^t) = 2 \sum_{1 \le i < j \le n} \mu_{ij} \cdot \mathcal{L}_{ij}^a + \frac{(d_i + d_j)m}{2n} \cdot \mathcal{L}_{ij}^r. \tag{12}$$

In fact, the above remedy of also pushing the negative samples does not affect the behavior of UMAP qualitatively, see for instance Figures 12 and 15.[2] In this light, we can treat $\tilde{\mathcal{L}}$ as the effective objective function that is optimized via SGD by UMAP's optimization procedure. It differs from UMAP's loss function (6), by having a drastically reduced repulsive weight of $\frac{(d_i+d_j)m}{2n}$ instead of $1 - \mu_{ij}$.

We illustrate our analysis on gene expression measurements of 86024 cells of C. elegans [14, 12]. We start out with a 100 dimensional PCA of the data[3] and use the cosine metric in high-dimensional space, consider $k = 30$ neighbors and optimize for 750 epochs as done in [12]. The resulting visualization is depicted in Figure 2a. On this dataset the average value of $1 - \mu_{ij}$ is 0.9999 but the maximal effective repulsive weight $\max_{ij} \frac{(d_i+d_j)m}{2n}$ is 0.0043, showing the dramatic reduction of repulsion due to negative sampling. After each optimization epoch, we log our effective loss $\tilde{\mathcal{L}}$ (12), the actual loss $\tilde{\mathcal{L}}^t$ (11) of each epoch computed based on the sampled (negative) pairs as well the purported UMAP loss $\mathcal{L}$ (6).[4] We always consider the embeddings at the end of each epoch. Note

[2] In fact, the Parametric UMAP [15] does include the update of negative samples.

[3] obtained from `http://cb.csail.mit.edu/cb/densvis/datasets/`. We informed the authors of our use of the dataset, which they license under CC BY-NC 2.0.

[4] Our code will be made publicly available upon acceptance. It is included in the supplementary material.

that UMAP's implementation updates each embedding e_i not just once at the end of the epoch but as soon as i is incident to a sampled edge or sampled as a negative sample. This difference does not change the actual loss much, see Appendix E. The result is plotted in Figure 3. We can see that our predicted loss matches its actual counterpart nearly perfectly. While both, $\tilde{\mathcal{L}}$ and $\tilde{\mathcal{L}}^t$, agree with the attractive part of the supposed UMAP loss, its repulsive part and thus the total loss are two orders of magnitude higher. Furthermore, driven by the repulsive part, the total intended UMAP loss increases during much of the optimization process, while the actual and effective losses decrease, exemplifying that UMAP really optimizes our effective loss $\tilde{\mathcal{L}}$ (12) instead of its purported loss $\mathcal{L}$ (6). Additional loss curves for UMAP on CIFAR-10 [9] can be found in Appendix F.

After having deduced the effective loss function of Non-Parametric UMAP, we conclude this Section with the corresponding but slightly different result for Parametric UMAP [15]:

Theorem 5.1. *The effective loss function of Parametric UMAP with batch size b is*

$$-\frac{1}{(m+1)\mu_{tot}} \sum_{1\le i<j\le n} \mu_{ij} \log\Big(\phi\big(f_\theta(x_i), f_\theta(x_j)\big)\Big) + m\frac{b-1}{b}\frac{d_i d_j}{2\mu_{tot}} \log\Big(1-\phi\big(f_\theta(x_i), f_\theta(x_j)\big)\Big). \tag{13}$$

Proof. The proof can be found in Appendix A and differs from the Non-Parametric case mostly because the negative samples come from the current batch instead of the full dataset. □

While Parametric UMAP's loss differs slightly from $\tilde{\mathcal{L}}$ (12) the same analysis holds unless explicitly mentioned. We mention a relation of Parametric UMAP to modularity clustering in Appendix A.1.

Algorithm 1: UMAP's optimization

input : input similarities μ_{ij},
initial embeddings $e_1, \dots, e_n$,
number of epochs T,
learning rate α,
number of negative samples m

output : final embeddings $e_1, \dots, e_n$

for $t = 0$ **to** T **do**
  for $ij \in \{1, \dots, n\}^2$ **do**
    $r \sim \text{Uniform}(0, 1)$
    if $r < \mu_{ij}$ **then**
      $e_i = e_i - \alpha \cdot \frac{\partial \mathcal{L}^a_{ij}}{\partial e_i}$
      $e_j = e_j - \alpha \cdot \frac{\partial \mathcal{L}^a_{ij}}{\partial e_j}$
      for $l = 1$ **to** m **do**
        $s \sim \text{Uniform}(\{1, \dots, n\})$
        $e_i = e_i - \alpha \cdot \frac{\partial \mathcal{L}^r_{is}}{\partial e_i}$
        `// Next line is omitted in UMAP's implementation, but included for our analysis`
        /* $e_s = e_s - \alpha \cdot \frac{\partial \mathcal{L}^r_{is}}{\partial e_s}$ */

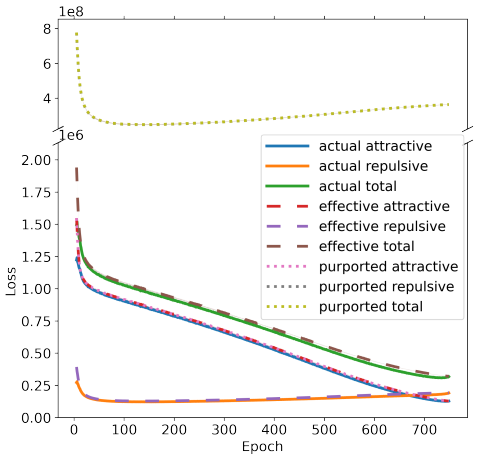


Figure 3: Loss curves for the optimization leading to Figure 2a. Our effective loss (12) closely matches the actual loss (11), while the purported UMAP loss (6) is two orders of magnitude higher. The total overlays the repulsive purported loss. An average over 7 runs is plotted with shaded areas of one standard deviation, see also Appendix E.1. Figure best viewed in color.

6 True target similarities

Since the effective objective function $\tilde{\mathcal{L}}$ (12) that UMAP optimizes is different from $\mathcal{L}$ (6), we cannot hope that UMAP truly tries to find a low-dimensional embedding whose similarities reproduce the high-dimensional similarities. Nevertheless, using the effective loss $\tilde{\mathcal{L}}$, we can compute the true target similarities ν^*_{ij} which UMAP tries to achieve in embedding space. The effective loss $\tilde{\mathcal{L}}$ is a sum of non-normalized binary cross entropy loss functions

$$-\left(\mu_{ij} \cdot \log(\nu_{ij}) + \frac{(d_i + d_j)m}{2n} \cdot \log(1 - \nu_{ij})\right) \tag{14}$$

which is minimal for

$$\nu^*_{ij} = \frac{\mu_{ij}}{\mu_{ij} + \frac{(d_i+d_j)m}{2n}} \begin{cases} = 0 \text{ if } \mu_{ij} = 0 \\ \approx 1 \text{ if } \mu_{ij} > 0. \end{cases} \tag{15}$$

The approximation holds in the typical case in which $\frac{(d_i+d_j)m}{2n} \approx 0$, discussed above. In other words, compared to the reduced repulsion weight even the smallest high-dimensional similarity appears maximal. Thus, the negative sampling essentially binarizes the input similarities. UMAP's high-dimensional similarities are non-zero exactly on the shared k-nearest neighbor graph edges of the high-dimensional data. Therefore, the binarization explains why Böhm et al. [3] find that using the binary weights of the shared k nearest neighbor graph does not deteriorate UMAP's performance much.[5] The binarization even helps UMAP to overcome disrupted high-dimensional similarities, as long as only the edges of the shared kNN graph have non-zero weight. In Figure 2b, we invert the original positive high-dimensional weights on the C. elegans dataset. That means that the k-th nearest neighbor will have higher weight than the nearest neighbor. The resulting visualization even improves on the original by keeping the layout more compact. This underpins Böhm et al. [3]'s claim that the elaborate theory used to compute the high-dimensional similarities is not the reason for UMAP's practical success. In fact, we show that UMAP's optimization scheme even actively ignores most information beyond the shared kNN graph. In Figure 4, we show histograms of the various notions of similarity for the C. elegans dataset. We see in panel 4b how the binarization equalizes the positive target similarities for the original and the inverted high-dimensional similarities.

The binary cross entropy terms in the effective loss $\tilde{\mathcal{L}}$ (12) are not normalized. This leads to a different weighing of the binary cross entropy terms for each pair ij

$$\tilde{\mathcal{L}} = 2 \sum_{1 \le i < j \le n} \mu_{ij} \cdot \mathcal{L}^a_{ij} + \frac{(d_i + d_j)m}{2n} \cdot \mathcal{L}^r_{ij} \tag{16}$$

$$= -2 \sum_{1 \le i < j \le n} \left(\mu_{ij} + \frac{(d_i + d_j)m}{2n} \right) \cdot \left(\nu^*_{ij} \log(\nu_{ij}) + (1 - \nu^*_{ij}) \log(1 - \nu_{ij}) \right). \tag{17}$$

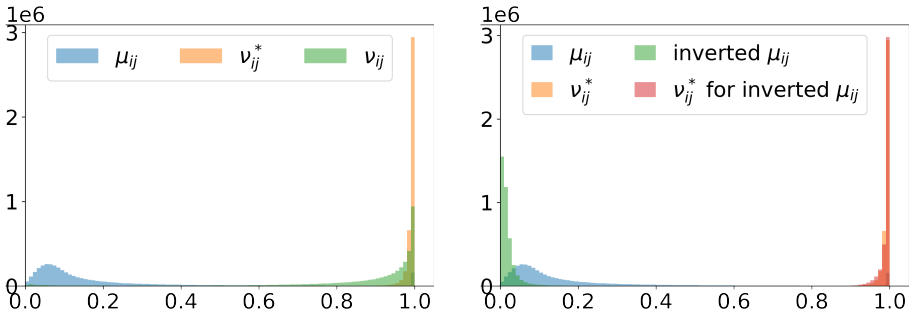
As $\frac{(d_i+d_j)m}{2n}$ is very small for large datasets, the term $\mu_{ij} + \frac{(d_i+d_j)m}{2n}$ is dominated by μ_{ij}. Hence, the reduced repulsion not only binarizes the high-dimensional similarities, it also puts higher weight on the positive than the zero target similarities. Therefore, we can expect that the positive target similarities are better approximated by the embedding similarities than the zero ones. Indeed, panel 4a shows that the low-dimensional similarities match the positive target similarities very well, as expected from the weighted BCE reading of the effective loss function (17).

6.1 Explaining artifacts in UMAP visualizations

We conclude this section by explaining the observed artifacts of UMAP's visualization in Figures 1 and 2 in the light of the above analysis. The normal UMAP optimization contracts the ring in Figure 1b even when initialized at the original layout because the reduced repulsion yields nearly binary target similarities. All pairs that are part of the kNN graph not only want to be sufficiently close that their high-dimensional similarity is reproduced, but so close that their similarity is one. The fact that the effective loss weighs the terms with target similarity near one much more than those with target similarity near zero reinforces this trend. As a result, the ring gets contracted to a circle. The same argument applies to the over contracted parts of the UMAP visualization of the C. elegans dataset in Figure 2.

Our framework can also explain the opposite behavior of UMAP when the dense similarities are used as input similarities, see Figure 1c. In this setting, the average degree of a node is about 100. With a `negative_sample_rate` of 5 and a dataset size of $n = 1000$ this yields repulsive weights of about $\frac{(d_i+d_j)m}{2n} \approx 0.5$. Thus, we increase the repulsion on pairs with high input similarity, but decrease it on pairs with low input similarity. Now, the target similarities are not a binarization of the input similarities, but instead skewed towards 0.5. Consequently, we can expect embedding points to increase their distance to nearest neighbors, but distant points to move closer towards each other. This is what we observe in Figure 1c, where the width of the ring has increased and the ring curves to bring distant points closer together.

[5] Böhm et al. [3] used a scaled version of the kNN graph, but the scaling factor cancels for the target weights.



(a) Similarities for $\mu_{ij} > 0$ (b) Original and inverted similarities for $\mu_{ij} > 0$

Figure 4: Histograms of high-dimensional (μ_{ij}), target (ν^*_{ij}) and low-dimensional (ν_{ij}) similarities on the C. elegans dataset [14, 12] for pairs with positive high-dimensional similarity. **4a:** The similarities of UMAP's low-dimensional embedding reproduce the target similarities instead of the high-dimensional ones. The target similarities are heavily skewed towards one. **4b:** Comparison of positive input and target similarities for the original and inverted input similarities. While the histograms of the input similarities differ noticeably, their target similarities do not. The binarization essentially ignores all information beyond the shared kNN graph.

7 Discussion

By deriving UMAP's true loss function and target similarities, we are able to explain several peculiar properties of UMAP visualizations. According to our analysis, UMAP does not aim to reproduce the high-dimensional UMAP similarities in low dimension but rather the binary weights of the shared kNN graph of the input data. This raises the question just what part of UMAP's optimization leads to its excellent visualization results. Apparently, the exact formula for the repulsive weights is not crucial as it differs for Non-Parametric UMAP and Parametric UMAP while both produce similarly high quality embeddings. A first tentative step towards an explanation might be the different weighing of the BCE terms in the effective loss function (17). Focusing more on the similar rather than the dissimilar pairs might help to overcome the imbalance between an essentially linear number of attractive and a quadratic number of repulsive pairs. Inflated attraction was found beneficial for t-SNE as well, in the form of early exaggeration [10].

Put another way, the decreased repulsive weights result in comparable total attractive and repulsive weights, which might facilitate the SGD based optimization. Indeed, the total attractive weight in UMAP's effective loss function is $2\mu_{\text{tot}} = \sum_{i,j=1}^{n} \mu_{ij}$ and the total repulsive weight amounts to $2m\mu_{\text{tot}} = \sum_{i,j=1}^{n} \frac{(d_i+d_j)m}{2n}$ for Non-Parametric UMAP. For Parametric UMAP the total attractive and repulsive weights are $1/(m+1)$ and $m/(m+1) \cdot (b-1)/b$, respectively. For the default value of $m = 5$, the total attractive and repulsive weights are of roughly the same order of magnitude. Moreover, we observe in Figure 3 that the resulting attractive and repulsive losses are also of comparable size. Using UMAP's purported loss function, however, would yield dominating repulsion. A more in-depth investigation as to why exactly balanced attraction and repulsion is beneficial for a useful embedding is interesting and left for future work.

8 Conclusion

In this work, we investigated UMAP's optimization procedure in depth. In particular, we computed UMAP's effective loss function analytically and found that it differs slightly between the non-parametric and parametric versions of UMAP and significantly from UMAP's alleged loss function. The optimal solution of the effective loss function is typically a binarized version of the high-dimensional similarities. This shows why the sophisticated form of the high-dimensional UMAP similarities does not add much benefit over the shared kNN graph. Instead, we conjecture that the resulting balance between attraction and repulsion is the main reason for UMAP's great visualization capability. Our analysis can explain some artifacts of UMAP visualizations.

References

[1] E. Becht, L. McInnes, J. Healy, C.-A. Dutertre, I. W. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell. Dimensionality reduction for visualizing single-cell data using UMAP. *Nature Biotechnology*, 37(1):38–44, 2019.

[2] M. Belkin and P. Niyogi. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002. URL `https://proceedings.neurips.cc/paper/2001/file/f106b7f99d2cb30c3db1c3cc0fde9ccb-Paper.pdf`.

[3] J. N. Böhm, P. Berens, and D. Kobak. A Unifying Perspective on Neighbor Embeddings along the Attraction-Repulsion Spectrum. *arXiv preprint arXiv:2007.08902*, 2020.

[4] B. Charlier, J. Feydy, J. A. Glaunès, F.-D. Collin, and G. Durif. Kernel Operations on the GPU, with Autodiff, without Memory Overflows. *Journal of Machine Learning Research*, 22(74):1–6, 2021. URL `http://jmlr.org/papers/v22/20-275.html`.

[5] E. Davis, S. Sethuraman, et al. Consistency of modularity clustering on random geometric graphs. *Annals of Applied Probability*, 28(4):2003–2062, 2018.

[6] W. Dong, C. Moses, and K. Li. Efficient K-Nearest Neighbor Graph Construction for Generic Similarity Measures. In *Proceedings of the 20th International Conference on World Wide Web*, pages 577–586, 2011.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[8] D. Kobak and G. C. Linderman. Initialization is critical for preserving global data structure in both t-SNE and UMAP. *Nature Biotechnology*, pages 1–2, 2021.

[9] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[10] G. C. Linderman and S. Steinerberger. Clustering with t-SNE, Provably. *SIAM Journal on Mathematics of Data Science*, 1(2):313–332, 2019.

[11] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[12] A. Narayan, B. Berger, and H. Cho. Assessing single-cell transcriptomic variability through density-preserving data visualization. *Nature Biotechnology*, pages 1–10, 2021.

[13] M. E. Newman. Analysis of weighted networks. *Physical review E*, 70(5):056131, 2004.

[14] J. S. Packer, Q. Zhu, C. Huynh, P. Sivaramakrishnan, E. Preston, H. Dueck, D. Stefanik, K. Tan, C. Trapnell, J. Kim, et al. A lineage-resolved molecular atlas of C. elegans embryogenesis at single-cell resolution. *Science*, 365(6459), 2019.

[15] T. Sainburg, L. McInnes, and T. Q. Gentner. Parametric UMAP: learning embeddings with deep neural networks for representation and semi-supervised learning. *arXiv preprint arXiv:2009.12981*, 2020.

[16] L. Van Der Maaten. Accelerating t-SNE using Tree-Based Algorithms. *Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[17] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] As claimed, we compute UMAP's true loss in Section 5 and show that most information beyond the shared kNN graph is ignored in Section 6 before proposing a new possible explanation for UMAP's success in Section 7, see also the end of Section 2 for this paper road map.
   (b) Did you describe the limitations of your work? [Yes] We acknowledge in Section 7 that our new explanation for UMAP's success is tentative and requires more in-depth validation.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We discuss societal impacts of our work in Appendix G.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Our assumption is UMAP's optimization procedure, which we outlined at the beginning of Section 5 and in Algorithm 1 as well as in Appendix A for Parametric UMAP. We also discuss a small deviation in the loss computation and its logging in Section 5 and in more detail in Appendices D and E.
   (b) Did you include complete proofs of all theoretical results? [Yes] The two major theoretical results are the effective loss functions of Non-Parametric and Parametric UMAP, which we derive in Section 5, together with Appendices C and D, and Appendix A, respectively.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include the code of all our experiments and scripts generating or downloading the data in the supplementary material. The README explains how to reproduce all numbers and figures.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We specified the hyperparameters used for the toy datasets in Section 4 and Figures 11, 12, 13. We mostly stuck to the default values and explained when we deviated from the default in Section 4. The hyperparameters for the C. elegans dataset are given in Section 5, where we also state that these are the same as in [12]. Finally, we used the default UMAP hyperparameters for the CIFAR-10 experiments, as stated in Appendix F. While the first two datasets do not have a split, we used the training and test set of CIFAR-10 jointly for the unsupervised UMAP dimension reduction.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Whenever we state loss values or run times of our experiments, we give averages and standard deviations over seven runs, see Tables 1, 2 and Figures 3, 7, 8, 10.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Confer Appendix E.2, where we discuss the hardware used, the run times of our main experiments and an estimate of the total compute needed to reproduce the results of the paper.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cited both the original creators of the C. elegans dataset, Packer et al. [14], and Narayan et al. [12], whose preprocessed version of the dataset we used in our experiments. Moreover, we cited Krizhevsky [9] for the CIFAR-10 dataset.
   (b) Did you mention the license of the assets? [Yes] We mention the license of the C. elegans dataset in footnote 3 and that of CIFAR-10 in footnote 7.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We included our code including scripts to generate the toy datasets in the supplemental material. We will make the code public upon acceptance.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We state in footnote 3 that we informed Narayan et al. [12] that we are using their version of the C. elegans dataset. Its license CC BY-NC 2.0 allows adaptation. The CIFAR-10 dataset is one of the most widely used datasets in machine learning. Its MIT license also permits adaptation.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] C. elegans is a 1mm roundworm.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] No crowdsourcing or research with human subjects was performed.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] No crowdsourcing or research with human subjects was performed.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] No crowdsourcing or research with human subjects was performed.