# Stat 185 Final Paper: t-SNE

**Stat 185**
*November 19, 2021* Max Guo

## 1 Introduction

High dimensional data is ubiquitous in our data-driven society, with common examples including gene combinations, images, and documents (viewed as collections of many words). Modern applications require methods for visualizing high dimensional data in order to better explore and understand the data. This problem is difficult, since we often wish for our visualization method to satisfy a number of properties, including: preserving local structures in the original high dimensional data, preserving global structure of the data, handling outliers appropriately, and computational feasibility. Naturally, a myriad of techniques exist for data visualization. We only briefly state these techniques, but papers such as [2] survey the field in more detail.

Geometric, iconographic, pixel-based, hierarchical, graph-based, and hybrid visualization techniques all attempt to provide tools for visualization but all suffer from interpretability when applied to high dimensional datasets [7]. Dimension reduction techniques offer another route. These techniques, including the linear method Principal Component Analysis and nonlinear methods such as Sammon Mapping, ISOMAP, Locally Linear Embedding, and Stochastic Neighbor Embedding (SNE), map high dimensional data to low dimensional data (e.g. 2, 3 dimensions) and then visualize the latter via a scatterplot. However, these techniques are often unable to capture both the local and global structures of the data [7]. These techniques also may fail when attempting to visualize real world data.

This motivates the t-SNE method for data visualization of high dimensional data. Introduced by Laurens van der Maaten and Geoffrey Hinton in 2008, the t-SNE method is based off of the aforementioned SNE [7]. Similar to SNE, t-SNE constructs a similarity matrix for the data and then attempts to construct low dimensional representations of the data by minimizing some notion of distance between the similarity matrix for the original data and the similarity matrix for the low dimensional data [3]. However, compared to SNE, t-SNE is better able to capture local clusters and a more global structure of the data. t-SNE has also been shown to perform well on messy real world datasets [7].

In this paper, we will formally introduce the method of t-SNE in Section 2, discuss the intuition behind the method and compare the model to SNE and selected techniques in more technical detail in Section 3, and concretely perform t-SNE on some example datasets in Section 4. We conclude in Section 5.

## 2 Method

At a high level, t-SNE constructs two notions of similarity via probabilities: one between the high dimensional data points, and one between the resulting low dimensional representations. It then minimizes the differences in the two similarity metrics by adjusting the low dimensional data points (via gradient descent). This results in low dimensional representations of the data whose internal similarities resemble the internal similarities of the original high dimensional data.

Our presentation of the method, including notation and order of presentation, unless otherwise noted, is based off of the original presentation of t-SNE by Laurens van der Maaten and Geoffrey Hinton. As in their paper, we will simultaneously introduce SNE, since t-SNE is built upon SNE. This will allow us to better motivate some of the techniques and draw comparisons between SNE and t-SNE. However, unlike the paper, we defer detailed discussion of the motivations behind t-SNE to Section 3 for the sake of brevity and clarity. For every intentional choice we will refer the reader to the appropriate justification later in the paper.

*Remark.* All notation will be introduced concurrently with the method as needed.

## 2.1 Problem Setup and High Level Intuition

Let $x_1, x_2, \ldots, x_n$ be the original high dimensional data points which we want to visualize, and let $y_1, y_2, \ldots, y_n$ be the corresponding low dimensional data points. The similarity notion we choose between high dimensional data points is captured within the quantities $p_{j|i}, 1 \leq i, j \leq n$, which can be viewed as the conditional probability that $x_i$ would choose $x_j$ as its neighbor over other points in proportion to its probability density under a Gaussian centered at $x_i$. Likewise, we choose a similarity notion between the corresponding low dimensional data points using the quantities $q_{j|i}, 1 \leq i, j \leq n$, which is the conditional probability that $y_i$ would choose $y_j$ as its neighbor over other points in proportion to its probability density under a separate distribution (which differs between SNE and t-SNE). Intuitively, then, our low dimensional visualization will be accurate if its similarities resemble the high dimensional similarities. This leads us to the first main idea behind t-SNE, which is also a main idea behind SNE.

**Main Idea 1 of t-SNE (and vanilla SNE)**

$y_i$ and $y_j$ model the similarity between $x_i$ and $x_j$ if $p_{j|i} \approx q_{j|i}$.

In the following section we will find a way to formally define the objective of finding low dimensional data representations to achieve the desired approximation.

## 2.2 Mathematical Formulation

In this section, we first define how similarities between points are represented via conditional probabilities in SNE and present a preliminary mathematical objective function for Main Idea 1. We then introduce symmetric SNE with its joint probability representations and refine our objective function. Finally, we describe the last major revision of t-SNE from SNE, which is the choice of a Student-t distribution for the low dimensional probability densities.

### 2.2.1 Conditional Probabilities in SNE

Let $\sigma_i$ be the standard deviation of the Gaussians centered around data points $x_i$. We will describe the approach to choose the standard deviations later in Section 2.2.5. Then we define:

$$p_{j|i} = \frac{\exp\left(-\frac{||x_i - x_j||^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{||x_i - x_k||^2}{2\sigma_i^2}\right)} \tag{1}$$

for all $1 \leq i, j \leq n$, $i \neq j$. We take $p_{i|i} = 0$ for all $i$, as we only wish to model pairwise similarities. Actually, Equation (1) implicitly encodes a core assumption behind t-SNE that we will discuss later on:

**Core Assumption 1 of t-SNE (and vanilla SNE)**

The underlying data manifold is locally linear and not pathological. (Section 3.3.1)

For the low dimensional similarities, SNE defines:

$$q_{j|i} = \frac{\exp(-||y_i - y_j||^2)}{\sum_{k \neq i} \exp(-||y_i - y_k||^2)}. \tag{2}$$

As before, $q_{i|i} = 0$ for all $1 \leq i \leq n$. Note that we take all of the standard deviations to be the same (i.e. $\sqrt{2}/2$) because we do not care too much about differentiating visualizations of denser clusters and sparser clusters.

Now, to formalize the objective, we can minimize the Kullback-Leibler divergences between the conditional distri-

butions $p_{j|i}$ and $q_{j|i}$ over all the points, which we define to be the *cost* $C$:

$$C = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}. \tag{3}$$

If $y_1^*, \ldots, y_n^*$ are our optimal low dimensional representations, then we wish to find:

$$y_1^*, \ldots, y_n^* = \underset{y_1, \ldots, y_n}{\text{argmin}}\, C, \tag{4}$$

and this is done via gradient descent on $C$ with respect to each of the $y_i$'s. However, the gradient descent is computationally difficult (see Section 3.1.1), which motivates reformulation of the similarities as joint probabilities.

### 2.2.2 Joint Probabilities in Symmetric SNE

In symmetric SNE, we define the joint probability distribution over the data points in the low dimensional space by the pairwise similarities:

$$q_{ij} = \frac{\exp\big(-||y_i - y_j||^2\big)}{\sum_{k \neq l} \exp(-||y_k - y_l||^2)}. \tag{5}$$

where $q_{ii} = 0$ for all $i$. Note that $q_{ij} = q_{ji}$ and that this is a valid joint distribution because $\sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} = 1$. For the data points in the high dimensional space, we define $p_{ij}$ in a different way in order to resolve outlier issues (Section 3.1.2):

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}. \tag{6}$$

Note also that $p_{ij} = p_{ji}$ for all $i, j$, and $p_{ii} = 0$ for all $i$. This is also a valid probability distribution:

$$\begin{aligned} \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} &= \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{p_{j|i} + p_{i|j}}{2n} \\ &= \frac{1}{2n} \left( \sum_{j=1}^{n} \sum_{i=1}^{n} p_{i|j} + \sum_{i=1}^{n} \sum_{j=1}^{n} p_{j|i} \right) \\ &= 1, \end{aligned}$$

since $\sum_{j=1}^{n} p_{j|i} = \sum_{i=1}^{n} p_{i|j} = 1$ for all $i, j$.

Now we wish to minimize the difference between the *joint* probability distributions of the high dimensional points and the low dimensional points.

**Main Idea 2 of t-SNE**

$y_i$ and $y_j$ model the similarity between $x_i$ and $x_j$ if $p_{ij} \approx q_{ij}$.

As in the case of SNE, symmetric SNE defines the cost to be:

$$C = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{7}$$

This has the gradient of the form:

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j=1}^{n} (p_{ij} - q_{ij})(y_i - y_j), \tag{8}$$

and we optimize $C$ via gradient descent to find the optimal low dimensional representations as in Equation (4).

#### 2.2.3 Using the Student-t Distribution

The final change that t-SNE implements is using a Student-t distribution with one degree of freedom in place of a Gaussian for the low dimensional joint probabilities. This solves the *crowding problem*, which will be discussed in Section 3.1.3.

**Main Idea 3 of t-SNE**

$y_i$ and $y_j$ better model the similarity between $x_i$ and $x_j$ if $q_{ij}$ is based on a Student-t probability density.

This encodes another underlying assumption we make about our data:

**Core Assumption 2 of t-SNE**

If two points are moderately far away compared to their respective nearest neighbors, these two points should repel each other in the visualization. (Section 3.1.3)

Thus, our new definition of $q_{ij}$ would be:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}} \tag{9}$$

using the probability density function of the Student-t distribution with one degree of freedom. Though this does not change the definition of the cost as in Equation (7), it does change the gradient:

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j=1}^{n} (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}. \tag{10}$$

#### 2.2.4 Optimization Techniques

Because Equation (7) is non-convex (Section 3.3), we use gradient descent to optimize the t-SNE objective. Furthermore, we utilize a momentum term in the gradient descent to prevent our optimization from returning local minima. In the original paper, the authors propose a number of additional optimization techniques including an adaptive learning rate scheme, early compression, and early exaggeration. Early compression encompasses techniques which forces the $y_i$'s to stay close together at the beginning and allows clusters to easily move through one another at the beginning. Early exaggeration involves multiplying the $p_{ij}$'s in the beginning so they are much too large to be modeled by the $q_{ij}$'s, so that the optimization focuses on modeling large $p_{ij}$'s with large $q_{ij}$'s and thus creates widely separated clusters that can then be moved around easily to find a decent global structure.

This finishes the overview of the algorithm for t-SNE. Finally, we return to the question of choosing $\sigma_i$ for each of the Gaussians in calculating $p_{j|i}$ in Equation (1).

#### 2.2.5 Choosing $\sigma_i$

Note that, if we have a sparse cluster and a dense cluster in the original high dimensional data, we will likely choose different values of $\sigma_i$ for the data points in the two different clusters. Intuitively, it does not make sense to have a large $\sigma$ value for a dense cluster, as then the similarity values within different points of the cluster will be approximately equally large, when in reality we wish to distinguish them. The same issue appears if we choose too small of a $\sigma$ value for a sparse cluster.

Let $P_i$ be the conditional probability distribution over all other data points given data point $x_i$ and $\sigma_i$. Then we

define:

$$H(P_i) = -\sum_{j=1}^{n} p_{j|i} \log_2(p_{j|i}) \tag{11}$$

as the *Shannon entropy* of $P_i$, and the *perplexity* of $P_i$ as

$$Perp(P_i) = 2^{H(P_i)}. \tag{12}$$

The perplexity of $P_i$ is an increasing function of $\sigma_i$. Thus, t-SNE can choose the $\sigma_i$ values by finding, for each data point $x_i$, the $\sigma_i$ value that results in a user specified perplexity. This can be done efficiently via binary search.

We conclude this section with pseudocode of the algorithm taken from the original paper:

**Algorithm 1**: Simple version of t-Distributed Stochastic Neighbor Embedding.

**Data**: dataset $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations $T$, learning rate $\eta$, momentum $\alpha(t)$.
**Result**: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
**begin**
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  **for** *t=1* **to** *T* **do**
    compute low-dimensional affinities $q_{ij}$ (using Equation 12)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 13)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  **end**
**end**

Figure 1: t-SNE Algorithm

# 3 Discussion

In the previous section, we detailed the steps of t-SNE as well as the intuition behind each step. In this section, we continue motivating some of the choices by discussing some of the improvements of t-SNE over the original SNE that we deferred from the previous section (Section 3.1). We also give an overview of the advantages of t-SNE over existing nonlinear dimension reduction methods in Section 3.2. In Section 3.3, we summarize some of the weaknesses encountered by t-SNE. Finally, we conclude this section by discussing some of the practical recommendations for measuring performance in Section 3.4 and some more recent work in the theory behind t-SNE in Section 3.5.

## 3.1 Advantages over SNE

### 3.1.1 Easier-to-compute Gradients

In the original SNE method, the gradient of the cost function looked like:

$$\frac{\partial C}{\partial y_i} = 2\sum_{j=1}^{n}(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j) \tag{13}$$

Comparing this to the final gradient in Equation (10), we note that the denominator of $q_{ij}$ need only be computed once in the gradient of t-SNE, whereas each $q_{j|i}$ has a different denominator for different $i$. Moreover, the $q_{ij}$ no longer has an exponential, making it much easier to compute. Thus, the change from asymmetric to symmetric SNE (and then to t-SNE) results in a more computationally efficient gradient.

### 3.1.2 Resolving Outliers

Other than Equation (6), another (perhaps more) natural choice of defining $p_{ij}$ would be:

$$p_{ij} = \frac{\exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)}{\sum_{k \neq l} \exp\left(-\frac{||x_k - x_l||^2}{2\sigma^2}\right)} \tag{14}$$

for some choice of $\sigma$. However, for an outlier $x_i$ far away from any data point, $p_{ij}$ is small for all $j$, which implies that the contribution of $y_i$ to $C$ is small and the location of $y_i$ cannot really be determined well. However, in Equation (6), we note that

$$\begin{aligned} \sum_{j=1}^{n} p_{ij} &> \frac{1}{2n} \sum_{j=1} p_{j|i} \\ &\geq \frac{1}{2n} \end{aligned}$$

In other words, each data point $x_i$ (even outliers) makes a significant contribution to the cost function, so their low dimensional representations are more well determined.

### 3.1.3 Resolving the Crowding Problem (Core Assumption 2)

A fundamental issue in representing high dimensional data points in a lower dimension is that there is "less space" in a lower dimension than in the high dimension. For example, in 3 dimensions, you can have four distinct points that have pairwise distances of 1 (e.g. consider a regular tetrahedron). However, this is impossible in 2 dimensions. In general, what tends to happen in SNE visualizations is that points that are well spaced out in the high dimensions are squished together in the low dimensional representation simply because there is not enough space to fit them all. This prevents natural clusters from appearing in the low dimensional representation.

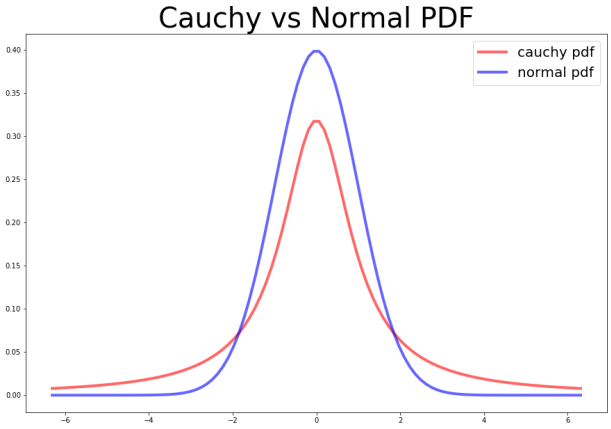


Figure 2: Student-t distribution with 1 degree of freedom (Cauchy) vs. Normal Distribution

To remedy this, t-SNE uses a Student-t distribution with 1 degree of freedom (also known as a Cauchy distribution) for the joint probabilities $q_{ij}$. Notice in Figure 2 how the Cauchy distribution has much fatter tails than the normal distribution. This means that, when matching the similarities $q_{ij}$ with $p_{ij}$, t-SNE will allow moderately spaced data points in the original high dimensions to retain their spacing in the low dimensional representations.

Theoretically, the authors note that a Student-t distribution is an infinite mixture of Gaussians. Moreover, Equa-

tion (9) approaches an inverse square as the pairwise distances $y_i - y_j$ grows larger, so the low dimensional representation is almost invariant to scale for large distances.

### 3.2 Advantages over other methods

Now that we've discussed some of the intuition behind t-SNE and its advantages over SNE, we give an overview of t-SNE's advantages over a few dimension reduction methods we covered in class.

*Linear Methods.* Note that PCA and Classical Multidimensional Scaling and similar linear methods for dimensionality reduction and visualization are quite restricted in their abilities to model nonlinear data manifolds. Similarly, they are unlikely to show separation in clusters for general distributions of clusters. PCA also requires the centering of data, which is not required in t-SNE because we only care about pairwise distances.

*Sammon Mapping.* In Sammon mapping, nonlinear data manifolds are represented via minimizing the following cost function (provided in [7]):

$$C = \frac{1}{\sum_{ij} ||x_i - x_j||} \sum_{i \neq j} \frac{(||x_i - x_j|| - ||y_i - y_j||)^2}{||x_i - x_j||} \tag{15}$$

As discussed in class, the denominator in the sum means that Sammon mapping places a large cost on data points which are close to each other in the high dimensional space. However, note that, between different small distances, the smallest ones contribute more to the cost. Intuitively, we should want clusters with varying small distances between points to contribute equally to the cost, and this is what t-SNE does. t-SNE, by employing Gaussian kernels with varying levels of variance for each point, regards less the magnitude of the small differences and instead captures the relative difference of inter-cluster and intra-cluster separations.

*ISOMAP.* Recall that ISOMAP creates a neighborhood graph and then attempts to estimate distances by using local Euclidean maps and then shortest path algorithms. However, this both suffers from computational complexity considerations (up to a cubic in time complexity, according to lecture.) Moreover, as mentioned in [7], ISOMAP can "short-circuit" if there are noisy points that allow for shorter paths between estimates. t-SNE is much more robust to noise because a noisy point does not affect too much its estimates of similarities, and it's also less resource-consuming time-wise.

*LLE.* Local Linear Embedding also uses a neighborhood graph in order to build a low dimensional representation that models the local geometry between points. However, in LLE there may be multiple ways of satisfying the constraint function that are non-optimal (e.g. using lots of points in the center of a map and a few points far away). With t-SNE, there are reasonable guarantees of performance (Section 3.5). Furthermore, both LLE and ISOMAP are incapable of representing disjoint manifolds because the neighborhood graphs will be disconnected, whereas t-SNE does well in representing disconnected clusters.

### 3.3 Weaknesses

Though t-SNE has many strengths in comparison with traditional techniques, there are also several weaknesses, or generally less desirable properties of t-SNE. These weaknesses are synthesized from [7] and [8].

1. *Higher Dimensions.* t-SNE as currently presented is optimal for only two or three dimensions (so primarily for data visualization) and may not generalize well to higher dimensions. In particular, the heavy tails of the Student-t distribution may need to be adjusted to accommodate for the space available in higher dimensions.
2. *Nonconvex Objective.* Note that the objective function of tSNE (Equation (7)) is nonconvex, so it requires much more complicated optimization techniques than standard convex methods and may lead to local optima.
3. *Hyperparameter Tuning.* Note that t-SNE has (at least) two hyperparameters that must be tuned, including

the perplexity and the step count (number of iterations to run Gradient Descent). With untuned parameters, t-SNE can result in poorly constructed visualizations.

4. *Non-interpretability.* Certain parts of the resulting t-SNE visualization are widely varying depending on how you choose your hyperparameters. For example, cluster sizes and the distances between clusters have little meaning in t-SNE due to their dependence on the perplexity and step count.

5. *Large Datasets.* Note that the t-SNE algorithm is quadratic in time and space complexity with respect to the data, so it cannot feasibly be run on more than $10^5$ data points. While random techniques exist, including a graph-based technique presented in [7], none of them circumvent at least one pass through the data of quadratic complexity.

#### 3.3.1 Local Linearity and the Curse of Dimensionality (Core Assumption 1)

Note that the numerator of $p_{j|i}$ as defined in Equation (1) is an exponential which, for small distances, can be Taylor approximated by:

$$\exp\left(-\frac{||x_i - x_j||^2}{2\sigma_i^2}\right) \approx 1 - \frac{||x_i - x_j||^2}{2\sigma_i^2} \tag{16}$$

For nearest neighbors, t-SNE solely uses Euclidean distances to determine the proportion. This assumes a non-pathological, linear data manifold. Thus, t-SNE may perform worse if the data has high intrinsic dimensionality.

### 3.4 Practical Recommendations

There are not too many recommended guidelines for measuring t-SNE's visualization performance other than quality of visualization because of t-SNE's weaknesses. However, we discuss a few ideas from [5] and [8].

First, because t-SNE is heavily dependent on hyperparameters and is non-deterministic due to optimization techniques for a non-convex objective function, it is recommended that the algorithm is run multiple times with different perplexities and step sizes, but this may run into the issue of obtaining different cluster forms with different hyperparameter choices ([8]).

Moreover, t-SNE doesn't preserve any of the global distances, so a natural benchmark of correlation between original distances and distances in the low dimensional representation may not capture the quality of the result from t-SNE. More recent methods like UMAP may result in better correlations than t-SNE ([5]).

One final idea that may or may not be helpful is that of clustering on the low dimensional representations generated by t-SNE. If clustering algorithms correctly cluster the original data and are also able to cluster the t-SNE representations correctly, that implies t-SNE successfully kept clusters apart. However, this idea is difficult in practice, especially because t-SNE may sometimes find clusters where they don't exist ([8]).

### 3.5 Theoretical Work

In the original work on t-SNE, no theoretical guarantees were provided, and the sole basis for the method was more intuition and experimentally based. However, due to the widespread usage of t-SNE since its publication, a small subfield now exists which examines the theoretical guarantees of the method. We only give an overview of a few interesting results from [1] and [4]. This is still an active field of research today.

In [4], four theoretical guarantees of t-SNE are provided. Informally, these are:

- *Exponential Convergence.* The t-SNE algorithm with parameters following a certain criteria converges exponentially, without needing optimization techniques such as momentum.
- *Spectral Clustering.* The t-SNE algorithm behaves like a spectral clustering algorithm and can thus be analyzed through the framework of spectral clustering.

- *Disjoint Clusters.* Though not guaranteed that clusters in the original data are disjoint in the low dimensional representation, it is more likely than not that this will be the case.
- *Independence of initialization.* None of these results are too overly dependent on the initialization of the low dimensional representation, within reason.

Finally, this next section gives a more detailed overview of the work in [1] that provides a theoretical basis for analyzing visualizations and applies it to t-SNE.

#### 3.5.1 Clusterability in Well-separated, Spherical Data

Now we will give a scenario in which t-SNE has provable guarantees for finding a good visualization. This requires defining what a good visualization is and what well, separated, spherical data is. Because of the complexity of the formalism in [1], we describe these informally.

In the following definitions, suppose that $\mathcal{X} = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$ has ground truth clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$. Let $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\} \subset \mathbb{R}^2$ be the 2-dimensional embedding of $\mathcal{X}$.

**Definition 3.1** (Visible Cluster). A cluster $\mathcal{C}_l$ in $\mathcal{X}$ is visible in $\mathcal{Y}$ if the corresponding points to $\mathcal{C}_l$ in $\mathcal{Y}$ are well-separated from the remaining points $\mathcal{Y}$.

**Definition 3.2** (Good Visualization). $\mathcal{Y}$ is a good visualization if every cluster in $\mathcal{X}$ is a visible cluster in $\mathcal{Y}$.

However, even the authors of [1] note that this line of theoretical work does not fully capture the human's capacity to visualize. For example, two intersecting parallel lines may be conceived as two clusters but would violate this definition. Even so, this notion of a good visualization nicely captures a number of scenarios, so we continue with this definition.

**Definition 3.3** (Well Separated, Spherical Data). If $\mathcal{X}$ has $n$ points and $k$ clusters and $|C_l| \geq 0.1(n/k)$ (cluster sizes are not too widely varying), then we say $\mathcal{X}$ is approximately spherical if, for every cluster, points in the same cluster are roughly concentrated around the same value. Furthermore, $\mathcal{X}$ is well separated if for any two clusters, the inter-cluster distance is somewhat larger than intra-cluster distances in both clusters.

Given the following definitions, we informally state the paper's main theorem:

**Theorem 3.4.** Suppose $\mathcal{X}$ is well separated and spherical. Then t-SNE using early exaggeration (Section 2.2.4) on input $\mathcal{X}$ outputs a good visualization $\mathcal{Y}$ with high probability.

## 4 Examples

To illustrate the results of t-SNE, we provide an example on the MNIST dataset with the generating code in the Appendix. We also include two toy examples from [8] that illustrate some of the failures of t-SNE.

### 4.1 MNIST

We also include an example of t-SNE data run on a subset of 5000 images from the MNIST handwritten digits dataset by using scikit-learn's TSNE implementation ([6]). The code was based off of [9], and is included in the Appendix.

We see in Figure 3, without any tuning of the parameters in t-SNE (using scikit-learn's standard perplexity of 30.0, for example), we are able to achieve a much better separation of different clusters of the MNIST dataset from t-SNE as opposed to PCA. Furthermore, both the scaled and unscaled versions of the data in t-SNE result in separation (the unscaled data even more so). This illustrates some of the strengths of t-SNE.

However, in the process of creating this example, I had to take a subset of the MNIST training data. This is because the entire training set of 60000 images required too much time to run (longer than 10 minutes), whereas PCA easily computed the dimension reduction on the entire dataset in a few seconds. This illustrates the downside of t-SNE having a quadratic time complexity in the number of data points.

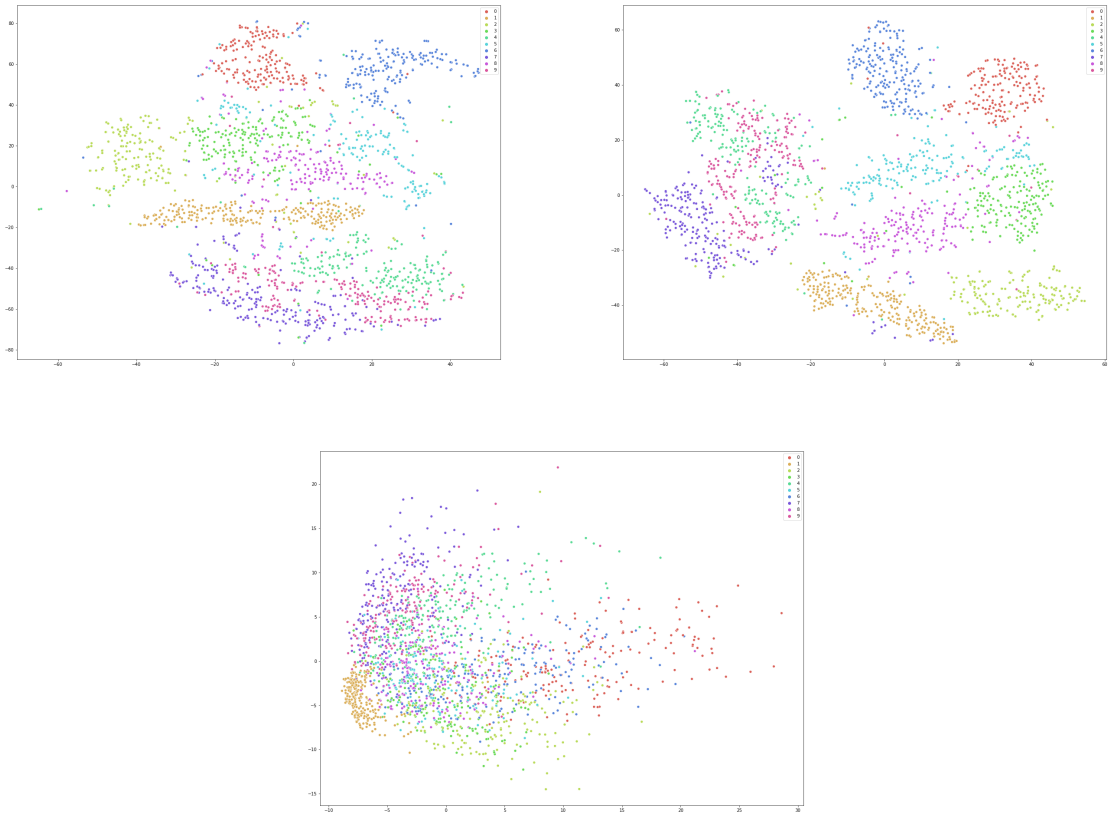

Figure 3: Top left: t-SNE on scaled data. Top right: t-SNE on unscaled data. Bottom: PCA on scaled data.

### 4.2 Toy Datasets from [8]

In an effort to demonstrate a few important points about t-SNE which may be interpreted as weaknesses, [8] creates three toy datasets, as shown in Figure 4. The first demonstrates the idea that cluster sizes and distances between clusters in the resulting low dimensional representations are mostly arbitrary and depend heavily on the hyperparameters. The second and third also emphasize the dependence of t-SNE on its hyperparameters. We can see in the second that, for fixed step size, some perplexities result in poor t-SNE performance for even this simple case of Gaussians. Similarly, in the second step we see that cluster shape is heavily dependent on step size, and there may be certain step sizes that are more desirable than others.

## 5 Conclusion

In conclusion, t-SNE is a nonlinear data visualization tool introduced in [7] that improves upon the previous Stochastic Neighborhood Embedding technique. The main idea of attempting to equate similarities between data points in the high dimension with similarities between data points in the low dimension remains the same. However, t-SNE uses joint probabilities to do both, whereas SNE uses conditional probabilities. Moreover, while a Gaussian kernel is used for both SNE and t-SNE in the original high dimension, a Student-t distribution is used in the low dimensional representation for t-SNE, which remedies the crowding problem. t-SNE also has many advantages over existing nonlinear dimension reduction methods, but it is not without weaknesses, especially in hyperparameter tuning, training, interpetability, and generalizability. Future work may include continual investigation into the theoretical guarantees into t-SNE as well as possible modifications of t-SNE that addresses some of its weaknesses. In the meantime, it remains one of the first resorts for scientists of many domains for visualizing real world data.

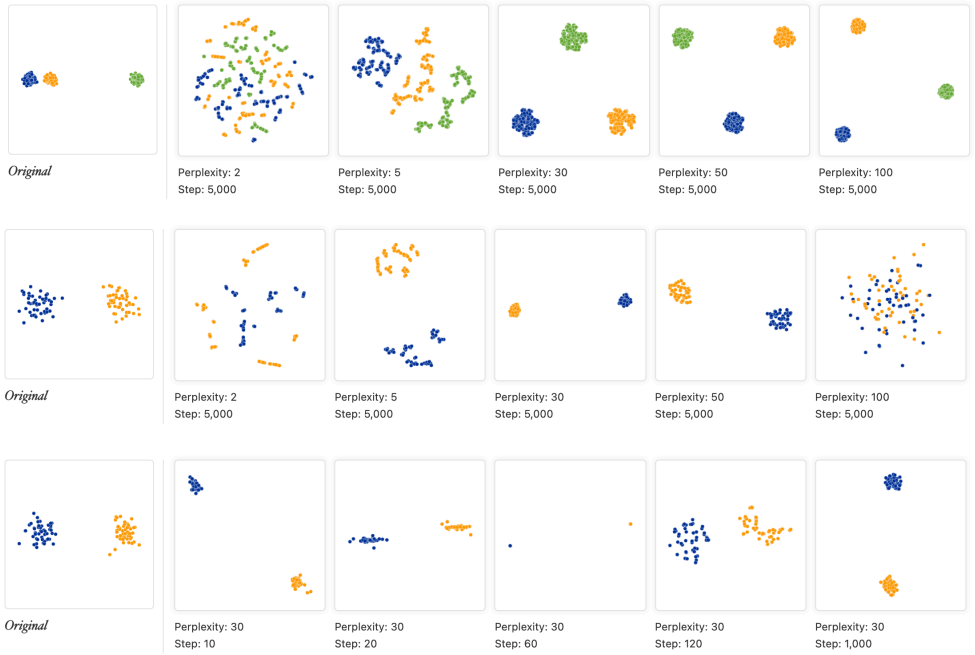


Figure 4: Top: t-SNE with varying perplexities on three different clusters. Middle: t-SNE with varying perplexities on two different clusters. Bottom: t-SNE with varying step counts on two different clusters. Each dataset is generated via a mixture of Gaussians (one per cluster).

## References

[1] Sanjeev Arora, Wei Hu, and Pravesh K. Kothari. "An Analysis of the t-SNE Algorithm for Data Visualization". In: *Proceedings of the 31st Conference On Learning Theory*. Ed. by Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet. Vol. 75. Proceedings of Machine Learning Research. PMLR, June 2018, pp. 1455–1462. URL: https://proceedings.mlr.press/v75/arora18a.html.

[2] M.C. Ferreira de Oliveira and H. Levkowitz. "From visual data exploration to visual data mining: a survey". In: *IEEE Transactions on Visualization and Computer Graphics* 9.3 (2003), pp. 378–394. DOI: 10.1109/TVCG.2003.1207445.

[3] Geoffrey Hinton and Sam T Roweis. "Stochastic neighbor embedding". In: *NIPS*. Vol. 15. Citeseer. 2002, pp. 833–840.

[4] George C Linderman and Stefan Steinerberger. "Clustering with t-SNE, provably". In: *SIAM Journal on Mathematics of Data Science* 1.2 (2019), pp. 313–332.

[5] Nikolay Oskolkov. *tSNE vs. UMAP: Global Structure*. Feb. 2021. URL: https://towardsdatascience.com/tsne-vs-umap-global-structure-4d8045acba17.

[6] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

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

[8] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. "How to Use t-SNE Effectively". In: *Distill* (2016). DOI: 10.23915/distill.00002. URL: http://distill.pub/2016/misread-tsne.

[9] Dehao Zhang. *Dimensionality Reduction using t-Distributed Stochastic Neighbor Embedding (t-SNE) on the MNIST...* Aug. 2020. URL: https://towardsdatascience.com/dimensionality-reduction-using-t-distributed-stochastic-neighbor-embedding-t-sne-on-the-mnist-9d36a3dd4521.

Class notes were also referenced throughout this paper.

# 6 Appendix

We include here the code used to generate the MNIST example in Figure 3.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.datasets import mnist
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE

np.random.seed(185)

# Obtain and scale data
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Choose small portion of the data
indices = np.random.choice(range(len(X_test)),
                           size=2000,
                           replace=False)

# Scaled data
small_data = StandardScaler().fit_transform(X_test[indices].reshape(-1, 784))

# run PCA
pca = PCA(n_components=2)
pca_res = pca.fit_transform(small_data)

plt.figure(figsize=(20, 15))
sns.scatterplot(x = pca_res[:,0],
                y = pca_res[:,1],
                hue = y_test[indices],
                palette = sns.hls_palette(10),
                legend = 'full')
plt.savefig("pca")
plt.show()

# run t-SNE with unscaled data
from sklearn.manifold import TSNE
tsne = TSNE(n_components = 2, random_state=0)
tsne_res = tsne.fit_transform(X_test[indices].reshape(-1, 784))
```

```
plt.figure(figsize=(20, 15))
sns.scatterplot(x = tsne_res[:,0],
                y = tsne_res[:,1],
                hue = y_test[indices],
                palette = sns.hls_palette(10),
                legend = 'full')
plt.savefig("tsne-mnist-no-scale")
plt.show()

# run t-SNE with scaled data
tsne = TSNE(n_components = 2, random_state=0)
tsne_res = tsne.fit_transform(small_data)

plt.figure(figsize=(20, 15))
sns.scatterplot(x = tsne_res[:,0],
                y = tsne_res[:,1],
                hue = y_test[indices],
                palette = sns.hls_palette(10),
                legend = 'full')
plt.savefig("tsne-mnist-small")
plt.show()
```