Analyzing Visualization and Dimensionality-Reduction Algorithms

Oliver Hayman[a] and Ashwin Narayan[b]

(a) Thomas Jefferson High School for Science and Technology, Alexandria, VA, USA
(b) Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

In order to find patterns among high dimensional data sets in scientific studies, scientists use mapping algorithms to produce representative two-dimensional or three-dimensional data sets that are easier to visualize. The most prominent of these algorithms is the t-Distributed Stochastic Neighbor Embedding algorithm (t-SNE). In this project, we create a metric for evaluating how clustered a data set is, and use it to measure how the perplexity parameter of the t-SNE algorithm affects the clustering of outputted data sets. Additionally, we propose a modification in which improved how well randomness is preserved in outputted data sets. Finally, we create a separate metric to test whether a group of points contains one or multiple clusters in a data set of centered clusters.

1 Introduction

1.1 Dimension-Reduction Algorithms

One of the largest steps of any scientific study is finding patterns from collected data that can be used to support a general claim. For two or three-dimensional data sets, we can find patterns through sight alone: we can easily determine whether points lie in a line or are grouped into clusters just by plotting the data set. However, many data sets in scientific studies depend on more than just two or three variables, and we can't think about four-dimensional distributions as easily as two or three-dimensional ones. This inability to visualize high-dimensional data sets creates a huge obstacle in drawing conclusions from data.

In order to resolve this issue, we use mapping algorithms to generate two-dimensional or three-dimensional representations of high dimensional data sets that preserve relationships between data points like clusters. For example, if we pass a data set consisting of points lying on the surface of a sphere through a dimension-reduction algorithm, the outputted two-dimensional data set would ideally look like a circle.

Some information about data is lost when mapping it to a lower dimension, so these algorithms aim at preserving only *useful* information. It is important for scientists to know what kinds of information is lost when using these algorithms.

1.2 t-Distributed Stochastic Neighbor Embedding

t-Distributed Stochastic Neighbor Embedding (t-SNE) is currently the most prominent dimension-reduction algorithm.

In t-SNE, we map a set of high dimensional points $X = \{x_1, x_2, \cdots, x_n\}$ to low dimensional points $Y = \{y_1, y_2, \cdots, y_n\}$ such that for all i, y_i is a two-dimensional representation of the point x_i.

Initially, we map the distances between pairs of points in X to a probability distribution, which we can define by using a Gaussian distribution.

Definition 1.1. A **Gaussian distribution** is a probability distribution with the probability density function

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x-\bar{x}}{2\sigma^2}},$$

where σ and $\bar{x}$ are parameters. For a Gaussian-distributed data set, σ is its standard deviation and $\bar{x}$ is its mean.

For each point x_i, we let the conditional probability $p_{j|i}$ denote the probability of choosing the point x_j from X, where the probability of choosing each point is proportional to its probability density under a Gaussian distribution centered at x_i. This gives the equation

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

where $||x_i - x_j||$ denotes the distance between points x_i and x_j [1]. Conceptually, $p_{j|i}$ gives the probability of x_i choosing point x_j as a "neighbor", because $p_{j|i}$ is high when x_j is close to x_i and low if they are far apart.

It is important to note that in t-SNE, every Gaussian distribution centered at some point x_i has a different variance σ_i^2. This can be explained by the interpretation of $p_{j|i}$ as the probability of x_i choosing x_j as a neighbor. If x_i is surrounded by many close points, it would make sense for x_i to be highly unlikely to choose a neighbor that isn't close to it, whereas if x_i is in a sparse region with few close neighbors it would choose a far away neighbor with much higher probability. We can account for this by increasing the value of σ_i^2 for x_i in sparse regions, as it increases the probability density for points that are far away from x_i in the Gaussian distribution under x_i, and would in turn increase $p_{j|i}$ for far x_j.

Sparser regions in a data set tend to reveal more about its overall distribution that small packets of dense points. Therefore, we can choose each value of σ_i^2 by quantifying how much information is given by a region of points.

Definition 1.2. The **entropy** of a discrete random variable is a measure of the average rate at which it produces information. The entropy of a random variable X is computed by

$$H(X) = \sum_{x \in X} -p(x) \log p(x),$$

where $p(x)$ denotes the probability of x. The conditional entropy between two random variables X, Y is given by

$$H(X|Y) = \sum_{x \in X, y \in Y} -p(x,y) \log \frac{p(x,y)}{p(x)}.$$

In order to find each variance σ_i^2, we define a parameter of t-SNE called the *perplexity*. Conceptually, the perplexity parameter can be thought of as the expected number of neighbors around each point in X that should be grouped together. Lower perplexities tend to result in data sets with small bundles of points, while higher perplexities display larger groups of points. However, we don't know the exact effect of perplexity on t-SNE's output.

We define the random variable P_i such that P_i gives point x_j with probability $p_{j|i}$ for each $x_j \in X$. Because $p_{j|i}$ depends on σ_i^2, for each P_i we can choose a value of σ_i^2 such that $s = 2^{H(P_i)}$. These chosen values are used in t-SNE [1].

We define

$$p_{ij} := \frac{p_{i|j} + p_{j|i}}{2n}$$

to be the probability of receiving the pair of points x_i, x_j from a randomly selected pair of points in X. Note that for any i, $\sum_j p_{ij} > \frac{1}{2n}$. This prevents any outlier points x_i from being difficult to map, as if we select a pair of points according to this probability distribution, each point has at least a $\frac{1}{2n}$ chance of being in the selected pair.

We can use a Student's t-Distribution to define a similar probability q_{ij} associated with each low dimensional pair of points (y_i, y_j).

Definition 1.3. A **Student's t-Distribution** is a probability distribution with a probability density function

$$f(x) = \frac{\Gamma\left(\frac{\upsilon+1}{2}\right)}{\sqrt{\upsilon\pi}\,\Gamma\left(\frac{\upsilon}{2}\right)} \left(1 + \frac{x^2}{\upsilon}\right)^{-\frac{\upsilon+1}{2}},$$

where υ is a parameter referred to as the degree of freedom of the distribution.

We can view a Student's t-Distribution as a Gaussian distribution with "heavy" tails - that is, the distributions look similar except in rare events, which have a higher probability of occurring under a Student's t-Distribution.

We define a probability q_{ij} associated with each pair of low dimensional points y_i, y_j using a Student's t-distribution (with $\upsilon = 1$) instead of a Gaussian distribution like so:

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

We use a Student's t-Distribution for q_{ij} instead of a Gaussian distribution because mappings given by a Gaussian distribution make it difficult to visualize every data point. More specifically, when some structures are mapped to two-dimensional or three-dimensional spaces using a Gaussian distribution, points that are close to one another in X are mapped extremely close to one another in Y, making them indistinguishable [1]. This means that in order to prevent this crowding, the distances between some points in Y will need to be increased, which can be accomplished by a Student's t-Distribution as its heavier tails mean that farther distances between points have a higher probability of occurring.

In order to find the optimal mapping of X, we want to find the set Y that minimizes how "different" the low-dimensional and high-dimensional data sets are. We define a cost function to minimize for each of the points—that is, a function that evaluates how well the low dimensional distribution preserves structure in the high dimensional distribution. In t-Distributed Stochastic Neighbor Embedding, we use the cost function [1]

$$C = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

We use this function because if points x_i and x_j are close to one another in X, the cost of mapping y_i and y_j far away from one another is high, so structures like clusters are preserved. However, if points x_i and x_j are far away from one another, mapping y_i and y_j close to each other does not cost much. This means that t-SNE is good at preserving clusters, but does not preserve the distances between clusters.

We can find the minimum value of C using the gradient descent method. The gradient of the cost function can be computed to be[1]

$$\frac{\delta C}{\delta y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||)^{-1}.$$

In order to approximate the minimum cost, we initially choose each of the points $y_1, y_2, ..., y_n$ randomly from an isotropic Gaussian distribution centered at the origin with a small variance. We then iterate each of the points to approach this minimum through the equation

$$\gamma^{(t)} = \gamma^{(t-1)} + \eta \frac{\delta C}{\delta \gamma} + \alpha(t)\left(\gamma^{(t-1)} - \gamma^{(t-2)}\right),$$

where $\alpha(t)$ is a function such that each iteration's points are equal to an exponentially decaying sum of the points in previous iterations, and η is a constant that controls the "learning" rate [1].

1.3 Barnes-Hut t-SNE

One major drawback of t-SNE is that the computation of $\frac{\delta C}{\delta y_i}$ requires computing $p_{ij} \log \frac{p_{ij}}{q_{ij}}$ between all pairs of points (x_i, x_j) and (y_i, y_j), giving a computational run time of $O(N^2)$ [2]. Therefore, an accelerated implementation of t-SNE, called **Barnes-Hut t-Distributed Stochastic Neighbor Embedding**, is more commonly used.

In Barnes-Hut t-SNE, we define η_i to be the set of the closest $\lfloor 3s \rfloor$ data points to x_i, where s denotes the perplexity. Note that points outside of η_i are far away from x_i, and therefore contribute very little to p_{ij}. Finding η_i can be found in $O(sN \log N)$ through the use of vantage point trees [4].

Now, we can redefine

$$p_{j|i} = \begin{cases} \frac{\exp\left(\frac{-||x_i - x_j||^2}{2\sigma_i^2}\right)}{\sum_{k \in \eta_i} \exp\left(\frac{-||x_i - x_k||^2}{2\sigma_i^2}\right)} & x_j \in \eta_i \\ 0 & x_j \notin \eta_i \end{cases}.$$

We keep that $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$.

We break apart the gradient of C into

$$\frac{\delta C}{\delta y_i} = 4\left(\sum_{i \neq j} p_{ij}(y_i - y_j)\left(1 + ||y_i - y_j||^2\right)^{-1}\right) - 4\left(\sum_{i \neq j} q_{ij}(y_i - y_j)\left(1 + ||y_i - y_j||^2\right)^{-1}\right).$$

Because we modified p_{ij}, we can find $4\left(\sum_{i \neq j} p_{ij}(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}\right)$ by simply summing over the nonzero values of p_{ij}, which can be done in $O(sN)$ time [2]. However, the computation of $4\left(\sum_{i \neq j} q_{ij}(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}\right)$ still has time complexity $O(N^2)$.

We can use the Barnes-Hut algorithm to compute $4\left(\sum_{i \neq j} q_{ij}(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}\right)$ in $O(N \log N)$ time. In the Barnes-Hut algorithm, we create a quadtree in order to carry out one computation for each group of

[1]For an explanation of how to calculate the gradient of C, refer to [1]

similar data points. A quadtree is a tree which consists of rectangular regions, or cells, of a data set where the children of non-leaf nodes are four cells $\frac{1}{4}$ the size of their parent in northwest, northeast, southwest, and southeast locations of their parent's center. For each cell, we find its center of mass, y_{cell} and its number of points N_{cell}. A quadtree can be constructed in $O(N)$ time [3].

For some point y_i, a cell is considered a good estimate for computing forces between its points and y_i if

$$\frac{l_{cell}}{||y_i - y_{cell}||^2} < \theta,$$

where θ is a parameter of the algorithm and l_{cell} is the length of the diagonal of the cell. If a cell is considered a good estimate, then the sum of $q_{ij}(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}$ for y_j in the cell is replaced by $N_{cell}q_{icell}(y_i - y_{cell}(1 + ||y_i - y_{cell}||^2)^{-1}$, where $q_{icell} = \frac{(1+||y_i - y_{cell}||^2)^{-1}}{\sum_{k \neq l}(1+||y_k - y_l||^2)^{-1}}$. This modification allows the forces $4\left(\sum_{i \neq j} q_{ij}(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}\right)$ to be computed in $O(N \log N)$ time, allowing the gradient to also be computed in $O(N \log N)$ time [3].

1.4 Research Overview

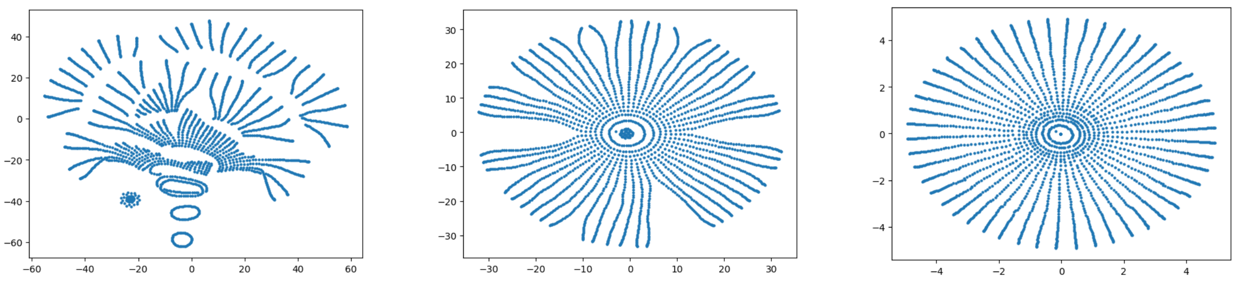
Because t-SNE is used widely across all scientific fields to analyze data sets, we aim to quantify how well certain characteristics of data sets are preserved by the algorithm. This project aims at utilizing metrics to better understand the specific behaviour of t-SNE, and to propose modifications to the algorithm that address its failures of preserving structures of certain types of data sets.

In Section 2, we propose a metric measuring how clustered a data set is and use this metric to determine how the perplexity parameter of t-SNE affects its outputted data sets. We then propose a method for computing the theoretical value of this metric. The effect of the perplexity parameter of t-SNE needs to be better understood, so creating some way to quantify the optimal perplexity for a data set would be very beneficial.

In Section 3, we propose a modification of t-SNE so that regions of high dimensional data sets that appear to be uniformly distributed also appear less clumped in low dimensional data sets. This greatly improves the interpretation of both uniform data sets and clustered data sets, because it removes the question of whether a perceived cluster in any low dimensional data set was actually part of the original data set or was created by t-SNE in an attempt to map random data.

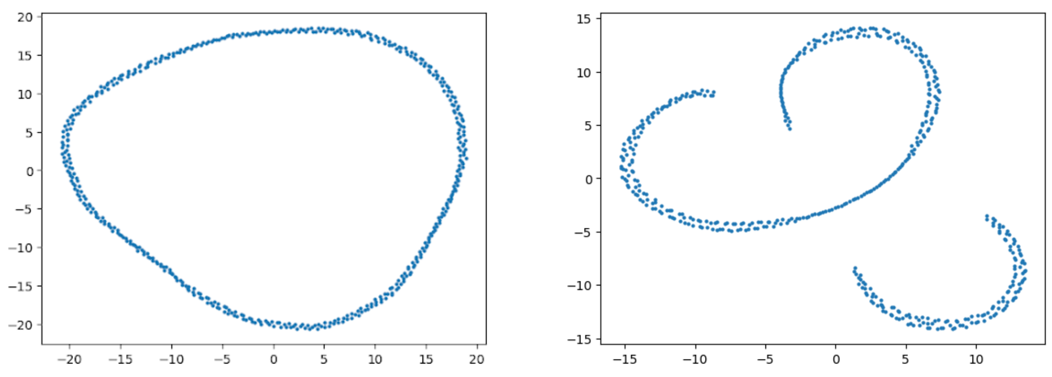
In Section 4, we propose a metric that evaluates whether a region of a high concentration of data points that would be interpreted as a cluster will actually consist of multiple centrally-distributed clusters. This aims at finding important structures of clusters that could be lost when inputted into t-SNE, and also aims at addressing failures of cluster-detection algorithms like db-SCAN [5] which would interpret multiple close-together clusters as a single cluster.

Barnes-Hut t-SNE is used in Section 2 and Section 4, while regular t-SNE is used in Section 3 as its cost function is easier to modify (although theoretically the modification could also be made to Barnes-Hut t-SNE).



(a) Perplexity = 10 (b) Perplexity = 30 (c) Perplexity = 80

Figure 1: t-SNE plots for points in a **circle** in the *xy*-plane.



(a) Perplexity = 30 (b) Perplexity = 50

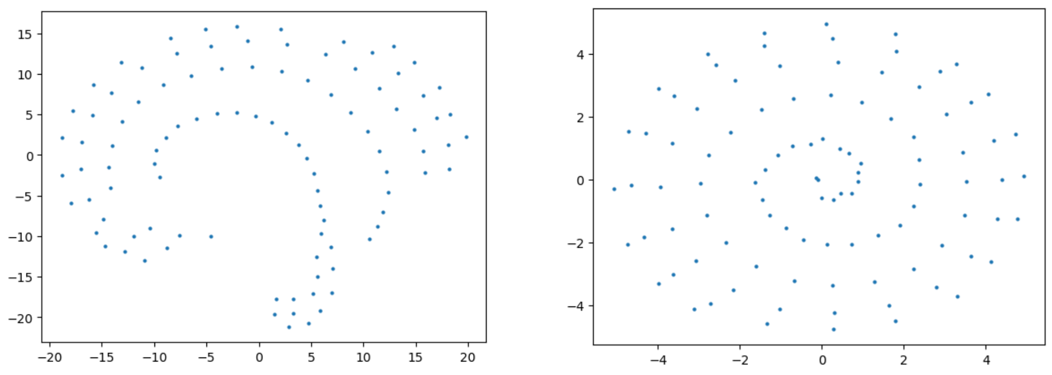
Figure 2: t-SNE plots for points in a **trefoil knot** in the *xy*-plane.

2 Evaluating the Effect of Perplexity on t-SNE

In order to modify t-SNE, we want to first develop a process to measure what specific types of inputs cause t-SNE to fail. t-SNE has two main parameters: a high-dimensional data set and a value for the perplexity. Therefore, we want to see what types of high-dimensional data sets cause t-SNE to fail and find a method to choose the best perplexity for a given data set.

We initially want to see the successes and failures of running t-SNE on data sets that form specific objects or patterns. All data sets we use are three-dimensional, because three-dimensional objects are easy to visualize, making it easy to understand what output to expect from t-SNE.

Plots of t-SNE plots with different parameters are shown for toy data sets, where points are arranged in a circle (Figure 1), a trefoil (Figure 2), and a spiral (Figure 3)



(a) Perplexity = 10 (b) Perplexity = 30

Figure 3: t-SNE plots for points in a **spiral** in the *xy*-plane.

It becomes apparent that dense structures such as points lying in a line could be pulled apart in t-SNE. This is likely because the heavier tails in the Student's t-Distribution make separating points by long distances less costly and more likely to occur, so the algorithm is more likely to separate points that are close to one another in a line.

It also becomes apparent that lower perplexities are generally weak at returning plots that resemble the expected output for a data set. The default value for the perplexity of 30 would seem like a fairly low and unreliable perplexity based off of the patterns we see, though this default could be caused by the fact that there is a cap for the highest perplexity of data sets depending on the number of points in them. In order to further investigate the behaviour of t-SNE, we focus on looking at how changing the perplexity affects the algorithm.

In order to analyze how the perplexity effects the mapping algorithm, we define a metric to measure how "clustered" a data set is, or more specifically, how well different groups of data points tend to center around some other data point.

Let $X = \{x_1, x_2, \cdots, x_n\}$ denote a high-dimensional data set inputted into t-SNE, and $Y = \{y_1, y_2, \cdots, y_n\}$ denote its low-dimensional output. Before we describe a clustering metric, we need to introduce some notation:

Definition 2.1. The function $\alpha(X, x_i, a)$ is defined to be

$$\alpha(X, x_i, a) := \left|\{x_j : x_j \in X, 0 < ||x_j - x_i|| \leq a\}\right|.$$

In other words, $\alpha(X, x_i, a)$ gives the number of points in X whose distance from x_i is less than a.

Definition 2.2. The function $\beta(X, x_i, a)$ is defined to be

$$\beta(X, x_i, a) := \min\{N : N \in \mathbb{R}, \alpha(X, x_i, N) = a\}.$$

In other words, $\beta(X, x_i, a)$ gives the distance from x_i to the ath closest point to x_i in X.

Definition 2.3. For a finite set $X = \{x_1, x_2, \cdots, x_n\}$ of d-dimensional data points and some positive integer parameter c, the metric M_{clust} is defined as:

$$M_{clust}(X, c) := \frac{1}{|X|} \sum_{x_i \in X} \alpha\left(x_i, \left(\frac{2}{c}\right)^{\frac{1}{d}} \beta(x_i, c)\right).$$

The intuition behind M_{clust} is as follows:

- First, around each point we find the radius of the smallest ball containing its c closest points (the value of β).
- We scale down this radius so that the number of points in a ball with the scaled down radius is expected to be 2. We find the actual number of data points in this ball (given by α).
- We average across all data points. This means the metric will be higher for clustered distributions, since more points are expected to lie towards the center of a ball around x_i.

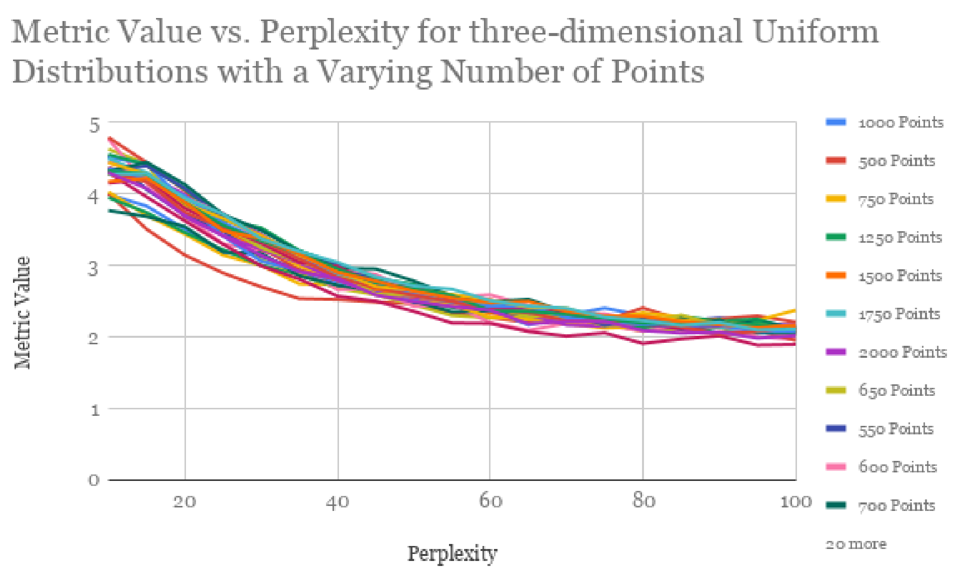


Figure 4: (Colored online) Plotting the value of our metric for various values of perplexities and number of points.

We give intuition for why this is true.

Note that the number of points captured by a $d-$ball of radius r in a d-dimensional uniform distribution is directly proportional to its volume, and is therefore directly proportional to r^d. Let's say the number of points in a uniform distribution inside a $d-$ball of radius r is c (which is the case if we set $r = \beta(x_i, c)$). Then, the number of points inside the same distribution inside a $d-$ball of radius $\left(\frac{2}{c}\right)^{\frac{1}{d}} r$ is $c \cdot \frac{\left(\left(\frac{2}{c}\right)^{\frac{1}{d}} r\right)^d}{r^d} = c \cdot \frac{2r^d}{cr^d} = 2$. This means that for uniformly distributed X,

$$\alpha\left(x_i, \left(\frac{2}{c}\right)^{\frac{1}{d}} \beta(x_i, c)\right) \approx 2.$$

Since M_{clust} is approximately 2 for all uniform distributions, the value of M_{clust} should be approximately independent of the dimension and size of any data set X, just like how how "clustered " a data set is shouldn't depend on either. This also means that the metric is very low for distributions that are close to uniform. However, for clustered distributions, since more points generally lie towards the center of a cluster, the value of this metric should be higher.

We use the value $c = 20$ for tests, as c should not be too small, or else the metric would be too small to distinguish between "clustered" and "not clustered" data sets, and it couldn't be too large or it would start looking at data points outside of clusters.

In order to empirically see if this metric gives similar values for similar data sets, we find M_{clust} values for t-SNE mappings of three-dimensional uniform data sets with different numbers of points for perplexities from 5 to 100, shown in Figure 4.

Since the graphs appear similar, this metric should be valid at evaluating how clustered the two-dimensional mappings given by t-SNE are.

When graphing the effect of perplexity on the metric for different data sets, we find that the metric lowers as perplexity increases, and changes in perplexity effect t-SNE's output much more for lower perplexities. Additionally, as the perplexity increases, the value returned by the metric appears to approach the metric value returned by the original data set. An example is shown in Figure 5.

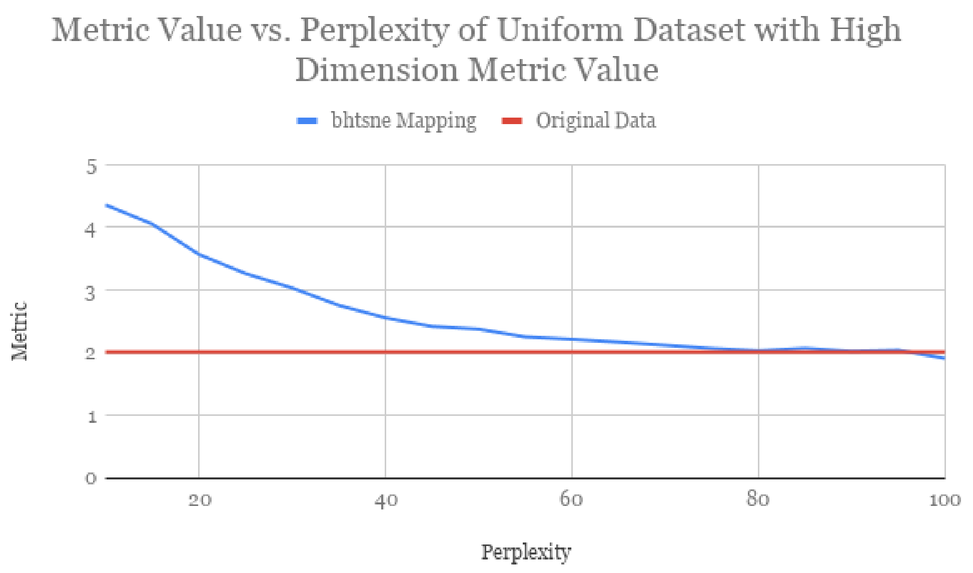


Figure 5: (Colored online) Plotting the value of our metric compared to that of a uniform data set.

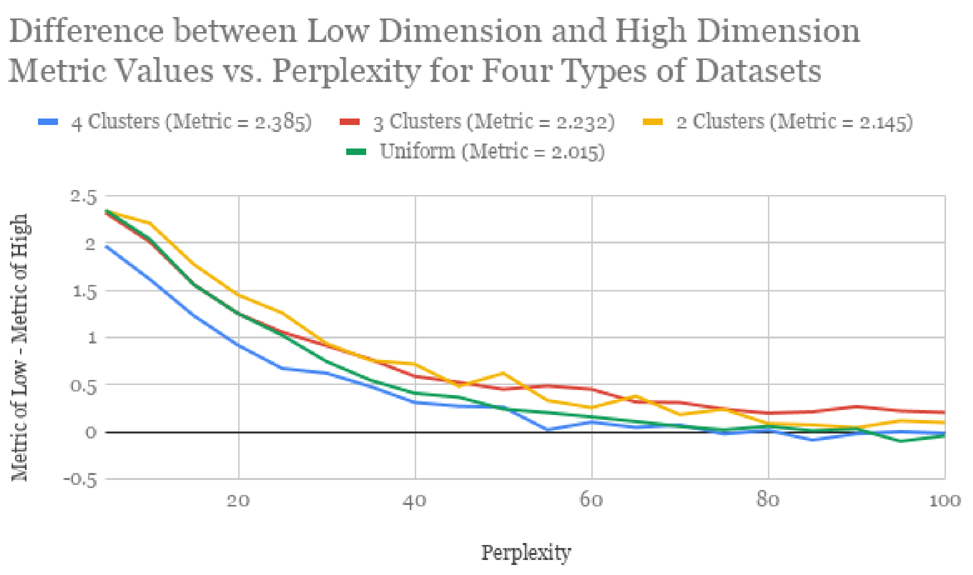


Figure 6: (Colored online) Plotting the value on our metric for variable types of data sets.

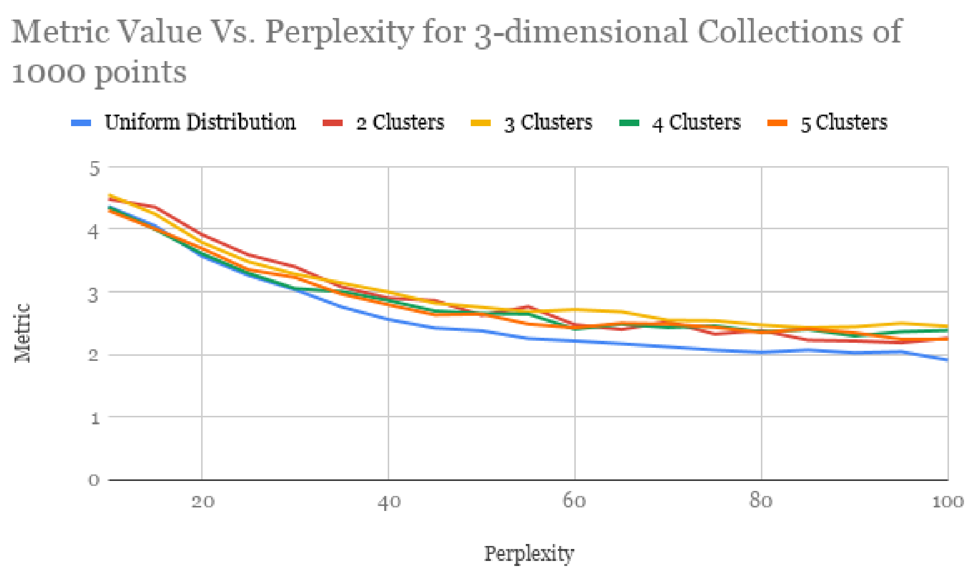


Figure 7: (Colored online) Metric value on clustered data sets.

Further evidence for this can be shown through graphs of the difference between the metric values of high-dimensional and low-dimensional data sets for different types of data sets, as evidenced by Figure 6.

In order to test how the number of clusters in a data set affects M_{clust}, we create a perplexity vs. M_{clust} graph for a 1000 point uniformly distributed data set and 1000 point data sets with 2, 3, 4, and 5 clusters. The M_{clust} values appear to be similar for distributions with at least one cluster, but look different from the values given by the uniform distribution. These results are displayed in Figure 7.

In order to get a better idea of how the perplexity behaves, we propose a general equation to approximate the perplexity and returned metric values for uniform distributions. We aim to find the best possible function $F(s) = e^{c+as} + b$ that models the data, where $F(s)$ is the value returned by the metric and s is the perplexity used to generate a t-SNE plot. Because it appears that the value of M_{clust} for t-SNE's output approaches the metric value given by the inputted data set as the perplexity increases, b is set to be equal to the input's M_{clust} value. Taking the natural log of both sides gives $\ln(F(s) - b) = as + c$. Linear regression can be used to find values for a and c for each data set. Then, we can create a plot of the number of data points in inputted uniform data sets and the values of a and c given by regression. Using linear regression on this plot gives that $c = 0.0002796x + 1.073$ and $a = 0.00000951x - 0.07147$, where x is the number of data points of the inputted distribution, are good models for the values.

To determine the exact behaviour of the metric on data sets and what metric values we might expect from mappings given by t-SNE, we want to find a method to theoretically evaluate the value of the metric for uniformly distributed data sets. We can approximate the expected value of M_{clust} for D-dimensional uniform distributions of n points inside the unit hypercube by making the assumption that a ball centered around a point will intersect no more than one of the faces of the hypercube (capital D is used to denote dimension in order to avoid confusion with dx terms in integrals). In this method it is assumed that any ball or hypercube mentioned is D-dimensional, and the value c is referring to the value used in the described metric.

Let the function $f(x,r)$ give the volume of a D-dimensional ball of radius r with a cap cut off of it from a distance x from its center. For example, in two dimensions,

$$f(x,r) = x\sqrt{r^2 - x^2} + \left(1 - \frac{\arccos\frac{x}{r}}{\pi}\right)(\pi r^2)$$

gives the area of a disk with some segment cut off with a chord of a distance x from the center of the disk.

Let the function $V_D(r)$ denote the volume of a D-dimensional ball with radius r. It is well known that

$$V_n(r) = \frac{\pi^{\frac{D}{2}} r^D}{\Gamma\left(\frac{D}{2}+1\right)}.$$

The function

$$g(r) = (1-2r)^D (V(r))^{c-1}(1-V(r))^{n-c-1} + \binom{D}{2} 2^{D-2} \int_0^r f(x,r)^{c-1}(1-f(x,r))^{n-c-2}(1-2x)^{D-1} dx$$

is proportional to the probability of the smallest ball containing the c closest points to a fixed center in this data set having radius r, as $c-1$ points need to lie in this ball and $n-c-1$ other points need to lie outside of it[2]. The integral represents the "frame" in which if the point that the ball is centered around lies within this frame, some of the ball around it is cut off from the unit hypercube, so the volume of the ball needs to be adjusted accordingly.

3 t-SNE Modification to Preserve Randomness

One of the largest failures of t-SNE is that when uniform random data sets are mapped to low dimensional data sets, their mappings look clustered and less random. This can become especially problematic when some areas of data sets are approximately uniform while other are clustered, because this could allow the misinterpretation of uniform regions of data as clustered or clustered regions of data as uniform.

The following modification to the t-SNE algorithm allows uniform distributions to look somewhat more unstructured, while the mapping of regions that are not uniform stays the same:

Definition 3.1. For a finite set X of d-dimensional datapoints $\{x_1, x_2, \cdots, x_n\}$, the function $\gamma(m,X,x_i)$ gives:

$$\gamma(m,X,x_i) := \left(\frac{m}{10}\right)^{\frac{1}{d}} \cdot \frac{d+1}{d} \cdot \frac{\sum_{x_k \in X} ||x_k - x_i||}{|X|}.$$

Consider the sets

$$S_{im} = \left\{x_j : x_j \in X, \gamma(m,X,x_i) > ||x_j - x_i|| > \gamma(m-1,X,x_i)\right\}$$

and sequences of sets

$$A_{i,j} = \begin{cases} S_{ij} \cup A_{i,j+1} & \frac{|X|}{10} - \frac{|X|}{80} < |S_{ij}| < \frac{|X|}{10} + \frac{|X|}{80} \\ \varnothing & \text{otherwise} \end{cases}.$$

To show that this sequence is well defined, we need to prove the following claim:

Claim. For a finite set X of d-dimensional data points $x_1, x_2 \cdots, x_n$, for every value of $1 \le i \le n$, there exists some value N_i such that for all $\varepsilon > N_i$, $A_{i,\varepsilon} = \varnothing$

[2]the other two points are the cth closest point, which lies on the boundary of the ball, and the ball's center

Proof. Note that since X is finite, we can define the value $M = \max\{||x_i - x_j|| : x_i, x_j \in X\}$. For each x_i, let the value $N_i = \left\lceil 10\left(M \cdot \frac{d}{d+1} \cdot \frac{|X|}{\sum_k ||x_k - x_i||}\right)\right\rceil + 1$. Note that $\gamma(N_i - 1, X, x_i) \geq M$. Consider some positive integer $\varepsilon > N_i$. This means that

$$S_{i(\varepsilon)} = \left\{x_j : x_j \in X, \gamma(\varepsilon, X, x_i) > ||x_j - x_i|| > \gamma(\varepsilon - 1, X, x_i)\right\}.$$

However, for x_j in this set, $||x_j - x_i|| > \gamma(\varepsilon - 1, X, x_i) > \gamma(N_i - 1, X, x_i) \geq M$, which contradicts $M = \max\{||x_i - x_j|| : x_i, x_j \in X\}$. This means that $S_{i\varepsilon} = \varnothing$, so $|S_{i\varepsilon}| = 0$ and $|S_{i\varepsilon}| < \frac{|X|}{10} - \frac{|X|}{80}$. Therefore, $A_{i,\varepsilon} = \varnothing$ for all $\varepsilon > N_i$. $\square$

In other words, if you try to compute $A_{i,1}$ and have to keep computing $A_{i,k}$ for progressively higher values of k due to the recursion $A_{i,j} = S_{ij} \cup A_{i,j+1}$, eventually you will find some value k such that $A_{i,k} = \varnothing$.

Let

$$T(i,j) = \begin{cases} 0.7 & x_i \in A_{j,1} \text{ and } x_j \in A_{i,1} \\ 1 & \text{otherwise} \end{cases}.$$

We now define

$$q_{ij} = \frac{(1 + ||x_i - x_j||^2)^{-T(i,j)}}{\sum_{k \neq l}(1 + ||x_i - x_j||^2)^{-T(i,j)}}.$$

We want to give some intuition as to why this modification will help at preserving randomness.

We can show that the number of points lying in the set $S_{im} = \left\{x_j : x_j \in X, \gamma(m, X, x_i) > ||x_j - x_i|| > \gamma(m-1, X, x_i)\right\}$ is approximately $\frac{|X|}{10}$ for small m.

Consider the random variable R representing the distance between a point selected randomly from a uniform distribution and another fixed point in the distribution. Let's assume that the uniform distribution takes place in some ball of radius k (this is clearly an incorrect assumption, but this should provide a good estimate of the behaviour of the distribution). Note that the surface area of a $D-$dimensional ball is proportional to r^{D-1}, where r is its radius, so since the selected point in R would lie on the surface of a D-dimensional ball, this probability distribution can be described by $P(R = r) = cr^{D-1}$, for some c. Then, we have that

$$\int_0^k cr^{D-1} dr = 1,$$

so $c = \frac{D}{k^D}$. This gives the probability distribution $P(R = r) = \frac{Dr^{D-1}}{k^d}$. The average of R over all of the points is expected to be

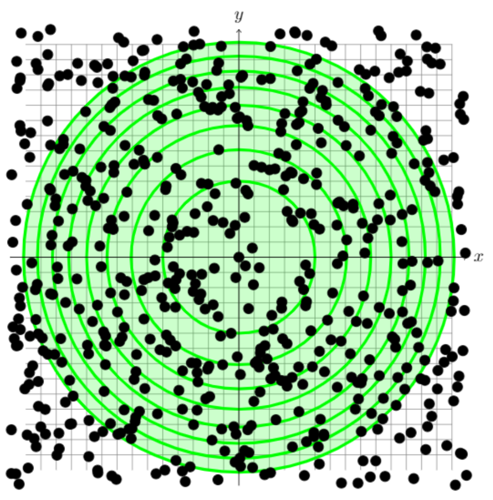
$$\int_0^k r \cdot P(R = r) dr = \int_0^k \frac{Dr^D}{k^D} dr = \frac{Dk}{D+1}.$$

Therefore, for a given point x_i and a randomly selected point x_j, the probability that $||x_i - x_j||$ lies between $\left(\frac{m-1}{10}\right) \cdot \frac{D+1}{D} \cdot \frac{\sum_k ||x_k - x_i||}{n}$ and $\left(\frac{m}{10}\right) \cdot \frac{D+1}{D} \cdot \frac{\sum_k ||x_k - x_i||}{n}$ is

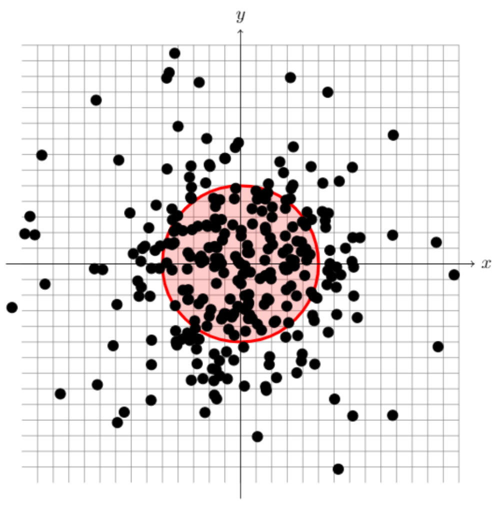
$$\int_{\left(\frac{m-1}{10}\right)^{\frac{1}{D}} \cdot \frac{D+1}{D} \cdot \frac{Dk}{D+1}}^{\left(\frac{m}{10}\right)^{\frac{1}{D}} \cdot \frac{D+1}{D} \cdot \frac{Dk}{D+1}} \frac{Dr^{D-1}}{k^D} dr = \frac{mk^D}{10k^D} - \frac{(m-1)k^D}{10k^D} = \frac{1}{10}.$$

By linearity of expectation, the expected number of points that lie a distance between $\left(\frac{m-1}{10}\right)^{\frac{1}{D}} \cdot \frac{D+1}{D} \cdot \frac{\sum_k ||x_k - x_i||}{n}$ and $\left(\frac{m}{10}\right)^{\frac{1}{D}} \cdot \frac{D+1}{D} \cdot \frac{\sum_k ||x_k - x_i||}{n}$ units away from a point x_i is approximately $\frac{|X|}{10}$.

Therefore, if the number of points in regions of this form is between $\frac{|X|}{10} - \frac{|X|}{80}$ and $\frac{|X|}{10} + \frac{|X|}{80}$, the distribution likely looks uniform.



(a) Uniform Data Set



(b) Clustered Data Set

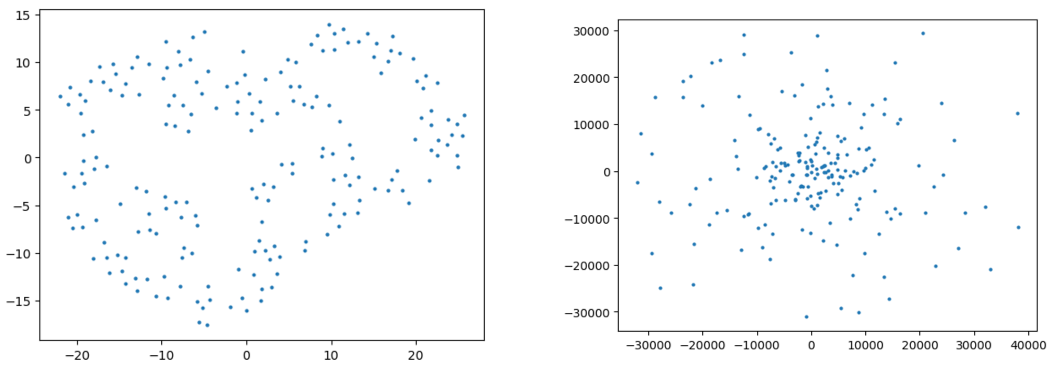
Figure 8: (Colored Online) Regions Tested for Uniformly Distributed Points around the Origin for Uniform and Clustered Data Sets.

Based on this claim, conceptually we are taking some point x_i, and then drawing a ball around this point that is expected to contain $\frac{|X|}{10}$ points if X is uniformly distributed. If this ball contains close to this number of points, we extend the radius of the ball so that it contains $\frac{2|X|}{10}$ points, and then keep doing this until eventually a region without the expected number of points is found. An example of this can be shown in Figure 8, in which the left figure shows regions in which each green region contains the expected number of points for each iteration of this process (note that the number of points in each region between two circles is the same), whereas in the clustered distribution in the right figure, the first circle contains too many points and the point finding process terminates. All of the points that are contained in regions with the expected number of points are considered uniform around x_i. For example, in the left plot of Figure 8, all points in the green region would be considered uniform. A pair of points x_i,x_j is considered uniform if x_i is uniform around x_j and x_j is uniform around x_i.

In order to address how to modify t-SNE so that randomness is more faithfully preserved, we need to find specific steps in the algorithm that cause the preservation of randomness to fail. A property of t-SNE is that mapping points that are close together in high dimensional data sets to points that are not close together in low dimensional data sets has a very high cost, whereas mapping points that are far apart in the high dimensional data set to points that are close together in the low dimensional data set has little effect on the cost. However, part of the appearance of randomness relies on points having an equal probability of being far apart or close to one another, so if we primarily preserve groups of points that are close to each other, the low dimensional data sets naturally look more clustered.

In order to change this, we need to increase the cost of mapping points that are far apart in high dimensional data sets to points that are close together in low dimensional data sets. In other words, we need to make the tails of q_{ij} even more "heavier".

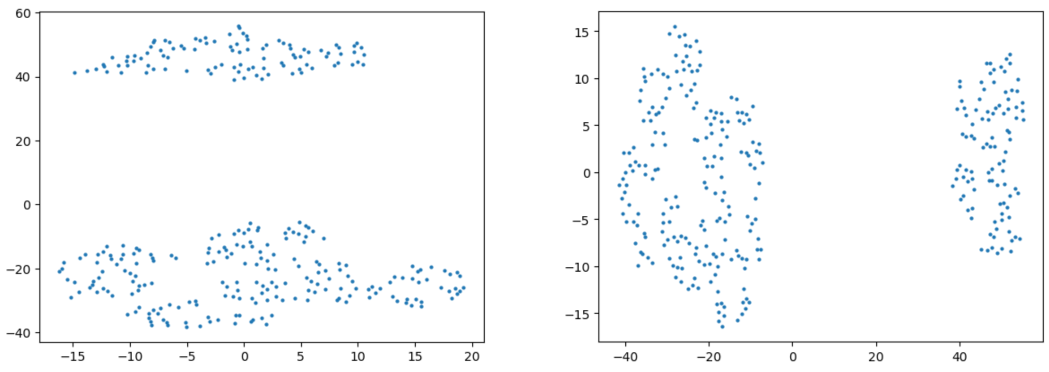
Consider the function $f(x) = \left(\frac{1}{1+x^2}\right)^{0.7}$. If x is close to 0 the value of $f(x)$ is approximately $\frac{1}{1+x^2}$, as $\frac{1}{1+x^2}$ and $f(x)$ would both be close to 1. However, for larger values of x, in which $\frac{1}{1+x^2} << 1$, then $f(x)$ is



(a) Original t-SNE

(b) Modified t-SNE

Figure 9: Comparison of our modification with original data set for uniformly distributed data.



(a) Original t-SNE

(b) Modified t-SNE

Figure 10: Comparison of our modification with original data set for clustered data.

significantly larger than $\frac{1}{1+x^2}$.

This means that by taking the function q_{ij} to the 0.7th power, we approximately preserve t-SNE's ability to map close points in X close to one another in Y. This is because, as explained above, q_{ij} should remain similar for values of $|x_i - x_j|$ close to 0 (it would be affected primarily by points that are far apart being weight more), but should be weighted more for larger values of $|x_i - x_j|$.

Therefore, because the function $(1+||x_i - x_j||^2)^{-T(i,j)}$ is equal to $(1+||x_i - x_j||^2)^{-0.7}$ if and only if points x_i and x_j are considered uniform, setting $q_{ij} = (1+||x_i - x_j||^2)^{-T(i,j)}$ allows uniform data sets to be mapped better while not changing how other data sets are mapped.

Results of this modification on uniform data and on clustered data are shown in Figure 9 and Figure 10, respectively. In Figure 9, the modified algorithm gives a mapping for a uniform data set that appears more uniform than the mapping given by unmodified t-SNE. In Figure 10, both modified and unmodified t-SNE give the same mapping for a clustered data set[3].

[3]The differences between the two mappings has to do with the set of points chosen randomly from an isotropic Gaussian distribution in the gradient descent method

4 Distinguishing Between a Single Normally-Distributed Cluster and Multiple Close Normally-Distributed Clusters

We have also looked at how to preserve information that is lost in t-SNE. We have looked at measuring if groups of points that are close to one another in a distribution are made up of two or more groups of points with similar characteristics. These groups of points in t-SNE will be interpreted as one cluster, making it difficult to interpret the data correctly once the data set is inputted into the algorithm.

It is impossible to determine if a large cluster is made up of multiple smaller clusters without restricting what the shape of a cluster may look like. Therefore, we make the assumption that clusters will be more dense closer to their center.

Consider the following metric used to determine whether some data set $X = \{x_1, x_2, \cdots, x_n\}$ is made up of multiple centrally distributed clusters.

Definition 4.1. We define the function $A(X, x_i)$ to be

$$A(X,x_i) := \frac{1}{|X|} \sum_{x_j \in X, j \neq i} ||x_j - x_i||.$$

A gives the average distance between the point x_i and all other data points in X.

Definition 4.2. We define the function $B(X, x_i)$ to be

$$B(X,x_i) := \frac{1}{20} \sum_{|x_j - x_i| \leq \alpha(X, x_i, 20)} ||x_j - x_i||.$$

B gives the average distance between x_i and its twenty closest points, using the definition of α in Section 2.

Definition 4.3. For a finite set $X = \{x_1, x_2, \cdots, x_n\}$ of d-dimensional data points, the metric $M_{multiclust}$ used to determine whether X consists of smaller clusters is defined as:

$$M_{multiclust}(X) := \frac{1}{|X|} \sum_{x_i \in X} \left(\left(\frac{20}{|X|} \right)^{\frac{1}{d}} \cdot \frac{A(X,x_i)}{B(X,x_i)} \right).$$

To show $M_{multiclust}$ is independent of the dimension and size of X, we want to show it gives the same value for any uniformly distributed X. According to results explained in Section 3, the value of $\frac{A(X,x_i)}{B(X,x_i)}$ is approximately

$$\frac{A(X,x_i)}{B(X,x_i)} \approx \frac{\frac{d}{d+1} k}{\frac{d}{d+1} \alpha(X, x_i, 20)} = \frac{k}{\alpha(X, x_i, 20)},$$

where k is the radius of some theoretical ball centered at x_i that contains all points in X. Since the expected number of points that lie in a uniformly distributed region is proportional to the region's volume and the volume of a d-dimensional ball with radius r is proportional to r^d, the fraction $\frac{k^d}{\alpha(X,x_i,20)^d}$ gives the approximate ratio of the number of points in X to the number of points in the ball of radius $\alpha(X, x_i, 20)$. This means $\left(\frac{k}{\alpha(X,x_i,20)} \right)^d \approx \frac{|X|}{20}$. Therefore,

$$\frac{1}{|X|} \sum_{x_i \in X} \left(\left(\frac{20}{|X|} \right)^{\frac{1}{d}} \cdot \frac{A(X,x_i)}{B(X,x_i)} \right) \approx \frac{1}{|X|} \sum_{x_i \in X} \left(\left(\frac{20}{|X|} \right)^{\frac{1}{d}} \cdot \left(\frac{|X|}{20} \right)^{\frac{1}{d}} \right) = 1$$

for all uniform distributions.

There is inherent structure in clusters made up of multiple centralized distributions which forces the value returned by $M_{multiclust}$ to be higher than for similar looking clusters that are not made up of multiple distributions.

For two data sets to look similar, it is likely that their average distance between data points are similar, meaning their values of $A(X, x_i)$ should be close to one another. As the distance between two or more normal distributions increases, the values for A increase, so single centralized distributions that look similar will also have larger values of A. Additionally, if you increase the spread of data in a normal distribution, $A(X, x_i)$ and $B(X, x_i)$ should remain proportional. However, the value of $B(X, x_i)$ for a cluster made up of multiple "sub-clusters" should cap at the value of $B(X, x_i)$ for each of its sub-clusters, as the twenty closest points in each sub-cluster do not change when two clusters are brought further apart. This means the metric will increase when the distance between two sub-clusters increases, but the metric of similar looking normal distributions would remain the same.

To demonstrate this, we can find A and B for pairs of similar looking data sets. For example, the ratio of the average value of A for data set (c) to the average value of A for data set (d) in Figure 12 was 1.010, whereas the ratios between their average values of B was 1.060 (although these numbers are both very close to 1, on the scale of thousands of data points a ratio of 1.060 indicates that B changes significantly more than A).

We can test $M_{multiclust}$ on clusters detected by the cluster detection algorithm db-SCAN [5]. We place some clusters extremely close to one another so db-SCAN interprets them as one cluster, and then use $M_{multiclust}$ to test if it could distinguish between these clusters and clusters that are made up of one centrally distributed distribution.

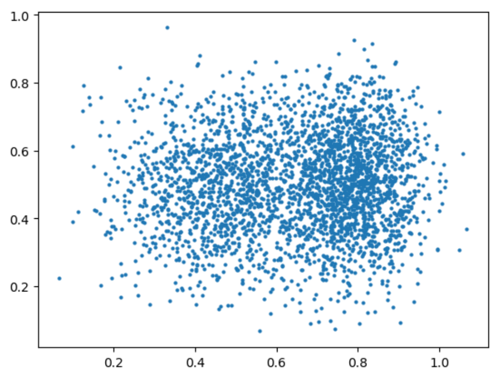
Based on the data sets tested, it appears that clusters that did not contain any "sub-clusters" tend to give $M_{multiclust}$ values of 1.23 to 1.26, whereas clusters that are actually a result of two or more clusters being close together tend to give metric values of 1.28 and above. It also appears that farther distances between sub-clusters result in higher $M_{multiclust}$ values. Results are shown in Figure 11.

This metric is effective at distinguishing whether a cluster is made up of multiple "sub-clusters" when the human eye cannot, as demonstrated in Figure 12.

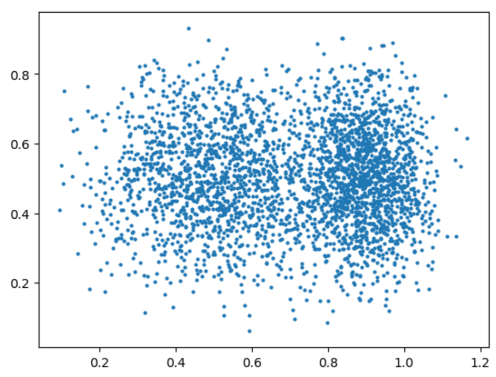
5 Further Work

In the future, one could propose different metrics to quantify how well t-SNE preserves characteristics other than just how clustered a data set is. Metric value vs. perplexity graphs for different distributions could reveal more about the perplexity's affect on specific distribution types other than just uniform distributions.

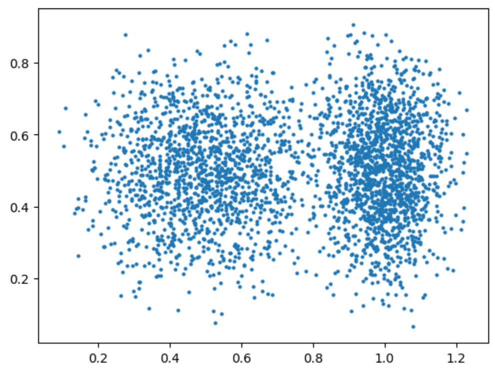
Additionally, experimenting with different cost function modifications could further improve how well t-SNE preserves uniformity or reduce the modified algorithm's run time. One could also try to modify the algorithm to become better at mapping distributions of centered clusters by using the $M_{multiclust}$ metric described in Section 4.



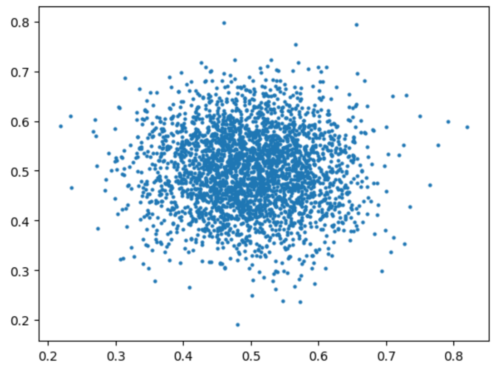
(a) Separation = 0.3, metric = 1.332



(b) Separation = 0.4, metric = 1.463

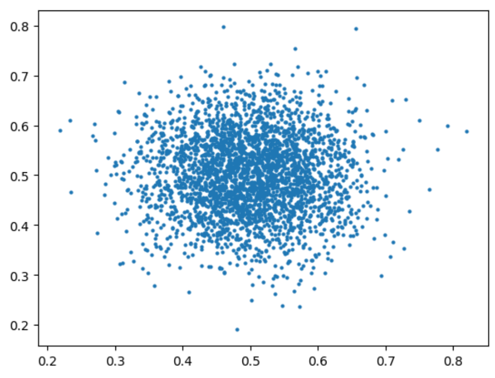


(c) Separation = 0.5, metric = 1.654

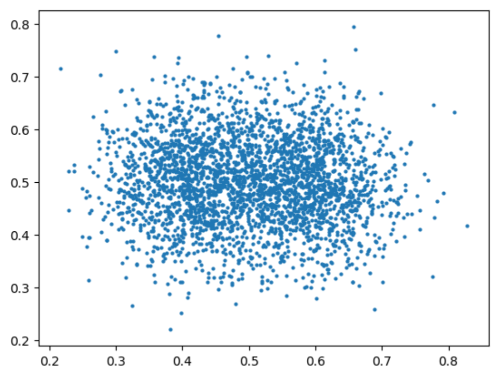


(d) Single cluster, metric = 1.245

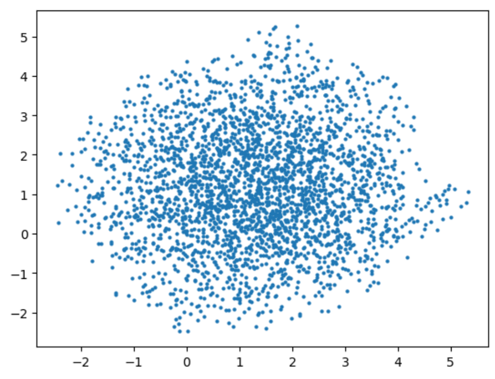
Figure 11: The metric $M_{multiclust}$ is able to differentiate when a data set is actually clustered.



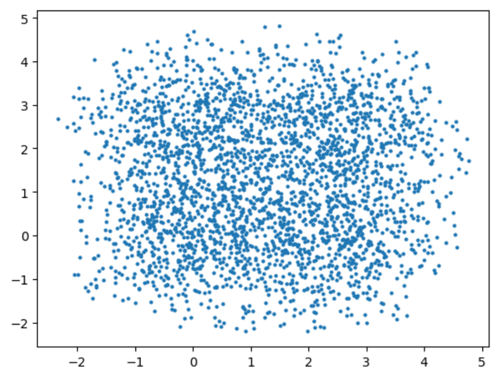
(a) Single centrally distributed, metric = 1.245



(b) Slightly separated, metric = 1.294



(c) Single centrally distributed, metric = 1.257



(d) Four separated distributions, metric = 1.319

Figure 12: The metric $M_{multiclust}$ differentiates clustered data sets even when the separation is not easy to see with the naked eye.

6 Acknowledgements

We would like to thank the MIT PRIMES-USA program for providing us with the opportunity to research, and would like to especially thank Dr. Tanya Khovanova for her feedback and guidance on planning out and writing research papers.

References

[1] L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9 (Nov): 2579-2605, 2008.

[2] L.J.P. van der Maaten. Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research 15 (Oct): 3221-3245, 2014.

[3] J. Barnes and P. Hut. A hierarchical O(N log N) force-calculation algorithm. Nature, 324 (4):446-449, 1986.

[4] P.N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, pages 311-321, 1993.

[5] Hahsler M, Piekenbrock M, Doran D (2019). “dbscan: Fast Density-Based Clustering with R.” Journal of Statistical Software, 91(1), 1-30. doi: 10.18637/jss.v091.i01.