CSCI 4968: Group U4 Project Progress Report

# t-SNE Data Visualization & Interpretation

Aksshat Gupta, Miles Harrison, Yuxiao Li

## 1 INTRODUCTION

Visualization of high-dimensional data is a powerful tool for data analysis and has been widely applied in many different domains. It helps extract meaningful and relevant information from these high-dimensional datasets for easier interpretation by human researchers. When visualizing these high-dimensional data sets, we usually convert them into 2-D or 3-D data that can be displayed clearly. Linear dimensionality reduction methods such as Principal Components Analysis (PCA), Linear Discriminant Analysis (LDA) and Classical Multidimensional Scaling (MDS) are usually poor in converting complex high-dimensional datasets. Meanwhile, most of the nonlinear dimensionality reduction techniques such as Sammon mapping, Isomap, and Locally Linear Embedding (LLE) fail to capture both the local and global structure of the data in a single map. This project progress report discusses a method named t-SNE which can be used for high-dimensional data visualization and clustering. It is based on the paper "Visualizing Data using t-SNE" authored by Laurens van der Maaten and Geoffrey Hinton and published in November of 2008 to the Journal of Machine Learning Research. [1]

### 1.1. MOTIVATION & INTUITION OF t-SNE

t-SNE (t-Distributed Stochastic Neighbor Embedding) is a nonlinear technique for high-dimensional data visualization in 2-D or 3-D and dimensionality reduction. It is a variation of SNE first proposed by Maaten and Hinton in 2008. [1] Though SNE provides powerful visualization of these datasets, it has a few drawbacks: 1) its cost function is difficult to optimize; 2) it has the “crowding” problem. t-SNE alleviates both of these problems by using a different cost function and a Student-t distribution. It retains both the global and local structure of the data in the low-dimensional map, especially the local structure as similar datapoints tend to cluster in the map. It also provides better visualization, especially in explicitly displaying the relationship among different clusters. However, it has a quadratic complexity and has difficulty choosing the ideal hyperparameters.

## 2 t-SNE

Knowing the advantages of t-SNE, we are now exploring how the algorithm works. Generally, it computes the similarities among the datapoints in both high dimension and low dimension, and then uses Kullback-Leibler divergence to optimize these similarities. A simple version of t-SNE is shown in Algorithm 1. [1]

## 2.1 t-SNE GENERAL WORKFLOW

*Step 1):*

t-SNE starts by computing the similarity among the high-dimensional datapoints.

It converts the high-dimension Euclidean distances between datapoints into conditional probabilities representing similarities. For each datapoint $x_i$, we compute the probability density of all other datapoints $x_j$ under a Gaussian distribution centered at $x_i$. Then we compute the conditional probability $p_{j|i}$ that $x_i$ would pick $x_j$ as its neighbor in proportion to their probability density. The conditional probability $p_{j|i}$ is given by

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

where $\sigma_i$ is the variance of the Gaussian that is centered on $x_i$. Note that we are computing the pairwise similarities among datapoints, we set $p_{i|i} = 0$. Thus, if datapoints $x_i$ and $x_j$ are close together, they will have a higher value of $p_{j|i}$ and if they are far apart, $p_{j|i}$ will be much smaller.

Let P denote the computed joint probability distribution in the high-dimensional space. We can manipulate the Gaussian distribution using perplexity. Perplexity can be considered as a smooth measure of the effective number of neighbors, i.e., the expected number of neighbors for each datapoint $x_i$. It is defined as

$$Perp(P_i) = 2^{H(P_i)}$$

where $P_i$ is the probability distribution induced by the variance $\sigma_i$ of the Gaussian centered over the datapoint $x_i$, and $H(P_i)$ is the Shannon entropy of $P_i$ measured in bits

$$H(P_i) = -\sum_j p_{j|i} log_2 p_{j|i}$$

Typical values of $Perp(P_i)$ are between 5 and 50. After choosing a fixed perplexity, we find a value of $\sigma_i$ which produces a $P_i$ with this fixed perplexity using binary search.

*Step 2):*

The similarity among high-dimensional datapoints is computed with a Gaussian distribution in the first step. Similarly, we now compute the similarity among the low-dimensional datapoints. Different from the SNE method which uses Gaussian distribution in low-dimensional datapoints affinities as well, t-SNE utilizes a Student t-distribution with one degree of freedom to convert distances into probabilities. Student t-distribution, also known as Cauchy distribution, is a probability distribution with heavier tails than a Gaussian distribution and thus better eliminates the unwanted attractive forces between map points that represent dissimilar datapoints. In the map representation of the joint probabilities, the representation of far-away points are hardly affected by changes in map scales. Meanwhile, it is faster to evaluate the density of a point under a Student t-distribution than under a Gaussian distribution as it does not contain an exponential. Let Q denote the computed joint probability distribution in the low-dimensional space. The joint probabilities $q_{ij}$ is defined as

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

**Algorithm 1:** Simple version of t-SNE

**Data:** $X = \{x_1, x_2, ..., x_n\}$,
cost function parameter: perplexity $Perp$,
optimization parameters: iterations $T$, learning rate $\eta$, momentum $\alpha(T)$.
**Result:** $\Upsilon^{(T)} = y_1, y_2, ..., y_n$.
Use binary search to find var $\sigma_i$ for each datapoint $x_i$ which produces $P_i$ with the given fixed perplexity $Perp$.
Compute pairwise affinities $p_{j|i}$ with $\sigma_i$
set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
sample initial solution $\Upsilon^{(0)} = y_1, y_2, ..., y_n$ from $(0, 10^{-4})I$
**for** $t = 1$ *to* $T$ **do**
  compute low-dimensional affinities $q_{ij}$
  compute gradient $\frac{C}{\Upsilon}$
  set $\Upsilon^{(t)} = \Upsilon^{(t-1)} + \eta \frac{\partial C}{\partial \Upsilon} + \alpha(t)(\Upsilon^{(t-1)} - \Upsilon^{(t-2)})$
**end**

*Step 3):*

After the similarities have been computed, we now minimize their differences using Kullback-Leibler divergences. Kullback-Leibler divergences measure the expected log difference between the probabilities of data in the two distributions. SNE minimizes the sum of Kull-Leibler divergences over all datapoints using a gradient descent method, which is difficult to optimize. Instead, t-SNE minimizes single Kullback-Leibler divergence between P and Q. The cost function is given by,

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \, log \frac{p_{ij}}{q_{ij}}$$

and we set $p_{ii} = q_{ii} = 0$. This is referred to as the symmetric SNE as $\forall i, j,\ p_{ij} = pji$ and $q_{ij} = q_{ji}$. Thus, we set

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

This ensures that each datapoint contributes significantly to the cost function.

The cost function is optimized using the gradient descent method, which will be illustrated below.

## 2.2 OPTIMIZATION OBJECTIVE FUNCTION

t-SNE optimizes the cost function using the gradient descent method. The gradient of the cost function is given by,

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

This ensures that each datapoint contributes significantly to the cost function. The gradient can also be given by:

$$\frac{\delta C}{\delta y_i} = 4 \sum_{i=1}^{N} \left( F_i^{attr} - F_i^{rep} \right)$$

, where

$$F_i^{attr} = \sum_{j \neq i}^{N} p_{ij}\, q_{ij} Z \left( y_i - y_j \right)$$

$$F_i^{rep} = \sum_{j \neq i}^{N} q_{ij}^2 Z \left( y_i - y_j \right)$$

$$Z = \sum_{k=1}^{N} \sum_{l \neq k}^{N} \left( 1 + \left|\left| y_k - y_l \right|\right| 2 \right)^{-1}$$

In this equation, the gradient descent is considered as a N-body simulation, where each datapoint exerts an attractive force and a repulsive force on all other datapoints. [2]

## 2.3 IMPORTANT t-SNE HYPERPARAMATERS

- Perplexity $Perp$:

Perplexity indicates the expected number of neighbors of each datapoint $x_i$ and typical values are between 5 and 50. In most implementations, it defaults to 30. In practice, we may need to analyze multiple t-SNE plots using different perplexities.

- Learning Rate η:

η determines the step size in gradient descent. Common values are between 10 and 1000.

- Exaggeration:

Exaggeration increases the attractive force among points and allows them to move more freely. This helps the datapoints separate in the high-dimensional space and form clear clusters.

## 2.4 t-SNE COMPLEXITY, PROS & CONS

The time and memory complexity for t-SNE are both quadratic.

The main advantages of t-SNE are better local and global structure preservation and map visualization. Compared with SNE, t-SNE has cost function which is easier to optimize and addresses the 'crowding' problem. The use of a Student t-distribution enables t-SNE to model dissimilar datapoints by means of large pairwise distances and model similar datapoints by means of small pairwise distances.

However, t-SNE suffers from its quadratic complexity which makes it infeasible to apply standard t-SNE to large datasets directly. It is also sensitive to the curse of intrinsic dimensionality of the data and may converge to a local optimum instead of a global optimum. The optimized hyperparameters for each dataset may also be difficult to determine.

### 2.5 COMPARISON WITH OTHER MODELS

Linear dimensionality reduction methods such as PCA are usually not good at modeling curved manifolds. Instead, they focus more on preserving the global structure than the local structure. Sammon mapping tries to solve this problem while still having major weaknesses in its cost function. Isomap is susceptible to 'short-circuiting' and mainly focuses on modeling large geodesic distances rather than smaller ones. LLE may collapse all datapoints into one single point and is not capable of visualizing data with more than two widely separated submanifolds. In the experiment section below, we will compare the resulting visualization using different methods with the results of t-SNE on the same datasets.

## 3 INTERPRETING DATA VISUALIZATIONS USING t-SNE

Hyperparameters have significant impacts on the resulting visualization. Different hyperparameters may lead to very distinct shapes of visualization. The plots shown below in this section are based on "How to Use t-SNE Effectively" by Wattenberg, et al. [3] The plots below are generated using different perplexities. As we can see that when perplexity is 2, the local structure is well retained. Normal values of perplexity are between 5 and 50 and should be smaller than the number of points.

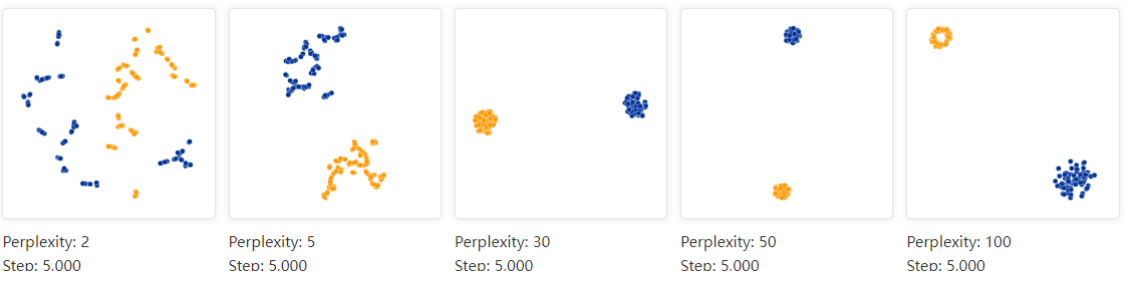



Figure 1 Same dataset using different perplexities

We usually choose the number of iterations which yields a stable result, which may not be fixed for different datasets. Multiple runs on most simple datasets give the same global shape while different results may be yield for certain datasets on different runs.

The sizes of clusters and the distances among them may not have any meaning. t-SNE expands the dense clusters and contracts the sparse clusters, thus the actual distances may be distorted. t-SNE visualizations of random noises may seem to contain some patterns.

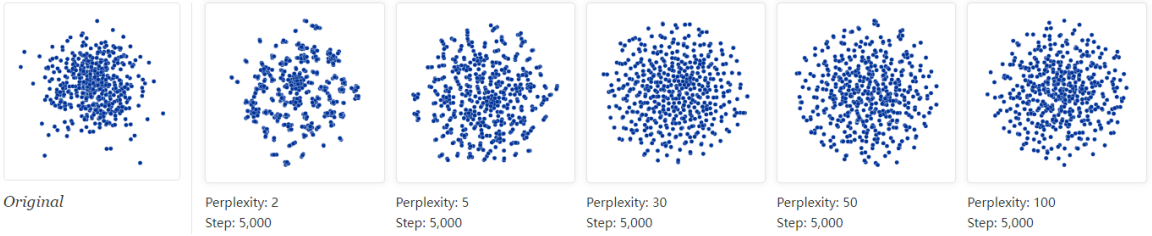
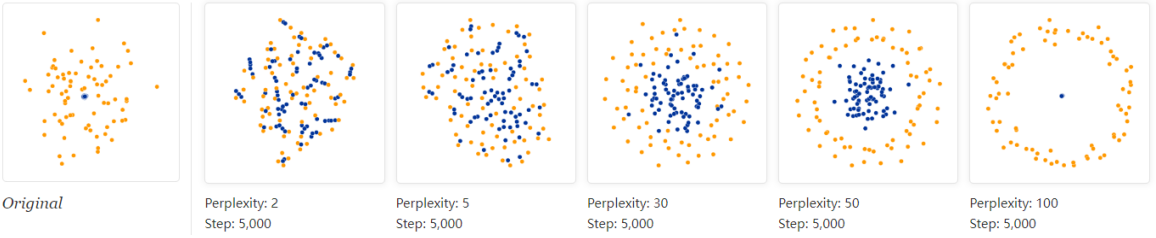



Figure 2 t-SNE visualization of random noises

The visualizations may also contain some shapes. To extract useful topological information, multiple plots may be needed.




## 4 EXPERIMENT

In this section, we will verify the performance of t-SNE and compare the results with visualization of other dimensionality-reduction methods. We implemented the experiment using the Fashion-MNIST dataset. In our future work, we will also perform experiments on the Iris dataset and the MNIST dataset. For each dataset, we will first perform self-implemented t-SNE and explore the optimized hyperparameters for the dataset. Then we will perform PCA, Sammong Mapping, Isomap and LLE and compare the visualization using similar hyperparameters.

Note that t-SNE has a quadratic complexity in the number of datapoints, it is infeasible to apply the standard version of t-SNE to datasets which are too large. Thus, we sampled random subsets with size of 10,000 points from datasets which are too large and performed experiment using these subsets.

## 4.1 FASHION-MNIST

1) t-SNE with perplexity 50:

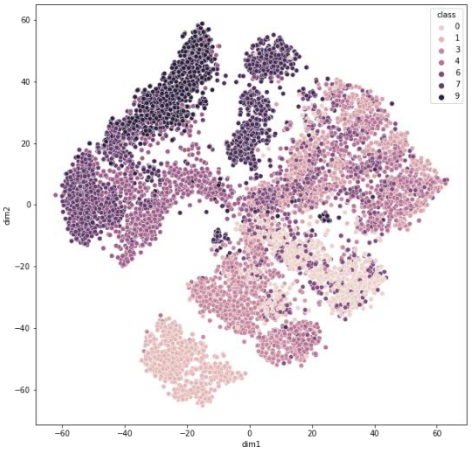



Figure 5.a FASHION-MNIST: t-SNE, perp=50

2) t-SNE with perplexity 10:

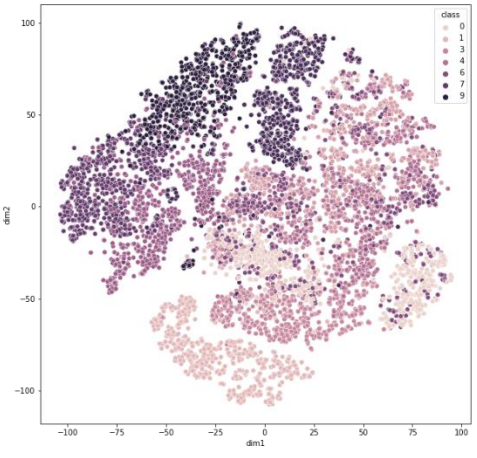



Figure 4.b FASHION-MNIST: t-SNE, perp=10

## 4.2 COMPARISON WITH OTHER MODELS:

1) PCA:

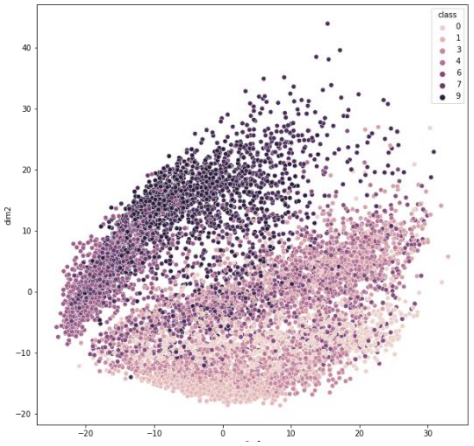



Figure 4.a FASHION-MNIST: PCA

2) Isomap:

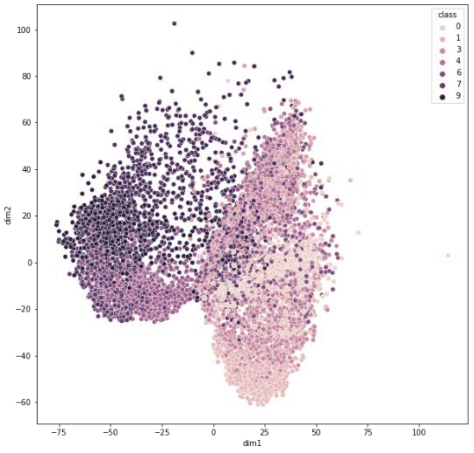



Figure 4.b FASHION-MNIST: Isomap

3) LLE:

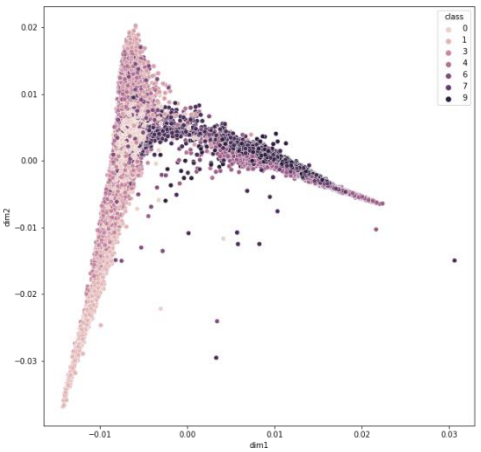



Figure 4.c FASHION-MNIST: LLE

## 5 PROBLEM SET

### 5.1 PROBLEM SET DESCRIPTION

In the problem set, you will need to implement a simple t-SNE and perform visualization using the MNIST dataset. The results should be compared with other methods such as PCA and SNE.

1) Data Preparation:

Load the MNIST dataset of handwritten digits and preprocess the data by normalizing the pixel values and removing any rows with missing values. Flatten the images and reduce the dimensionality using PCA for faster results (example: n=50).

2) t-SNE Implementation:

Implement the t-SNE algorithm from scratch in Python using only NumPy and matplotlib/seaborn. Start with a perplexity of 30 and a learning rate of 200 for the optimization. Use the algorithm provided above. The cost function and its gradient are provided here for reference.

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \, log \frac{p_{ij}}{q_{ij}}$$

$$\frac{\delta C}{\delta y_i} = 4 \sum_{i=1}^{N} \left( F_i^{attr} - F_i^{rep} \right)$$

3) Hyperparameter Tuning:

Experiment with different perplexity values (e.g., 5, 10, 50) and learning rates (e.g., 10, 100, 500) to see how they affect the quality of the t-SNE visualization. Visualize the results using matplotlib and compare the different parameter settings.

4) Visualization:

Use the t-SNE algorithm with the best hyperparameters to visualize the MNIST dataset in two dimensions. Plot the resulting embeddings and color-code the points based on their true labels. Interpret the resulting visualization.

5) Evaluation:

Calculate the trustworthiness and continuity of the t-SNE visualization using scikit-learn. Use trustworthiness_score and continuity_score from sklearn.metrics. Compare the results to those of SNE, PCA.

### 5.2 PROBLEM SET PEDAGOGICAL VALUE

1) Understanding the importance of dimensionality reduction:

t-SNE can help students understand the need for reducing the high-dimensional data to a lower-dimensional space to perform efficient analysis and visualization.

2) Exploring hyperparameter tuning:

t-SNE has several hyperparameters that affect the resulting visualization, such as perplexity and learning rate. Students can explore the effect of changing these hyperparameters and learn about the importance of hyperparameter tuning.

3) Visualizing high-dimensional data:

t-SNE can be used to visualize datasets in lower-dimensional space, which can help students understand the relationships, similarities, and differences between different clusters.

4) Understanding the limitations of visualization:

While t-SNE can provide useful visualizations, it's important to understand that they may not always reveal all the underlying patterns or relationships in the data. Students can learn about the limitations of visualization and explore other analysis techniques that can complement t-SNE.

## 6 CONCLUSION

The main result of “Visualizing Data using t-SNE” is a comparison of t-SNE to other existing methods for visualizing data in high dimensions. t-SNE has significant advantages over other techniques, but also various weaknesses. Some positives are that t-SNE retains the local structure of the data while also revealing global structure. Because of this, t-SNE can outperform existing state of the art techniques for visualizing some real-world data sets.

t-SNE also has a few main weaknesses. Although it is good for visualization, it is unclear how well it performs at reducing to dimensionality $d$ when $d > 3$. It is also sensitive to the intrinsic dimensionality of the data and may not work well on data sets with high intrinsic dimensionality. Unlike many other state of the art dimensionality reduction techniques, t-SNE’s cost function is not convex. So, t-SNE relies on good choices for hyper-parameters which can vary depending on the dataset. Additionally, t-SNE is quadratic in computational and memory complexity which makes it difficult to apply to large datasets. Performing a

random walk along the dataset is one solution proposed by “Visualizing Data using t-SNE” to speed up run times.

In our experiments, we found that t-SNE works well when handled correctly. Due to the size of the data sets we worked with, we found it useful to sample 10% of the datapoints for t-SNE to speed up calculations and still get meaningful results. Additionally, different values for hyperparameters were tested and a perplexity of 50 seemed to work well for the Fashion-MNIST data set. As shown in the plots in section 4.1, t-SNE shows more separation between each class of data than the other techniques. PCA and Isomap also show some separation, but have less distinction between the classes. LLE shows the least distinction between classes and is not very useful for visual interpretation.

## REFERENCES

[1] Laurens van der Maaten; Geoffrey Hinton: Visualizing Data using t-SNE. *Journal of Machine Learning Research 2008*, 2579-2605

[2] N. Pezzotti, B. P. F. Lelieveldt, L. v. d. Maaten, T. Höllt, E. Eisemann and A. Vilanova, "Approximated and User Steerable tSNE for Progressive Visual Analytics," in *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 7, pp. 1739-1752, 1 July 2017, doi: 10.1109/TVCG.2016.2570755.

[3] Wattenberg, et al., "How to Use t-SNE Effectively", *Distill*, 2016. http://doi.org/10.23915/distill.00002