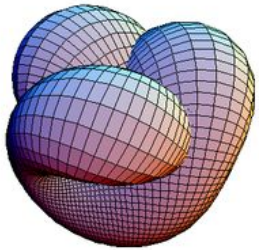
An application of t-SNE Dimension Reduction and Visualization

By: Zak Wilson

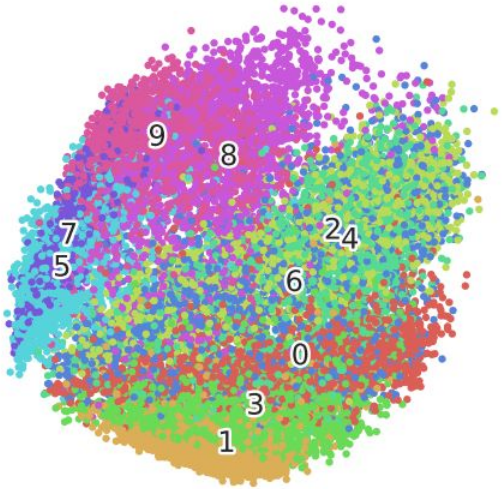
t-SNE



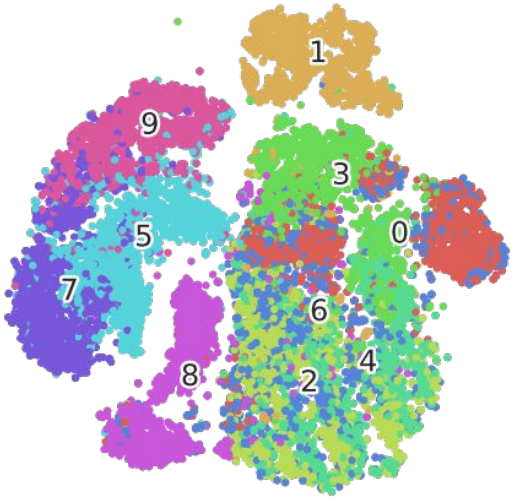
- “T-distributed stochastic neighbor embedding”
- Developed in 2008 by Laurens van der Maaten and Geoffrey Hinton
- What is it?
  - An unsupervised **non-linear** dimension reduction algorithm used to visualize high dimensional data sets.
- Goal: Take high dimensional data and map it into a low dimensional space (typically 2-D) while preserving the global and local structure from the high dimensional space as best as possible.
- Typical linear projections such as PCA are not as accurate for data visualization since PCA cannot always capture non-linear details in high dimensions.

PCA vs t-SNE Comparison

PCA

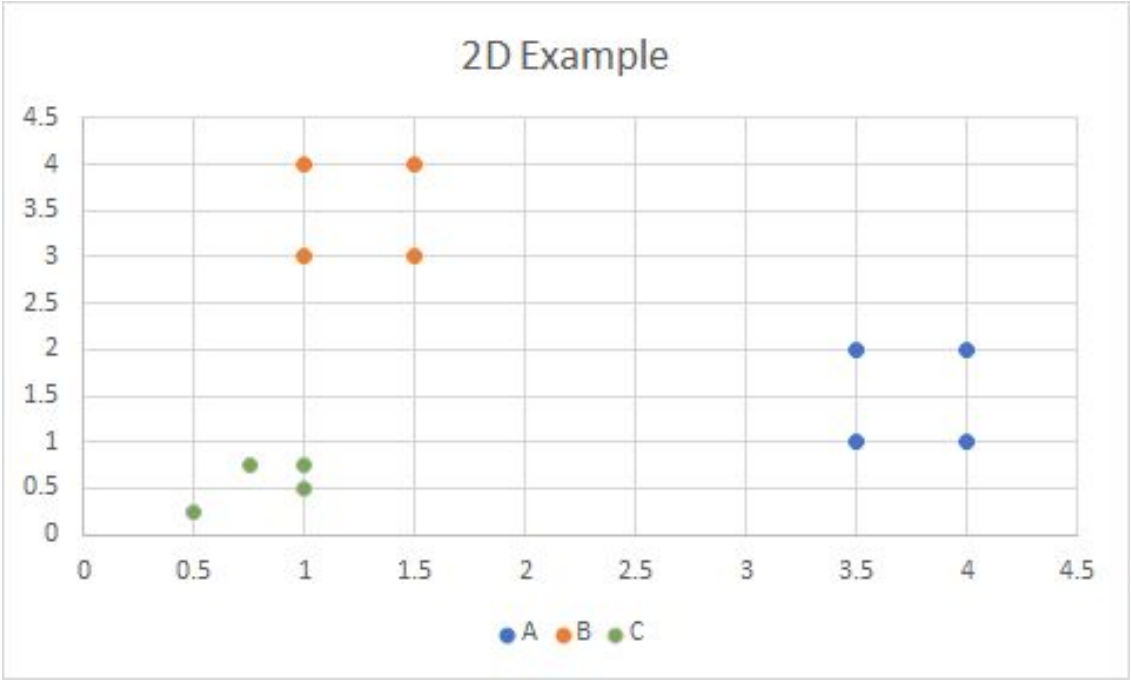


t-SNE



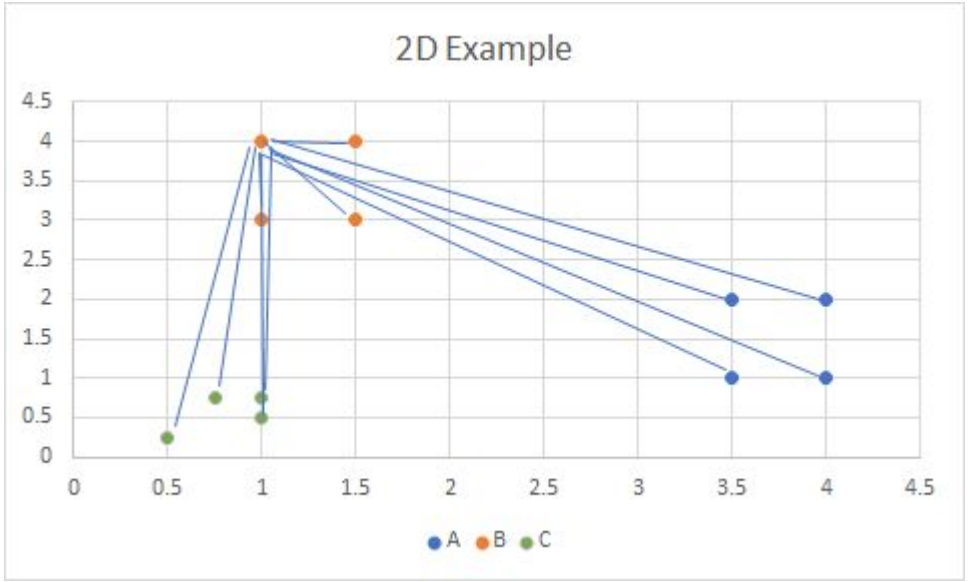
Source: https://www.datacamp.com/community/tutorials/introduction-t-sne

t-SNE: How?



t-SNE: How?

- Take pairwise Euclidean distance, between all points.



t-SNE: How?

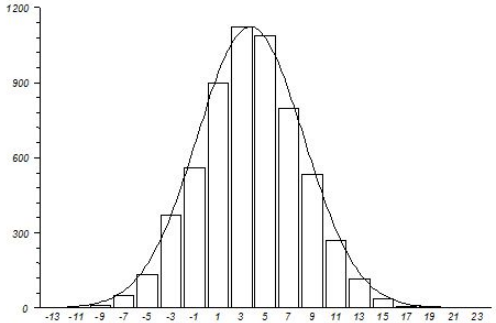
- Calculate probabilities (p_{ij}) that are proportional to the similarity of the points (say x_i and x_j) using a Gaussian function centered at x_i in the high dimensional space.

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}, \qquad p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

- Note: σ_i (called bandwidth in t-SNE) is selected based on a hyperparameter called “perplexity” which is set by the user before performing tSNE. More on this later.
- Example using Excel.

What is “Similarity”?

- Since we are creating a probability distribution about a point (x_i), we are quantifying this “similarity” as the probability that our main point(x_i) will pick the other pairwise point (x_j) as its “neighbor”.
- Result: The closer the points are in distance, the higher the probability of that point selecting the other point as it’s neighbor.
- Note: We set the similarity of a point with itself to 0, since we are only interested in pairwise distances. This gives a diagonal of 0s in the final similarity matrix.

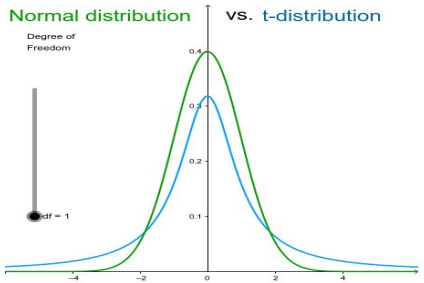


t-SNE: How?

- Goal: Learn a low dimensional (typically 2D) map that reflects the “similarities” in the high dimensional space.
- How: In the lower dimensional space, calculate probabilities (q_{ij}) that are proportional to the similarity of the points (say y_i and y_j) using a t - Distribution with one degree of freedom.

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|\mathbf{y}_i - \mathbf{y}_k\|^2)^{-1}}$$

Why the t-distribution?



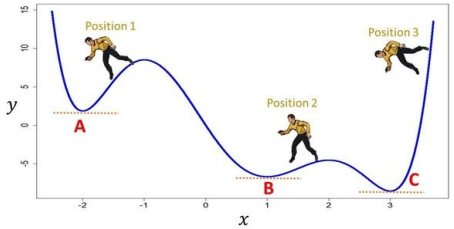
- Why t-distribution in lower dimensional space**?**
- Due to the curse of dimensionality, when bringing high dimensional data to a lower dimension space, there can be a crowding problem which makes it virtually impossible to really see the differences in similarity/distance between points.
- By using the t-distribution with 1 df we give higher probability density to far away points (due to heavy tail), which allows for it to be mapped farther apart in the lower dimensional map. This effectively “spreads” the data out.
- Without “t” we wouldn’t be able to see!

t-SNE, How?

- Now, we have a Gaussian probability distribution "P_i" about a point in the high dimensional space.
- Now, we have a Student's-t probability distribution "Q_i" about the same point in the lower dimensional map.
- Optimize the mapping by minimizing Kullback - Leibler Divergence between P_i and Q_i using Gradient Descent. The goal is to mimic the high dimensional probability distribution, for the lower dimension map.

$$\mathrm{KL}\left(P \parallel Q\right) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Gradient Descent



- What is it? A common optimization algorithm used to minimize a cost function (K-L Divergence in our case), iteratively, by moving in the direction of the steepest decrease by the negative of the gradient.
  - Maths for optimizing K - L Divergence using Gradient Descent:

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

- Can adjust these movements or “step sizes” with t-SNE as well to get desired results. This is generally referred to as the “learning rate” in machine learning algorithms.

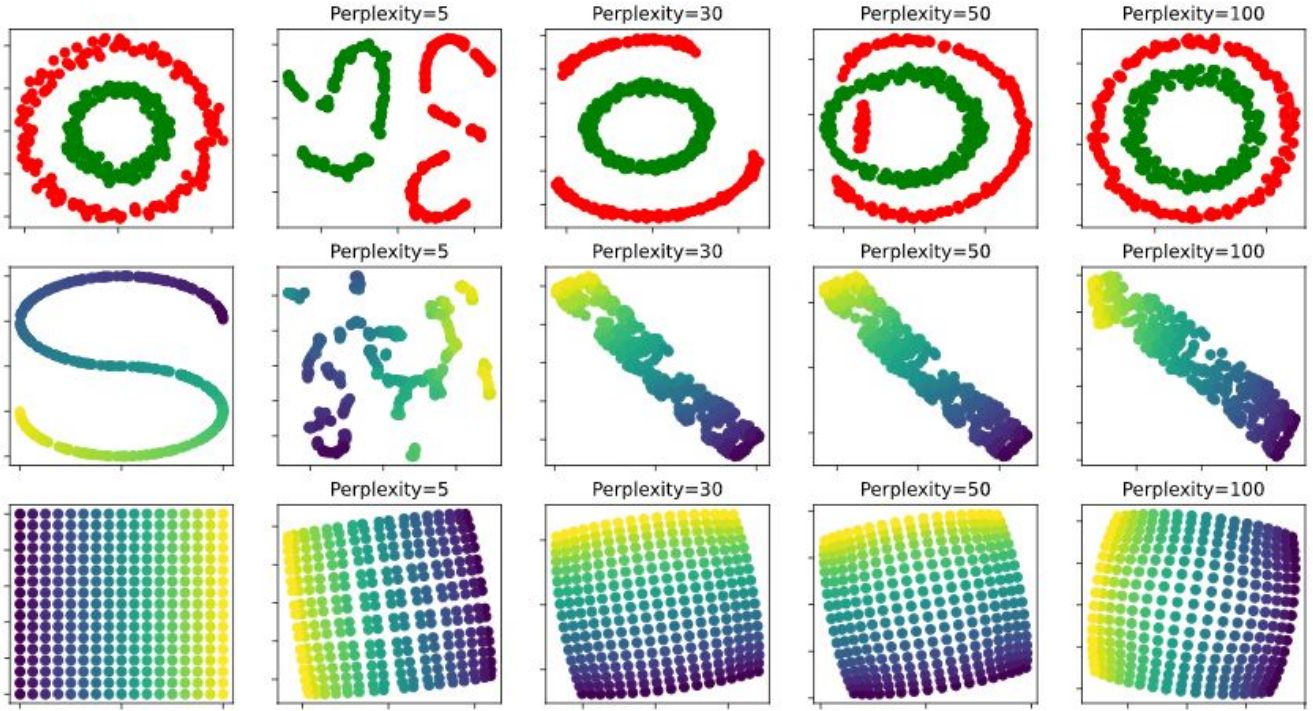
Perplexity

- Perplexity is a hyperparameter which is set by the user before performing t-SNE.
- SNE performs a binary search for the value of bandwidth (σ_i) that produces a probability distribution, P_i, with the user-provided perplexity value.

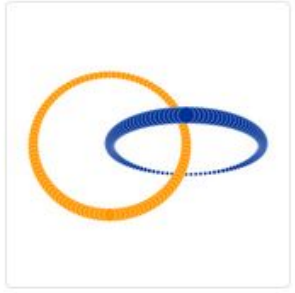
$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}. \qquad Perp(P_i) = 2^{H(P_i)},$$

- According to Van Der Maaten and Hinton, perplexity can be interpreted as a measure of the “effective number of neighbors” of each point.
- Larger perplexity - mimics global structure.
- Smaller perplexity - mimics local structure.

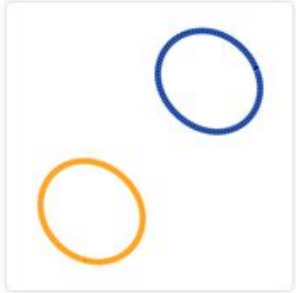
Perplexity Examples



Source Code: https://scikit-learn.org/stable/auto_examples/manifold/plot_t_sne_perplexity.html#sphx-glr-auto-examples-manifold-plot-t-sne-perplexity-py



Original



Perplexity: 2
Step: 5,000

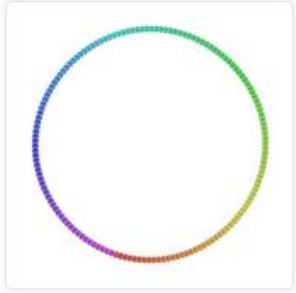
Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000

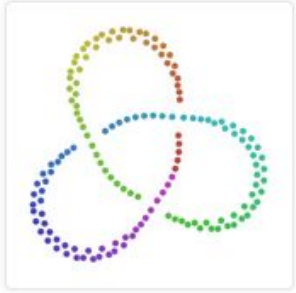
Perplexity: 100
Step: 5,000

Original

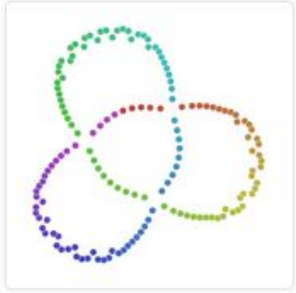


Perplexity: 2
Step: 5,000

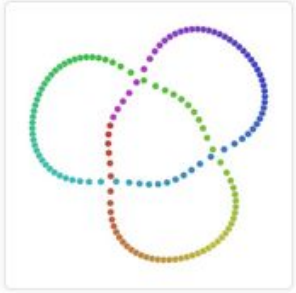
Perplexity: 5
Step: 5,000



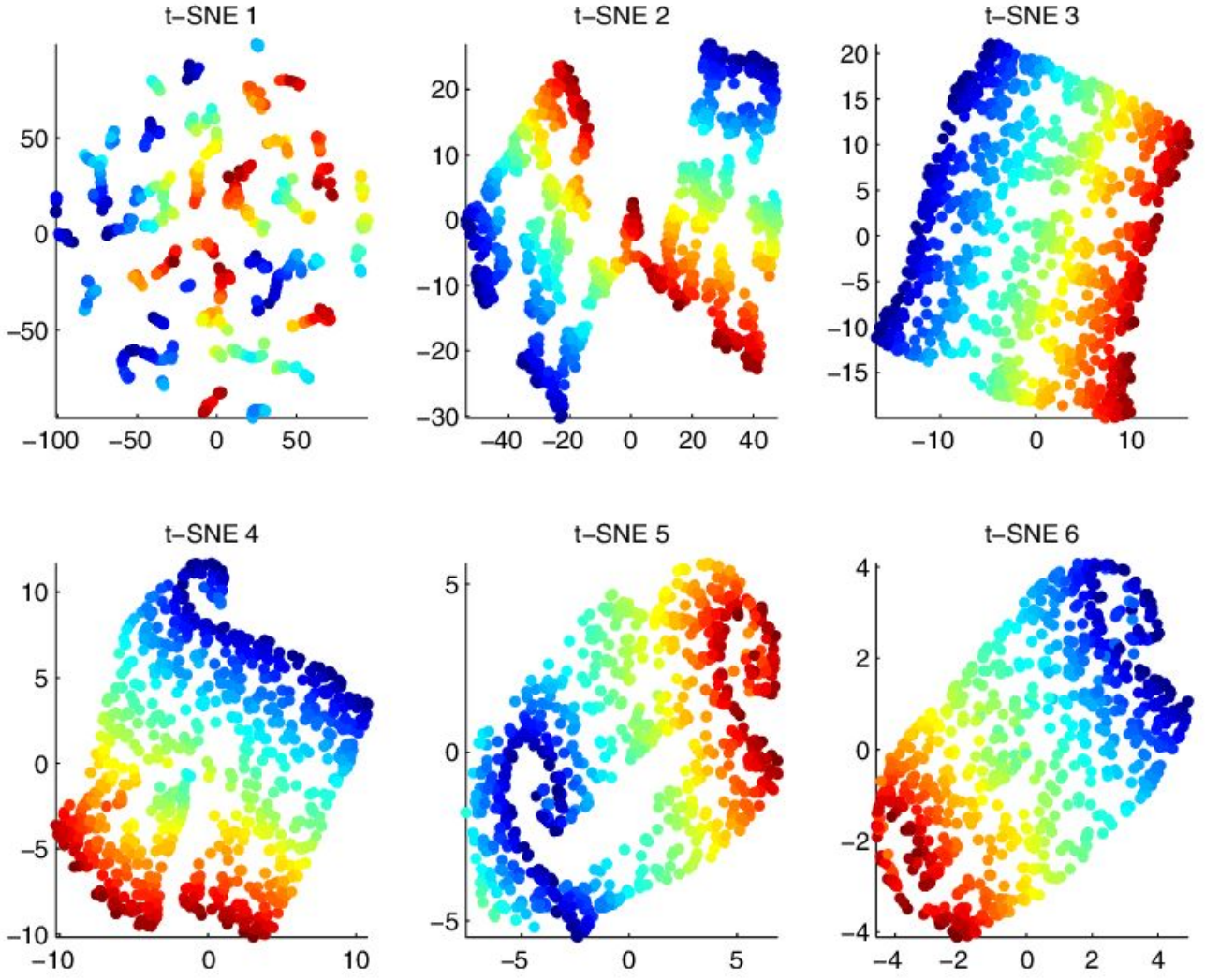
Perplexity: 30
Step: 5,000

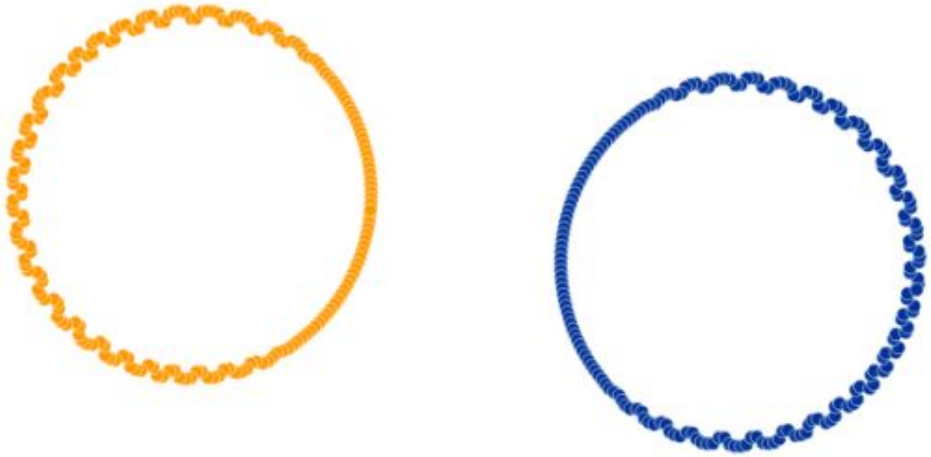


Perplexity: 50
Step: 5,000



Perplexity: 100
Step: 5,000


t–SNE 1
t–SNE 2
t–SNE 3
t–SNE 4
t–SNE 5
t–SNE 6



Step
59,820

Points arranged in 3D, on two linked circles. Different runs may give different results.

Share this view

Number Of Points 200

Perplexity 50

Epsilon 4

More Perplexity Examples

https://distill.pub/2016/misread-tsne/

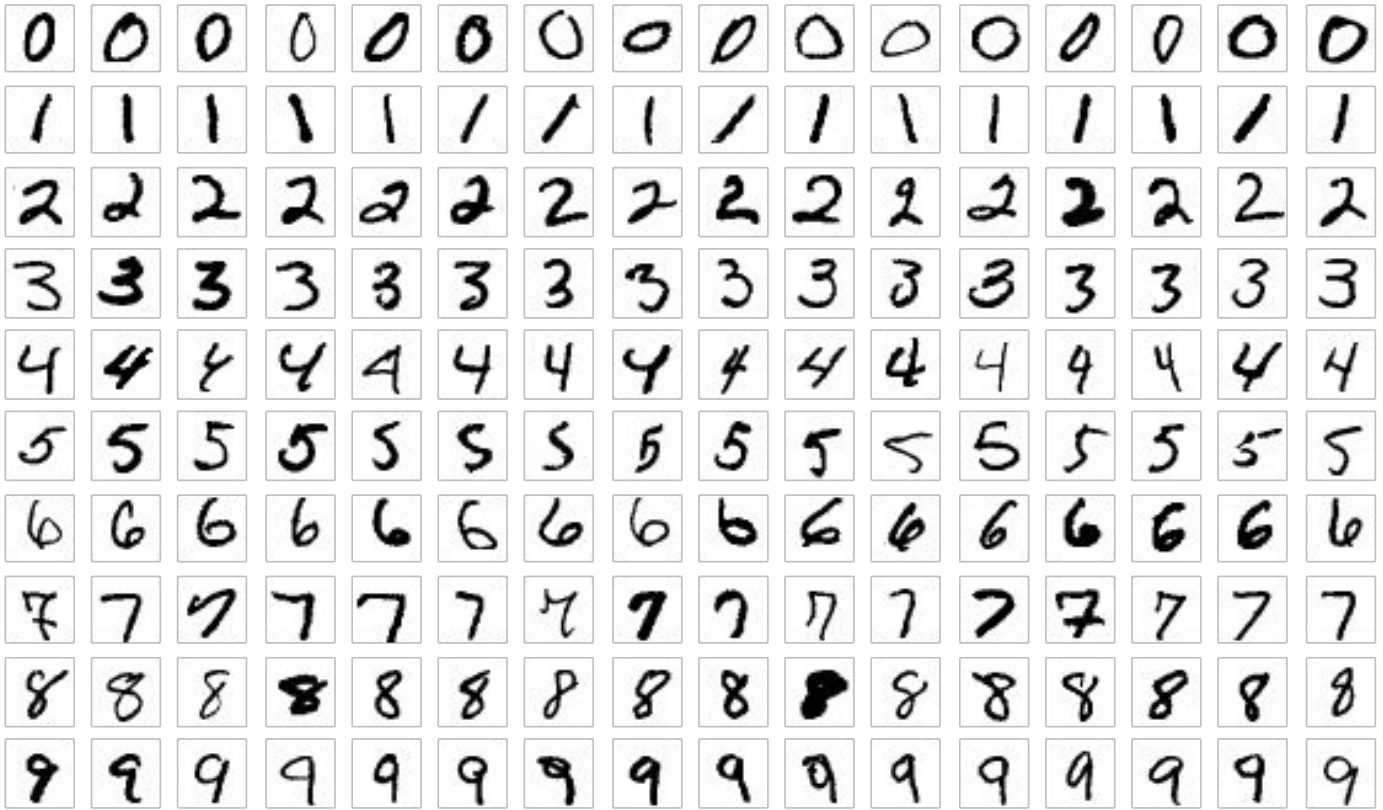
t-SNE: How? Summary

- **Step 1:** Generate conditional probabilities using distance and Gaussian function centered about each of the points in the high dimensional space in your data.
- **Step 2:** Generate t-distribution of the same points in the lower dimensional space. This is the map we will use to visualize our results at the end.
- **Step 3:** Optimize the placement of the points in the low dimensional map by using gradient descent to minimize K - L divergence between Step 1 and Step 2.
- **Step 4:** Plot data map to visualize clusters, similarities, etc. of your high dimensional data.

Case Study: MNIST

- Data: MNIST Digits
- What is it?
  - About 60 thousand handwritten digits
  - Digits range from 0-9
  - Images with 28x28 pixel values (784 dimensions in the data)
  - Data is provided in grayscale pixel values from 0-255 (0 is black, 255 is white)
  - Goal: Figure out similarities between digits, as well as provide visual structure to the 10 different digit labels.
- All of my code is ran with 4000 iterations.

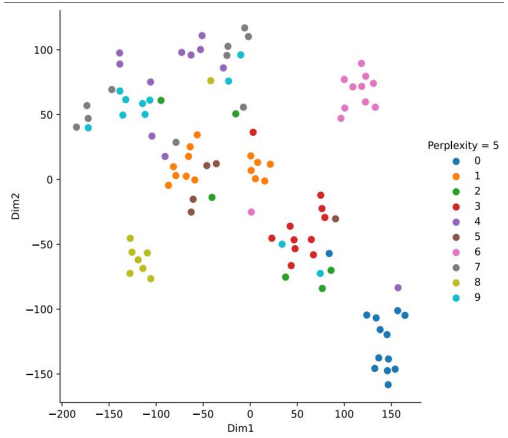
MNIST Digits

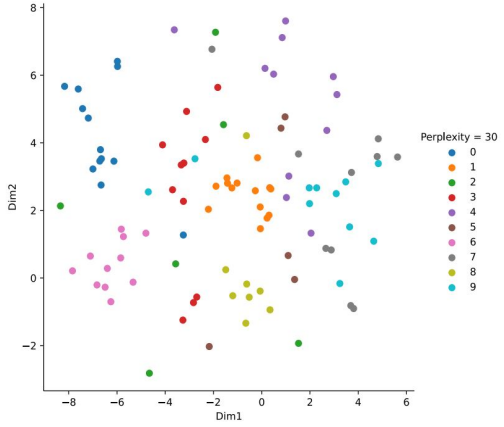


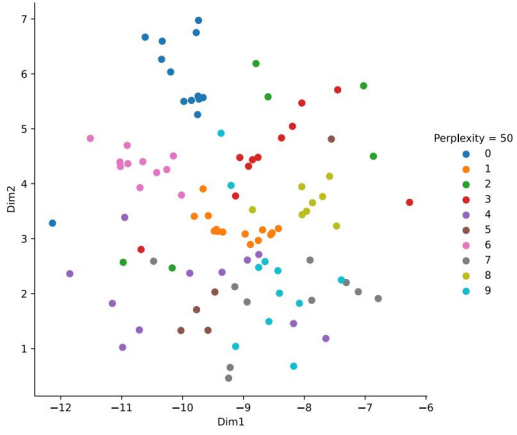
MNIST Data Snapshot in Excel

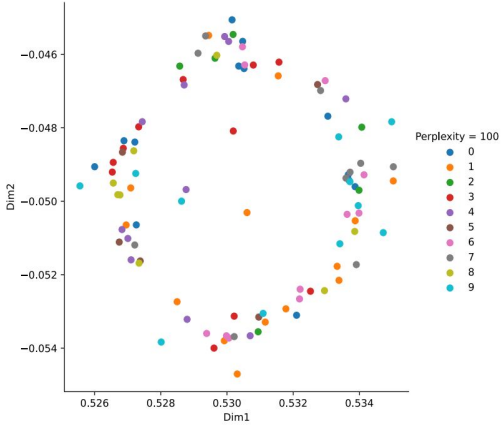
A	B	C	D	E	F	G	H	I	J	K	L
label	1x1	1x2	1x3	1x4	1x5	1x6	1x7	1x8	1x9	1x10	1x11
5	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0

N = 100

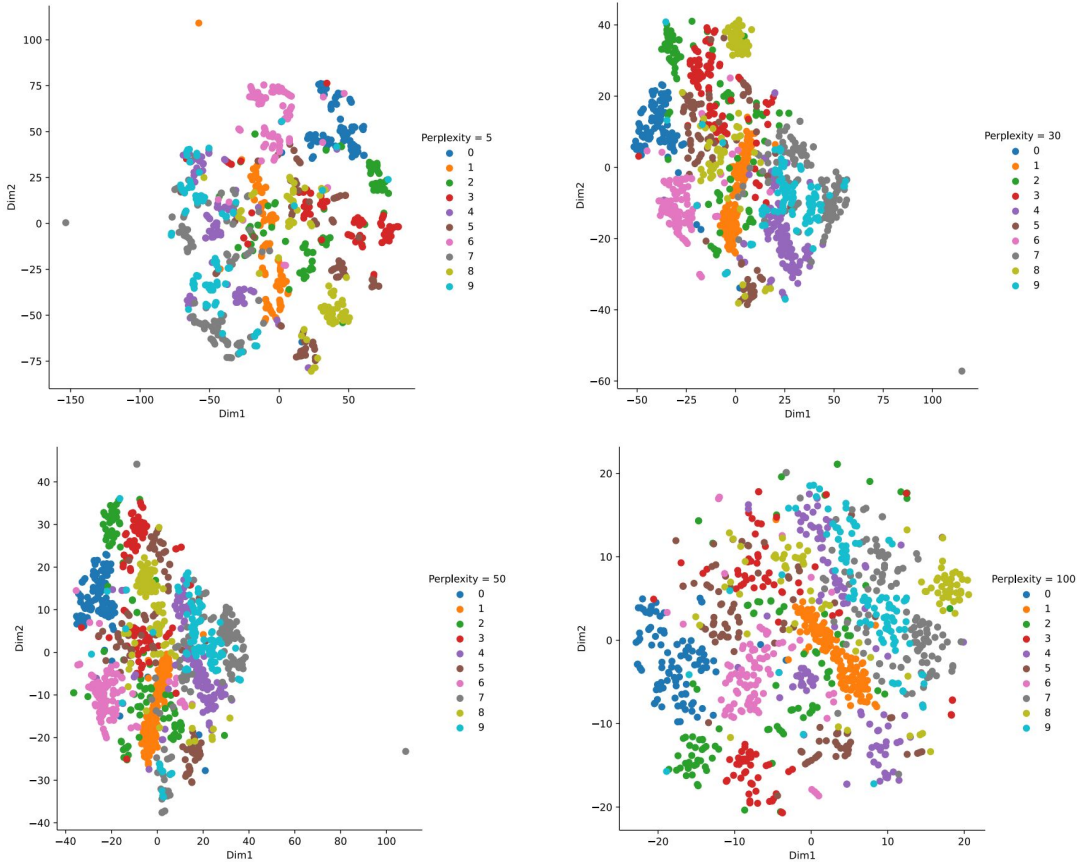

Perplexity = 5
0
1
2
3
4
5
6
7
8
9
Dim1
Dim2


Perplexity = 30
0
1
2
3
4
5
6
7
8
9
Dim1
Dim2

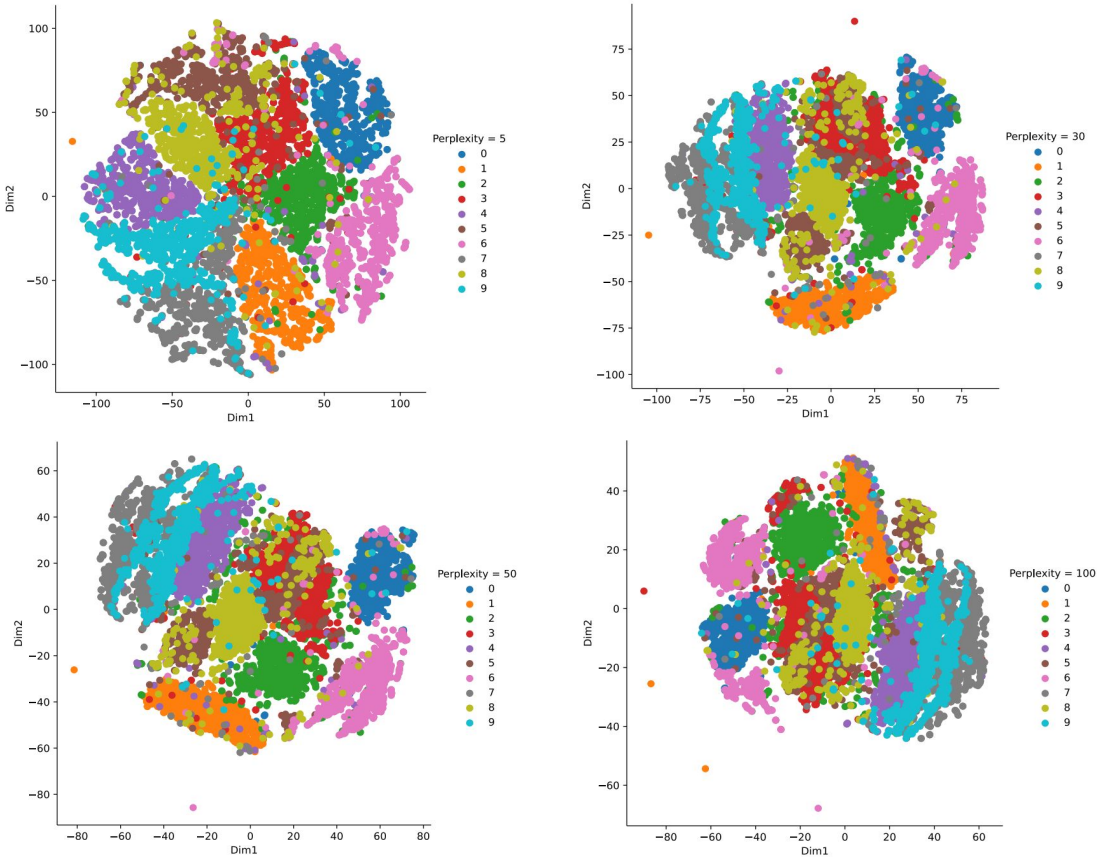

Perplexity = 50
0
1
2
3
4
5
6
7
8
9
Dim1
Dim2


Perplexity = 100
0
1
2
3
4
5
6
7
8
9
Dim1
Dim2

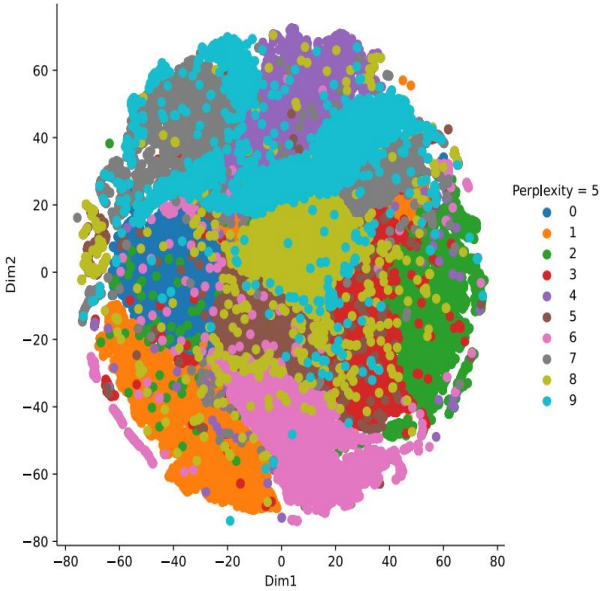
N = 1,000

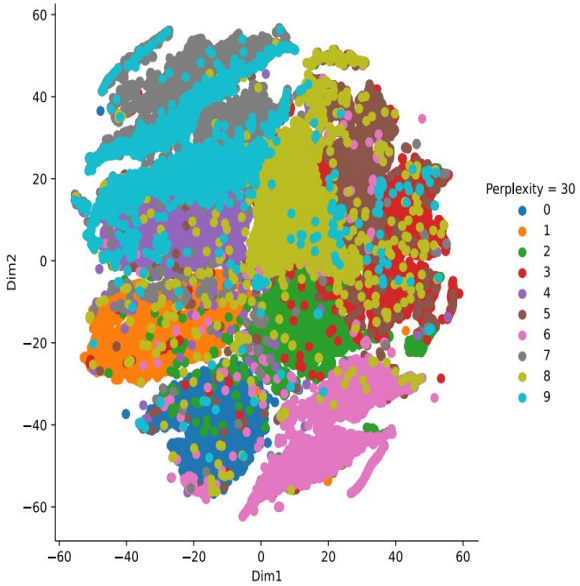

Perplexity = 5
Perplexity = 30
Perplexity = 50
Perplexity = 100
0
1
2
3
4
5
6
7
8
9
Dim1
Dim2

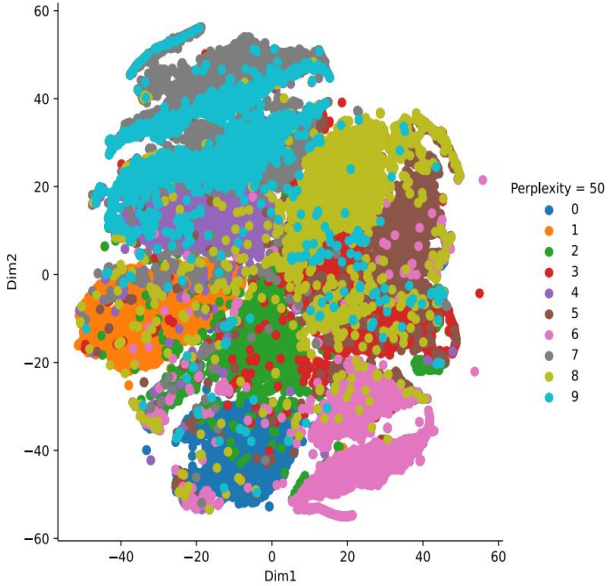
N = 10,000


Perplexity = 5
Perplexity = 30
Perplexity = 50
Perplexity = 100
0
1
2
3
4
5
6
7
8
9
Dim1
Dim2

N = 60000


Perplexity = 5
0
1
2
3
4
5
6
7
8
9
Dim1
Dim2


Perplexity = 30
0
1
2
3
4
5
6
7
8
9
Dim1
Dim2


Perplexity = 50
0
1
2
3
4
5
6
7
8
9
Dim1
Dim2

Drawbacks to t-SNE

1. Non-Convexity of optimization: t-SNE has multiple local minima and is difficult to optimize.

2. t-SNE is not deterministic (it says Stochastic in the name!) which means that running the algorithm with the same hyperparameters may result in slightly varying results.

3. Since t-SNE uses Euclidean distance (which assumes linearity), complex local structure that are non-linear can pose problems for the algorithm. Using other methods such as autoencoders may be a useful alternative.

4. t-SNE on large data sets can be very inefficient. There are ways to improve the speed of the algorithm but at the cost of accuracy. To run my Python script using t-SNE it took several hours to finish the visualizations for the N = 60,000 case.

Recommended videos/articles

- Simple Illustration Example: https://www.youtube.com/watch?v=NEaUSP4YerM
- Google Tech Talk: https://www.youtube.com/watch?v=RJVL80Gg3lA
- Amazing visualizations using various hyperparameters:
  - https://distill.pub/2016/misread-tsne/
- MNIST Database Overview: http://yann.lecun.com/exdb/mnist/
- MNIST in CSV format: https://www.kaggle.com/oddrationale/mnist-in-csv#mnist_train.csv
- ML Explained: https://mlexplained.com/2018/09/14/paper-dissected-visualizing-data-using-t-sne-explained/
- Python Script I used: https://github.com/zwilson999/tsne_project
- t_SNE Paper: http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf