CS 5683: Algorithms & Methods for Big Data Analytics

t-Distributed Stochastic Neighbor Embedding (t-SNE)

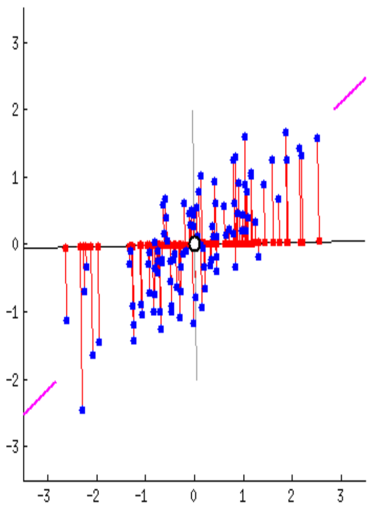
Arunkumar Bagavathi

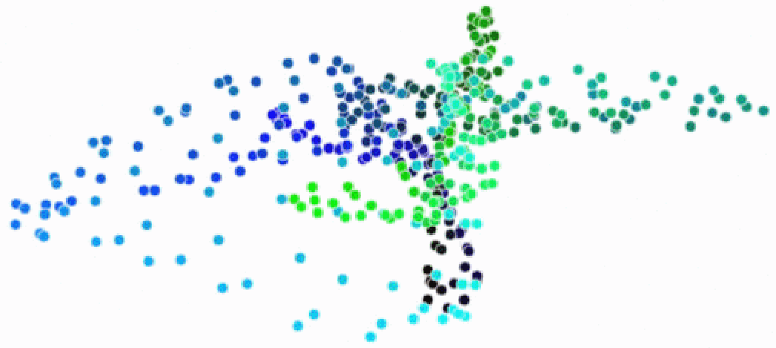
Department of Computer Science

Oklahoma State university

What is t-SNE?

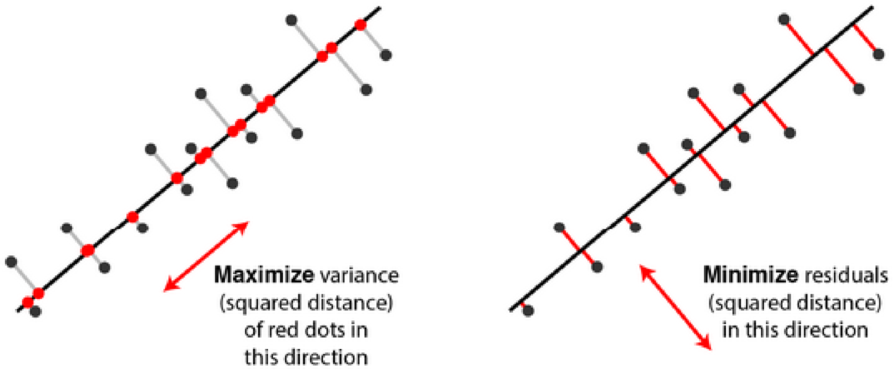
- t-SNE is a dimensionality reduction algorithm
- Like PCA – but not similar to PCA
- t-SNE is something in the category of non-linear dimensionality reduction algorithm



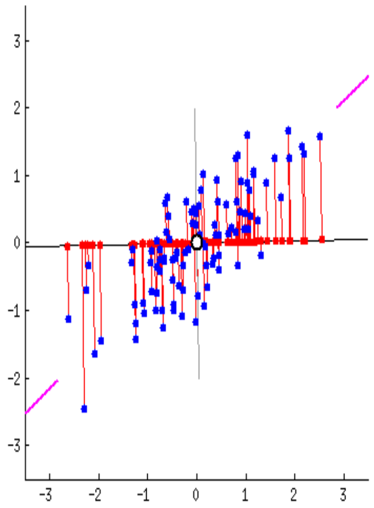


Interpretation of PCA

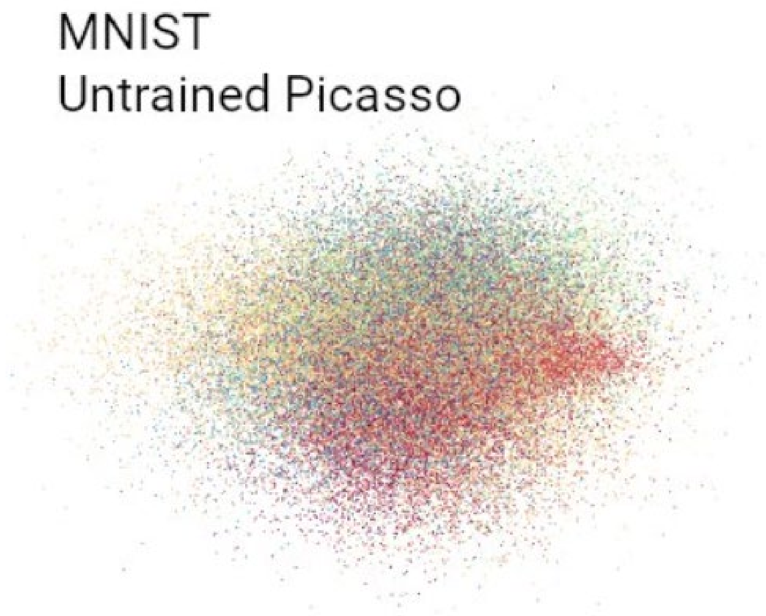
- Two ways of interpretation for PCA:
  - ***Maximize variance*** of projections along each component (dimension)
  - ***Minimize reconstruction error*** between the original and projected coordinates
- Family of PCA algorithms are considered to linear algorithms
- Ideally, they preserve distances in the projected data space rather than giving importance to the neighborhood

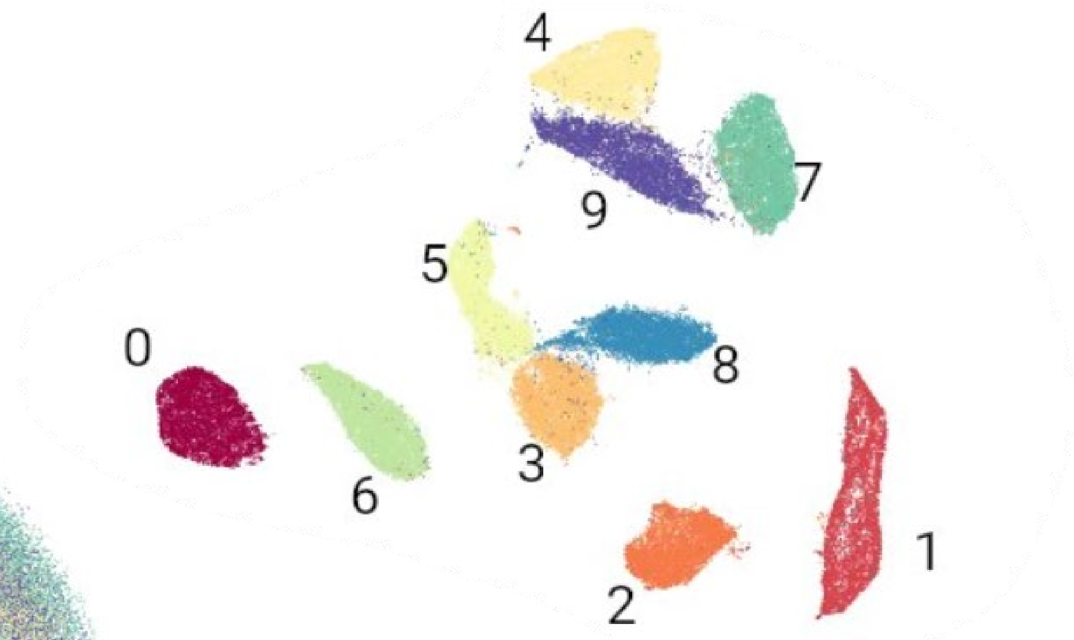


Two equivalent views of principal component analysis.



PCA Limitations Visualized

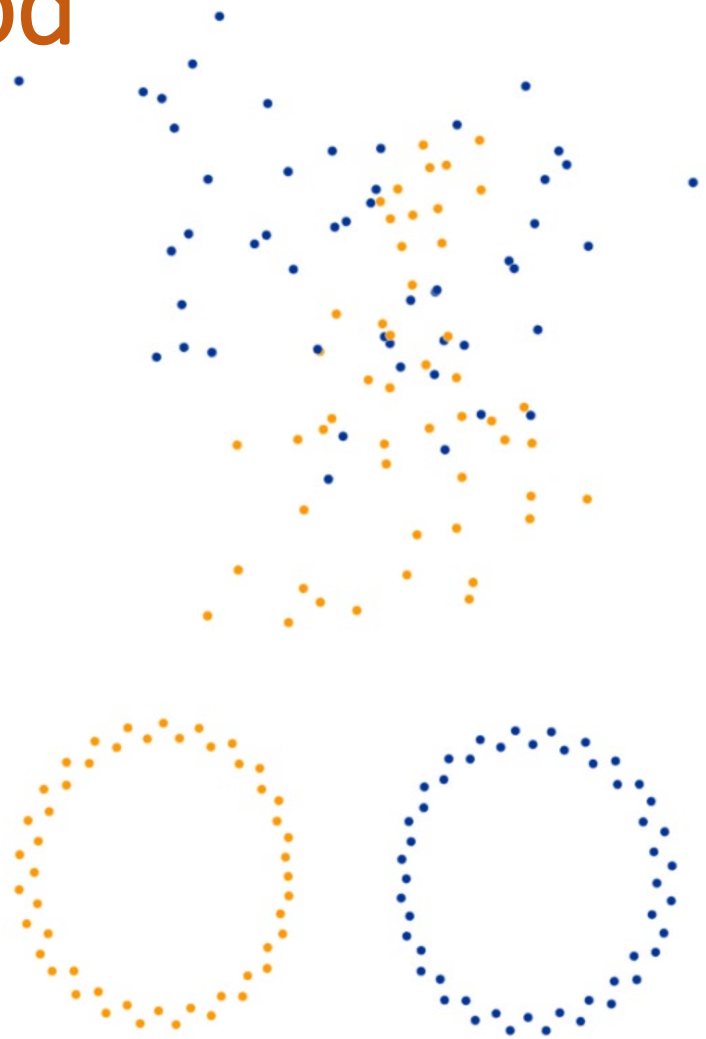




What we expect

Can we do better?

Preserving Neighborhood



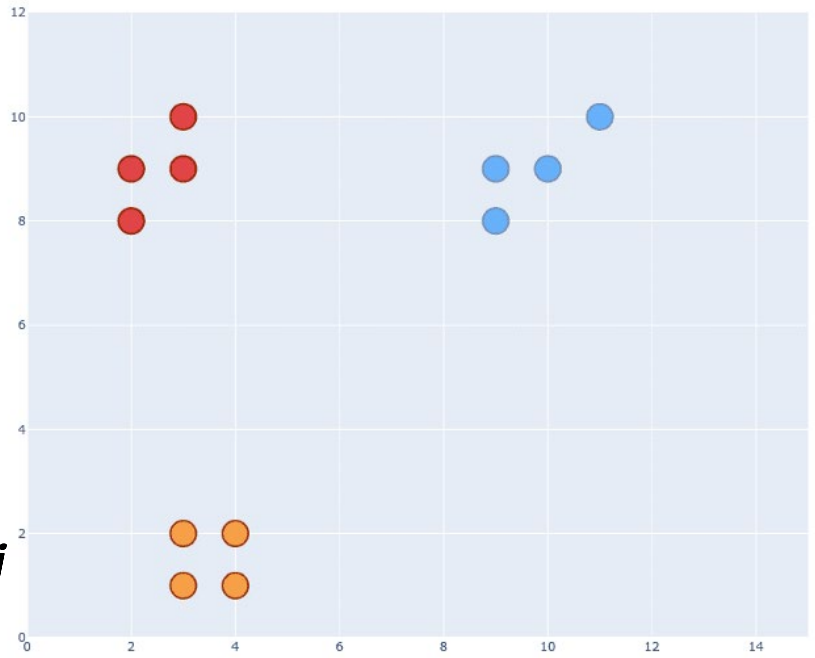
- Reduce the number of dimensions along with preserving neighborhood
- Neighbors are important notion of data mining. For example, consider social networks, twitter followers, and professional networks
- ***Neighbors:*** Data points nearby, measured using some metric space, to the given data
- The problem can be projected as an unsupervised learning task

Neighborhood in SNE

- Neighborhood is identified using a ***similarity*** metric. For example, *Euclidean Distance*

- In SNE, similarity is measured using ***probability distribution***

- Similarity of datapoint x_j to datapoint x_i is considered to be the conditional probability $\boldsymbol{p_{j|i}}$



- Meaning that x_i would pick x_j as its neighbor with probability $p_{j|i}$

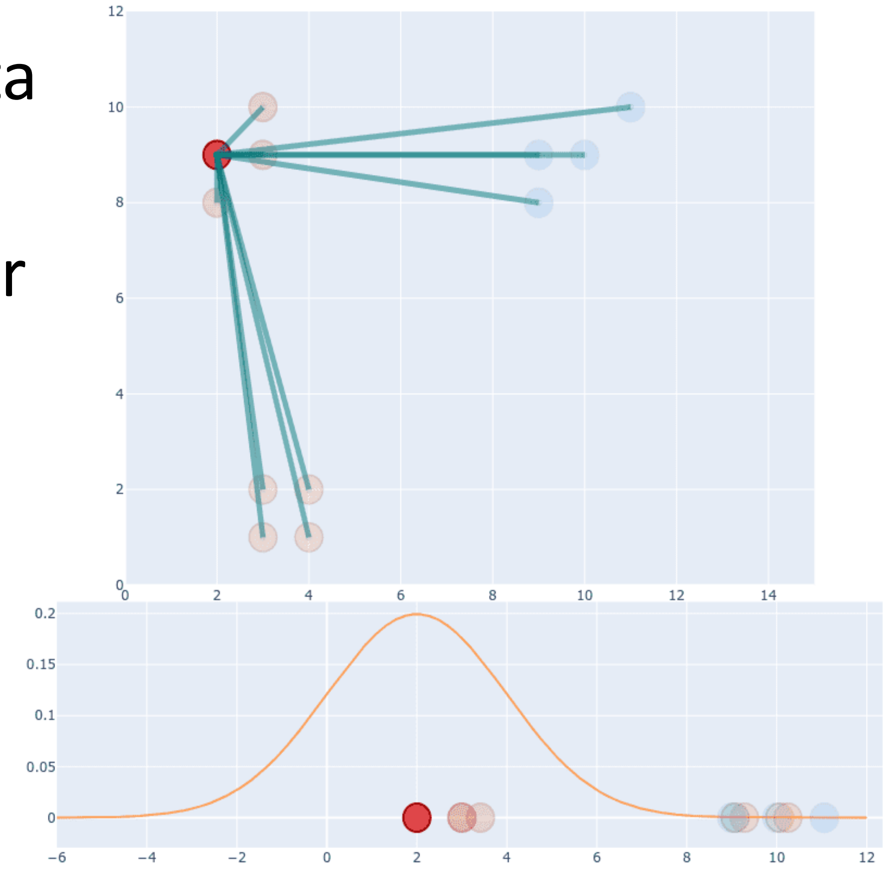
Determining the Neighborhood in SNE

- Pick a random data point (x_i) in the original data

- Measure similarity of the data point to all other data points. Example: Euclidean Distance

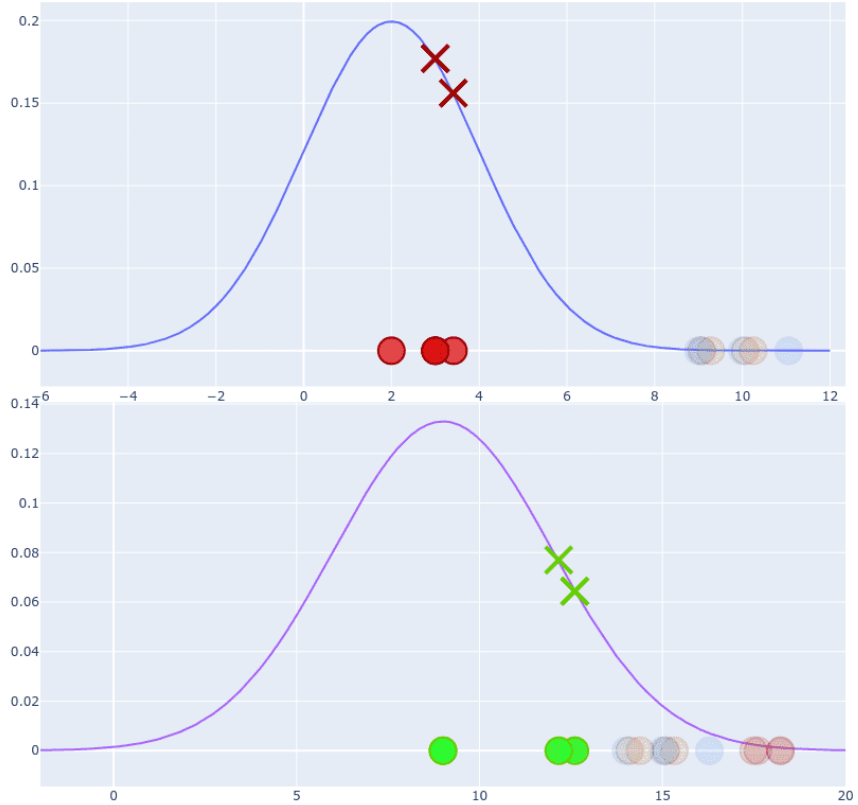
- Plot the points according to the similarity metric on a 1D plane

- The measured similarity metric must be proportional to the probability density under a Gaussian curve centered at x_i

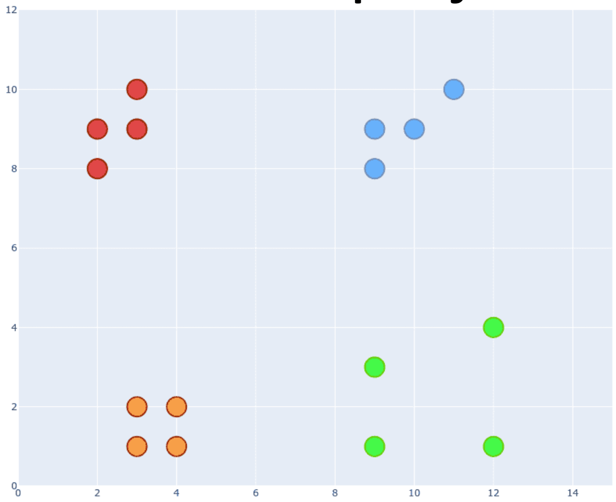


Determining the Neighborhood in SNE

- Including variance (σ^2) and considering scattered clusters



- Follow the same process as before for the data projections



- We can still distinguish similar and non-similar data points
- Notice the probability value is much smaller for the scattered cluster

Tweaking the Probability

- Normalize the projection

$$p_{j|i} = \frac{|x_i - x_j|}{\sum_{k \neq i} |x_i - x_k|}$$

- Normalization scales all values to have sum equal to 1

$$\sum_j p_{j|i} = 1$$

- Also note that $p_{i|i} = 0$

- Let's calculate the probability for our red and green nodes now!

- Red node:

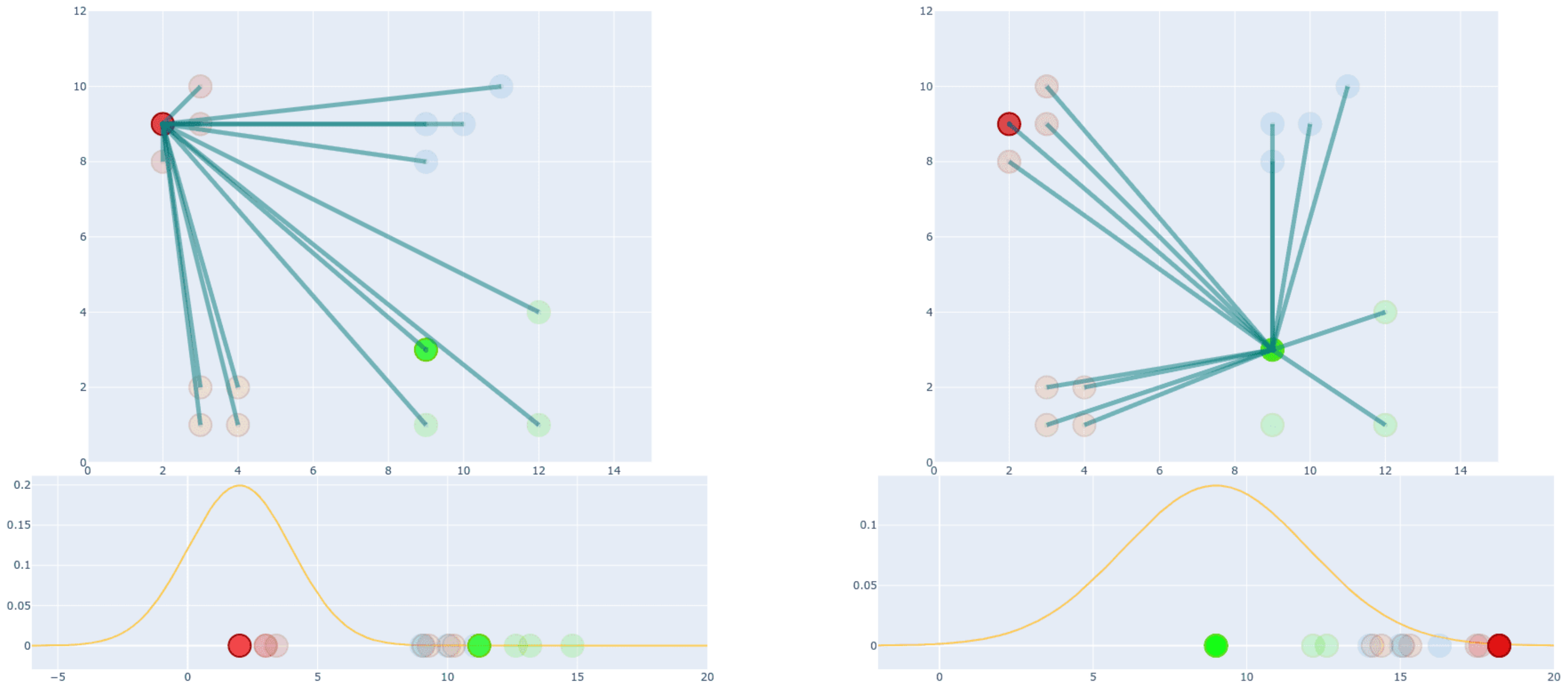
$$p_{j|i} = \frac{0.177}{0.177 + 0.177 + 0.164 + 0.0014 + 0.0013 + \ldots} \approx 0.34$$

- Green node:

$$p_{j|i} = \frac{0.077}{0.077 + 0.064 + 0.064 + 0.032 + 0.031 + \ldots} \approx 0.27$$

Finding Distance Between Points

- Remember: We “tweaked” the probability



Finding Distance Between Points

- The two conditional probabilities $p_{i|j}$ and $p_{j|i}$ will be different
- ***WHY?***
- Fix:

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

N = number of dimensions in the data

Give me SNE!

- The real SNE uses a different probability function:

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

- ***Perplexity:*** Consider this to be a measure of the effective number of neighbors to the data point
  - Important hyper-parameter for all SNE algorithms
  - Ideally choice of perplexity would be between 5 and 50

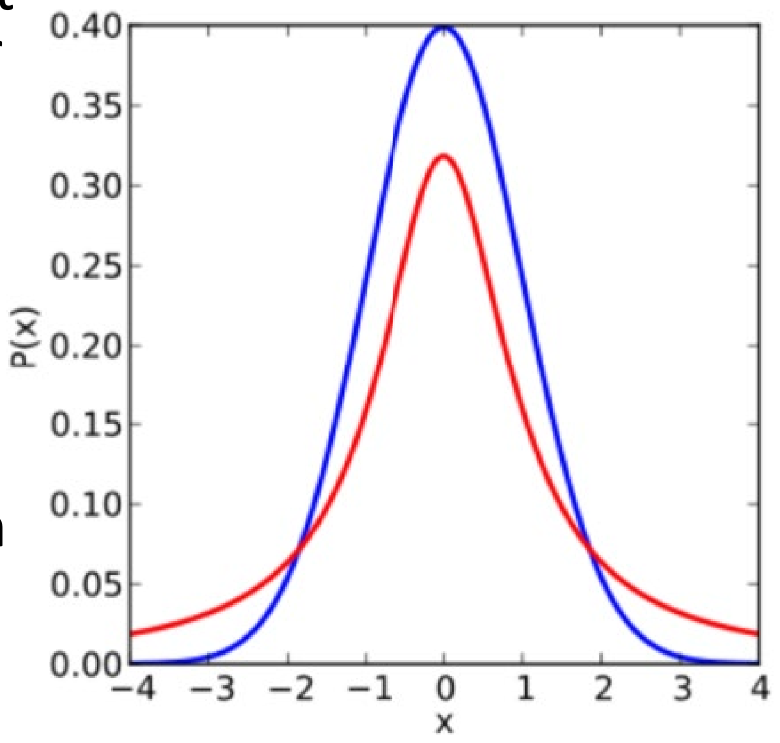
- Variance of a data point is determined by the perplexity
  - The actual SNE algorithms perform a binary search for σ_i using the perplexity hyperparameter

Give me t-SNE!

- The goal of t-SNE is to project the high-dimensional space (p) into some low-dimensional space (q)
- We can start with spreading the data points randomly in the low-dimensional space
- **The real deal:** The algorithm must find the optimal probability distribution in the low-dimensional space.
- **But** we cannot have the low-dimensional space to be Gaussian

The Crowding Problem

- One of the properties of the Gaussian Distribution is “Short Tails” and it causes the data to be crowded near the center
- When embedding neighbors from high-dimensional space to low-dimensional space, there is too little space near a given data point for moderately distant datapoints
- Some far away points are crowded near the center (a given data point)
- We need some other “long tail” distribution for the low-dimensional space!

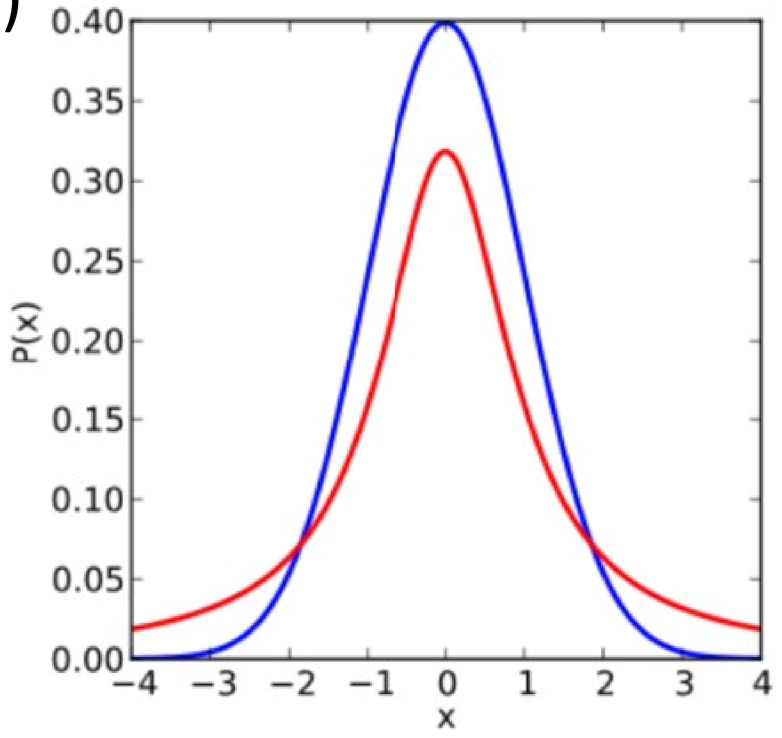


The Crowding Problem Fix

- t-SNE algorithm uses the student t-distribution (or) simply ***t-distribution***

- It falls rapidly near the center and it has long tail

- The neighborhood probability falls less rapidly near the tail. Thus far away points are still a little far from center and cluster points are still close together in the center



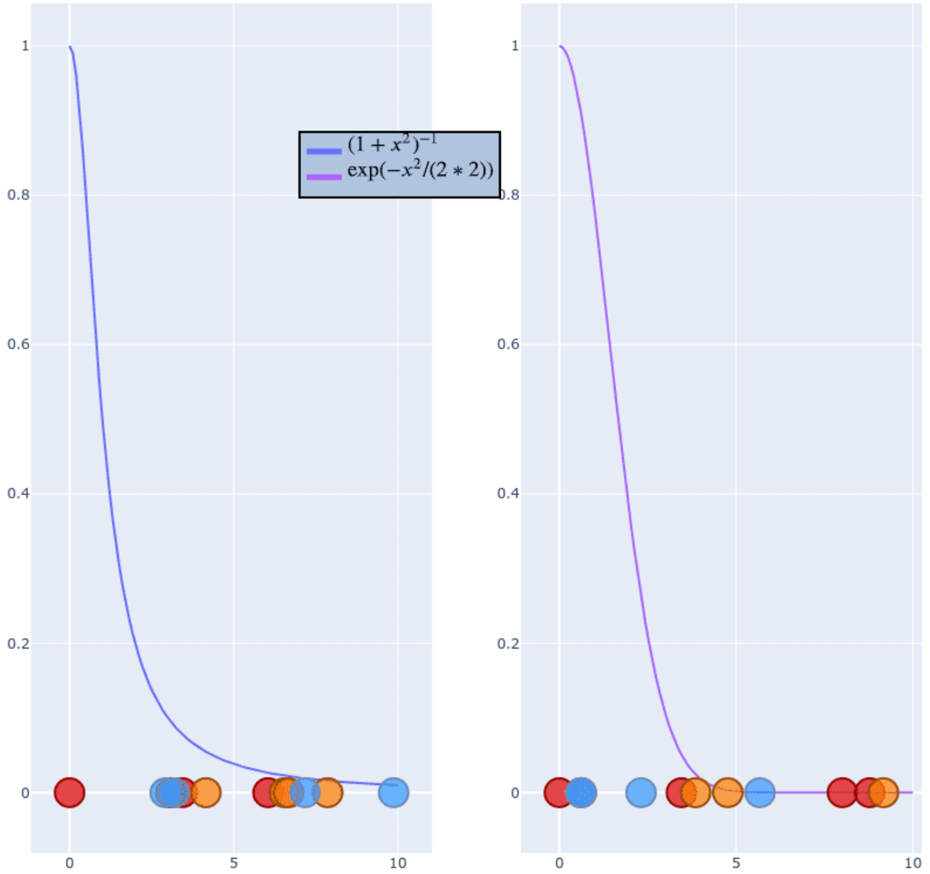
t-SNE Projections

- t-SNE algorithm uses a new formula to get the probability distribution of the low-dimensional space (q):

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

- Instead of the formula used for high-dimensional distribution:

$$q_{ij} = \frac{\exp(-\|y_i - y_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq l} \exp(-\|y_k - y_l\|^2 / 2\sigma_i^2)}$$



t-SNE Objective

- ***Given:*** $x^1, x^2, \ldots\ldots x^m \in \mathbb{R}^N$, we define distributions p_{ij}

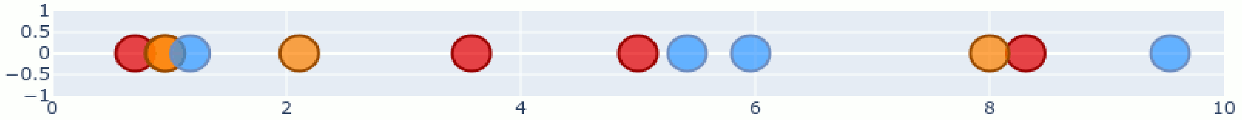
$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2/2\sigma_i^2)}$$

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

- ***Goal:*** Find a good embedding $y^1, y^2, \ldots y^m \in \mathbb{R}^n$ for some $n << N$ (usually 2 or 3). We define the distribution (q_{ij}) for this embedding

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

- ***Objective:*** Optimize $\mathbf{q_{ij}}$ to be as close as $\mathbf{p_{ij}}$

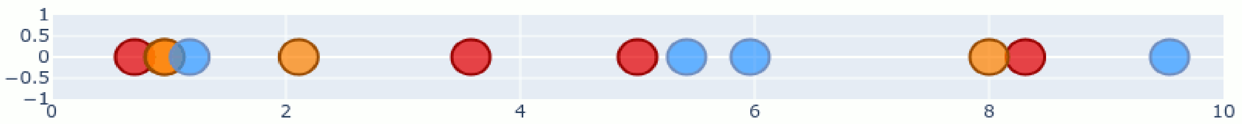


t-SNE Minimization

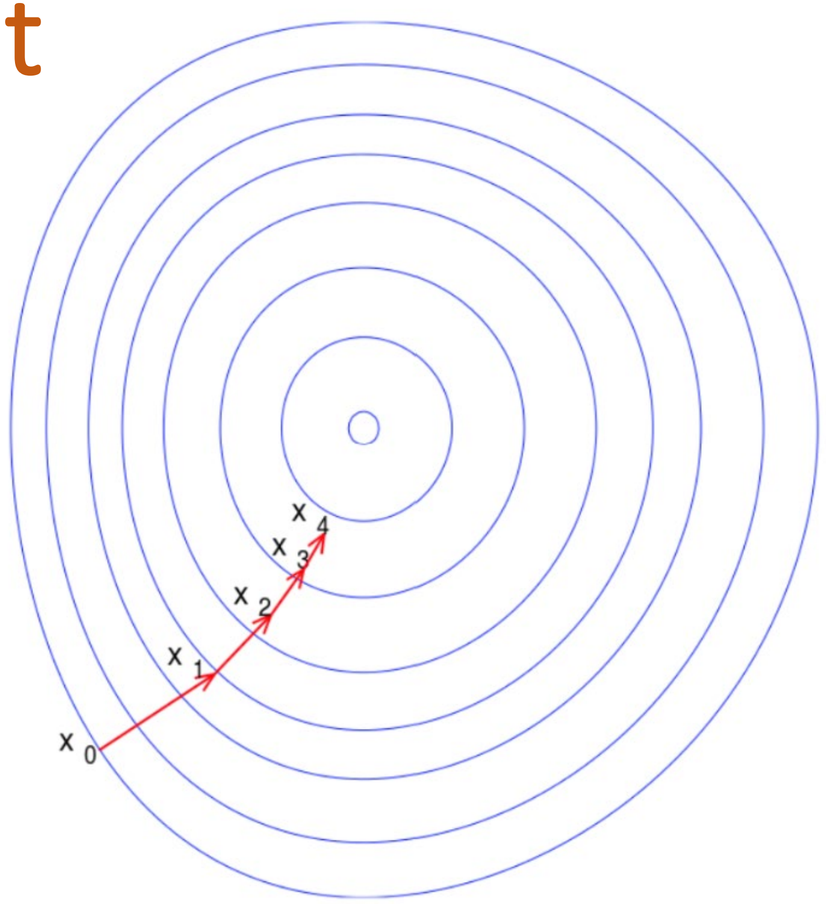
- t-SNE performs the optimization using ***Kullback-Leibler (KL) Divergence*** *between conditional probabilities p_{ij} and q_{ij}*

$$C = D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

- Minimize the divergence between two probabilities
- ***How?***

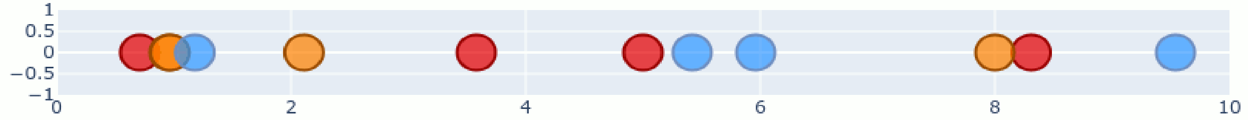


Gradient Descent



- Adjust the output coordinates (y) according to the gradient of the given optimization function (C)
- **Gradient Descent:** Iterative process to find minimal of a function
- **Steps:**
  - Start with a random initial output distribution
  - Iteratively calculate the gradient
  - In each iteration, treat the gradient as the force to push and pull points to make the input and output distributions more similar

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



Implementing t-SNE

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

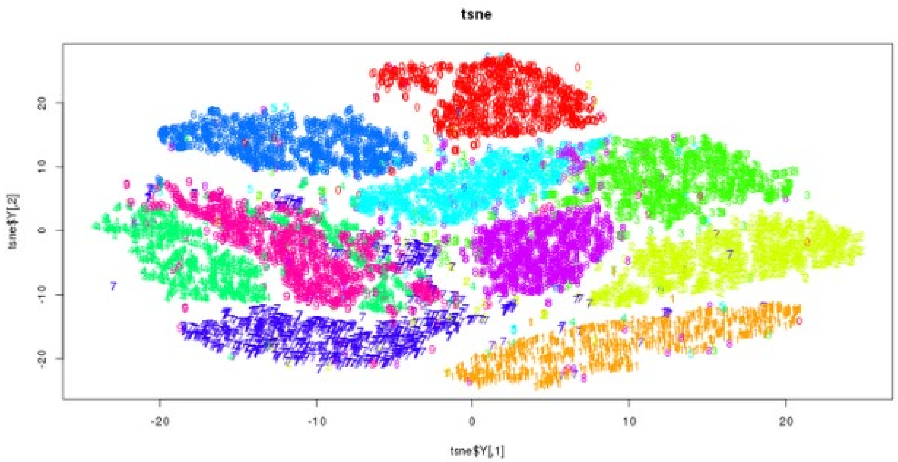
Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4} I)$
  for t=1 **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    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

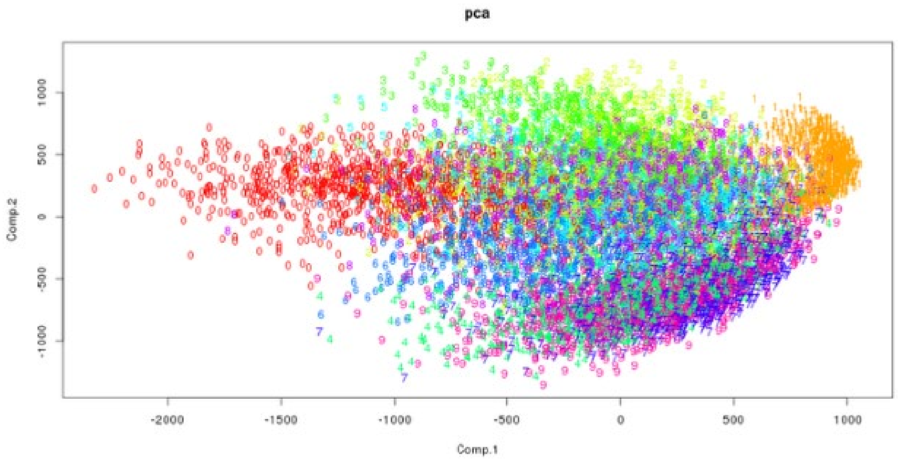
$$p_{j|i} = \frac{exp(-||x_i - x_j||^2 / 2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2 / 2\sigma_i^2)}$$

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

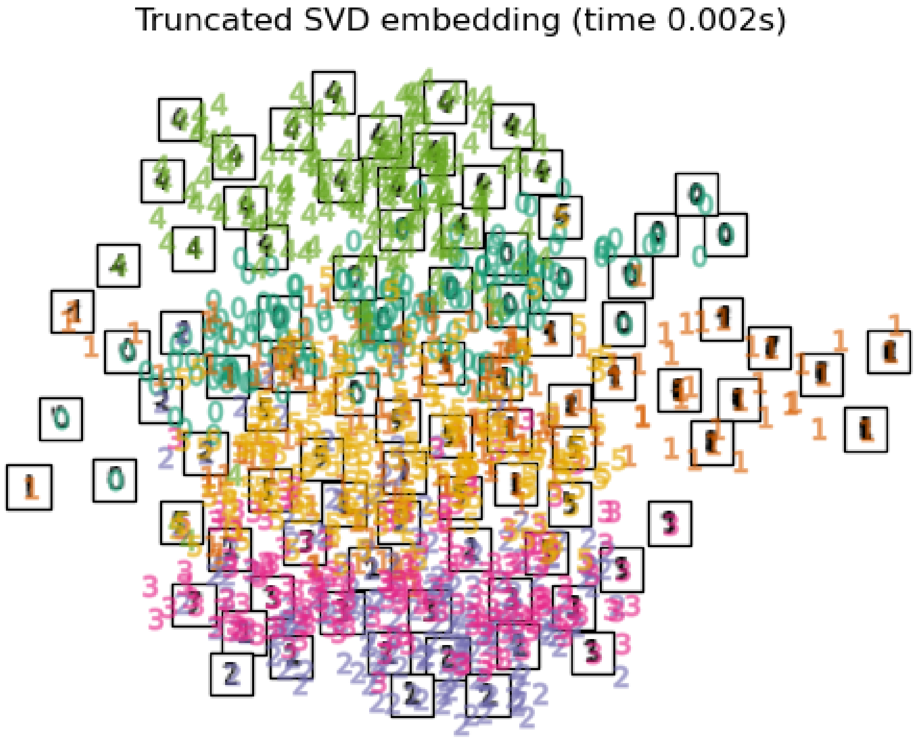
$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$$

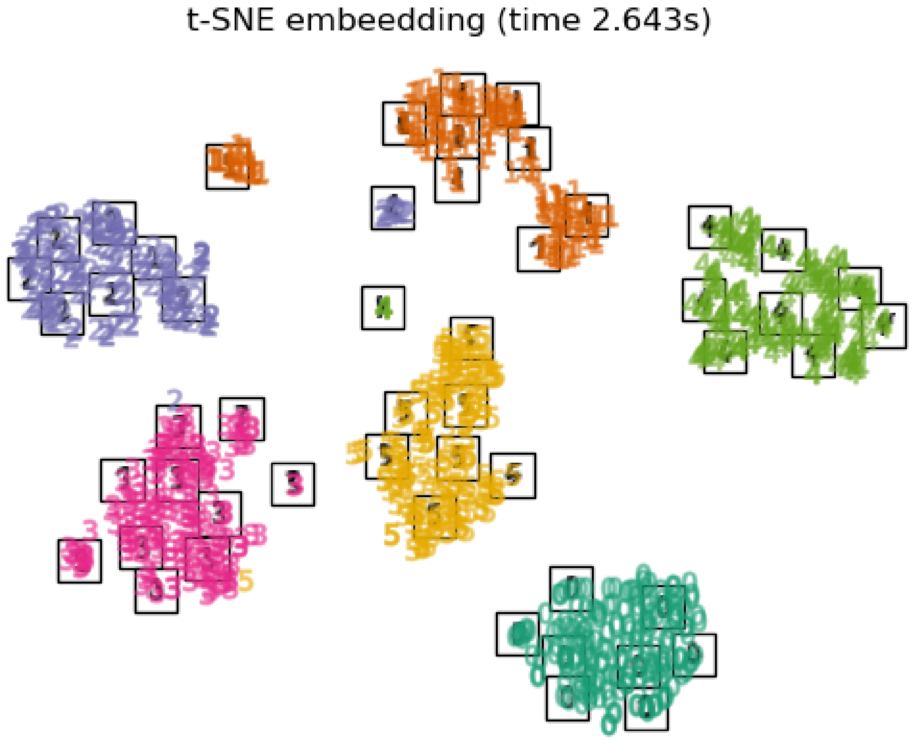
t-SNE vs. PCA (1)





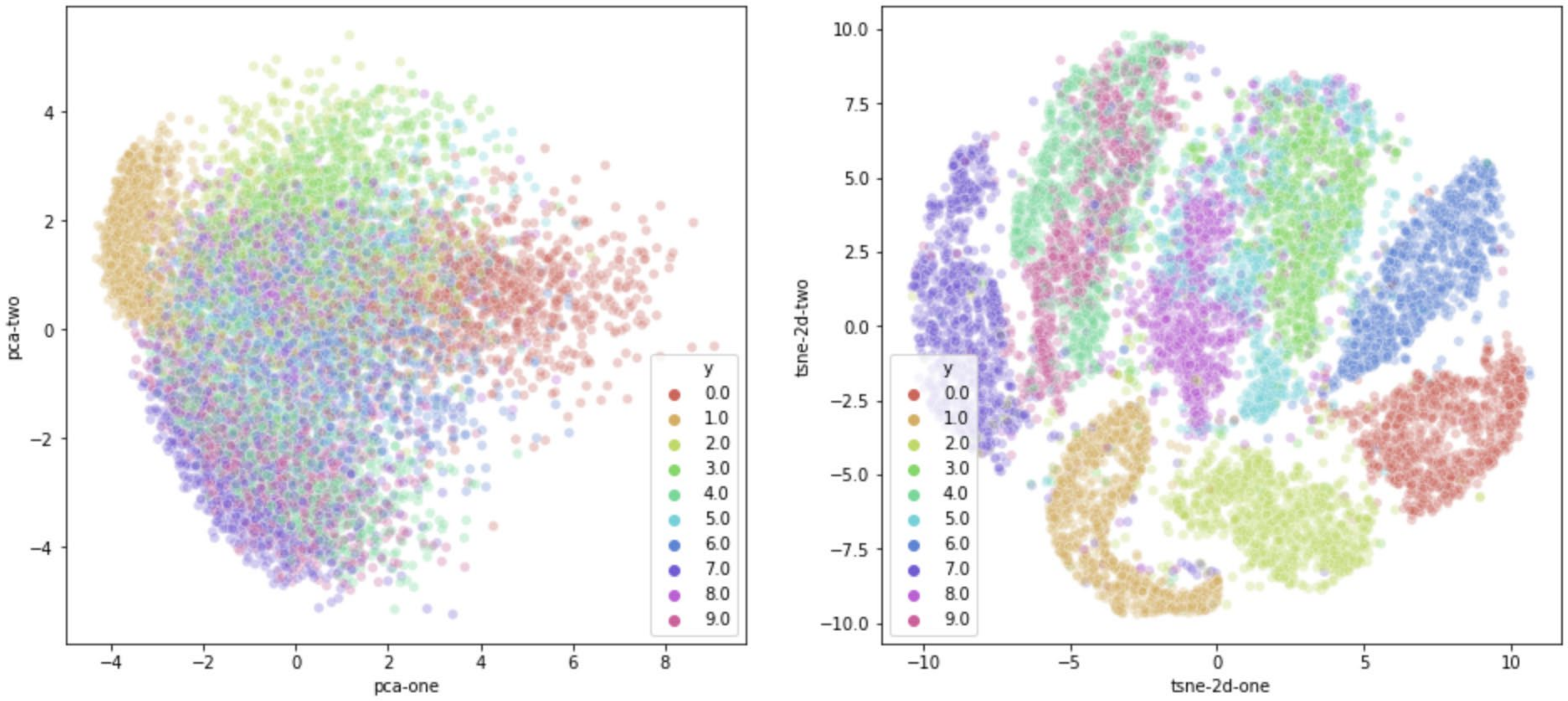
t-SNE vs. PCA (2)





http://scikit-learn.org/stable/auto_examples/manifold/plot_lle_digits.html

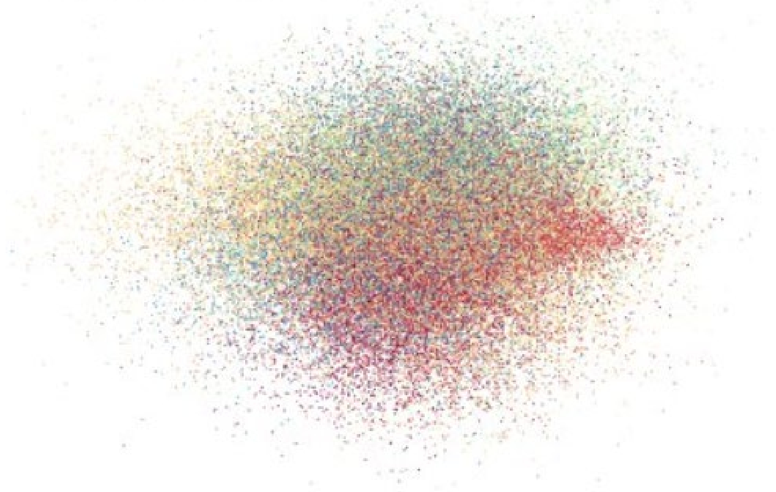
t-SNE vs. PCA (3)



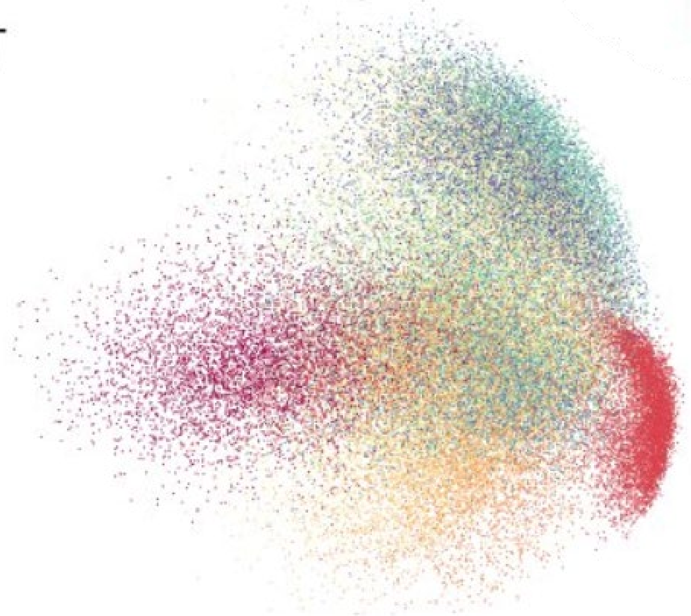
https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b

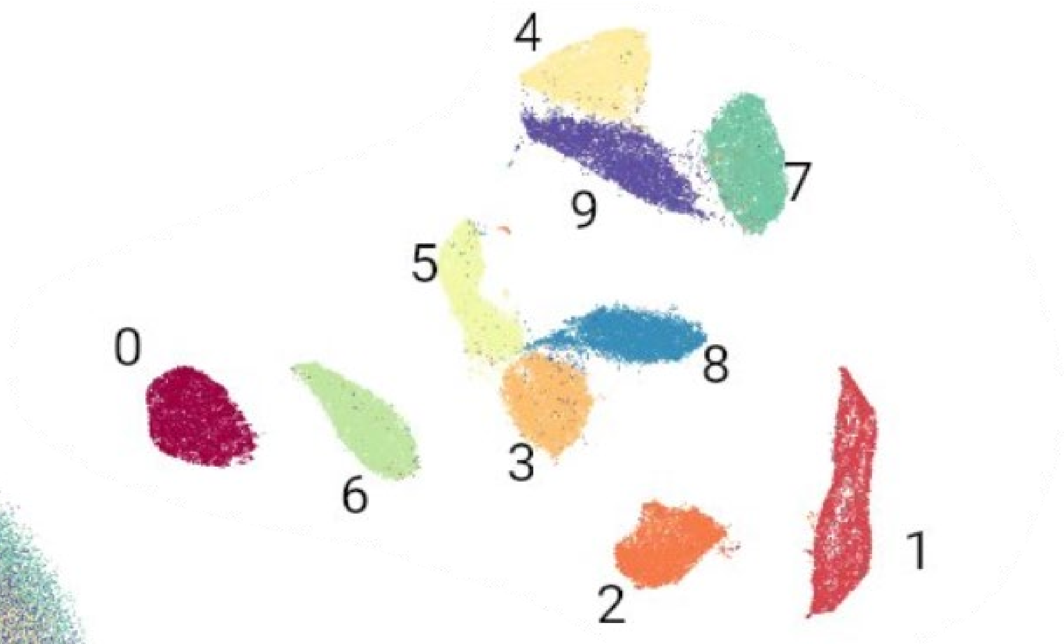
t-SNE vs. PCA (4)

MNIST
Untrained Picasso



MNIST
PCA





MNIST t-SNE

Playing with t-SNE

- https://lvdmaaten.github.io/tsne/
- https://scikit-learn.org/stable/auto_examples/manifold/plot_t_sne_perplexity.html
- https://distill.pub/2016/misread-tsne/

Limitations of t-SNE

- **Extremely slow!** Pairwise conditional probabilities for each data point
  - ***Solution:*** Use PCA and t-SNE in conjunction. Use PCA to reduce the number of dimensions and then use t-SNE to reduce further the number of dimensions
- **Non-deterministic algorithm**. Each execution of the algorithm on the same dataset may produce different result
- **Hyperparameter tuning**

Questions???

References

- Some figures are taken from https://towardsdatascience.com/t-sne-clearly-explained-d84c537f53a
- Some slides are motivated from http://www.sci.utah.edu/~beiwang/teaching/cs6965-spring-2018/Lecture03-tSNE.pdf and https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec13_handout.pdf