T-distributed Stochastic Neighbor Embedding (tSNE)

Tuan Nguyen

Ngày 28 tháng 12 năm 2022

Overview

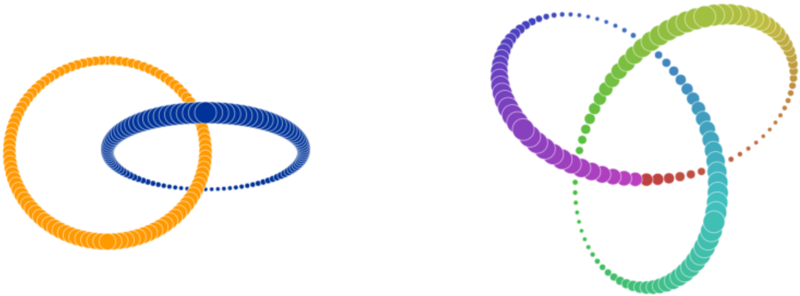
PCA - linear projection

t-SNE introduction

SNE algorithm

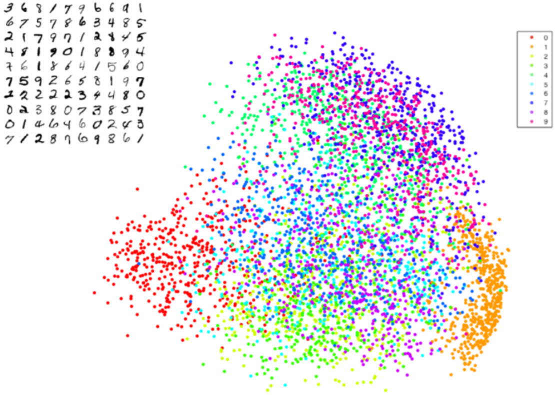
t-SNE algorithm

PCA - linear projection

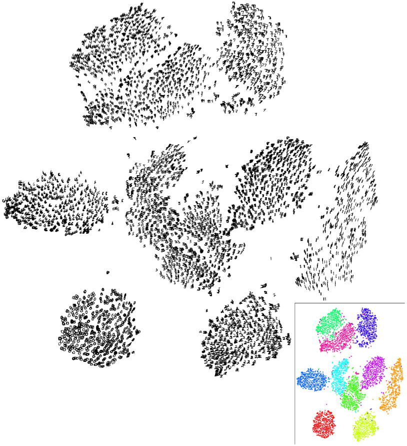


Hình 1: Linearly nonseparable data

PCA - MNIST



tSNE - MNIST



t-SNE

- t-SNE is an alternative dimensionality reduction algorithm
- PCA tries to find a global structure
  - Low dimensional subspace
  - Can lead to local inconsistencies -> Far away point can become nearest neighbors
- t-SNE tries to perserve local structure
  - Low dimensional neighborhood should be the same as original neighborhood

SNE

SNE basic idea:

- "Encode" high dimensional neighborhood information as a distribution
- Find low dimensional points such that their neighborhood distribution is similar
- Intuition: Random walk between data points
  - High probability to jump to a close point
- How do you measure distance between distributions?
  - Most common measure: KL divergence

Neighborhood Distribution

SNE basic idea:

- Consider the neighborhood around an input data point $x_i \in \mathbb{R}^d$
- Imagine that we have a Gaussian distribution centered around x_i
- Then the probability that x_i chooses some other datapoint x_j as its neighbor is in proportion with the density under this Gaussian
- A point closer to x_i will be more likely than one further away

The probability that point x_i chooses x_j as it neighbor:

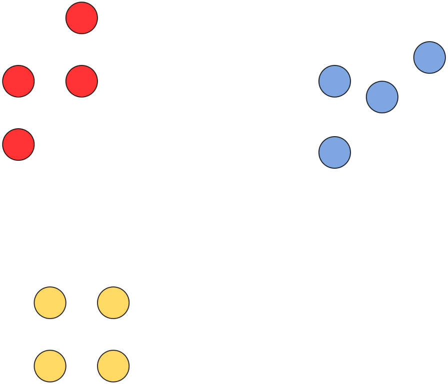
$$p_{j|i} = \frac{\exp\left\{-\left\|x^{(i)} - x^{(j)}\right\|^2 / 2\sigma_i^2\right\}}{\sum_{k \neq i} \exp\left\{-\left\|x^{(i)} - x^{(k)}\right\|^2 / 2\sigma_i^2\right\}} \tag{1}$$

with $p_{i|i} = 0$

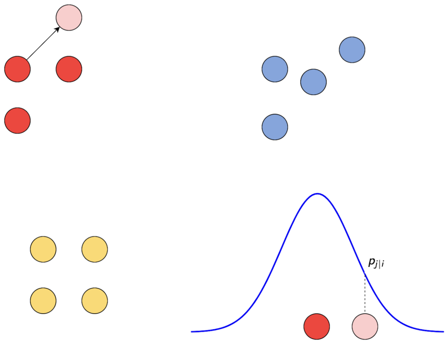
Final distribution over pairs is symmetrized:

$$p_{ij} = \frac{1}{2N}(p_{i|j} + p_{j|i}) \tag{2}$$

Example 1



Example 2



Perplexity

- The parameter σ_i sets the size of the neighborhood
  - Very low σ_i - all the probability is in the nearest neighbor
  - Very high σ_i - Uniform weights
- Here we set σ_i differently for each data point
- Results depend heavily on σ_i - it defines the neighborhoods we are trying to preserve.

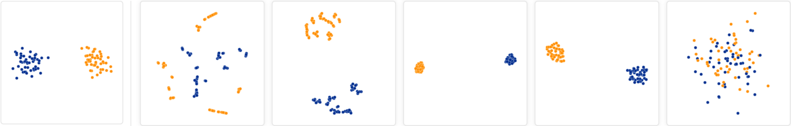
Perplexity (cont.)

- For each distribution $P_{j|i}$ (depends on σ_i) we define the perplexity

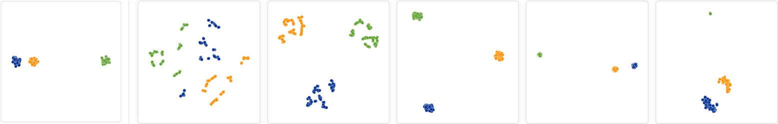
$$perp(P_{j|i}) = 2^{H(P_{j|i})}, H(P) = -\sum_i P_i \log(P_i) \text{ is the entropy} \quad (3)$$

- If P is uniform over k elements - perplexity is k
  - Low perplexity = small σ
  - High perplexity = large σ
- Values between 5-50 usually work well
- Important parameter - different perplexity can capture different scales in the data

Perplexity (cont.)



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

SNE objective

- Given $x^{(1)}, .., x^{(N)} \in \mathbb{R}^D$ we define the distribution P_{ij}
- Goal: Find good embedding $y^{(1)}, .., y^{(N)} \in \mathbb{R}^d$ for some d < D (normally 2 or 3)
- For points $y^{(1)}, .., y^{(N)} \in \mathbb{R}^d$ we can define distribution Q similarly the same (notice no σ_i and not symmetric)

$$Q_{ij} = \frac{\exp\left\{-\left\|y^{(i)} - y^{(j)}\right\|\right\}^2}{\sum_k \sum_{l \neq k} \exp\left\{-\left\|y^{(l)} - y^{(k)}\right\|^2\right\}} \tag{4}$$

- Optimize Q to be close to P: Minimize KL - divergence -> to find the embedding (parameter) $y^{(1)}, .., y^{(N)} \in \mathbb{R}^d$

Example 2

Measure the distance between two distributions, P and Q:

$$KL(Q||P) = \sum_{ij} Q_{ij} \log\left(\frac{Q_{ij}}{P_{ij}}\right) \tag{5}$$

KL Properties:

- KL(Q||P) $\geq$ 0 and zero only when Q $=$ P
- KL(Q||P) is a convex function

SNE algorithm

- We have P, and are looking for $y^{(1)}, .., y^{(N)} \in \mathbb{R}^d$ such that the distribution Q we infer will minimize $\mathrm{L(Q)} = \mathrm{KL(P||Q)}$
- Note that

$$KL(P||Q) = \sum_{ij} P_{ij} \log\left(\frac{P_{ij}}{Q_{ij}}\right) = -\sum_{ij} P_{ij} \log(Q_{ij}) + \text{const} \tag{6}$$

- Can show that

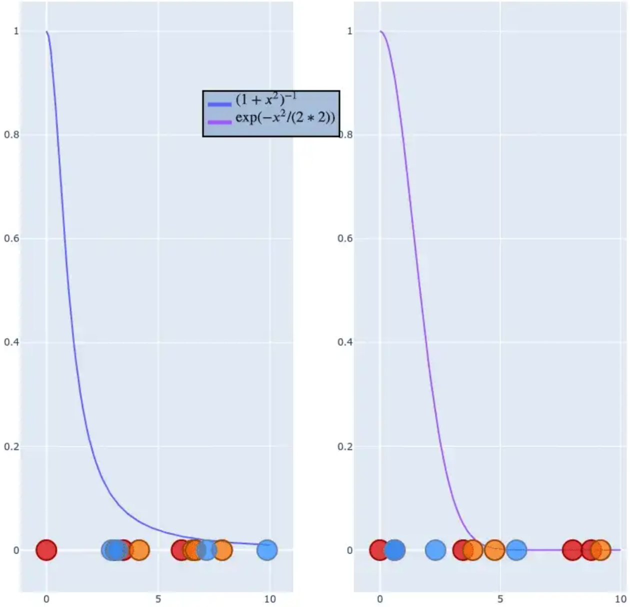
$$\frac{\partial L}{\partial y^{(i)}} = \sum_j (P_{ij} - Q_{ij})(y^{(i)} - y^{(j)}) \tag{7}$$

- Crowding problem...

Crowding problem

- In high dimension we have more room, points can have a lot of different neighbors
- In 2D a point can have a few neighbors at distance one all far from each other - what happens when we embed in 1D?
- This is the "crowding problem" - we don't have enough room to accommodate all neighbors.
- This is one of the biggest problems with SNE.
- t-SNE solution: Change the Gaussian in Q to a heavy tailed distribution -> if Q changes slower, we have more "wiggle room" to place points at.

Crowding problem (cont.)



t-SNE

t-Distributed Stochastic Neighbor Embedding

- Probability goes to zero much slower then a Gaussian
- We can now redefine Q_{ij} as

$$Q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|y_k - y_l\|^2)^{-1}} \tag{8}$$

- We use the same P_{ij}
- the gradients of t-SNE objective are

$$\frac{\partial L}{\partial y^{(i)}} = \sum_j (P_{ij} - Q_{ij})(y^{(i)} - y^{(j))}(1 + \|y_i - y_j\|^2)^{-1} \tag{9}$$