

# CSC 411 Lecture 13:t-SNE

Ethan Fetaya, James Lucas and Emad Andrews

University of Toronto

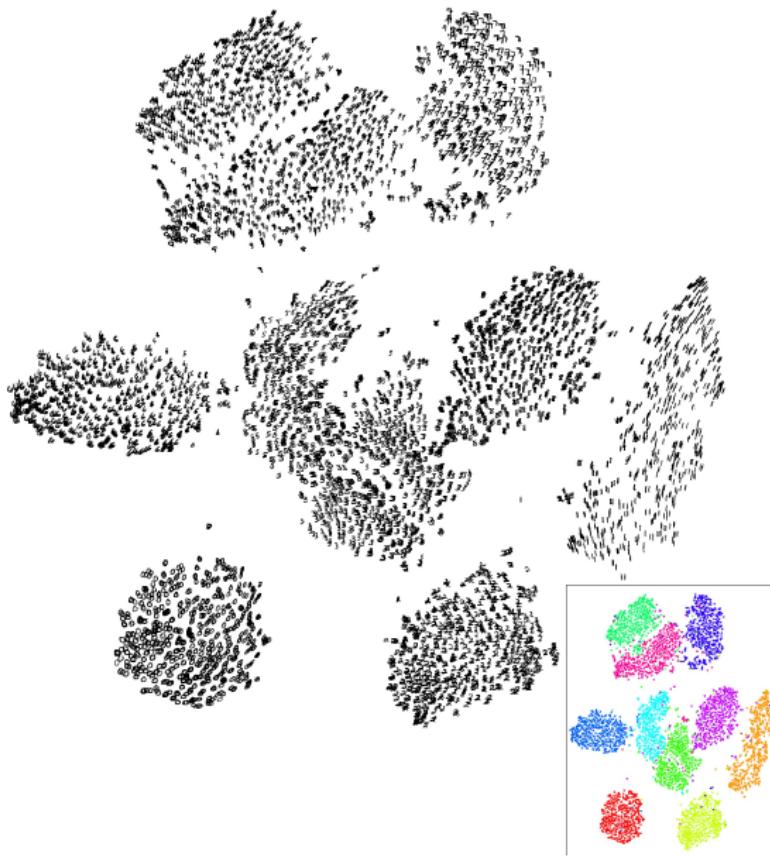
# Today

- SNE - Stochastic Neighbor Embedding
- t-SNE
- KL divergence

# Local embedding

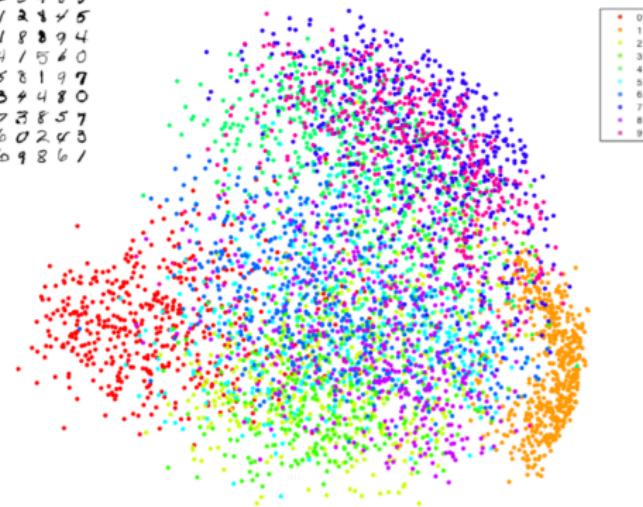
- t-SNE is an alternative dimensionality reduction algorithm.
- PCA tries to find a **global** structure
  - ▶ Low dimensional subspace
  - ▶ Can lead to local inconsistencies
    - ▶ Far away point can become nearest neighbors
- t-SNE tries to preserve local structure
  - ▶ Low dimensional neighborhood should be the same as original neighborhood.
- Unlike PCA almost only used for visualization
  - ▶ No easy way to embed new points

# tSNE 2 dimensions embedding for MNIST



# PCA 2 dimensions embedding for MNIST

```
3 6 8 1 7 9 6 6 9 1  
6 7 5 7 8 6 3 4 8 5  
2 1 7 9 7 1 2 4 4 5  
4 8 1 9 0 1 8 8 9 4  
7 6 1 8 4 1 5 6 0  
7 5 9 2 6 5 8 1 9 7  
1 2 2 2 2 3 4 4 8 0  
0 2 3 8 0 7 3 8 5 7  
0 1 4 6 4 6 0 2 4 3  
7 1 2 8 7 6 9 8 6 1
```



# Stochastic Neighbor Embedding (SNE)

SNE basic idea:

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

# Neighborhood Distributions

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

# Probabilities $P_{ij}$

The  $i \rightarrow j$  probability (should be familiar from A1Q2), is the probability that point  $\mathbf{x}_i$  chooses  $\mathbf{x}_j$  as its neighbor

$$P_{j|i} \equiv \frac{\exp(-\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}^{(i)} - \mathbf{x}^{(k)}\|^2 / 2\sigma_i^2)}$$

With  $P_{i|i} = 0$

- The parameter  $\sigma_i$  sets the size of the neighborhood
  - ▶ Very low  $\sigma_i$  - all the probability is in the nearest neighbor.
  - ▶ Very high  $\sigma_i$  - Uniform weights.
- Here we set  $\sigma_i$  **differently** for each data point
- Results depend heavily on  $\sigma_i$  - it defines the neighborhoods we are trying to preserve.
- Final distribution over pairs is symmetrized:  $P_{ij} = \frac{1}{2N}(P_{i|j} + P_{j|i})$ 
  - ▶ Pick  $i$  (or  $j$ ) uniformly and then "jump" to  $j$  ( $i$ ) according to  $P_{j|i}$  ( $P_{i|j}$ )

# Perplexity

- For each distribution  $P_{j|i}$  (depends on  $\sigma_i$ ) we define the perplexity
  - ▶  $\text{perp}(P_{j|i}) = 2^{H(P_{j|i})}$  where  $H(P) = -\sum_i P_i \log(P_i)$  is the entropy.
- If  $P$  is uniform over  $k$  elements - perplexity is  $k$ .
  - ▶ Smooth version of  $k$  in  $kNN$
  - ▶ Low perplexity = small  $\sigma^2$
  - ▶ High perplexity = large  $\sigma^2$
- Define the desired perplexity and set  $\sigma_i$  to get that (bisection method)
- Values between 5-50 usually work well
- Important parameter - different perplexity can capture different scales in the data
- If you're interested - try A1Q2 which uses a fixed perplexity instead (let me know how it worked)!

# Perplexity



[Pic credit: <https://distill.pub/2016/misread-tsne/>]

## SNE objective

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

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

- Optimize  $Q$  to be close to  $P$ 
  - Minimize KL-divergence
- The embeddings  $\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(N)} \in \mathbb{R}^d$  are the parameters we are optimizing.
  - How do you embed a new point? No embedding function!

# KL divergence

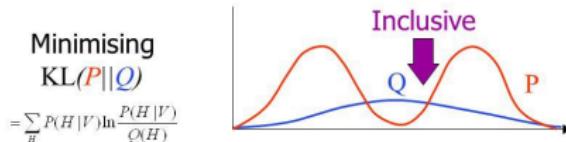
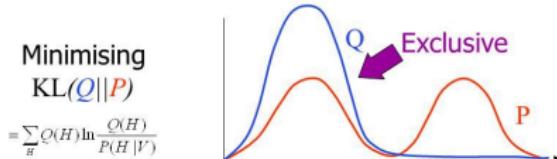
Measures distance between two distributions,  $P$  and  $Q$ :

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

- Not a metric function - not symmetric!
- Code theory intuition: If we are transmitting information that is distributed according to  $P$ , then the optimal (lossless) compression will need to send on average  $H(P)$  bits.
- What happens you expect  $P$  (and design your compression accordingly) but the actual distribution is  $Q$ ?
  - ▶ will send on average  $H(Q) + KL(Q||P)$
  - ▶  $KL(Q||P)$  is the "penalty" for using wrong distribution

# KL Properties

- $KL(Q||P) \geq 0$  and zero only when  $Q = P$  (a.s)
- $KL(Q||P)$  is a convex function.
- if  $P_{ij} = 0$  but  $Q_{ij} > 0$  then  $KL(Q||P) = \infty$



[Pic credit: <https://timvieira.github.io/blog/post/2014/10/06/kl-divergence-as-an-objective-function/>]

# SNE algorithm

- We have  $P$ , and are looking for  $\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(N)} \in \mathbb{R}^d$  such that the distribution  $Q$  we infer will minimize  $L(Q) = KL(P||Q)$  (notice Q on right, uncommon).
- Note that  $KL(P||Q) = \sum_{ij} P_{ij} \log \left( \frac{P_{ij}}{Q_{ij}} \right) = -\sum_{ij} P_{ij} \log (Q_{ij}) + const$
- Can show that  $\frac{\partial L}{\partial \mathbf{y}^{(i)}} = \sum_j (P_{ij} - Q_{ij})(\mathbf{y}^{(i)} - \mathbf{y}^{(j)})$
- Not a convex problem! No guarantees, can use multiple restarts.
- Main issue - 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.

# t-SNE

## t-Distributed Stochastic Neighbor Embedding

- Student-t Probability density  $p(x) \propto (1 + \frac{x^2}{v})^{-(v+1)/2}$ 
  - ▶ for  $v = 1$  we get  $p(x) \propto \frac{1}{1+x^2}$
- Probability goes to zero much slower than a Gaussian.
- Can show it is equivalent to averaging Gaussians with some prior over  $\sigma^2$
- We can now redefine  $Q_{ij}$  as

$$Q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

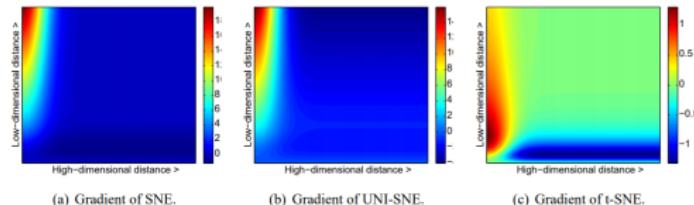
- We leave  $P_{ij}$  as is!

# t-SNE gradients

- Can show that the gradients of t-SNE objective are

$$\frac{\partial L}{\partial \mathbf{y}^{(i)}} = \sum_j (P_{ij} - Q_{ij})(\mathbf{y}^{(i)} - \mathbf{y}^{(j)})(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$$

- Compare to the SNE gradients:  $\frac{\partial L}{\partial \mathbf{y}^{(i)}} = \sum_j (P_{ij} - Q_{ij})(\mathbf{y}^{(i)} - \mathbf{y}^{(j)})$



- Both repulse close dissimilar points and attract far similar points, but the  $t-SNE$  has a smaller attraction term to solve crowding.

[Image credit: "Visualizing Data using t-SNE"]

# Algorithm

---

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

---

**Data:** data set  $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ ,

cost function parameters: perplexity  $Perp$ ,

optimization parameters: number of iterations  $T$ , learning rate  $\eta$ , momentum  $\alpha(t)$ .

**Result:** low-dimensional data representation  $\mathcal{Y}^{(T)} = \{y_1, y_2, \dots, y_n\}$ .

**begin**

    compute pairwise affinities  $p_{j|i}$  with perplexity  $Perp$  (using Equation 1)

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

    sample initial solution  $\mathcal{Y}^{(0)} = \{y_1, y_2, \dots, y_n\}$  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)

$$\text{set } \mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) (\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)})$$

**end**

**end**

---

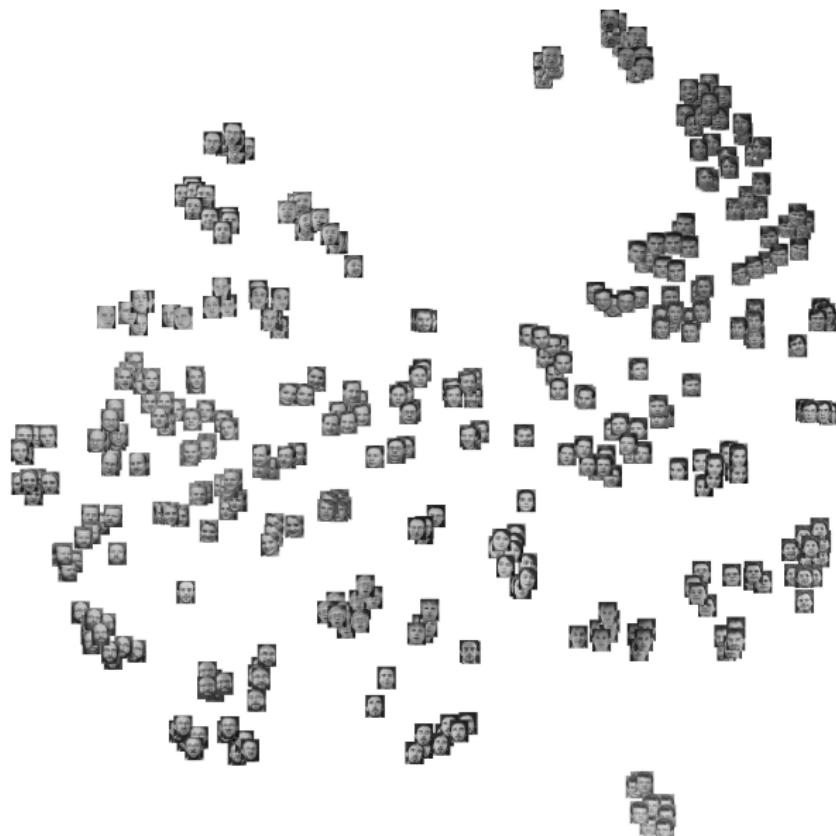
[Slide credit: "Visualizing Data using t-SNE"]

# CNN features example



[Image credit: <http://cs.stanford.edu/people/karpathy/cnnembed/>]

# CNN features example



[Image credit: <https://lvdmaaten.github.io/tsne/>]

## Recap

- t-SNE is a great way to visualize data
- Helps understand "black-box" algorithms like DNN.
- Reduced "crowding problem" with heavy tailed distribution.
- Non-convex optimization - solved by GD with momentum.
- Less suitable for SGD (think about the parameters), some alternative speedups exist ("Barnes-Hut t-SNE").
- Great extra resource: <https://distill.pub/2016/misread-tsne/>