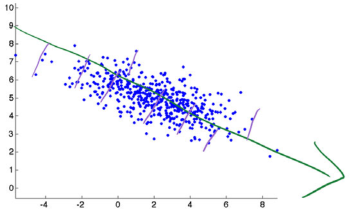
CS 189/289

Today's lecture outline

1. Finish PCA.
2. Non-linear dimensionality reduction (t-SNE)

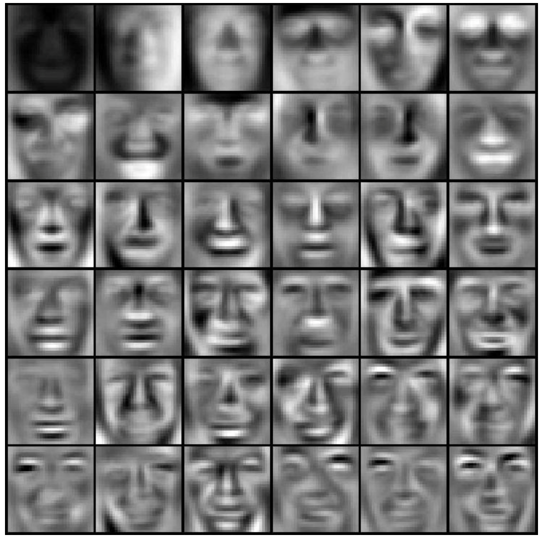
Recall: PCA



- Goal: find $k < d$ dimensions w most of the "information" in the original data, $X \in \mathbb{R}^{n \times d}$.
- Directions accounting for most variance and yielding lowest $\|X_{recon-k} - X\|_F$.

5K original faces, $x_j \in \mathbb{R}^{1024}$

1024 PCA basis vectors, $u_i \in \mathbb{R}^{1024}$ (from $X^T X = UDU^T$)



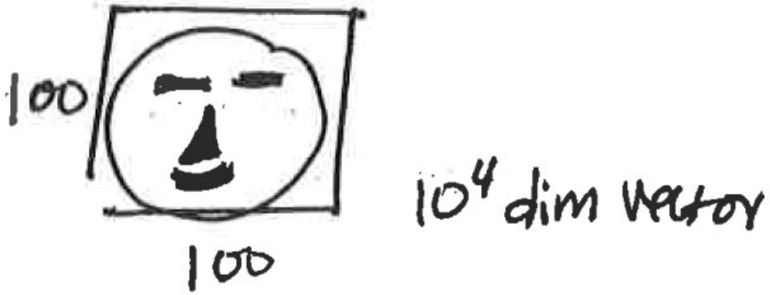
Reconstruction, $X_{recon-k} = XU_k U_k^T$

Linear algebra for PCA

$\bar{X} \in \mathbb{R}^{n \times d}$

- Need to compute the principal axes, $\boldsymbol{Q}$, of $\bar{X}^T\bar{X} = QDQ^T \in \mathbb{R}^{d \times d}$.
- This is an *eigendecomposition* of the matrix $\bar{X}^T\bar{X}$.

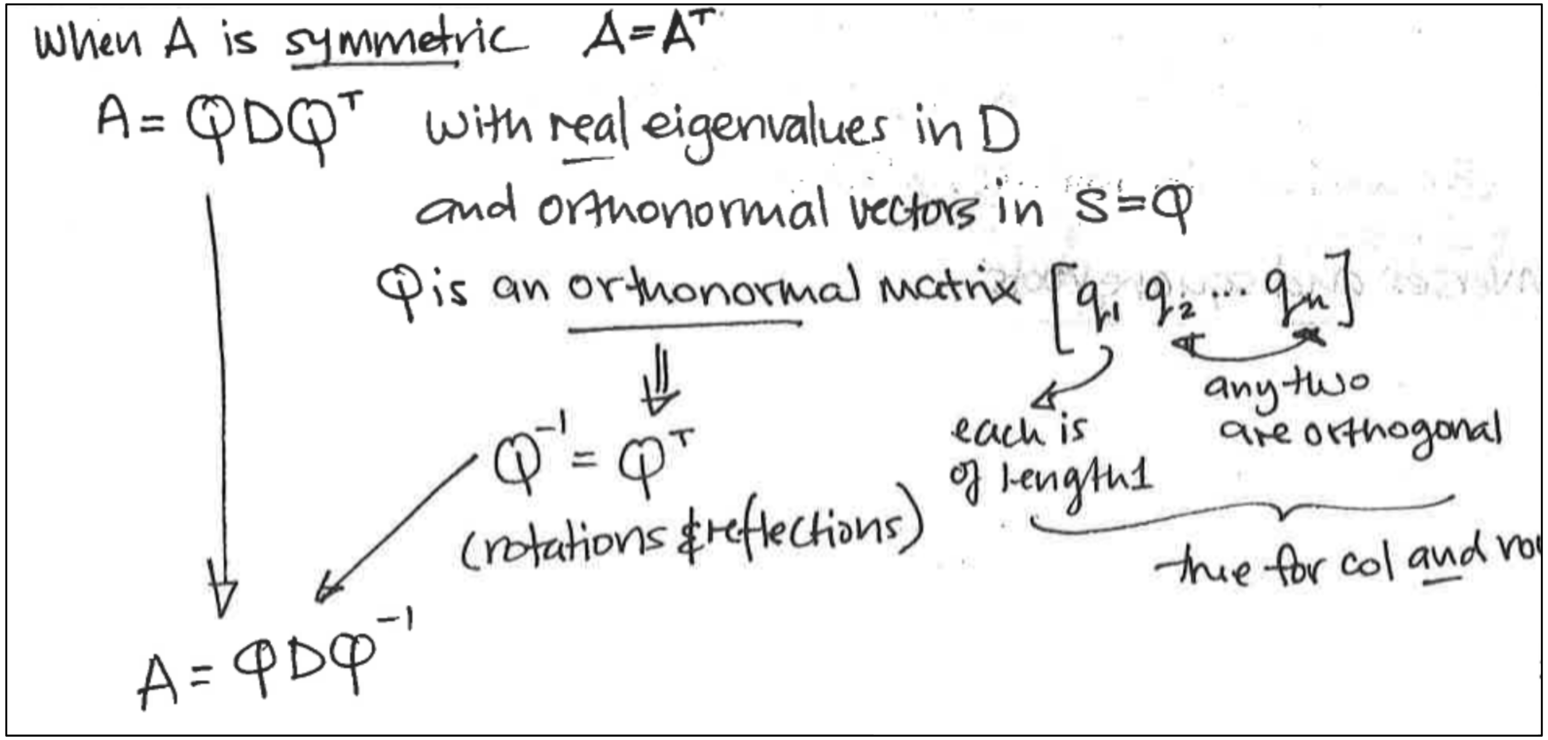
- Computing its eigendecomposition has time complexity $\boldsymbol{O(d^3)}$.
- What if $\boldsymbol{d \gg n}$? *e.g.*, images of $\mathbf{d = 100 \times 100 = 10^4}$ pixels, and $\boldsymbol{n = 1{,}000}$ images.
- Could we do something cheaper?
- Yes. Need to understand the SVD.



Linear algebra for PCA

mean centered data, $\bar{X} \in \mathbb{R}^{n \times d}$

Recall the spectral theorem from MVG lecture, which gives a *spectral (eigen) decomposition:.*



When A is symmetric $A=A^T$

$A=\Phi D \Phi^T$ with real eigenvalues in D

and orthonormal vectors in $S=\Phi$

Φ is an orthonormal matrix $[q_1\ q_2 \cdots q_n]$

each is of length 1

any two are orthogonal

true for col and ro

$\Downarrow$

$\Phi^{-1}=\Phi^T$

(rotations & reflections)

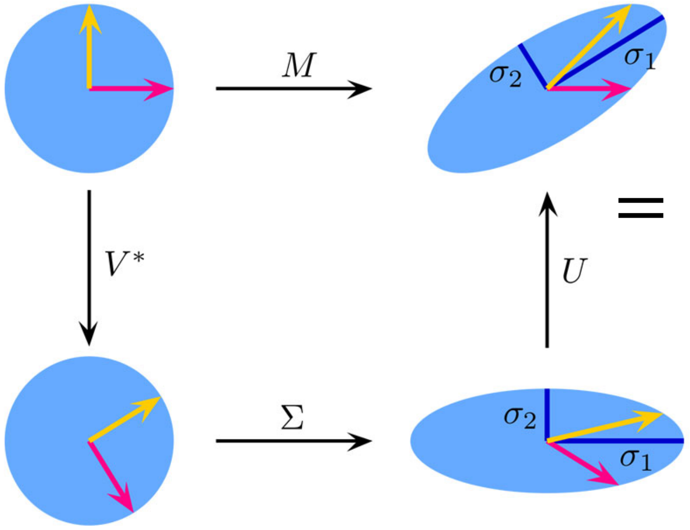
$A=\Phi D \Phi^{-1}$

- The covariance matrix for PCA, $\bar{X}^T\bar{X}$, is symmetric (and PSD).
- It turns out, there is a generalization of the spectral decomposition, for non-symmetric and non-square matrices, the SVD that will be helpful.

Singular Value Decomposition (SVD)

Can think of M as linear transformation broken down into three steps, by looking at its effect on the unit disc and the two canonical unit vectors e_1 and e_2:

1. **Left:** V^T rotates the disc and unit vectors.
2. **Bottom: Σ** stretches scales axes by $\sigma_i = \Sigma_{i,i}$ (singular values).
3. **Right: U** performs another rotation.

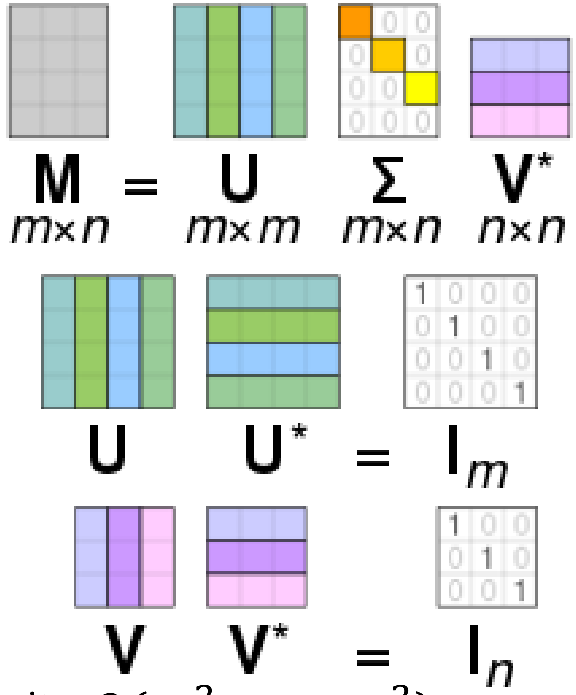


$$M = U \cdot \Sigma \cdot V^*$$

$$= M\begin{bmatrix} e_2 \\ e_1 \end{bmatrix} = U\Sigma V^* \begin{bmatrix} e_2 \\ e_1 \end{bmatrix}$$

https://en.wikipedia.org/wiki/Singular_value_decomposition

Can be applied to any matrix M.



$$\underset{m\times n}{\mathbf{M}} = \underset{m\times m}{\mathbf{U}}\ \underset{m\times n}{\mathbf{\Sigma}}\ \underset{n\times n}{\mathbf{V}^*}$$

$$\mathbf{U}\ \mathbf{U}^* = \mathbf{I}_m$$

$$\mathbf{V}\ \mathbf{V}^* = \mathbf{I}_n$$

$$\begin{aligned} Mv_1 &= \Sigma_{1,1}u_1 \\ Mv_2 &= \Sigma_{2,2}u_2 \\ &\cdots \\ Mv_r &= \Sigma_{r,r}u_r \end{aligned}$$

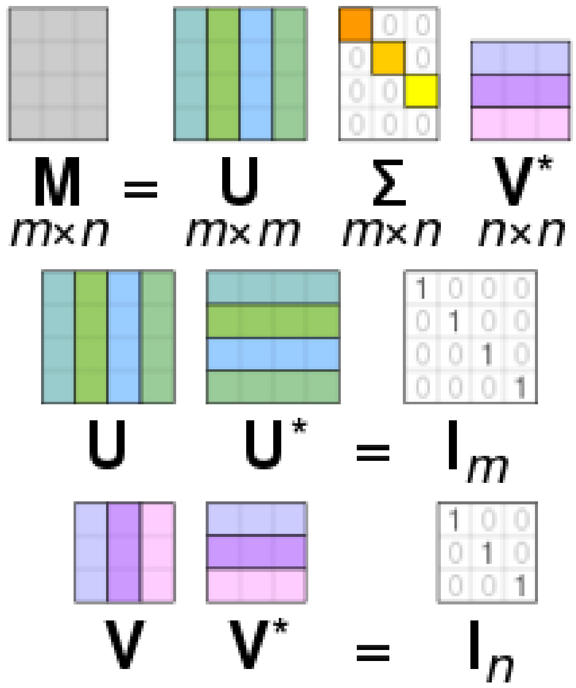
$$\begin{aligned} r &= rank(M) \\ &\le \min(m, n) \end{aligned}$$

- *Has time complexity $O(m^2n + mn^2)$.*
- *Σ is unique (if in descending order), but V and U are generally not: e.g. sign flips.*
- *(Eigendecomposition is unique if all eigenvalues are unique)*
- *If M is square+symmetric, yields the spectral decomposition.*

Singular Value Decomposition (SVD)

- Columns in $\boldsymbol{U}$ are the eigenvectors of $\boldsymbol{MM^T}$, called the *left singular vectors of M* ($MM^T = U\Sigma V^T V \Sigma^T U^T = U\Sigma^2 U^T$).
- Columns in $\boldsymbol{V}$ are the eigenvectors of $\boldsymbol{M^T M}$, called the *right singular vectors of M* ($M^T M = V\Sigma^T U^T U \Sigma V^T = V\Sigma^2 V^T$).
- Both spectral decompositions at once!
- Eigenvalues are the same, given by $\boldsymbol{\lambda_i} = \Sigma_{i,i}^2$ ($\boldsymbol{\Sigma_{\mathrm{i}}, \mathrm{i}}$ are the *singular values* of $\boldsymbol{M}$):
  - ➢ Since $\boldsymbol{v_i}$ is an eigenvector for $\boldsymbol{M^T M}$, it follows that $M^T M v_i = \lambda_i v_i$. It follows that…
  - ➢ ….$(MM^T)Mv_i = \lambda_i(Mv_i)$ thus $\boldsymbol{Mv_i}$ is an eigenvector for $\boldsymbol{MM^T}$ with eigenvalue $\boldsymbol{\lambda_i}$!

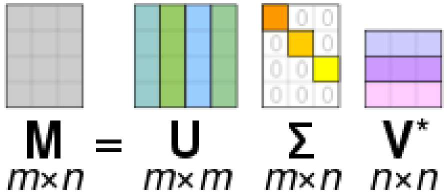
Can be applied to any matrix $\boldsymbol{M}$.



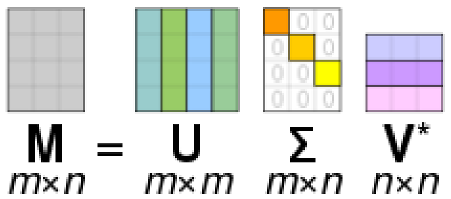
Singular Value Decomposition (SVD)

$X \in \mathbb{R}^{n \times d}$

- Recall this example with $d \gg n$,e.g. $d = 10^4$ pixels, $n = 1000$ images.
- How can we make use of what we just learned to do PCA faster than the eigendecomposition $O(d^3)$?
- Instead of spectral decomposition of $X^T X$…
- …directly use SVD of the data matrix: $SVD(X) = U\Sigma V^T$
- Because columns in V are the eigenvectors of $X^T X$.
- $\lambda_i = \Sigma_{i,i}^2$ are needed eigenvalues.
- SVD has time complexity $O(dn^2)$.



Singular Value Decomposition (SVD)



For PCA we want projections onto top $\boldsymbol{k}$ PCs.

- When we used a spectral decomposition, $\boldsymbol{X^TX = QDQ^T}$, we compute: $\boldsymbol{X_k} = \boldsymbol{X}\boldsymbol{Q_k} \in \mathbb{R}^{n\times k}$ ($\boldsymbol{Q}$ are eigvecs of $\boldsymbol{X^TX}$).
- When using the SVD of $\boldsymbol{X}$, we can instead get this from:
- $X_k = XV_{:,1:k} = U_{:,1:k}\Sigma_{1:k,1:k} \in \mathbb{R}^{n\times k}$ ("scores" in PCA basis).
- We don't need to compute covariance matrix, or do the projections, we just need $\boldsymbol{SVD(X)}$!

$$Xv_i = \sigma_i u_i$$

“Eckart Young theorem” 1936

- The SVD “k-reconstruction” produces the best $\boldsymbol{k}$-rank approximation by the matrix norm, $\|X - X_{\boldsymbol{recon-k}}\|_{\boldsymbol{F}}$.
- First proven by Schmidt (of Gram-Schmidt fame) in 1907 for Froebenius norm.
- Later rediscovered by Eckart & Young 1936, also generalized to other norms..
- <u>Thus, PCA provides the best low rank approximation to the data matrix.</u>

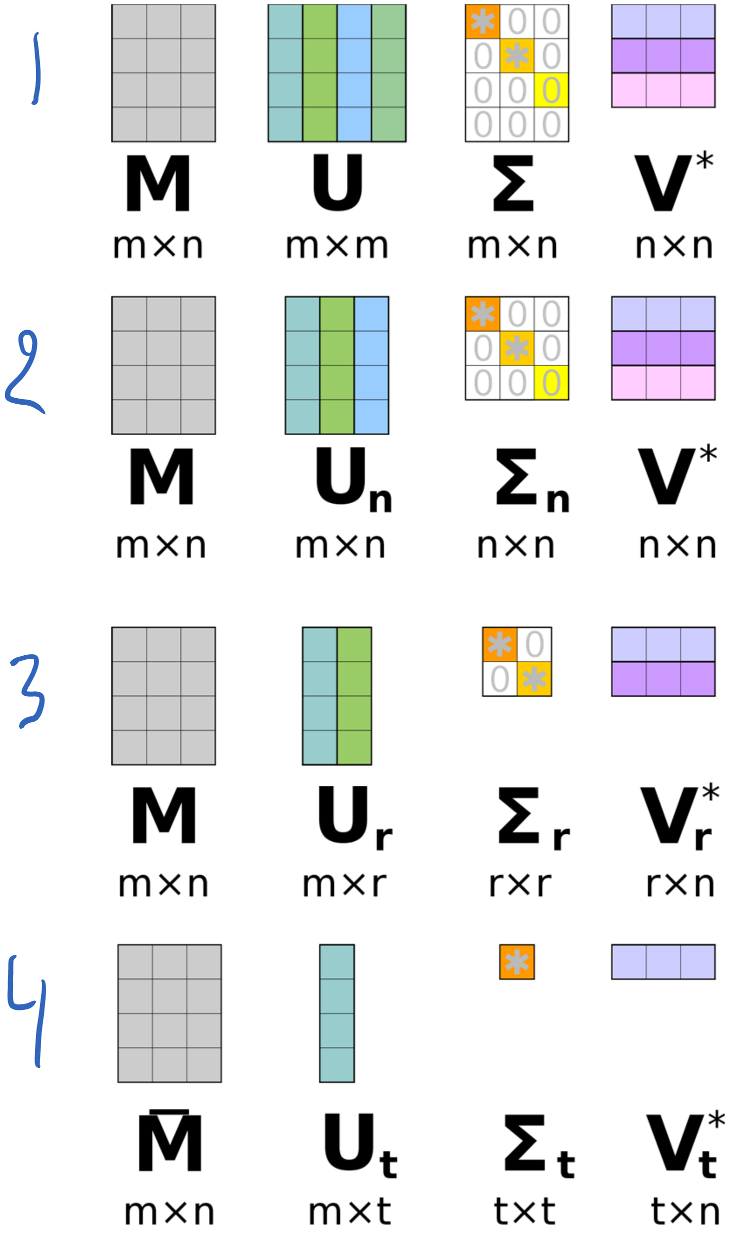
$$\|\mathsf{A}\|_F \equiv \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{i\,j}|^2}$$

https://en.wikipedia.org/wiki/Low-rank_approximation#Proof_of_Eckart%E2%80%93Young%E2%80%93Mirsky_theorem_(for_spectral_norm)

Practicalities: Reduced SVDs

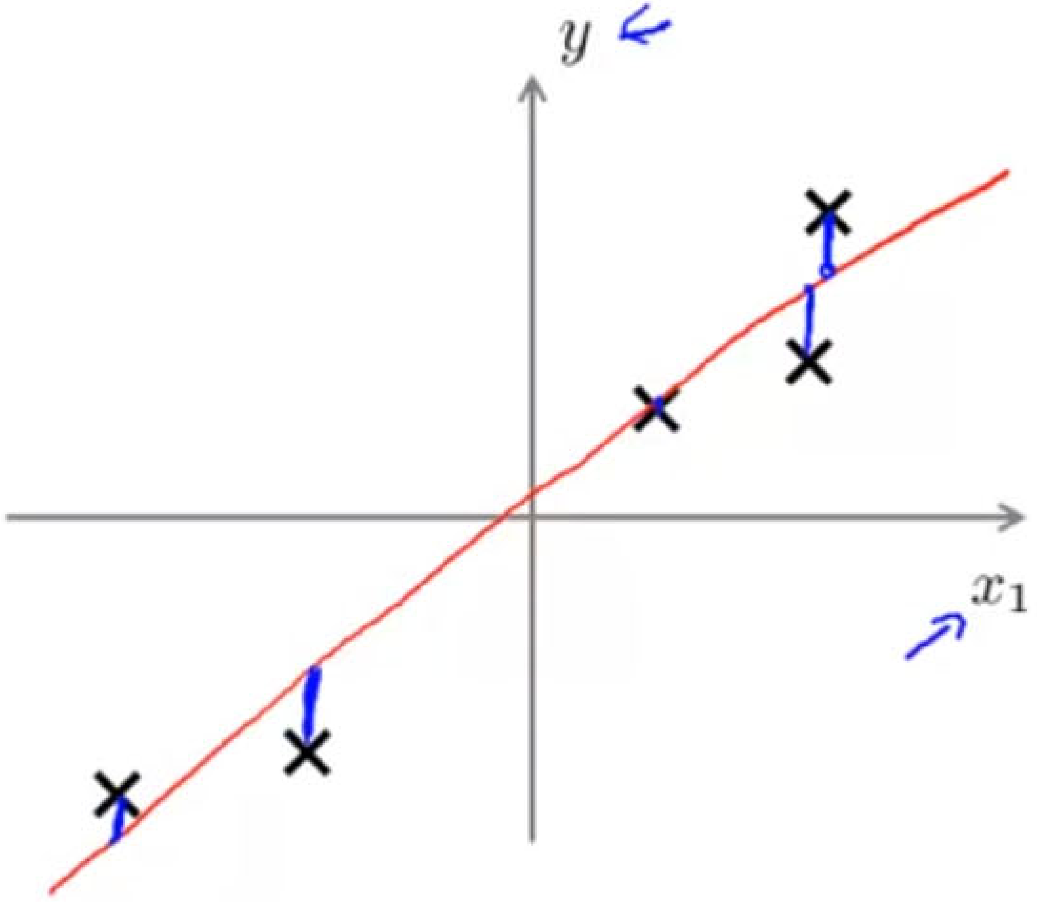
For PCA and other applications, don't need the entire SVD, and can make do with "trimmed down" versions:

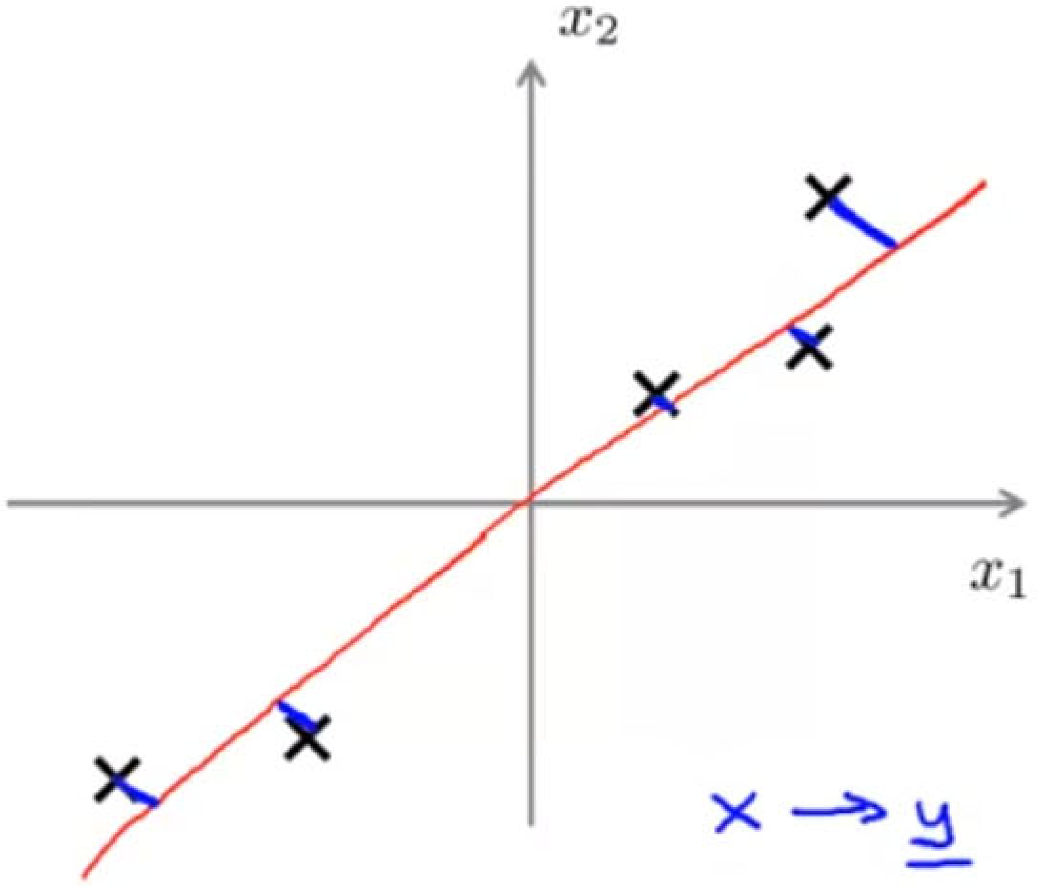
1. Full SVD
2. Thin SVD (remove columns of U not corresponding to rows of V*)
3. Compact SVD (remove vanishing singular values and corresponding columns/rows in U and V*),
4. Truncated SVD (keep only largest t singular values and corresponding columns/rows in U and V*)



https://en.wikipedia.org/wiki/Singular_value_decomposition

PCA is not linear regression





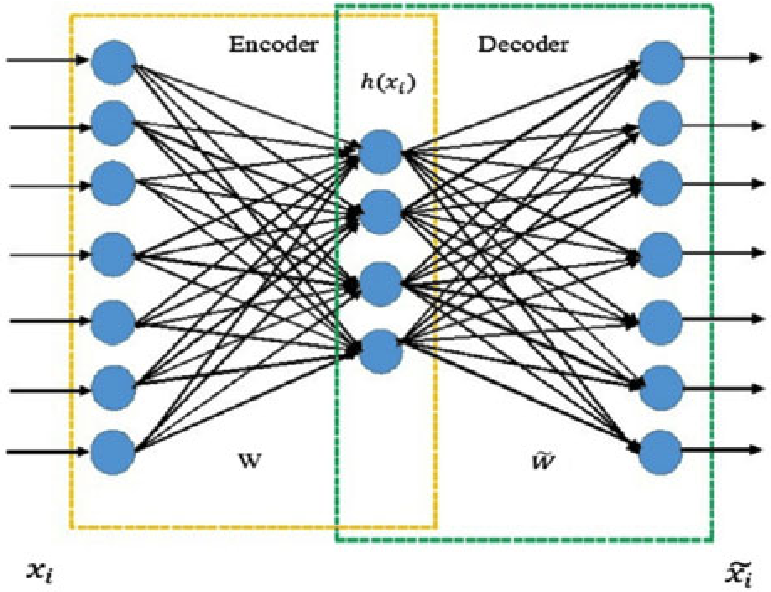
$x \rightarrow \underline{y}$

$x_1, x_2, \ldots, x_n$

Andrew Ng

PCA from neural networks!

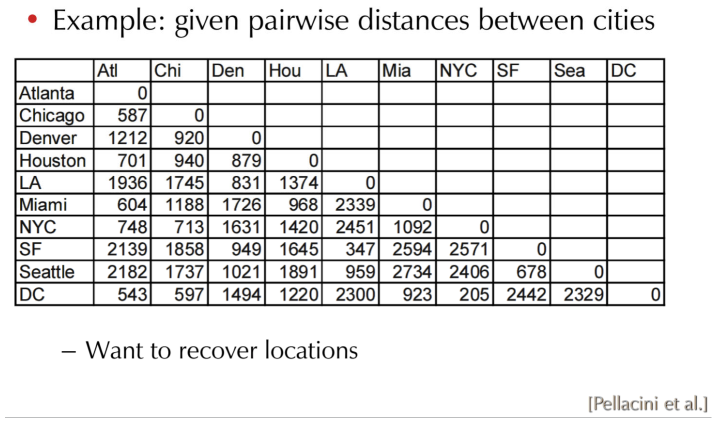
- Special kind of neural network: an *autoencoder*.
- Learned parameters with MLE recovers the same subspace as PC-k (one layer, all linear, k nodes).
- Can generalize by making non-linear transfers, and more layers, etc.



https://towardsdatascience.com/understanding-pca-autoencoders-algorithms-everyone-can-understand-28ee89b570e2

Can we do PCA if don’t have $\boldsymbol{X}$ explicitly?

- Suppose you’re given pairwise distances between $\boldsymbol{n}$ cities, $\boldsymbol{M} \in \mathbb{R}^{n \times n}$, and asked you to find a 2D representation?
- Think of $\boldsymbol{M} = \boldsymbol{X}\boldsymbol{X}^T \in \mathbb{R}^{n \times n}$ for some unobserved $\boldsymbol{X}$ (instead of $\boldsymbol{X}^T\boldsymbol{X}$).
- Now new basis is in $\boldsymbol{U}$ from: $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$.
- We just performed *Multidimensional Scaling* (MDS):
- Implicitly assumes some latent space $\boldsymbol{X}$ of unknown dimension for which the distance is an inner product distance, $\mathbf{d}(\mathbf{x}', \mathbf{x}) = x'x^T$.
- Could be non-linear distance function of actual $\boldsymbol{x}$ (*e.g.*, if latent space had a polynomial expansion).



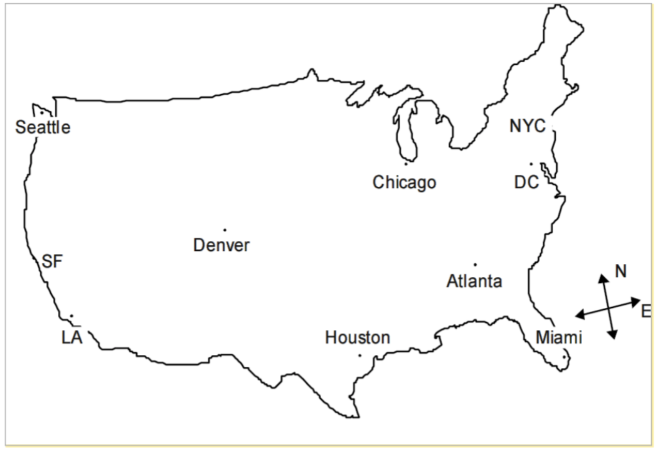
- Example: given pairwise distances between cities

	Atl	Chi	Den	Hou	LA	Mia	NYC	SF	Sea	DC
Atlanta	0									
Chicago	587	0								
Denver	1212	920	0							
Houston	701	940	879	0						
LA	1936	1745	831	1374	0					
Miami	604	1188	1726	968	2339	0				
NYC	748	713	1631	1420	2451	1092	0			
SF	2139	1858	949	1645	347	2594	2571	0		
Seattle	2182	1737	1021	1891	959	2734	2406	678	0	
DC	543	597	1494	1220	2300	923	205	2442	2329	0

  - Want to recover locations

[Pellacini et al.]

- Result ($k = 2$):



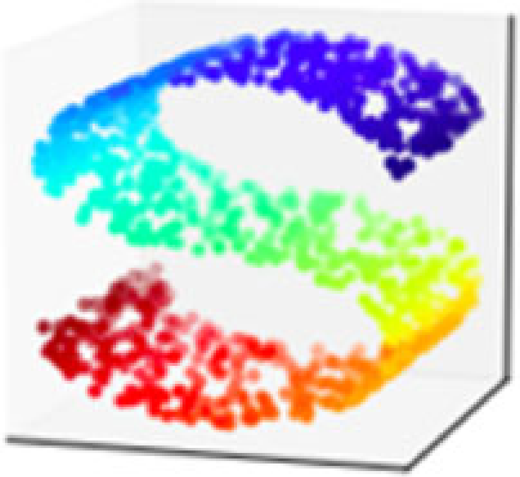
https://www.cs.princeton.edu/courses/archive/fall13/cos323/notes/cos323_f13_lecture11_dim_red.pdf

CS 189/289

Today’s lecture outline

1. Finish PCA.
2. Non-linear dimensionality reduction (t-SNE)

Recall the “swiss roll” type of data from last class



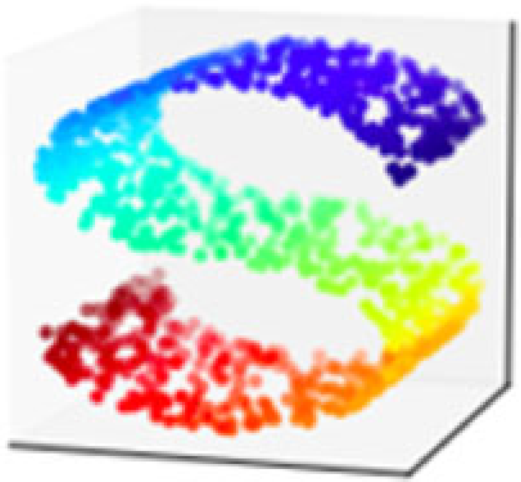
What would happen if we applied PCA with two dimensions to these data?

Ambient dimension=3
Manifold dimensions=2

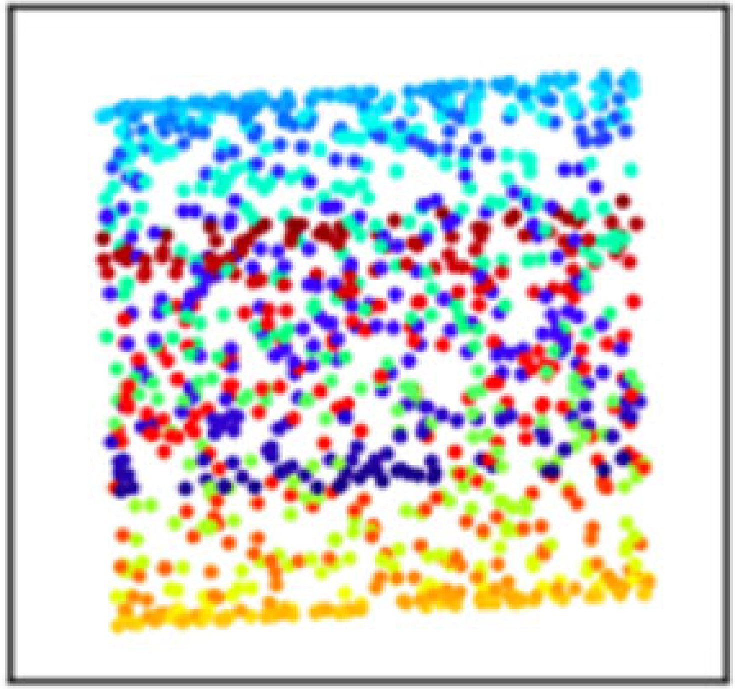
Hint: PCA is a *linear* transformation,
$X_k' = XU_k$

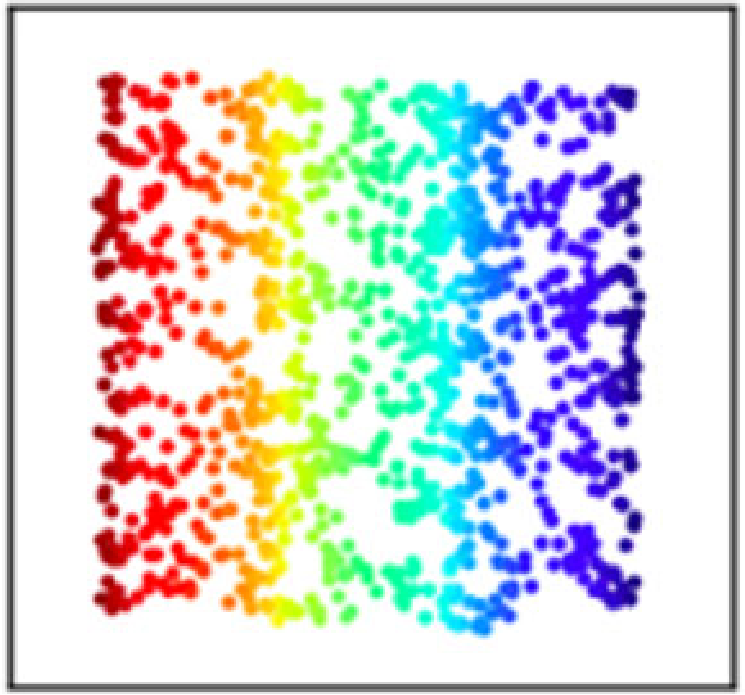
https://jhui.github.io/2017/01/15/Machine-learning-Multi-dimensional-scaling-and-visualization/

Recall the “swiss roll” type of data from last class



PCA projection

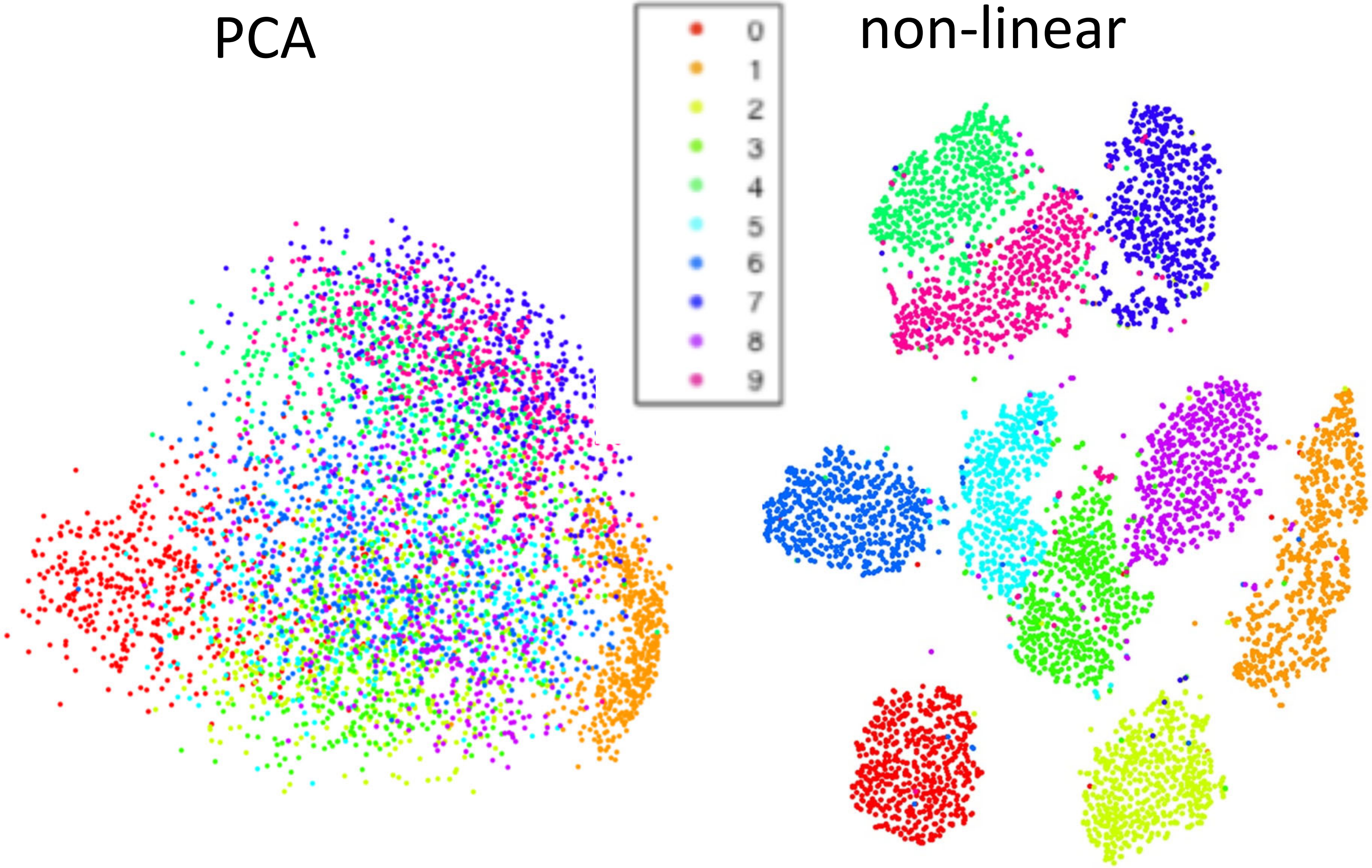




- This is not what we want!
- These data require a non-linear mapping.

https://jhui.github.io/2017/01/15/Machine-learning-Multi-dimensional-scaling-and-visualization/

PCA vs non-linear embedding on MNIST



PCA

non-linear

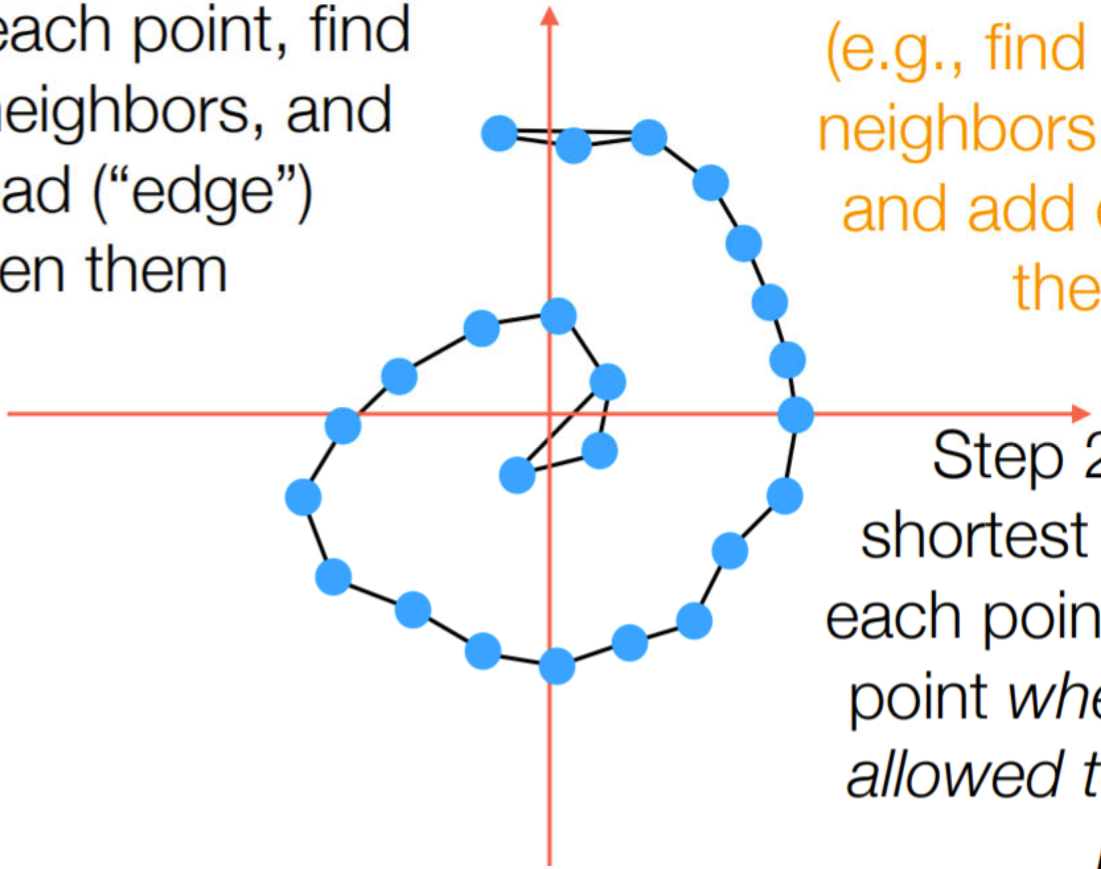
Neighborhood embeddings

Explicitly try to find a low dimensional embedding that is good at preserving neighborhoods of original points in ambient space.

1. NE: *neighbor embedding (e.g. IsoMap* Tenenbaum *et al* 2000 *)*
2. SNE: *stochastic* NE (Hinton & Roweis 2002)
3. t-SNE: *t-distributed* SNE (Van der Maaten & Hinton 2008)

Neighborhood Embedding (NE) aka *Isomap*

Step 1: For each point, find its nearest neighbors, and build a road (“edge”) between them



(e.g., find closest 2 neighbors per point and add edges to them)

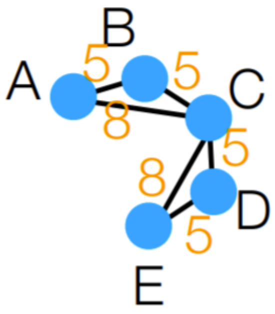
Step 2: Compute shortest distance from each point to every other point *where you’re only allowed to travel on the roads*

Step 3: Given this set of pairwise distances, put them in a matrix, $\boldsymbol{M}$, and perform MDS.

https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

Example of *Isomap*

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

M matrix in MDS

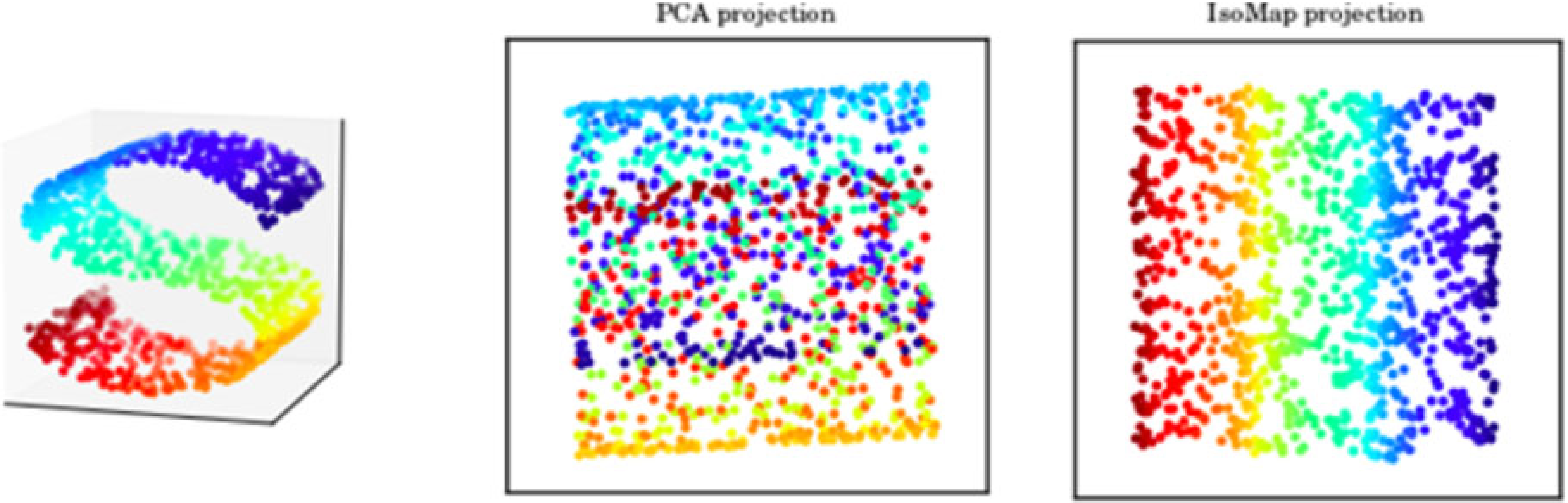
Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Shortest distances between every point to every other point *where we are only allowed to travel along the roads*

	A	B	C	D	E
A	0	5	8	13	16
B	5	0	5	10	13
C	8	5	0	5	8
D	13	10	5	0	5
E	16	13	8	5	0

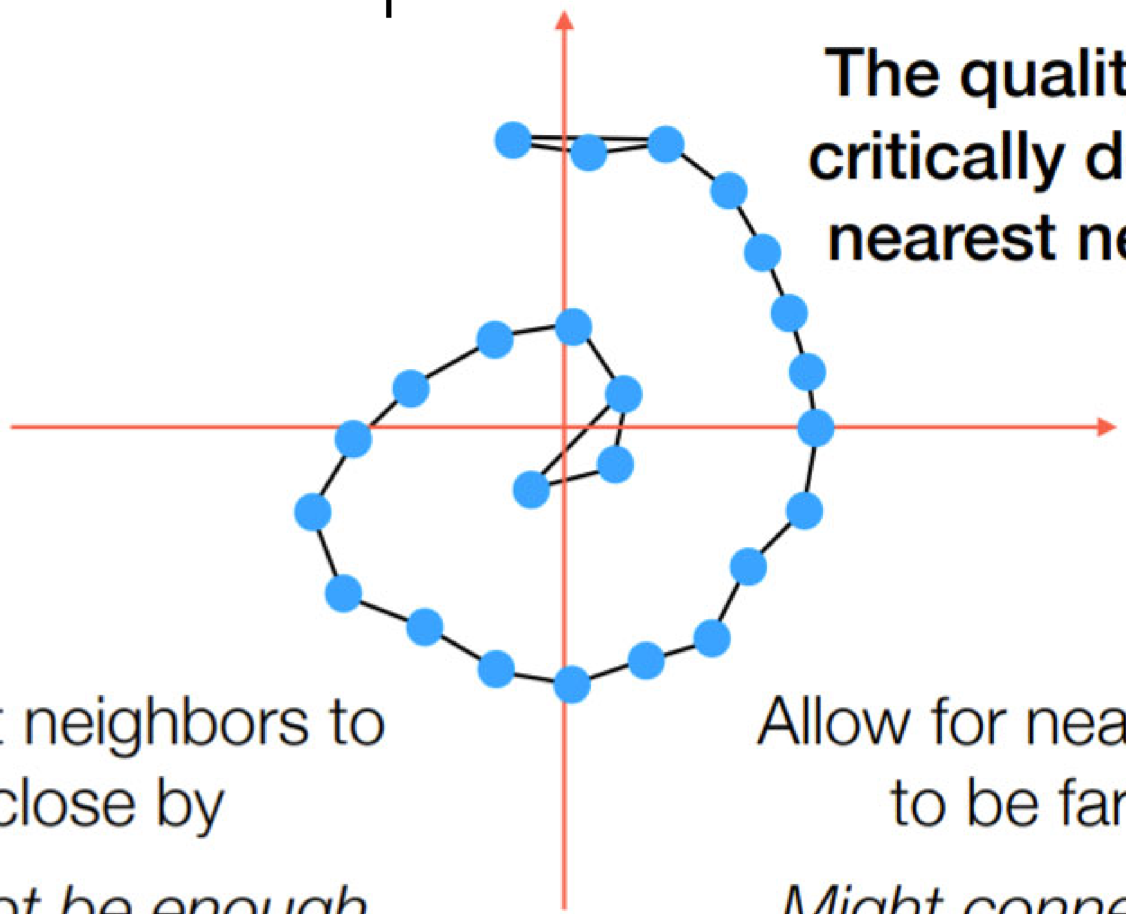
https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

IsoMap on Swiss Roll type example



https://jhui.github.io/2017/01/15/Machine-learning-Multi-dimensional-scaling-and-visualization/

Comments on Isomap



The quality of the result critically depends on the nearest neighbor graph

Ask for nearest neighbors to be really close by

There might not be enough edges

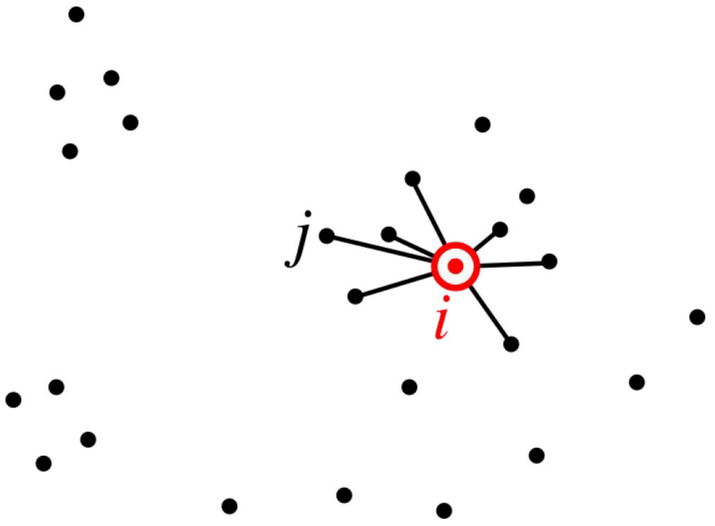
Allow for nearest neighbors to be farther away

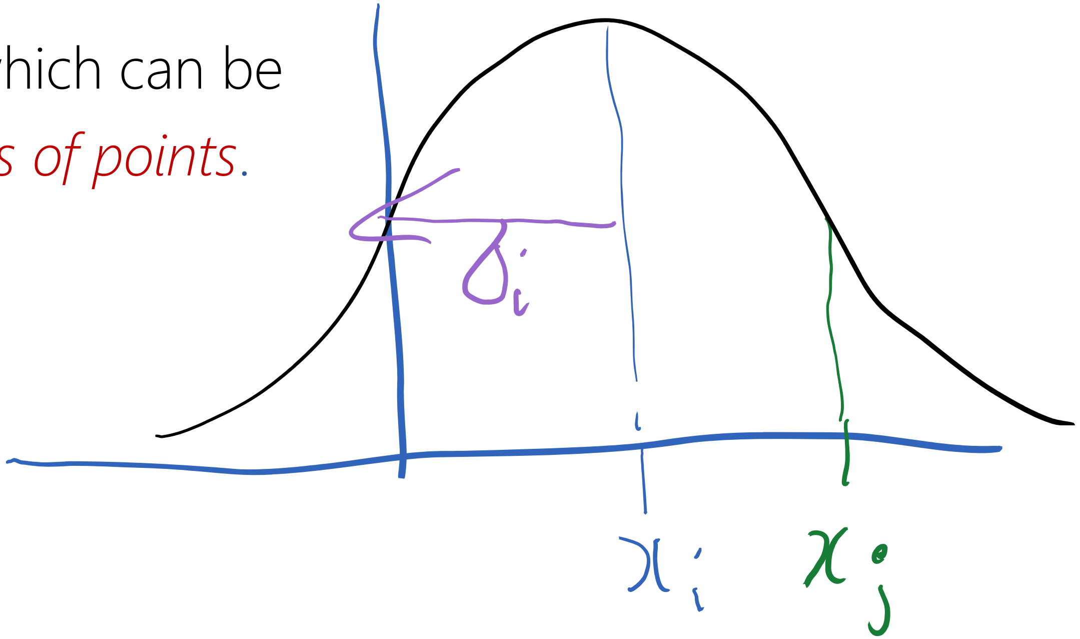
Might connect points that shouldn't be connected

https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

From NE to *stochastic NE* (SNE)

- Make the event of two samples being neighbors a random variable:
- The probability that x_i "chooses" x_j as its neighbor ($x \in \mathbb{R}^d$) is given by $P_{j \leftarrow i} = \frac{exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{k \neq i} exp\left(-\|x_i - x_k\|^2/2\sigma_i^2\right)}$, and $P_{j \leftarrow j} = 0$
- Smells like a Gaussian, but normalized so that $1 = \sum_j P_{j \leftarrow i}$.
- Symmetrize & normalize, $P_{ij} = P_{ji} = \frac{1}{2n}\left(P_{j \leftarrow i} + P_{i \leftarrow j}\right)$ which can be interpreted *as probability to pick this pair out of all pairs of points*.
- Set σ_i^2 adaptively such that entropy of $P_{: \leftarrow i}$ is constant, $\sum_{j \neq i} P_{j \leftarrow i} \log P_{j \leftarrow i}$.



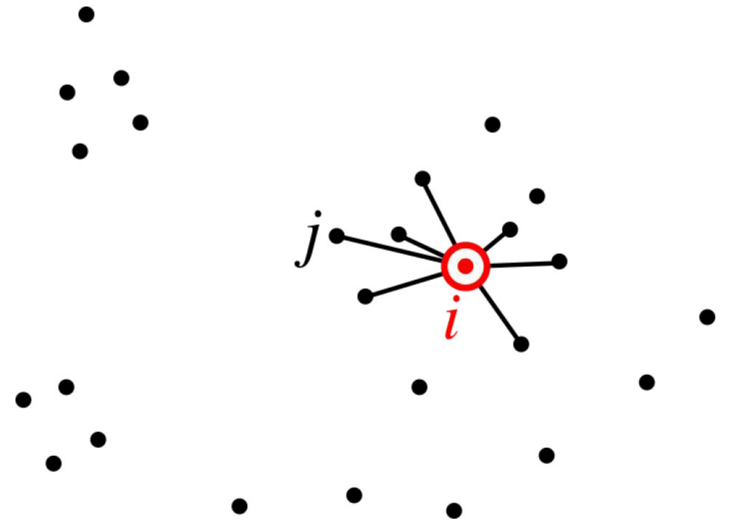


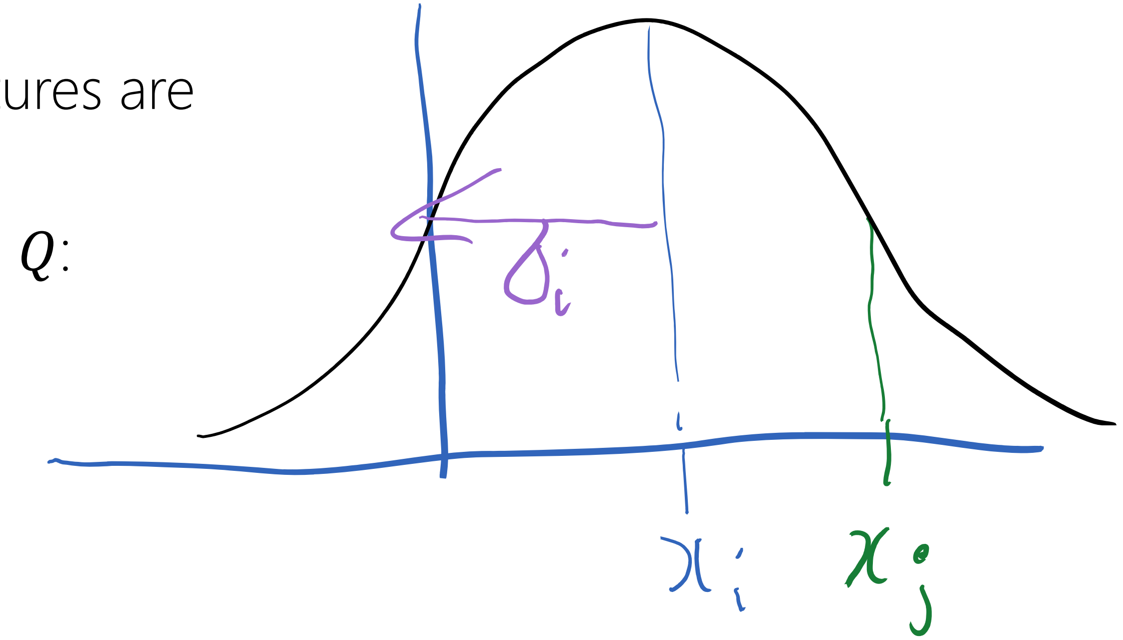
From NE to *stochastic NE* (SNE)

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

- From original data, $X \in \mathbb{R}^{n\times d}$, we have defined stochastic neighborhoods with probability distribution, $P = \{P_{ij}\}$.
- Now <u>posit low-dimensional representations</u>, $Y \in \mathbb{R}^{n\times k}$, and define stochastic neighborhoods for them, $Q = \{Q_{ij}\}$,
- $Q_{ji} = \frac{exp\left(-\|y_i - y_j\|^2\right)}{\sum_{l\neq k} exp(-\|y_l - y_k\|^2)}$ (setting $\sigma^2 = \frac{1}{2}$)
- Goal: find Y such that stochastic neighborhood structures are preserved ($Q \approx P$).
- Solution: minimize the KL-divergence between P and Q:

$$\hat{Y} = \underset{Y}{\text{argmin}}\, KL(P||Q) = \sum_{i,j} P_{ij}\, log\frac{P_{ij}}{Q_{ij}}.$$





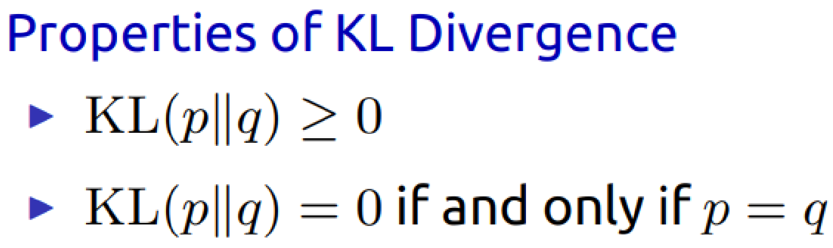
Aside: Kullback-Leibler (KL) Divergence

- Also called *Relative Entropy*.
- Measures how much one distribution diverges from another.
- For discrete probability distributions, $\boldsymbol{P}$ and $\boldsymbol{Q}$, it is defined as:

$$D_{KL}(P||Q) = \sum_x P(x) \ln \frac{P(x)}{Q(x)}$$

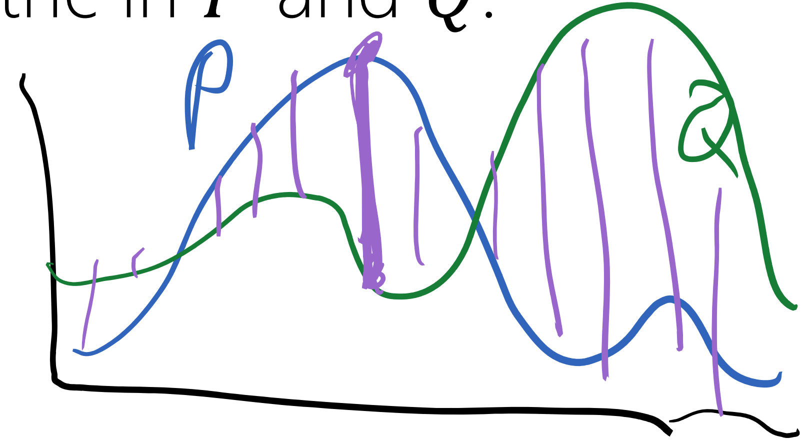
- Not a true distance metric because not symmetric in $\boldsymbol{P}$ and $\boldsymbol{Q}$:

$$D_{KL}(P||Q) \neq D_{KL}(Q||P)$$



Properties of KL Divergence

- $\mathrm{KL}(p||q) \geq 0$
- $\mathrm{KL}(p||q) = 0$ if and only if $p = q$



https://www.cs.ox.ac.uk/people/varun.kanade/teaching/ML-MT2016/slides/slides03.pdf

Aside: Kullback-Leibler (KL) Divergence

$$D_{KL}(P||Q) = \sum_{x} P(x) log \frac{P(x)}{Q(x)}$$

$$= E_{P(x)}\left[log \frac{1}{Q(X)}\right] - E_{P(x)}\left[log \frac{1}{P(X)}\right]$$

Aside: Kullback-Leibler (KL) Divergence

$$
\begin{aligned}
D_{KL}(P||Q) &= \sum_{x} P(x) log \frac{P(x)}{Q(x)} \\
&= E_{P(x)}\left[log \frac{1}{Q(X)}\right] - E_{P(x)}\left[log \frac{1}{P(X)}\right] \\
&= \underbrace{H(P,Q)}_{\text{cross-entropy}} - \underbrace{H(P)}_{\text{entropy}}
\end{aligned}
$$

- Consider data, $\boldsymbol{D}$ where $x_{\mathrm{i}} \sim \hat{p}_{data}$ and a model with params $\boldsymbol{\theta}$, $p(x|\theta)$.
- If minimizing the KL divergence (instead of MLE), $argmin_{\theta} D_{KL}(\hat{p}_{data}||p(x|\theta)) =)) =$

Aside: Kullback-Leibler (KL) Divergence

$$D_{KL}(P||Q) = \sum_{x} P(x) log \frac{P(x)}{Q(x)}$$

$$= E_{P(x)}\left[log \frac{1}{Q(X)}\right] - E_{P(x)}\left[log \frac{1}{P(X)}\right]$$

$$= \underbrace{H(P,Q)}_{\text{cross-entropy}} - \underbrace{H(P)}_{\text{entropy}}$$

- Consider data, D where $x_{\mathrm{i}} \sim \hat{p}_{data}$ and a model with params θ, $p(x|\theta)$.
- If minimizing the KL divergence (instead of likelihood for MLE),

$$argmin_{\theta} D_{KL}(\hat{p}_{data}||p(x|\theta)) =)) = argmin_{\theta}\, H(\hat{p}_{data}, p(x|\theta)) + \overset{\text{no dependence on } \Theta}{H(\hat{p}_{data})}$$

$$= argmax \underbrace{E_{\hat{p}_{data}}[\log p(x|\theta))]}_{\text{negative cross-entropy}}$$

Aside: Kullback-Leibler (KL) Divergence

$$D_{KL}(P||Q) = \sum_x P(x) log \frac{P(x)}{Q(x)}$$

$$= E_{P(x)}\left[log\frac{1}{Q(X)}\right] - E_{P(x)}\left[log\frac{1}{P(X)}\right]$$

$$= \underbrace{H(P,Q)}_{\text{cross-entropy}} - \underbrace{H(P)}_{\text{entropy}}$$

MLE problem

- Consider data, D where $x_i \sim \hat{p}_{data}$ and a model with params θ, $p(x|\theta)$.
- If minimizing the KL divergence (instead of MLE),

$$argmin_\theta D_{KL}(\hat{p}_{data}||p(x|\theta)) =)) = argmin_\theta\, H(\hat{p}_{data}, p(x|\theta)) + H(\hat{p}_{data})$$

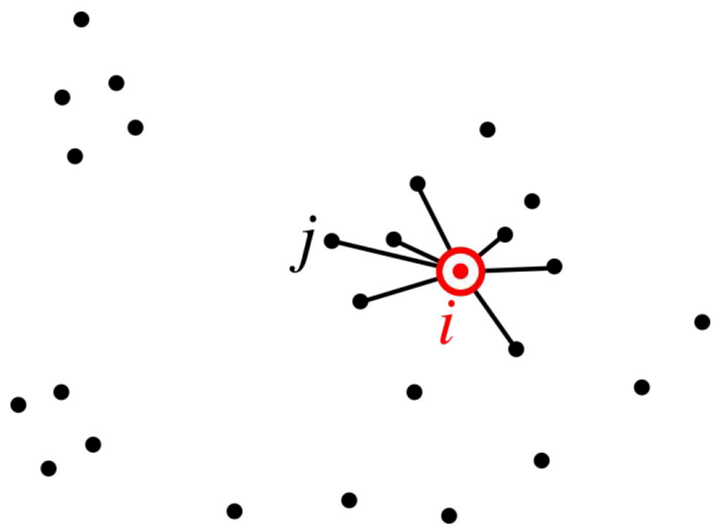
$$= argmax E_{\hat{p}_{data}}[\log p(x|\theta)] = \boxed{argmax \sum_i^N \log p(x_i|\theta)}.$$

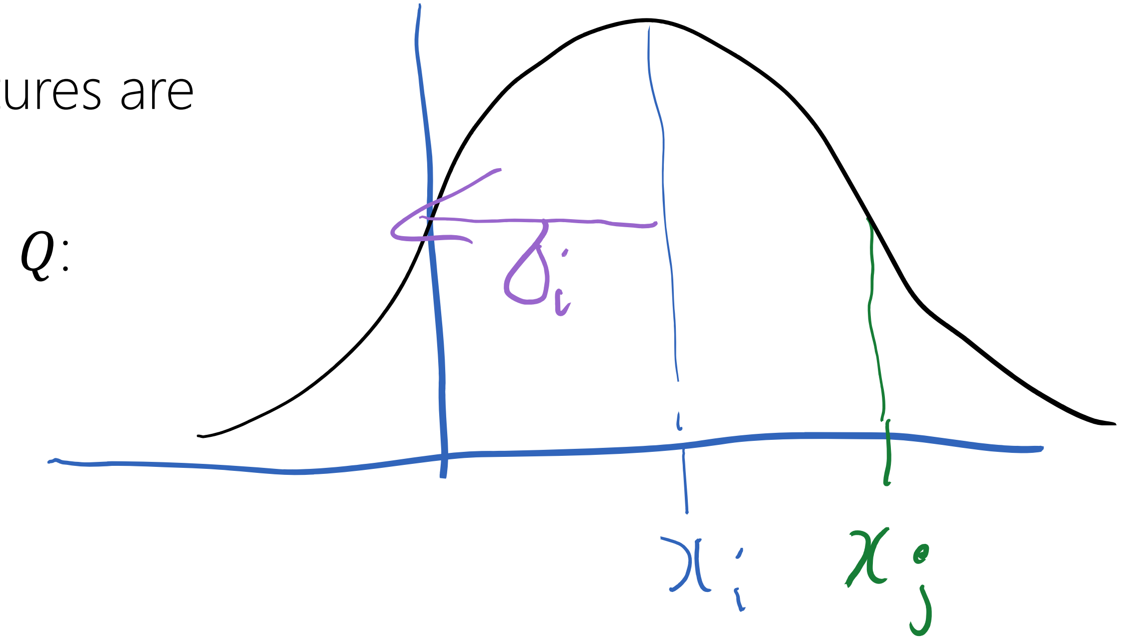
From NE to *stochastic NE* (SNE)

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

- From original data, $X \in \mathbb{R}^{n\times d}$, we have defined stochastic neighborhoods with probability distribution, $P = \{P_{ij}\}$.
- Now posit low-dimensional representations, $Y \in \mathbb{R}^{n\times k}$, and define stochastic neighborhoods for them, $Q = \{Q_{ij}\}$,
- $Q_{ji} = \frac{exp\left(-\|y_i - y_j\|^2\right)}{\sum_{l\neq k} exp(-\|y_l - y_k\|^2)}$ (setting $\sigma^2 = \frac{1}{2}$)
- Goal: find Y such that stochastic neighborhood structures are preserved ($Q \approx P$).
- Solution: minimize the KL-divergence between P and Q:

- $\hat{Y} = \underset{Y}{\text{argmin}}\, KL(P||Q) = \sum_{i,j} P_{ij}\, log \frac{P_{ij}}{Q_{ij}}$.





SNE: miminizing the loss function

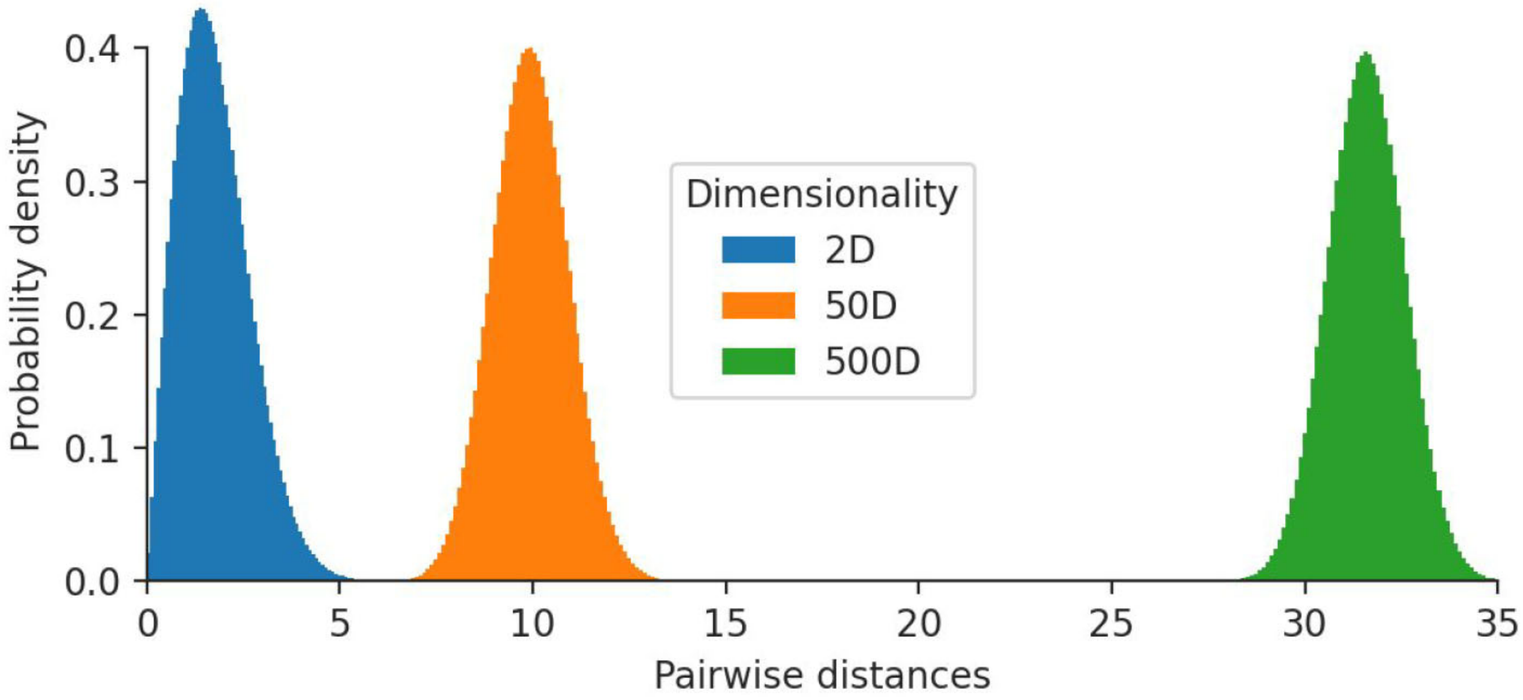
$$\hat{Y} = \underset{Y}{\operatorname{argmin}}\, KL(P||Q) = \sum_{i,j} P_{ij}\, log\, \frac{P_{ij}}{Q_{ij}}.$$

- Use gradient descent to find embedded points, $\{y_i\}$
- Can show that $\frac{\partial Loss}{\partial y_i} = \sum_j (P_{ij} - Q_{ij})(y_i - y_j)$.
- Not a convex optimization problem, so there are many local minima.

Problem with SNE

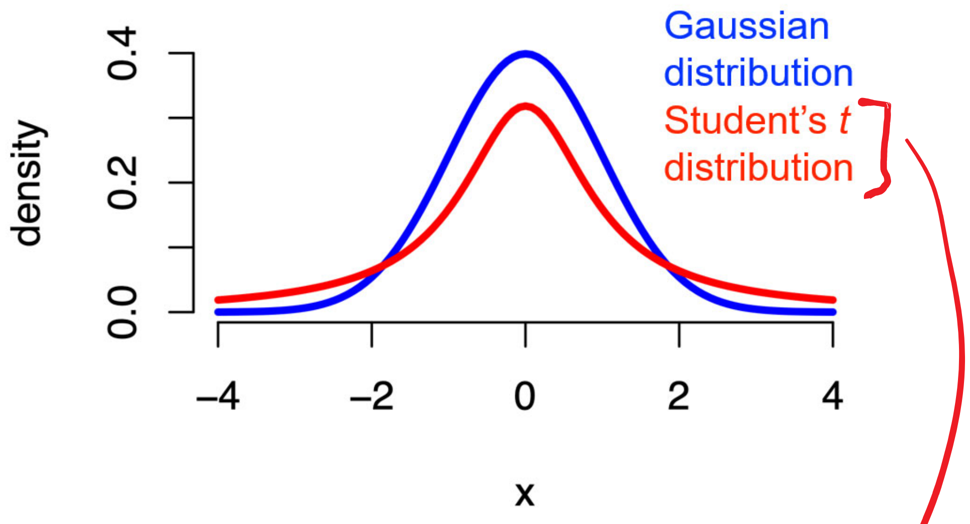
When we try to maintain the same neighborhood probabilities between the low and high dimensional spaces, we end up "crowding" the points in the lower dimensional space.

Pairwise distances between points in a standard Gaussian:



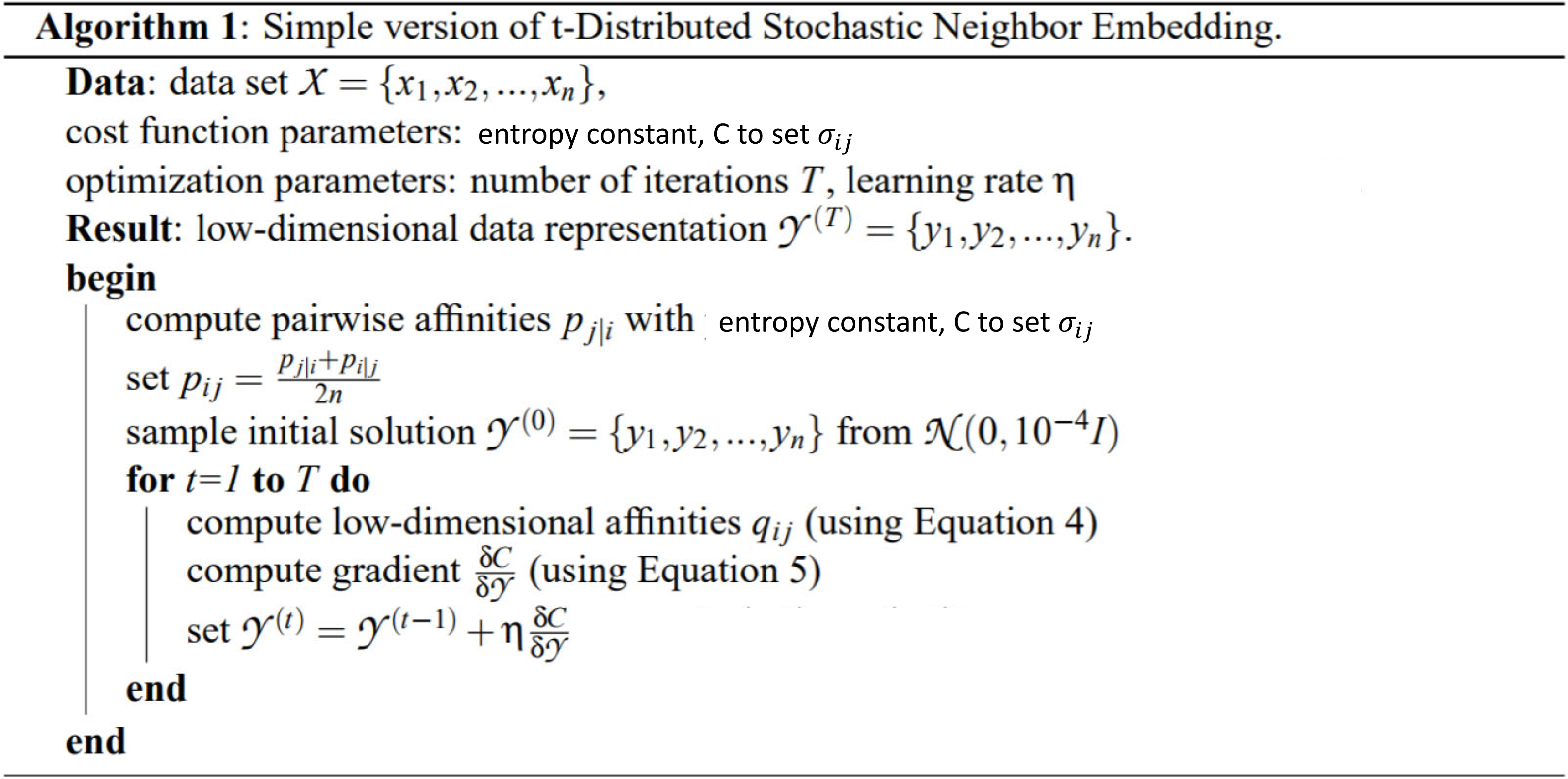
Fixing SNE with t-SNE

- Change the distribution in the embedding space, Q_{ij} to have a heavier-tailed distribution, a t-distribution.
- SNE used $Q_{j\leftarrow i} = \frac{exp\left(-\|y_i - y_j\|^2\right)}{\sum_{i\neq k} exp(-\|y_i - y_k\|^2)}$
- t-SNE uses $Q_{j\leftarrow i} = \frac{\left(1+\|y_i - y_j\|^2\right)^{-1}}{\sum_{i\neq k}(1+\|y_i - y_k\|^2)^{-1}}$
- Everything else remains the same as in SNE.



$p(x) \propto (1 + \frac{x^2}{\nu})^{-(\nu+1)/2}$

for $\nu = 1$ we get $p(x) \propto \frac{1}{1+x^2}$

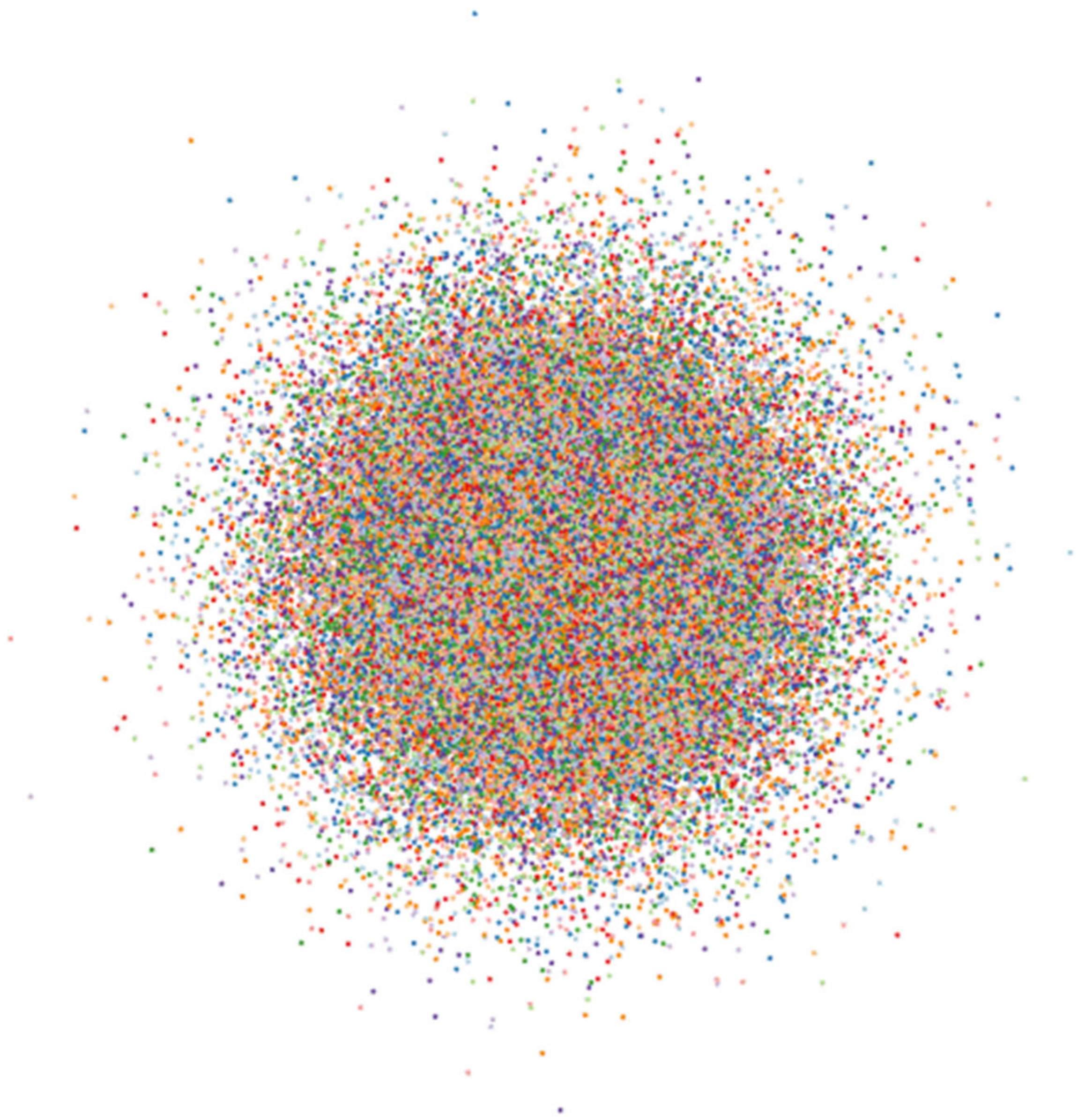


Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

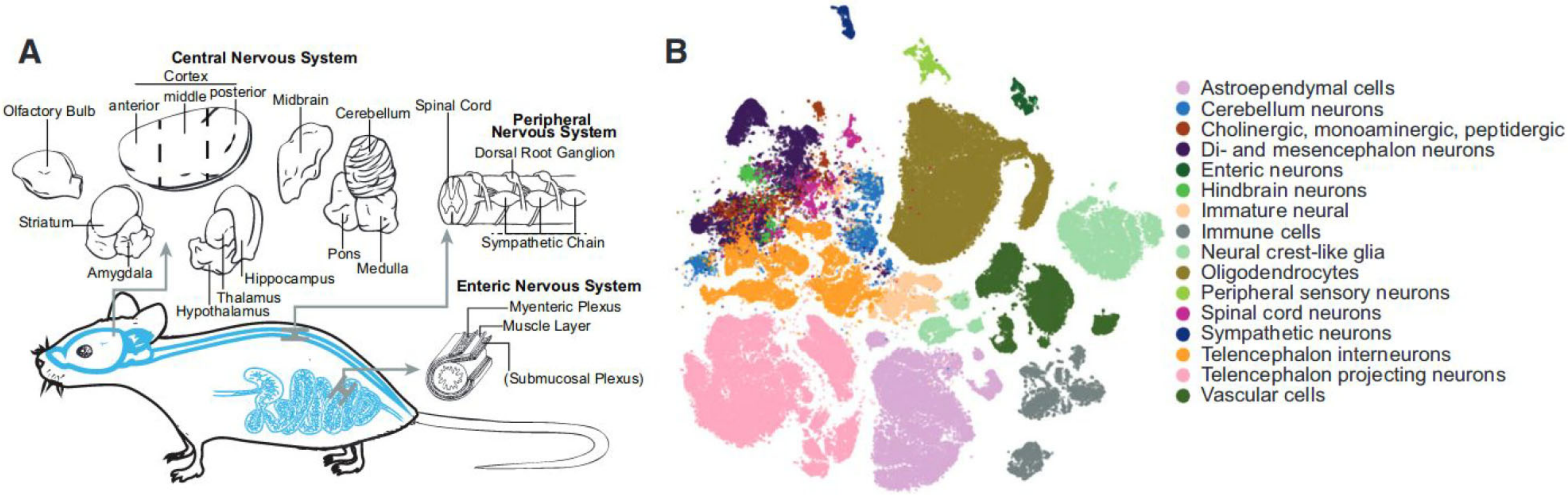
```
Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: entropy constant, C to set $\sigma_{ij}$
optimization parameters: number of iterations $T$, learning rate η
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
    compute pairwise affinities $p_{j|i}$ with entropy constant, C to set $\sigma_{ij}$
    set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
    sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ 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 $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}}$
    end
end
```

t-SNE on MNIST

(each color labels one of the 10 digits)

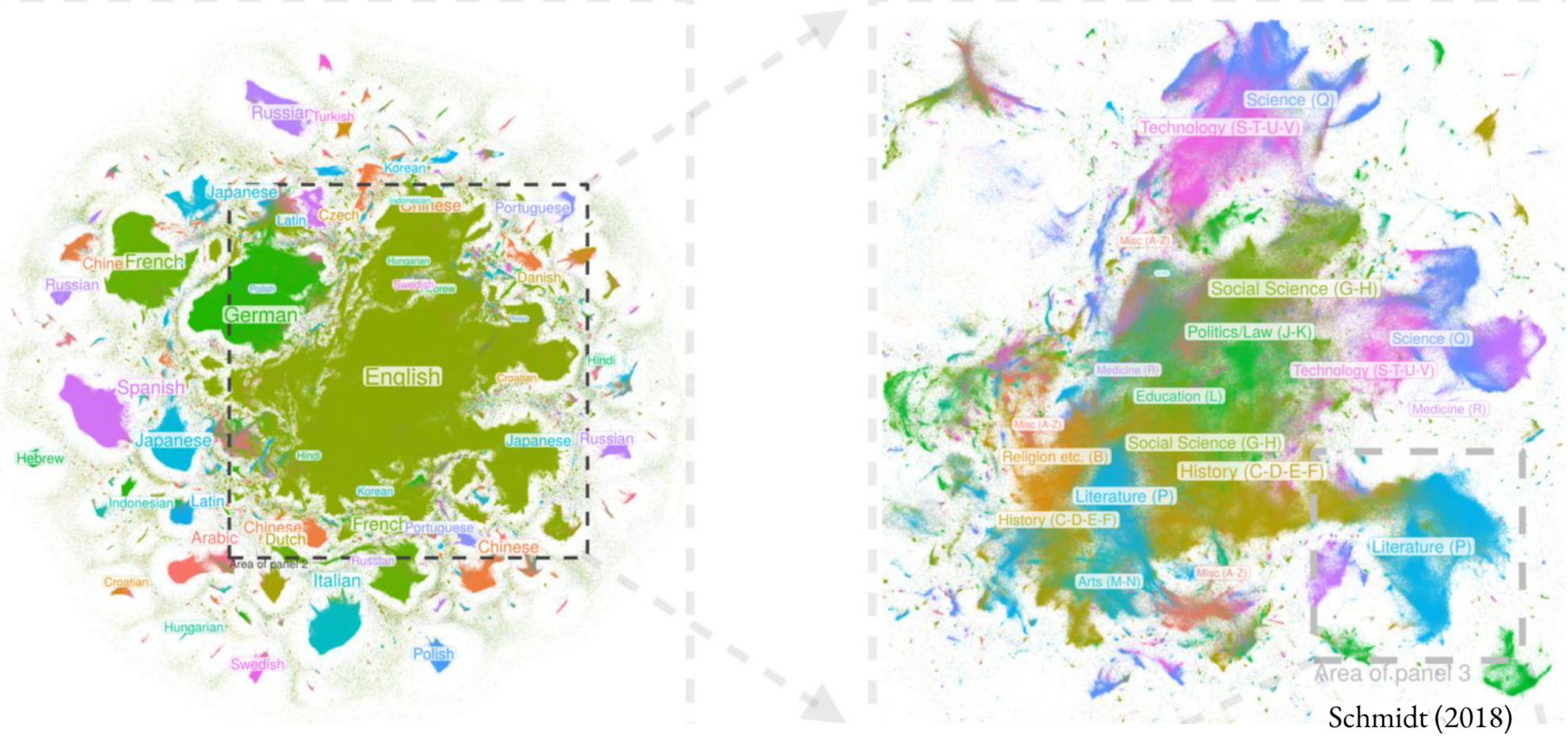


Single-cell transcriptomics (single-cell RNA sequencing): samples are cells, features are genes.



Zeisel et al. (2018)
$n \approx 500{,}000$

Digital humanities: samples are books, features are words.



Schmidt (2018)
$n \approx 15{,}000{,}000$

Some pros and cons: PCA vs t-SNE

PCA—captures global structure via shared PC axes.

- No issues with local minima / sub-optimal solutions---SVD gives minimal.
- Projection axes can be examined for interpretability.
- Can be computed efficiently (SVD)
- Can embed new points without updating the embedding space.
- Limited to capturing linear structures (but can be generalized using augmented feature space/kernels).

t-SNE—captures local structure by preserving neighborhoods

- Can model non-linear manifolds.
- Thus can preserve richer local structures (e.g. think swiss roll).
- Expensive to compute, but there are tricks to speed it up.
- Subject to local minima during optimization.
- No explicit projection mapping, so:
  - less potential for interpretability
  - cannot be applied to new points (no explicit mapping given).---must re-run it.