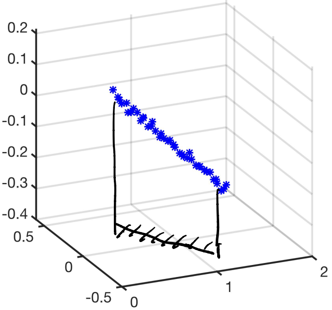
DDA4210/AIR6002 Advanced Machine Learning
Lecture 07 Non-Linear Dimensionality Reduction

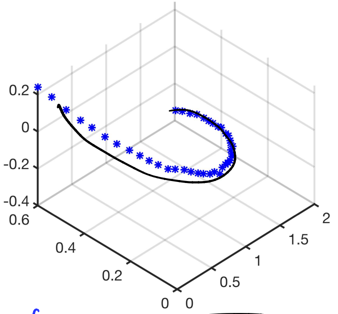
Tongxin Li

School of Data Science, CUHK-Shenzhen

Spring 2024

Overview





Linear DR

eigenvalues of $X^T X$

$\lambda_1 = 1$

$\lambda_2 = 0.8$

$\lambda_3 = 0.01$

⋮

Non-Linear DR

$\lambda_1 = 0.5$

$\lambda_2 = 0.49$?

$\lambda_3 = 0.4$

⋮

Popular linear DR algorithms

- PCA
- LDA (supervised)
- Multidimensional Scaling (MDS)

Introduction: Multidimensional Scaling (MDS)

- **Problem:** Given euclidean distances among points, recover the position of the points!

$$\mathbf{D} \in \mathbb{R}^{N \times N} \longrightarrow \mathbf{X} \in \mathbb{R}^{D \times N}$$

- **Example:** The road distances between 21 European cities (almost euclidean, but not quite) are as follows:

$D :=$

	Athens	Barcelona	Brussels	Calais	Cherbourg
Barcelona	3313				
Brussels	2963	1318			
Calais	3175	1326	204		
Cherbourg	3339	1294	583	460	
Cologne	2762	1498	206	409	785

$\vdots$

Can we construct a map?

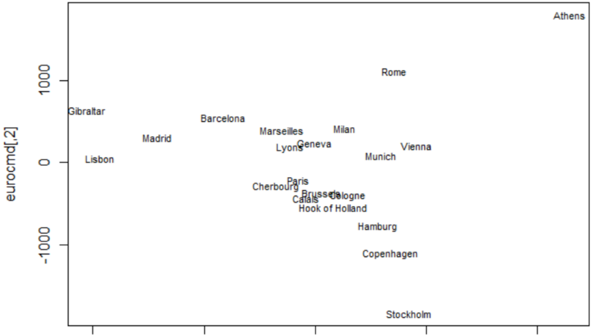
Introduction: Multidimensional Scaling (MDS)

Road distance

	Athens	Barcelona	Brussels	Calais	Cherbourg
Barcelona	3313				
Brussels	2963	1318			
Calais	3175	1326	204		
Cherbourg	3339	1294	583	460	
Cologne	2762	1498	206	409	785

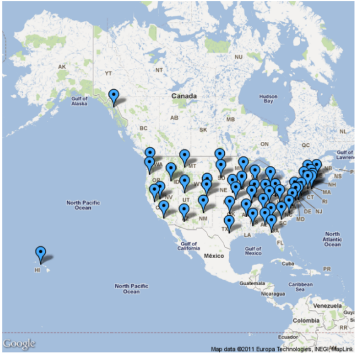
⋮

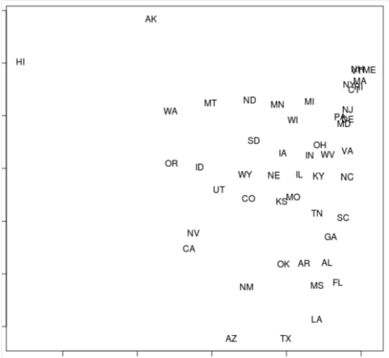
Constructed 2-D map (data visualization)



Introduction: Multidimensional Scaling

Transform US city distances to city locations (2D)

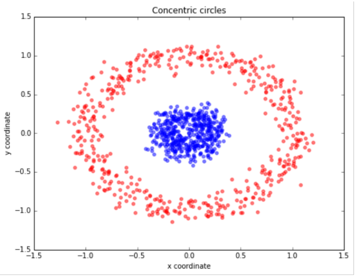


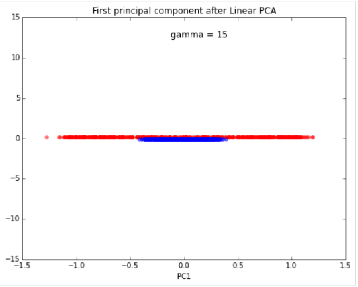


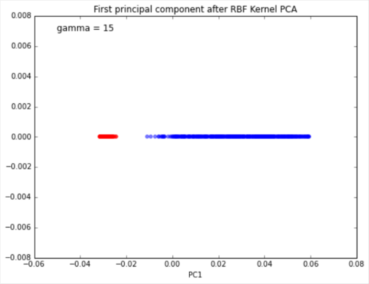
More about MDS

- `https://www.stat.pitt.edu/sungkyu/course/2221Fall13/lec8_mds_combined.pdf`
- Cox, M., Cox, T. (2008). Multidimensional Scaling. In: Handbook of Data Visualization. Springer Handbooks Comp.Statistics. Springer, Berlin, Heidelberg. `https://link.springer.com/content/pdf/10.1007/`

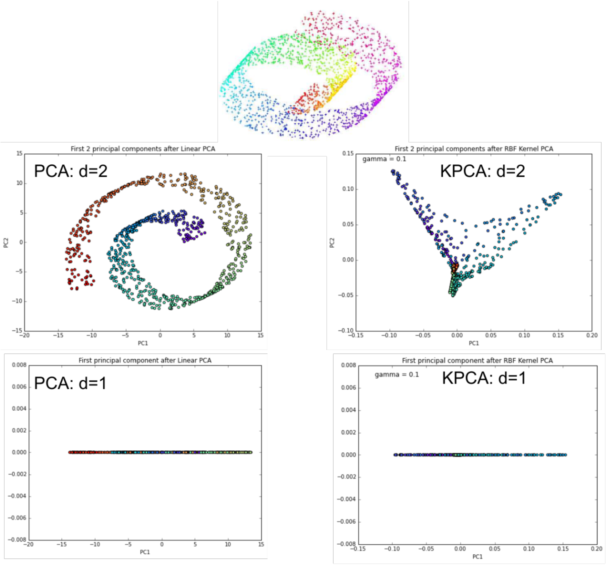
Introduction: A Toy Example of NLDR







Introduction: A Toy Example of NLDR



Introduction: NLDR

Important NLDR algorithms

- Kernel PCA[1]
- Locally Linear Embedding[2]
- Isomap[3]
- Autoencoder[4]
- t-SNE[5]
- UMAP[6]

---

[1] B. Scholkopf, A. Smola, and K.-R. Muller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation, vol. 10, no. 5, pp. 1299-1319, 1998.

[2] Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, v.290 no.5500 , Dec.22, 2000. pp.2323-2326

[3] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. Science, 290(5500):2319-2323, 2000.

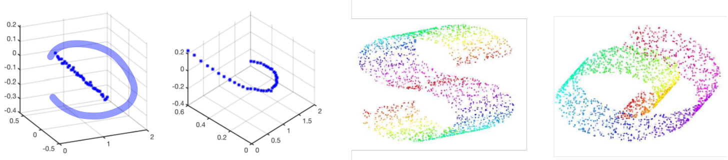
[4] Hinton, Geoffrey E. and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science 313, no. 5786 (2006): 504-507.

[5] Van der Maaten, L.J.P. and Hinton, G.E. Visualizing Data Using t-SNE. Journal of Machine Learning Research. 2008, 9: 2579-2605.

[6] McInnes, Leland, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv: 1802.03426,2018.

Locally Linear Embedding (LLE)

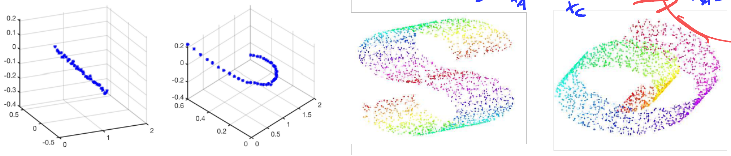
- Based on a simple geometric intuition of local linearity
- Assume each sample and its neighbors lie on or close to a locally linear patch of the manifold
  - Manifold: low-dimentional surface embedded (nonlinearly) in high-dimentional space.
  - Examples of manifold



- LLE assumption: projection should preserve the neigborhood
  - Projected data point should have the same neigborhood as the original point
  - Locally linear representation should be preserved

Locally Linear Embedding (LLE)

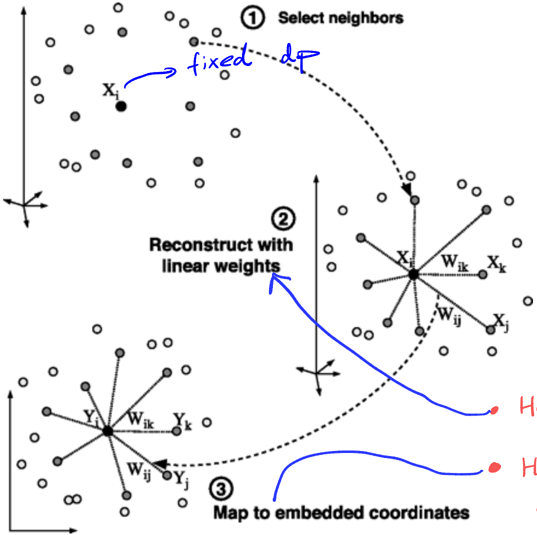
- Based on a simple geometric intuition of local linearity
- Assume each sample and its neighbors lie on or close to a locally linear patch of the manifold
  - Manifold: low-dimentional surface embedded (nonlinearly) in high-dimentional space.
  - Examples of manifold



- LLE assumption: projection should preserve the neigborhood
  - Projected data point should have the same neigborhood as the original point
  - Locally linear representation should be preserved

LLE: Main Idea

- Neighborhood-preserved projection



Base model

Some weights I can change.

$$\mathbf{x}_i \approx \sum_{j \in \mathcal{N}_i^k} w_{ij} \mathbf{x}_j$$

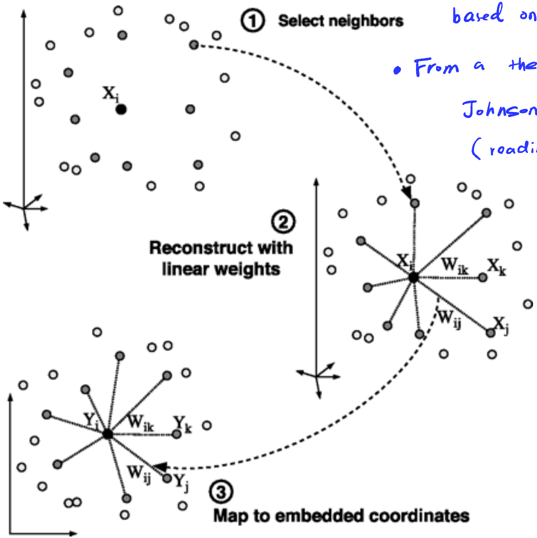
→ set of neighbors

$$\sum_{j \in \mathcal{N}_i^k} w_{ij} = 1$$

- How to get the weights?
- How can we map to embedded coordinates?

LLE: Main Idea

- Neighborhood-preserved projection



based on some distance

- From a theoretical point of view:

  Johnson-Lindenstrauss Lemma

  (reading material)

$$\mathbf{x}_i \approx \sum_{j \in \mathcal{N}_i^k} w_{ij} \mathbf{x}_j$$

$$\sum_{j \in \mathcal{N}_i^k} w_{ij} = 1$$

LLE: Algorithm

Goal : Dimension Reduction · $D \times N \to d \times N$ $D \gg d$

- Input: D-dimensional data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, d, k ($k \geq d+1$)
  1. For each data point $\mathbf{x}_i$, find its k nearest neighbors $\mathcal{N}_i^k$ (# of neighbors.)
  2. Find the locally linear reconstruction weights by solving

$$\mathbf{W} = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{x}_i - \sum_{j \in \mathcal{N}_i^k} w_{ij}\mathbf{x}_j\|^2, \quad \text{s.t. } \forall i \sum_{j \in \mathcal{N}_i^k} w_{ij} = 1$$

  3. Use $\mathbf{W}$ to compute the low-dimensional projections

$$\mathbb{R}^{d\times N} \ni \mathbf{Z} = \underset{\mathbf{Z}}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{z}_i - \sum_{j \in \mathcal{N}_i^k} w_{ij}\mathbf{z}_j\|^2, \quad \text{s.t. } \sum_{i=1}^{N} \mathbf{z}_i = 0, \ \frac{1}{N}\mathbf{Z}\mathbf{Z}^\top = \mathbf{I}$$

- Output: Low-dimensional embedding $\mathbf{Z} \in \mathbf{R}^{d \times N}$

LLE: Algorithm

- Input: D-dimensional data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, d, k ($k \geq d+1$)
  1. For each data point $\mathbf{x}_i$, find its k nearest neighbors $\mathcal{N}_i^k$
  2. Find the locally linear reconstruction weights by solving

$\|X - XW\|_F^2$

$$\mathbf{W} = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{x}_i - \sum_{j \in \mathcal{N}_i^k} w_{ij} \mathbf{x}_j\|^2, \quad \text{s.t. } \forall i \sum_{j \in \mathcal{N}_i^k} w_{ij} = 1$$

original dp   linear approximation.

  3. Use $\mathbf{W}$ to compute the low-dimensional projections

$$\mathbf{Z} = \underset{\mathbf{Z}}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{z}_i - \sum_{j \in \mathcal{N}_i^k} w_{ij} \mathbf{z}_j\|^2, \quad \text{s.t. } \sum_{i=1}^{N} \mathbf{z}_i = 0, \ \frac{1}{N} \mathbf{Z}\mathbf{Z}^\top = \mathbf{I}_d$$

- Output: Low-dimensional embedding $\mathbf{Z} \in \mathbf{R}^{d \times N}$

LLE: Algorithm

- Input: D-dimensional data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, d, k ($k \geq d+1$)
  1. For each data point $\mathbf{x}_i$, find its k nearest neighbors $\mathcal{N}_i^k$
  2. Find the locally linear reconstruction weights by solving

$$\mathbf{W} = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{x}_i - \sum_{j \in \mathcal{N}_i^k} w_{ij}\mathbf{x}_j\|^2, \quad \text{s.t. } \forall i \sum_{j \in \mathcal{N}_i^k} w_{ij} = 1$$

avoid degenerate solutions (unit covariance)

  3. Use $\mathbf{W}$ to compute the low-dimensional projections

$\|Z - ZW\|_F^2$ self-expression!

$$\mathbf{Z} = \underset{\mathbf{Z}}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{z}_i - \sum_{j \in \mathcal{N}_i^k} w_{ij}\mathbf{z}_j\|^2, \quad \text{s.t.} \sum_{i=1}^{N} \mathbf{z}_i = 0, \ \frac{1}{N}\mathbf{Z}\mathbf{Z}^\top = \mathbf{I}_d$$

remove translation freedom (need to be centered at origin)

- Output: Low-dimensional embedding $\mathbf{Z} \in \mathbf{R}^{d \times N}$

What is Z? How to compute it?

LLE: Algorithm

- Computation of $\mathbf{W}$ → step 2
  See the paper [7] if your are interested in it.

- Computation of $\mathbf{Z}$ → step 3.

$$\sum_{i=1}^{N} \|\mathbf{z}_i - \sum_{j \in \mathcal{N}_i^k} w_{ij}\mathbf{z}_j\|^2 = \|\mathbf{Z} - \mathbf{Z}\mathbf{W}\|_F^2 = \mathrm{trace}\left(\mathbf{Z}(\mathbf{I} - \mathbf{W})(\mathbf{I} - \mathbf{W}^\top)\mathbf{Z}^\top\right)$$

// Z(I-W)

(A detailed derivation is in the reading materials on course web; optional; similar to PCA)

Therefore, $\mathbf{Z}$ should be composed of the d eigenvectors of $(\mathbf{I} - \mathbf{W})(\mathbf{I} - \mathbf{W}^\top)$ corresponding to the smallest d nonzero eigenvalues, i.e., Q: why $\lambda_1 = 0$? what is v_1?

$$\mathbf{Z} = [\mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_{d+1}]^\top$$

[7] Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, v.290 no.5500 , Dec.22, 2000. pp.2323-2326

LLE: Algorithm

- Computation of $\mathbf{W}$
  See the paper [7] if your are interested in it.

- Computation of $\mathbf{Z}$

min. $\mathbf{Z}$

$$\sum_{i=1}^{N} \|\mathbf{z}_i - \sum_{j \in \mathcal{N}_i^k} w_{ij}\mathbf{z}_j\|^2 = \|\mathbf{Z} - \mathbf{Z}\mathbf{W}\|_F^2 = \text{trace}\left(\mathbf{Z}(\mathbf{I} - \mathbf{W})(\mathbf{I} - \mathbf{W}^\top)\mathbf{Z}^\top\right)$$

$\| \mathbf{Z}(I-W)\|_F^2$

Therefore, $\mathbf{Z}$ should be composed of the d eigenvectors of $(\mathbf{I} - \mathbf{W})(\mathbf{I} - \mathbf{W}^\top)$ corresponding to the smallest d nonzero eigenvalues, i.e.,

Remaining eigenvalues are often non-zeros in practice.

$$\mathbf{Z} = [\mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_{d+1}]^\top$$

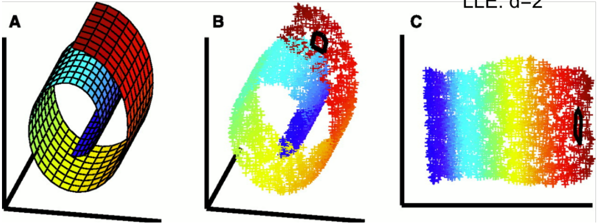
all-one vector

Note $W \cdot 1 = 1 \Rightarrow (I-W)1 = 0 \Rightarrow (I-W^T)(I-W)1 = 0$

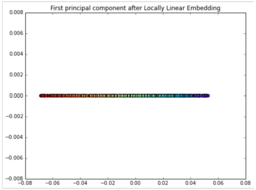
[7] Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, v.290 no.5500 , Dec.22, 2000. pp.2323-2326

Toy examples

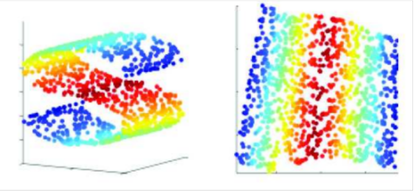
LLE: d=2



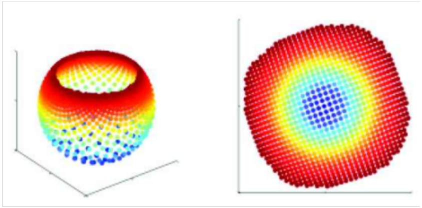
LLE: d=1



LLE: d=2

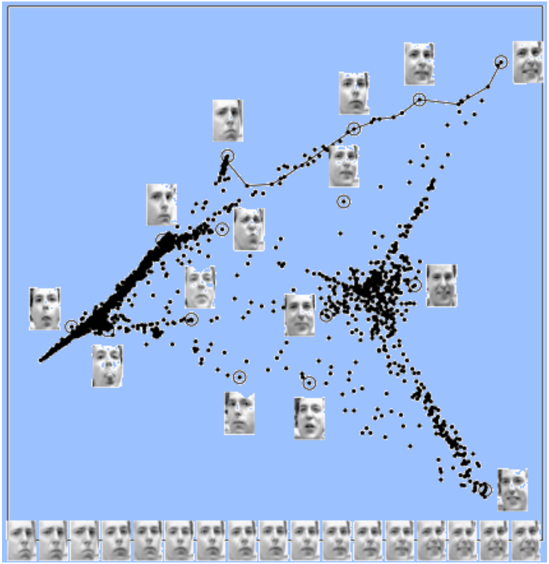


LLE: d=2



LLE: Applications

Face images



Limitations of Local Methods

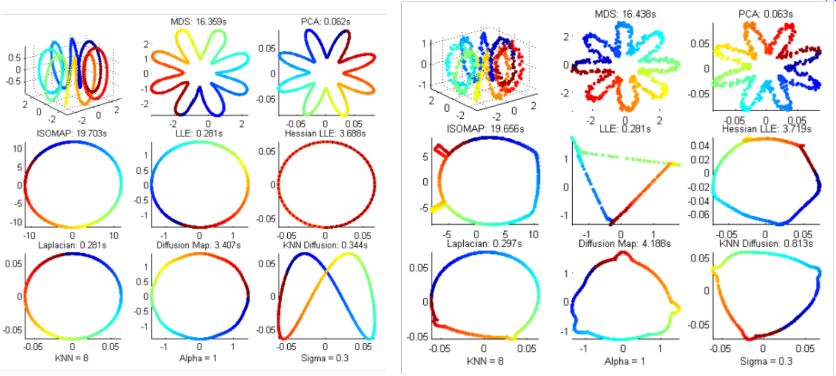
- Sensitive to noise

• Local Method: LLE sensitive to noise

• Global Method: PCA, and t-SNE (will be discussed later!) more robust to noise

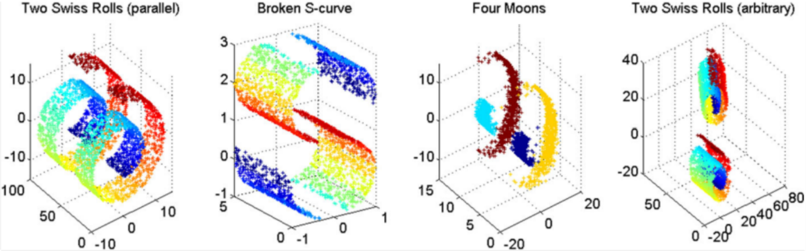
Clean

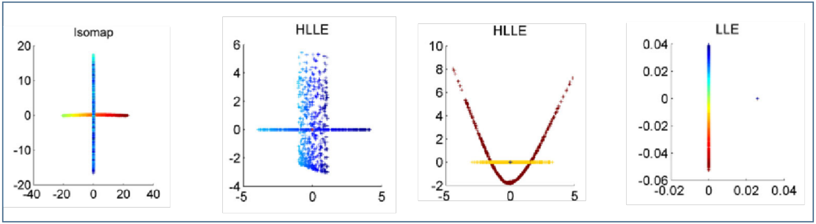
Noisy



Limitations of Local Methods

- Cannot handle disconnected data





Recall: • LLE focuses on local information.

• sensitive to noise

• How to improve ?

→ neighbor set ⟶ ? Stochastic relation.

[hard-threshold] [soft constraints]

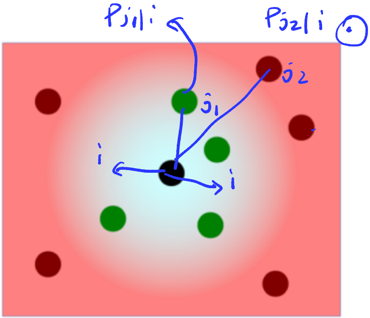
t-SNE: Neighborhood Probability

- Given D high-dimensional data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, let

$$p_{j|i} = \begin{cases} \dfrac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)}, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases}$$

$$\sum_{j=1}^{N} p_{j|i} = 1$$

- $p_{j|i}$ denotes the probability that $\mathbf{x}_j$ is a neighbor of $\mathbf{x}_i$
- The parameter σ_i sets the size of the neighborhood of $\mathbf{x}_i$
- Set σ_i differently for each data point (according to “Perplexity")
- Results depend heavily on σ_i-it defines the neighborhoods we are trying to preserve

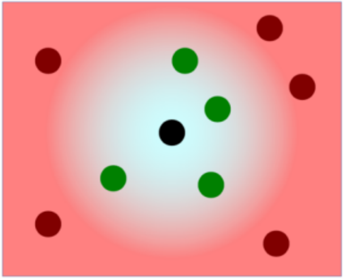


t-SNE: Neighborhood Probability

- Given D high-dimensional data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, let

$$p_{j|i} = \begin{cases} \dfrac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)}, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases}$$

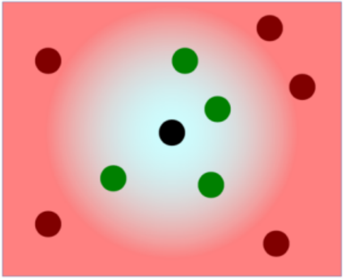
- Let $p_{ij} = \dfrac{p_{j|i} + p_{i|j}}{2N}$ $\forall$ $i = 1, \cdots, N$, $j = 1, \cdots, N$
- Then $p_{ij} = p_{ji}$, $p_{ii} = 0$, $\sum_{i,j} p_{ij} = 1$



t-SNE: Neighborhood Probability

- Given D high-dimensional data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, let

$$p_{j|i} = \begin{cases} \dfrac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases}$$



- Let $p_{ij} = \dfrac{p_{j|i} + p_{i|j}}{2N}$
- Then $p_{ij} = p_{ji}, \; p_{ii} = 0, \; \sum_{i,j} p_{ij} = 1$

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ \vdots & \ddots & & \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{bmatrix} = \{p_{ij}\}_{i,j}$$

↓ later we will use this matrix

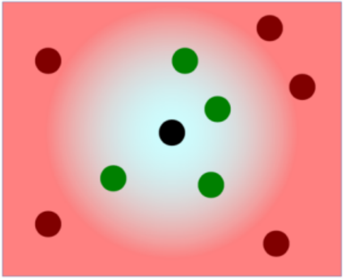
t-SNE: Neighborhood Probability

- Given D high-dimensional data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, let

$$p_{j|i} = \begin{cases} \dfrac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases}$$

Q: How to choose $\{\sigma_i\}$?

- Let $p_{ij} = \dfrac{p_{j|i} + p_{i|j}}{2N}$
- Then $p_{ij} = p_{ji}$, $p_{ii} = 0$, $\sum_{i,j} p_{ij} = 1$



t-SNE: Neighborhood Probability

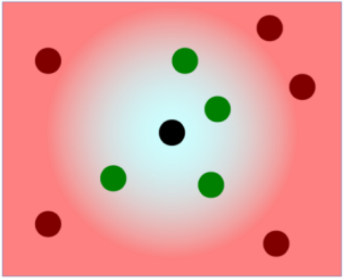
- Given D high-dimensional data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, let

$$p_{j|i} = \begin{cases} \dfrac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases}$$

Idea: Choosing $\{\sigma_i\}$ is hard, but we can define a new information theoretic metric [next slide]!

perplexity

- Let $p_{ij} = \dfrac{p_{j|i} + p_{i|j}}{2N}$
- Then $p_{ij} = p_{ji}$, $p_{ii} = 0$, $\sum_{i,j} p_{ij} = 1$

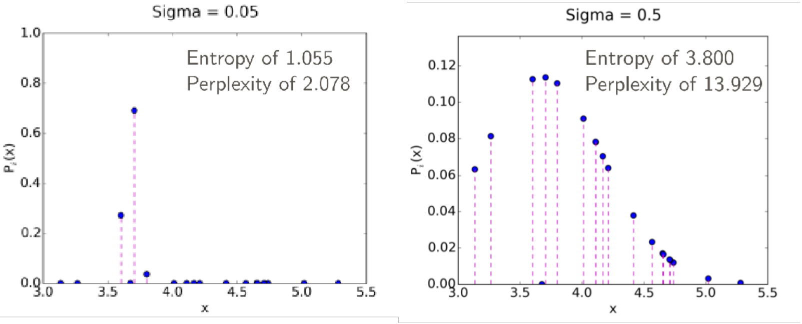


t-SNE: Perplexity (to determine σ_i) (optional)

- For each data point, define the perplexity:

$$\text{perp}(i) = 2^{H(p_{j|i})}, \qquad H(p_{j|i}) = -\sum_{j=1}^{N} p_{j|i} \log p_{j|i}$$

  - A low perplexity indicates the probability distribution is good at predicting the sample.
  - Define the desired perplexity and set σ_i to get that (e.g. bisection)
  - Values between 5-50 usually work well



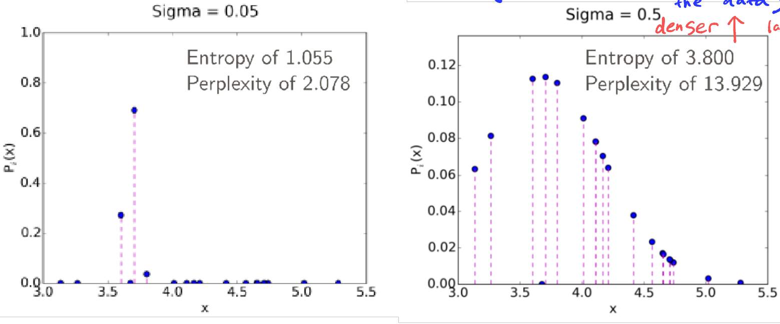
t-SNE: Perplexity (to determine σ_i) (optional)

- For each data point, define the perplexity: → entropy of $\{P_{j|i}\}_{j=1}^{N}$

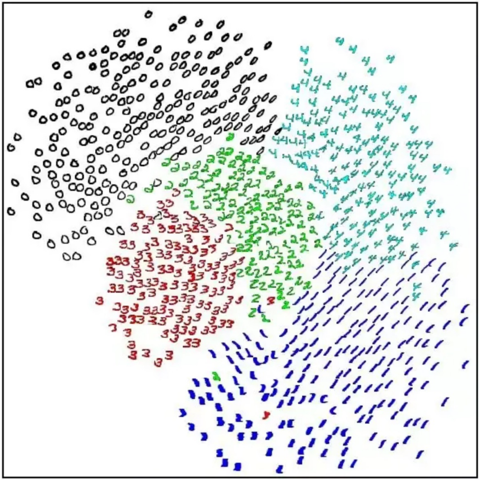
$$\text{perp}(i) = 2^{H(p_{j|i})}, \qquad H(p_{j|i}) = -\sum_{j=1}^{N} p_{j|i} \log p_{j|i}$$

→ The performance of t-SNE is fairly robust under different settings.

  - A low perplexity indicates the probability distribution is good at predicting the sample.
  - Define the desired perplexity and set σ_i to get that (e.g. bisection)
  - Values between 5-50 usually work well (depending on the density of the data). denser ↑ larger ↑



t-SNE: Objective



Another example of approximating using q_{ij}.

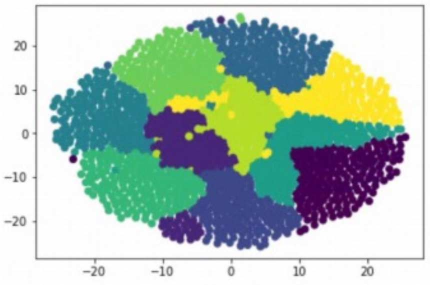
This is called SNE

Solution ?

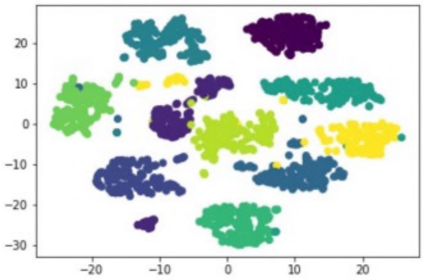
[8]Not $q_{ji} = \frac{\exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2)}{\sum_k \sum_{l \neq k} \exp(-\|\mathbf{z}_l - \mathbf{z}_k\|^2)}$, which corresponds to SNE. It has a crowding problem: in 2D or 3D, we do not have enough room to accommodate all neighbors when using Gaussian distribution.

t-SNE: Objective

- Idea : Make the lower dimensional distribution more uniform so the datapoints spread out.
- How ?



Crowding Problem



TSNE solves crowding problem

---

[8] Not $q_{ji} = \frac{\exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2)}{\sum_k \sum_{l \neq k} \exp(-\|\mathbf{z}_l - \mathbf{z}_k\|^2)}$, which corresponds to SNE. It has a crowding problem: in 2D or 3D, we do not have enough room to accommodate all neighbors when using Gaussian distribution.

t-SNE: Objective

- Learn a d-dimensional embedding $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N$ ($d < D$ and usually $d = 2$ or 3)[8]

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

Such that $Q = [q_{ij}]_{N \times N}$ is close to $P = [p_{ij}]_{N \times N}$.

[8] Not $q_{ji} = \frac{\exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2)}{\sum_k \sum_{l \neq k} \exp(-\|\mathbf{z}_l - \mathbf{z}_k\|^2)}$, which corresponds to SNE. It has a crowding problem: in 2D or 3D, we do not have enough room to accommodate all neighbors when using Gaussian distribution.

t-SNE: Objective

- Learn a d-dimensional embedding $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N$ ($d < D$ and usually $d = 2$ or 3)[8]

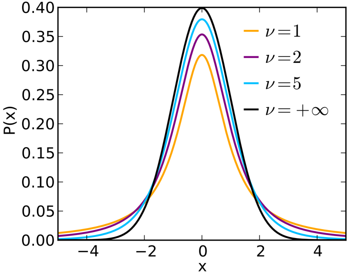
$$q_{ij} = \frac{(1 + \|\mathbf{z}_i - \mathbf{z}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{z}_k - \mathbf{z}_l\|^2)^{-1}}$$

Such that $Q = [q_{ij}]_{N \times N}$ is close to $P = [p_{ij}]_{N \times N}$.

Student-t probability density

$$p(x) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\,\Gamma(\frac{\nu}{2})} \left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

- When $\nu = \infty$, $p(x)$ is Gaussian
- When $\nu = 1$, $p(x) \propto \left(1 + x^2\right)^{-1}$



[8] Not $q_{ji} = \frac{\exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2)}{\sum_k \sum_{l \neq k} \exp(-\|\mathbf{z}_l - \mathbf{z}_k\|^2)}$, which corresponds to SNE. It has a crowding problem: in 2D or 3D, we do not have enough room to accommodate all neighbors when using Gaussian distribution.

t-SNE: Objective

- Minimize the KL-divergence[9]

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

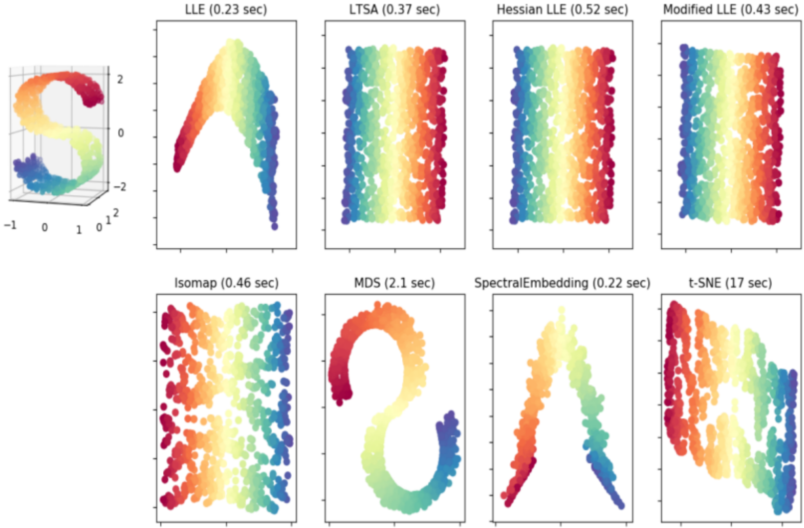
- Gradient based optimization:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}_i} = \sum_j (p_{ij} - q_{ij})(\mathbf{z}_i - \mathbf{z}_j)(1 + \|\mathbf{z}_i - \mathbf{z}_j\|^2)^{-1}, \quad i = 1, \ldots N$$

[9] A loss function to measure the difference between two distributions, similar to the cross entropy loss.

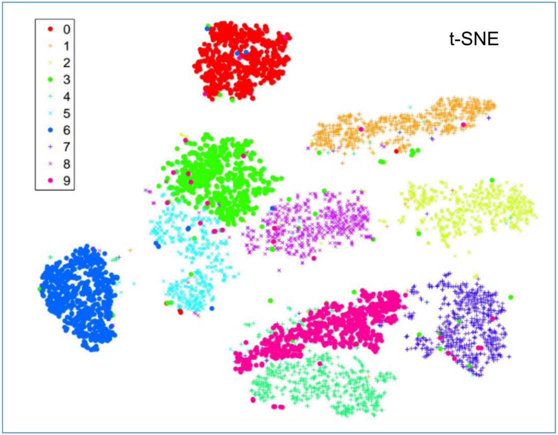
t-SNE: Applications

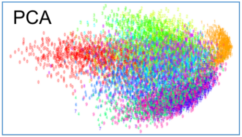
- Synthetic data

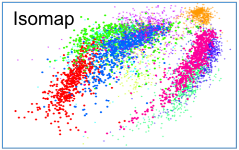


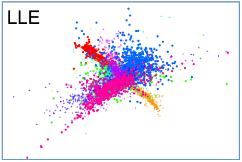
t-SNE: Applications

- MNIST handwritten digits (10 classes)

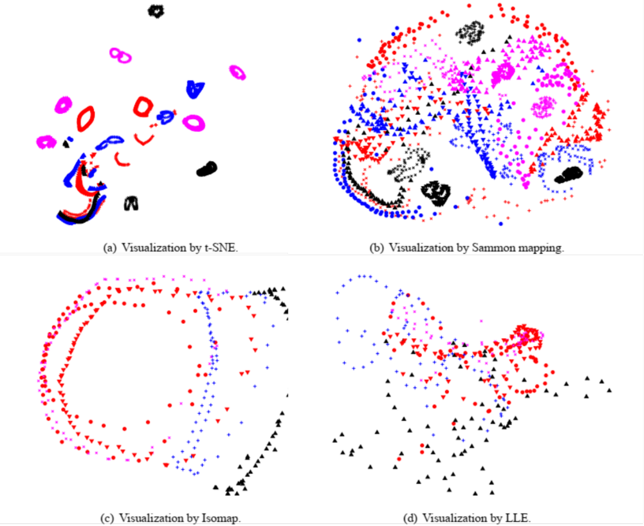








- COIL20 image data (20 objects with 72 different poses)

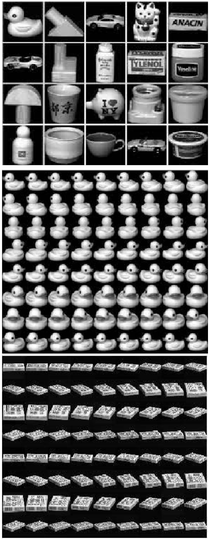


(a) Visualization by t-SNE.

(b) Visualization by Sammon mapping.

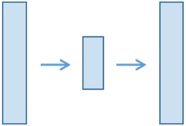
(c) Visualization by Isomap.

(d) Visualization by LLE.



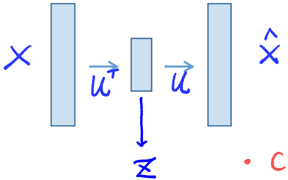
Autoencoder: Basic Structure

- Recall PCA: $\mathbf{Z} = \mathbf{U}^\top \mathbf{X}$, $\widehat{\mathbf{X}} = \mathbf{U}\mathbf{Z} = \mathbf{U}\mathbf{U}^\top \mathbf{X}$



Autoencoder: Basic Structure

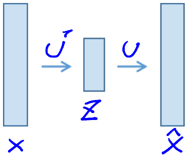
- Recall PCA: $\mathbf{Z} = \mathbf{U}^\top \mathbf{X}$, $\widehat{\mathbf{X}} = \mathbf{UZ} = \mathbf{UU}^\top \mathbf{X}$



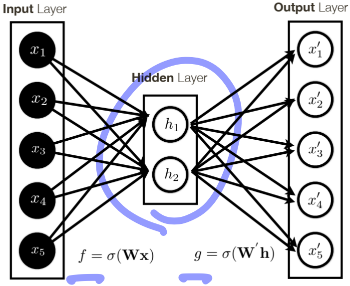
→ this is an encoding of X!

- Can we generalize this idea to neural networks?

- Recall PCA: $\mathbf{Z} = \mathbf{U}^\top \mathbf{X}, \ \widehat{\mathbf{X}} = \mathbf{U}\mathbf{Z} = \mathbf{U}\mathbf{U}^\top \mathbf{X}$



- Autoencoder: a neural network with output=input



Encoder/Decoder architecture

- Encoder: $f = \sigma(\mathbf{W}\mathbf{x})$
- Decoder: $g = \sigma(\mathbf{W}'\mathbf{h})$
- Hidden layer dimension<input dimension
- Predict the input by itself: $\mathbf{x} \approx g(f(\mathbf{x}))$

Autoencoder: Deep Model

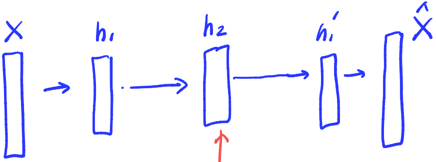
- Stacked Autoencoders (SAE)
  - Use the middle layer as a representation
  - Out-of-sample extension: just feed new data into the encoder
  - Question: out-of-sample extension for PCA, LLE, and t-SNE?

- Stacked Autoencoders (SAE)
  - Use the middle layer as a representation
  - Out-of-sample extension: just feed new data into the encoder
  - Question: out-of-sample extension for PCA, LLE, and t-SNE? →

Q: What if we obtain some new data after getting the models?

need to retrain the model

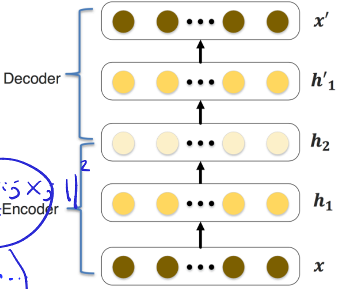
(online/streaming versions of PCA, LLE, tSNE exist)



$$\hat{x}_{new} = g(f(x_{new}))$$

Autoencoder: Deep Model

- Stacked Autoencoders (SAE)
  - Use the middle layer as a representation
  - Out-of-sample extension: just feed new data into the encoder
  - Question: out-of-sample extension for PCA, LLE, and t-SNE?



- Train the Autoencoder
  - For example, solve

$$\sum_{i=1}^{N} \| x_i - \sum_{j \in N_i} w_{ij} x_j \|^2$$

+ regularizer...

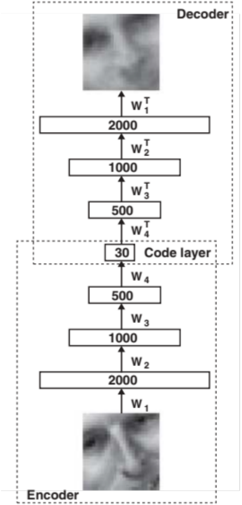
$$\underset{\theta}{\text{minimize}} \ \frac{1}{2} \sum_{i=1}^{N} \|\mathbf{x}_i - g(f(\mathbf{x}_i))\|^2$$

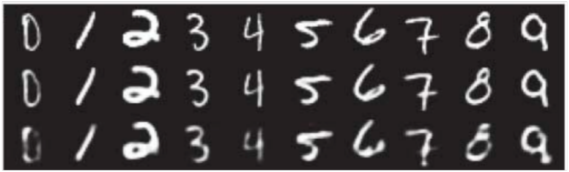
θ: network parameters

  - Backpropagation
  - Gradient-based optimization

* Autoencoder can be extended to convolutional neural networks—CAE

Autoencoder: Application-Data Compression





G. E. Hinton and R. R. Salakhutdinov. Science. 2006

MNIST 2D visualization

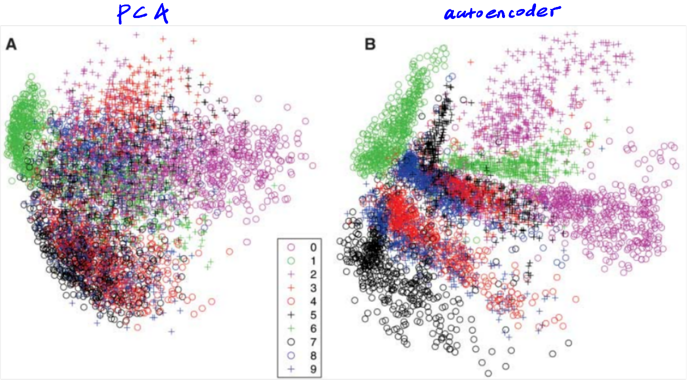
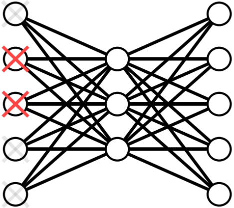


Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).

G. E. Hinton and R. R. Salakhutdinov. Science. 2006

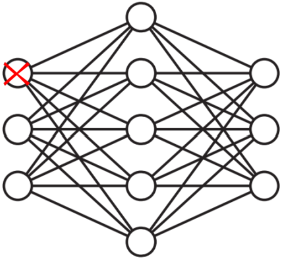
Autoencoder: Application-Image Denoising

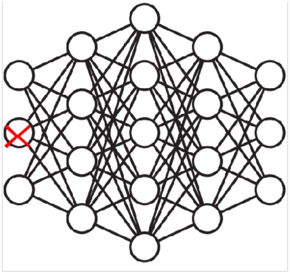
- Denoising Autoencoder



Reconstruction from corrupted data:
For each input sample, some of its components are randomly selected and set to 0, but the reconstruction error is computed by comparing to the original, non-corrupted data.

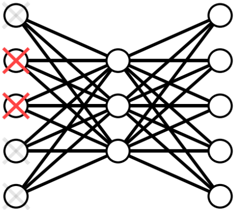
The size of hidden layer can be larger than the input size.





Autoencoder: Application-Image Denoising

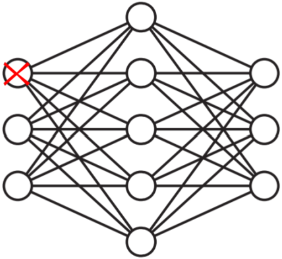
- Denoising Autoencoder

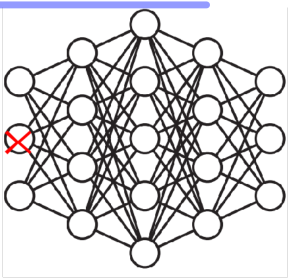


Reconstruction from corrupted data:
For each input sample, some of its components are randomly selected and set to 0, but the reconstruction error is computed by comparing to the original, non-corrupted data.

different from the normal setting.

The size of hidden layer can be larger than the input size.





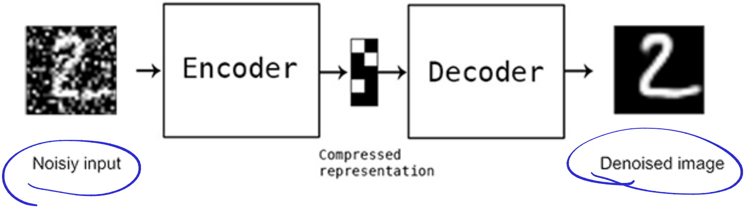
Autoencoder: Application-Image Denoising

- Train denoising Autoencoder, e.g.,

$$\underset{\theta}{\text{minimize}} \ \frac{1}{2}\sum_{i=1}^{N} \|\mathbf{x}_i - g(f(\tilde{\mathbf{x}}_i))\|^2$$

corrupted

  - $\tilde{\mathbf{x}}_i$: corrupted $\mathbf{x}_i$
  - May use other loss such as L1 norm
  - May add regularization to the loss function



Generative Models

Limitation of Autoencoder

- Cannot generate meaningful data using the decoder

Generative models

- Variational Autoencoder (VAE) (Kingma and Welling.ICLR 2014)
- Generative Adversarial Network (GAN) (Goodfellow et al. arXiv 2014)

Generative Models

Limitation of Autoencoder

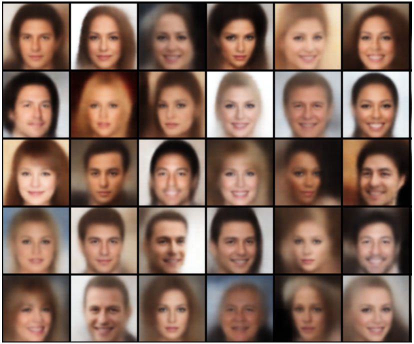
- Cannot generate meaningful data using the decoder

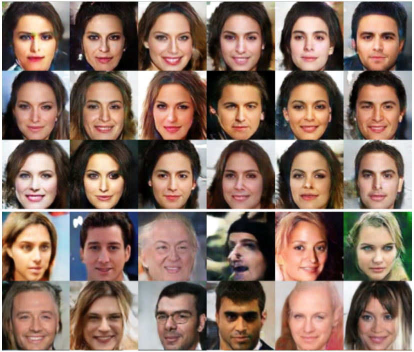
Generative models

- Variational Autoencoder (VAE) (Kingma and Welling.ICLR 2014)
- Generative Adversarial Network (GAN) (Goodfellow et al. arXiv 2014)

Examples of generated images (CelebA dataset)

Left: VAE. Right: GAN. Image from Pieters and Wiering 2018.





Learning Outcomes

- Understand the basic ideas of LLE, t-SNE, and Autoencoder
- Know the limitations of LLE, t-SNE, and Autoencoder
- Be able to use t-SNE to visualize real data
- Be able to use Autoencoder to reduce the noise of real data