# EECS 545: Machine Learning

# Lecture 17. Unsupervised Learning and Embedding Methods 2

Honglak Lee

3/17/2025

UNIVERSITY OF MICHIGAN

# Logistics

- HW3 due 03/18/25 (tomorrow).
- Project progress report late due date is tonight (with 3 late days applied)

# Outline

- Independent Component Analysis
- Sparse Coding
- Multidimensional Scaling
- Isomap
- t-distributed stochastic neighbor embedding (t-SNE)
- Autoencoder

# Independent Component Analysis

# Independent Component Analysis

- Independent Component Analysis (ICA)
  - Also called: “blind source separation”
- Suppose *m* independent signals are mixed, and sensed by *m* independent sensors.
  - Cocktail party with speakers and microphones [demo]
  - EEG with brain wave sources and sensors
    - Brain Computer Interface videos: [demo1, demo2, demo3].
  - etc.
- Can we reconstruct the original signals, given the mixed data from the sensors?

# Independent Component Analysis

- The sources **s** must be independent.
  - And they must be non-Gaussian.
  - (If Gaussian, then there is no way to find unique independent components.)
- Linear mixing to get the sensor signals **x**.
  - $\mathbf{x} = \mathbf{A}\mathbf{s}$
  - *or* $\mathbf{s} = \mathbf{W}\mathbf{x}$ (i.e., $\mathbf{W} = \mathbf{A}^{-1}$)
- **A** is called bases; **W** is called filters

# Algorithm for ICA

- There are several formulations of ICA:
  - **Maximum likelihood**
  - Maximizing non-Gaussianity

# Maximum-likelihood

- Maximum likelihood learning for W
  - By definition, the sources are independent

$$p(\mathbf{s}) = \prod_{j=1}^{m} p_s(s_j)$$

  - Then, the observed data distribution is given as:

$$p(\mathbf{x}) = \prod_{j=1}^{m} p_s(\mathbf{w}_j^T \mathbf{x}) \cdot |W|$$

  - We model CDF of source distribution as sigmoid:

$$\int_{-\infty}^{s} p_s(s')\, ds' = g(s) \rightarrow p_s(s) = g'(s)$$

$$g(s) = 1/\left(1 + e^{-s}\right) \qquad = g(s)(1 - g(s))$$

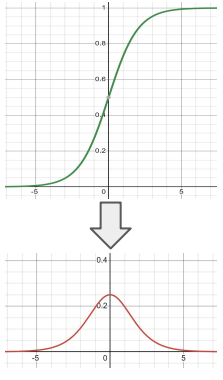
Use “change of variables” trick given:

$$\mathbf{s} = \mathbf{W}\mathbf{x}$$

$$p(\mathbf{x})Vol(d\mathbf{x}) = p(\mathbf{s})Vol(d\mathbf{s})$$

$$\begin{aligned} p(\mathbf{x})|d\mathbf{x}| &= p(\mathbf{s})|d\mathbf{s}| \\ &= p(\mathbf{s})|W\, d\mathbf{x}| \\ &= p(\mathbf{s})|W| \cdot |d\mathbf{x}| \end{aligned}$$

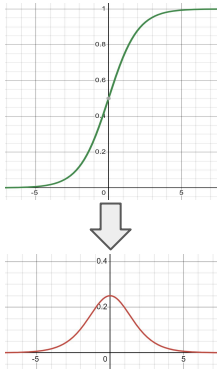
$$p(\mathbf{x}) = p(\mathbf{s})|W|$$



CDF g(s)

PDF g’(s)

# Maximum-likelihood (cont'd)



CDF g(s)

PDF g'(s)

- Maximum likelihood learning for W
  - We model CDF of source distribution as sigmoid:

$$p_s(s) = g'(s) = g(s)(1 - g(s)) \qquad g(s) = 1/\left(1 + e^{-s}\right)$$

  - Our loss is the log-likelihood of data

$$\ell(W) = \sum_{i=1}^{N} \left( \sum_{j=1}^{m} \log g'(\mathbf{w}_j^\top \mathbf{x}^{(i)}) + \log |W| \right)$$

# Maximum-likelihood (cont'd)

- Maximum likelihood learning for W
  - To get the update rule,

$$\ell(W) = \sum_{i=1}^{N} \left( \sum_{j=1}^{m} \log g'(\mathbf{w}_j^\top \mathbf{x}^{(i)}) + \log |W| \right)$$

  - SGD by taking derivative and using $\nabla_W |W| = |W| \left(W^{-1}\right)^\top$

$$W := W + \alpha \left( \begin{bmatrix} 1 - 2g(\mathbf{w}_1^\top \mathbf{x}^{(i)}) \\ 1 - 2g(\mathbf{w}_2^\top \mathbf{x}^{(i)}) \\ \vdots \\ 1 - 2g(\mathbf{w}_m^\top \mathbf{x}^{(i)}) \end{bmatrix} \mathbf{x}^{(i)\top} + \left(W^\top\right)^{-1} \right)$$

# Algorithm for ICA

- There are several formulations of ICA:
  - Maximum likelihood
  - **Maximizing non-Gaussianity**

# ICA by Maximizing non-Gaussianity

- Common steps of ICA (e.g., FastICA):
  - Apply PCA whitening (aka sphering) to the data
  - Find orthogonal unit vectors along which that the non-Gaussianity are maximized

$$\max_{W} L(W\tilde{\mathbf{x}})$$
$$\text{s.t.} \quad WW^{\top} = I$$

  - where L(x) can be Kurtosis, L1-norm, etc.

# PCA Whitening

- To whiten the input data,
  - We want a linear transformation

$$\widetilde{\mathbf{x}} = \mathbf{V}\mathbf{x}$$

  - So the components are uncorrelated:

$$\mathbb{E}\left[\widetilde{\mathbf{x}}\widetilde{\mathbf{x}}^\top\right] = \mathbf{I}$$

  - From PCA transformation matrix, $\Sigma = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$
    - We can use

$$\mathbf{V} = \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^\top$$

  - Because

$$\mathbb{E}\left[\widetilde{\mathbf{x}}\widetilde{\mathbf{x}}^\top\right] = \mathbb{E}\left[\mathbf{V}\mathbf{x}\mathbf{x}^\top\mathbf{V}^\top\right] = \mathbf{I}$$

# Maximizing non-Gaussianity

- Kurtosis

$\mu_4$ is the fourth central moment

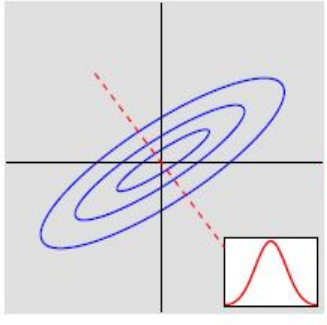
$$\operatorname{Kurt}[X] = \mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \frac{\mathbb{E}\left[(X-\mu)^4\right]}{\left(\mathbb{E}\left[(X-\mu)^2\right]\right)^2} = \frac{\mu_4}{\sigma_4}$$

  - Measure the “tailed-ness” of a distribution
  - All Gaussian distributions have Kurt=3
  - By maximizing Kurtosis, we can increase the “non-gaussianity”.

Kurtosis - Wikipedia https://en.wikipedia.org/wiki/Kurtosis

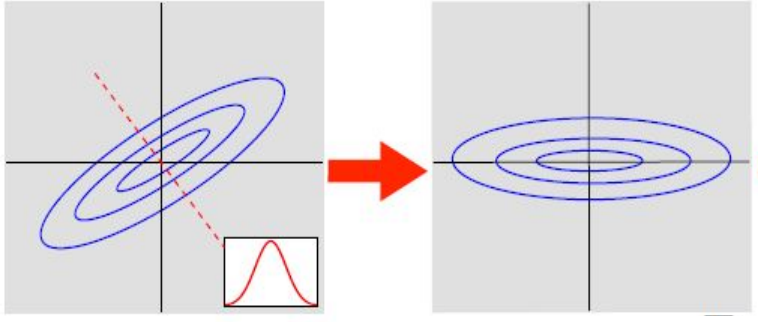
# PCA whitening (preprocessing for ICA): data from Gaussian

- Apply PCA: $\Sigma = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$



# PCA whitening (preprocessing for ICA): data from Gaussian

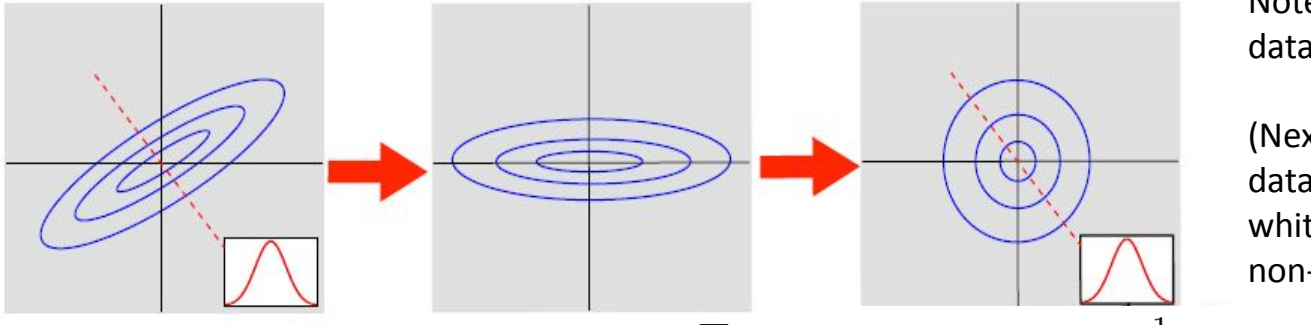
- Apply PCA: $\Sigma = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$
- Project (rotate) to the principal components



$\mathbf{x}$ $\quad$ $\mathbf{x}_{\mathrm{PCA}} = \mathbf{U}^\top\mathbf{x}$

# PCA whitening (preprocessing for ICA): : data from Gaussian

- Apply PCA: $\Sigma = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$
- Project (rotate) to the principal components
- “Scale” each axis so that the transformed data has identity as covariance.



$\mathbf{x}$ $\quad \mathbf{x}_{\text{PCA}} = \mathbf{U}^\top \mathbf{x}$ $\quad \mathbf{x}_{\text{white}} = \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^\top \mathbf{x}$

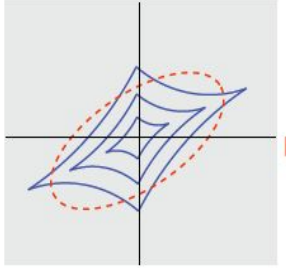
Note: this is a visualization for data with Gaussian distribution.

(Next slide:) For non-Gaussian data, ICA further rotates the whitened data to maximize non-gaussianity along each axis.

# ICA illustration: data from non-Gaussian distribution

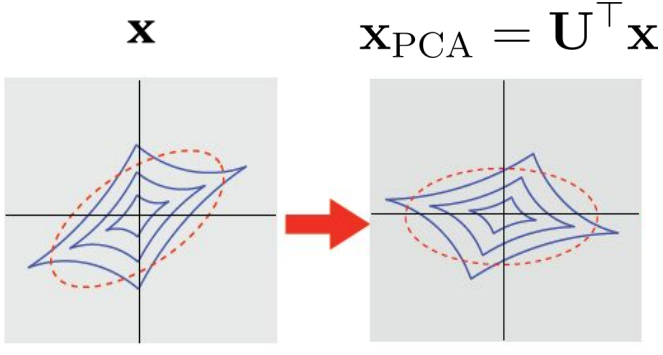
- PCA whitening
  - Apply PCA: $\Sigma = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$

$\mathbf{x}$



# ICA illustration: data from non-Gaussian distribution

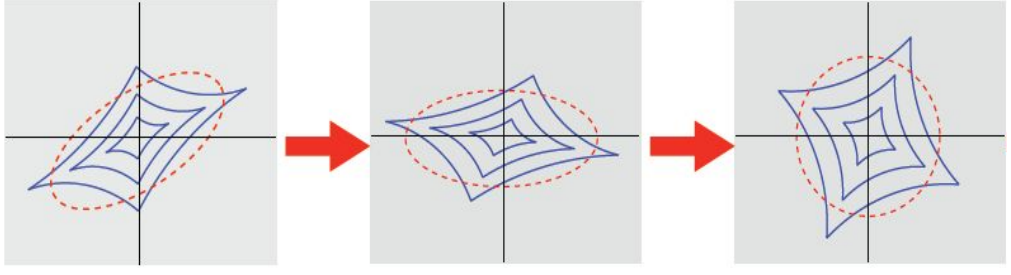
- PCA whitening
  - Apply PCA: $\Sigma = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$
  - Project (rotate) to the principal components



# ICA illustration: data from non-Gaussian distribution

- PCA whitening
  - Apply PCA: $\Sigma = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$
  - Project (rotate) to the principal components
  - “Scale” each axis so that the transformed data has identity as covariance.

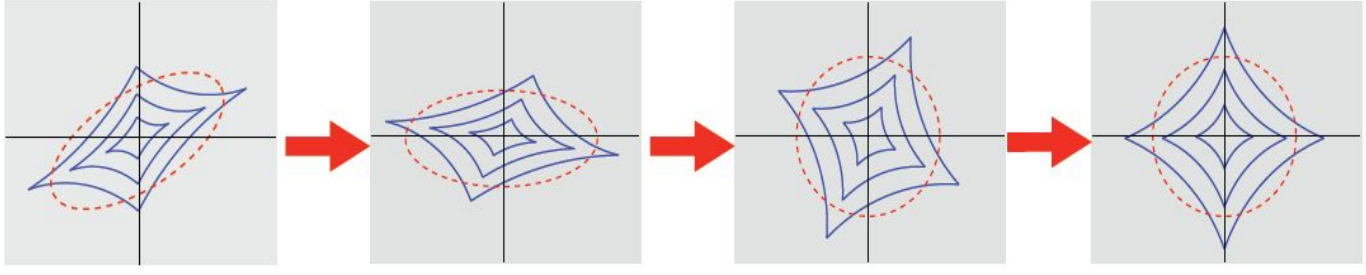
$\mathbf{x}$ $\quad \mathbf{x}_{\text{PCA}} = \mathbf{U}^\top\mathbf{x} \quad \mathbf{x}_{\text{white}} = \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^\top\mathbf{x}$



# ICA illustration: data from non-Gaussian distribution

- PCA whitening
  - Apply PCA: $\Sigma = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$
  - Project (rotate) to the principal components
  - “Scale” each axis so that the transformed data has identity as covariance.
- Rotate to maximize non-Gaussianity

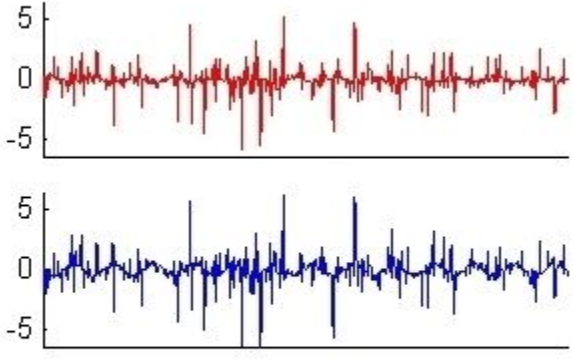
$\mathbf{x}$ $\quad \mathbf{x}_{\text{PCA}} = \mathbf{U}^\top \mathbf{x}$ $\quad \mathbf{x}_{\text{white}} = \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^\top \mathbf{x}$ $\quad \mathbf{x}_{\text{ICA}} = \mathbf{W}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^\top \mathbf{x}$



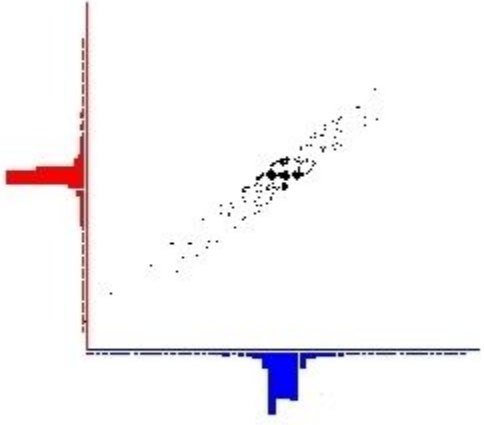
# Independent Component Analysis

- Mixture example.

Signals




Joint Density

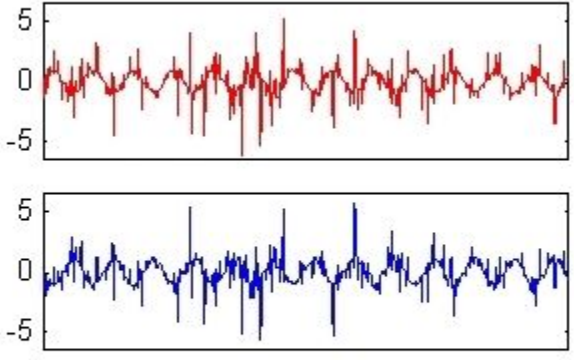


**Input signals and density**

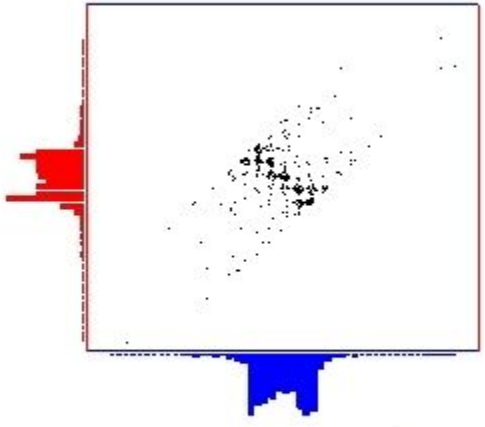
# Independent Component Analysis

- Remove correlations by whitening (sphering) the data.

Signals




Joint Density

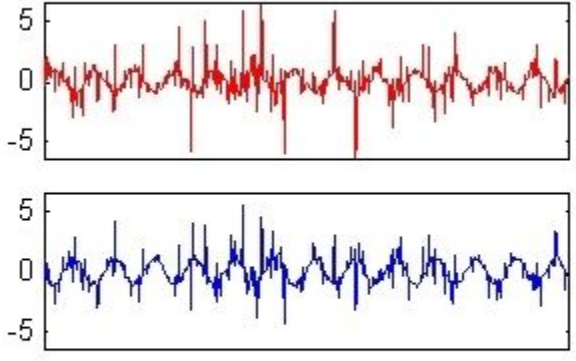


**Whitened signals and density**

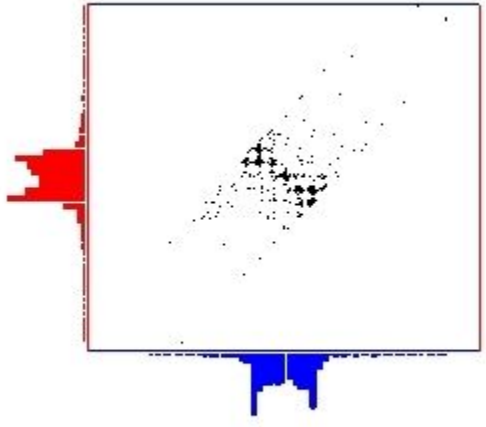
# Independent Component Analysis

- Step 1 of FastICA

Signals




Joint Density

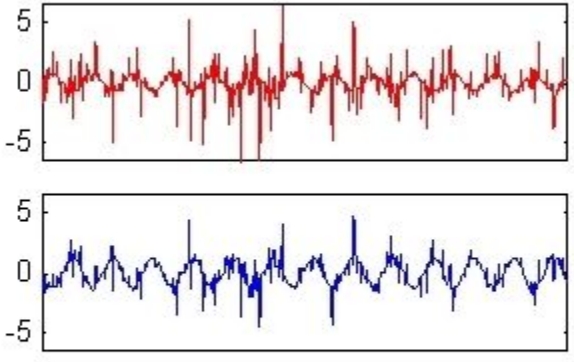


**Separated signals after 1 step of FastICA**

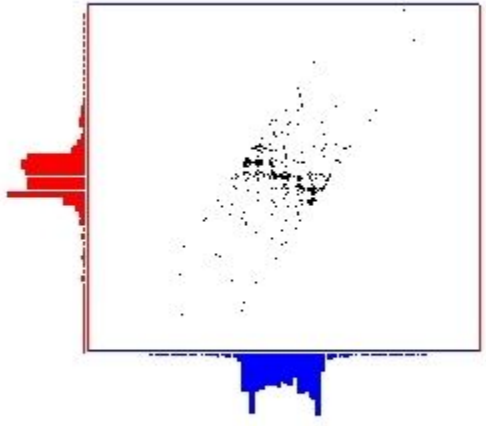
# Independent Component Analysis

- Step 2 of FastICA

Signals




Joint Density

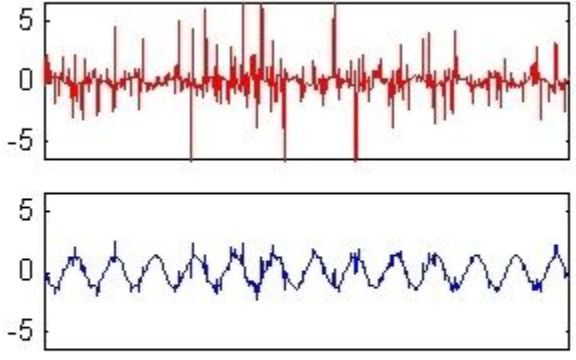


**Separated signals after 2 steps of FastICA**

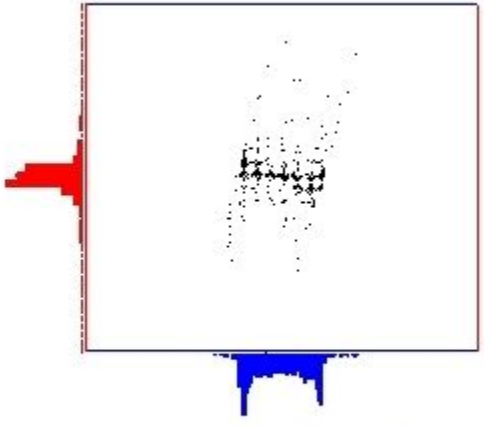
# Independent Component Analysis

- Step 3 of FastICA

Signals




Joint Density

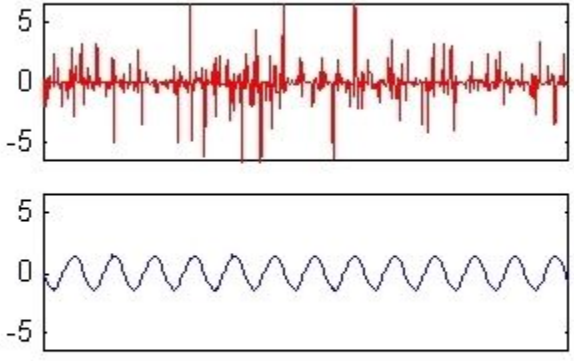


**Separated signals after 3 steps of FastICA**

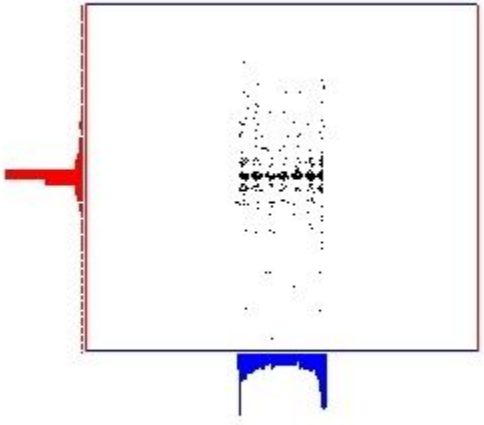
# Independent Component Analysis

- Step 4 of FastICA

Signals




Joint Density

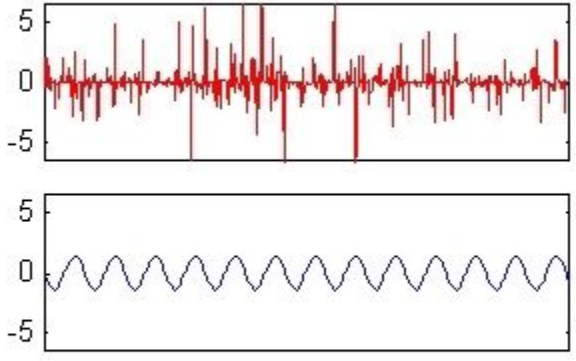


**Separated signals after 4 steps of FastICA**

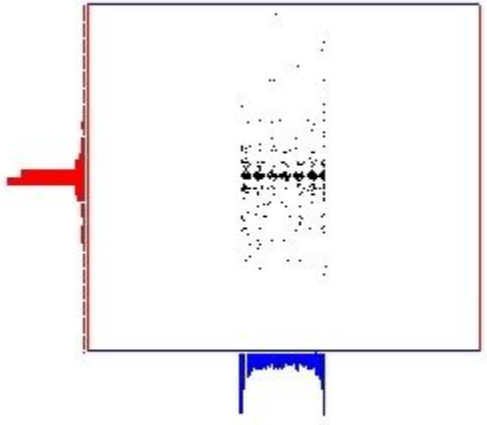
# Independent Component Analysis

- Step 5: note that $p(\mathbf{x}_{\mathrm{ICA},1}, \mathbf{x}_{\mathrm{ICA},2}) = p(\mathbf{x}_{\mathrm{ICA},1})p(\mathbf{x}_{\mathrm{ICA},2})$

Signals




Joint Density



**Separated signals after 5 steps of FastICA**

# ICA: summary

- Learning is done by PCA whitening followed by maximizing non-Gaussianity after transformations (kurtosis maximization).
- ICA is widely used for “blind-source separation.”
- The ICA components can be used for features.
- Limitations:
  - Difficult to learn overcomplete bases due to the orthogonality constraint
  - Difficult to handle situations where mixing is non-linear.

Blind Source Separation: Audio Examples
https://www.kecl.ntt.co.jp/icl/signal/sawada/demo/bss2to4/index.html
https://cnl.salk.edu/~tewon/Blind/blind_audio.html

# Sparse Coding

# Sparse coding

- Sparse coding [Olshausen and Field, 1997]
  - Objective: Given input data {**x**}, search for a set of bases {$\mathbf{b}_j$} such that

$$\mathbf{x} = \sum_j s_j \mathbf{b}_j$$

where $s_j$ are mostly zeros.

- Main intuition:
  - Build compact/succinct representations.
  - Learn interpretable and discriminative features.

# Two objectives in sparse coding

- Preserve information
  - Minimize the reconstruction error

$$||\mathbf{x}^{(i)} - \sum_j s_j^{(i)} \mathbf{b}_j||^2$$

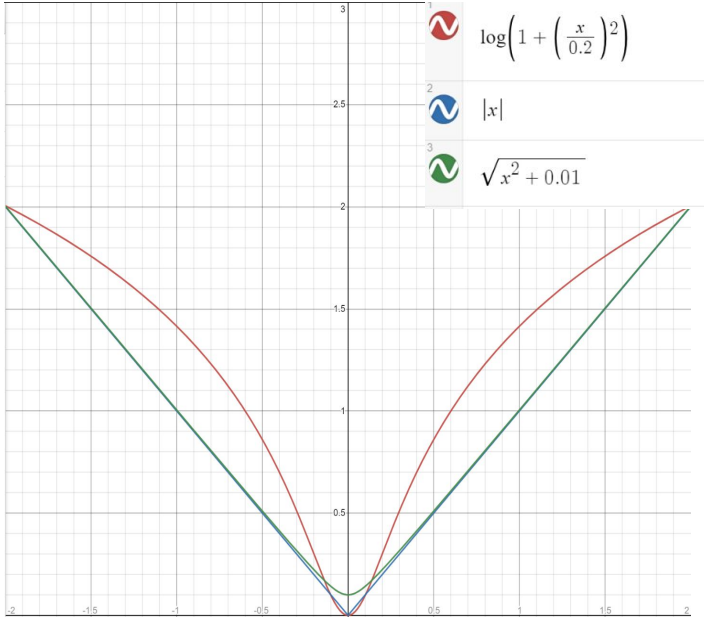
- Sparseness of coefficients
  - Minimize the sparsity penalty

$$\sum_{ij} \Psi\left(s_j^{(i)}\right)$$

# Sparsity penalty

Many choices for inducing (approximately) sparse coefficients:

$$\Psi(s) = \begin{cases} I(s \neq 0) & \mathrm{L}_0 \text{ penalty} \\ \log(1+s^2) & \log \text{ penalty} \\ |s| & \mathrm{L}_1 \text{ penalty} \\ \sqrt{s^2+\epsilon} & \text{epsilon } \mathrm{L}_1 \text{ penalty} \end{cases}$$



# Learning bases: optimization

Given input data $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$, we want to find good bases $\{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$:

$$\min_{\mathbf{b},\mathbf{s}} = \sum_i ||\mathbf{x}^{(i)} - \sum_j s_j^{(i)} \mathbf{b}_j||_2^2 + \beta \sum_i ||\mathbf{s}^{(i)}||_1$$

Reconstruction error

Sparsity penalty

$$\forall j: \ \| b_j \| \leq 1$$

Normalization constraint

Tradeoff between "quality of approximation" and "sparsity" (compactness).

# Sparse Coding Implementation

$$\min_{\mathbf{b},\mathbf{s}} = \sum_i ||\mathbf{x}^{(i)} - \sum_j s_j^{(i)} \mathbf{b}_j||_2^2 + \beta \sum_i ||\mathbf{s}^{(i)}||_1$$

Reconstruction error

Sparsity penalty

$\forall j: \ ||b_j|| \leq 1$

Normalization constraint

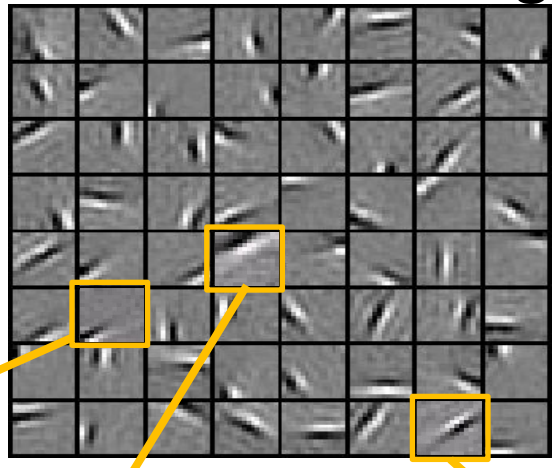
- Alternating optimization:
  - Optimize with either **b** (bases) and **s** (coefficients) while fixing others.
  - The problem is convex for each sub-problems, but not jointly convex

- Fast inference and learning algorithm
  - http://web.eecs.umich.edu/~honglak/softwares/nips06-sparsecoding.htm

# Sparse coding for images

## Natural Images

## Learned bases: “Edges”



## Test example


[0, 0, …, 0, **0.8**, 0, …, 0, **0.3**, 0, …, 0, **0.5**, …] = coefficients (feature representation)

Compact & easily interpretable

# Sparse coding: summary

- Sparse coding is a popular “dictionary learning” method in machine learning
- It can learn a large overcomplete set of bases; the coefficients are not a linear function of input.
- The coefficients can be used as features; Successful applications in object recognition, and many other tasks
- Limitation: computationally expensive; representation is unstable.

# PCA, ICA and Sparse Coding Summary

- **PCA (Principal Component Analysis)** reduces dimensionality by identifying orthogonal directions of maximum variance to efficiently represent data.
- **ICA (Independent Component Analysis)** decomposes data into statistically independent components by maximizing non-Gaussianity.
- **Sparse coding** represents data as sparse linear combinations of basis functions, aiming for efficient encoding with minimal active components.

**PCA** and **ICA** both seek linear transformations of data, but **PCA** focuses on variance maximization, while ICA prioritizes statistical independence.

**Sparse coding** extends **ICA** by not only seeking statistically independent components but explicitly enforcing a sparsity constraint to ensure most coefficients in the representation are near-zero, promoting interpretability and efficiency.

# Outline

- Independent Component Analysis
- Sparse Coding
- Multidimensional Scaling
- Isomap
- t-distributed stochastic neighbor embedding (t-SNE)
- Autoencoder

Examples of Embedding methods

# Learning Embedding

Problem definition:

- Given set X of input samples, find a mapping from input to embedding X⟶ Y such that Y reflects semantics or similarity/neighborhood structures.

$$\mathcal{X} = \left\{x^{(1)}, x^{(2)}, \ldots, x^{(N)} \in \mathbb{R}^h\right\} \rightarrow \mathcal{Y} = \left\{y^{(1)}, y^{(2)}, \ldots, y^{(N)} \in \mathbb{R}^l\right\}$$

- The embedding may be a parameterized function (e.g., y = f(x) by a neural network), or alternatively, you can directly optimize Y (i.e., all embedding coordinates) corresponding to the entire input dataset X.
- The specific formulation differs depending on the objective function.

# Learning Embedding

Examples of methods for learning embedding.

- Multidimensional Scaling
- ISOMAP
- tSNE (t-Distributed Stochastic Neighbor Embedding)
- Autoencoder

Here, we will primarily cover methods used for visualization via dimensionality reduction of the original data

# Multidimensional scaling

# Multidimensional scaling (MDS)

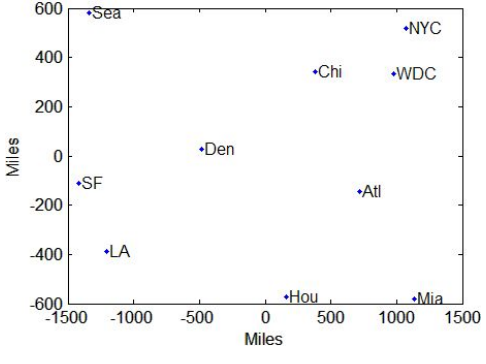
- Given pairwise distances between data points, MDS tries to "recover" the linear (Euclidean) embedding that can preserve the original distances (as much as possible).
  - i.e., Any pairwise distance -> Euclidean space
- Often used for data visualization, or dimensionality reduction
- Can be used for nonlinear dimensionality reduction (ISOMAP; covered later)

# Multidimensional scaling (MDS)

- Given pairwise distances between data points (cities in the USA), can you locate the points (cities) in a 2D map?

| | Atl | Chi | Den | Hou | LA | Mia | NYC | SF | Sea | WDC |
|---|---|---|---|---|---|---|---|---|---|---|
| Atl | 0 | 587 | 1212 | 701 | 1936 | 604 | 748 | 2139 | 2182 | 543 |
| Chi | 587 | 0 | 920 | 940 | 1745 | 1188 | 713 | 1858 | 1737 | 597 |
| Den | 1212 | 920 | 0 | 879 | 831 | 1726 | 1631 | 949 | 1021 | 1494 |
| Hou | 701 | 940 | 879 | 0 | 1374 | 968 | 1420 | 1645 | 1891 | 1220 |
| LA | 1936 | 1745 | 831 | 1374 | 0 | 2339 | 2451 | 347 | 959 | 2300 |
| Mia | 604 | 1188 | 1726 | 968 | 2339 | 0 | 1092 | 2594 | 2734 | 923 |
| NYC | 748 | 713 | 1631 | 1420 | 2451 | 1092 | 0 | 2571 | 2408 | 205 |
| SF | 2139 | 1858 | 949 | 1645 | 347 | 2594 | 2571 | 0 | 678 | 2442 |
| Sea | 2182 | 1737 | 1021 | 1891 | 959 | 2734 | 2408 | 678 | 0 | 2329 |
| WDC | 543 | 597 | 1494 | 1220 | 2300 | 923 | 205 | 2442 | 2329 | 0 |

- Desired output:



# Algorithm for MDS

- Given pairwise distances *D* (note: x is unknown)

$$\mathbf{D}_{ij} = \left\| \mathbf{x}^{(i)} - \mathbf{x}^{(j)} \right\|^2$$

- Compute pairwise inner products (Gram matrix)

$$\mathbf{B} = -\frac{1}{2}\mathbf{H}\mathbf{D}\mathbf{H}$$

$$\text{where } \mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$$

**1**: a Nx1 vector of all ones
**1** $\mathbf{1}^\top$: outer product of **1** and $\mathbf{1}^\top$, which is a NxN matrix of all ones

  - Then it can be shown that $\mathbf{B} = \mathbf{X}^\top\mathbf{X}$. where $\mathbf{X} : N_{\text{dim}} \times N_{\text{examples}}$

- Embed with *Y* (i.e. approximate $\mathbf{B} \approx \mathbf{Y}^\top\mathbf{Y}$)
  - SVD: $\mathbf{B} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$

- Solution: $\mathbf{Y} = \sqrt{\mathbf{\Lambda}}\mathbf{V}^\top$
  - Often truncate with top-K eigenvectors/eigenvalues

# Detailed Derivation for MDS

$$HDH = \left(I - \frac{1}{N}11^T\right) D \left(I - \frac{1}{N}11^T\right) = D - \frac{1}{N}11^T D - D\frac{1}{N}11^T + \frac{1}{N}11^T D\frac{1}{N}11^T$$

$$\begin{aligned}
(HDH)_{ij} &= D_{ij} - \frac{1}{N}\left(1^T D\right)_j - \frac{1}{N}\left(D1\right)_i + \frac{1}{N^2}\sum_{kl} D_{kl} \\
&= \left(||x^{(i)}||^2 + ||x^{(j)}||^2 - 2x^{(i)T}x^{(j)}\right) \\
&\quad -\frac{1}{N}\sum_i \left(||x^{(i)}||^2 + ||x^{(j)}||^2 - 2x^{(i)T}x^{(j)}\right) - \frac{1}{N}\sum_j \left(||x^{(i)}||^2 + ||x^{(j)}||^2 - 2x^{(i)T}x^{(j)}\right) \\
&\quad +\frac{1}{N^2}\sum_{ij} \left(||x^{(i)}||^2 + ||x^{(j)}||^2 - 2x^{(i)T}x^{(j)}\right) \\
&= \left(||x^{(i)}||^2 + ||x^{(j)}||^2 - 2x^{(i)T}x^{(j)}\right) - \frac{1}{N}\sum_i \left(||x^{(i)}||^2 + ||x^{(j)}||^2\right) - \frac{1}{N}\sum_j \left(||x^{(i)}||^2 + ||x^{(j)}||^2\right) \\
&\quad +\frac{1}{N^2}\sum_{ij} \left(||x^{(i)}||^2 + ||x^{(j)}||^2\right) \\
&= ||x^{(i)}||^2 + ||x^{(j)}||^2 - 2x^{(i)T}x^{(j)} - \frac{2}{N}\sum_i ||x^{(i)}||^2 - ||x^{(j)}||^2 - ||x^{(i)}||^2 + \frac{2}{N}\sum_i ||x^{(i)}||^2 \\
&= -2x^{(i)T}x^{(j)}
\end{aligned}$$

# Summary: MDS

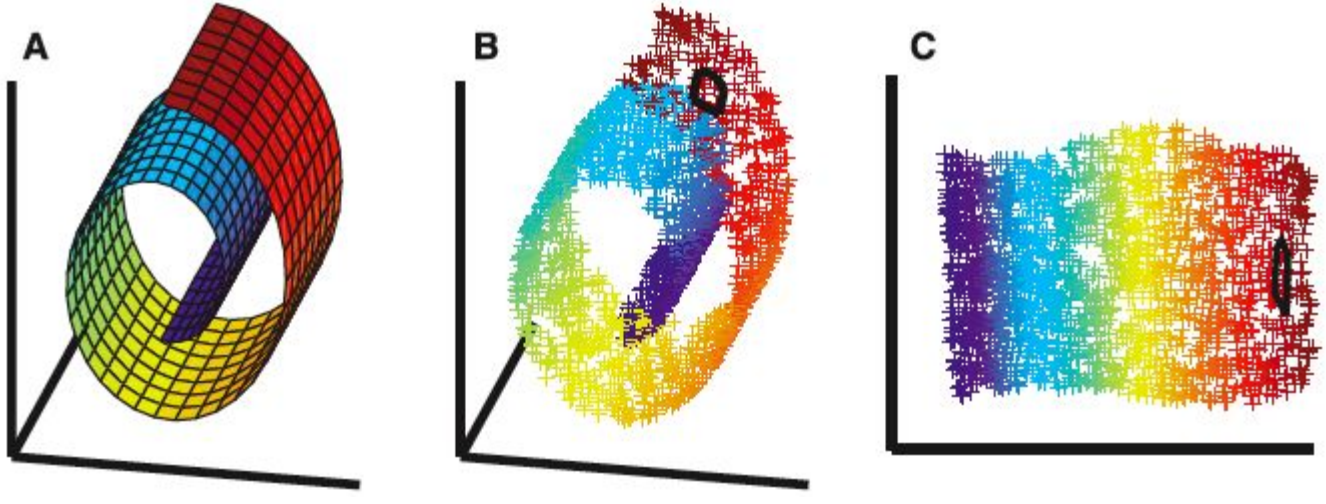
- MDS is a “linear” dimensionality reduction method
  - If the original distance is defined over the Euclidean space, then MDS can at best recover the original space (or approximation with reduced dimensionality)

- However, MDS can be combined with nonlinear distance metric to discover “manifold” structure in the data (ISOMAP)

# ISOMAP

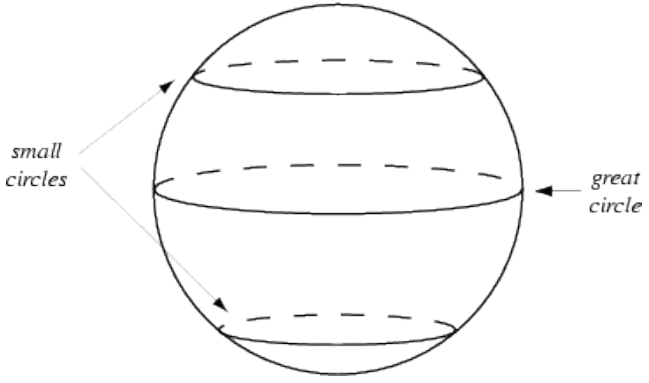
# Nonlinear Manifolds

- PCA fails on seriously nonlinear manifolds like the “Swiss roll”

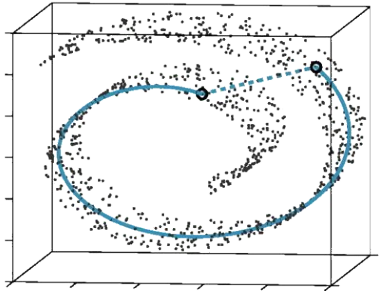


# Isometric Feature Mapping (ISOMAP)

- Geodesic: the shortest curve on a manifold that connects two points on the manifold
  - e.g. on a sphere, geodesics are great circles



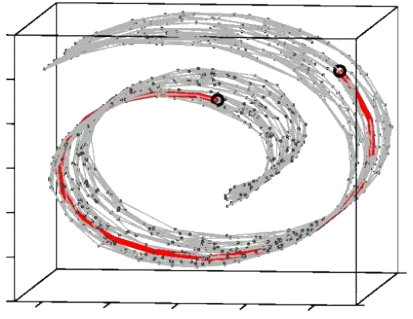

- Geodesic distance: length of the geodesic



- Points far apart measured by geodesic dist. appear close measured by Euclidean dist.

# ISOMAP

- Take a distance matrix as input
- Construct a weighted graph G based on neighborhood relations
- Estimate pairwise geodesic distance by “a sequence of short hops” on G
- Apply MDS to the geodesic distance matrix
  - MDS “unfolds” the manifold into Euclidean space

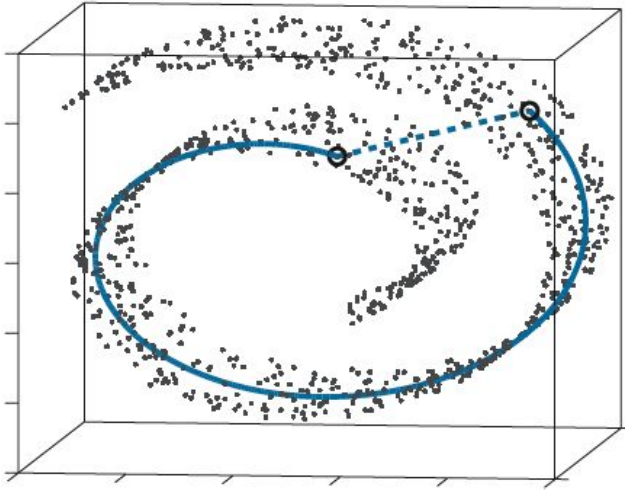


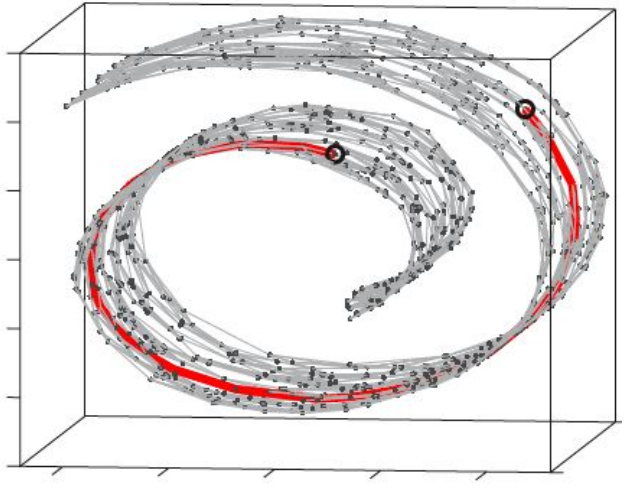
# Isomap

- Define local neighborhoods by small distance or *k*-nearest-neighbors.
  - Link each pair of points in a neighborhood with their distance in the neighborhood.
- Distance between all other pairs is shortest path distance in the connectivity graph.
- Compute eigenvalues; select dimension.
- Do multidimensional scaling into desired low-dimensional space.

# Isomap: the Swiss roll

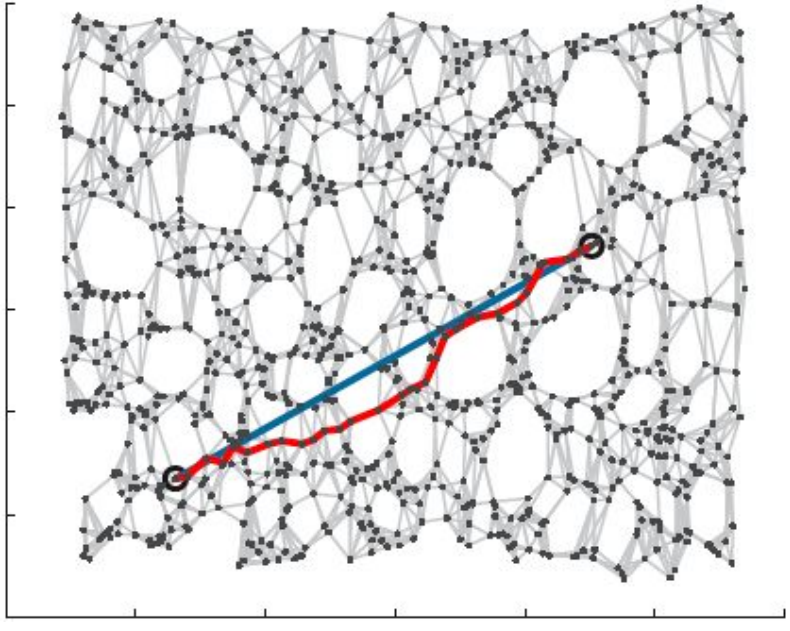
- Distance may be longer along a geodesic in the manifold than in the embedding space.





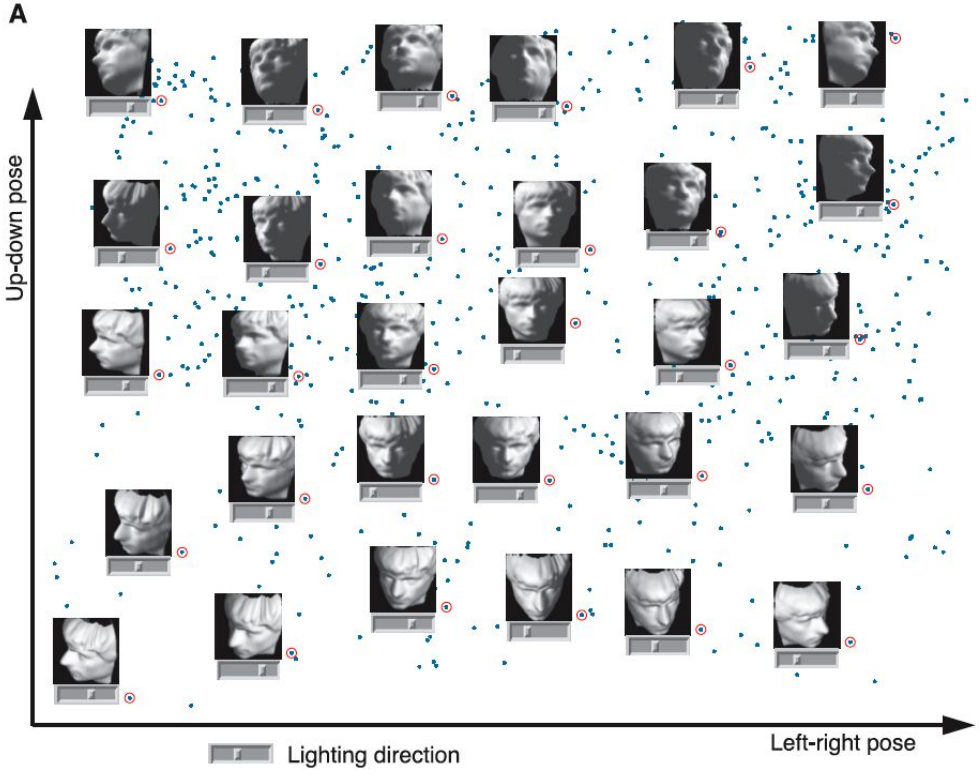
# Unrolling the Swiss Roll

- The resulting 2D structure reflects the geodesic distances along the manifold.



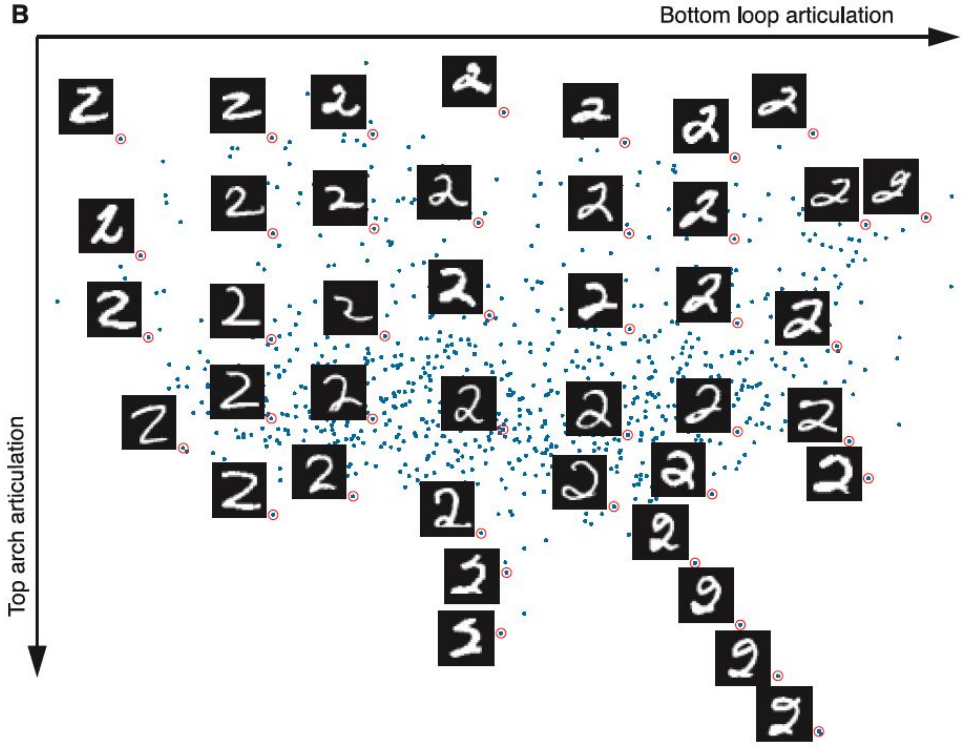
# Faces: pose x illumination

- 64x64 = 4096 dimensions



# Hand-written “2”s

- Mapping groups the digits by “style”



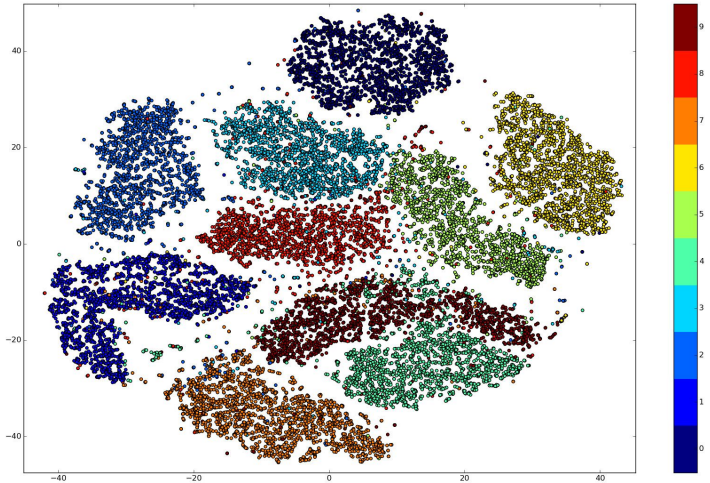
# Summary: ISOMAP

- Nonlinear dimensionality reduction methods can be used to find intrinsic manifold structure from the data; also useful for visualization
- Limitation:
  - computationally quite expensive; doesn’t scale to very large data
  - Requires dense data points for good estimates
  - Sensitive to noise
- Other related algorithms:
  - Hessian LLE, Laplacian eigenmap, etc.

# tSNE (t-Distributed Stochastic Neighbor Embedding)

# Data visualization with embeddings

- Embeddings are vector representations that reflect semantic meaning in feature space (i.e., feature vectors live in neighborhoods based on meaning)
- We can visualize these with techniques such as t-SNE



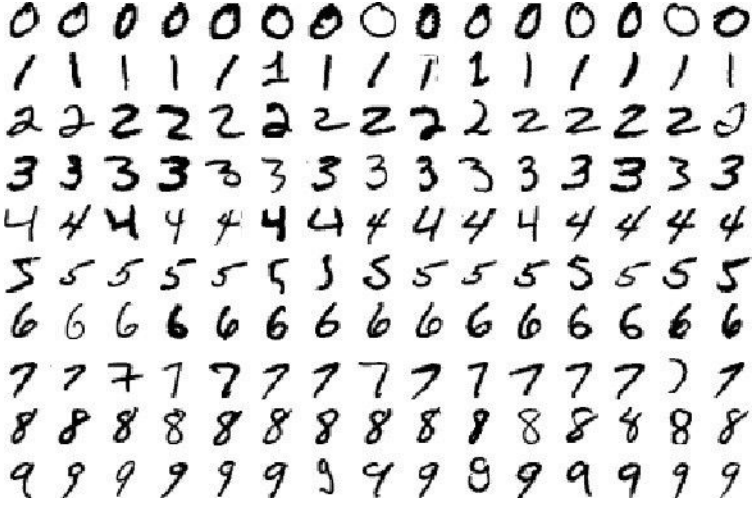

tSNE visualization of handwritten digits (MNIST) in 2d space.

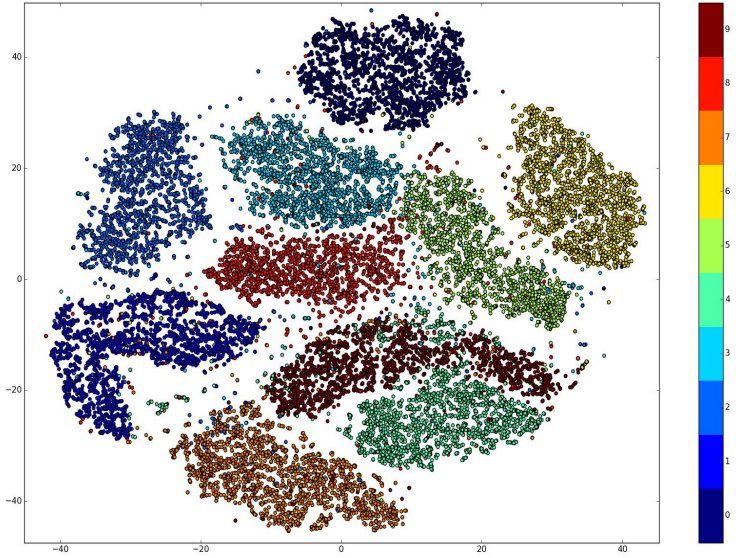


Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, *9*(Nov), 2579-2605.

# t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Given a collection of N high-dimensional data $x^{(1)}$, $x^{(2)}$, … $x^{(N)}$, how can we get a sense of how they are arranged in data space?



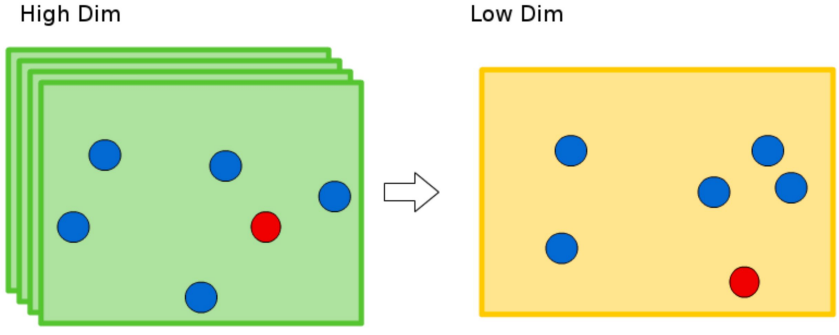





Slide credit: Kai-Wen Zhao

# t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Compress the high dimensional data into a lower dimensional space
- We want to preserve distance and neighborhood structure






Slide credit: Kai-Wen Zhao

# Preliminary: Stochastic Neighbor Embedding (SNE)

SNE converts euclidean distances to similarities, that can be interpreted as probabilities $P^i$ over neighbors of $x^i$. Then we find embedding $Q^i$ that approximates $P^i$.

$P^i = \{p^{1|i}, p^{2|i}, ..., p^{N|i}\}$ and $Q^i = \{q^{1|i}, q^{2|i}, ..., q^{N|i}\}$ are the distributions on the neighbors (e.g., transition prob.) of datapoint *i*.

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

$$q^{j|i} = \frac{\exp\left(-\left\|y^{(i)} - y^{(j)}\right\|^2\right)}{\sum_{k \neq i} \exp\left(-\left\|y^{(i)} - y^{(k)}\right\|^2\right)}$$

$$p^{i|i} = 0, \; q^{i|i} = 0$$

SNE minimizes the following cost:

$$C = \sum_i KL\left(P^i \| Q^i\right) = \sum_i \sum_j p^{j|i} \log \frac{p^{j|i}}{q^{j|i}}$$



Slide credit: Simon Carbonnelle

# t-Distributed Stochastic Neighbor Embedding (t-SNE)

## SNE

### Modelisation:

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

$$q^{j|i} = \frac{\exp\left(-\left\|y^{(i)} - y^{(j)}\right\|^2\right)}{\sum_{k \neq i} \exp\left(-\left\|y^{(i)} - y^{(k)}\right\|^2\right)}$$

### Cost Function:

$C = \sum_i KL(P_i||Q_i)$

### Derivatives:

$$\frac{dC}{dy} = 2\sum_j \left(p^{j|i} - q^{j|i} + p^{ij} - q^{ij}\right)\left(y^{(i)} - y^{(j)}\right)$$

# t-Distributed Stochastic Neighbor Embedding (t-SNE)

## SNE

### Modelisation:

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

$$q^{j|i} = \frac{\exp\left(-\left\|y^{(i)} - y^{(j)}\right\|^2\right)}{\sum_{k \neq i} \exp\left(-\left\|y^{(i)} - y^{(k)}\right\|^2\right)}$$

### Cost Function:

$$C = \sum_i KL(P_i||Q_i)$$

### Derivatives:

$$\frac{dC}{dy} = 2\sum_j \left(p^{j|i} - q^{j|i} + p^{ij} - q^{ij}\right)\left(y^{(i)} - y^{(j)}\right)$$

## Symmetric SNE

### Modelisation:

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

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

### Cost Function:

$$C = KL(P||Q)$$

### Derivatives:

$$\frac{dC}{dy} = 4\sum_j \left(p^{ij} - q^{ij}\right)\left(y^{(i)} - y^{(j)}\right)$$

→ Faster optimization!

# t-Distributed Stochastic Neighbor Embedding (t-SNE)

## SNE

### Modelisation:

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

$$q^{j|i} = \frac{\exp\left(-\left\|y^{(i)} - y^{(j)}\right\|^2\right)}{\sum_{k \neq i} \exp\left(-\left\|y^{(i)} - y^{(k)}\right\|^2\right)}$$

### Cost Function:

$$C = \sum_i KL(P_i||Q_i)$$

### Derivatives:

$$\frac{dC}{dy} = 2\sum_j \left(p^{j|i} - q^{j|i} + p^{ij} - q^{ij}\right)\left(y^{(i)} - y^{(j)}\right)$$

## Symmetric SNE

### Modelisation:

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

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

### Cost Function:

$$C = KL(P||Q)$$

### Derivatives:

$$\frac{dC}{dy} = 4\sum_j \left(p^{ij} - q^{ij}\right)\left(y^{(i)} - y^{(j)}\right)$$

→ Faster optimization!

## t-SNE

### Modelisation:

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

$$q^{ij} = \frac{\left(1 + \left\|y^{(i)} - y^{(j)}\right\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \left\|y^{(k)} - y^{(l)}\right\|^2\right)^{-1}}$$

### Cost Function:

$$C = KL(P||Q)$$

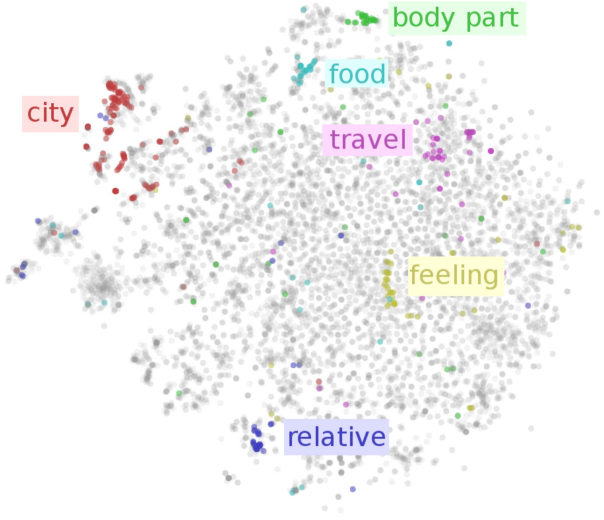
### Derivatives:

$$\frac{dC}{dy^{(i)}} = 4\sum_j \left(p^{ij} - q^{ij}\right)\left(y^{(i)} - y^{(j)}\right)\left(1 + \left\|y^{(i)} - y^{(j)}\right\|^2\right)^{-1}$$

→ Improved robustness
(to noise/uncertainties)

# Embedding Methods Basics

**Using t-SNE to explore word embeddings**

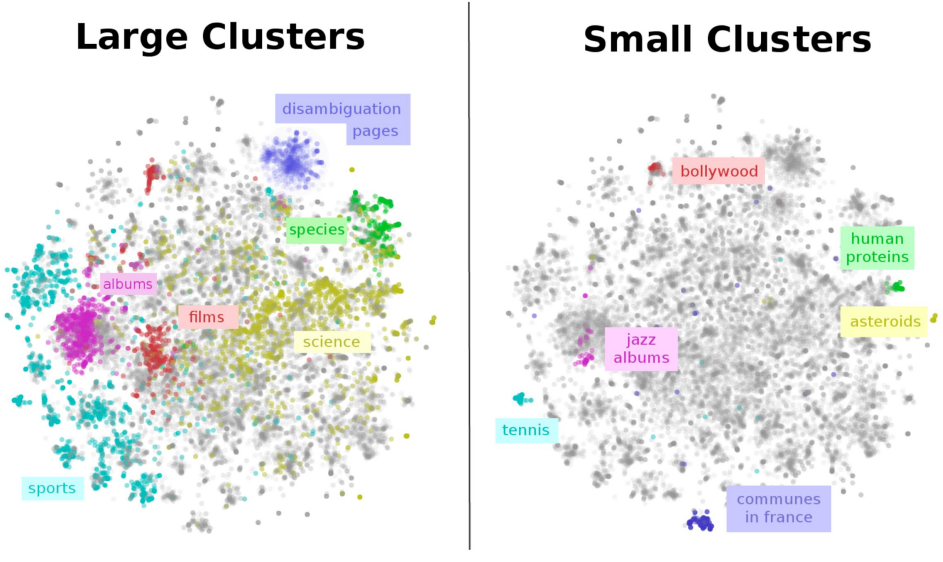





Slide credit: Simon Carbonelle

# Embedding Methods Basics

**Using t-SNE to explore Wikipedia article embeddings**



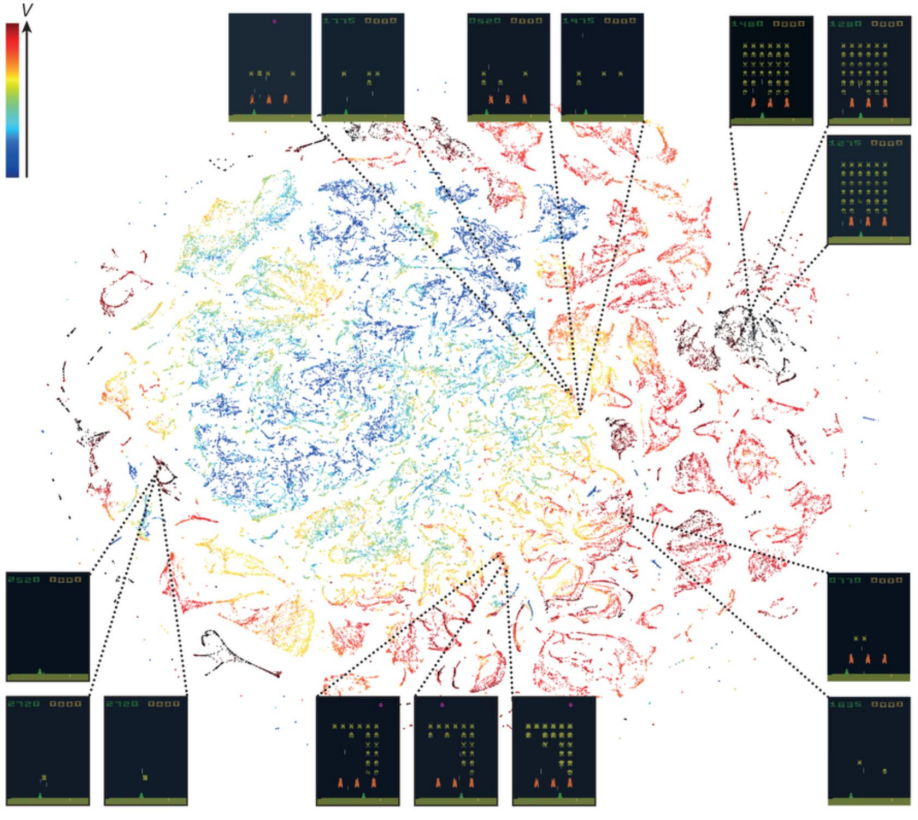



Slide credit: Simon Carbonelle

# Embedding Methods Basics

**Using t-SNE to explore game state representations**

- DQN network that learns to play video games learns embeddings that group game states that look similar



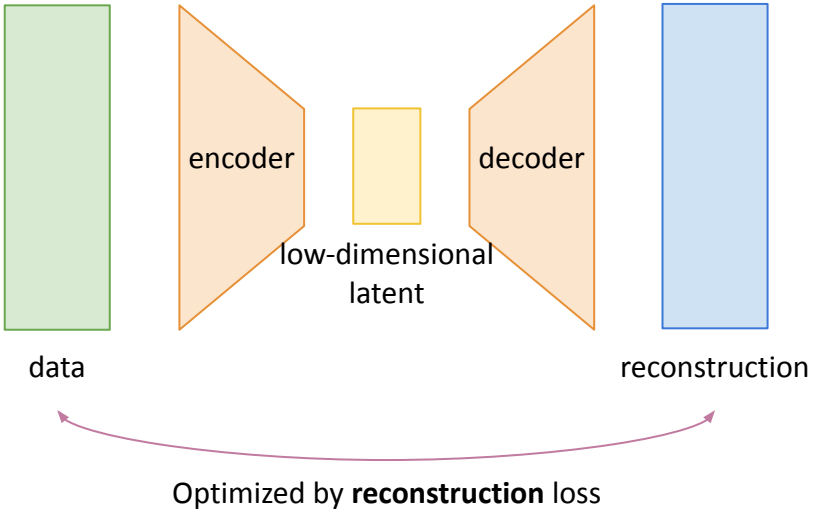

Human-level control through deep reinforcement learning, V. Mnih et Al. (Nature,2015)

Slide credit: Simon Carbonelle

# Autoencoder

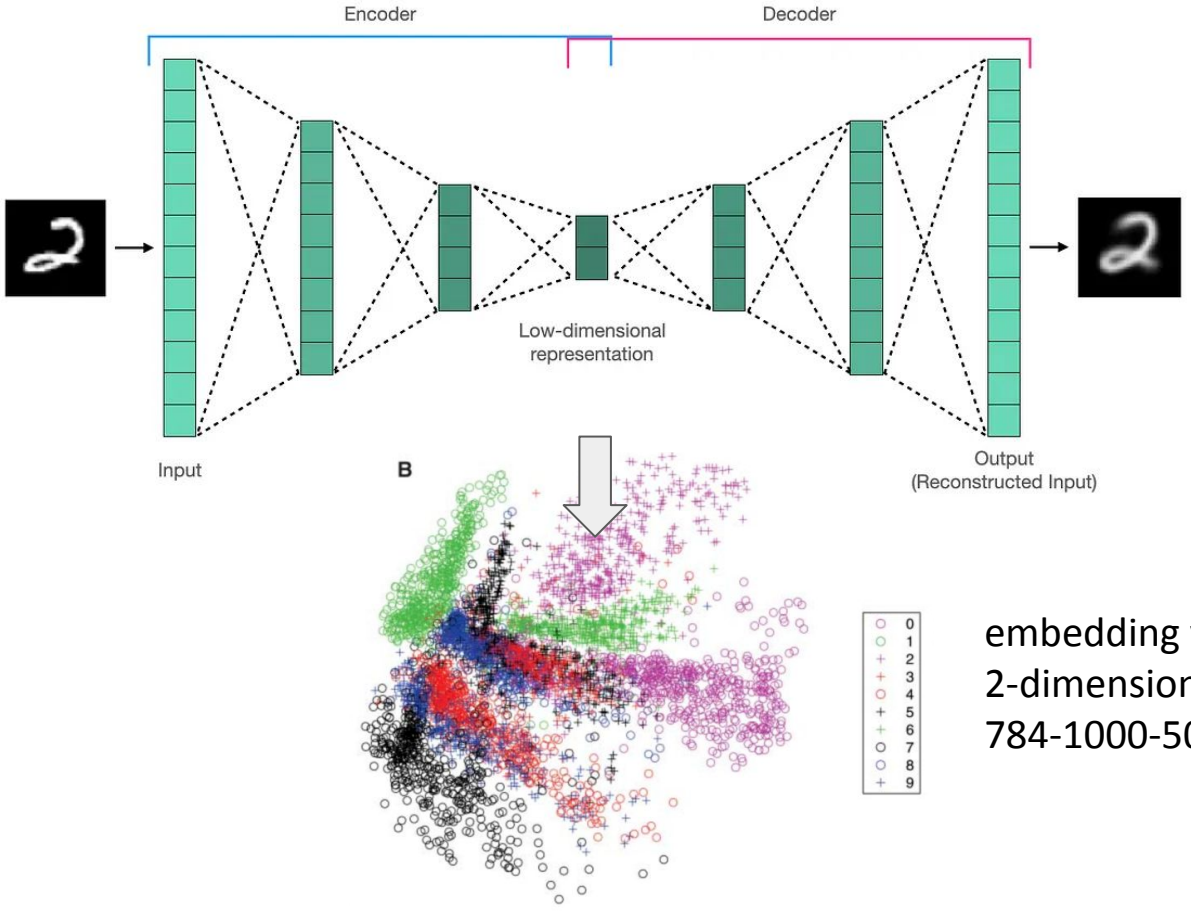
# Autoencoders

- Compress the data and learn data reconstruction.
  - Reconstruction loss can take L2 loss (for real-valued inputs) or cross-entropy loss (for binary inputs)
- The learned features reduce dimension, the reconstruction will not be exactly the same as inputs.




Problem: Autoencoder may overfit if # parameters > # data

# Learning Autoencoder with handwritten digits

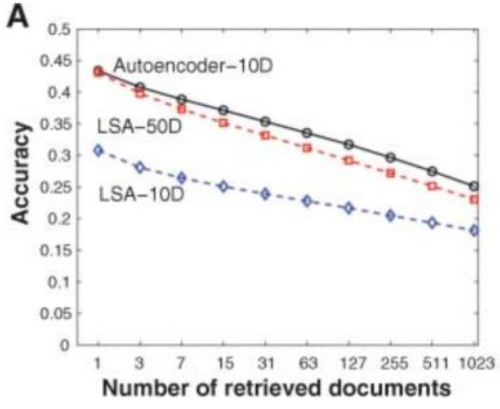


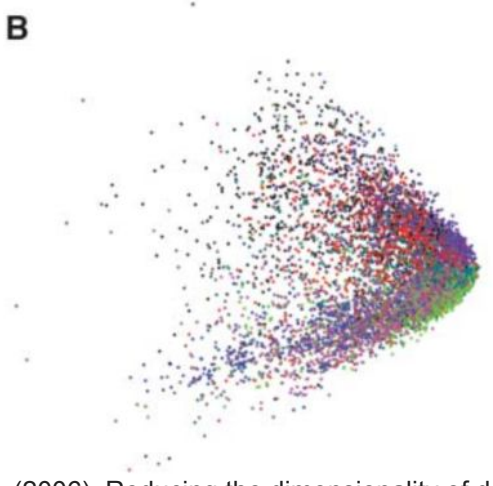

embedding visualization:
2-dimensional codes found by a
784-1000-500-250-2 autoencoder

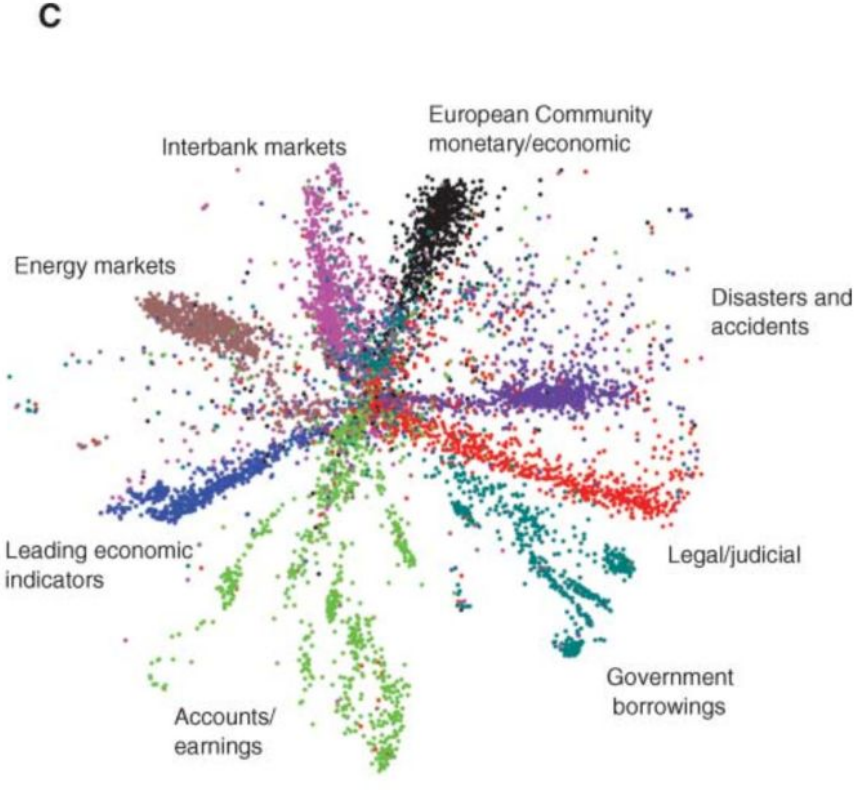
Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504-507.

# Learning Autoencoder with documents

**Fig. 4.** (**A**) The fraction of retrieved documents in the same class as the query when a query document from the test set is used to retrieve other test set documents, averaged over all 402,207 possible queries. (**B**) The codes produced by two-dimensional LSA. (**C**) The codes produced by a 2000-500-250-125-2 autoencoder.



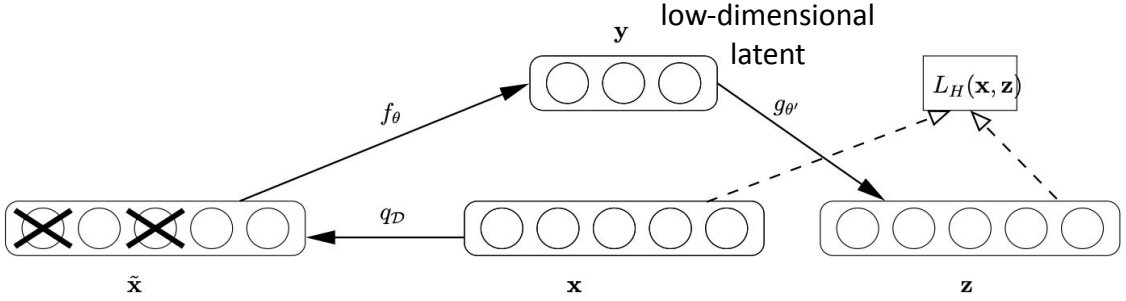




Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504-507.

# Denoising autoencoders (DAEs)

- Problem: Autoencoders can memorize training data too closely, reducing their effectiveness on unseen data.
- Denoising: reconstruct from **corrupted**, partially destroyed data
  - Binary masking noise: masked a part of the component to 0
  - Additive isotropic Gaussian noise: $\widetilde{x} = x + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I)$




- DAE can be used as building blocks for deep neural networks
  - greedy layer-wise pre-training, followed by fine-tuning with task objective

Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *Proceedings of the 25th international conference on Machine learning*. 2008.

# Next

- Deep Generative Models
  - Autoencoders/Variational autoencoders
  - Generative adversarial networks
  - Autoregressive models

# Any feedback (about lecture, slide, homework, project, etc.)?

(via **anonymous** google form: https://forms.gle/fpYmiBtG9Me5qbP37)

Change Log of lecture slides:
https://docs.google.com/document/d/e/2PACX-1vSSIHJjkIypK7rKFSR1-5GYXyBCEW8UPtpSfCR9AR6M1l7K9ZQEmxfFwaWaW7kLDxusthsF8WlCyZJ-/pub