16. Dimensionality Reduction
STA3142 Statistical Machine Learning

Kibok Lee
Assistant Professor of
Applied Statistics / Statistics and Data Science

{Apr 30, May 2}, 2024

Announcement

- No class @ week 10 (May 7, 9)

- No class & no final exam @ week 16
  - Assignment 5 is the replacement
  - You should submit A5 for your attendance @ week 16

Midterm Grading

- Ongoing; we are trying to release it this week

- If you don't agree with the Honor code – your midterm score is 0.
  - If you didn't write the pledge and your name on the first page properly, you receive 0 point.
  - If you did so, your submission will be graded after you complete it.
  - Your academic career is built on academic honesty.

Post-Midterm

- Let's solve some questions that you felt difficult.
- A survey will be out together with midterm results.
  - To determine questions we are going to solve together

- If you feel you didn't do well,
  - You are not alone; other students would too.
  - Problem-solving skills can be improved by practice.
    - E.g., Derive ML algorithms we have learned from scratch
    - Don't just memorize them

Assignment 3

- Due **Friday 5/3, 11:59pm**
- Topics
  - (Programming) K-Nearest Neighbors
  - (Math) MLE vs. MAP
  - (Math) Kernel Methods
  - (Math/Programming) SVM Primal
- Please read the instruction carefully!
  - Submit one <u>pdf</u> and one <u>zip</u> file separately
  - Write your code only in the designated spaces
  - Do not import additional libraries
  - …
- If you feel difficult, consider to take **option 2**.

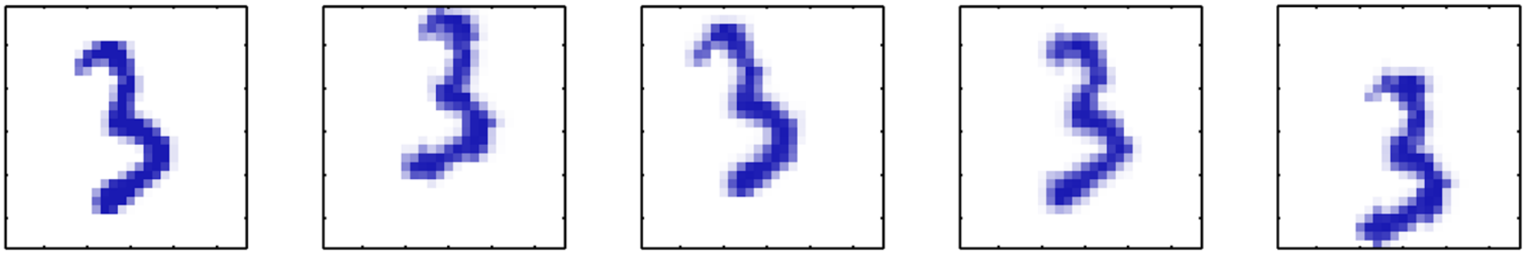
Outline

- Principal component analysis (PCA)
  - vs. Fisher's Linear Discriminant (FLD)
  - Kernel PCA
- Independent Component Analysis (ICA)
  - Maximum Likelihood Estimation
  - Maximizing Non-Gaussianity
- t-Distributed Stochastic Neighbor Embedding (t-SNE)

Principal Component Analysis

Dimensionality Reduction

- High-dimensional data may have a low-dimensional structure.
- E.g., Digit images below consist of 28x28 pixels.



- Only 3 degrees of freedom in this example: horizontal, vertical translations and rotations.
- All **variabilities** can be represented with 3 numbers with a nonlinear mapping: $[0,1]^{28\times 28} \to (t_x, t_y, r)$

Principal Component Analysis

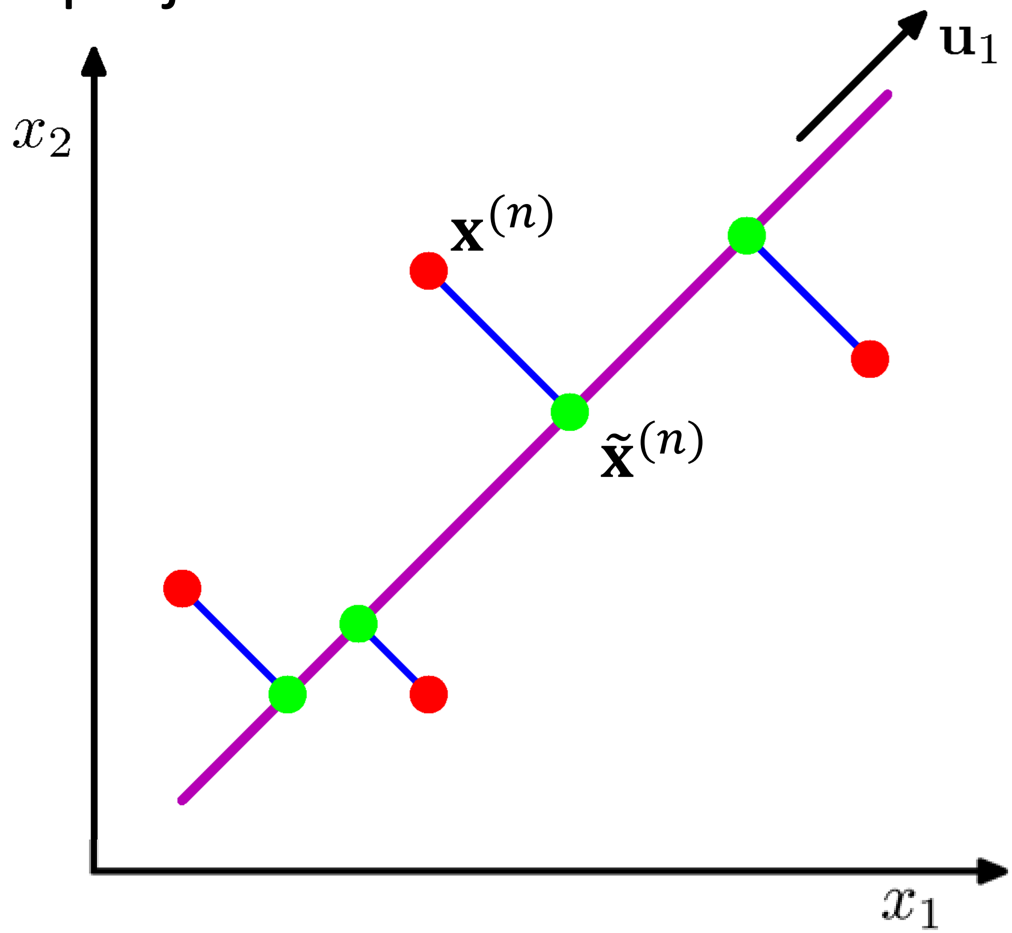
- Given a set of data $\{\mathbf{x}^{(n)}\}_{n=1}^{N}$ in a D-dimensional space, (D can be arbitrarily large)

- Find a subspace of dimension $M < D$ that captures most of its **variability**

- PCA can be described as either:
  - Maximizing the **variance** of the projection
  - Minimizing the **squared approximation error**

PCA: Two Different Intuitions

- Approximate with the projection:

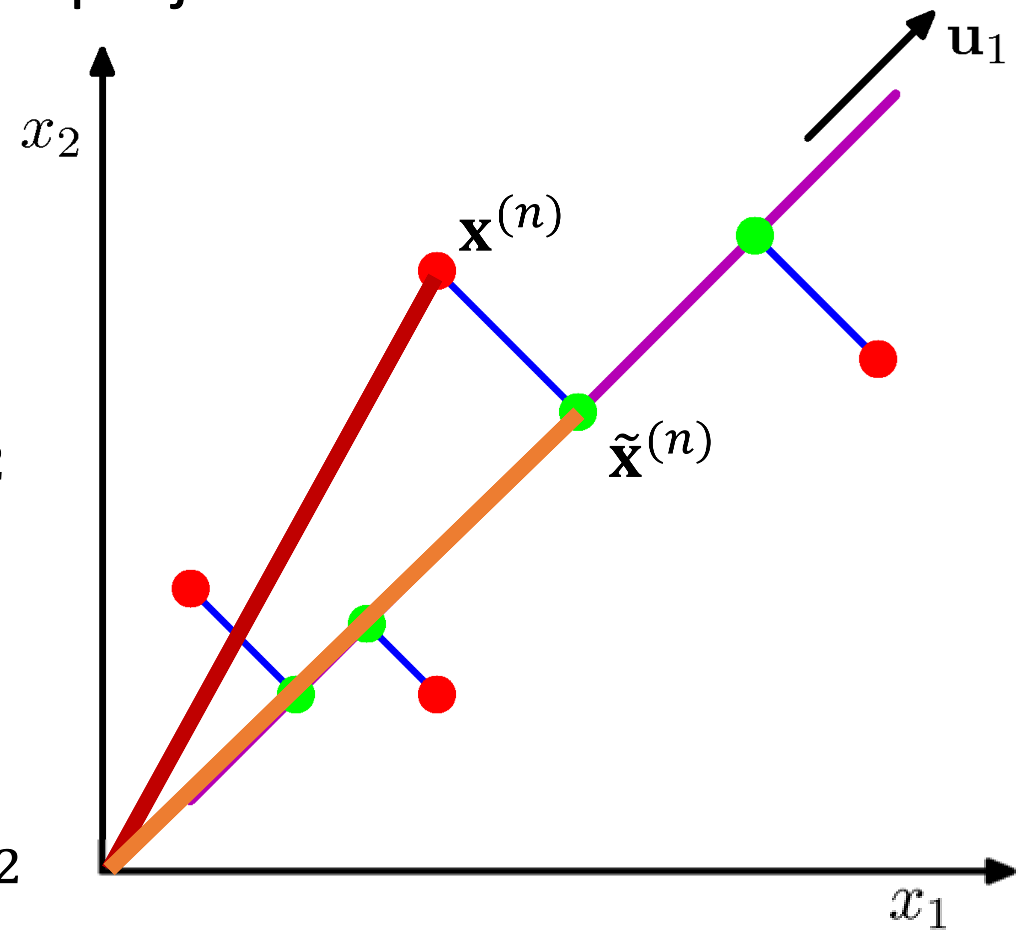


- Maximize the **variance**

- Minimize the **squared error**

PCA: Two Different Intuitions

- Approximate with the projection:
- With a **constraint** $\sum_n \left(\mathbf{x}^{(n)}\right)^2 = const$
- Maximize the **variance**: $\sum_n \left(\tilde{\mathbf{x}}^{(n)}\right)^2$
- Minimize the **squared error**: $\sum_{n=1}^{N} \left(\mathbf{x}^{(n)} - \tilde{\mathbf{x}}^{(n)}\right)^2$



PCA: Formulation

- Given a set of data $\{\mathbf{x}^{(n)}\}_{n=1}^{N}$ where $\mathbf{x}^{(n)} \in \mathbb{R}^D$,
  - Sample mean:
$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}^{(n)} \in \mathbb{R}^D$$
  - Data covariance:
$$S = \frac{1}{N}\sum_{n=1}^{N} \left(\mathbf{x}^{(n)} - \bar{\mathbf{x}}\right)\left(\mathbf{x}^{(n)} - \bar{\mathbf{x}}\right)^T \in \mathbb{R}^{D\times D}$$
- Let $\mathbf{u}_1$ be the first principal component.
  - Length 1: $\mathbf{u}_1^T\mathbf{u}_1 = 1$
  - Projection of $\mathbf{x}^{(n)}$ onto $\mathbf{u}_1$: $\mathbf{u}_1^T\mathbf{x}^{(n)}$

PCA: Derivation

- Maximize the variance on the projected space:

$$\frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{u}_1^T\mathbf{x}^{(n)} - \mathbf{u}_1^T\bar{\mathbf{x}}\right)^2 = \mathbf{u}_1^T S\mathbf{u}_1$$

  - Subject to $\mathbf{u}_1^T\mathbf{u}_1 = 1$
- Using the Lagrange multiplier:

$$\max_{\mathbf{u}_1} \mathbf{u}_1^T S\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^T\mathbf{u}_1)$$

- Derivative is zero when $S\mathbf{u}_1 = \lambda_1\mathbf{u}_1$
  - i.e., the maximum variance is the largest eigenvalue $\mathbf{u}_1^T S\mathbf{u}_1 = \lambda_1$, and $\mathbf{u}_1$ is the largest eigenvector.

PCA: Derivation

- Repeat to find the M eigenvectors of the data covariance matrix S corresponding to the M **largest eigenvalues** (power iteration)
- Or, perform eigenvalue decomposition $S = U\Lambda U^T$, and then take the first M eigenvectors

- We can arrive at the same result by minimizing the sum of the **squared error** of the projection.

PCA: Two Different Formulations

- Maximizing the **variance**

$$\max_{U} J_1(U) = \mathrm{tr}(U^T S U)$$

- Equivalent to minimizing the **squared error**

$$\min_{U} J_2(U) = \frac{1}{N} \sum_{n=1}^{N} \left\| \mathbf{x}^{(n)} - U U^T \mathbf{x}^{(n)} \right\|^2$$

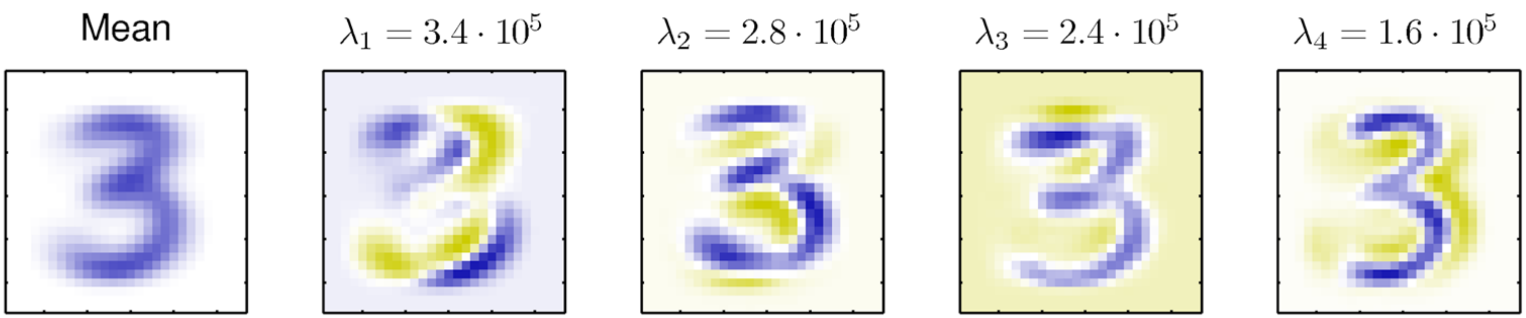
- Cf. Fisher's linear discriminant (FLD)

$$\max_{W} J_{\mathrm{FLD}}(W) = \mathrm{tr}\big((W^T S W)^{-1} (W^T S_B W)\big)$$

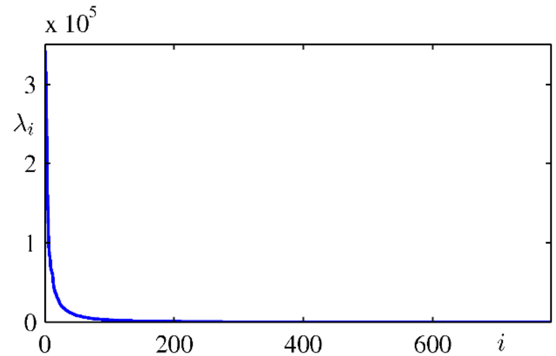
  - Supervised learning; S_B requires supervision (labels)

PCA Example: Digit Image

- The mean and first four PCA eigenvectors.



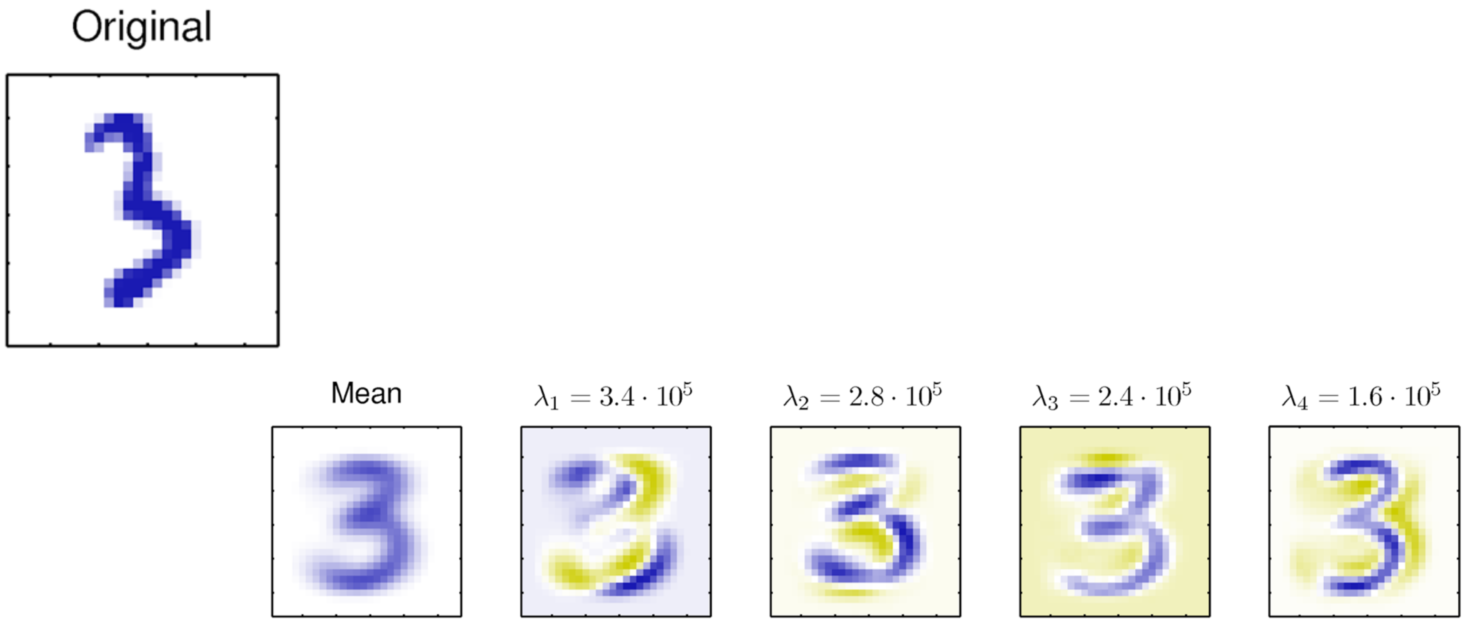
- The eigenvalue spectrum:
  - We do not need many eigenvectors to approximate images.



(a)

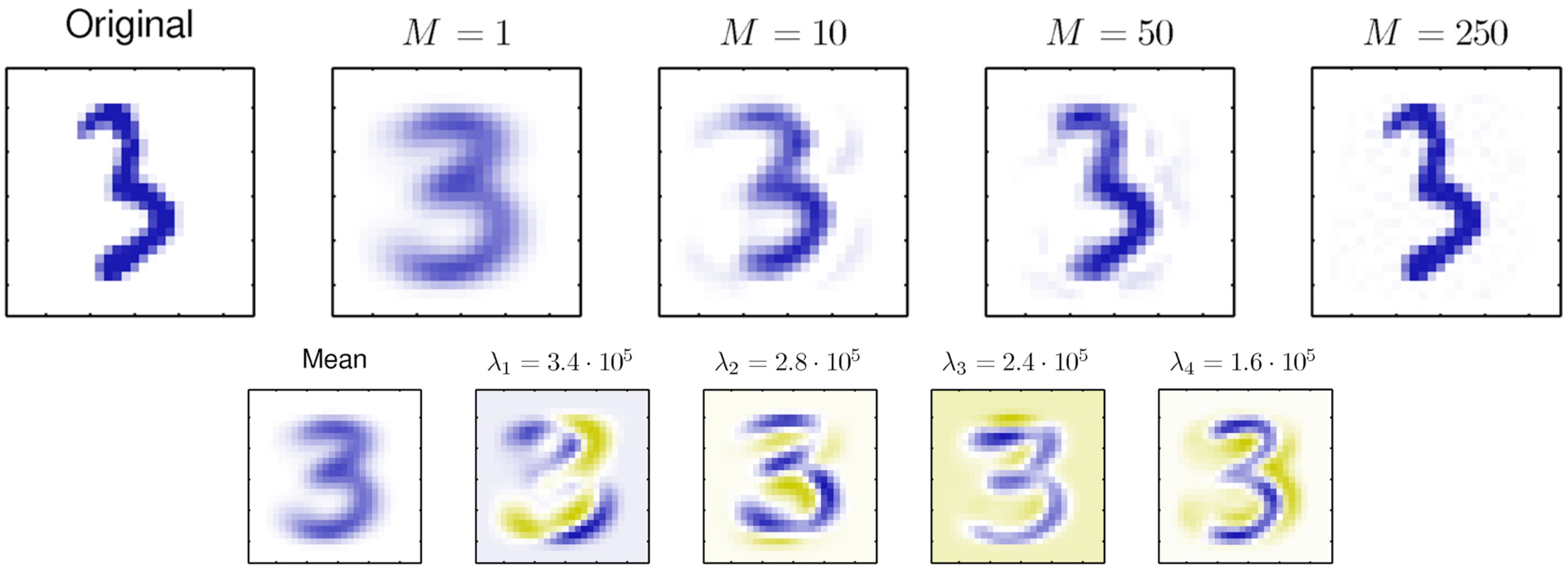
PCA Example: Digit Image

- Compress the image representation by using the first M eigenvectors and discarding the less important information.



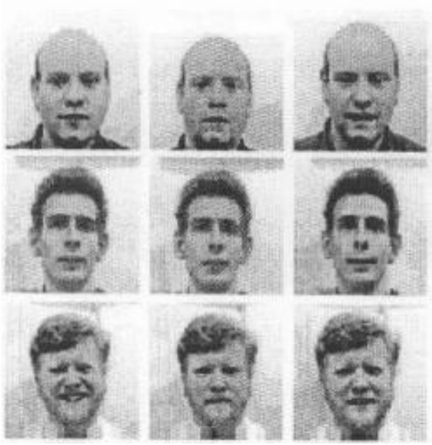
PCA Example: Digit Image

- Compress the image representation by using the first M eigenvectors and discarding the less important information.



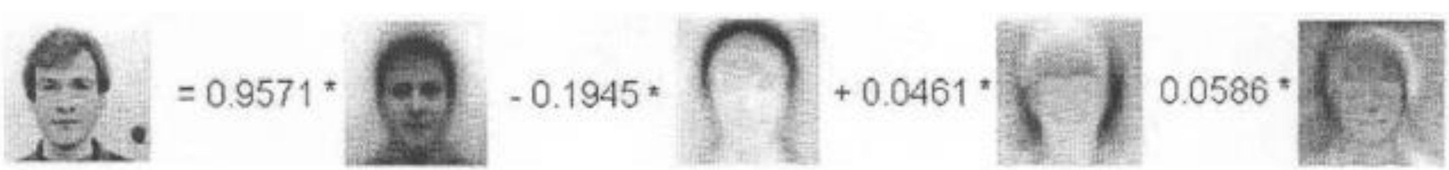
PCA Example: Eigenfaces

Training face images



Learned PCA bases

Test example



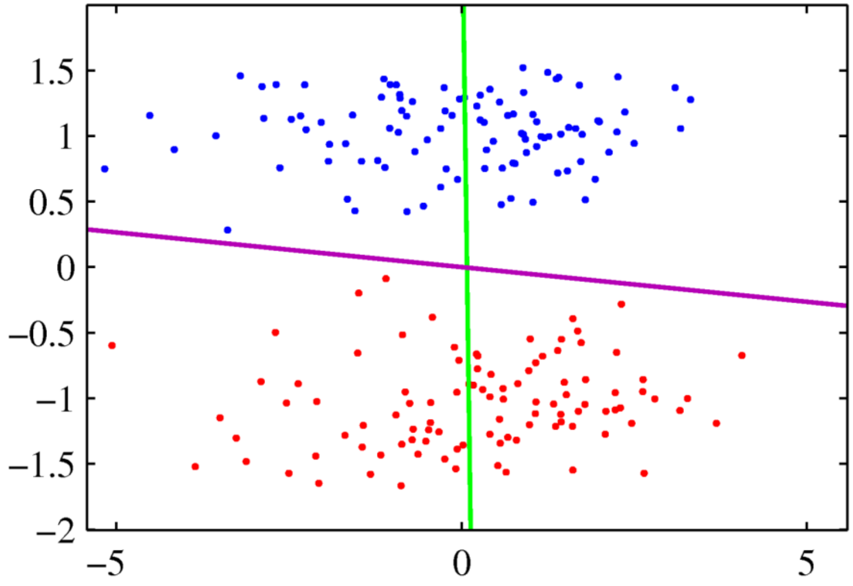
Images from www.cse.unr.edu/~bebis/CS485/Lectures/Eigenfaces.ppt

PCA: Limitations

- Maximizing the variance is not always the best way to make the structure visible.
- Comparison with Fisher's linear discriminant (FLD)

FLD gives better class separation

PCA leads to strong class overlap



PCA vs. FLD

- **Principal component analysis (PCA)**
  - Unsupervised learning
  - Learning objective:
$$\max_U J_{\mathrm{PCA}}(U) = \mathrm{tr}(U^T S U)$$
  - Solved by eigenvalue decomposition
$$SU = U\Lambda$$
- **Fisher's linear discriminant (FLD)**
  - Supervised learning
  - Learning objective:
$$\max_W J_{\mathrm{FLD}}(W) = \mathrm{tr}\big((W^T S W)^{-1}(W^T S_B W)\big)$$
  - Solved by generalized eigenvalue decomposition
$$S_B W = S W \Lambda$$

PCA vs. FLD

- **Principal component analysis (PCA)**
  - Unsupervised learning
  - Learning objective:
    $$\max_U J_{\text{PCA}}(U) = \text{tr}(U^T S U)$$
  - Reduced dimension M: $1 \leq M \leq \boxed{D}$
    - By taking the first M largest eigenvectors

Dimension of input data

- **Fisher's linear discriminant (FLD)**
  - Supervised learning
  - Learning objective:
    $$\max_W J_{\text{FLD}}(W) = \text{tr}\left((W^T S W)^{-1}(W^T S_B W)\right)$$
  - Reduced dimension M: $1 \leq M \leq \text{rank}(S_B) \leq \boxed{K} - 1$
    - By taking the first M largest eigenvectors

Number of classes

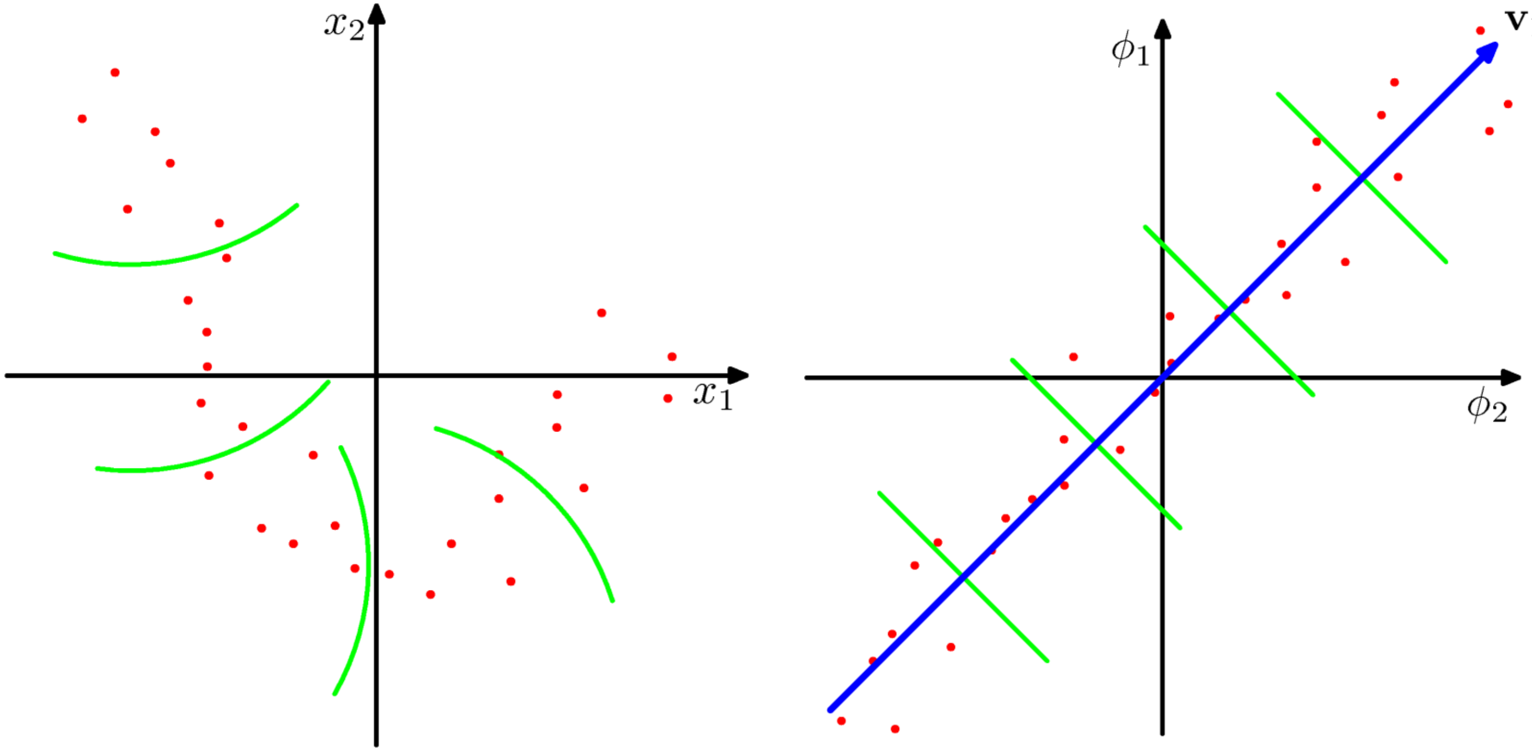
PCA vs. FLD

- **Principal component analysis (PCA)**
  - Unsupervised learning
  - Learning objective:
$$\max_U J_{\mathrm{PCA}}(U) = \mathrm{tr}(U^T S U)$$
  - **Can be kernelized**

- **Fisher's linear discriminant (FLD)**
  - Supervised learning
  - Learning objective:
$$\max_W J_{\mathrm{FLD}}(W) = \mathrm{tr}\left((W^T S W)^{-1}(W^T S_B W)\right)$$
  - **Can be kernelized**

Kernel PCA

- Suppose the regularity that allows dimensionality reduction is nonlinear.

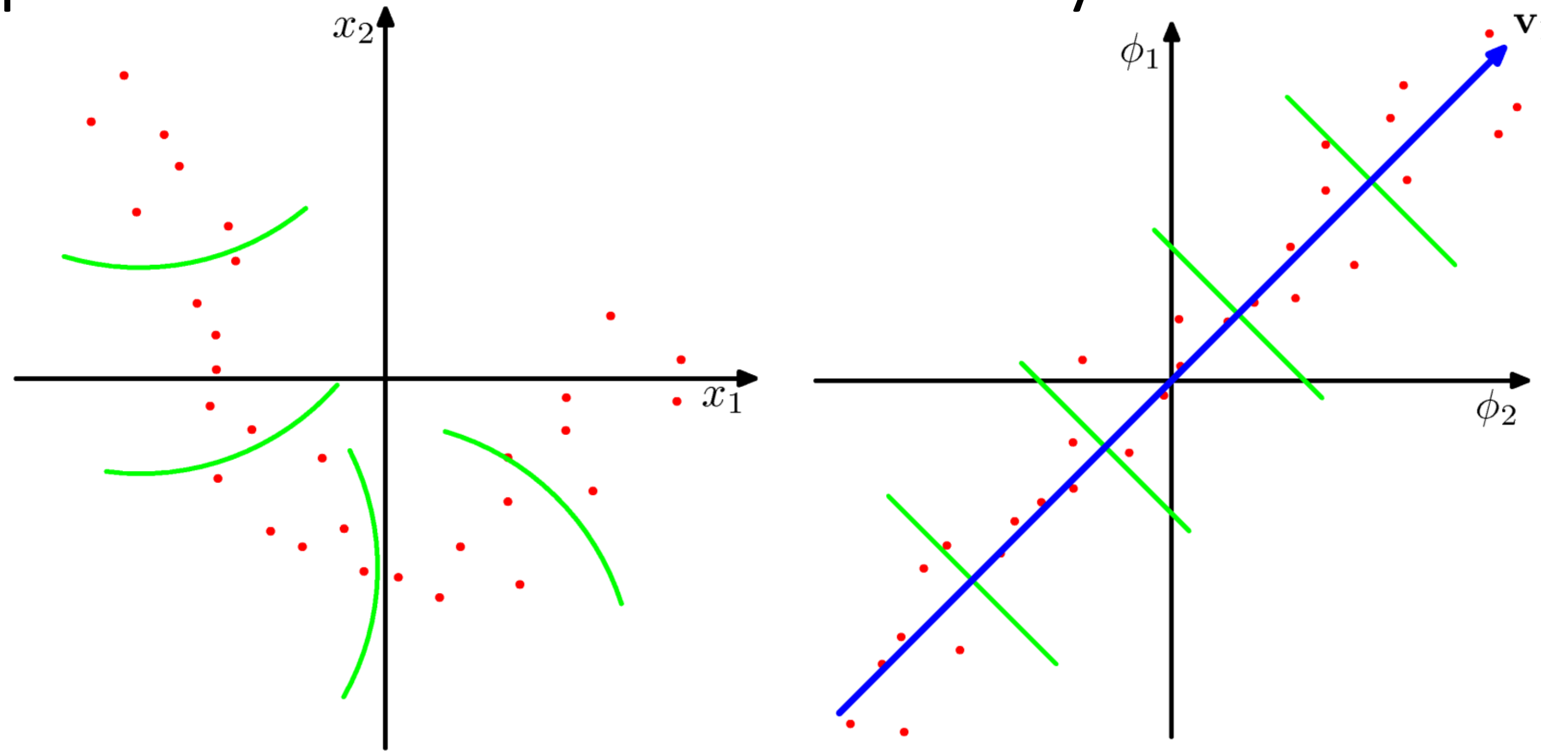


Kernel PCA

- (Linear) PCA can be performed on a feature space:

$$\{\mathbf{x}^{(n)}\}_{n=1}^{N} \rightarrow \{\phi(\mathbf{x}^{(n)})\}_{n=1}^{N}$$

- Together with a nonlinear feature mapping ϕ, PCA performs a nonlinear dimensionality reduction.



Kernel PCA

- Define a **kernel** to avoid having to evaluate the feature vectors explicitly
$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$$
- PCA can be expressed in terms of the kernel.
  - Need centering the kernel matrix (to zero mean)
$$K' = K - LK - KL + LKL$$
  - Where $L = \frac{1}{N} 1_N 1_N^T$ is the $N \times N$ matrix with all $1/N$.
- Solved by eigenvalue decomposition

- FLD can be kernelized in a similar way.

Independent Component Analysis

Independent Component Analysis

- ICA is used for blind source separation.

- Suppose m independent signals are mixed, and sensed by m independent sensors.
  - E.g., Cocktail party with speakers and microphones.
  - E.g., EEG with brain wave sources and sensors.

- Can we reconstruct the original signals, given the mixed data from the sensors?

Independent Component Analysis

- The sources s must be independent and non-Gaussian.
  - If not, there is no way to find unique independent components.
- Assume the sensor signals x is a linear mixing (transformation) of the sources s.
  - $x = As$
  - $s = Wx \ (W = A^{-1})$
- A is called bases; W is called filters.

ICA Algorithms

- **Maximum Likelihood Estimation**
  - Bell and Sejnowski (1995)

- Maximizing Non-Gaussianity

ICA: Maximum Likelihood

- Let $p_s(s_i)$ be the density function of each source s_i.
- By definition, all sources are independent:

$$p(s) = \prod_{i=1}^{M} p_s(s_i)$$

- Substituting $s = Wx$ to get the data distribution:

$$p(x) = \prod_{j=1}^{M} p_s\left(w_j^T x\right) \cdot |W|$$

- Modeling the CDF of source distribution as sigmoid:

$$\int_{-\infty}^{s} p_s(s')ds' = g(s) = \frac{1}{1 + \exp(-s)}$$

ICA: Maximum Likelihood

- Substituting $s = Wx$ to get the data distribution:

$$p(x) = \prod_{j=1}^{M} p_s\left(w_j^T x\right) \cdot |W|$$

- Modeling the CDF of source distribution as sigmoid:

$$\int_{-\infty}^{s} p_s(s')ds' = g(s) = \frac{1}{1 + \exp(-s)}$$

- Log-likelihood

$$\ell(W) = \sum_{i=1}^{N} \left(\sum_{j=1}^{M} \log g'\left(w_j^T x^{(i)}\right) + \log|W| \right)$$

ICA: Maximum Likelihood

- Log-likelihood

$$\ell(W) = \sum_{i=1}^{N} \left(\sum_{j=1}^{M} \log g'\left(w_j^T x^{(i)}\right) + \log|W| \right)$$

- Update rule (Note: $\nabla_W |W| = |W| W^{-T}$)

$$W := W + \alpha \left(\begin{bmatrix} 1 - 2g\left(w_1^T x^{(i)}\right) \\ 1 - 2g\left(w_2^T x^{(i)}\right) \\ \vdots \\ 1 - 2g\left(w_M^T x^{(i)}\right) \end{bmatrix} \left(x^{(i)}\right)^T + W^{-T} \right)$$

ICA Algorithms

- Maximum Likelihood Estimation
  - Bell and Sejnowski (1995)

- **Maximizing Non-Gaussianity**

ICA: Maximizing non-Gaussianity

- Common steps of ICA (e.g., FastICA):

1. Preprocessing: $\tilde{\mathbf{x}} \leftarrow \mathbf{x}$
   - Centering: Subtracting data mean
   $$\mathbb{E}[\tilde{\mathbf{x}}] = \mathbf{0}$$
   - PCA Whitening: Decorrelation and standardization
   $$\mathbb{E}[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T] = I$$
2. Find orthogonal unit vectors along which that the non-Gaussianity are maximized:
   $$\max_W L(W\tilde{\mathbf{x}}) \text{ s.t. } WW^T = I$$
   - Where $L(\cdot)$ can be log cosh, Kurtosis, L1-norm, …

ICA Preprocessing: PCA Whitening

- Whitening can be done by a linear transformation:

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

- Such that they are uncorrelated & unit variance:

$$\mathbb{E}[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T] = I$$

- From the eigenvalue decomposition of covariance

$$S = U\Lambda U^T$$

- PCA whitening transformation matrix:

$$V = \Lambda^{-1/2}U^T$$

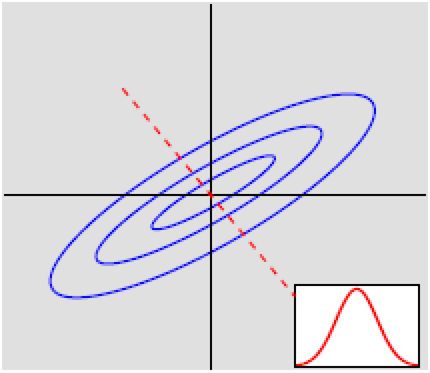
- Because

$$\mathbb{E}[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T] = \mathbb{E}[V\mathbf{x}\mathbf{x}^T V^T] = I$$

ICA Preprocessing: PCA Whitening

- PCA whitening transformation matrix:

$$V = \Lambda^{-1/2} U^T$$



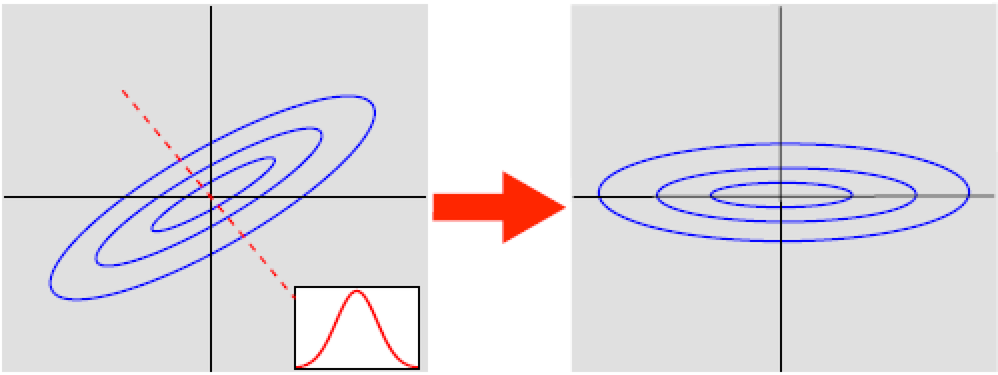
$\mathbf{x}$

ICA Preprocessing: PCA Whitening

- PCA whitening transformation matrix:

$$V = \Lambda^{-1/2} U^T$$

- Decorrelation: Project to the principal components



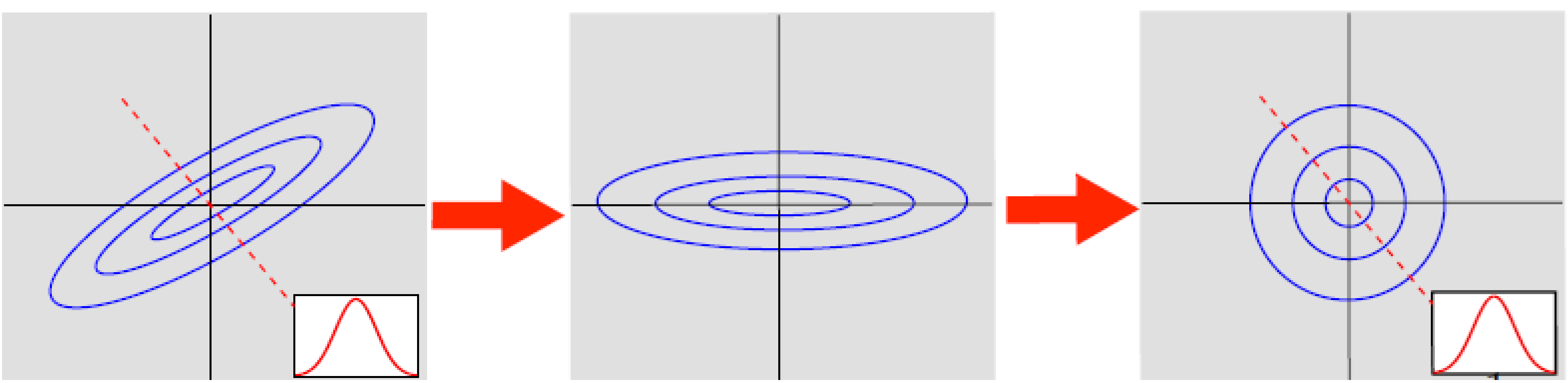
$\mathbf{x}$ $\qquad\qquad$ $U^T\mathbf{x}$

ICA Preprocessing: PCA Whitening

- PCA whitening transformation matrix:

$$V = \Lambda^{-1/2} U^T$$

- Decorrelation: Project to the principal components
- Standardization: Scale each axis to have unit var.



$\mathbf{x}$ $\quad$ $U^T\mathbf{x}$ $\quad$ $V\mathbf{x} = \Lambda^{-1/2} U^T \mathbf{x}$

ICA: Maximizing non-Gaussianity

- Kurtosis is a measure of the **tailedness**:

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

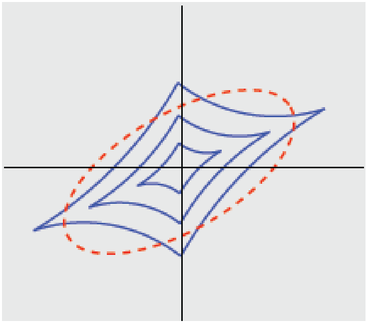
- All Gaussian random variables have a Kurtosis of 3.
- Maximizing Kurtosis results in increasing the non-Gaussianity.

ICA: Maximizing non-Gaussianity

- PCA whitening transformation matrix:

$$V = \Lambda^{-1/2} U^T$$

- Decorrelation: Project to the principal components
- Standardization: Scale each axis to have unit var.



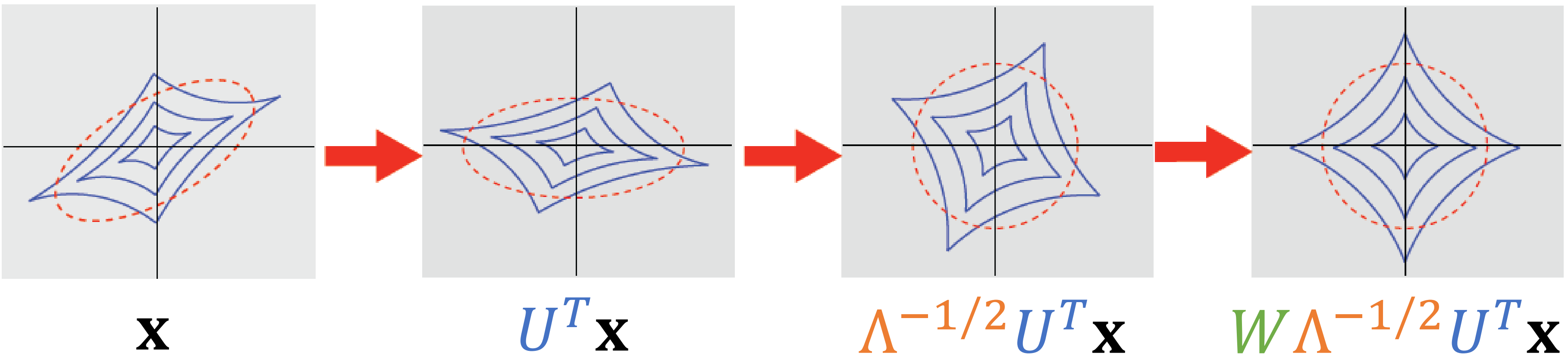
$\mathbf{x}$

ICA: Maximizing non-Gaussianity

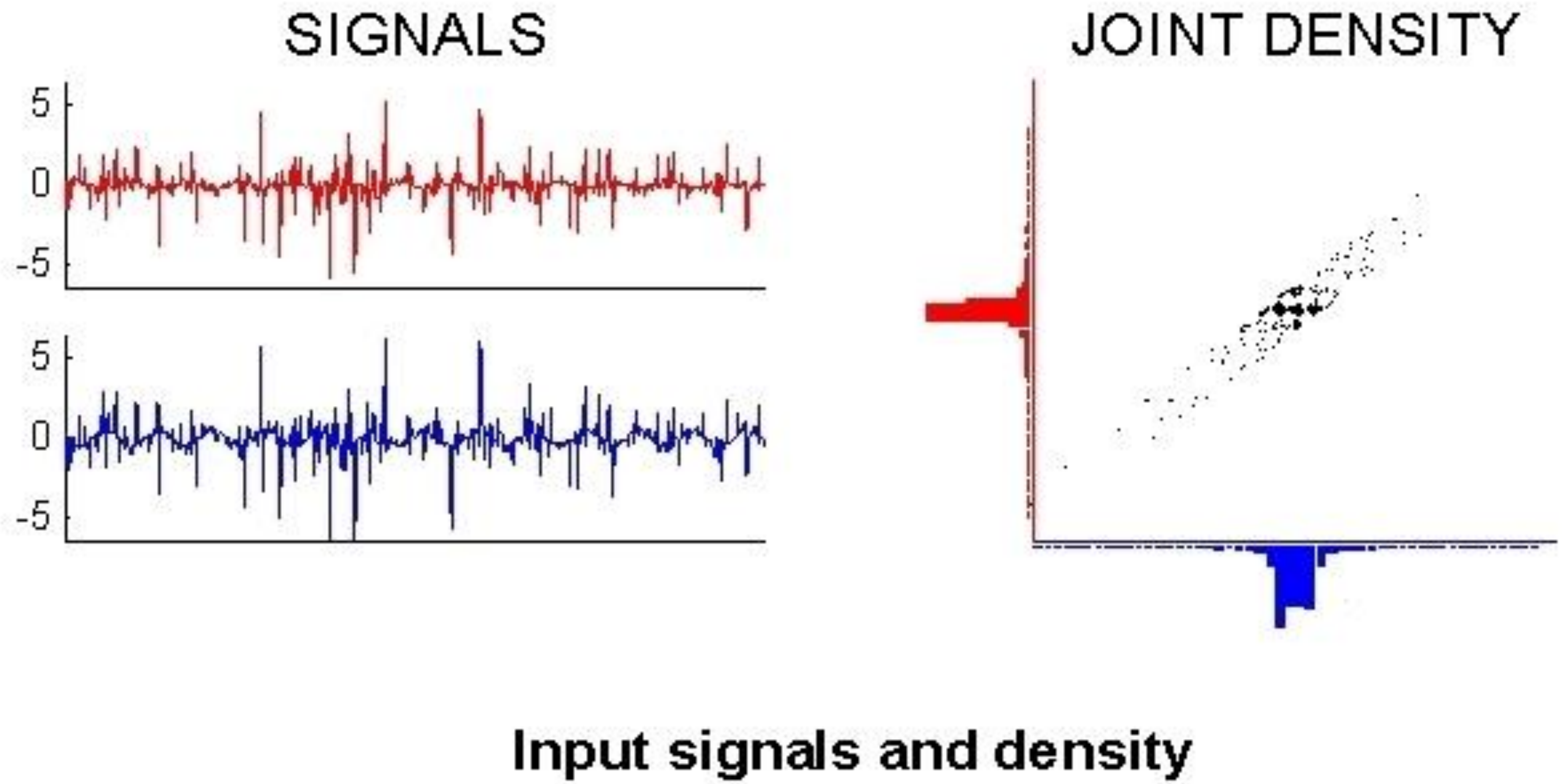
- PCA whitening transformation matrix:

$$V = \Lambda^{-1/2} U^T$$

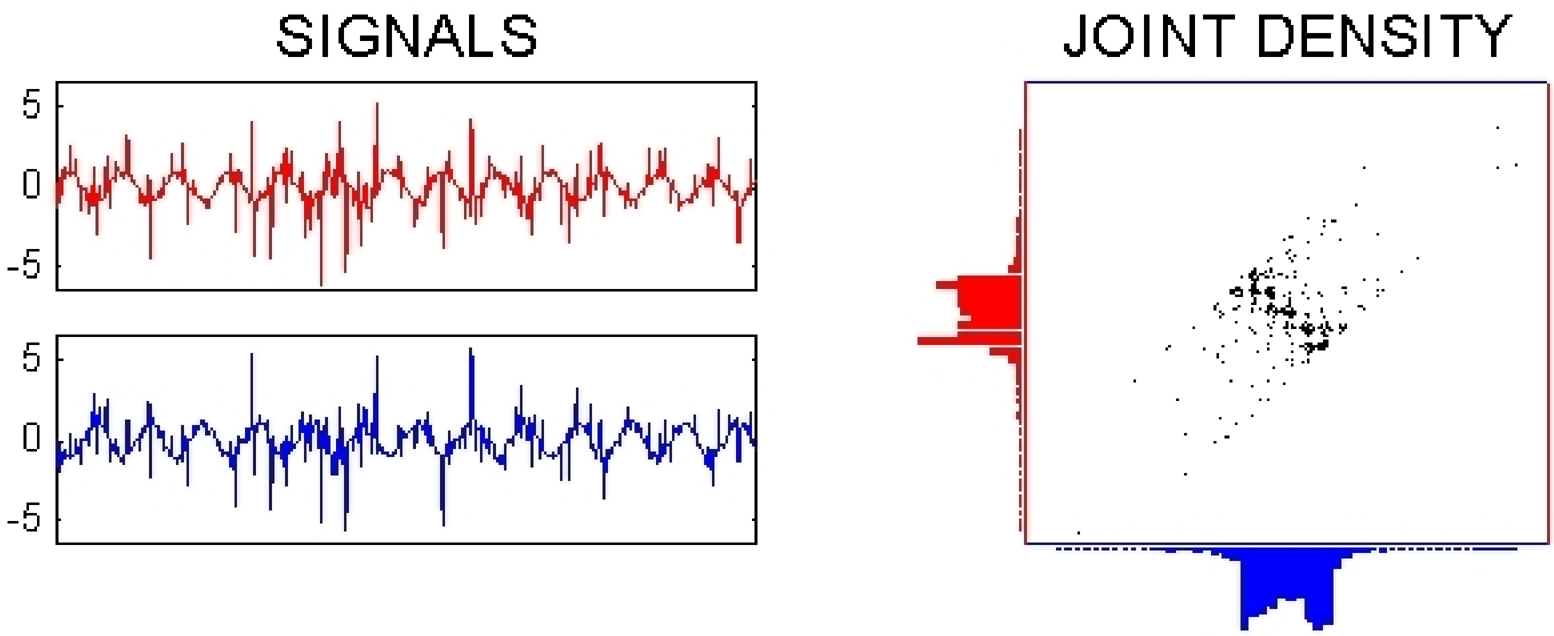
- Decorrelation: Project to the principal components
- Standardization: Scale each axis to have unit var.
- Kurtosis maximization: Rotate to maximize non-Gaussianity



ICA: Example

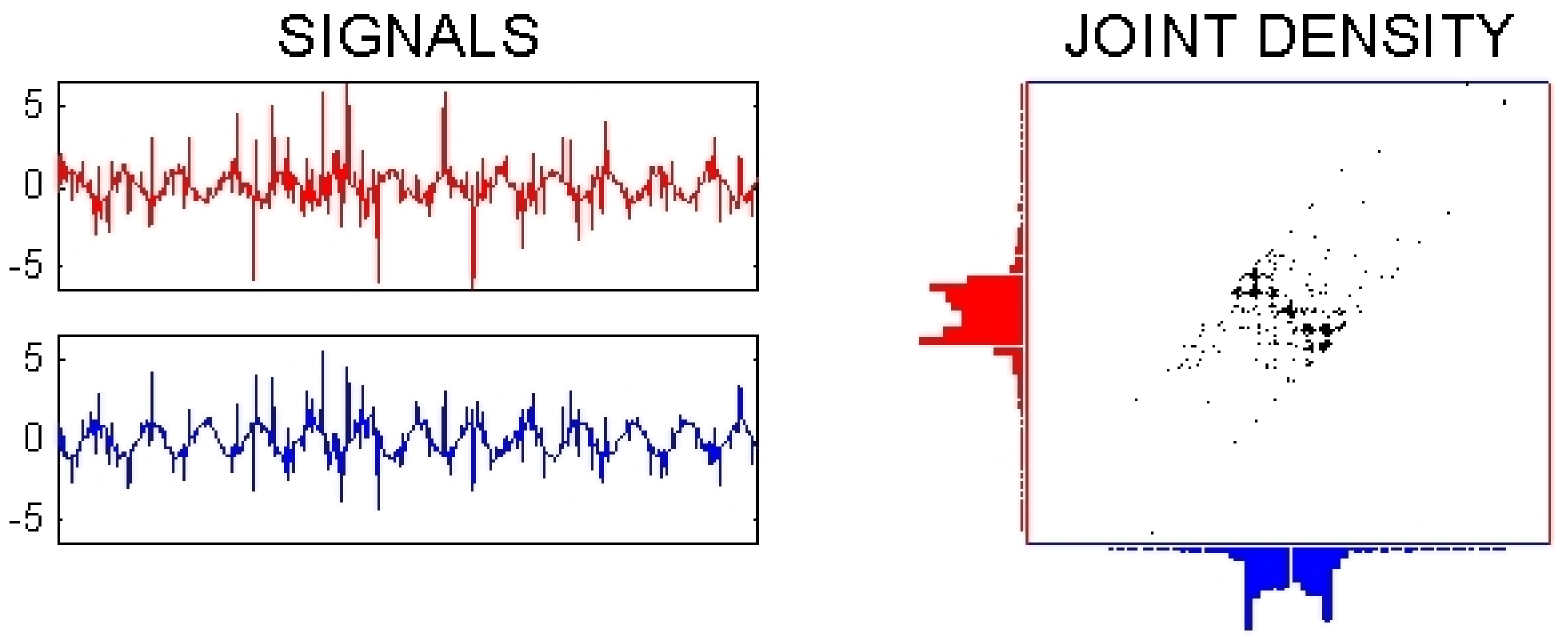


ICA: Example



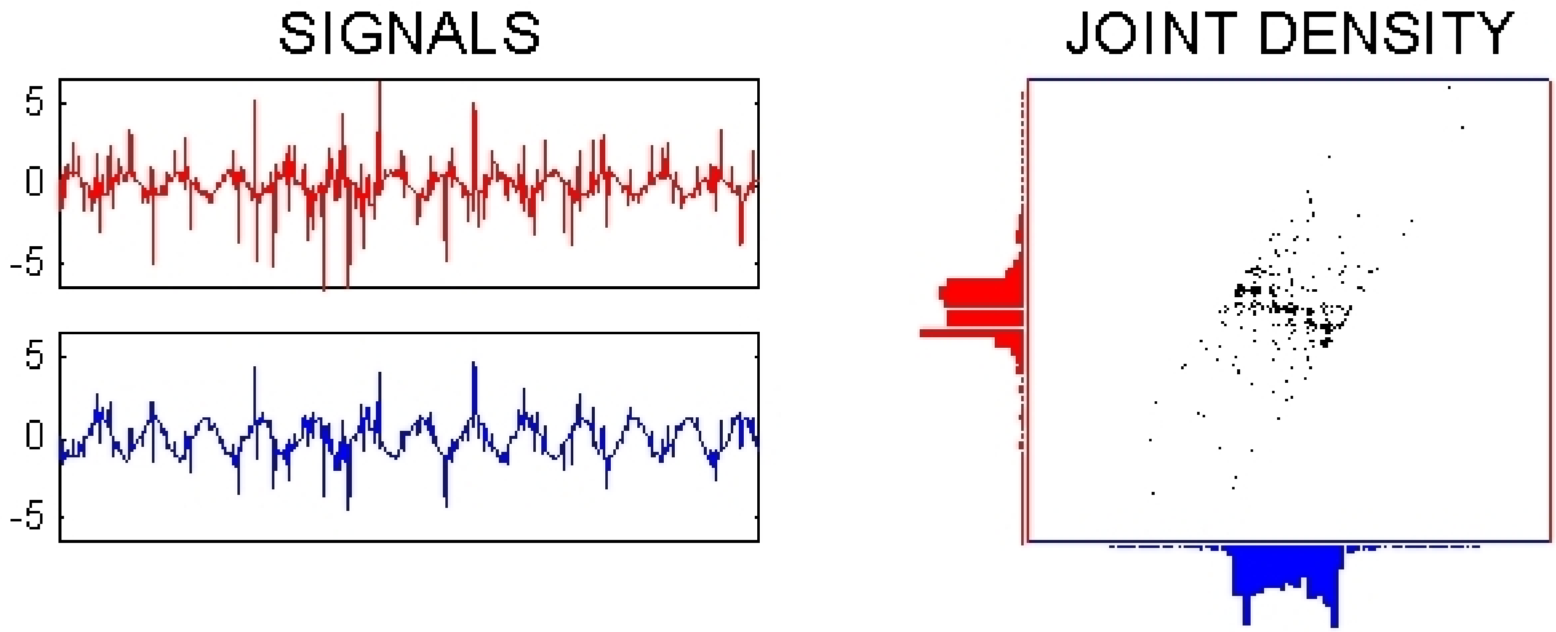
Whitened signals and density

ICA: Example



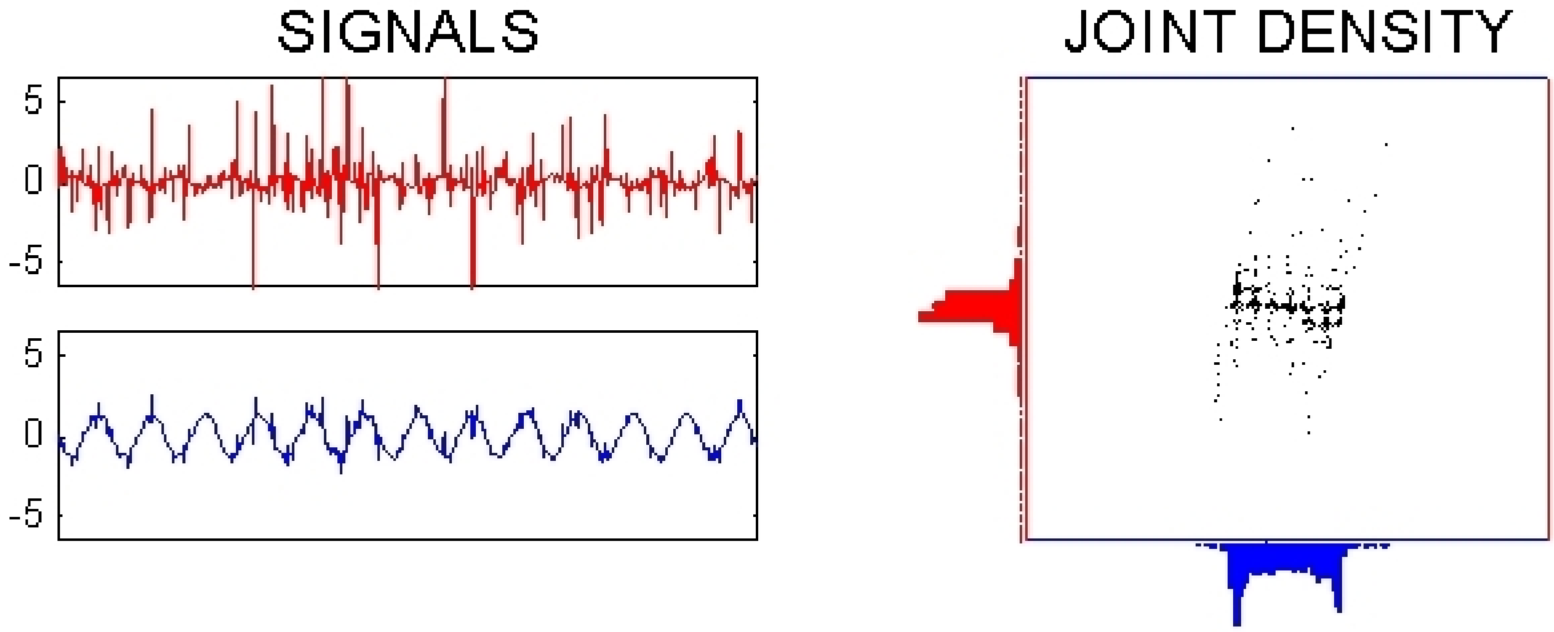
Separated signals after 1 step of FastICA

ICA: Example



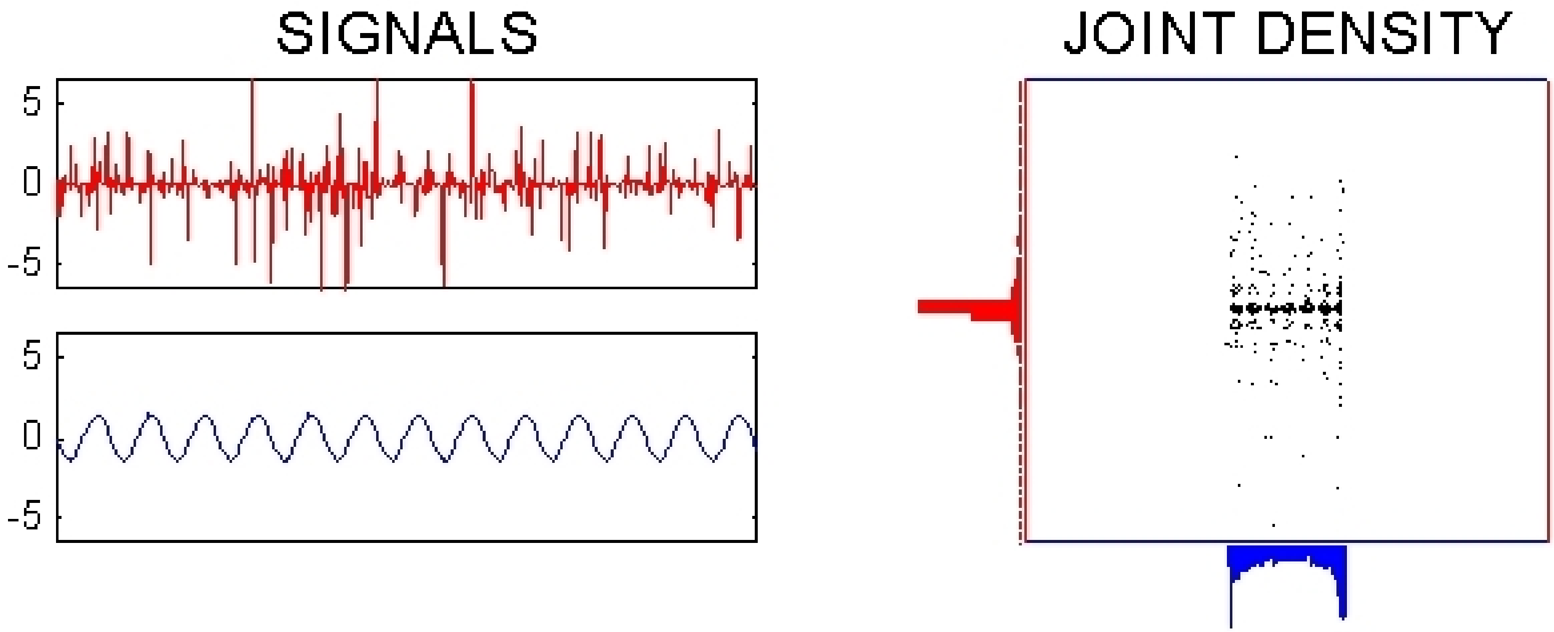
Separated signals after 2 steps of FastICA

ICA: Example



Separated signals after 3 steps of FastICA

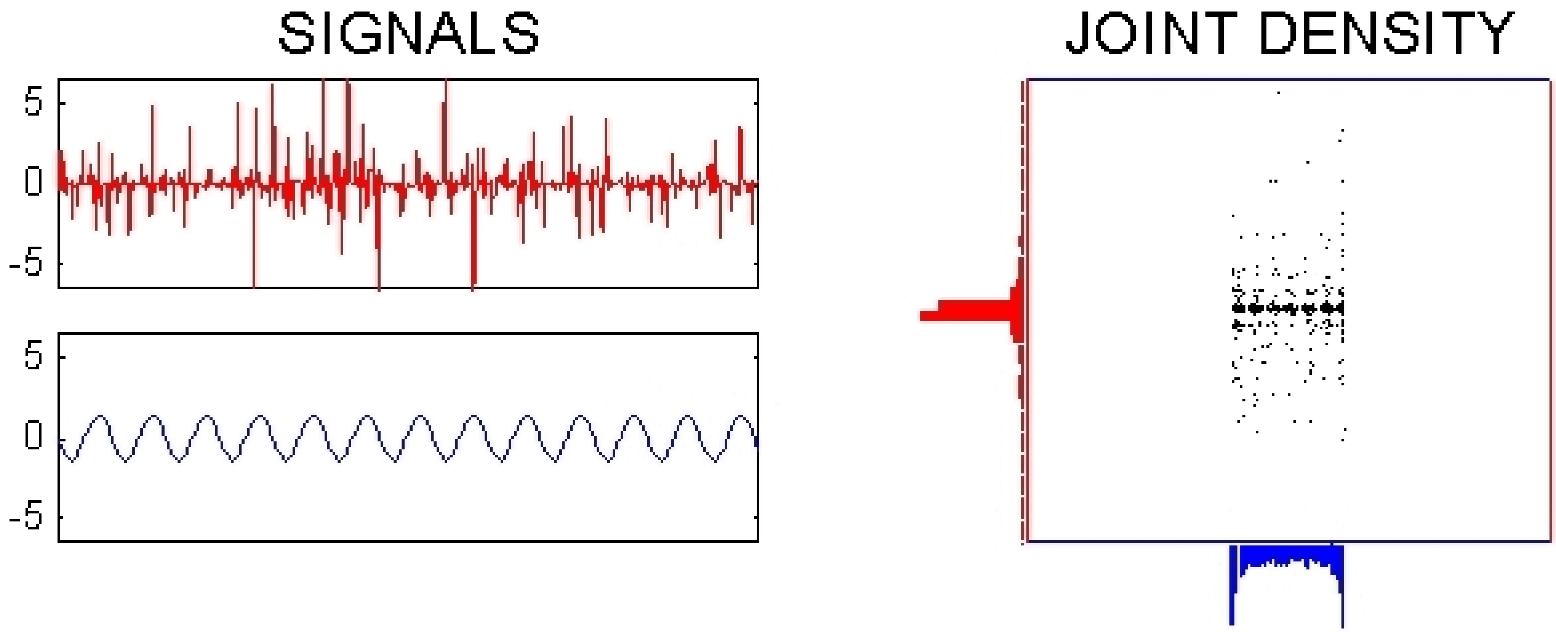
ICA: Example



Separated signals after 4 steps of FastICA

ICA: Example

- $p(\mathbf{x}_{ICA,1}, \mathbf{x}_{ICA,2}) = p(\mathbf{x}_{ICA,1})p(\mathbf{x}_{ICA,2})$



Separated signals after 5 steps of FastICA

ICA: Summary

- ICA is used for **blind-source separation**.
- ICA can be done by
  - Maximum likelihood estimation
  - Maximizing non-Gaussianity
    - E.g., PCA whitening followed by kurtosis maximization
- ICA components can be used for features.
- Difficult to learn overcomplete (or redundant) bases due to the orthogonality constraint
  - Overcompleteness is good in the sense that
    - Representation can be more compact/sparse
    - Robust to noise

Linear Dimensionality Reduction

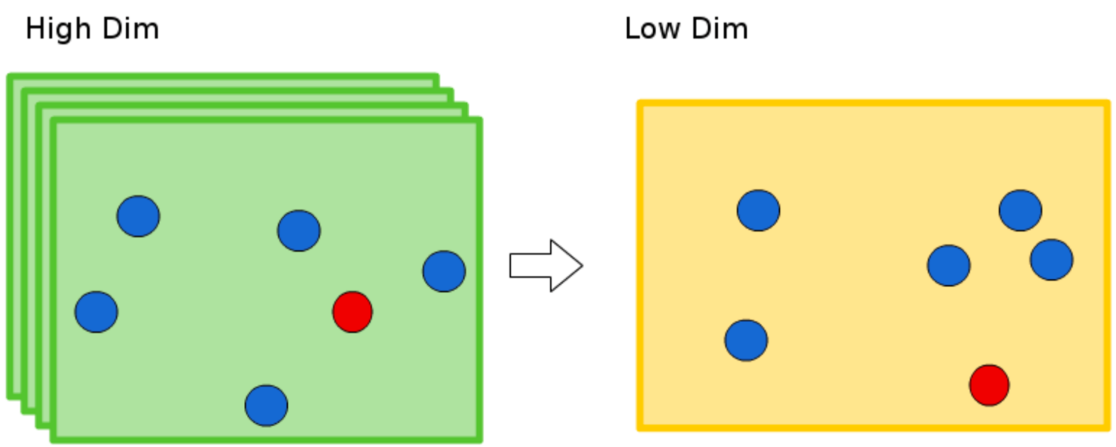
- Fisher's linear discriminant (FLD)
    - Supervised learning
- **Principal component analysis (PCA)**
- Independent component analysis (ICA)
- Sparse coding
- Multidimensional scaling (MDS)

t-Distributed Stochastic Neighbor Embedding

van der Maaten and Hinton, JMLR’08

t-SNE: Motivation

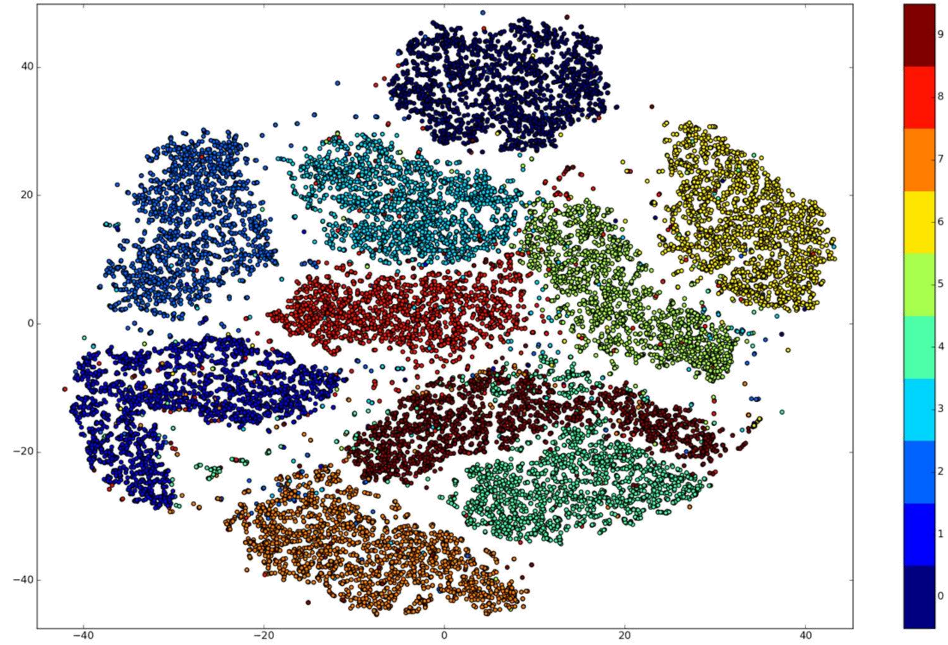
- Given a collection of high-dimensional data $\{x^{(1)}, \ldots, x^{(N)}\}$, how can we get a sense of how they are arranged in data space?
- We would like to compress the high-dimensional data into a lower-dimensional space while preserving distance and neighborhood structure.



Slide credit: Kai-Wen Zhao

t-SNE: Motivation

- Given a collection of high-dimensional data $\{x^{(1)}, \dots, x^{(N)}\}$, how can we get a sense of how they are arranged in data space?
- t-SNE visualization of MNIST:



Slide credit: Kai-Wen Zhao

Stochastic Neighbor Embedding

- SNE converts Euclidean distances to similarities.
  - Interpreted as probabilities P^i over neighbors of $x^{(i)}$.
- Then, we find embedding Q^i that approximates P^i.

$$P^i = \left\{p^{1|i}, p^{2|i}, \ldots, p^{N|i}\right\} \qquad p^{j|i} = \frac{\exp\left(-\left\|x^{(i)} - x^{(j)}\right\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\left\|x^{(i)} - x^{(k)}\right\|^2 / 2\sigma_i^2\right)}$$

$$Q_i = \left\{q^{1|i}, q^{2|i}, \ldots, q^{N|i}\right\} \qquad q^{j|i} = \frac{\exp\left(-\left\|y^{(i)} - y^{(j)}\right\|^2\right)}{\sum_{k \neq i} \exp\left(-\left\|y^{(i)} - y^{(k)}\right\|^2\right)}$$

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

- **SNE** minimizes the KL divergence:

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

Slide credit: Simon Carbonnelle

Symmetric SNE

- SNE converts Euclidean distances to similarities.
  - Interpreted as probabilities P^i over neighbors of $x^{(i)}$.
- Symmetrize p and q:

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

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

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

- **Symmetric SNE** minimizes the KL divergence:

$$C = KL(P||Q) = \sum_{i,j} p^{ij} \log \frac{p^{ij}}{q^{ij}}$$

Slide credit: Simon Carbonnelle

t-SNE

- SNE converts Euclidean distances to similarities.
  - Interpreted as probabilities P^i over neighbors of $x^{(i)}$.
- Symmetrize p and q, Student's t-distribution for q:

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

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

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

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

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

- **t-SNE** minimizes the KL divergence:

$$C = KL(P||Q) = \sum_{i,j} p^{ij} \log \frac{p^{ij}}{q^{ij}}$$

Slide credit: Simon Carbonnelle

From SNE to t-SNE

SNE

- Modelization:

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

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

- 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

- Modelization:

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

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

- 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

- Modelization:

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

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

- Cost function:

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

- Derivatives:

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

- Faster optimization
- More robust to noise (heavy tailed t-dist.)

Slide credit: Simon Carbonnelle

t-SNE: Summary

- Advantages:
  - Nonlinear dimensionality reduction while preserving distance and neighborhood structure
  - Probabilistic interpretation: Minimizing the KL divergence between the pdf of inputs and outputs
    - Input is modeled by Gaussian distribution
    - Output is modeled by Student's t-distribution
- Disadvantages:
  - Quadratic w.r.t. the number of data: $O(N^2)$
  - Non-convex optimization
  - Non-parametric; cannot reuse learned mappings
    - Cf. Parametric t-SNE (van der Maaten and Hinton, AISTATS'09)
  - Hyperparameter tuning required

Nonlinear Dimensionality Reduction

- Isometric feature mapping (ISOMAP)
- Locally linear embedding (LLE)
- **t-distributed stochastic neighbor embedding (t-SNE)**
  - van der Maaten and Hinton (2008)
- **Uniform manifold approximation and projection (UMAP)**
  - McInnes et al. (2018)

- **t-SNE and UMAP are commonly used for visualizing features in deep learning**

Next: Neural Networks