Talk 12 Unsupervised Learning

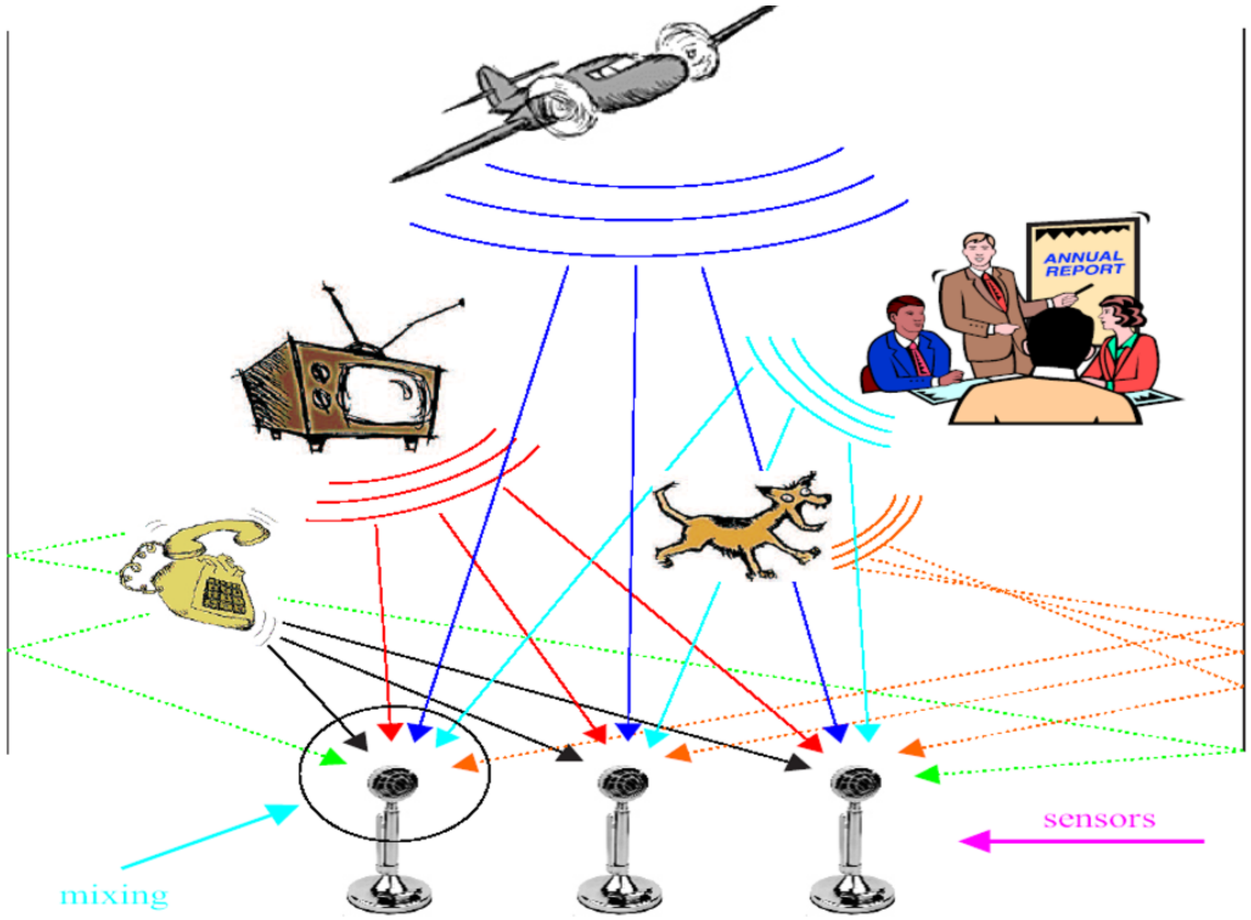
Part 2

Unsupervised Learning

1. Introduction
2. Association Rules & Cluster Analysis
4. Self-Organizing Maps
5. Principal Components, Curves and Surfaces
6. Non-negative Matrix Factorization
7. Independent Component Analysis
8. Multidimensional Scaling
9. Nonlinear Dimension Reduction
10. The Google PageRank Algorithm

6. Independent Component Analysis (ICA)

- Speech Separation (Cocktail Party Problem)

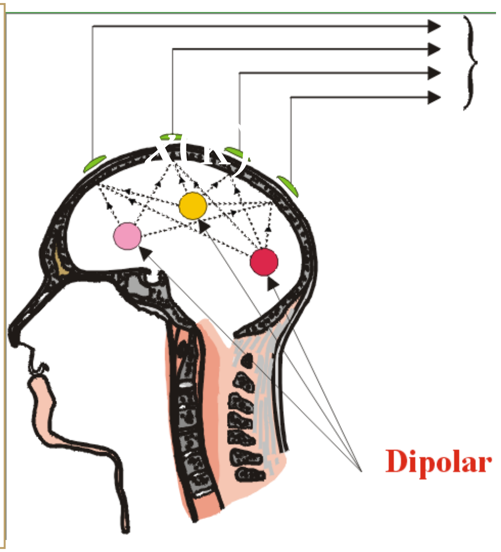


Mathematical Formulation

Mixing Model

$$x_i(k) = a_{i1}s_1(k) + a_{i2}s_2(k) + \cdots + a_{in}s_n(k) + \varepsilon(k)$$

$$\mathbf{x}(k) = \mathbf{A}\mathbf{s}(k) + \boldsymbol{\varepsilon}(k)$$



- $\boldsymbol{s}$(k)= $(s_1(k), \ldots, s_n(k))^T$: the vector of n-source signals;
- $\boldsymbol{x}$(k)= $(x_1(k), \ldots, x_m(k))^{\mathrm{T}}$: the vector of m-sensor signals;
- $\boldsymbol{\varepsilon}$ (k): the vector of sensor noises.
- $\boldsymbol{A}$ is the mixing matrix.

Demixing Model

Problem: to estimate the source signals (or event-related potentials) by using the sensor signals

$$\mathbf{y}(k) = \mathbf{W}\,\mathbf{x}(k)$$

- $\boldsymbol{y}(\mathrm{k}) = (y_1(k), \ldots, y_m(k))^T$: the vector of recovered signals
- $\mathbf{W}$ is the demixing matrix.

$$\boldsymbol{s}(\boldsymbol{k}) \longrightarrow \boxed{A} \xrightarrow{\boldsymbol{x}(\boldsymbol{k})} \boxed{W} \xrightarrow{\boldsymbol{y}(\boldsymbol{k})}$$

Basic Theory

- **Assumption:**
  - ✓The source signal are mutually independent
- **Model:**
  - ✓Linear instantaneous mixture
  - ✓Linear convolutive mixture
- **Principles**
  - ✓Maximum Entropy
  - ✓Minimum Mutual Information
  - ✓Joint Diagonalization of Cross-correlations
  - ✓Linear Predictability
  - ✓Sparseness Maximization

6. Independent Component Analysis (ICA)

- One of Approaches to recover independent component one by one:
  - Maximize the NonGaussianity
- Let mixing model: $\mathbf{x} = \mathbf{A}\mathbf{s} + \boldsymbol{\varepsilon}; \quad \mathbf{s} = \left(s_1, s_2, \cdots, s_n\right)^T$
- We attempt to extract a single component from $\mathbf{x}$

$$y(k) = \mathbf{b}^T \mathbf{x}(k) = \mathbf{b}^T \mathbf{A}\mathbf{s}(k)$$

- Let $\mathbf{z} = \mathbf{A}^T \mathbf{b}$

$$y(k) = \mathbf{b}^T \mathbf{A}\mathbf{s}(k) = \mathbf{z}^T \mathbf{s}(k)$$

What we can derive from this observation?
CLT provides us a criterion for the ICA.

CLT – Central Limit theory

Assume $X_1, X_2, \cdots, X_n$ are iid samples from a probability with mean μ and variance σ^2, define

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i; \quad \mathrm{U}_n = \frac{\bar{X}-\mu}{\sigma/\sqrt{n}}$$

Then the random variable U_n converges to the nornal distribution as the sample size n tends to infinite.

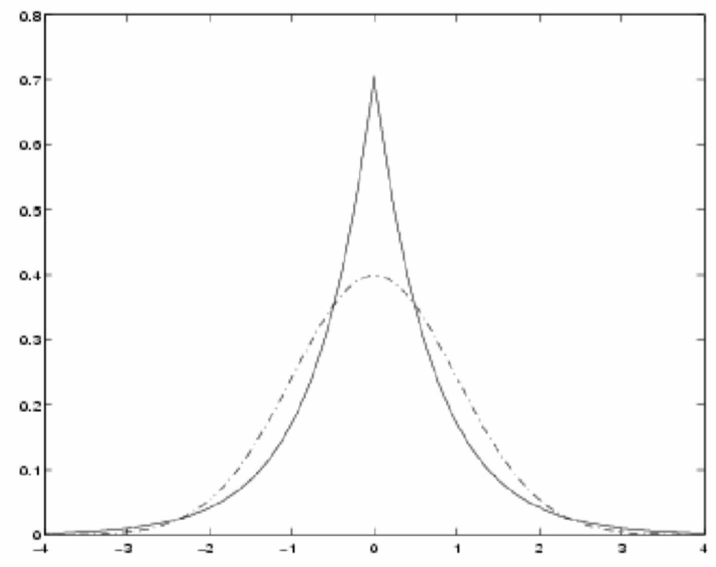
$$\boxed{\mathrm{U}_n = \frac{\bar{X}-\mu}{\sigma/\sqrt{n}} \to N(0,1)}$$

6. Independent Component Analysis (ICA)

- **Kurtosis is a measure of non-gaussianity**

$$\text{kurtosis}(y) = E[y^4] - 3\left(E[y^2]\right)^2$$

$$\hat{b} = \operatorname{var}\max_b \left|\text{kurtosis}(b^T \mathbf{x})\right|$$



FastICA algorithm is derived

$$\text{kurtosis}(y) = E[y^4] - 3\left(E[y^2]\right)^2 \begin{cases} > 0, & \text{for SuperGaussian} \\ = 0, & \text{for Gaussian} \\ < 0, & \text{for SubGaussian} \end{cases}$$

Cost Function

Kullback-Leibler (KL) divergence between

$$p(y_1, \cdots, y_K) \quad \text{and} \quad \prod_{k=1}^{K} q_k(y_k)$$

$$KL(W) = \int p(\mathbf{y}) \log \frac{p(\mathbf{y})}{\prod_{k=1}^{K} q_k(y_k)} d\mathbf{y}$$

$$= -H(\mathbf{Y};W) + \sum_{k=1}^{K} H(Y_k;W)$$

1. Joint Entropy of $\mathbf{y}$ **2. Sum of marginal entropy of y_k**

- Minimized when y_k are mutually independent

Cost Function

Assume that $\mathbf{y}=\mathbf{W}\mathbf{x},$ the differential entropy can be expressed by

$$H(\mathbf{y})=H(\mathbf{x})+\log\left|\det(\mathbf{W})\right|$$

Assume that $\mathbf{y}=\mathbf{W}\mathbf{x},$ $\mathbf{W}$ is an invertable matrix, the probablity density function of $\mathbf{y}$ is given by $p_y(\mathbf{y})=\left|\det(\mathbf{W})\right|^{-1} p_x(\mathbf{x})$

$$\begin{aligned}
H(\mathbf{y}) &= -\int p(y)\log(p(y))dy \\
&= -\int \left|\det(\mathbf{W})\right|^{-1} p_x(\mathbf{x})\log(\left|\det(\mathbf{W})\right|^{-1} p_x(\mathbf{x}))\left|\det(\mathbf{W})\right| d\mathbf{x} \\
&= -\int p_x(\mathbf{x})\log(p_x(\mathbf{x}))d\mathbf{x}+\log(\left|\det(\mathbf{W})\right|)
\end{aligned}$$

Cost Function

Assume that $\mathbf{y} = \mathbf{W}\mathbf{x}$, the differential entropy can be expressed by

$$H(\mathbf{y}) = H(\mathbf{x}) + \log\left|\det(\mathbf{W})\right|$$

$$KL(W) = \int p(\mathbf{y}) \log \frac{p(\mathbf{y})}{\prod_{k=1}^{K} p(y_k)} d\mathbf{y}$$

$$= -H(\mathbf{Y};W) + \sum_{k=1}^{K} H(Y_k;W)$$

$$= -H(x) - \log\left|\det(W)\right| - \sum_{k=1}^{K} E\left[\log(p_k(y_k))\right]$$

Gradient Descent

To update W along the negative gradient of $KL(W)$

$$\Delta \boldsymbol{W} \propto -\frac{\partial KL(\boldsymbol{W})}{\partial \boldsymbol{W}} = \left((\boldsymbol{W}^{\mathrm{T}})^{-1} - \int p(x)\phi(\boldsymbol{y})\boldsymbol{x}^{\mathrm{T}} d\boldsymbol{x} \right)$$

$$= \left((\boldsymbol{W}^{\mathrm{T}})^{-1} - \mathrm{E}_x\left[\phi(\boldsymbol{y})\boldsymbol{x}^{\mathrm{T}}\right] \right)$$

$$= \left(\boldsymbol{I} - \mathrm{E}_y\left[\phi(\boldsymbol{y})\boldsymbol{y}^{\mathrm{T}}\right] \right) (\boldsymbol{W}^{\mathrm{T}})^{-1}$$

Nonlinear Function 2 $\Rightarrow$ To be diagonalized

where

$$\phi(\boldsymbol{y}) \equiv \left[\frac{\partial \log p(y_1)}{\partial y_1}, \ldots, \frac{\partial \log p(y_K)}{\partial y_K} \right]^{\mathrm{T}}$$

This can be approximated by Sigmoid Function in speech signal.

Gradient Descent

To update W along the negative gradient of $KL(W)$

$$\Delta \mathbf{W} = \alpha \left(\mathbf{I} - \mathrm{E}_y \left[\varphi(\mathbf{y}) \mathbf{y}^{\mathrm{T}} \right] \right) \left(\mathbf{W}^{\mathrm{T}} \right)^{-1}$$

Modified Learning Algorithm

$$\phi(\boldsymbol{y}) \equiv \left[\frac{\partial \log p(y_1)}{\partial y_1}, \ldots, \frac{\partial \log p(y_K)}{\partial y_K} \right]^{\mathrm{T}}$$

$$\Delta \mathbf{W} = \alpha \left(\mathbf{I} - \mathrm{E}_y \left[\varphi(\mathbf{y}) \mathbf{y}^{\mathrm{T}} \right] \right) \mathbf{W}$$

The natural gradient of Nonsingular Matrix Manifold

$$\nabla l(\mathbf{W}) = \frac{\partial l(\mathbf{W})}{\partial \mathbf{W}} \mathbf{W}^T \mathbf{W}$$

ICA theoretical problems

- Convergence of gradient descent

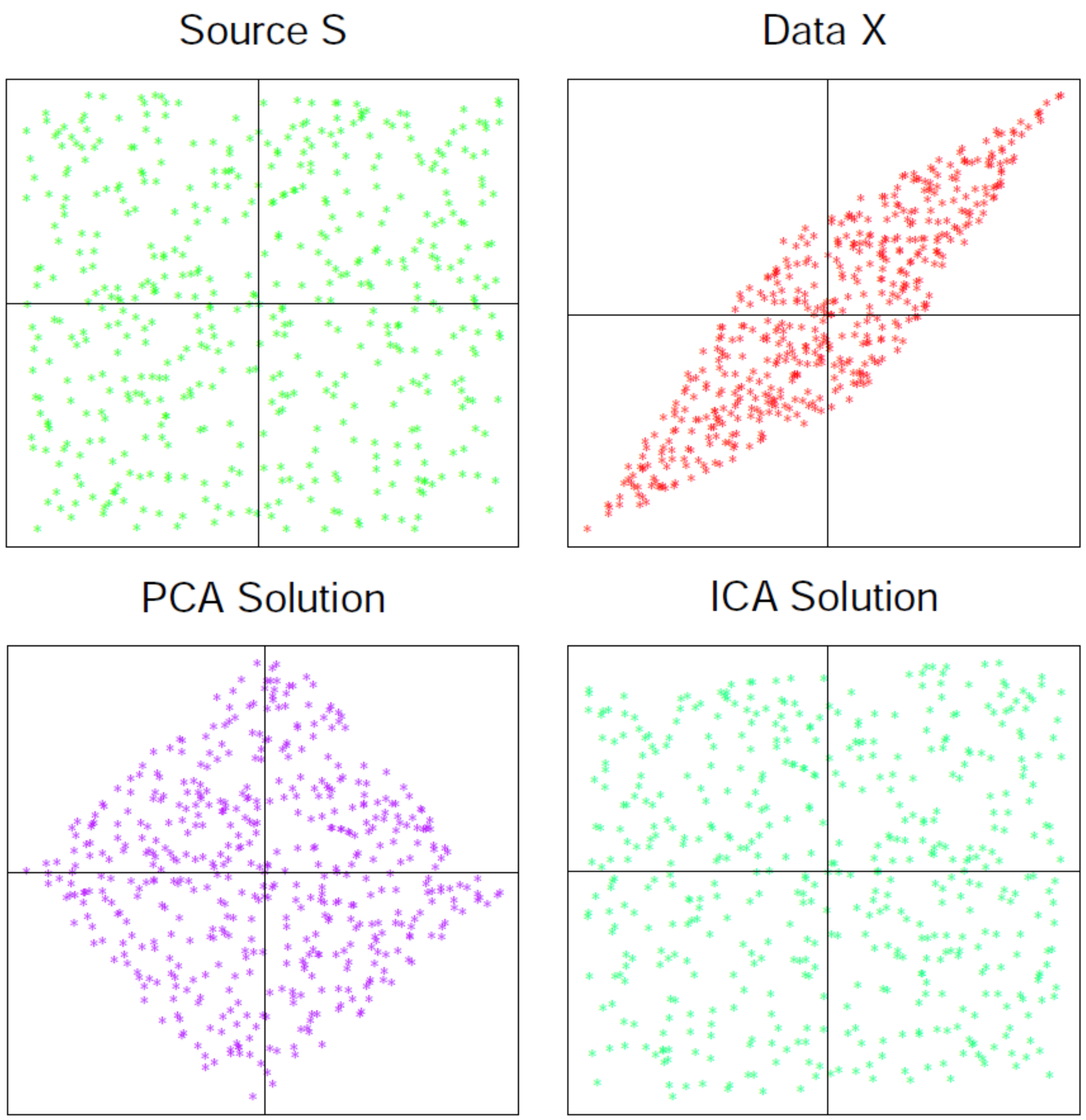
$$\Delta \mathbf{W} = \alpha\left(\mathbf{I} - \mathrm{E}_y\left[\varphi(\mathbf{y})\mathbf{y}^{\mathrm{T}}\right]\right)\mathbf{W}$$

  - Depends on the choice of activation function
  - For super Gaussian, $\varphi(y) = \tanh(y)$
  - For sub Gaussian, $\varphi(y) = y^3$
- Adaptation of activation function
  - Using generalized Gaussian family

$$f(x) = \frac{c_p}{\alpha}\exp\left(-\frac{|x-\mu|^p}{2\alpha^p}\right), \quad c_p = \frac{p}{2^{(p+1)/p}\Gamma(1/p)}, \quad \mathrm{p}>0$$

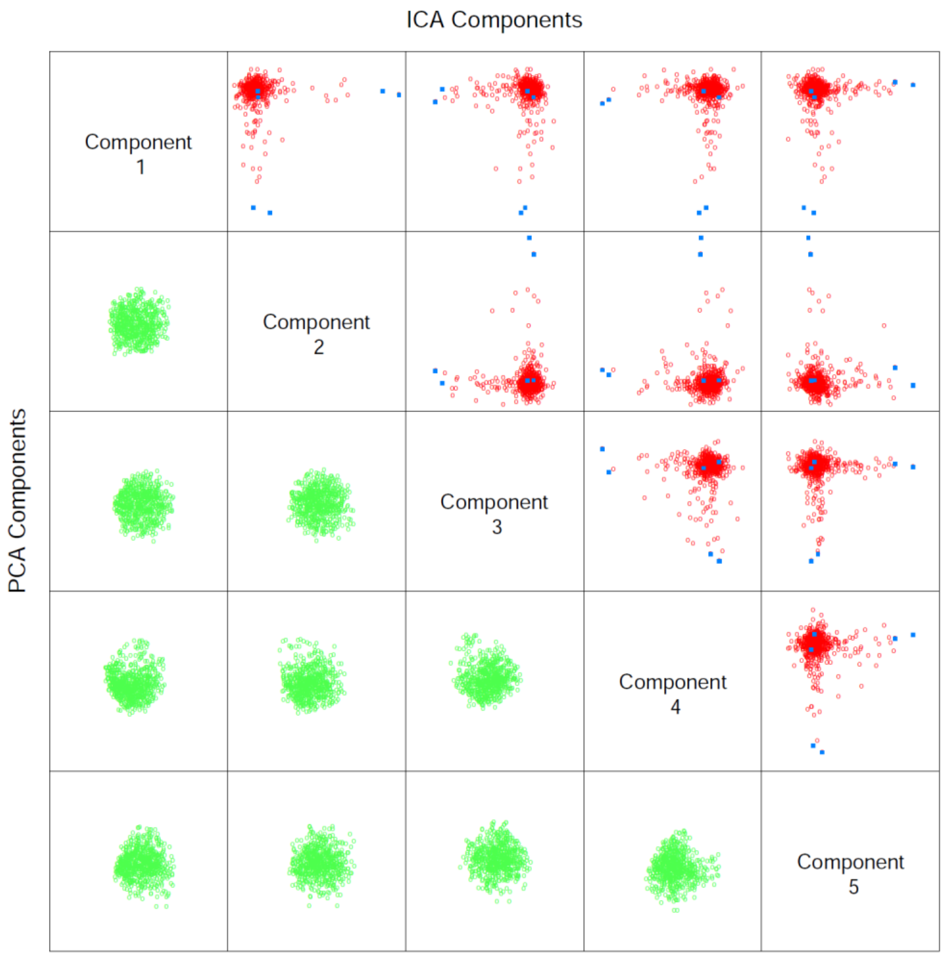
- Determination of the number of ICs

ICA Solution (Non-Uniqueness)



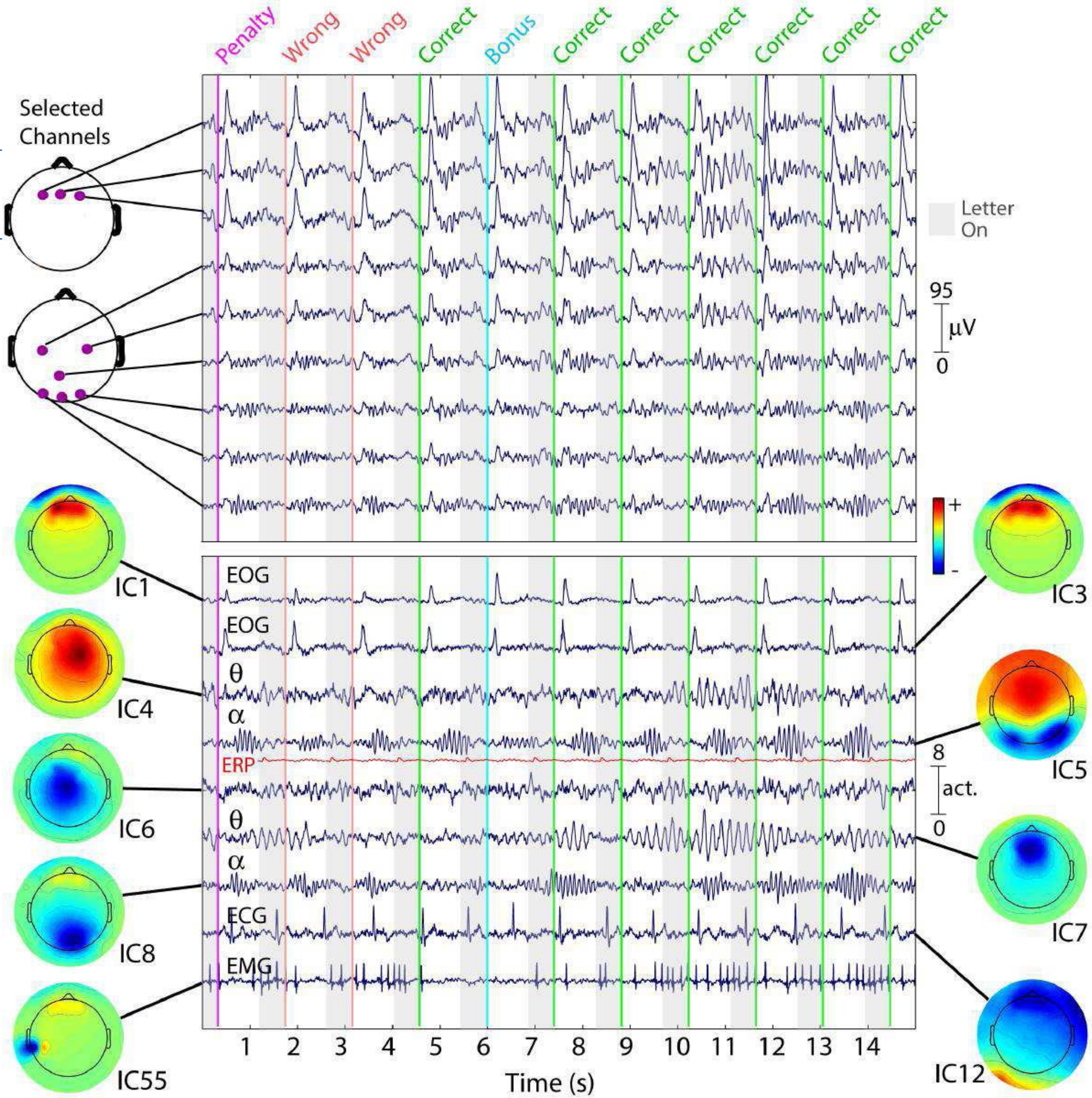
6. Independent Component Analysis (ICA)

- ICA can be used for handwritten digits



- Example:

- EEG feature extraction



Unsupervised Learning

7. Multi-dimensional Scaling (MDS)

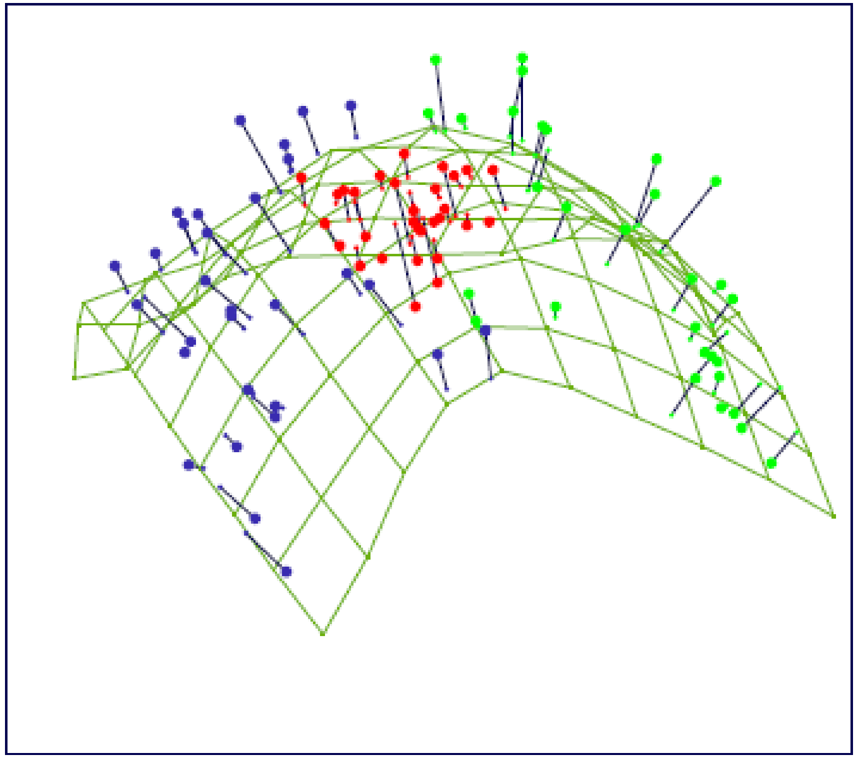
- Objectives: Data dimension reduction: different from SOMs, PCurves, MDS is to preserve the pairwise distance
- Data:

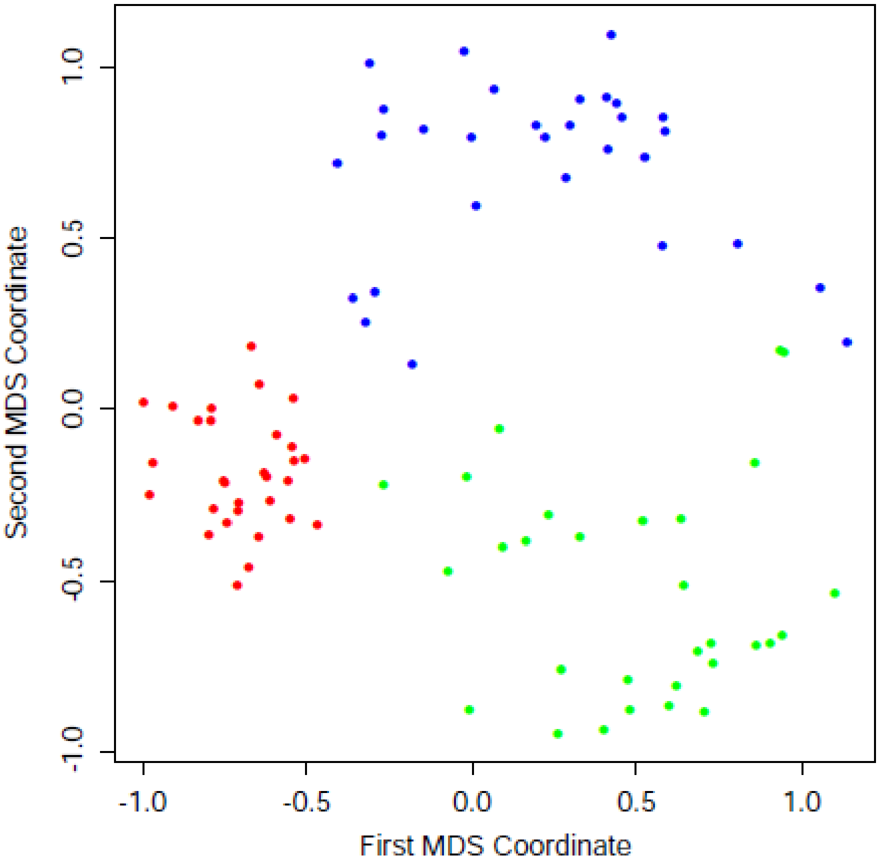
$$D=\{\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_N\},\mathbf{x}_i\in R^p \Rightarrow S=\{\mathbf{s}_1,\mathbf{s}_2,\cdots,\mathbf{s}_N\},\mathbf{s}_i\in R^q(q<p)$$

- Distance or Dissimilarity $d_{ij}=\|\mathbf{x}_i-\mathbf{x}_j\|$
- Goal: find low dimension data $\boldsymbol{S}$ such that

$$S_D(z_1,z_2,\cdots,z_N)=\sqrt{\sum_{i\neq j}\left(d_{ij}-\|z_i-z_j\|\right)^2}$$

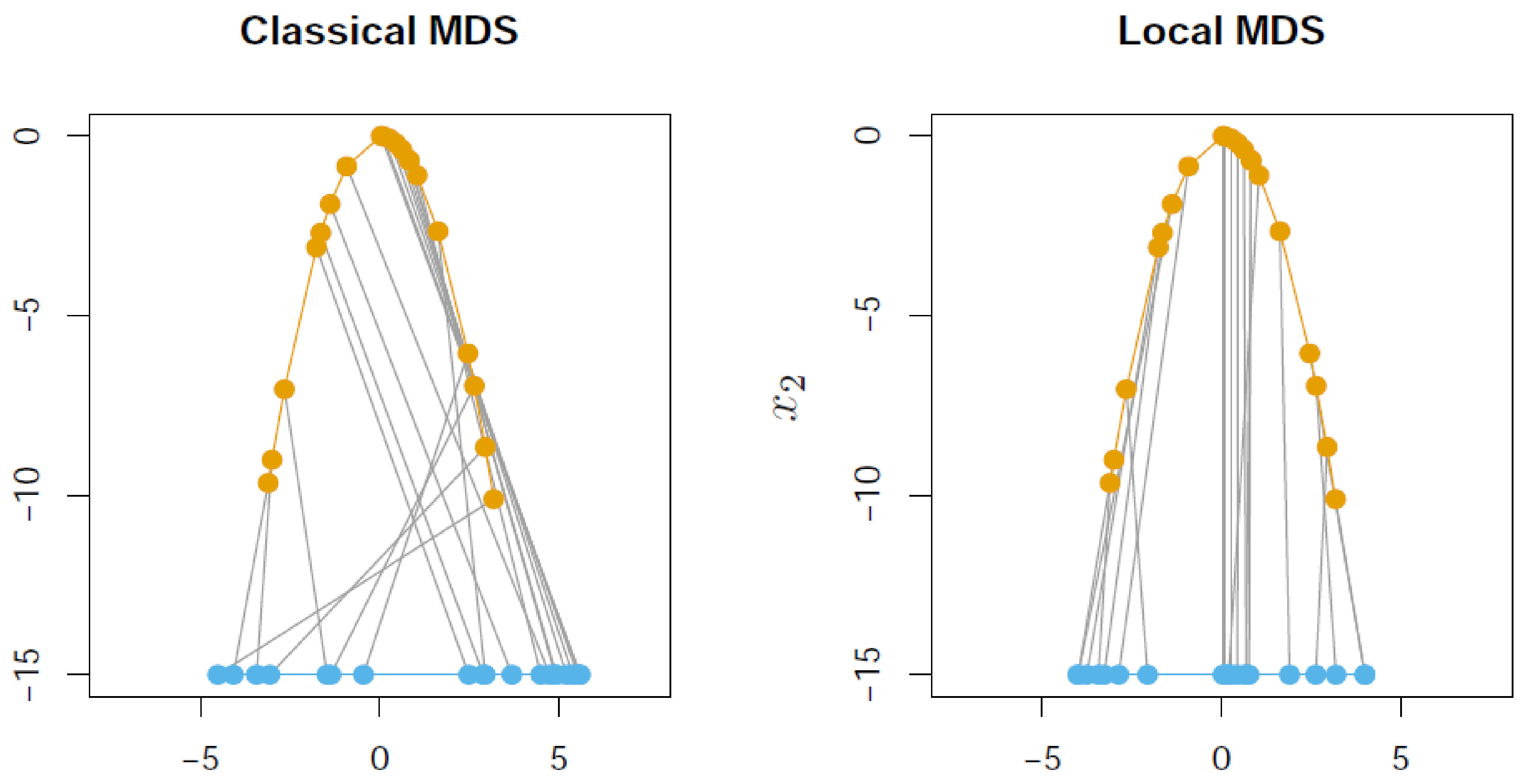
7. Multi-dimensional Scaling (MDS)





14.9 Nonlinear Dimension Reduction

- Problem?



Unsupervised Learning

1. Introduction
2. Association Rules & Cluster Analysis
4. Self-Organizing Maps
5. Principal Components, Curves and Surfaces
6. Non-negative Matrix Factorization
7. Independent Component Analysis
8. Multidimensional Scaling
9. Nonlinear Dimension Reduction
10. The Google PageRank Algorithm

14.9 Nonlinear Dimension Reduction

- Local MDS is to preserve the ordering of the points along the curve.
- Three new approaches to nonlinear dimension reduction and manifold mapping.
  - Isometric feature mapping (ISOMAP) (Tenenbaum et al., 2000)
  - Local linear embedding (Roweis and Saul, 2000)
  - Local MDS (Chen and Buja, 2008)

Isometric feature mapping (ISOMAP)

- Isometric feature mapping (ISOMAP) constructs a graph to approximate the geodesic distance between points along the manifold.
- For each data point we find its neighbors—points within some small Euclidean distance of that point.
- We construct a graph with an edge between any two neighboring points.
- The geodesic distance between any two points is then approximated by the shortest path between points on the graph.

Local linear embedding

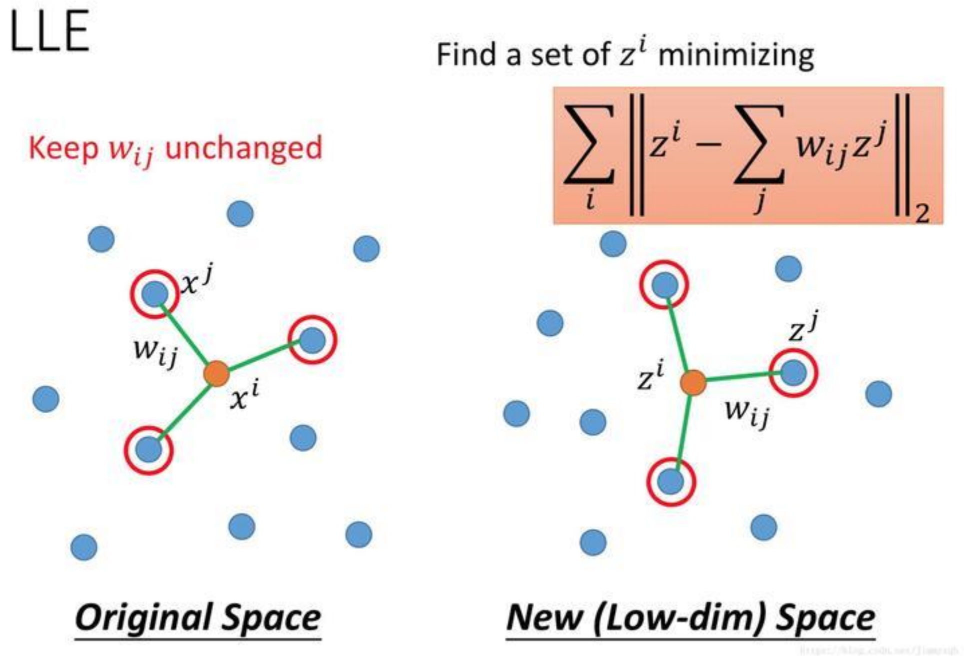
- Local linear embedding takes a very different approach, trying to preserve the local affine structure of the high-dimensional data.
- Each data point is approximated by a linear combination of neighboring points. Then a lower dimensional representation is constructed that best preserves these local approximations.

Local linear embedding

- For each data point x_i in p dimensions, we find its K-nearest neighbors $N(i)$ in Euclidean distance.
- We approximate each point by an affine mixture of the points in its neighborhood:

$$\min_{W_{ik}} \left\| x_i - \sum_{k \in N(i)} w_{ik} x_k \right\|^2$$

$$w_{ik} = 0, \quad k \notin N(i); \quad \sum_{k=1}^{N} w_{ik} = 1 .$$



- We must have K < p.

Local linear embedding

- Given w_{ik} , we find points $y_i \in R^d$ in a space of dimension d < p to minimize

$$\min_{y_i} \sum_{i=1}^{N} \left\| y_i - \sum_{k=1} w_{ik} y_k \right\|^2$$

- In compact form:

$$\min_{Y} tr\left[(Y - WY)^T (Y - WY) \right] = tr\left[Y^T (I - W)^T (I - W) Y \right]$$

where $W \in \mathrm{R}^{N \times N}, Y \in \mathrm{R}^{N \times d}$. The solution is the trailing eigenvectors of

$$\mathrm{M} = (I - W)^T (I - W)$$

- Since 1 is a trivial eigenvector with eigenvalue 0, we discard it and keep the next d. This has the side effect that $1^T Y = 0$

Local MDS

- Define N to be the symmetric set of nearby pairs of points; specifically a pair (i, i') is in N if point i is among the K-nearest neighbors of i', or vice-versa.

$$S(\{z_i\}_{i=1}^{N}) = \sum_{i,i' \in N} \left\| d_{ii'} - \|z_i - z_{i'}\| \right\|^2$$

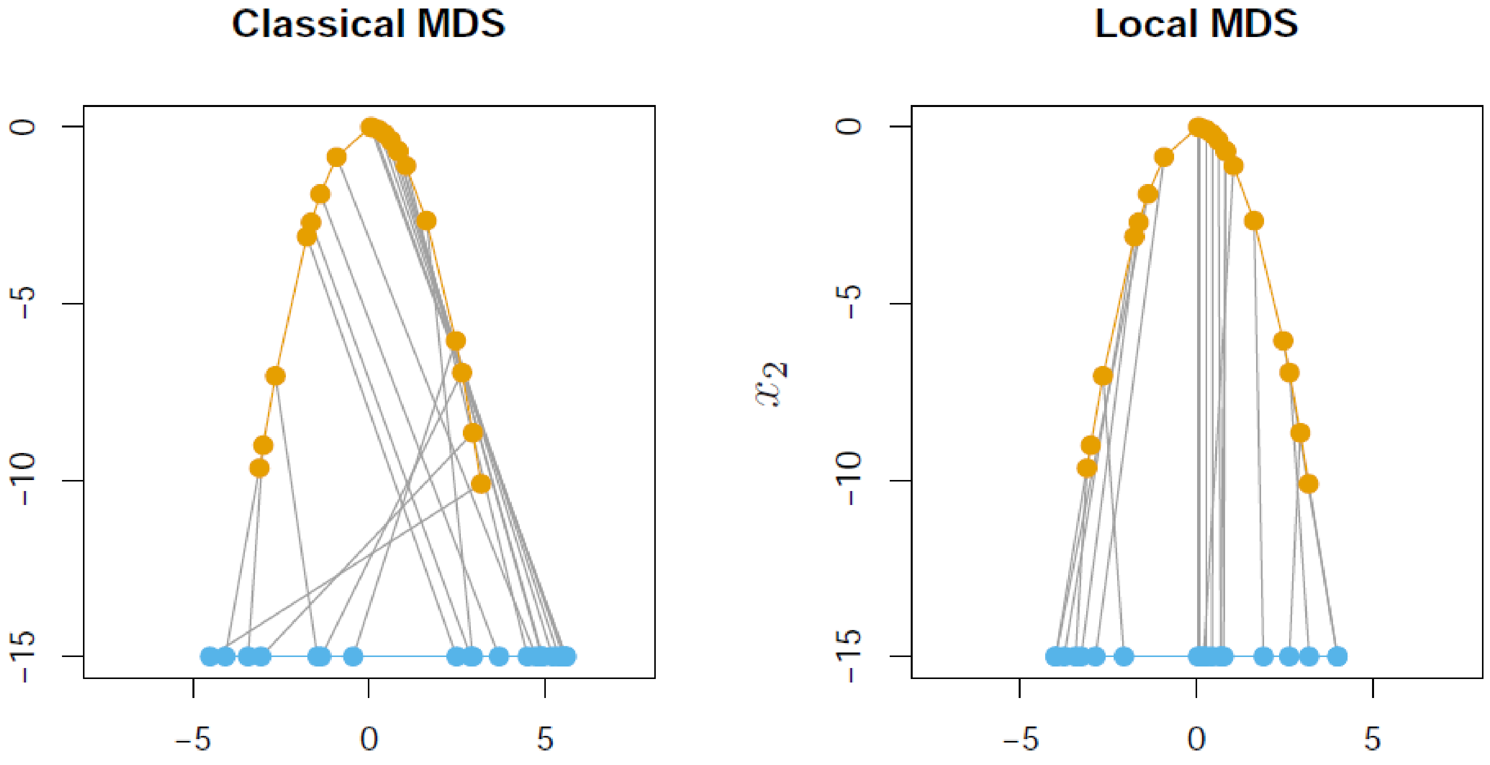
$$+ \sum_{i,i' \notin N} w \left\| D - \|z_i - z_{i'}\| \right\|^2$$

$$w \sim 1/D$$

Local MDS

- Take $\tau = 2wD$, for example: $\tau = 0.01$

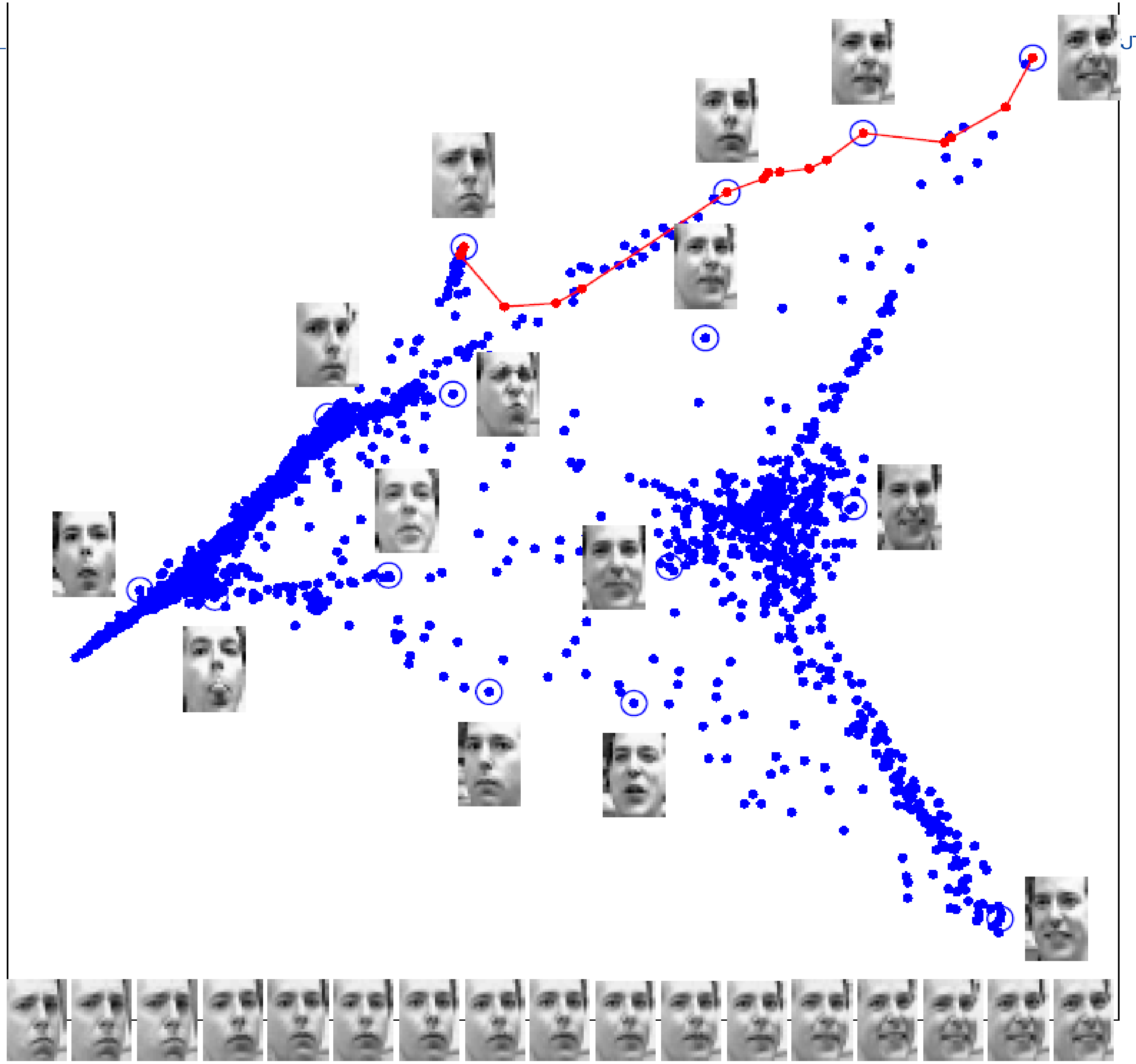
$$S(\{z_i\}_{i=1}^{N}) = \sum_{i,i' \in N} \left\| d_{ii'} - \| z_i - z_{i'} \| \right\|^2 - \tau \sum_{i,i' \notin N} \| z_i - z_{i'} \|$$



- LLE example

20X28 image

- Similar results by LMDS



Stochastic Neighbor Embedding

- For many real-world dataset with non-linear inherent structures(e.g. MNIST), both linear methods like PCA and classical manifold learning algorithms like ***Isomap*** and ***LLE*** fail.
- Why?   How to solve it?

(a) Visualization by Isomap.

(b) Visualization by LLE.

Stochastic Neighbor Embedding

- Solution:
  - SNE (Hinton and Roweis, 2002)
  - t-SNE (van der Maaten and Hinton, 2008)
- Problem to be solved:
  - Measure of similarity inherited
  - How to define data distribution
- Technical Approach:
  - KL divergence between the two probability distributions at each point
- Algorithm:
  - Iteratively learn low-dimensional embedding by minimizing the cost function

Stochastic Neighbor Embedding

- For each data point $x_i \in R^p$, we find points $y_i \in R^d$ in a space of dimension d < p to minimize *KL* divergence between two distributions.
- Similarity matrix at high dimension:

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}$$

- Similarity matrix at low dimension:

$$q_{j|i} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_j\|^2)}$$

- The cost function:

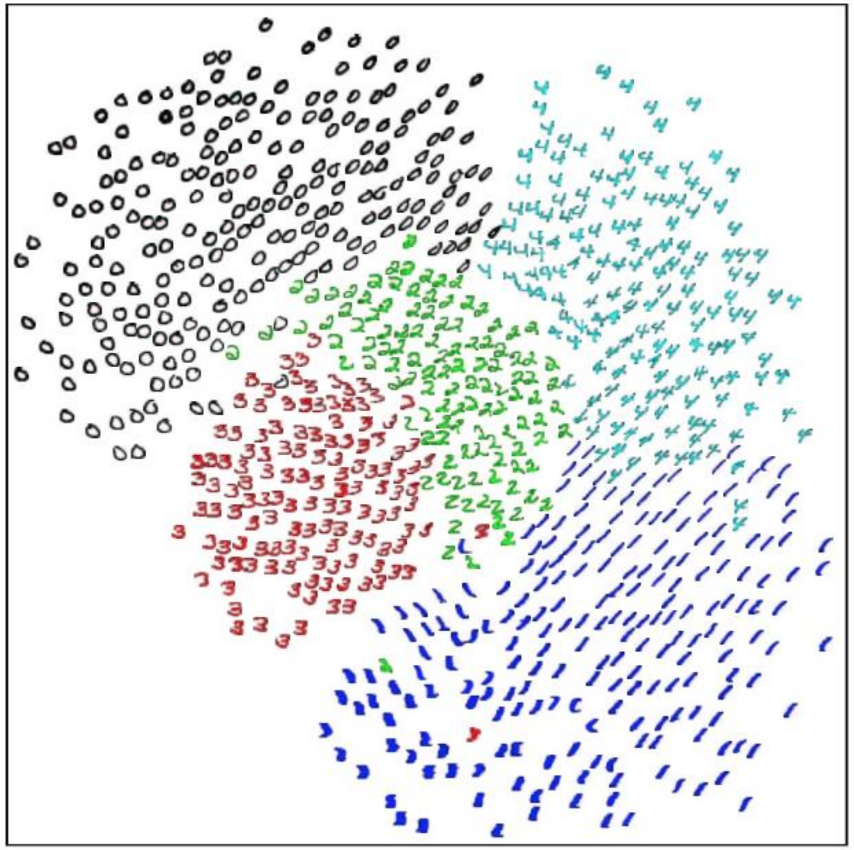
$$\mathrm{L} = \sum_i KL(P_i \mid Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

Stochastic Neighbor Embedding

- The cost function: $$L = \sum_i KL(P_i \,|\, Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$
- It has gradient:

$$\frac{\partial L}{\partial y_i} = 2\sum_j (y_i - y_j)(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})$$

- SNE suffers from the "crowding problem": The area of the 2D map that is available to accommodate moderately distant data points will not be large enough compared with the area available to accommodate nearby data points.



t-SNE: Student-t distribution SNE

- For each data point $x_i \in R^p$, we find points $y_i \in R^d$ in a space of dimension d < p to minimize *KL* divergence between two distributions.
- Student-t distribution SNE:
  - A symmetrized version of the SNE cost function with simpler gradients.
  - A Student-t distribution to compute the similarity in the low-dimensional
- Symetry :

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

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}$$

- A Student-t distribution for $y_i \in R^d$

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_i - y_j\|^2\right)^{-1}}$$

$$p_{st}(z \mid p) = \frac{\Gamma\left(\frac{p+1}{2}\right)\left(1 + \|z\|^2 / p\right)^{-\frac{p+1}{2}}}{\sqrt{p\pi}\,\Gamma(p/2)}$$

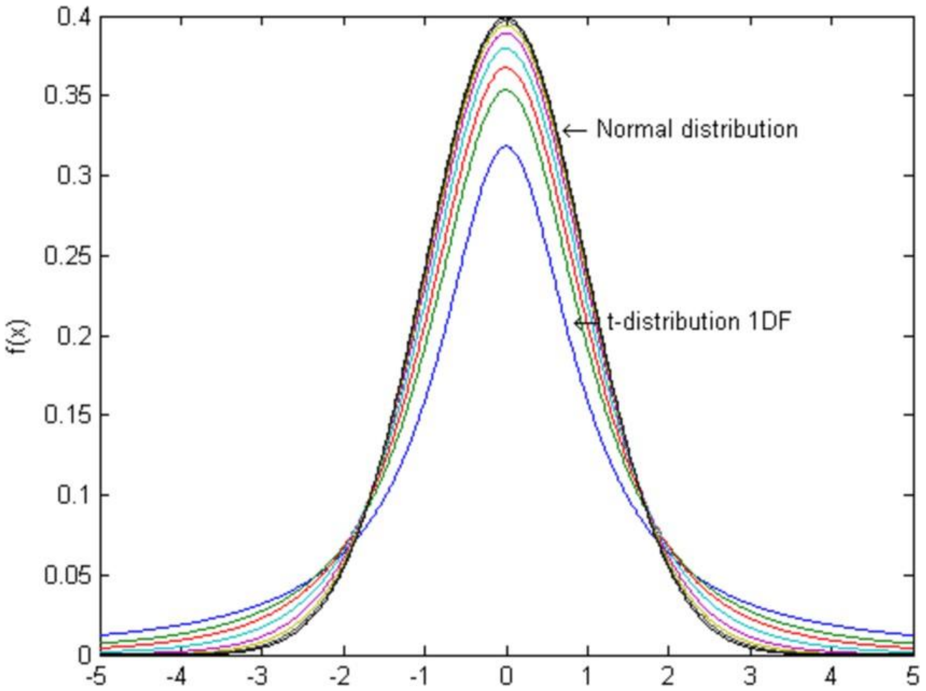
Student t-Distribution

- Suppose that $\{x_k\}_{k=1}^{N}$ are N samples from Gaussian distribution

$$\bar{x} = \frac{1}{N}\sum_{k=1}^{N} x_k; \qquad \bar{S}^2 = \frac{1}{N-1}\sum_{k=1}^{N}(x_k - \bar{x})(x_k - \bar{x})^T$$

- $\dfrac{\bar{x}-\mu}{\bar{S}/\sqrt{N}} = \dfrac{\bar{x}-\mu}{\sigma/\sqrt{N}}\dfrac{1}{\bar{S}/\sigma}$ follows the student t-distribution with N-1 dfs, i.e. p=N-1

$$p_{st}(z\,|\,p) = \frac{\Gamma\left(\frac{p+1}{2}\right)\left(1+\|z\|^2/p\right)^{-\frac{p+1}{2}}}{\sqrt{p\pi}\,\Gamma\left(p/2\right)}$$



t-SNE: Student-t distribution SNE

- Cost function:

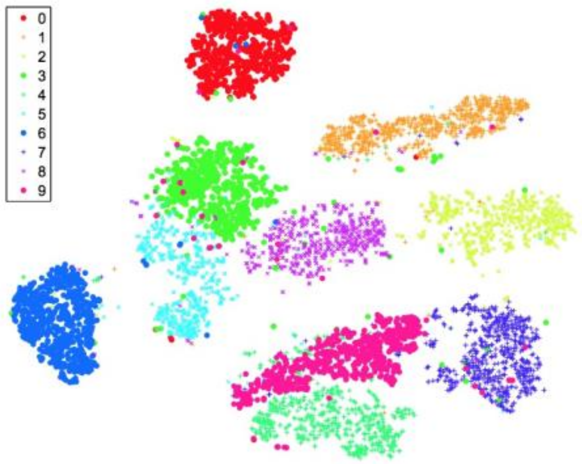
$$L = \sum_i KL(P_i \| Q_i) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

- The gradient

$$\frac{\partial L}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) \left(1 + \left\| y_i - y_j \right\|^2\right)^{-1} (y_i - y_j)$$

- The heavy tails of the normalized Student-t kernel allow dissimilar input objects $x_i, x_j \in R^p$ to be modeled by low-dimensional counterparts $y_i, y_j \in R^d$ that are too far apart because q_{ij} is not very small for two embedded points that are far apart.

Comparisons



(a) Visualization by t-SNE.

(a) Visualization by Isomap.

(b) Visualization by Sammon mapping.

(b) Visualization by LLE.

Unsupervised Learning

1. Introduction
2. Association Rules & Cluster Analysis
4. Self-Organizing Maps
5. Principal Components, Curves and Surfaces
6. Non-negative Matrix Factorization
7. Independent Component Analysis
8. Multidimensional Scaling
9. Nonlinear Dimension Reduction

10. The Google PageRank Algorithm

The Google PageRank Algorithm

- We suppose that we have N web pages and wish to rank them in terms of importance.
- The PageRank algorithm considers a webpage to be important if many other webpages point to it.
- The linking webpages that point to a given page are not treated equally: the algorithm also takes into account both the importance (PageRank) of the linking pages and the number of outgoing links that they have.

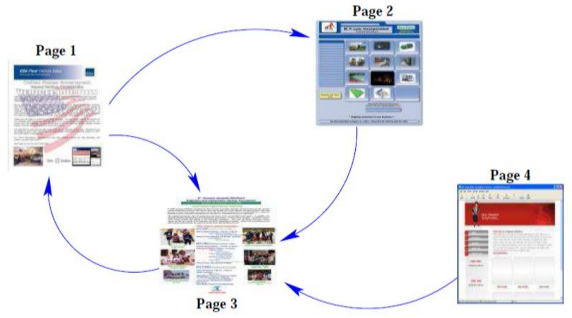


FIGURE 14.46. *PageRank algorithm: example of a small network*

PageRank Formulation

- Let $L_{ij} = 1$ if page j points to page i, and zero otherwise.
- Let $c_j = \sum_{i=1} L_{ij}$ equals the number of pages pointed to by page j (number of outlinks).

$L_{12} = 1$ or $L_{21} = 1$?

$c_1 = L_{11} + L_{21} + L_{31} + L_{41} = 2$

$c_3 = ?$

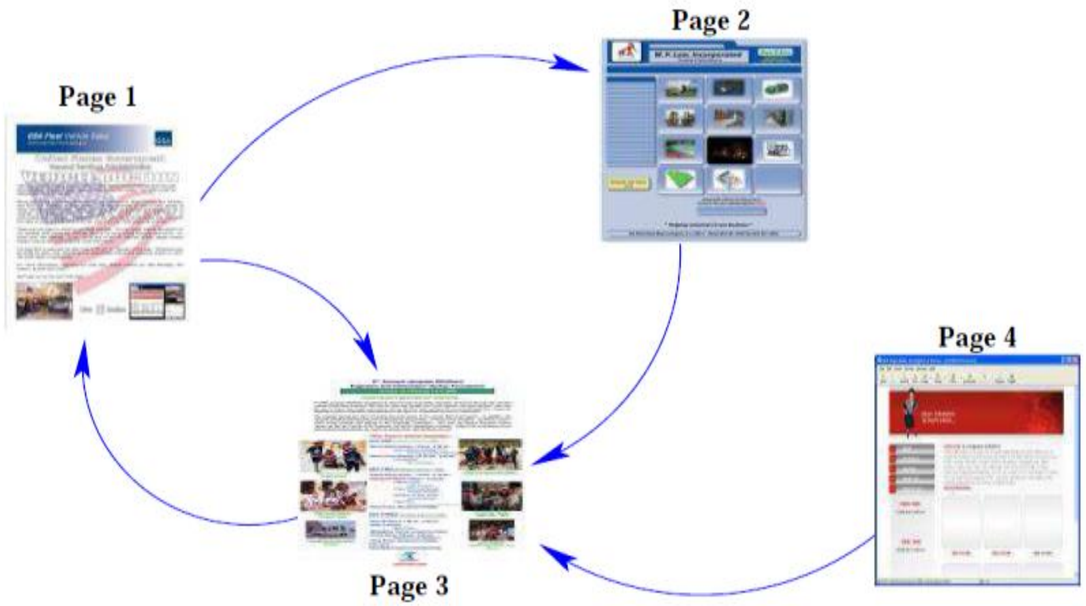


FIGURE 14.46. *PageRank algorithm: example of a small network*

PageRank Formulation

- **Assumption:** The importance of page i is the sum of the importance of other pages that point to that page.
- The **Google PageRanks** p_i are defined by the recursive relationship

$$p_i = (1-d) + d\sum_{j=1}^{N}\frac{L_{ij}}{c_j}p_j,$$

d is a positive constant (say $d=0.85$). In matrix form

$$\mathbf{p} = (1-d)\mathbf{e} + d\mathbf{L}\mathbf{D}_c^{-1}\mathbf{p}$$

$\mathbf{e}$ is a vector of N ones; $\mathbf{D}_c$=diag(c)

Page Rank Algorithm

$$\mathbf{p} = (1-d)\mathbf{e} + d\mathbf{L}\mathbf{D}_c^{-1}\mathbf{p}$$

- Introducing the normalization $\mathbf{e}^T\mathbf{p} = N$ (i.e., the average PageRank is 1), rewrite the above equation

$$\mathbf{p} = \left[(1-d)\mathbf{e}\mathbf{e}^T / N + d\mathbf{L}\mathbf{D}_c^{-1}\right]\mathbf{p} = \mathbf{A}\mathbf{p}$$

- Exploiting a connection with Markov chains (see below), it can be shown that the matrix $\mathbf{A}$ has a real eigenvalue equal to one, and one is its largest eigenvalue.

PageRank Algorithm

- The page rank solution

$$\mathbf{p} = \left[(1-d)\mathbf{e}\mathbf{e}^T / N + d\mathbf{L}\mathbf{D}_c^{-1}\right]\mathbf{p} = \mathbf{A}\mathbf{p}$$

$$\mathbf{e}^T\mathbf{p} = N$$

- Find by the power method: initialize p_0

$$\mathbf{p}_k \leftarrow \mathbf{A}\mathbf{p}_{k-1}; \qquad \mathbf{p}_k \leftarrow N\frac{\mathbf{p}_k}{\mathbf{e}^T\mathbf{p}_k}$$

PageRank Algorithm

- Viewing PageRank as a Markov chain makes clear why the matrix $\mathbf{A}$ has a maximal real eigenvalue of 1.
- Since $\mathbf{A}$ has positive entries with each column summing to one, Markov chain theory tells us that it has a unique eigenvector with eigenvalue one, corresponding to the stationary distribution of the chain (Bremaud, 1999).

Simple Example

$$L = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

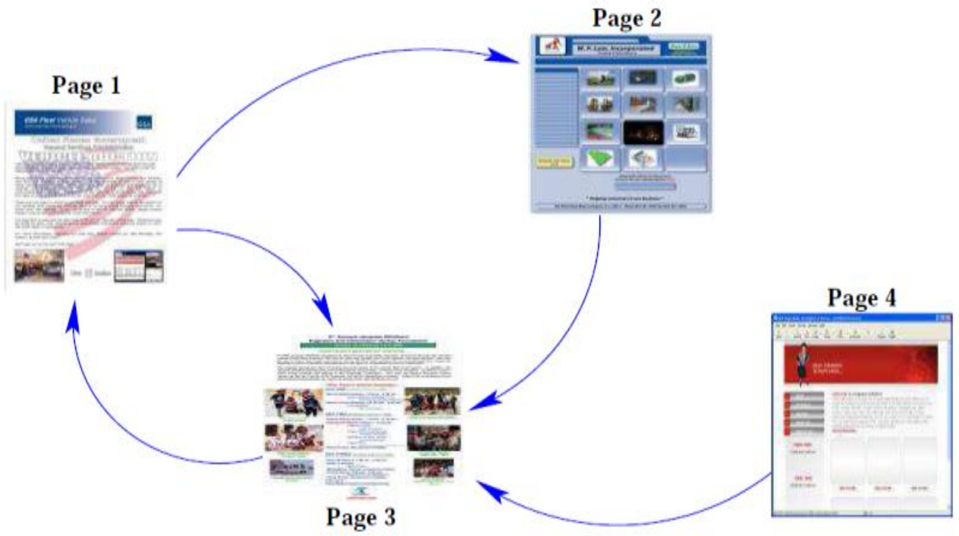


FIGURE 14.46. *PageRank algorithm: example of a small network*

The number of outlinks $c = (2,1,1,1)$.

$$p = (1.49, 0.78, 1.58, 0.15)$$

Summary

1. Introduction
2. Association Rules & Cluster Analysis
4. Self-Organizing Maps
5. Principal Components, Curves and Surfaces
6. Non-negative Matrix Factorization
7. Independent Component Analysis
8. Multidimensional Scaling
9. Nonlinear Dimension Reduction
10. The Google PageRank Algorithm

The End of Talk