第十一章：特征提取与选择
Part 2

张煦尧(xyz@nlpr.ia.ac.cn)

2018年1月10日

助教：何文浩(wenhao.he@nlpr.ia.ac.cn)
杨红明(hongming.yang@nlpr.ia.ac.cn)

Class Separation Problem

$$\max_{W \in \mathbb{R}^{d \times d'}} \quad \mathrm{tr}\left(W^\top S_b W\right) \quad \text{s.t.} \quad W^\top S_w W = I$$

$$\max_{W \in \mathbb{R}^{d \times d'}} \quad \mathrm{tr}\left(W^\top W_{\text{whiten}}^\top S_b W_{\text{whiten}} W\right)$$
$$\text{s.t.} \quad W^\top W = I \,.$$

is equivalent to

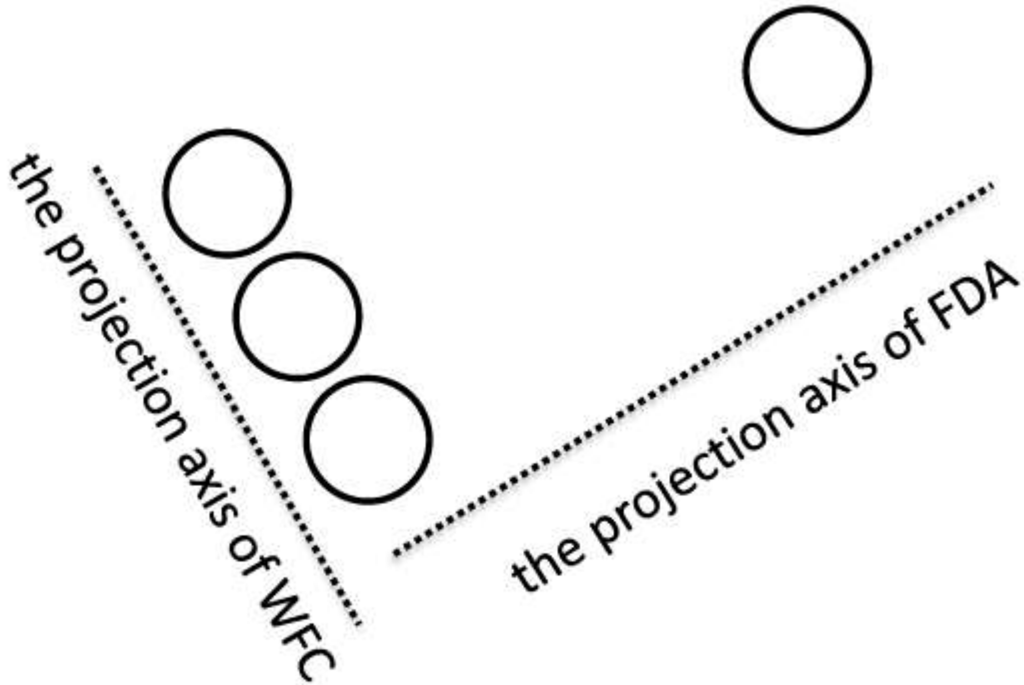
$$\boxed{\max_{W \in \mathbb{R}^{d \times d'}} \quad \sum_{i,j=1}^{K} p_i p_j \Delta_{ij} \qquad \text{s.t.} \quad W^\top W = I \,.}$$

$$\Delta_{ij} = \left\| W^\top W_{\text{whiten}}^\top (\mu_i - \mu_j) \right\|_2^2$$

- ✓ Maximize the sum of all the pairwise distances in the reduced space
- ✓ PCA transformation among $W_{\text{whiten}}^\top \mu_1, \cdots, W_{\text{whiten}}^\top \mu_K$
- ✓ PCA is a global model
- ✓ The local information for distinguishing one class from another may be lost

Class Separation Problem

$$\max_{W \in \mathbb{R}^{d \times d'}} \sum_{i,j=1}^{K} p_i p_j \triangle_{ij} \quad \text{s.t.} \quad W^\top W = I \ .$$



Class Separation Problem

$$\max_{W \in \mathbb{R}^{d \times d'}} \sum_{i,j=1}^{K} p_i p_j \triangle_{ij} \quad \text{s.t.} \quad W^\top W = I \,.$$

To solve the class separation problem:

- ✓ maximize the geometric mean [Tao2009PAMI]

$$\{\max \textstyle\sum_{i \neq j} p_i p_j \log \triangle_{ij}\}$$

- ✓ maximize the harmonic mean [Bian2008ICPR]

$$\{\max - \textstyle\sum_{i \neq j} p_i p_j \triangle_{ij}^{-1}\}$$

- ✓ maximize the minimal distance [Zhang2010NIPS, Xu2010ICPR, Yu2011PR, Bian2011PAMI]

$$\{\max \min_{i \neq j} \triangle_{ij}\}$$

- ✓ maximize all the distances simultaneously [Abou-Moustafa2010CVPR]

$$\{\max \triangle_{12}, \max \triangle_{13}, \cdots, \max \triangle_{K-1,K}\}$$

Class Separation Problem

Method	Optimization	Experiments (K classes)
[30]	steepest gradient	UCI and USPS ($K \leq 10$)
[3]	conjugate gradient	UCI and Objects ($K \leq 20$)
[39]	constrained concave-convex procedure (CCCP)	UCI and Face ($K \leq 100$)
[34]	semi-definite programming (SDP)	UCI and Face ($K \leq 40$)
[4]	Sequential SDP	UCI and Face ($K \leq 50$)
[1]	gradient descent	Image and UCI ($K \leq 40$)

✓complex iterative optimization procedures
✓not scalable for large category (e.g. thousands of classes) problems

Weighted Fisher Criterion

$$\max_{W \in \mathbb{R}^{d \times d'}} \sum_{i,j=1}^{K} f_{ij} p_i p_j \Delta_{ij} \quad \text{s.t.} \quad W^\top W = I$$

- ✓ weighting function $f_{ij} \geq 0$
- ✓ Solving the class separation problem: setting larger weights for the most confusable classes
- ✓ still an eigen-decomposition problem

Xu-Yao Zhang, Cheng-Lin Liu. Evaluation of weighted Fisher criteria for large category dimensionality reduction in application to Chinese handwriting recognition. Pattern Recognition, vol. 46, pp. 2599-2611, September 2013.

Weighted Fisher Criterion

$$\max_{W \in \mathbb{R}^{d \times d'}} \quad \sum_{i,j=1}^{K} f_{ij} p_i p_j \triangle_{ij} \qquad \text{s.t.} \quad W^\top W = I$$

is equivalent to

$$\max_{W \in \mathbb{R}^{d \times d'}} \quad \mathrm{tr}\left(W^\top \widehat{S_b} W\right) \qquad \text{s.t.} \quad W^\top W = I$$

$$\widehat{S_b} = \sum_{i,j=1}^{K} f_{ij} p_i p_j (\widehat{\mu_i} - \widehat{\mu_j})(\widehat{\mu_i} - \widehat{\mu_j})^\top$$

$$\widehat{\mu_i} = W_{\text{whiten}}^\top \mu_i$$

eigen-decomposition of $\widehat{S_b}$

Weighted Fisher Criterion

(1) Step 1 (Algorithm 1): $W_{\text{whiten}} \in \mathbb{R}^{d \times d}$

(2) Step 2 (Algorithm 2): $W_{\text{WFC}} \in \mathbb{R}^{d \times d'}$

(3) Final transformation $W_{\text{final}} = W_{\text{whiten}} W_{\text{WFC}} \in \mathbb{R}^{d \times d'}$

Key problem: the definition of the weighting matrix

$$F = \{f_{ij}\} \in \mathbb{R}^{K \times K}$$

Weighting Functions

Five weighting functions are compared

(1) FDA $\quad \mathrm{FDA}: \quad f_{ij} = 1, \forall i, j = 1, \cdots, K$

(2) aPAC [Loog2001PAMI]

$$f_{ij} = \frac{1}{2d_{ij}^2} \operatorname{erf}\left(\frac{d_{ij}}{2\sqrt{2}}\right)$$

$$d_{ij} = \|\widehat{\mu_i} - \widehat{\mu_j}\|_2 = \left\|W_{\text{whiten}}^\top(\mu_i - \mu_j)\right\|_2$$

Weighting Functions

(3) POW [Lotlikar2000PAMI]

$$\text{POW}: \quad f_{ij} = d_{ij}^{-m}$$

(4) CDM: confused distance maximization (CDM)

confusion matrix

$$\text{CDM}: \quad f_{ij} = \begin{cases} \dfrac{N_{i \rightsquigarrow j}}{N_i}, & i \neq j \\ 0, & i = j \end{cases}$$

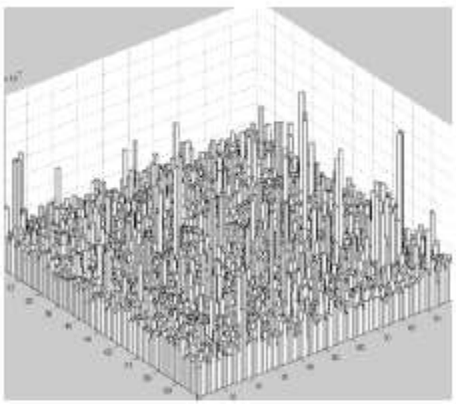
Weighting Functions

(5) KNN

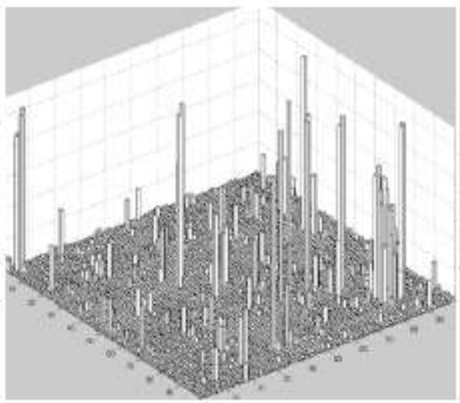
$$\text{KNN}: \quad f_{ij} = \begin{cases} 1, & \text{if } \widehat{\mu_j} \in \text{KNN}(\widehat{\mu_i}) \\ 0, & \text{else} \end{cases}$$

- ✓ focusing on the nearest class pairs, and removing the influence of the large-distance class pairs
- ✓ the geometrical relationship of different classes is preserved by the connection and propagation between each class and its nearest neighbors
- ✓ the fast construction and sparsity of the weighting matrix can significantly reduce the computational complexity of WFC
- ✓ the KNN weighting matrix is nearly space invariant, that means the KNN relationship between the class pairs is nearly the same either in the original feature space or the final reduced space

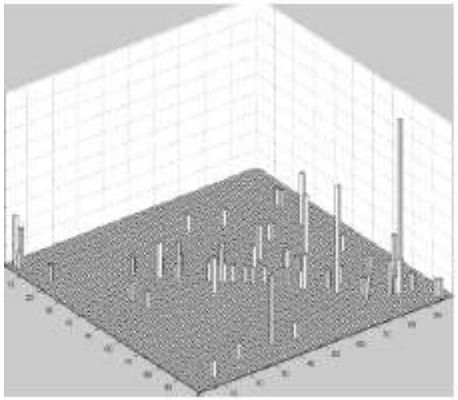
Comparing Weighting Functions



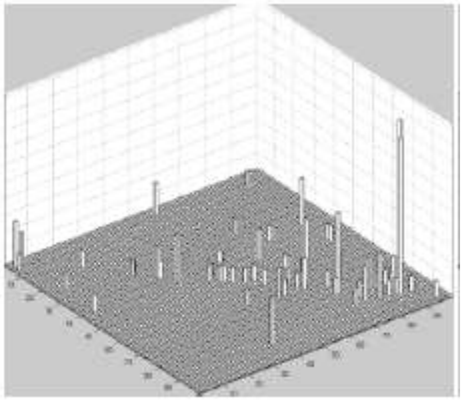
aPAC



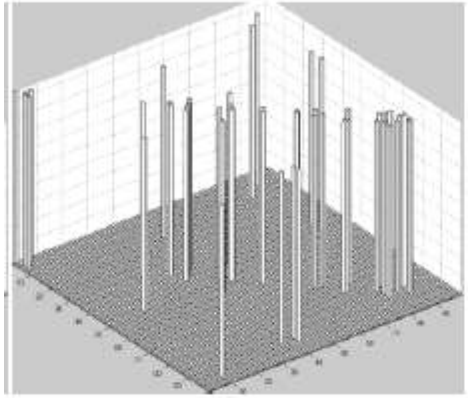
POW9



CDM (NCM)



CDM (MQDF)



KNN5

Comparing Weighting Functions

	class separation	locality	classifier-specific	space invariant
FDA				√
aPAC	√			
POW	√			
CDM	√	√	√	
KNN	√	√		√

- ✓ Class separation: by setting larger weights for the most confusable classes, aPAC, POW, CDM and KNN can solve the class separation problem.
- ✓ Locality: the weighting matrices of CDM and KNN are much sparser than other weighting functions
  - ➢ focusing on the most confusable classes
  - ➢ fast computation of the between-class scatter matrix
- ✓ Classifier-specific: the confusion matrix used by CDM is classifier-specifc
- ✓ Space invariant: the weighting matrices in the original and final reduced space are nearly the same.

Weighting Spaces

- learn a transformation from $\mathbb{R}^d$ to $\mathbb{R}^{d'}$.
- the classification performance is directly evaluated in $\mathbb{R}^{d'}$.

The optimal weighting function
should be defined in the **final reduced space (FRS)** $\mathbb{R}^{d'}$.

However, the two following problems are in a chicken-and-egg flavor

1. the WFC transformation learning $W \in \mathbb{R}^{d \times d'}$
2. the weighting matrix estimation in $\mathbb{R}^{d'}$.

Therefore, we propose three weighting spaces to approximate the weighting functions in FRS.

Weighting Spaces

(1) original space:

- define weighting functions in $\mathbb{R}^d$
- lowest computational complexity
- may be significantly different from FRS

(2) low-dimensional space:

- estimate a weighting matrix in $\mathbb{R}^d$
- learn a WFC transformation $W \in \mathbb{R}^{d \times d'}$
- re-estimate a weighting matrix in $\mathbb{R}^{d'}$
- re-learn the WFC with the new weighting matrix

$$W \in \mathbb{R}^{d \times d'} \rightleftarrows \mathbb{R}^{d'}$$

convergence not guaranteed

Weighting Spaces

(3) fractional space [Lotlikar2000PAMI]

- The dimensionality is reduced in small fractional steps
- the relevant class pairs to be more correctly weighted

$$\mathbb{R}^d \xrightarrow[\mathrm{WFC}]{F} \mathbb{R}^{d-t} \xrightarrow[\mathrm{WFC}]{F} \mathbb{R}^{d-2t} \ldots \xrightarrow[\mathrm{WFC}]{F} \mathbb{R}^{d'}$$

- the weighting matrix is estimated in the higher input space
- WFC is used to reduce the dimensionality by a small step t
- the weighting matrix is re-estimated in the lower output space.

✓ $t < 1$ means many sub-steps are involved to reduce the dimensionality by 1

✓ To lighten the computation burden for large category applications, we only consider the fractional-step t to be integer, (e.g. 1, 5, 10).

Comparing Weighting Spaces

- ✓ All the five weighting functions (FDA, aPAC, POW, CDM and KNN) can be defined in the three weighting spaces.

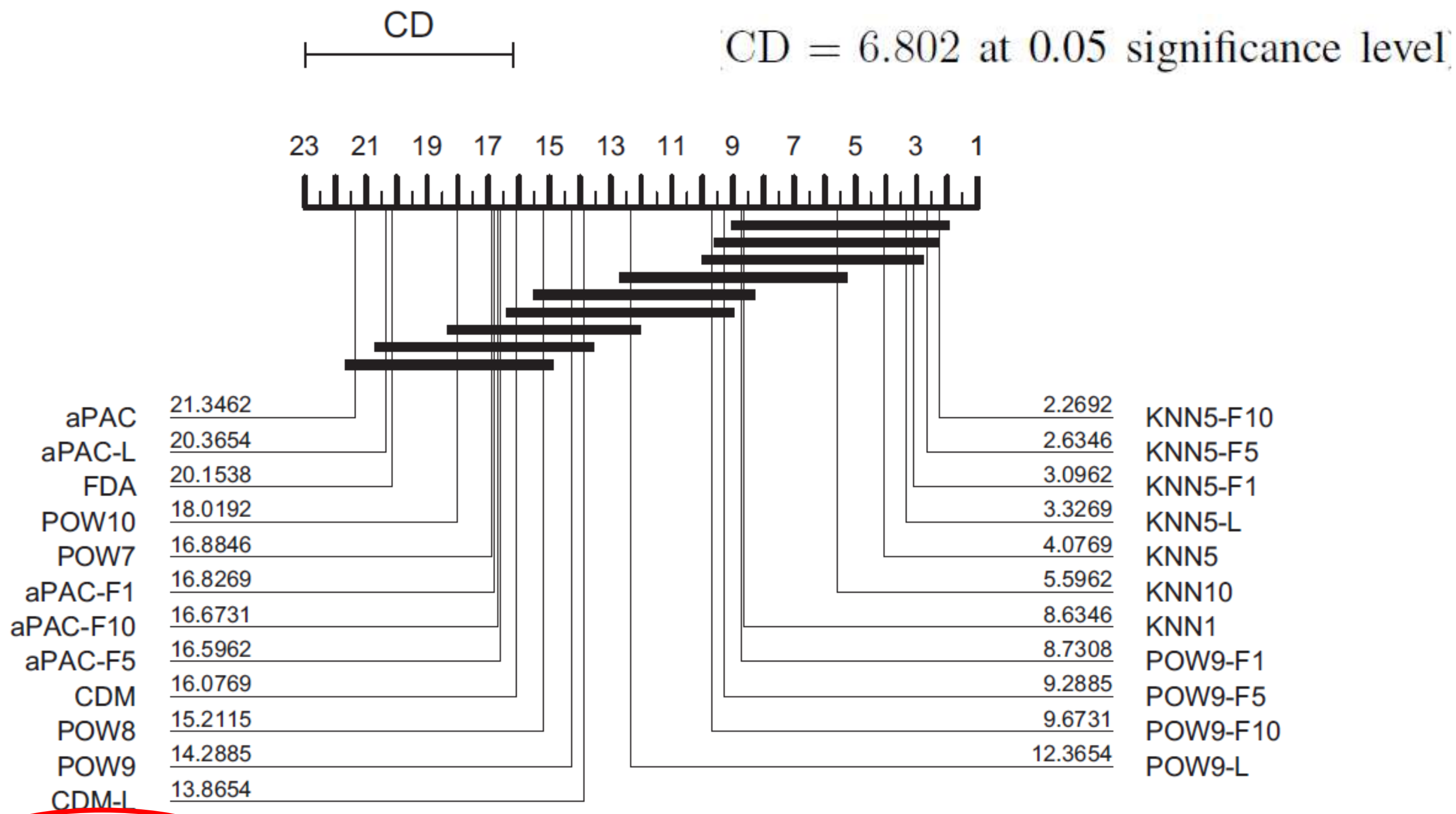
- ✓ The three weighting spaces have increasing computational complexities, but lead to better approximations of the weighting function in the FRS.

- ✓ If the weighting function is not changed in different weighting spaces (space invariant), then we can simply define the weighting function in the original space which has the lowest computational complexity.

Comparison

Totally 23 models with 26 evaluations.

We compare their average ranks [Demsar2009JMLR].



The critical difference (CD) diagram of different methods.

Comparison

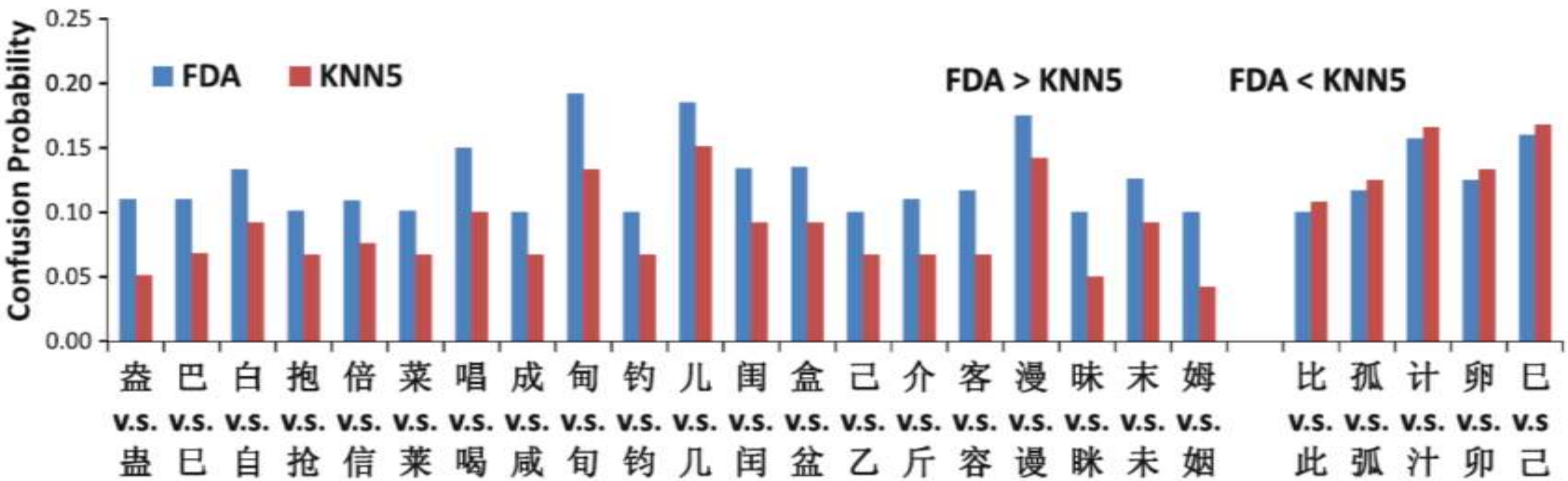
$$\mathrm{aPAC} \xrightarrow{1.1924} \mathrm{FDA} \xrightarrow{4.0769} \mathrm{CDM}$$

$$\xrightarrow{1.7884} \mathrm{POW9} \xrightarrow{10.2116} \mathrm{KNN5}$$

$$\mathrm{aPAC} \xrightarrow{0.9808} \mathrm{aPAC\text{-}L} \xrightarrow{3.7692} \mathrm{aPAC\text{-}F5}\,,$$

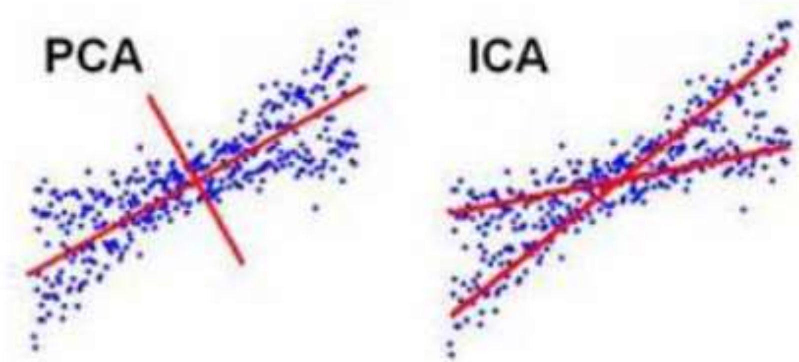
$$\mathrm{POW9} \xrightarrow{1.9231} \mathrm{POW9\text{-}L} \xrightarrow{3.6346} \mathrm{POW9\text{-}F1}\,,$$

$$\mathrm{KNN5} \xrightarrow{0.7500} \mathrm{KNN5\text{-}L} \xrightarrow{1.0577} \mathrm{KNN5\text{-}F10}\,.$$



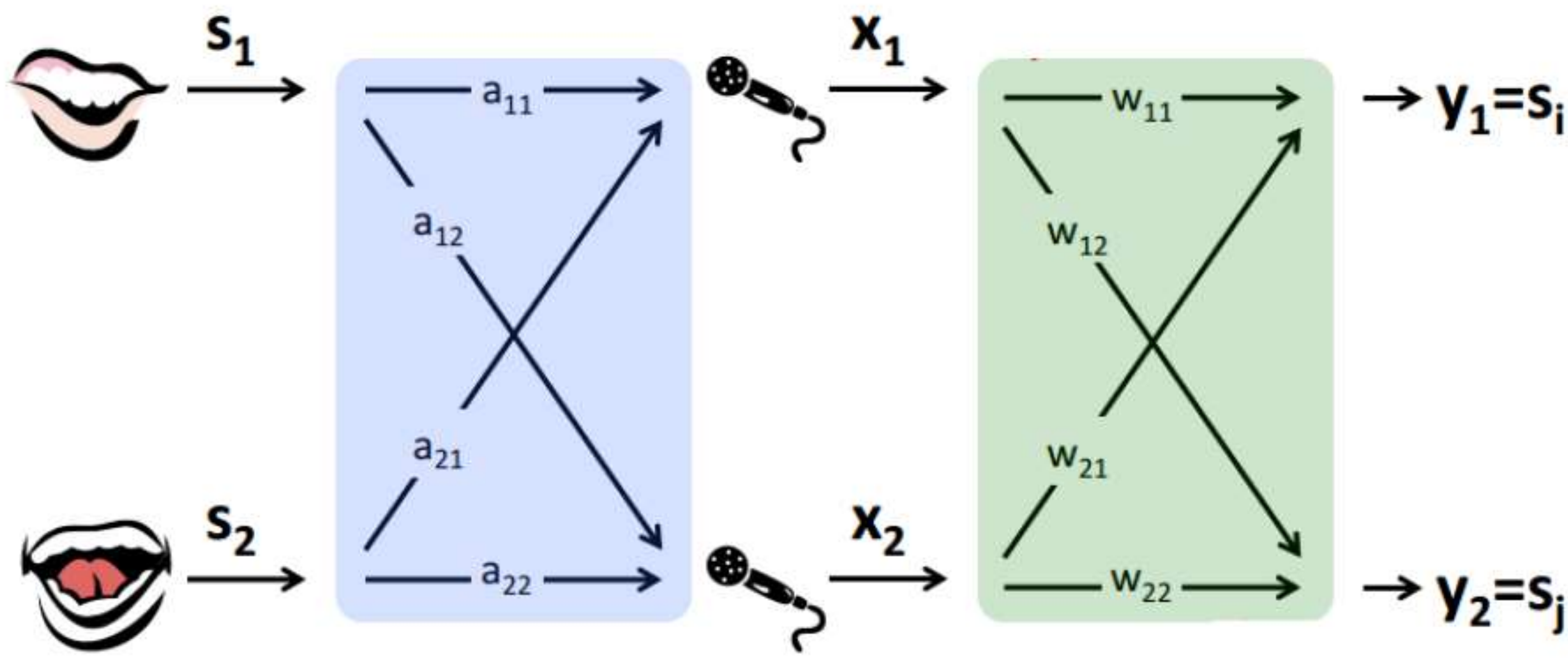
Independent Component Analysis (ICA)

- 独立成分分析 ICA
  - □ 求解线性变换 $Y = WX$，使得 $Y=\{y1,y2,\cdots,ym\}$ 各个分量之间相互独立
  - □ 和PCA不同，ICA追求的是输出的变量相互独立，而非仅不相关，因此，ICA需要利用数据分布的高阶统计信息而非仅二阶信息
  - □ 在很多应用中，ICA提取的特征好于PCA

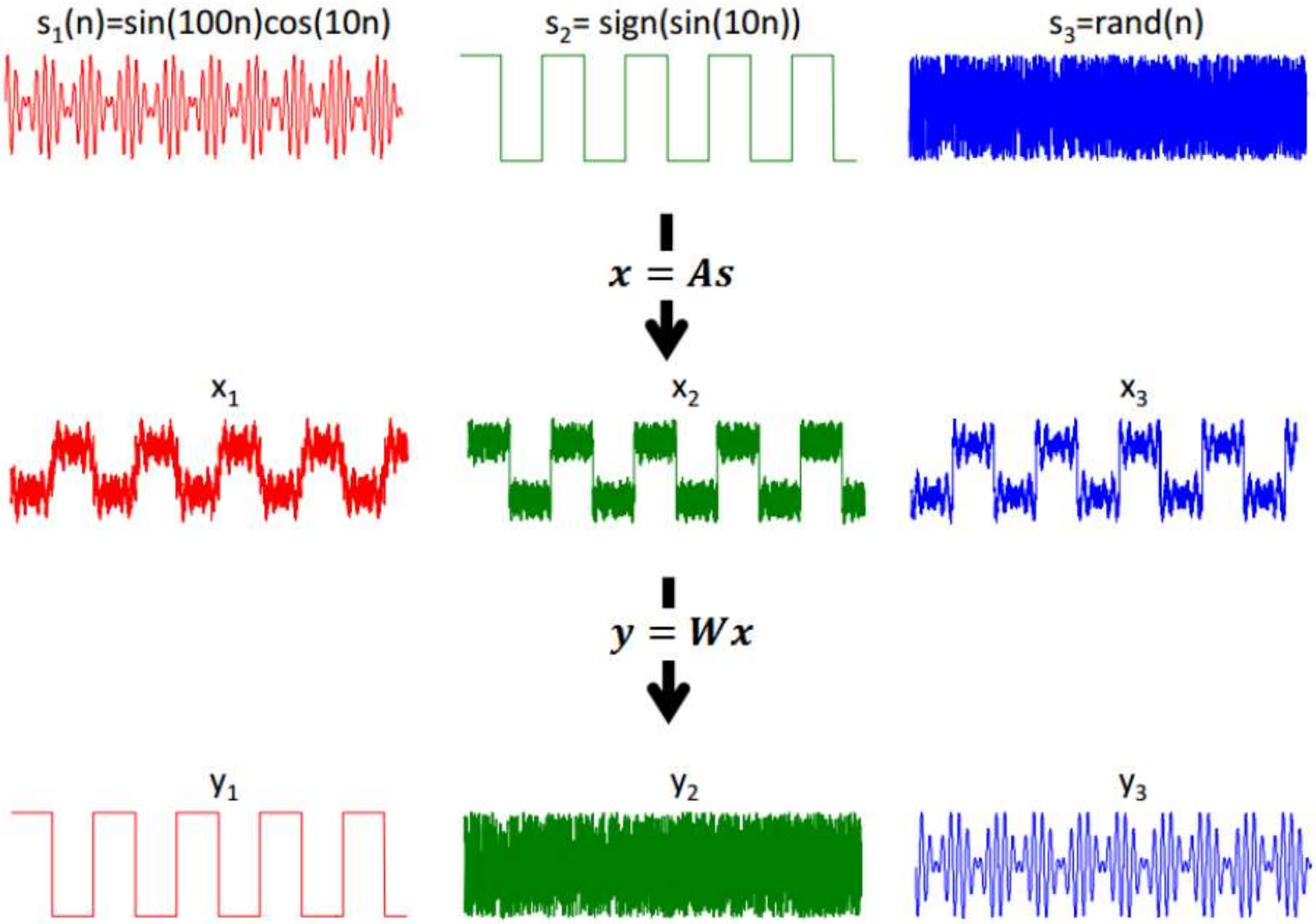


Independent Component Analysis (ICA)

- 鸡尾酒会问题（cocktail party problem）
  - 在一个酒会中，人们可以分辨出不同的声音，



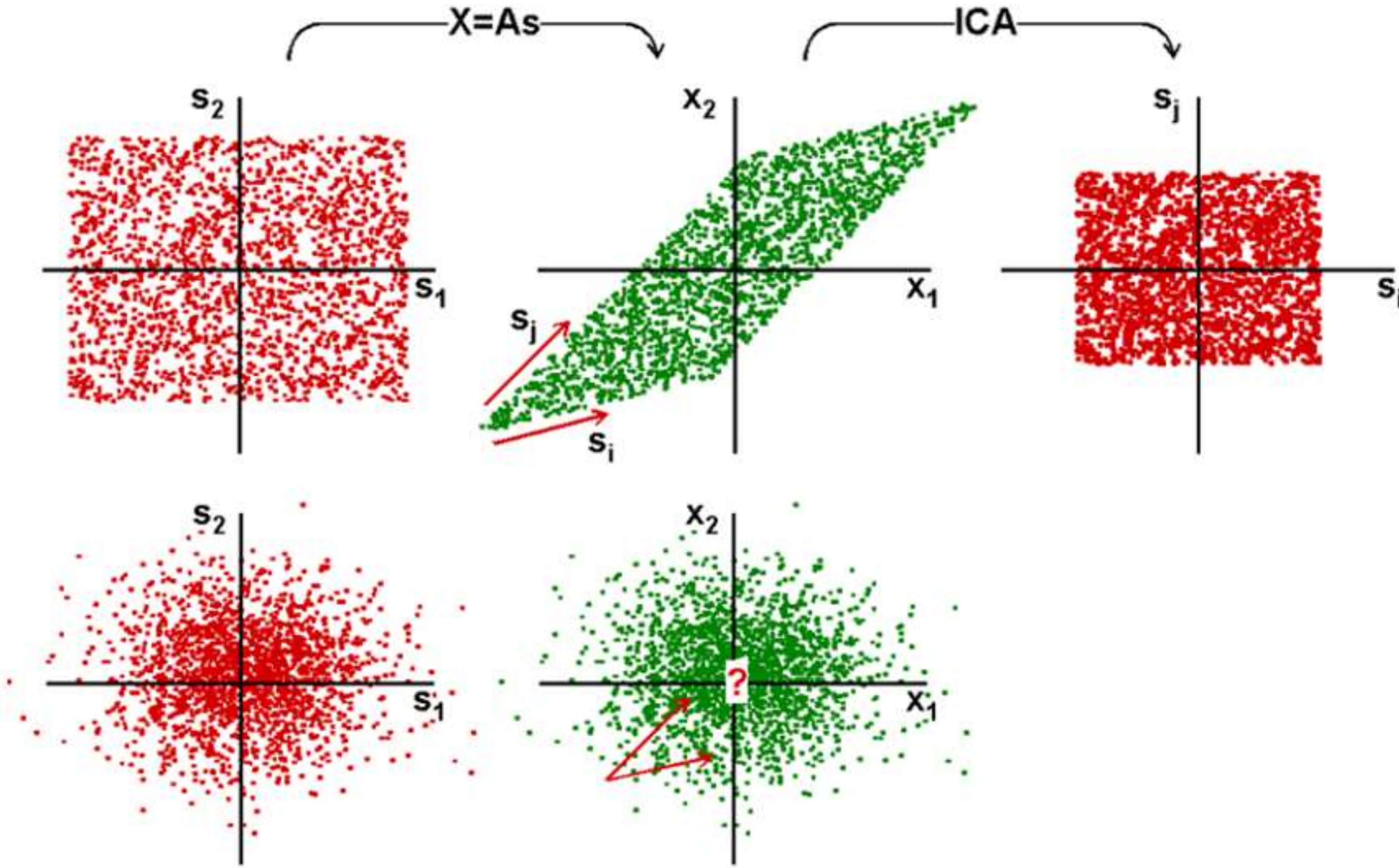
Independent Component Analysis (ICA)



Independent Component Analysis (ICA)

- 由于W和Y都是未知的，导致ICA的两类不确定性
  - Y的幅值Scale不确定，一般Y的方差限制为1
  - Y的排列顺序不确定
- 独立 vs 不相关
  - 独立： $p(y_1, y_2) = p(y_1)p(y_2)$
  - 不相关： $E(y_1 y_2) = 0$
  - 独立意味着不相关，但反之不亦然
  - 高斯分布意味着独立等价不相关
- 信号需要是非高斯的
  - 高斯的话可以利用PCA
  - 高斯分布的对称性导致ICA失效

Independent Component Analysis (ICA)



Independent Component Analysis (ICA)

- 非高斯最大化 Non-Gaussianity Maximization
- ***y = Wx = WAs*** 每一维y可以看成s的线性组合
- 根据中心极限定理，y要比s更趋于高斯分布，而只有当y等于s的时候才具有最低的高斯性
  - ✓ The sum of independent variables converges to the normal distribution
- 非高斯性一般利用信号的四阶统计量，峰度（kurtosis）来衡量

峰度（kurtosis）最大化

- 在统计学中，峰度（Kurtosis）衡量实数随机变量概率分布的峰态。峰度高就意味着方差增大是由低频度的大于或小于平均值的极端差值引起的。
- 4阶统计量

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

$$Z = \frac{X-\mu}{\sigma}, \qquad \kappa = E(Z^4) = \operatorname{var}(Z^2) + [E(Z^2)]^2 = \operatorname{var}(Z^2) + 1,$$

$$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^2\right)^2} - 3$$

- 减3是为了让正态分布的峰度为0

Fast ICA Algorithm

- Given whitened data $\mathbf{z}$
- Estimate the 1^{st} ICA component:

Probably the most famous ICA algorithm

$\star\ y = \mathbf{w}^T\mathbf{z},\ \|\mathbf{w}\| = 1, \qquad \Leftarrow \mathbf{w}^T = 1^{st}$ row of $\mathbf{W}$

$$\boxed{\begin{array}{l}\star \text{ maximize kurtosis } f(\mathbf{w}) \doteq \kappa_4(y) \doteq \mathbb{E}[y^4]\text{-}3 \\ \qquad \text{with constraint } h(\mathbf{w}) = \|\mathbf{w}\|^2 - 1 = 0\end{array}}$$

$\star$ At optimum $f'(\mathbf{w}) + \lambda h'(\mathbf{w}) = \mathbf{0}^T$ (λ Lagrange multiplier)

$$\Rightarrow 4\mathbb{E}[(\mathbf{w}^T\mathbf{z})^3\mathbf{z}] + 2\lambda\mathbf{w} = 0$$

Solve this equation by Newton–Raphson's method.

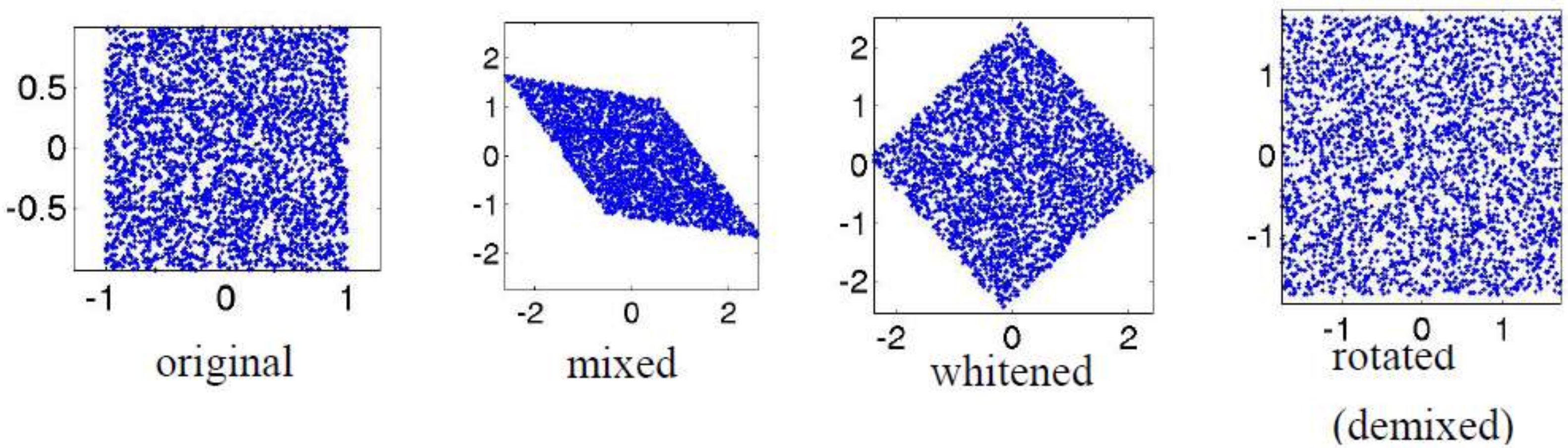
Independent Component Analysis (ICA)

ICA task: Given **x**,

- ❑ find **y** (the estimation of **s**),
- ❑ find **W** (the estimation of $\mathbf{A}^{-1}$)

ICA solution: **y=Wx**

- ❑ Remove mean, E[**x**]=0
- ❑ Whitening, $E[\mathbf{x}\mathbf{x}^T]=\mathbf{I}$
- ❑ Find an orthogonal **W** optimizing an objective function



2D Extensions

Tensor Data

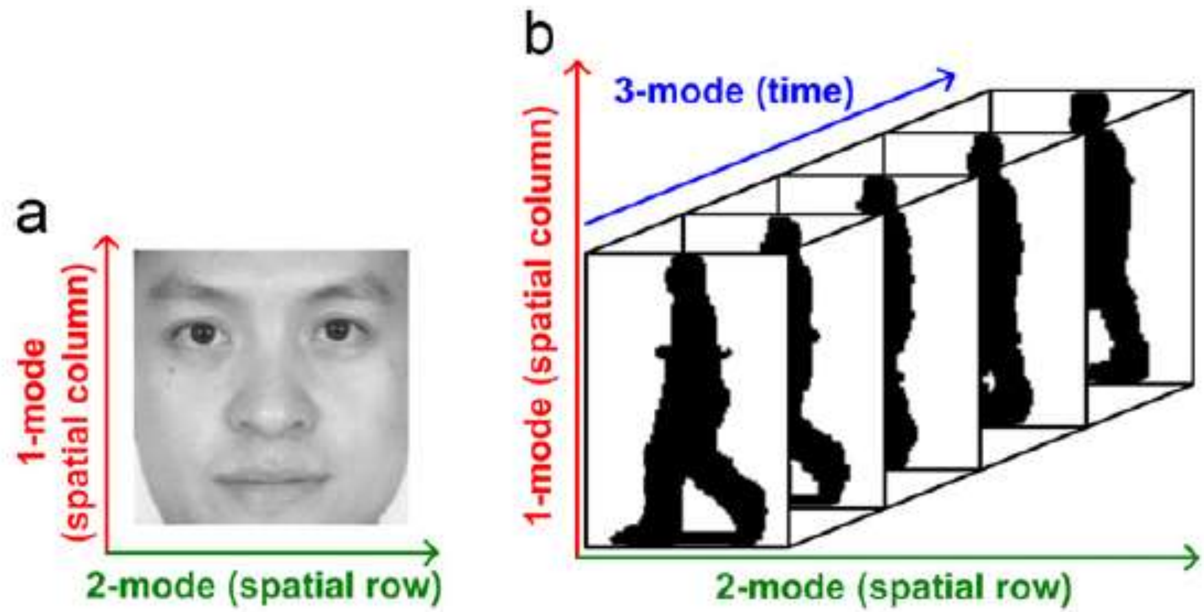


Fig. 1. Illustration of real-world data in their natural tensor representation: (a) a 2D face and (b) a 3D silhouette sequence.

2D PCA (Yang)

- ***A*** be a ***m*n*** random matrix.
- Learn a lot of ***n*1*** vectors ***x1,x2,…,xd***
- To form a ***m*d*** feature matrix: ***[Ax1,Ax2,…,Axd]***

- Image covariance (scatter) matrix

$$\mathbf{G}_t = \frac{1}{M}\sum_{j=1}^{M}(\mathbf{A}_j - \bar{\mathbf{A}})^T(\mathbf{A}_j - \bar{\mathbf{A}}).$$

J. Yang, D. Zhang, A.F. Frangi, J.-y. Yang, Two-Dimensional PCA: A New Approach to Appearance-Based Face Representation and Recognition, *IEEE Trans. PAMI*, 2004.

2D PCA (Yang)

- Solution: $$J(\mathbf{X}) = \mathbf{X}^T \mathbf{G}_t \mathbf{X},$$

$$\begin{cases} \{\mathbf{X}_1, \cdots, \mathbf{X}_d\} = \arg\max J(\mathbf{X}) \\ \mathbf{X}_i^T \mathbf{X}_j = 0, i \neq j, i, j = 1, \cdots, d. \end{cases}$$

- Feature extraction:

$$\mathbf{Y}_k = \mathbf{A}\mathbf{X}_k, k = 1, 2, \cdots, d.$$

- Feature matrix

$$\mathbf{B} = [\mathbf{Y}_1, \cdots, \mathbf{Y}_d].$$

- Classification:

$$d(\mathbf{B}_i, \mathbf{B}_j) = \sum_{k=1}^{d} \left\| \mathbf{Y}_k^{(i)} - \mathbf{Y}_k^{(j)} \right\|_2,$$

2D PCA (Yang)

- Experiments: 92*112 pixels face image

# Training samples / class	1 *	2 *	3	4 *	5
PCA (Eigenfaces)	66.9 (39)	84.7 (79)	88.2 (95)	90.8 (60)	93.5 (37)
2DPCA	76.7 (112×2)	89.1 (112×2)	91.8 (112×6)	95.0 (112×5)	96.0 (112×3)

- Here, 2D PCA is equal to line-based PCA

L. Wang, X. Wang, X. Zhang, J. Feng, The equivalence of two-dimensional PCA to line-based PCA, *Pattern Recognition Letters*, 2005.

- block-based PCA may work even better!

2D PCA (Ye)

- Each datum is represented by a matrix and the collection of data is represented by a sequence of matrices.
- ***N*** data points:

$$A_i \in R^{r\times c}, \text{ for } i = 1, \cdots, n$$

- Generalized low rank approximation:

$$\min_{\substack{L \in R^{r\times \ell_1}\,:\; L^T L = I_{\ell_1} \\ R \in R^{c\times \ell_2}\,:\; R^T R = I_{\ell_2} \\ \boxed{D_i \in R^{\ell_1\times \ell_2}}\,:\; i = 1, \cdots, n}} \sum_{i=1}^{n} ||A_i - LD_iR^T||_F^2.$$

Left transformation

Right transformation

J. Ye, Generalized Low Rank Approximations of Matrices, *ICML 2004*. (Outstanding Student Paper Award)

2D PCA (Ye)

- Given ***L*** and ***R*** $D_i = L^T A_i R$
- The problem become:

$$\max_{\substack{L \in R^{r \times \ell_1} : \quad L^T L = I_{\ell_1} \\ R \in R^{c \times \ell_2} : \quad R^T R = I_{\ell_2}}} \sum_{i=1}^{n} ||L^T A_i R||_F^2.$$

- Iterative algorithm:

1. *For a given* R, L *consists of the* ℓ_1 *eigenvectors of the matrix* $M_L = \sum_{i=1}^{n} A_i R R^T A_i^T$ *corresponding to the largest* ℓ_1 *eigenvalues.*

Can be viewed as iteratively using Yang's 2D PCA
(1) on the rows (right PCA)
(2) on the columns (left PCA)

IMLDA

- Image matrix-based LDA (IMLDA)
- image between-class (within-class) scatter matrix:

$$\mathbf{G}_b = \frac{1}{M}\sum_{i=1}^{c} M_i(\bar{\mathbf{A}}_i - \bar{\mathbf{A}})^{\mathrm{T}}(\bar{\mathbf{A}}_i - \bar{\mathbf{A}}),$$

$$\mathbf{G}_w = \frac{1}{M}\sum_{i=1}^{c}\sum_{j=1}^{M_i}(\mathbf{A}_j^{(i)} - \bar{\mathbf{A}}^{(i)})^{\mathrm{T}}(\mathbf{A}_j^{(i)} - \bar{\mathbf{A}}^{(i)}).$$

J. Yang, J.-y. Yang, A.F. Frangi, D. Zhang, Uncorrelated projection discriminant analysis and its application to face image feature extraction, *IJPRAI*, 2003.

M. Li, B. Yuan, 2D-LDA: A statistical linear discriminant analysis for image matrix, *Pattern Recognition Letters*, 2005.

IMLDA

- Solution:

$$J(\varphi) = \frac{\varphi^{\mathrm{T}} \mathbf{G}_b \varphi}{\varphi^{\mathrm{T}} \mathbf{G}_w \varphi}.$$

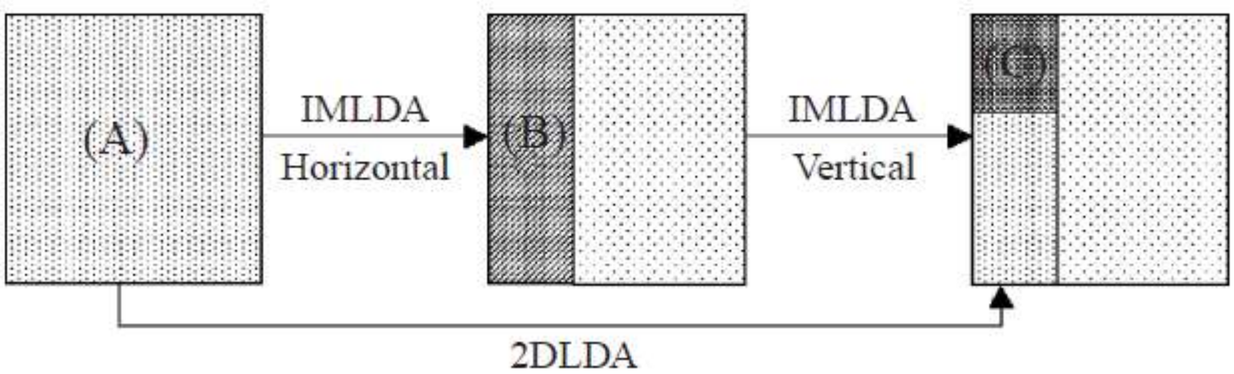
- Feature extraction:

$$\mathbf{B} = \mathbf{A}\mathbf{U}, \quad \text{where } \mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_q).$$

- A simple extension of Yang's 2D PCA !

2D LDA (Yang)

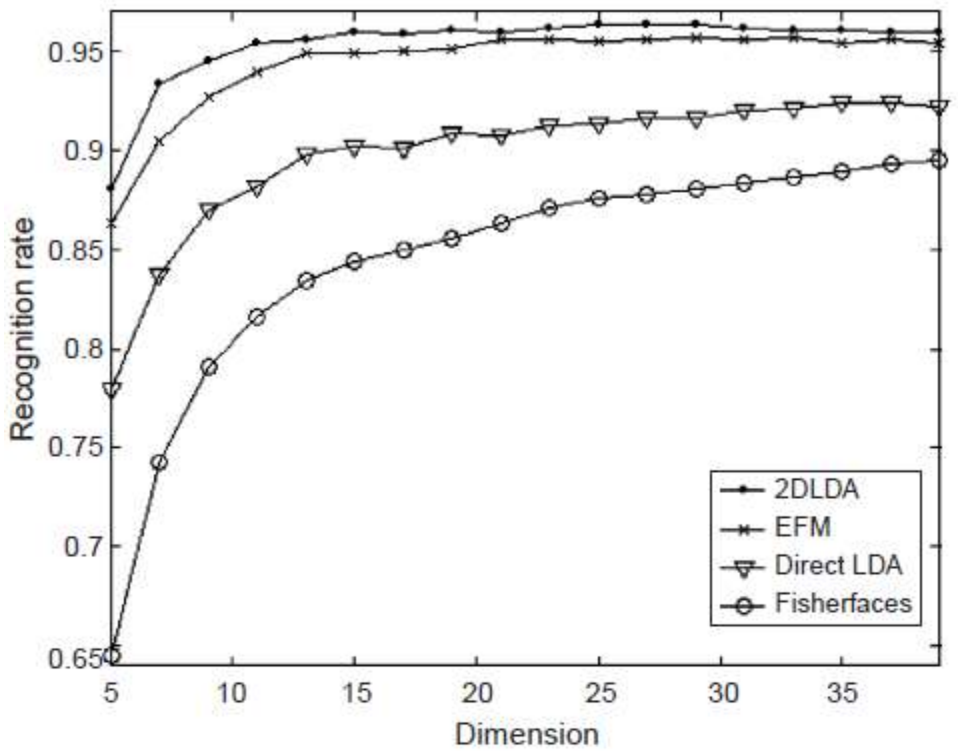
- **Just to perform IMLDA twice:**
- **(1) in horizontal direction**
- **(2) in vertical direction**



J. Yang, D. Zhang, X. Yong, J.-y. Yang, Two-dimensional discriminant transform for face recognition, *Pattern Recognition*, 2005.

2D LDA (Yang)

- Experiments:



2D LDA (Ye)

- Image matrix: $X \in \mathbb{R}^{r \times c}$
- Left transform: $L = [u_1, \cdots, u_{\ell_1}] \in \mathbb{R}^{r \times \ell_1}$
- Right transform: $R = [v_1, \cdots, v_{\ell_2}] \in \mathbb{R}^{c \times \ell_2}$
- Transformation: $L^T X R \in R^{\ell_1 \times \ell_2}$
- Within/between class distance:

$$\tilde{D}_w = \text{trace}\left(\sum_{i=1}^{k} \sum_{X \in \Pi_i} L^T (X - M_i) R R^T (X - M_i)^T L \right),$$

$$\tilde{D}_b = \text{trace}\left(\sum_{i=1}^{k} n_i L^T (M_i - M) R R^T (M_i - M)^T L \right).$$

J. Ye, R. Janardan, Q. Li, Two-Dimensional Linear Discriminant Analysis, *NIPS* 2004.

Tensor Subspace Analysis

H. Lu, K.N. Plataniotis, A.N. Venetsanopoulos, A survey of multilinear subspace learning for tensor data, *Pattern Recognition*, 2011.

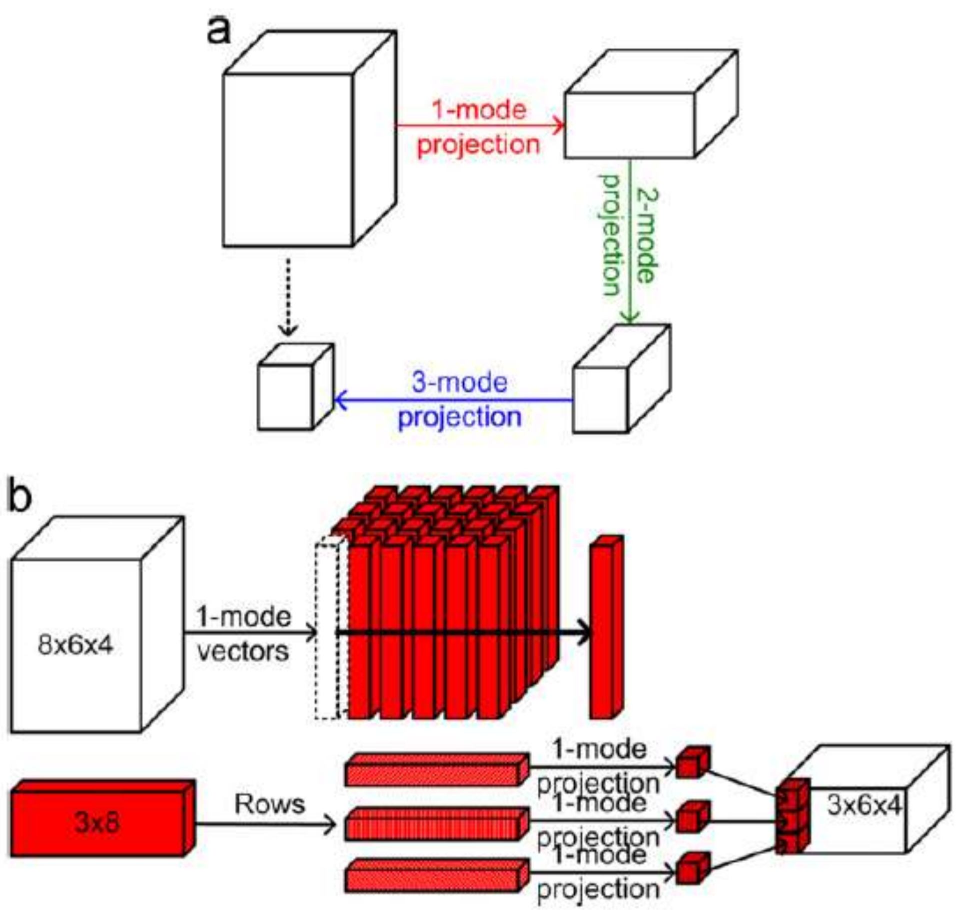
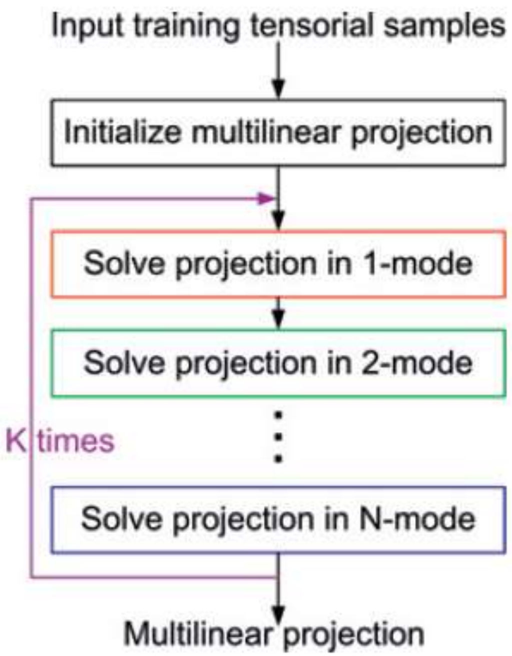


Fig. 5. Illustration of tensor-to-tensor projection: (a) projection of a tensor in all modes and (b) projection of a tensor in one mode.

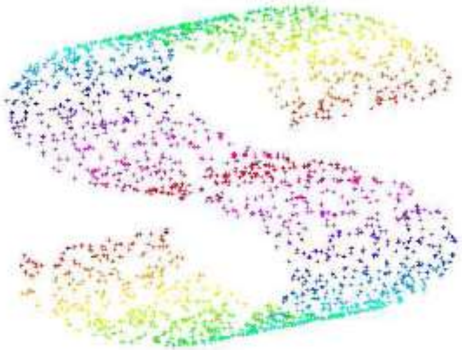
(1) Minimum reconstruction error (PCA)
(2) Maximum separation criterion (LDA)



Nonlinear Extensions

Nonlinear Extensions

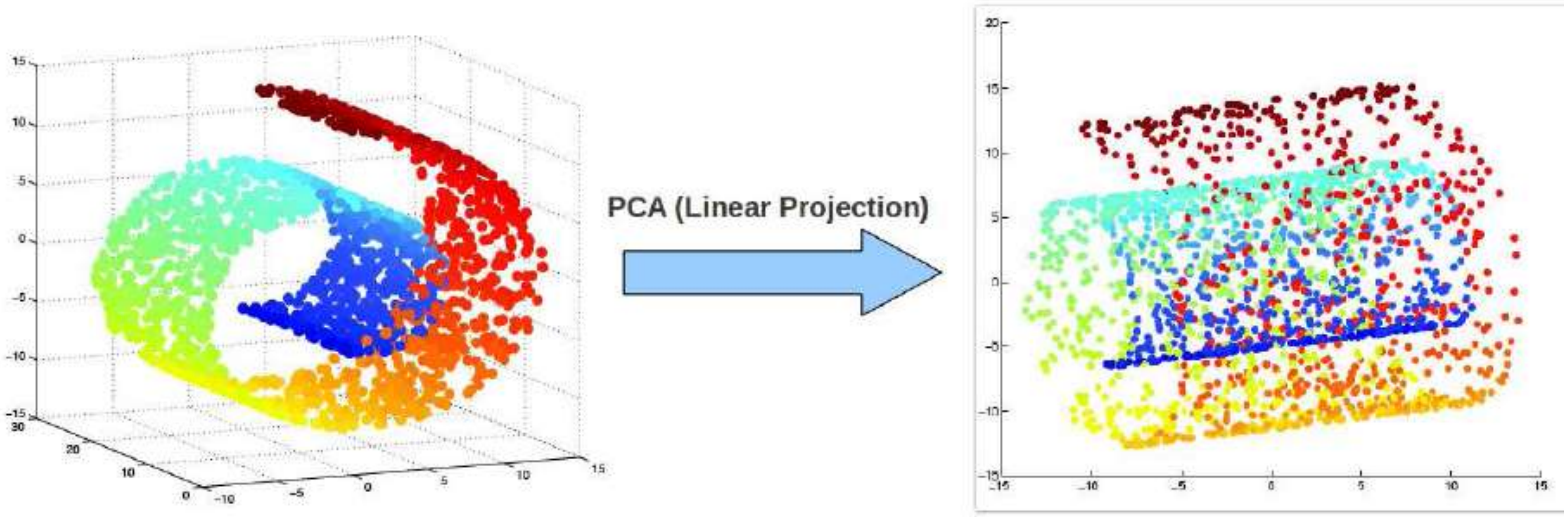
- Given: Low-dim. surface embedded **nonlinearly** in high-dim. space
  - Such a structure is called a **Manifold**



- Goal: Recover the low-dimensional surface

Nonlinear Extensions

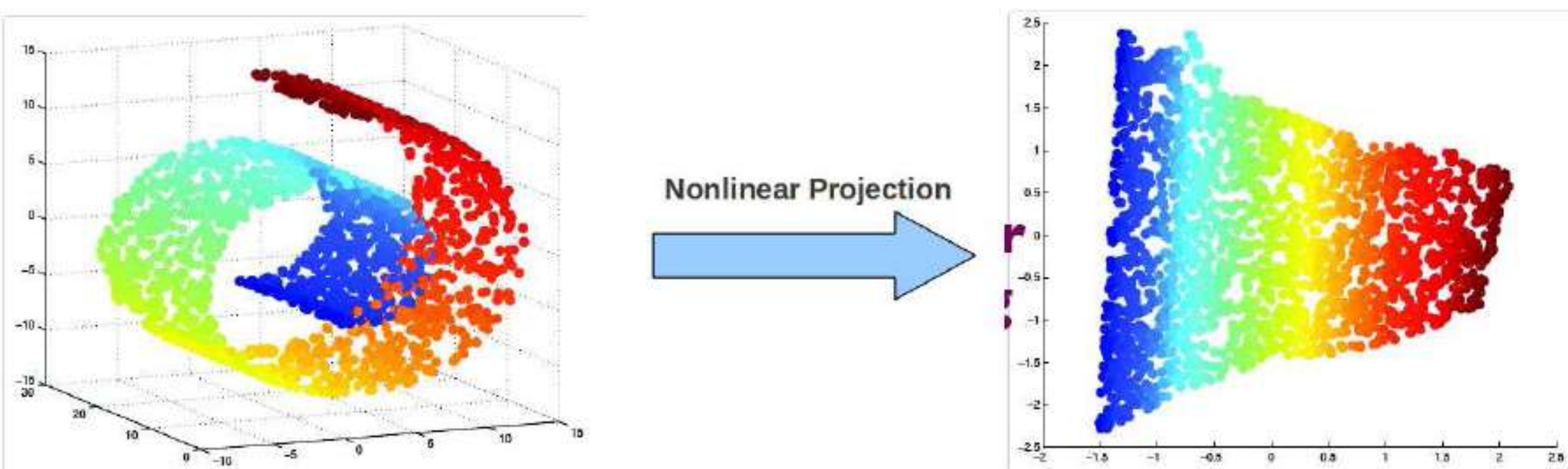
- Consider the swiss-roll dataset (points lying close to a manifold)



- Linear projection methods (e.g., PCA) can't capture intrinsic nonlinearities

Nonlinear Extensions

- We want to do nonlinear projections
- Different criteria could be used for such projections
- Most nonlinear methods try to preserve the neighborhood information
  - Locally linear structures (locally linear $\Rightarrow$ globally nonlinear)
  - Pairwise distances (along the nonlinear manifold)
- Roughly translates to "unrolling" the manifold



Nonlinear Extensions

Two ways of doing it:

- Nonlinearize a linear dimensionality reduction method. E.g.:
  - **Kernel PCA (nonlinear PCA)**
- Using manifold based methods. E.g.:
  - **Locally Linear Embedding (LLE)**
  - **Isomap**
  - Maximum Variance Unfolding
  - Laplacian Eigenmaps
  - And several others (Hessian LLE, Hessian Eigenmaps, etc.)

Kernel PCA

- Given N observations $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\forall \mathbf{x}_n \in \mathbb{R}^D$, define the $D \times D$ covariance matrix (assuming centered data $\sum_n \mathbf{x}_n = \mathbf{0}$)

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^\top$$

- Linear PCA: Compute eigenvectors $\mathbf{u}_i$ satisfying: $\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i \; \forall i = 1, \ldots, D$
- Consider a nonlinear transformation $\phi(\mathbf{x})$ of $\mathbf{x}$ into an M dimensional space
- $M \times M$ covariance matrix **in this space** (assume centered data $\sum_n \phi(\mathbf{x}_n) = \mathbf{0}$)

$$\mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^\top$$

- Kernel PCA: Compute eigenvectors $\mathbf{v}_i$ satisfying: $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i \; \forall i = 1, \ldots, M$
- Ideally, we would like to do this without having to compute the $\phi(\mathbf{x}_n)$'s

Kernel PCA

- Kernel PCA: Compute eigenvectors $\mathbf{v}_i$ satisfying: $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$
- Plugging in the expression for $\mathbf{C}$, we have the eigenvector equation:

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\mathbf{x}_n)\{\phi(\mathbf{x}_n)^\top \mathbf{v}_i\} = \lambda_i \mathbf{v}_i$$

- Using the above, we can write $\mathbf{v}_i$ as: $\mathbf{v}_i = \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n)$
- Plugging this back in the eigenvector equation:

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^\top \sum_{m=1}^{N} a_{im}\phi(\mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n)$$

- Pre-multiplying both sides by $\phi(\mathbf{x}_l)^\top$ and re-arranging

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\mathbf{x}_l)^\top\phi(\mathbf{x}_n)\sum_{m=1}^{N} a_{im}\phi(\mathbf{x}_n)^\top\phi(\mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_l)^\top\phi(\mathbf{x}_n)$$

Kernel PCA

- Using $\phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$, the eigenvector equation becomes:

$$\frac{1}{N} \sum_{n=1}^{N} k(\mathbf{x}_l, \mathbf{x}_n) \sum_{m=1}^{N} a_{im} k(\mathbf{x}_n, \mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in} k(\mathbf{x}_l, \mathbf{x}_n)$$

- Define $\mathbf{K}$ as the $N \times N$ **kernel matrix** with $K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m)$
  - $\mathbf{K}$ is the similarity of two examples $\mathbf{x}_n$ and $\mathbf{x}_m$ in the ϕ space
  - ϕ is implicitly defined by kernel function k (which can be, e.g., RBF kernel)
- Define $\mathbf{a}_i$ as the $N \times 1$ vector with elements a_{in}
- Using $\mathbf{K}$ and $\mathbf{a}_i$, the eigenvector equation becomes:

$$\mathbf{K}^2 \mathbf{a}_i = \lambda_i N \mathbf{K} \mathbf{a}_i \quad \Rightarrow \quad \boxed{\mathbf{K} \mathbf{a}_i = \lambda_i N \mathbf{a}_i}$$

- This corresponds to the original Kernel PCA eigenvalue problem $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$
- For a projection to $K < D$ dimensions, top K eigenvectors of $\mathbf{K}$ are used

Kernel PCA: Centering Data

- In PCA, we centered the data before computing the covariance matrix
- For kernel PCA, we need to do the same

$$\tilde{\phi}(\mathbf{x}_n) = \phi(\mathbf{x}_n) - \frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_l)$$

- How does it affect the kernel matrix $\mathbf{K}$ which is eigen-decomposed?

$$\begin{aligned}
\tilde{K}_{nm} &= \tilde{\phi}(\mathbf{x}_n)^\top \tilde{\phi}(\mathbf{x}_m) \\
&= \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m) - \frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_l) - \frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_l)^\top \phi(\mathbf{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}\phi(\mathbf{x}_j)^\top \phi(\mathbf{x}_l) \\
&= k(\mathbf{x}_n, \mathbf{x}_m) - \frac{1}{N}\sum_{l=1}^{N}k(\mathbf{x}_n, \mathbf{x}_l) - \frac{1}{N}\sum_{l=1}^{N}k(\mathbf{x}_l, \mathbf{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}k(\mathbf{x}_l, \mathbf{x}_l)
\end{aligned}$$

- In matrix notation, the centered $\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N\mathbf{K} - \mathbf{K}\mathbf{1}_N + \mathbf{1}_N\mathbf{K}\mathbf{1}_N$
- $\mathbf{1}_N$ is the $N \times N$ matrix with every element $= 1/N$
- Eigen-decomposition is then done for the centered kernel matrix $\tilde{\mathbf{K}}$

Kernel PCA: The Projection

- Suppose $\{\mathbf{a}_1, \ldots, \mathbf{a}_K\}$ are the top K eigenvectors of kernel matrix $\tilde{\mathbf{K}}$
- The K-dimensional KPCA projection $\mathbf{z} = [z_1, \ldots, z_K]$ of a point $\mathbf{x}$:

$$z_i = \phi(\mathbf{x})^\top \mathbf{v}_i$$

- Recall the definition of $\mathbf{v}_i$

$$\mathbf{v}_i = \sum_{n=1}^{N} a_{in} \phi(\mathbf{x}_n)$$

- Thus

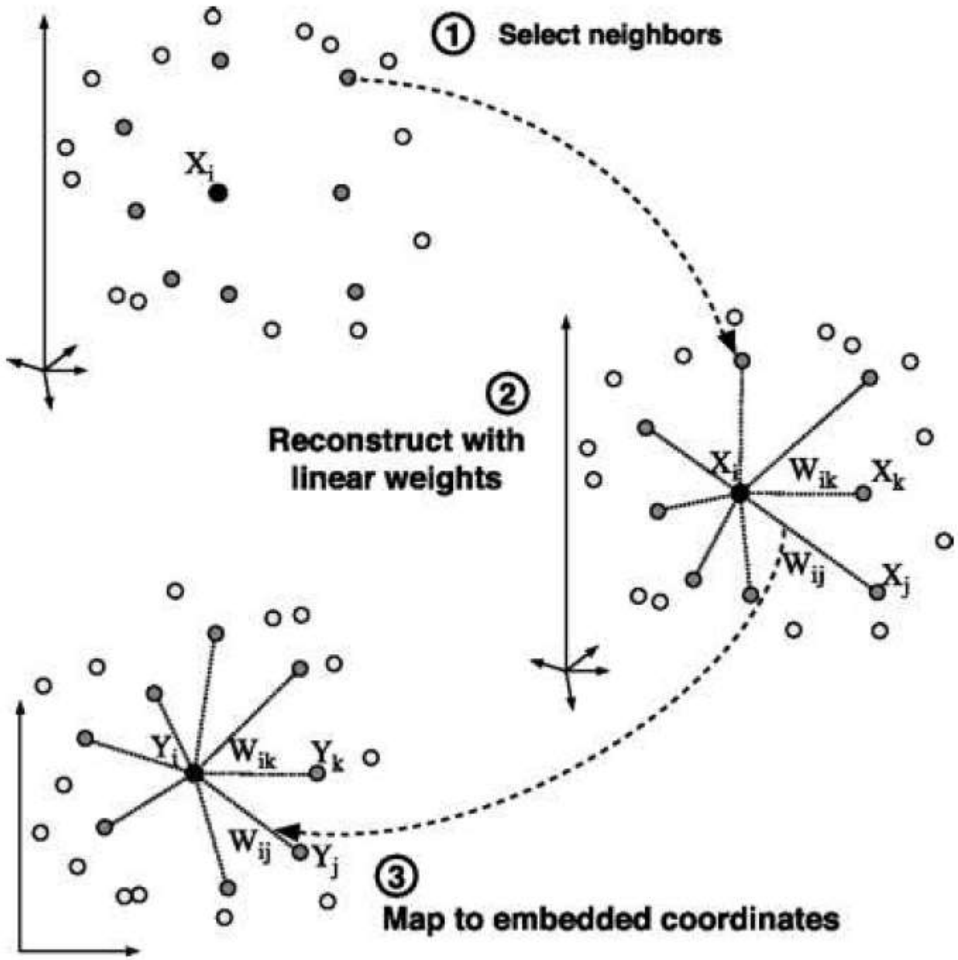
$$z_i = \phi(\mathbf{x})^\top \mathbf{v}_i = \sum_{n=1}^{N} a_{in} k(\mathbf{x}, \mathbf{x}_n)$$

Manifold Learning

- **Locally Linear Embedding (LLE)**
- **Isomap**
- Maximum Variance Unfolding
- Laplacian Eigenmaps
- And several others (Hessian LLE, Hessian Eigenmaps, etc.)

Locally Linear Embedding (LLE)

- Based on
- Assume e… …cally linear patch of t…
- LLE assur… …d
  - Proje… …riginal point



Locally Linear Embedding (LLE)

- Given D dim. data $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, compute K dim. projections $\{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$
- For each example $\mathbf{x}_i$, find its L nearest neighbors
- Assume $\mathbf{x}_i$ to be a weighted linear combination of the L nearest neighbors

$$\mathbf{x}_i \approx \sum_{j \in \mathcal{N}} W_{ij} \mathbf{x}_j \qquad \text{(so the data is assumed locally linear)}$$

- Find the weights by solving the following least-squares problem:

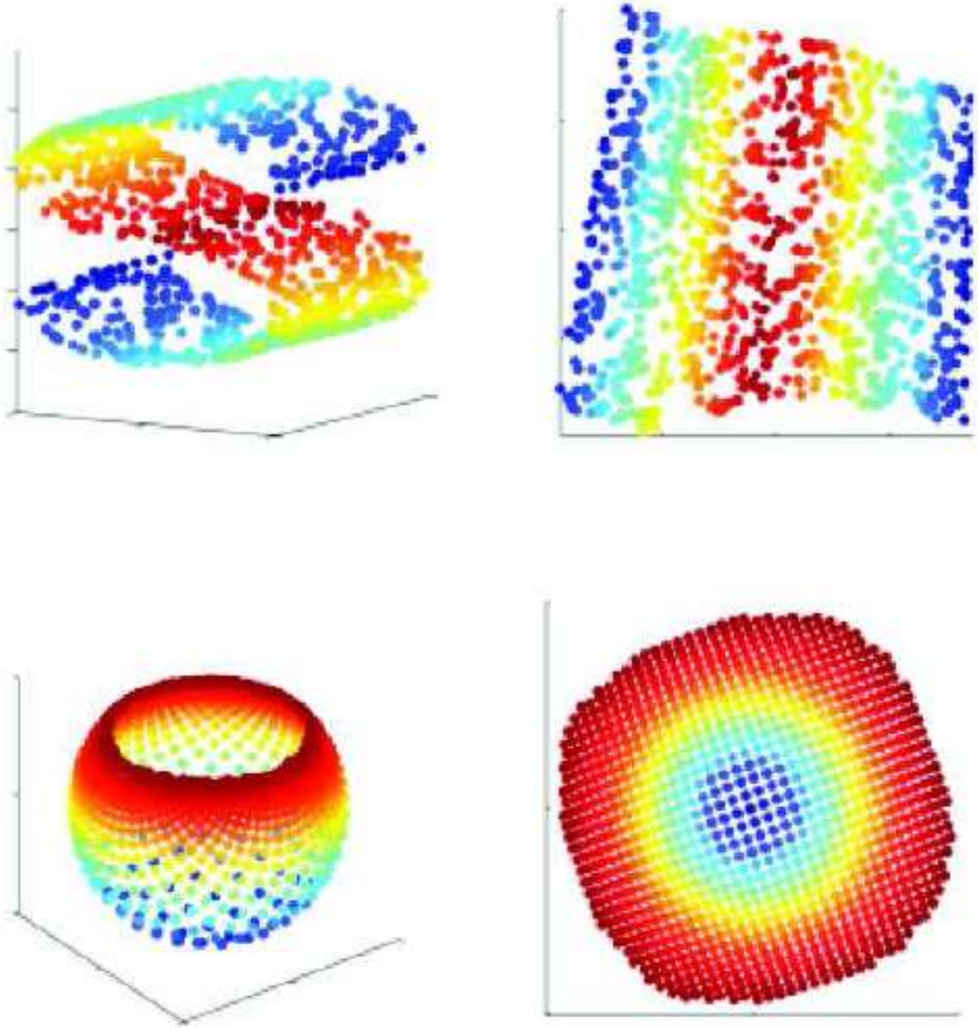
$$\boxed{W = \arg\min_W \sum_{i=1}^{N} ||\mathbf{x}_i - \sum_{j \in \mathcal{N}_i} W_{ij} \mathbf{x}_j||^2} \qquad s.t. \forall i \quad \sum_j W_{ij} = 1$$

- $\mathcal{N}_i$ are the L nearest neighbors of $\mathbf{x}_i$ (note: should choose $L \geq K + 1$)

- Use W to compute low dim. projections $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ by solving:

$$\boxed{\mathbf{Z} = \arg\min_{\mathbf{Z}} \sum_{i=1}^{N} ||\mathbf{z}_i - \sum_{j \in \mathcal{N}} W_{ij} \mathbf{z}_j||^2} \qquad s.t. \forall i \quad \sum_{i=1}^{N} \mathbf{z}_i = 0, \quad \frac{1}{N} \mathbf{Z}\mathbf{Z}^\top = \mathbf{I}$$

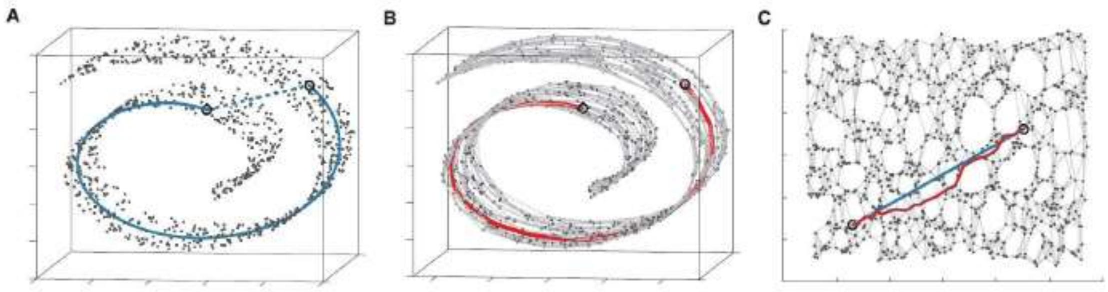
LLE: Examples



Isometric Feature Mapping (ISOMAP)

A graph based algorithm based on constructing a matrix of geodesic distances

- Identify the L nearest neighbors for each data point (just like LLE)
- Connect each point to all its neighbors (an edge for each neighbor)
- Assign weight to each edge based on the Euclidean distance
- Estimate the geodesic distance d_{ij} between any two data points i and j
    - Approximated by the sum of arc lengths along the shortest path between i and j in the graph (can be computed using Djikstras algorithm)
- Construct the $N \times N$ distance matrix $\mathbf{D} = \{d_{ij}^2\}$



Isometric Feature Mapping (ISOMAP)

- Use the distance matrix $\mathbf{D}$ to construct the Gram Matrix

$$\mathbf{G} = -\frac{1}{2}\mathbf{HDH}$$

where $\mathbf{G}$ is $N \times N$ and

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

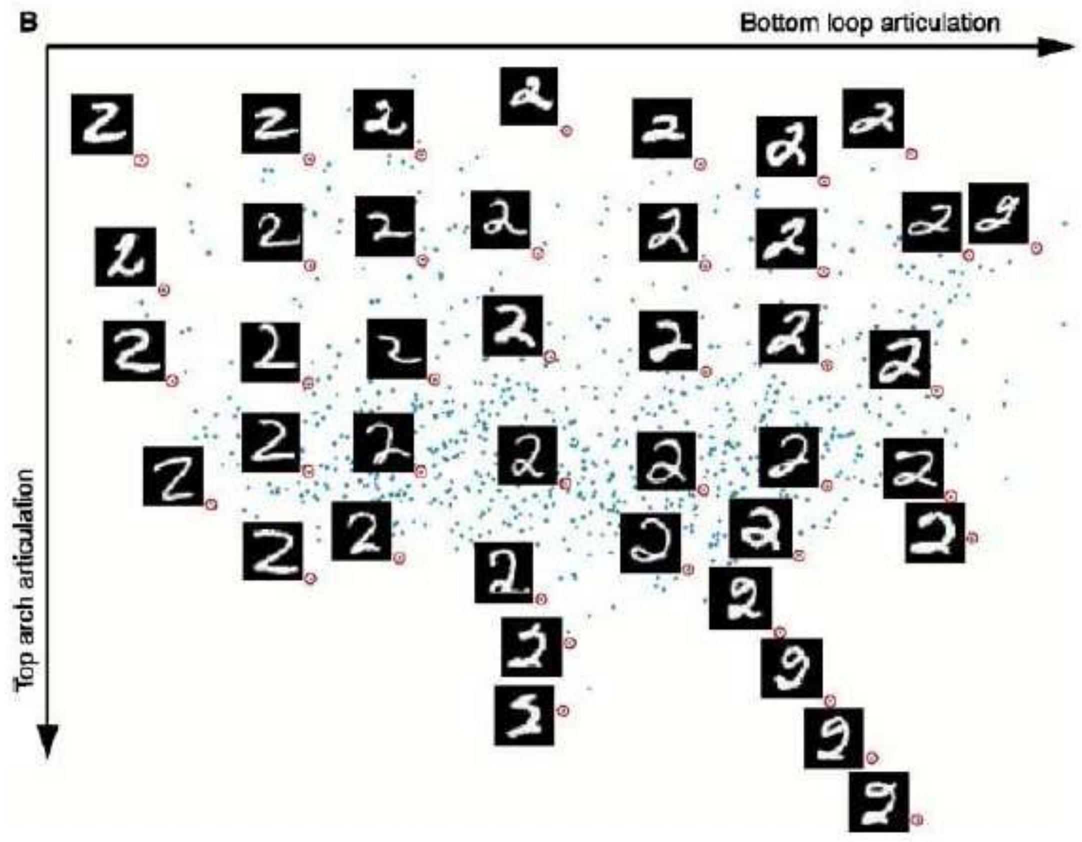
$\mathbf{I}$ is $N \times N$ identity matrix, $\mathbf{1}$ is $N \times 1$ vector of 1s

- Do an eigen decomposition of $\mathbf{G}$
- Let the eigenvectors be $\{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ with eigenvalues $\{\lambda_1, \ldots, \lambda_N\}$
  - Each eigenvector $\mathbf{v}_i$ is N-dimensional: $\mathbf{v}_i = [v_{1i}, v_{2i}, \ldots, v_{Ni}]$
- Take the top K eigenvalue/eigenvectors
- The K dimensional embedding $\mathbf{z}_i = [z_{i1}, z_{i2}, \ldots, z_{iK}]$ of a point $\mathbf{x}_i$:

$$z_{ik} = \sqrt{\lambda_k} v_{ki}$$

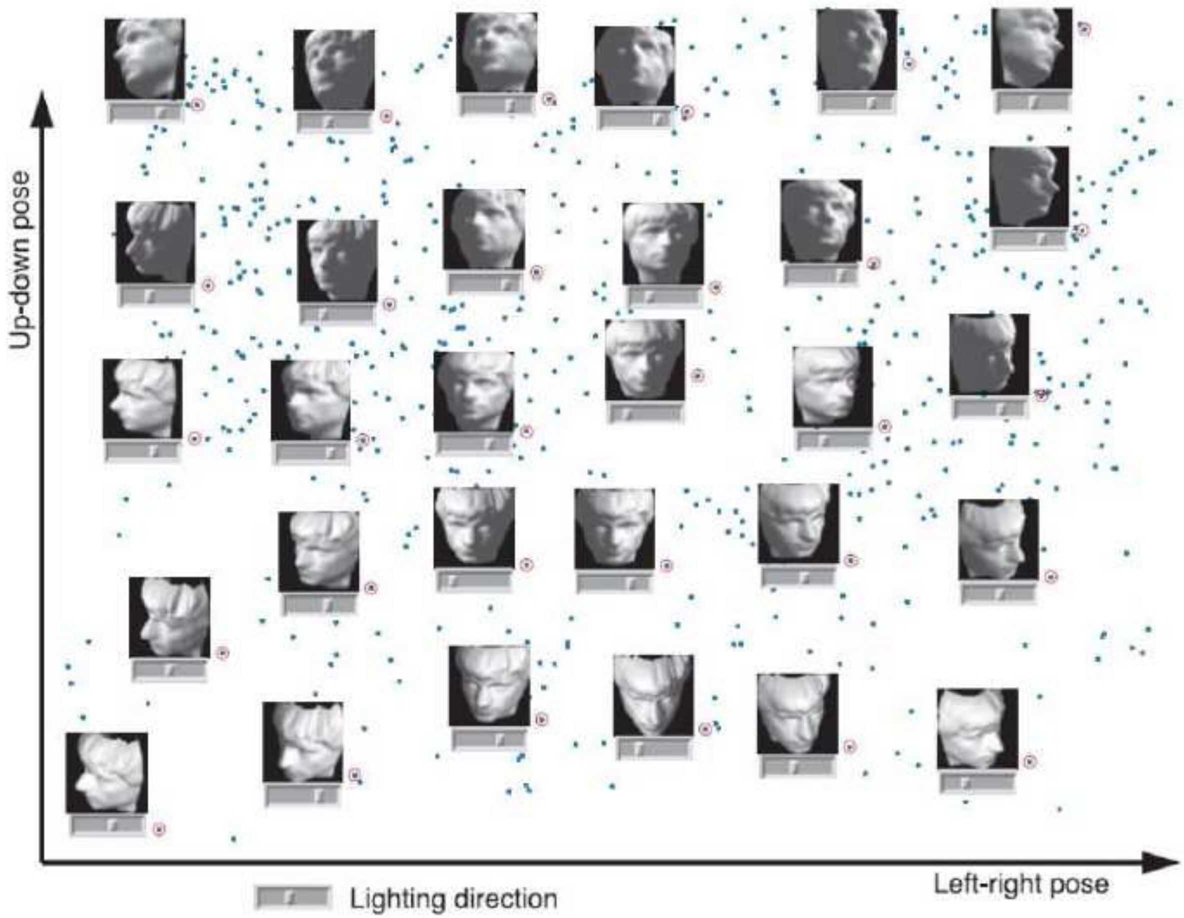
ISOMAP: Example

Digit images projected down to 2 dimensions



ISOMAP: Example

Face images with varying poses



LPP: Locality Preserving Projection

- **Out-of-sample problem:**
  - LLE and ISOMAP are computationally intensive
  - The embedding is only defined on actual data points.
- Solution:
  - LPP is a **linear method that approximates nonlinear methods** (specifically, the Laplacian Eigenmap.)
  - LPP is a linear approximation to nonlinear methods, which takes locality into account

$$\min \sum_{ij} (y_i - y_j)^2 S_{ij}$$

$$S_{ij} = \begin{cases} \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / t), & \|\mathbf{x}_i - \mathbf{x}_j\|^2 < \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

$$S_{ij} = \begin{cases} \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / t), & \text{if } \mathbf{x}_i \text{ is among } k \text{ nearest neighbors of } \mathbf{x}_j \\ & \text{or } \mathbf{x}_j \text{ is among } k \text{ nearest neighbors of } \mathbf{x}_i \\ 0 & \text{otherwise,} \end{cases}$$

LPP: Locality Preserving Projection

$$
\begin{aligned}
&\tfrac{1}{2}\textstyle\sum_{ij}(y_i - y_j)^2 S_{ij} \\
&= \tfrac{1}{2}\textstyle\sum_{ij}(\mathbf{w}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{x}_j)^2 S_{ij} \\
&= \textstyle\sum_{ij}\mathbf{w}^T\mathbf{x}_i S_{ij}\mathbf{x}_i^T\mathbf{w} - \sum_{ij}\mathbf{w}^T\mathbf{x}_i S_{ij}\mathbf{x}_j^T\mathbf{w} \\
&= \textstyle\sum_i \mathbf{w}^T\mathbf{x}_i D_{ii}\mathbf{x}_i^T\mathbf{w} - \mathbf{w}^T XSX^T\mathbf{w} \\
&= \mathbf{w}^T XDX^T\mathbf{w} - \mathbf{w}^T XSX^T\mathbf{w} \\
&= \mathbf{w}^T X(D-S)X^T\mathbf{w} \\
&= \mathbf{w}^T XLX^T\mathbf{w}
\end{aligned}
$$

The matrix D provides a natural measure on data points → a measure importance of the ith image

So a constraint can be imposed

$$
\begin{aligned}
&\mathbf{y}^T D\mathbf{y} = 1 \\
\Rightarrow\ &\mathbf{w}^T XDX^T\mathbf{w} = 1
\end{aligned}
$$

Thus the optimization problem is:

$$
\underset{\substack{\mathbf{w} \\ \mathbf{w}^T XDX^T\mathbf{w}=1}}{\arg\min}\ \mathbf{w}^T XLX^T\mathbf{w}
$$

- The solution is the Generalized Eigenvalue problem.
- The solution is also called Laplacianfaces.

Face Recognition

- Eigenface (PCA) – preserves global structure of image space (unsupervised)
- Fischerface (LDA) – preserves discriminating information (supervised)
- Laplacianface (LPP) – preserves local structure of image space (unsupervised)

TABLE 1
Performance Comparison on the Yale Database

Approach	Dims	Error Rate
Eigenfaces	33	25.3%
Fisherfaces	14	20.0%
Laplacianfaces	**28**	**11.3%**

TABLE 2
Performance Comparison on the PIE Database

Approach	Dims	Error Rate
Eigenfaces	150	20.6%
Fisherfaces	67	5.7%
Laplacianfaces	**110**	**4.6%**

特征选择
Feature Selection

降维 vs 特征选择

- Dimensionality reduction
  - All original features are used
  - The transformed features are linear combinations of the original features
- Feature selection
  - Only a subset of the original features are selected
- Continuous versus discrete

Feature Selection

- Definitions of subset optimality
- Perspectives of feature selection
  - Subset search and feature ranking
  - Feature/subset evaluation measures
  - Models: filter vs. wrapper
  - Results validation and evaluation

An Example for Optimal Subset

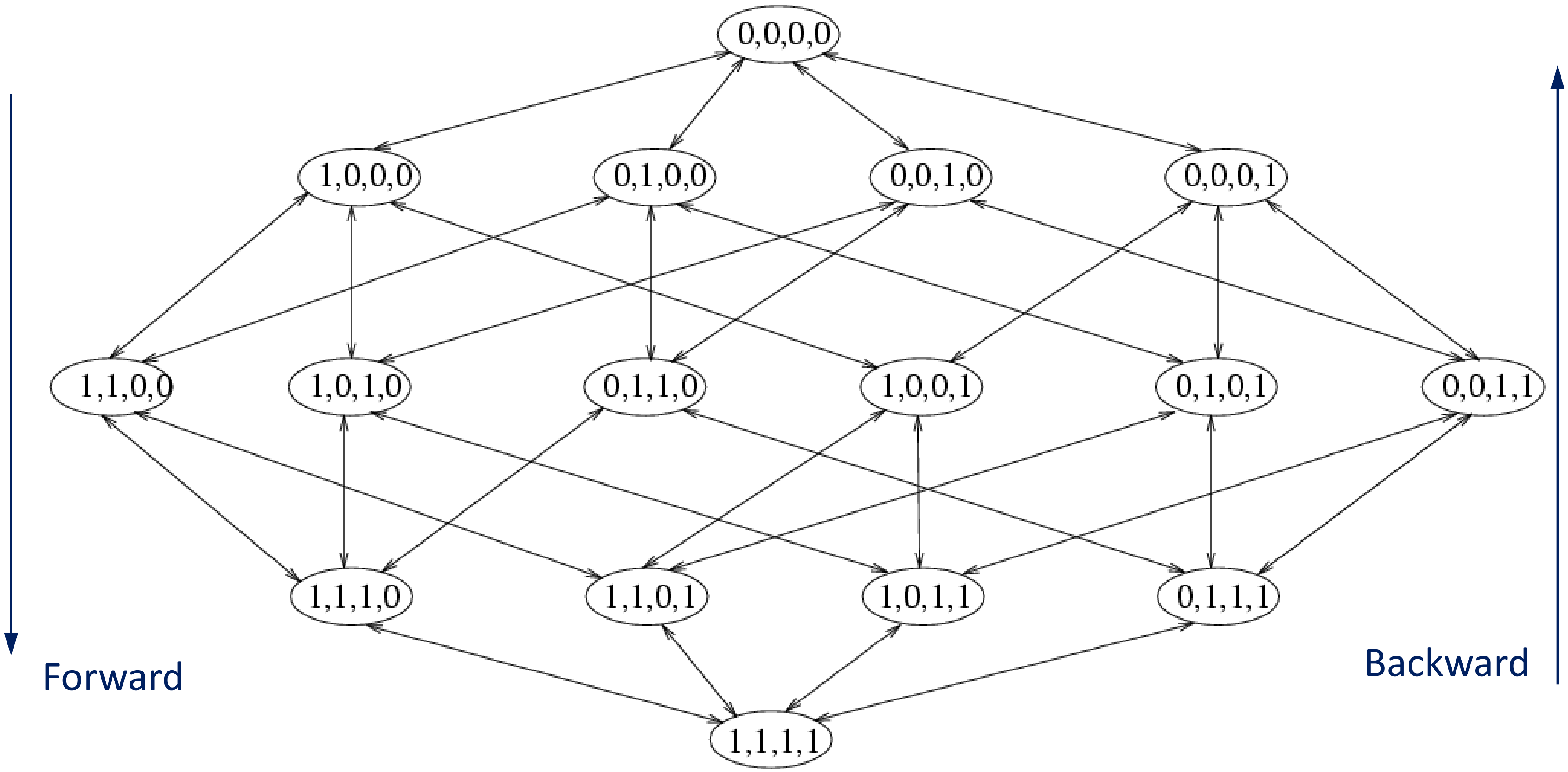
F_1	F_2	F_3	F_4	F_5	C
0	0	1	0	1	0
0	1	0	0	1	1
1	0	1	0	1	1
1	1	0	0	1	1
0	0	1	1	0	0
0	1	0	1	0	1
1	0	1	1	0	1
1	1	0	1	0	1

- Data set (whole set)
  - Five Boolean features
  - $C = F_1 \vee F_2$
  - $F_3 = \neg F_2$, $F_5 = \neg F_4$
  - Optimal subset:

    $\{F_1, F_2\}$ or $\{F_1, F_3\}$
- Combinatorial nature of searching for an optimal subset

Subset Search Problem

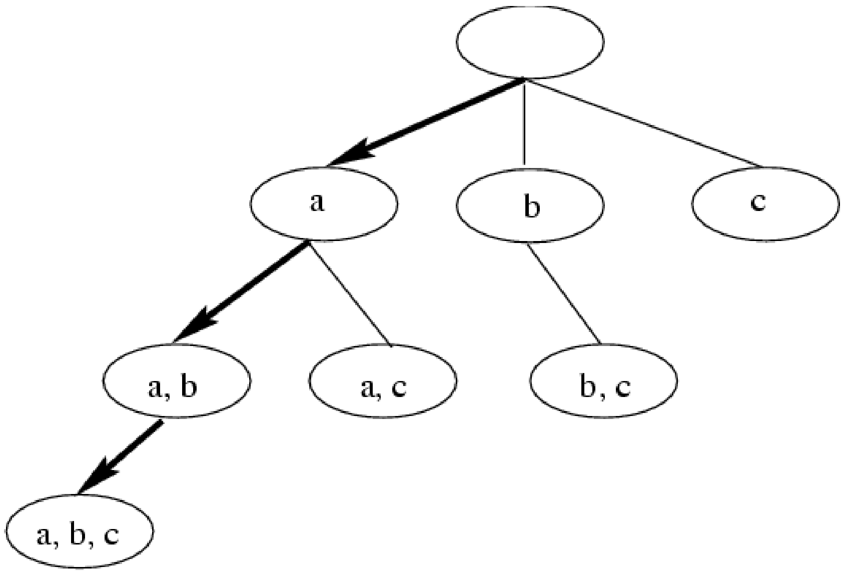
- An example of search space (*Kohavi & John 1997*)



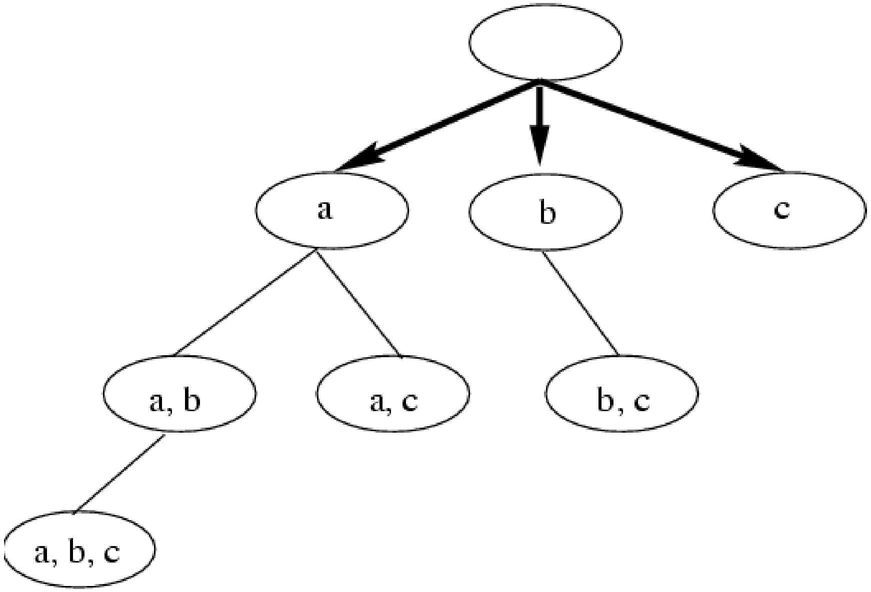
Different Aspects of Search

- Search starting points
  - Empty set
  - Full set
  - Random point
- Search directions
  - Sequential forward selection
  - Sequential backward elimination
  - Bidirectional generation
  - Random generation
- Search Strategies
  - Exhaustive/complete search
  - Heuristic search
  - Nondeterministic search

Illustration of Search Strategies



Depth-first search



Breadth-first search

Feature Ranking

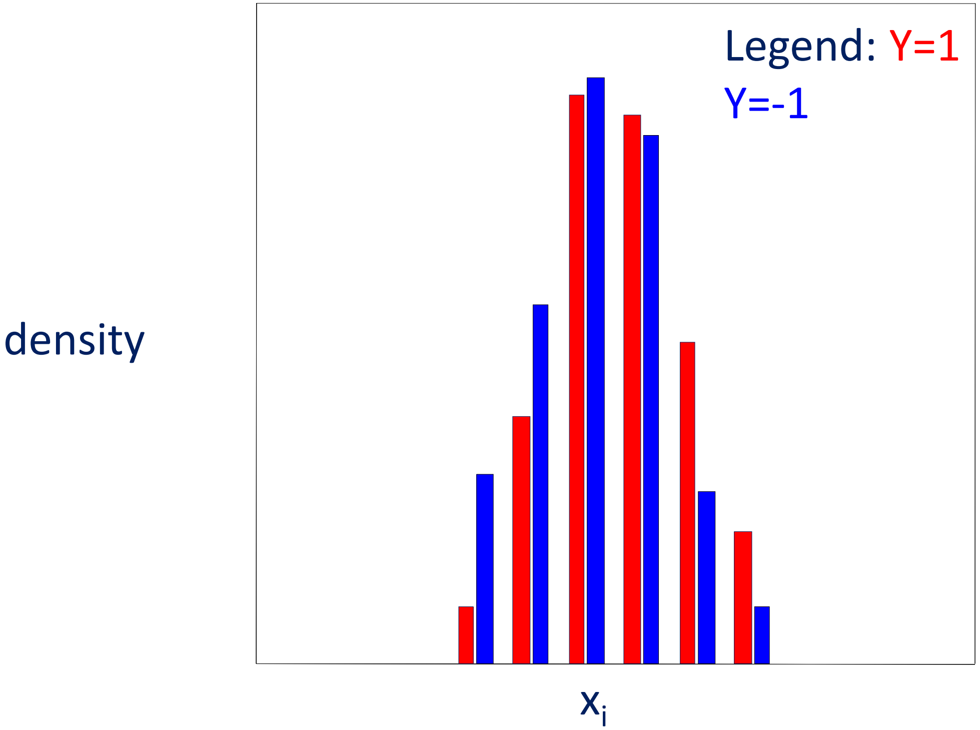
- Weighting and ranking individual features
- Selecting top-ranked ones for feature selection
- Advantages
  - Efficient: $O(N)$ in terms of dimensionality N
  - Easy to implement
- Disadvantages
  - Hard to determine the threshold
  - Unable to consider correlation between features

Individual Feature Irrelevance

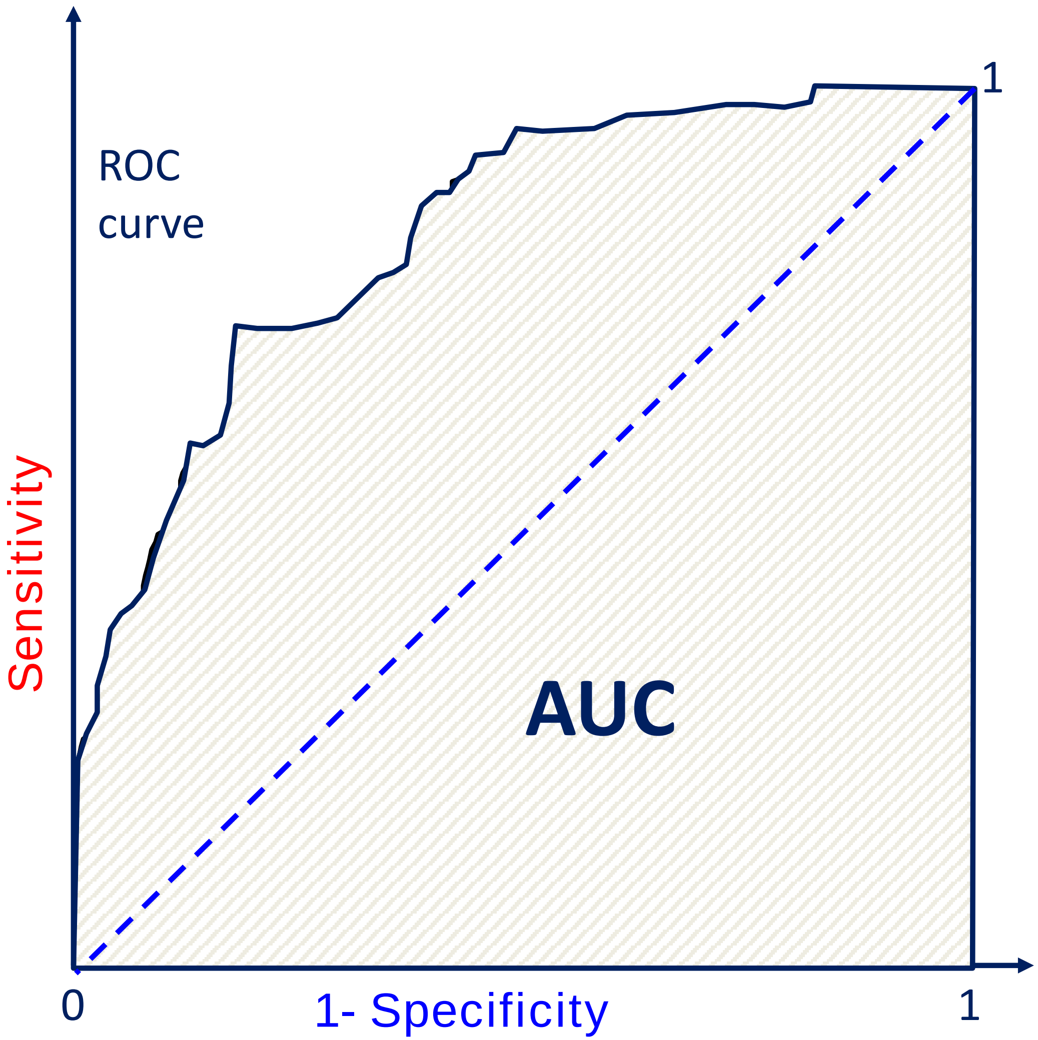
$$P(X_i, Y) = P(X_i)\, P(Y)$$

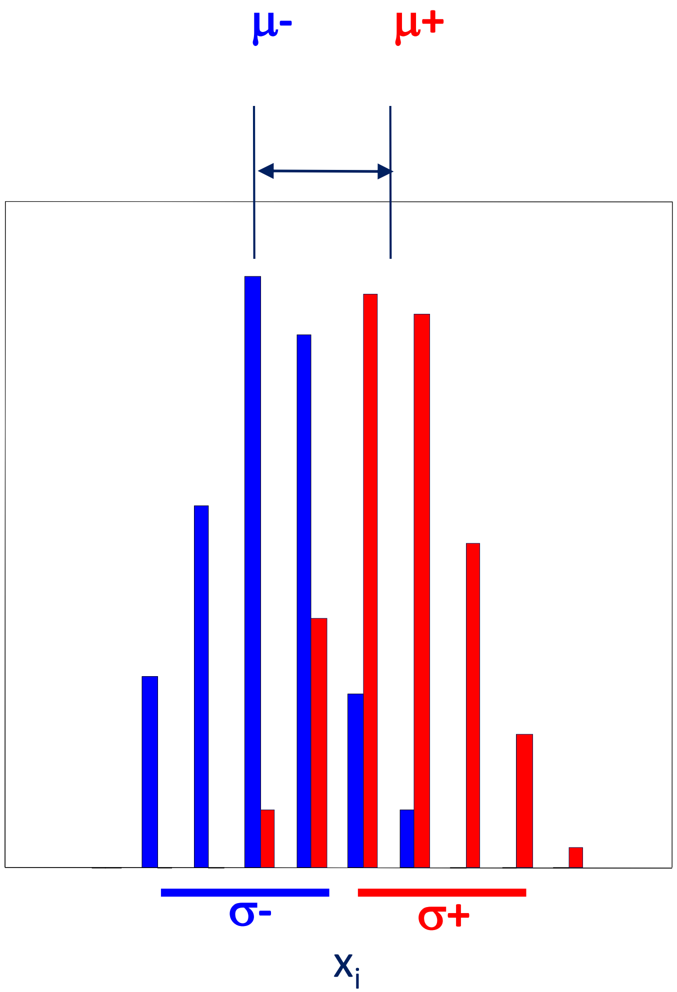
$$P(X_i | Y) = P(X_i)$$

$$P(X_i | Y=1) = P(X_i | Y=-1)$$

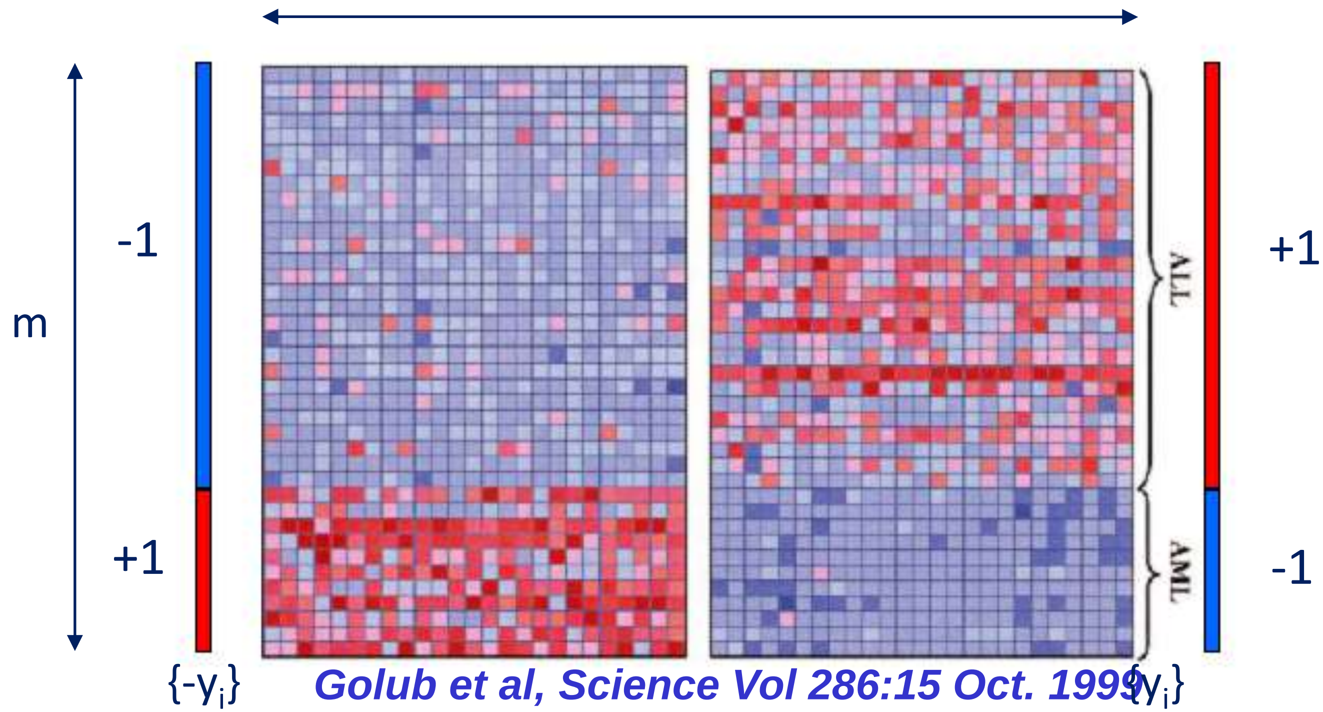


Individual Feature Relevance



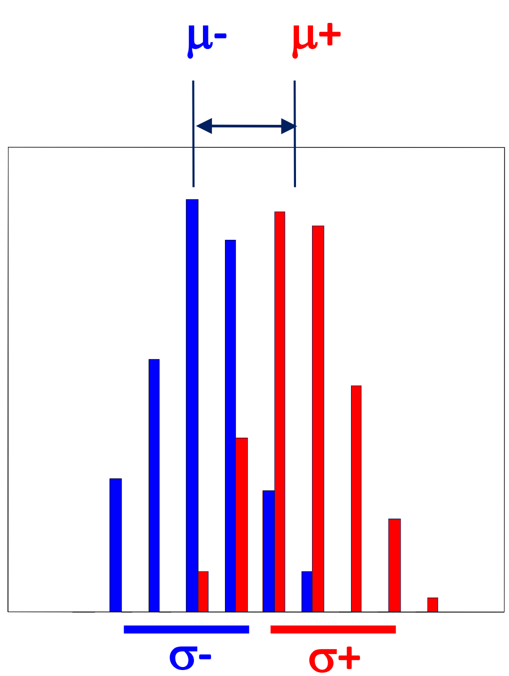


Fisher Ratio

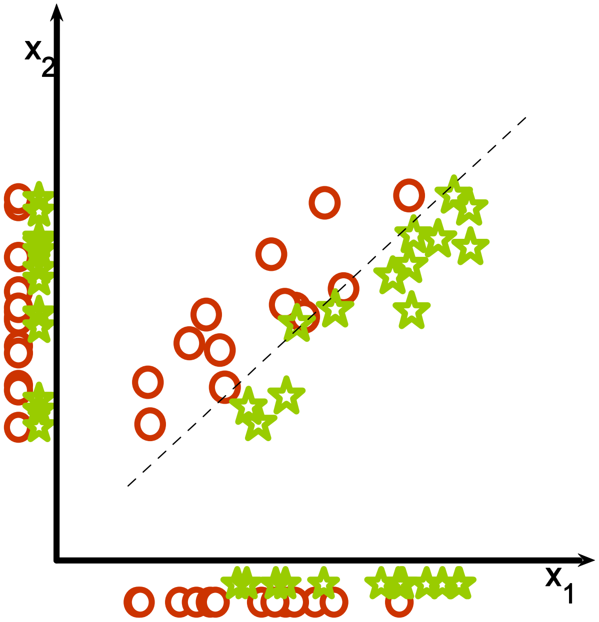


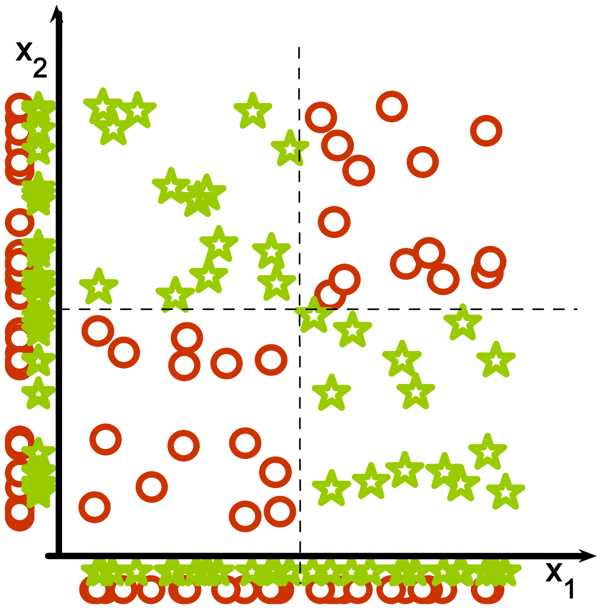
Golub et al, Science Vol 286:15 Oct. 1999

$$S2N = \frac{|\mu+ - \mu-|}{\sigma+ + \sigma-}$$



Univariate Selection May Fail





Guyon-Elisseeff, JMLR 2004; Springer 2006

Evaluation Measures

- The goodness of a feature/feature subset is dependent on measures
- Various measures
  - Information measures (Yu & Liu 2004, Jebara & Jaakkola 2000)
  - Distance measures (Robnik & Kononenko 03, Pudil & Novovicov 98)
  - Dependence measures (Hall 2000, Modrzejewski 1993)
  - Consistency measures (Almuallim & Dietterich 94, Dash & Liu 03)
  - Accuracy measures (Dash & Liu 2000, Kohavi&John 1997)

Illustrative Data set

	Hair	Height	Weight	Lotion	Result
i_1	1	2	1	0	1
i_2	1	3	2	1	0
i_3	2	1	2	1	0
i_4	1	1	2	0	1
i_5	3	2	3	0	1
i_6	2	3	3	0	0
i_7	2	2	3	0	0
i_8	1	1	1	1	0

Sunburn data

	Result (Sunburn)	
	No	Yes
P(Result)	5/8	3/8
P(Hair=1|Result)	2/5	2/3
P(Hair=2|Result)	3/5	0
P(Hair=3|Result)	0	1/3
P(Height=1|Result)	2/5	1/3
P(Height=2|Result)	1/5	2/3
P(Height=3|Result)	2/5	0
P(Weight=1|Result)	1/5	1/3
P(Weight=2|Result)	2/5	1/3
P(Weight=3|Result)	2/5	1/3
P(Lotion=0|Result)	2/5	3/3
P(Lotion=1|Result)	3/5	0

Priors and class conditional probabilities

Information Measures

- Entropy of variable *X*

$$H(X) = -\sum_i P(x_i) \log_2(P(x_i))$$

- Entropy of *X* after observing *Y*

$$H(X|Y) = -\sum_j P(y_j) \sum_i P(x_i|y_j) \log_2(P(x_i|y_j))$$

- Information Gain

$$IG(X|Y) = H(X) - H(X|Y)$$

Distance Measures

– Distance Measures.

- Measures of separability, discrimination or divergence measures. The most typical is derived from distance between the class conditional density functions.

	Mathematical form
Euclidean distance	$D_e = \left\{\sum_{i=1}^{m} (x_i - y_i)^2\right\}^{\frac{1}{2}}$
City-block distance	$D_{cb} = \sum_{i=1}^{m} \lvert x_i - y_i \rvert$
Cebyshev distance	$D_{ch} = \max_i \lvert x_i - y_i \rvert$
Minkowski distance of order m	$D_M = \left\{\sum_{i=1}^{m} (x_i - y_i)^m\right\}^{\frac{1}{m}}$
Quadratic distance Q, positive definite	$D_q = \sum_{i=1}^{m} \sum_{j=1}^{m} (x_i - y_i) Q_{ij} (x_j - y_j)$
Canberra distance	$D_{ca} = \sum_{i=1}^{m} \frac{\lvert x_i - y_i \rvert}{x_i + y_i}$
Angular separation	$D_{as} = \frac{\sum_{i=1}^{m} x_i \cdot y_i}{\left[\sum_{i=1}^{m} x_i^2 \sum_{i=1}^{m} y_i^2\right]^{\frac{1}{2}}}$

Consistency Measures

- Consistency measures
  - Trying to find a minimum number of features that separate classes as consistently as the full set can
  - They aim to achieve **P(C|FullSet) = P(C|SubSet)**.
  - An inconsistency is defined as two instances having the same feature values but different classes
    - E.g., one inconsistency is found between instances i4 and i8 if we just look at the first two columns of the data table

	Hair	Height	Weight	Lotion	Result
i_1	1	2	1	0	1
i_2	1	3	2	1	0
i_3	2	1	2	1	0
i_4	1	1	2	0	1
i_5	3	2	3	0	1
i_6	2	3	3	0	0
i_7	2	2	3	0	0
i_8	1	1	1	1	0

Dependence Measures

- Dependence Measures.
  - known as measures of association or correlation.
  - Its main goal is to quantify how strongly two variables are correlated or present some association with each other, in such way that knowing the value of one of them, we can derive the value for the other.
  - *Pearson correlation* coefficient:

$$\rho(X,Y) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\left[\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2\right]^{\frac{1}{2}}}$$

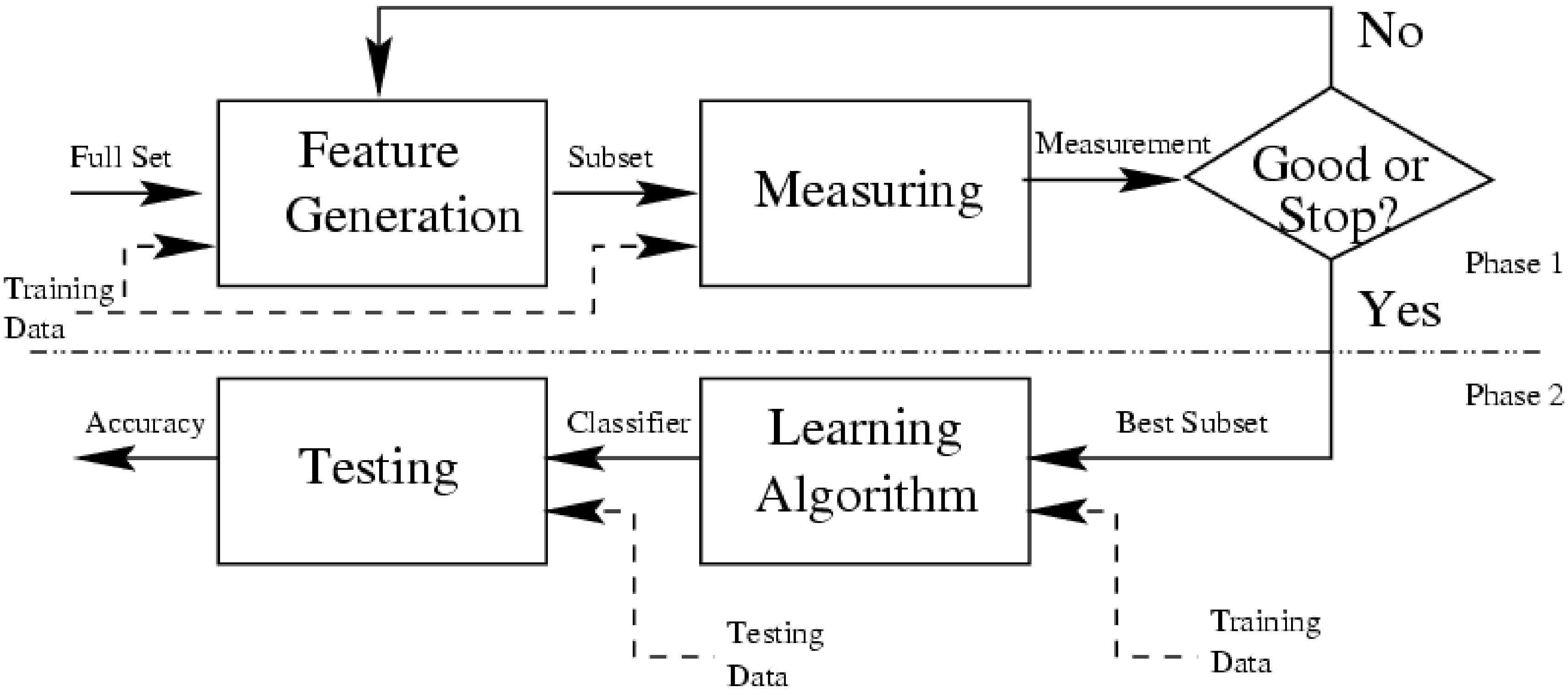
Accuracy Measures

- Using classification accuracy of a classifier as an evaluation measure
- Factors constraining the choice of measures
  - Classifier being used
  - The speed of building the classifier
- Compared with previous measures
  - Directly aimed to improve accuracy
  - Biased toward the classifier being used
  - More time consuming

Models of Feature Selection

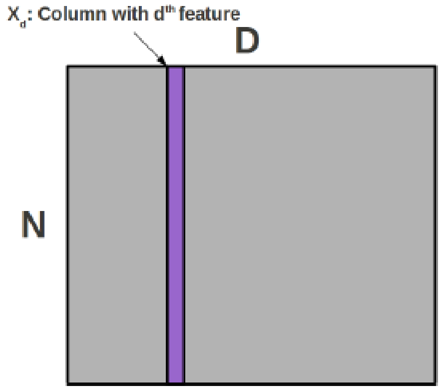
- Filter model
  - Separating feature selection from classifier learning
  - Relying on general characteristics of data (*information, distance*, *dependence*, *consistency*)
  - No bias toward any learning algorithm, fast
- Wrapper model
  - Relying on a predetermined classification algorithm
  - Using predictive accuracy as goodness measure
  - High accuracy, computationally expensive
- Embedded model
  - Feature selected during learning process

Filter Model



Filter Feature Selection

- Uses heuristics but is much faster than wrapper methods



- **Correlation Critera:** Rank features in order of their correlation with the labels

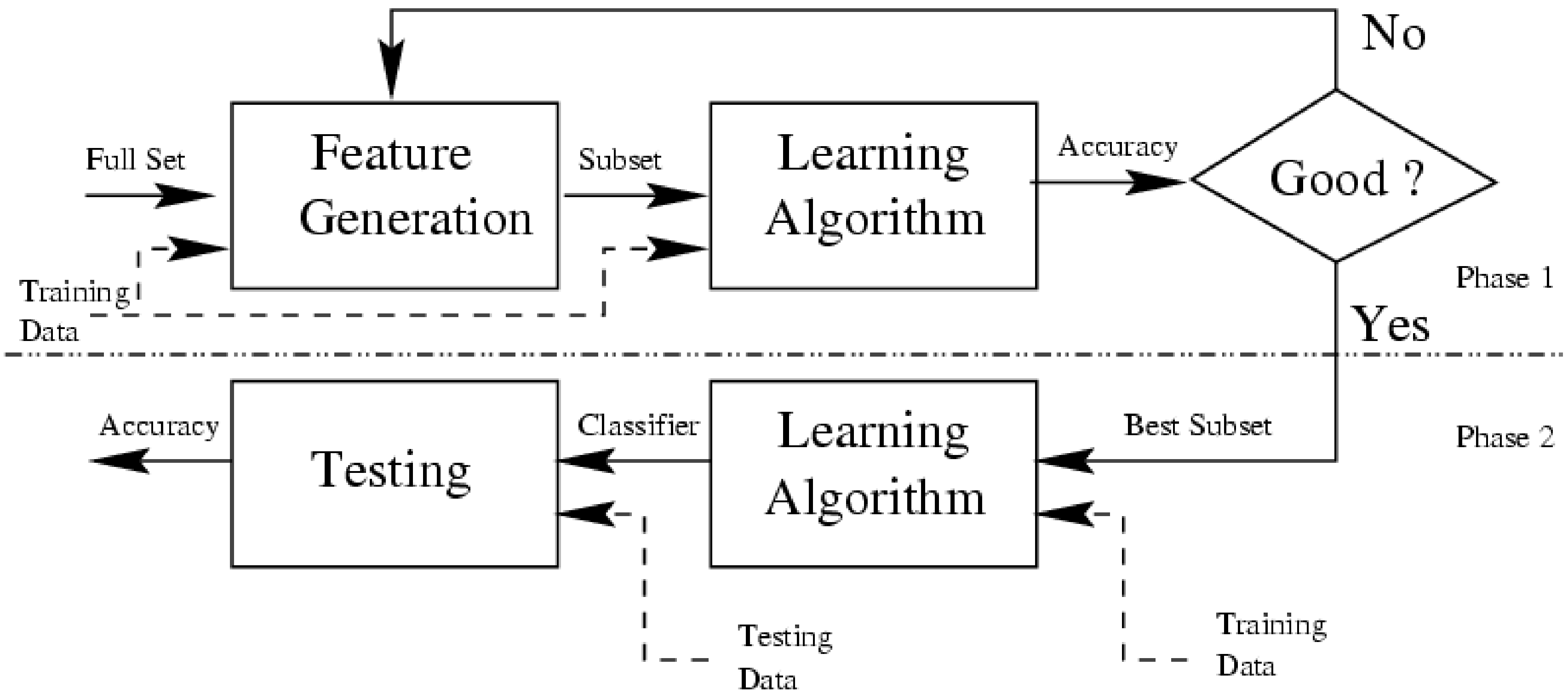
$$R(X_d, Y) = \frac{cov(X_d, Y)}{\sqrt{var(X_d)var(Y)}}$$

- **Mutual Information Criteria:**

$$MI(X_d, Y) = \sum_{X_d \in \{0,1\}} \sum_{Y \in \{-1,+1\}} P(X_d, Y) \frac{\log P(X_d, Y)}{P(X_d)P(Y)}$$

  - High mutual information mean high relevance of that feature

Wrapper Model



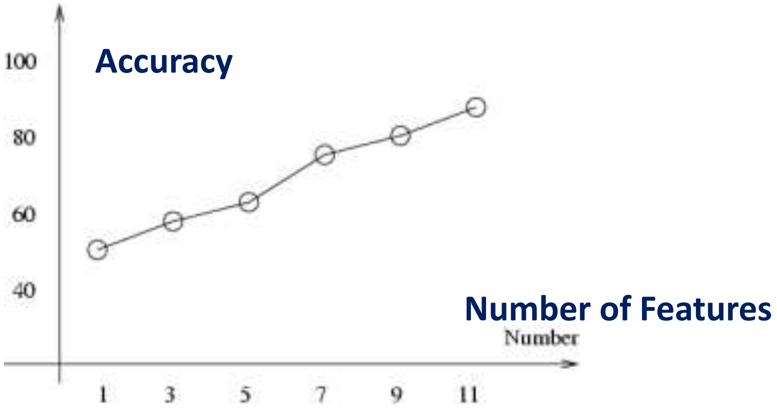
Wrapper Feature Selection

- **Forward Search**
  - Let $\mathcal{F} = \{\}$
  - While not selected desired number of features
  - For each unused feature f:
    - Estimate model's error on feature set $\mathcal{F} \bigcup f$ (using cross-validation)
  - Add f with lowest error to $\mathcal{F}$
- **Backward Search**
  - Let $\mathcal{F} = \{\text{all features}\}$
  - While not reduced to desired number of features
  - For each feature $f \in \mathcal{F}$:
    - Estimate model's error on feature set $\mathcal{F} \backslash f$ (using cross-validation)
  - Remove f with lowest error from $\mathcal{F}$

How to Validate Selection Results

- Direct evaluation (if we know *a priori* …)
  - Often suitable for artificial data sets
  - Based on prior knowledge about data
- Indirect evaluation (if we don't know …)
  - Often suitable for real-world data sets
  - Based on
    - number of features selected
    - performance on selected features (e.g., predictive accuracy, goodness of resulting clusters)
    - interpretability, speed

Methods for Result Evaluation



- Learning curves
  - For results in the form of a ranked list of features
- Before-and-after comparison
  - For results in the form of a minimum subset
- Comparison using different classifiers
  - To avoid learning bias of a particular classifier
- Repeating experimental results
  - For non-deterministic results

Representative Algorithms

- Filter algorithms
  - Feature ranking algorithms
    - Example: Relief (*Kira & Rendell 1992*)
  - Subset search algorithms
    - Example: consistency-based algorithms
      - Focus (*Almuallim & Dietterich, 1994*)

- Wrapper algorithms
  - Feature ranking algorithms
    - Example: SVM
  - Subset search algorithms
    - Example: RFE

Relief Algorithm

Relief

Input: $\mathbf{x}$ - features
m - number of instances sampled
τ - adjustable relevance threshold

initialize: $\mathbf{w} = 0$
for $i = 1$ to m
begin
randomly select an instance I
find nearest-hit H and nearest-miss J
for $j = 1$ to N
$\mathbf{w}(j) = \mathbf{w}(j) - \texttt{diff}(j, I, H)^2/m + \texttt{diff}(j, I, J)^2/m$
end

Output: $\mathbf{w}$ greater than τ

Focus Algorithm

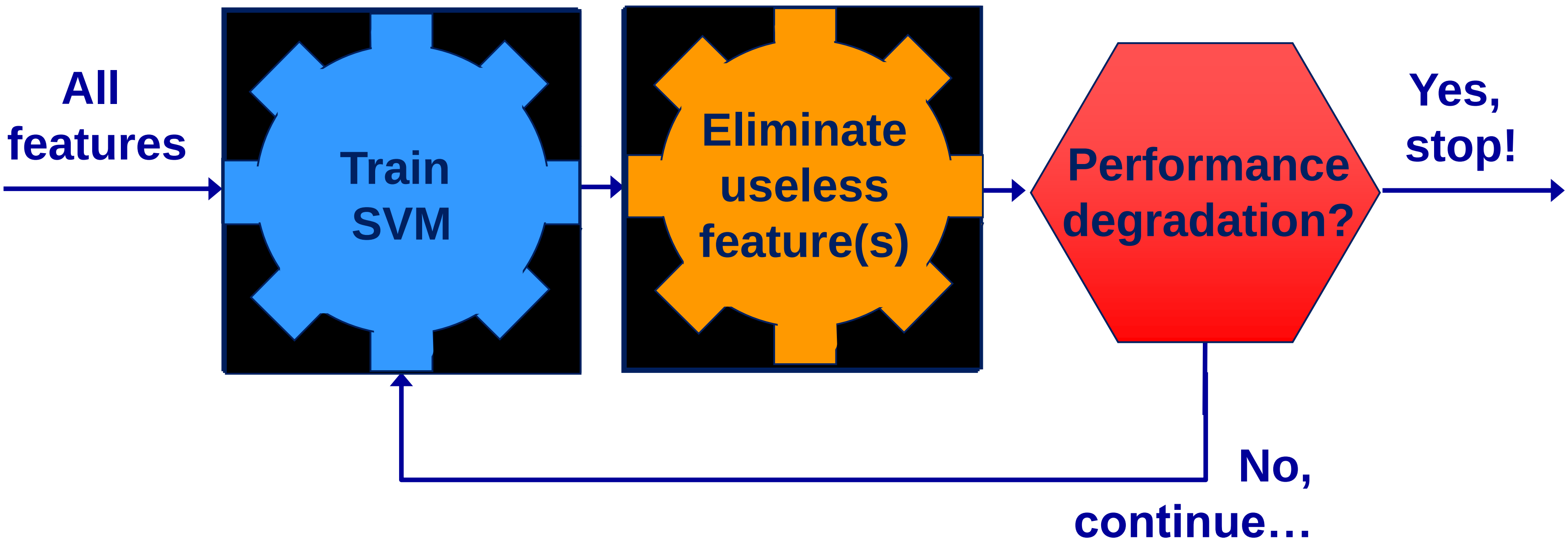
Focus

Input: F - all features x in data D
U - inconsistency rate as evaluation measure

initialize: $S = \{\}$

for $i = 1$ to N
  for each subset S of size i
    if $\text{Cal}U(S, D) = 0$ /* *Cal*$U(S, D)$ *returns inconsistency* */
      return S
Output: S - a minimum subset that satisfies U

Embedded Methods (RFE)



Recursive Feature Elimination (RFE) SVM. *Guyon-Weston, 2000. US patent 7,117,188*

Feature Selection via Regularization

- Data: $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, $i = 1, \ldots, n$
- Minimize with respect to function $f : \mathcal{X} \to \mathcal{Y}$:

$$\sum_{i=1}^{n} \ell(y_i, f(x_i)) \quad + \quad \frac{\lambda}{2}\|f\|^2$$

Error on data + Regularization

Loss & function space ? Norm ?

- Two theoretical/algorithmic issues:

1. Loss
2. **Function space / norm**

Ridge Regression and LASSO

- Compared methods to reach the least-square solution
  - Ridge regression: $\min_{w \in \mathbb{R}^p} \frac{1}{2}\|y - Xw\|_2^2 + \frac{\lambda}{2}\|w\|_2^2$
  - Lasso: $\min_{w \in \mathbb{R}^p} \frac{1}{2}\|y - Xw\|_2^2 + \lambda\|w\|_1$
  - Forward greedy:
    - Initialization with empty set
    - Sequentially add the variable that best reduces the square loss

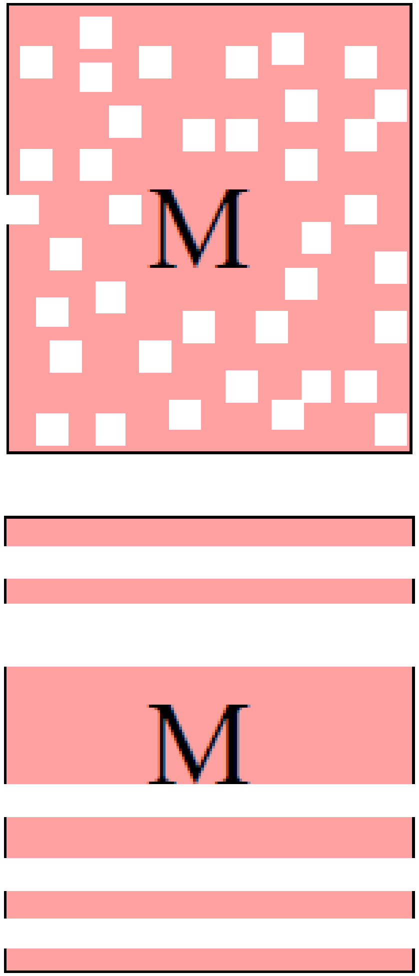
- Each method builds a path of solutions from 0 to ordinary least-squares solution

Group Sparsity (Multi-Class)

$$\min_{\mathbf{W},\mathbf{b}} \sum_{i=1}^{n} \left\| \mathbf{W}^T \mathbf{x}_i + \mathbf{b} - \mathbf{y}_i \right\|_2^2 + \lambda \|\mathbf{W}\|_F^2$$

$$\|\mathbf{W}\|_{2,1} = \|\bar{\mathbf{w}}\|_1 = \sum_{i=1}^{m} \sqrt{\sum_{j=1}^{c} W_{ij}^2}.$$

$$\min_{\mathbf{W},\mathbf{t},\mathbf{M}} \ ||\mathbf{X}\mathbf{W} + \mathbf{e}_n \mathbf{t}^T - \mathbf{Y} - \mathbf{B} \odot \mathbf{M}||_{2,1} + \lambda ||\mathbf{W}||_{2,1}$$
$$\text{s.t.} \quad \mathbf{M} \geq \mathbf{0}$$



S. Xiang et al, Discriminative Least Squares Regression for Multiclass Classification and Feature Selection, IEEE Trans. NNLS, 2012.

Future Work

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 39, NO. 12, DECEMBER 2017

Forward Selection Component Analysis: Algorithms and Applications

Luca Puggini, *Student Member, IEEE*, and Seán McLoone

Abstract—Principal Component Analysis (PCA) is a powerful and widely used tool for dimensionality reduction. However, the principal components generated are linear combinations of all the original variables and this often makes interpreting results and root-cause analysis difficult. Forward Selection Component Analysis (FSCA) is a recent technique that overcomes this difficulty by performing variable selection and dimensionality reduction at the same time. This paper provides, for the first time, a detailed presentation of the FSCA algorithm, and introduces a number of new variants of FSCA that incorporate a refinement step to improve performance. We then show different applications of FSCA and compare the performance of the different variants with PCA and Sparse PCA. The results demonstrate the efficacy of FSCA as a low information loss dimensionality reduction and variable selection technique and the improved performance achievable through the inclusion of a refinement step.

References

Part of slides come from web, and they reserve the corresponding copyrights:

– Fei-Fei Li, CS231n: Convolutional Neural Networks for Visual Recognition
– Piyush Rai, CS5350/6350: Machine Learning
– Lei Yu, Dimensionality Reduction for Data Mining, SDM07
– Galen Andrew, Deep Canonical Correlation Analysis, ICML2013

Thank You!
Q&A