EE 6885 Statistical Pattern Recognition

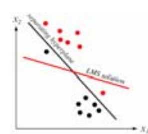
Fall 2005
Prof. Shih-Fu Chang
http://www.ee.columbia.edu/~sfchang

Review: Final Exam (12/12/2005)

- Final Exam
  - Dec. 16th Friday 1:10-3 pm, Mudd Rm 644

- Chap 5: Linear Discriminant Functions

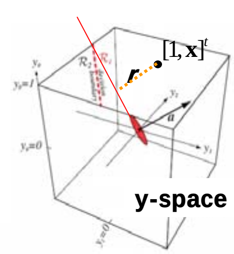
Linear Discriminant Classifiers



$$g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0 \quad \Rightarrow \text{find weight } \mathbf{w} \text{ and bias } w_o$$

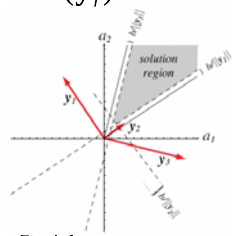
- Augmented Vector $\quad \mathbf{y} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$

$$\Rightarrow g(\mathbf{x}) = g(\mathbf{y}) = \mathbf{a}^t \mathbf{y}$$



map $\mathbf{y}$ to class ω_1 if $g(\mathbf{y})$>0, otherwise class ω_2

- Normalization $\quad \forall \mathbf{y}_i$ in class ω_2, $\mathbf{y}_i \leftarrow -(\mathbf{y}_i)$
- Design Objective $\quad \mathbf{a}^t \mathbf{y}_i > \mathrm{b}, \ \forall \mathbf{y}_i$



a-space

Minimal Squared-Error Solution

$$Y = \begin{bmatrix} \mathbf{y}_1^t \\ \mathbf{y}_2^t \\ \vdots \\ \mathbf{y}_n^t \end{bmatrix}$$

Training sample matrix

dimension: n x (d+1)

Objective: $\mathbf{a}^t\mathbf{y}_i = b, \ \forall \mathbf{y}_i$

$$\Rightarrow \text{define } J_s = \sum_{i=1}^{n} (\mathbf{a}^t\mathbf{y}_i - b_i)^2$$

$$= \|Y\mathbf{a} - \mathbf{b}\|^2 = (Y\mathbf{a} - \mathbf{b})^t (Y\mathbf{a} - \mathbf{b})$$

$$\nabla_{\mathbf{a}} J_s = 2Y^t (Y\mathbf{a} - \mathbf{b}) = 0$$

$$\Rightarrow \mathbf{a} = \left(Y^t Y\right)^{-1} Y^t \mathbf{b} = Y^\dagger \mathbf{b}$$

$$Y^\dagger = \left(Y^t Y\right)^{-1} Y^t$$ **pseudo-inverse : (d+1) x n**

- **Example**

  training samples: $class\ \omega_1 : (1,2)^t, (2,0)^t \quad class\ \omega_2 : (3,1)^t, (2,3)^t$

$$Y = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 0 \\ -1 & -3 & -1 \\ -1 & -2 & -3 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

find $Y^\dagger$, then compute $\mathbf{a}^* = Y^\dagger \mathbf{b}$

Vector Derivative (Gradient) and Chain Rule

Consider scalar function of vector input: $J(\mathbf{x})$

- **Vector derivative (gradient)** $\nabla_{\mathbf{x}} J(\mathbf{x}) = [\partial J / \partial x_1, \partial J / \partial x_2, \cdots, \partial J / \partial x_d]^t$
- **inner product** $J = \mathbf{a}^t\mathbf{b} = \sum_k a_k b_k$

$$\Rightarrow \nabla_{\mathbf{a}} \mathbf{a}^t\mathbf{b} = \mathbf{b} \qquad \nabla_{\mathbf{b}} \mathbf{a}^t\mathbf{b} = \nabla_{\mathbf{b}} \mathbf{b}^t\mathbf{a} = \mathbf{a}$$

- **Matrix-vector multiplication** $\nabla_{\mathbf{b}} J = \nabla_{\mathbf{b}} \mathbf{Ab} = \mathbf{A}^t$
- **Hermitian** $J = \mathbf{x}^t A\mathbf{x} = \sum_i \sum_j x_i A_{ij} x_j$ $\Rightarrow \nabla_{\mathbf{x}} \mathbf{x}^t A\mathbf{x} = A\mathbf{x} + A^t\mathbf{x}$
- **Generalized chain rule**

  now consider $\mathbf{x} = A\mathbf{x}'$, $i.e.\ x_i = \sum_j A_{ij} x_j' \Rightarrow \delta x_i / \delta x_j' = A_{ij}$

$$\nabla_{\mathbf{x}'} J = \left(\frac{\delta x_i}{\delta x_j'} \right)^t \nabla_{\mathbf{x}} J \qquad \Rightarrow \nabla_{\mathbf{x}'} J = A^t \nabla_{\mathbf{x}} J$$

- **HW#5 P.1**

- Chap. 5.11 and Burges '98 paper: Support Vector Machine

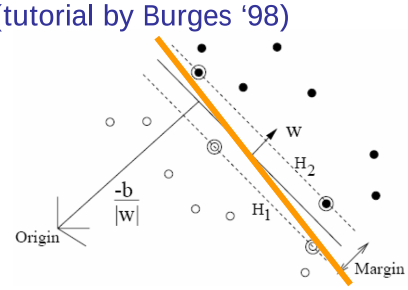
Support Vector Machine (tutorial by Burges '98)

- **Look for separation plane with the highest margin**

  Decision boundary

  $H_0: \ \mathbf{w}^t\mathbf{x}+b=0$

- **HW#5 P.2**



- **Linearly separable**

$\mathbf{w}^t\mathbf{x}_i+b\geq+1 \quad \forall\mathbf{x}_i$ in class ω_1 i.e. $y_i=+1$

$\mathbf{w}^t\mathbf{x}_i+b\leq-1 \quad \forall\mathbf{x}_i$ in class ω_2 i.e. $y_i=-1$

Inequality constraints : $y_i(\mathbf{w}^t\mathbf{x}_i+b)-1\geq 0 \ , \ \forall \ i$

- **Two parallel hyperplanes defining the margin**

  hyperplane $H_1(H_+):\mathbf{w}^t\mathbf{x}_i+b=+1$

  hyperplane $H_2(H_-):\mathbf{w}^t\mathbf{x}_i+b=-1$

- **Margin: sum of distances of the closest points to the separation plane**

  margin = $2/\|\mathbf{w}\|$

- **Best plane defined by w and b**

Finding the maximal margin

minimize $\frac{1}{2}\|\mathbf{w}\|^2$ subject to inequality constraints

$y_i(\mathbf{w}^t\mathbf{x}_i + b) - 1 \geq 0 \quad i = 1, \cdots, l$

- **Use the Lagrange multiplier technique for the constrained opt. problem**

minimize L_p w.r.t. $\mathbf{w}$ and b

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i (y_i(\mathbf{w}^t\mathbf{x}_i + b) - 1)$$

$\alpha_i \geq 0$

$$\frac{dL_p}{d\mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$$

$$\frac{dL_p}{db} = 0 \quad \Rightarrow \quad \sum_{i=1}^{l} \alpha_i y_i = 0$$

Primal Problem

↔

maximize L_D w.r.t. $\mathbf{w}$ and b

$$L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

with conditions :

$$\sum_{i=1}^{l} \alpha_i y_i = 0$$

$\alpha_i \geq 0$

- **Quadratic Programming**

Dual Problem

- **HW#6 P.1**

KKT conditions for separable case

$$\frac{\partial}{\partial w_\nu} L_P = w_\nu - \sum_i \alpha_i y_i x_{i\nu} = 0 \qquad \nu = 1, \cdots, d \quad \longrightarrow \quad \mathbf{w}^* = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$$

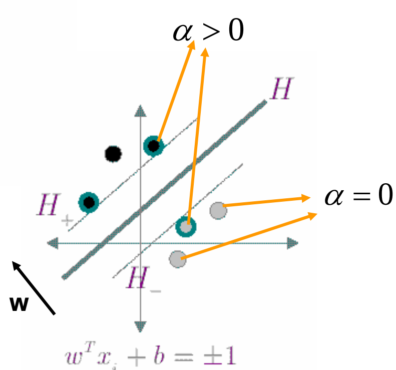
$$\frac{\partial}{\partial b} L_P = -\sum_i \alpha_i y_i = 0$$

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \qquad i = 1, \cdots, l$$

$$\alpha_i \geq 0 \qquad \forall i$$

$$\alpha_i(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1) = 0 \qquad \forall i$$

- **How to compute w and b?**
- **How to classify new data?**



if $\alpha_i > 0$, $\mathbf{x}_i$ is on H_+ or H_- and is a support vector

Non-separable

- **Add slack variables** ξ_i

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 - \xi_i \quad \text{for } y_i = +1$$
$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 + \xi_i \quad \text{for } y_i = -1$$
$$\xi_i \geq 0 \;\forall i.$$

if $\xi_i > 1$, then $\mathbf{x}_i$ is misclassified (i.e. training error)

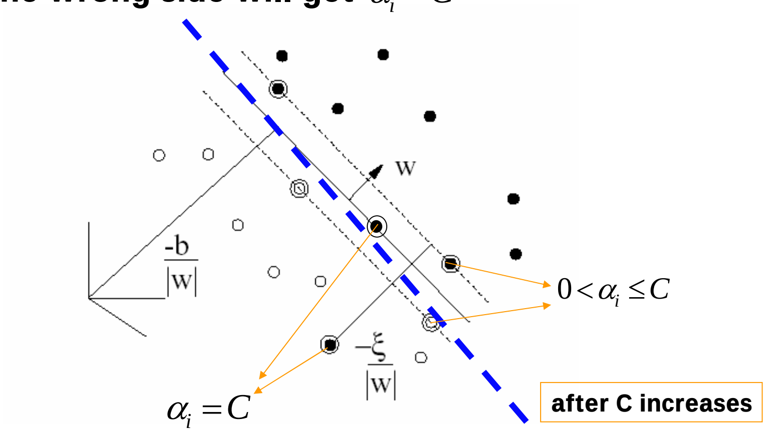
Lagrange multiplier: minimize

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i\{y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i\} - \sum_i \mu_i \xi_i$$

New objective function

Ensure positivity

- **All the points located in the margin gap or the wrong side will get** $\alpha_i = C$



- **When C increases, samples with errors get more weights**
  - **better training accuracy, but smaller margin**
  - **less generalization performance**

Mapping to Higher-Dimension Space

$\Phi : \mathbf{R}^d \mapsto \mathcal{H}.$ Map to a high dimensional space, to make the data separable $\Phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}\, x_1 x_2 \\ x_2^2 \end{pmatrix}$

- **Find the SVM in the high-dim space (embedding space)**

$$g(\mathbf{x}) = \underbrace{\sum_{i=1}^{N_s} \alpha_i y_i \Phi(\mathbf{s}_i)}_{\mathbf{w}} \cdot \Phi(\mathbf{x}) + b$$

- **define kernel** $K(\mathbf{s}_i, \mathbf{x}) = \Phi(\mathbf{s}_i) \cdot \Phi(\mathbf{x})$

$$\Rightarrow g(\mathbf{x}) = \sum_{i=1}^{N_s} \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + b$$

- **We can use the same method (Dual Problem) to maximize L_D to find** α_i

$$L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

$$= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

- **HW#5 P.2**

- Chap. 9 : Analysis of Learning Algorithms

Bias vs. variance for estimator

Assume F is a quantity whose value is to be estimated

data source → **Randomly draw n samples** $D=\{x_1, x_2 \dots, x_n\}$ → **Learn** g_D **to estimate F**

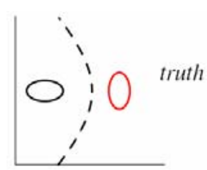
Repeat multiple times

expected estimation error: $E_D\left[\left|g_D - F\right|^2\right]$

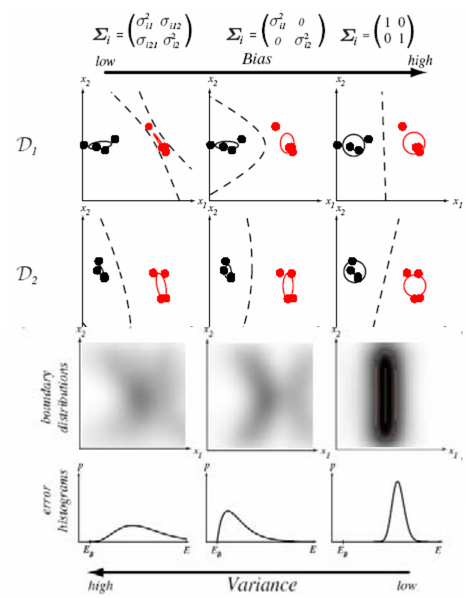
$$=\underbrace{\left[E_D(g_D)-F\right]^2}_{\textbf{Bias}^2}+\underbrace{E_D\left[\left|g_D - E_D(g_D)\right|^2\right]}_{\textbf{Variance}}$$

Bias vs. variance for classification

- **Ground truth: 2D Gaussian**

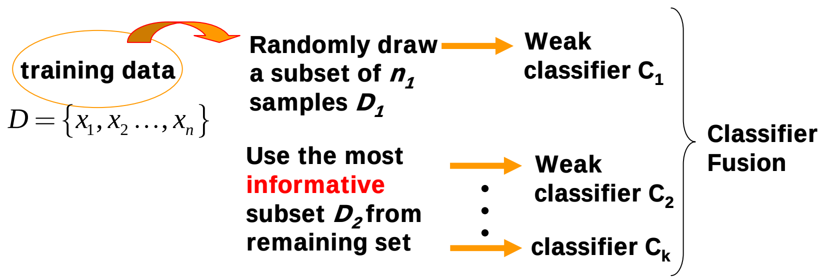


- **Complex models have smaller biases, more variances than simple models**
- **Increasing training pool size helps reduce the variance**
- **Occam's Razor principle**



Boosting

- **For each component classifier, use the subset of data that is most informative given the current set of component classifiers**



HW#7 P.2

Algorithm AdaBoost

Input: set of N labeled examples $\{(1, c(1)), \ldots, (N, c(N))\}$
distribution D over the examples
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

As in AdaBoost Ref.

Initialize the weight vector: $w_i^1 = D(i)$ for $i = 1, \ldots, N$

Do for $t = 1, 2, \ldots, T$

1. Set

$$\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}$$

2. Call **WeakLearn**, providing it with the distribution $\mathbf{p}^t$; get back a hypothesis h_t.
3. Calculate the error of h_t: $\epsilon_t = \sum_{i=1}^N p_i^t |h_t(i) - c(i)|$.
4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
5. Set the new weights vector to be

$$w_i^{t+1} = w_i^t \beta_t^{1 - |h_t(i) - c(i)|}$$

Final Classifier h_f

$$h_f(i) = \begin{cases} 1, & \sum_{t=1}^{T}\left(\log \frac{1}{\beta_t}\right) h_t(i) \geq \frac{1}{2}\sum_{t=1}^{T} \log \frac{1}{\beta_t} \\ 0, & \text{otherwise} \end{cases}.$$

- **When will the final classifier be incorrect?**
- **Suppose c(i)=0, then *$h_f(i)$* is incorrect if**

$$\sum_{t=1}^{T} (\log \beta_t^{-1}) h_t(i) \geq \frac{1}{2}\sum_{t=1}^{T} \log(\beta_t^{-1})$$

$$\text{namely} \prod_{t=1}^{T} \beta_t^{-h_t(i)} \geq \prod_{t=1}^{T} \beta_t^{-1/2} \quad \Rightarrow \quad \prod_{t=1}^{T} \beta_t^{1-h_t(i)} \geq \prod_{t=1}^{T} \beta_t^{1/2}$$

- **In general**

$$h_f(i) \text{ is incorrect if } \prod_{t=1}^{T} \beta_t^{1-|h_t(i)-c(i)|} \geq \left(\prod_{t=1}^{T} \beta_t\right)^{1/2} \qquad \times D(i)$$

$$\Rightarrow D(i)\prod_{t=1}^{T} \beta_t^{1-|h_t(i)-c(i)|} \geq D(i)\left(\prod_{t=1}^{T} \beta_t\right)^{1/2} \quad \Rightarrow \quad w_i^{T+1} \geq D(i)\left(\prod_{t=1}^{T} \beta_t\right)^{1/2}$$

?

$$h_f(i) \text{ is incorrect if } w_i^{T+1} \geq D(i)\left(\prod_{t=1}^{T} \beta_t\right)^{1/2}$$

$$\sum_{i,\, h_f(i)\neq c(i)}^{N} w_i^{T+1} \geq \sum_{i,\, h_f(i)\neq c(i)}^{N} D(i)\left(\prod_{t=1}^{T} \beta_t\right)^{1/2} = E\left(\prod_{t=1}^{T} \beta_t\right)^{1/2}$$

Ref.

Theorem 1 in Ref. $\sum_{i=1}^{N} w_i^{t+1} \leq \sum_{i=1}^{N} w_i^t (1-(1-\beta_t)(1-E_t))$

$$\beta_t = \frac{E_t}{1-E_t}$$

$$\sum_{i=1}^{N} w_i^{t+1} \leq \sum_{i=1}^{N} w_i^t (2E_t) \quad \Rightarrow \sum_{i=1}^{N} w_i^{T+1} \leq \prod_{t=1}^{T} (2E_t)$$

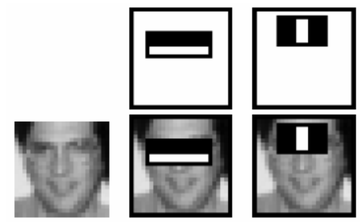
? ?

$$\therefore \quad E \leq \prod_{t=1}^{T} (2E_t) \Big/ \left(\prod_{t=1}^{T} \beta_t\right)^{1/2} = \prod_{t=1}^{T} \left(2\sqrt{E_t(1-E_t)}\right)$$

... Fill in details to complete HW7 P.2

AdaBoost Learning

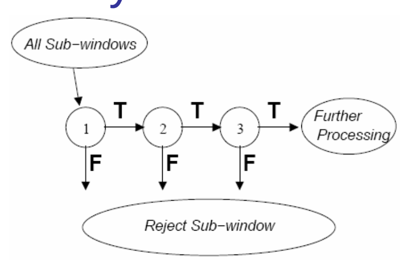
- **The first two features after feature selection**



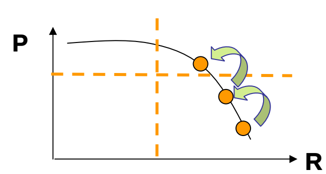
- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2}\sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Cascade classifier for efficiency

- **Break a large classifier into cascade of smaller classifiers**
  - **E.g., 200 features to {1, 10, 25, 50, 50}**
- **Adjust threshold in early stage so that it rejects unlikely regions quickly**

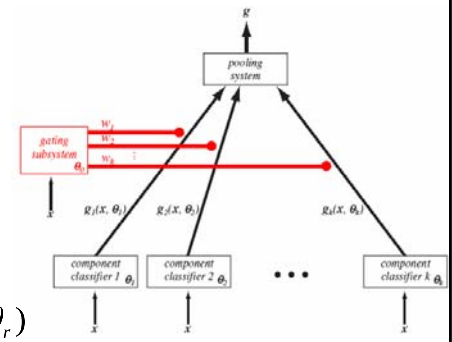


- **Design tradeoffs**
  - **Number of features in each classifier**
  - **Threshold uses in each classifier**
  - **Number of classifiers**
- **Add stages until objective in P-R is met**



Mixture of Experts

- Each component classifier is treated as an expert
- The predictions from each expert are pooled and fused by a gating subsystem



$$P(\mathbf{y} \mid \mathbf{x}, \Theta) = \sum_{r=1}^{k} P(r \mid \mathbf{x}, \theta_0) P(\mathbf{y} \mid \mathbf{x}, \theta_r)$$

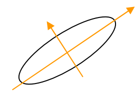
where $\mathbf{x}$ is the input pattern, $\mathbf{y}$ is the output

- **Determine** $P(r \mid \mathbf{x}, \theta_0)$ **, i.e., mixture priors?**
- **Maximize data likelihood** $l(D, \Theta) = \sum_i \ln \sum_{r=1}^{k} P(r \mid \mathbf{x}^i, \theta_0) P(\mathbf{y}^i \mid \mathbf{x}^i, \theta_r)$
  - **gradient decent or EM**

- Chap. 10 :
  feature dimension reduction and clustering

PCA for feature dimension reduction

- Approximate data with reduced dimensions



1-D approximation $\hat{\mathbf{x}} = \mathbf{m} + a\mathbf{e}$, $\mathbf{m}$: mean

Approximation Error $J_1(\mathbf{e}) = \sum_{k=1}^{n} \|\hat{\mathbf{x}}_k - \mathbf{x}_k\| = \sum_{k=1}^{n} \|(\mathbf{m} + a_k\mathbf{e}) - \mathbf{x}_k\|^2$

$$= \sum_{k=1}^{n} a_k^2 \|\mathbf{e}\|^2 - 2\sum_{k=1}^{n} a_k \mathbf{e}^t(\mathbf{x}_k - \mathbf{m}) + \sum_{k=1}^{n} \|\mathbf{x}_k - \mathbf{m}\|^2 = -\sum_{k=1}^{n} \left[\mathbf{e}^t(\mathbf{x}_k - \mathbf{m})\right]^2 + \sum_{k=1}^{n} \|\mathbf{x}_k - \mathbf{m}\|^2$$

$$= -\mathbf{e}^t \underbrace{\left[\sum_{k=1}^{n} (\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^t\right]}_{\mathbf{S}} \mathbf{e} + \sum_{k=1}^{n} \|\mathbf{x}_k - \mathbf{m}\|^2 = -\mathbf{e}^t\mathbf{S}\mathbf{e} + \sum_{k=1}^{n} \|\mathbf{x}_k - \mathbf{m}\|^2$$

S: scatter matrix $= (n-1) \times$ sample covariance

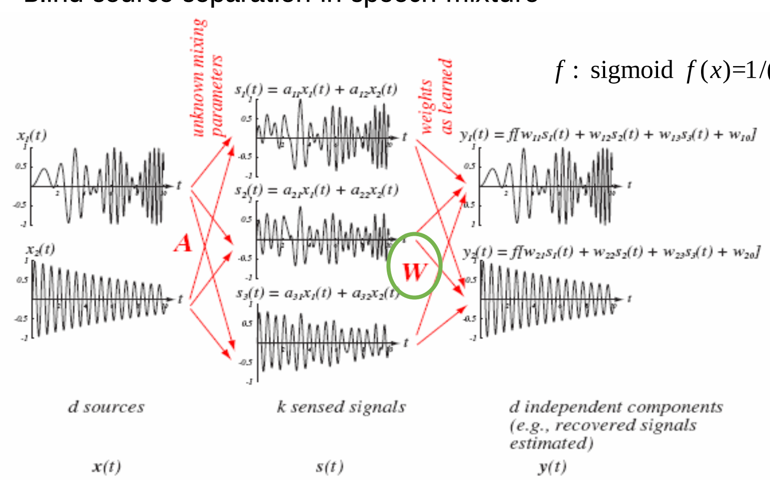
Optimal **e** minimizing error J_1 -- eigenvector of **S** with the largest eigenvalue

Multi-Dim. approximation $\mathbf{x} = \mathbf{m} + \sum_{i=1}^{d'} a_i \mathbf{e}_i$ → what are the optimal $\mathbf{e}_i$?

Independent Component Analysis

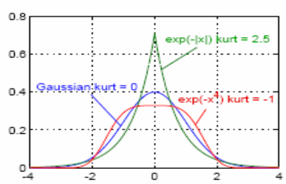
- Seek most independent directions, instead of minimize representation errors (sum-squared-error) as in PCA
- Blind source separation in speech mixture

f : sigmoid $f(x) = 1/(1+e^{-x})$



- Find the best weights to make the output components independent
- How to measure independence?
  - Linear combination of random variables leads to Normal distribution
  - Use the high-order statistics to measure Non-Gaussianity
  - Gradient Decent to weights for discovering each component
    - **Measures of deviations from Gaussianity: 4th moment is Kurtosis ("bulging")**

$$kurt(y) = E\left[\left(\frac{y-\mu}{\sigma}\right)^4\right] - 3$$

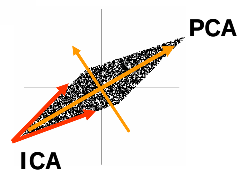


(from Ellis)

-kurtosis of Gaussian is zero (this def.)
-'heavy tails' $\rightarrow kurt > 0$
-closer to uniform dist. $\rightarrow kurt < 0$

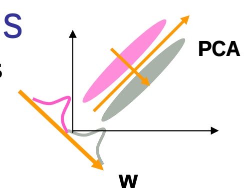
•Directly related to KL divergence from Gaussian PDF

- FastICA Matlab package : http://www.cis.hut.fi/projects/ica/fastica/



LDA: Linear Discriminant Analysis

Given a set of data $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, and their class labels
Find the best projection dimension, $y_i = \mathbf{w}^t \mathbf{x}_i$
so that y_i are most separable



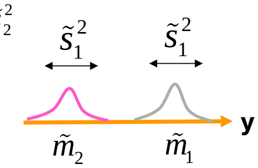
$$\tilde{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{w}^t \mathbf{x} = \mathbf{w}^t \mathbf{m}_i$$

$\mathbf{m}_i$: sample means
$\tilde{m}_i$: sample means of projected points

$$\tilde{s}_i^2 = \frac{1}{n_i} \sum_{\mathbf{y} \in Y_i} (y - \tilde{m}_i)^2$$

$\tilde{s}_1^2 + \tilde{s}_2^2$: within-class scatter

LDA maximizes criterion function: $J(\mathbf{w}) = \dfrac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$



LDA Scatter Matrices

- **P.25-28 of Chap 10**

before projection: $\mathbf{S}_i = \sum_{\mathbf{x}\in D_i} (\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^t$

after projecttion: $\tilde{s}_i^2 = \mathbf{w}^t \mathbf{S}_i \mathbf{w}$

$$\tilde{s}_1^2 + \tilde{s}_2^2 = \mathbf{w}^t(\mathbf{S}_1+\mathbf{S}_2)\mathbf{w} = \mathbf{w}^t\mathbf{S}_\mathbf{w}\mathbf{w}$$

$\mathbf{S}_\mathbf{w} = \mathbf{S}_1 + \mathbf{S}_2$: within-class scatter matrix

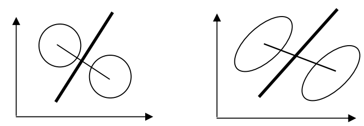
Similarly, between-class scatter matrix $\mathbf{S}_\mathbf{B} = (\mathbf{m}_1-\mathbf{m}_2)(\mathbf{m}_1-\mathbf{m}_2)^t$

$$\Rightarrow J(\mathbf{w}) = \frac{\mathbf{w}^t\mathbf{S}_\mathbf{B}\mathbf{w}}{\mathbf{w}^t\mathbf{S}_\mathbf{w}\mathbf{w}}$$

$$\Rightarrow \mathbf{w}_{opt} = \arg\max J(\mathbf{w}) = \mathbf{S}_\mathbf{w}^{-1}(\mathbf{m}_1-\mathbf{m}_2)$$

Recall the Gaussian Cases

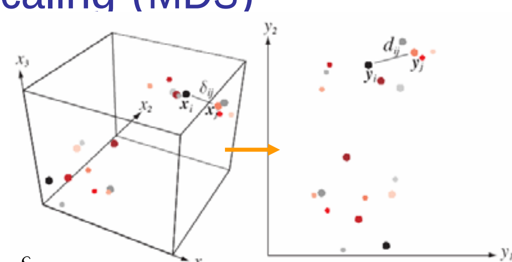
$$w = \Sigma^{-1}(\mu_i - \mu_j)$$



Mean difference vector in the PCA space

Multi-Dimensional Scaling (MDS)

- **Visualize the data points in a lower-dim space**
- **How to preserve the original structure (e.g., distance)?**
- **Optimization Criterion**

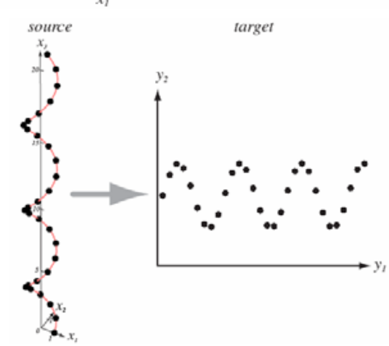


$$J_{ee} = \frac{\sum_{i<j}(d_{ij}-\delta_{ij})^2}{\sum_{i<j}\delta_{ij}^2} \qquad J_{ff} = \sum_{i<j}\left(\frac{d_{ij}-\delta_{ij}}{\delta_{ij}}\right)^2$$

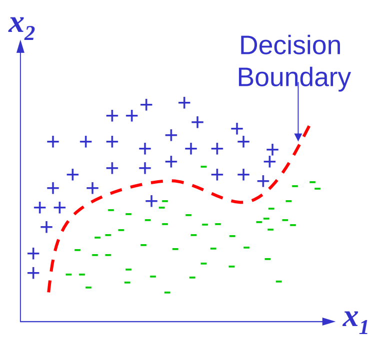
- **Gradient Decent to find new locations**

$$\nabla_{\mathbf{y}_k} J_{ee} = \frac{2}{\sum_{i<j}\delta_{ij}^2}\sum_{j\neq k}\left(d_{kj}-\delta_{kj}\right)\frac{\mathbf{y}_k-\mathbf{y}_j}{d_{kj}}$$

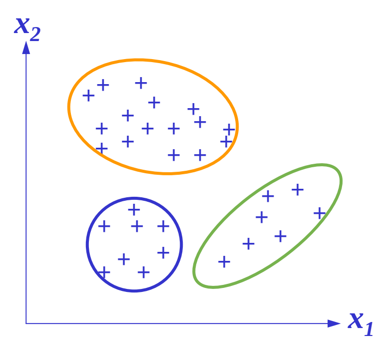
- **Sometimes rank order is more important**



Classification vs. Clustering

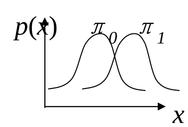


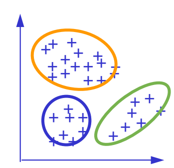
- **Data with labels**
- **Supervised**
- **Find decision boundaries**



- **Data without labels**
- **Unsupervised**
- **Find data structures and clusters**

Review: Mixture Of Gaussians





- Model data distributions as GMM

$$p(x) = \sum_z p(z)\,p(x \mid z)$$

$$= \sum_z \pi_z N\left(x \middle| \mu_z, \Sigma_z\right) \quad = \sum_{z=1}^{Z} \pi_z \frac{1}{(2\pi)^{D/2}\sqrt{|\Sigma_z|}} e^{-\frac{1}{2}(x-\mu_z)^T \Sigma_z^{-1}(x-\mu_z)}$$

- Given data $x_1, \ldots, x_N$, log-likelihood:

$$l = \sum_{n=1}^{N} \log\left(\pi_0 N(x_n|\mu_0,\Sigma_0) + \pi_1 N(x_n|\mu_1,\Sigma_1)\right)$$

- Posterior probability of x being generated by a cluster i

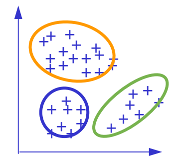
$$posteriers = \tau^i = p\left(z = i \middle| x, \theta\right) \qquad parameter: \;\; \theta = \{\mu_0, \Sigma_0, \mu_1, \Sigma_1\}$$

- Optimization

find $\{\mu_0, \Sigma_0, \mu_1, \Sigma_1\}$ and mixture priors π_z to max. likelihood

GMM for Clustering

- Given the estimated GMM model, compute the probability that x is generated by cluster i



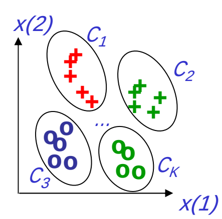
$$posteriers = \tau^i = p\left(z = i \middle| x, \theta\right), \quad \theta = \{\mu_0, \Sigma_0, \mu_1, \Sigma_1, \pi_0\}$$

$$Expectation: \tau_n^{i(t)} = \frac{\pi_i^{(t)} N\left(x_n \mid \mu_i^{(t)}, \Sigma_i^{(t)}\right)}{\sum_j \pi_i^{(t)} N\left(x_n \mid \mu_j^{(t)}, \Sigma_j^{(t)}\right)}$$

- Each sample is assigned to every cluster with a 'soft' decision.

Comparison: K-Mean Clustering

- K-mean clustering
  - Fix K values
  - Choose initial representative of each cluster
  - Map each sample to its closest cluster



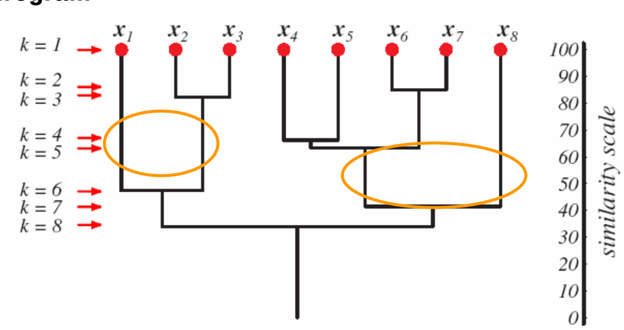
$$\begin{aligned} &for\ i=1,2,...,N, \\ &\quad x_i \rightarrow C_k, if\ \ Dist(x_i, C_k) < Dist(x_i, C_{k'}), k \neq k' \\ &end \end{aligned}$$

Hard decision

  - Re-compute the centers
- Can be used to initialize the EM for GMM

Hierarchical Clustering

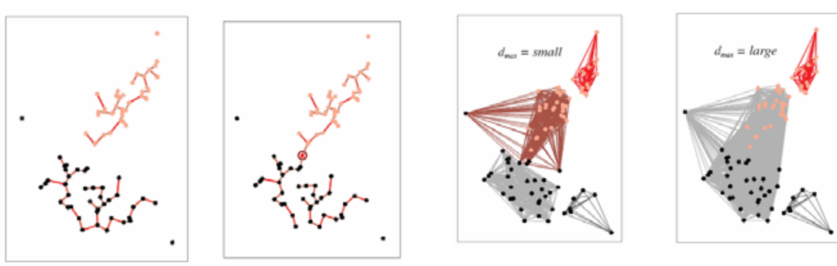
- **Add hierarchical structures to clusters**
  - **many real-world problems have such hierarchical structures**
  - **e.g., biological, semantic taxonomy**
- **Agglomerative vs. Divisive**
- **Dendrogram**



- **Use large gap of similarity to find a suitable number of clusters → clustering validity**

distances or similarity for merging

$$d_{\min}(D_i, D_j) = \min_{\mathbf{x}\in D_i, \mathbf{x}'\in D_j} \|\mathbf{x}-\mathbf{x}'\| \qquad d_{\max}(D_i, D_j) = \max_{\mathbf{x}\in D_i, \mathbf{x}'\in D_j} \|\mathbf{x}-\mathbf{x}'\|$$



- **Nearest neighbor algorithm, minimal algorithm**
- **Merging results in the min. distance spanning tree**
- **But sensitive to noise / outlier**

- **Farthest neighbor algorithm, maximum algorithm**
- **Use distance threshold to avoid large-diameter clusters**
- **Discourage forming elongated clusters**

- **HW#8 P.2**