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

\[ g(x) = w^T x + w_0 \quad \Rightarrow \text{find weight } w \text{ and bias } w_0 \]

- Augmented Vector

\[
y = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \quad \quad a = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}
\]

\[ \Rightarrow g(x) = g(y) = a^T y \]

map \( y \) to class \( \omega_1 \) if \( g(y) > 0 \), otherwise class \( \omega_2 \)

- Normalization

\[ \forall y_i \text{ in class } \omega_2, \quad y_i \leftarrow -(y_i) \]

- Design Objective

\[ a^T y_i > b, \quad \forall y_i \]
Minimal Squared-Error Solution

Training sample matrix:

\[
Y = \begin{bmatrix}
\ y_1^t \\
\ y_2^t \\
\vdots \\
\ y_n^t \\
\end{bmatrix}
\]

dimension: \( n \times (d+1) \)

Objective: \( a^t y_i = b, \ \forall y_i \)

\[\Rightarrow \text{define } J = \sum_{i=1}^{n} (a^t y_i - b)^2 \]

\[\Rightarrow \|a - b\|^2 = (Ya - b)^t(Ya - b) \]

\[
\nabla_a J = 2Y^t(Ya - b) = 0
\]

\[\Rightarrow a = \left(Y^t Y\right)^{-1} Y^t b = Y^t b \]

\[Y^\dagger = \left(Y^t Y\right)^{-1} Y^t \text{ pseudo-inverse : } (d+1) \times n \]

- **Example**
  - training samples: class \( \omega_1: (1,2)^t, (2,0)^t \) class \( \omega_2: (3,1)^t, (2,3)^t \)
  - \[ Y = \begin{bmatrix}
1 & 1 & 2 \\
1 & 2 & 0 \\
1 & 1 & 1 \\
1 & 2 & 3 \\
\end{bmatrix} \]
  - \[ b = \begin{bmatrix}
1 \\
1 \\
1 \\
1 \\
\end{bmatrix} \]

EE6887-Chang

Vector Derivative (Gradient) and Chain Rule

Consider scalar function of vector input: \( J(x) \)

- **Vector derivative (gradient)**
  \[ \nabla_x J(x) = [\partial J / \partial x_1, \partial J / \partial x_2, \ldots, \partial J / \partial x_d]' \]

- **inner product**
  \[ J = a^t b = \sum_k a_k b_k \]
  \[ \Rightarrow \nabla_x a^t b = b \quad \nabla_b a^t b = \nabla_b b^t a = a \]

- **Matrix-vector multiplication**
  \[ \nabla_b J = \nabla_b A b = A' \]

- **Hermitian**
  \[ J = x^t A x = \sum_j \sum_i x_i A_{ij} x_j \]
  \[ \Rightarrow \nabla_{x^t} A x = A x + A' x \]

- **Generalized chain rule**
  now consider \( x = Ax' \), \( i.e. \ x_i = \sum_j A_{ij} x_j' \)
  \[ \Rightarrow \delta x_i / \delta x_j' = A_{ij} \]
  \[ \nabla_{x^t} J = \left( \begin{array}{c}
\frac{\delta x_i}{\delta x_j'}
\end{array} \right) \nabla_x J \]
  \[ \Rightarrow \nabla_{x^t} J = A' \nabla_x J \]

EE6887-Chang

Review Final-5

Review Final-6

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*
  
  \[ H_b: \ w'x + b = 0 \]

- **Linearly separable**
  
  \[ w'x_i + b \geq +1 \quad \forall x_i \text{ in class } \omega_i \text{ i.e. } y_i = +1 \]
  
  \[ w'x_i + b \leq -1 \quad \forall x_i \text{ in class } \omega_i \text{ i.e. } y_i = -1 \]

  Inequality constraints: \( y_i(w'x_i + b) - 1 \geq 0 \), \( \forall i \)

- **Two parallel hyperplanes defining the margin**
  
  hyperplane \( H_1(H_+): w'x + b = +1 \)

  hyperplane \( H_1(H_-): w'x + b = -1 \)

- **Margin**: sum of distances of the closest points to the separation plane
  
  \[ \text{margin} = \frac{2}{\|w\|} \]

  Best plane defined by \( w \) and \( b \)
Finding the maximal margin

\[
\text{minimize } \frac{1}{2} \|w\|^2 \quad \text{subject to inequality constraints } \quad y_i (w^T x_i + b) - 1 \geq 0 \quad i = 1, \ldots, l
\]

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

Primal Problem

\[
L_w = \frac{1}{2} \|w\|^2 - \sum_{i=1}^l \alpha_i (y_i (w^T x_i + b) - 1) \\
\alpha_i \geq 0 \\
\frac{dL_w}{dw} = 0 \quad \Rightarrow \quad w = \sum_{i=1}^l \alpha_i y_i x_i \\
\frac{dL_w}{db} = 0 \quad \Rightarrow \quad \sum_{i=1}^l \alpha_i y_i = 0
\]

Dual Problem

\[
L_\alpha = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\
\text{with conditions: } \\
\sum_{i=1}^l \alpha_i y_i = 0 \\
\alpha_i \geq 0
\]

KKT conditions for separable case

\[
\frac{\partial}{\partial w} L_P = w - \sum_{i=1}^{l} \alpha_i y_i x_i = 0 \quad \nu = 1, \ldots, d \quad \Rightarrow \quad w^* = \sum_{i=1}^{l} \alpha_i y_i x_i
\]

\[
\frac{\partial}{\partial b} L_P = - \sum_{i=1}^{l} \alpha_i y_i = 0
\]

\[
y_i (x_i \cdot w + b) - 1 \geq 0 \quad i = 1, \ldots, l
\]

\[
\alpha_i \geq 0 \quad \forall i
\]

\[
\alpha_i (y_i (w \cdot x_i + b) - 1) = 0 \quad \forall i
\]

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

if \( \alpha_i > 0 \), \( x_i \) is on \( H_+ \) or \( H_- \) and is a support vector
Non-separable

- Add slack variables $\xi_i$

\[
x_i \cdot w + b \geq +1 - \xi_i \quad \text{for } y_i = +1
\]
\[
x_i \cdot w + b \leq -1 + \xi_i \quad \text{for } y_i = -1
\]
$\xi_i \geq 0 \ \forall i$.

if $\xi_i > 1$, then $x_i$ is misclassified (i.e. training error)

Lagrange multiplier: minimize

\[
L_P = \frac{1}{2}||w||^2 + C \sum_i \xi_i - \sum_i \alpha_i \{y_i(x_i \cdot w + b) - 1 + \xi_i\} - \sum_i \mu_i \xi_i
\]

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

\[ \Phi : \mathbb{R}^d \mapsto \mathcal{H}. \]

Map to a high dimensional space, to make the data separable

\[ \Phi(x) = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix} \]

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

\[ g(x) = \sum_{i=1}^{N} \alpha_i y_i \Phi(s_i) \cdot \Phi(x) + b \]

- define kernel

\[ K(s_i, x) = \Phi(s_i) \cdot \Phi(x) \]

\[ \Rightarrow g(x) = \sum_{i=1}^{N} \alpha_i y_i K(s_i, x) + b \]

- We can use the same method (Dual Problem) to maximize \( L_\alpha \) to find \( \alpha_i \)

\[ L_\alpha = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \Phi(x_i) \cdot \Phi(x_j) \]

\[ = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \]

Chap. 9 : Analysis of Learning Algorithms
Bias vs. variance for estimator

Assume $F$ is a quantity whose value is to be estimated

Randomly draw $n$ samples

$$D = \{x_1, x_2, \ldots, x_n\}$$

Learn $g_D$ to estimate $F$

Repeat multiple times

Expected estimation error:

$$E_D \left[ \left| g_D - F \right|^2 \right]$$

$$= \left[ E_D \left( g_D \right) - F \right]^2 + E_D \left[ g_D - E_D \left( g_D \right) \right]^2$$

Bias

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.

Randomly draw a subset of \( n_1 \) samples \( D_1 \)

Use the most informative subset \( D_2 \) from remaining set

Weak classifier \( C_1 \)

Weak classifier \( C_2 \)

Weak classifier \( C_k \)

Classifier Fusion

---

**Algorithm AdaBoost**

**Input:** set of \( N \) labeled examples \( \{(1, c(1)), \ldots, (N, c(N))\} \)

- distribution \( D \) over the examples
- weak learning algorithm \texttt{WeakLearn}
- integer \( T \) specifying number of iterations

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

**Do for** \( t = 1, 2, \ldots, T \)

1. Set
   \[
   p^t = \frac{w^t}{\sum_{i=1}^{N} w_i^t}
   \]

2. Call \texttt{WeakLearn}, providing it with the distribution \( p^t \); get back a hypothesis \( h_t \).

3. Calculate the error of \( h_t \): \( \varepsilon_t = \sum_{i=1}^{N} p_i^t |h_t(i) - c(i)| \).

4. Set \( \beta_t = \varepsilon_t / (1 - \varepsilon_t) \).

5. Set the new weights vector to be
   \[
   w_i^{t+1} = w_i^t \beta_t^{-|h_t(i) - c(i)|}
   \]

**HW#7 P.2**

As in AdaBoost Ref.
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
  $$\sum_{t=1}^{T} (\log \beta_t^{-1}) h_t(i) \geq \frac{1}{2} \sum_{t=1}^{T} \log(\beta_t^{-1})$$
  namely
  $$\prod_{t=1}^{T} \beta_t^{-h_t(i)} \geq \prod_{t=1}^{T} \beta_t^{-1/2} \Rightarrow \prod_{t=1}^{T} \beta_t^{1-h_t(i)} \geq \prod_{t=1}^{T} \beta_t^{1/2}$$

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

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

**Theorem 1 in Ref.**

$$\sum_{i=1}^{N} w_i^{T+1} \geq \sum_{i=1}^{N} w_i(1 - (\beta_i)(1 - E_i))$$

$$\sum_{i=1}^{N} w_i^{T+1} = \sum_{i=1}^{N} w_i(1 - (\beta_i)(1 - E_i))$$

Ref.

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

$$\therefore E \leq \prod_{t=1}^{T} (2E_i) \left( \prod_{t=1}^{T} \beta_t \right)^{1/2} = \prod_{t=1}^{T} (2\sqrt{E_i}(1 - E_i))$$
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 \(w_{1,j} = \frac{1}{n}, \frac{1}{m}\) 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,
     \[
     u_{t,j} = \frac{w_{t,j}}{\sum_{j=1}^{m} w_{t,j}}
     \]
     so that \(u_{t,j}\) 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 \(w_t\), \(e_j = \sum_{i=1}^{n} w_t \cdot \left| h_j(x_i) - y_i \right|\).
  3. Choose the classifier, \(h_j\), with the lowest error \(e_j\).
  4. Update the weights:
     \[
     w_{t+1,j} = w_{t,j} u_{t,j}^{1-e_j}
     \]
     where \(e_j = 0\) if example \(x_i\) is classified correctly, \(e_j = 1\) otherwise, and \(u_{t,j} = \frac{1}{\sum_{j=1}^{m} u_{t,j}}\).
- The final strong classifier is:
  \[
  h(x) = \begin{cases} 
  1 & \sum_{i=1}^{T} \alpha_i h_i(x) \geq \frac{1}{2} \sum_{i=1}^{T} \alpha_i \\
  0 & \text{otherwise}
  \end{cases}
  \]
  where \(\alpha_i = \log \left( \frac{1}{e_i} \right)\)

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

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

where \( x \) is the input pattern, \( y \) is the output

- Determine \( P(r \mid x, \theta_0) \), i.e., mixture priors?
- Maximize data likelihood
  
  \[ l(D, \Theta) = \sum_i \ln \sum_{r=1}^{k} P(r \mid x^i, \theta_0)P(y^i \mid x^i, \theta_r) \]

Chap. 10:  
feature dimension reduction and clustering
PCA for feature dimension reduction

- Approximate data with reduced dimensions

1-D approximation \( \hat{x} = m + ae, \) \( m: \) mean

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

\[
= \sum_{k=1}^{n} a_e^2 \| e \|^2 - 2 \sum_{k=1}^{n} a_e (x_k - m) + \sum_{k=1}^{n} \| x_k - m \|^2 = - \sum_{k=1}^{n} e' (x_k - m) \| e \|^2 + \sum_{k=1}^{n} \| x_k - m \|^2
\]

\[
= -e' \left[ \sum_{k=1}^{n} (x_k - m)(x_k - m) \right] e + \sum_{k=1}^{n} \| x_k - m \|^2 = -e'Se + \sum_{k=1}^{n} \| x_k - m \|^2
\]

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

Optimal \( e \) minimizing error \( J_1 \) -- eigenvector of \( S \) with the largest eigenvalue

Multi-Dim. approximation \( x = m + \sum_{i=1}^{d} a_i e_i \rightarrow \) what are the optimal \( e_i ? \)

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 \( f(x) = \frac{1}{1 + e^{-x}} \)
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")
    \[ kurt(y) = E\left[\left(\frac{y - \mu}{\sigma}\right)^4\right] - 3 \]
  - 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:

---

**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, \( y_i = \mathbf{w}' \mathbf{x}_i \)

so that \( y_i \) are most separable

\[
\bar{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{w}' \mathbf{x} = \mathbf{w}' \mathbf{m}_i \quad \text{m}_i: \text{sample means} \\
\bar{m}_i: \text{sample means of projected points}
\]

\[
\bar{s}_i^2 = \frac{1}{n_i} \sum_{y \in \mathcal{Y}_i} (y - \bar{m}_i)^2 \quad \bar{s}_1^2 + \bar{s}_2^2: \text{within-class scatter}
\]

LDA maximizes criterion function:
\[
J(\mathbf{w}) = \frac{\left|\bar{m}_1 - \bar{m}_2\right|^2}{\bar{s}_1^2 + \bar{s}_2^2}
\]
LDA Scatter Matrices

before projection: \( S_i = \sum_{x \in D_i} (x - m_i)(x - m_i)' \)

after projection:
\[
\tilde{\sigma}_i^2 = w' S_i w \\
\tilde{\sigma}_{2}^2 = w' (S_i + S_2) w = w' S_w w
\]

\( S_w = S_i + S_2 \): within-class scatter matrix

Similarly, between-class scatter matrix \( S_B = (m_1 - m_2)(m_1 - m_2)' \)
\[ \Rightarrow J(w) = \frac{w' S_B w}{w' S_w w} \]
\[ \Rightarrow w_{opt} = \arg \max J(w) = S_w^{-1}(m_1 - m_2) \]

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
\[
J = \sum_{i<j} (d_{ij} - \delta_{ij})^2 \\
J_y = \sum_{i<j} \left( \frac{d_{ij} - \delta_{ij}}{\delta_{ij}} \right)^2
\]

- Gradient Decent to find new locations
\[
\nabla_y J_{y} = \sum_{i<j} \frac{2}{\delta_{ij}} \sum_{j=k} (d_{ij} - \delta_{ij}) \frac{y_k - y_j}{d_{ij}}
\]

- 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
  \[ p(x) = \sum_z p(z) p(x | z) = \sum_z \pi_z N(x | \mu_z, \Sigma_z) = \sum_z \pi_z \frac{1}{(2\pi)^{D/2} \sqrt{|\Sigma_z|}} e^{-\frac{1}{2}(x-\mu_z)^T \Sigma_z^{-1} (x-\mu_z)} \]

- Given data \( x_1, \ldots, x_N \), log-likelihood:
  \[ l = \sum_{n=1}^N \log(\pi_0 N(x_n | \mu_0, \Sigma_0) + \pi_1 N(x_n | \mu_1, \Sigma_1)) \]

- Posterior probability of \( x \) being generated by a cluster \( i \)
  \[ \text{posterior} = \tau^i = p(z = i | x, \theta) \quad \text{parameter: } \theta = \{\mu_0, \Sigma_0, \mu_1, \Sigma_1\} \]

- Optimization
  find \( \{\mu_0, \Sigma_0, \mu_1, \Sigma_1\} \) and mixture priors \( \pi_z \) to max. likelihood
GMM for Clustering

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

$$ post{\text{erior}}s = \tau^i = p(z = i | x, \theta), \quad \theta = \{\mu_0, \Sigma_0, \mu_i, \Sigma_i, \pi_0\} $$

$$ \text{Expectation: } \tau^{(1)}_n = \frac{\pi_i^{(1)} N(x_n | \mu_i^{(1)}, \Sigma_i^{(1)})}{\sum_j \pi_j^{(1)} N(x_n | \mu_j^{(1)}, \Sigma_j^{(1)})} $$

- 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

for $i = 1, 2, ..., N$,

$$x_i \rightarrow C_k \text{ if } \text{Dist}(x_i, C_k) < \text{Dist}(x_i, C_{k'}), k \neq k'$$

end

- 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

\[ d_{\text{min}}(D_i, D_j) = \min_{x \in D_i, x' \in D_j} \|x - x'\| \quad d_{\text{max}}(D_i, D_j) = \max_{x \in D_i, x' \in D_j} \|x - x'\| \]

- 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