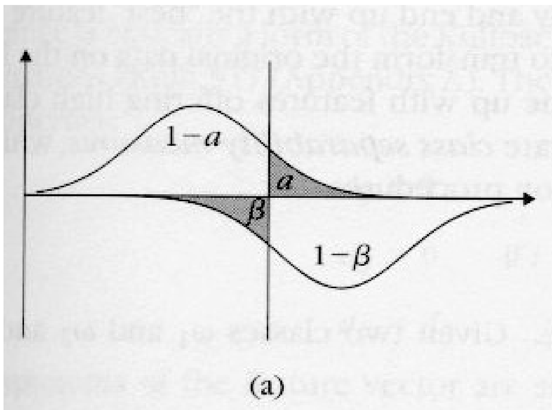
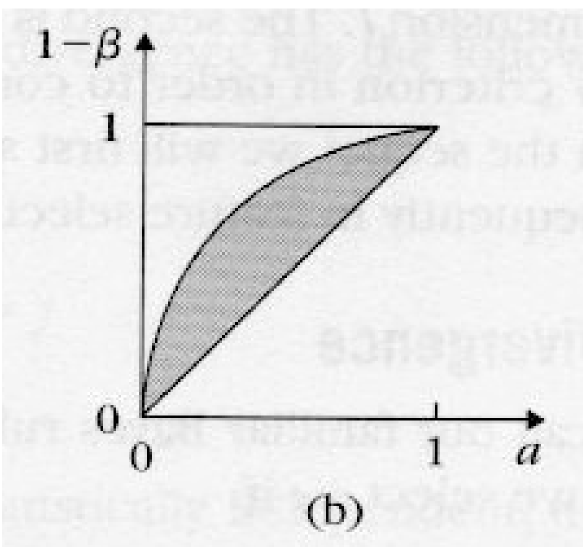
R·I·T

Feature Selection: ROC and Subset Selection

Theodoridis 5.5-5.7

Using ROC for Feature Selection





Hypothesis Tests Examined (e.g. t-test):

Useful for discarding features

But does not tell us about overlap between classes for a feature!

At Left (a): Feature for two class prob.

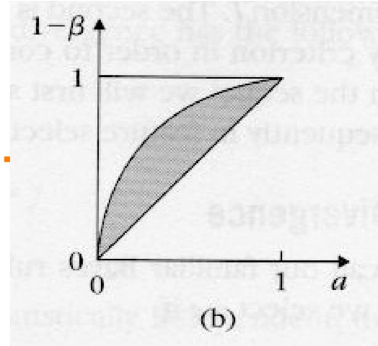
a: P(error for ω1) right of threshold

1-β: P(correct for ω2) right of threshold

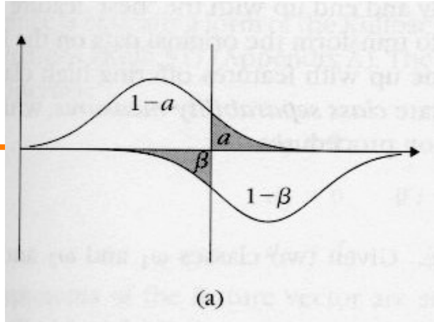
ROC:

Sweep the threshold over the feature value range, record a, 1-β

R·I·T



ROC Cont'd



Metric for Class Discrimination by Feature

Area of the upper-left triangle in the ROC

- Complete overlap: 0 (a = 1 - β everywhere)
- Complete separation: 1/2

In practice, can be estimated using a training sample, sweeping the threshold through the feature value range

R·I·T

Measuring Class Separation Using Multiple Features

Applications

- Identify best feature or fixed-length feature vector
- Define criteria used in transforming original data to produce features that better separate classes

Divergence

Recall: Bayes Rule for 2 classes

Choose ω_1 if $\quad P(\omega_1|\mathbf{x}) > P(\omega_2|\mathbf{x})$

The mean ratio of the class-conditional pdfs can be used to quantify discrimination of class 1 vs. class 2 based on features (similar for class 2, D_{21}):

$$D_{12} = \int_{-\infty}^{+\infty} p(\mathbf{x}|\omega_1) \ln \frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} d\mathbf{x}$$

Divergence is defined by:

$$d_{12} = D_{12} + D_{21}$$

R·I·T

Divergence: Multiple Classes

Compute divergence for every pair of classes:

$$d_{ij} = D_{ij} + D_{ji} = \int_{-\infty}^{+\infty} (p(\mathbf{x}|\omega_i) - p(\mathbf{x}|\omega_j)) \ln \frac{p(\mathbf{x}|\omega_i)}{p(\mathbf{x}|\omega_j)} d\mathbf{x}$$

Then compute the average divergence:

$$d = \sum_{i=1}^{|\Omega|} \sum_{i=1}^{|\Omega|} P(\omega_i) P(\omega_j) d_{ij}$$

Limitation:

Divergence directly related to Bayes Error for Gaussian (normal) distributions, but not more general distributions

- For normal distributions with equal covariance, divergence becomes the Mahalanobis distance between the mean vectors

$$d_{ij} = (\mu_\mathbf{i} - \mu_\mathbf{j})^T \Sigma^{-1} (\mu_\mathbf{i} - \mu_\mathbf{j})$$

Chernoff Bound

Provides

An upper bound for error of a two-class Bayesian classifier:

$$P_e = \int_{-\infty}^{+\infty} \min[P(\omega_i)p(\mathbf{x}|\omega_i), P(\omega_j)p(\mathbf{x}|\omega_j)]d\mathbf{x}$$

using the inequality:

$$\min[a, b] \leq a^s b^{1-s} \text{ for } a, b \geq 0, \text{ and } 0 \leq s \leq 1$$

Chernoff Bound, Continued

$$P_e \leq P(\omega_i)^s P(\omega_j)^{1-s} \int_{-\infty}^{\infty} p(\mathbf{x}|\omega_i)^s p(\mathbf{x}|\omega_j)^{1-s}\, d\mathbf{x} \equiv \epsilon_{CB} \tag{5.25}$$

ϵ_{CB} is known as the *Chernoff bound*. The minimum bound can be computed by minimizing ϵ_{CB} with respect to s. A special form of the bound results for $s = 1/2$:

$$P_e \leq \epsilon_{CB} = \sqrt{P(\omega_i)P(\omega_j)} \int_{-\infty}^{\infty} \sqrt{p(\mathbf{x}|\omega_i)p(\mathbf{x}|\omega_j)}\, d\mathbf{x} \tag{5.26}$$

For Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i), \mathcal{N}(\boldsymbol{\mu}_j, \Sigma_j)$ and after a bit of algebra, we obtain

$$\epsilon_{CB} = \sqrt{P(\omega_i)P(\omega_j)} \exp(-B)$$

where

$$B = \frac{1}{8}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T \left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) + \frac{1}{2} \ln \frac{\left|\frac{\Sigma_i + \Sigma_j}{2}\right|}{\sqrt{|\Sigma_i||\Sigma_j|}} \tag{5.27}$$

B: *Bhattacharyya distance*

R·I·T

Bhattacharyya Distance

$$\epsilon_{CB} = \sqrt{P(\omega_i)P(\omega_j)}\exp(-B)$$

where

$$B = \frac{1}{8}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T \left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) + \frac{1}{2}\ln \frac{\left|\frac{\Sigma_i + \Sigma_j}{2}\right|}{\sqrt{|\Sigma_i||\Sigma_j|}} \qquad (5.27)$$

This is the optimal Chernoff bound for identical covariance matrices, Σi, Σj

- Bhattacharyya distance becomes proportional to Mahalanobis distance

Scatter Matrices

Class Separability Criteria so far...

Not easily computed, unless we assume Gaussian distributions

And so now...

We'll look directly at the distribution of our samples in feature space

Measuring Scatter

1. Within-class scatter matrix

Average feature variance per class $$S_w = \sum_{1=1}^{|\Omega|} P(\omega_i)\Sigma_i$$

2. Between-class scatter matrix

Average variance of class means vs. global mean (μ_0)

$$S_b = \sum_{i=1}^{|\Omega|} P(\omega_i)(\mu_i - \mu_0)(\mu_i - \mu_0)^T \qquad \mu_0 = \sum_{i=1}^{|\Omega|} P(\omega_i)\mu_i$$

3. Mixture scatter matrix

Feature covariance with respect to global mean:

$$S_m = S_w + S_b$$

R·I·T

Class Separability Criteria Using Scatter Matrices

$$J_1 = \frac{trace(S_m)}{trace(S_w)}$$

Large when samples cluster tightly around their class means, and classes are well-separated

Top: sum of feature variances around the global mean

Bottom: measure of average feature variance across classes

Related criterion (*invariant under linear transformations*):

$$J_3 = trace\{S_w^{-1} S_m\}$$

(Note: trace is the sum of diagonal elements in a matrix)

Fisher's Discriminant Ratio

For one dimensional, two class problems

Can use sample-based mean and variance estimates

$$FDR = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$

For multi-class problems, we can use the average FDR value across all class pairs

Feature Subset Selection

Problem:

Select *k* of *m* available features, with the goal of maximizing class separation

Approaches:

- *Scalar feature selection*: treat features individually (ignores feature correlations)
- *Feature vector selection*: consider feature sets (and feature correlations)

Scalar Feature Selection

Procedure:

1. Compute class separability criterion for each feature
   - e.g. ROC, FDR, or divergence
   - Average values needed in multi-class case, or can use minimum between-class criterion values ('maxmin' strategy)
2. Rank features in descending order of criterion values
3. Select the *k* highest ranking features

Taking Correlation into account

Cross-correlation coefficients may be included in a weighted criterion (see p. 283-284 of Theodoridis)

R·I·T

Brute-Force Feature Vector Selection

'Filter' Approach

Find the optimal feature vector of length k by evaluating class separation criterion for all possible feature vectors

For m features, vectors of size k:

$$\binom{m}{k} = \frac{m!}{k!(m-k)!}$$

- e.g. m = 20, k = 5 : 15, 504 length 5 vectors
- *worse* if we want to try over different k

Brute Force, Part 2: Wrapper Approach

Evaluate Features Using Classifiers

...not class separation criterions. Again, simplest approach is brute-force.

Can be more expensive than 'Filter' approach (due to expense in training classifiers, e.g. a neural net, decision tree, or SVM)

Suboptimal Search for Feature Vector of Size k

Backward Selection

Start with all features in a vector (*m* features)

Iteratively eliminate one feature, compute class separability criterion

Keep combination with the highest criterion value

Repeat with chosen combination until we have a vector of size k

Number of Combinations Generated

$$1 + \frac{(m+1)m - k(k+1)}{2}$$

Suboptimal Search, Cont'd

Forward Search

1. Compute criterion value for each feature
2. Select feature with best value
3. Form all possible pairings of best vector with another unused feature
    - Evaluate each using the criterion, select best vector
4. Repeat step 3 until we have a vector of size k

Combinations Generated:

$$km - \frac{k(k-1)}{2}$$

*less efficient than backward search for k close to m

R·I·T

Floating Search (forward direction)

Heuristic search that alternates ('floats') between adding and removing features in order to improve the criterion value

Rough idea: as we add a feature (forward), check smaller feature sets to see if we do better with this feature replacing a previously selected feature (backward). Terminate when k features selected.

(see p. 287 for pseudo code)

Optimal Approaches

If criterion is monotonic (non-decreasing as features are added), we have more efficient methods to find the optimal feature set of size k (vs. brute force)

Dynamic Programming

Branch-and-Bound