Pattern Classification – E186 Handout

General Concepts of Pattern Recognition

Pattern Recognition (PR) is a general term which may mean any of the following with subtle conceptual differences.

- **Recognition**

  Given a pattern (e.g., image objects such as a human face, printed or written text, a natural scene, etc.), a name (perceptual category) is generated as the output.

- **Classification/Categorization**

  The input patterns of different classified/categorized into a set of classes/categories.

- **Association**

  The associator learns to establish the connection between two patterns (concepts).

- **Completion**

  similar to the content addressable nature of the brain.

Different methodologies can be used for PR, such as *statistical PR*, *syntactic (structural) PR*, and *neural network PR*. And PR is in general a two-stage process:

- **Training (learning)**

  The computer is trained by one of the two strategies:

  – *Supervised learning* (learning with teacher) *a priori* knowledge about the patterns to be recognized is assumed to be available for teaching the computer how to recognize patterns;

  – *Unsupervised learning* (learning without teacher) no such *a priori* knowledge is available. Computer has to learn by itself.

- **Testing**

  The trained computer recognizes patterns.

Basic Definitions

- **Feature Space**

  We assume a set of n features can be extracted from the image to represent the image objects of interest. They form an n-dimensional space called *feature space*, in which each axis represents the measurement of a certain feature.

- **Patterns**

  The measurements of the n features of an object to be recognized are obtained and represented by an n-dimensional vector $X = [x_1, \cdots, x_n]^T$. This vector is called a pattern, or a sample, and can be represented by a point in the feature space: $X \in R^n$.

- **Pattern Classes (categories)**

  Assume there are c possible classes to which a given pattern X may belong. These classes are members of a class set:

  $$\Omega = \{\omega_1, \cdots, \omega_c\}$$

- **Pattern Classification**

  Given an image pattern $X \in R^n$, find a class $\omega \in \Omega$ to which X most likely belongs. Classification process can be be carried out as

  $$X \sim \omega_k \quad iff \quad D_k(X) = max\{D_i, \ i = 1, \cdots, c\}$$

  where $D_i(X)$ is the *discriminant function* of class ω_i which can be obtained according to the specific classification method used.

Classification can be considered as a partitioning of the feature space into c regions each corresponding to a class, by the boundaries

$$D_i(X) = D_j(X), \quad for\ all\ i \neq j$$

- **Feature Selection**

  The number of features actually used in classification can be reduced from n to m by *feature selection* for two reasons: (1) to use only those features which are most relevant to a specific application, (2) to reduce the computational cost.

  Feature selection may be achieved in two different ways:

  - Choosing m features directly from the n available features (C_n^m ways to do so.)
  - Using m linear combinations of the n features as new features:

    $$Y = A^T X$$

    where X is the n-dimensional pattern, A is an $n \times m$ matrix, and $Y = [y_1, \cdots, y_m]^T$ is an m-dimensional pattern. Each y_i is a new feature which is a linear combination of all n original features $x_1, \cdots, x_n$.

  For either method, some criterion is needed to guide the selection (i.e., which m features to choose, how to find the matrix A).

Distance Measurements

Any *distance metric* $D(X,Y)$ between two points X and Y must satisfy:

1. $D(X,Y) \geq 0, (D(X,Y) = 0 \text{iff} X = Y)$;
2. $D(X,Y) = D(Y,X)$;
3. $D(X,Y) \leq D(X,Z) + D(Z,Y)$.

where Z is an arbitrary point.

Several specific distance measurements can be used in the feature space in the process of pattern classification.

- **Distance between two points**

$$D_L(X,Y) = [\sum_{i=1}^{n} |x_i - y_i|^L]^{1/L}$$

Where L can take any value such as 1, 2, and infinity.

When $L = 2$, $D_L(X,Y)$ becomes Euclidean distance:

$$D_E(X,Y) \triangleq \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} = [(X-Y)^T(X-Y)]^{1/2}$$

When $L = 1$, $D_L(X,Y)$ is called *city-block distance*

$$D_c(X,Y) \triangleq \sum_{i=1}^{n} |x_i - y_i|$$

which is much less costly to compute than the Euclidean distance, as absolute operation is much easier than multiplication and square root operation.

When $L = \infty$, $D_L(X,Y)$ is so called *maximum distance*

$$D_m(X,Y) = max\{|x_i - y_i|\,, \;\; i = 1, \cdots, n\}$$

- **Distance between a point and a distribution**

  The *Mahananobis distance* can be used to measure the distance between a pattern X and a class ω_i represented by its mean vector M_i and the covariance matrix Σ_i:

$$D_M(X, \omega_i) = (X - M_i)^T \Sigma_i^{-1} (X - M_i)$$

- **Distance between two distributions**

  The *Bhattacharrya distance* can be used to measure the distance between two classes ω_i and ω_j represented by their mean vectors and covariance matrices:

$$D_B(\omega_i, \omega_j) = \frac{1}{4}(M_i - M_j)^T \left[\frac{\Sigma_i + \Sigma_j}{2}\right]^{-1} (M_i - M_j) + log\left[\frac{\left|\frac{\Sigma_i + \Sigma_j}{2}\right|}{(|\Sigma_i|\ |\Sigma_j|)^{1/2}}\right]$$

Separability Criteria for Feature Selection

In feature selection, we need to evaluate how separable a set of classes are in an m-dimensional feature space by some criteria such as the one discussed here.

- Total number of samples:

$$N = \sum_{i=1}^{c} N_i$$

  where N_i is the number of samples in class ω_i

- Overall mean vector:

$$M = \frac{1}{N}\sum_{X} X = \frac{1}{N}\sum_{i=1}^{c} N_i(\frac{1}{N_i}\sum_{X\sim\omega_i} X) = \sum_{i=1}^{c}\frac{N_i}{N}M_i = \sum_{i=1}^{c} P_i M_i$$

  where $P_i = N_i/N$ is the *a priori* probability of class ω_i

- Scatter matrix of class ω_i (same as the covariance matrix of the class):

$$S_i = \frac{1}{N_i}\sum_{X\sim\omega_i}(X-M_i)(X-M_i)^T = \Sigma_i$$

- Within-class scatter matrix:

$$S_W = \sum_{i=1}^{c} P_i S_i = \sum_{i=1}^{c} P_i \Sigma_i$$

- Between-class scatter matrix:

$$S_B = \sum_{i=1}^{c} P_i(M_i - M)(M_i - M)^T$$

- Total scatter matrix:

$$S_T = \frac{1}{N}\sum_{X}(X-M)(X-M)^T = \frac{1}{N}\sum_{i=1}^{c}\sum_{X\sim\omega_i}(X-M)(X-M)^T$$

We can show that $S_T = S_W + S_B$, i.e., the total scatter is the sum of within-class scatter and between-class scatter.

$$
\begin{aligned}
S_T &= \frac{1}{N}\sum_{i=1}^{c}\sum_{X\sim\omega_i}(X-M)(X-M)^T \\
&= \frac{1}{N}\sum_{i=1}^{c}\sum_{X\sim\omega_i}(X-M_i+M_i-M)(X-M_i+M_i-M)^T \\
&= \frac{1}{N}\sum_{i=1}^{c}\sum_{X\sim\omega_i}[(X-M_i)(X-M_i)^T+(X-M_i)(M_i-M)^T \\
&\quad +(M_i-M)(X-M_i)^T+(M_i-M)(M_i-M)^T] \\
&\overset{*}{=} \frac{1}{N}\sum_{i=1}^{c}\sum_{X\sim\omega_i}(X-M_i)(X-M_i)^T+\frac{1}{N}\sum_{i=1}^{c}\sum_{X\sim\omega_i}(M_i-M)(M_i-M)^T \\
&= \sum_{i=1}^{c}\frac{N_i}{N}S_i+\sum_{i=1}^{c}\frac{N_i}{N}(M_i-M)(M_i-M)^T \\
&= S_W+S_B
\end{aligned}
$$

* You should be able to show why the two middle terms disappeared.

Now we can define:

$$J_a = tr(S_W^{-1}S_B)$$

or

$$J_b = det(S_W^{-1}S_B)$$

where $tr(A)$ and $det(A)$ represent trace and determinant of matrix A, respectively.

These J's are measurements of the separability among all classes and can be used as criteria in feature selection, i.e., to obtain m from the n features to form a sub-feature space in which the separability is maximized.

Nearest Neighbor Classifier

- **Training**

  This is a supervised classification method as a set of *training samples* (patterns of known classes) is assumed to be available for all c classes:

  $$\{X_1^{(k)}, \cdots, X_{N_k}^{(k)}\} \quad (k = 1, \cdots, c)$$

  The classification is directly based on these training samples with no further training needed.

- **Classification**

  First define the *nearest neighbor distance* between a pattern X and a class represented by its training samples as

  $$Dist(X, \omega_k) \triangleq min\{D_L(X, X_i^{(k)}), \quad i = 1, \cdots, N_k\}$$

  Then a pattern X of unknown class is classified to its nearest neighbor's class:

  $$X \sim \omega_k \quad iff \quad Dist(X, \omega_k) = min\{Dist(X, \omega_j), \quad j = 1, \cdots, c\}$$

Since this method depends highly on individual training samples, it is sensitive to noise.

Minimum Distance Classifier

- **Training**

  The kth class ω_k is represented by its mean vector M_k and covariance matrix which can be estimated from the training samples:

$$M_k = \frac{1}{N_k}\sum_{i=1}^{N_k} X_i^{(k)} \quad (k = 1, \cdots, c)$$

  and

$$\Sigma_k = \frac{1}{N_k}\sum_{i=1}^{N_k}(X_i^{(k)} - M_k)(X_i^{(k)} - M_k)^T$$

- **Classification**

  A given pattern X of unknown class is classified to ω_k if its Mahalanobis distance to ω_k is smaller than those to all other classes:

$$X \sim \omega_k \quad iff \quad D_M(X, \omega_k) = min\{D_M(X, \omega_i) \;\; i = 1, \cdots, c\}$$

For simplicity, the distance $D_L(X, M_i)$ can be used to replace $D_M(X, \omega_i)$ above. As now only the mean vector of each class is used, the classification does not take into account how the classes are distributed in the feature space.

Bayes Classifier — the Optimal Classifier

The basic principle

- $P(\omega_k)$: the *a priori* probability that an arbitrary pattern belongs to class ω_k.
- $P(\omega_k/X)$: the *a posteriori* conditional probability that a specific pattern X belongs to class ω_k.
- $p(X)$: the density distribution of all patterns.
- $p(X/\omega_k)$: the conditional density distribution of all patterns belonging to ω_k.

  Note that $p(X)$ is the weighted sum of all $p(X/\omega_i)$ for $i = 1, \cdots, c$:

  $$p(X) = \sum_{i=1}^{c} p(X/\omega_i)P(\omega_i)$$

- **The Bayes' Theorem**

  $$P(\omega_k/X) = \frac{p(X/\omega_k)P(\omega_k)}{\sum_{i=1}^{c} p(X/\omega_i)P(\omega_i)} = \frac{p(X/\omega_k)P(\omega_k)}{p(X)}$$

- **Training**

  The *a priori* probability $P(\omega_i)$ can be estimated from the training samples as $P(\omega_i) = P_i = N_i/N$, assuming the training samples are randomly chosen from all the patterns.

  We also need to estimate $p(X/\omega_i)$. If we don't have any good reason to believe otherwise, we will assume the density to be a normal distribution:

  $$p(X/\omega_i) = N(X, M_i, \Sigma_i) = \frac{1}{(2\pi)^{n/2} \left|\Sigma_i\right|^{1/2}} exp[-\frac{1}{2}(X-M_i)^T\Sigma_i^{-1}(X-M_i)]$$

  where the mean vector M_i and the covariance matrix Σ_i can be estimated from the training samples as shown before.

- **Classification**

  A given pattern X of unknown class is classified to ω_k if "it is most likely that X belongs to ω_k" (that's why it is called optimal classifier), i.e.:

  $$X \sim \omega_k \quad iff \quad P(\omega_k/X) = max\{P(\omega_i/X), \;\; i = 1, \cdots, c\}$$

  As shown above, the likelihood $P(\omega_k/X)$ can be written as

  $$P(\omega_k/X) = \frac{p(X/\omega_k)P(\omega_k)}{p(X)}$$

  and the denominator $p(X)$ can be dropped as it is common in all $P(\omega_k/X)$'s, therefore a discriminant function

  $$D_i(X) = p(X/\omega_k)P(\omega_i)$$

  can be used in the classification:

  $$X \sim \omega_k \quad iff \quad D_k(X) = max\{D_i(X) \quad i = 1, \cdots, c\}$$

Error analysis

First consider 2-class case ($c = 2$). Let $P(X \in R_i \cap X \sim \omega_j)$ denote the joint probability that X belongs to ω_j but is in region R_i, then the total probability of error (misclassification) is:

$$\begin{aligned} P(error) &= P(X \in R_2 \cap X \sim \omega_1) + P(X \in R_1 \cap X \sim \omega_2) \\ &= P(X \in R_2/\omega_1)\, P(\omega_1) + P(X \in R_1/\omega_2)\, P(\omega_2) \\ &= \int_{R_2} p(X/\omega_1) dX\, P(\omega_1) + \int_{R_1} p(X/\omega_2) dX\, P(\omega_2) \end{aligned}$$

Next consider multi-class case. As there are many different ways to have a wrong classification and only one way to get it right, consider

$$\begin{aligned} P(correct) &= \sum_{i=1}^{c} P(X \in R_i \cap X \sim \omega_i) \\ &= \sum_{i=1}^{c} P(X \in R_i/\omega_i)\, P(\omega_i) \\ &= \sum_{i=1}^{c} P(\omega_i) \int_{R_i} P(X/\omega_i) dX \end{aligned}$$

Some special cases

As $p(X/\omega_i)$ is only used relatively among all classes, it can be replaced by a monotonic log function and the discriminant function becomes

$$\begin{aligned} D_i(X) &= ln\, p(X/\omega_i)\, P(\omega_i) = ln\, p(X/\omega_i) + ln\, P(\omega_i) \\ &= -\frac{1}{2}(X - M_i)^T \Sigma_i^{-1} (X - M_i) - \frac{n}{2}\, ln\, 2\pi - \frac{1}{2}\, ln\, |\Sigma_i| + ln\, P(\omega_i) \end{aligned}$$

The second term $-n\, ln\, 2\pi/2$ is a constant common to all D_i's and can be dropped.

Now consider several special cases:

- All classes have equal *a priori* probability:

  $$P(\omega_i) = P(\omega_j) \quad for\ all\ i, j$$

  then the last term of $D_i(X)$ can be dropped.

- All classes have the same isotropic distribution:

  $$\Sigma_i = \sigma^2\, I = diag[\sigma^2, \cdots, \sigma^2]$$

  then

  $$|\Sigma_i| = \sigma^{2n}$$

  and $D_i(X)$ becomes

  $$D_i(X) = -\frac{|X - M_i|^2}{2\sigma^2} + ln\, P(\omega_i)$$

Note that the term $ln\,|\Sigma_i|$ has been dropped from the original expression of $D_i(X)$ as it is now the same for all classes.

Consider the boundary in the feature space between ω_i and ω_j:

$$D_i(X) = D_j(X)$$

This can be simplified to a linear equation:

$$W^T X - w = 0$$

where W is a vector

$$W = M_i - M_j$$

and w is a scalar

$$w = -(M_i^T M_i - M_j^T M_j) + 2\sigma^2 \; ln \, \frac{P(\omega_i)}{P(\omega_j)}$$

This linear equation represents a hyperplane between the two points M_i and M_j and perpendicular to the straight line passing through these points.

Further, when all classes have the same $P(\omega_i)$, $D_i(X)$ becomes

$$D_i(X) = -\,|X - M_i|^2 = -(X - M_i)^T (X - M_i) = D_2(X, M_i)$$

and the Bayes classifier becomes minimum distance classifier using Euclidean distance (maximizing $D_i(X)$ is equivalent to minimizing $D_2(X, M_i)$).

- All classes have the same distribution:

$$\Sigma_i = \Sigma \quad (i = 1, \cdots, c)$$

but different M_i. $D_i(X)$ becomes

$$D_i(X) = -\frac{1}{2}(X - M_i)^T \Sigma^{-1} (X - M_i) + ln\, P(\omega_i)$$

When all classes have the same $P(\omega_i)$, the classifier is equivalent to minimum distance classifier using Mahananobis distance.

Consider the boundary in the feature space between ω_i and ω_j:

$$D_i(X) = D_j(X)$$

This can be simplified to a linear equation:

$$W^T X - w = 0$$

where

$$W = \Sigma^{-1}(M_i - M_j)$$

and

$$w = -\frac{1}{2}(M_i^T \Sigma^{-1} M_i - M_j^T \Sigma^{-1} M_j) + ln \frac{P(\omega_i)}{P(\omega_j)}$$

This linear equation represents a hyperplane between the two points M_i and M_j and perpendicular to the straight line $\Sigma^{-1}(M_i - M_j)$ (the straight line $M_i - M_j$ rotated by matrix Σ^{-1}).

- All classes have different Σ_i.

  This is the most general case and the boundary between any two classes ω_i and ω_j

  $$D_i(X) = D_j(X)$$

  becomes a quadric (multivariable quadratic) equation:

  $$X^T \mathbf{W} X + W^T X + w = 0$$

  where **W** is an n by n matrix:

  $$\mathbf{W} = -\frac{1}{2}(\Sigma_i^{-1} - \Sigma_j^{-1})$$

  $$W = \Sigma_i^{-1} M_i - \Sigma_j^{-1} M_j$$

  and

  $$w = -\frac{1}{2}(M_i^T \Sigma^{-1} M_i - M_j^T \Sigma^{-1} M_j) - \frac{1}{2} ln \frac{|\Sigma_i|}{|\Sigma_j|} + ln \frac{P(\omega_i)}{P(\omega_j)}$$

  These boundaries in the nD feature space are in general quadric hyper-surfaces such as hyper-sphere, hyper-ellipsoid, hyper-parabola, hyper-hyperbola, etc.

Unsupervised Classification – Clustering

Given a set of samples $\{X_i,\ i = 1, 2, ..., N\}$ (where each $X_i = [x_1^{(i)}, \cdots, x_n^{(i)}]^T$ is a column vectors representing a point in the n-dimensional feature space), group them into a set of clusters according to their natural distribution in the feature space. Clustering is unsupervised classification as no *a priori* knowledge (such as samples of known classes) is assumed to be available.

The K-Means Algorithm

- Step 1. Arbitrarily choose from the given sample set k initial cluster centers $M_1^{(0)},\ M_2^{(0)}, ..., M_k^{(0)}$ (e.g., the first k samples of the sample set). Set $l = 0$;

- Step 2. Assign each of the samples $\{X_i,\ i = 1, ..., N\}$ to one of the clusters according to the distance between the sample and the center of the cluster:

  $$X \sim \omega_j \quad if \quad D_L(X, M_j^{(l)}) = min\ \{\ D_L(X, M_i^{(l)}), \quad i = 1, \cdots, k\}$$

  where ω_j denotes the ith cluster of samples whose center is $M_j^{(l)}$ at the lth iteration;

- Step 3. Update the cluster centers to get $M_j^{(l+1)}$

  $$M_j^{(l+1)} = \frac{1}{N_j} \sum_{X \sim \omega_j} X, \qquad (j = 1, \cdots, k)$$

  where $N_j^{(l)}$ is the number of samples currently in $\omega_j^{(l)}$ at the lth iteration, and

  $$\sum_{j=1}^{k} N_j^{(l)} = N$$

  .

  By doing so the sum of the distances from all points in ω_j to the new center is minimized (you should be able to derive it), i.e.,

  $$\sum_{X \sim \omega_j^{(l)}} D_L(X, M_j^{(l+1)}) \rightarrow min. \qquad (j = 1, \cdots, k)$$

- Step 4. Terminate if the algorithm has converged:

$$M_j^{(l+1)} = M_j^{(l)} \quad (j = 1, \cdots, k)$$

  or a preset maximum number of iterations is exceeded.

  Otherwise, $l \leftarrow l + 1$, goto Step 2.

This method is simple, but has obvious drawbacks. For example, the user has to guess the number of clusters k, which stays fixed even it may turn out later that more or fewer clusters would fit the data better.

The Isodata Algorithm

Isodata stands for *Iterative Self-Organizing Data Analysis Techniques.* This is a more sophisticated algorithm which allows the number of clusters to be automatically adjusted during the iteration by merging similar clusters and splitting clusters with large standard deviations. We first define the following parameters:

1. K = number of clusters desired;
2. I = maximum number of iterations allowed;
3. P = maximum number of pairs of cluster which can be merged;
4. Θ_N = a threshold value for minimum number of samples in each cluster can have (used for discarding clusters);
5. Θ_S = a threshold value for standard deviation (used for split operation);
6. Θ_C = a threshold value for pairwise distances (used for merge operation).

The algorithm:

- Step 1. Arbitrarily choose k (not necessarily equal to K) initial cluster centers: $M_1,\ M_2, ..., M_k$ from the data set $\{X_i,\ \ i = 1, 2, ..., N\}$.

- Step 2. Assign each of the N samples to the closest cluster center:

$$X \sim \omega_j \quad if \quad D_L(X, M_j) = max \; \{ \; D_L(X, M_i), \quad i = 1, \cdots, k)$$

- Step 3. Discard clusters with fewer than Θ_N members, i.e., if for any j, $N_j < \Theta_N$, then discard ω_j and $k \leftarrow k - 1$.

- Step 4. Update each cluster center:

$$M_j = \frac{1}{N_j} \sum_{X \sim \omega_j} X \qquad (j = 1, \cdots, k)$$

- Step 5. Compute the average distance D_j of samples in cluster ω_j from their corresponding cluster center:

$$D_j = \frac{1}{N_j} \sum_{X \sim \omega_j} D_L(X, M_j) \qquad (j = 1, \cdots, k)$$

- Step 6. Compute the overall average distance of the samples from their respective cluster centers:

$$D = \frac{1}{N} \sum_{j=1}^{k} N_j D_j$$

- Step 7. If $k \leq K/2$ (too few clusters), go to Step 8; else if $k > 2K$ (to many clusters), go to Step 11; else go to Step 14.

  (Steps 8 through 10 are for split operation, Steps 11 through 13 are for merge operation.)

- Step 8. Find the standard deviation vector $\Sigma_j = [\sigma_1^{(j)}, \cdots, \sigma_n^{(j)}]^T$ for each cluster:

$$\sigma_i^{(j)} = \sqrt{\frac{1}{N_j} \sum_{X \sim \omega_j} (x_i - m_i^{(j)})^2}, \qquad (i = 1, \cdots, n, \;\; j = 1, \cdots, k)$$

  where $m_i^{(j)}$ is the ith component of M_j and σ_i is the standard deviation of the samples in ω_j along the ith coordinate axis. N_j is the number of samples in ω_j.

- Step 9. Find the maximum component of each Σ_j and denote it by $\sigma^{(j)}_{max}$; Do this for all $j = 1, \cdots, k$.
- Step 10.

  If for any $\sigma^{(j)}_{max}$, $\quad (j = 1, \cdots, k)$, all of the following are true

  - $\sigma^{(j)}_{max} > \Theta_S$,
  - $D_j > D$,
  - $N_j > 2\Theta_N$

  then *split* M_j into two new cluster centers M_j^+ and M_j^- by adding $\pm\delta$ to the component of M_j corresponding to $\sigma^{(j)}_{max}$, where δ can be $\alpha\, \sigma^{(j)}_{max}$, for some $\alpha > 0$. Then delete M_j and let $k \leftarrow k + 1$. Goto Step 2

  else Go to Step 14.

- Step 11. Compute the pairwise distances D_{ij} between every two cluster centers:

  $$D_{ij} = D_L(M_i, M_j), \qquad (for\ \ all\ \ i \neq j)$$

  and arrange these $k(k-1)/2$ distances in ascending order.

- Step 12. Find no more than P smallest D_{ij}'s which are also smaller than Θ_C and keep them in ascending order:

  $$D_{i_1 j_1} \leq D_{i_2 j_2} \leq \cdots \leq D_{i_P j_P}$$

- Step 13. Perform *pairwise merge*: for $l = 1, \cdots, P$, do the following:

  If neither of M_{i_l} and M_{j_l} has been used in this iteration,

  Then merge them to form a new center:

  $$M = \frac{1}{N_{i_l} + N_{j_l}} [N_{i_l} M_{i_l} + N_{j_l} M_{j_l}]$$

  Delete M_{i_l} and M_{j_l}, and let $k \leftarrow k - 1$.

  Go to Step 2.

- Step 14. Terminate if maximum number of iterations I is reached. Otherwise go to Step 2.

The Isodata algorithm is more flexible than the K-mean method. But the user has to choose empirically many more parameters listed previously.

The Tree Classifiers

When both the number of classes c and the number of features n are large, the feature selection and classification discussed before encounter difficulties because

- feature selection is no longer effective as it is difficult to find m features from n which are suitable for separating all the c classes (some features may be good from some classes but not good for others).
- classification is costly as a large number of features are necessary.

The solution is to do classification in several steps implemented as a *tree classifier*. One method to design the tree classifier is the bottom-up merge algorithm described in the following steps, which is considered as the training process.

1. From the training samples of each class $\omega_i \quad (i = 1, \cdots, c)$, estimate the mean and covariance:
$$M_i = \frac{1}{N_i} \sum_{X \sim \omega_i} X$$
and
$$\Sigma_i = \frac{1}{N_i} \sum_{X \sim \omega_i} (X - M_i)(X - M_i)^T$$

2. Compute Bhattacharrya distances for every pair of different classes ($c(c-1)/2$ of them in total):
$$D_{ij} = \frac{1}{4}(M_i - M_j)^T \left[\frac{\Sigma_i + \Sigma_j}{2}\right]^{-1} (M_i - M_j) + log \left[\frac{\left|\frac{\Sigma_i + \Sigma_j}{2}\right|}{(|\Sigma_i| \ |\Sigma_j|)^{1/2}}\right]$$
$$for \ \ all \ \ i \neq j$$

3. Merge the two classes with the smallest D_{ij} to form a new class: $\omega_i \cup \omega_j = \omega_n$ and compute its mean and covariance:
$$M_n = \frac{1}{N_i + N_j}[N_i M_i + N_j M_j]$$

and

$$\Sigma_n = \frac{1}{N_i + N_j}[N_i(\Sigma_i + (M_i - M_n)(M_i - M_n)^T) + N_j(\Sigma_j + (M_j - M_n)(M_j - M_n)^T)]$$

Delete the old classes ω_i and ω_j.

4. Compute the distance between the new class ω_n and all other classes (excluding ω_i and ω_j).

5. Repeat the above steps until eventually all classes are merged into one and a binary tree structure is thus obtained.

6. At each node of the tree build a 2-class classifier to be used to classify a sample into one of the two children G_l and G_r representing the two groups of classes. According to the classification method used, we find the discriminant functions $D_l(X)$ and $D_r(X)$.

7. At each node of the tree adaptively select features that are best for separating the two groups of classes G_l and G_r. Any feature selection method can be used here, such as directly choosing m from n features using between-class distance (Bhattacharyya distance) as the criterion, or feature selection using some orthogonal transform (KLT, DFT, WHT, etc.). Only a small number of selected features may be needed as here only two groups of classes need to be distinguished.

After the classifier is built and trained, the classification is carried out in the following manner:

A testing sample X of unknown class enters the classifier at the root of the tree and is classified to either the left or the right child of the node according to

$$X \sim \begin{cases} G_l & if \;\; D_l(X) > D_r(X) \\ G_r & if \;\; D_l(X) < D_r(X) \end{cases}$$

This process is repeated recursively at the child node (either G_l or G_r) and its child and so on, until eventually X reaches a leaf node corresponding to a single class, to which the sample X is therefore classified.