

Classification



Ying-Chao Hung

Department of Statistics

National Chengchi University

Email: hungy@nccu.edu.tw

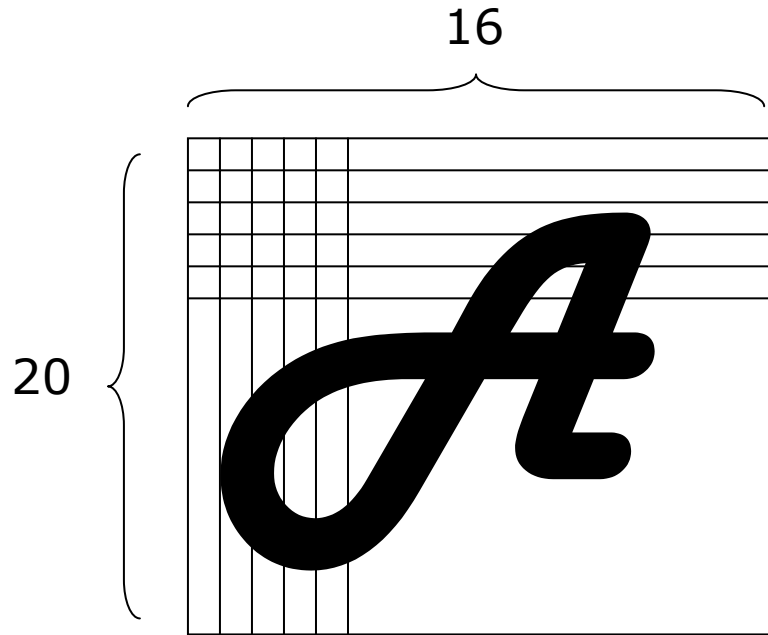
Part I. An Overview



The following material is mainly based on the lecture notes of “Applied Multivariate Analysis” by Professor George Michailidis from the Department of Statistics, University of Michigan, Ann arbor.

Handwritten Letter and Digit Recognition

■ Example:



■ There are 20×16 pixels, each has 8-bit (256) possible intensities.

Data

<u>Y</u>	<u>X_1</u>	<u>X_2</u>	\cdots	<u>X_{320}</u>
A	0	1		1
A	0	0		0
A	15	23		0
B	168	117		5
B	255	0		0
C	2	0		0
C	59	16		0
.	.	.		
.	.	.		
.	.	.		

Task and Algorithm

□ Task:

To “predict” the identity of each image from the set
 $\{ A, B, C, \dots, Z, 0, 1, \dots, 9 \}$

□ Algorithm:

High degree of accuracy → could be used as part of an automated hand-writing recognition system.

□ Supervised Learning Problem:

Also known as **classification**, **discriminant analysis**, etc.

Basic Statistical Decision Theory

Setup:

- ❑ Certain objects are to be classified from one of a fixed number of classes, e.g., $C_i \in \{1, 2, \dots, K\}$
- ❑ Each object i is comprised of a vector of variables x_i (or feature vector)
- ❑ The proportion (or probability) of class $C = k$ is given by π_k , which can be known or unknown.
- ❑ Feature vectors x from class k have the density $P_k(x)$.
- ❑ To classify an object (with feature vector x) to one of $k + 2$ classes $1, 2, \dots, K, D, O$.
(D : being in doubt, O : an outlier)

Bayes Rules for Known Distributions

-- An ideal case when $P_k(x)$ and π_k are known.

- Not directly applicable in real life but will serve a guideline.
- After observing the feature vector X , a classifier (or decision rule) is denoted by

$$\hat{c}(X): \mathcal{X} \mapsto \{1, 2, \dots, K, D\}$$

- To determine the quality of the classifier, the following overall criteria are considered:
 - (1) misclassification probabilities
 - (2) doubt probabilities

Criteria of the Classifier

▣ Misclassification Probabilities:

$$pmc(k) = P(\hat{c}(X) \neq k, \hat{c}(X) \in \{1, \dots, K\} \mid C = k)$$

▣ Doubt Probabilities:

$$pd(k) = P(\hat{c}(X) = D \mid C = k)$$

➔ To penalize wrong decisions, we need a **loss function**.

Loss Function (of decisions)

Let $L(k, l)$ be the loss incurred by making decision l when the true class is $C = k$.

A reasonable choice:

$$L(k, l) = \begin{cases} 0 & \text{if } l = k \text{ (right decision)} \\ 1 & \text{if } l \neq k \text{ and } l \in \{1, \dots, K\} \text{ (wrong decision)} \\ d & \text{if } l = D \text{ (being in doubt)} \end{cases}$$

Notes:

- (1) This loss function treats all mistakes **equally serious**.
- (2) In many practical situations a **false positive** is much less serious than a **false negative**.

Risk Function

- The **risk function** for a classifier \hat{c} is the expected loss as a function of the unknown class k :

$$\begin{aligned} R(\hat{c}, k) &= E[L(k, \hat{c}(X)) | C = k] \\ &= \sum_{l=1}^K L(k, l) P(\hat{c}(X) = l | C = k) + L(k, D) P(\hat{c}(X) = D | C = k) \\ &= 1 \cdot pmc(k) + d \cdot pd(k) \end{aligned}$$

- The **total risk** is the total expected loss (since k unknown):

$$\begin{aligned} R(\hat{c}) &= E[R(\hat{c}, C)] \\ &= \sum_{k=1}^K \pi_k \cdot pmc(k) + d \cdot \sum_{k=1}^K \pi_k \cdot pd(k) \end{aligned}$$

The Best Decision Rule

- Introduce the posterior probability of class k given $X = x$:

$$p(k | x) = P(C = k | X = x) = \frac{\pi_k \cdot P_k(x)}{\sum_{l=1}^K \pi_l \cdot P_l(x)}$$

- **Proposition**: The decision rule (classifier) which minimizes the total risk under the proposed loss function is:

$$c^*(x) = \begin{cases} k & \text{if } p(k | x) = \max_{l=1, \dots, K} p(l | x) \text{ and } > 1 - d. \\ D & \text{if each } p(k | x) \leq 1 - d \end{cases}$$

Some Remarks

- The optimal classifier is known as the **Bayes rule**, its risk is called the **Bayes risk** (or Bayes error).
- When two or more classes attain the maximal $P(k|x)$, the tie can be broken arbitrarily.
- The Bayes risk is the best value one can achieve if π_k and $P_k(x)$ are known.
This provides a benchmark for other procedures.
- The classification problem can be viewed as a regression problem with a **qualitative variable** as the dependent variable.

Classification in Practice

The posterior probability $P(k|x)$ plays a central role in determining the best classifier.

However, $P(k|x)$ are in general **unknown**.

Thus, we have to estimate $P(k|x)$ from the observed data.

Two types of methods to estimate $P(k|x)$:

- ❑ Parametric method: fixed model, determined by a set of parameters
- ❑ Non-parametric method: data driven, more flexible

Example (Parametric Method)

- π_k : known, but $P_k(x)$: unknown.
- Suppose we believe that the class densities are **normally distributed**, then $P_k(x)$ is denoted by $P_k(x; \theta)$ where θ corresponds to the **mean** and **variance**.
- Using the **training data**, we can come up with an estimate $\hat{\theta}$.
- The most popular choice for $\hat{\theta}$ is the Maximal Likelihood Estimator (MLE).
- Once we obtain the MLE $\hat{\theta}$, we can calculate the approximate posterior probability using:

$$\hat{p}(k | x) = \frac{\pi_k \cdot P_k(x; \hat{\theta})}{\sum_{l=1}^K \pi_l \cdot P_l(x; \hat{\theta})}$$

Example (Parametric Method)

- The resulting decision rule is known as a **plug-in classifier**, which is widely used by some classification methods.
- To validate if the class densities are normally distributed, an easy approach is to check out the **quantile-quantile plot (Q-Q plot)** of each standardized variable within the same class.

Non-parametric Methods for Density Estimation

- The most popular choice in low dimensional spaces is the “**kernel method**” (Diggle 1985, Silverman 1986, Jones 1990):

$$\hat{P}_k(x) = \frac{1}{n_k} \sum_{i=1}^{n_k} \frac{1}{h} K\left(\frac{x - x_i}{h}\right)$$

where K is a positive kernel function with $\int K(x)dx = 1$.

Note: The kernel method might encounter the problem of “sparseness” in high-dimensional spaces.

Kernel Density Estimation for $P_k(x)$

- Remember a pdf at some point x can be viewed as a limiting probability

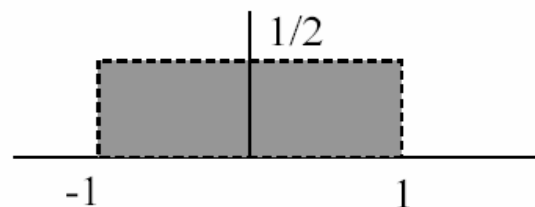
$$p(x) = \lim_{h \rightarrow 0} \frac{1}{2h} P(x-h \leq X \leq x+h)$$

- An estimator is then

$$\hat{p}_k(x) = \frac{1}{2h} \cdot \frac{\text{\# of objects in } [x-h, x+h]}{n_k} = \frac{1}{n_k} \cdot \sum_{i=1}^{n_k} \frac{1}{h} \cdot w\left(\frac{x-x_i}{h}\right)$$

if the **weight function** w is defined as

$$w(s) = \begin{cases} 1/2 & \text{if } |s| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$



Kernel Density Estimation for $P_k(x)$

- ▣ Replace $w(\cdot)$ with a general kernel function $K(\cdot)$, then

$$\hat{p}_k(x) = \frac{1}{n_k} \cdot \sum_{i=1}^{n_k} \frac{1}{h} \cdot K\left(\frac{x - x_i}{h}\right) \quad \text{where}$$

h : is called the **bandwidth**

K : is a bounded function satisfying that $\int K(x)dx = 1$

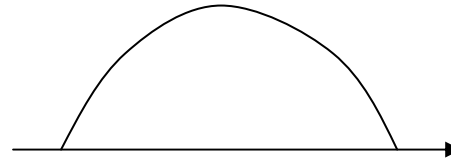
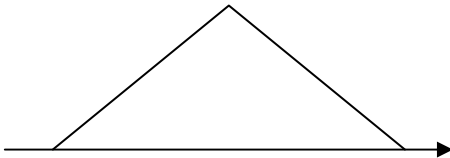
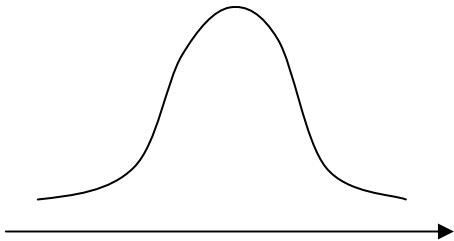
- ▣ The choice of h controls the **smoothness** of the estimate.
One way to choose the best h is using CV (discussed later).

Silverman (1986) gave a suggestion of choosing h , which minimizes the *asymptotic mean squared error (AMSE)* for Gaussian kernels.

Choice of Kernel Functions

□ Possible choices for K are:

Gaussian, rectangular, triangular, Epanechnikov, etc.



Estimation of π_k

- The most popular choice is to use the **observed proportion of class k** in the training data, say

$$\hat{\pi}_k = \frac{\sum_{i=1}^N I\{\text{object } i \text{ belongs to class } k\}}{N}$$

for each $k = 1, \dots, K$.

Performance Assessment in Practice

□ Apparent Error Rate:

The error rate of the decision rule \hat{c} when applied to the **training data**.

Theoretically, this can be reduced to **zero** (e.g. a saturated regression model)

□ True Error Rate:

The error rate of the decision rule \hat{c} when applied to new data (the new data set is called a **test data set**).

True Error Rate

- ❑ It is more important to estimate the true error rate (remember our goal is to predict the class for new data).

Problem: Need to get a new data set → Waste data !!
(since we can then construct a better classifier)

Question: How do we estimate the true error rate without requiring a new data set ?

Cross-Validation (CV)

The following procedure, called **cross-validation**, can be used to estimate the true error rate.

Step 1: Hold out a fraction (say, 5-10% or even one observation) of the training data.

Step 2: Use the remaining portion of the training data to develop a decision rule.

Step 3: Apply the rule to the “holdout sample” and estimate the error rate.

Step 4: Now leave out a different fraction and repeat Step 2 and 3.

➔ The estimated true error rate will be the average of the error rates obtained in Step 3.

Some Remarks

- The CV error rate tends to be closer to the true error rate than the apparent error rate.
- The leave-one-out version of CV is most popular.
However, it requires a large amount of computation for a large data set.

Part II. Specific Methods of Classification



In this part we will introduce five different
classification methods:

- (i) LDA (ii) QDA (iii) Nearest Neighbor
- (iV) Logistic discrimination
- (v) Classification Tree

Linear Discriminant Analysis (LDA)

- ❑ The oldest method by Fisher (1936), also called Fisher's LDA.
- ❑ Primary purpose: separate groups.
- ❑ The training data set : T , the feature/attribute vector : X

Idea: Taking linear combinations $a'X$ of the attributes in T so that different classes can be best distinguished.

Algorithm of LDA

Step 1: Calculate the mean vector \bar{x}_k for each group (class)
 $k = 1, \dots, K$ in T .

Step 2: Calculate the overall mean \bar{x} of T .

Step 3: Calculate the **between-group variance matrix**

$$B = \sum_{k=1}^K n_k (\bar{x}_k - \bar{x})(\bar{x}_k - \bar{x})'$$

where n_k is the # of objects in group (class) k .

Step 4: Calculate the **within-group variance matrix**

$$W = \sum_{k=1}^K (n_k - 1) S_k$$

where S_k is the covariance matrix of group k .

Algorithm of LDA

Step 5: Find the optimal values a by maximizing the ratio

$$\max_a \frac{a' B a}{a' W a}$$

Q: What is the intuition in Step 5 ?

- The solution a^* corresponds to the **eigenvectors of** $W^{-1}B$.
(generalized eigenvalue problem)
- The number of possible solutions is $\min(K-1, m)$, where K is the # of groups and m is the # of attributes.

Classifying New Objects

- The **first linear discriminant** $a_1'X$ corresponds to the first eigenvector, **the second linear discriminant** $a_2'X$ corresponds to the second eigenvector, so on and so forth.
- **Question:** How can we use the derived linear discriminants to classify new objects ?
- **Answer:** Let x_0 be the attribute vector of a new object we want to classify. The classification rule is to **allocate the object to class k** if

$$\sum_{j=1}^r [a_j'(x_0 - \bar{x}_k)]^2 \leq \sum_{j=1}^r [a_j'(x_0 - \bar{x}_i)]^2 \quad \text{for all } i \neq k,$$

where the number of discriminants $r \leq \min\{k-1, m\}$.

The LDA Solution

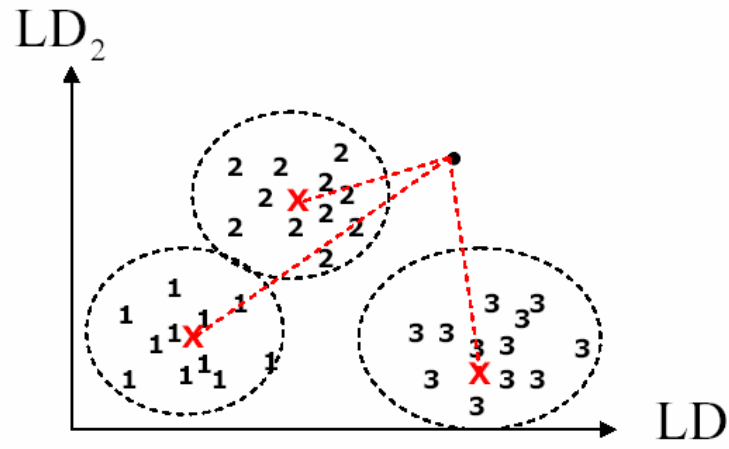
- It is noted that the classification rule can be written as

$$\hat{c}(x_0) = \arg \min_k \left\{ \sum_{j=1}^r [a_j' (x_0 - \bar{x}_k)]^2 \right\}$$

That is, if the new object is closest to the center of class k , say, closest to \bar{x}_k in the r -dimensional LD space, then we allocate it to class k .

- Example:**

(choose $r = 2$)



Example: Iris data

Apply LDA in R we get the following result:

Prior probabilities of groups:

```
      SETOSA VERSICOLOR VIRGINICA  
0.3333333  0.3333333 0.3333333
```

Group means:

	petallen	petalwid	sepallen	sepalwid
SETOSA	14.62	2.46	50.06	34.28
VERSICOLOR	42.60	13.26	59.36	27.70
VIRGINICA	55.52	20.26	65.88	29.74

Coefficients of linear discriminants:

	LD1	LD2
petallen	0.22012117	-0.093192121
petalwid	0.28104603	0.283918785
sepallen	-0.08293776	0.002410215
sepalwid	-0.15344731	0.216452123

Proportion of trace:

LD1	LD2
0.9912	0.0088

Example: Iris data

Apply LDA in R we get the following result:

Confusion Matrix:

	Setosa	Versicolor	Virginica
Setosa	50	0	0
Versicolor	0	48	2
Virginica	0	1	49

TABLE 1. Confusion matrix for the iris data based on linear discrimination

➔ Misclassification error (apparent error) rate = $3/150 = 2\%$.

Q: How about true error rate? (see Lab)

Example: Iris data

The output on the LD space:

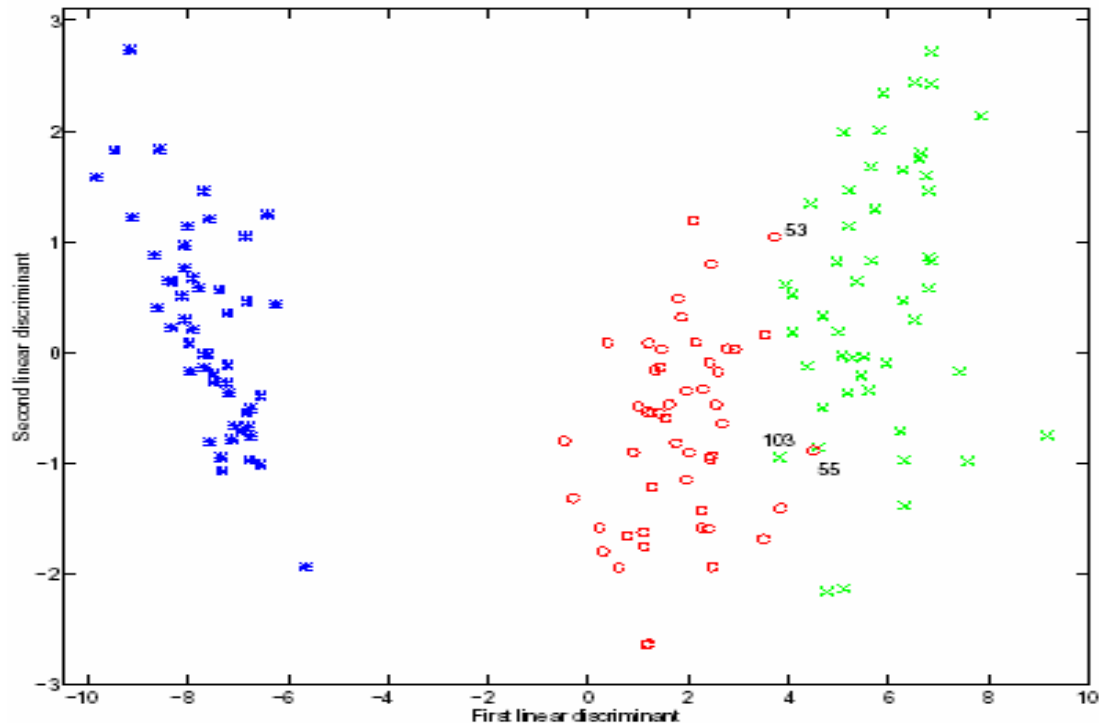


FIGURE 4. Linear discriminant analysis of the iris data (blue *=setosa, green x=virginica, red o=versicolor)

Quadratic Discriminant Analysis (QDA)

- A parametric method based on Normality assumption.

Assumptions: Objects in class k are assumed to be normally distributed with mean vector μ_k and covariance matrix Σ_k .

Decision Rule: Remember the best decision rule is to

$$\begin{aligned} & \text{maximize } p(k | x) = \text{maximize } \{\pi_k \cdot P_k(x)\} \\ & = \text{maximize } \{2 \log \pi_k + 2 \log P_k(x)\} \\ & = \text{minimize } \{-2 \log \pi_k - 2 \log P_k(x)\} \\ & = \text{minimize } \left\{ -2 \log \pi_k - 2 \log \left(\underbrace{\frac{1}{(2\pi)^{m/2} |\Sigma_k|^{1/2}} \exp\left[-1/2(x - \mu_k)' \Sigma_k^{-1} (x - \mu_k)\right]}_{\text{Multivariate normal pdf}} \right) \right\} \end{aligned}$$

Decision Rule (continued)

$$\begin{aligned} &= \text{minimize } \left\{ -2 \log \pi_k + \log |\Sigma_k| + (x - \mu_k)' \Sigma_k^{-1} (x - \mu_k) \right\} \\ &= \text{minimize } \{ Q_k \} \end{aligned}$$

Squared Mahalanobis distance
of x to the center of class k



- For the 2-class problem, if $Q_1 < Q_2$ for some x , then x is classified to class 1.
- The boundary between the two classes is given by a quadratic function of the attributes (x).
- ➔ The method is known as Quadratic Discriminant Analysis.

Some Remarks

- QDA is also applied to the general K -class problem.
- For the 2-class problem with equal covariance matrix (i.e. $\Sigma_1 = \Sigma_2$), the decision rule reduces to Fisher's LDA.
- Usually we estimate μ_k and Σ_k from the training data T .
 - ➔ Using \bar{x}_k and S_k .

Nearest Neighbor Method (NN)

- A simple case of the non-parametric kernel methods.
 - ➔ Choose kernel K to be constant over the nearest r objects and zero elsewhere.
- Can estimate $P(k | x)$ using the proportions of the classes among the nearest r objects.

$$\hat{P}(k | x) = \frac{\text{\# of class } k \text{ objects}}{r} = \frac{n_k}{r}$$

NN Procedure

Step 1: Choose r .

Step 2: Determine the r nearest neighbors (usually in Euclidean distance) of the object x in T .

Step 3: Classify the object to the class that **contains the majority of the r nearest neighbors** (i.e. the maximum of n_k / r)

Q: How to choose r ?

A: Usually chosen by cross-validation which corresponds to the smallest “true error rate”.

Note: If $n_i / r = n_j / r \rightarrow$ randomly select one class.

Example: Glass data (6 classes)

The confusion matrices for NN with $r = 1$ and $r = 3$:

	A	B	C	D	E	F
A	70	0	0	0	0	0
B	0	74	0	0	0	0
C	0	0	17	0	0	0
D	0	0	0	13	0	0
E	0	0	0	0	9	0
F	0	0	0	0	0	29

TABLE 5. Confusion matrix for the glass data based on the $r = 1$ nearest neighbors

	A	B	C	D	E	F
A	64	3	3	0	0	0
B	10	62	1	2	1	0
C	7	0	10	0	0	0
D	0	0	0	12	0	1
E	0	2	0	0	6	1
F	0	1	1	1	1	25

TABLE 4. Confusion matrix for the glass data based on the 3 nearest neighbors

Logistic Discrimination

- Assume that $P_k(x)$ are **normal** and we have **common covariance matrix** Σ_k .
- If we compare the posterior probability of class k with class 1, then

$$\begin{aligned} 2 \log \frac{p(k | x)}{p(1 | x)} &= (x - \mu_1)' \Sigma^{-1} (x - \mu_1) - (x - \mu_k)' \Sigma^{-1} (x - \mu_k) + 2 \log \frac{\pi_k}{\pi_1} \\ &= 2(\mu_k - \mu_1)' \Sigma^{-1} x - (\mu_k + \mu_1)' \Sigma^{-1} (\mu_k - \mu_1) + 2 \log \frac{\pi_k}{\pi_1} \\ &= \beta_k' x + \alpha_k \end{aligned}$$

➔ A linear function of x .

Logistic Discrimination

- A 2-class problem:

$$\log \frac{p(2|x)}{p(1|x)} = \log \frac{p(2|x)}{1 - p(2|x)} = \alpha + \beta'x \Rightarrow \text{logistic regression}$$

↘ Logit (or log-odds)

- General form:

$$\log \frac{p(k|x)}{p(1|x)} = \alpha_k + \beta_k'x$$

where α_k and β_k can be estimated by \bar{x}_k , S_k , and π_k .

Estimation of Posterior Probability

□ Since

$$\frac{\exp\{\alpha_k + \beta_k'x\}}{\sum_{i=1}^K \exp\{\alpha_i + \beta_i'x\}} = \frac{\frac{p(k|x)}{p(1|x)}}{\frac{p(1|x)}{p(1|x)} + \dots + \frac{p(K|x)}{p(1|x)}} = \frac{p(k|x)}{\underbrace{p(1|x) + \dots + p(K|x)}_1} = p(k|x)$$

$$\rightarrow \hat{p}(k|x) = \frac{\exp\{\hat{\alpha}_k + \hat{\beta}_k'x\}}{\sum_{i=1}^K \exp\{\hat{\alpha}_i + \hat{\beta}_i'x\}}$$

□ The decision rule chooses the maximal value of $\hat{p}(k|x)$.

Example: Iris and Glass data

The confusion matrices for Logistic discrimination:

	Setosa	Versicolor	Virginica
Setosa	50	0	0
Versicolor	0	49	1
Virginica	0	1	49

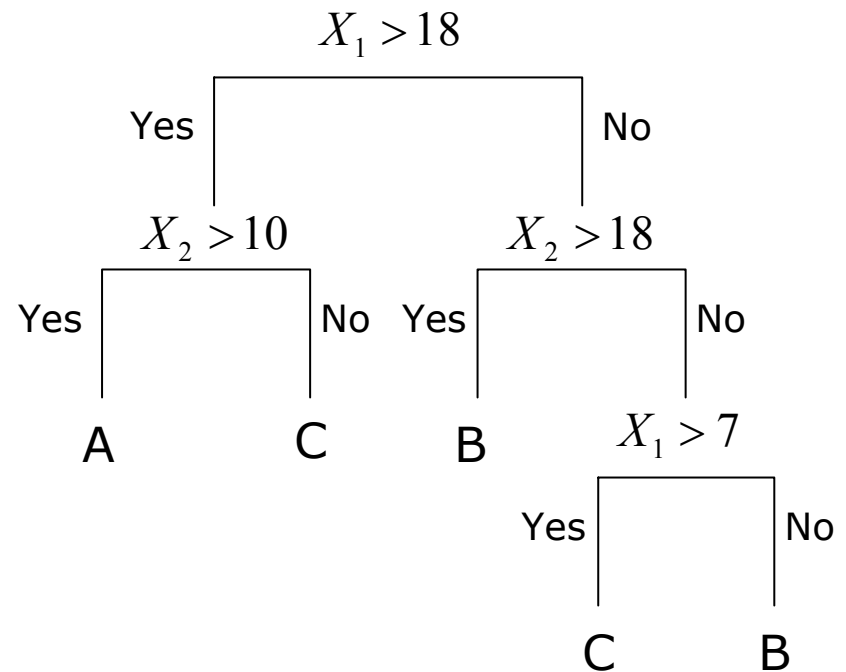
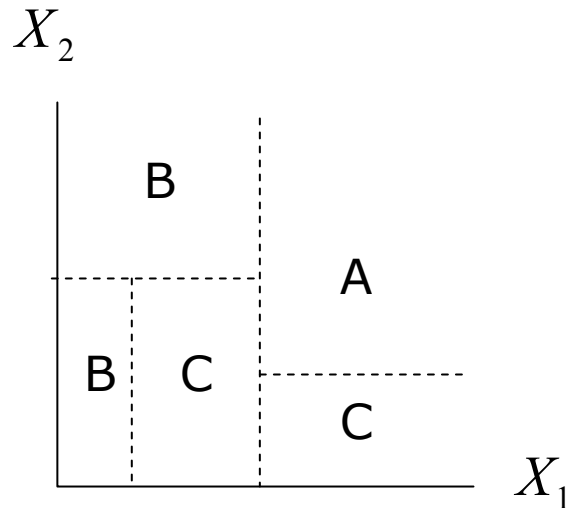
TABLE 6. Confusion matrix for the iris data based on logistic discrimination

	A	B	C	D	E	F
A	52	19	10	0	0	0
B	18	54	7	3	0	2
C	9	7	0	0	0	0
D	0	1	0	9	0	0
E	0	0	0	0	9	0
F	0	2	0	1	0	26

TABLE 7. Confusion matrix for the glass data based on logistic discrimination

Classification Tree

- The goal is to partition the sample space (attribute space) into hypercubes and assign a class to every hypercube.



Constructing a Classification Tree

Main Issue: Theoretically we can achieve 0% apparent error rate by growing a large-size tree. To avoid one object per hypercube(overfitting the model), we need to prune the tree.

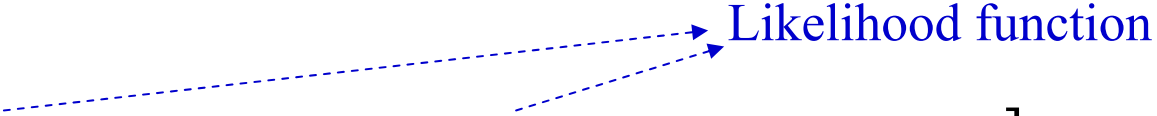
Setup:

- At each node v , denote the probability of class k by $p(k|v)$.
- Given attributes x_i , we learn the numbers of objects n_v assigned to every node v .
- Denote the numbers n_{vk} of class k at node v and it has a multinomial distribution with probability

$$p(k | v) = \frac{p(\text{class } k \text{ and node } v)}{p(v)}$$

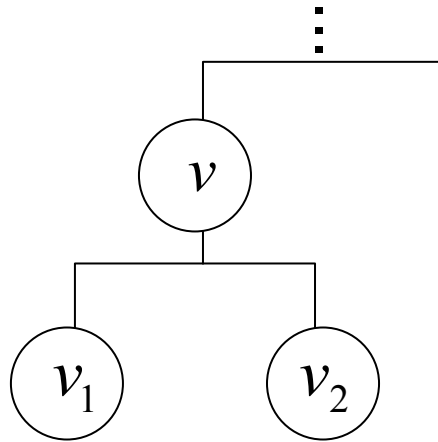
Constructing a Classification Tree

- The Maximal Likelihood Estimate of $p(k|v)$ is $\hat{p}(k | v) = \frac{n_{vk}}{n_v}$
- For a perfect model, it is clear that $n_{vk} / n_v = 1$ (only one class)
- The (scaled) deviance for node v is defined as


$$\begin{aligned} D_v &= 2[\log l(\text{perfect model}) - \log l(\text{model using MLE})] \\ &= 2\left[0 - \log\left(\prod_k \hat{p}(k | v)^{n_{vk}}\right)\right] \\ &= -2\sum_k n_{vk} \log \hat{p}(k | v) = -2\sum_k n_{vk} \cdot \log\left(\frac{n_{vk}}{n_v}\right) \end{aligned}$$

Constructing a Classification Tree

- For the whole tree, the total deviance is $D = \sum_v D_v$
- For each node v , how do we split the node into nodes v_1 and v_2 ?



➔ The split is the one that **maximizes the reduction of deviance**,
i.e., $D_v - (D_{v_1} + D_{v_2})$.

Prune the Tree

- To avoid growing a very large tree (over-fitting the training data), we need to prune the tree.

That is, when shall we stop growing the tree ?

- Define an **cost-complexity measure** (or loss function):

$$C_{\alpha}(\text{Tree}) = R(\text{Tree}) + \alpha \cdot \text{Size}(\text{Tree})$$

Misclassification rate

of terminal nodes

- The goal is to minimize $C_{\alpha}(\text{Tree})$ over all sizes of trees.

Prune the Tree

- When $\alpha \rightarrow 0$, we are interested in minimizing $R(\text{Tree})$.
 - ➔ This usually comes up with a large size.
- When $\alpha \rightarrow \infty$, we are interested in minimizing $\text{Size}(\text{Tree})$.
 - ➔ This usually comes up with a small size.

Q: How to choose the best α ?

Choosing the Best Tree

Step 1: Start with a small α and find the best size of the tree that minimizes $C_\alpha(\text{Tree})$.

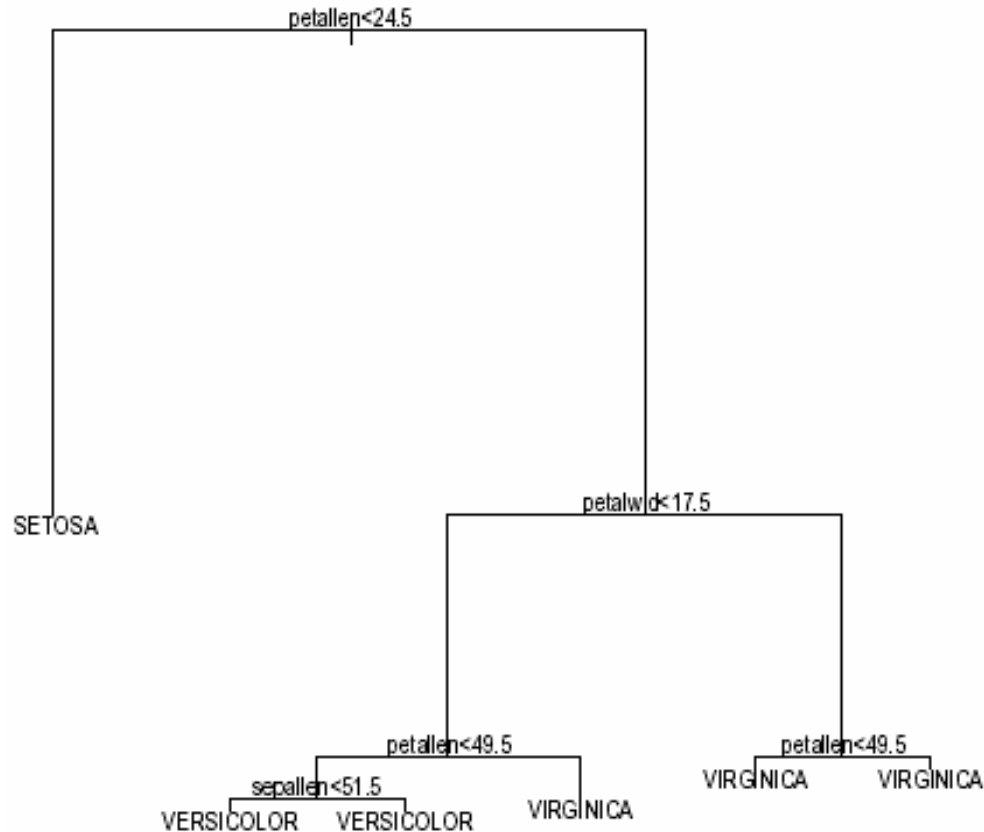
Step 2: Increase α and find the best tree for each given α .

Step 3: Choose the best α such that the overall “true error rate” is minimized (using CV).

- ❑ The 1-SE rule can be used to further reduce the tree size.
- ❑ Besides cost-complexity measure, other criteria such as Akaike's Information Criterion (AIC) can be used.

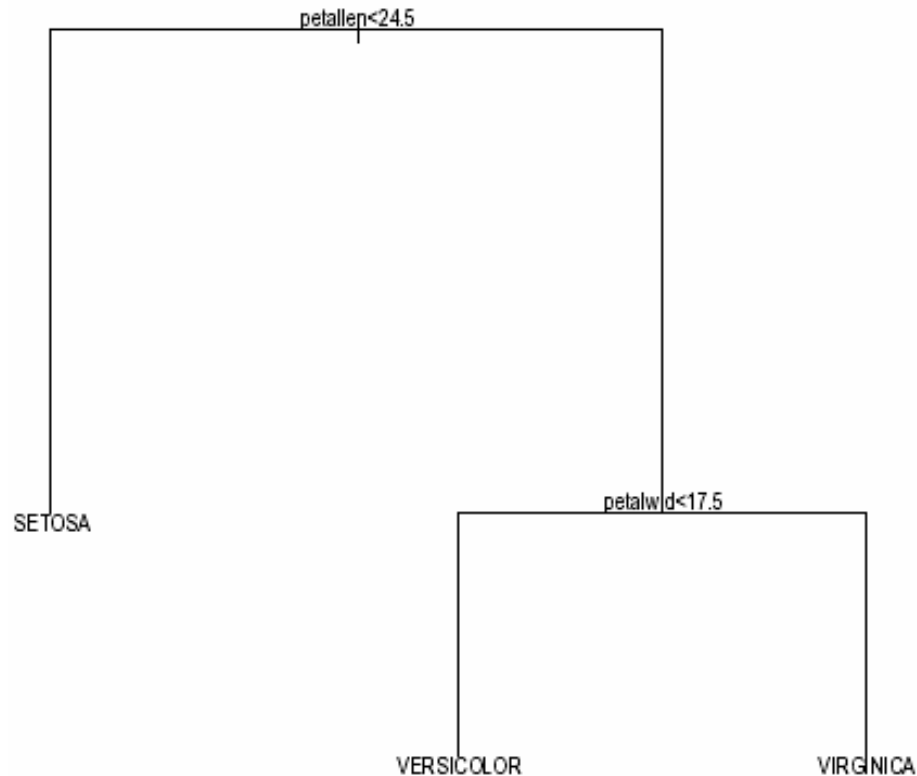
Example: Iris data

A classification tree with size = 6.



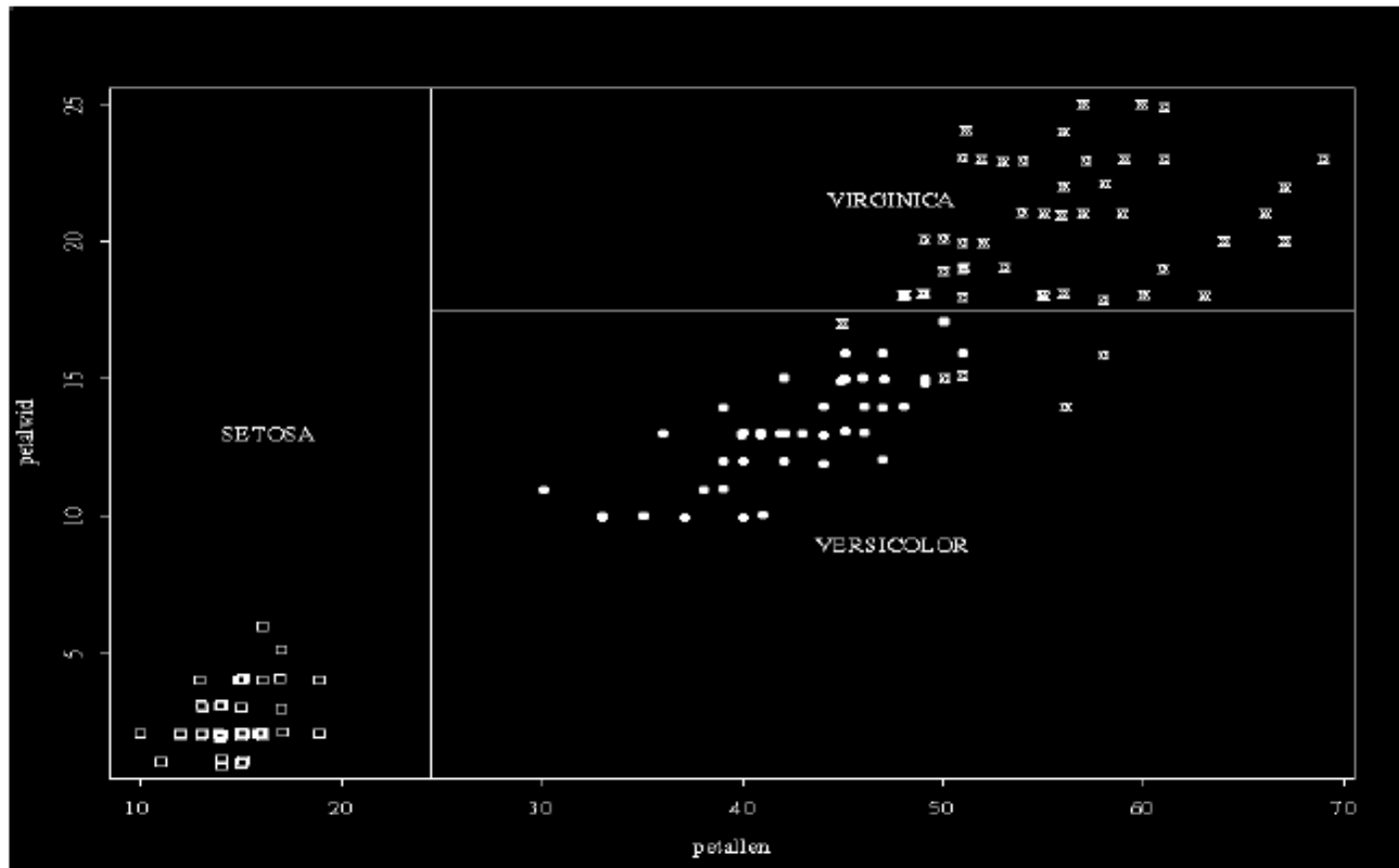
Example: Iris data

The pruned tree with size = 3, which uses **only 2 variables for partition**.



Example: Iris data

Partition of the pruned tree with size = 3.



Variable Selection

- ❑ Including “irrelevant” attributes will (i) probably have a negative effect on the performance of the classifier; (ii) cost money.
- ❑ In LDA, one can (i) check loadings; (ii) examine the changes in Wilk’s U statistic, say, $U = |W| / |W+B|$.
- ❑ The Tree method will automatically select the variables. However, one can examine the changes in deviance.
- ❑ In Logistic classification, can examine the changes in deviance.
- ❑ Non-parametric methods do not allow us to study directly the influence of particular variables.

Aggregating Classifiers - 1

□ **Bagging** [Breiman, 1996]:

Derive a **perturbed version** T^P of the same size as the training data T based on **Bootstrap replicates** (i.e. draw objects at random with replacement from T).

Suppose we decide to use LDA (or other methods) and a new object is assigned to A, A, A, B from classifiers built from 4 different perturbed versions of T .

➔ The “winning” class will be “A” by **plurality voting**.

Note: Breiman showed that this improve prediction accuracy.

Aggregating Classifiers - 2

- Tree methods are unstable (easily affected by outliers), so it benefits most from Bagging predictors.
- LDA and NN are more stable, so they benefit least.
- One can also combine the classifiers obtained from different approaches and assign the new object to the majority class.
Such a procedure can also improve the prediction accuracy.

Concluding Remarks

- ❑ Various methods have been suggested in the literature.
 - Boosting (Freund and Schapire, 1996)
 - Arcing (Breiman, 1998)
 - Error Correcting Output Coding (Diettrich & Bakiri)
 - Support Vector Machine (Vapnik, 1998)
- ❑ Classification problems in large data set
- ❑ Imbalanced data problems (i.e. the # of objects for each class are quite different)