Chapter 6 Object Recognition

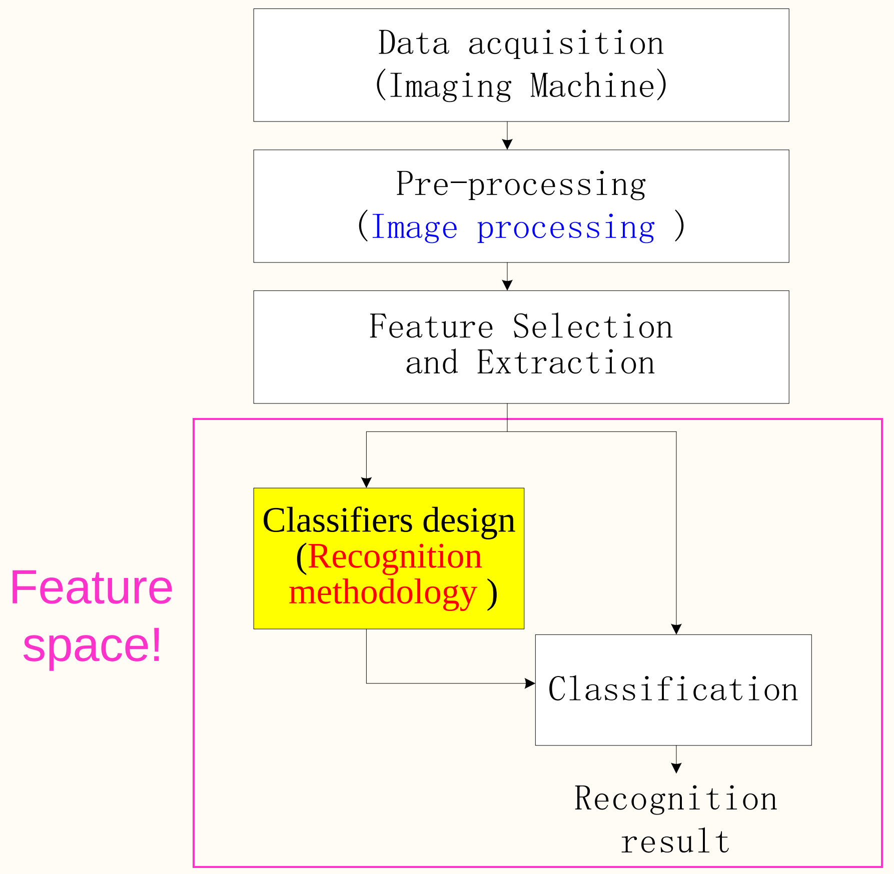
Part II   Classifier

Outline

- 6.1 Pattern Recognition System
- 6.2 Feature Selection and Extraction
- 6.3 Pattern Matching
- 6.4 Bayesian Decision Theory
- 6.5 Linear Classifiers
- 6.6 Clustering

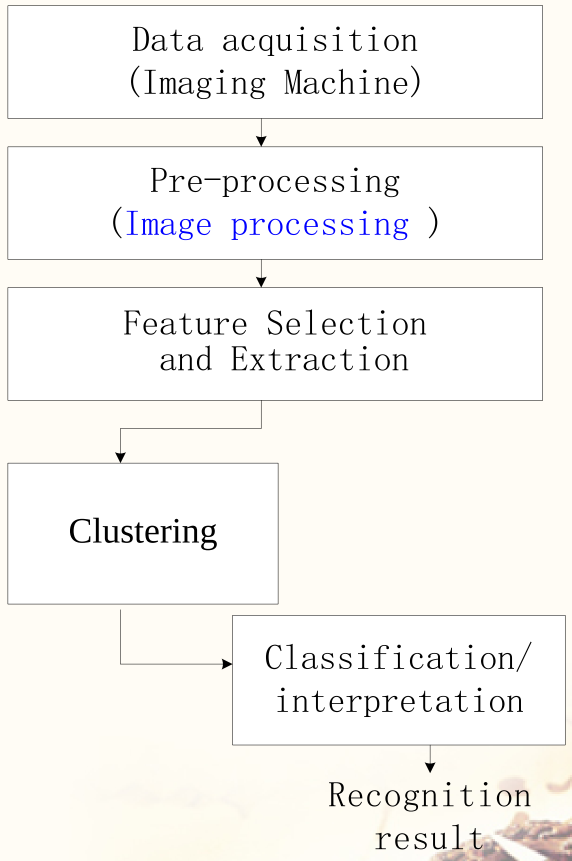
Reference:
S. Theodoridis, K. Koutroumbas. Pattern Recognition (Fourth Edition). 2004

Illustration for pattern recognition system



Data acquisition (Imaging Machine)
Pre-processing (Image processing)
Feature Selection and Extraction
Feature space!
Classifiers design (Recognition methodology)
Classification
Recognition result

Supervised pattern recognition



Data acquisition (Imaging Machine)
Pre-processing (Image processing)
Feature Selection and Extraction
Clustering
Classification/ interpretation
Recognition result

Unsupervised pattern recognition

6.4 Bayesian Decision Theory

Most Propable!

Basic Assumptions

- The decision problem is posed in probabilistic terms
- All of the relevant probability values are known

Bayes Formula

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j)P(\omega_j)}{p(x)}$$

$$p(x) = \sum_{j=1}^{2} p(x \mid \omega_j)P(\omega_j)$$

$$posterior = \frac{likelihood \times prior}{evidence}$$

Decide ω_1 if $P(\omega_1 \mid x) > P(\omega_2 \mid x)$; otherwise decide ω_2

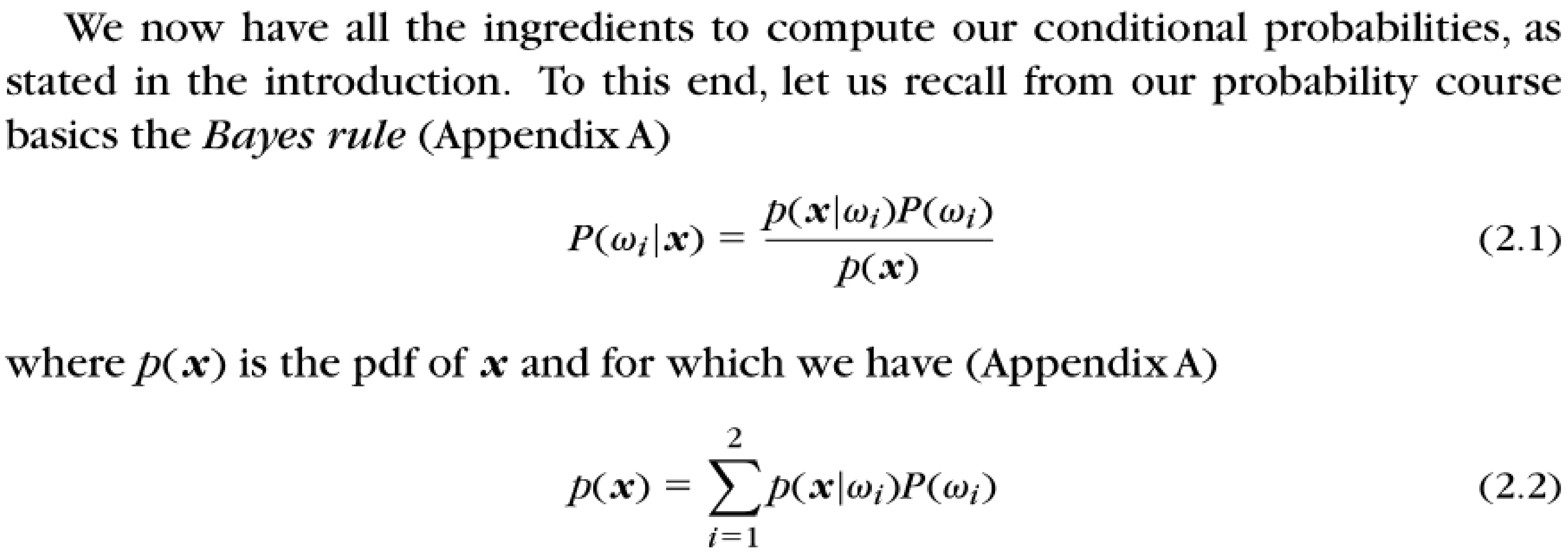
Bayes Formula

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j)P(\omega_j)}{p(x)}$$

$$p(x) = \sum_{j=1}^{2} p(x \mid \omega_j)P(\omega_j)$$

Decide ω_1 if $P(\omega_1 \mid x) > P(\omega_2 \mid x)$; otherwise decide ω_2

$$posterior = \frac{likelihood \times prior}{evidence}$$



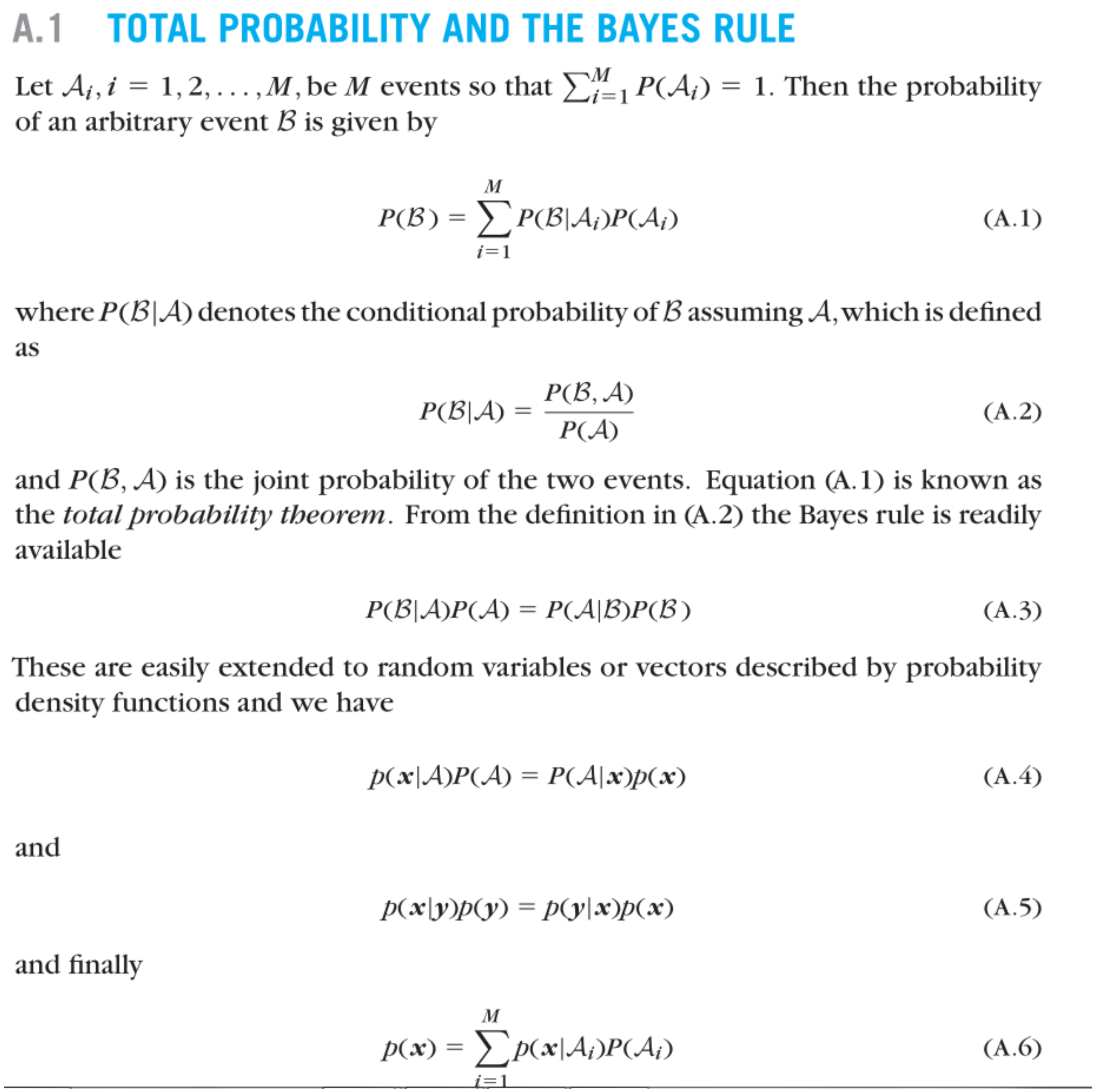
We now have all the ingredients to compute our conditional probabilities, as stated in the introduction. To this end, let us recall from our probability course basics the *Bayes rule* (Appendix A)

$$P(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_i)P(\omega_i)}{p(\boldsymbol{x})} \quad (2.1)$$

where $p(\boldsymbol{x})$ is the pdf of $\boldsymbol{x}$ and for which we have (Appendix A)

$$p(\boldsymbol{x}) = \sum_{i=1}^{2} p(\boldsymbol{x}|\omega_i)P(\omega_i) \quad (2.2)$$

Bayes Formula



A.1 TOTAL PROBABILITY AND THE BAYES RULE

Let $\mathcal{A}_i, i = 1, 2, \ldots, M$, be M events so that $\sum_{i=1}^{M} P(\mathcal{A}_i) = 1$. Then the probability of an arbitrary event $\mathcal{B}$ is given by

$$P(\mathcal{B}) = \sum_{i=1}^{M} P(\mathcal{B}|\mathcal{A}_i)P(\mathcal{A}_i) \tag{A.1}$$

where $P(\mathcal{B}|\mathcal{A})$ denotes the conditional probability of $\mathcal{B}$ assuming $\mathcal{A}$, which is defined as

$$P(\mathcal{B}|\mathcal{A}) = \frac{P(\mathcal{B}, \mathcal{A})}{P(\mathcal{A})} \tag{A.2}$$

and $P(\mathcal{B}, \mathcal{A})$ is the joint probability of the two events. Equation (A.1) is known as the *total probability theorem*. From the definition in (A.2) the Bayes rule is readily available

$$P(\mathcal{B}|\mathcal{A})P(\mathcal{A}) = P(\mathcal{A}|\mathcal{B})P(\mathcal{B}) \tag{A.3}$$

These are easily extended to random variables or vectors described by probability density functions and we have

$$p(\boldsymbol{x}|\mathcal{A})P(\mathcal{A}) = P(\mathcal{A}|\boldsymbol{x})p(\boldsymbol{x}) \tag{A.4}$$

and

$$p(\boldsymbol{x}|\boldsymbol{y})p(\boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) \tag{A.5}$$

and finally

$$p(\boldsymbol{x}) = \sum_{i=1}^{M} p(\boldsymbol{x}|\mathcal{A}_i)P(\mathcal{A}_i) \tag{A.6}$$

Comparison of posterior probability

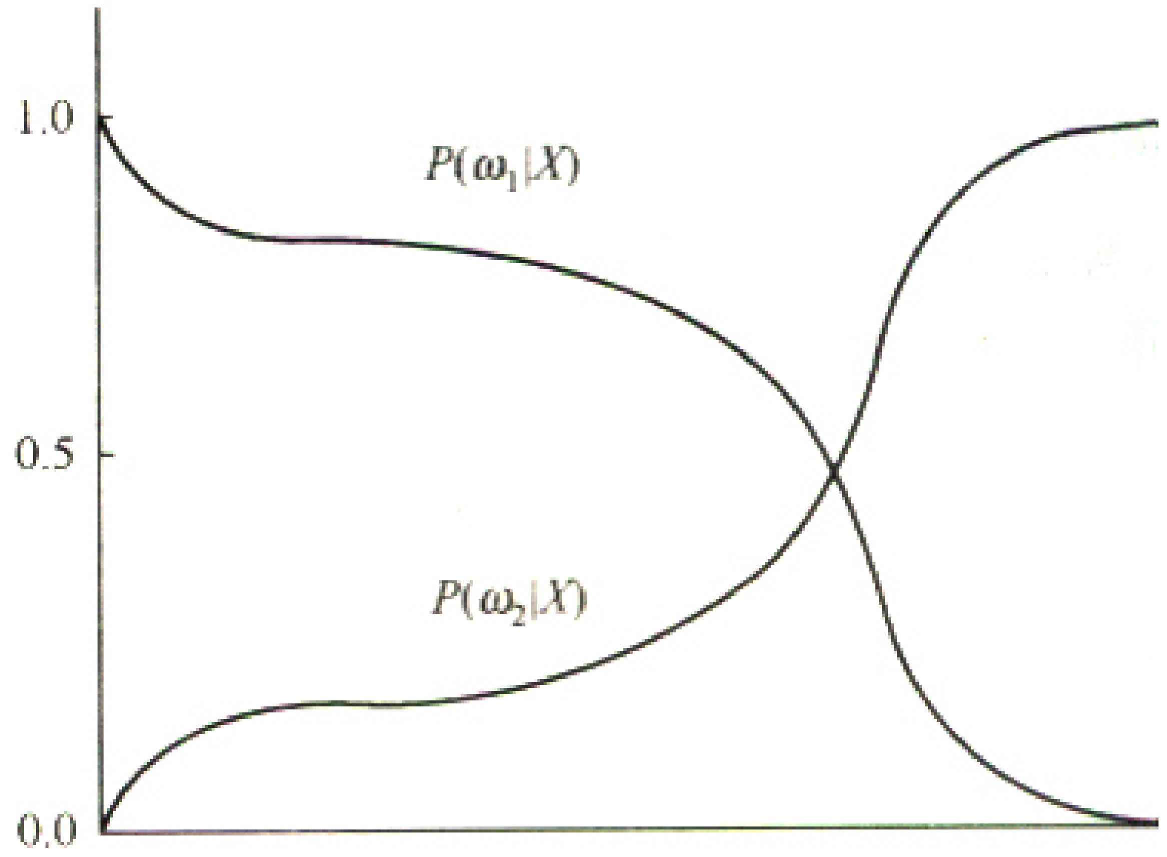


图 4-4　后验概率比较图

Bayes Decision Rule

- Probability of error

$$P(error \mid x) = \begin{cases} P(\omega_1 \mid x) \text{ if we decide } \omega_2 \\ P(\omega_2 \mid x) \text{ if we decide } \omega_1 \end{cases}$$

$$P(error) = \int_{-\infty}^{\infty} p(error, x)dx = \int_{-\infty}^{\infty} p(error \mid x)p(x)dx$$

- Bayes decision rule

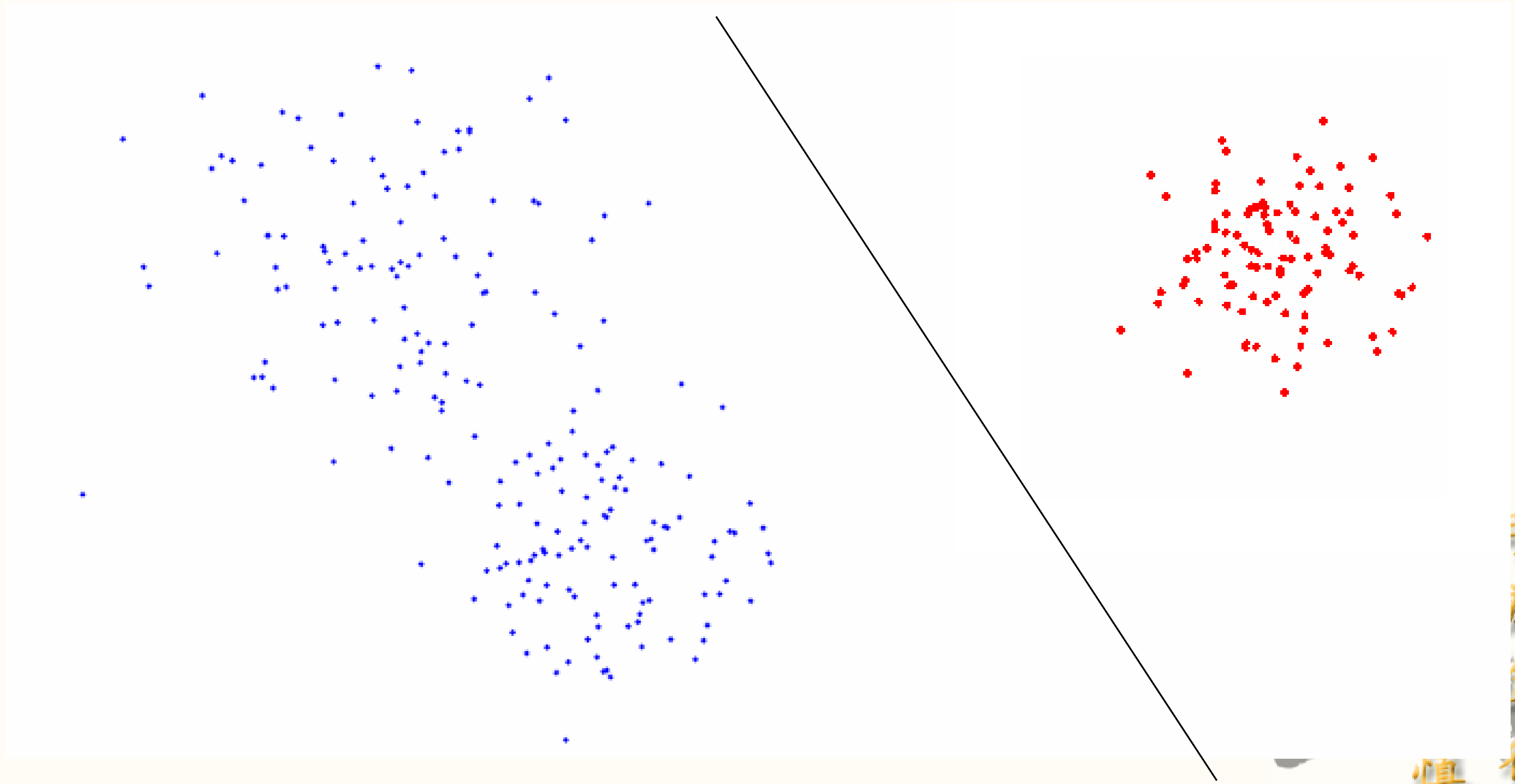
$\text{Decide } \omega_1 \text{ if } P(\omega_1 \mid x) > P(\omega_2 \mid x); \text{otherwise decide } \omega_2$

$\text{Or, decide } \omega_1 \text{ if } p(x \mid \omega_1)P(\omega_1) > p(x \mid \omega_2)P(\omega_2);$

$\text{otherwise decide } \omega_2$

6.5 Linear Classifiers

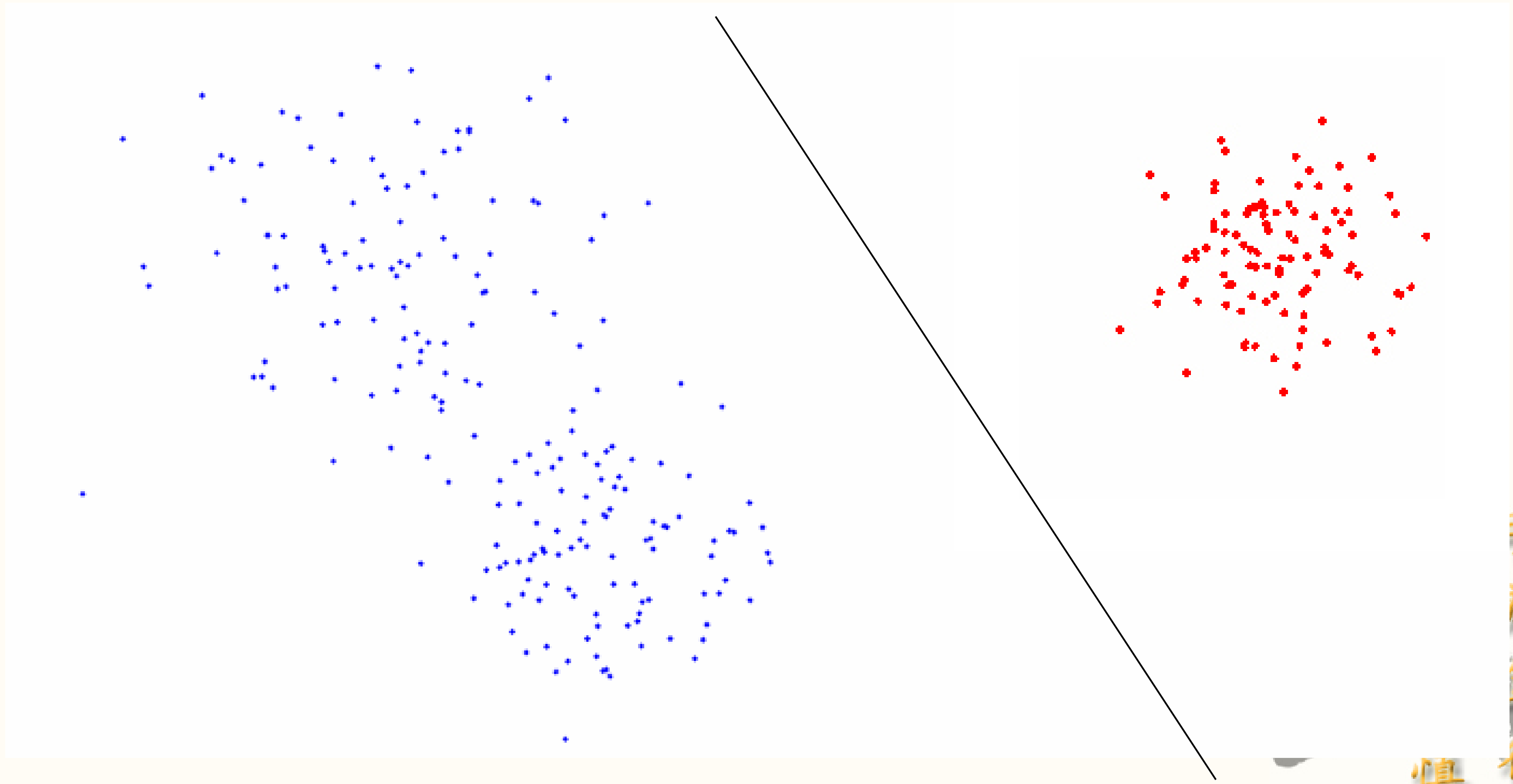
Linearly separable



Linear Discriminant Functions

- Proper forms for discriminant functions are known
- Use samples to estimate the values of parameters of the classifiers
- Not require knowledge of the forms of underlying probability distributions
- Linear in some given set of functions
- Simple, may not be optimal

Linearly separable



Linear Combination of Components: Two-Category Case

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + \mathbf{w}_0$$

Linear Machines

- Decision regions for linear machines are convex
  - Limits flexibility and accuracy
- Every decision region is singly connected
  - More suitable for which $p(\mathbf{x}|\omega_i)$ is unimodal

Augmented Vectors

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i = \sum_{i=0}^{d} w_i x_i, \quad x_0 = 1$$

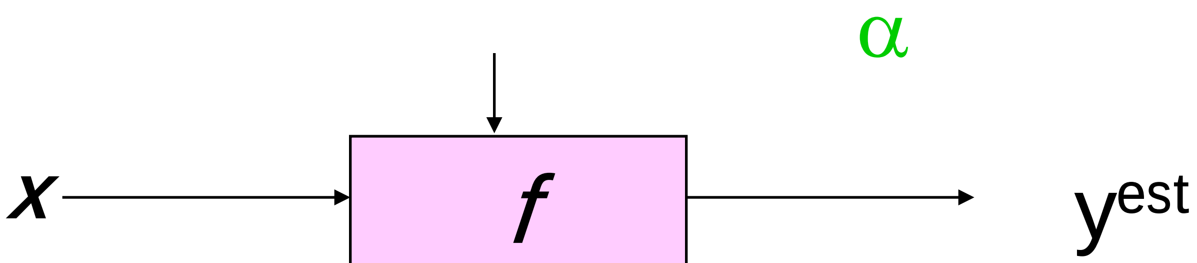
$$\mathbf{y} = \begin{bmatrix} 1 \\ x_1 \\ \text{M} \\ x_d \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} w_0 \\ w_1 \\ \text{M} \\ w_d \end{bmatrix} = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$$

$$g = WX$$

Two-Category Linearly Separable Case

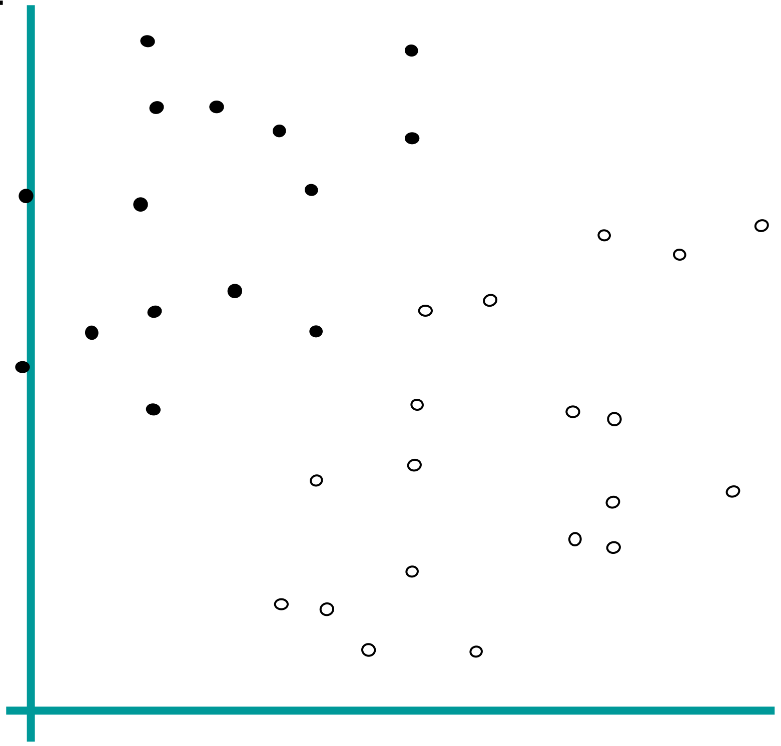
- Set of n sample: $\mathbf{y}_1, \ldots, \mathbf{y}_n$
- Labels: ω_1, ω_2
- To determine a linear discriminant function $g(\mathbf{x})=WX$
- Linearly separable
  - Weight vector that classifies all of the samples correctly
- Normalization
- Separating vector (solution vector)

Linear Classifiers



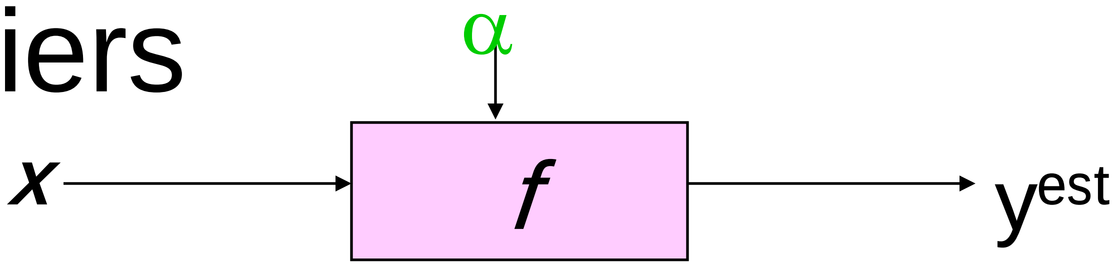
$f(\mathbf{x},\mathbf{w},b) = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x} - b)$

- ● denotes +1
- ○ denotes -1



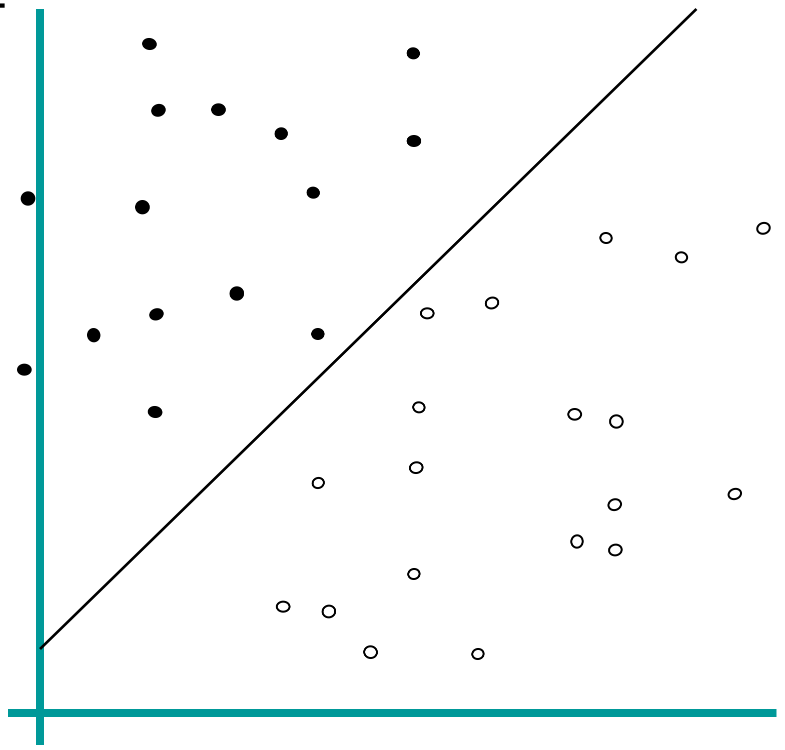
How would you classify this data?

Linear Classifiers



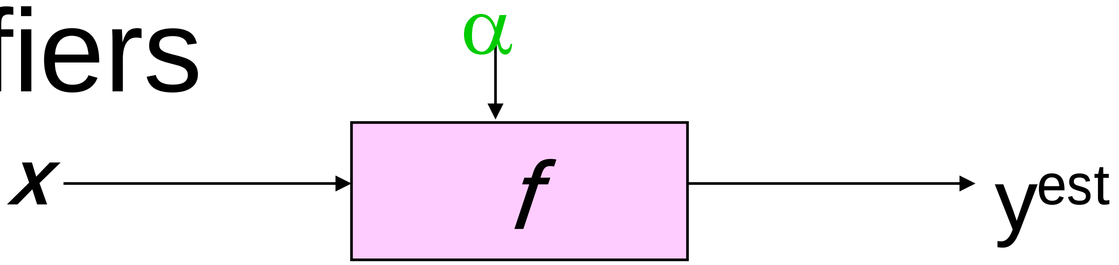
$f(x, w, b) = \text{sign}(w \cdot x - b)$

- ● denotes +1
- ○ denotes -1

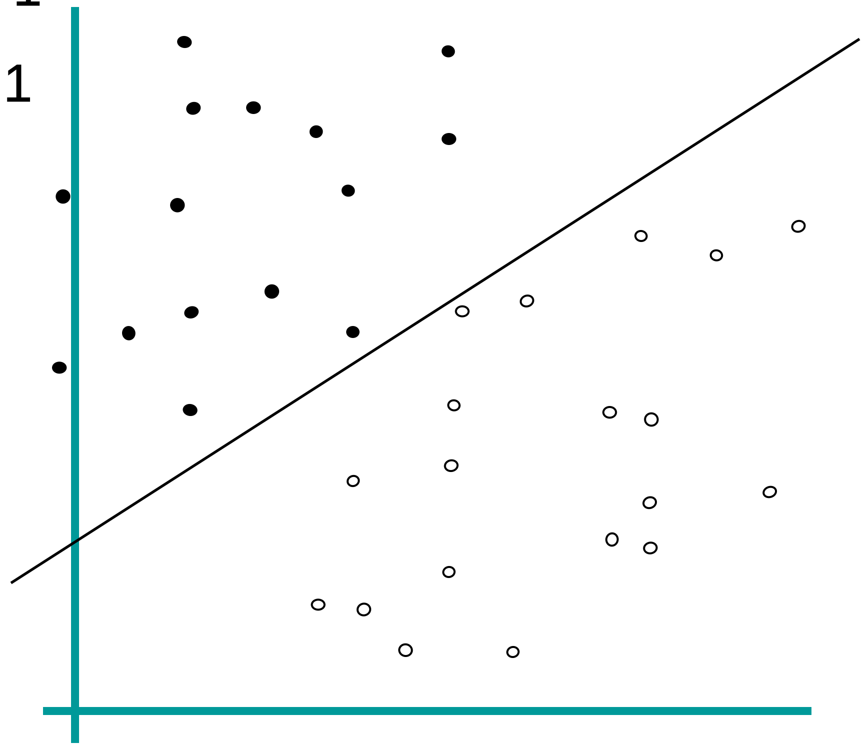


How would you classify this data?

Linear Classifiers



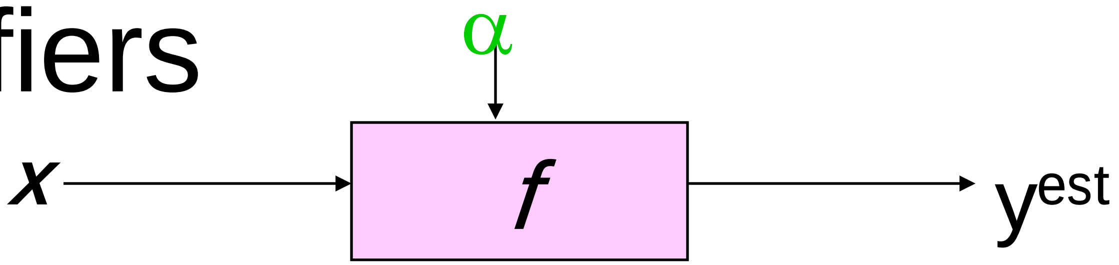
$f(x,w,b) = sign(w. x - b)$



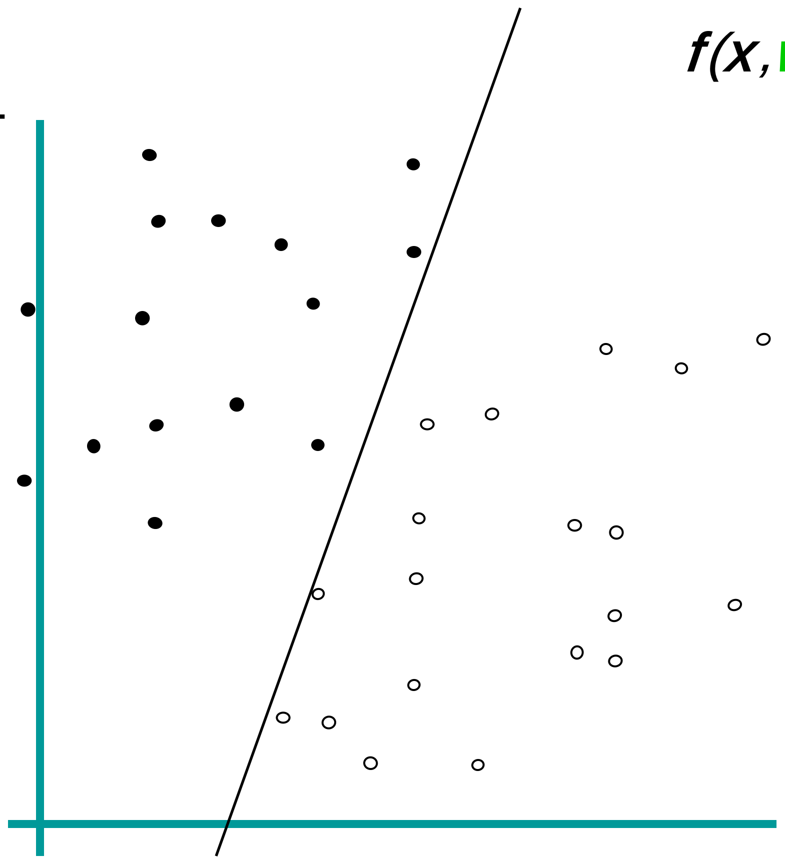
- ● denotes +1
- ○ denotes -1

How would you classify this data?

Linear Classifiers



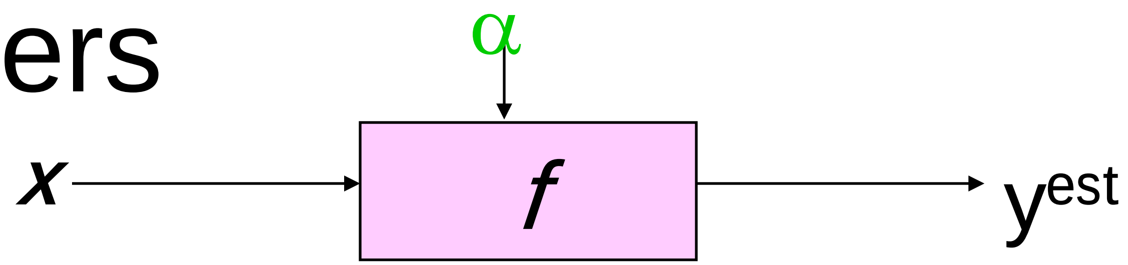
$f(x, w, b) = \text{sign}(w \cdot x - b)$



- ● denotes +1
- ○ denotes -1

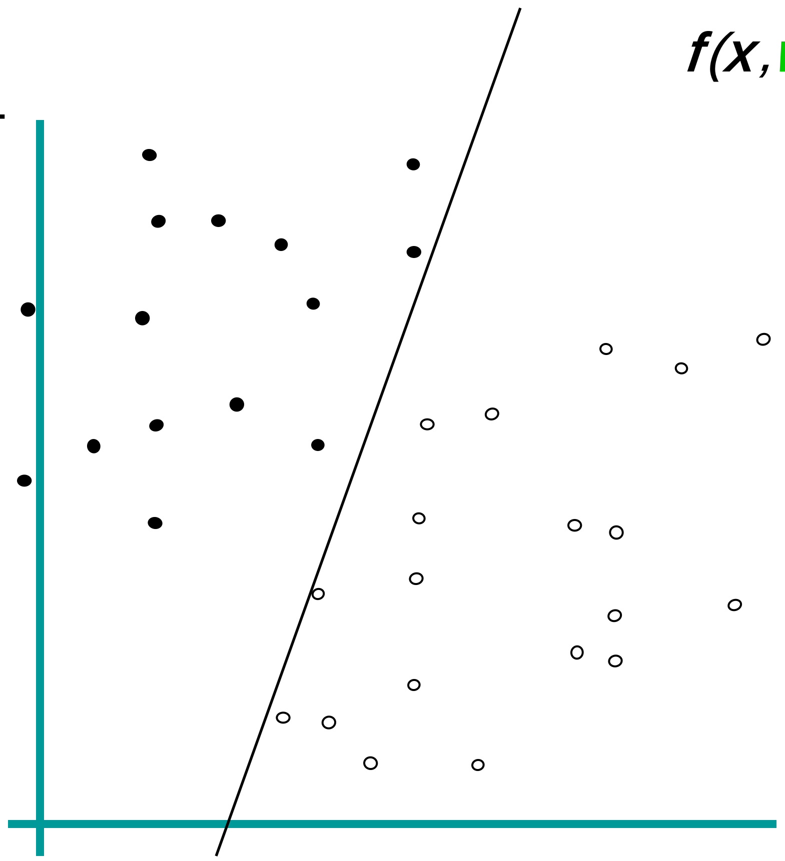
How would you classify this data?

Linear Classifiers



$f(x, w, b) = \text{sign}(w \cdot x - b)$

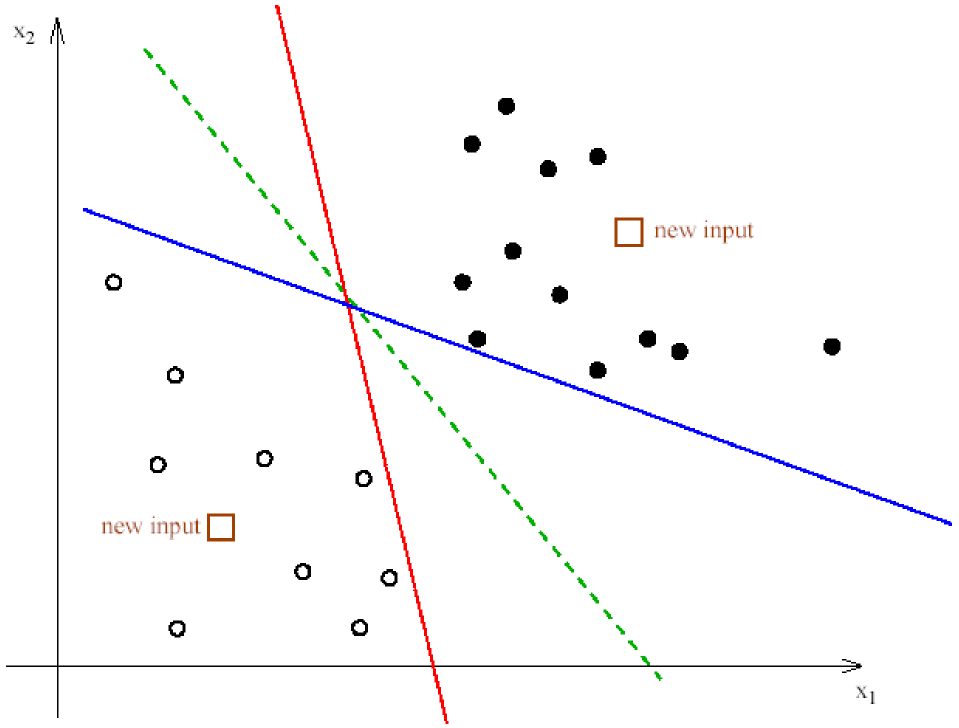
- ● denotes +1
- ○ denotes -1



How would you classify this data?

Classification Hyperplane

- Training set: ($\mathbf{x}_i$, y_i), i=1,2,…N; $y_i \in \{+1,-1\}$
- Hyperplane: $\mathbf{wx}+b=0$
  - This is fully determined by ($\mathbf{w}$,b)

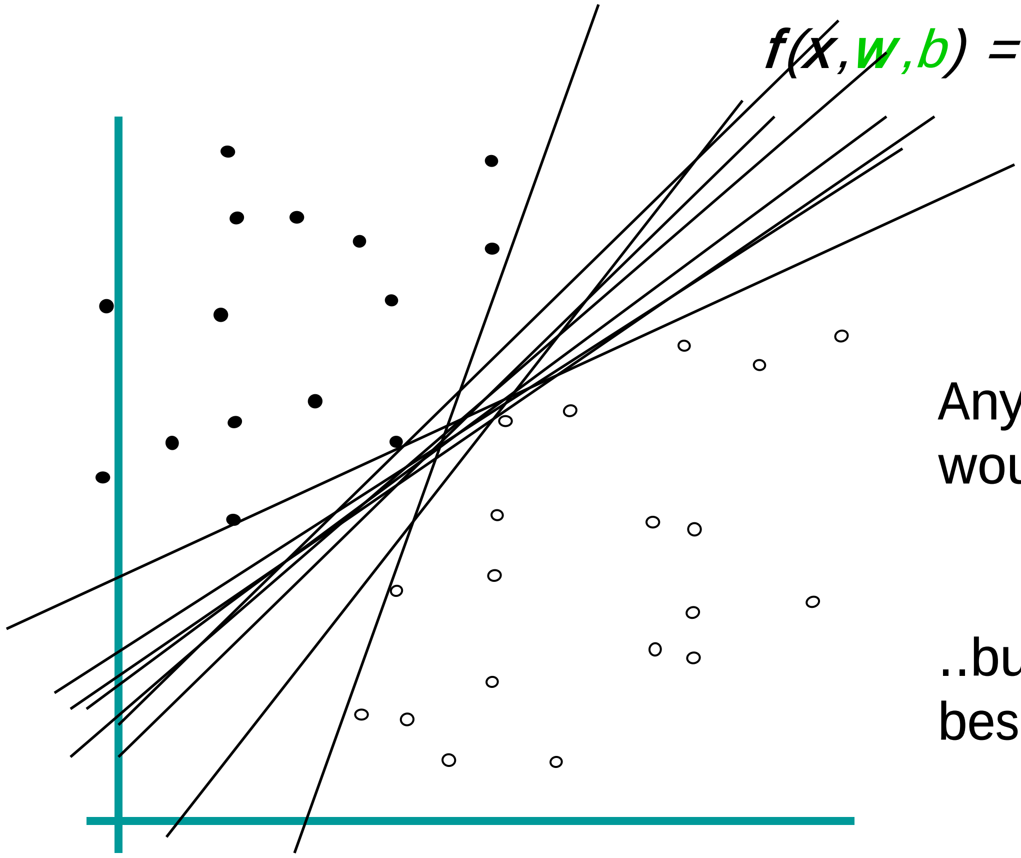


Linear Classifiers

$$x \longrightarrow \boxed{f} \longrightarrow y$$

$f(x, w, b) = sign(w . x - b)$

●	+1
○	-1



Any of these would be fine..

..but which is best?

$$g = WX$$

1. How to computer the unknown parameters w_i defining the decision hyperplane ?

Classifiers based on cost function optimization!

2. Approach the problem as a typical optimization task.

3. Design the corresponding classifier using different type of criterion.

Four Learning Criteria

The Perceptron Algorithm

The Perceptron Cost

$$J_p(w) = \sum_{\mathbf{y} \in Y} \left(-w^t \mathbf{y} \right),$$

where Y is the subset of the training vectors, which are misclassified by the hyperplane defined by the weight vector w.

We will approach the problem as a typical optimization task (Appendix C). Thus we need to adopt (a) an appropriate cost function and (b) an algorithmic scheme to optimize it. To this end, we choose the *perceptron cost* defined as

$$J(\boldsymbol{w}) = \sum_{\boldsymbol{x} \in Y} (\delta_x \boldsymbol{w}^T \boldsymbol{x}) \tag{3.6}$$

where Y is the subset of the training vectors, which are misclassified by the hyperplane defined by the weight vector $\boldsymbol{w}$. The variable δ_x is chosen so that $\delta_x = -1$ if $\boldsymbol{x} \in \omega_1$ and $\delta_x = +1$ if $\boldsymbol{x} \in \omega_2$. Obviously, the sum in (3.6) is always positive, and it becomes zero when Y becomes the empty set, that is, if there are not misclassified vectors $\boldsymbol{x}$. Indeed, if $\boldsymbol{x} \in \omega_1$ and it is misclassified, then $\boldsymbol{w}^T \boldsymbol{x} < 0$ and $\delta_x < 0$, and the product is positive. The result is the same for vectors originating from class ω_2. When the cost function takes its minimum value, 0, a solution has been obtained, since all training feature vectors are correctly classified.

Minimizing Perceptron Criterion

$$\min J_p'(\mathbf{a}) = \sum_{i \in Y'} \left(b_i - \mathbf{a}^t \mathbf{y}_i \right), \quad Y' = \left\{ i \mid \mathbf{a}^t \mathbf{y}_i \le b_i \right\}$$

Equivalent problem:

$$\min_{\mathbf{u}} z = \boldsymbol{\alpha}^t \mathbf{u} \text{ subject to } \mathbf{A}\mathbf{u} \ge \boldsymbol{\beta}, \mathbf{u} \ge 0$$

$$\mathbf{u} = \begin{bmatrix} \mathbf{a}^+ \\ \mathbf{a}^- \\ \boldsymbol{\tau} \end{bmatrix}, \mathbf{A} = \begin{bmatrix} \mathbf{y}_1^t & -\mathbf{y}_1^t & 1 & 0 & \cdots & 0 \\ \mathbf{y}_2^t & -\mathbf{y}_2^t & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}_n^t & -\mathbf{y}_n^t & 0 & 0 & \cdots & 1 \end{bmatrix}, \boldsymbol{\alpha} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{1}_n \end{bmatrix}, \boldsymbol{\beta} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Optimization method
- iterative scheme

1. the gradient descent method

$$w(t+1) = w(t) - \eta(t)\frac{\partial J(w)}{\partial w}\Big|_{w=w(t)}$$

A Good Reference for Optimization

- R. Fletcher, *Practical Methods of Optimization*, Wiley, 2nd ed., 1987.

Batch Perceptron

$$J_p(\mathbf{a}) = \sum_{\mathbf{y}\in Y}\left(-\mathbf{a}^t\mathbf{y}\right),\ Y: \text{set of samples misclassified by } \mathbf{a}$$

$$\nabla J_p = \sum_{\mathbf{y}\in Y}\left(-\mathbf{y}\right), \quad \mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\sum_{\mathbf{y}\in Y}\mathbf{y}$$

```
initialize $\mathbf{a}, \eta(\bullet)$, criterion $\theta, k \leftarrow 0$
    do $k \leftarrow k+1$
        $\mathbf{a} \leftarrow \mathbf{a} + \eta(k)\sum_{\mathbf{y}\in Y_k}\mathbf{y}$
    until $\left|\eta(k)\sum_{\mathbf{y}\in Y_k}\mathbf{y}\right| < \theta$
    return $\mathbf{a}$
end
```

Batch Relaxation with Margin

```
initialize $\mathbf{a}, \eta(\bullet), b, k \leftarrow 0$
  do $k \leftarrow (k+1) \bmod n$
     $Y_k = \{\ \}, \quad j \leftarrow 0$
     do $j \leftarrow j+1$
        if $\mathbf{a}^t \mathbf{y}^j \leq b$ then Append $\mathbf{y}^j$ to $Y_k$
     until $j = n$
     $\mathbf{a} \leftarrow \mathbf{a} + \eta(k) \sum_{\mathbf{y} \in Y_k} \frac{b - \mathbf{a}^t \mathbf{y}}{\|\mathbf{y}\|^2} \mathbf{y}$
  until $Y_k = \{\ \}$
  return $\mathbf{a}$
end
```

Single-Sample Relaxation with Margin

```
initialize $\mathbf{a}, \eta(\bullet), k \leftarrow 0$
  do $k \leftarrow (k+1) \bmod n$
      if $\mathbf{a}^t\mathbf{y}^k \le b$ then $\mathbf{a} \leftarrow \mathbf{a} + \eta(k)\dfrac{b - \mathbf{a}^t\mathbf{y}^k}{\|\mathbf{y}\|^2}\mathbf{y}^k$
  until $\mathbf{a}^t\mathbf{y}^k > b$ for all $\mathbf{y}^k$
  return $\mathbf{a}$
end
```

Least Mean Squared-Error Algorithm (LMSE)

Minimum Squared-Error Procedures

$$\begin{pmatrix} y_{10} & y_{11} & \cdots & y_{1d} \\ y_{20} & y_{21} & \cdots & y_{2d} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ y_{n0} & y_{n1} & \cdots & y_{nd} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$J_s(\mathbf{a}) = \sum_{i=1}^{n} \left(\mathbf{a}^t \mathbf{y}_i - b_i \right)^2 = \| \mathbf{Y}\mathbf{a} - \mathbf{b} \|^2$$

Minimum Squared-Error Procedures

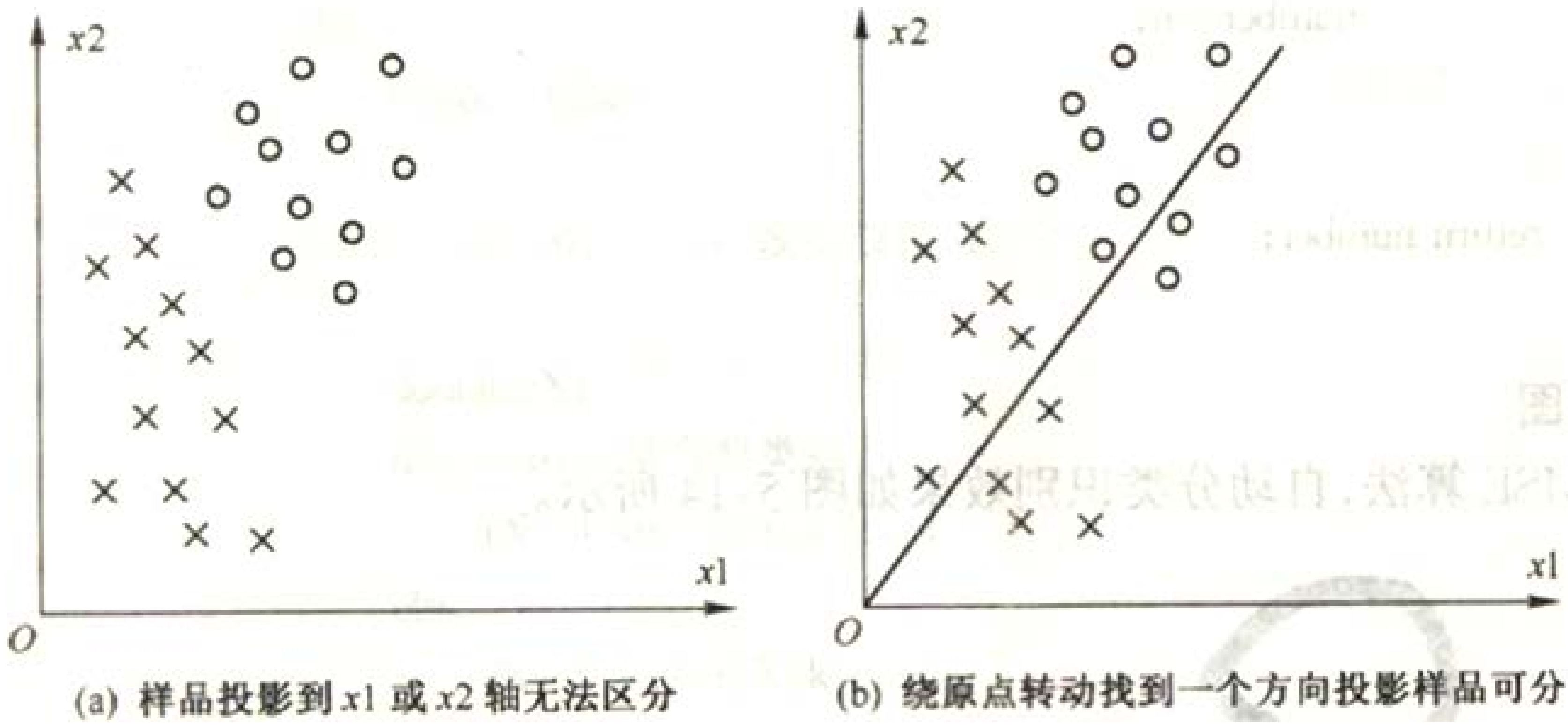
$$\nabla J_s = \sum_{i=1}^{n} 2\left(\mathbf{a}^t \mathbf{y}_i - b_i\right)\mathbf{y}_i = 2\mathbf{Y}^t\left(\mathbf{Y}\mathbf{a} - \mathbf{b}\right) = 0$$

$$\mathbf{Y}^t\mathbf{Y}\mathbf{a} = \mathbf{Y}^t\mathbf{b}$$

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

$$\text{pseudoinverse}\;\mathbf{Y}^+ = \left(\mathbf{Y}^t\mathbf{Y}\right)^{-1}\mathbf{Y}^t$$

Fisher Algorithm


x2
x1
O
(a) 样品投影到 x1 或 x2 轴无法区分
x2
x1
O
(b) 绕原点转动找到一个方向投影样品可分

Principle of Fisher method

Relation to Fisher's Linear Discriminant

$$D_1 = \{\mathbf{x}_1, \ldots, \mathbf{x}_{n_1}\},\ D_2 = \{\mathbf{x}_{n_1+1}, \ldots, \mathbf{x}_{n_1+n_2}\}$$

$$\text{augmented pattern}: \mathbf{y}_i = \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix}$$

$$\mathbf{Y} = \begin{pmatrix} \mathbf{1}_1 & \mathbf{X}_1 \\ -\mathbf{1}_2 & \mathbf{X}_2 \end{pmatrix},\ \mathbf{a} = \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix},\ \mathbf{b} = \begin{pmatrix} \frac{n}{n_1}\mathbf{1}_1 \\ \frac{n}{n_2}\mathbf{1}_2 \end{pmatrix}$$

Relation to Fisher's Linear Discriminant

$$\begin{pmatrix} \mathbf{1}_1^t & -\mathbf{1}_2^t \\ \mathbf{X}_1^t & -\mathbf{X}_2^t \end{pmatrix} \begin{pmatrix} \mathbf{1}_1 & \mathbf{X}_1 \\ -\mathbf{1}_2 & -\mathbf{X}_2 \end{pmatrix} \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix} = \begin{pmatrix} \mathbf{1}_1^t & -\mathbf{1}_2^t \\ \mathbf{X}_1^t & -\mathbf{X}_2^t \end{pmatrix} \begin{pmatrix} \frac{n}{n_1}\mathbf{1}_1 \\ \frac{n}{n_2}\mathbf{1}_2 \end{pmatrix}$$

$$\begin{pmatrix} n & (n_1\mathbf{m}_1 + n_2\mathbf{m}_2)^t \\ (n_1\mathbf{m}_1 + n_2\mathbf{m}_2) & S_W + n_1\mathbf{m}_1\mathbf{m}_1^t + n_2\mathbf{m}_2\mathbf{m}_2^t \end{pmatrix} \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix} = \begin{pmatrix} 0 \\ n(\mathbf{m}_1 - \mathbf{m}_2) \end{pmatrix}$$

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{x}, \quad \mathbf{S}_W = \sum_{i=1}^{2} \sum_{\mathbf{x} \in D_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t$$

Relation to Fisher's Linear Discriminant

$$w_0 = -\mathbf{m}^t \mathbf{w}$$

$$\left[\frac{1}{n}\mathbf{S}_W + \frac{n_1 n_2}{n^2}\left(\mathbf{m}_1 - \mathbf{m}_2\right)\left(\mathbf{m}_1 - \mathbf{m}_2\right)^t\right]\mathbf{w} = \mathbf{m}_1 - \mathbf{m}_2$$

$$\frac{n_1 n_2}{n^2}\left(\mathbf{m}_1 - \mathbf{m}_2\right)\left(\mathbf{m}_1 - \mathbf{m}_2\right)^t \mathbf{w} = \left(1 - \alpha\right)\left(\mathbf{m}_1 - \mathbf{m}_2\right)$$

$$\mathbf{w} = \alpha n \mathbf{S}_W^{-1}\left(\mathbf{m}_1 - \mathbf{m}_2\right)$$

Fisher's linear discriminant

- Fisher's linear discriminant is a classification method that projects high-dimensional data onto a line and performs classification in this one-dimensional space.
- The projection maximizes the distance between the means of the two classes while minimizing the variance within each class.

- This defines the Fisher criterion, which is maximized over all linear projections, ***w***:

$$J(w) = \frac{|m_1 - m_2|^2}{s_1^2 + s_2^2}$$

- where ***m*** represents a mean, **s**^{2} represents a variance, and the subscripts denote the two classes.

Fisher's linear discriminant

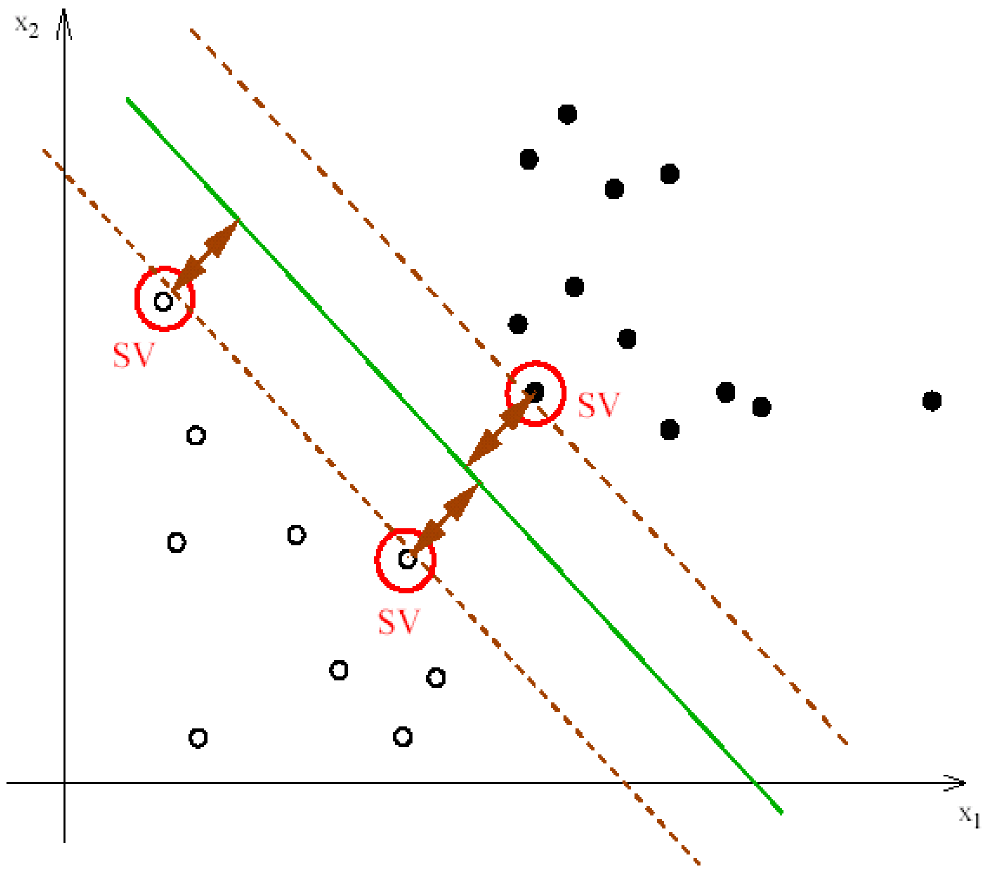
- In signal theory, this criterion is also known as the signal-to-interference ratio. Maximizing this criterion yields a closed form solution that involves the inverse of a covariance-like matrix. This method has strong parallels to linear perceptrons. We learn the threshold by optimizing a cost function on the training set.

Support Vector Machines

结构风险最小化归纳原则支持向量机(SVM)

- SVMs are learning systems that
  - use a hyperplane（超平面） of *linear functions*
  - in a high dimensional feature space — *Kernel function*
  - trained with a learning algorithm from optimization theory — *Lagrange*
  - Implements a learning bias derived from statistical learning theory — *Generalisation* SVM is a classifier derived from statistical learning theory by Vapnik and Chervonenkis

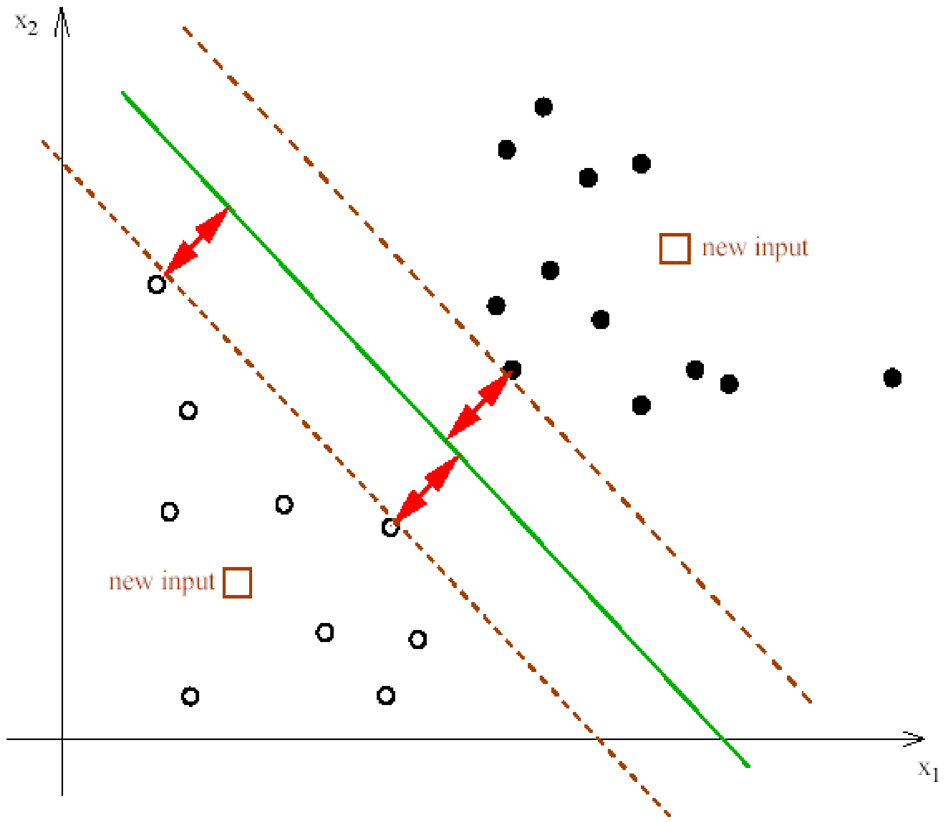
支持向量



The training points that are nearest to the separating function are called support vectors.

What is the output of our decision function for these points?

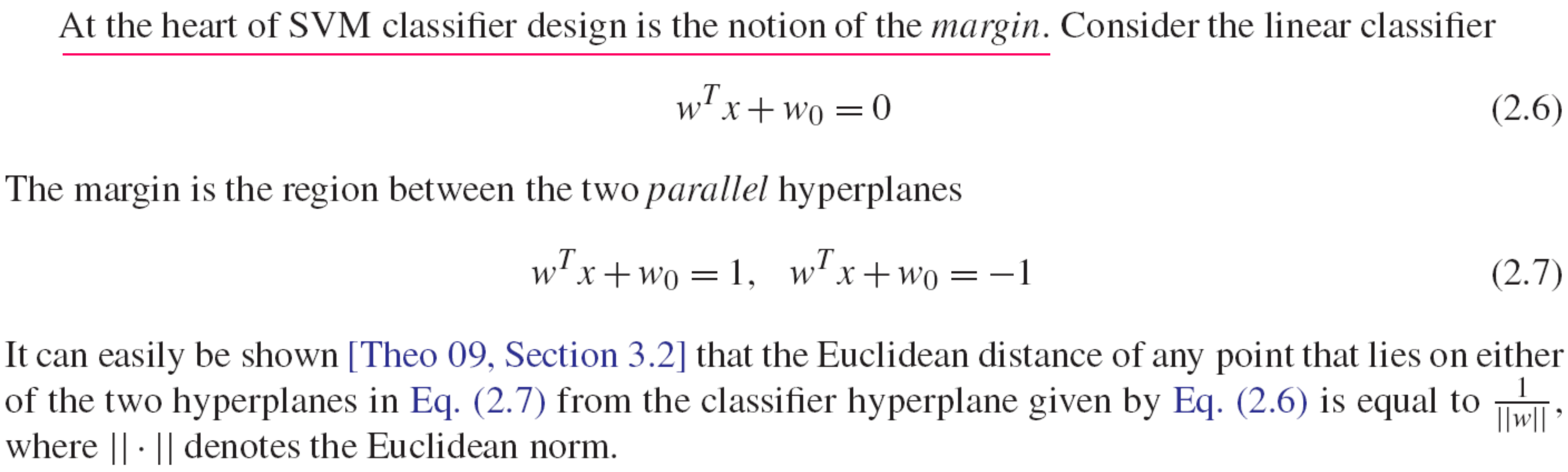
Maximising Margin



According to a theorem from Learning Theory, from all possible linear decision functions the one that maximises the margin of the training set will minimise the generalisation error.

最大间隔原则

怎么定？怎么算？



At the heart of SVM classifier design is the notion of the *margin*. Consider the linear classifier

$$w^T x + w_0 = 0 \tag{2.6}$$

The margin is the region between the two *parallel* hyperplanes

$$w^T x + w_0 = 1, \quad w^T x + w_0 = -1 \tag{2.7}$$

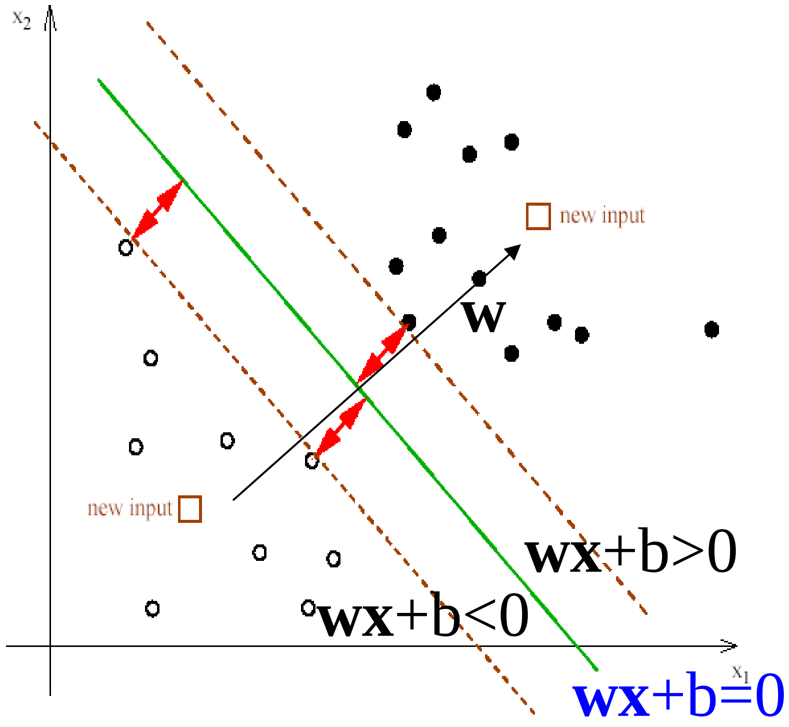
It can easily be shown [Theo 09, Section 3.2] that the Euclidean distance of any point that lies on either of the two hyperplanes in Eq. (2.7) from the classifier hyperplane given by Eq. (2.6) is equal to $\frac{1}{||w||}$, where $||\cdot||$ denotes the Euclidean norm.

Margin 间隔！

最大间隔原则

最优超平面定义的分类决策函数为

$$f(x) = \mathrm{sgn}(g(x)) = \mathrm{sgn}((w \cdot x) + b)$$



Note1: decision functions (**w**,b) and (c**w**, cb) are the same

Note2: but margins as measured by the outputs of the function **x**→**wx**+b are not the same if we take (c**w**, cb).

Definition: *geometric margin*: the margin given by the *canonical decision function*规范化决策函数, which is when c=1/||**w**||

Strategy:

1) we need to maximise the geometric margin! (cf result from learning theory)

在规范化的分类超平面！

2) subject to the constraint that training examples are classified correctly

A question sometimes raised by a newcomer in the field is why the margin is defined by these two "magic" numbers, $+1$ and -1. The answer is that this is not an issue. Let us consider a hyperplane in space—for example, Eq. (2.6), as shown in Figure 2.3 by the full line and two parallel to it hyperplanes (*dotted lines*) $w^T x + w_0 = \pm d$. The parameter d can take any value, which means that the two planes can be close to or far away from each other. Fixing the value of d and dividing both sides of the previous equation by d, we obtain ± 1 on the right side. However, the direction and the position in space of the two hyperplanes do not change. The same applies to the hyperplane described by Eq. (2.6). Normalization by a constant value d has no effect on the points that lie on (and define) a hyperplane.

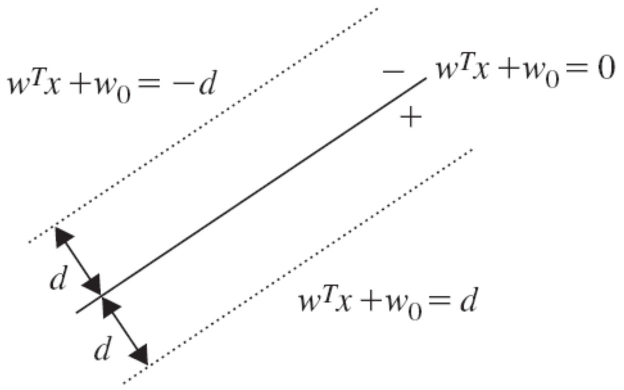


FIGURE 2.3

Line and its margin of size $2d$.

$$w^T x + w_0 = 0 \tag{2.6}$$

最大间隔原则（进一步补充）

According to Note1, we can demand the function output for the nearest points to be +1 and –1 on the two sides of the decision function. This removes the scaling freedom.

Denoting a nearest positive example x_+ and a nearest negative example x_-, this is $\mathbf{w}\mathbf{x}_+ + b = +1$ and $\mathbf{w}\mathbf{x}_- + b = -1$

Computing the geometric margin (that has to be maximised):

$$\frac{1}{2}\left(\frac{\mathbf{w}}{\|\mathbf{w}\|}\mathbf{x}_+ + \frac{b}{\|\mathbf{w}\|} - \frac{\mathbf{w}}{\|\mathbf{w}\|}\mathbf{x}_- - \frac{b}{\|\mathbf{w}\|}\right) = \frac{1}{2\|\mathbf{w}\|}(\mathbf{w}\mathbf{x}_+ + b - \mathbf{w}\mathbf{x}_- - b) = \frac{1}{\|\mathbf{w}\|}$$

And here are the constraints:

$$\left.\begin{array}{ll} \mathbf{w}\mathbf{x}_i + b \geq +1 & \text{for } y_i = +1 \\ \mathbf{w}\mathbf{x}_i + b \leq -1 & \text{for } y_i = -1 \end{array}\right\} \Leftrightarrow y_i(\mathbf{w}\mathbf{x}_i + b) - 1 \geq 0 \quad \text{for all } i$$

Margin = $\dfrac{2}{\|\mathbf{w}\|}$ .....(1)

H1平面： $\mathbf{W} \bullet \mathbf{X}_1 + b \geq 1$

H2平面： $\mathbf{W} \bullet \mathbf{X}_2 + b \leq -1$

$y_i[(\mathbf{W} \bullet \mathbf{X}_i) + b] - 1 \geq 0$ .....(2)

利用二次优化求解

$$\frac{1}{2}\mathbf{w}.\mathbf{w}$$

Minimize

$$y_k(\mathbf{w} \,.\, \mathbf{x}_k + b) >= 1$$

subject to

$$k=1,2,\ldots,n$$

链接约束条件的优化计算

Support Vector Machines

distance from a hyperplane to $\mathbf{y}$: $\dfrac{|g(\mathbf{y})|}{\|\mathbf{a}\|}$

positive margin b,

$$\frac{z_k g(\mathbf{y}_k)}{\|\mathbf{a}\|} \geq b, k = 1, \ldots, n$$

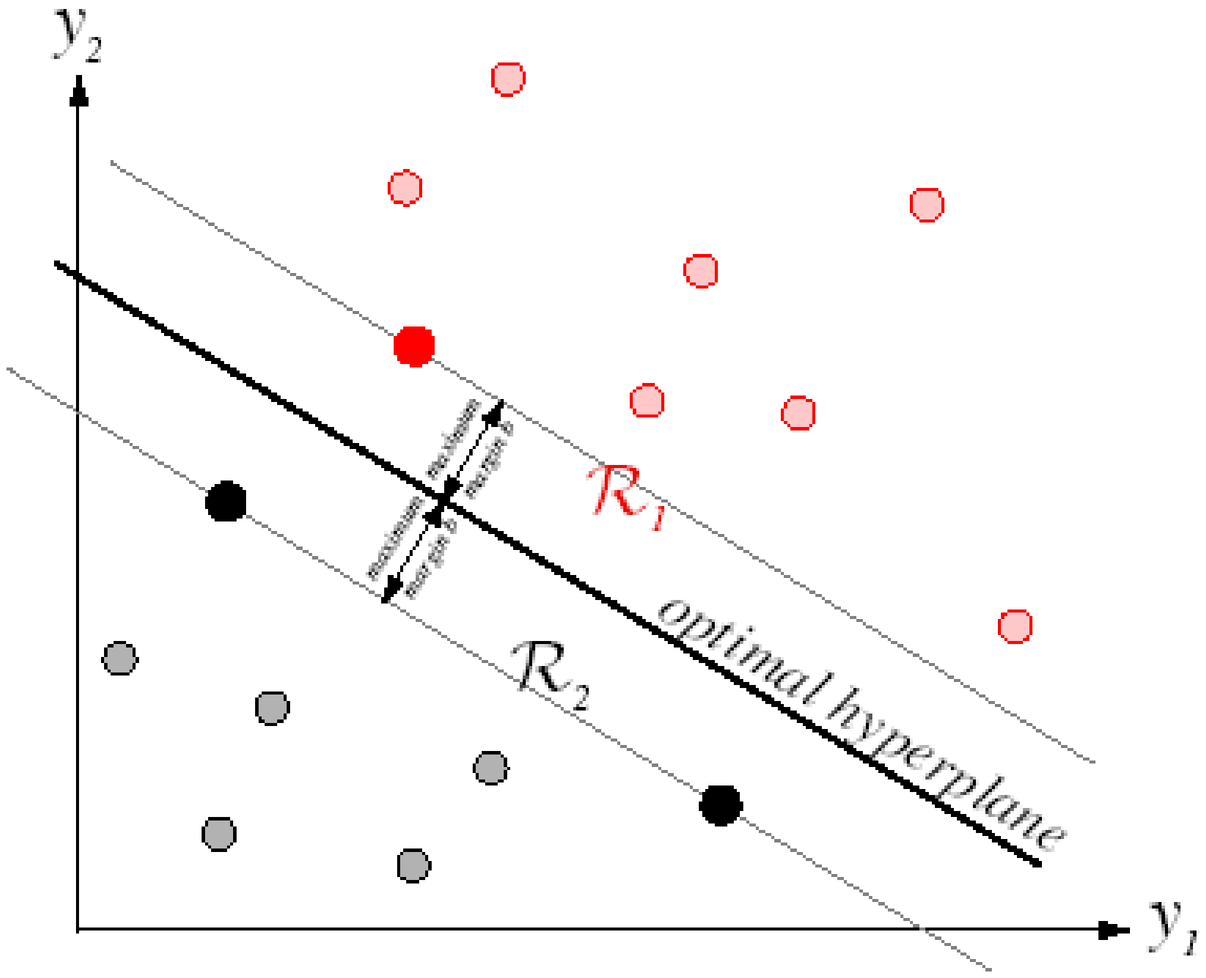
Goal : find $\mathbf{a}$ to maximize b

Uniqueness constraint : $b\|\mathbf{a}\| = 1$

Alternative problem :

$$\min_{\mathbf{a}} \|\mathbf{a}\|^2 \text{ subject to } z_k g(\mathbf{y}_k) \geq 1, k = 1, \ldots, n$$

Support Vector Machines



A Simple SVM Training Method

- A modification the Perceptron training rule
- Trained by choosing the current *worst*-classified pattern to update the weight vector
- At the end of the training period, such a pattern will be one of the support vectors
- Computationally expensive

A question sometimes raised by a newcomer in the field is why the margin is defined by these two "magic" numbers, $+1$ and -1. The answer is that this is not an issue. Let us consider a hyperplane in space—for example, Eq. (2.6), as shown in Figure 2.3 by the full line and two parallel to it hyperplanes (*dotted lines*) $w^T x + w_0 = \pm d$. The parameter d can take any value, which means that the two planes can be close to or far away from each other. Fixing the value of d and dividing both sides of the previous equation by d, we obtain ± 1 on the right side. However, the direction and the position in space of the two hyperplanes do not change. The same applies to the hyperplane described by Eq. (2.6). Normalization by a constant value d has no effect on the points that lie on (and define) a hyperplane.

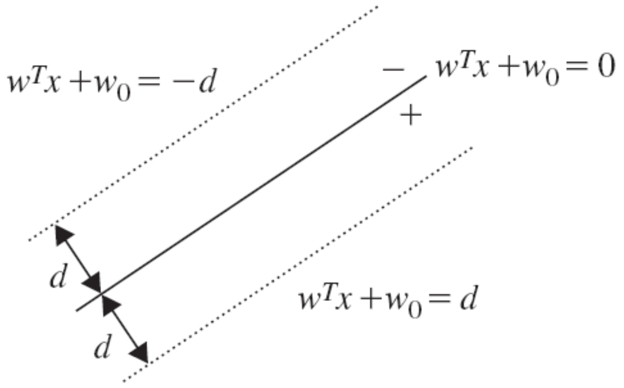


FIGURE 2.3

Line and its margin of size $2d$.

$$w^T x + w_0 = 0 \tag{2.6}$$

Learning by Optimization

$$\min_{\mathbf{a}} L(\mathbf{a}, \boldsymbol{\alpha})$$

$$L(\mathbf{a}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{a}\|^2 - \sum_{k=1}^{n} \alpha_k \left[z_k \mathbf{a}^t \mathbf{y}_k - 1\right],$$

$$\boldsymbol{\alpha} \geq 0$$

Kuhn-Tucker Theorem

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to } g_i(\mathbf{x}) \geq 0, i = 1,2,\ldots,r$$

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{i=1}^{r} \lambda_i g_i(\mathbf{x})$$

Necessary conditions (KT conditions)

$$\frac{\partial f}{\partial x_j}(\mathbf{x}^*) - \sum_{i=1}^{r} \lambda_i^* \frac{\partial g_i}{\partial x_j}(\mathbf{x}^*) = 0, \ j = 1,2,\ldots,n$$

$$g_i(\mathbf{x}^*) \geq 0, \quad i = 1,2,\ldots,r$$

$$\lambda_i^* g_i(\mathbf{x}^*) = 0, \quad i = 1,2,\ldots,r$$

$$\lambda_i^* \geq 0, \quad i = 1,2,\ldots,r$$

Equivalent Quadratic Programming Problem

$$\max_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}) \text{ subject to } \sum_{k=1}^{n} z_k \alpha_k = 0, \boldsymbol{\alpha} \geq 0$$

$$L(\boldsymbol{\alpha}) = \sum_{k=1}^{n} \alpha_k - \frac{1}{2} \sum_{k,j}^{n} \alpha_k \alpha_j z_k z_j \mathbf{y}_j^t \mathbf{y}_k$$

Benefits of SVM

- Complexity of the resulting classifiers is characterized by the number of support vectors rather than the dimensionality of the transformed space
- Tends to be less prone to problems of overfitting than some other methods

Multicategory Generalizations

generalized linear discriminant functions

$$g_i(\mathbf{x}) = \mathbf{a}_i^t \mathbf{y}(\mathbf{x}), i = 1, \ldots, c$$

samples are linearly separable if there exists a set of $\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_c$ such that if $\mathbf{y}_k \in Y_i$, then

$$\hat{\mathbf{a}}_i^t \mathbf{y}_k > \hat{\mathbf{a}}_j^t \mathbf{y}_k \text{ for all } j \neq i$$

A Good Reference for Optimization

- R. Fletcher, *Practical Methods of Optimization*, Wiley, 2nd ed., 1987.

Basic Gradient Descent Algorithm

define a criterion function $J(\mathbf{a})$

minimized if $\mathbf{a}$ is a solution vector

```
initialize $\mathbf{a}$, threshold $\theta, \eta(\bullet), k \leftarrow 0$
    do $k \leftarrow k+1$
        $a \leftarrow a - \eta(k)\nabla J(\mathbf{a})$
    until $\left|\eta(k)\nabla J(\mathbf{a})\right| < \theta$
    return $\mathbf{a}$
end
```

Choice of Learning Rate: Minimizing Quadratic Approx.

$$J(\mathbf{a}) \approx J(\mathbf{a}(k)) + \nabla J^t \left(\mathbf{a} - \mathbf{a}(k)\right) + \frac{1}{2}\left(\mathbf{a} - \mathbf{a}(k)\right)^t \mathbf{H}\left(\mathbf{a} - \mathbf{a}(k)\right)$$

$$\mathbf{H} : \text{Hessian matrix, } H_{ij} = \left.\frac{\partial^2 J}{\partial a_i \partial a_j}\right|_{\mathbf{a}=\mathbf{a}(k)}$$

$$\mathbf{a}(k+1) = \mathbf{a}(k) - \eta(k)\nabla J(\mathbf{a}(k))$$

$$J(\mathbf{a}(k+1)) \approx J(\mathbf{a}(k)) - \eta(k)\left\|\nabla J\right\|^2 + \frac{1}{2}\eta^2(k)\nabla J^t \mathbf{H} \nabla J$$

$$\text{minimize } J(\mathbf{a}(k+1)) \text{ by choosing } \eta(k) = \frac{\left\|\nabla J\right\|^2}{\nabla J^t \mathbf{H} \nabla J}$$

Newton's Algorithm

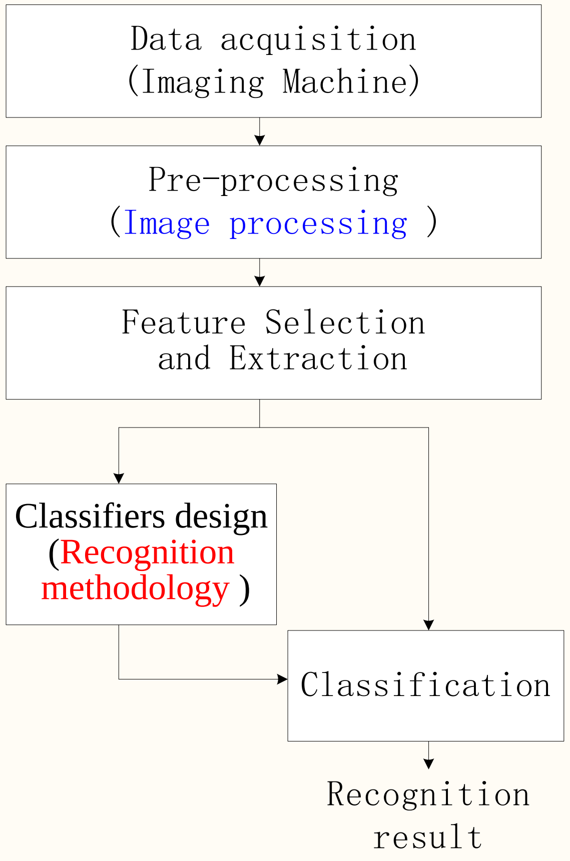
$$J(\mathbf{a}) \approx J(\mathbf{a}(k)) + \nabla J^t \left(\mathbf{a} - \mathbf{a}(k)\right) + \frac{1}{2}\left(\mathbf{a} - \mathbf{a}(k)\right)^t \mathbf{H}\left(\mathbf{a} - \mathbf{a}(k)\right)$$

$$\mathbf{H} : \text{Hessian matrix, } H_{ij} = \left.\frac{\partial^2 J}{\partial a_i \partial a_j}\right|_{\mathbf{a}=\mathbf{a}(k)}$$

$$\text{minimize } J(\mathbf{a}) \text{ by choosing } \mathbf{a} = \mathbf{a}(k+1) = \mathbf{a}(k) - \mathbf{H}^{-1}\nabla J$$

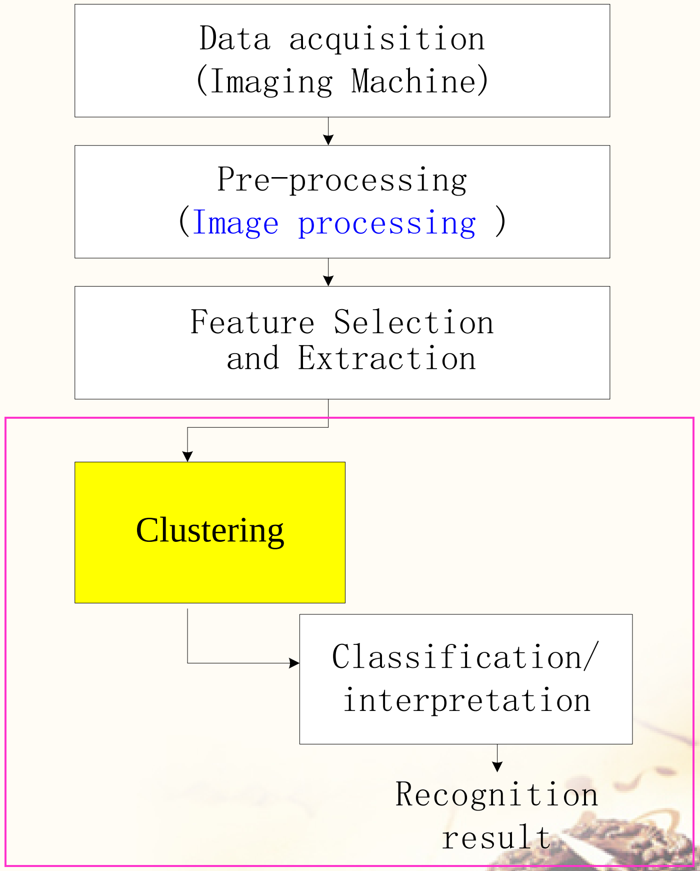
```
initialize $\mathbf{a}$, threshold $\theta$
    do $\mathbf{a} \leftarrow \mathbf{a} - \mathbf{H}^{-1}\nabla J(\mathbf{a})$
    until $\left|\mathbf{H}^{-1}\nabla J(\mathbf{a})\right| < \theta$
    return $\mathbf{a}$
end
```

Illustration for Pattern recognition system



Data acquisition (Imaging Machine)
Pre-processing (Image processing)
Feature Selection and Extraction
Classifiers design (Recognition methodology)
Classification
Recognition result

Supervised pattern recognition



Data acquisition (Imaging Machine)
Pre-processing (Image processing)
Feature Selection and Extraction
Clustering
Classification/ interpretation
Recognition result

Feature space!

Unsupervised pattern recognition

7.6 Clustering

Supervised vs. Unsupervised Learning

- Supervised training procedures
  - Use samples labeled by their category membership
- Unsupervised training procedures
  - Use unlabeled samples

Reasons for interest

- Collecting and labeling a large set of sample patterns can be costly
  - e.g., speech
- Training with large amount of unlabeled data, and using supervision to label the groupings found
  - For “data mining” applications
- Improved performance for data with slow changes of characteristics of patterns by tracking in an unsupervised mode
  - Automated food classification when seasons change

Reasons for interest

- Can use unsupervised methods to find features that will then be useful for categorization
  - Data dependent “smart preprocessing” or “smart feature extraction”
- Perform exploratory data analysis and gain insights into the nature or structure of the data
  - Discovery of distinct clusters may suggest us to alter the approach to designing the classifier

k-Means Clustering

$\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}})$ is large when $\left(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i\right)^t \hat{\boldsymbol{\Sigma}}_i^{-1}\left(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i\right)$ is small

merely compute $\left\|\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i\right\|^2$, find $\hat{\boldsymbol{\mu}}_m$ nearest to $\mathbf{x}_k$, approximate $\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}})$ as

$$\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}}) = \begin{cases} 1 & \text{if } i = m \\ 0 & \text{otherwise} \end{cases}$$

iteratively apply $\hat{\boldsymbol{\mu}}_i = \dfrac{\sum_{k=1}^{n} \hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}})\mathbf{x}_k}{\sum_{k=1}^{n} \hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}})}$

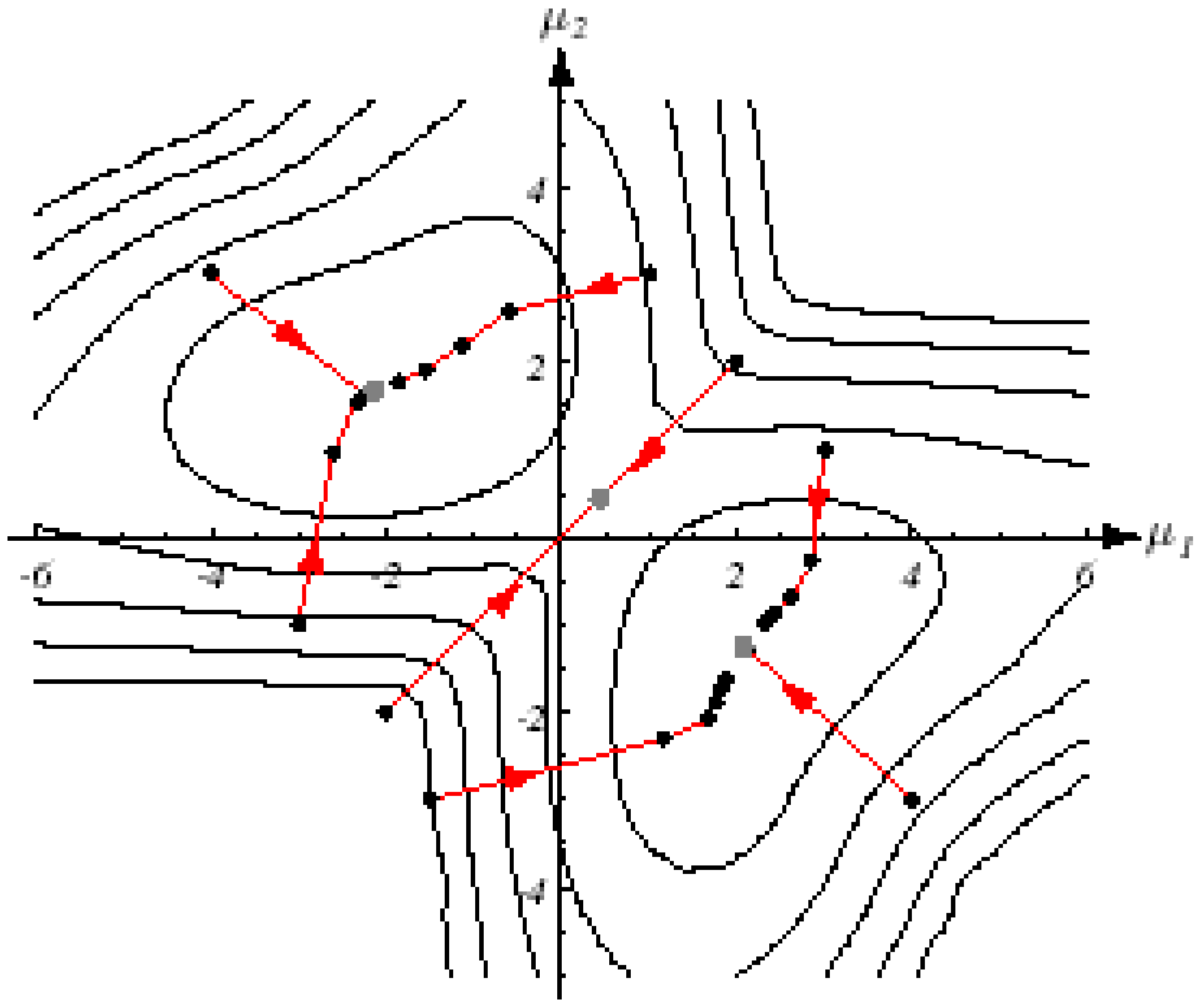
k-Means Clustering

```
initialize $n, c, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_c$
    do classify $n$ samples according to nearest $\boldsymbol{\mu}_i$
        recompute $\boldsymbol{\mu}_i$
    until no change in $\boldsymbol{\mu}_i$
    return $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_c$
end
```

k-Means Clustering

- Complexity $O(ndcT)$
- In practice, the number of iterations T is generally much less than the number of samples
- The values obtained can be accepted as the answer, or can be used as starting points for more exact computations

k-Means Clustering



$\hat{\mu}_1 \approx -2.176$

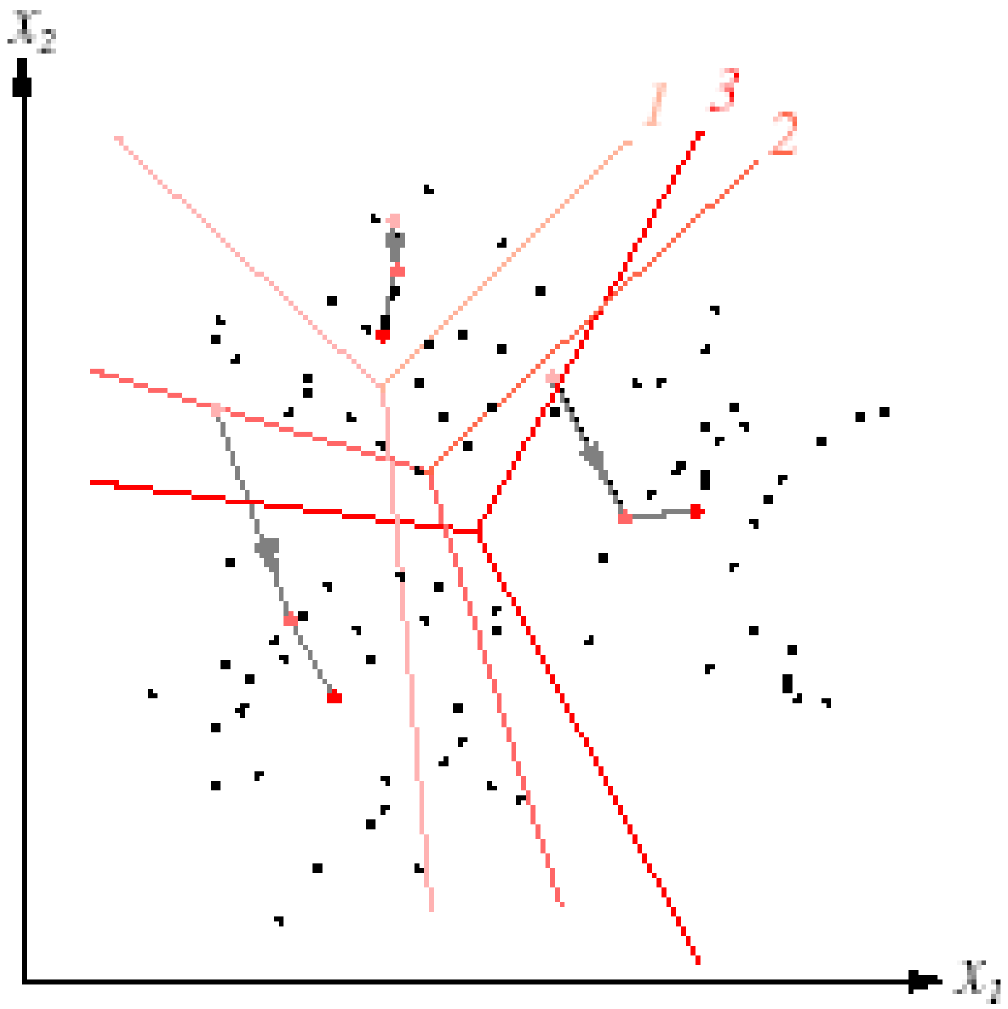
$\hat{\mu}_2 \approx 1.684$

maximum - likelihood

$\hat{\mu}_1 \approx -2.130$

$\hat{\mu}_2 \approx 1.688$

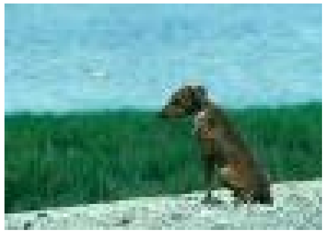
k-Means Clustering



K-means算法在图像分割上的简单应用

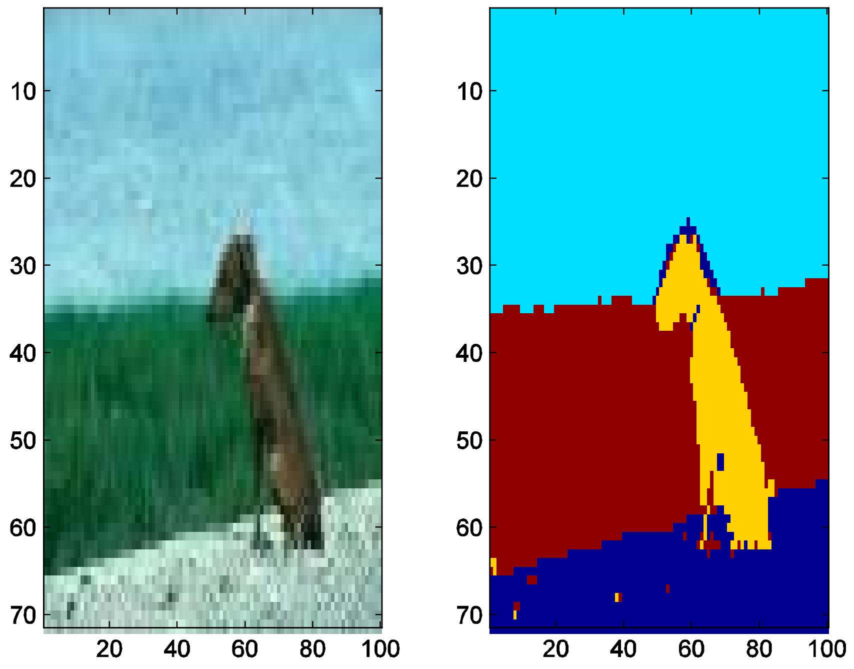
例1：

1. 图片：一只遥望大海的小狗；
2. 此图为100 x 100像素的JPG图片，每个像素可以表示为三维向量（分别对应JPEG图像中的红色、绿色和蓝色通道）；
3. 将图片分割为合适的背景区域（三个）和前景区域（小狗）；
4. 使用K-means算法对图像进行分割。



K-均值聚类
补充内容

在图像分割上的简单应用



分割后的效果

注：最大迭代次数为20次，需运行多次才有可能得到较好的效果。

Knowledge points

- Bayesian Decision Theory
- Linear and Nonlinear Classifiers
- Clustering

Questions and Practices

- 1)PLS Work hard to finish project 1 in time.
- 2)PLS Work hard to finish project 2 in time.