

Pattern Recognition

Dr P. N. Suganthan

Room No.: S2-B2a-21

Tel : 67905404

E-mail: epnsugan@ntu.edu.sg

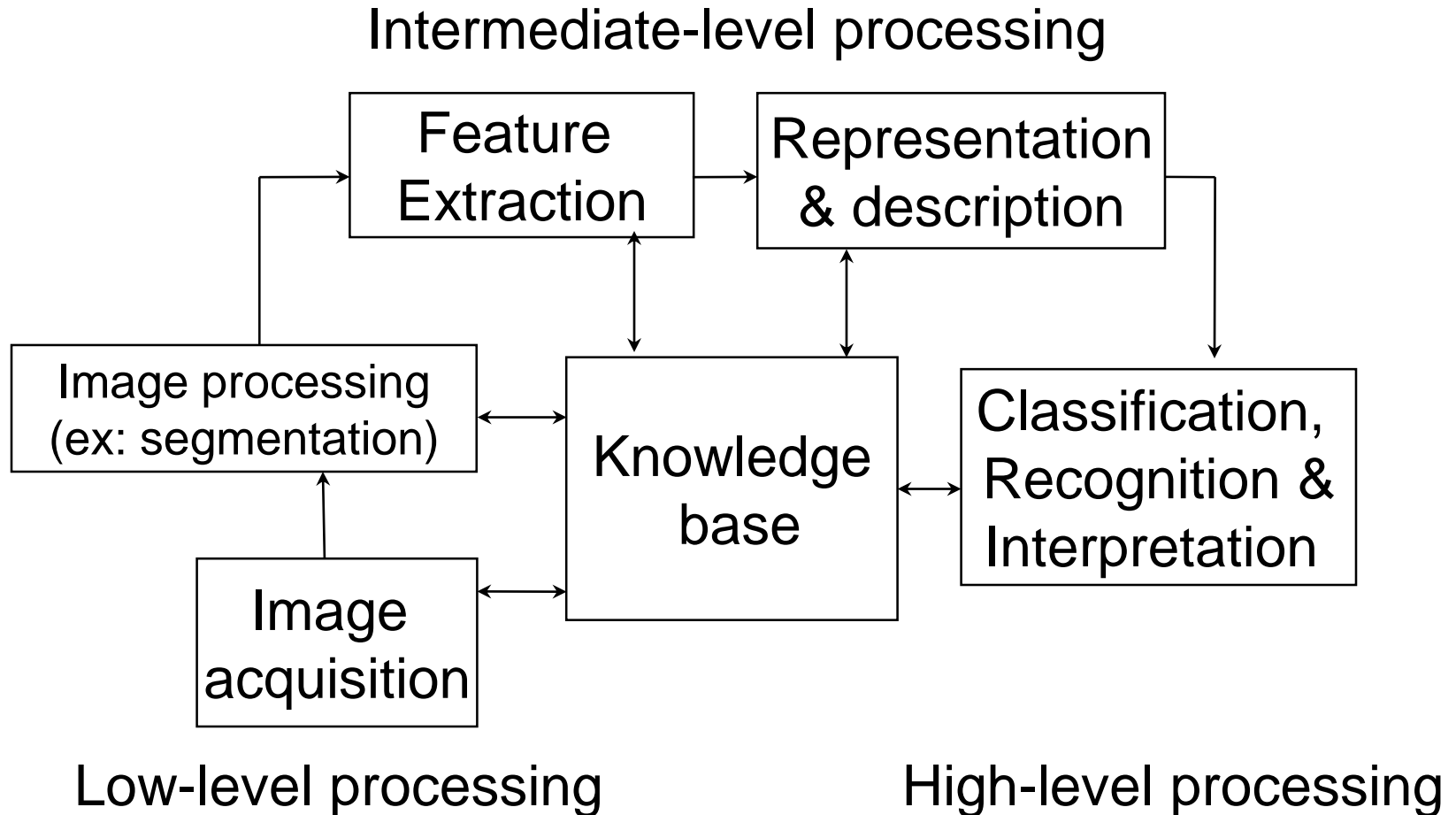
Outline

- Pattern Recognition (weeks 5 - 7)
 - Model based approach: template matching (covered in week 3)
 - Statistical approaches
 - Introduction (slides 4 - 20)
 - Supervised
 - Minimum distance (slide 21), minimum risk (slide 30), knn (slide 49), Fisher's discriminant (slide 55), Random Vector Functional Link (slide 70), Random Forest (slide 81), Convolutional Neural Nets (slide 88).
 - Unsupervised: Clustering algorithms (slide 104)

Recommended Reference Books

1. R.C. Gonzalez & R. E. Woods, “Digital image processing”, Addison-Wesley, 2010.
2. RO Duda, E Hart and DG Stork, “Pattern Classification 2nd ed.” John Wiley & Sons, Inc 2001. [Duda01]
3. Bow ST, *Pattern recognition and image preprocessing*, M. Dekker, TK7882.P3B784P 2002.

Overview of image analysis



Design Cycle

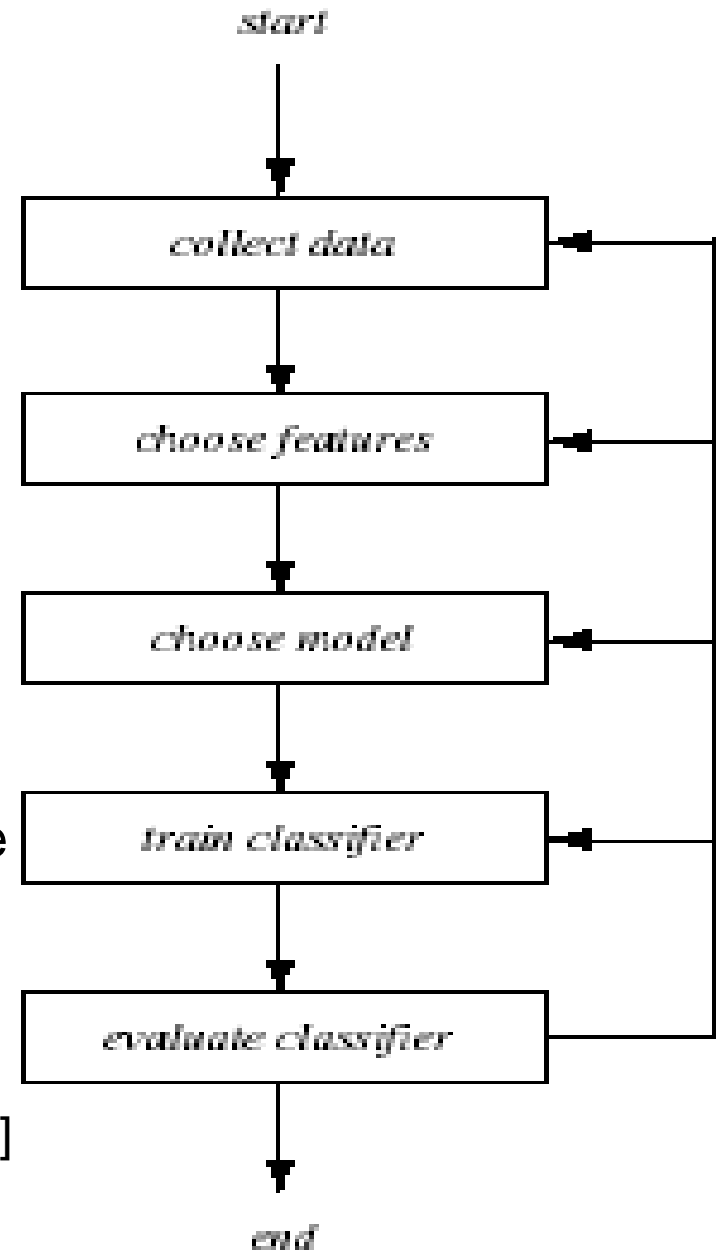
Iterative process.
No universal approach.
Feature selection not mandatory.

*prior knowledge
(e.g., invariances)*

Training – use data to determine optimum
Classifier or providing optimal decision surface

Evaluation of error for improvement

[Duda01]



What is a feature ?

A feature is a characteristic primitive or attribute of image patterns.

(e.g. pixel value of an image point or a distribution of pixel values within an image region, shape characteristics of patterns, etc.)

A pattern may be characterized by one or more features.

Examples of features used to characterize regions/objects

Features used to represent image regions (objects) are :

area (A): counting the no. of pixels within a region. This gives a measure of the size of the region.

Perimeter (P): counting the no. of pixels on the boundary of an image region.

roundness (R): a measure of circularity (R=1 for a circle): $R = \frac{P^2}{4\pi A}$

A lot more on features in lecture set 1.

Statistical features:

- Statistical measures commonly used to represent features and the variations of feature values.
- Two basic types of statistical measures are:
 - First order statistics - e.g. mean, variance & standard deviation of feature distribution.
(assumed normal distribution).
 - Second order statistics - features obtained from co-occurrence matrix.
- By analyzing feature values, shapes/patterns can be classified and recognized.
- Statistical features can be any extractable measurements.

Supervised Learning and Feature Space

One of the most used paradigms of machine learning/pattern recognition is the **supervised** (appearance-based/instance-based) **learning**. This type of learning uses the information of available examples and also to which object-class the examples belong to.

Usually, one has to extract certain characteristics or features from the set of objects to recognised. These features then represent the objects. For example, to distinguish between the object classes oranges, apples and watermelon, one might choose the weight and the smoothness of the skin as features. Here we have three classes, $\{\mathcal{C}_o, \mathcal{C}_a, \mathcal{C}_w\}$ and two-dimensional feature vector $\mathbf{f} = [f_1 \ f_2]^T$.

Supervised learning essentially has as input for learning the training set $\{f_i, y_i\}$ for $i = 1$ to k . That is, there are k examples each of which has label y_i that associates f_i with class C_i .

If we only have $\{f_i\}$ without the labels, then we need to apply some **unsupervised learning** first to find the classes or subclasses.

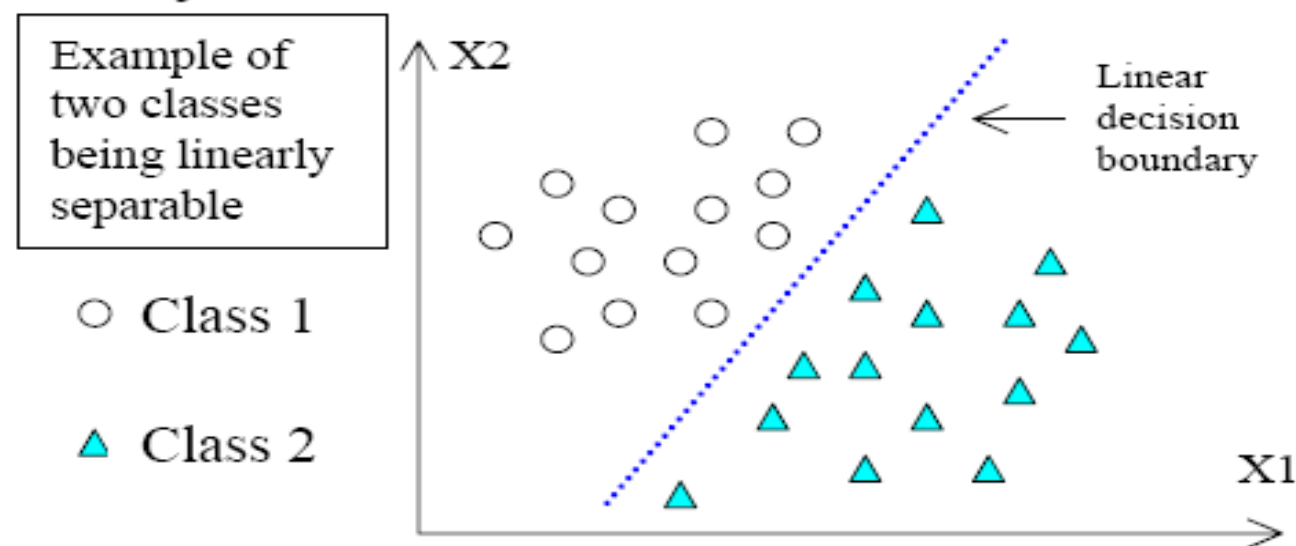
As in the fruit classification example above, it is necessary in all recognition tasks to first of all extract some features from the objects. Obviously, the ideal would be to select the fewest that can be used to distinguish one class from the others.

In computer vision, we use images for recognition. Thus the features must be extracted from intensity and/or colour and spatial relationships of the images. We could use all kinds of techniques --- fourier transforms, wavelets, edges, colour, textures, geometric shapes etc. Or we could use the raw pixel information itself as features. In this particular case, the result would be a very high dimensional feature space. How do we handle this ?

In this course, we will learn how to handle high dimensional feature spaces. In fact, there is good reason to work in high dimensional space. We study several abstract ways of extracting good features.

Decision Boundaries

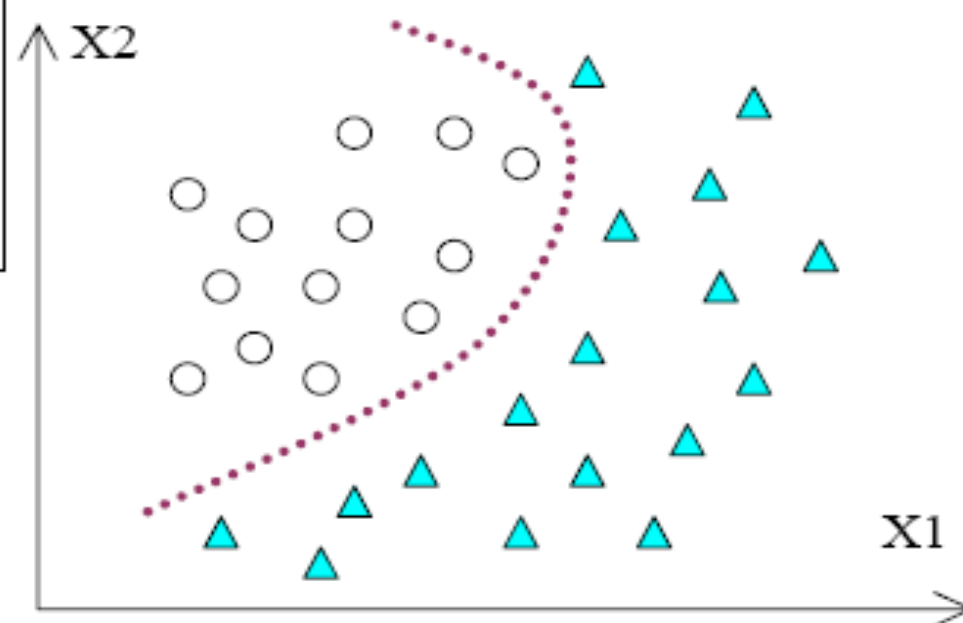
It is of course desirable that two object classes can be separated by a curve (hypersurface). If we can find a line (hyperplane) to separate the two classes, then we say they are **linearly separable**. Recall that minimum distance classifier generates a linear decision boundary.



Example of
non-linear
separation.

○ Class 1

▲ Class 2



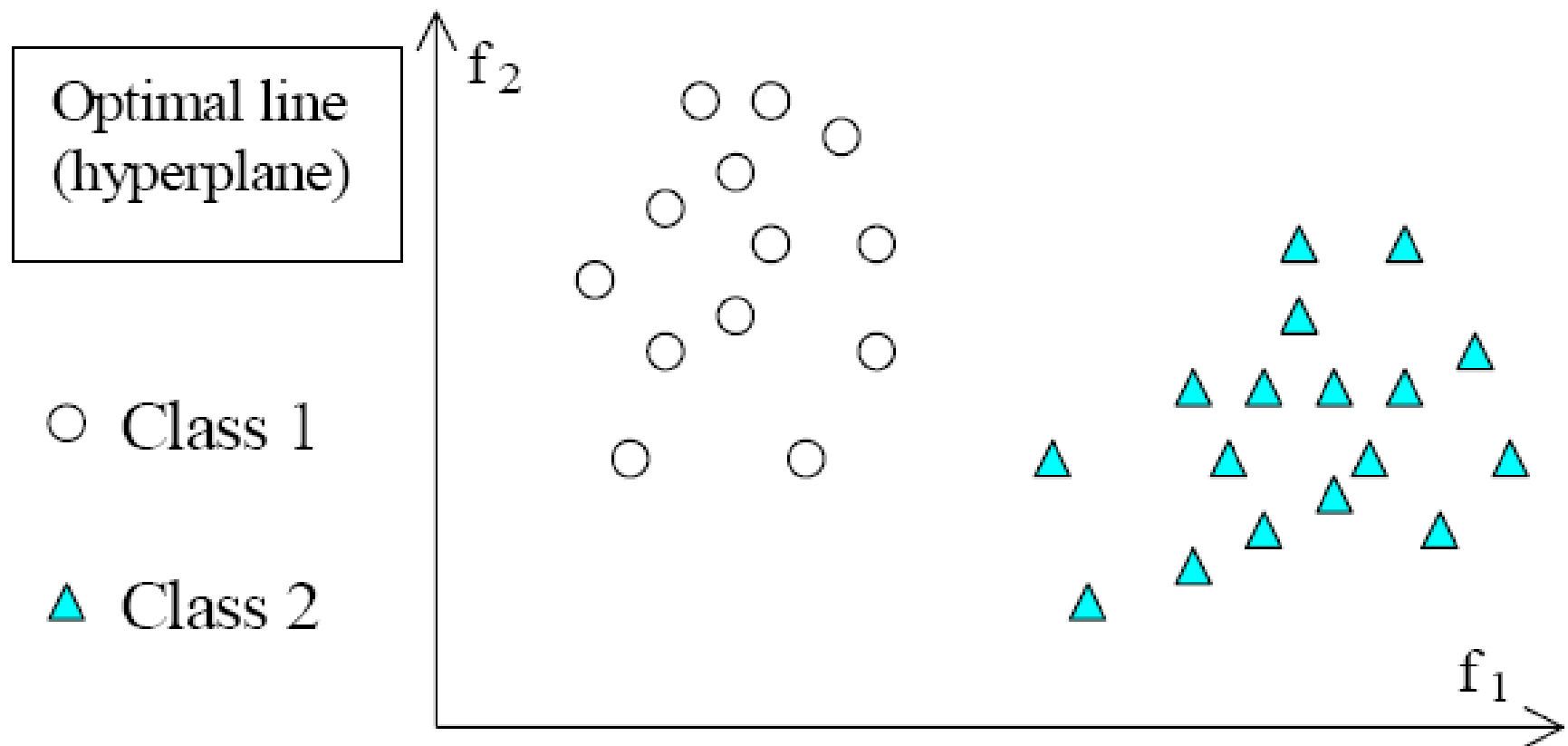
For some other cases, the two classes are separable but not linearly separable (eg in above example). Bayes classifier with 0-1 loss function and assuming gaussian distribution in the feature space yields in general quadratic decision boundaries.

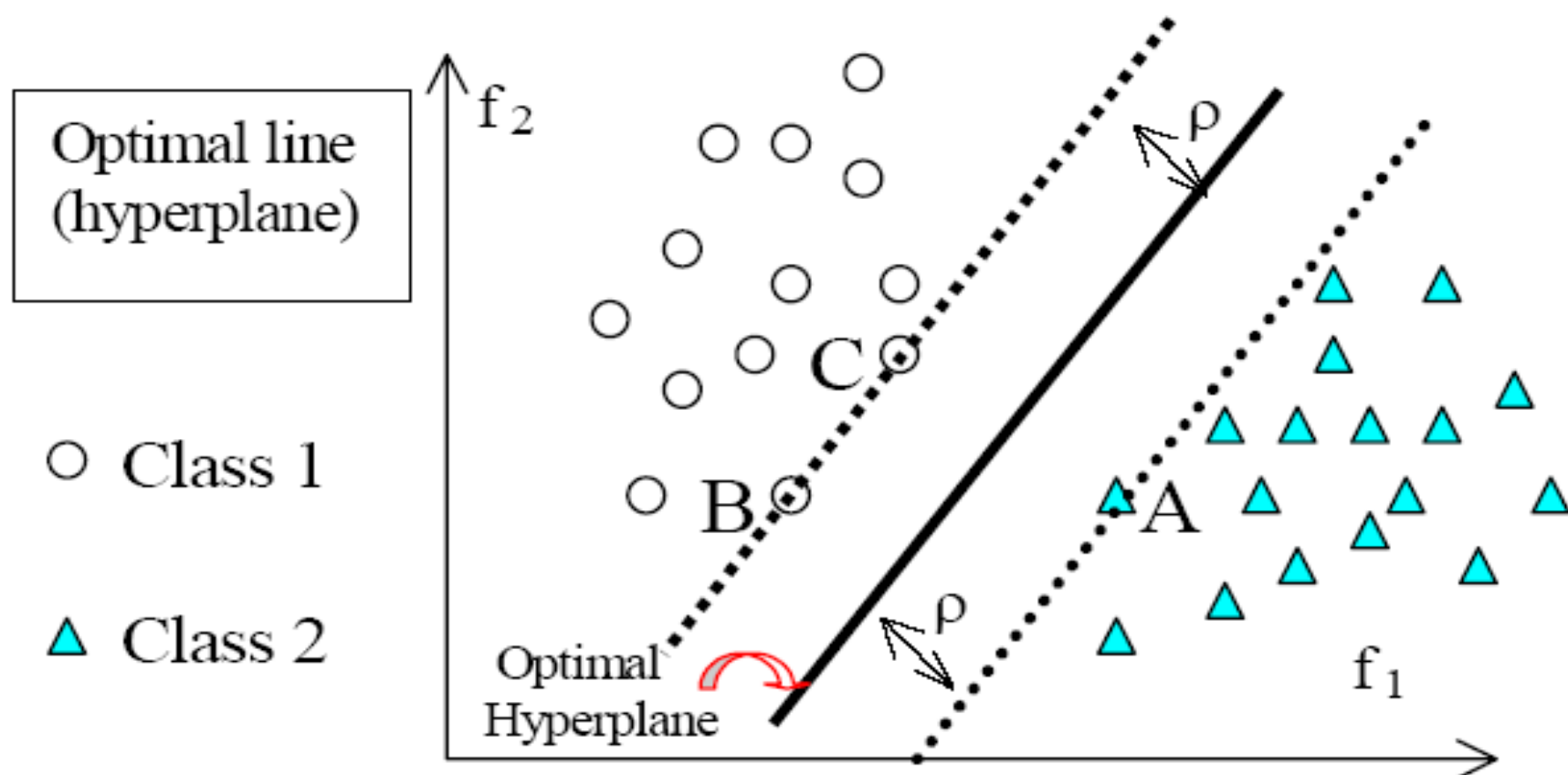
If two classes are linearly separable in the feature space, is there an “optimal” decision line (hyperplane) that separates the two classes for a given set of training samples for each class ?

Vector Support Machine - “the vector support machine is a linear machine that constructs a hyperplane as the decision surface in such a way that the margin of separation between one class examples and the other class examples is maximised.” - [Neural Networks by S Haykins]

Example : Given the two Classes 1 and 2 with training samples as shown.

Where is the optimal decision hyperplane ?





By observation, the **maximum separation** is ρ and the **optimal hyperplane** is as shown. A, B and C are called **support vectors**.

“Since the support vectors are those sample points that lie closest to the optimal hyperplane, they are therefore the most difficult to classify”.

Hence they have direct bearing on the optimum location of the decision surface.

[For practitioners of neural network implementations, the number of support vectors indicate the number of perceptrons needed. For further information of how to calculate this optimal hyperplane, refer to “Neural Networks” by Simon Haykins. Also note that one can relax the condition in that the two classes need not be separable.]

To handle non-linear separable situations, a support vector machine should have a preprocessing transformation to convert the initial feature space into an abstract feature space so that in the latter, the two classes are linearly separable. This transformation may not easily be found.

But some typical transformations are of the form:

$(\mathbf{x}^T \mathbf{x}_i)^p$ --- polynomial learning machine

$\exp(-\|\mathbf{x} - \mathbf{x}_i\|^2 / 2\sigma^2)$ -- radial-basis network

$\tanh(\beta \mathbf{x}^T \mathbf{x}_i + \gamma)$ --- two-layer perceptron

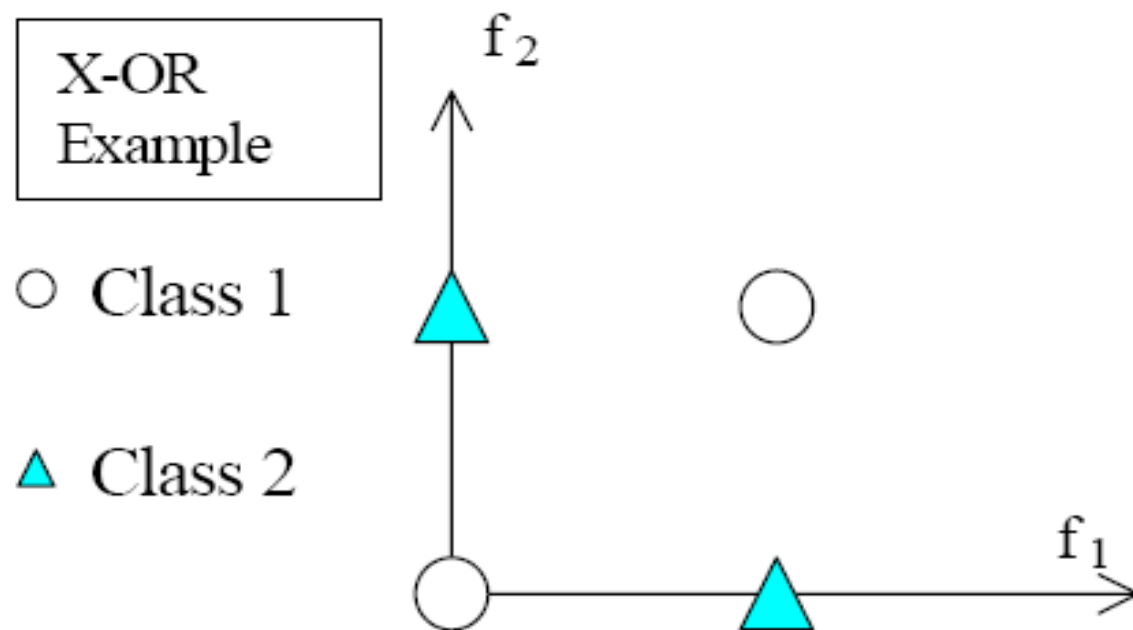
Cover's Theorem on the Separability of Patterns (1965)

“A complex pattern classification problem cast in a high-dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space”

Example 1 [The X-OR Problem]

Let f_1 and f_2 be the features. For class 1, $(0,0)^T$ and $(1,1)^T$ are the samples while $(1,0)^T$ and $(0,1)^T$ belong to class 2.

The two classes are obviously not linearly separable.



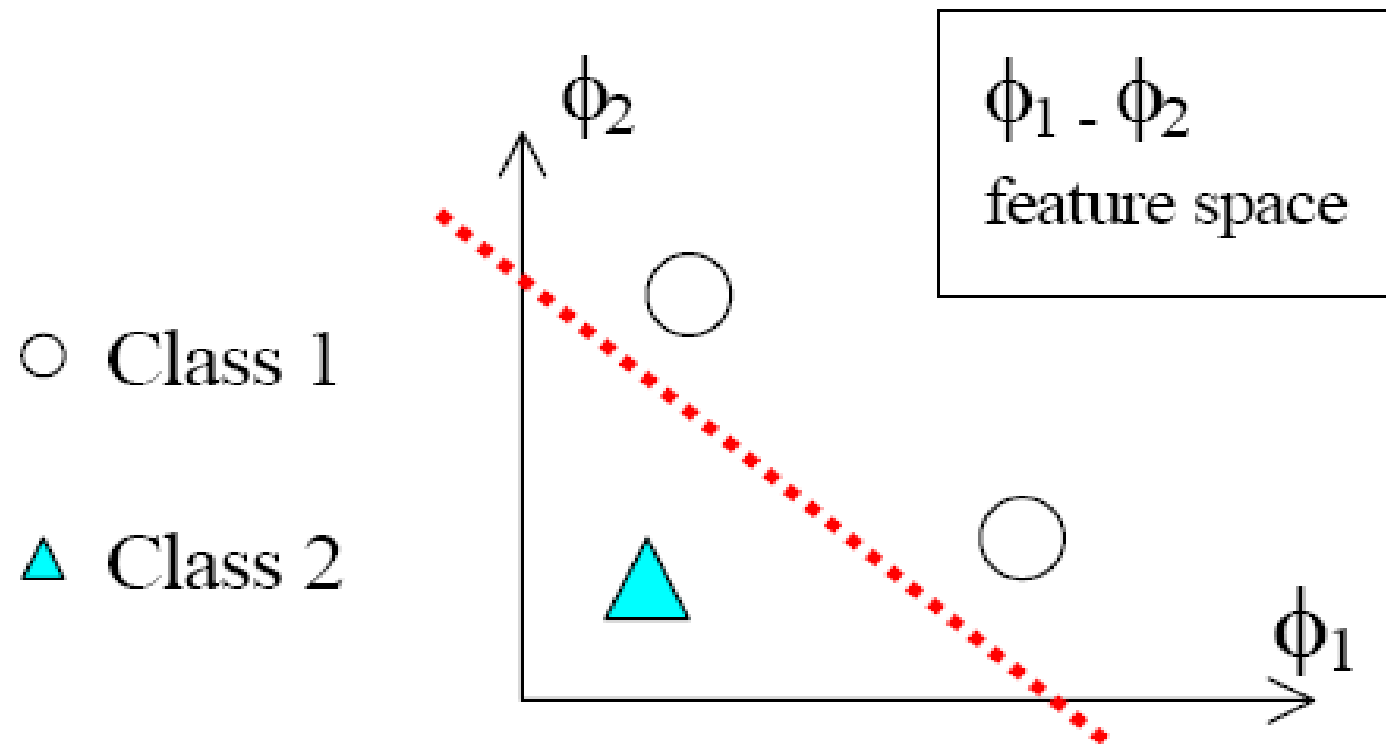
Now if we take the transformations defined by :

$$\phi_1(\mathbf{f}) = e^{-\|\mathbf{f} - \mathbf{t}_1\|^2} ; \quad \phi_2(\mathbf{f}) = e^{-\|\mathbf{f} - \mathbf{t}_2\|^2}$$

where $\mathbf{t}_1 = (1,1)$ and $\mathbf{t}_2 = (0,0)$.

<u>Input pattern \mathbf{f}</u>	<u>abs feature $\phi_1(\mathbf{f})$</u>	<u>abs feature $\phi_2(\mathbf{f})$</u>
(0,0)	0.1353	1
(1,1)	1	0.1353
(0,1)	0.3678	0.3678
(1,0)	0.3678	0.3678

Constructing the new feature space, yields :



In the new feature space, the two classes are now linearly separable.

Types of decision function:

Minimum Distance Classifier (MDC)

Example (Bean – red/grn; big/small):

$$\omega_1 : \mathbf{m}_1 = (x_1^{(1)}, x_2^{(1)})^T$$

$$\omega_2 : \mathbf{m}_2 = (x_1^{(2)}, x_2^{(2)})^T$$

$$\omega_3 : \mathbf{m}_3 = (x_1^{(3)}, x_2^{(3)})^T$$

$$\omega_4 : \mathbf{m}_4 = (x_1^{(4)}, x_2^{(4)})^T$$

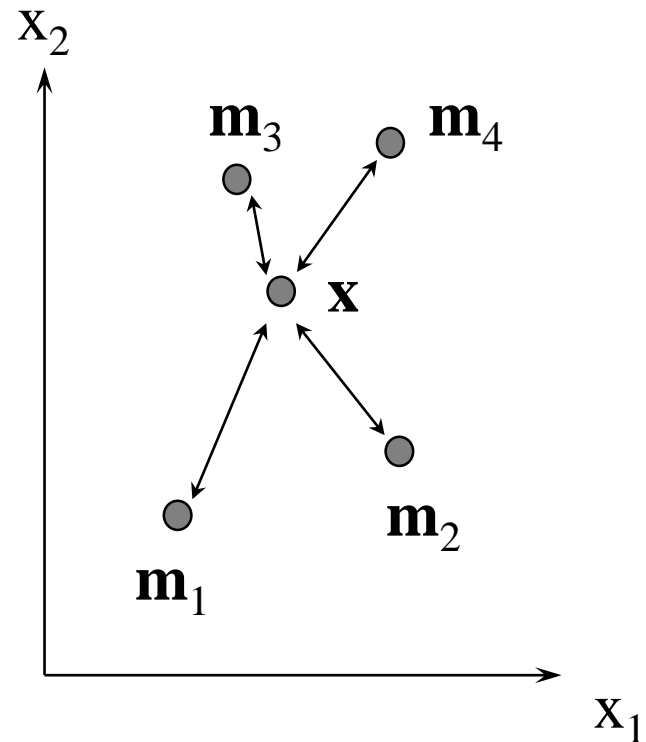
$$\|\mathbf{x} - \mathbf{m}_3\| < \|\mathbf{x} - \mathbf{m}_1\|$$

$$\|\mathbf{x} - \mathbf{m}_3\| < \|\mathbf{x} - \mathbf{m}_2\|$$

$$\|\mathbf{x} - \mathbf{m}_3\| < \|\mathbf{x} - \mathbf{m}_4\|$$

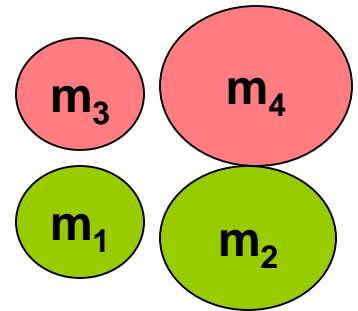
\mathbf{x} is closest to \mathbf{m}_3

Therefore, \mathbf{x} belongs to ω_3 .



- Each pattern class is represented by a prototype :

$$\mathbf{m}_j = \frac{1}{N_j} \sum_{\mathbf{x} \in \omega_j} \mathbf{x} \quad j = 1, 2, \dots, M$$



where \mathbf{m}_j is a mean vector for class ω_j ,
 N_j is the no. of pattern vectors from class ω_j .

- Given an unknown pattern vector \mathbf{x} , find the closest prototype by finding the **minimum Euclidean Distance**:

$$D_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{m}_j\| \quad j = 1, 2, \dots, M$$

where $\|a\| = [a^T a]^{1/2} =$ Euclidean norm of vector \mathbf{a}

$\therefore \mathbf{x} \rightarrow \omega_i$ if $D_i(\mathbf{x}) < D_j(\mathbf{x}) \quad j = 1, 2, \dots, M; j \neq i$

- Recognition is achieved by finding

$$d_i(\mathbf{x}) = \min_j \{D_j(\mathbf{x})\} \quad j = 1, 2, \dots, M;$$

- Decision function for MDC can be defined as

$$d_j(\mathbf{x}) = -D_j(\mathbf{x})$$

- Recognition is *then* achieved by finding

$$d_i(\mathbf{x}) = \max_j \{d_j(\mathbf{x})\} \quad j = 1, 2, \dots, M;$$

- By expanding $D_j(\mathbf{x})$ (on slide 22) and removing the common terms, $d_j(\mathbf{x})$ is simplified to

$$d_j(\mathbf{x}) = \mathbf{x}^T \mathbf{m}_j - \frac{1}{2} \mathbf{m}_j^T \mathbf{m}_j$$

How to simplify the decision function of MDC ?

$$D_j(\mathbf{x}) = (\mathbf{x} - \mathbf{m}_j)^T (\mathbf{x} - \mathbf{m}_j)$$

$$d_j(\mathbf{x}) = -(\mathbf{x} - \mathbf{m}_j)^T (\mathbf{x} - \mathbf{m}_j) \quad j = 1, \dots, M$$

$$= -(\mathbf{x}^T - \mathbf{m}_j^T)(\mathbf{x} - \mathbf{m}_j)$$

$$= -(\mathbf{x}^T \mathbf{x} - \mathbf{m}_j^T \mathbf{x} - \mathbf{x}^T \mathbf{m}_j + \mathbf{m}_j^T \mathbf{m}_j)$$

$$\because \mathbf{m}_j^T \mathbf{x} = \mathbf{x}^T \mathbf{m}_j$$

$$\therefore d_j(\mathbf{x}) = -(\mathbf{x}^T \mathbf{x} - 2\mathbf{x}^T \mathbf{m}_j + \mathbf{m}_j^T \mathbf{m}_j)$$

$\mathbf{x}^T \mathbf{x}$ is independent of j , i.e. a common term.

When $\max\{d_j(\mathbf{x})\}$ is sought, $\mathbf{x}^T \mathbf{x}$ can be dropped.

Therefore, dividing d_j by 2, we have

$$\therefore d_j(\mathbf{x}) = \mathbf{x}^T \mathbf{m}_j - \frac{1}{2} \mathbf{m}_j^T \mathbf{m}_j$$

Iris Recognition

versicolor $\omega_1 : \mathbf{m}_1 = (4.3, 1.3)^T$

setosa $\omega_2 : \mathbf{m}_2 = (1.5, 0.3)^T$

$$d_1(\mathbf{x}) = \mathbf{x}^T \mathbf{m}_1 - \frac{1}{2} \mathbf{m}_1^T \mathbf{m}_1 = 4.3x_1 + 1.3x_2 - 10.1$$

$$d_2(\mathbf{x}) = \mathbf{x}^T \mathbf{m}_2 - \frac{1}{2} \mathbf{m}_2^T \mathbf{m}_2 = 1.5x_1 + 0.3x_2 - 1.17$$

Rewrite in matrix form

$$\begin{pmatrix} 4.3 & 1.3 & -10.1 \\ 1.5 & 0.3 & -1.17 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}$$

If a petal has length 2 cm and width 0.5cm,

$$\text{i.e. } \mathbf{x} = (2, 0.5)^T$$

$$\begin{pmatrix} 4.3 & 1.3 & -10.1 \\ 1.5 & 0.3 & -1.17 \end{pmatrix} \begin{pmatrix} 2 \\ 0.5 \\ 1 \end{pmatrix} = \begin{pmatrix} -0.85 \\ 1.98 \end{pmatrix} \leftarrow$$

Therefore, it is Iris Setosa.

If a petal has length 5 cm and width 1.5cm,

$$\text{i.e. } \mathbf{x} = (5, 1.5)^T$$

$$\begin{pmatrix} 4.3 & 1.3 & -10.1 \\ 1.5 & 0.3 & -1.17 \end{pmatrix} \begin{pmatrix} 5 \\ 1.5 \\ 1 \end{pmatrix} = \begin{pmatrix} 13.35 \\ 6.78 \end{pmatrix} \leftarrow$$

Therefore, it is Iris Versicolor.

Decision boundary of MDC

- it is the line that separates two classes (for 2 Dimensional case).

Property of the decision boundary in MDC

the decision boundary is a perpendicular bisector of the line joining two class prototypes.

At the decision boundary (surface), $d_1(\mathbf{x}) = d_2(\mathbf{x})$

$$\therefore d_{12}(\mathbf{x}) = d_1(\mathbf{x}) - d_2(\mathbf{x}) = 2.8x_1 + 1.0x_2 - 8.9 = 0$$

for Iris classification,

$$\mathbf{x} \rightarrow \omega_1 \text{ if } d_{12}(\mathbf{x}) > 0$$

$$\mathbf{x} \rightarrow \omega_2 \text{ if } d_{12}(\mathbf{x}) < 0$$

Example:

$\mathbf{x} = (2, 0.5)^T$; $d_{12}((2, 0.5)^T) = -2.8 \rightarrow \omega_2$: Iris setosa

$\mathbf{x} = (5, 1.5)^T$; $d_{12}((5, 1.5)^T) = 6.6 \rightarrow \omega_1$: Iris versicolor

\therefore Decision boundary can be used for recognition

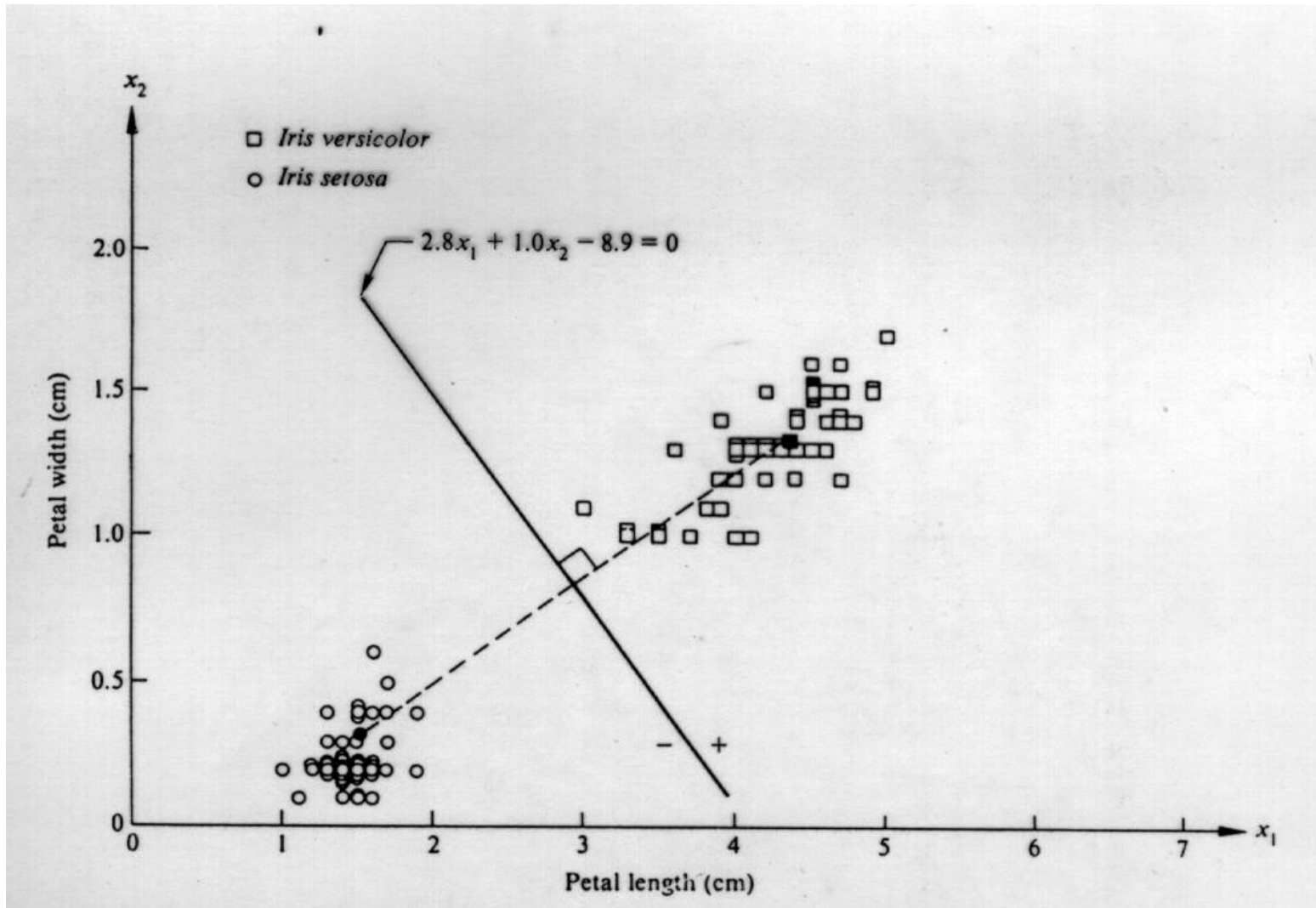
Definition of decision function:

Let the decision function be a line or surface in the pattern space separating **the 2-classes**.

Recognition is achieved by testing the sign of the decision function when presented with a pattern vector.

Example:

Classify Iris versicolor and Iris setosa by petal length, x_1 , and petal width, x_2 .



Use of Probability Theory in Recognition

- Probability theory is widely used in statistical pattern recognition.
- In particular, Bayes probability theory is used in formulating the optimal classifier (recognizer).
- Feature distributions (**derived from training data**) are used to form a priori **probability distributions** and they are represented as

$$p(\mathbf{x}|\omega_j) \quad \text{for } j = 1, 2, \dots, M.$$

- The above is also known as the likelihood function of class j object (**prior probability**).

Optimal Statistical Classifier - The Bayes Classifier

Recognition problem: given \mathbf{x} , find ω_i .

Let the probability of deciding \mathbf{x} belonging to ω_i be $p(\omega_i|\mathbf{x})$. (Given \mathbf{x} probability that it belongs to ω_i)

If \mathbf{x} should belong to ω_i but is wrongly classified to ω_j , the conditional average risk or conditional average loss is defined as

$$r_j(\mathbf{x}) = \sum_{k=1}^M L_{kj} p(\omega_k | \mathbf{x})$$

where L_{kj} is the loss incurred due to misclassification.

$r_j(\mathbf{x})$ = conditional average risk

- $p(\omega_i|\mathbf{x})$ is a *posterior* probability and is generally unknown.

From Bayes' rule,

$$p(\omega_k | \mathbf{x}) = \frac{p(x | \omega_k) p(\omega_k)}{p(\mathbf{x})}$$

$$\therefore r_j(\mathbf{x}) = \frac{1}{p(\mathbf{x})} \sum_{k=1}^M L_{kj} p(\mathbf{x} | \omega_k) p(\omega_k)$$

$p(\mathbf{x})$ is independent of j and is a common term,

$\therefore r_j$ can be reduced to

$$r_j(\mathbf{x}) = \sum_{k=1}^M L_{kj} p(\mathbf{x} | \omega_k) p(\omega_k)$$

$p(\mathbf{x}|\omega_k)$: probability density function of the patterns in each class;
 $p(\omega_k)$: probability of occurrence of each class.

- $p(\mathbf{x} | \omega_k)$ and $p(\omega_k)$ are both *a priori* probabilities that can be estimated from training set.
- Recognition is achieved by

$$\mathbf{x} \rightarrow \omega_i \quad \text{if} \quad r_i = \min_j \{r_j\} \quad \text{for } j = 1, 2, \dots M.$$

Decision function for Bayes classifier is

$$d_j(\mathbf{x}) = - \sum_{k=1}^M L_{kj} p(\mathbf{x} | \omega_k) p(\omega_k)$$

$$\therefore \mathbf{x} \rightarrow \omega_i \quad \text{if} \quad d_i = \max_j \{d_j\} \quad \text{for } j = 1, 2, \dots M.$$

Example illustrating the concept of Bayes classifier

For a 2-class problem, show how a feature \mathbf{x} coming from class 2 can be correctly classified using the Bayes classifier. Assume that patterns from both classes are equally likely to be seen. For correct classification, there should be no penalty. For incorrect classification, the penalty is $L_{ij}=1, i \neq j$.

From above, we know : $M = 2$, $p(\omega_1) = p(\omega_2)$, and $\mathbf{x} \in \omega_2$
Also, $L_{11} = L_{22} = 0$, $L_{12}=L_{21}=1$.

$$r_1 = L_{11} p(\mathbf{x} | \omega_1) p(\omega_1) + L_{21} p(\mathbf{x} | \omega_2) p(\omega_2)$$

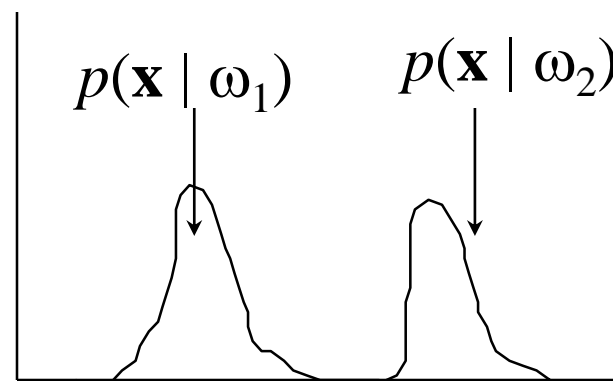
$$r_2 = L_{12} p(\mathbf{x} | \omega_1) p(\omega_1) + L_{22} p(\mathbf{x} | \omega_2) p(\omega_2)$$

Since $p(\omega_1) = p(\omega_2)$, and $L_{11} = L_{22} = 0$, $L_{12}=L_{21}=1$.

$$\begin{aligned}\therefore r_1 &= p(\mathbf{x} \mid \omega_2) & r_2 &= p(\mathbf{x} \mid \omega_1) \\ d_1 &= -p(\mathbf{x} \mid \omega_2) & d_2 &= -p(\mathbf{x} \mid \omega_1)\end{aligned}$$

$$\begin{aligned}\text{If } \mathbf{x} \in \omega_2 & \quad p(\mathbf{x} \mid \omega_2) > p(\mathbf{x} \mid \omega_1) \\ \therefore & \quad r_1 > r_2\end{aligned}$$

$$\begin{aligned}\text{If } \mathbf{x} \in \omega_2 & \quad -p(\mathbf{x} \mid \omega_2) < -p(\mathbf{x} \mid \omega_1) \\ \therefore & \quad d_1 < d_2\end{aligned}$$



Bayes classifier with 0-1 loss function.

- In the **penalty** approach,

$L_{ii} = 0$ for correct decision, i.e. $i = j$.

$L_{ij} = 1$ for incorrect decision, i.e. $i \neq j$.

- In the **reward** approach,

$\delta_{ii} = 1$ for correct decision, i.e. $i = j$.

$\delta_{ij} = 0$ for incorrect decision. i.e. $i \neq j$.

Let $L_{ij} = 1 - \delta_{ij}$ and $\delta_{ij} = 1$ when $i = j$, $\delta_{ij} = 0$ when $i \neq j$.

$$d_j(\mathbf{x}) = - \sum_{k=1}^M (1 - \delta_{kj}) p(\mathbf{x} | \omega_k) p(\omega_k) \quad \leftarrow \text{Penalty approach. From slide 33}$$

The summation can be separated into 2 parts:

$$d_j(\mathbf{x}) = - \sum_{k=1}^M p(\mathbf{x} | \omega_k) p(\omega_k) + \sum_{k=1}^M \delta_{kj} p(\mathbf{x} | \omega_k) p(\omega_k)$$

But,

$$p(\mathbf{x}) = \sum_{k=1}^M p(\mathbf{x} | \omega_k) p(\omega_k)$$

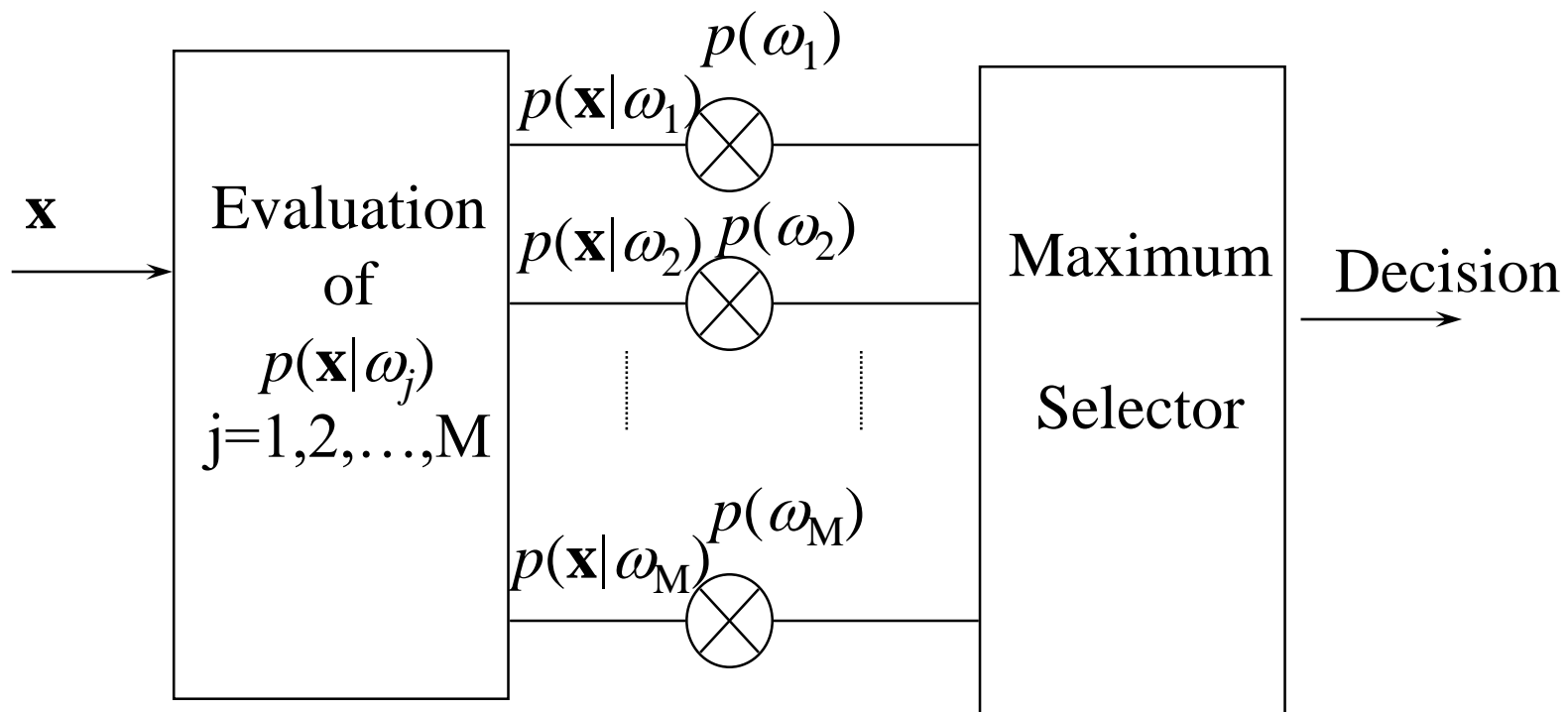
$$\therefore d_j(\mathbf{x}) = -p(\mathbf{x}) + \sum_{k=1}^M \delta_{kj} p(\mathbf{x} | \omega_k) p(\omega_k)$$

Since $-p(\mathbf{x})$ is constant and $\delta_{ij} = 0$ except when $k = j$,

$$\boxed{d_j(\mathbf{x}) = p(\mathbf{x} | \omega_j) p(\omega_j) , \quad j = 1, 2, \dots, M}$$

$$\therefore \mathbf{x} \rightarrow \omega_i \quad \text{if} \quad d_i = \max_j \{d_j(\mathbf{x})\}, \quad j = 1, 2, \dots, M$$

A Bayes Classifier with 0-1 loss



Bayes classifier with 0-1 loss for Gaussian patterns

A pattern class with a Gaussian distribution of its feature values can be represented by

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2\right]$$

Gaussian probability density function

where $m = E\{x\}$

$$\sigma^2 = E\{(x-m)^2\}$$

$E\{.\}$ denotes the expected values.

A 2-class example of Gaussian patterns

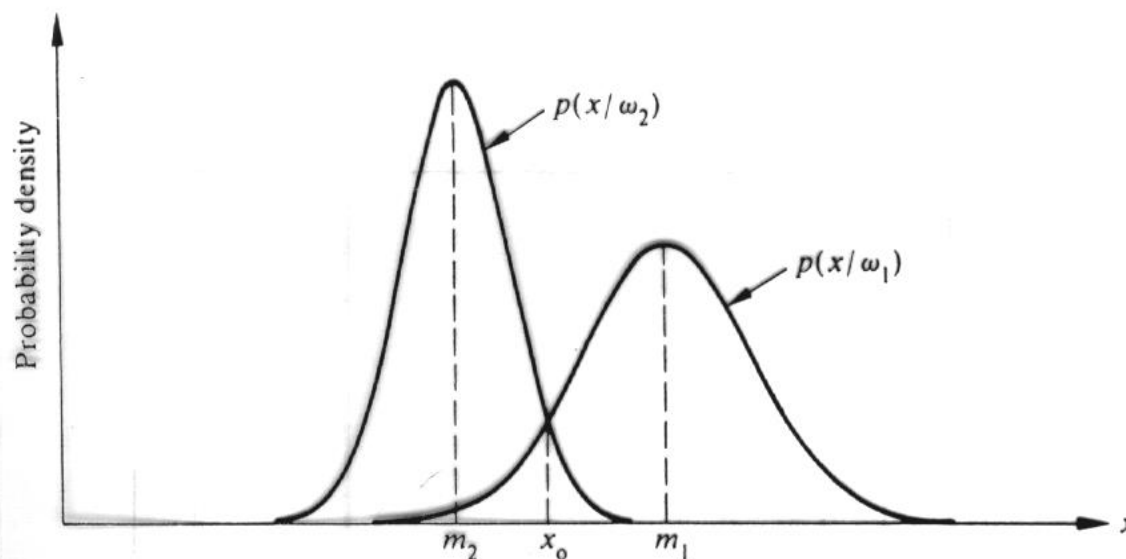
Suppose only one feature is used to classify two patterns.

Let x be the feature, the Bayes decision function is

$$d_j(x) = p(x | \omega_j) p(\omega_j) , \quad j = 1, 2$$

Therefore,

$$d_j(x) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left\{-\frac{1}{2}\left(\frac{x - m_j}{\sigma_j}\right)^2\right\} p(\omega_j) , \quad j = 1, 2$$



Probability density functions for two 1-D pattern classes. The point x_0 is the decision boundary if the two classes are equally likely to occur.

At the decision boundary , $d_1(x_0)=d_2(x_0)$

If $p(\omega_2) = p(\omega_1)$, $p(x_0 | \omega_1) = p(x_0 | \omega_2)$.

If $p(\omega_1) > p(\omega_2)$, x_0 will be shifted to the left.

If $p(\omega_1) < p(\omega_2)$, x_0 will be shifted to the right.

For **n dimensional feature vectors**, Gaussian distribution is represented by

$$p(\mathbf{x} | \omega_j) = \frac{1}{(2\pi)^{n/2} |\mathbf{C}_j|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \mathbf{m}_j)^T \mathbf{C}_j^{-1} (\mathbf{x} - \mathbf{m}_j) \right]$$

where \mathbf{m}_j is the mean vector of class j , $= E_j \{ \mathbf{x} \}$

\mathbf{C}_j is the covariance matrix of class j , $= E_j \{ (\mathbf{x} - \mathbf{m}_j)(\mathbf{x} - \mathbf{m}_j)^T \}$

$|\mathbf{C}_j|$ is the determinant of \mathbf{C}_j ,

n is the dimension of feature vector.

Approximating the Expected value, $E\{\bullet\}$, with its average value:

$$\mathbf{m}_j = \frac{1}{N_j} \sum_{\mathbf{x} \in \omega_j} \mathbf{x} \quad \mathbf{C}_j = \frac{1}{N_j} \sum_{\mathbf{x} \in \omega_j} \mathbf{x}\mathbf{x}^T - \mathbf{m}_j \mathbf{m}_j^T$$

The Bayes decision function for Gaussian patterns is:

$$d_j(\mathbf{x}) = \frac{1}{(2\pi)^{n/2} |\mathbf{C}_j|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \mathbf{m}_j)^T \mathbf{C}_j^{-1}(\mathbf{x} - \mathbf{m}_j)\right] p(\omega_j)$$

$$d'_j(\mathbf{x}) = \ln d_j(\mathbf{x})$$

$$d'_j(\mathbf{x}) = -\frac{n}{2} \ln(2\pi) - \frac{1}{2} \ln |\mathbf{C}_j| - \frac{1}{2} [(\mathbf{x} - \mathbf{m}_j)^T \mathbf{C}_j^{-1}(\mathbf{x} - \mathbf{m}_j)] + \ln p(\omega_j)$$

Remove the constant term, $d'_j(\mathbf{x})$ can be redefined as

$$d'_j(\mathbf{x}) = -\frac{1}{2} [(\mathbf{x} - \mathbf{m}_j)^T \mathbf{C}_j^{-1}(\mathbf{x} - \mathbf{m}_j)] + \left[\ln p(\omega_j) - \frac{1}{2} \ln |\mathbf{C}_j| \right]$$

This is the general form of Bayes classifier for Gaussian pattern classes with 0-1 loss function

The Covariance matrix \mathbf{C}_j

\mathbf{C}_j is a symmetric square matrix of n by n .

$$\mathbf{C}_j = E_j \{ (\mathbf{x} - \mathbf{m}_j)(\mathbf{x} - \mathbf{m}_j)^T \}$$

$$\mathbf{C}_j = E_j \left\{ \begin{bmatrix} x_1 - m_1 \\ x_2 - m_2 \\ \vdots \\ x_n - m_n \end{bmatrix} \begin{bmatrix} x_1 - m_1 & x_2 - m_2 & \cdot & \cdot & \cdot & x_n - m_n \end{bmatrix} \right\}$$

$$\mathbf{C}_j = \begin{bmatrix} E\{(x_1 - m_1)(x_1 - m_1)\} & E\{(x_1 - m_1)(x_2 - m_2)\} & \dots & E\{(x_1 - m_1)(x_n - m_n)\} \\ E\{(x_2 - m_2)(x_1 - m_1)\} & E\{(x_2 - m_2)(x_2 - m_2)\} & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & & & \cdot \\ E\{(x_n - m_n)(x_1 - m_1)\} & \dots & \dots & E\{(x_n - m_n)(x_n - m_n)\} \end{bmatrix}_j$$

$$\mathbf{C}_j = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & & & \cdot \\ \cdot & & \cdot & \cdot \\ c_{n1} & \dots & \dots & c_{nn} \end{bmatrix}_j$$

- c_{ii} is the variance
- c_{ij} is the covariance; $i \neq j$.

- Covariance matrix can be estimated by expanding the original definition and simplifying by using averages to:

$$\mathbf{C}_j = E_j[\mathbf{xx}^T] - \mathbf{m}_j \mathbf{m}_j^T = \left(\frac{1}{N_j} \sum_{\mathbf{x} \in \omega_j} \mathbf{xx}^T \right) - \mathbf{m}_j \mathbf{m}_j^T$$

- For features that are statistically independent (uncorrelated), \mathbf{C}_j is a diagonal matrix. ($c_{ij} = 0$)

If all covariance matrices are equal and are identity matrices and $P(\omega_j) = 1/M$ for all $j = 1, 2, \dots, M$, then:

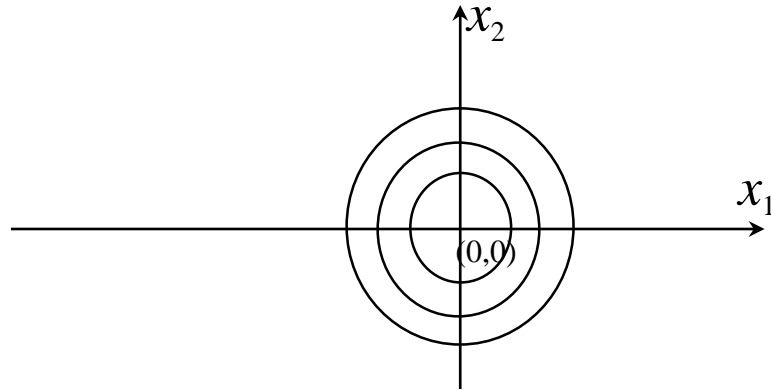
$$\therefore d_j(\mathbf{x}) = \mathbf{x}^T \mathbf{m}_j - \frac{1}{2} \mathbf{m}_j^T \mathbf{m}_j = \text{MDC (Minimum Distance Classifier)}$$

Thus, MDC is optimal in the Bayes sense if:

1. Pattern classes are Gaussian
2. All covariances are equal to identity matrix
3. All classes are equally likely to occur

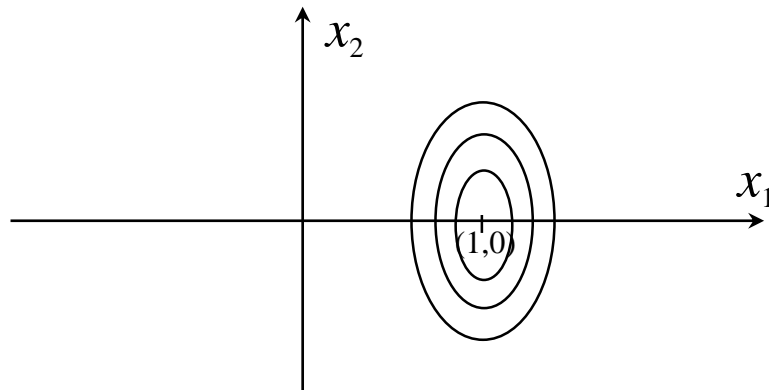
Special cases of the covariance matrix (2D examples):

(a) assume $\mathbf{m} = 0$, $\mathbf{C} = \sigma^2 \mathbf{I}$



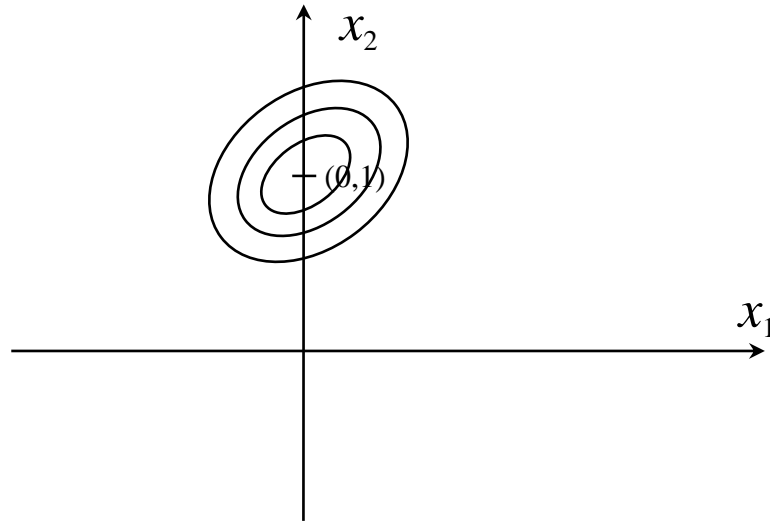
(b) assume $\mathbf{m} = [1, 0]^T$, $c_{22} > c_{11}$

$$\mathbf{C} = \begin{bmatrix} c_{11} & 0 \\ 0 & c_{22} \end{bmatrix}$$



(c) assume $\mathbf{m} = [0, 1]^T$,

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$



k-Nearest Neighbour

- We are concerned with the following problem: we wish to label some observed pattern x with some class category θ .
- Two possible situations with respect to x and θ may occur:
 - We may have complete statistical knowledge of the distribution of observation x and category θ . In this case, a standard Bayes analysis yields an optimal decision procedure.
 - We may have no knowledge of the distribution of observation x and category θ aside from that provided by pre-classified samples. In this case, a decision to classify x into category θ will depend only on a collection of correctly classified samples.
- The nearest neighbor rule is concerned with the latter case. Such problems are classified in the domain of non-parametric statistics. No optimal classification procedure exists with respect to all underlying statistics under such conditions.

The NN Rule

- Recall that the only knowledge we have is a set of i points of x correctly classified into categories θ : (x_i, θ_i) . By intuition, it is reasonable to assume that observations that are close together -- for some appropriate metric -- will have the same classification.
- Thus, when classifying an unknown sample x , it seems appropriate to weight the evidence of the nearby x_i 's heavily.
- One simple non-parametric decision procedure of this form is the nearest neighbor rule or NN-rule. This rule classifies x in the category of its nearest neighbor. More precisely, we call

$$x_m \in (x_1, x_2, x_3, \dots, x_n)$$

x_m is a nearest neighbor to x if

$$\min d(x_i, x) = d(x_m, x) \quad \text{where } i = 1, 2, \dots, n.$$

The nearest neighbor rule chooses to classify x to the category θ_m , where x_m is the nearest neighbor to x and belongs to class θ_m . A mistake is made if θ_m is not the same as θ which is the true class that x belongs to. Also, notice that the NN-rule only uses the nearest neighbor x_m as a classifier, while ignoring the remaining pre-labeled data points.

Figure 1: The NN rule

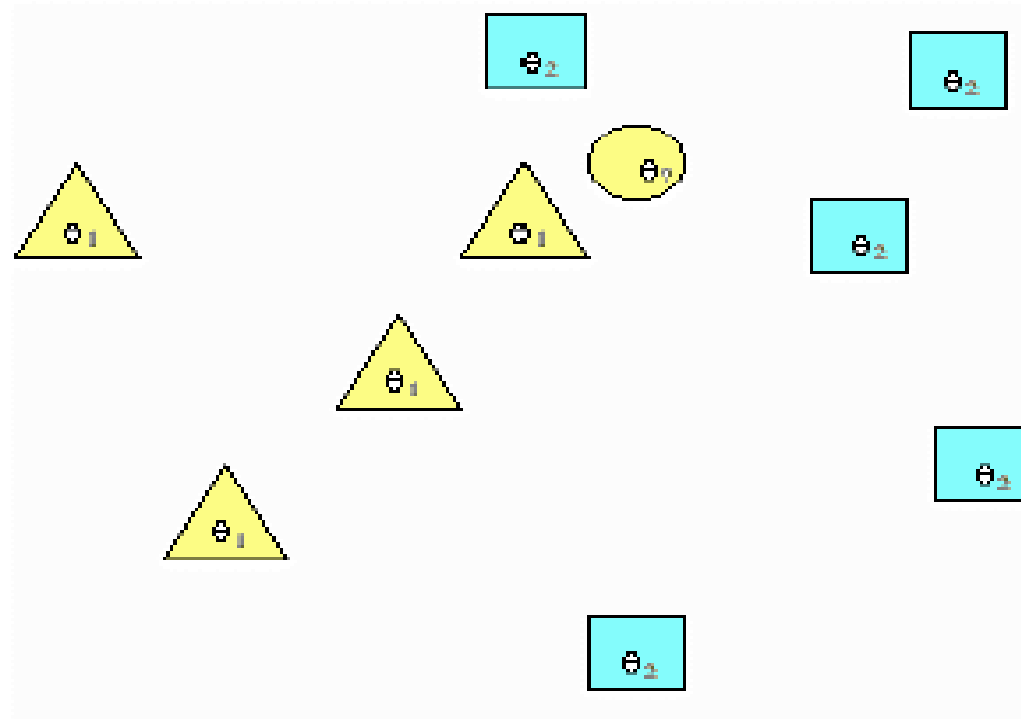


Figure 1 shows an example of the NN rule. In this problem, there are two classes: (yellow triangles) and (blue squares). The circle represents the unknown sample x and as its nearest neighbor comes from class x_1^1 , it is labeled as class θ_1 .

The k-NN Rule

If the number of pre-classified points is large it makes good sense to use, instead of the single nearest neighbor, the majority vote of the nearest k neighbors. This method is referred to as the k -NN rule.

The number k should be:

- 1) large to minimize the probability of misclassifying x .
- 2) small (with respect to the number of samples) so that the points are close enough to x to give an accurate estimate of the true class of x .

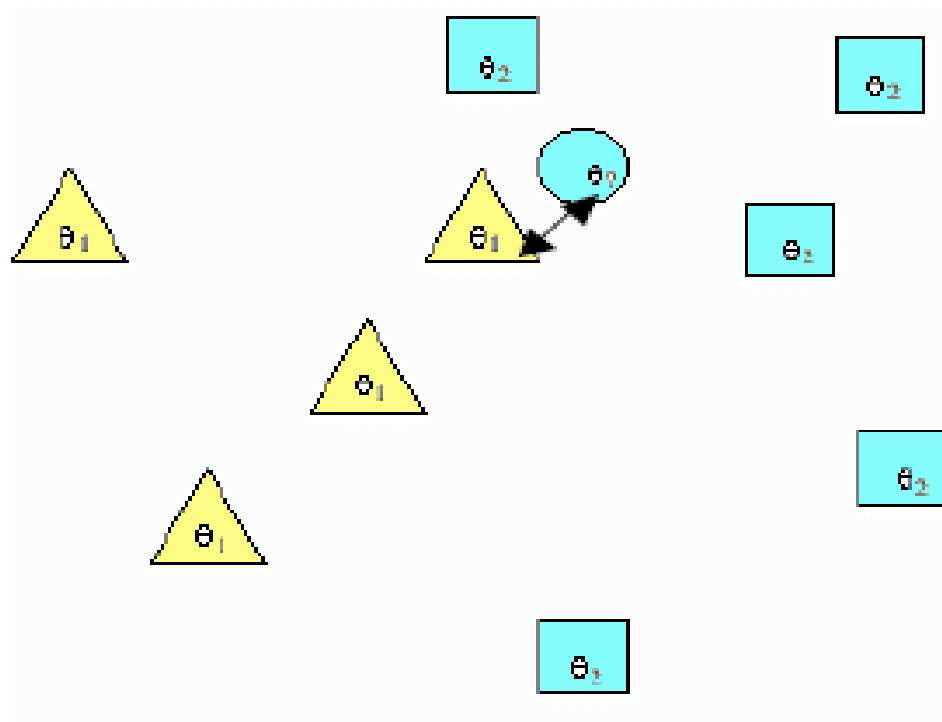


Figure 2: The k-NN rule with $k=3$

Figure 2 shows an example of the k-NN rule, with $k=3$. As before, there are two classes: θ_1 (yellow triangles) and θ_2 (blue squares). The circle represents the unknown sample x and as two of its nearest neighbors come from class θ_2 , it is labeled class θ_2 .

Fisher's Linear Discriminant Analysis

Figure 1 summarizes the two major approaches --- decision boundaries and projection subspaces.

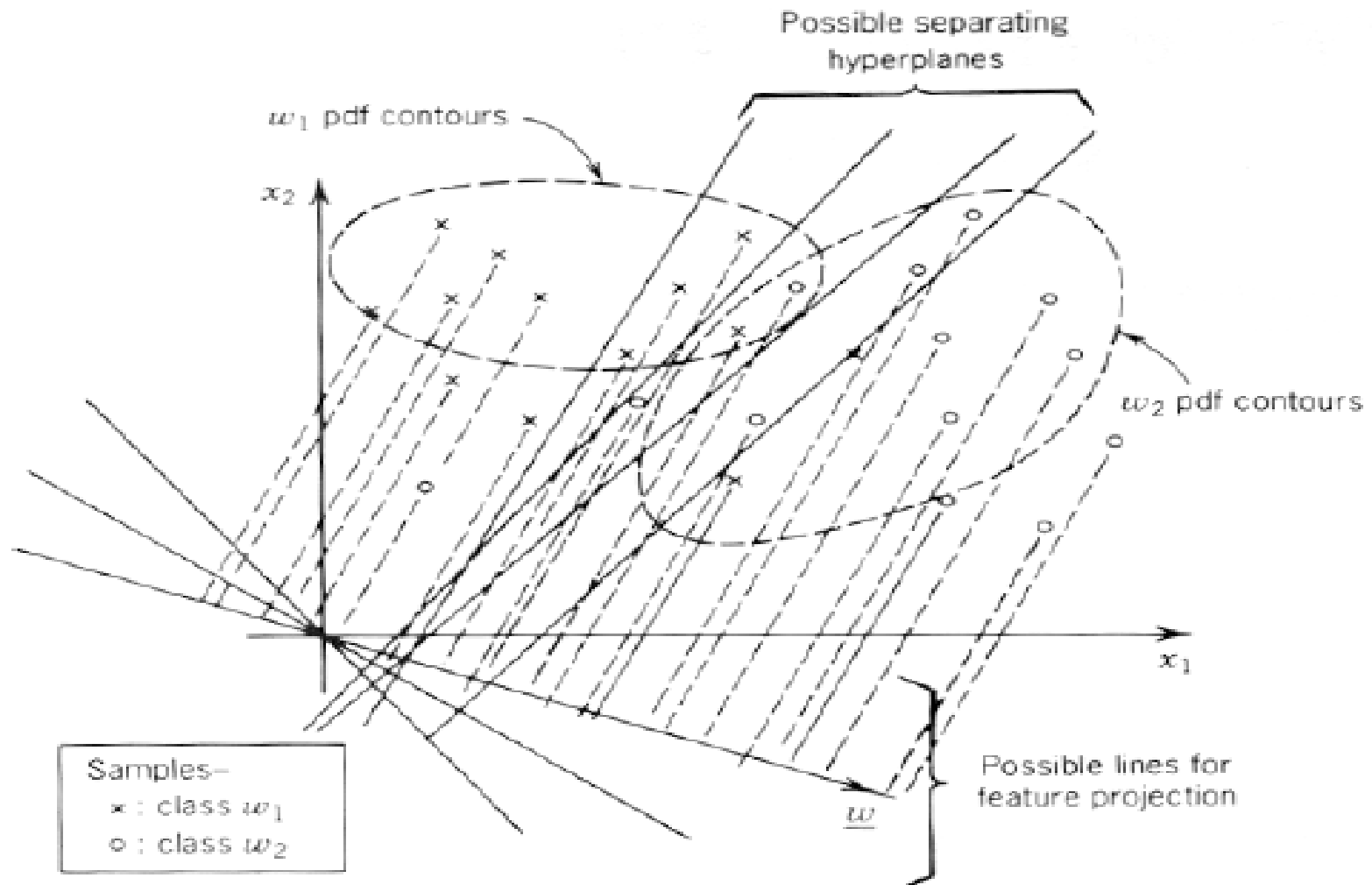


Figure 1 Two Classification Paradigms.

Fisher's Linear Discriminant

The Fisher approach [R. A. Fisher 1936] is based on *projection* of d -dimensional data onto a *line*. The hope is that these projections onto a linear will be well *separated by class*. Thus, the line is oriented to maximize this class separation. The real value of this approach is when the size of the feature vector is very large, for example $d = 10000$ or larger. We begin with a $c = 2$ class example. The given training set

$$H = \{\underline{x}_1, \underline{x}_2, \dots, \underline{x}_n\} = \{H_1, H_2\} \quad (4.2)$$

is partitioned into $n_1 \leq n$ training vectors in subset H_1 , corresponding to class w_1 , and $n_2 \leq n$ training vectors in set H_2 , corresponding to class w_2 , where $n_1 + n_2 = n$. The feature vector projections are formed via

$$y_i = \underline{w}^T \underline{x}_i = \langle \underline{w}, \underline{x}_i \rangle \quad \text{for } i = 1, 2, \dots, n \quad (4-3)$$

If we further constrain $\| \underline{w} \| = 1$, each y_i is the projection of \underline{x}_i onto a line *in the direction of* \underline{w} . Note that this line always goes through the origin in R^d . The problem is to choose the direction of \underline{w} , given H , such that y_i from H_1 and y_i from H_2 fall into (ideally) distinct clusters along the line, denoted Y_1 and Y_2 . Ideally, if $y_i \in Y_1$, then $\underline{x}_i \in H_1$ and if $y_i \in Y_2$, then $\underline{x}_i \in H_2$.

Measures of Projected Data Class Separation. One measure of separation of the projections is the *difference of the means of the projections*. For example, $|\mu_{Y_1} - \mu_{Y_2}|^2$ is such a measure, where, using (4-3)

$$\mu_{Y_i} = E\{y_i \mid \underline{x}_i \in w_i\} = E\{\underline{w}^T \underline{x}_i \mid \underline{x}_i \in w_i\} \quad (4-4)$$

This measure may be shown to be related to the H_1, H_2 sample means through \underline{w} :

$$\underline{m}_i = \frac{1}{n_i} \sum_{x_i \in H_i} x_i \quad (4-5a)$$

Thus, the *projection mean* for each class is a scalar, given by

$$\begin{aligned} \bar{m}_i &= \frac{1}{n_i} \sum_{x_i \in H_i} \underline{w}^T x_i = \frac{1}{n_i} \sum_{y_i \in \underline{y}_i} y_i \\ &= \underline{w}^T \frac{1}{n_i} \sum_{x_i \in H_i} x_i = \underline{w}^T \underline{m}_i \end{aligned} \quad (4.5b)$$

where m_i is the sample mean of the vectors in H_i . The difference of the projection means using sample data is therefore

$$|\underline{\bar{m}}_1 - \underline{\bar{m}}_2| = |\underline{w}^T(\underline{m}_1 - \underline{m}_2)| \quad (4-5c)$$

The difference between the means of the projected data alone is insufficient for a good classifier, as is shown in Figure 2. Although we want well-separated (class) projections, *they should not be intermingled*. To achieve this, we need to consider variances of y_i in Y_i *relative* to the means. Therefore, a **better class separation measure** is the ratio **(difference of means)/(variance of within-class data)**. For example, a reasonable criterion in the $c = 2$ case is

$$J(w) = \frac{(\mu_{r_1} - \mu_{r_2})^2}{\sigma_{r_1}^2 + \sigma_{r_2}^2} \quad (4-6a)$$

or, in the case of sample data,

$$J(\underline{w}) = \frac{(\underline{\bar{m}}_1 - \underline{\bar{m}}_2)^2}{\hat{\sigma}_1^2 + \hat{\sigma}_2^2} \quad (4-6b)$$

where σ_i^2 is a measure of within-class scatter of the projected data. Instead of using variances in (4-6b), we could estimate the within-class scatter of the projection data by

$$\bar{s}_i^2 = \sum_{y \in \mathcal{Y}_i} (y - \bar{m}_i)^2 \quad (4-7a)$$

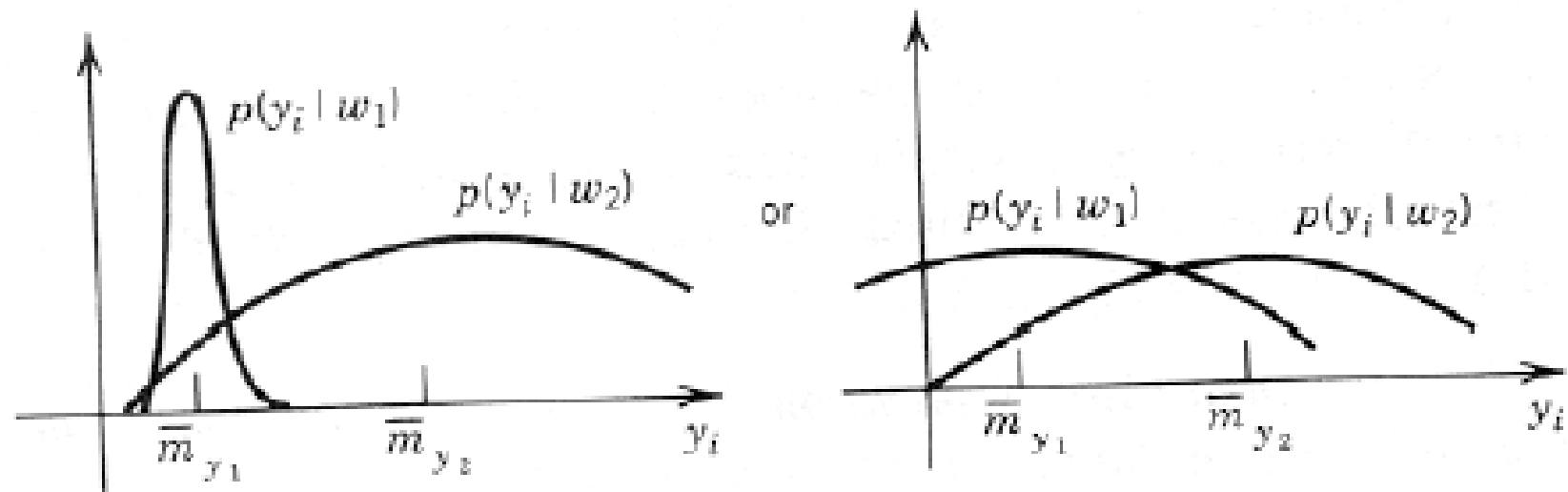


Figure 2 The importance of defining good separation measures for projected data.

Fisher showed that a reasonable measure of projected data separability is the criterion function that is a scaled version of (4-6b):

$$J(\underline{w}) = \frac{(\overline{m}_1 - \overline{m}_2)^2}{\overline{s}_1^2 + \overline{s}_2^2} \quad (4-8)$$

The value of \underline{w} that maximizes (4-6a), (4-6b), or (4-8) is used in a linear discriminant function (of the form $\underline{w}^T \underline{x}$) to yield *Fisher's linear discriminant*. To proceed further, it is necessary to rewrite J in (4-8) as an *explicit function* of \underline{w} . Notice \underline{w} influences *both* the numerator and the denominator of (4-8), since both m_i and s_i are functions of y_i .

We therefore have created an optimization problem, that is, to determine the direction of \underline{w} such that the criterion of (4-8) is maximum.

Solution for Criteria of (4-6a) (when Exact Means and Covariances are known). Denoting the mean and covariance of the d -dimensional vectors in class w_i as $\underline{\mu}_i$ and $\underline{\Sigma}_i$ respectively, it is straightforward to show

$$\mu_{y_i} = \underline{w}^T \underline{\mu}_i \quad \text{for } i = 1, 2 \quad (4-9a)$$

and

$$\sigma_{y_i}^2 = \underline{w}^T \underline{\Sigma}_i \underline{w} \quad \text{for } i = 1, 2 \quad (4-9b)$$

A maximum of (4-6a) is found by setting $\partial J / \partial \underline{w} = 0$. This requires

$$\frac{\partial J}{\partial \underline{w}} = \frac{\partial J}{\partial \mu_{y_1}} \frac{\partial \mu_{y_1}}{\partial \underline{w}} + \frac{\partial J}{\partial \mu_{y_2}} \frac{\partial \mu_{y_2}}{\partial \underline{w}} + \frac{\partial J}{\partial \sigma_{y_1}^2} \frac{\partial \sigma_{y_1}^2}{\partial \underline{w}} + \frac{\partial J}{\partial \sigma_{y_2}^2} \frac{\partial \sigma_{y_2}^2}{\partial \underline{w}} = 0 \quad (4-9c)$$

Equation 4-9c together with (4-9a) and (4-9b) yields the constraint on the optimal \underline{w} , denoted $\hat{\underline{w}}$:

$$\frac{2(\mu_{r_1} - \mu_{r_2})}{\sigma_{r_1}^2 + \sigma_{r_2}^2}(\underline{\mu}_1 - \underline{\mu}_2) - \frac{2(\mu_{r_1} - \mu_{r_2})^2}{(\sigma_{r_1}^2 + \sigma_{r_2}^2)^2}(\Sigma_1 + \Sigma_2)\hat{\underline{w}} = \underline{0} \quad (4-9d)$$

or

$$\hat{\underline{w}} = k(\Sigma_1 + \Sigma_2)^{-1}(\underline{\mu}_1 - \underline{\mu}_2) \quad (4-9e)$$

where

$$k = \frac{\sigma_{r_1}^2 + \sigma_{r_2}^2}{\mu_{r_1} - \mu_{r_2}} \quad (4-9f)$$

is merely a scale term that affects only $\|\underline{w}\|$. Equation 4-9e is intuitively pleasing and bears a striking similarity to Bayesian results in the Gaussian case. Therefore, by estimating Σ_i and $\underline{\mu}_i$ from H_i , the direction of $\hat{\underline{w}}$ may be determined.

Solution for Criterion of (4-8), Based on Sample Data. An alternative procedure is now shown. Defining a *scatter matrix* S_i as

$$S_i = \sum_{\underline{x} \in H_i} (\underline{x} - \underline{m}_i)(\underline{x} - \underline{m}_i)^T \quad \text{for } i = 1, 2 \quad (4-10a)$$

and

$$S_W = S_1 + S_2 \quad (4-10b)$$

the denominator of (4-8) may be formulated as

$$\bar{s}_1^2 + \bar{s}_2^2 = \underline{w}^T S_W \underline{w} \quad (4-10c)$$

Similarly, the numerator (4-8), using (4-5c), may be rewritten in terms of the sample means as

$$(\overline{m}_1 - \overline{m}_2)^2 = \underline{w}^T (\underline{m}_1 - \underline{m}_2)(\underline{m}_1 - \underline{m}_2)^T \underline{w} = \underline{w}^T S_B \underline{w} \quad (4-10d)$$

where S_B is the *between-class scatter matrix*. Since S_B is the outer product of a vector with itself, it has rank one. Therefore, (4-8) becomes

$$J(\underline{w}) = \frac{\underline{w}^T S_B \underline{w}}{\underline{w}^T S_W \underline{w}} \quad (4-11a)$$

where the sample data in H_1 and H_2 are used to determine S_W and S_B . Forming $\partial J / \partial \underline{w} = \underline{0}$ leads to

$$S_W \hat{\underline{w}} (\hat{\underline{w}}^T S_B \hat{\underline{w}}) (\hat{\underline{w}}^T S_W \hat{\underline{w}})^{-1} = S_B \hat{\underline{w}} \quad (4-11b)$$

which yields a *generalized eigenvector problem*, where the scalar term

$$\left(\underline{\hat{w}}^T S_B \underline{\hat{w}}\right) \left(\underline{\hat{w}}^T S_W \underline{\hat{w}}\right)^{-1} = \lambda .$$

Thus, we seek a solution to

$$\lambda S_W \underline{\hat{w}} = S_B \underline{\hat{w}} \tag{4-11c}$$

If S_W^{-1} exists, a simple solution for the *direction* of $\hat{\underline{w}}$ is

$$\lambda \hat{\underline{w}} = (S_W^{-1} S_B) \hat{\underline{w}} \quad (4-11d)$$

and a solution for \underline{w} may be found by solving for an *eigenvector* of $(S_W^{-1} S_B)$. An alternative solution is based on the fact that $S_B \hat{\underline{w}}$ in (4-11b), has direction $\underline{m}_1 - \underline{m}_2$, since $(\underline{m}_1 - \underline{m}_2)(\underline{m}_1 - \underline{m}_2)^T \hat{\underline{w}} = (\underline{m}_1 - \underline{m}_2)k$. Therefore,

$$\hat{\underline{w}} = S_W^{-1}(\underline{m}_1 - \underline{m}_2) \quad (4-12)$$

Figure 3 shows an example of the Fisher approach in the $c = 2$ class case.

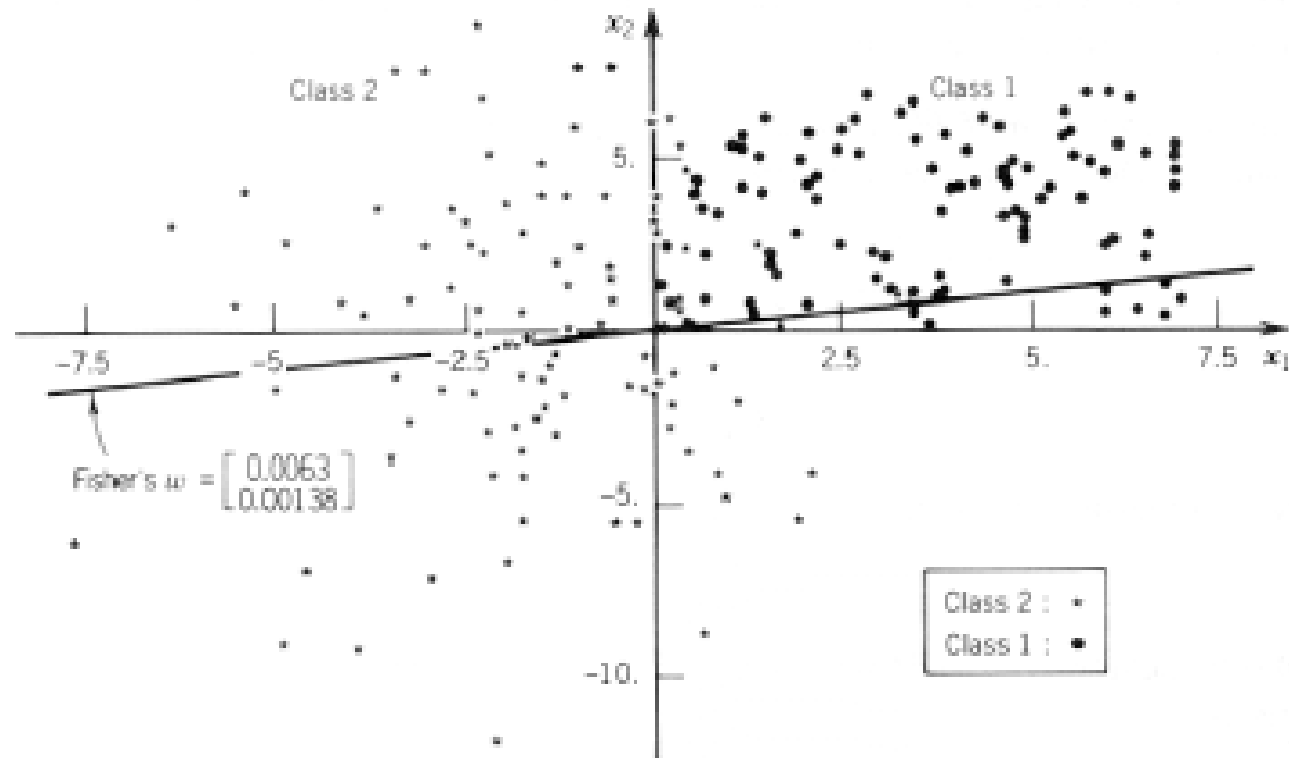


Figure 3 An example of Fisher's linear discriminant function. The line onto which feature data are projected is shown. (The line passes through the origin.)

Q : Is LDA a supervised or unsupervised learning methodology ?

Why ?

Also refer to the book by Robert Schalkoff, “Pattern Recognition ...”, on page 89- for more details on LDA Q327.S297

Homework:

Derive the objective functions and the optimal directions for the sampled data and the known mean-variance cases.

PS: You need to separate the mixed derivations in the lecture slides.

Random Vector Functional Link (RVFL)

Introduction

SLFN: Single Layer Feedforward Network

From Error back-propagation (back-prop) to randomized learning

Notations of RVFL

Learning process in RVFL

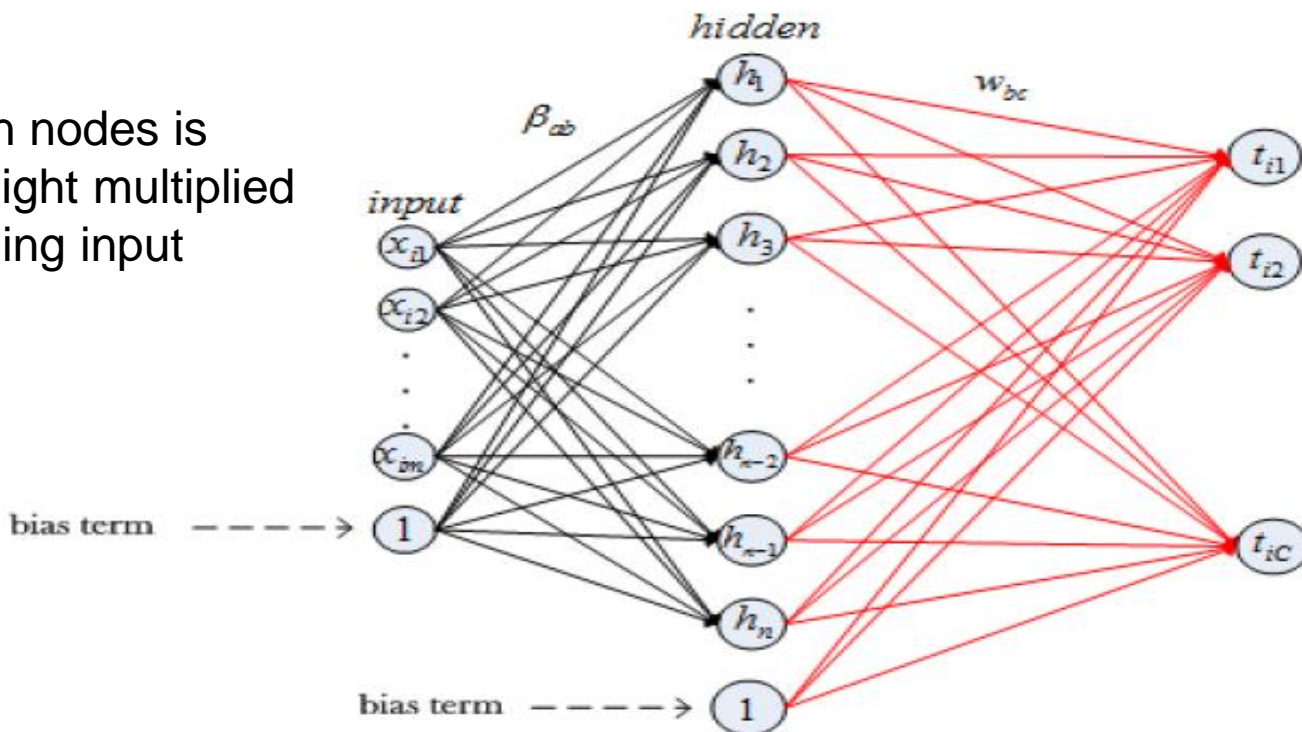
Theoretical part of RVFL

Classification ability of RVFL

property of RVFL variants

Structure of Single Layer Feedforward Neural Network

Input to hidden nodes is sum of link weight multiplied by corresponding input features.



Details

- Bias : $x \times \beta + b = [x, 1] \times \tilde{\beta}$. i.e, extending the input features by adding additional feature which equals to 1.
- For C classes, we can have C 0-1 output neurons. When we present an input observation belonging to c^{th} class, we expect the c^{th} neuron to give an output near one while all others to give near zero. Circles in the hidden and output layers have non-linear activation functions. Inputs to activation functions are weighted sums.

Learning in Neural Networks

learning the mapping $t_i = f(x_i)$, where i is the index for the data. The pipeline can be formalized as follows:

- ▶ **Input:** Input consists of a set of N data $[x_i, y_i]$, where x_i is the features and y_i is the actual target given with the dataset. We refer to this data as the training set used to learn network parameters, etc.
- ▶ **Learning:** Use the training set to learn network structure and parameters. We refer to this step as training a classifier.
- ▶ **Classification:** In the end, we evaluate the quality of the classifier by asking it to predict target t for an input data that it has never seen before. We will then compare the true target y of the data to the t predicted by the model. Prediction t is selected by a max operator among the C outputs.

Approximation Power of Neural Networks

Theoretical proof about the approximation ability of standard multilayer feedforward network can be found in¹.

With the following conditions:

- ▶ Arbitrary squashing functions.
- ▶ Sufficiently many hidden units are available.

Standard multilayer feedforward network can approximate virtually any function to any desired degree of accuracy.

¹Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks 2.5* (1989), pp. 359–366.

Why Randomized Neural Networks

Some weaknesses of back-prop

- ▶ Error surface can have multiple local minimum.
- ▶ Slow convergence due to iteratively adjusting the weights on every link. These weights are first randomly generated.
- ▶ Sensitivity to learning rate setting.

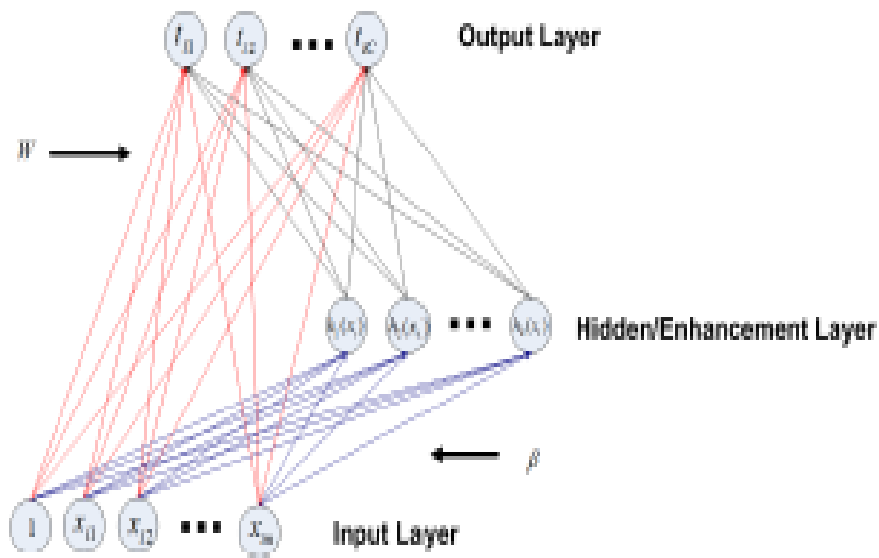
Alternative for back-prop

- ▶ In², the authors show the parameters in hidden layers can be randomly and properly generated without learning.
- ▶ In³, the parameters in the enhancement nodes can be randomly generalized with proper range.

²Wouter F Schmidt, Martin Kraaijveld, Robert PW Duin, et al. "Feedforward neural networks with random weights". In: *1992 11th IAPR International Conference on Pattern Recognition*. IEEE. 1992, pp. 1–4.

³Yoh-Han Pao, Gwang-Hoon Park, and Dejan J Sobajic. "Learning and generalization characteristics of the random vector functional-link net". In: *Neurocomputing*, Vol. 6, pp. 163-180, 1994.

Structure of RVFL



- ▶ Original features (from the input layer) are concatenated with enhanced feature (from hidden layer) and referred to as d .
- ▶ Learning aims at solving $t_i = d_i * W, i = 1, 2, \dots, N$, where N is the number of training sample, W (indicated in red and gray) are the weights in the output layer, and t_i, d_i represent the output values and the combined features, respectively.

Details

- ▶ Parameters β (indicated in blue) in enhancement nodes are randomly generated in proper range and kept fixed.

Details of RVFL: Notations

- ▶ $X = [x_1, x_2, \dots, x_N]'$: input data (N samples and m features).
- ▶ $\beta = [\beta_1, \beta_2, \dots, \beta_m]'$: the weights for the enhancement nodes ($m \times k$, k is the number of enhancement nodes).
- ▶ $b = [b_1, b_2, \dots, b_k]$ the bias for the enhancement node.
- ▶ $H = h(X * \beta + \text{ones}(n, 1) * b)$: feature matrix after the enhancement nodes and h is the activation function. No activation in output nodes.

Commonly used activation functions in hidden layer

activation function	formulation: s is the input, y is the output	
sigmoid	$y = \frac{1}{1+e^{-s}}$	
sine	$y = \text{sine}(s)$	
hardlim	$y = (\text{sign}(s) + 1)/2$	
tribas	Triangular basis	$y = \max(1 - s , 0)$
radbas	Radial basis	$y = \exp(-s^2)$
sign	$y = \text{sign}(s)$	
relu	Rectified linear Unit	$y = \max(0, x)$

Details of RVFL: Notations

$$\blacktriangleright H = \begin{bmatrix} h(x_1) \\ h(x_2) \\ \vdots \\ h(x_N) \end{bmatrix} = \begin{bmatrix} h_1(x_1) & \cdots & h_k(x_1) \\ h_1(x_2) & \cdots & h_k(x_2) \\ \vdots & \ddots & \vdots \\ h_1(x_N) & \cdots & h_k(x_N) \end{bmatrix}$$

$$\blacktriangleright D = [H, X] = \begin{bmatrix} d(x_1) \\ d(x_2) \\ \vdots \\ d(x_N) \end{bmatrix} = \begin{bmatrix} h_1(x_1) & \cdots & h_k(x_1) & x_{11} & \cdots & x_{1m} \\ h_1(x_2) & \cdots & h_k(x_2) & x_{21} & \cdots & x_{2m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ h_1(x_N) & \cdots & h_k(x_N) & x_{N1} & \cdots & x_{Nm} \end{bmatrix}$$

► Target y , can be class labels for C class classification problem $(1, 2, \dots, C)$.

► For classification, y is usually transformed into 0-1 coding of the class

label. For example, if we have 3 classes $y = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, i.e. we

have 3 neurons, 1st neuron goes to 1 for 1st classes while others at zero, and so on.

Learning in RVFL

Define the problem

Once the random parameters β and b are generated in proper range, the learning of RVFL aims at solving the following problem:

$$t_i = d_i * W, i = 1, 2, \dots, N,$$

where N is the number of training sample, W are the weights in the output layer, and t_i , d_i represent the output and the combined features.

Solutions

► closed form:

► In prime space:

$$W = (\lambda I + D^T D)^{-1} D^T Y; \quad \text{If } N > m+k \quad (1)$$

► In dual space:

$$W = D^T (\lambda I + D D^T)^{-1} Y \quad \text{If } N < m+k \quad (2)$$

where λ is the regularization parameter, when $\lambda \rightarrow 0$, the methods converges to the Moore-Penrose pseudoinverse solution.

$$Y = [y_1, y_2, \dots, y_N]';$$

► Iterative methods: tune the weight based on the gradient of the error function in each step. In assignment, you'll use above Eqns.


Approximation Ability of RVFL

Method

- ▶ It formulates a limit-integral representation of the function to be approximated.
- ▶ Evaluates the integral with the Monte-Carlo Method.

Conclusion

- ▶ RVFL is a universal approximator for continuous function on bounded finite dimensional sets.
- ▶ RVFL is an efficient universal approximator with the rate of approximation error converge to zero of order $O(C/\sqrt{n})$, where n is the number of basis functions and C is independent of n .

⁵Bons IgelNIK and Yoh-Han Pao. "Stochastic choice of basis functions in adaptive function approximation and the functional-link net". In: *IEEE Transactions on Neural Networks* 6.6 (1995), pp. 1320–1329. 

Classification Ability of RVFL

Evaluation

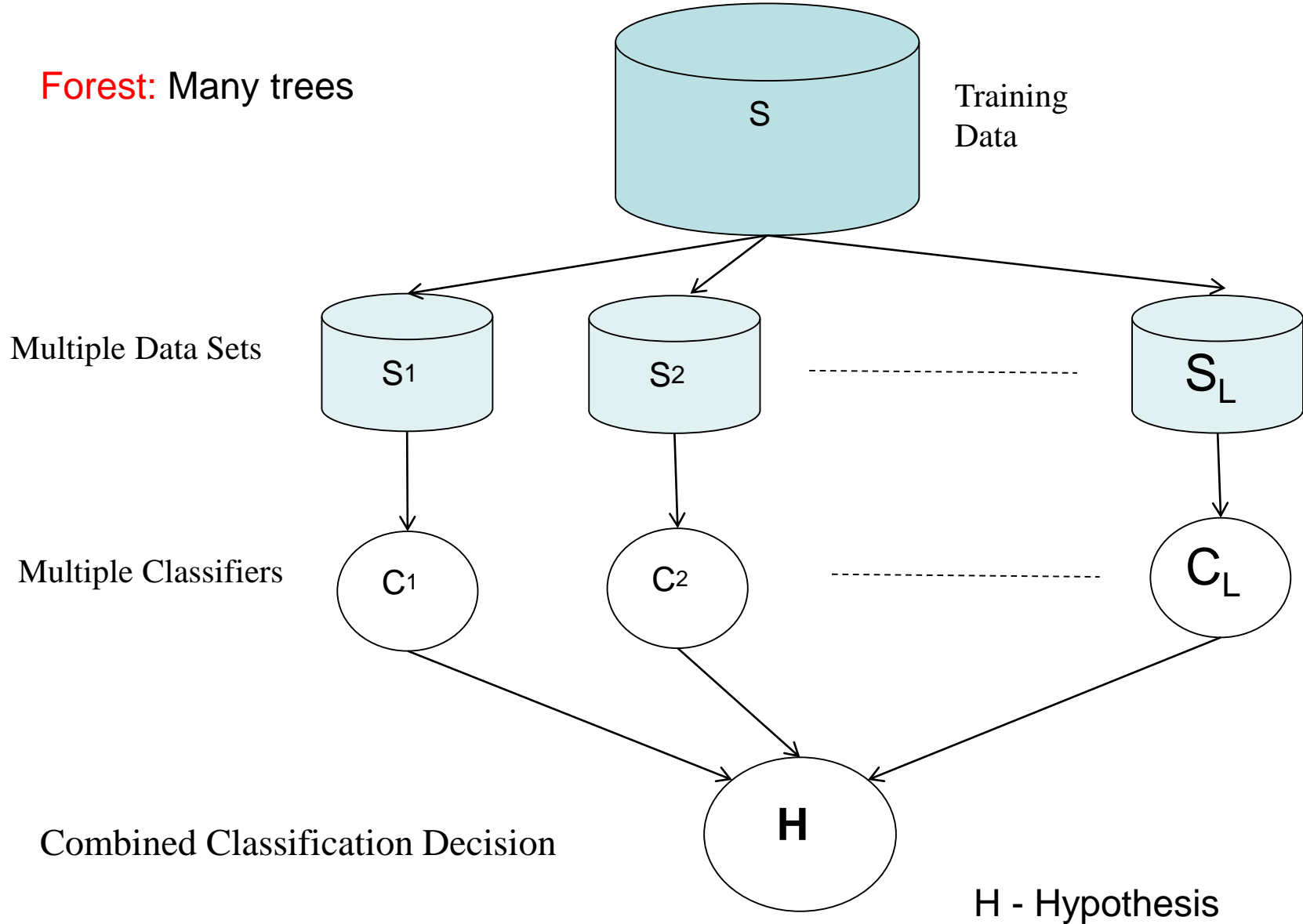
The following issues can be investigated by using 121 UCI datasets

- ▶ Effect of direct links from the input layer to the output layer.
- ▶ Effect of the bias in the output neuron.
- ▶ Effect of scaling the random features before feeding them into the activation function.
- ▶ Performance of 6 commonly used activation functions: *sigmoid*, *radbas*, *sine*, *sign*, *hardlim*, *tribas* .
- ▶ Performance of Moore-Penrose pseudoinverse and ridge regression (or regularized least square solutions) for the computation of the output weights.

L. Zhang, P. N. Suganthan, "[A Comprehensive Evaluation of Random Vector Functional Link Networks](#)," Information Sciences, **DOI:** 10.1016/j.ins.2015.09.025, [Volumes 367–368](#), pp. 1094–1105, Nov 2016.
([Codes Available](#): 2016-RVFL-Comp-Eval-Classification)

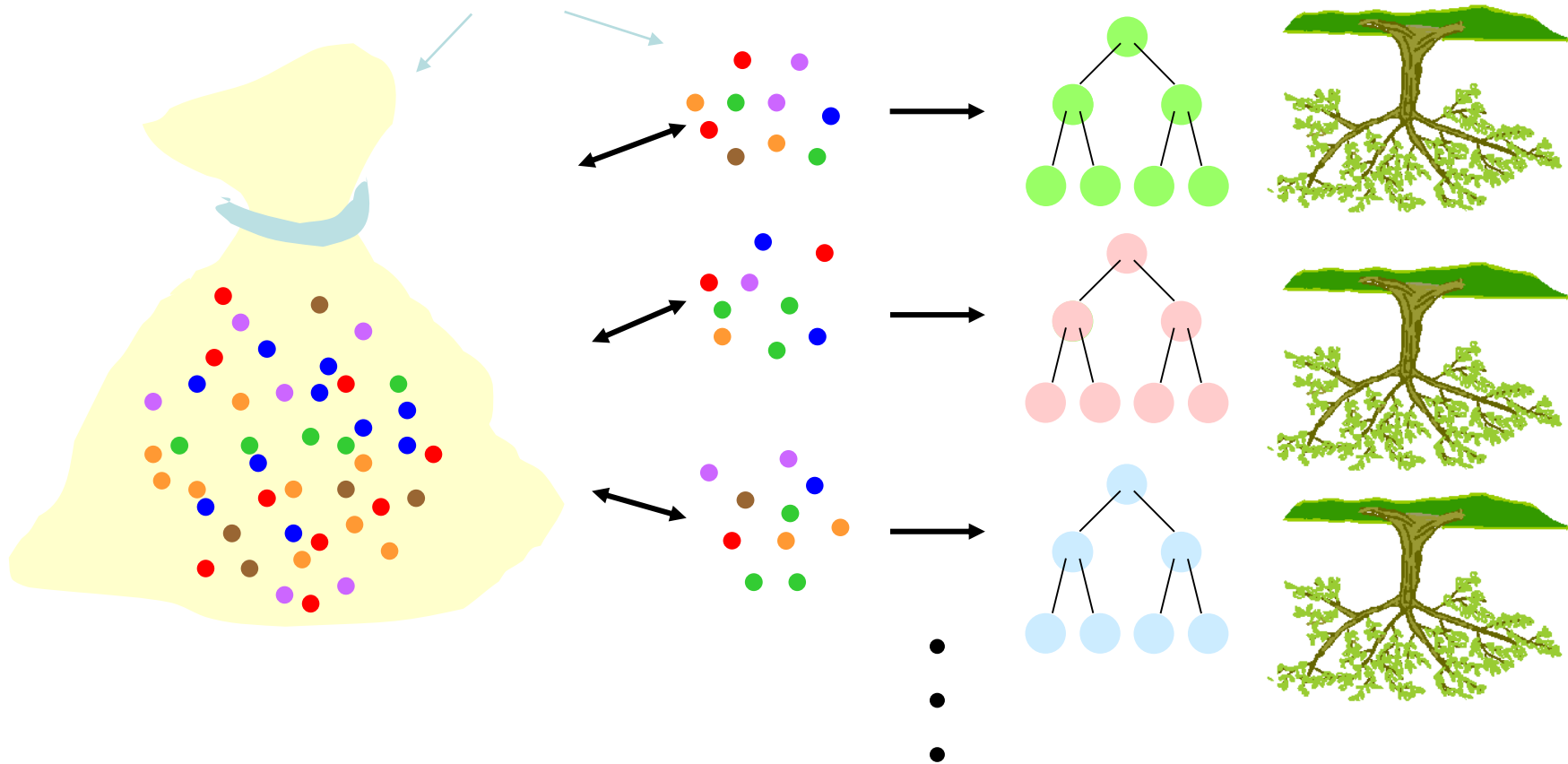
Random Forest: An Ensemble Classifier

Forest: Many trees



Bagging(1)

Same number of training samples



Total number of trees - L

Bagging(2)

1. For $\ell = 1$ to L / L ...number of iterations
 - a) draw(with replacement) from the training set to generate the training data S_ℓ
 - b) learn a classifier C_ℓ on S_ℓ .
2. For each test example
 - a) try all classifiers C_ℓ
 - b) Predict the class that receives the highest number of votes.

Random Forests: 100s of Decision Trees

Random Forest Algorithm:

· Training phase:

Given:

$X := N \times M$ is the training dataset, where N is the number of the training data, M is the dimension of each data.

$Y := N \times 1$ is the labels of the training set.

L is the ensemble size, which means the number of trees in the forests.

T_i refers to each random tree in the forest, $i = 1 \dots L$.

m is the number of features randomly selected to split in each non-leaf node.

$minleaf$ is the number of maximum number of samples in an impure node.

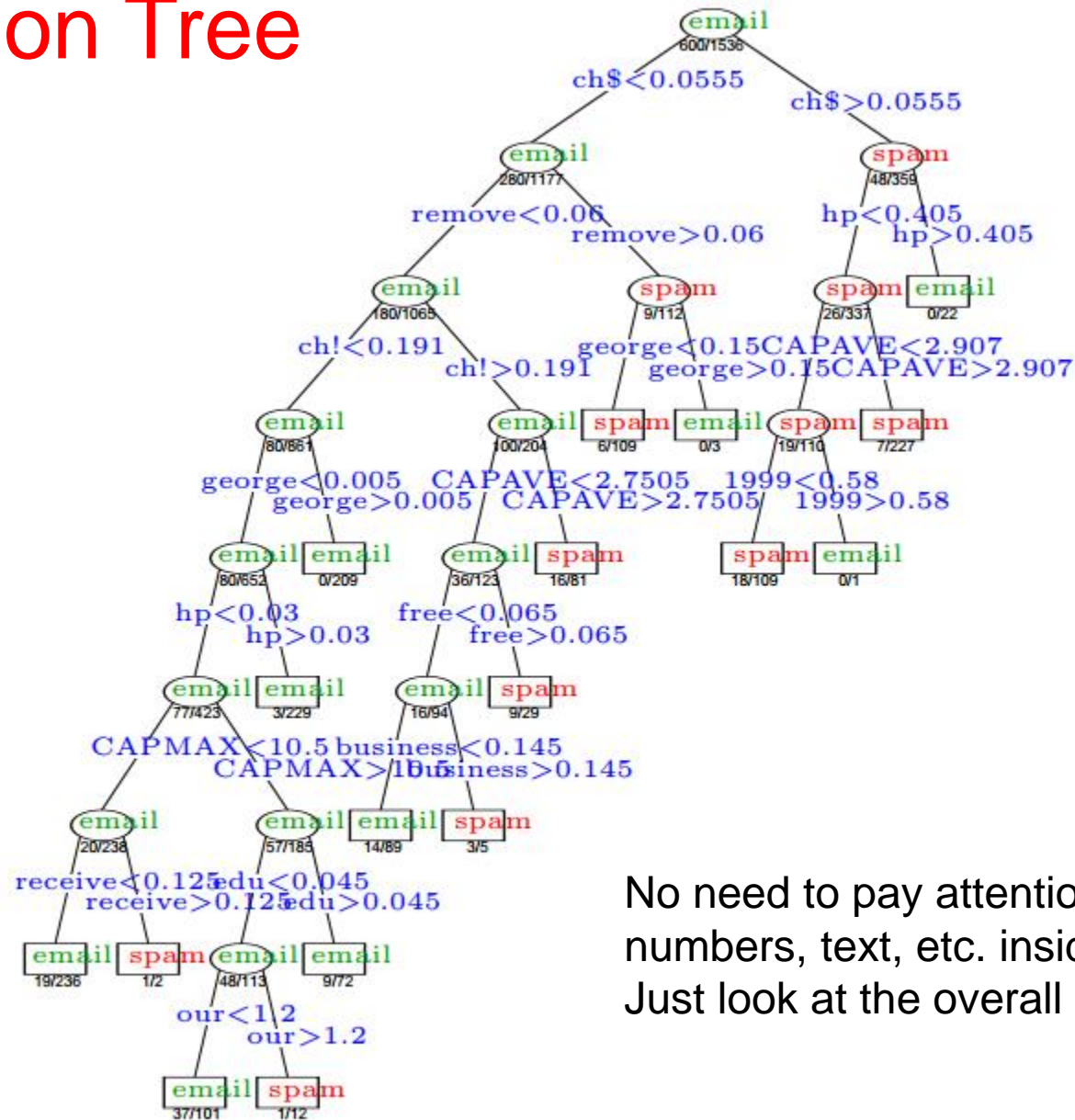
1. Generate the training set for T_i by sampling N times from all N available training cases with replacement.
2. At each node the best split is calculated using the m randomly chosen features in the training set for T_i .
3. Go to Step 2 until T_i is fully grown until reach an pure node or the number of samples are smaller than " $minleaf$ ".

· Classification phase:

For a given sample, it is pushed down each tree in the forests and each tree in the forests will give one vote on the predicted label of this sample. In this case, the predicted label of this sample is determined as the one which has the most votes in the forests.

Pure Node: All samples belong to the same class.

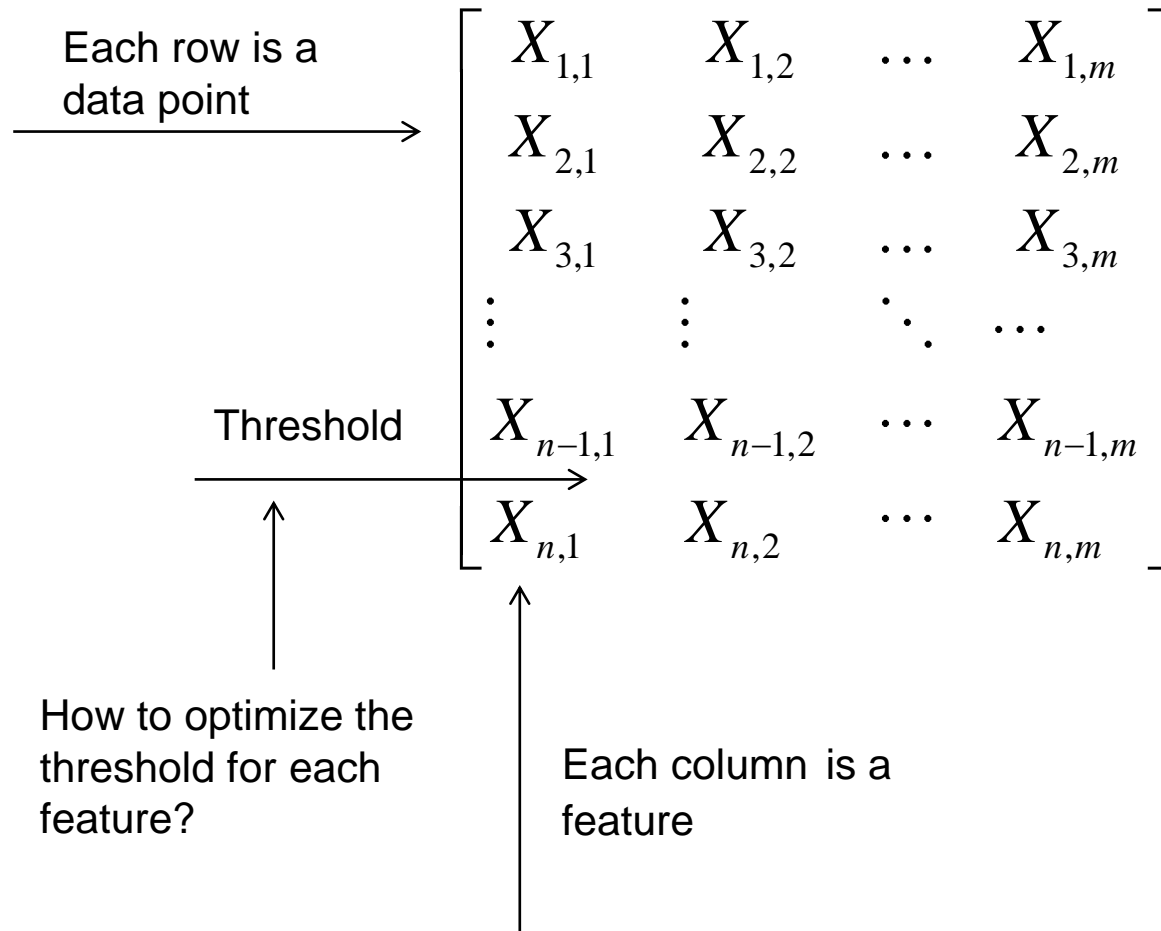
Decision Tree



No need to pay attention to the numbers, text, etc. inside the tree. Just look at the overall structure only.

A Node of Decision Tree

Assume that n samples reach the node. m features are chosen to find the split.



Optimize the threshold over all features

- Gini impurity, the threshold should be optimized so that $Gini_{before\ split} - Gini_{after\ split}$ is the largest.

$$Gini(t)_{before\ split} = 1 - \sum_{i=1}^c \left(\frac{n_{w_i}}{n} \right)^2$$

$$Gini(t)_{after\ split} = \frac{n^l}{n} \left[1 - \sum_{i=1}^c \left(\frac{n_{w_i}^l}{n^l} \right)^2 \right] + \frac{n^r}{n} \left[1 - \sum_{i=1}^c \left(\frac{n_{w_i}^r}{n^r} \right)^2 \right]$$

c – number of classes ; n^r – training samples going to right branch
 n^l – training samples going to left branch ; n – total training samples

There are other impurity criteria such as information gain, etc.

Convolutional Neural Networks

Deep Learning

Deep Neural Network

Deep Neural Network has been popular since the work².

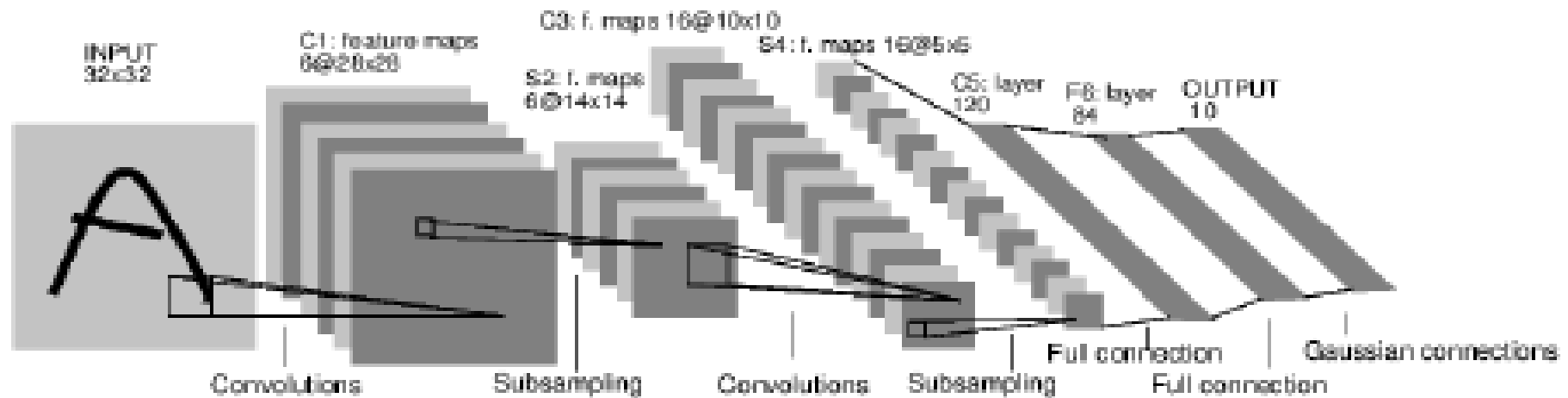
- ▶ Many hidden layers.
- ▶ Greedy layer-wise unsupervised training.²
- ▶ Sufficient training set: large datasets.

CNN

- ▶ CNNs are biologically-inspired variants of MLPs.
- ▶ Among all deep neural networks, Deep Convolutional Neural Network (CNN) has lead to a great success in computer Vision and Machine learning tasks.

²Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *Science* 313.5786 (2006), pp. 504–507.

Convolutional Neural Networks – Lenet - 5



Layers

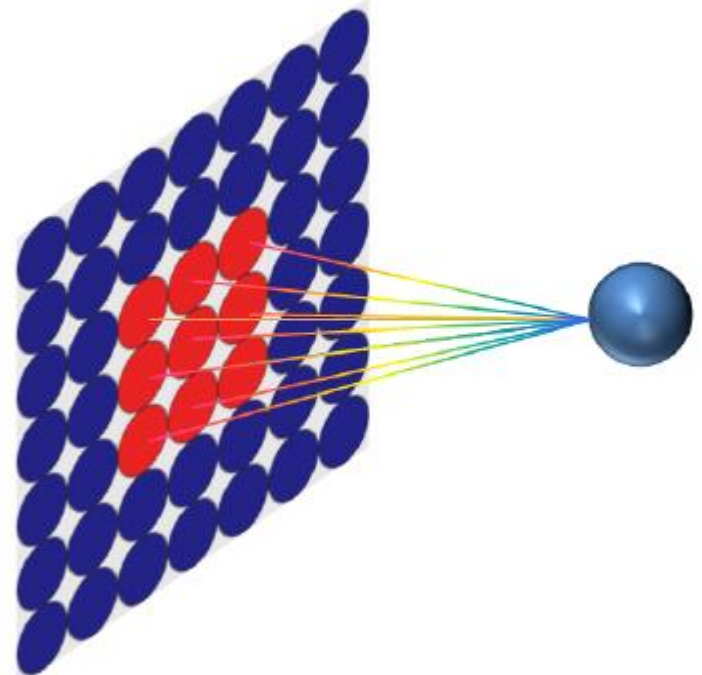
A typical CNN may consist of the following layers:

- ▶ Convolutional Layer: C1, C3, C5
- ▶ Sub-sampling (Pooling) Layer: S2, S4.
- ▶ Fully Connected Layer: F6.

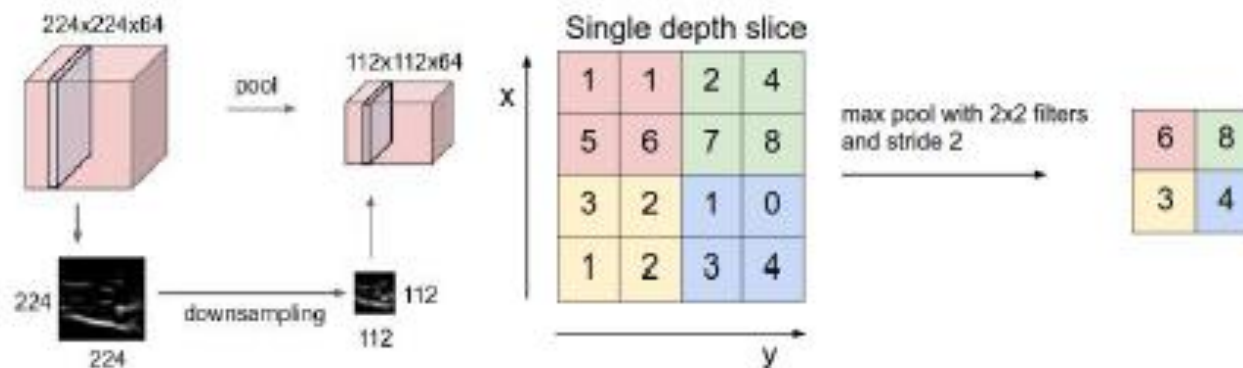
One can have the flexibility of designing different CNN models by combining different basic layers for different application.

Properties: Local Receptive Fields and Shared Weights

- ▶ Local receptive fields: neurons in the convolutional layer is only locally connected to some neighbourhood in previous layer.
- ▶ Shared weights: All neurons in the convolutional layer use the same weights.
- ▶ Convolve the input with a set of filter banks to generate the feature maps as the output of the convolutional layer.



Properties: Subsampling / Pooling



Effect

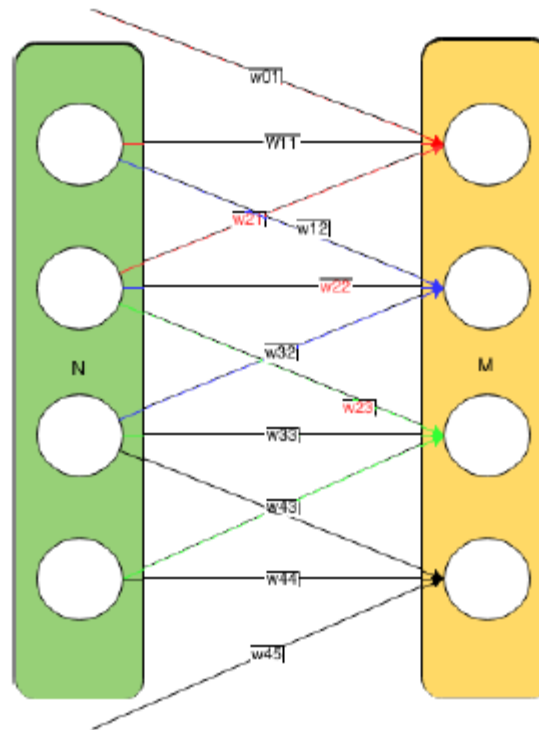
Convolutional layer and pooling layer makes CNN to have shift, scale and distortion invariances.

Training

- ▶ Most CNN are trained with back-propagation. Requires a lot of training data.
- ▶ Limited data: fine-tune the CNN model pretrained from auxiliary data in the same domain. Also known as transfer learning

Fully Connected Layer

- ▶ A fully connected layer takes all neurons in the previous layer (be it fully connected, pooling, or convolutional) and connects it to every single neuron it has.
- ▶ Usually appears before the classifier to perform dimensional reduction for features.

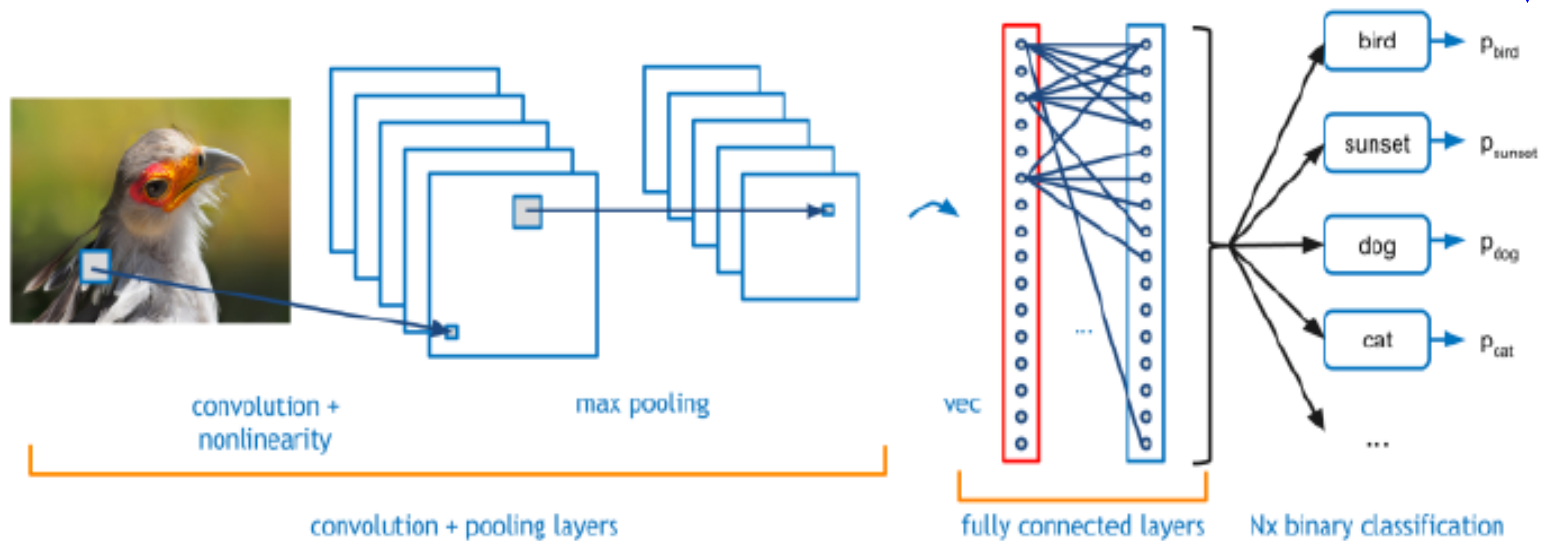


Why CNN Popular

End to end learning

Joint feature extraction and classification/regression.

Output probabilities



Why CNN Popular

One may have numerous choices for CNN.

C/C++: Caffe

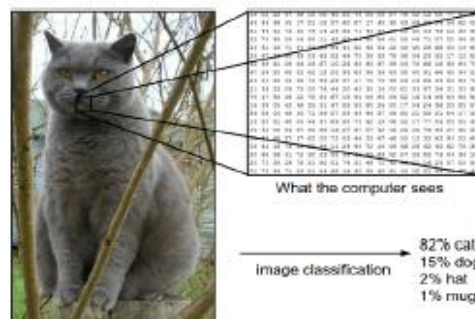
- ▶ Designed by the Berkeley Vision and Learning Center (BVLC).
- ▶ Expressive architecture : define by configuration without hard-coding.
- ▶ Extensible code: forked by over 1,000 developers.
- ▶ Speed: fastest convnet implementation available.
- ▶ Reproducibility: Implementations of state-of-the-art CNN models can be found in Model Zoo.
- ▶ Community: powers academic research projects, startup prototypes, and even large-scale industrial applications in vision, speech, and multimedia.

C/C++: Cuda-convnet

- ▶ Designed by Alex Krizhevsky.
- ▶ Efficient implementation of convolution in CUDA.
- ▶ Modular design: easy to add new layer types, neuron activation functions, or objectives.
- ▶ Mostly avoids use of temporary memory where it isn't strictly needed. ☰

Applications of CNN: Image Classification

- ▶ An image classification model takes a single image and assigns probabilities to 4 labels, {cat, dog, hat, mug}.
- ▶ Image is represented as one large 3-dimensional array of numbers (the cat image is 248 pixels wide, 400 pixels tall, and has three color channels Red, Green, Blue (or RGB for short). the image consists of $248 \times 400 \times 3$ numbers (or a total of 297,600 numbers). Each number is an integer that ranges from 0 (black) to 255 (Maximum)
- ▶ Our task is to turn this quarter of a million numbers into a single label, such as "cat".³



³Andrej Karpathy Fei-Fei Li. *Stanford CS class CS231n: Convolutional Neural Networks for Visual Recognition.*

Challenges in Image Classification

Though it is relatively trivial for a human to perform. It is very challenging for computer:

- ▶ Viewpoint variation: A single instance of an object can be oriented in many ways with respect to the camera.
- ▶ Scale variation: Visual classes often exhibit variation in their size (size in the real world, not only in terms of their extent in the image).
- ▶ Deformation: Many objects of interest are not rigid bodies and can be deformed in extreme ways.
- ▶ Occlusion: The objects of interest can be occluded. Sometimes only a small portion of an object (as little as few pixels) could be visible.
- ▶ Illumination conditions: The effects of illumination are drastic on the pixel level.
- ▶ Background clutter: The objects of interest may blend into their environment, making them hard to identify.
- ▶ Intra-class variation: The classes of interest can often be relatively broad, such as chair. There are many different types of these objects, each with their own appearance.

A good image classification model must be invariant to the cross product of all these variations, while simultaneously retaining sensitivity to the inter-class variations.⁴

Challenges of Image Classification

Viewpoint variation



Scale variation



Deformation



Occlusion



Illumination conditions



Background clutter



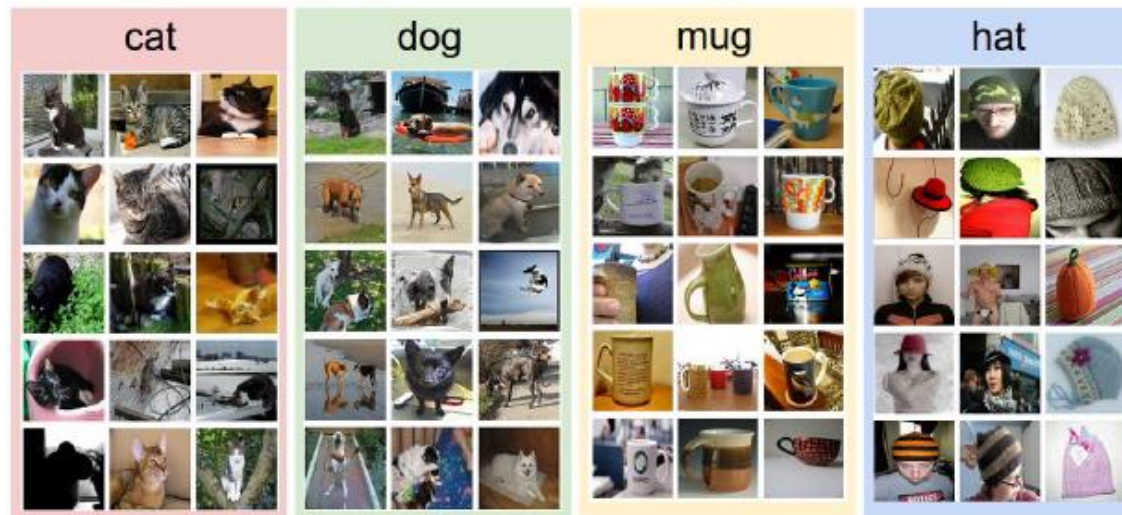
Intra-class variation



How to write an algorithm

- ▶ The approach that we will take is not unlike one you would take with a child: we're going to provide the computer with many examples of each class and then develop learning algorithms that look at these examples and learn about the visual appearance of each class.
- ▶ This approach is referred to as a data-driven approach, since it relies on first accumulating a training dataset of labeled images.⁵

An example of a training set:



The image classification pipeline

Image Classification is to take an array of pixels that represents a single image and assign a label to it. Our complete pipeline can be formalized as follows:

- ▶ **Input:** Our input consists of a set of N images, each labeled with one of K different classes. We refer to this data as the training set.
- ▶ **Learning:** Our task is to use the training set to learn what every one of the classes looks like. We refer to this step as training a classifier, or learning a model.
- ▶ **Evaluation:** In the end, we evaluate the quality of the classifier by asking it to predict labels for a new set of images that it has never seen before. We will then compare the true labels of these images to the ones predicted by the classifier. Intuitively, we're hoping that a lot of the predictions match up with the true answers (which we call the ground truth)⁶.

Applications of CNN

Scene recognition



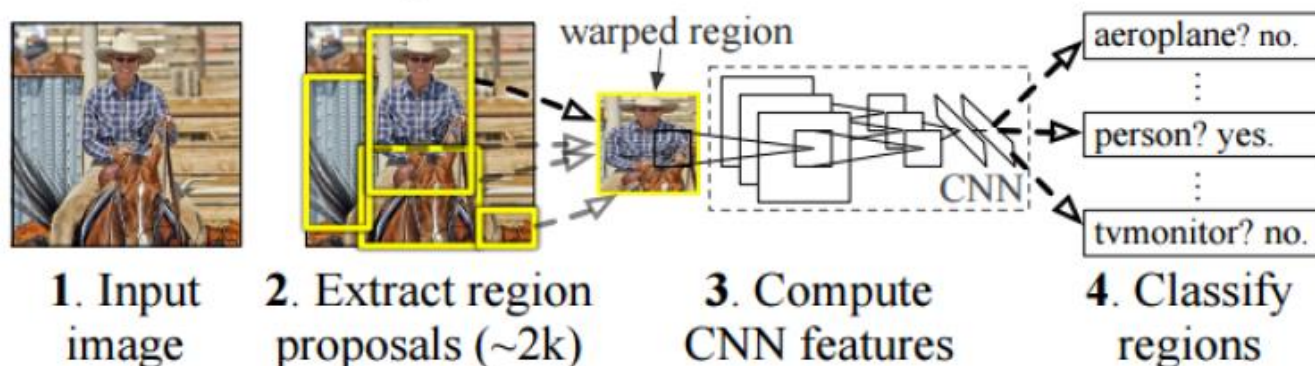
- ▶ large datasets like ImageNet (the sub-dataset for classification task in 2014 contains 10+ million images belonging to 400+ unique scene categories) is not sufficient to train a deep CNN model for scene recognition.
- ▶ Introduce a new scene-centric database called Places with over 7 million labeled pictures of scenes.¹⁰

¹⁰Bolei Zhou et al. "Learning deep features for scene recognition using places database". In: *Advances in Neural Information Processing Systems*. 2014, pp. 487–495.

Applications of CNN

Object Detection

R-CNN: Region-based Convolutional Network

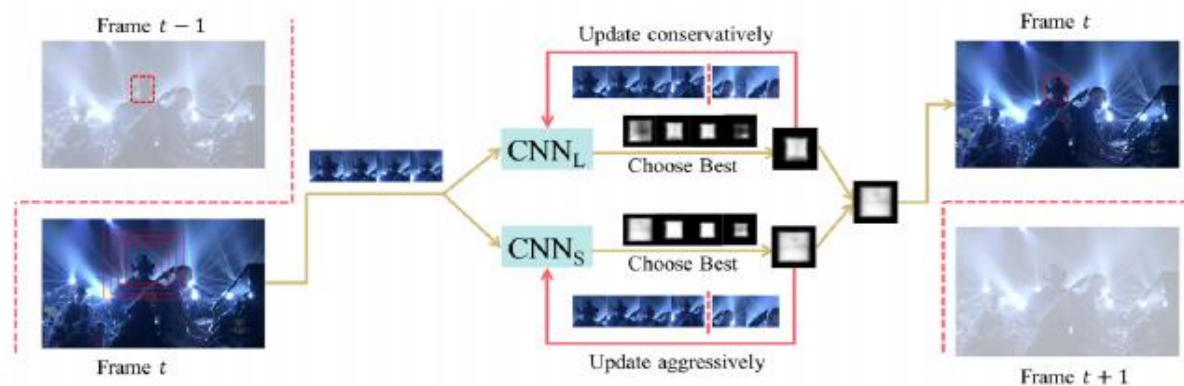


- ▶ Apply high-capacity CNNs to bottom-up region proposals (These proposals define the set of candidate detections available to our detector) in order to localize and segment objects.
- ▶ When labeled training data are scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, boosts performance significantly.¹¹

¹¹R. Girshick et al. "Region-based Convolutional Networks for Accurate Object Detection and Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2015), pp. 1-1. ISSN: 0162-8828.

Applications of CNN

Visual Tracking



- ▶ Visual tracking: self-taught learning (training data only exists in the first frame).
- ▶ Lack of properly labeled training data: a major hurdle that hinders the application of CNN to visual tracking.
- ▶ Pre-training a CNN offline on ImageNet 2014 detection dataset and then transferring the rich feature hierarchies learned to online tracking.
- ▶ Two trackers: One updates conservatively and the other updates aggressively.¹²

¹²Naiyan Wang et al. "Transferring Rich Feature Hierarchies for Robust Visual Tracking". In: *arXiv preprint arXiv:1501.04587* (2015).

Further Reading

Recent Deep Learning CNNs have 1000s of layers and billions of connection weights developed by Google, Microsoft, etc.

- ▶ UFLDL Tutorial (http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial).
- ▶ CS231n: Convolutional Neural Networks for Visual Recognition (<http://cs231n.stanford.edu/>).
- ▶ Anderson de Andrade. “Best Practices for Convolutional Neural Networks Applied to Object Recognition in Images.”
- ▶ Jake Bouvrie. “Notes on Convolutional Neural Networks”.

Supervised Training Vs Unsupervised Training

Supervised training

- A supervisor teaches the system how to classify a known set of patterns with class labels before using it.
- need *a priori* class label information

Unsupervised training

- does not depend on *a priori* class label information
- used when *a priori* class label information is not available or proper training sets are not available.

Other cases

- Semi-supervised: Some data samples labeled while most not.
- Sample generation using generative adversarial networks (GAN) using a/few real image or sketches, descriptions, etc.

Unsupervised training by clustering

To seek regions such that every pattern falls into one of these regions and no pattern falls in two or more regions, i.e.

$$S_1 \cup S_2 \cup S_3 \dots \cup S_K = S ; \quad S_i \cap S_j = \emptyset \quad \forall i \neq j$$

- * Algorithms classify objects into clusters by natural association according to some similarity measures. e.g. Euclidean distance
- * Clustering algorithms are based either on heuristics or optimization principles.

Clustering with a known number of classes

Require knowledge about the number of classes, e.g.
K classes (or clusters).

The K-means algorithm is based on the
minimization of the sum of squared distances

$$J_j = \sum_{\mathbf{x} \in S_j(k)} \left\| \mathbf{x} - \mathbf{z}_j \right\|^2 \quad j = 1, 2, \dots, K$$

where $S_j(k)$ is the cluster domain for cluster
Centre \mathbf{z}_j at the k^{th} iteration.

K-Means Algorithm

1. Arbitrarily choose K samples as initial cluster centres. i.e. $\mathbf{z}_1(k), \mathbf{z}_2(k), \mathbf{z}_3(k), \dots, \mathbf{z}_K(k)$; **iteration, $k = 1$** .
2. At k th iterative step, distribute the pattern samples \mathbf{x} among the K clusters according to the following rule:

$$\mathbf{x} \in S_j(k) \quad \text{if} \quad \|\mathbf{x} - \mathbf{z}_j(k)\| < \|\mathbf{x} - \mathbf{z}_i(k)\|$$

for $i = 1, 2, \dots, K$; $i \neq j$;

3. Compute $\mathbf{z}_j(k+1)$ and J_j for $j = 1, 2, \dots, K$.

$$\mathbf{z}_j(k+1) = \frac{1}{N_j} \sum_{\mathbf{x} \in S_j(k)} \mathbf{x} \quad ;$$

$$J_j = \sum_{\mathbf{x} \in S_j(k)} \left\| \mathbf{x} - \mathbf{z}_j(k+1) \right\|^2$$

4. If $\mathbf{z}_j(k+1) = \mathbf{z}_j(k)$ for $j = 1, 2, \dots, K$, stop.

Otherwise go to step 2.

Advantages :

- simple and efficient.
- minimum computations are required

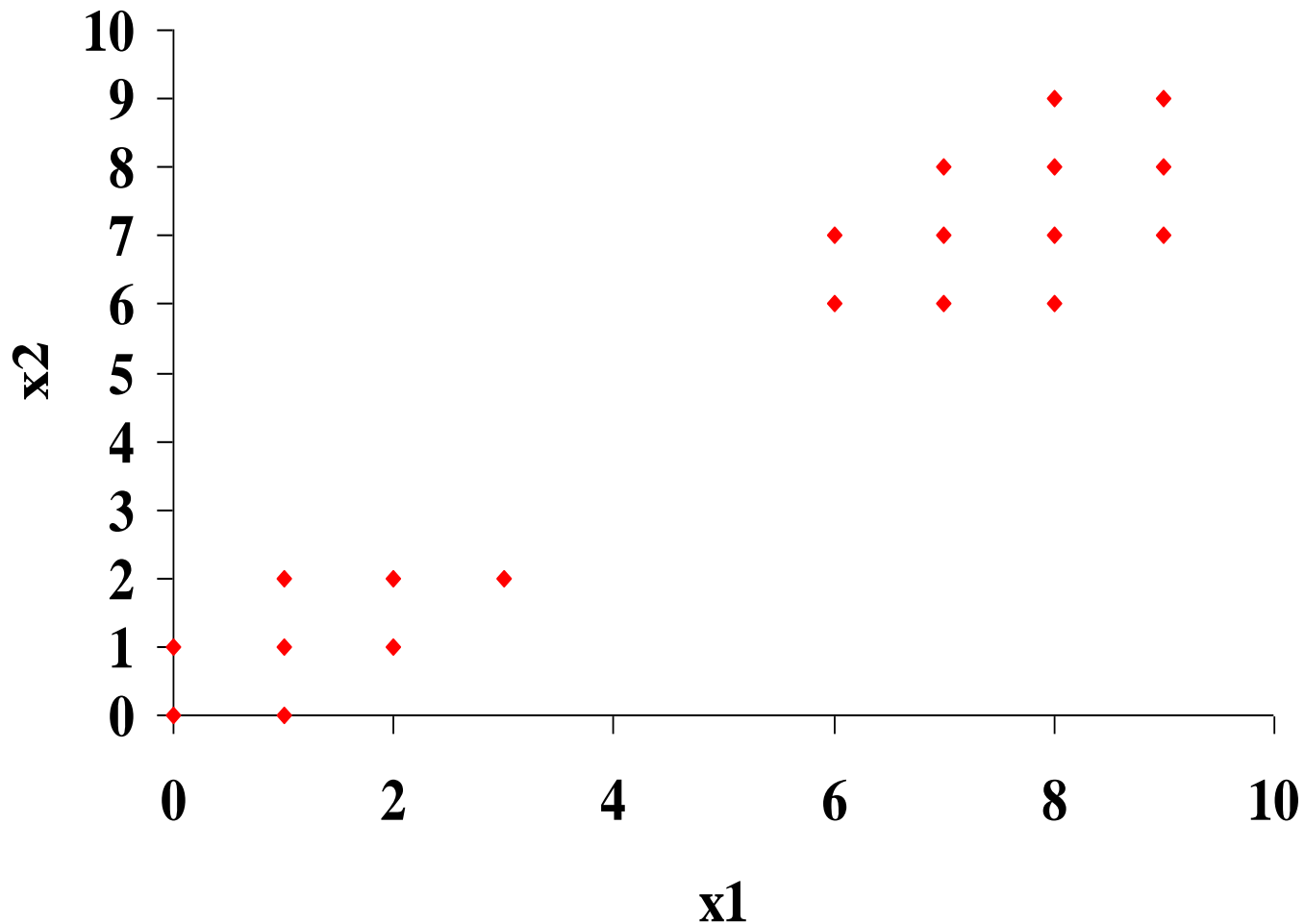
Drawbacks :

- K must be decided
- Individual Clusters themselves should be tight, and clusters should be widely separated from one another.
- Results usually depend on the order of presentation of training samples and the initial cluster centers.

Example

Apply K-means algorithm to the following data for $K = 2$:

$$\begin{aligned} \mathbf{x}_1 &= (0,0), & \mathbf{x}_2 &= (0,1), & \mathbf{x}_3 &= (1,0), \\ \mathbf{x}_4 &= (1,1), & \mathbf{x}_5 &= (2,1), & \mathbf{x}_6 &= (1,2), \\ \mathbf{x}_7 &= (2,2), & \mathbf{x}_8 &= (3,2), & \mathbf{x}_9 &= (6,6), \\ \mathbf{x}_{10} &= (7,6) & \mathbf{x}_{11} &= (8,6), & \mathbf{x}_{12} &= (6,7), \\ \mathbf{x}_{13} &= (7,7), & \mathbf{x}_{14} &= (8,7), & \mathbf{x}_{15} &= (9,7), \\ \mathbf{x}_{16} &= (7,8), & \mathbf{x}_{17} &= (8,8), & \mathbf{x}_{18} &= (9,8), \\ \mathbf{x}_{19} &= (8,9), & \mathbf{x}_{20} &= (9,9) \end{aligned}$$



Homework – make use of this dataset with class labels to try out Fisher's Discriminant method to compute the projection vector \hat{w} .

Example

Iteration 1:

Step 1. Let $K = 2$,

$$\mathbf{z}_1(1) = \mathbf{x}_1 = (0,0)^T, \quad \mathbf{z}_2(1) = \mathbf{x}_2 = (0,1)^T.$$

(These starting points are selected randomly).

Step 2.

Calculate distances of all patterns from \mathbf{z}_1 and \mathbf{z}_2 .

The patterns are assigned to $S_1(1)$ and $S_2(1)$ as

$$S_1(1) = \{\mathbf{x}_1, \mathbf{x}_3\} ; \quad S_2(1) = \{\mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_5, \dots, \mathbf{x}_{20}\}$$

Step 3.

Update the cluster centers:

$$\mathbf{z}_1(2) = (0.5 \quad 0.0)^T ; \quad \mathbf{z}_2(2) = (5.67 \quad 5.33)^T$$

Step 4.

$\mathbf{z}_j(1)$ and $\mathbf{z}_j(2)$ are different, go back to step 2

Iteration 2:

Step 2.

Calculate distances of all patterns from \mathbf{z}_1 and \mathbf{z}_2 .
The patterns are assigned to $S_1(2)$ and $S_2(2)$ as

$$S_1(2) = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_8\} ; \quad S_2(2) = \{\mathbf{x}_9, \mathbf{x}_{10}, \dots, \mathbf{x}_{20}\}$$

Step 3.

Update the cluster centers:

$$\mathbf{z}_1(3) = (1.25 \quad 1.13)^\top ; \mathbf{z}_2(3) = (7.67 \quad 7.33)^\top$$

Step 4.

$\mathbf{z}_j(3)$ and $\mathbf{z}_j(2)$ are different, go to step 2

Iteration 3:

Step 2.

Calculate distances of all patterns from \mathbf{z}_1 and \mathbf{z}_2 .

No change in S_1 and S_2 .

Step 3.

- Update the cluster centers:
- $\mathbf{z}_1(3) = (1.25 \ 1.13)^\top$; $\mathbf{z}_2(3) = (7.67 \ 7.33)^\top$

Step 4.

- $\mathbf{z}_j(3)$ and $\mathbf{z}_j(4)$ are the same, stop.
- Final Centers :
- $\mathbf{z}_1(4) = (1.25 \ 1.13)^\top$; $\mathbf{z}_2(4) = (7.67 \ 7.33)^\top$

Batchelor and Wilkin's algorithm

This algorithm forms clusters without requiring the specification of the number of clusters, K . But instead, it needs to specify a parameter, namely the proportion p .

The algorithm:

1. Arbitrarily, take the first sample as first cluster. i.e. $\mathbf{z}_1 = \mathbf{x}_1$
2. Choose the farthest pattern sample from \mathbf{z}_1 to be \mathbf{z}_2 .

3. Compute distance from each remaining pattern sample to \mathbf{z}_1 and \mathbf{z}_2 .
4. Save the minimum distance for each pair of these computations.
5. Select the maximum of these minimum distances.
6. If the maximum distance is greater than a fraction p of $d(\mathbf{z}_1, \mathbf{z}_2)$, assign the corresponding sample to a new cluster \mathbf{z}_3 .
Otherwise, stop.

7. Compute distance from each remaining pattern sample to each of the 3 clusters and save the minimum distance of every group of 3 distances.
8. Then select the maximum of these minimum distances.
9. If the maximum distance is greater than a fraction p of the *typical* previous maximum distance, assign the corresponding sample to a new cluster \mathbf{z}_4 . Otherwise, stop.

10. Repeat steps 7 to 9 until no new cluster is formed.
11. Assign each sample to its nearest clusters.

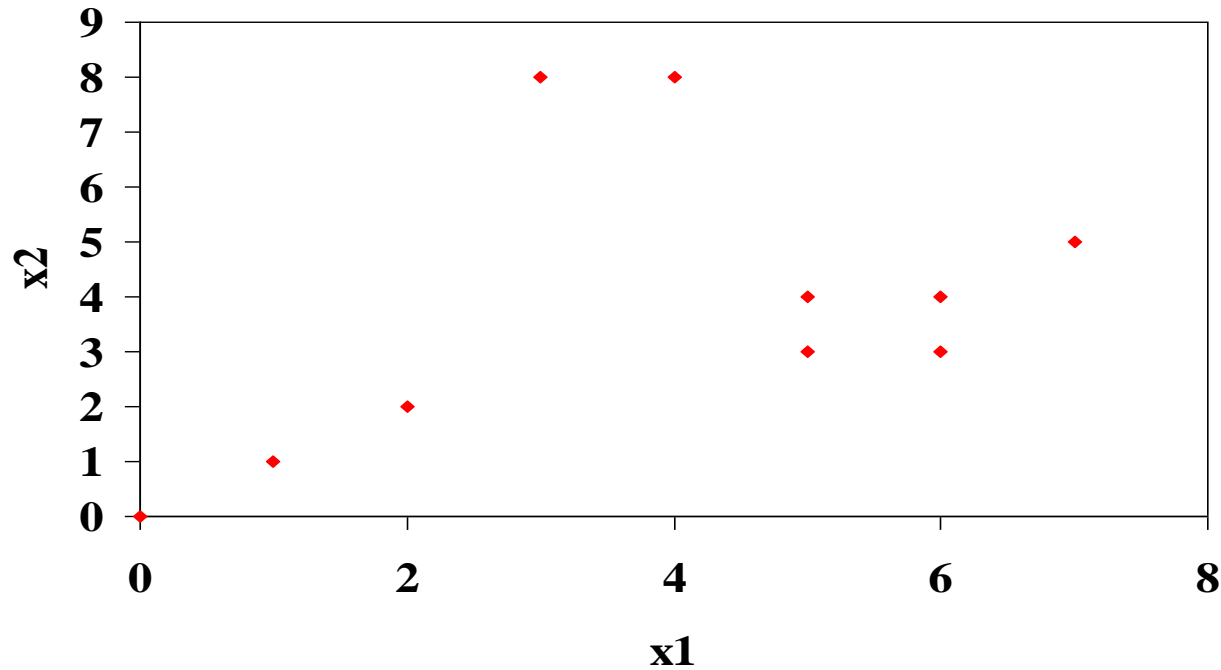
Example Apply Batchelor and Wilkin's algorithm to find the cluster groups for the following samples :

$$\begin{aligned}\mathbf{x}_1 &= (0,0), \mathbf{x}_2 = (3,8), \mathbf{x}_3 = (2,2), \mathbf{x}_4 = (1,1), \\ \mathbf{x}_5 &= (5,3), \mathbf{x}_6 = (4,8), \mathbf{x}_7 = (6,3), \mathbf{x}_8 = (5,4), \\ \mathbf{x}_9 &= (6,4), \mathbf{x}_{10} = (7,5)\end{aligned}$$

By using the algorithm, the result will be

$$S_1 = \{\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_4\}; \quad S_2 = \{\mathbf{x}_2, \mathbf{x}_6\}$$

$$S_3 = \{\mathbf{x}_5, \mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9, \mathbf{x}_{10}\};$$



Q : Apply Batchelor and Wilkens algorithm to the above training data. Select the proportion parameter as 0.5. Start with $\mathbf{x}_1 = (0,0)$.

Q : What effect do you think the proportion parameter has on the final clustering ?

The **Advantages** of this heuristic algorithm are as follows :

- i) It determines K,
- ii) it self-organises, and
- iii) it is efficient for small-to-medium numbers of classes.

The **disadvantages** are :

- i) the proportion p must be given,
- ii) the ordering of the samples influences the clustering,
- iii) the value of p influences the clustering,
- iv) the separation is linear,
- v) the cluster centers are not recomputed to better represent the classes,
- vi) there is no measure of how good the clustering is,
- vii) an outlier can upset the whole process.

In practice, it is suggested that the process be run for several values of p and the mean-squared error of all distances of the samples from their center be computed for each k th cluster ($k = 1, 2, \dots, K$) and then summed over all classes k to obtain the total mean-squared error. The value of p that provides the lowest total mean-square error is an optimal one. The final clusters should be averaged to obtain a new center and all sample vectors reassigned via minimum distance to the new centers.