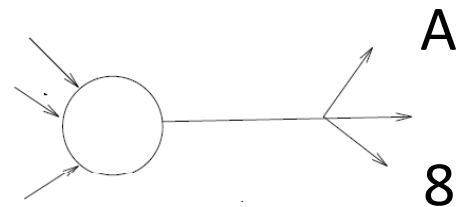


# 4º CAPÍTULO

## Conceitos de Reconhecimento de Padrões

```
00000000000000000000 00000000000000000000
00000000010000000000 00000000011100000000
00000000011000000000 00000001100001100000
00000000101000000000 00000011000000110000
00000001100110000000 00000100000000010000
00000001000010000000 00001100000000011000
00000010000010000000 00001000000000010000
00001100000010000000 00001000000000011000
00001000000011000000 00001000000000010000
00001000000001100000 00001000000000011000
00001000000001000000 00000011100000010000
00001100111111100000 00000011100000010000
00001111110000010000 00000000111100000000
00011000000000011000 00000011000111000000
00010000000000001000 00000110000001100000
00010000000000001100 00001100000000011000
00110000000000000100 00011000000000011000
00110000000000000110 00110000000000010000
00100000000000000010 00100000000000001100
00100000000000000010 00010000000000011000
01000000000000000000 00011000000000010000
00000000000000000000 00001000000000011000
00000000000000000000 00001110000001110000
00000000000000000000 00000001111110000000
00000000000000000000 00000000000000000000
```



Prof. Arnaldo Abrantes

# Reconhecimento de padrões

- **Definição:** ao processo de atribuir uma etiqueta (classe) à instância de objecto observado designa-se por **classificação**
  - Porquê a necessidade da classe de rejeição?
  - Exemplo: reconhecimento de caracteres, classificação de frutos
- **Definição:** ao processo de fazer corresponder (ou não) uma instância de um objecto com um protótipo específico designa-se por **verificação**
  - Exemplo: verificação de identidade num ATM

00000000000000000000	00000000000000000000
00000000001000000000	00000000001111000000
00000000011000000000	0000000011000001100000
00000000010100000000	0000000110000000110000
0000000011001100000000	0000001000000000010000
0000000010000100000000	0000110000000000110000
0000000100000100000000	0000100000000000010000
0000001100000010000000	0000100000000000110000
0000001000000010000000	0000110000000000100000
0000001000000001100000	0000001000000000110000
0000100000000001000000	0000001110000000100000
0000110011111111000000	0000000111001111000000
0000111111000001000000	0000000011110000000000
0001100000000000110000	0000000110001110000000
0001000000000000010000	0000001100000001100000
0001000000000000011000	0000110000000001100000
0011000000000000001000	0001100000000000110000
0011000000000000001100	0011000000000000010000
0010000000000000000100	0010000000000000011000
0010000000000000000100	0001000000000000110000
0110000000000000000100	0001100000000000100000
0100000000000000000000	0000100000000001100000
0000000000000000000000	0000111000000111000000
0000000000000000000000	00000011111110000000
0000000000000000000000	0000000000000000000000

# Modelo de sistema de classificação

- Componentes dum sistema de classificação

- Classes  $C_1, C_2, \dots, C_{m-1}$

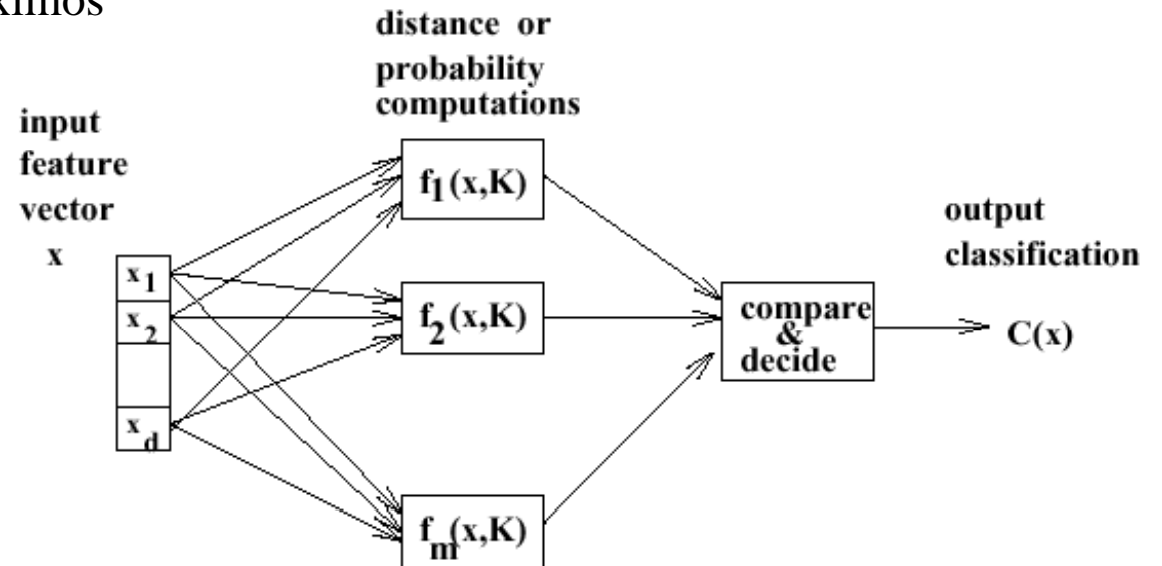
- classe de rejeição  $C_m = C_r$

- Sensores

- Extracção de características (*features*)

- Classificador

- assinatura mais próxima
    - k-vizinhos mais próximos
    - MAP
    - rede neuronal
    - árvore de decisão



- Definições:
  - o classificador comete um *erro de classificação* sempre que classifica um objecto na classe  $C_i$  e a verdadeira classe é  $C_j$  ( $i \neq j; C_i \neq C_r$ )
  - *taxa experimental de erros* é igual ao número de erros de classificação, cometidos nos dados de teste independentes, a dividir pelo número de testes efectuadas
  - *taxa experimental de rejeições* é igual ao número de rejeições feitas nos dados de teste independentes a dividir pelo número de testes efectuados
  - designa-se por *dados de teste independentes* a um conjunto de amostras de classe conhecida, incluindo objectos da classe rejeição, que não foram usados no desenvolvimento do algoritmo de extracção de características nem no algoritmo de classificação (não foram usados no treino)

## Avaliação do sistema (cont.)

- Matriz de confusão

		class j output by the pattern recognition system											
		'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	'R'	
true object class  i	'0'	97	0	0	0	0	0	1	0	0	1	1	
	'1'	0	98	0	0	1	0	0	1	0	0	0	
	'2'	0	0	96	1	0	1	0	1	0	0	1	
	'3'	0	0	2	95	0	1	0	0	1	0	1	
	'4'	0	0	0	0	98	0	0	0	0	2	0	
	'5'	0	0	0	1	0	97	0	0	0	0	2	
	'6'	1	0	0	0	0	1	98	0	0	0	0	
	'7'	0	0	1	0	0	0	0	98	0	0	1	
	'8'	0	0	0	1	0	0	1	0	96	1	1	
	'9'	1	0	0	0	3	0	0	0	1	95	0	

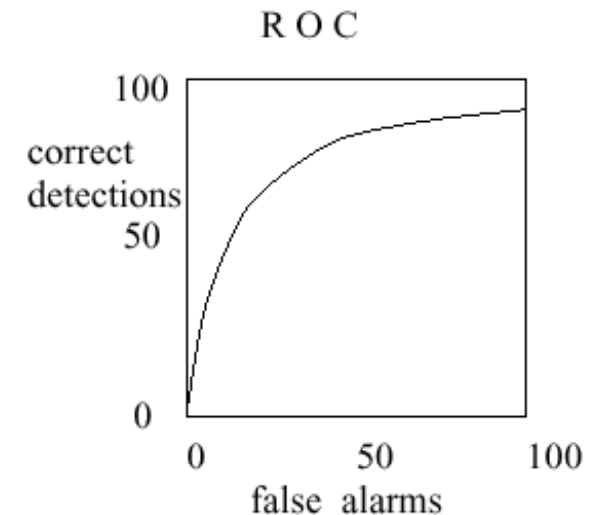
- Problemas envolvendo apenas 2 classes

- Exemplos

- objecto em bom estado / objecto em mau estado
    - objecto presente na imagem / objecto ausente
    - pessoa com doença D / pessoa sem doença D

- Conceitos

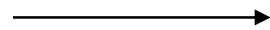
- Falsos alarmes (falso positivo) vs  
falhas de detecção (falso negativo)
    - Precisão vs rechamada



# Vectores de *features* e classificação

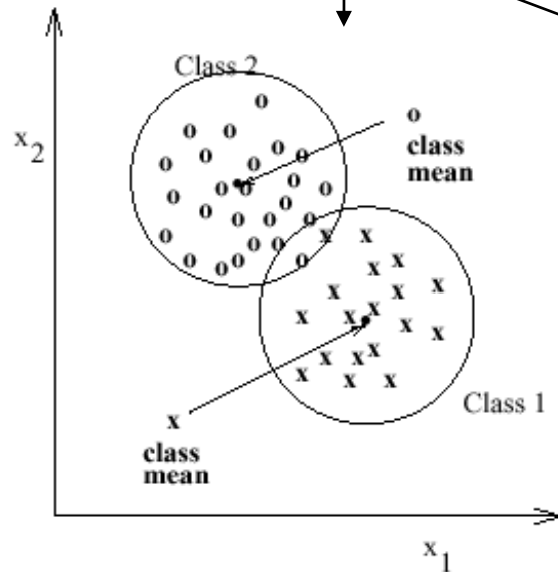
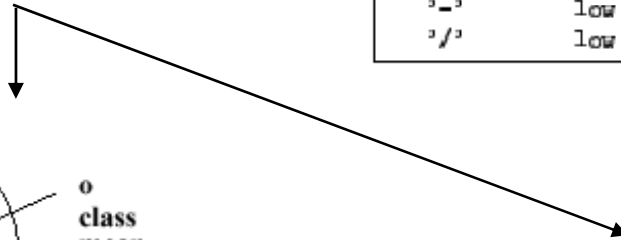
- Vectores de *features*

- 8D

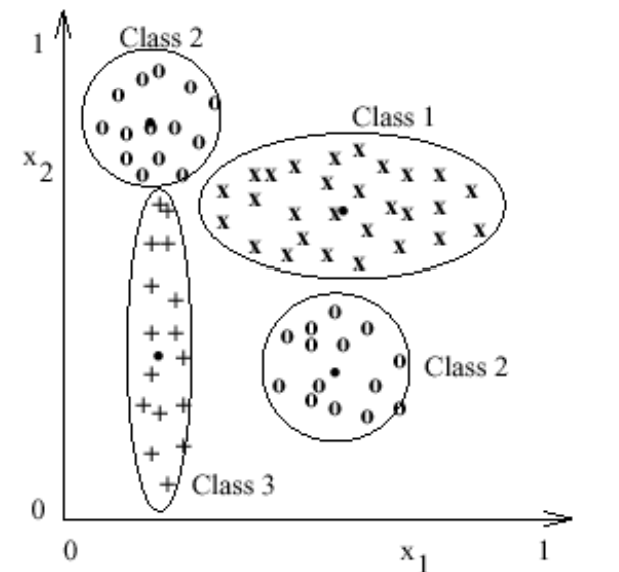


(class)				number	number	(cx,cy)	best	least
character	area	height	width	#holes	#strokes	center	axis	inertia
'A'	medium	high	3/4	1	3	1/2,2/3	90	medium
'B'	medium	high	3/4	2	1	1/3,1/2	90	large
'8'	medium	high	2/3	2	0	1/2,1/2	90	medium
'0'	medium	high	2/3	1	0	1/2,1/2	90	large
'1'	low	high	1/4	0	1	1/2,1/2	90	low
'W'	high	high	1	0	4	1/2,2/3	90	large
'X'	high	high	3/4	0	2	1/2,1/2	?	large
'*'	medium	low	1/2	0	0	1/2,1/2	?	large
'-'	low	low	2/3	0	1	1/2,1/2	0	low
'/'	low	high	2/3	0	1	1/2,1/2	60	low

- 2D



$$\|\mathbf{x} - \mathbf{x}_c\| = \sqrt{\sum_{i=1}^d (\mathbf{x}_1[i] - \mathbf{x}_2[i])^2}$$



$$\|\mathbf{x} - \mathbf{x}_c\| = \sqrt{\sum_{i=1}^d ((\mathbf{x}_1[i] - \mathbf{x}_2[i]) / \sigma_i)^2}$$

# Classificador – k-vizinhos mais próximos

---

**S** is a set of  $n$  labeled class samples  $s_i$  where  $s_i.x$  is a feature vector and  $s_i.c$  is its integer class label.

**x** is the unknown input feature vector to be classified.

**A** is an array capable of holding up to  $k$  samples in sorted order by distance  $d$ .

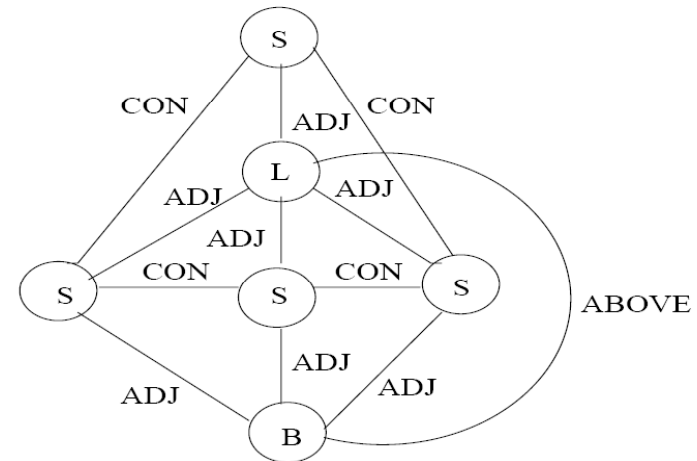
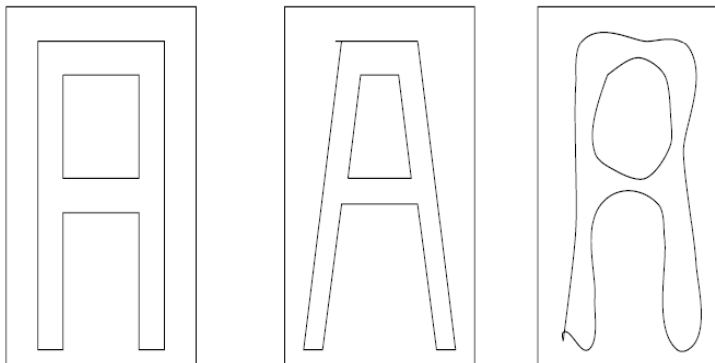
The value returned is a class label in the range  $[1, m]$

```
procedure K_Nearest_Neighbors(x, S)
{
  make A empty;
  for all samples  $s_i$  in S
  {
     $d$  = Euclidean distance between  $s_i$  and x;
    if A has less than  $k$  elements then insert  $(d, s_i)$  into A;
    else if  $d$  is less than max A
      then {
        remove the max from A;
        insert  $(d, s_i)$  in A;
      }
  } ;
  assert A has  $k$  samples from S closest to x;
  if a majority of the labels  $s_i.c$  from A are class  $c_0$ 
    then classify x into class  $c_0$ ;
    else classify x into the reject class;
  return(class-of-x);
}
```

# Representação por Grafo de Estrutura

0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	0
0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	1	0
0	1	0	0	0	1	1	1	0	0
0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0
0	1	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	1	1	1	1	1	1	0
0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	1	0
0	0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0





# Classificador - árvore de decisão

**input:** feature vector with [ #holes, #strokes, moment of inertia ]

**output:** class of character

case of #holes

0: character is 1, W, X, \*, -, or /

case of moment about axis of least inertia

low: character is 1, -, or /

case of best axis direction

0: character is -

60: character is /

90: character is 1

large: character is W or X

case of #strokes

2: character is X

4: character is W

1: character is A or 0

case of #strokes

0: character is o

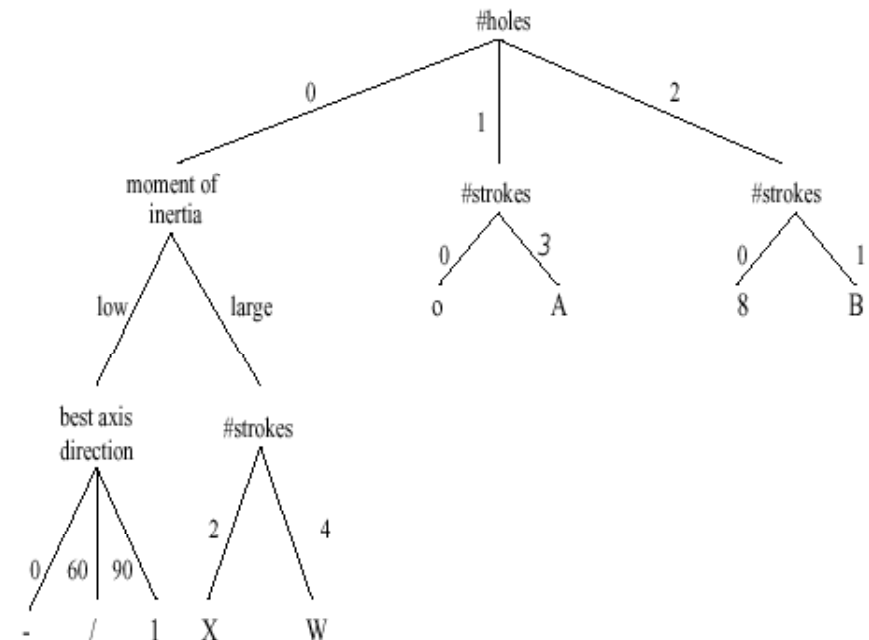
3: character is A

2: character is B or 8

case of #strokes

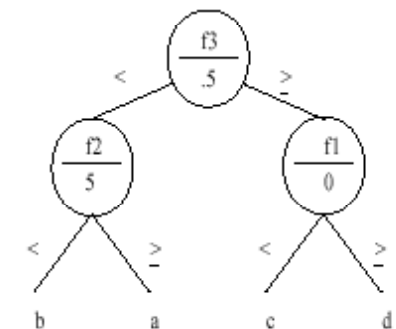
0: character is 8

1: character is B



f1	f2	f3	CLASS
3	6	0	a
-5	9	1	c
4	5	1	d
7	4	0	b
-1	10	0	a
2	6	1	d
-2	2	1	c
-1	3	0	b

Training Data

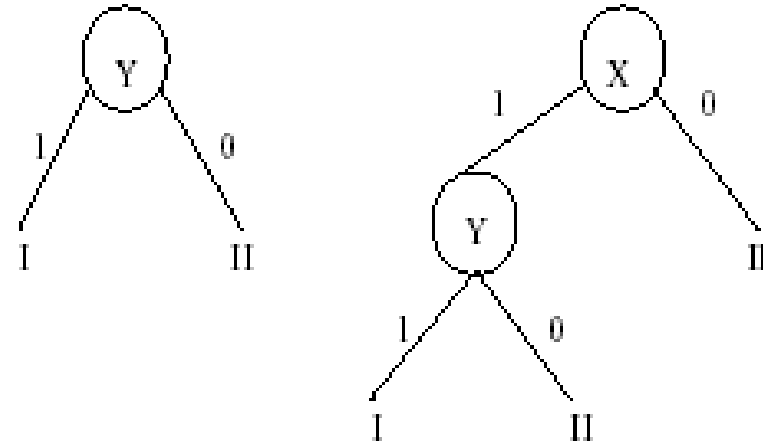


Decision Tree

## Exemplo simples

---

X	Y	Z	Class C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II



Training Data

Two Possible Decision Trees

- Necessidade de automatizar o processo: problemas reais podem facilmente envolver dezenas ou mesmo centenas de *features*
- No caso das *features* tomarem valores não binários, como escolher os limiares de decisão?

## Teoria de informação e as árvores de decisão

---

- A entropia de um conjunto de acontecimentos  $x = \{x_1, x_2, \dots, x_n\}$  é dada por

$$H(x) = - \sum_{i=1}^n p_i \log_2 p_i$$

onde  $p_i$  é a probabilidade do acontecimento  $x_i$

- Dado um conjunto de *features* seleccionadas, quais são aquelas que são mais informativas para o processo de classificação?
- O conteúdo de informação que a *feature*  $F$  introduz, para a determinação da classe,  $C$ , é dada pela seguinte expressão

$$I(C; F) = \sum_{i=1}^m \sum_{j=1}^d P(C = c_i, F = f_j) \log_2 \frac{P(C = c_i, F = f_j)}{P(C = c_i)P(F = f_j)}$$

↑  
Informação mútua

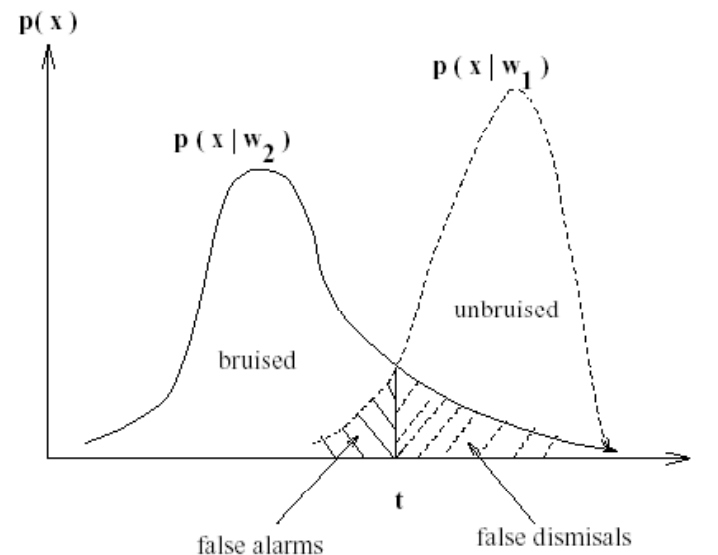
# Classificador MAP

- Classificador MAP: um objecto é classificado na classe que é mais provável pertencer, após observação das suas *features*

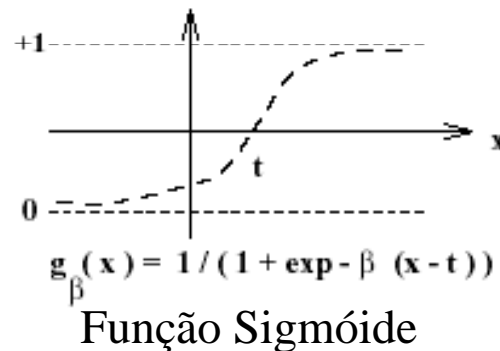
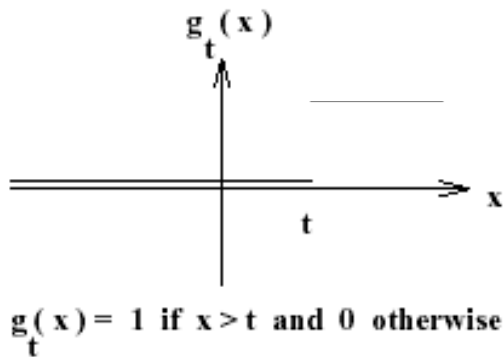
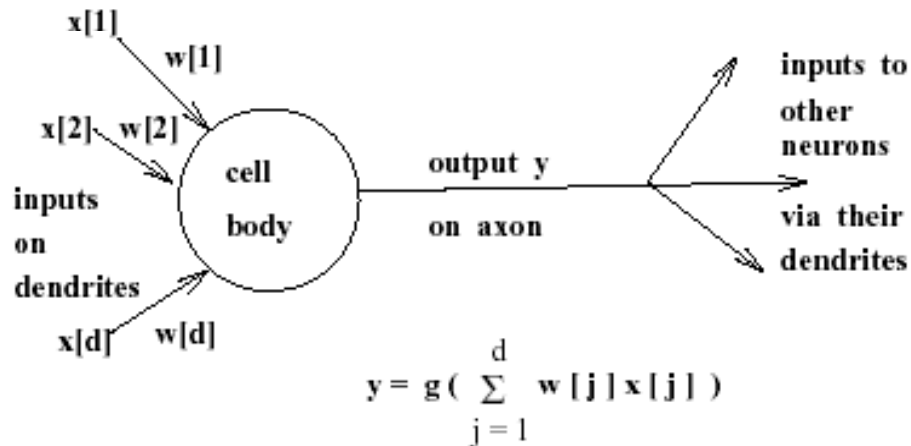
$$P(w_i | x) = \frac{p(x | w_i)P(w_i)}{p(x)}$$

- Informação necessária:

- Distribuição do vector de *features*, para cada classe  $\longrightarrow p(x | w_i)$
- Probabilidades a priori de cada classe  $\longrightarrow P(w_i)$



Artificial Neuron (AN)



- Equação do hiperplano

$$\sum_{j=1}^d w_j x_j - t = 0$$

$$\sum_{j=0}^d w_j x_j = 0 \quad \begin{matrix} w_0 = -t \\ x_0 = 1.0 \end{matrix}$$

- Espaço de features aumentado:

$$\tilde{x} = (1 \quad x)$$

- Entrada/saída em cada neurónio

$$y = g \left( \sum_{j=0}^d w_j x_j \right)$$

# Discriminantes lineares – percepção de incremento simples

---

Aprender  $\leftrightarrow$  Adaptar pesos

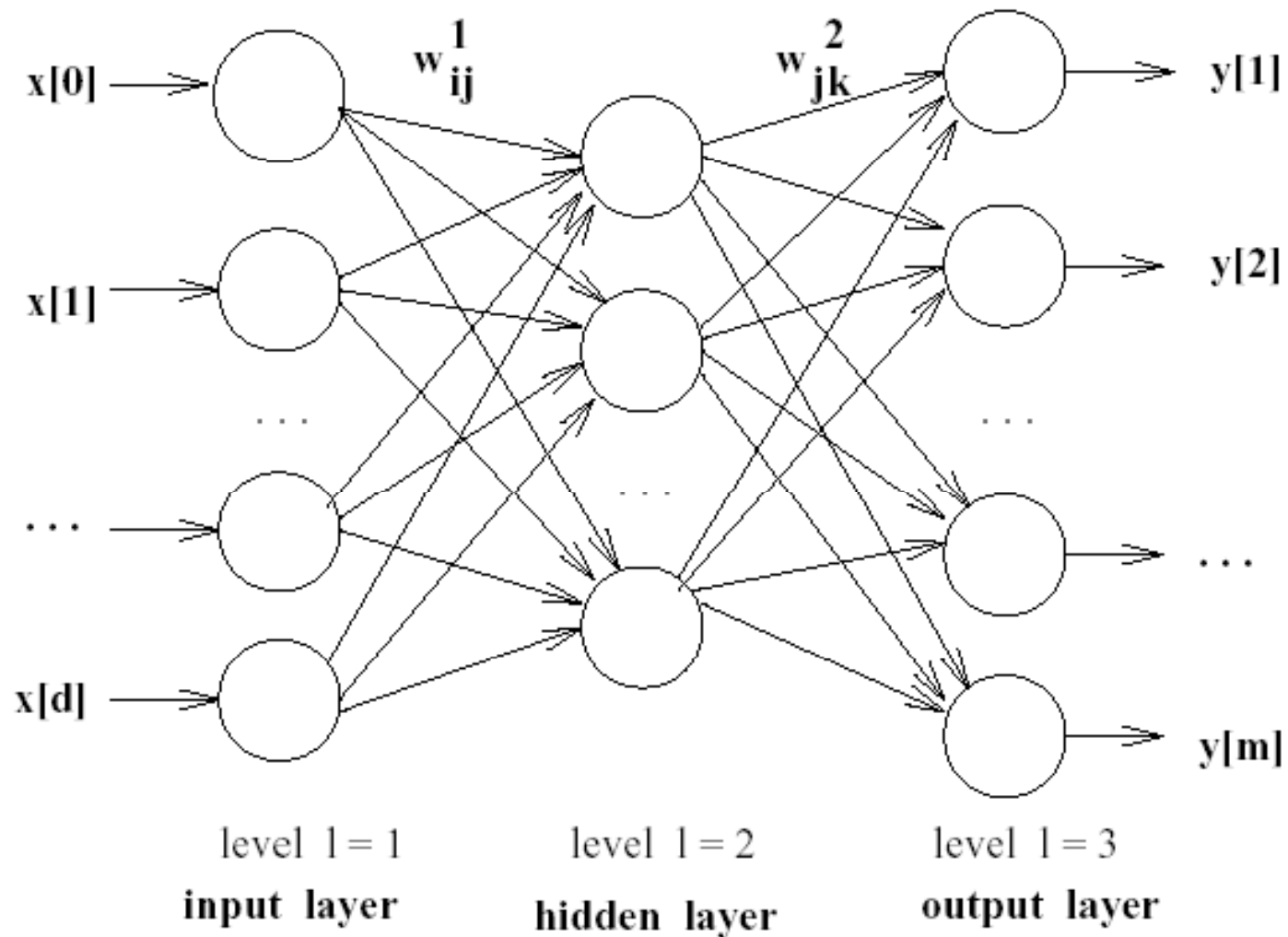
Compute weight vector  $w$  to discriminate Class 1 from 2.  
 $S1$  and  $S2$  are sets of  $n$  samples each.  
 $gain$  is a scale factor used to change  $w$  when  $x$  is misclassified.  
 $max\_passes$  is maximum number of passes through all training samples.

```
procedure Perceptron_Learning( $gain, max\_passes, S1, S2$ )
{
  input sample sets  $S1$  and  $S2$ ;
  choose weight vector  $w$  randomly;
  "let NE be the total number of samples misclassified"
   $NE = check\_samples ( S1, S2, w )$ ;
  while (  $NE > 0$  and  $passes < max\_passes$  )
  {
    training_pass (  $S1, S2, w, gain$  );
     $NE = check\_samples ( S1, S2, w )$ ;
     $gain = 0.5 * gain$ ;
     $passes = passes + 1$ ;
  }
  report number of errors NE and weight vector  $w$ ;
}

procedure training_pass (  $S1, S2, w, gain$  );
{
  for  $i$  from 1 to size of  $S_k$ 
  {
    "scalar, or dot, product  $\circ$  implements AN computation"
    take next  $x$  from  $S1$ ;
    if (  $w \circ x > 0$  )  $w = w - gain * x$ ;
    take next  $x$  from  $S2$ ;
    if (  $w \circ x < 0$  )  $w = w + gain * x$ ;
  }
}
```

## Rede *feedforward* multicamada

---



Aprendizagem  $\rightarrow$  algoritmo de retropropagação