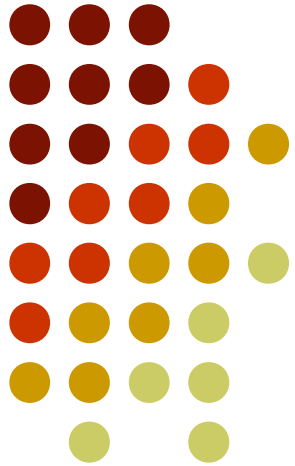
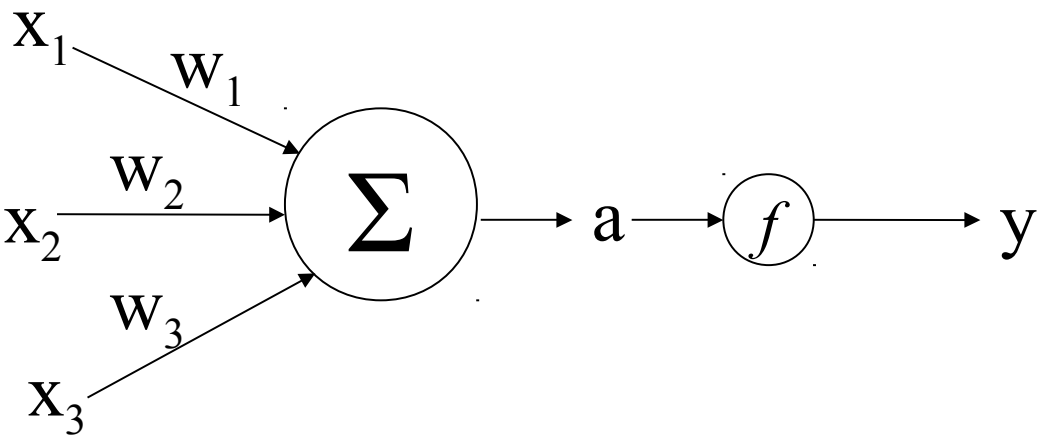
PGCA009 –Inteligência Computacional
Aula 3 Redes Neurais

Prof. Angelo Loula

Mestrado em Computação Aplicada (UEFS)



Neurônios Artificiais

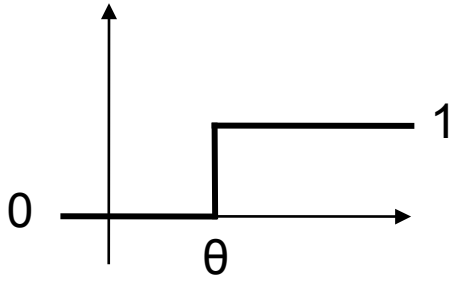


Threshold Unit or Threshold Logic Unit (TLU)

$$a = x_1 w_1 + x_2 w_2 + x_3 w_3$$

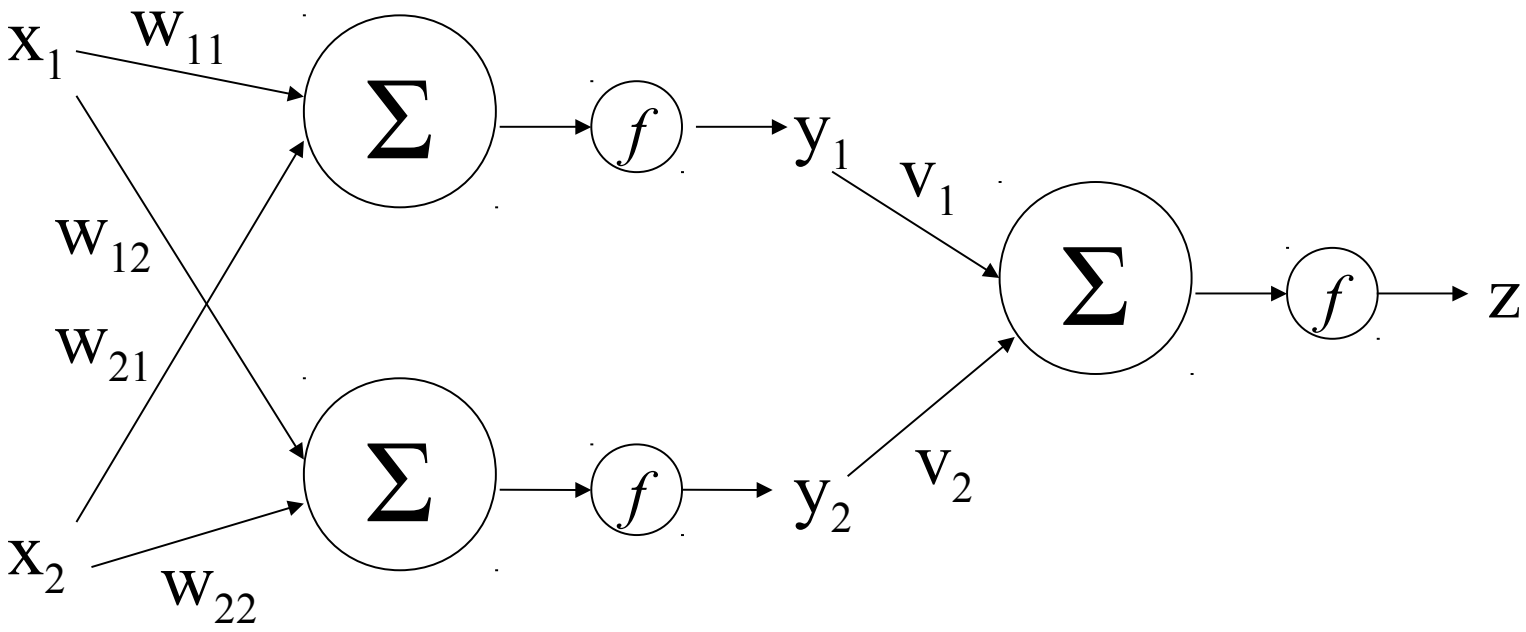
$$y = f(x_1 w_1 + x_2 w_2 + x_3 w_3)$$

$$f(a) = \begin{cases} 1, & a => \theta \\ 0, & a < \theta \end{cases}$$



limiar

Redes Neurais Artificiais



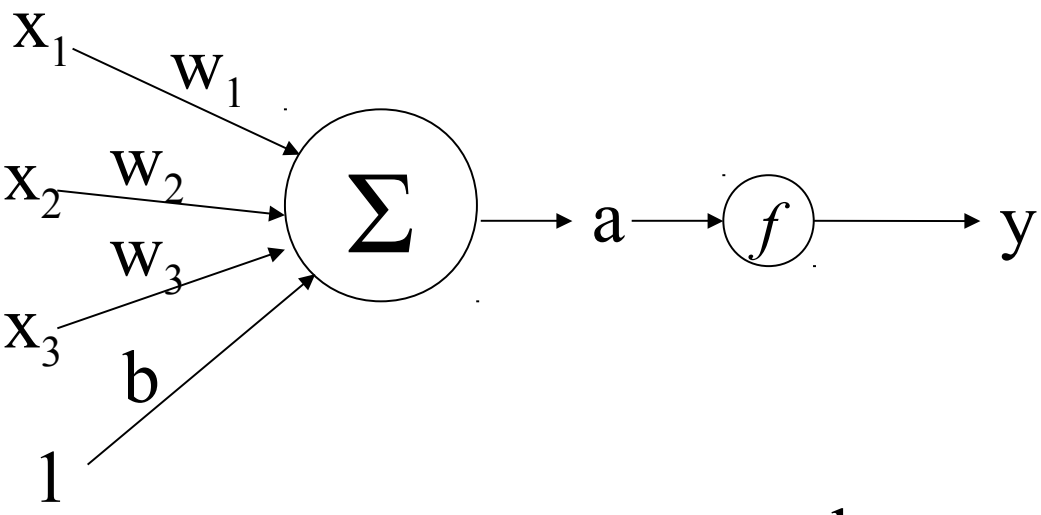
Redes Neurais Artificiais

- Redes de Neurônios TLU
  - Realizam funções booleanas
  - Parecem com portas lógicas
  - Difíceis de projetar para problemas mais complexos
  - Sem um algoritmo de treinamento/aprendizado

Redes Neurais Artificiais

- Perceptron
  - Descrito por Rosenblatt, 1958
  - $X \in [0,1] \subset \Re$
  - Rede de uma única camada ainda limitada
    - Separação linear
  - Rede multicamada pode realizar muito mais
    - Separação de conjuntos convexos ou arbitrários

Neurônios Artificiais

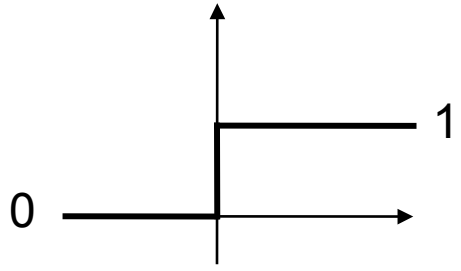


$$a = x_1w_1 + x_2w_2 + x_3w_3 + b$$

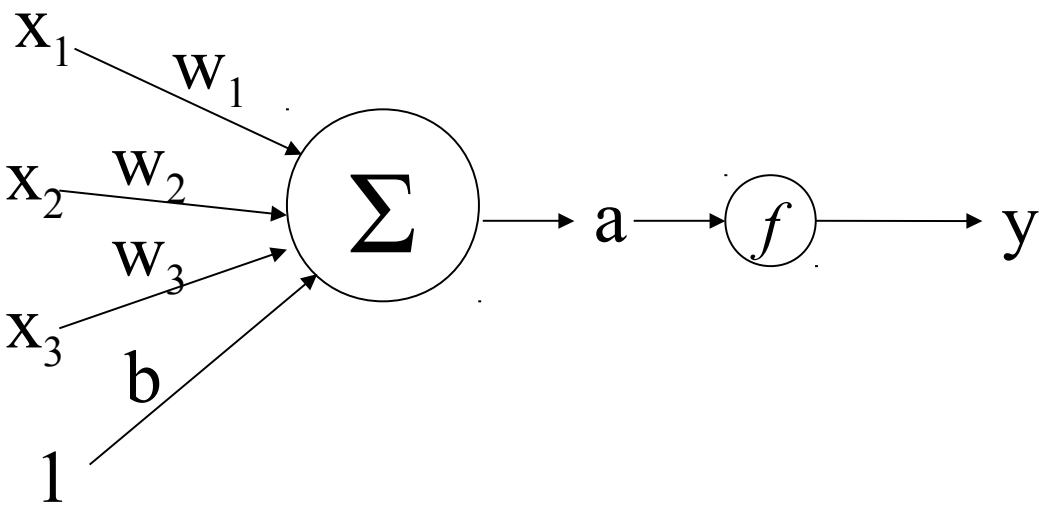
$$y = f(x_1w_1 + x_2w_2 + x_3w_3 + b)$$

b → bias
substitui o limiar
E se não houvesse?

$$f(a) = \begin{cases} 1, & a => 0 \\ 0, & a < 0 \end{cases}$$



Neurônios Artificiais



b pode ser visto como um peso para uma entrada constante 1 então

$$X = (x_1, x_2, x_3, 1)$$
$$W = (w_1, w_2, w_3, b)$$
$$a = X \cdot W$$

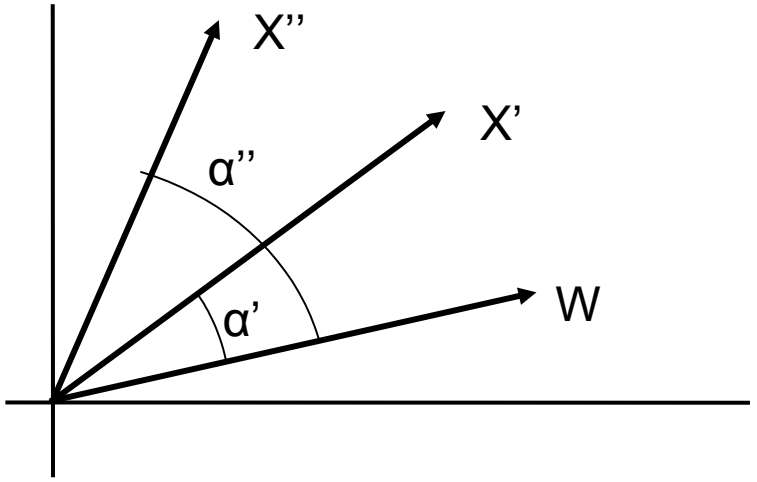
$$y = f(a) = \begin{cases} 1, & a => 0 \\ 0, & a < 0 \end{cases}$$

Neurônios Artificiais

$X = (x_1, x_2, x_3, 1)$

$W = (w_1, w_2, w_3, b)$

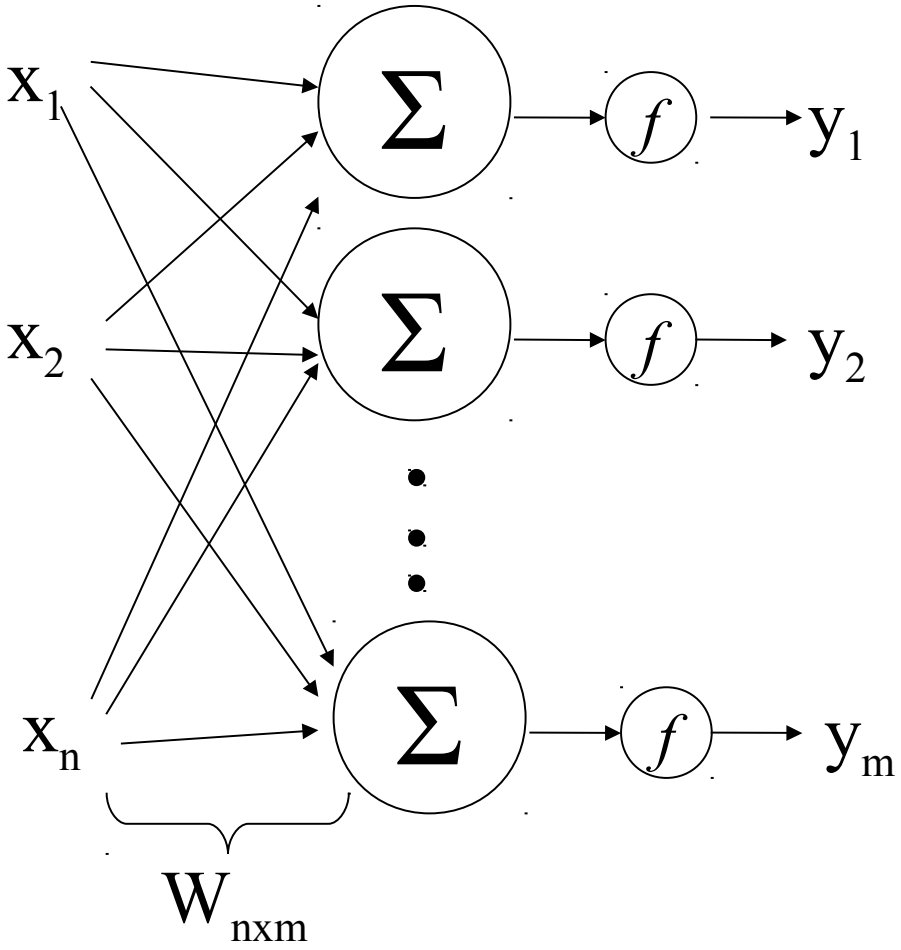
$a = X \cdot W = ||X||.||W||.\cos \alpha$



Ativação de um neurônio é a projeção de X em W, quanto X se assemelha a W

Redes Neurais Artificiais

- Uma camada



Redes Neurais Artificiais

- Uma camada
  - Ativação de cada neurônio para uma mesma entrada identifica os neurônios que reconhecem (são mais similares) à entrada
    - Mas somente se os pesos estiverem normalizados

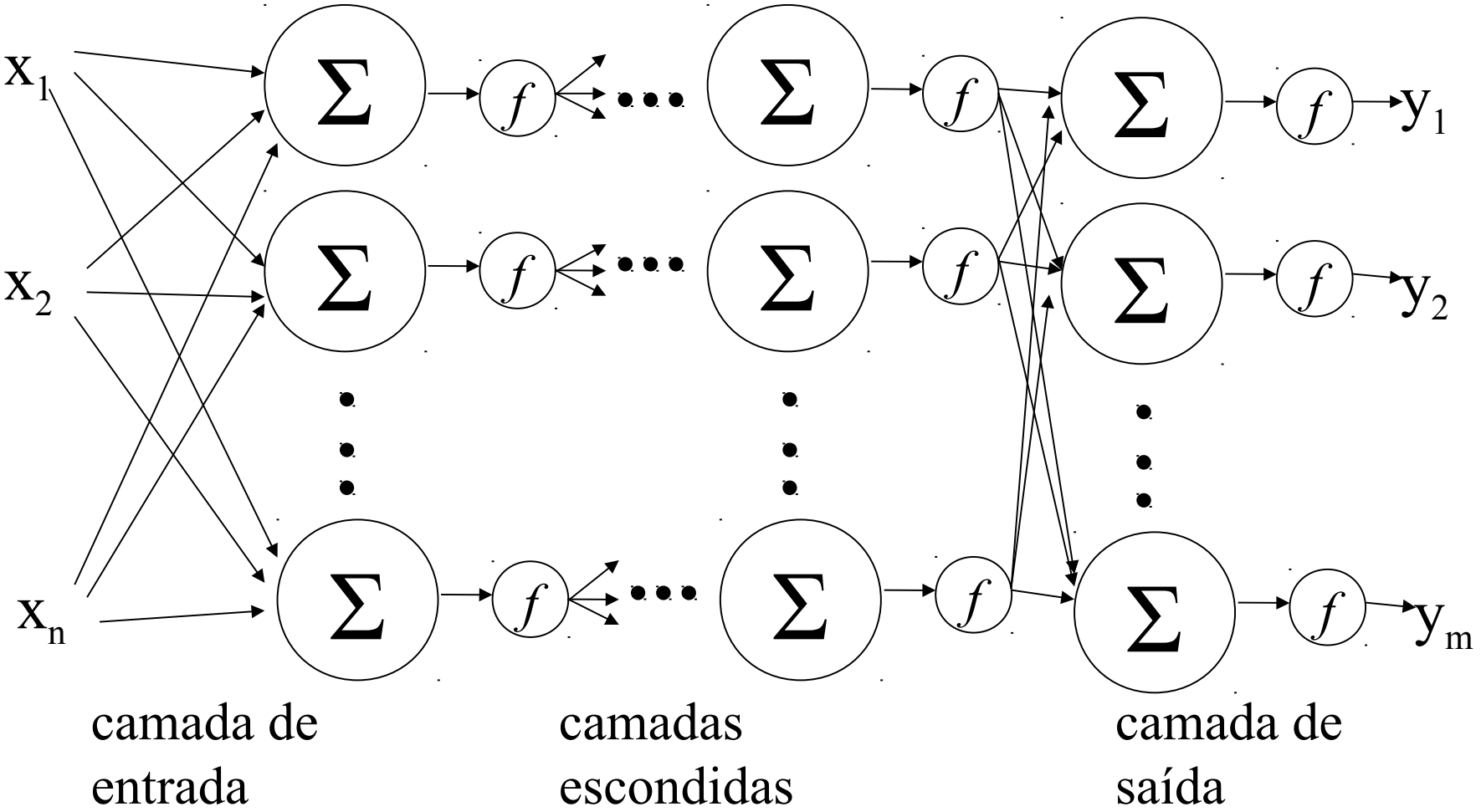
$$X = (x_1, x_2, x_3, 1)$$

$$W = (w_1, w_2, w_3, b)$$

$$a = X \cdot W = \|X\| . \|W\| . \cos \alpha$$

Redes Neurais Artificiais

- Múltiplas camada



Redes Neurais Artificiais

- Múltiplas camada
  - Neurônios de uma camada somente são conectados em neurônios da camada seguinte
  - Normalmente não há necessidade para mais de 2 camadas
  - Cada conexão entre camadas envolvem pesos entre cada par de neurônios
  - Aumento considerável do número de pesos a ajustar!
  - Não há um método exato para determinar o número de neurônios na camada intermediária(escondida)

Redes Neurais Artificiais

- Múltiplas camada
  - Uma rede perceptron com uma camada intermediária tem capacidade de **aproximação universal**!
    - Mas depende do número de neurônios mas
    - É preciso ajustar sua flexibilidade de aproximação...

Redes Neurais Artificiais

- Múltiplas camada
  - O projeto de uma rede multi-camada envolve
    - topologia da rede
    - função de transferência de cada neurônio individual
    - estratégia de aprendizado
    - dados de treinamento
  - Normalmente especificamos por projeto topologia e função de transferência
    - mas isso pode não ser o mais adequado ao problema

Redes Neurais Artificiais

- Funções de transferência

Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$		hardlims
Linear	$a = n$		purelin
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$		satlins

Redes Neurais Artificiais

- Funções de transferência

Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \le n \le 1$ $a = 1 \quad n > 1$		satlins
Log-Sigmoid	$a = \frac{1}{1+e^{-n}}$		logsig
Hyperbolic Tangent Sigmoid	$a = \frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$		tansig
Positive Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \le n$		poslin
Competitive	$a = 1$ neuron with max n $a = 0$ all other neurons		compet

Redes Neurais Artificiais

- Entradas e Saídas
  - Dimensão da entrada/ saída → Quantidade de neurônios de entrada/saída
  - Codificação: dependente do problema
  - Codificação Local: um neurônio por objeto
  - Codificação Distribuída: cada característica dos objetos em um neurônio
  - Normalização
  - Pré-processamento e Pós-processamento

Redes Neurais Artificiais

- Precisamos de um método de treinamento da rede neural artificial
  - Aprendizado supervisionado
    - Exemplos de entrada e saída: conjunto de treino
    - Comparação entre saída atual e esperada
  - Aprendizado por reforço
    - Sem saída esperada, somente uma nota
  - Aprendizado não-supervisionado
    - Ajustes baseados somente nas entradas

Redes Neurais Artificiais

- Treinamento da rede neural de uma camada
  - Regra iterativa de aprendizado supervisionado

```
Faça
   W=W+ΔW
Até(critério de parada)
```

  - ΔW?

Redes Neurais Artificiais

- Regra de Aprendizado Perceptron
  - Um neurônio com função de transferência hardlim

erro e = t - y

y: saída da rede t: saída esperada

$W_{novo} = W_{anterior} + \rho \cdot e \cdot X$ ρ:taxa de aprendizado entre 0 e 1

  - Entradas e saídas binárias

$$\text{erro } e = \begin{cases} 0,\ t = y \text{ então nada a ajustar} \\ 1,\ t = 1\ y = 0 \text{ então W se aproxima de X} \\ -1,\ t = 0\ y = 1 \text{ então W se afasta de X} \end{cases}$$

Neurônios Artificiais

- Treinamento de um neurônio

Considere a seguinte função e treine um neurônio com pesos iniciais aleatórios para realizar esta função!

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Neurônios Artificiais

bias

x_1	x_2	x_3	t
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	1

Inicialize os pesos com valores pequenos e distribuídos em torno de zero.

w	x	y	e	w_{novo}

x_1	x_2	x_3 bias	t
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	1

$\rho = 0.5$

	W	X	y	e	W_{novo}
1	(-0.1,-0.2,0.2)	(0,0,1)	1	-1	(-0.1,-0.2,-0.3)
2	(-0.1,-0.2,-0.3)	(0,1,1)	0	0	
3	(-0.1,-0.2,-0.3)	(1,0,1)	0	0	
4	(-0.1,-0.2,-0.3)	(1,1,1)	0	1	(0.4,0.3,0.2)
5	(0.4,0.3,0.2)	(0,0,1)	1	-1	(0.4,0.3,-0.3)
6	(0.4,0.3,-0.3)	(0,1,1)	1	-1	(0.4,-0.2,-0.8)
7	(0.4,-0.2,-0.8)	(1,0,1)	0	0	
8	(0.4,-0.2,-0.8)	(1,1,1)	0	1	(0.9,0.3,-0.3)

x_1	x_2	x_3 bias	t
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	1

$\rho = 0.5$

	W	X	y	e	W_{novo}
	...	...	...	...	...
	...	...	...	...	...
33	(1.4,0.8,-1.8)	(0,0,1)	0	0	
34	(1.4,0.8,-1.8)	(0,1,1)	0	0	
35	(1.4,0.8,-1.8)	(1,0,1)	0	0	
36	(1.4,0.8,-1.8)	(1,1,1)	1	0	

Redes Neurais Artificiais

- Treinamento como otimização
  - Erro quadrático médio

$$MSE = \frac{1}{N} \sum_{j=1}^{N} \sum_{i} \left(t_i^j - y_i^j \right)^2$$

N: quantidade de exemplos para treino

$$MSE = \frac{1}{N} \sum_{j=1}^{N} \sum_{i} \left(t_i^j - f\left(\sum_{k} w_{jk} \cdot x_k^j \right) \right)^2$$

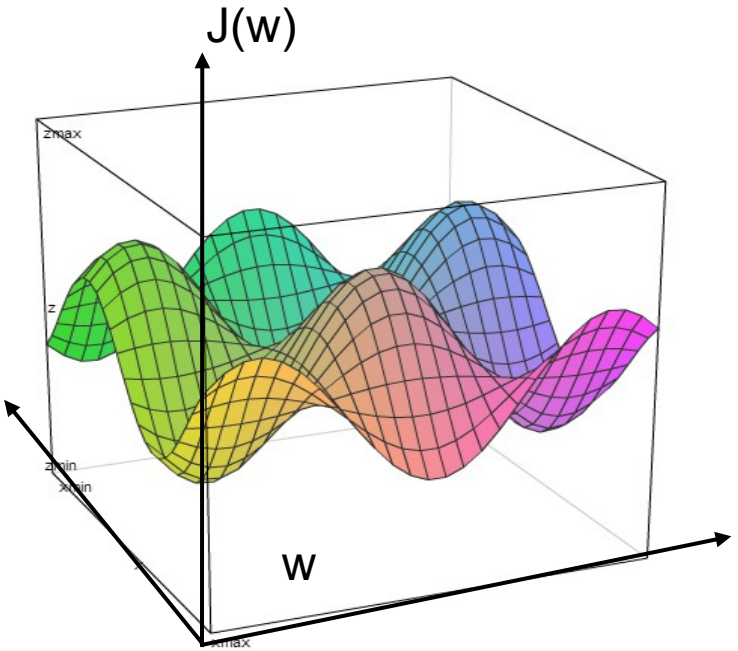
  - MSE depende fundamentalmente do ajuste dos pesos *w*

Redes Neurais Artificiais

- Treinamento como otimização

$$J(w) = MSE = \frac{1}{N} \sum_{j=1}^{N} \sum_{i} \left(t_i^j - f\left(\sum_k w_{jk} \cdot x_k^j \right) \right)^2$$

  - Define uma superfície de erro



Redes Neurais Artificiais

- Treinamento como otimização

$$J(w) = MSE = \frac{1}{N} \sum_{j=1}^{N} \sum_{i} \left(t_i^j - f\left(\sum_k w_{jk} \cdot x_k^j \right) \right)^2$$

  - Podemos usar o método do gradiente para buscar a direção que leva a um valor de w que minimize J(w)
    - Converge para um mínimo local (depende da escolha dos pesos iniciais)

Redes Neurais Artificiais

- Regra de Aprendizado Delta
  - Um neurônio com função de transferência f

$$\Delta w_{ij} = -\rho . \frac{\partial J(w)}{\partial w}$$

$$\Delta w_{ij} = \rho (t_j - y_j) f'(a_j) x_i$$

$$a_j = \sum_i w_{ij} x_i$$

  - É necessário que f seja diferenciável!

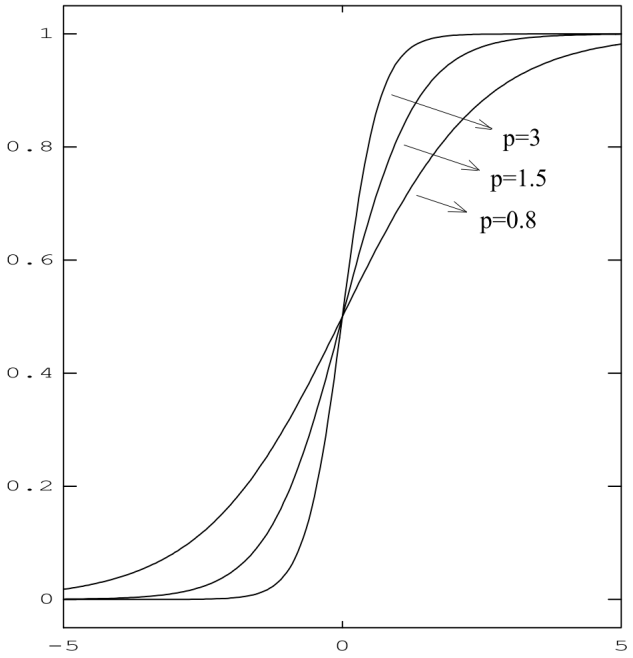
Redes Neurais Artificiais

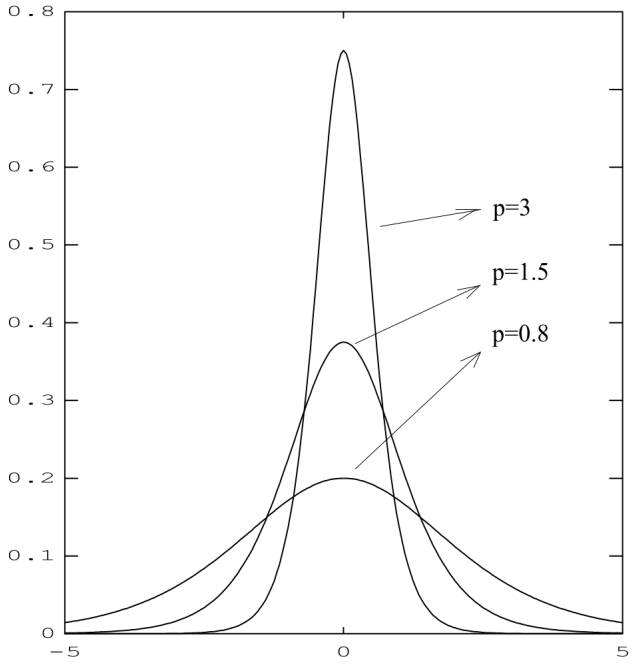
- Sigmóide

$$f(x) = \frac{1}{1 + e^{-px}}$$

$$f'(x) = \left(\frac{1}{1 + e^{-px}}\right)\left(\frac{1 + e^{-px} - 1}{1 + e^{-px}}\right)$$

$$f'(x) = p \,.\, f(x)(1 - f(x))$$





Redes Neurais Artificiais

- Regra de Aprendizado Delta
  - Para uma sigmóide:

$$\Delta w_{ij} = \rho \left(t_j - y_j \right) p \,.\, f\left(a_j \right) \left(1 - f\left(a_j \right) \right) x_i$$

$$a_j = \sum_i w_{ij} \, x_i$$