



Redes neurais

Jones Granatyr


Contexto

- ▶ Problemas que são resolvidos por algoritmos pré-determinados (sistemas de recomendação, buscas, grafos, ordenações)
- ▶ Algoritmo pré-definido para reconhecimento facial?
Processamento de linguagem natural?
- ▶ Algumas aplicações
 - Descoberta de novos remédios
 - Entendimento de linguagem natural
 - Carros autônomos
 - Reconhecimento facial
 - Cura para doenças
 - Bolsa de valores
 - Encontrar soluções para controle de tráfego
- ▶ Muitos dados e problemas complexos

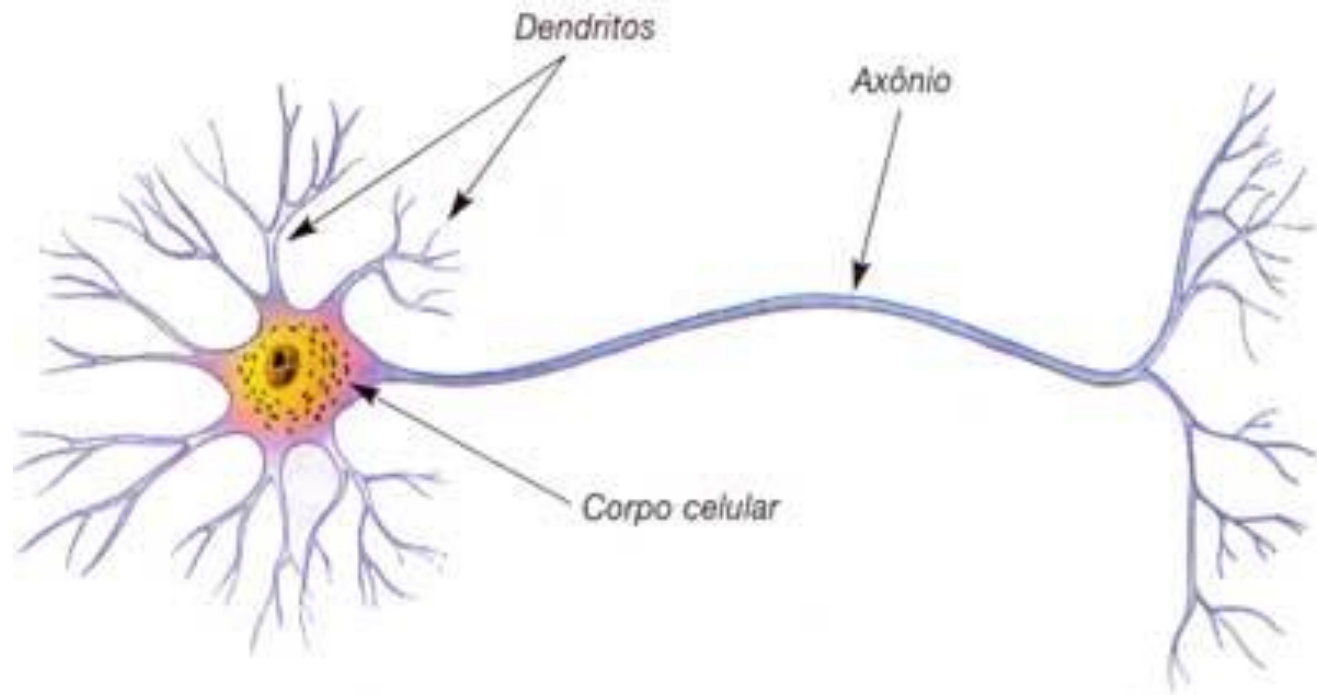
Redes neurais



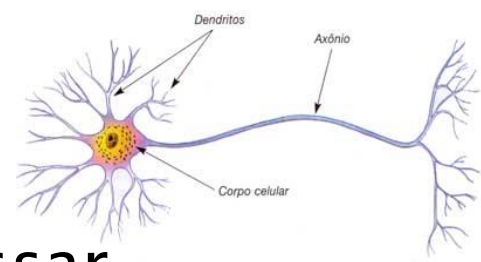
Redes neurais

- ▶ Imitar o sistema nervoso de humanos no processo de aprendizagem
 - ▶ Inspirada em redes neurais biológicas
 - ▶ Parecido com a troca de informações em uma rede biológica
 - ▶ Com deep learning (aprendizagem profunda) as redes neurais ficaram populares novamente
- 

Neurônio



Neurônio



- ▶ **Neurônios:** o cérebro usa para processar informações
- ▶ **Axônio:** transmite o sinal de um neurônio para outro (sinais elétricos, sinapses) – conecta os neurônios
- ▶ Substâncias químicas são lançadas das sinapses e entram pelos dendritos, aumentando ou baixando o potencial elétrico do corpo da célula
- ▶ O neurônio dispara se a entrada é maior que um número definido (liga ou não liga)

Redes neurais

- ▶ Fornece um valor de entrada, a rede processa e retorna uma resposta
- ▶ O neurônio é ativado somente se o valor for maior que um limiar

Entrada

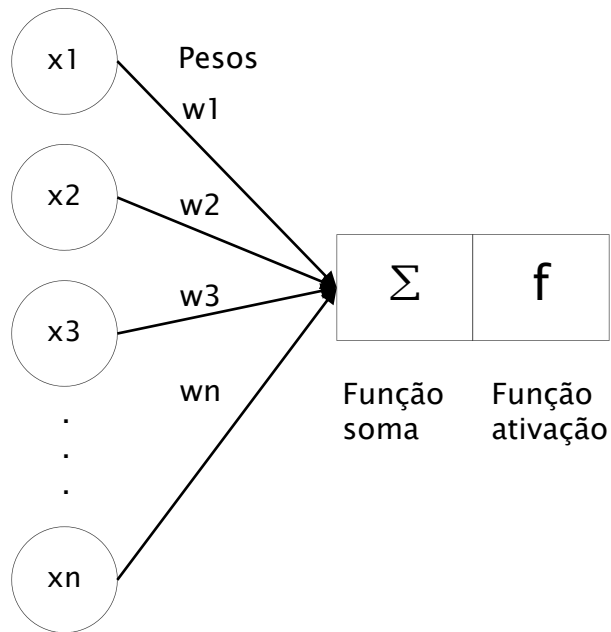
A blue rectangular box with a double border, representing a neural network component.

Neurônios e
axônios

Saída

Neurônio artificial

Entradas



1943 – McCulloch e Pitts

1958 – Frank Rosenblatt (perceptron)

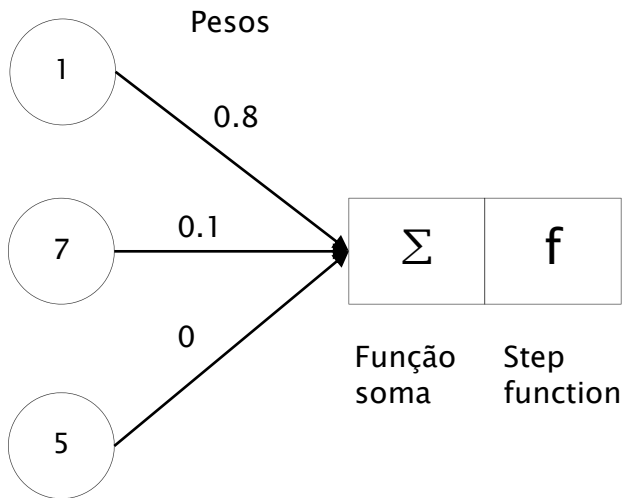
$$soma = \sum_{i=1}^n x_i * w_i$$

Neurônio artificial

$$soma = \sum_{i=1}^n x_i * w_i$$

$$soma = (1 * 0.8) + (7 * 0.1) + (5 * 0)$$

Entradas



Step function (função Degrau)

Maior do que zero = 1
Caso contrário = 0

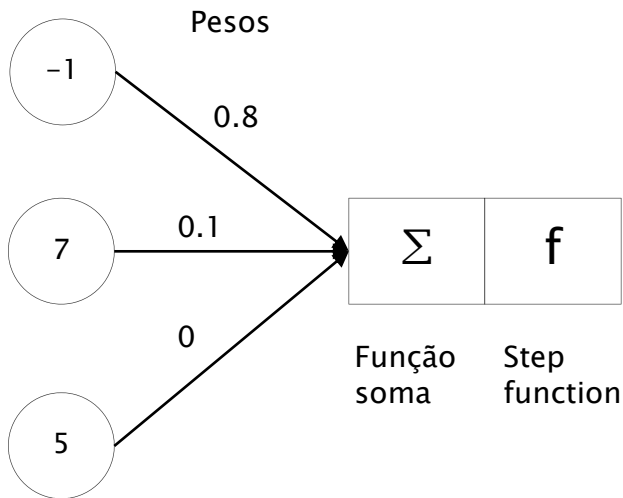
Representação tudo ou nada

Neurônio artificial

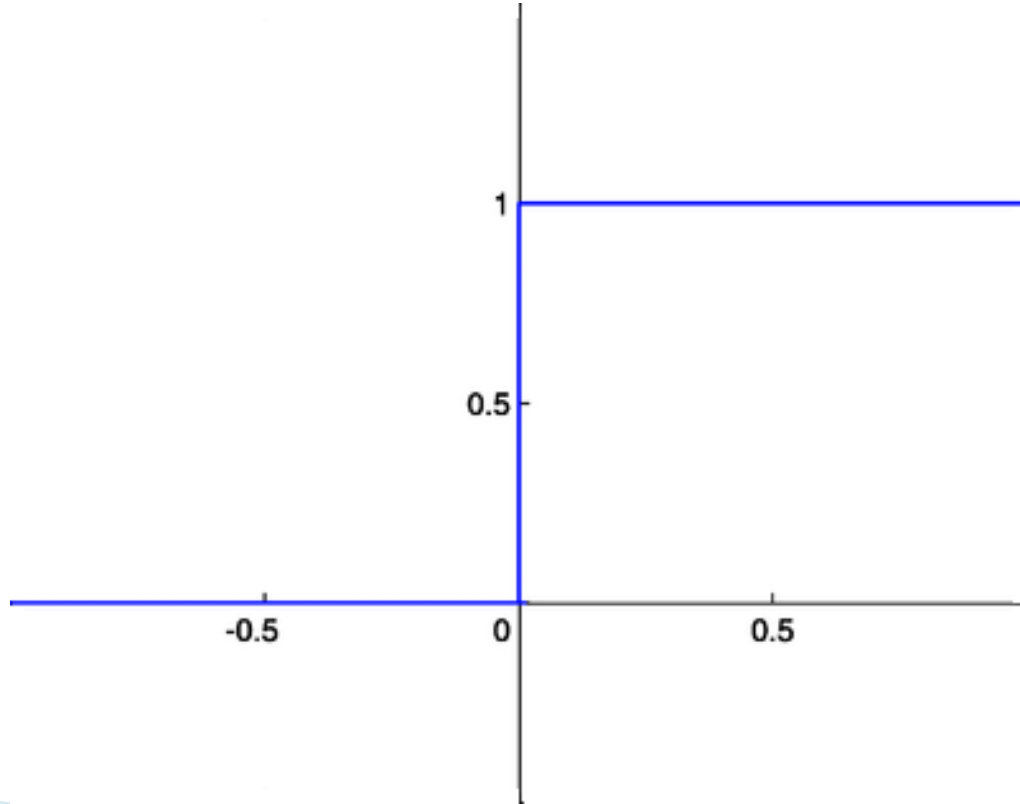
$$soma = \sum_{i=1}^n x_i * w_i$$

$$soma = (-1 * 0.8) + (7 * 0.1) + (5 * 0)$$


Entradas



Step function



Redes neurais

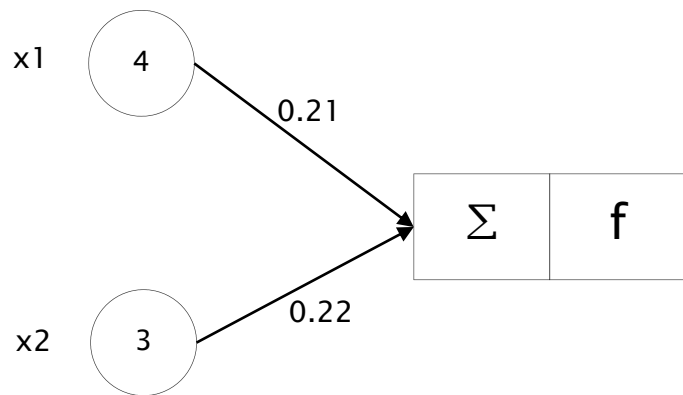
- ▶ Peso positivo – sinapse excitadora
 - ▶ Peso negativo – sinapse inibidora
 - ▶ Pesos são sinapses
 - ▶ Pesos amplificam ou reduzem o sinal de entrada
 - ▶ Conhecimento da rede neural são os pesos
- 

Classificação

x1 – comprimento do parafuso

x2 – diâmetro do parafuso

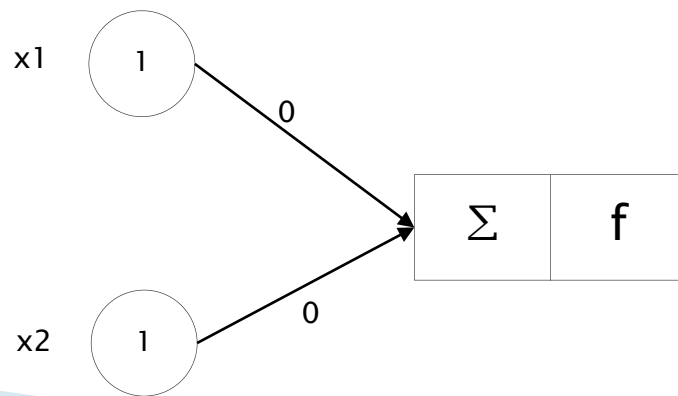
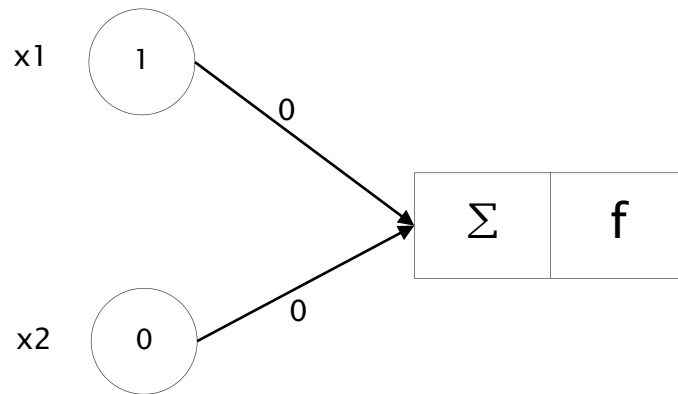
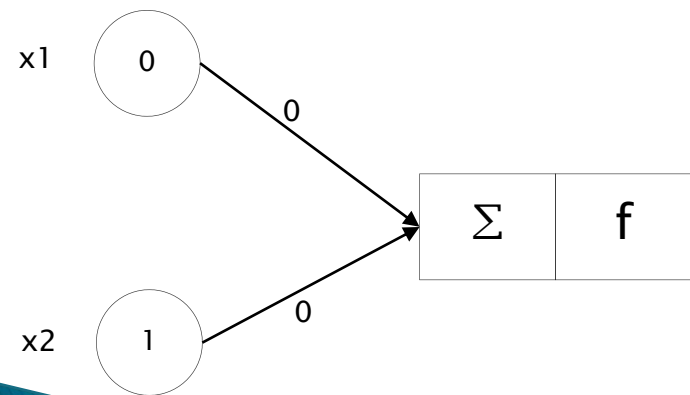
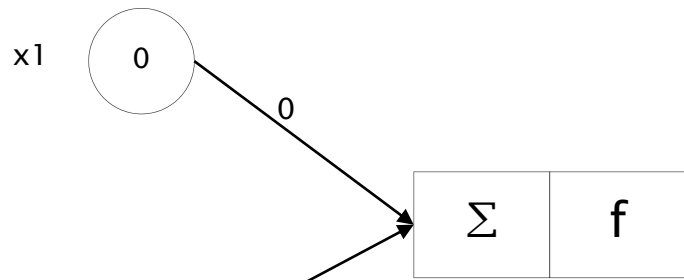
Classe A (0) e Classe B (1)



$$soma = \sum_{i=0}^n x_i * w_i$$

Operador E

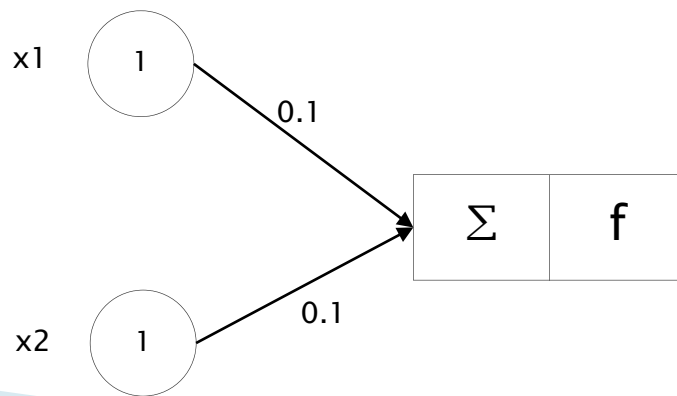
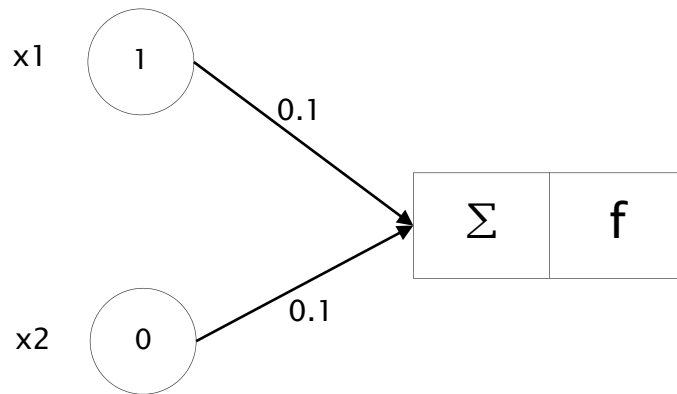
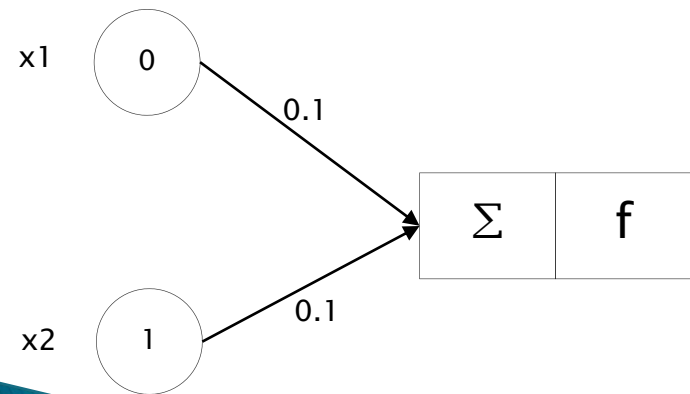
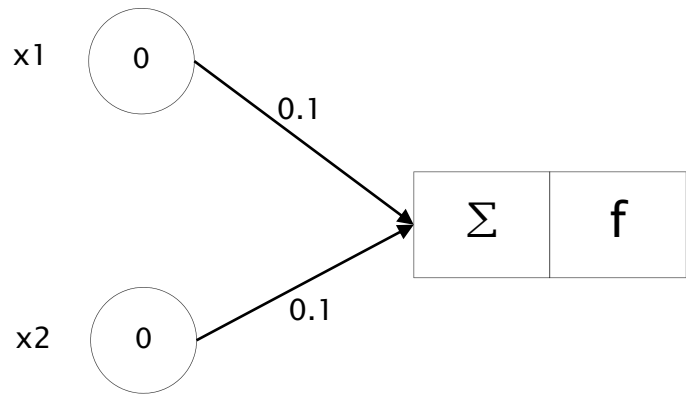
x1	x2	Classe
0	0	0
0	1	0
1	0	0
1	1	1



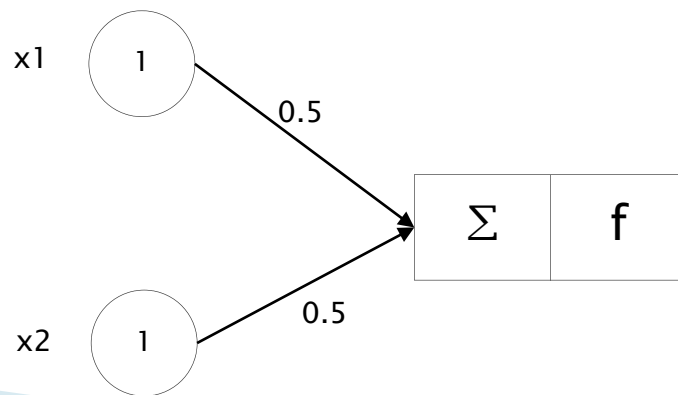
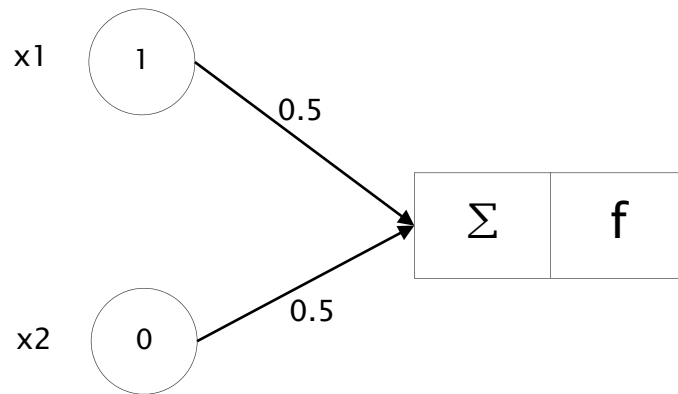
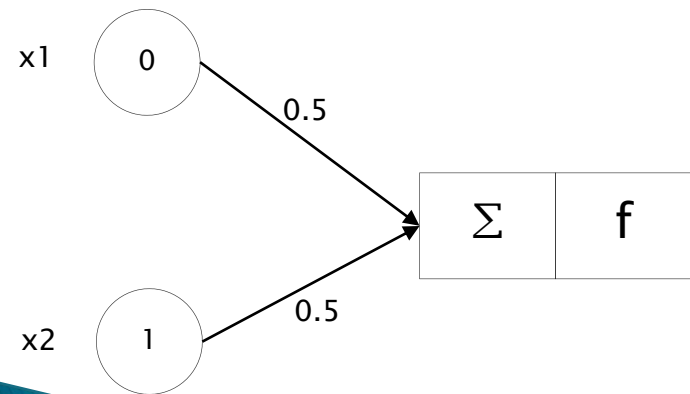
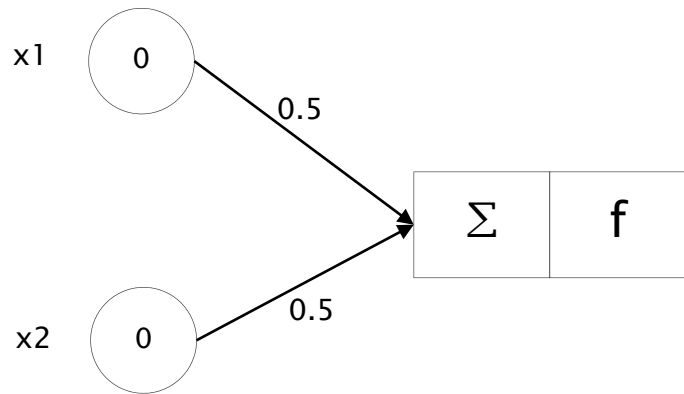
x1	x2	Classe
0	0	0
0	1	0
1	0	0
1	1	1

Erro

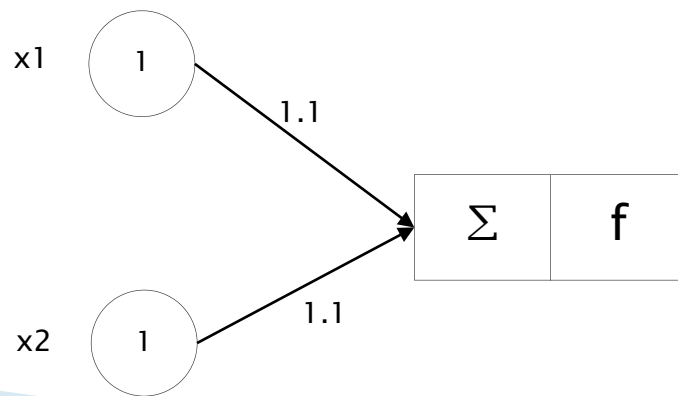
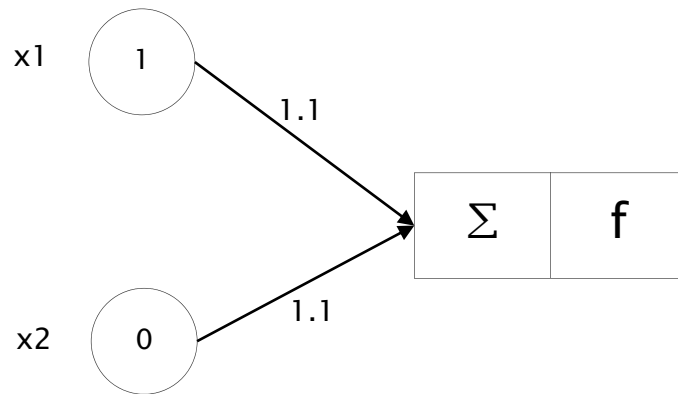
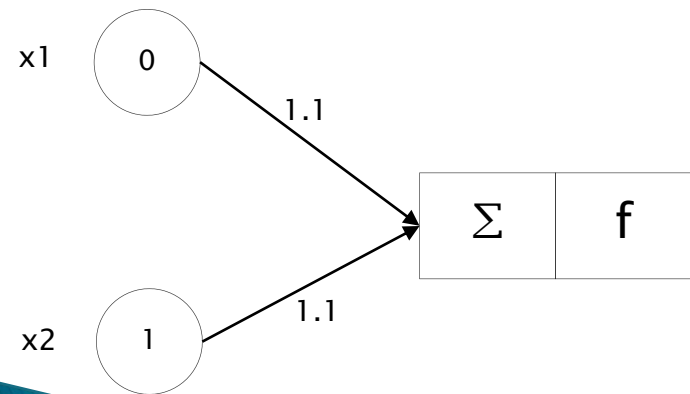
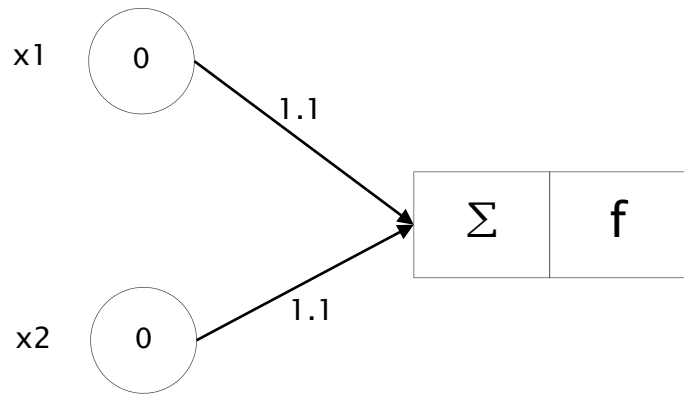
- ▶ Algoritmo mais simples
 - $\text{erro} = \text{respostaCorreta} - \text{respostaCalculada}$
- ▶ Os pesos são atualizados até os erros serem pequenos
 - $\text{peso}(n + 1) = \text{peso}(n) + (\text{taxaAprendizagem} * \text{entrada} * \text{erro})$



x1	x2	Classe
0	0	0
0	1	0
1	0	0
1	1	1



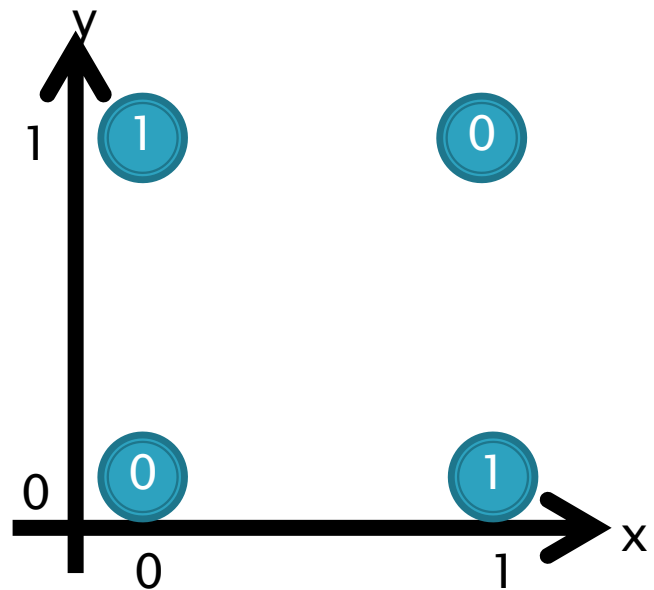
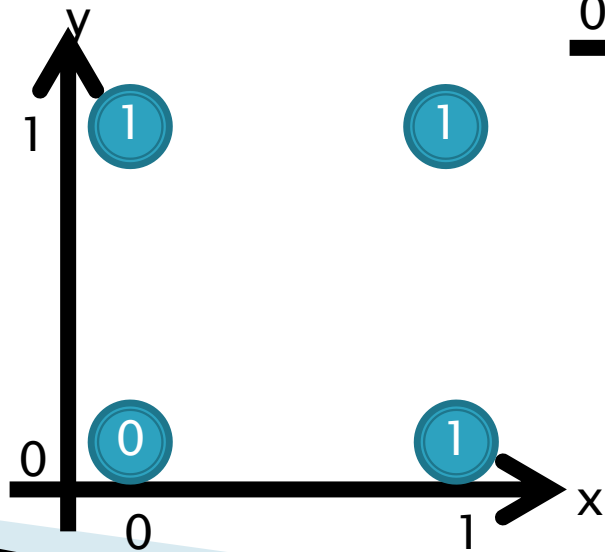
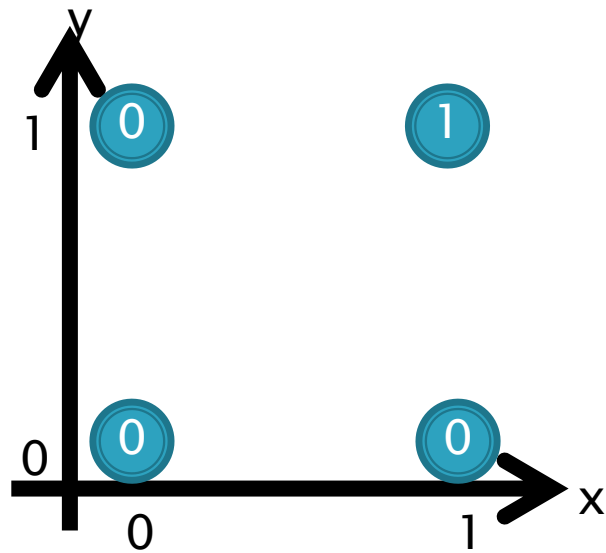
x1	x2	Classe
0	0	0
0	1	0
1	0	0
1	1	1



x1	x2	Classe
0	0	0
0	1	1
1	0	1
1	1	1

Algoritmo

- ▶ Enquanto o erro for diferente de zero
 - Para cada registro
 - Calcula a saída com os pesos atuais
 - Compara a saída esperada com a saída calculada, somando o erro
 - Para cada peso da rede
 - Atualiza o peso – $\text{peso}(n + 1) = \text{peso}(n) + (\text{taxaAprendizagem} * \text{entrada} * \text{erro})$



Conclusão