MÉTODOS E MODELOS AVANÇADOS EM CIÊNCIA DE DADOS

Aula 02B - Perceptron Multicamadas
(Multilayer Perceptron - MLP)

Prof. Rafael G. **Mantovani**

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Universidade Tecnológica Federal do Paraná (UTFPR)
Especialização em Ciência de Dados

Roteiro

Roteiro

1 Introdução

2 Multilayer Perceptron

3 Exemplo

4 Formalização / Treinamento

5 Função de Ativação / Backpropagation

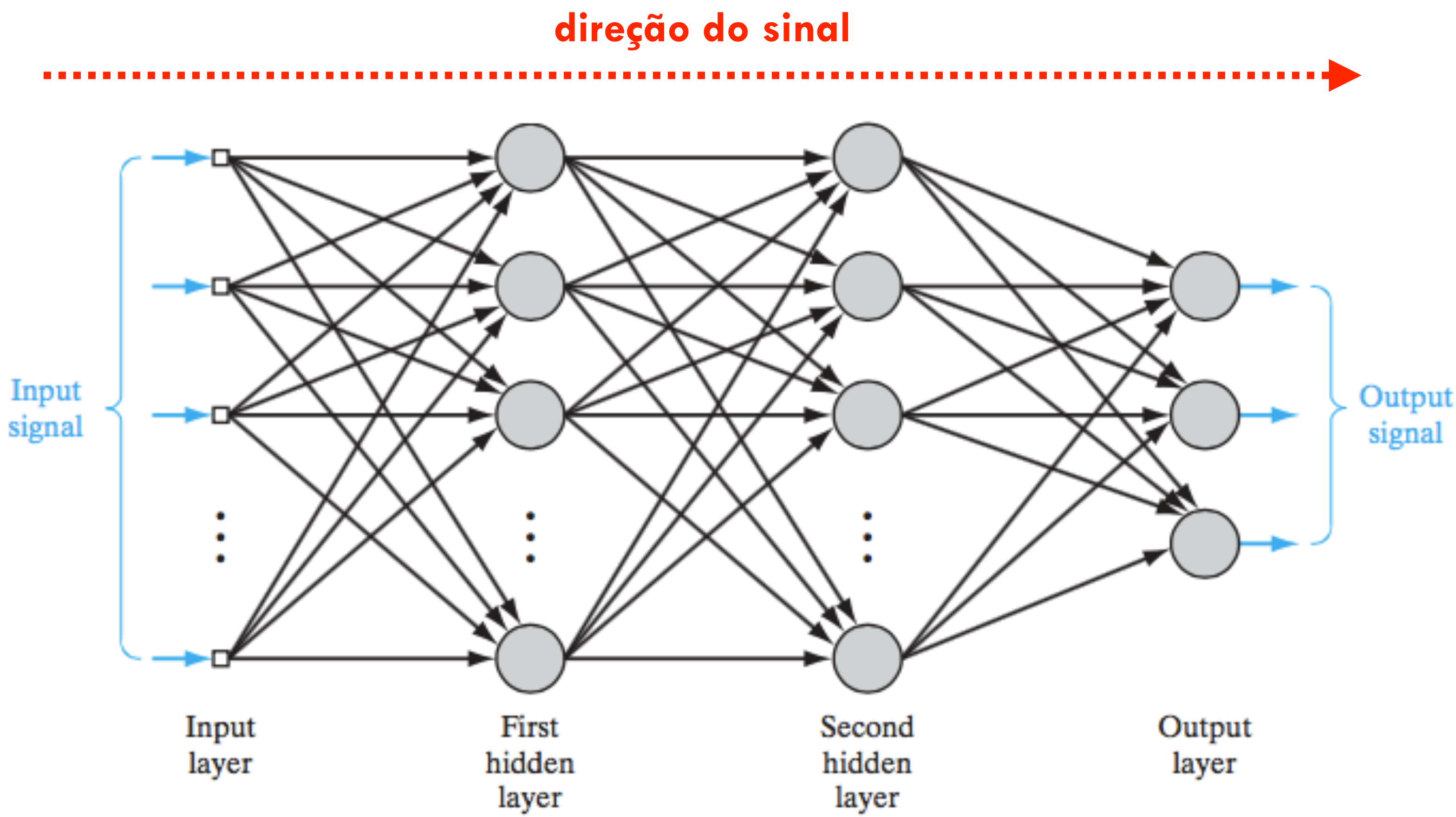
6 Síntese / Próximas Aulas

7 Referências

Introdução

- **Multilayer Perceptron:**
  - Supera as limitações práticas do Perceptron
    - neurônios possuem uma função de ativação **não-linear e diferenciável**
    - contém uma ou mais camadas escondidas
    - a rede possui alto grau de conectividade

Introdução


direção do sinal
Input signal
Output signal
Input layer
First hidden layer
Second hidden layer
Output layer

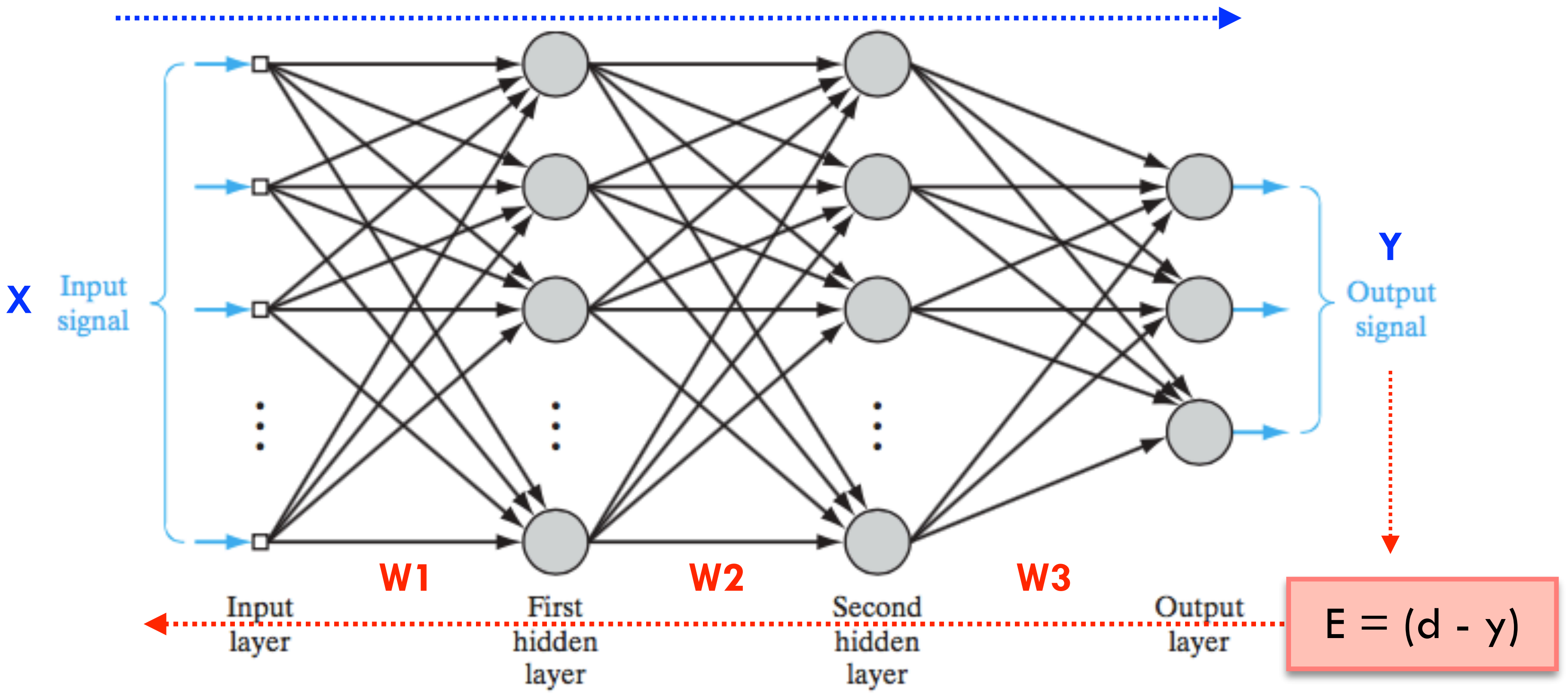
Introdução

- **Deficiências:**
  - análise teórica é difícil → há muitas conexões e funções não-lineares
  - muitos neurônios → difícil de visualizar o processo de aprendizado
  - Aprendizado → difícil: há um espaço muito maior de funções. Há mais representações dos padrões de entrada

Roteiro

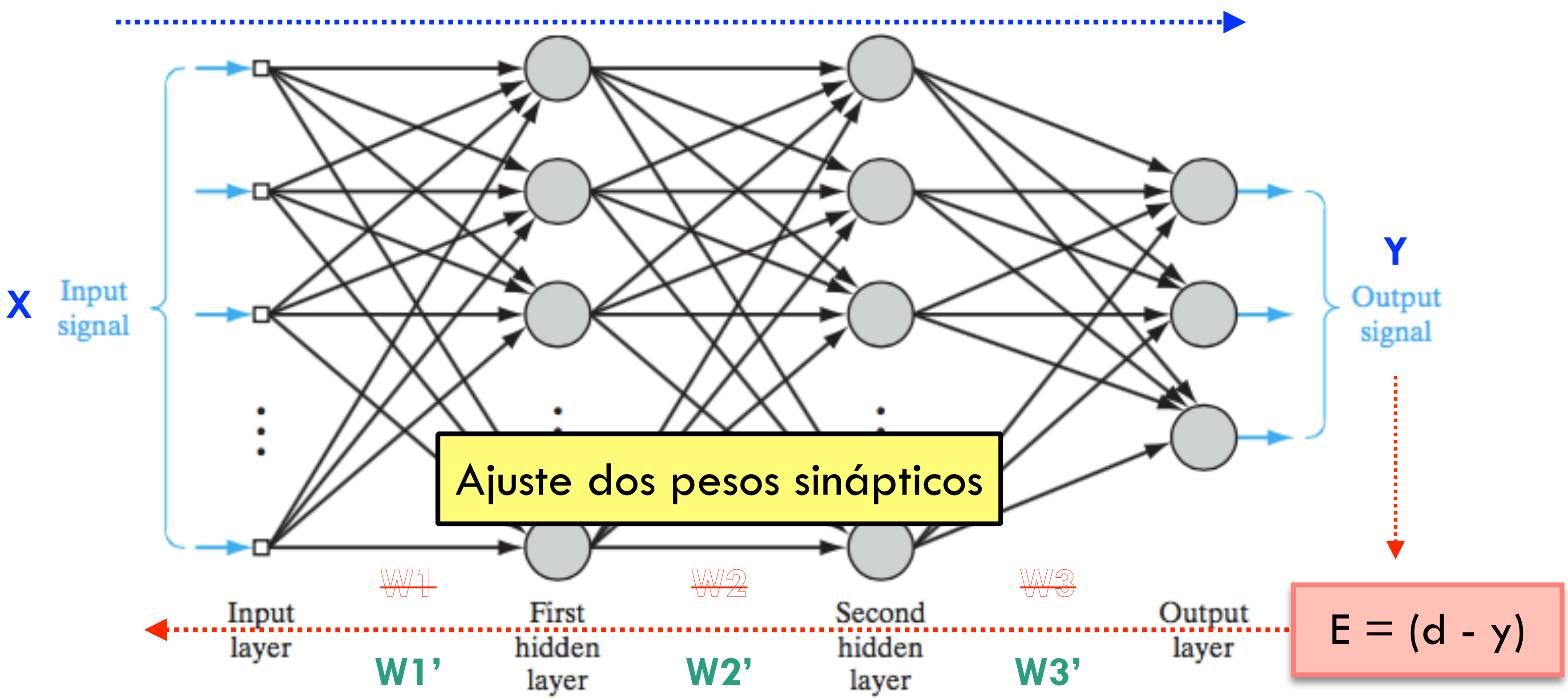
Multilayer Perceptron

- ***Backpropagation***



Multilayer Perceptron

- ***Backpropagation***



Multilayer Perceptron

- **Fases:**
  - ***Forward***: propagação do sinal

    Entrada → Camadas Ocultas → Saída

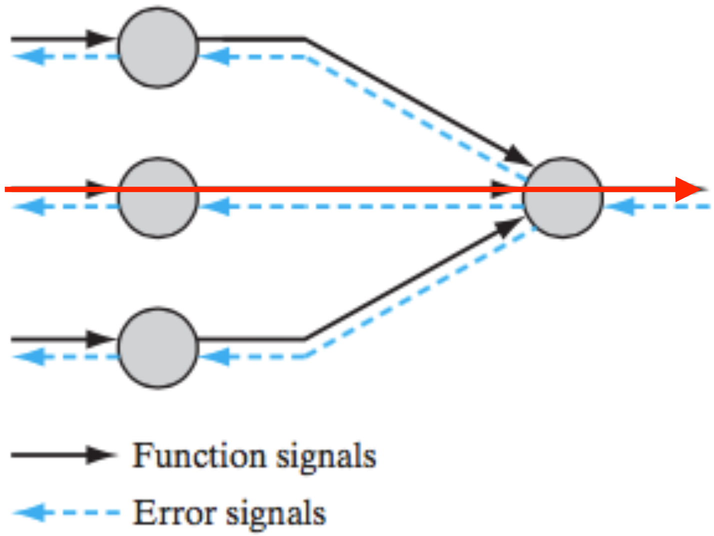
  - ***Backward***: sinal de erro é retropropagado

    Entrada ← Camadas Ocultas ← Saída

    Ajustes sinápticos

Multilayer Perceptron

- Neurônios da camada oculta, executam dois cálculos:

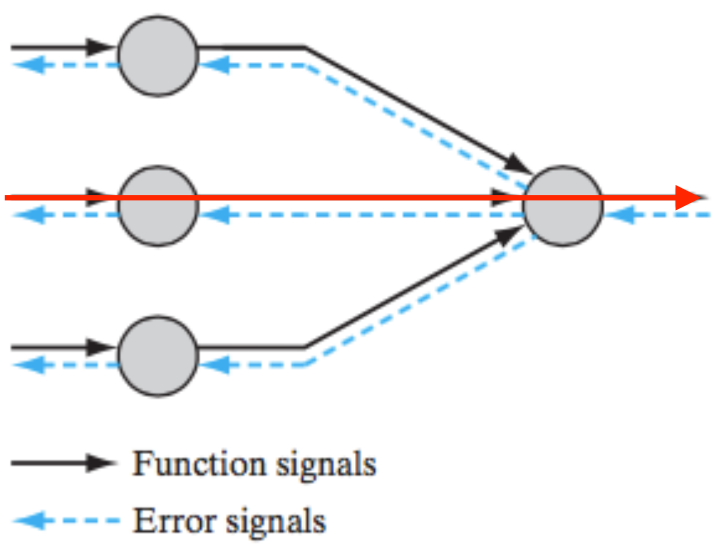


1

- sinal de função (y_{ij})
    - ϕ não-linear
    - ϕ contínua

Multilayer Perceptron

- Neurônios da camada oculta, executam dois cálculos:

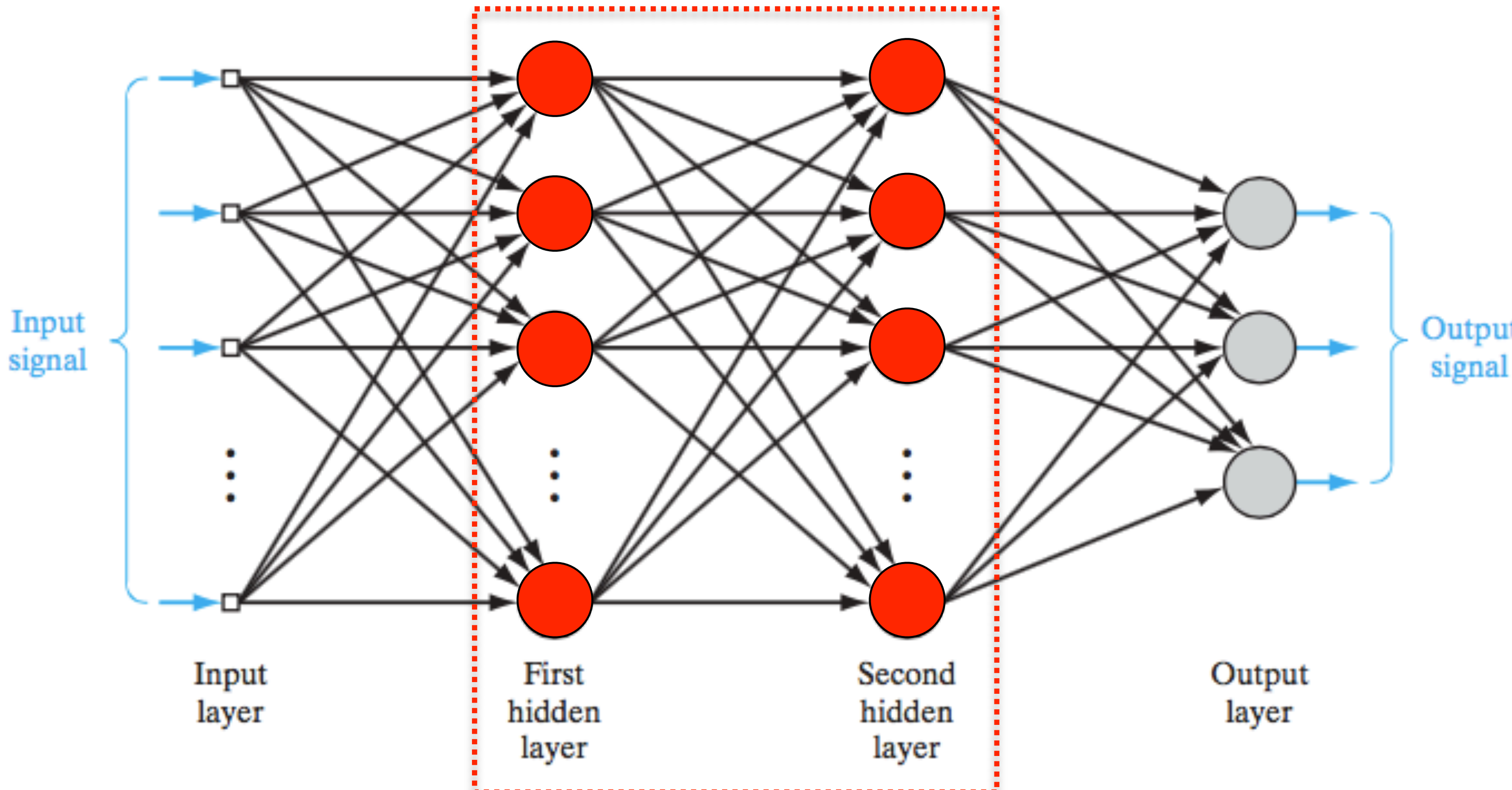


2

- estimativa do vetor gradiente (δ_{ij})
  - gradiente da superfície de erro
  - **retropropagação**

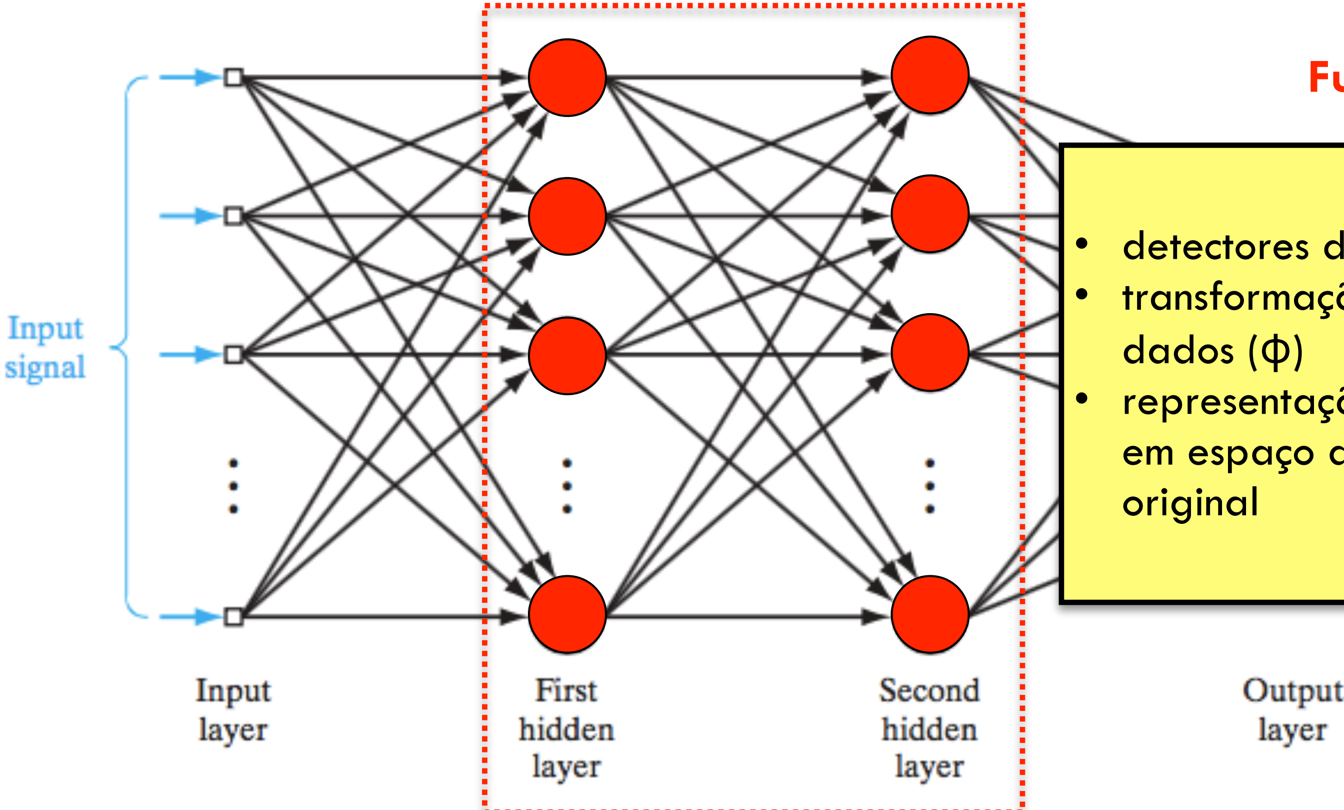
Multilayer Perceptron

- Neurônios das camadas ocultas



Multilayer Perceptron

- Neurônios das camadas ocultas

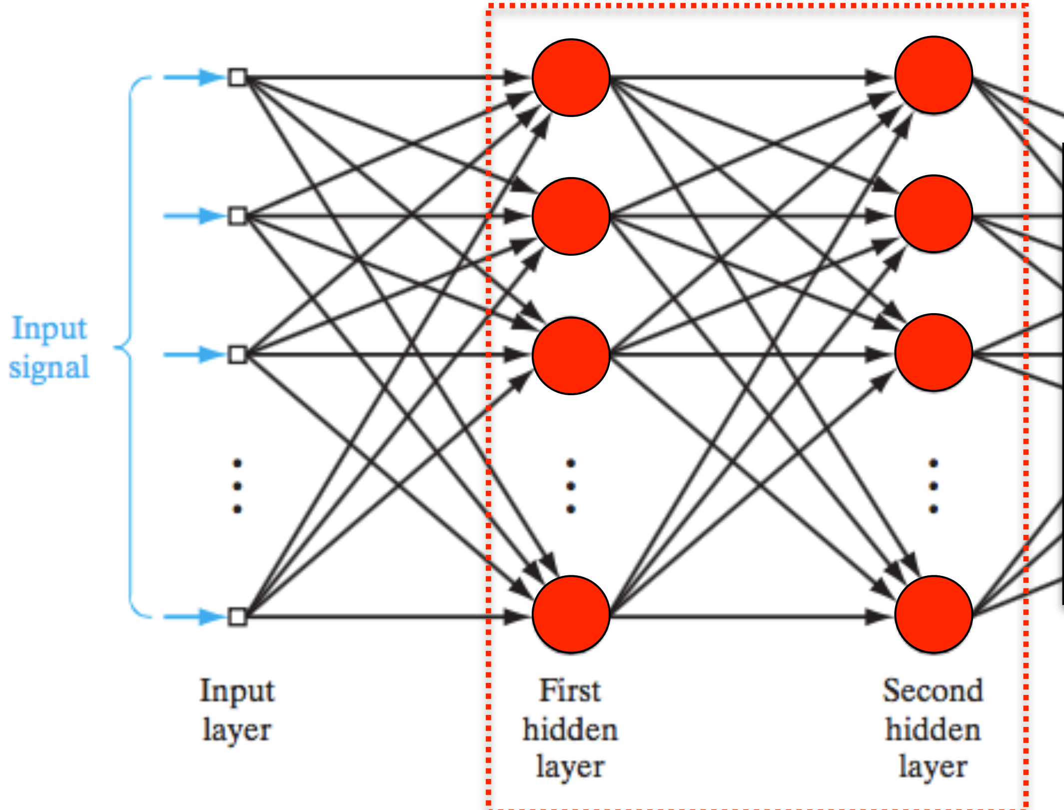


Função

- detectores de atributos
- transformação não-linear dos dados (ϕ)
- representação do problema em espaço diferente do original

Multilayer Perceptron

- Neurônios das camadas ocultas



Função

- detectores de atributos
- transformação não-linear dos dados (ϕ)
- representação do problema em espaço diferente do original

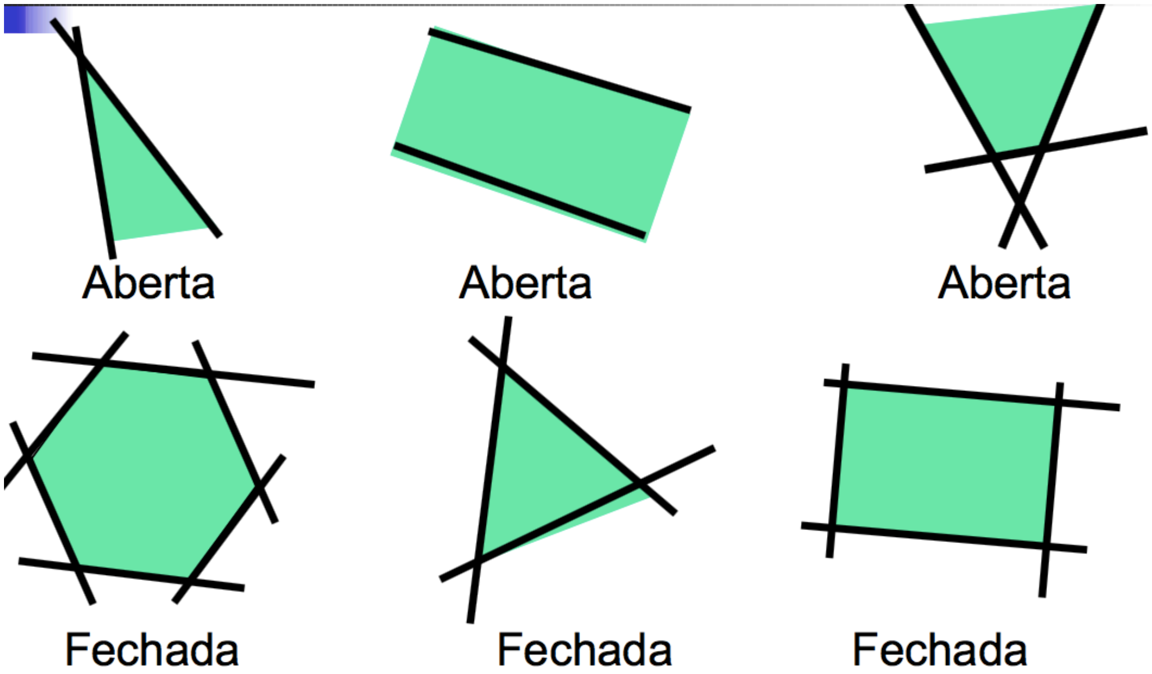
Multilayer Perceptron

- **Camadas intermediárias formam?**

1 camada → **linhas** retas no espaço de decisão
2 camada → combina as linhas da camada anterior em **regiões convexas**
3 camada → combina as regiões convexas produzindo **formatos abstratos**
...

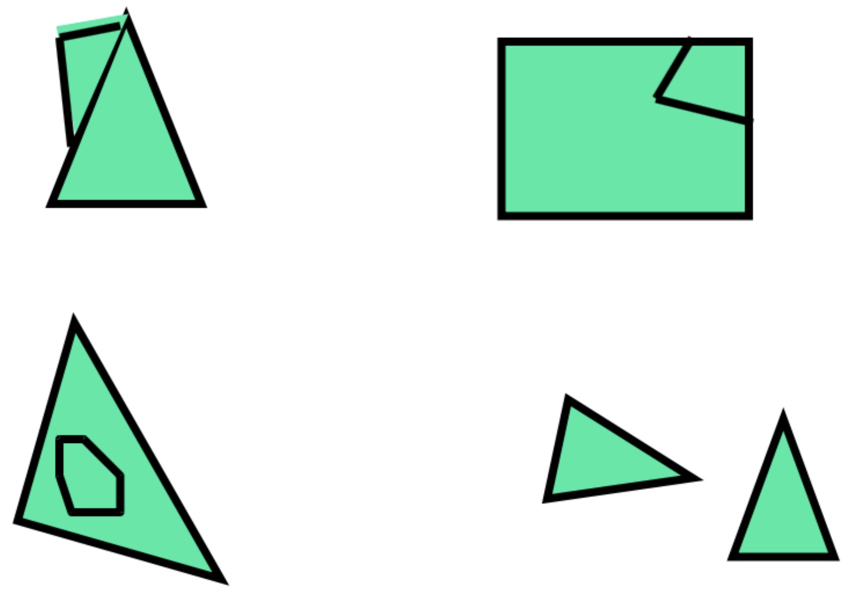
Regiões convexas

- **Combinações de hiperplanos**



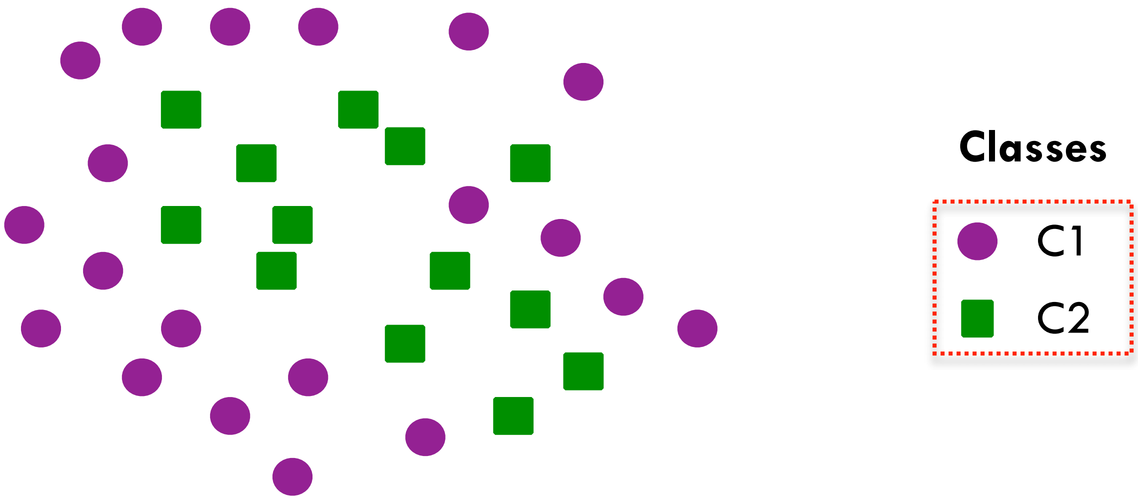
Figuras convexas

- **Combinações de regiões convexas**

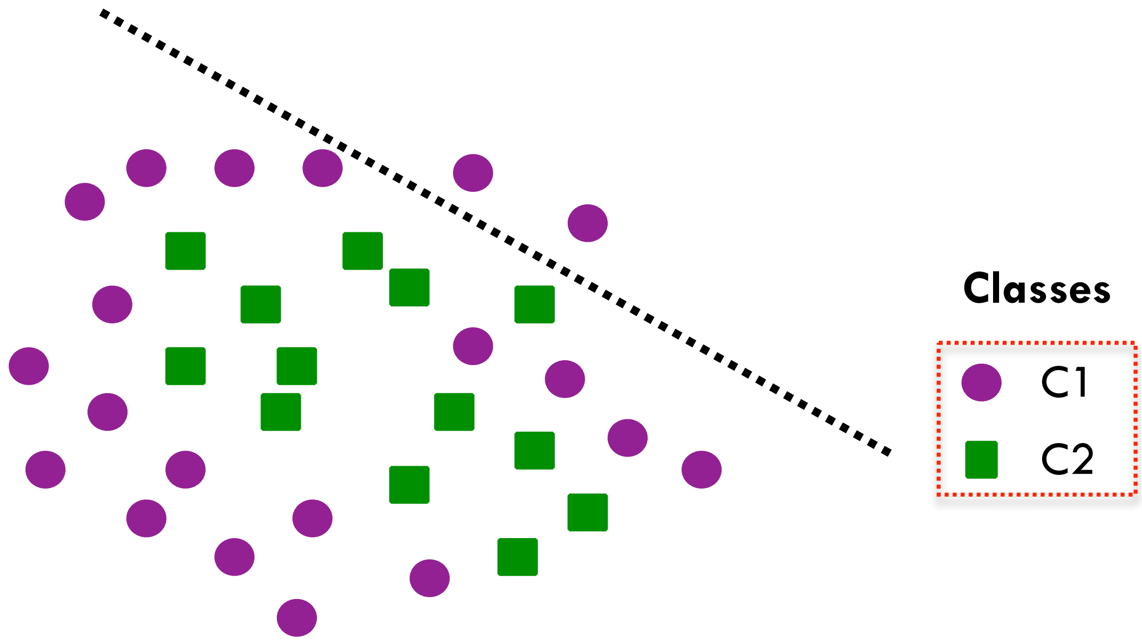


Roteiro

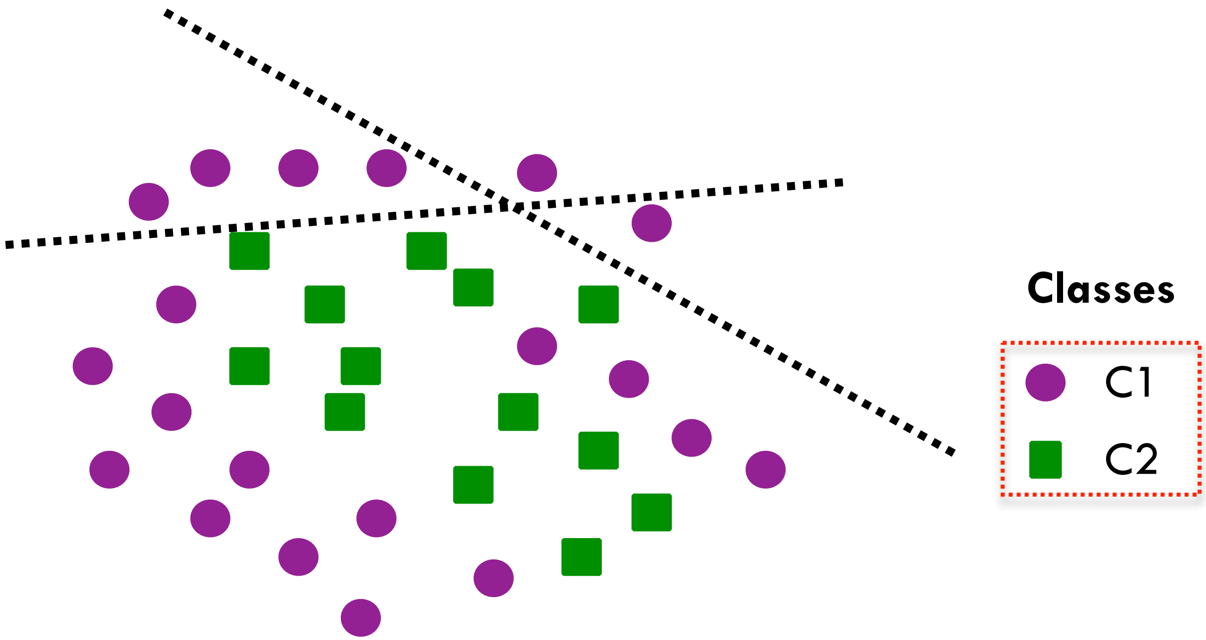
Exemplo



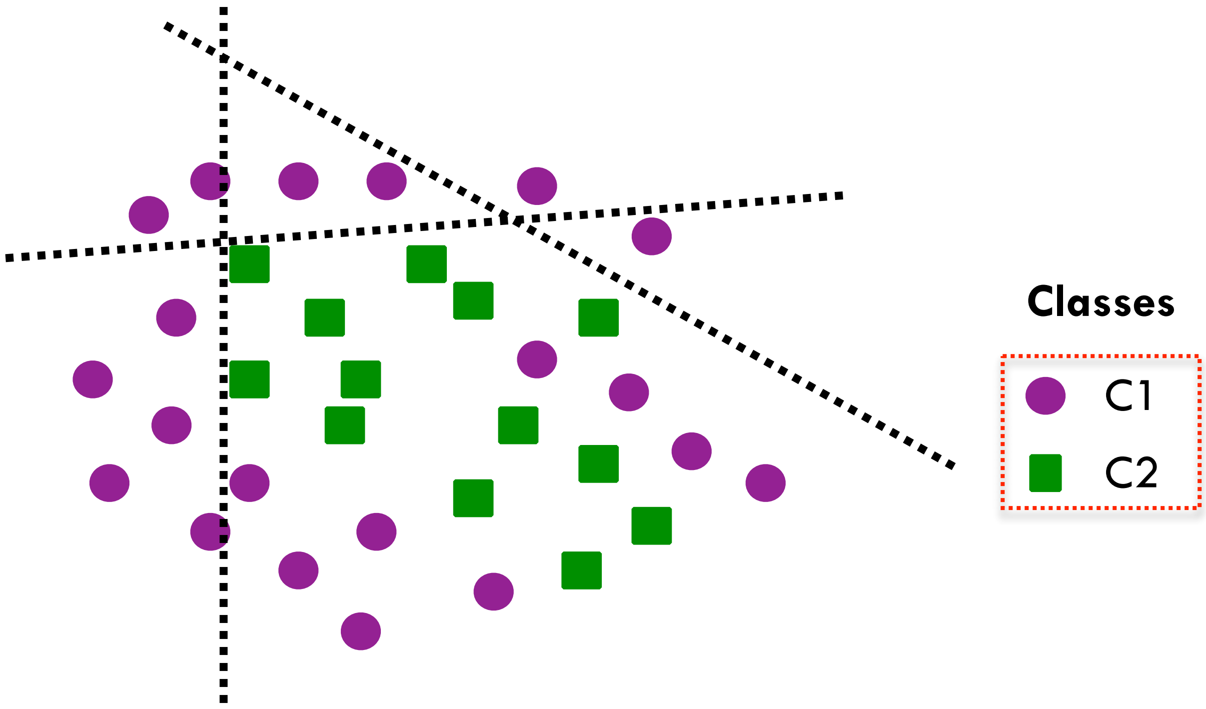
Exemplo



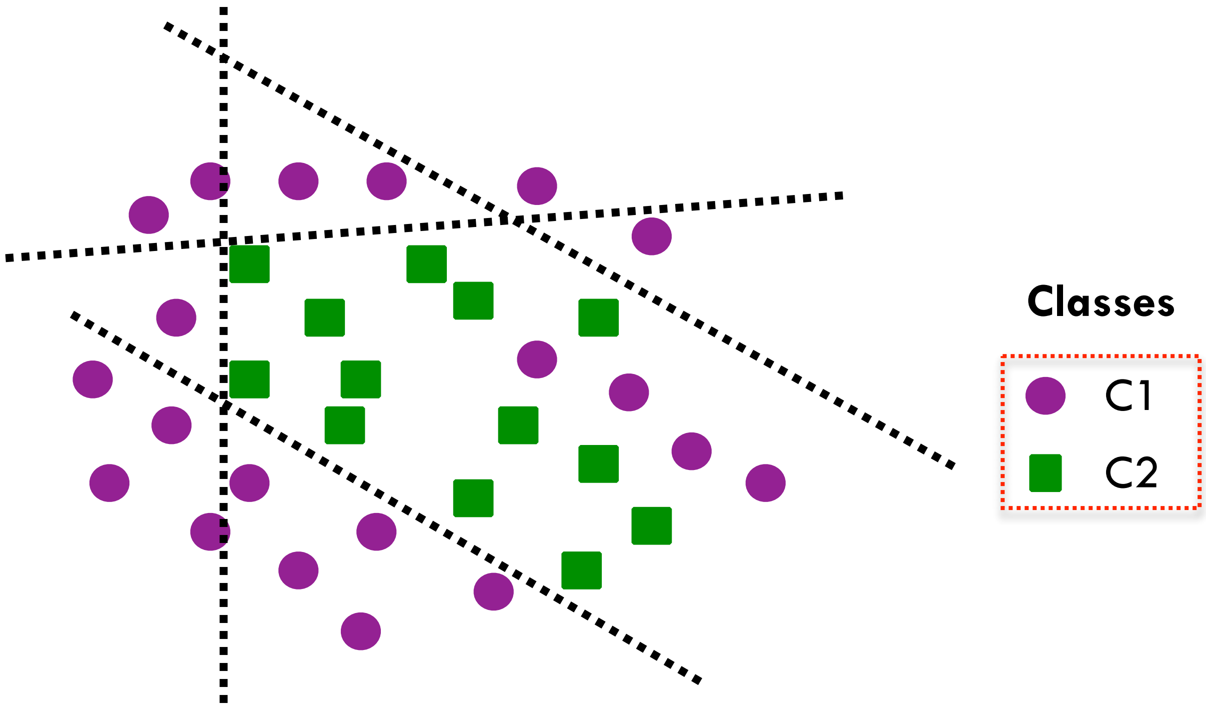
Exemplo



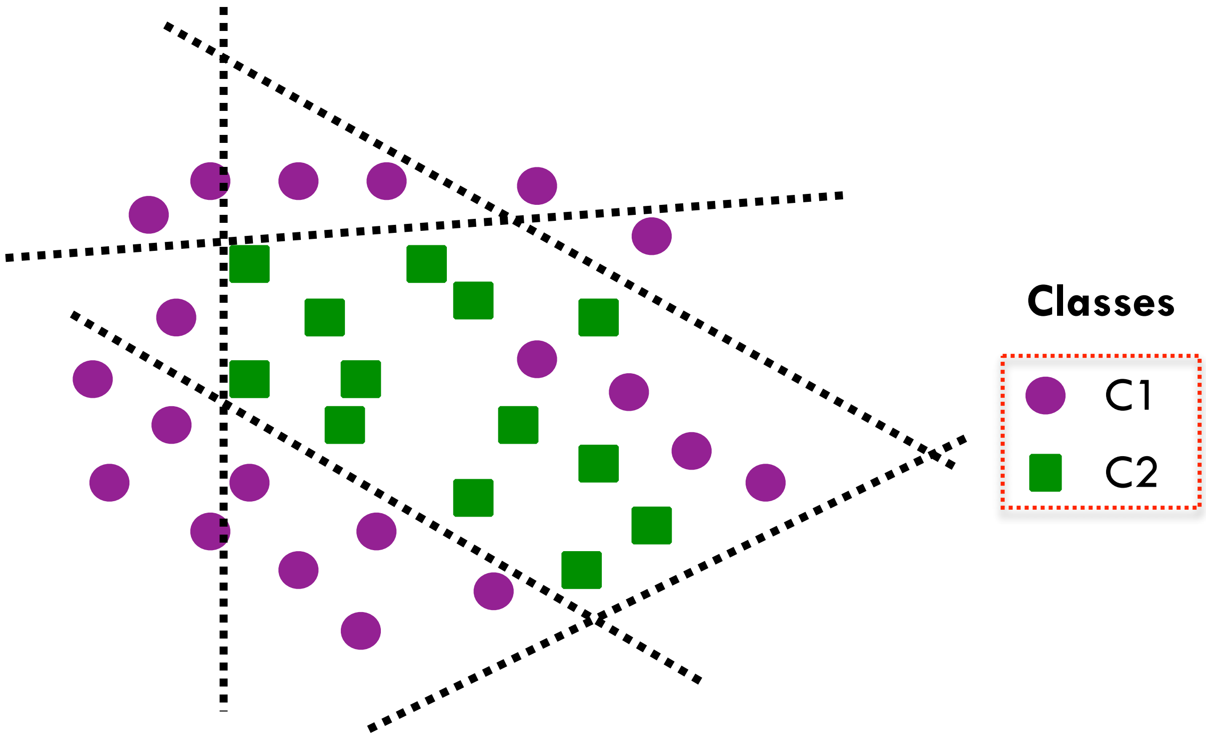
Exemplo



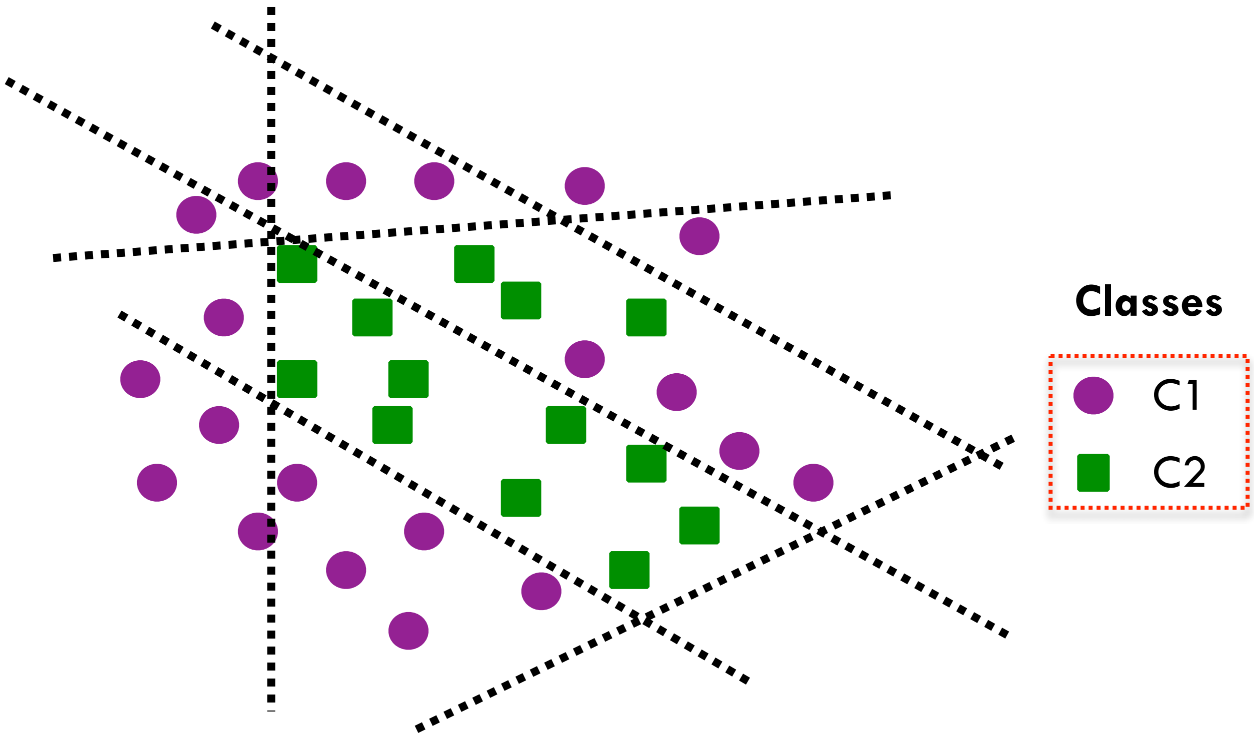
Exemplo



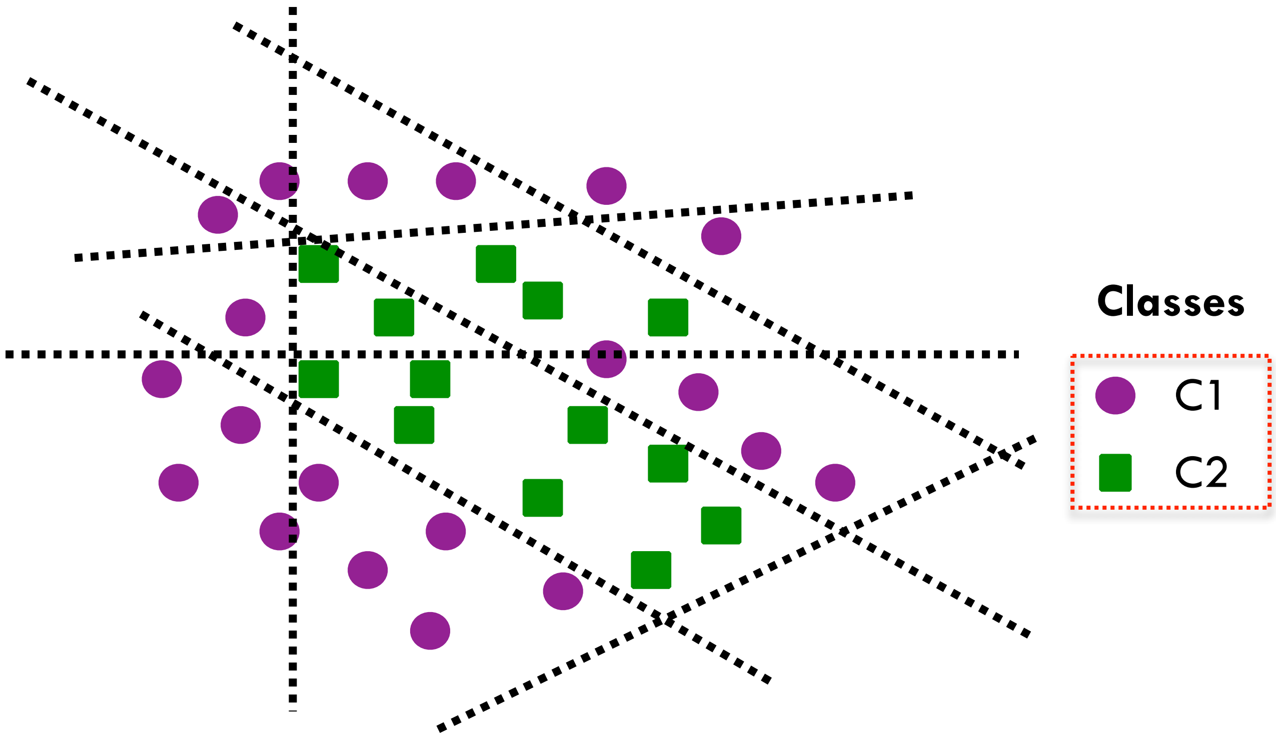
Exemplo



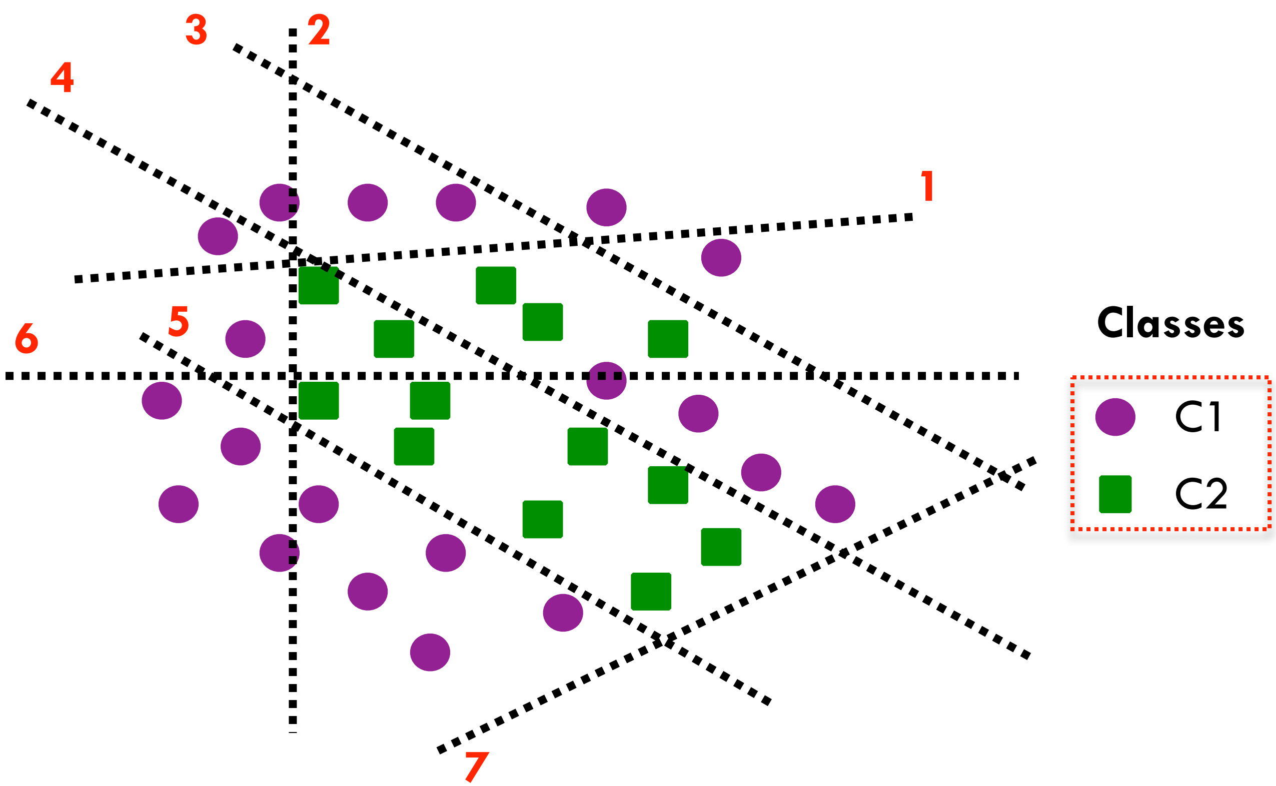
Exemplo



Exemplo



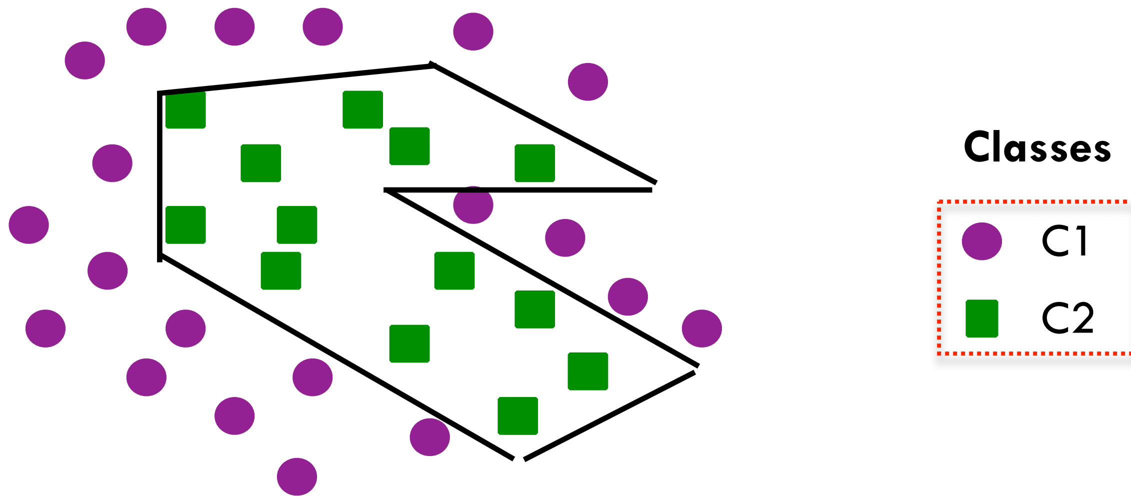
Exemplo



7 hiperplanos = 7 neurônios

Exemplo

No nosso exemplo hipotético, precisaríamos de uma região convexa com 7 hiperplanos para separar as classes corretamente.



7 hiperplanos = 1 região convexa

Roteiro

1. Introdução
2. Multilayer Perceptron
3. Exemplo
4. Formalização / Treinamento
5. Função de Ativação / Backpropagation
6. Síntese / Próximas Aulas
7. Referências

Formalização

- **MLP:**
  - τ = [x(n), d(n)] → exemplo de treinamento
  - **$y_j(n)$** → sinal produzido na saída do neurônio j na camada de saída, estimulado por x(n), aplicado na camada de entrada
  - Sinal do erro produzido pelo neurônio j é:
    - $e_j(n) = d_j(n) - y_j(n)$

Formalização

- O erro instantâneo produzido no neurônio j é dado por:

$$\varepsilon_j(n) = \frac{1}{2} e_j^2(n)$$

- Somando os erros de todos os neurônios da camada de saída:

$$\varepsilon(n) = \sum_{j \in C} \varepsilon_j(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)$$

- C é o conjunto de neurônios de saída. Se houverem N exemplos de treinamento, o **erro médio** sobre todos os exemplos (risco empírico) é dado por:

$$\varepsilon_{av}(N) = \frac{1}{N} \sum_{n=1}^{N} \varepsilon(n) = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j \in C} e_j^2(n)$$

Formalização

- O erro instantâneo produzido no neurônio j é dado por:

$$\varepsilon_j(n) = \frac{1}{2} e_j^2(n)$$

- Somando os erros de todos os neurônios da camada de saída:

$$\varepsilon(n) = \sum_{j \in C} \varepsilon_j(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)$$

Erro quadrático médio da época considera todos os neurônios da camada de saída C) e todos os exemplos do conjunto de treinamento (N)

- por:

$$\varepsilon_{av}(N) = \frac{1}{N} \sum_{n=1}^{N} \varepsilon(n) = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j \in C} e_j^2(n)$$

Treinamento

- **Alternativa 1 - modo de treinamento batch:**
  - apresenta todos os exemplos → ajuste dos pesos
  - 1 ajuste p uma época completa
- Curva de aprendizado
  - ε_{av} (erro médio) x épocas
  - Em cada época os exemplos são embaralhados/reordenados
- Vários experimentos, iniciando W com valores diferentes
  - Média do desempenho

Treinamento

- **Alternativa 1 - modo de treinamento batch:**

  - estimativa mais precisa do vetor de gradientes
    - derivada da função de custo ε_{av} em relação a W
    - suscetível a ficar prezo em um mínimo local
  - Paralelização do processo de aprendizado

  - Porém é mais difícil de detectar mudanças pequenas nos dados
  - Se há exemplos redundantes, não consegue identificar (pois ajusta os pesos para todos os exemplos)

Treinamento

- **Alternativa 2 - modo de treinamento online:**
    - ajuste de W após a apresentação de cada exemplo

- A busca no espaço de pesos multidimensional torna-se estocástica
    - método estocástico

- Menos suscetível a ficar preso em mínimos locais
- Quando há redundância, tira vantagem ao ajustar os pesos
- Detecta melhor pequenas mudanças nos dados de treinamento
- Simples de implementar / Bons resultados em problemas difíceis

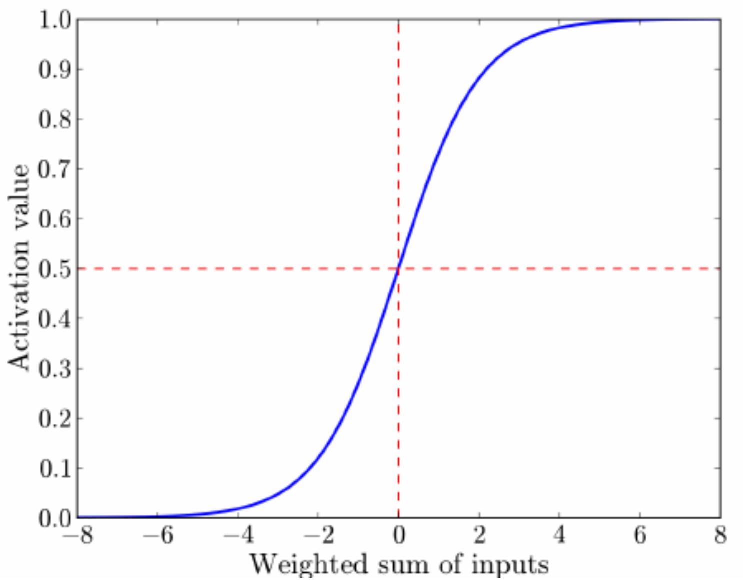
Roteiro

1. Introdução
2. Multilayer Perceptron
3. Exemplo
4. Formalização / Treinamento
5. Função de Ativação / Backpropagation
6. Síntese / Próximas Aulas
7. Referências

Funções de Ativação

- **ϕ deve ser diferenciável:**
  - funções sigmoidal / logística

$$\varphi_j(v_j(n)) = \frac{1}{1+exp(-av_j(n))}, \qquad a > 0$$

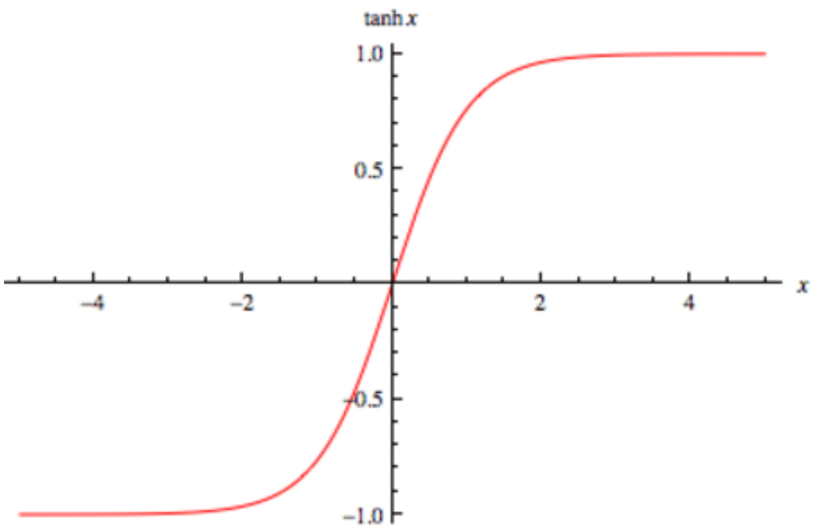


Funções de Ativação

- função tangente hiperbólica:

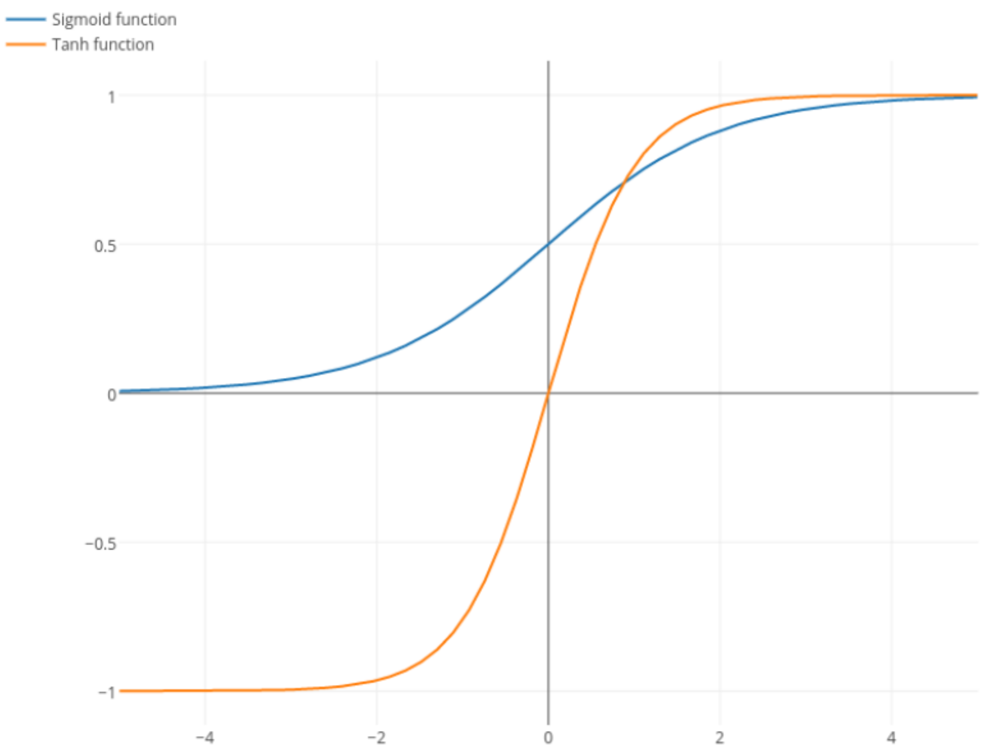
$$\varphi_j(v_j(n)) = a \tanh(b v_j(n))$$

  - a e b são constantes positivas
  - amplitude do sinal de saída: $-a \leq y_j \leq +a$

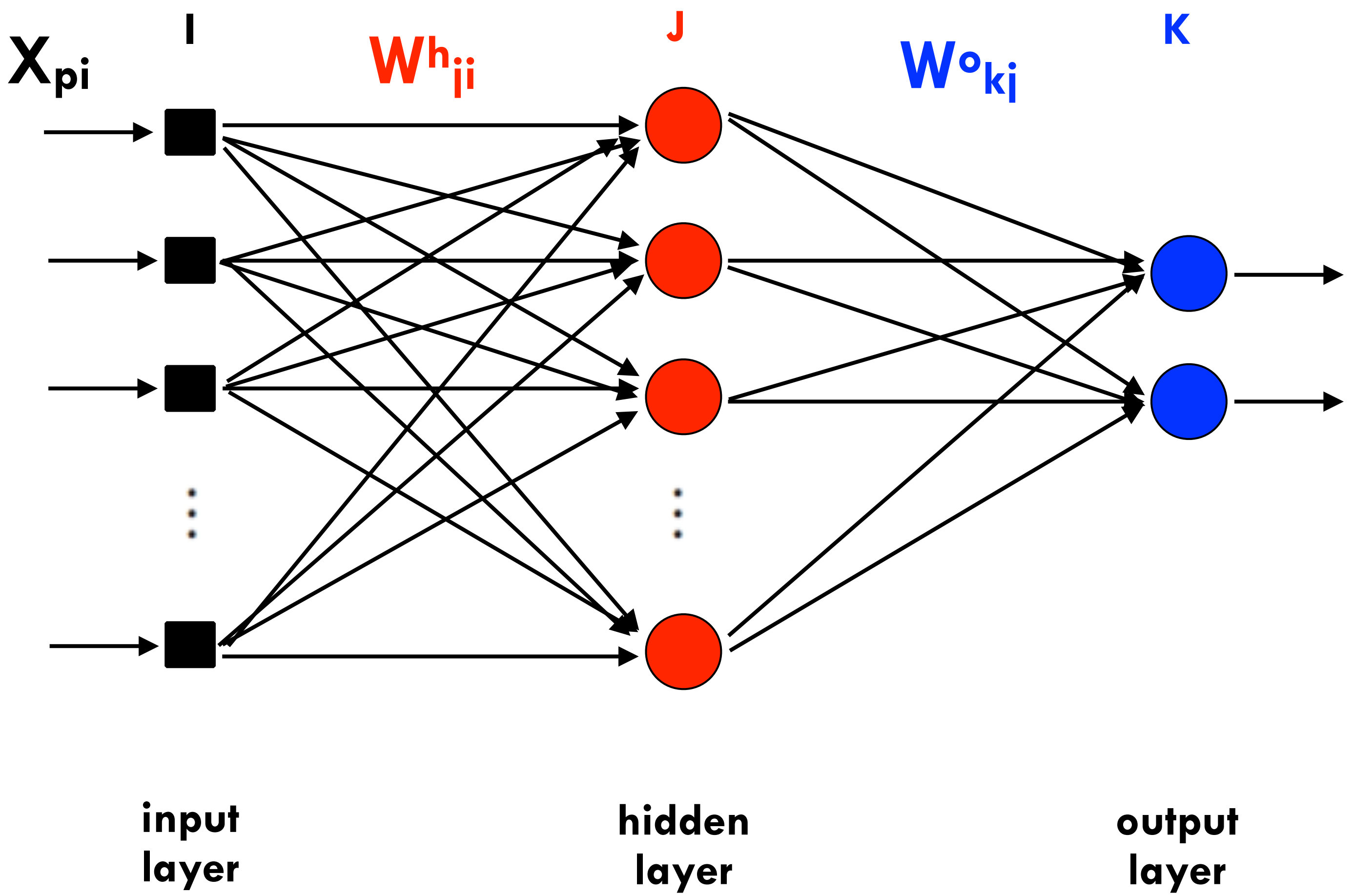


Funções de Ativação

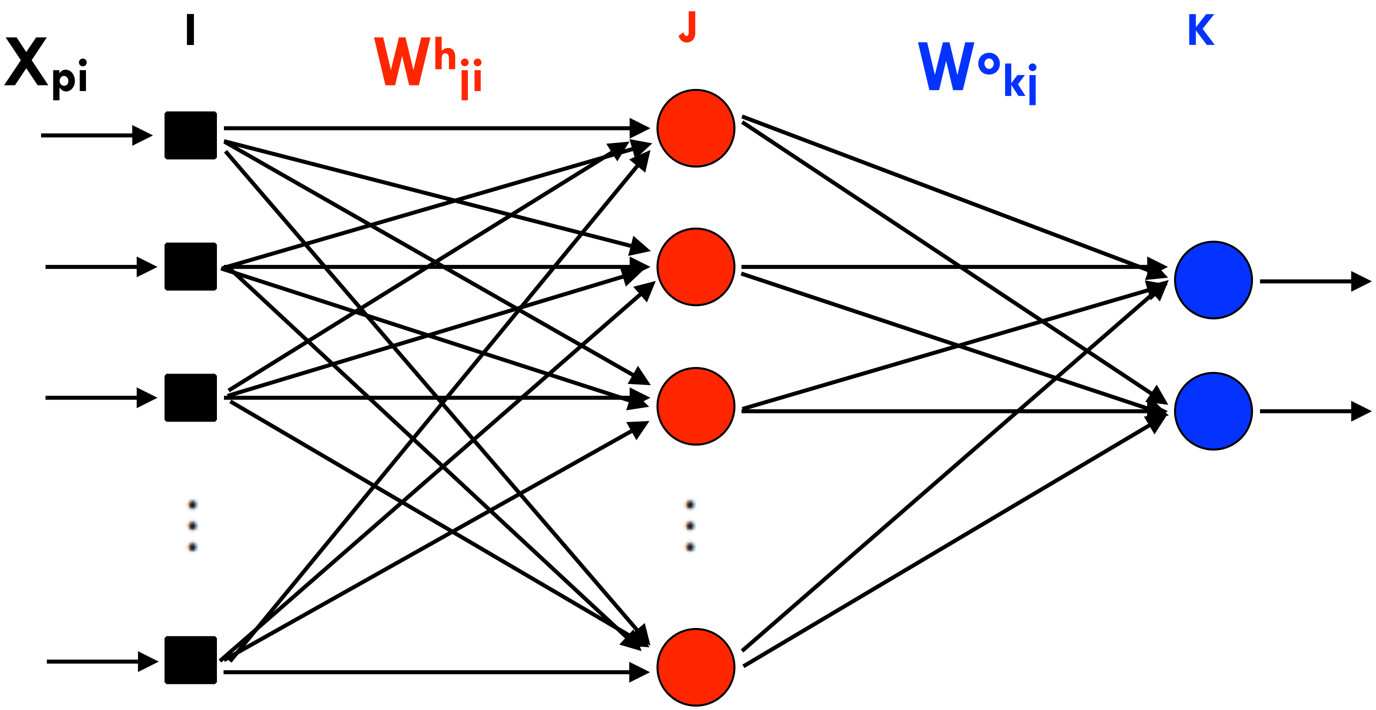
- Comparativo entre as duas formas de funções de ativação
  - tahn x sigmoid



Backpropagation

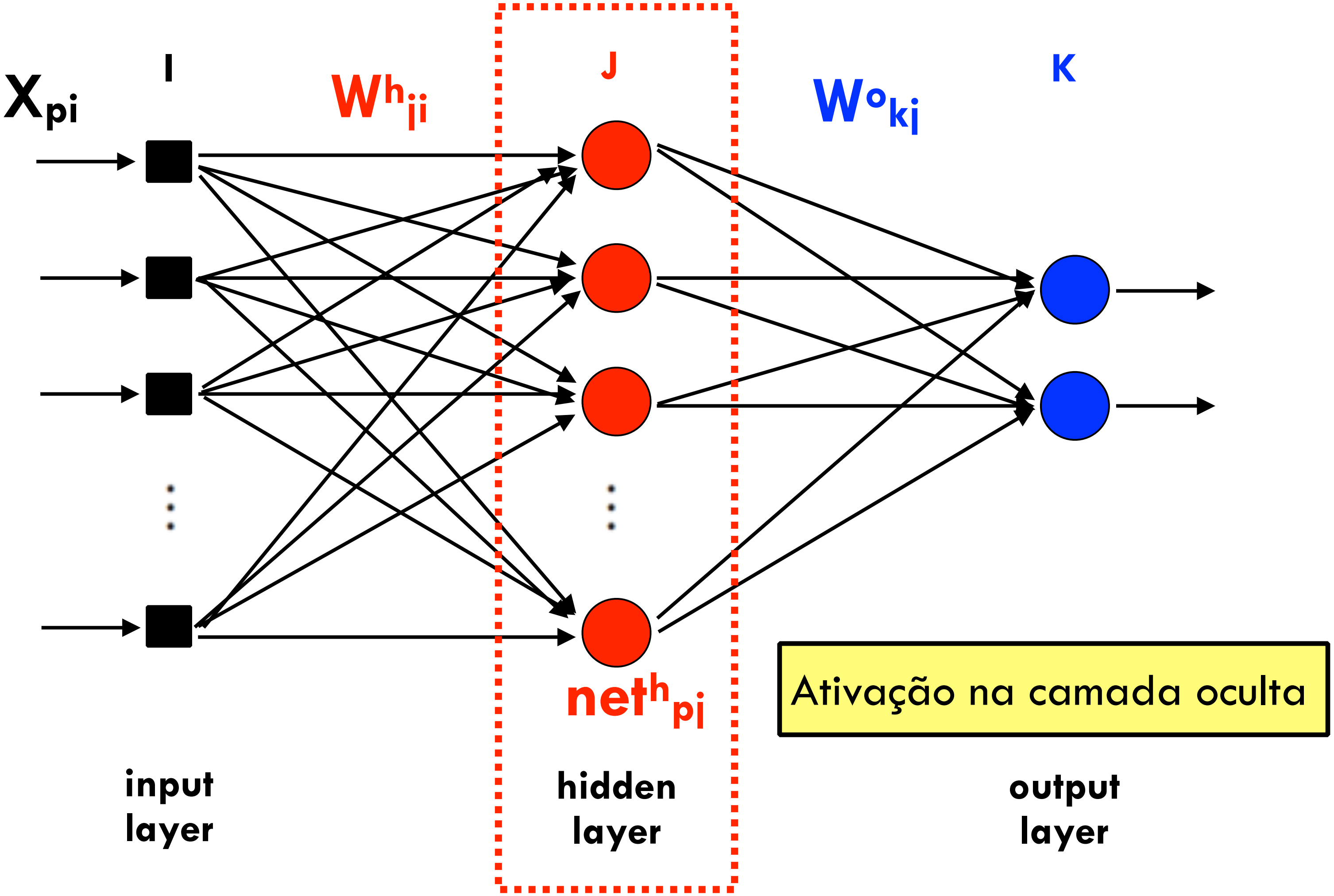


Backpropagation



Os pesos sinápticos são matrizes:
Wh - conecta camada **oculta** e a camada de **entrada**
Wo - conecta a camada de **saída** e a camada **oculta**

Backpropagation



Backpropagation

- Sinal no neurônio de índice J na camada escondida:

$$\mathbf{net}_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + \theta_{j}^{h}$$

N → número de neurônios na camada de entrada

w^h_{ji} → peso da conexão com o neurônio de entrada i

x_{pi} → padrão inserido na entrada da rede

θ^h_j → bias do neurônio

Backpropagation

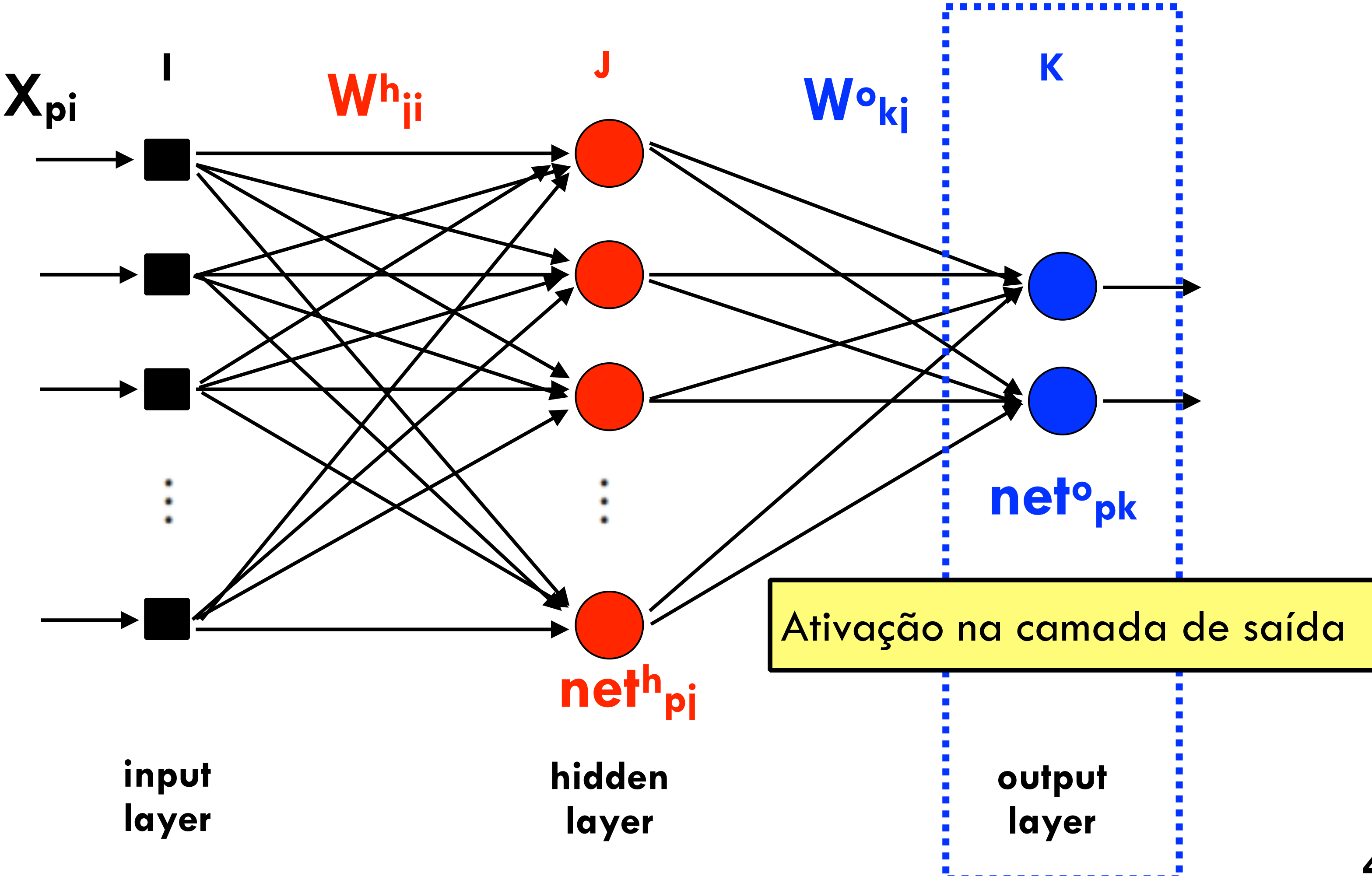
- Sinal no neurônio de índice J na camada escondida:

$$\mathbf{net}_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + \theta_j^h$$

- Ativação desse neurônio é igual a:

$$i_{pj} = f^h_j\,(net^h_{pj})$$

Backpropagation



Backpropagation

- Sinal no neurônio de índice K na camada de saída:

$$\mathbf{net}^{o}_{pk} = \sum_{j=1}^{L} w^{o}_{kj} i_{pj} + \theta^{o}_{k}$$

L → número de neurônios na camada de saída

w^{o}_{pj} → peso da conexão com o neurônio j da camada escondida

i_{pj} → valor de ativação do neurônio j da camada escondida

θ^{o}_{k} → bias do neurônio

Backpropagation

- Sinal no neurônio de índice K na camada de saída:

$$\mathbf{net}^{o}_{pk} = \sum_{j=1}^{L} w^{o}_{kj} i_{pj} + \theta^{o}_{k}$$

- Saída desse neurônio é igual a:

$$o_{pk} = f^{o}_{k}\,(net^{o}_{pk})$$

Algoritmo resumido

Inicio algoritmo

1. Aplicar um vetor de entrada para a rede (X) e calcular os valores de saída
2. Comparar as saídas atuais com as saídas desejadas e obter uma medida de erro
3. Determinar em qual direção (+ ou -) se deve modificar os pesos para minimizar o erro
4. Determinar a quantidade para se modificar cada peso
5. Aplicar as correções aos pesos
6. Repetir de 1 a 5 com todos os exemplos de treinamento, até que uma margem de erro de treinamento seja atingida

Fim algoritmo

Algoritmo resumido

1. Aplicar o valor de entrada Xp na rede

2. Calcular os valores de entrada na camada oculta (net^h_{pj})

$$\mathbf{net}^h_{pj} = \sum_{i=1}^{N} w^h_{ji} x_{pi} + \theta^h_j$$

3. Calcular as saídas da camada oculta (i_{pj})

$$i_{pj} = f^h_j (net^h_{pj})$$

Algoritmo resumido

4. Avance para a camada de saída. Calcular os valores de entrada para cada unidade de saída (net^o_{pk})

$$\mathbf{net}^o_{pk} = \sum_{j=1}^{L} w^o_{kj} i_{pj} + \theta^o_k$$

5. Calcular as saídas da camada de saída (o_{pk})

$$o_{pk} = f^{ok}(net^h_{pk})$$

Algoritmo resumido

6. Calcular os termos de erro para as unidades de saída:

$$\delta^{o}_{pk} = (y_{pk} - o_{pk}) * f^{o\prime}_{k}(net^{o}_{pk})$$

7. Calcular os termos de erro para as unidades ocultas:

$$\delta^{h}_{pj} = f^{h\prime}_{j}(net^{h}_{pj}) \sum_{k} (\delta^{o}_{pk} * W^{o}_{kj})$$

Obs: o erro das unidades ocultas é calculado **ANTES** do ajuste de pesos da camada de saída

Algoritmo resumido

8. Atualizar os pesos da camada de saída

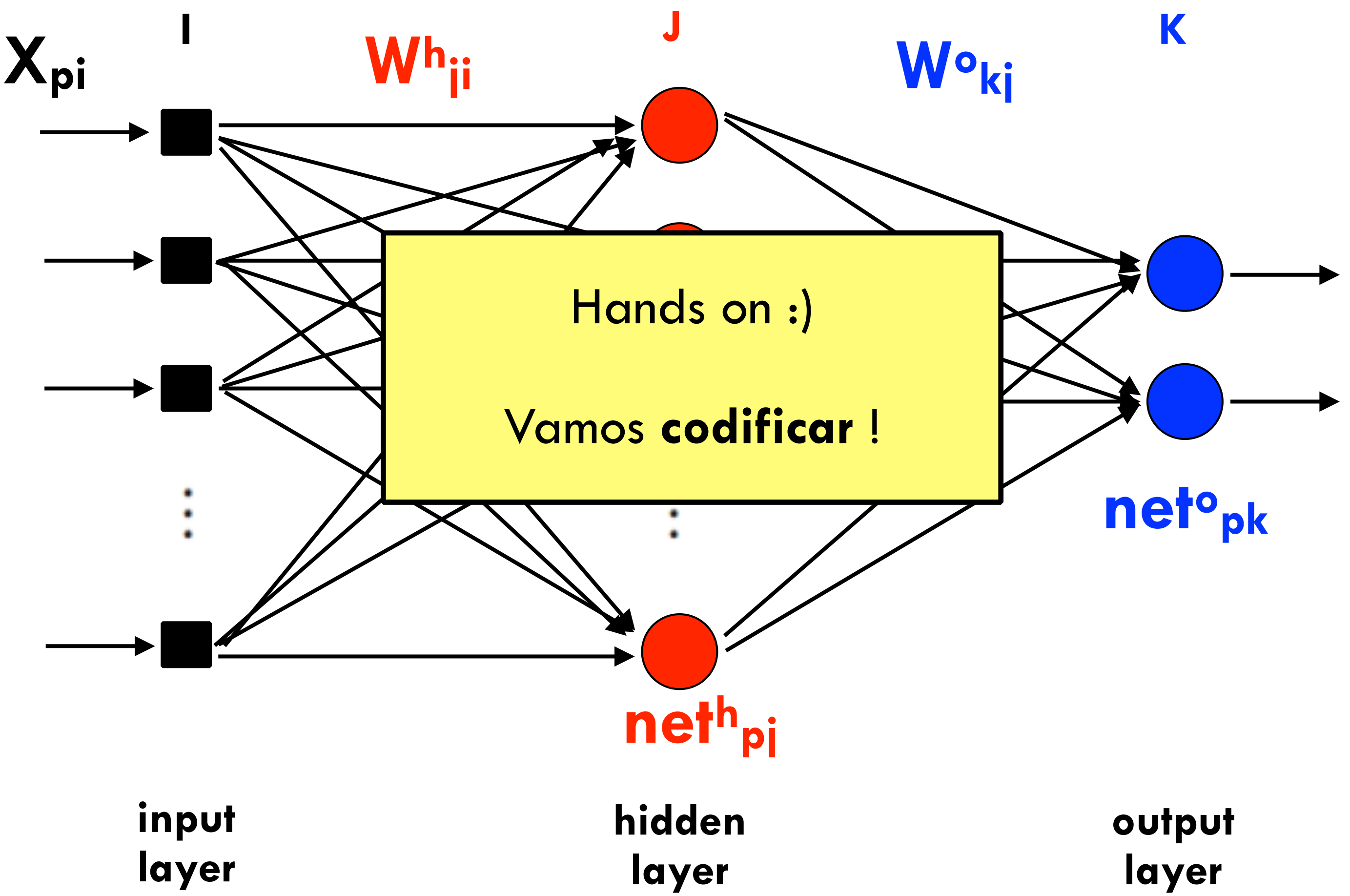
$$W^o_{kj}(t+1) = W^o_{kj}(t) + \eta * \delta^o_{pk} * i_{pj}$$

9. Atualizar os pesos da camada oculta

$$W^h_{ji}(t+1) = W^h_{ji}(t) + \eta * \delta^h_{pj} * x_{pi}$$

10. Calcular o erro total da época
    a. Indica o quão bem a rende está aprendendo.
    b. quando for menor que um limiar, parar

Exemplo(s)



X_{pi}
I
W^h_{ji}
J
W^o_{kj}
K
Hands on :)
Vamos **codificar** !
net^o_{pk}
net^h_{pj}
input layer
hidden layer
output layer

Roteiro

1. Introdução
2. Multilayer Perceptron
3. Exemplo
4. Formalização / Treinamento
5. Função de Ativação / Backpropagation
6. Síntese / Próximas Aulas
7. Referências

Síntese/Revisão

- MLP
  - perceptron multicamadas
  - neurônio j - sinais de ativação, sinais de erro
  - função de ativação **diferenciável**

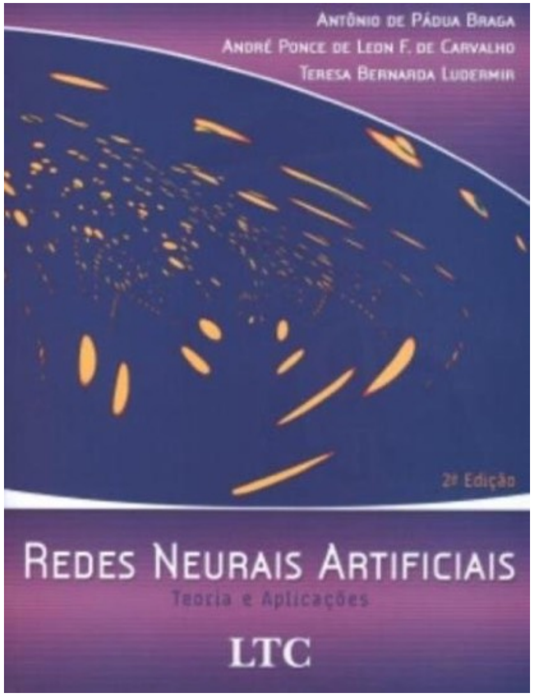
  - combinação de hiperplanos / regiões convexas
  - Backpropagation
    - forward → propaga sinal
    - backward → propaga o erro

Roteiro

1. Introdução
2. Multilayer Perceptron
3. Exemplo
4. Formalização / Treinamento
5. Função de Ativação / Backpropagation
6. Síntese / Próximas Aulas
7. **Referências**

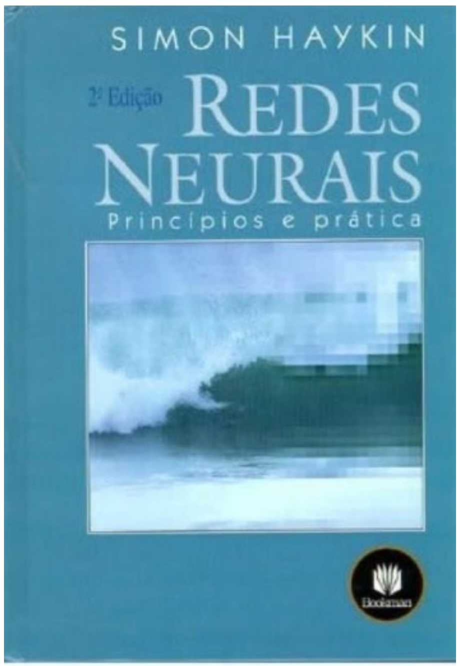
Literatura Sugerida

[Faceli et al, 2011]

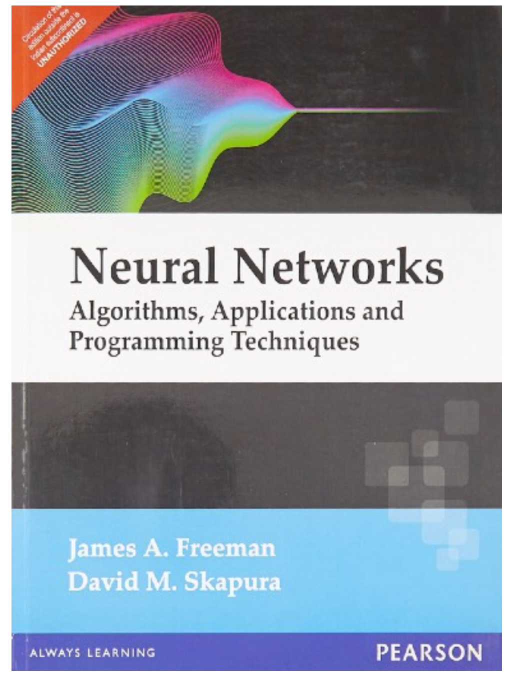


[Braga et al, 2007]

Literatura Sugerida



(Haykin, 1999)



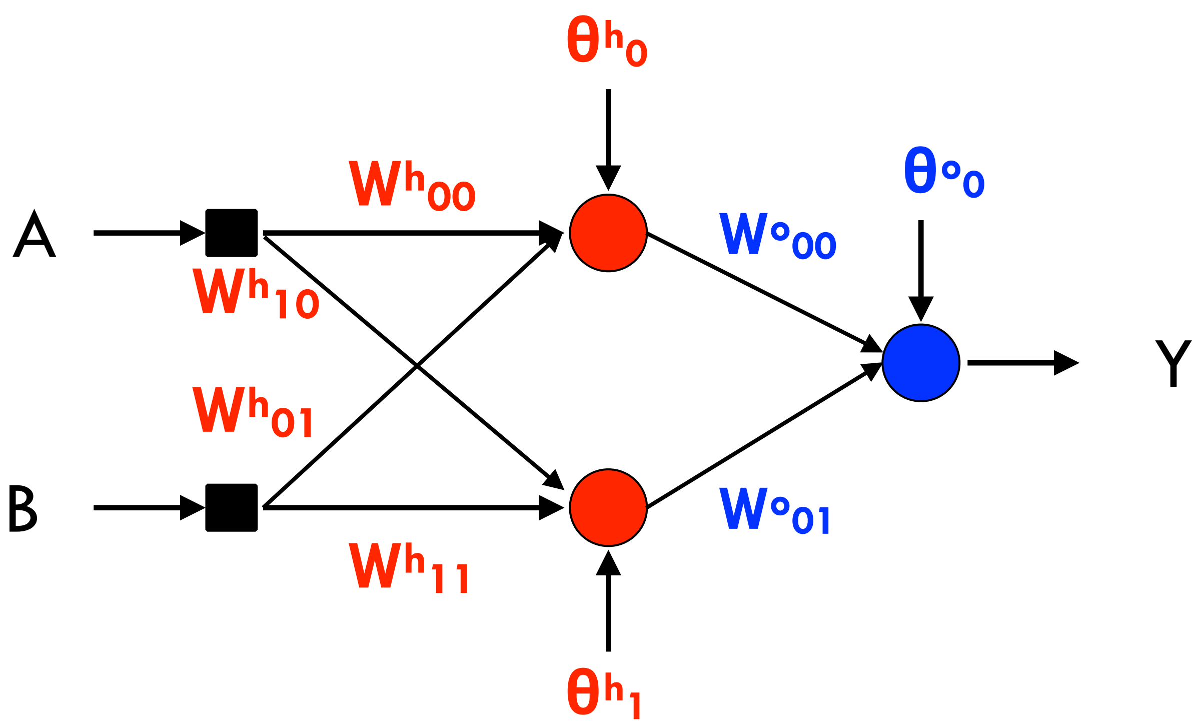
(Freeman & Skapura, 1991)

Perguntas?

Prof. Rafael G. Mantovani

rgmantovani@gmail.com

Exemplo



XOR dataset

A	*B*	*Y*
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*

Exemplo

epoch	$\theta^h 0$	$\theta^h 1$	$\theta^o 0$	$W^h 00$	$W^h 10$	$W^h 01$	$W^h 11$	$W^o 00$	$W^o 01$
0	0.05	0.06	0.07	0.2	0.15	0.35	0.18	0.10	0.12
1									
2									

- $\eta = 0.2$
- $f(net) = 1/(1+ \exp^{-net})$
- $f'(net) = net\ (1 - net)$

Exercício

epoch	θ^h0	θ^h1	θ^o0	W^h00	W^h10	W^h01	W^h11	W^o00	W^o01
0	0.05	0.06	0.07	0.2	0.15	0.35	0.18	0.10	0.12
1									
2									

- X = XOR dataset
- $\eta = 0.2$
- **$f(net) = net^3 + 0.5$**
- **$f'(net) = 3*net^2$**