

## 2. Redes Neurais Artificiais

**Prof. Renato Tinós**

Depto. de Computação e Matemática (FFCLRP/USP)

# 2.3. Perceptron Multicamadas - MLP

---

## 2.3.1. Introdução ao MLP

## 2.3.2. Treinamento

### 2.3.2.1. Backpropagation

### 2.3.2.2. Dificuldades do treinamento

## 2.3.3. Projeto de MLPs

## 2.3.4. Reconhecimento de Padrões

## 2.3.5. Aproximação de Funções

## 2.3.6. Generalização

## 2.3.1. Introdução ao MLP

- **Problema**

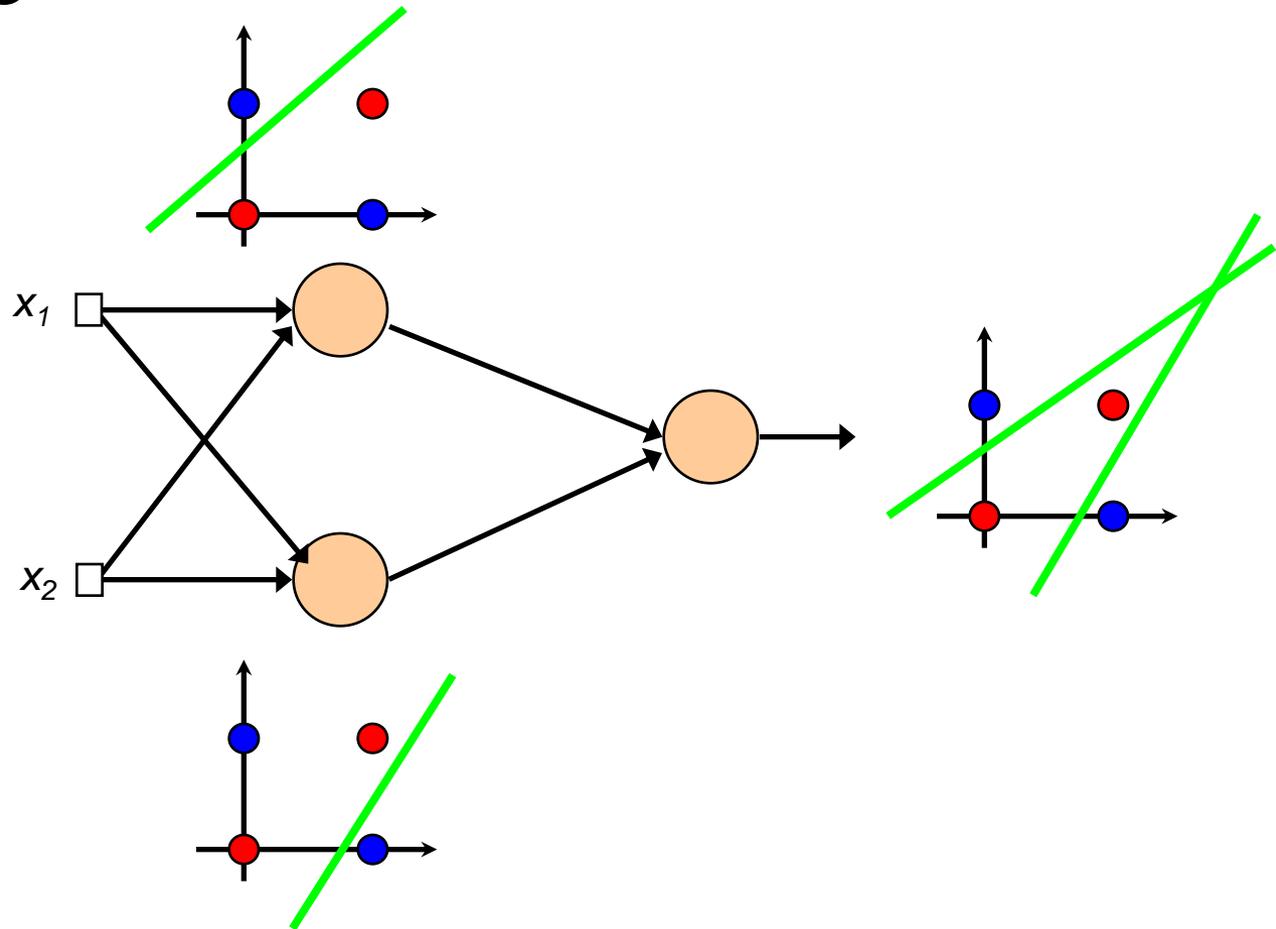
- RNAs com uma única camada de perceptrons resolvem apenas problemas linearmente separáveis

- **Solução**

- Utilizar mais de uma camada
  - » Camada 1: uma rede Perceptron para cada grupo de entradas linearmente separáveis
  - » Camada 2: uma rede combina as saídas das redes da primeira camada, produzindo a classificação final

# 2.3.1. Introdução ao MLP

- Exemplo

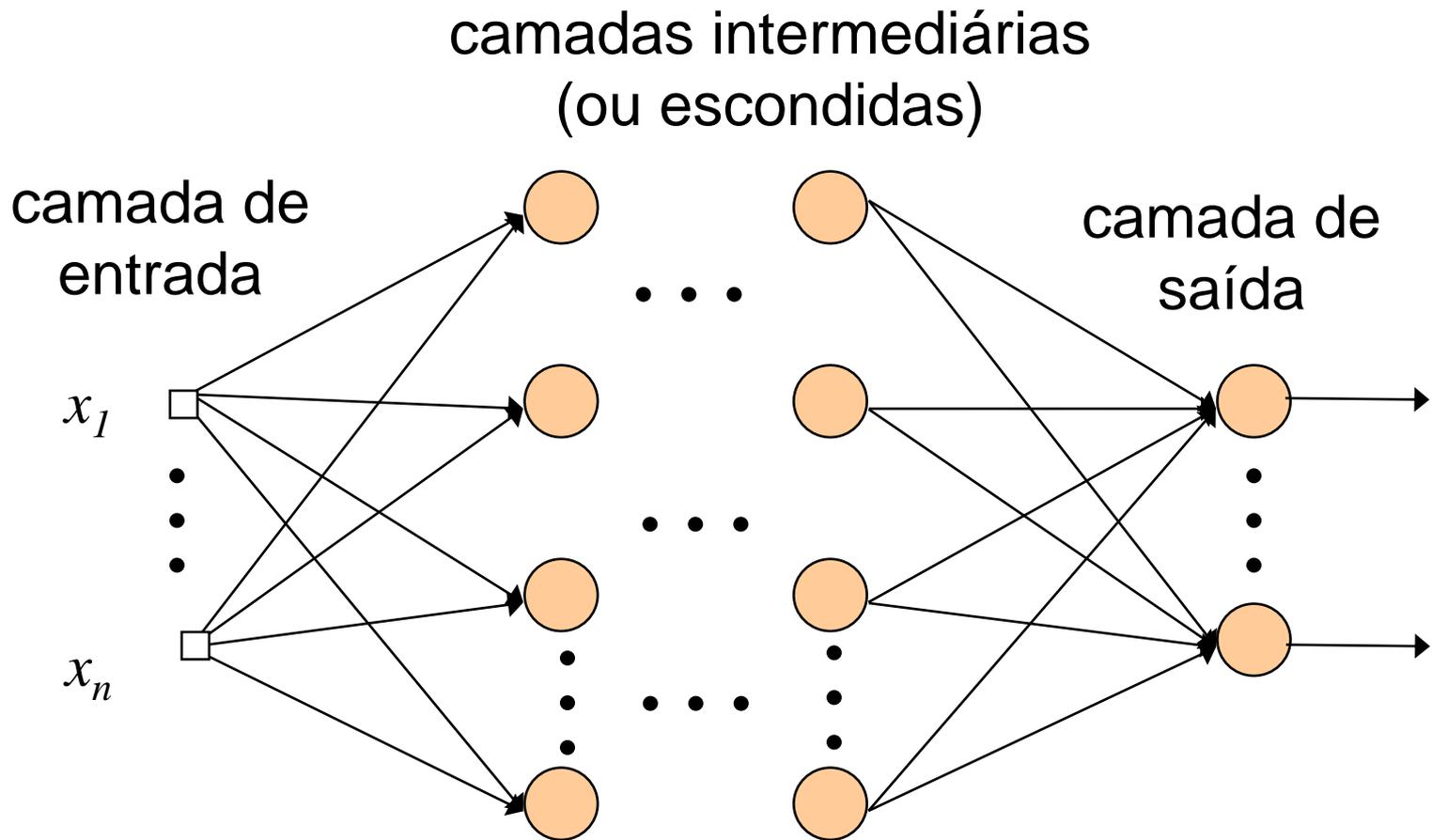


## 2.3.1. Introdução ao MLP

- **Modelo de RNA mais popular**
  - Tem sido aplicado com sucesso em uma série de problemas difíceis
    - » Resolve também os problemas que não são linearmente separáveis
- **O MLP consiste em**
  - Um conjunto de unidades sensoriais (nós fontes) que constituem a **camada de entrada**
  - Uma ou mais **camadas ocultas** (ou intermediárias) de nós computacionais (neurônios)
  - Uma **camada de saída** de nós computacionais (neurônios)

# 2.3.1. Introdução ao MLP

- Arquitetura



## 2.3.1. Introdução ao MLP

- No MLP, o sinal de entrada se propaga para a frente, passando pelas camadas ocultas em direção à camada de saída
  - Saída do neurônio  $j$

$$y_j(n) = \varphi(u_j(n)) \quad (2.30)$$

$$u_j(n) = \sum_{i=0}^m w_{ji}(n) o_i(n) \quad (2.31)$$

Se o neurônio  $j$  estiver na primeira camada oculta

$$o_i(n) = x_i(n) \quad (2.32)$$

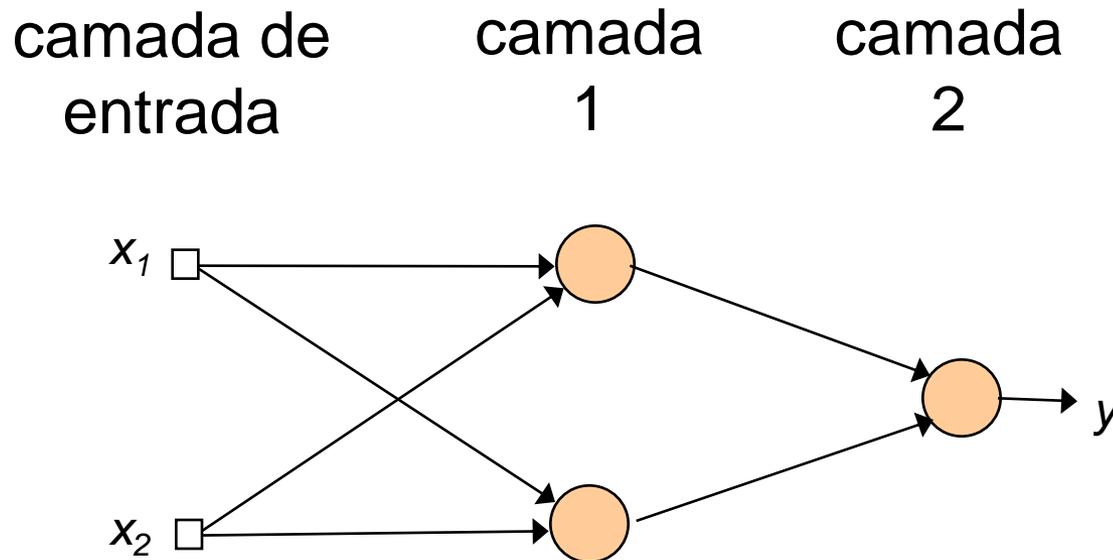
caso contrário

$$o_i(n) = y_i(n) \quad (2.33)$$

sendo  $y_i$  a saída do neurônio  $i$  da camada oculta anterior

## 2.3.1. Introdução ao MLP

### Exemplo 2.3.1. MLP que resolve a função XOR



Pesos:

Camada 1

$$w_{10} = -0,5$$

$$w_{11} = +1$$

$$w_{12} = +1$$

$$w_{20} = -1,5$$

$$w_{21} = +1$$

$$w_{22} = +1$$

Camada 2

$$w_{10} = -0,5$$

$$w_{11} = +1$$

$$w_{12} = -2,0$$

# 2.3.1. Introdução ao MLP

- **Funções de ativação**

- **Função de ativação linear**

- » Composição de funções lineares é uma função linear

- Sempre vai existir uma rede com uma camada equivalente a uma rede multicamadas com funções de ativação lineares

- Não resolve os problemas que não são linearmente separáveis

- **Funções não-lineares**

- » **Função sinal**

- Muda de valor abruptamente

- Não é diferenciável

- » **Funções sigmoidais**

- Contínua

- Diferenciável

## 2.3.1. Introdução ao MLP

- **Camadas ocultas com funções de ativação não-lineares**
  - Alteram a dimensão do problema
  - Transformações não-lineares sucessivas
  - Transformam o problema não-linearmente separável em um problema linearmente separável
    - » simplificam o problema para a camada de saída
  - Permite composição de funções

## 2.3.2. Treinamento

- **Treinamento da rede**

- Porque não utilizar o algoritmo de treinamento do perceptron?
  - » Qual o erro dos neurônios da camada intermediária?
- Grande variedade de algoritmos de treinamento
  - » Geralmente supervisionados
  - » Tipos
    - Estáticos
      - Não alteram estrutura da rede
    - Construtivos
      - Alteram estrutura da rede

## 2.3.2. Treinamento

- **Treinamento estático**

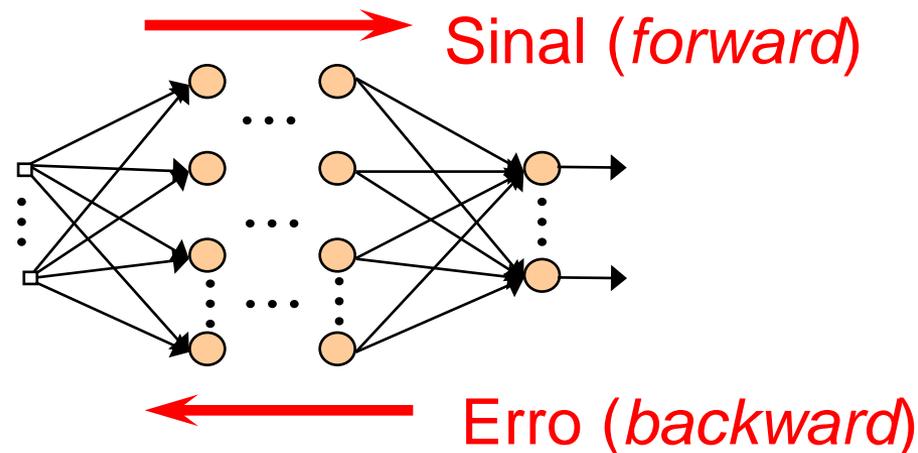
- MLPs com formatos e tamanhos diferentes podem utilizar mesma regra de aprendizado
- Algoritmo de retropropagação do erro (*Backpropagation*)
  - » Algoritmo mais utilizado
  - » Regra de aprendizado por correção de erro
  - » Generalização do algoritmo do mínimo quadrado médio (LMS)
    - Regra Delta
    - Utilização do gradiente descendente do erro

# 2.3.2.1. Backpropagation

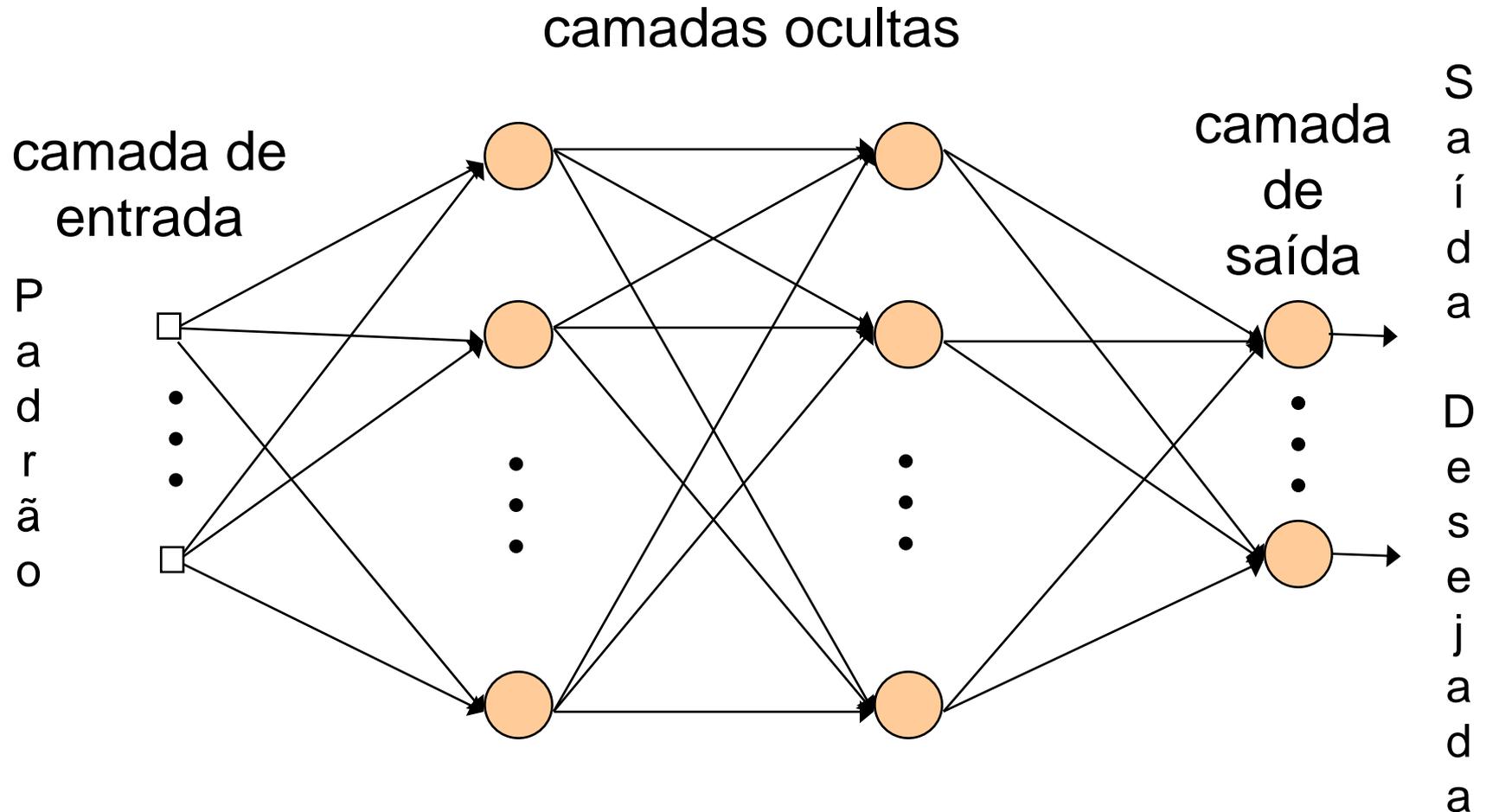
## • Backpropagation

- Desenvolvido por Rumelhart, Hinton e Williams (1986)
- Rede é treinada com pares entrada-saída
- Cada entrada de treinamento está associada a uma saída desejada
- Treinamento em duas fases, cada uma percorrendo a rede em um sentido

- » Fase *forward*
- » Fase *backward*



# 2.3.2.1. Backpropagation



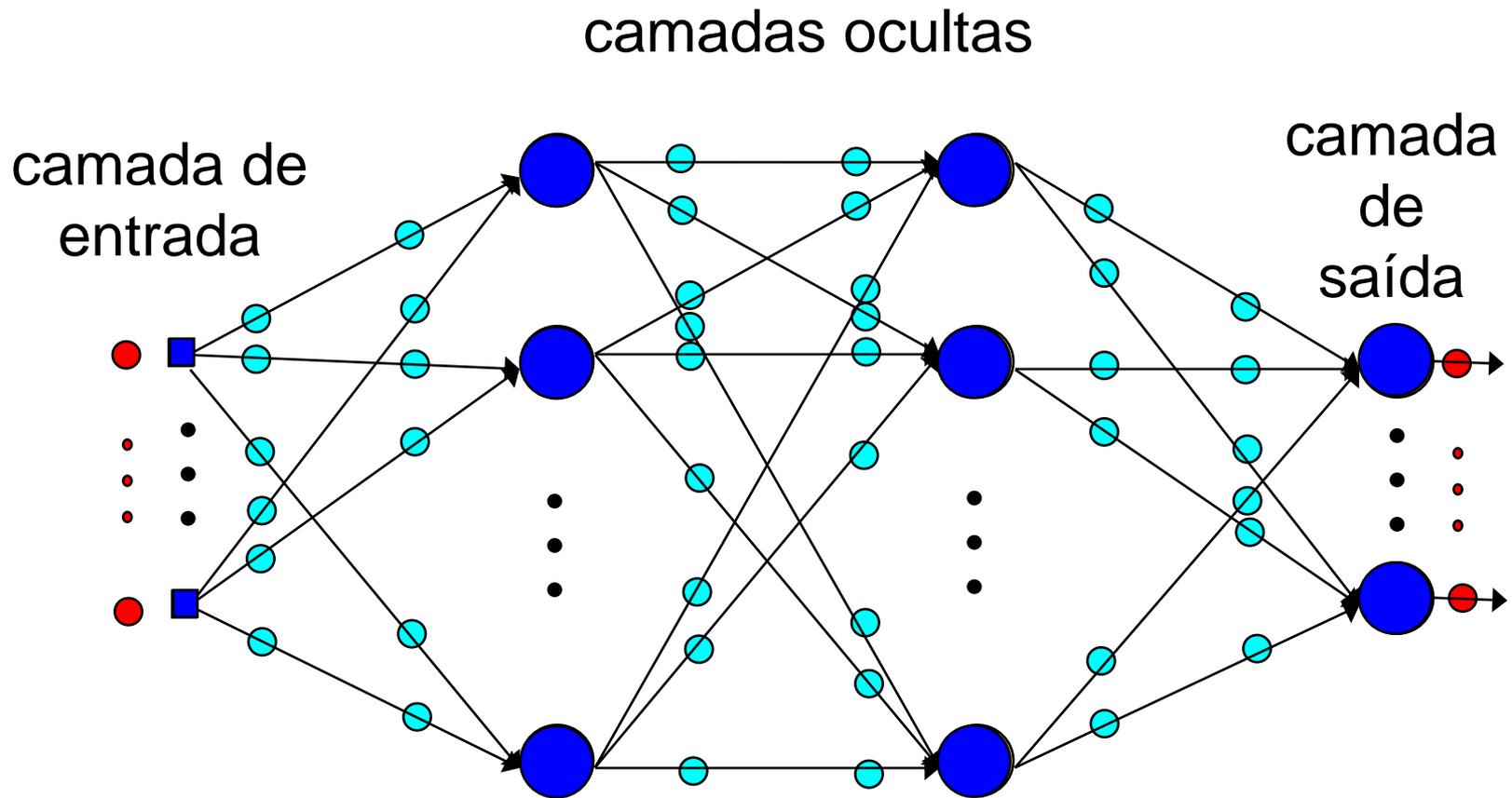
## 2.3.2.1. Backpropagation

- **Fase *forward***

- Entrada é apresentada à primeira camada da rede
- Após os neurônios de uma camada calcularem seus valores de saída, os neurônios da camada posterior utilizam estes valores para definir suas saídas
- Saídas produzidas pelos neurônios da última camada são comparadas às saídas desejadas
- Erro para cada neurônio da camada de saída é calculado

# 2.3.2.1. Backpropagation

- **Fase *forward***

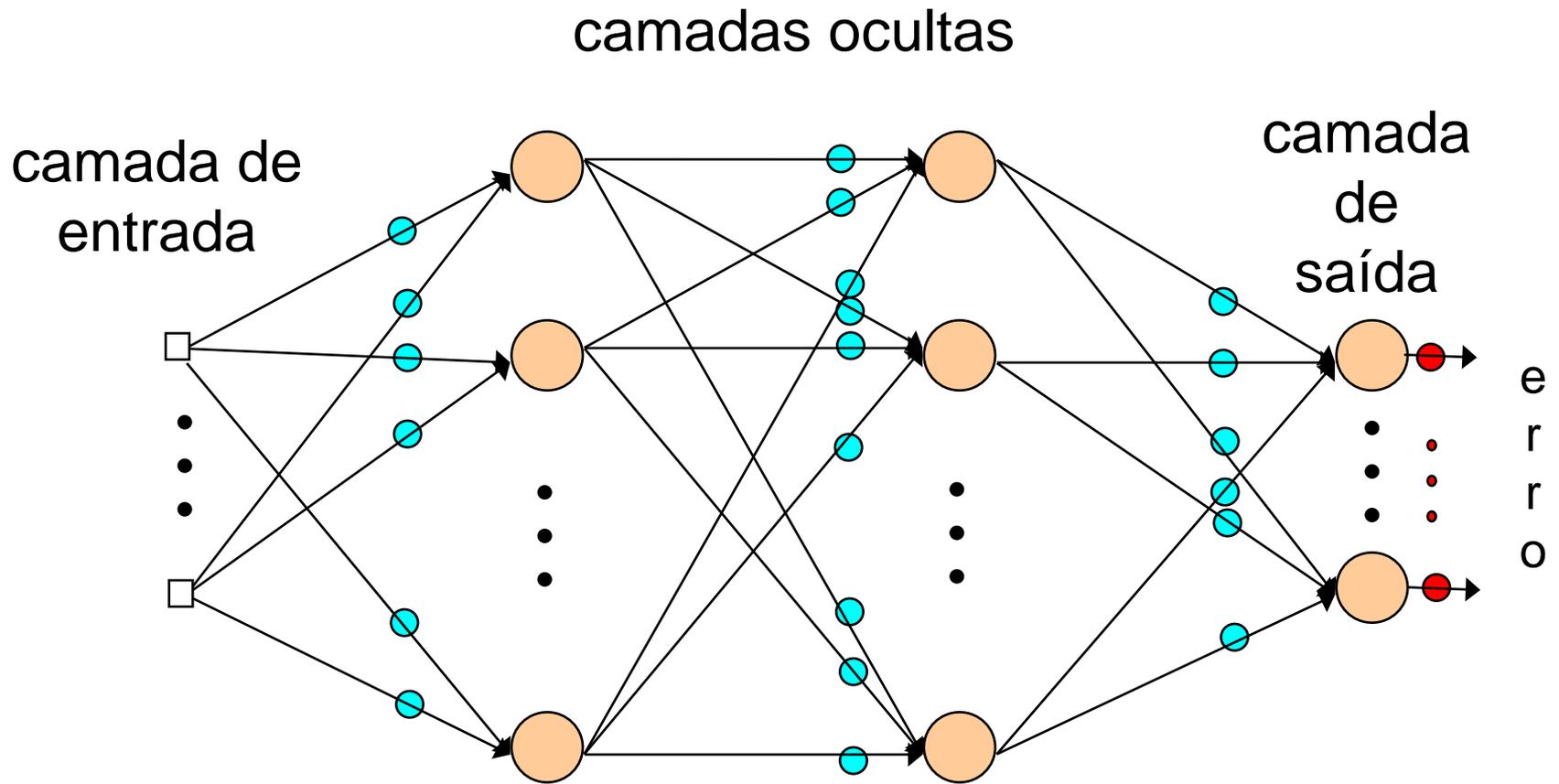


## 2.3.2.1. Backpropagation

- **Fase *backward***
  - Cada nodo ajusta seus pesos de modo a reduzir seu erro
  - Cada nodo das camadas anteriores tem seu erro definidos por:
    - » Erros dos nodos da camada seguinte conectados a ele, ponderados pelos pesos das conexões entre ele e estes nodos

# 2.3.2.1. Backpropagation

- Fase *backward*



## 2.3.2.1. Backpropagation

- Durante o treinamento, busca-se minimizar uma função de custo baseada no erro instantâneo
  - Erro instantâneo entre a saída produzida pelo neurônio  $j$  da camada de saída na iteração  $n$  e o seu valor desejado

$$e_j(n) = d_j(n) - y_j(n) \quad (2.34)$$

- Definindo a energia do erro para o neurônio  $j$  como  $e_j^2(n)/2$ , o valor instantâneo da energia total de erro é dada por

$$E(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n) \quad (2.35)$$

sendo que o conjunto  $C$  engloba todos os neurônios da camada de saída

## 2.3.2.1. Backpropagation

- Podemos ainda definir a **energia média do erro quadrático (ou erro quadrático médio)** para o conjunto de treinamento

$$E_{med} = \frac{1}{N} \sum_{n_{norm}=1}^N E(n_{norm}) \quad (2.36)$$

na qual  $n_{norm}$  é o valor de  $n$  normalizado entre 1 e  $N$  (tamanho do conjunto de treinamento)

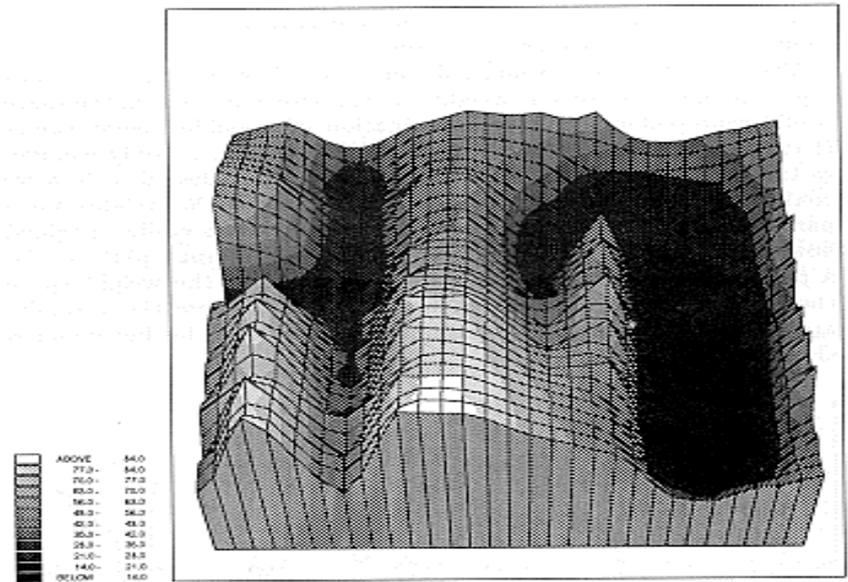
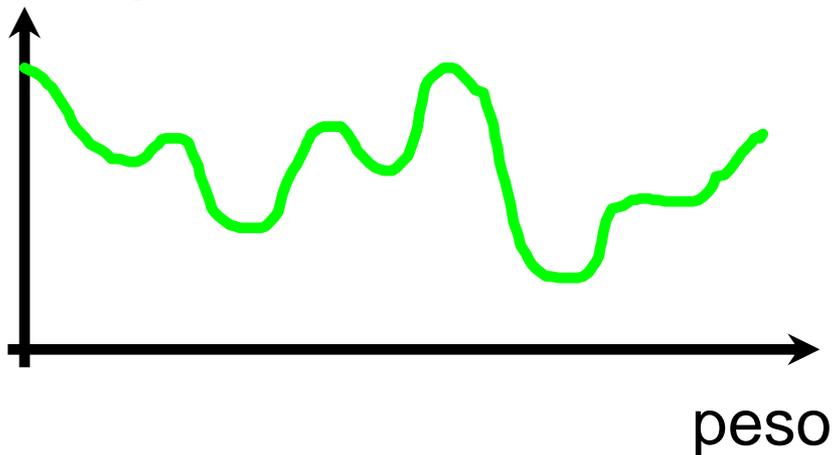
## 2.3.2.1. Backpropagation

- Os erros produzidos são funções dos parâmetros livres do MLP
- Minimização do erro ocorre através do ajuste dos parâmetros livres
  - Deve-se buscar o mínimo da superfície de erro
  - Utilização do gradiente descendente da energia total do erro

# 2.3.2.1. Backpropagation

- **Superfície de erro**
  - Exemplos

energia total do erro



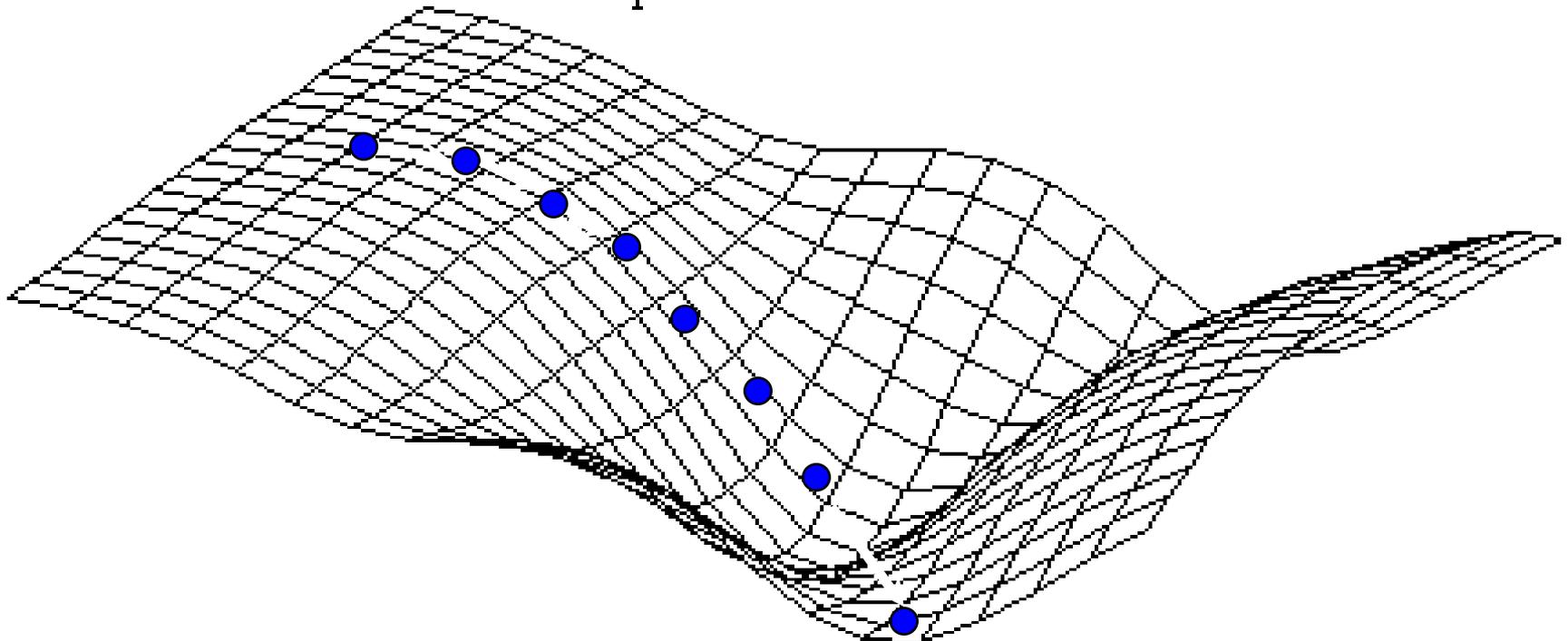
## 2.3.2.1. Backpropagation

- **Backpropagation aproxima trajetória no espaço de pesos computada pelo método do gradiente descendente**
- **Gradiente da função erro (energia total do erro)**
  - está na direção e sentido em que a função de energia do erro tem a taxa de variação máxima
  - para um dado peso é dado pela derivada parcial da função erro em relação ao peso

## 2.3.2.1. Backpropagation

- Exemplo bidimensional

Superfície de Erro



## 2.3.2.1. Backpropagation

### • Treinamento

- Em cada iteração  $n$ , uma correção  $\Delta w_{ji}(n)$  é aplicada no peso sináptico entre o neurônio  $i$  da camada anterior (ou entrada no caso da primeira camada oculta) e o neurônio  $j$
- A correção  $\Delta w_{ji}(n)$  é proporcional à derivada parcial (regra delta)

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \quad (2.37)$$

» O uso do sinal negativo indica a descida do gradiente na superfície de erro

□ Busca-se uma direção para a mudança do peso que reduza o valor de  $E(n)$

## 2.3.2.1. Backpropagation

### • Treinamento

- Substituindo as Eqs. (2.30), (2.31), (2.34) e (2.35) na Eq. (2.37) e resolvendo as derivadas parciais (ver desenvolvimento na lousa), temos a **regra delta** de ajuste de pesos

$$\Delta w_{ji}(n) = \eta \delta_j(n) o_i(n) \quad (2.38)$$

na qual  $o_i(\cdot)$  é a entrada  $i$  do neurônio  $j$  (Eqs. 2.32 e 2.33),  $\eta$  é a taxa de aprendizagem e o **gradiente local**  $\delta_j(\cdot)$  é definido como

$$\delta_j(n) = - \frac{\partial E(n)}{\partial u_j(n)} \quad (2.39)$$

## 2.3.2.1. Backpropagation

- **Treinamento**

- A resolução da Eq. (2.39) resulta em (ver desenvolvimento na lousa) em

$$\delta_j(n) = \varphi'(u_j(n)) e_j(n) \quad (2.40)$$

se o neurônio  $j$  está na camada de saída, e

$$\delta_j(n) = \varphi'(u_j(n)) \sum_k \delta_k(n) w_{kj}(n) \quad (2.41)$$

na qual  $k$  indica as conexões de saída do neurônio  $j$

se o neurônio  $j$  está em uma camada oculta.

## 2.3.2.1. Backpropagation

- **Treinamento**

- Se a função logística (sigmoïdal) é utilizada (ver Eq. 2.6) considerando-se entradas finitas e  $a > 0$ , então a derivada da função de ativação é dada por (ver desenvolvimento na lousa)

$$\varphi'(u_j(n)) = a y_j(n) [1 - y_j(n)] \quad (2.42)$$

# 2.3.2.1. Backpropagation

**Algoritmo**  $\text{mlp}( \mathbf{X}_T, \mathbf{d}_T, \eta )$

*Iniciar todas as conexões com valores aleatórios*

$n \leftarrow 1$

**faça**

*// fase forward: apresentação do par de treinamento  $\mathbf{x}(n)$ ,  $\mathbf{d}(n)$*

**para** cada camada  $k$  do MLP (começando da primeira camada)

**para** cada neurônio  $j$  da camada  $k$

*Calcular a saída  $y_j$*

*// Eqs. (2-30-2.33)*

**fim para**

**fim para**

*Calcular o valor instantâneo da energia total do erro  $E(n)$*

*// Eqs. (2.34) e (2.35)*

*// fase backward: correção dos pesos*

**se**  $(E(n) > 0)$

**para** cada camada  $k$  do MLP (começando da última camada)

**para** cada neurônio  $j$  da camada  $k$

*atualizar pesos  $w_{ji}(n)$  para todo  $i$  // Eqs. (2.38), (2.40-2.42)*

**fim para**

**fim para**

**fim se**

$n \leftarrow n+1$

**enquanto** *(critério de convergência não tiver sido satisfeito)*

## 2.3.2.2. Dificuldades do Treinamento

- **Convergência**

- Em geral, o treinamento de RNAs é lenta se comparado com outros métodos de AM
- Nem sempre converge para o **ótimo global**
  - » Configuração não-adequada
    - Ex.: escolha incorreta de parâmetros
  - » Presença de **ótimos locais**
    - Estratégia: iniciar com pesos aleatórios e executar o algoritmo diversas vezes

## 2.3.2.2. Dificuldades do Treinamento

- **Convergência**

- Outros métodos têm sido propostos para acelerar o treinamento (em relação ao *Backpropagation*)
  - » Ex.: Levenberg–Marquardt
    - Uso de segundas derivadas
  - » No entanto, estes algoritmos são melhores sob certas circunstâncias
- Metaheurísticas populacionais também têm sido utilizadas
  - » No entanto, o treinamento é geralmente lento

- **Escolha dos parâmetros**

- É difícil determinar *a priori* qual a melhor configuração
- Influência especialmente no treinamento
- Novamente aqui, metaheurísticas populacionais tem sido utilizadas

- **Referências**

- **Haykin, S. S.. *Redes neurais: princípios e prática*. 2ª ed., Bookman, 2001.**
  - » Capítulo 4
- **Braga, A.P.; Carvalho, C.P.L.F. & Ludermir, T.B.. *Redes neurais artificiais: Teoria e Aplicações*. LTC, 2000.**
  - Capítulo 4
- **Principe, J. C.; Euliano, N. R. & Lefebvre, W. C. *Neural and Adaptive Systems: Fundamentals Through Simulations*. John Wiley & Sons, Inc. 2000**
  - Capítulos 2, 3, 4 e 5