4. Perceptron Multicamadas

Prof. Renato Tinós

Depto. de Computação e Matemática (FFCLRP/USP)

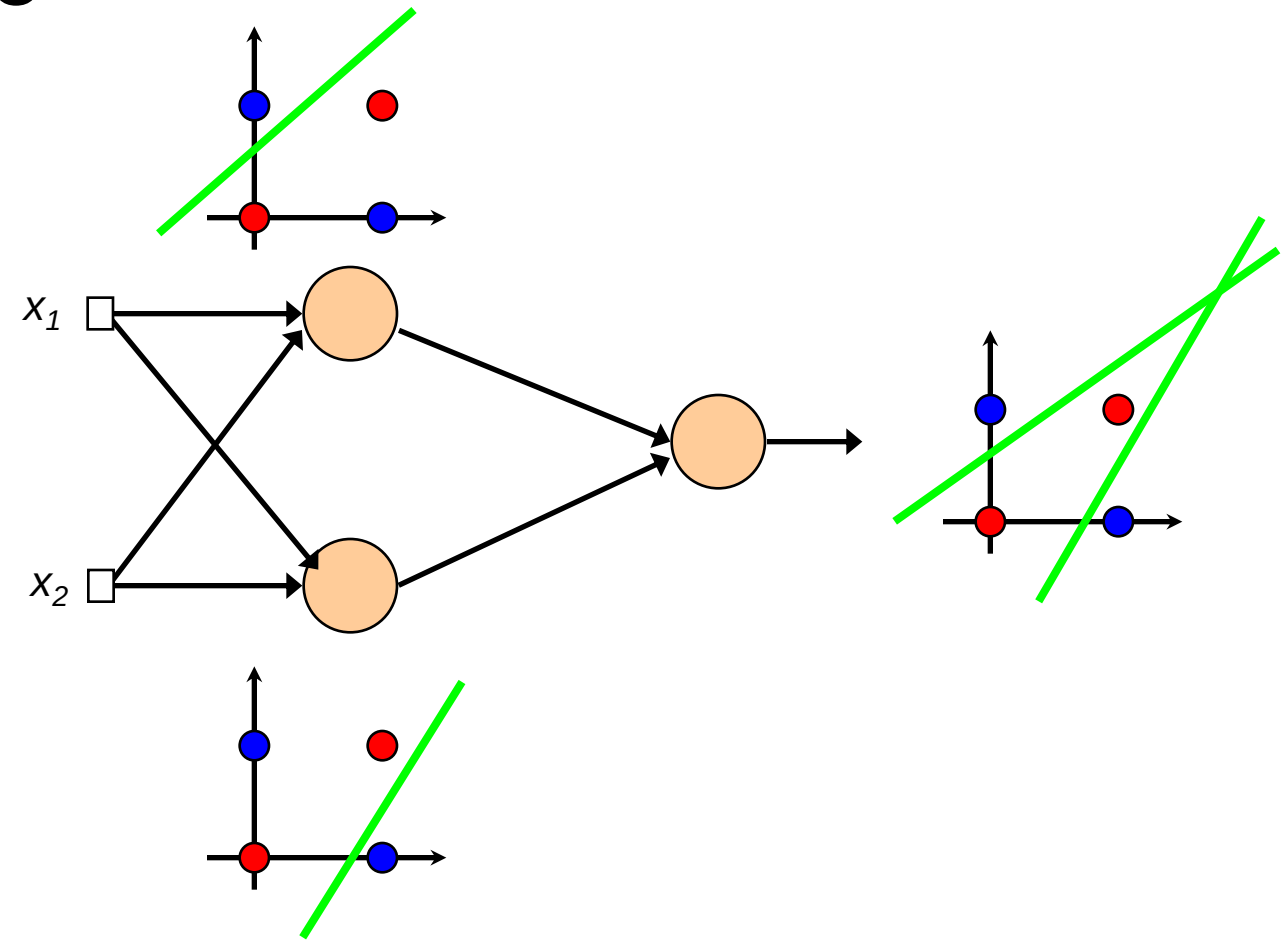
4. Perceptron Multicamadas - MLP

4.1. 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

4.1. Introdução ao MLP

- **Exemplo**

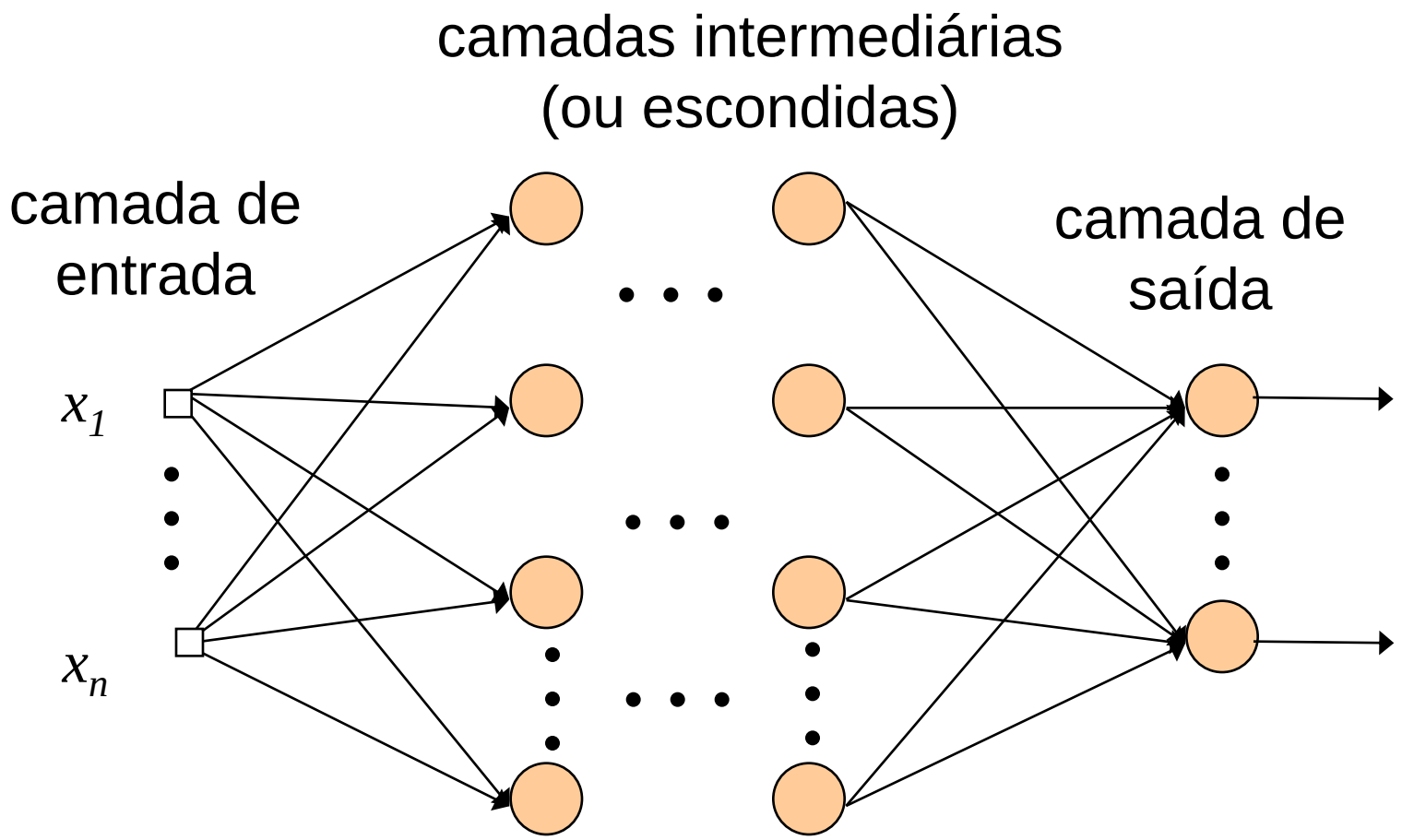


4.1. Introdução ao MLP

- **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)

4.1. Introdução ao MLP

- **Arquitetura**



4.1. 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*

$$y_j(n) = \varphi(u_j(n)) \tag{4.1}$$

$$u_j(n) = \sum_{i=0}^{m} w_{ji}(n) o_i(n) \tag{4.2}$$

Se o neurônio *j* estiver na primeira camada oculta

$$o_i(n) = x_i(n) \tag{4.3}$$

caso contrário

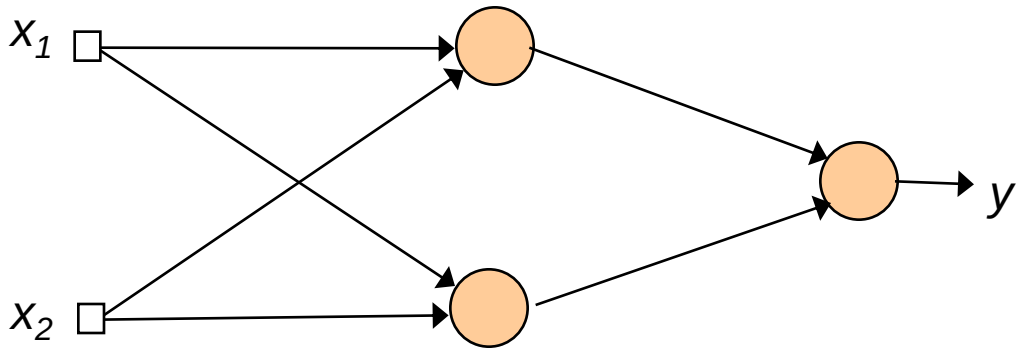
$$o_i(n) = y_i(n) \tag{4.4}$$

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

4.1. Introdução ao MLP

Exemplo 4.1. MLP que resolve a função XOR

camada de entrada | camada 1 | camada 2



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

$w_{10} = -0,5$

$w_{11} = +1$

$w_{12} = -2,0$

4.1. 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

4.1. 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

4.2. Treinamento

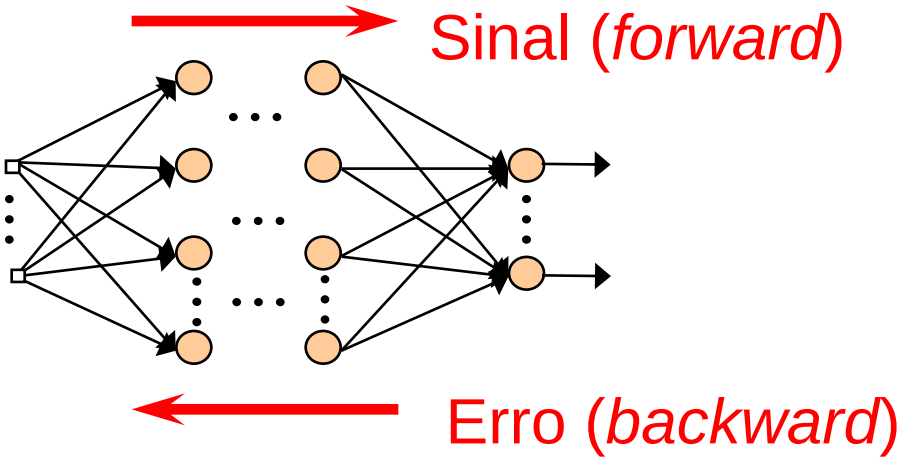
- **Treinamento da rede**
  - Porque não utilizar o algoritmo de treinamento do perceptron?
    - » Qual o erro dos neurônios da camada intermediária?
  - Grande variedade de algoritmos de treinamento
    - » Geralmente supervisionados
    - » Tipos
      - ❑ Estáticos
        - ➢ Não alteram estrutura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

4.2. Treinamento

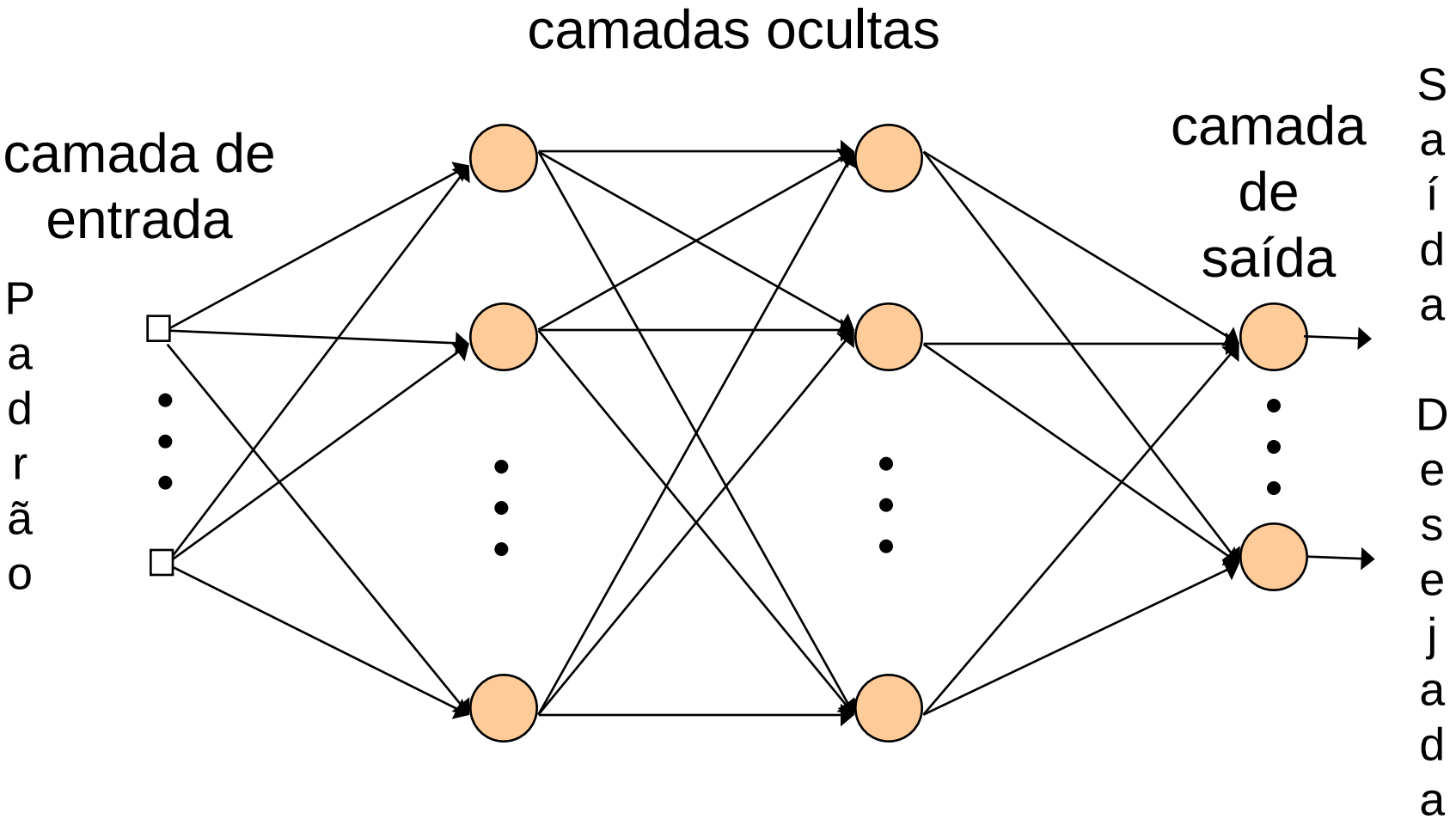
- **Treinamento estático**
  - MLPs com formatos e tamanhos diferentes podem utilizar mesma regra de aprendizado
  - Algoritmo de retropropagação do erro (*Backpropagation*)
    - » Algoritmo mais utilizado
    - » Regra de aprendizado por correção de erro
    - » Generalização do algoritmo do mínimo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

4.2.1. Backpropagation

- **Backpropagation**
  - Desenvolvido por Rumelhart, Hinton e Willians (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*



4.2.1. Backpropagation

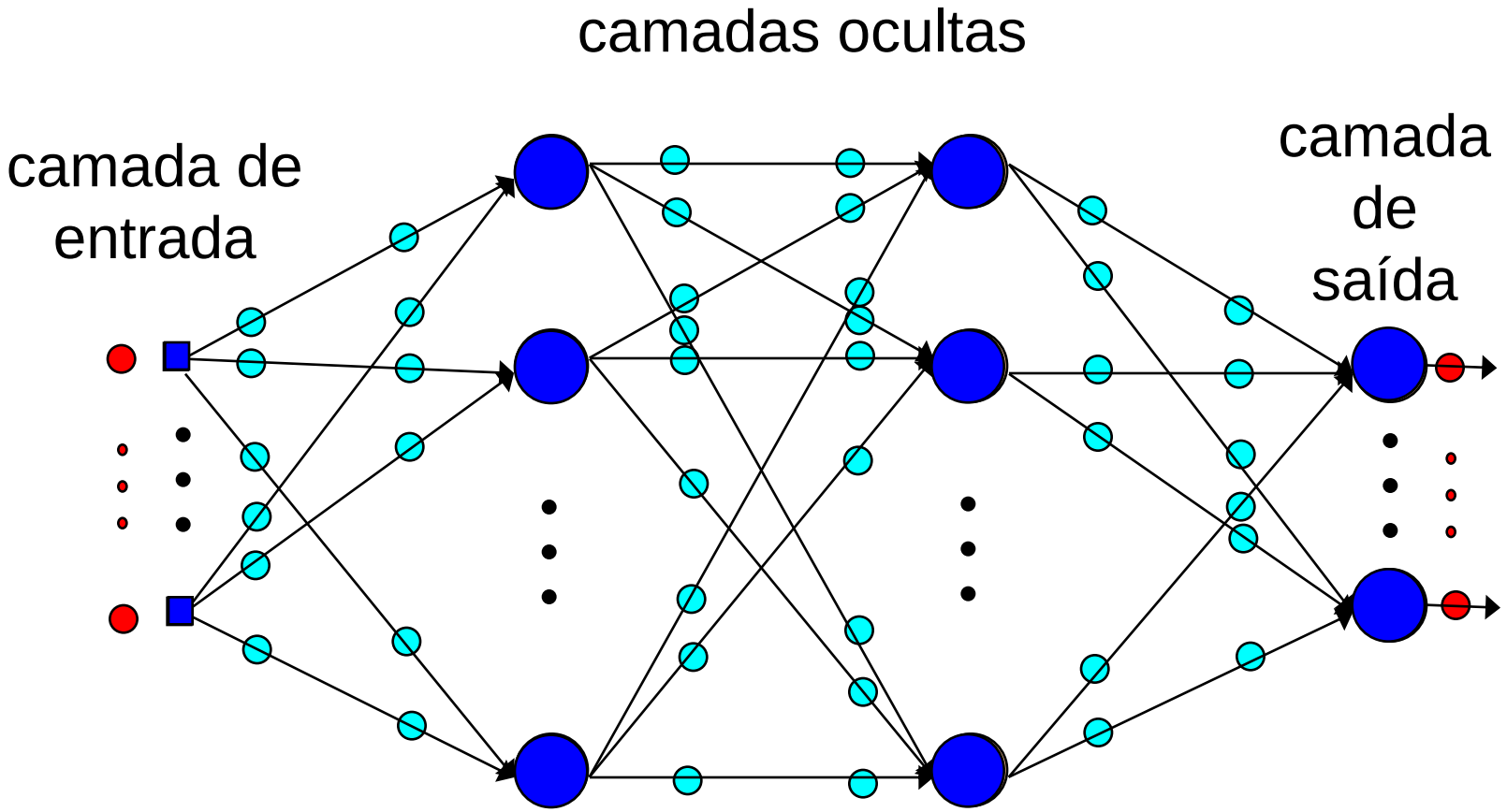


4.2.1. Backpropagation

- **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

4.2.1. Backpropagation

- **Fase *forward***

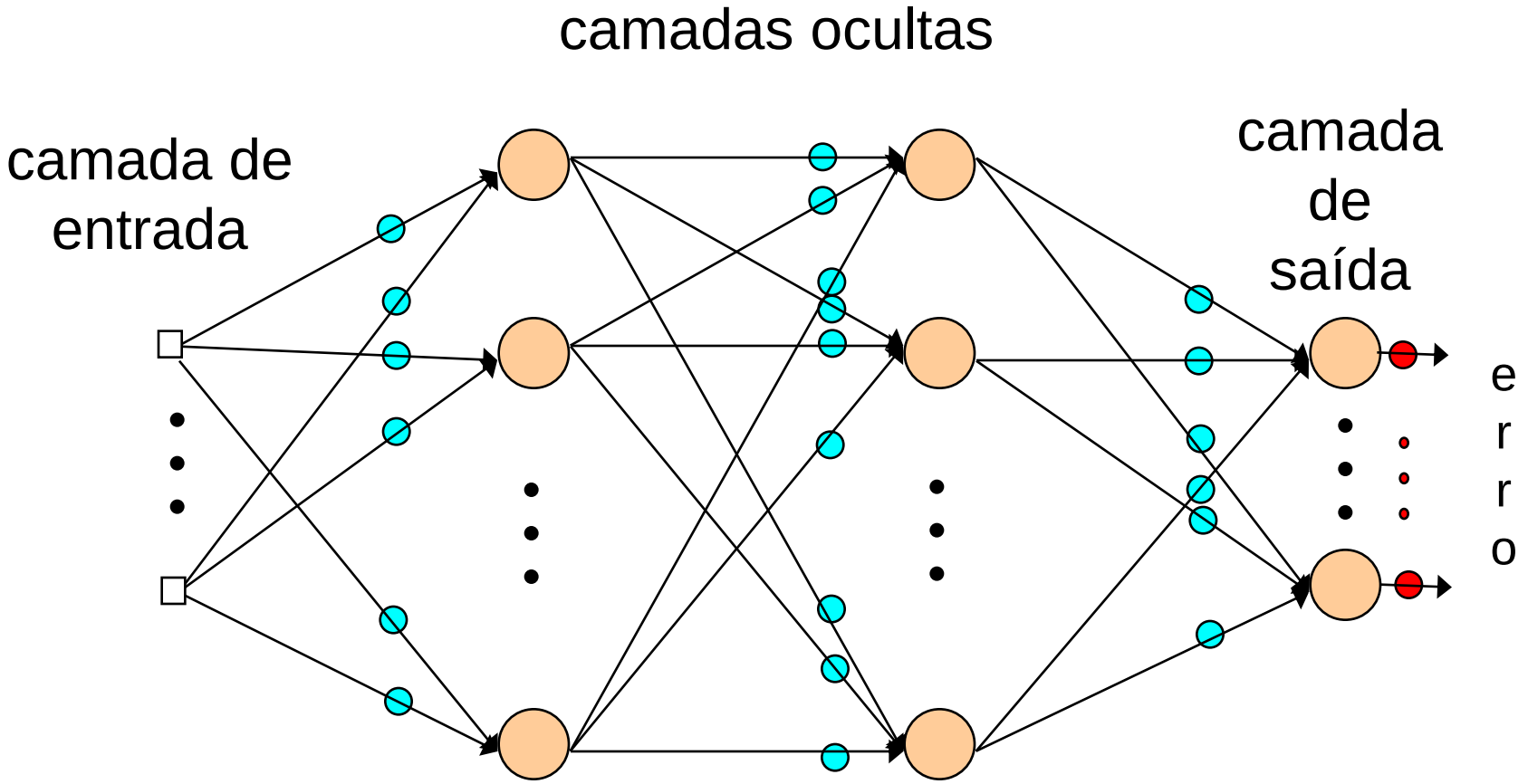


4.2.1. Backpropagation

- **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

4.2.1. Backpropagation

- **Fase *backward***



4.2.1. 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

$$e_j(n) = d_j(n) - y_j(n) \qquad (4.5)$$

  - 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

$$E(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) \qquad (4.6)$$

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

4.2.1. Backpropagation

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

$$E_{med} = \frac{1}{N} \sum_{n_{norm}=1}^{N} E(n_{norm}) \qquad (4.7)$$

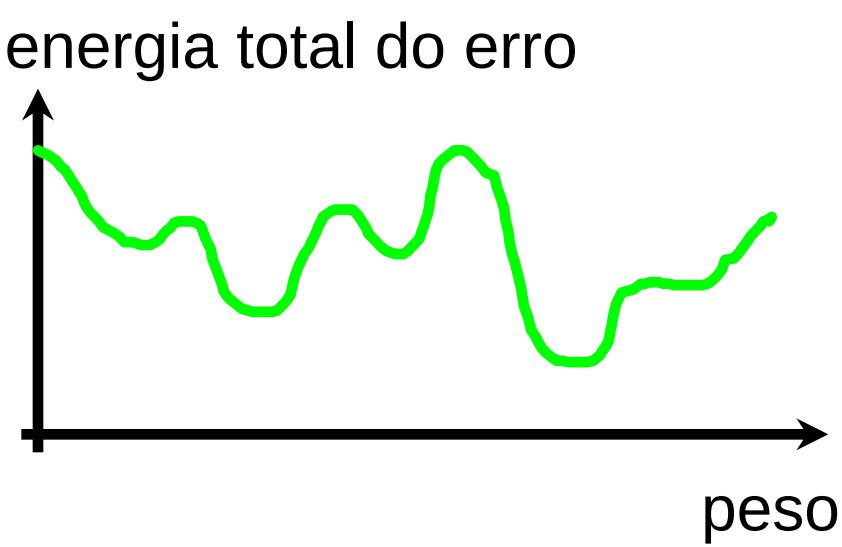
na qual n_{norm} é o valor de n normalizado entre 1 e N (tamanho do conjunto de treinamento)

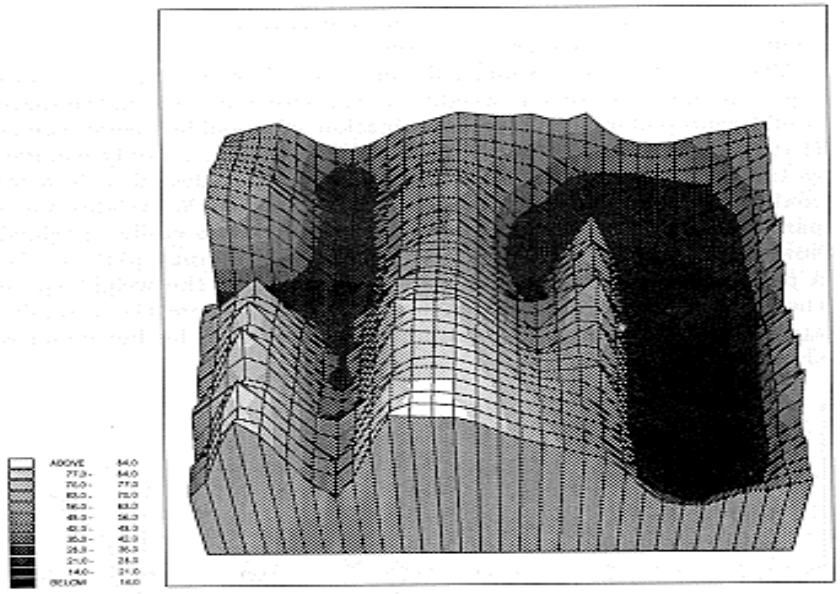
4.2.1. Backpropagation

- **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

4.2.1. Backpropagation

- **Superfície de erro**
  - Exemplos



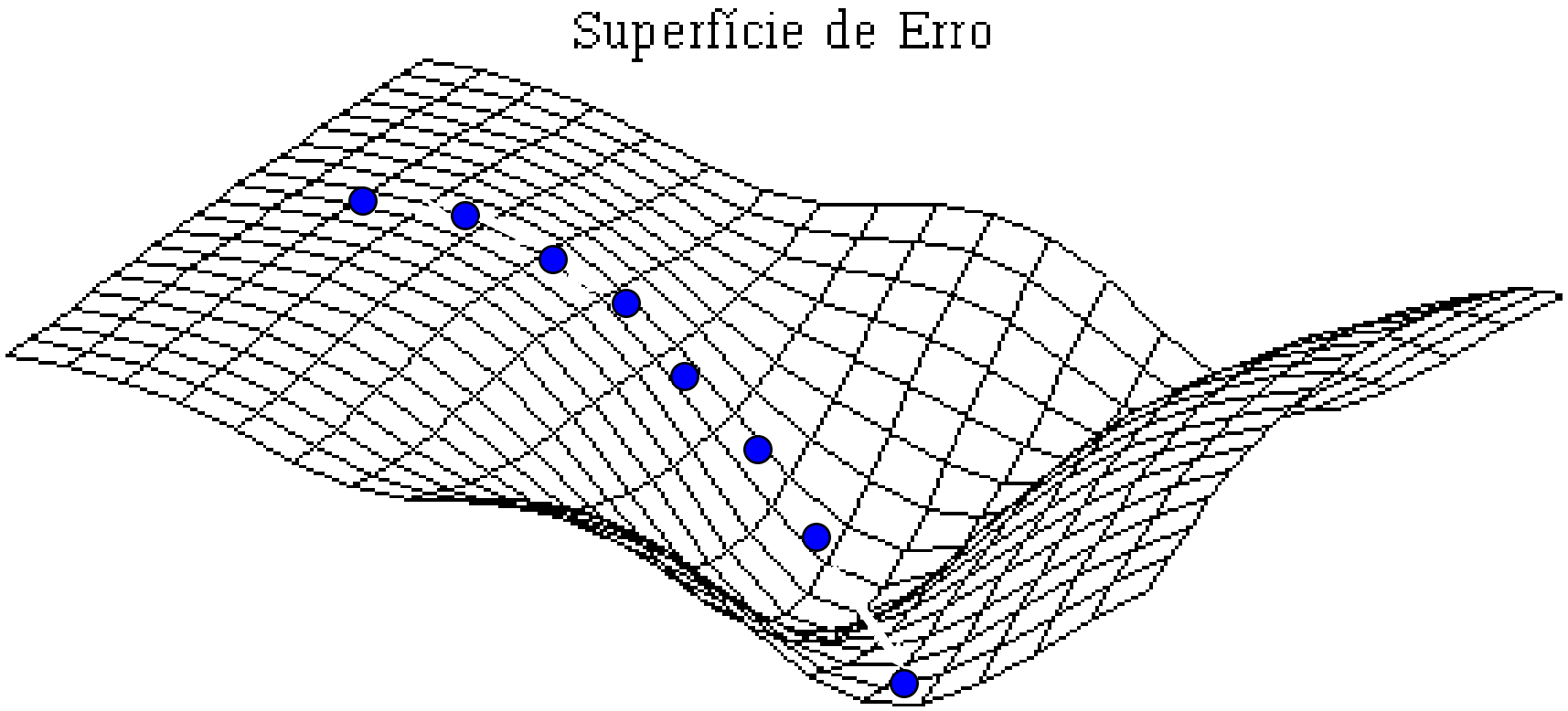


4.2.1. Backpropagation

- **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

4.2.1. Backpropagation

- Exemplo bidimensional



4.2.1. 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)

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \qquad (4.8)$$

    - » 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 $E(n)$

4.2.1. Backpropagation

- **Treinamento**
  - Substituindo as Eqs. (4.1)-(4.6) na Eq. (4.8) e resolvendo as derivadas parciais (ver desenvolvimento na lousa), temos a regra delta de ajuste de pesos

$$\Delta w_{ji}(n) = \eta\, \delta_j(n)\, o_i(n) \qquad (4.9)$$

na qual $o_i(.)$ é a *i*-ésima entrada do neurônio *j* (Eqs. 4.3 e 4.4), η é a taxa de aprendizagem e o gradiente local $\delta_j(.)$ é definido como

$$\delta_j(n) = -\frac{\partial E(n)}{\partial u_j(n)} \qquad (4.10)$$

4.2.1. Backpropagation

- **Treinamento**
  - A resolução da Eq. (4.10) resulta em (ver desenvolvimento na lousa) em

$$\delta_j(n) = \varphi'\left(u_j(n)\right) e_j(n) \qquad (4.11)$$

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

$$\delta_j(n) = \varphi'\left(u_j(n)\right) \sum_k \delta_k(n) w_{kj}(n) \qquad (4.12)$$

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

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

4.2.1. Backpropagation

- **Treinamento**
  - Se a função logística (sigmoidal) 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)

$$\varphi'\left(u_j(n)\right) = a\, y_j(n)\left[1 - y_j(n)\right] \qquad (4.13)$$

4.2.1. Backpropagation

Algoritmo mlp($\mathbf{X_T}$, $\mathbf{d_T}$, η)

Iniciar todas as conexões com valores aleatórios
$n \leftarrow 1$

faça
    // fase forward: apresentação do par de treinamento ***x****(n) ,* ***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. (4.1-4.4)*
        fim para
    fim para

    // fase backward: correção dos pesos
    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. (4.9), (4.11-4.13)*
        fim para
    fim para

    $n \leftarrow n+1$
enquanto *(critério de convergência não tiver sido satisfeito)*

4.2.2. 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

4.2.2. 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

Comentários

- **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