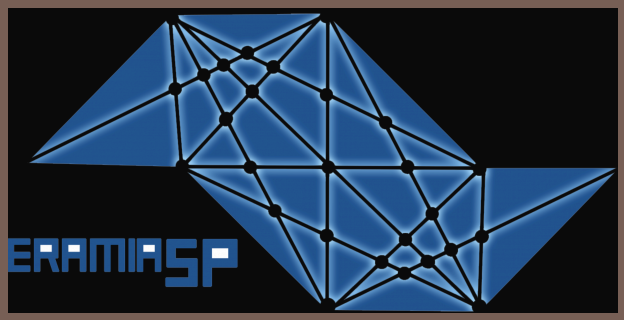


MULTILAYER PERCEPTRON

PROF. RICARDO CERRI

ERAMIA 2020

Universidade Federal de São Carlos
Departamento de Computação - DC/UFSCar

1

Multilayer Perceptron

2

- Supera as limitações práticas do Perceptron
  - O modelo de cada neurônio inclui uma função de ativação não linear e diferenciável
  - Contém uma ou mais camadas escondidas entre a camada de entrada e a camada de saída
  - A rede possui alto grau de conectividade

2

Multilayer Perceptron

3

- Deficiências
  - Análise teórica difícil, pois há muitas conexões e funções não lineares
  - Muitos neurônios escondidos tornam difícil a visualização do processo de aprendizado
  - O aprendizado é mais difícil, pois há um espaço muito maior de possíveis funções. Há mais representações dos padrões de entrada

3

Multilayer Perceptron

4

- Como aprender? Back-propagation
  - **Forward phase**: pesos fixos e o sinal é propagado através da rede, camada por camada, até a saída
  - Mudanças só ocorrem nos potenciais de ativação e nas saídas dos neurônios da rede

4

Multilayer Perceptron

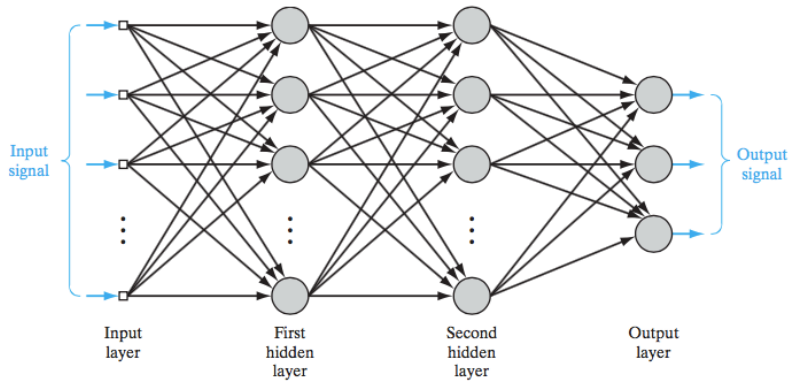
5

- Como aprender? Back-propagation
  - **Backward phase**: um sinal de erro é produzido comparando a saída desejada com a obtida
  - O erro é retropropagado através da rede, camada por camada
  - Ajustes são realizados nos pesos sinápticos da rede

5

Multilayer Perceptron

6



6

Multilayer Perceptron

7

- **Sinais de função**: sinal (estímulo) que vem da entrada da rede, é propagado neurônio a neurônio, e sai no fim da rede como uma sinal de saída
  - Presume-se que vá desempenhar uma função útil na saída da rede
  - Em cada neurônio, o sinal é calculado como uma função das entradas e pesos associados

7

Multilayer Perceptron

8

- **Sinais de erro**: sinal que tem origem na saída da rede (neurônio de saída) e é retropropagado camada à camada através da rede
  - Seu cálculo, para cada neurônio da rede, envolve uma função que depende do erro obtido na saída da rede

8

Multilayer Perceptron

9

- Sinais de função e sinais de erro

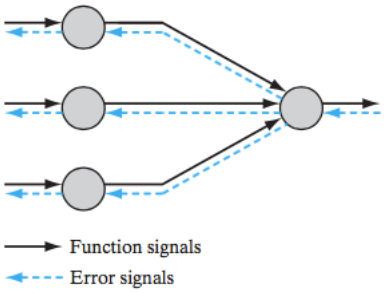


FIGURE 4.2 Illustration of the directions of two basic signal flows in a multilayer perceptron: forward propagation of function signals and back propagation of error signals.

9

Multilayer Perceptron

10

- Os neurônios de saída constituem a camada de saída da rede. Os neurônios restantes constituem as camadas escondidas
- Os neurônios escondidos não são parte nem da entrada, nem da saída da rede
  - A primeira camada escondida é alimentada com a saída da camada de entrada.
  - A saída resultante é então aplicada à segunda camada escondida, e assim em diante para toda a rede

10

Multilayer Perceptron

11

- Cada neurônio, de saída ou escondido, é desenvolvido para executar dois cálculos
  - Do sinal de função que aparece na saída de cada neurônio, expresso como uma função não linear contínua do sinal de entrada e pesos sinápticos associados
  - Da estimativa do vetor gradiente (gradientes da superfície do erro com respeito aos pesos conectados às entradas dos neurônios). Necessário para a fase de retropropagação

11

Multilayer Perceptron

12

- Função dos neurônios escondidos
  - Agem como detectores de atributos. Conforme o aprendizado progride, esses neurônios começam a descobrir os atributos que caracterizam os dados de treinamento
  - Isso é feito por meio da transformação não linear dos dados de entrada em um novo espaço chamado de espaço de características
  - Nesse novo espaço, classes (por exemplo em um problema de classificação) podem ser mais facilmente separadas umas das outras do que no espaço de entrada original

12

Multilayer Perceptron

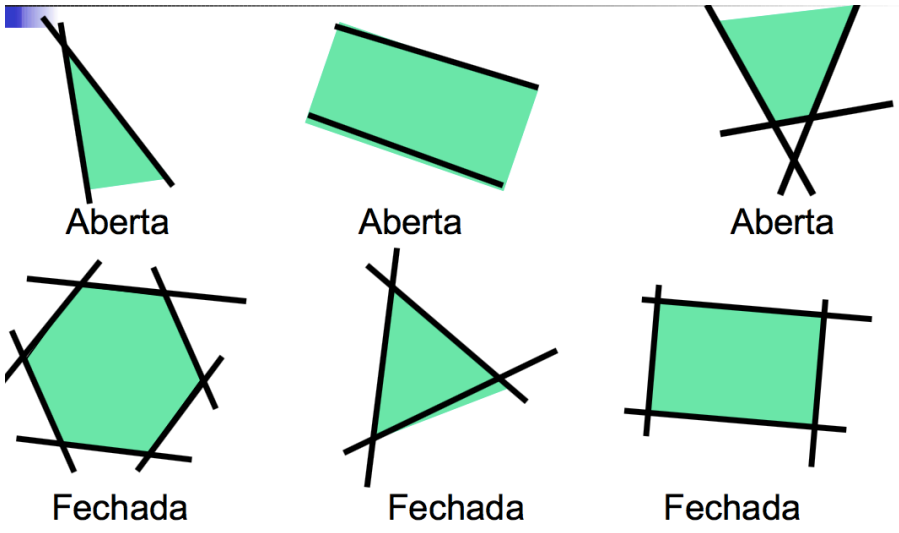
13

- Camadas intermediárias
  - Primeira camada: linhas retas no espaço de decisão
  - Segunda camada: combina as linhas da camada anterior para formar regiões convexas
  - Terceira camada: combina figuras convexas produzindo formatos abstratos

13

Multi-layer Perceptrons – Regiões Convexas

14

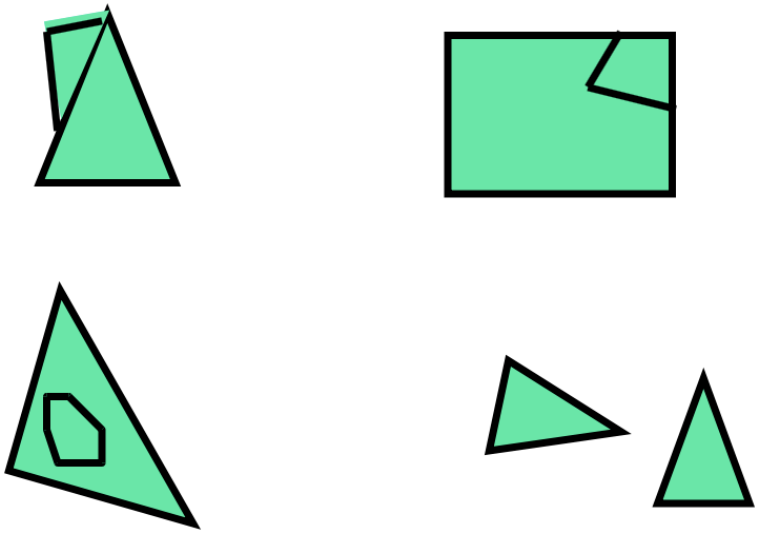


14

Multi-layer Perceptrons

Combinações de Regiões Convexas

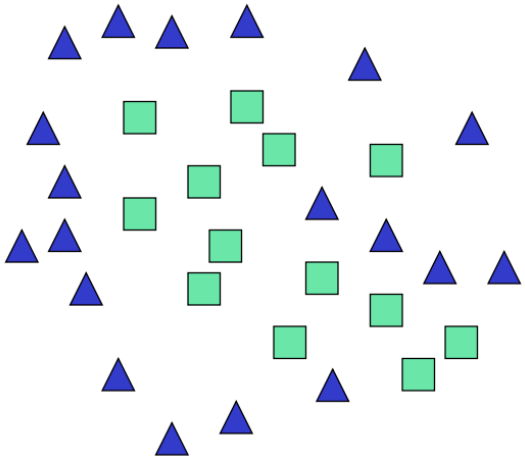
15



15

Multilayer Perceptron

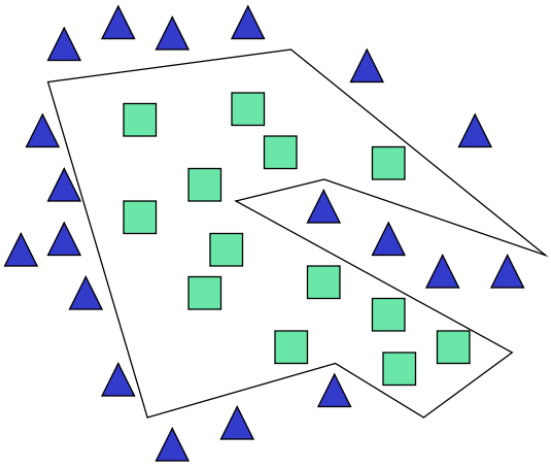
16



16

Multilayer Perceptron

17



17

Multilayer Perceptron

18

- Considere um Multilayer Perceptron.
- Considere $\tau = \{\mathbf{x}(n), \mathbf{d}(n)\}_{n=1}^{N}$ um exemplo de treinamento. Seja $y_j(n)$ o sinal produzido na saída do neurônio j na camada de saída, estimulado por $\mathbf{x}(n)$ aplicado na camada de entrada
- O sinal de erro produzido na saída do neurônio j é dado por $e_j(n) = d_j(n) - y_j(n)$

18

Multilayer Perceptron

19

- O sinal de erro produzido na saída do neurônio j é dado por $e_j(n) = d_j(n) - y_j(n)$, em que $d_j(n)$ é o j–ésimo elemento do vetor de respostas desejadas $\mathbf{d}(n)$

- O erro instantâneo do neurônio j é dado por

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

19

Multilayer Perceptron

20

- Somando os erros de todos os neurônios da camada de saída, o erro total de toda a rede é dado por

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

- C é o conjunto de todos os neurônios de saída. Em um conjunto de treinamento com N exemplos, o erro médio sobre todos os exemplos (risco empírico) é dado por

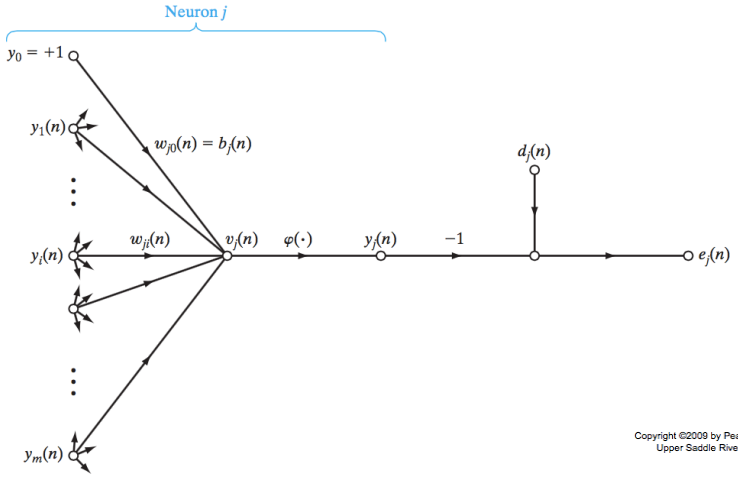
$$\varepsilon_{av}(N) = \frac{1}{N} \sum_{n=1}^{N} \varepsilon(n) = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j \in C} e_j^2(n)$$

20

O Algoritmo Back-propagation

21

- Neurônio j sendo alimentado por um conjunto de sinais



21

O Algoritmo Back-propagation

22

- O potencial de ativação $v_j(n)$ produzido na entrada da função de ativação associada é dado por

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n)$$

  - m : número total de entradas (excluindo o bias)
  - w_{j0} : peso aplicado a entrada fixa y_0=+1 (bias)

22

O Algoritmo Back-propagation

23

- O sinal $y_j(n)$ na saída do neurônio j na iteração n é:

$$y_j(n) = \varphi_j(v_j(n))$$

- O algoritmo aplica um correção $\Delta w_{ji}(n)$ no peso sináptico $w_{ji}(n)$, proporcional a derivada parcial:

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

23

O Algoritmo Back-propagation

24

- A derivada parcial $\partial \varepsilon(n) / \partial w_{ji}(n)$ determina a direção da busca por w_{ji} no espaço de pesos

- Diferenciando a equação abaixo em ambos os lados com relação a $e_j(n)$:

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

24

O Algoritmo Back-propagation

25

- A derivada parcial $\partial \varepsilon(n) / \partial w_{ji}(n)$ determina a direção da busca por w_{ji} no espaço de pesos

- Diferenciando a equação abaixo em ambos os lados com relação a $y_j(n)$:

$$e_j(n) = d_j(n) - y_j(n) \rightarrow \frac{\partial e_j(n)}{\partial y_j(n)} = -1$$

25

O Algoritmo Back-propagation

26

- A derivada parcial $\partial \varepsilon(n) / \partial w_{ji}(n)$ determina a direção da busca por w_{ji} no espaço de pesos

- Diferenciando a equação abaixo em ambos os lados com relação a $v_j(n)$:

$$y_j(n) = \varphi_j(v_j(n)) \rightarrow \frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n))$$

26

O Algoritmo Back-propagation

27

- A derivada parcial $\partial\varepsilon(n)/\partial w_{ji}(n)$ determina a direção da busca por w_{ji} no espaço de pesos

- Diferenciando a equação abaixo em ambos os lados com relação a $w_{ji}(n)$:

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \rightarrow \frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n)$$

27

O Algoritmo Back-propagation

28

- A derivada parcial $\partial\varepsilon(n)/\partial w_{ji}(n)$ determina a direção da busca por w_{ji} no espaço de pesos

- Substituindo as equações obtidas na equação da regra da cadeia, obtemos

$$\frac{\partial\varepsilon(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'(v_j(n))y_i(n)$$

28

O Algoritmo Back-propagation

29

- A correção $\Delta w_{ji}(n)$ aplicada a $w_{ji}(n)$ é definida pela regra delta:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)}$$

  - η : taxa de aprendizado do algoritmo
  - O sinal negativo refere-se ao gradiente descendente no espaço de pesos
- Fazendo uma substituição na equação acima:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$$

29

O Algoritmo Back-propagation

30

- O gradiente local $\delta_j(n)$ é definido por

$$\begin{aligned}\delta_j(n) &= -\frac{\partial \varepsilon(n)}{\partial v_j(n)} \\ &= -\frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \\ &= e_j(n) \varphi_j'(v_j(n))\end{aligned}$$

- O gradiente local define a mudança necessária nos pesos. Ele é dados pelo produto do erro e da derivada da função de ativação do neurônio

30

O Algoritmo Back-propagation

31

- O sinal de erro do neurônio de saída é o fator chave no cálculo do ajuste dos pesos
- Assim, dois casos para o cálculo do erro podem ser identificados
  - O neurônio está localizado na última camada (saída)
  - O neurônio está localizado em uma camada escondida

31

O Algoritmo Back-propagation

32

- Neurônio j é um neurônio de saída
  - Nesse caso, o neurônio está diretamente associado com a saída desejada. Assim, calcula-se o erro diretamente:

$$e_j(n) = d_j(n) - y_j(n)$$

  - Tendo calculado o erro, o gradiente local é calculado de maneira direta:

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n))$$

32

O Algoritmo Back-propagation

33

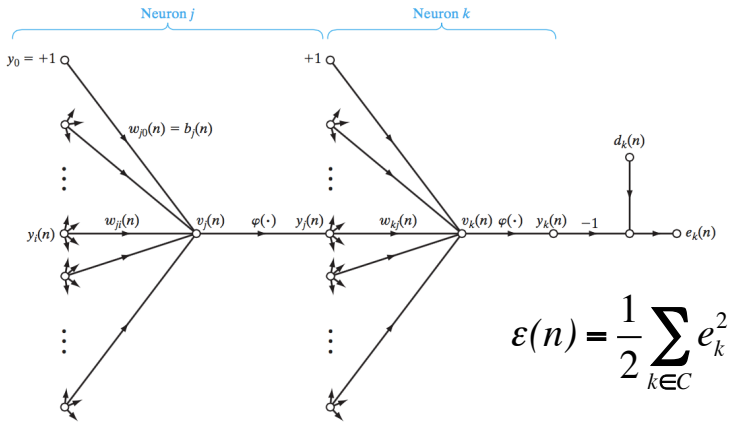
- Neurônio j é um neurônio escondido
  - Nesse caso, não há saída desejada específica associada ao neurônio.
  - O sinal de erro deve ser calculado recursivamente, em termos dos sinais de erro de todos os neurônios os quais o neurônio j está diretamente conectado
  - Nesse ponto o desenvolvimento do back-propagation torna-se mais complicado (mas nem tanto)

33

O Algoritmo Back-propagation

34

- Neurônio j é um neurônio escondido



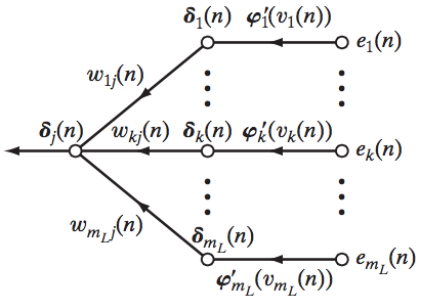
$$\varepsilon(n) = \frac{1}{2}\sum_{k \in C} e_k^2(n)$$

34

O Algoritmo Back-propagation

35

- Neurônio j é um neurônio escondido
  - Representação gráfica de $\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n)$



35

O Algoritmo Back-propagation

36

- **Resumindo:** a correção aplicada nos pesos conectando um neurônio i com um neurônio j é dada pela regra delta

$$\begin{pmatrix} \text{Correção} \\ \text{pesos} \\ \Delta w_{ji}(n) \end{pmatrix} = \begin{pmatrix} \text{Taxa} \\ \text{aprendizado} \\ \eta \end{pmatrix} \times \begin{pmatrix} \text{Gradiente} \\ \text{local} \\ \partial_j(n) \end{pmatrix} \times \begin{pmatrix} \text{Sinal entrada} \\ \text{neurônio } j \\ y_i(n) \end{pmatrix}$$

- **Saída:** $\delta_j(n)$ é igual ao produto da derivada $\varphi_j'(v_j(n))$ e do sinal de erro $e_j(n)$, ambos associados ao neurônio j
- **Escondido:** $\delta_j(n)$ é igual ao produto da derivada associada $\varphi_j'(v_j(n))$ e da soma ponderada dos δs calculados para os neurônios da próxima camada, ou da camada de saída, conectados ao neurônio j

36

Funções de Ativação

37

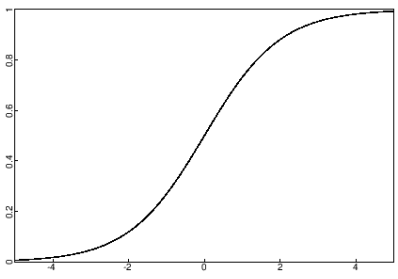
- Para calcular o δ para cada neurônio, precisamos conhecer a derivada da função de ativação $\varphi(\cdot)$ associada ao neurônio
- Para existir a derivada, $\varphi(\cdot)$ deve ser contínua
- Assim, ser **diferenciável** é o único requisito para a função de ativação
- Função contínua diferenciável comumente utilizada: **sigmoidal**

37

Funções de Ativação

38

- **Função logística:** $\varphi_j(v_j(n)) = \dfrac{1}{1+exp(-av_j(n))}, \quad a > 0$
  - Amplitude do sinal de saída: $0 \leq y_j \leq 1$



38

Funções de Ativação

39

- **Função logística:** $\varphi_j(v_j(n)) = \dfrac{1}{1+exp(-av_j(n))}, \quad a > 0$
  - Amplitude do sinal de saída: $0 \leq y_j \leq 1$
  - A derivada da função com relação a $v_j(n)$ fornece:

$$\varphi'_j(v_j(n)) = \frac{a\ exp(-av_j(n))}{\left[1+exp(-av_j(n))\right]^2}$$

  - Com $y_j(n) = \varphi_j(v_j(n))$, podemos reescrever a derivada:

$$\varphi'_j(v_j(n)) = ay_j(n)\left[1-y_j(n)\right]$$

39

Funções de Ativação

40

$$\varphi'_j(v_j(n)) = ay_j(n)\left[1-y_j(n)\right]$$

- Para um neurônio j da camada de saída, $y_j(n) = o_j(n)$. O gradiente local do neurônio j é dado por:

$$\begin{aligned}\delta_j(n) &= e_j(n)\varphi'_j(v_j(n)) \\ &= a\left[d_j(n)-o_j(n)\right]o_j(n)\left[1-o_j(n)\right]\end{aligned}$$

40

Funções de Ativação

41

$$\varphi_j'(v_j(n)) = a y_j(n)\left[1 - y_j(n)\right]$$

- Para um neurônio j de uma camada escondida, o gradiente local do neurônio j é dado por:

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n)$$

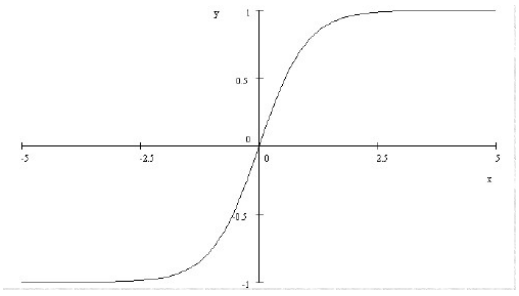
$$= a y_j(n)\left[1 - y_j(n)\right] \sum_k \delta_k(n) w_{kj}(n)$$

41

Funções de Ativação

42

- **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: $-a \leq y_j \leq a$



42

Funções de Ativação

43

- **Função tangente hiperbólica:** $\varphi_j(v_j(n)) = a\tanh(bv_j(n))$
  - a e b são constantes positivas
  - Amplitude do sinal de saída: $-1 \le y_j \le 1$
  - A derivada da função com relação a $v_j(n)$ fornece:

$$\begin{aligned}\varphi_j'(v_j(n)) &= ab\ sech^2(bv_j(n)) \\ &= ab\left(1 - tanh^2(bv_j(n))\right) \\ &= \frac{b}{a}\left[a - y_j(n)\right]\left[a + y_j(n)\right]\end{aligned}$$

43

Funções de Ativação

44

$$\varphi_j'(v_j(n)) = \frac{b}{a}\left[a - y_j(n)\right]\left[a + y_j(n)\right]$$

- Para um neurônio j da camada de saída, o gradiente local do neurônio j é dado por:

$$\begin{aligned}\delta_j(n) &= e_j(n)\varphi_j'(v_j(n)) \\ &= \frac{b}{a}\left[d_j(n) - o_j(n)\right]\left[a - o_j(n)\right]\left[a + o_j(n)\right]\end{aligned}$$

44

Funções de Ativação

45

$$\varphi_j'(v_j(n)) = \frac{b}{a}\left[a - y_j(n)\right]\left[a + y_j(n)\right]$$

- Para um neurônio j de uma camada escondida, o gradiente local do neurônio j é dado por:

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n)$$

$$= \frac{b}{a}\left[a - y_j(n)\right]\left[a + y_j(n)\right] \sum_k \delta_k(n) w_{kj}(n)$$

45

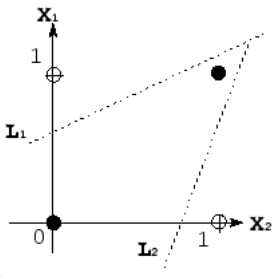
O Problema XOR

46

- O Perceptron de Rosenblatt não pode classificar padrões não linearmente separáveis

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$Y = X_1 \oplus X_2$



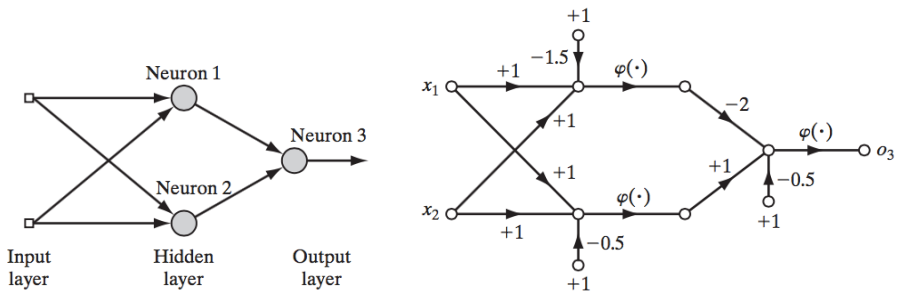
  - Um único Perceptron consegue traçar apenas um hiperplano

46

O Problema XOR

47

- Pode-se resolver o problema usando uma camada escondida com dois neurônios



$$w_{11} = w_{12} = +1 \qquad b_1 = -\frac{3}{2}$$

$$w_{21} = w_{22} = +1 \qquad b_2 = -\frac{1}{2}$$

$$w_{31} = -2 \qquad w_{32} = +1 \qquad b_3 = -\frac{1}{2}$$

47

O Problema XOR

48

- Pode-se resolver o problema usando uma camada escondida com dois neurônios

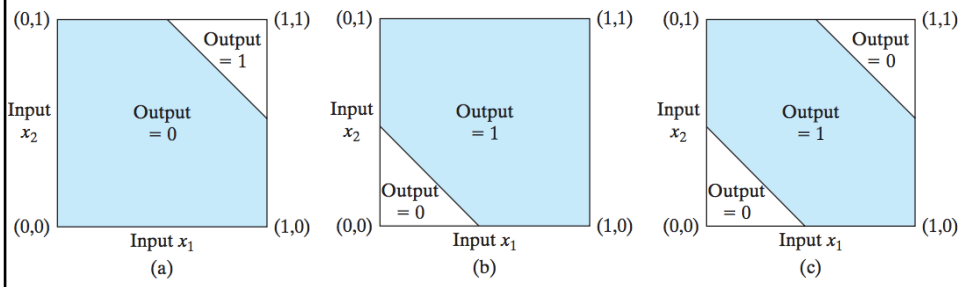


FIGURE 4.9 (a) Decision boundary constructed by hidden neuron 1 of the network in Fig. 4.8. (b) Decision boundary constructed by hidden neuron 2 of the network. (c) Decision boundaries constructed by the complete network.

48