Redes Neurais de Uma Camada

Prof. Dr. Ajalmar Rocha

Disciplina: Inteligência Computacional Aplicada (ICA)

Programa de Pós-Graduação em Eng. de Telecomunicações (PPGET)
Instituto Federal do Ceará (IFCE)

Agosto/2013

Outras funções para Cálculo da Saída do Neurônio

- Há outras funções relevantes - além da degrau ($y = sinal(u)$) e linear - para cálculo da variável de saída do neurônio .
- Duas das mais importantes são do tipo **Sigmóide**.
  (i) Logística

$$y = \frac{1}{1 + exp(-u)} \tag{1}$$

  (ii) Tangente Hiperbólica

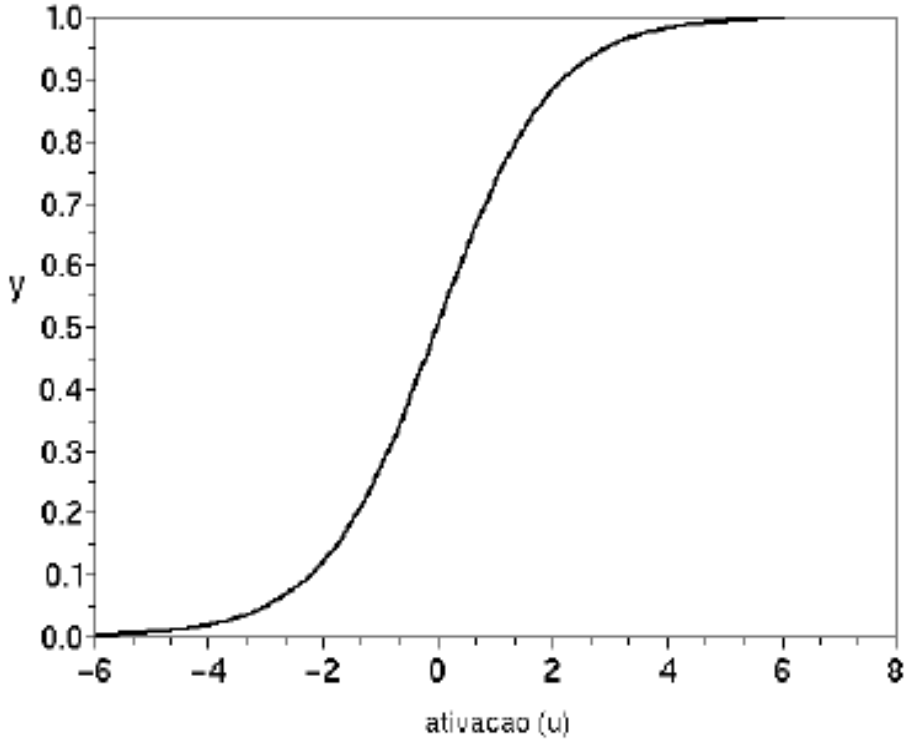
$$y = \frac{1 - exp(-u)}{1 + exp(-u)} \tag{2}$$

- Estas funções tem características especias importantes: (i) são diferenciáveis e (ii) possuem trechos centrais praticamente linear.

Função Sigmóide Logística

$$y_i(t) = \frac{1}{1+\exp(-u_i(t))}$$

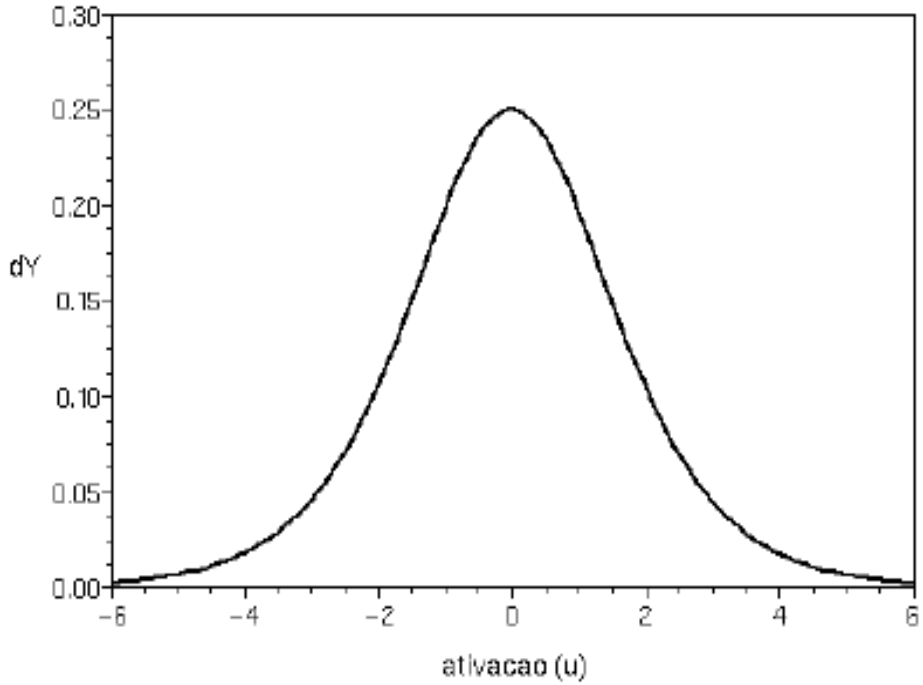
$$y_i(t) \in (0,1)$$



Derivada da Função Sigmóide Logística

$$y_i'(t) = \frac{d\, y_i(t)}{d\, u_i(t)}$$

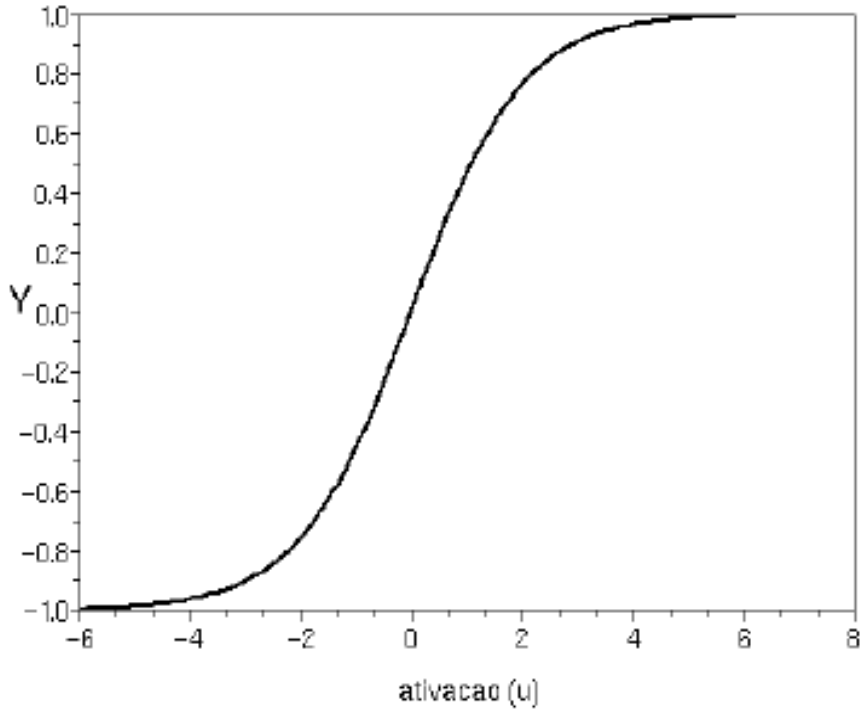
$$y_i'(t) = y_i(t)[1 - y_i(t)]$$



Função Sigmóide Tangente Hiperbólica

$$y_i(t) = \frac{1 - \exp(-u_i(t))}{1 + \exp(-u_i(t))}$$

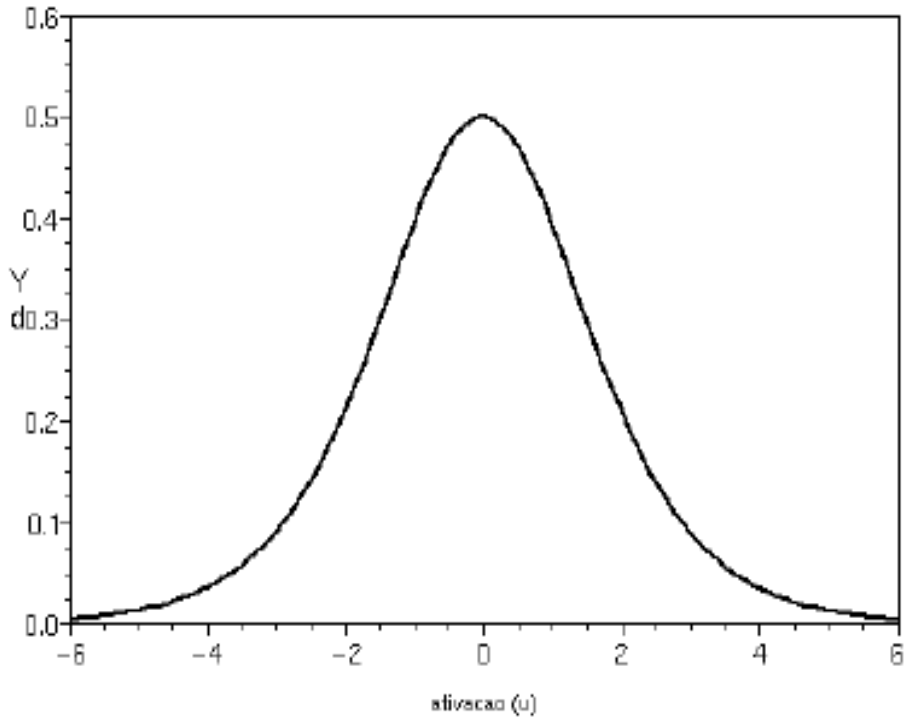
$$y_i(t) \in (-1, 1)$$



Derivada da Função Sigmóide Tangente Hiperbólica

$$y_i'(t) = \frac{d\, y_i(t)}{d\, u_i(t)}$$

$$y_i'(t) = 0{,}5[1 - y_i^2(t)]$$



Revisitando o problema de otimização

Minimização da Função Custo

- Considerando o erro quadrático como critério de desempenho, a partir de um valor inicial para o vetor de pesos, objetiva-se que o vetor de pesos se aproxime gradativamente do mínimo global.
- No entanto, busca-se atingir o mínimo da superfície J resultante, a saber:

$$J = \frac{1}{2}e^2$$

$$J = \frac{1}{2}(d_i - y(u))^2 \tag{3}$$

Minimização da Função Custo

- Com o intuito de minimizar a função custo

$$J = \frac{1}{2}(d_i - y(u))^2$$

é necessário que seja obtido a direção do ajuste a ser aplicado no vetor de pesos para aproximar a solução do mínimo de J.

- Para tal, usa-se o gradiente da função custo J no ponto $\mathbf{w}(t)$.
- Sabe-se que o gradiente possui a mesma direção da maior variação do erro (aponta na direção de crescimento da função), portanto o ajuste deve ocorrer na direção contrária ao gradiente. Logo, a varação dos pesos pode ser descrita por:

$$\Delta\mathbf{w}(t) \propto -\nabla J. \tag{4}$$

Gradiente da Função Custo

Consideração

Sabe-se que a função custo J depende de e, bem como sabe-se que e depende de y, e ainda sabe-se que y depende de $\mathbf{w}$.

- Cada componente do gradiente da função custo ∇J pode ser obtida com base na regra da cadeia, a saber:

$$\frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial e}\frac{\partial e}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial w_i} \tag{5}$$

- Sabendo ainda que

$$\frac{\partial J}{\partial e} = e, \qquad \frac{\partial e}{\partial y} = -1, \qquad \frac{\partial y}{\partial u} = y' \quad \text{e} \quad \frac{\partial u}{\partial w_i} = x_i,$$

Gradiente da Função Custo

- A i-ésima componente do gradiente da função custo $\frac{\partial J}{\partial w_i}$ considerando que $\frac{\partial J}{\partial e} = e$, $\frac{\partial e}{\partial y} = -1$, $\frac{\partial y}{\partial u} = y'$ e $\frac{\partial u}{\partial w_i} = x_i$ é igual a

$$\frac{\partial J}{\partial w_i} = -x_i y' e \tag{6}$$

em que

$$y' = y(1-y), \quad \textbf{para a função logística}$$
$$y' = \frac{1}{2}(1-y^2), \quad \textbf{para a função tangente hiperbólica}$$

Regra de Aprendizagem

- A regra de aprendizagem pode então ser obtida como segue.

$$\Delta \mathbf{w} = \mathbf{w}(t+1) - \mathbf{w}(t) \tag{7}$$
$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta \mathbf{w}$$

- Considerando que $\Delta \mathbf{w}(t) \propto -\nabla J$, pode se escrever

$$\Delta \mathbf{w} = \mathbf{w}(t+1) - \mathbf{w}(t) \tag{8}$$
$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta \mathbf{w}$$
$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla J$$

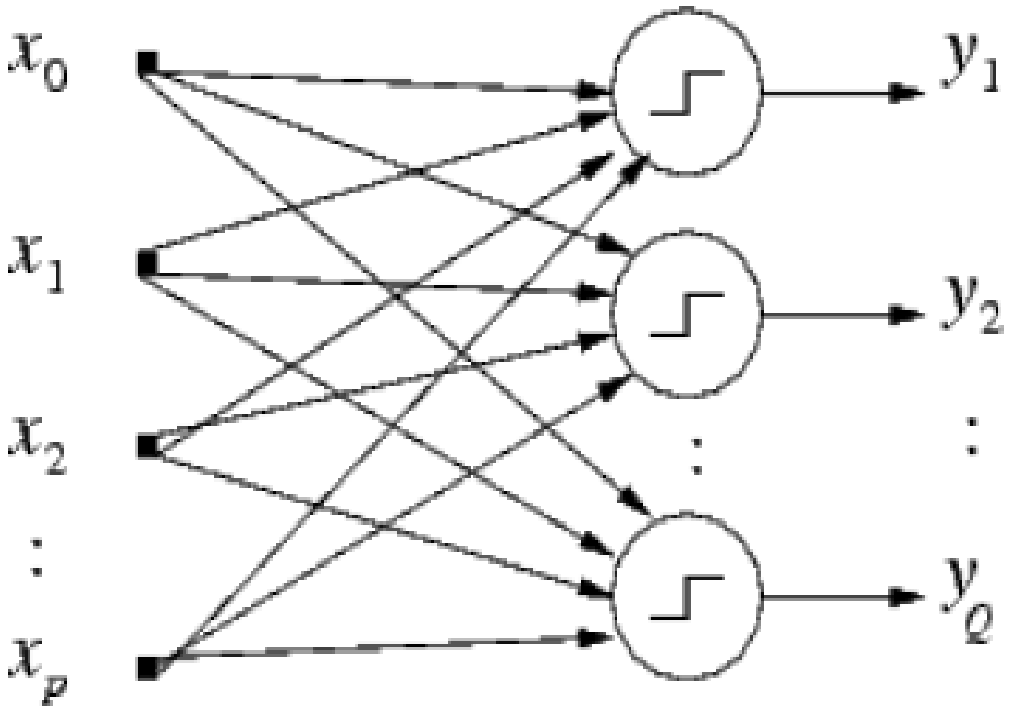
- Além disto, sabendo que $\frac{\partial J}{\partial \mathbf{w}} = -\mathbf{x}e$ temos que

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta e(t) y'(t) \mathbf{x}(t) \tag{9}$$

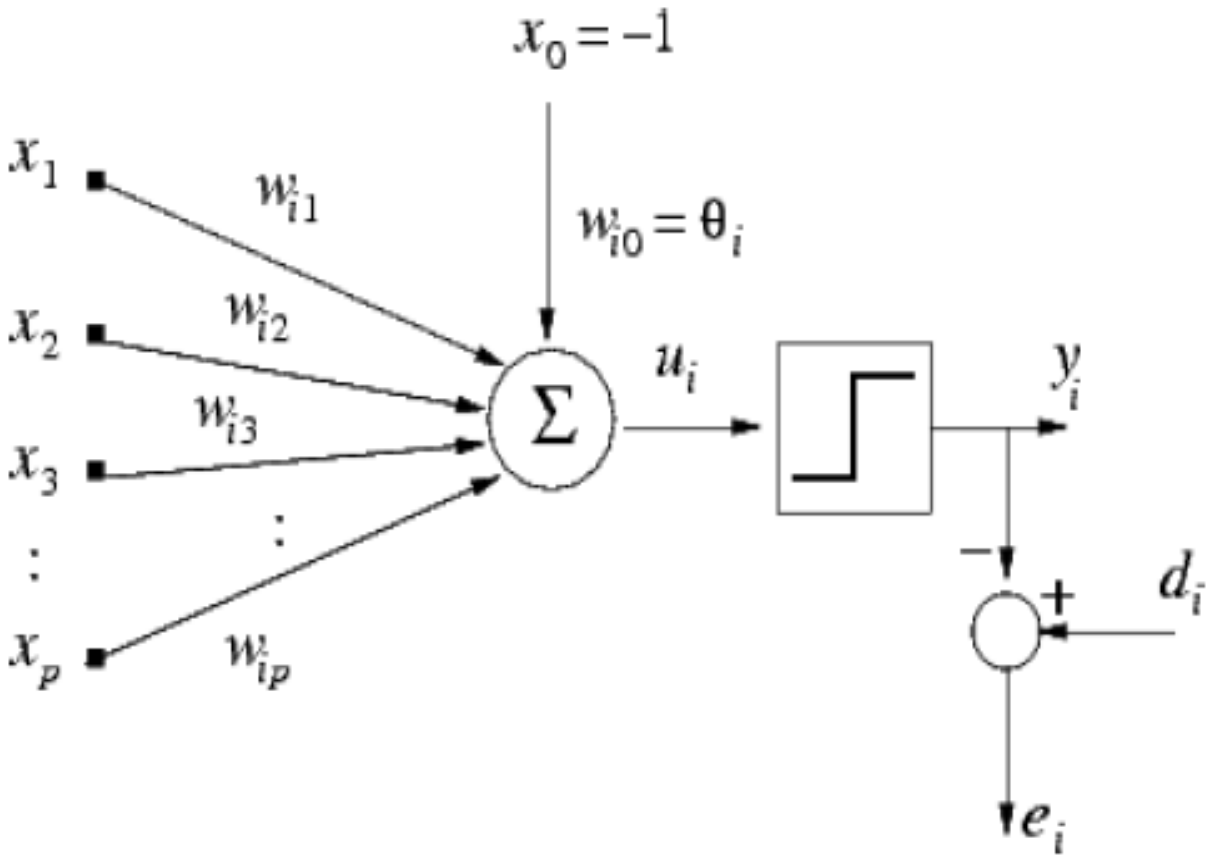
Rede Perceptron

- Com apenas um neurônio podemos classificar apenas $C = 2$ classes, $c_1 = 0, c_2 = 1\}$.
- Como visto, o Perceptron é bastante adequado para um problema binário linearmente separável.
- Problemas multiclasses com $C > 2$ exige a utilização de mais de um neurônio em paralelo.
- A utilização deste neurônios em paralelo recebendo simultaneamente os mesmos valores de entrada caracterizam uma camada de uma rede.

Rede Perceptron



Perceptron para o i-ésimo Neurônio



Codificação 1-out-of-C

- Nesse método o número de neurônios é igual ao número de classes C.
- Exemplo: problema com duas classes 2 neurônios, com 3 classes 3 neurônios, etc.

$$\text{Classe 1: } \mathbf{d} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \text{Classe 2: } \mathbf{d} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \text{Classe 3: } \mathbf{d} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Variável de Ativação e Saída para Rede Perceptron

- A i-ésima variável de ativação pode, portanto, ser calculada por

$$u_i = x_0 w_{i0} + x_1 w_{i1} + x_2 w_{i2}. \tag{10}$$

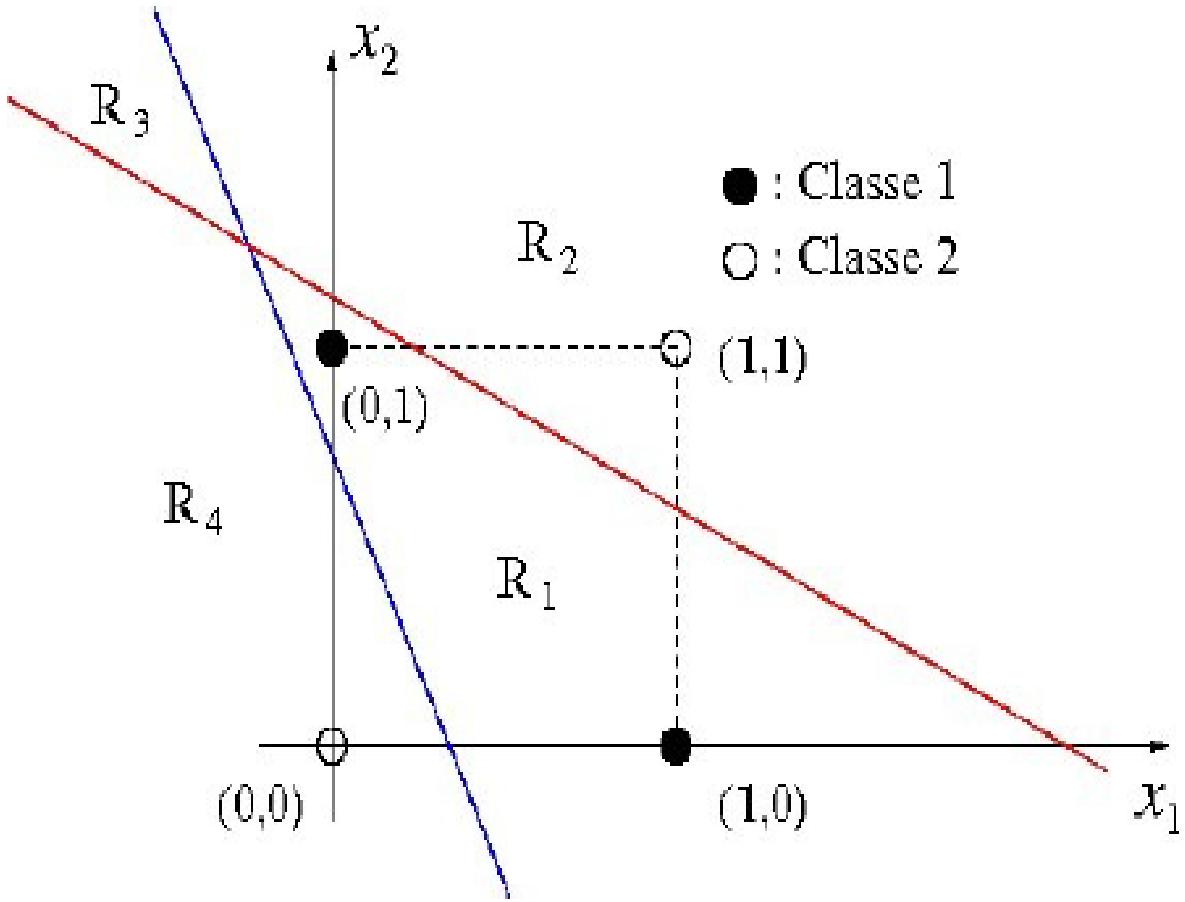
- Enquanto a i-ésima variável de saída pode ser calculada por

$$y_i = \phi(u_i) \tag{11}$$

- Assim, a saída para a camada é

$$[y_1 \dots y_2 \dots y_C] \tag{12}$$

Perceptron para o *i*-ésimo Neurônio



Obrigado