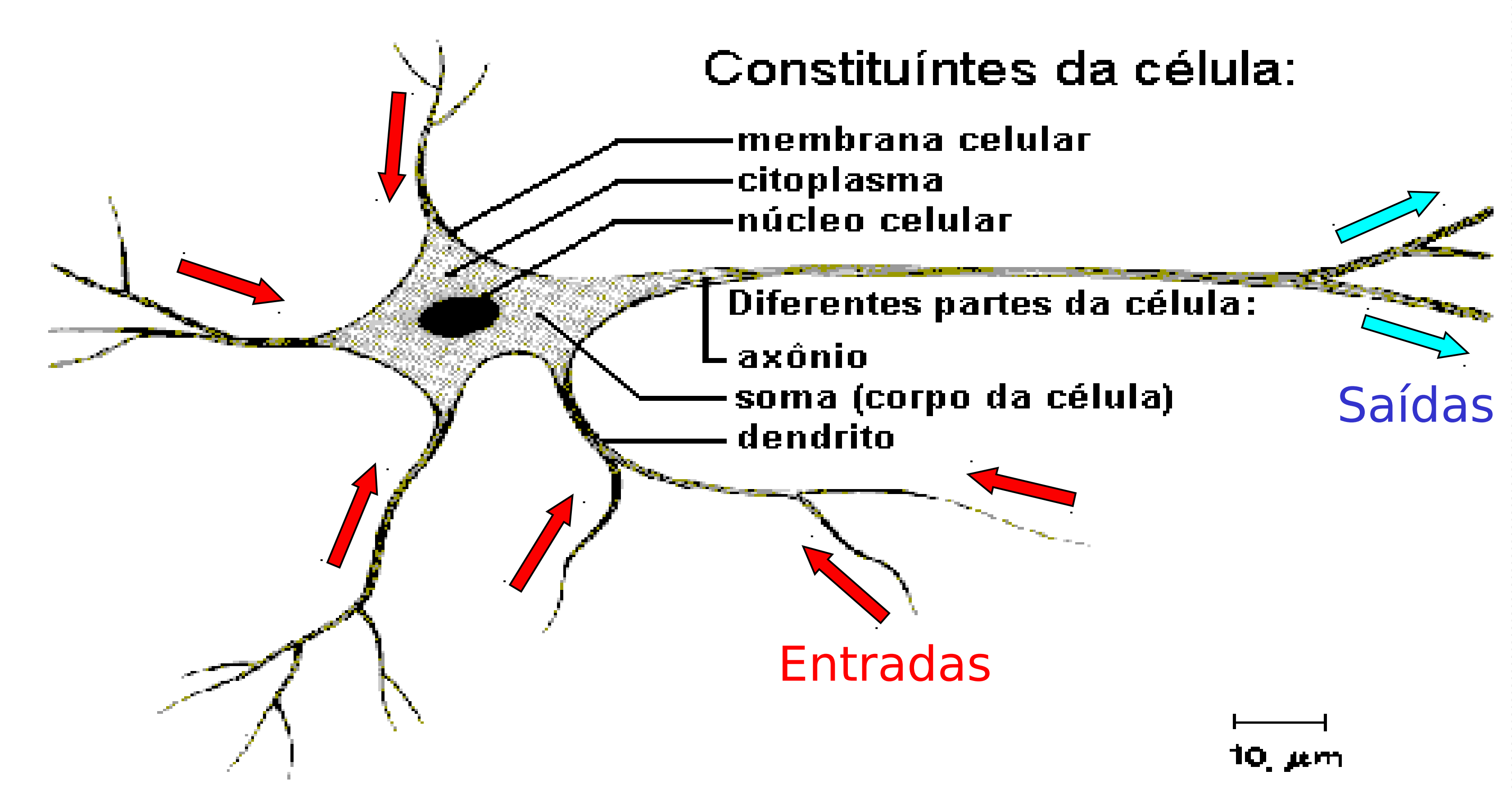
Redes Neurais Artificiais

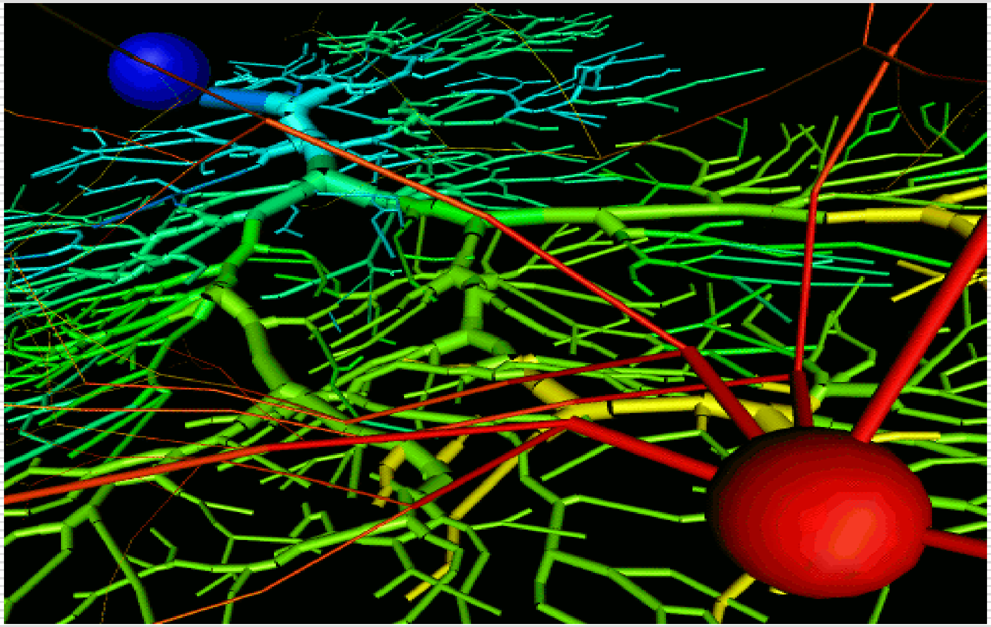
Sistemas Inteligentes – Prof. Flávio Varejão
Programa de Pós-Graduação em Informática
Universidade Federal do Espírito Santo

Modelo Biológico - Neurônio



Modelo Biológico – Rede Neural

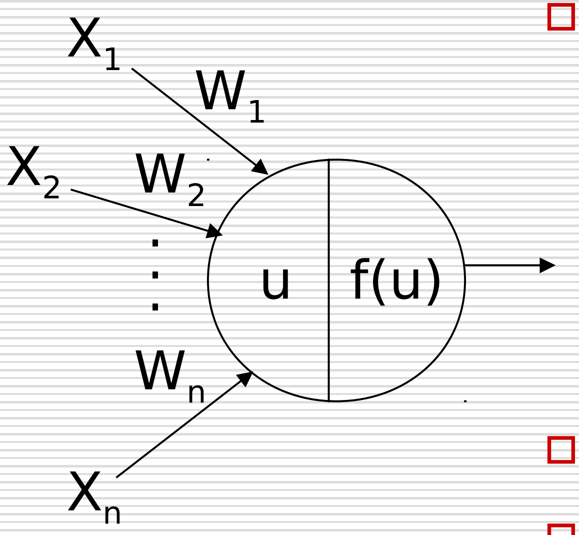
- Entradas
  - Recebidas pelos dendritos
  - Sinapses vindas de outros neurônios



- Processamento
  - Núcleo celular
- Saídas
  - Propagadas pelo axônio
  - Sinapses para outros neurônios

Cérebro humano
10^{11} Neurônios
10^{14} Sinapses

Neurônio MCP



- Entrada: $X = [X_1, X_2, ..., X_n]^T$
- Pesos: $W = [W_1, W_2, ..., W_n]^T$
- Saída linear (net): $u = W^T . X$

  $u = \sum_i W_i X_i$

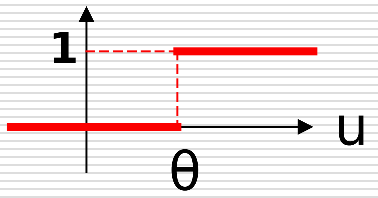
$y = f(u)$

- Saída de ativação: $y = f(u)$
- Função de ativação: $f(\bullet)$

Funções de Ativação

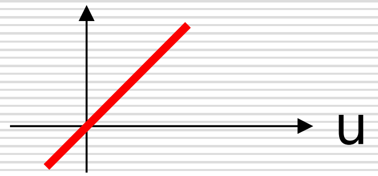
- Função de limiar (*threshold*)

$$f(u) = \begin{cases} 0 \; , & \text{se } u < \theta \\ 1 \; , & \text{se } u \geq \theta \end{cases}$$



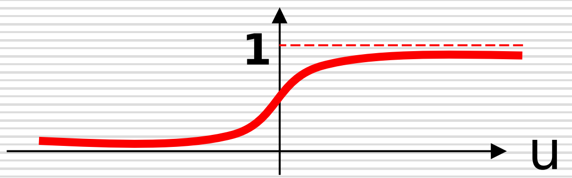
- Função identidade

$f(u) = u$



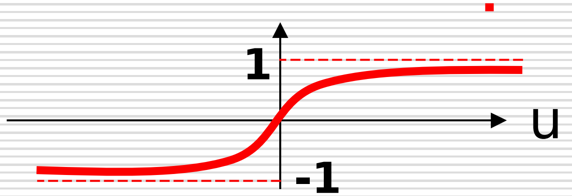
- Função sigmóide
  - Logística

    $f(u) = (1 + e^{-u})^{-1}$



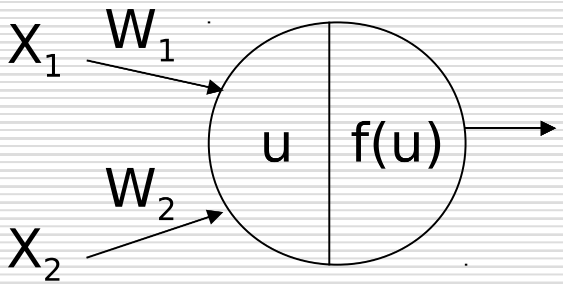
  - Tangente hiperbólica

    $f(u) = \tanh(u)$



Parâmetros Livres

- Um neurônio artificial pode mapear funções diferentes, dependendo de seus parâmetros livres: pesos e limiar (*bias*)
- Consideremos, por exemplo, um neurônio MCP com duas entradas X_1 e X_2 e função de limiar:



$$f(u) = \begin{cases} 0, & \text{se } u < \theta \\ 1, & \text{se } u \geq \theta \end{cases}$$

Parâmetros Livres - Exemplos

- Exemplo 1
  - $W_1 = 1,0$
  - $W_2 = 1,0$
  - $\theta \ = 0,5$

X_1	X_2	u	f(u)
0	0	0	0
0	1	1	1
1	0	1	1
1	1	2	1

Função “OU” lógico

- Exemplo 2
  - $W_1 = 1,0$
  - $W_2 = 1,0$
  - $\theta \ = 1,5$

X_1	X_2	u	f(u)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	2	1

Função “E” lógico

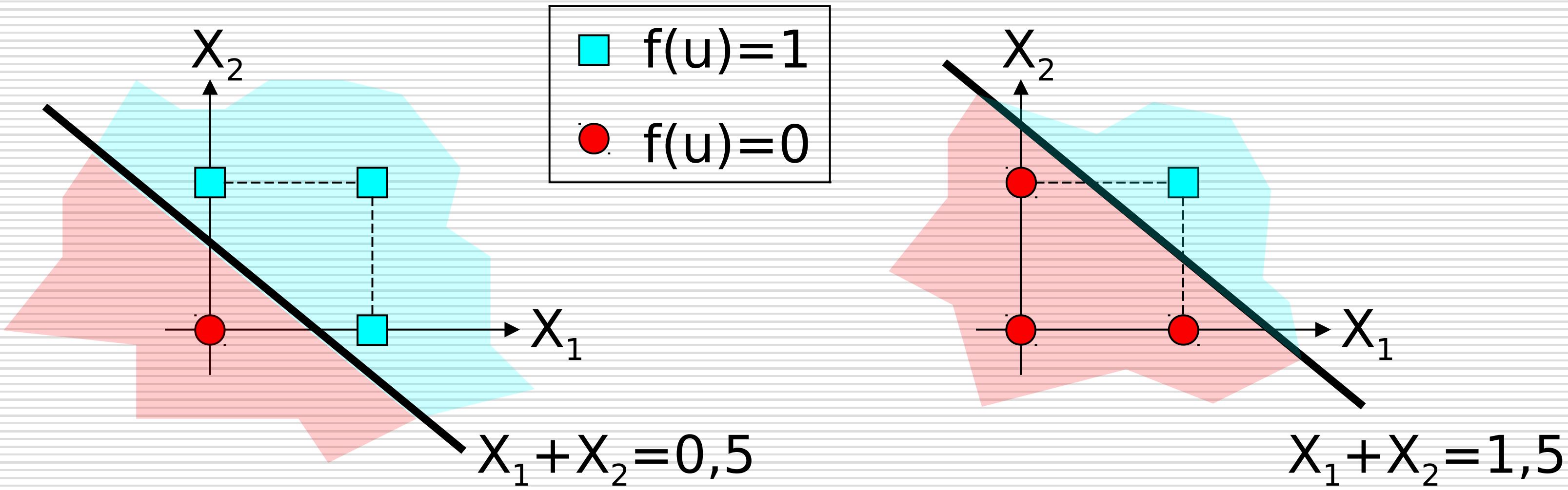
Visualização Geométrica

- Nos exemplos, a equação de decisão por meio da qual o neurônio ativa ou não a sua saída é:

$$\boxed{W_1 X_1 + W_2 X_2 = \theta} \quad \text{ou} \quad \boxed{W_0 + W_1 X_1 + W_2 X_2 = 0}$$

Exemplo 1

Exemplo 2

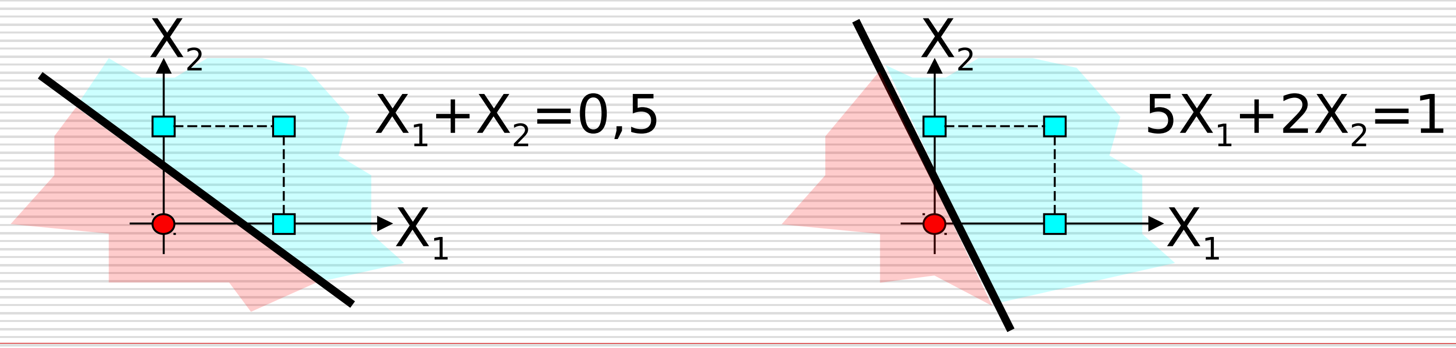


Visualização Geométrica

- Num espaço de duas variáveis, X_1 e X_2, o neurônio MCP estabelece uma reta que divide esse espaço em duas regiões
- Alterando-se os parâmetros W_0, W_1 e W_2, teremos retas com diferentes ângulos e posições
- Num espaço de três dimensões, os parâmetros W_0, W_1, W_2 e W_3 definem um plano
- Num espaço de *n* dimensões, W_0, W_1, W_2 ... W_n definem um hiperplano de dimensão (n–1), que divide o hiperespaço em duas regiões

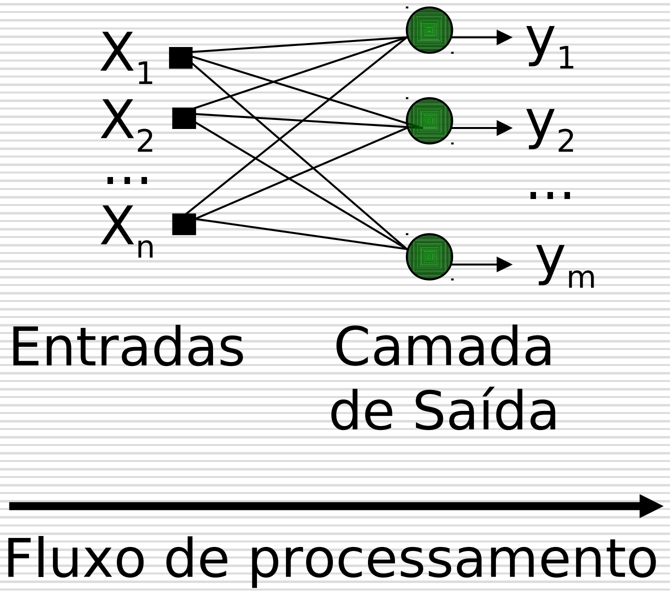
Adaptação dos Parâmetros Livres

- A representação do conhecimento em uma RNA está na topologia da rede e no valor dos parâmetros livres
- O aprendizado de uma RNA consiste em encontrar o conjunto de pesos W (incluindo o *bias* W_0) que melhor se adapta ao problema (solução ótima)
- Para o problema da função "OU" lógico, existem infinitas soluções para W_0, W_1 e W_2



Arquiteturas de Redes Neurais

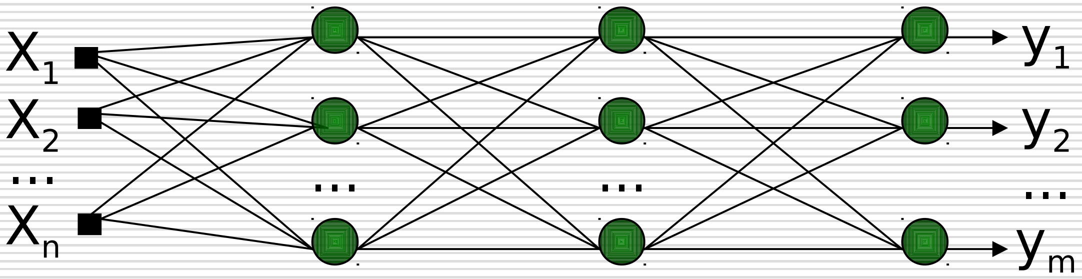
- Redes *feed-forward* com uma camada
  - Ex: Perceptron Simples, Máquina Linear



- Resolvem problemas linearmente separáveis
- Não resolvem o problema da função “XOR”

Arquiteturas de Redes Neurais

- <u>Redes *feed-forward* com múltiplas camadas</u>
  - Ex: Perceptron de Múltiplas Camadas



Entradas

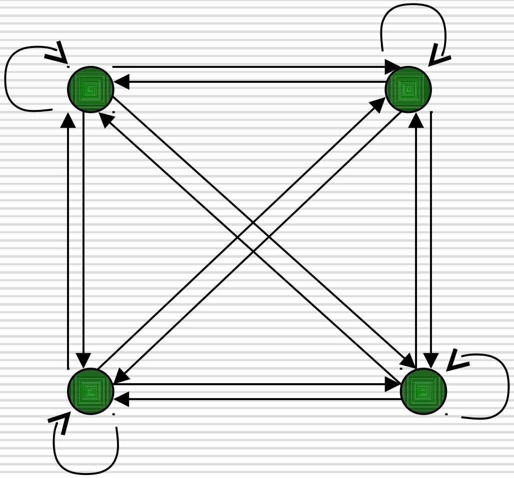
Camadas Intermediárias

Camada de Saída

Fluxo de processamento

Arquiteturas de Redes Neurais

- Redes recorrentes (*feed-back*)
  - Ex: Rede de Hopfield



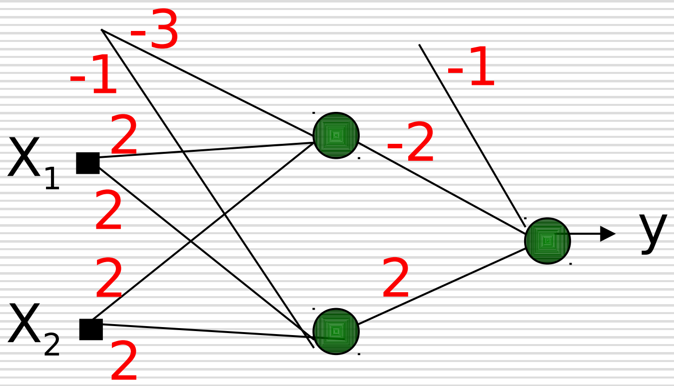
- Há laços de realimentação
- Comportamento dinâmico
- X(t+1) = f(X(t))

Perceptron de Múltiplas Camadas

- Aproximações universais de funções multivariáveis contínuas
- O número de entradas e saídas depende da dimensionalidade dos dados
- O número de neurônios nas camadas intermediárias depende da complexidade do problema
- Quanto maior o número de neurônios, mais complexas são as funções mapeadas com a RNA, porém maior é o tempo de treinamento

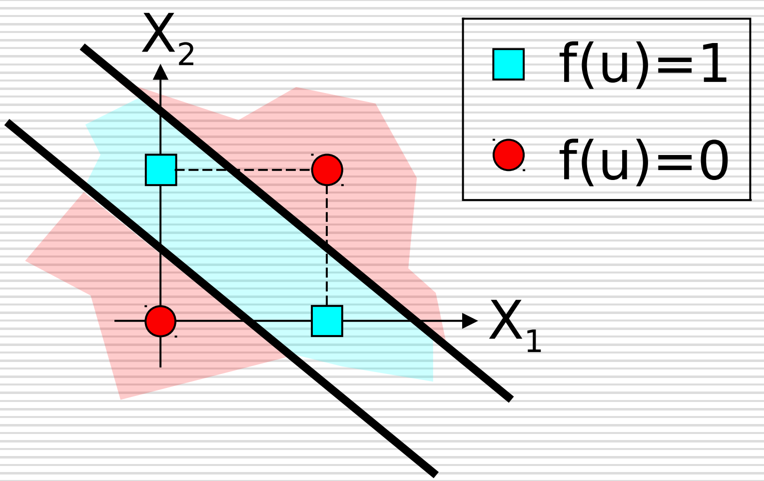
Perceptron de Múltiplas Camadas

- Exemplo: Problema XOR



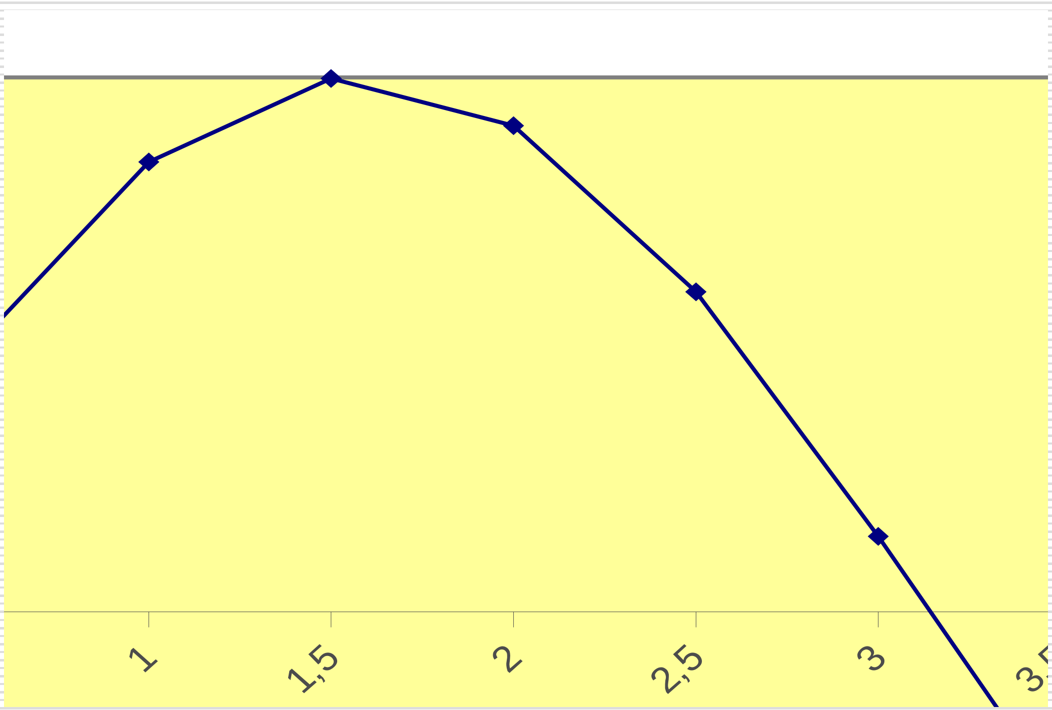
$$f(u) = \begin{cases} 0 \text{ , se } u < 0 \\ 1 \text{ , se } u \geq 0 \end{cases}$$

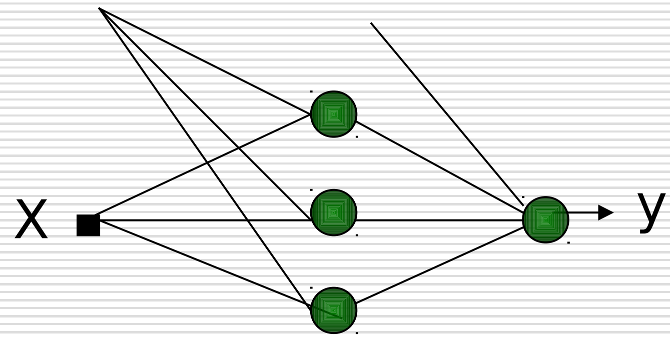
X_1	X_2	u_1	u_2	f_1	f_2	u	f(u)
0	0	-3	-1	0	0	-1	0
0	1	-1	1	0	1	1	1
1	0	-1	1	0	1	1	1
1	1	1	3	1	1	-1	0



Aplicações

- Aproximação de Funções (Regressão)
  - Problema de interpolação





- Camada escondida
  - f(u) = tanh(u)
- Camada de saída
  - f(u) = u

Aprendizado

- RNAs caracterizam-se pelo aprendizado por meio de exemplos
- Para um determinado conjunto de dados, o algoritmo de aprendizado deve adaptar os parâmetros livres da rede
- Em um número finito de iterações do algoritmo deve haver convergência para uma solução
- Critério de convergência
  - Minimização de uma função-objetivo, como por exemplo o erro de saída da rede

Paradigmas de Aprendizado

- Aprendizado Supervisionado
  - Caracteriza-se pela existência de um "professor" que tem a função de monitorar as respostas
  - O conjunto de treinamento é formado por pares de entrada e saída (x, y_d)
  - O ajuste de pesos é feito de maneira que a resposta y da rede se aproxime da desejada y_d dentro de limites de tolerância estabelecidos
  - Utilizado em problemas de aproximação de funções, classificação de dados e modelagem de sistemas

Paradigmas de Aprendizado

- <u>Aprendizado Não Supervisionado</u>
  - Não há um “professor” externo
  - A única informação disponível é o conjunto de entradas x
  - O objetivo é a descoberta de estruturas entre os dados (clusters)
  - Utilizado em problemas de categorização
  - Exemplo: Mapa Auto-organizável de Kohonen

Paradigmas de Aprendizado

- <u>Aprendizado por Reforço</u>
  - Pode ser considerado um paradigma intermediário entre o supervisionado e o não supervisionado
  - O conjunto de treinamento é formado apenas pelas entradas, mas há um “crítico” externo que retorna um sinal de reforço ou penalidade associado à última ação da rede
  - O algoritmo visa a maximização do reforço e conseqüente melhora do desempenho da rede
  - Exemplo: controle do pêndulo invertido

Algoritmo *Backpropagation*

- Rumelhart et al, 1986
- Baseia-se na retropropagação dos erros para realizar ajustes nos pesos das camadas intermediárias
- Ajuste do vetor de pesos na n-ésima iteração:

$$w(n+1) = w(n) + \Delta w(n)$$

- Método da descida de gradiente

$$\Delta w(n) = -\eta \nabla E(w(n))$$

- Regra de Delta Generalizada

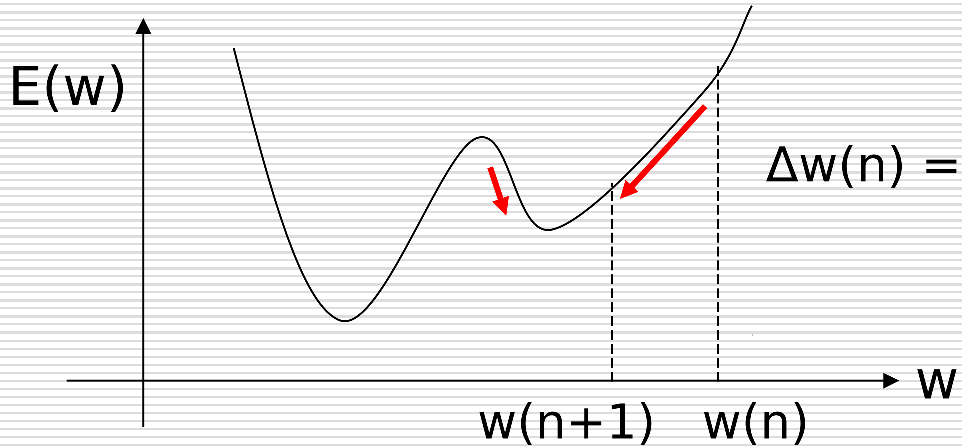
$$\Delta w_{ji}(n) = -\eta \, \partial E / \partial w_{ji}(n)$$

Algoritmo *Backpropagation*

- E : Erro quadrático médio das respostas da RNA

  $E(w(n)) = \frac{1}{2} \sum_j e_j(w(n))^2 = \frac{1}{2} \sum_j (y_j^d – f_j(u_j(n)))^2$

- η : Taxa de aprendizado
  - Parâmetro do algoritmo de treinamento
  - Quanto maior a taxa, maior o valor de Δw
  - Pode caminhar para um mínimo local, em vez do global



Algoritmo *Backpropagation*

- Fase *Feed-Forward*:

1. Na primeira fase do treinamento, as entradas X_i são propagadas para frente, calculando $u_h=\sum_i w_{hi} x_i$ e $f_h(u_h)$ para cada neurônio *h* nas camadas escondidas
2. As saídas das camadas escondidas $f_h(u_h)$ são propagadas para frente, até chegar ao cálculo de $u_j=\sum_h w_{jh} f_h(u_h)$ e $f_j(u_j)$, para cada neurônio *j* na camada de saída
3. Como os valores de saída desejados são conhecidos, calculam-se os erros: $e_j(w)=y_j^d - f_j(u_j)$

Métodos Conexionistas – Resumo

- Vantagens
  - Aquisição automática de conhecimento
  - Manipulação de dados quantitativos, mesmo aproximados ou com ruídos
  - Grande poder de representação de conhecimento
- Desvantagens
  - Dificuldade para se definir a topologia e os parâmetros ideais para cada problema
  - O conhecimento adquirido não fica explicitado numa linguagem compreensível ao ser humano
  - Lentidão do processo de aprendizado