

# KDD E MINERAÇÃO DE DADOS

## Métodos Baseados em Redes Neurais

**Prof. Ronaldo R. Goldschmidt**

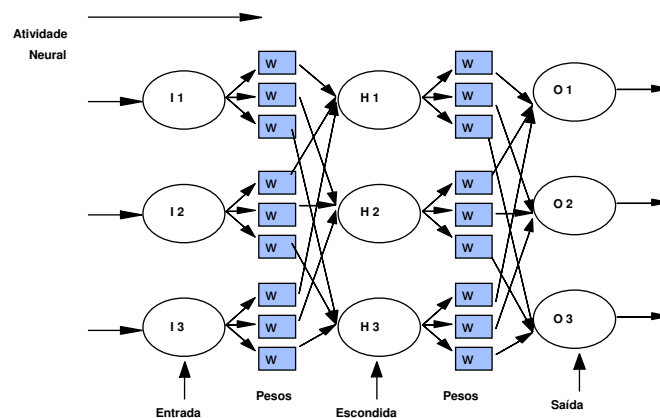
[ronaldo@de9.ime.eb.br](mailto:ronaldo@de9.ime.eb.br)

[rribeiro@univercidade.br](mailto:rribeiro@univercidade.br)

[geocities.yahoo.com.br/ronaldo\\_goldschmidt](http://geocities.yahoo.com.br/ronaldo_goldschmidt)

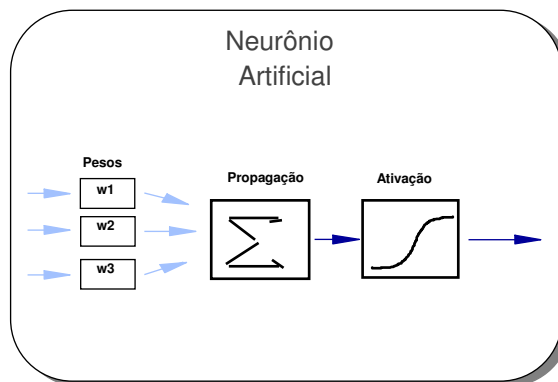
### REDES NEURAIS

#### Estrutura de uma Rede Neural:



## REDES NEURAIIS

### Elementos Básicos de um Neurônio Artificial:



## REDES NEURAIIS

### Elementos Básicos de um Neurônio Artificial:

- **Estado de Ativação**  $\rightarrow S_j$
- **Conexões entre Processadores** - a cada conexão existe um peso sináptico que determina o efeito da entrada sobre o processador  $\rightarrow W_{ij}$
- **Função de Ativação** - determina o novo valor do Estado de Ativação de Ativação do processador  $\rightarrow s_j = F(\text{net}_j)$

## REDES NEURAIS

### Regra de Propagação:

Combina as entradas de um processador com os pesos sinápticos associados às conexões que chegam a tal processador.

$$\text{net}_j = \sum W_{ij} * O_i$$

Onde:

$\text{net}_j$  – Entrada do processador j

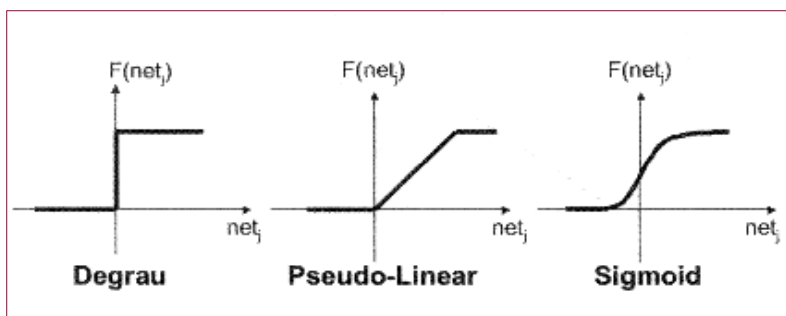
$O_i$  – Saída do processador i

$W_{ij}$  – Peso da conexão entre os neurônios i e j

## REDES NEURAIS

### Funções de Ativação:

É a função que determina o nível de ativação do Neurônio Artificial -  $S_j = F(\text{net}_j)$



## REDES NEURAIS

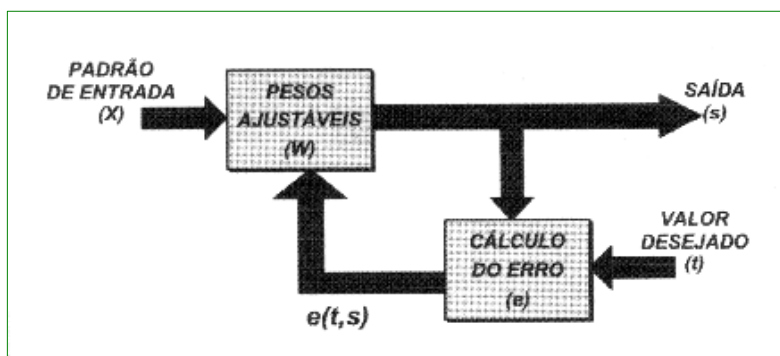
### Modelos de Redes Neurais – Exemplos:

- **Back-Propagation**
- **Hopfield**
- **Kohonen**
- **ART**
- **SVM**

Possuem diferentes propósitos e caracterizações.

## BACK-PROPAGATION

### Princípio Básico:



## BACK-PROPAGATION

Os valores de entrada dos neurônios após a camada de entrada são calculados pela **regra de propagação**, em geral o produto escalar:

$$\text{netj}(n) = \sum w_{ji}(n) y_i(n)$$

onde,

$w_{ji}(n)$  corresponde aos pesos das conexões que chegam ao neurônio  $j$  na  $n$ -ésima iteração.

$y_i(n)$  é o  $i$ -ésimo sinal de entrada do neurônio  $j$  na  $n$ -ésima iteração.

## BACK-PROPAGATION

Em todos os neurônios, a função de ativação  $\phi$  deve ser diferenciável. A **função de ativação** é aplicada ao potencial de ativação de cada neurônio ( $\text{netj}(n)$ ):

$$y_j(n) = \phi_j(\text{netj}(n))$$

É muito comum a utilização da **função logística sigmoidal**:

$$y_j(n) = 1 / (1 + e^{-\text{netj}(n)})$$

## BACK-PROPAGATION

Uma vez geradas as saídas dos neurônios da camada de saída da rede, o algoritmo Back-Propagation inicia a segunda etapa do treinamento para o padrão apresentado. Neste momento, o **signal de erro** produzido pelo neurônio  $j$  da camada de saída na iteração  $n$  é calculado por:

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

O **gradiente** local de cada neurônio  $j$  da camada de saída no instante  $n$  é calculado pela expressão:

$$\delta_j(n) = e_j(n) \phi'_j(\text{net}_j(n))$$

## BACK-PROPAGATION

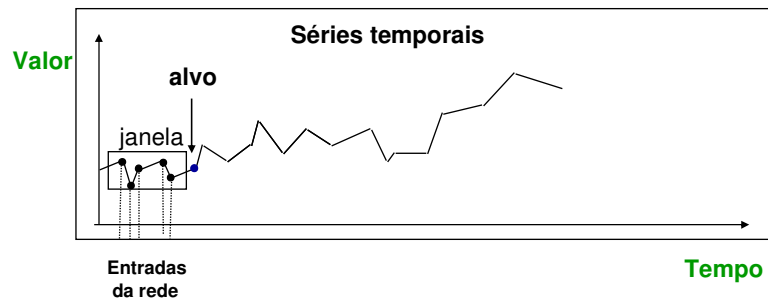
Esta etapa prossegue, passando os sinais de erro da direita para a esquerda, calculando o gradiente local associado a cada neurônio não pertencente à **camada de saída (1)**, e ajustando os **pesos das conexões (2)** associadas a ele.

$$\delta_j(n) = \phi'_j(\text{net}_j(n)) \sum_k \delta_k(n) w_{kj}(n) \quad (1)$$

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \quad (2)$$

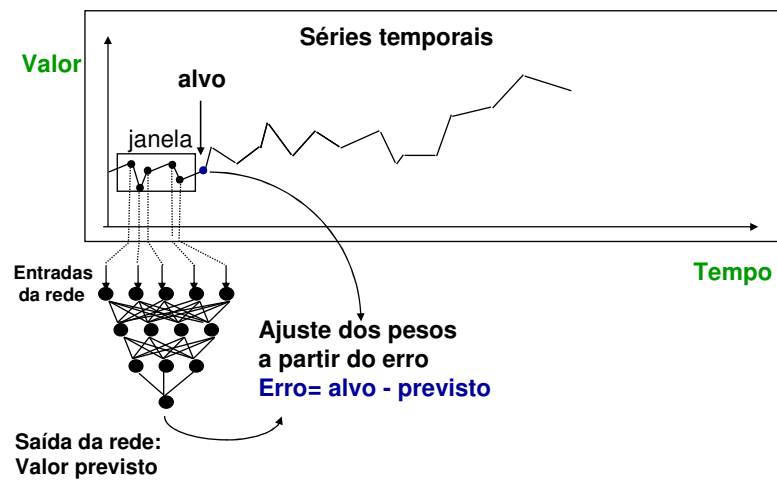
## BACK-PROPAGATION

### Aplicação em Previsão de Séries Temporais



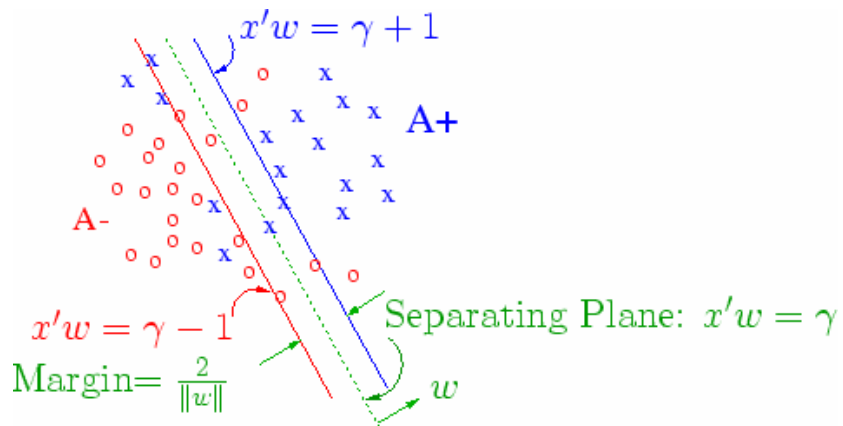
## BACK-PROPAGATION

### Aplicação em Previsão de Séries Temporais



## SVM – SUPPORT VECTOR MACHINES

TAREFA: CLASSIFICAÇÃO



## MODELOS NEURO-FUZZY HIERÁRQUICOS

TAREFA: CLASSIFICAÇÃO

