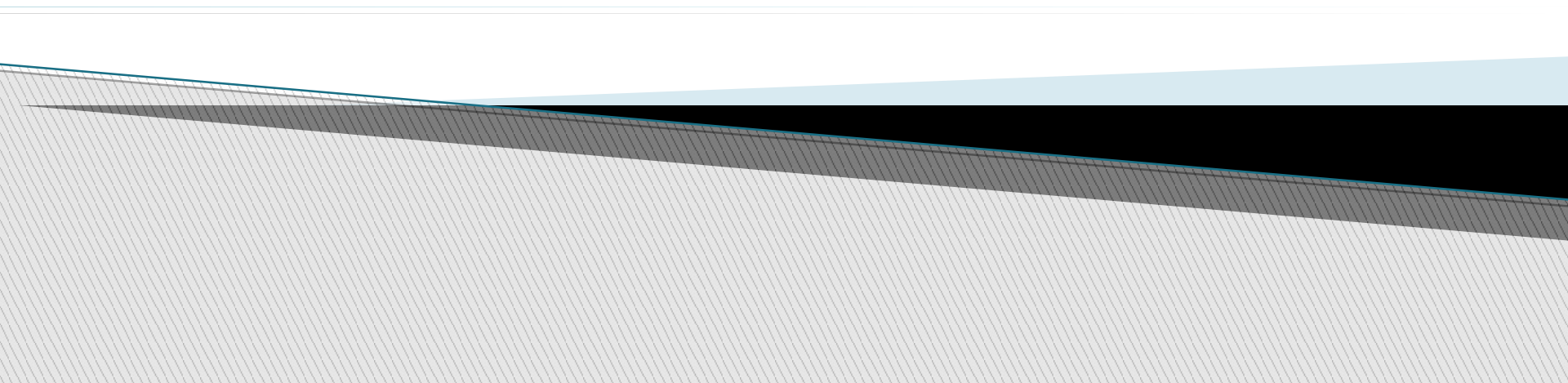


Inteligência Artificial

II

Redes Neurais MLP
Prof. Tales Bitelo Viegas



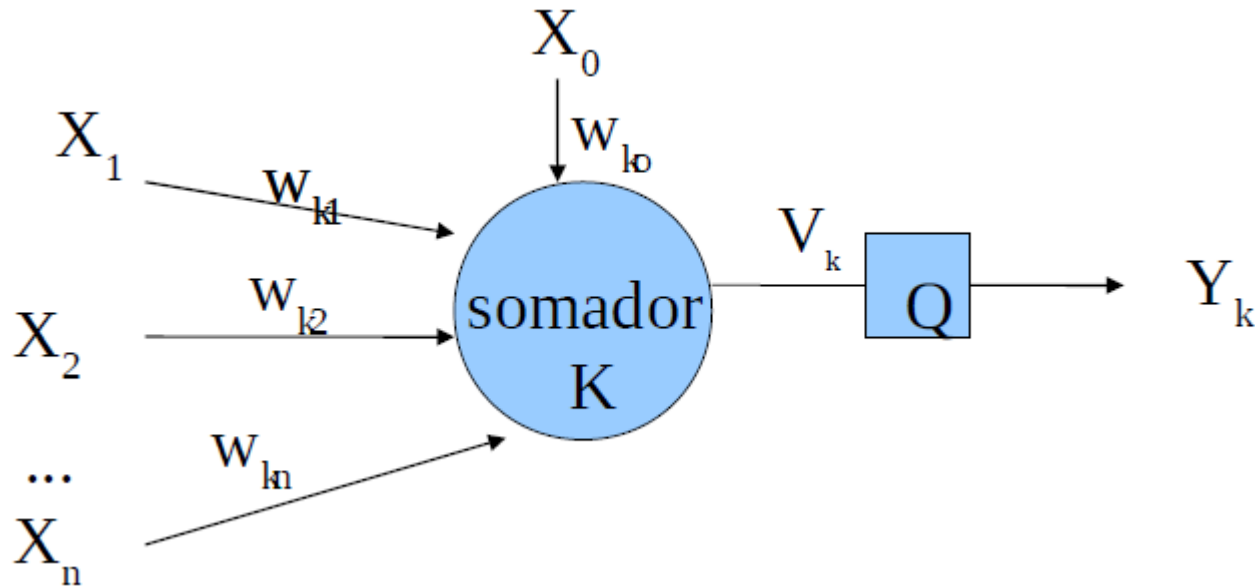
Conceito de Rede Neural Artificial (RNA)

- ▶ “Uma Rede Neural é um processador paralelamente distribuído, constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para uso.” (Haykin, 2001)

Conceito de Rede Neural Artificial (RNA)

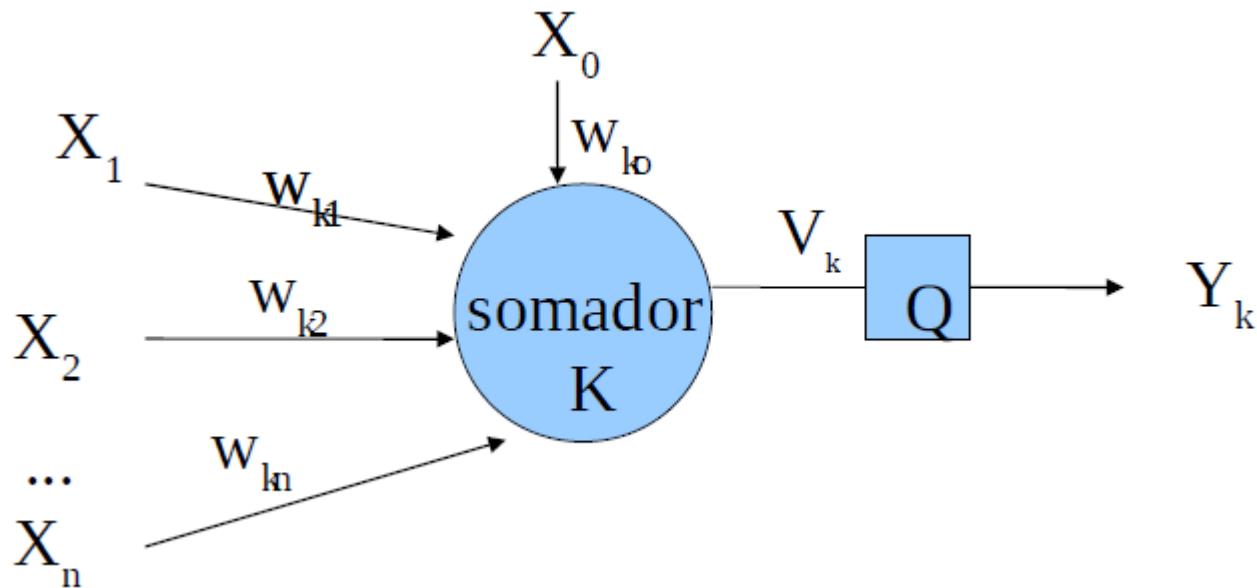
- ▶ Uma Rede Neural se assemelha ao cérebro em dois aspectos (Haskin, 2001):
 - O conhecimento é adquirido pela rede a partir do ambiente, através de um processo de aprendizagem
 - Forças de conexão entre os neurônios, conhecidas como pesos sinápticos, são usados para armazenar o conhecimento adquirido.

Modelo de Neurônio



- ▶ Modelo desenvolvido por McCulloch & Pitts na década de 1940.
- ▶ Elementos do neurônio:
 - Entradas X_1, X_2, \dots, X_n . X_0 é sempre 1
 - Pesos sinápticos: W_{kn} , onde k é o neurônio e n a entrada
 - W_{k0} : peso do bias
 - V_k : campo local induzido
 - Q : função de transferência ou ativação
 - Y_k : saída do neurônio k

Neurônio Artificial



- ▶ $V_k = \sum (W_{ki} * x_i)$ para $i=0$ até n
- ▶ $Y_k = Q(V_k)$

Neurônio Artificial

- ▶ Função de Transferência

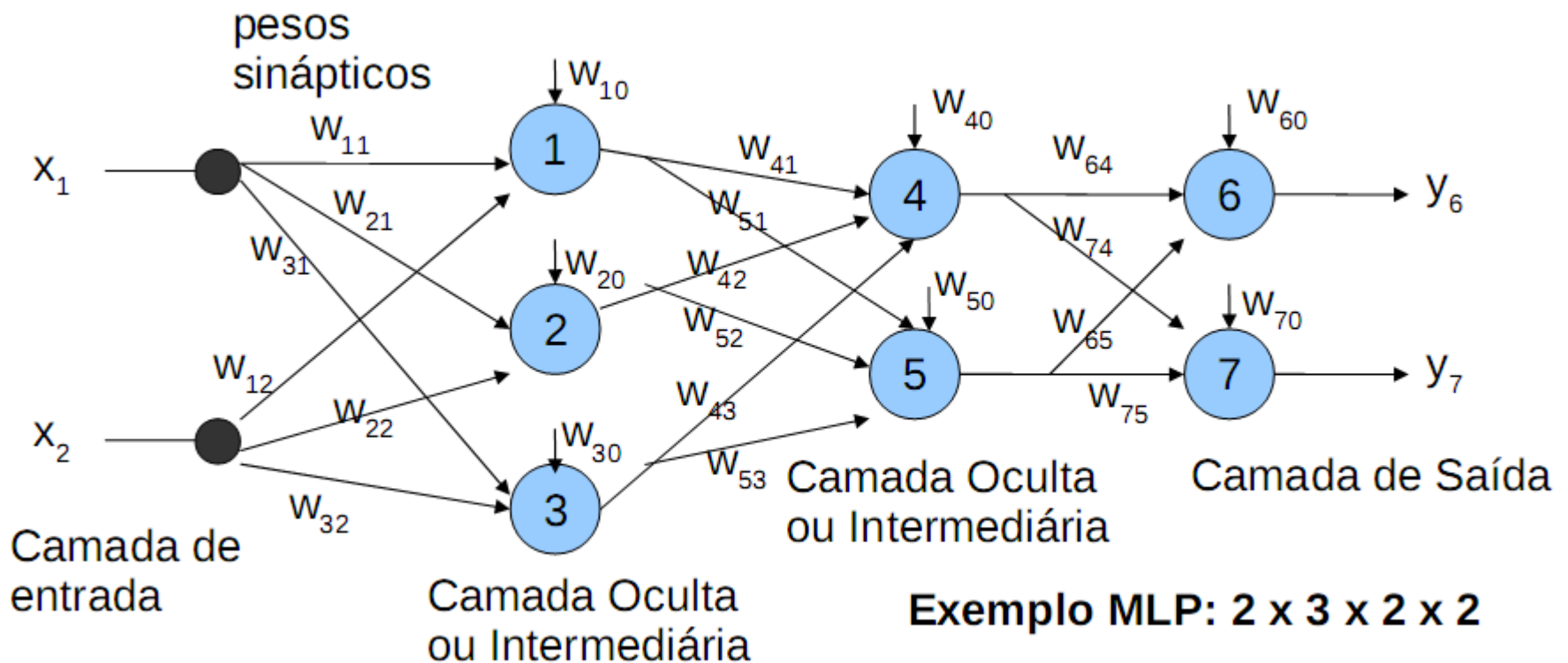
- Limiar

- ▮ $Q(V_k) = 1$ se $V_k \geq 0$
 0 se $V_k < 0$

Rede Neurais Artificiais

MLP

- ▶ **MLP: MultiLayer Perceptron**
- ▶ Redes perceptron de múltiplas camadas (*feed forward*), totalmente conectadas, contendo uma ou mais camadas ocultas



Fases de Construção

- ▶ Fase de **Aprendizagem**
 - Pré-processamento dos dados
 - Treinamento da rede
- ▶ Fase de **Generalização**
 - Pré-processamento dos dados
 - Teste da rede
 - Pós-processamento



Aprendizagem

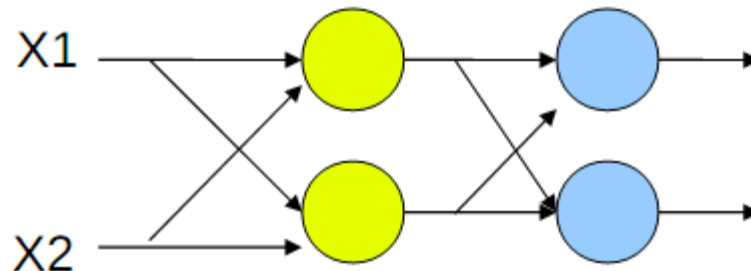
- ▶ Tanto na fase de aprendizagem quanto de generalização dos dados (padrões, amostras de entrada rotuladas) devem ser pré-processados.
- ▶ Os dados são particionados em dois subconjuntos distintos: conjunto de **treino** e conjunto de **teste**.
 - O conjunto de **treino** é usado no treinamento da rede – fase aprendizagem (~80% das amostras)
 - O conjunto de **teste** é usado para validar a rede – fase de generalização (~20% das amostras)

Pré-Processamento

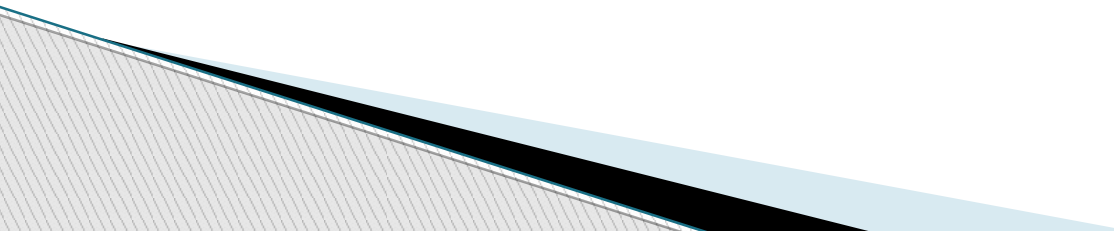
- ▶ A rede trabalha apenas com valores de entrada numéricos, os quais devem pertencer ao intervalo $[0..1]$

Topologia da Rede

- ▶ Para estimar o número de neurônios de uma camada oculta pode-se usar a seguinte heurística:
 - $\text{numNeuronios} = \text{raiz}(\text{numEntradas} \times \text{numSaídas})$
 - Ex: $\text{numEntradas} = 2$, $\text{numSaídas} = 2$
 $\text{numNeuronios} = 2$



Topologia da Rede

- ▶ Uma rede com 10 entradas e 5 saídas terá ~7 neurônios na camada oculta.
 - ▶ Esta heurística pode ser usada para definir o número de neurônios de qualquer camada oculta.
 - ▶ Definida a topologia, a rede pode ser então treinada para reconhecer os padrões
- 

Treinamento

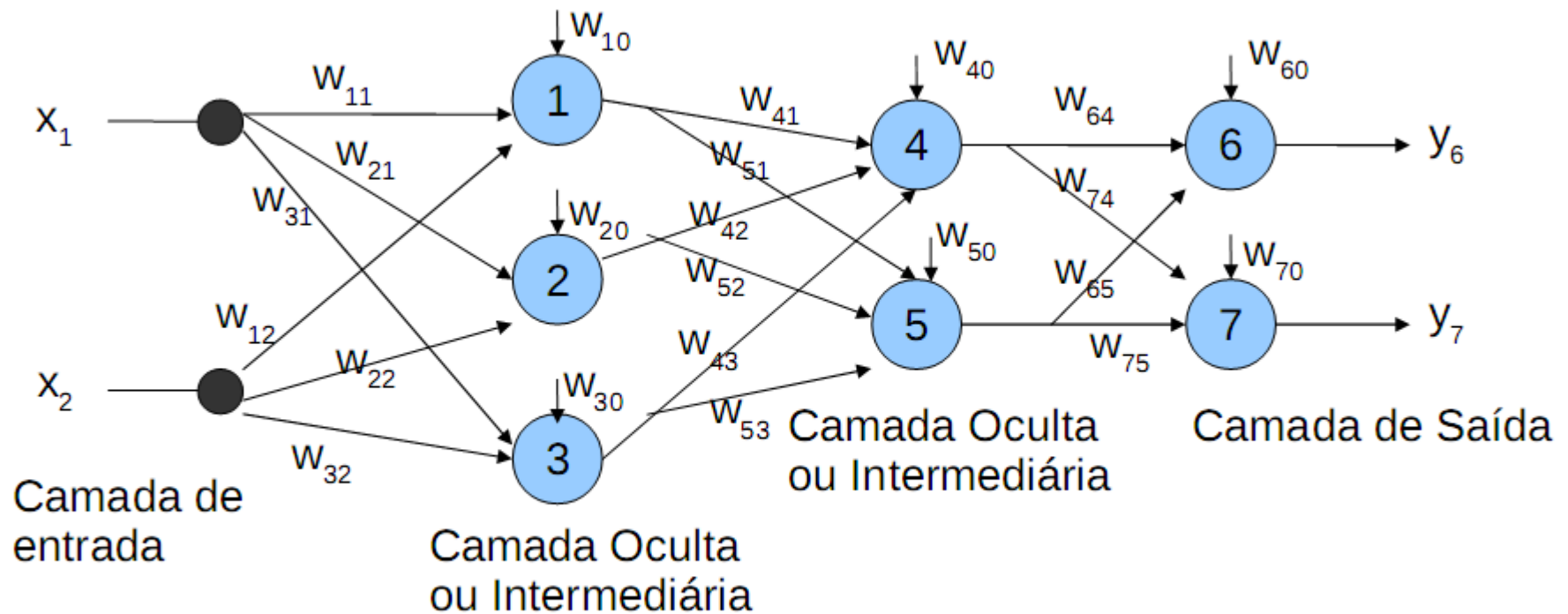
- ▶ O algoritmo **Error Backpropagation** é usado para treinar a rede. Ele possui 2 etapas: *forward* e *backward*. Em cada etapa a rede é percorrida em um sentido
- ▶ A fase *forward* (para frente) – **propagação** – é utilizada para definir a saída da rede para um dado padrão de entrada.
- ▶ A fase *backward* (para trás) – **retropropagação** – utiliza a saída desejada e a saída gerada pela rede para atualizar os pesos de suas conexões.

Treinamento

- ▶ Os ciclos de treinamento de uma rede são medidos em **épocas**
- ▶ Uma **época** corresponde a passagem de todos os padrões do conjunto de treino uma vez pela rede
- ▶ Para treinar uma rede são necessárias várias épocas

Treinamento

Propagação (fase forward)



Retropropagação (fase backward)

Algoritmo Error - Backpropagation

- ▶ O conjunto de treino da rede são N pares de valores (X,D) , onde $X=[x_1, x_2, \dots, x_z]$ e $D=[d_1, d_2, \dots, d_m]$, sendo **z** o número de atributos de cada padrão de entrada e **m** o número de saídas desejadas.
- ▶ Etapas:
 - 1 - Iniciar todos os pesos da rede com valores arbitrários não nulos
 - 2 - Apresentar cada padrão **n** de entrada do conjunto de treino e **propagá-lo** até a saída da rede (geração dos Y 's da última camada, a camada de saída), onde $n=1$ até N

Algoritmo Error - Backpropagation

- ▢ **Propagação:** para cada neurônio k da rede:
 - ▢ $V_k(n) = \sum (w_{ki}(n) * x_i(n))$, onde k é o neurônio e i a entrada, para $i=0$ até z (total de entradas, ou seja, total de atributos de u padrão \mathbf{n} do conjunto de treino). Quando o neurônio for de uma camada oculta, x_i será y_i (saída do neurônio i da camada anterior).
 - ▢ $Y_k(n) = Q(v_k(n))$, a saída é gerada pela aplicação da função de transferência Q sobre o campo local induzido v .

Algoritmo Error - Backpropagation

- 3 - Iniciar a **retropropagação**
 - ▢ Calcular o **erro**. Para cada neurônio k da camada de saída:
 - ▢ $\text{erro}_k(n) = d_k(n) - y_k(n)$, onde d_k é a saída desejada, o rótulo (a classe) do padrão n

Algoritmo Error - Backpropagation

- ❑ Calcular a energia do **erro instantâneo** para o padrão propagado:
 - ❑ $\varepsilon(n) = \frac{1}{2} * \sum (\text{erro}_k(n))^2$, onde n identifica o padrão e $k=1$ até o número de neurônios da camada de saída.
- ❑ Calcular os **gradientes δ da camada de saída**. Para cada neurônio k da camada de saída.
 - ❑ $\delta_k(n) = Q'(v_k(n)) * \text{erro}_k(n)$
- ❑ Calcular o ajuste dos pesos de k :
 - ❑ $\Delta w_{ki}(n) = \delta_k(n) * \eta * y_i(n)$, onde η é a taxa de aprendizagem - intervalo típico (0;1]

Algoritmo Error - Backpropagation

- ▢ Ajustar os pesos dos neurônios da camada de saída:
 - ▢ $w_{ki}(n+1) = w_{ki}(n) + \Delta w_{ki}(n)$
 - ▢ Pode-se usar ainda a constante de momento α , intervalo típico $[0;1]$:
 - ▢ $w_{ki}(n+1) = w_{ki}(n) + \Delta w_{ki}(n) + \alpha * w_{ki}(n-1)$
- ▢ Calcular os gradientes **δ da camadas ocultas**. Para cada neurônio k de uma camada oculta:
 - ▢ $\delta_k(n) = Q'(v_k(n)) * \sum(\delta_j(n) * w_{jk}(n+1))$, onde j são os neurônios com os quais o neurônio k tem conexão a direita. Como a camada à direita já sofreu retropropagação, seus pesos já foram atualizados, por isto aparece $n+1$.

Algoritmo Error - Backpropagation

- Da mesma forma, calcular o **ajuste dos pesos** de k:
 - $\Delta w_{ki}(n) = \delta_k(n) * \eta * y_i(n)$, quando o ajuste for a primeira camada oculta, a mais a esquerda, y_i será x_i .
- Da mesma forma, ajustar os pesos dos neurônios da camada oculta:
 - $w_{ki}(n+1) = w_{ki}(n) + \Delta w_{ki}(n)$
 - Pode-se usar ainda a constante de momento α :
 - $w_{ki}(n+1) = w_{ki}(n) + \Delta w_{ki}(n) + \alpha * w_{ki}(n-1)$

Algoritmo Error - Backpropagation

- 4 - Terminada a retropropagação do padrão n atual, repetir os passos 2 e 3 para o padrão seguinte ($n+1$) do conjunto de treino até que todos os padrões tenham sido processados pela rede, caracterizando, assim, uma **época**.

Algoritmo Error - Backpropagation

- 5 - Ao final de cada época, calcular o **erro médio quadrado E**:
 - ▢ $E_{\text{época}} = 1/N * \sum[\epsilon(j)]$, onde $j=1$ até N .
 - ▢ O erro médio quadrado é a média aritmética dos erros instantâneos dos padrões do conjunto de treino

Algoritmo Error - Backpropagation

- 6 – Parar o algoritmo quando:
 - ▢ For atingido um número pré-definido de épocas
 - ▢ O erro médio quadrado for menor que um valor pré-definido
 - ▢ A variação do erro médio quadrado nas últimas x épocas for inferior a um valor pré-definido
 - ▢ Número de padrões corretamente classificados não se alterar
 - ▢ Uma combinação destes critérios
- A seguir, algumas heurísticas para melhorar o algoritmo

Heurísticas para as funções de transferência

- ▶ Funcionam melhor quando são usadas com algumas constantes (Haykin, 2001)
 - Logística: $Q(V_k) = 1 / (1 + \exp(-\mathbf{a} * V_k))$
 - Tangente hiperbólica: $Q(V_k) = \mathbf{a} * \tanh(\mathbf{b} * V_k)$,
▮ Onde $\mathbf{a}=1,7159$ e $\mathbf{b}=2/3$

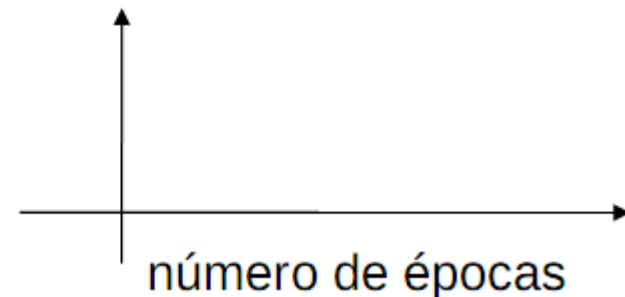
Heurísticas para as funções de Transferência

- ▶ Derivadas das funções:
 - Logística:
 - ▮ $Q'(V_k) = Q(V_k) * (1 - Q(V_k))$
 - Tangente hiperbólica:
 - ▮ $\tanh'(V_k) = 1 - \tanh^2(V_k)$

Heurísticas para a taxa de aprendizagem η e a constante de momento α

- ▶ Algumas taxas de aprendizagem usadas: 0.01, 0.1, 0.3, 0.5 e 0.9
- ▶ Algumas constantes de momento usadas: 0, 0.1, 0.5 e 0.9
- ▶ O valor que proporcionar a melhor curva de aprendizagem, determinada pelo erro médio, é a melhor escolha.

erro médio quadrado



Heurísticas para a taxa de aprendizagem η e a constante de momento α

- ▶ A taxa de aprendizagem pode ser atualizada ao longo da fase de treinamento:
 - $\eta(n) = \eta_0 / (1 + (n / \tau))$, onde η_0 é o valor inicial da taxa e deve ser alto; τ é uma constante que ajudará a atualizar a taxa.

Heurísticas para a taxa de aprendizagem η e a constante de momento α

- ▶ Para melhorar o desempenho da rede, pode-se:
 - Submeter os padrões do conjunto de treino aleatoriamente
 - Submeter primeiramente o padrão que na época anterior gerou o maior erro

Generalização

- ▶ Feito o treinamento da rede, os pesos que mapeiam os padrões de entrada nas saídas desejadas foram encontrados.
- ▶ A generalização consiste em **propagar** pela rede os padrões pré-processados do conjunto de teste e analisar os resultados gerados pela rede.
- ▶ Visto que as funções de transferência geram valores contínuos é comum um **pós-processamento da saída gerada**

Pós-Processamento

- ▶ É uma regra de decisão, geralmente baseada em algum valor limiar que auxilia a definir a que classe o padrão de teste pertence.
 - Ex: se $Y \geq 0.8$ então $Y = 1$