REDES NEURAIS ARTIFICIAIS

AULA 3 – REDE *ADALINE*

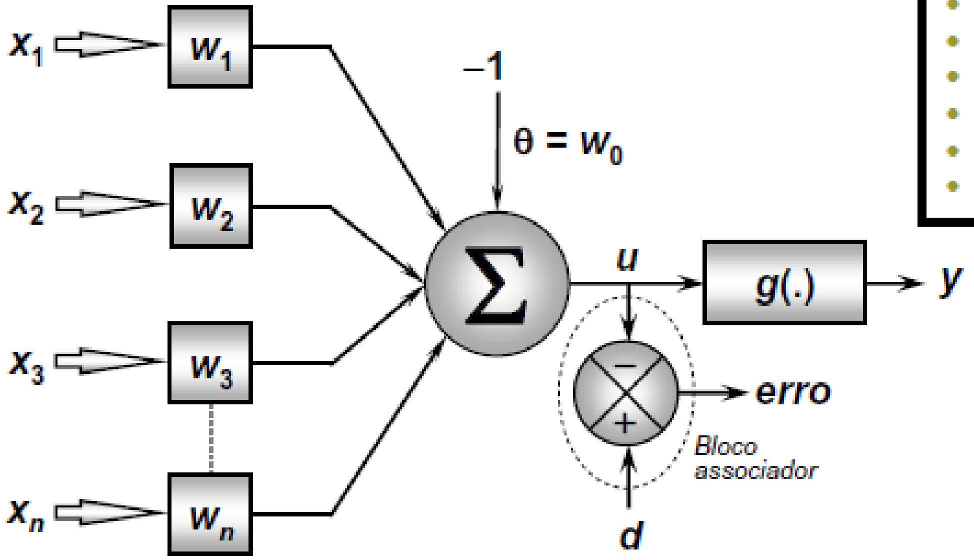
Prof. Rodrigo Palácios

rodrigopalacios@utfpr.edu.br

REDE *ADALINE*

- O ***ADALINE*** (*Adaptive Linear Element*) foi idealizado por Widrow & Hoff em 1960.
- Sua principal aplicação estava em sistemas de chaveamento de circuitos telefônicos.
- Esta foi uma das primeiras aplicações industriais que envolveram efetivamente a utilização das redes neurais artificiais.
- Embora seja também um tipo de rede bem simples (um único neurônio), o ***ADALINE*** promoveu alguns outros avanços para a área de redes neurais, isto é:
  - Desenvolvimento do algoritmo de aprendizado **regra Delta**.
  - Aplicações em diversos problemas práticos envolvendo processamento de sinais analógicos.
  - Primeiras aplicações industriais de redes neurais artificiais.

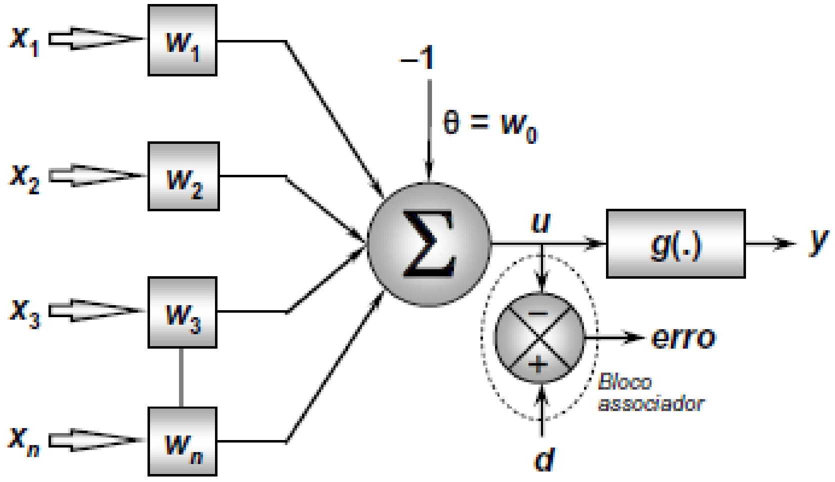
REDE *ADALINE* – ASPECTOS TOPOLÓGICOS



- Sinais de entrada ➔ $\{x_1, x_2, \ldots, x_n\}$
- Pesos sinápticos ➔ $\{w_1, w_2, \ldots, w_n\}$
- Combinador linear ➔ $\{\Sigma\}$
- Limiar de ativação ➔ $\{\theta\}$
- Potencial de ativação ➔ $\{u\}$
- Função de ativação ➔ $\{g\}$
- Sinal de saída ➔ $\{y\}$
- Valor desejado ➔ $\{d\}$
- Valor de erro ➔ $\{erro\}$ entre $\{d\}$ e $\{u\}$

1. Rede ***ADALINE*** é constituída de *n* sinais de entrada e somente uma saída (composta de um único neurônio).
2. Assim como o ***Perceptron***, a rede ***ADALINE*** possui também arquitetura *feedforward* de camada única, pois o fluxo de informações é realizado sempre adiante.
3. Assim como o ***Perceptron***, a rede ***ADALINE***, devida ainda à sua simplicidade estrutural, tem sido também mais utilizada em problemas de **“Classificação de Padrões”**, envolvendo apenas duas classes distintas.

REDE *ADALINE* – PASSOS PARA OBTENÇÃO DA RESPOSTA (SAÍDA)

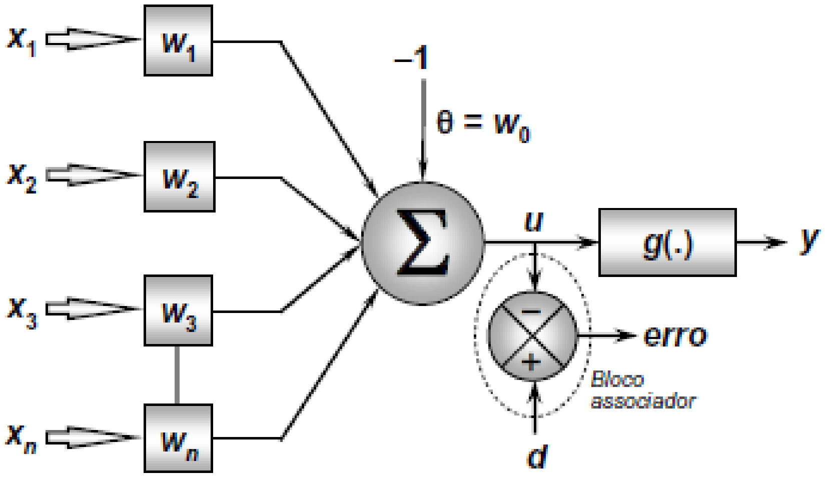


$$\begin{cases} u = \sum_{i=1}^{n} w_i \cdot x_i - \theta = \sum_{i=0}^{n} w_i \cdot x_i \\ y = g(u) \end{cases}$$

$$\boxed{erro = d - u}$$

1. Apresentação de um conjunto de valores que representam as variáveis de entrada do neurônio.
2. Multiplicação de cada entrada pelo seu respectivo peso sináptico.
3. Obtenção do potencial de ativação produzido pela soma ponderada dos sinais de entrada, subtraindo-se o limiar de ativação.
4. Aplicação de uma função de ativação apropriada, tendo-se como objetivo limitar a saída do neurônio.
5. Compilação da saída a partir da aplicação da função de ativação neural em relação ao seu potencial de ativação.
6. Presença de bloco associador, cuja função é produzir o sinal de "*erro*" entre "*d*" e "*u*" a fim de auxiliar no treinamento da rede.

REDE *ADALINE* – ASPECTOS DE APLICABILIDADE

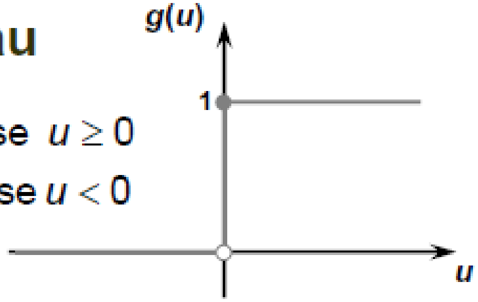


$$\begin{cases} u = \sum_{i=1}^{n} w_i \cdot x_i - \theta = \sum_{i=0}^{n} w_i \cdot x_i \\ y = g(u) \end{cases}$$

1. Assim como no ***Perceptron***, devido às suas características estruturais, as funções de ativação do ***ADALINE*** são a **"degrau"** ou **"degrau bipolar"**.
2. Assim, tem-se também apenas **"duas possibilidades"** de valores a serem produzidos pela sua saída, ou seja, valor 0 ou 1 (para a função de ativação Degrau), ou ainda, valor -1 ou 1 (para a função Degrau Bipolar).

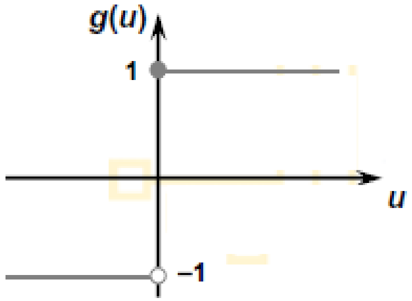
Degrau

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

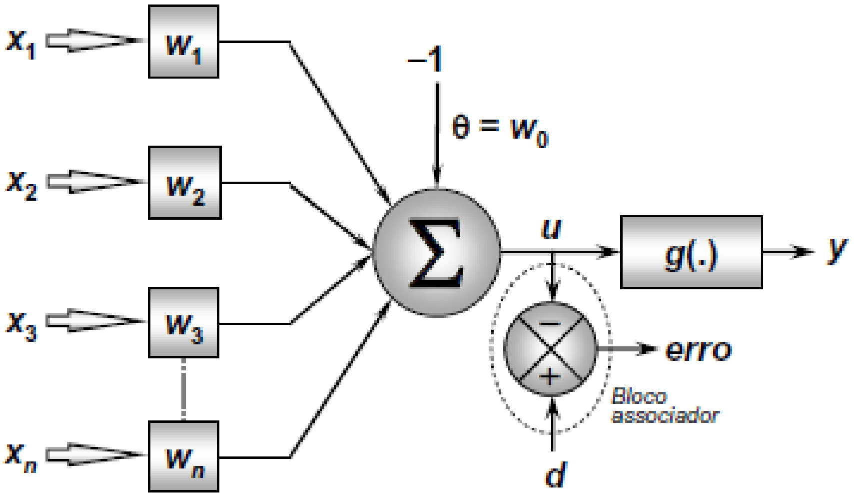


Degrau Bipolar

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



REDE *ADALINE* – ASPECTOS DE APLICABILIDADE



- Sinais de entrada ➔ $\{x_1, x_2, \ldots, x_n\}$
- Pesos sinápticos ➔ $\{w_1, w_2, \ldots, w_n\}$
- Combinador linear ➔ $\{\Sigma\}$
- Limiar de ativação ➔ $\{\theta\}$
- Potencial de ativação ➔ $\{u\}$
- Função de ativação ➔ $\{g\}$
- Sinal de saída ➔ $\{y\}$
- Valor desejado ➔ $\{d\}$
- Valor de erro ➔ $\{erro\}$ entre $\{d\}$ e $\{u\}$

Parâmetro	Variável Representativa	Tipo Característico
Entradas	x_i (*i*-ésima entrada)	Reais ou Binária (advindas externamente)
Pesos Sinápticos	w_i (associado a x_i)	Reais (iniciados aleatoriamente)
Limiar	θ	Real (iniciado aleatoriamente)
Saída	y	Binária
Função de Ativação	$g(.)$	Degrau ou Degrau Bipolar
Processo de Treinamento	-----	Supervisionado
Regra de Aprendizado	-----	Regra Delta

REDE *ADALINE* – ASPECTOS DO TREINAMENTO SUPERVISIONADO

Parâmetro	Variável Representativa	Tipo Característico
Entradas	x_i (i-ésima entrada)	Reais ou Binária (advindas externamente)
Pesos Sinápticos	w_i (associado a x_i)	Reais (iniciados aleatoriamente)
Limiar	θ	Real (iniciado aleatoriamente)
Saída	***y***	**Binária**
Função de Ativação	***g*(.)**	**Degrau ou Degrau Bipolar**
Processo de Treinamento	**-----**	**Supervisionado**
Regra de Aprendizado	**-----**	**Regra Delta**

1) Assim como no ***Perceptron***, o ajuste dos pesos e limiar do ***ADALINE*** é efetuado utilizando processo de treinamento "**Supervisionado**".

2) Então, para cada amostra dos sinais de entrada se tem a respectiva saída (resposta) desejada.

3) Como o ***ADALINE*** é tipicamente usado em problemas de classificação de padrões, a sua saída pode assumir somente dois valores possíveis, os quais se associam às **"duas classes"** que o mesmo estará identificando.

4) A diferença principal entre o ***ADALINE*** e o ***Perceptron*** está principalmente na regra de aprendizado utilizada para os ajustes dos pesos e limiar.

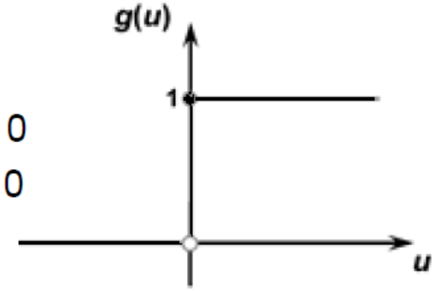
REDE *ADALINE* – ASPECTOS DE APLICABILIDADE

1) Assim como no ***Perceptron***, para problemas de classificação dos sinais de entrada, tem-se então duas classes possíveis para a saída do ***ADALINE***, denominadas de ***Classe A*** e ***Classe B***.

2) Então, como se tem somente **"duas possibilidades"** de valores a serem produzidos na saída do ***ADALINE***, podem-se ter as seguintes associações para a sua saída { $y = g(u)$ }:

Degrau

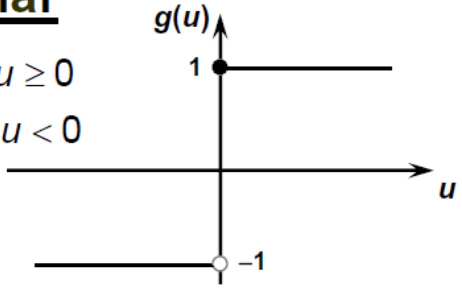
$$g(u) = \begin{cases} 1, & \text{se } u \geq 0 \\ 0, & \text{se } u < 0 \end{cases}$$



$$g(u) = \begin{cases} 1, & \text{se } u \geq 0 \Rightarrow \boldsymbol{x} \in \textit{Classe A} \\ 0, & \text{se } u < 0 \Rightarrow \boldsymbol{x} \in \textit{Classe B} \end{cases}$$

Degrau Bipolar

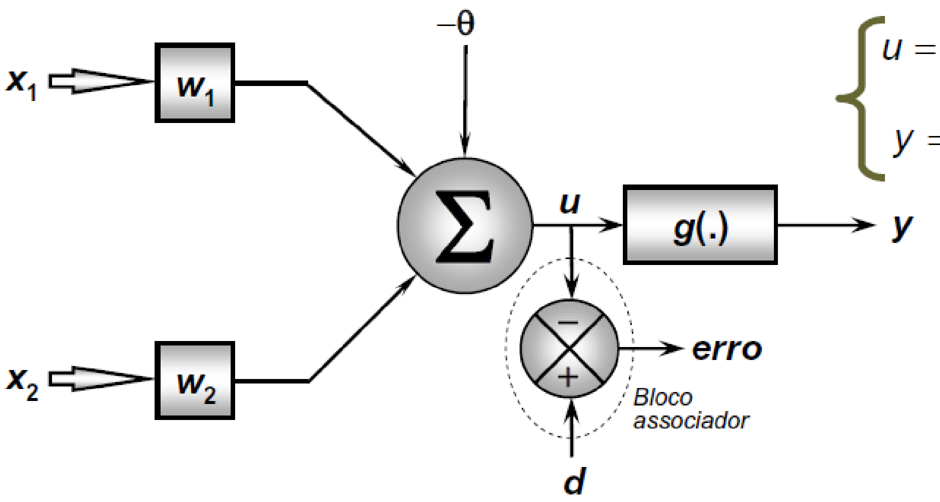
$$g(u) = \begin{cases} 1, & \text{se } u \geq 0 \\ -1, & \text{se } u < 0 \end{cases}$$



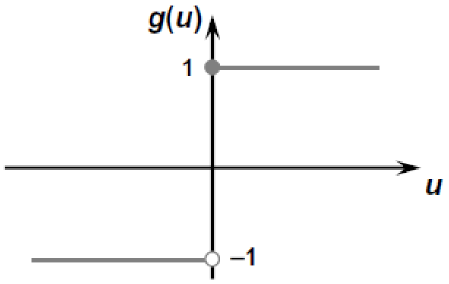
$$g(u) = \begin{cases} 1, & \text{se } u \geq 0 \Rightarrow \boldsymbol{x} \in \textit{Classe A} \\ -1, & \text{se } u < 0 \Rightarrow \boldsymbol{x} \in \textit{Classe B} \end{cases}$$

REDE *ADALINE* – ANÁLISE MATEMÁTICA

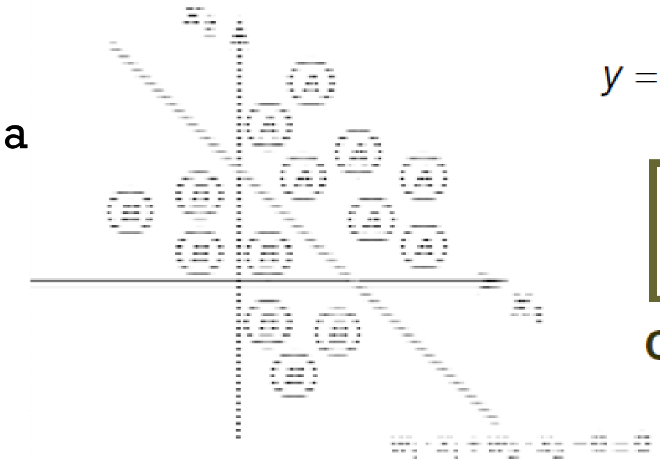
Assumindo-se aqui um ***ADALINE*** com apenas duas entradas, tem-se:



$$\begin{cases} u = \sum_{i=1}^{n=2} w_i \cdot x_i - \theta \Rightarrow w_1 \cdot x_1 + w_2 \cdot x_2 - \theta \\ y = g(u) \end{cases}$$



Usando a função de ativação sinal, a saída do ***ADALINE*** seria dada por:



$$y = g(u) = \begin{cases} 1, \text{ se } \sum w_i \cdot x_i - \theta \geq 0 \Leftrightarrow w_1 \cdot x_1 + w_2 \cdot x_2 - \theta \geq 0 \\ -1, \text{ se } \sum w_i \cdot x_i - \theta < 0 \Leftrightarrow w_1 \cdot x_1 + w_2 \cdot x_2 - \theta < 0 \end{cases}$$

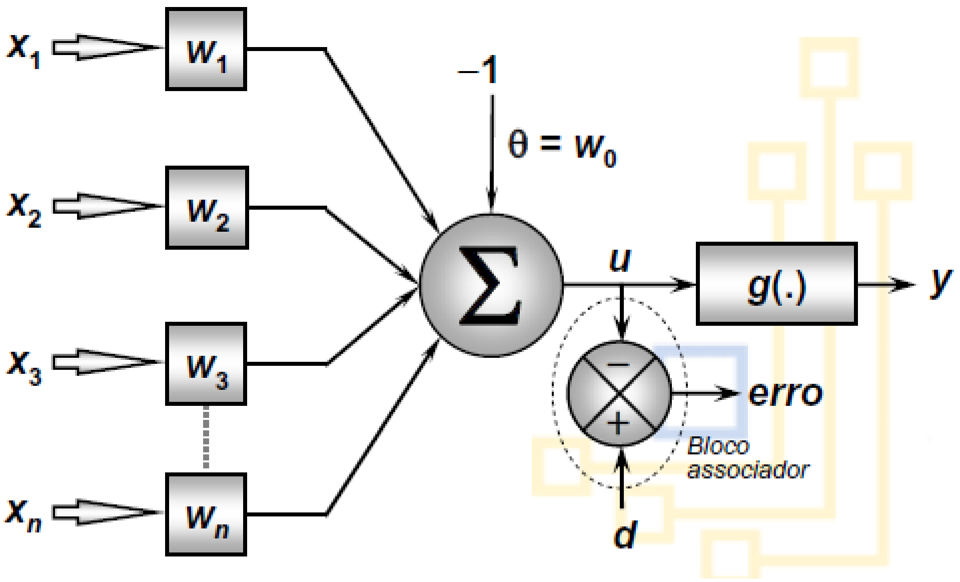
$$\boxed{g(u) = \begin{cases} 1, & \text{se } u \geq 0 \Rightarrow \mathbf{x} \in \textit{Classe A} \\ -1, & \text{se } u < 0 \Rightarrow \mathbf{x} \in \textit{Classe B} \end{cases}}$$

Conclusão: Assim como o ***Perceptron***, o ***ADALINE*** é também um **classificador linear** que pode ser usado para classificar duas classes.

REDE *ADALINE* – PROCESSO DE TREINAMENTO (REGRA DELTA)

- O processo de ajuste dos pesos e limiar do ***ADALINE*** é baseado no algoritmo de aprendizado denominado de regra Delta, também conhecida pelos seguintes nomes:
  - Regra de aprendizado de Widrow-Hoff.
  - Algoritmo LMS (Least Mean Square).
  - Método do Gradiente Descendente.
- Assumindo-se a disponibilidade de ***p*** amostras de treinamento, o princípio envolvido com a aplicação da regra Delta, objetivando ajustar os pesos e limiar do neurônio, é a seguinte:
  - Minimizar a diferença ***(erro)*** entre a saída desejada ***d*** e a resposta do combinador ***u***, considerando-se para tanto todas as ***p*** amostras.
  - Mais especificamente, utiliza-se a minimização do Erro Quadrático entre ***u*** e ***d*** com o intuito de ajustar o vetor de pesos **{*w*}** da rede.

$$\mathbf{w} = [\theta \quad w_1 \quad w_2 \ \ldots \ w_n]^T$$



REDE *ADALINE* – DERIVAÇÃO DO ALGORITMO (REGRA DELTA)

- Consiste de obter um ***w**** ótimo tal que o **Erro Quadrático** {$E(\mathbf{w}^*)$} sobre todo o conjunto de amostras seja o mínimo possível, isto é:

$$E(\mathbf{w}^*) \le E(\mathbf{w}), \quad \text{para } \forall \mathbf{w} \in \Re^{n+1}$$

- A função **Erro Quadrático** em relação às ***p*** amostras de treinamento é definida por:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{p}(d^{(k)} - u)^2$$

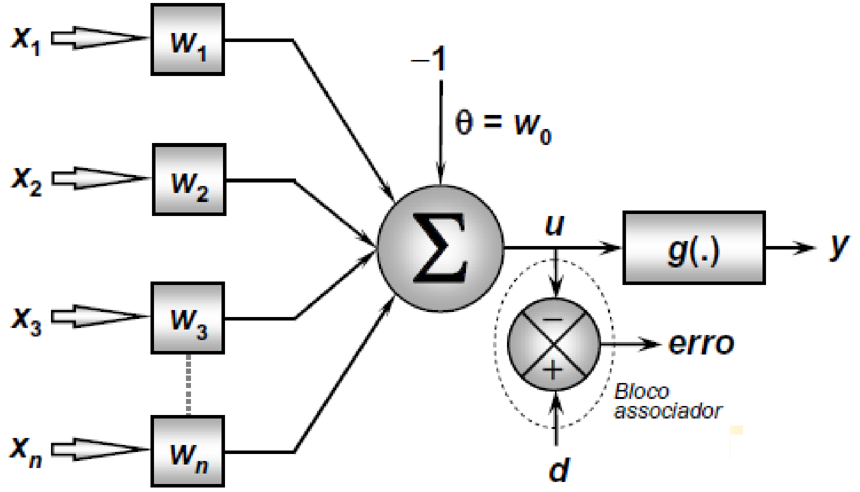
- Usando o valor de ***u*** na expressão acima, obtém-se:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{p}(d^{(k)} - (\sum_{i=1}^{n} w_i \cdot x_i^{(k)} - \theta))^2$$

- Em forma vetorial, tem-se:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{p}(d^{(k)} - (\mathbf{w}^T \cdot \mathbf{x}^{(k)} - \theta))^2$$

- **Conclusão:** Esta expressão acima totaliza o erro quadrático contabilizando-se as ***p*** amostras de treinamento.



$$\begin{cases} u = \sum_{i=1}^{n} w_i \cdot x_i^{(k)} - \theta \\ y = g(u) \end{cases}$$

REDE *ADALINE* – DERIVAÇÃO DO ALGORITMO (REGRA DELTA)

- **Aplicação do operador Gradiente:**
  - Consiste de aplicar o operador gradiente em relação ao vetor ***w***, tendo-se como objetivo a busca do valor ótimo para o erro quadrático, isto é:

$$\nabla E(\mathbf{w}) = \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$$

  - Considerando o valor de *E(**w**)* na expressão acima, obtém-se:

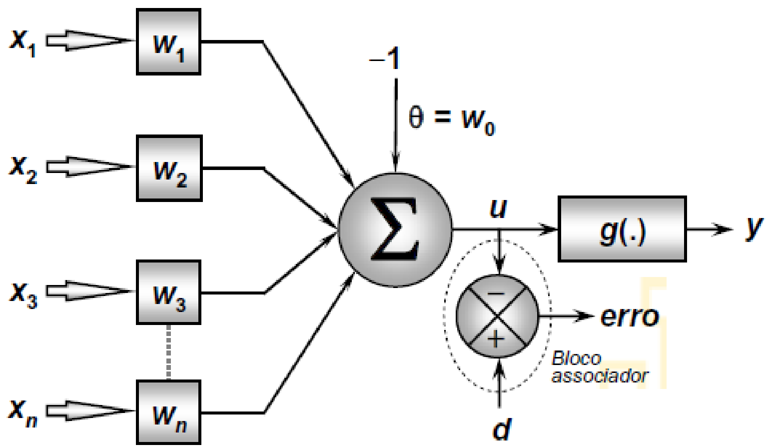
$$\nabla E(\mathbf{w}) = \sum_{k=1}^{p} (d^{(k)} - (\underbrace{\mathbf{w}^T \cdot \mathbf{x}^{(k)} - \theta}_{u})) \cdot (-\mathbf{x}^{(k)})$$

  - Retornando o valor de ***u*** na expressão acima, tem-se:

$$\nabla E(\mathbf{w}) = -\sum_{k=1}^{p} (d^{(k)} - u) \cdot \mathbf{x}^{(k)}$$

  - **Conclusão**: Esta expressão acima fornece o valor do gradiente do erro em relação a todas as amostras de treinamento.

$$\boxed{E(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^{p} (d^{(k)} - (\mathbf{w}^T \cdot \mathbf{x}^{(k)} - \theta))^2}$$



$$\begin{cases} u = \sum_{i=1}^{n} w_i \cdot x_i^{(k)} - \theta \\ y = g(u) \end{cases}$$

REDE *ADALINE* – DERIVAÇÃO DO ALGORITMO (REGRA DELTA)

- **Ajuste do Vetor de Pesos:**
  - Agora, deve-se associar o valor do gradiente ao procedimentos de ajustes do vetor de pesos {***w***}.
  - Assim, a adaptação do vetor de pesos devem ser efetuado na direção oposta àquela do gradiente, pois o objetivo da otimização é minimizar o erro quadrático médio entre ***d*** e ***u***.
  - Nesta condição, a variação Δ***w*** a ser efetivada no vetor de pesos do ***ADALINE*** é dada por:

$$\Delta \mathbf{w} = -\eta \cdot \nabla E(\mathbf{w})$$

$$\boxed{\nabla E(\mathbf{w}) = -\sum_{k=1}^{p} (d^{(k)} - u) \cdot \mathbf{x}^{(k)}}$$

  - Inserindo o resultado de ∇*E*(***w***) na expressão acima, tem-se:

$$\Delta \mathbf{w} = \eta \sum_{k=1}^{p} (d^{(k)} - u) \cdot \mathbf{x}^{(k)}$$

  - De forma complementar, pode-se exprimir Δ***w*** por:

$$\mathbf{w}^{atual} = \mathbf{w}^{anterior} + \eta \sum_{k=1}^{p} (d^{(k)} - u) \cdot \mathbf{x}^{(k)}$$

REDE *ADALINE* – DERIVAÇÃO DO ALGORITMO (REGRA DELTA)

- **Simplificação Algorítmica:**
  - Por razões de simplificação computacional, a atualização de **w** pode ser também realizada de forma discreta.
  - Neste caso, um passo de ajuste em **w** é realizado após a apresentação de cada *k*-ésima amostra de treinamento, ou seja:

$$\mathbf{w}^{atual} = \mathbf{w}^{anterior} + \eta \sum_{k=1}^{p} (d^{(k)} - u) \cdot \mathbf{x}^{(k)}$$

$$\mathbf{w}^{atual} = \mathbf{w}^{anterior} + \eta \cdot (d^{(k)} - u) \cdot \mathbf{x}^{(k)}, \quad k = 1 \ldots p$$

  - Em notação algorítmica, tem-se:

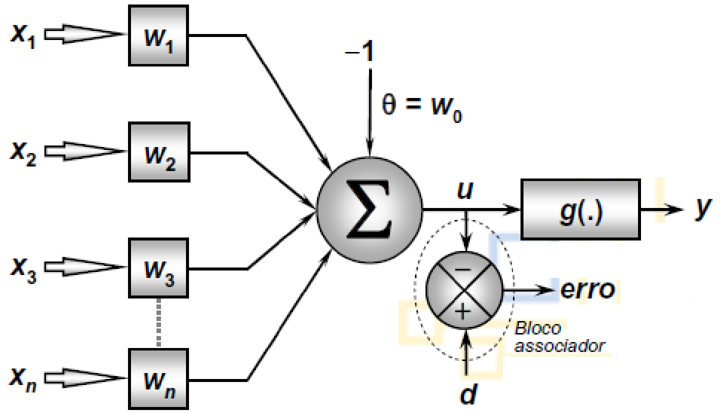
$$\boxed{\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot (d^{(k)} - u) \cdot \mathbf{x}^{(k)}, \quad k = 1 \ldots p}$$

onde: $\mathbf{w} = [\theta \quad w_1 \quad w_2 \ \ldots \ w_n]^T$

$\mathbf{x}^{(k)} = [-1 \quad x_1^{(k)} \quad x_2^{(k)} \ \ldots \ x_n^{(k)}]^T$

$d^{(k)}$ é o valor desejado para *k*-ésima amostra de treinamento.

η é a taxa de aprendizagem $\{0 < \eta < 1\}$.



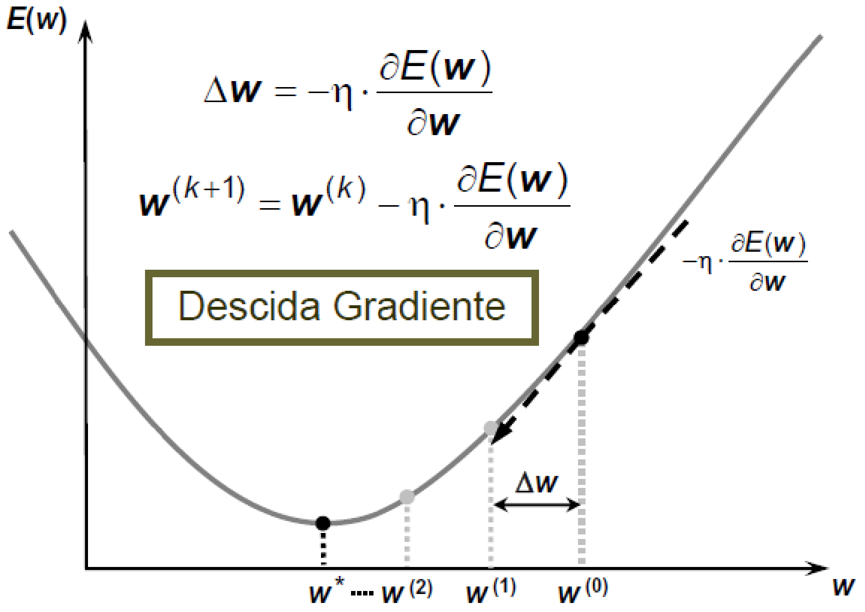
REDE *ADALINE* – DERIVAÇÃO DO ALGORITMO (REGRA DELTA)

- **Interpretação geométrica (processo de convergência do ADALINE):**
  - A interpretação geométrica frente aos passos de atualização der ***w***, rumo ao ponto de minimização ***w**** da função erro quadrático {$E(\mathbf{w})$}, é dada pela seguinte ilustração:

    1) Parte-se de um valor inicial $\mathbf{w}^{(0)}$;

    2) O próximo valor de ***w*** (dado por $\mathbf{w}^{(1)}$) será obtido considerando-se a direção oposta ao vetor gradiente em relação a $\mathbf{w}^{(0)}$;

    3) Para o próximo passo de atualização, o ajuste de ***w*** (representado agora por $\mathbf{w}^{(2)}$) será realizado considerando-se o valor do gradiente em relação à $\mathbf{w}^{(1)}$;
    4) Aplicando-se tais passos sucessivamente, haverá a convergência em direção a ***w****;
    5) Este valor de ***w**** será a configuração ótima para os parâmetros livres do ***ADALINE***;
    6) O valor de $E(\mathbf{w}^*)$ será então sempre menor que quaisquer $E(\mathbf{w})$ calculados nos passos anteriores.



REDE *ADALINE* – DERIVAÇÃO DO ALGORITMO (REGRA DELTA)

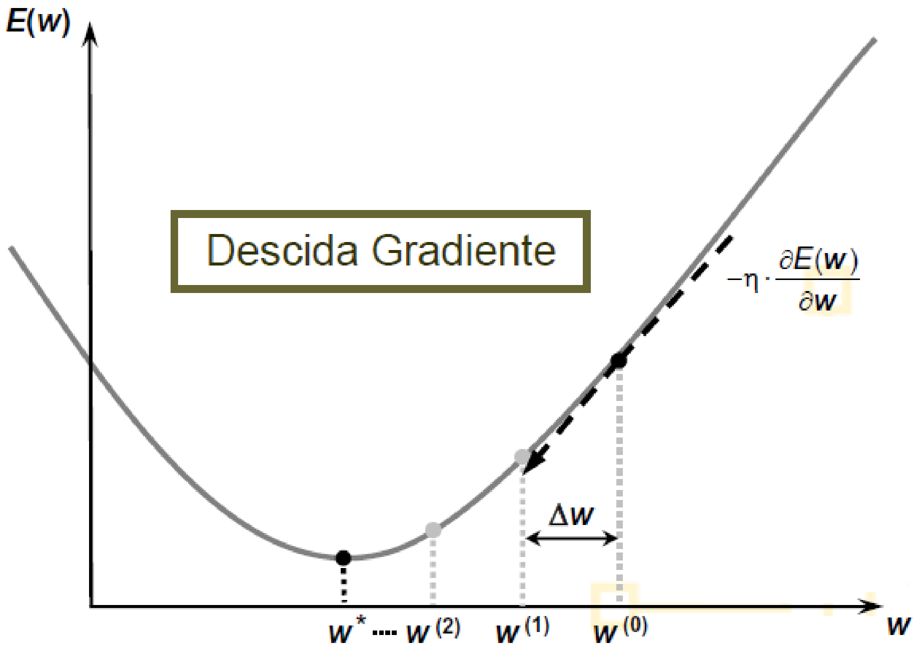
- **Critério de Parada do Algoritmo:**
  - Com a aplicação dos passos iterativos anteriores, o vetor ***w*** caminhará em direção ao seu valor ótimo {***w****}.
  - Para se detectar o alcance de ***w****, a fim de cessar o processo de convergência, há a necessidade de se estabelecer um critério de parada.
  - Para tanto, utiliza-se o valor do erro quadrático médio $\{E_{qm}(\mathbf{w})\}$ em relação a todas as amostras de treiname sendo este definido por:

$$E_{qm}(\mathbf{w}) = \frac{1}{p}\sum_{k=1}^{p}(d^{(k)} - u)^2$$

O algoritmo converge quando o erro quadrático médio entre duas épocas sucessivas for bem pequeno, isto é:

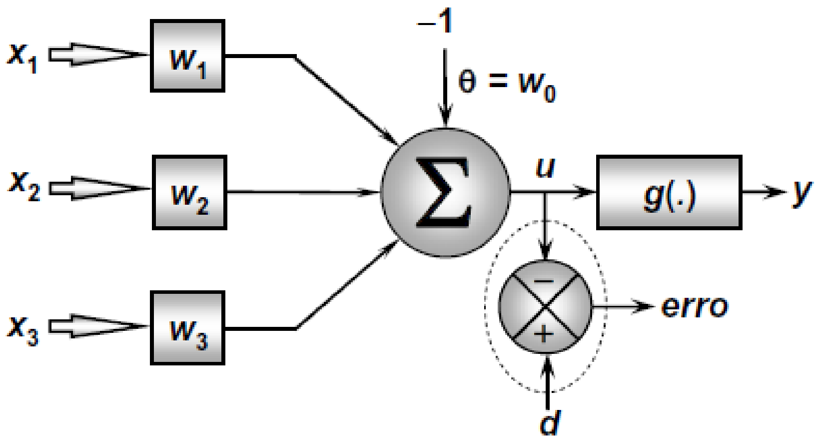
$$| E_{qm}(\mathbf{w}^{atual}) - E_{qm}(\mathbf{w}^{anterior}) | \leq \varepsilon$$

onde ε é a precisão requerida para o processo de convergência.



REDE *ADALINE* – IMPLEMENTAÇÃO COMPUTACIONAL

- **Montagem de conjuntos de treinamento:**
  - Supõe-se que um problema a ser mapeado pelo ***ADALINE*** tenha três entradas { x_1, x_2, x_3 }, conforme a figura ao lado.
  - Assume-se que se tem quatro amostras, constituída dos seguintes valores de entrada:
    - **Amostra 1** → Entrada: [0,1 0,4 0,7] → Saída desejada: [1]
    - **Amostra 2** → Entrada: [0,3 0,7 0,2] → Saída desejada: [-1]
    - **Amostra 3** → Entrada: [0,6 0,9 0,8] → Saída desejada: [-1]
    - **Amostra 4** → Entrada: [0,5 0,7 0,1] → Saída desejada: [1]



- Então, de forma similar ao ***Perceptron***, pode-se converter tais sinais para que estes possam ser usados no treinamento do ***ADALINE***:
- Geralmente, as amostras de treinamento s disponibilizadas em sua forma matricial (por meio de arquivo texto ou planilha).

Conjunto de treinamento

$\boldsymbol{x}^{(1)} = [-1 \quad 0{,}1 \quad 0{,}4 \quad 0{,}7]^T \rightarrow \text{com } d^{(1)} = 1$

$\boldsymbol{x}^{(2)} = [-1 \quad 0{,}3 \quad 0{,}7 \quad 0{,}2]^T \rightarrow \text{com } d^{(2)} = -1$

$\boldsymbol{x}^{(3)} = [-1 \quad 0{,}6 \quad 0{,}9 \quad 0{,}8]^T \rightarrow \text{com } d^{(3)} = -1$

$\boldsymbol{x}^{(4)} = [-1 \quad 0{,}5 \quad 0{,}7 \quad 0{,}1]^T \rightarrow \text{com } d^{(4)} = 1$

forma matricial

$$\Omega^{(x)} = \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{cccc} \boldsymbol{x}^{(1)} & \boldsymbol{x}^{(2)} & \boldsymbol{x}^{(3)} & \boldsymbol{x}^{(4)} \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 \\ 0{,}1 & 0{,}3 & 0{,}6 & 0{,}5 \\ 0{,}4 & 0{,}7 & 0{,}9 & 0{,}7 \\ 0{,}7 & 0{,}2 & 0{,}8 & 0{,}1 \end{bmatrix} \end{array};$$

$$\Omega^{(d)} = \begin{array}{c} \begin{array}{cccc} d^{(1)} & d^{(2)} & d^{(3)} & d^{(4)} \end{array} \\ \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix} \end{array}$$

REDE *ADALINE* – IMPLEMENTAÇÃO COMPUTACIONAL

- **Pseudocódigo para a Fase de Treinamento:**

Início {Algoritmo *Adaline* – Fase de Treinamento}

<1> Obter o conjunto de amostras de treinamento { $\boldsymbol{x}^{(k)}$ };

<2> Associar a saída desejada { $d^{(k)}$ } para cada amostra obtida;

<3> Iniciar o vetor $\boldsymbol{w}$ com valores aleatórios pequenos;

<4> Especificar taxa de aprendizagem { η } e precisão requerida { ε };

<5> Iniciar o contador de número de épocas { $época \leftarrow 0$ };

<6> Repetir as instruções:

<6.1> $E_{qm}^{anterior} \leftarrow E_{qm}(\boldsymbol{w})$;

<6.2> Para todas as amostras de treinamento { $\boldsymbol{x}^{(k)}$, $d^{(k)}$ }, fazer:

<6.2.1> $u \leftarrow \boldsymbol{w}^T \cdot \boldsymbol{x}^{(k)}$;

<6.2.2> $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \cdot (d^{(k)} - u) \cdot \boldsymbol{x}^{(k)}$;

<6.3> $época \leftarrow época + 1$;

<6.4> $E_{qm}^{atual} \leftarrow E_{qm}(\boldsymbol{w})$;

Até que: $\left| E_{qm}^{atual} - E_{qm}^{anterior} \right| \leq \varepsilon$

Fim {Algoritmo *Adaline* – Fase de Treinamento}

Critério de parada → O algoritmo converge quando o erro quadrático médio {E_{qm}} entre duas épocas sucessivas for menor ou igual à precisão {ε} requerida ao problema mapeado pelo ***ADALINE***.

REDE *ADALINE* – IMPLEMENTAÇÃO COMPUTACIONAL

- **Pseudocódigo para a Fase de Operação:**

Início {Algoritmo *Adaline* – Fase de Operação}

<1> Obter uma amostra a ser classificada { $\boldsymbol{x}$ };

<2> Utilizar o vetor $\boldsymbol{w}$ ajustado durante o treinamento;

<3> Executar as seguintes instruções:

<3.1> $u \leftarrow \boldsymbol{w}^T \cdot \boldsymbol{x}$;

<3.2> $y \leftarrow \text{sinal}(u)$;

<3.3> Se $y = -1$

<3.3.1> Então: amostra $\boldsymbol{x} \in$ *{Classe A}*

<3.4> Se $y = 1$

<3.4.1> Então: amostra $\boldsymbol{x} \in$ *{Classe B}*

Fim {Algoritmo *Adaline* – Fase de Operação}

Obs. 1 → A "Fase de Operação" é usada somente após a fase de treinamento, pois aqui a rede já está apta para ser usada no processo.

Obs. 2 → A "Fase de Operação" é então utilizada para realizar a tarefa de classificação de padrões frente às novas amostras que serão apresentadas à rede.

Obs. 3 → Lembrar de incluir o valor -1 dentro do vetor ***x***, pois será multiplicado por ***w***.

$$\boldsymbol{x} = [-1 \quad x_1 \quad x_2 \ldots \; x_n]^T$$

$$\boldsymbol{w} = [\theta \quad w_1 \quad w_2 \; \ldots \; w_n]^T$$

REDE *ADALINE* – IMPLEMENTAÇÃO COMPUTACIONAL

- **Pseudocódigo para o Erro Quadrático Médio:**

Início {Algoritmo EQM}

<1> Obter a quantidade de padrões de treinamento { p };

<2> Iniciar a variável E_{qm} com valor zero { $E_{qm} \leftarrow 0$ };

<3> Para todas as amostras de treinamento { $\boldsymbol{x}^{(k)}$, $d^{(k)}$ }, fazer:

<3.1> $u \leftarrow \boldsymbol{w}^T \cdot \boldsymbol{x}^{(k)}$;

<3.2> $E_{qm} \leftarrow E_{qm} + (d^{(k)} - u)^2$;

<4> $E_{qm} \leftarrow \dfrac{E_{qm}}{p}$;

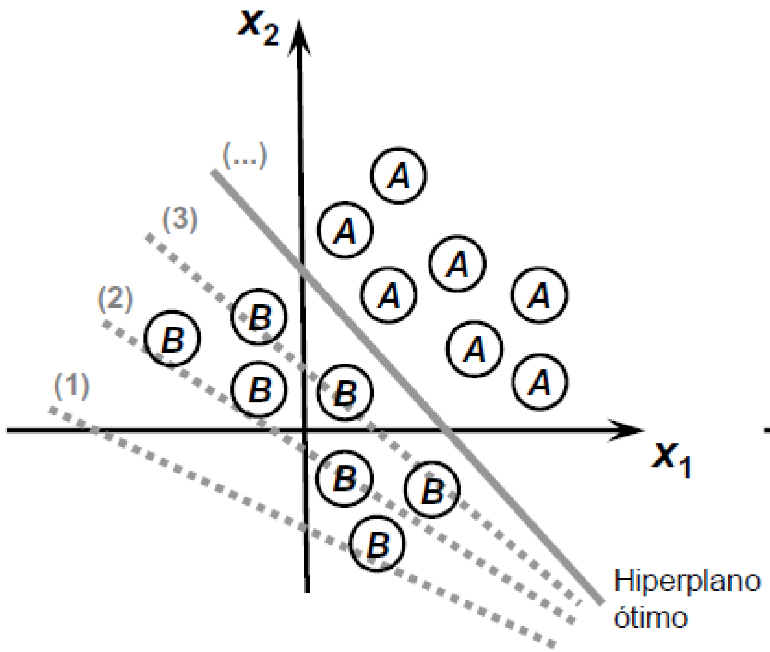
Fim {Algoritmo EQM}

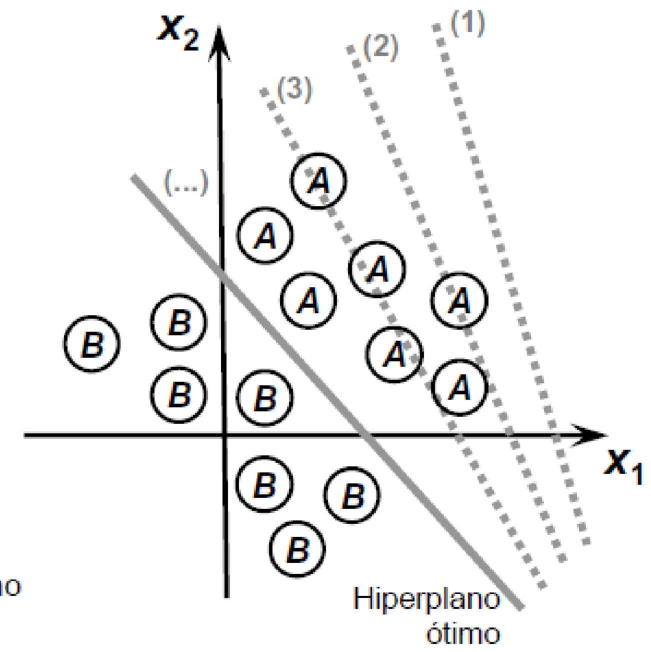
Obs. 1 → O algoritmo do erro quadrático médio deve ser implementado por meio de uma sub-rotina (função).

Obs. 2 → Todas as amostras de treinamento devem ser contabilizadas no cálculo do valor do erro quadrático médio.

REDE *ADALINE* – IMPLEMENTAÇÃO COMPUTACIONAL

- **Processo de Convergência:**
  - Processo de treinamento do ***ADALINE*** tende a mover continuamente o seu vetor de pesos, tendo-se o intuito de minimizar o erro quadrático em relação a todas as amostras disponíveis para o aprendizado.
  - As figuras mostram duas situações que ilustram a convergência do ***ADALINE*** rumo à estabilização, considerando-se para fins didáticos apenas duas entradas $\{x_1$ e $x_2\}$.

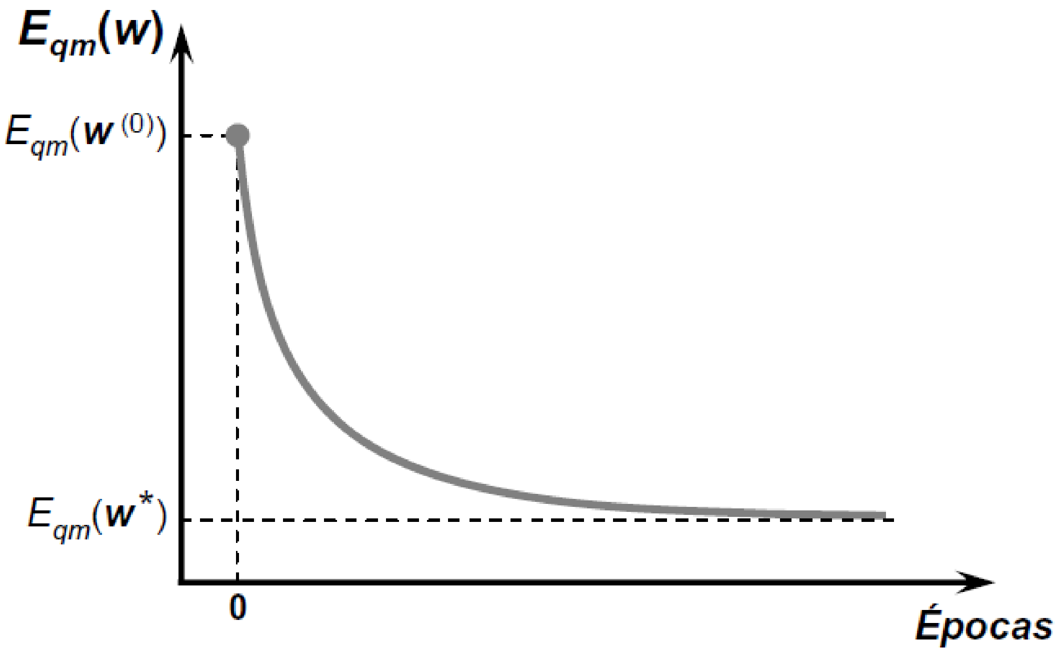




- Observa-se que nestas duas situações, cujos vetores de pesos iniciais $\boldsymbol{w}^{(0)}$ foram iniciados em posições distintas, o processo de convergência encaminha o hiperplano sempre para o mesmo local.
- Nesses casos, o valor correspondente a $\boldsymbol{w}^*$ é sempre aquele que minimiza a função erro quadrático.

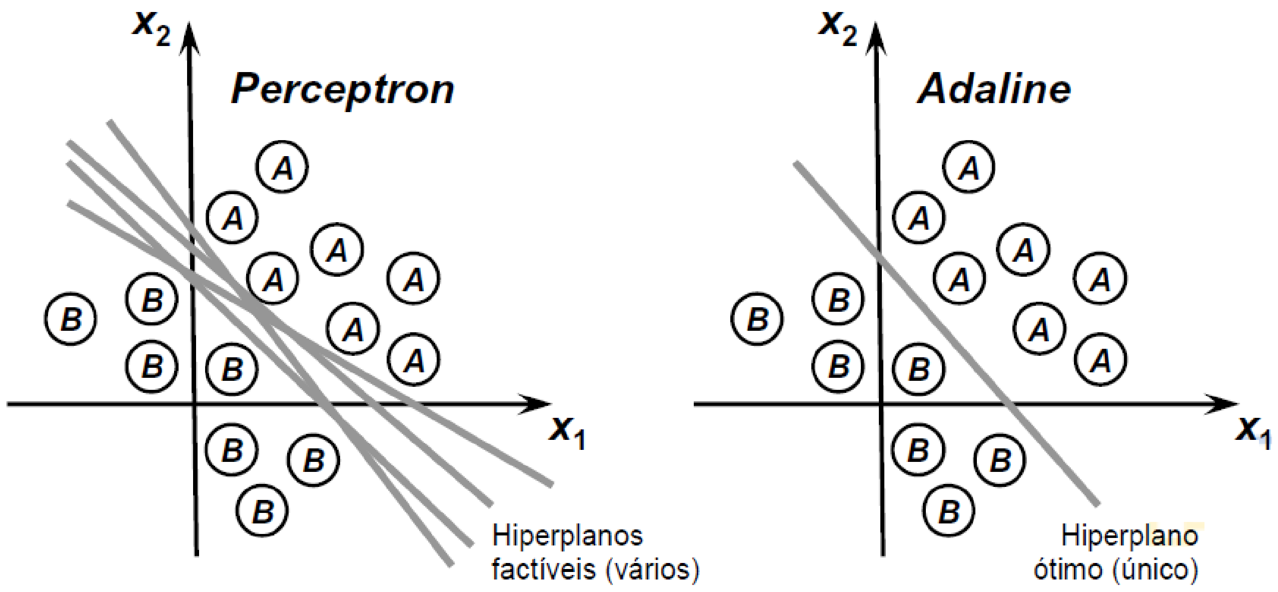
REDE *ADALINE* – IMPLEMENTAÇÃO COMPUTACIONAL

- **Comportamento da Função Erro Quadrático:**
  - Para fins de interpretação, a figura mostra o comportamento do erro quadrático médio (E_{qm}) em função do número de épocas de treinamento.
    - Constata-se então que a curva do erro quadrático médio para o ***ADALINE*** é sempre descendente.
    - Esta decresce na medida em que as épocas de treinamento vão sendo executadas.
    - Finalmente, estabiliza-se num valor constante quando o ponto de mínimo da função erro quadrático médio for alcançado



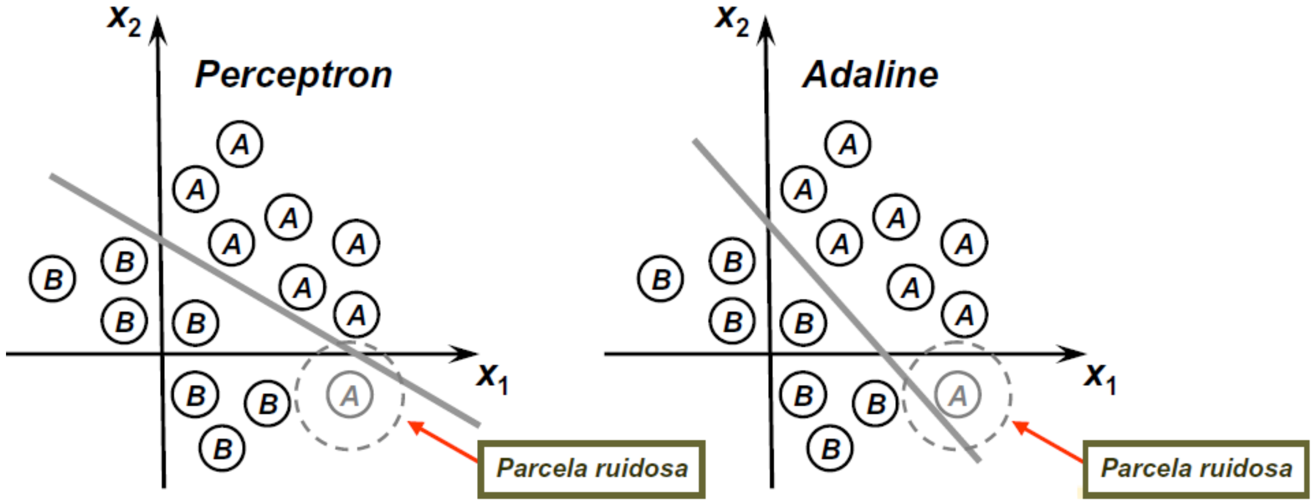
COMPARAÇÃO ENTRE A REDE *ADALINE* E PERCEPTRON

- **Diferenças entre treinamento do *ADALINE* e *Perceptron***
  - ***ADALINE*** - Treinamento usando Regra Delta (Minimização do erro em relação a todas as amostras de treinamento).
    - Independentemente dos valores iniciais atribuídos a ***w***, o hiperplano de separabilidade (após convergência) sempre será o mesmo.
  - ***Perceptron*** - Treinamento usando Regra de Hebb (Avaliação das respostas produzidas após apresentação de cada amostra de treinamento).
    - Quaisquer hiperplanos posicionados dentro da faixa de separabilidade entre as classes são considerados soluções factíveis.



COMPARAÇÃO ENTRE A REDE *ADALINE* E PERCEPTRON

- **Aspectos de Robustez do *ADALINE* e *Perceptron* (Fase de Operação)**
  - ***ADALINE*** - A inclinação do hiperplano é ajustada por intermédio do método dos mínimos quadrados dos erros (processo otimizado).
    - **Maior tendência** de imunidade a eventuais ruídos que podem afetar o processo em que a mesma está mapeando.
  - ***Perceptron*** - O hiperplano que separa as classes pode ter infinitas disposições, pois sua configuração final é fortemente do valores inicial de ***w***.
    - **Menor tendência** de imunidade a eventuais ruídos que podem afetar o processo em que a mesma está mapeando.



- ***ADALINE*** **→** probabilidade menor de classificar a amostra incorretamente.
- ***Perceptron*** **→** probabilidade maior de classificar a amostra incorretamente.

REFERÊNCIA

- SILVA, Ivan Nunes da e SPATTI, Danilo Hernane e FLAUZINO, Rogério Andrade. Redes neurais artificiais para engenharia e ciências aplicadas. São Paulo: Artliber Editora. . Acesso em: 23 dez. 2023. , 2010