Instituto Nacional de Telecomunicações

C210 – Inteligência Computacional

Redes Adaline

Prof. Me. Marcelo Vinícius Cysneiros Aragão
marcelovca90@inatel.br

1º semestre / 2022

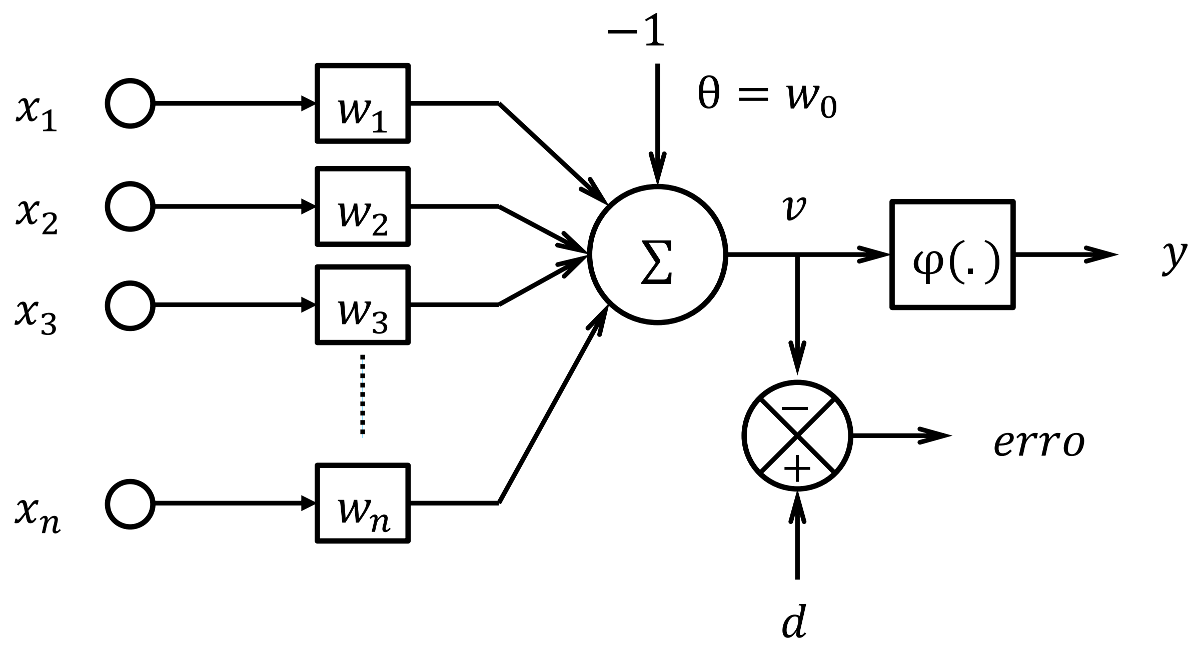
www.inatel.br

Rede *Adaline*

- *Adaline* (*Adaptive Linear Neuron*) foi idealizado por Widrow e Hoff em 1960 e sua principal aplicação se destinava a sistemas de chaveamento de circuitos eletrônicos;
- Apesar de ser uma rede simples, promoveu alguns avanços importantes para o progresso da área de redes neurais:
  - Desenvolvimento do algoritmo de aprendizado regra Delta;
  - Aplicações em diversos problemas práticos envolvendo processamento de sinais analógicos;
  - Primeiras aplicações industriais de redes neurais artificiais.
- Sua grande contribuição foi a introdução da regra Delta, que hoje é utilizado para treinamento de redes *Perceptron* de camadas múltiplas.

Rede *Adaline*, cont.

- Configuração estrutural similar ao *Perceptron*, composto de apenas uma camada de neurônios com arquitetura *feedforward*.



Princípio de funcionamento

- Cada entrada $\{x_j\}$ será ponderada pelos respectivos pesos sinápticos;
- O potencial de ativação $\{v\}$ é computado através da soma das contribuições advindas das multiplicações de todos os sinais x_i por w_i, incluindo o seu limiar $\{\theta\}$.
- A saída $\{y\}$ é a aplicação da função de ativação $\varphi(\mathrm{v})$, representada tipicamente pela função degrau ou função bipolar.
- A saída $erro$ é utilizada no processo de treinamento da rede;

$$v = \sum_{i=1}^{n} w_i . x_i - \theta \leftrightarrow v = \sum_{i=0}^{n} w_i . x_i$$

$$y = \varphi(v)$$

$$erro = (d - v)$$

Treinamento do *Adaline*

- O processo de ajuste dos pesos e limiar do *Adaline* é baseado na regra Delta, também conhecido como algoritmo LMS (*least mean squares*) ou método do Gradiente Descendente;
- Assumindo-se p amostras de treinamento, a ideia básica está em minimizar a diferença entre a saída desejada $\{d\}$ e a resposta do combinador linear $\{v\}$, considerando todas as amostras;
- O critério utilizado é a minimização do erro quadrático médio entre v e d ajustando o vetor de pesos sinápticos $w = [\theta\ w_1 w_2 \ldots w_n]^T$ da rede;

Treinamento do *Adaline, cont.*

- O objetivo é encontrar o w^* ótimo para o qual o erro quadrático $\{E(w^*)\}$ seja mínimo para todo o conjunto de amostras;

$$E(w^*) \leq E(w), para \; \forall \; w \; \in R^{n+1}$$

- A função do erro quadrático médio em relação às p amostras é definida por:

$$E(w) = \frac{1}{p}\sum_{k=1}^{p}(d^{(k)} - v)^2 \; = \frac{1}{p}\sum_{k=1}^{p}(d^{(k)} - (\sum_{i=0}^{n} w_i . {x_i}^{(k)}))^2$$

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

- Como minimizar a função do erro $E(w)$?

Instituto Nacional de Telecomunicações

www.inatel.br

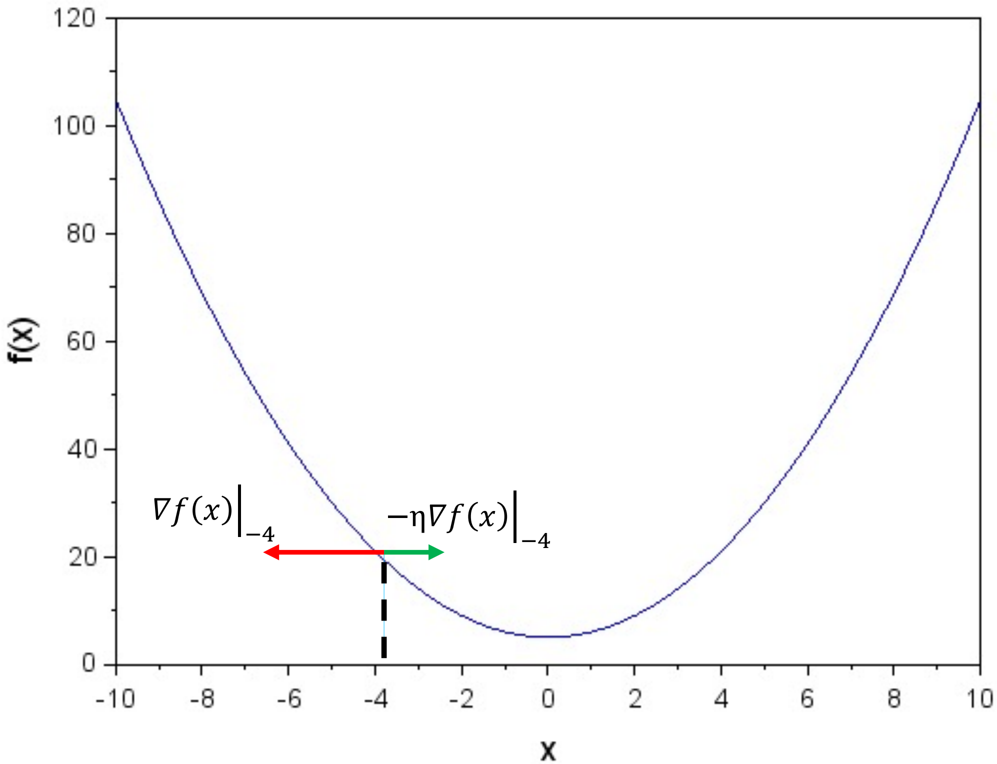
Gradiente

- Como minimizar o erro $E(w)$?

$$f(x) = x^2 + 5 \rightarrow \nabla f(x) = 2x$$

$$\begin{cases} x = -4 \rightarrow \nabla f(x) = -8 \\ x = -3 \rightarrow \nabla f(x) = -6 \\ x = -2 \rightarrow \nabla f(x) = -4 \\ x = -1 \rightarrow \nabla f(x) = -2 \\ x = \ \ \ 0 \rightarrow \nabla f(x) = \ \ \ 0 \\ x = +1 \rightarrow \nabla f(x) = +2 \\ x = +2 \rightarrow \nabla f(x) = +4 \\ x = +3 \rightarrow \nabla f(x) = +6 \\ x = +4 \rightarrow \nabla f(x) = +8 \end{cases}$$

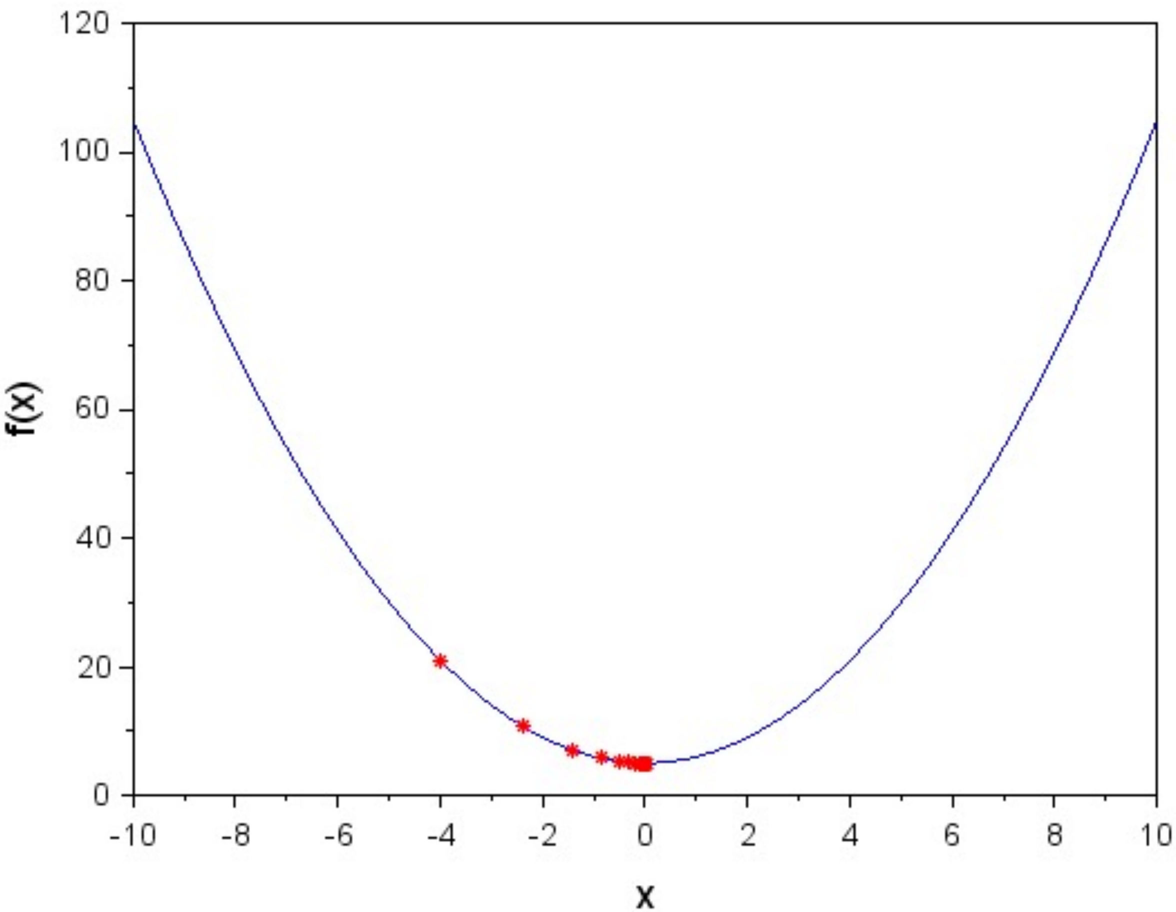
$$\nabla f(x^*) = 0 \rightarrow x^* = (0{,}0)$$



- Partindo de $x = -4$, qual o ajuste a ser feito em x para minimizar $f(x)$?

Gradiente, cont.

$$x(k+1) = x + \eta.\nabla f(x(k)) \rightarrow f(x(k+1)) < f(x(k))$$



Regra Delta

- O vetor gradiente aponta para a direção e sentido de maior crescimento da função de custo (para uma função de ativação linear);

$$E(w) = \frac{1}{p}\sum_{k=1}^{p}\left(d^{(k)} - \left(w^T * x^{(k)}\right)\right)^2$$

$$\nabla E(w) = \frac{\partial E(w)}{\partial w} = \sum_{k=1}^{p}\left(d^{(k)} - \left(w^T * x^{(k)}\right)\right).\left(-x^{(k)}\right)$$

$$\nabla E(w) = -\sum_{k=1}^{p}\left(d^{(k)} - v\right).\left(x^{(k)}\right)$$

- O ajuste dos pesos deve considerar a mesma direção e o sentido contrário ao do vetor gradiente da função de custo $E(w)$ porque o objetivo é minimizar o erro quadrático médio.

Regra Delta, cont.

- O ajuste Δw no vetor de pesos sinápticos é dada por:

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

$$\Delta w = \eta . \sum_{k=1}^{p} \left(d^{(k)} - v\right) . \left(x^{(k)}\right)$$

- A partir do ajuste Δw, atualiza-se o vetor w:

$$w^{atual} = w^{anterior} + \eta . \sum_{k=1}^{p} \left(d^{(k)} - v\right) . \left(x^{(k)}\right)$$

- O ajuste também pode ser realizado após a apresentação de cada k-ésima amostra de treinamento:

$$w^{atual} = w^{anterior} + \eta . \left(d^{(k)} - v\right) . \left(x^{(k)}\right), onde\ k = 1, \dots, p$$

Regra Delta, cont.

- Assim como no *Perceptron*, a taxa de aprendizagem $\{\eta\}$ exprime o quão rápido o processo de treinamento da rede estará rumando em direção ao ponto de minimização da função de erro quadrático médio;

- Normalmente adotam-se valores pertencentes ao intervalo compreendido em $0 < \eta < 1$, comumente expressos em potências de 10. Exemplos:
  - 0,1 → 10^{-1} ou 1E-1, que significa $1{,}0 \times 10^{-1}$
  - 0,01 → 10^{-2} ou 1E-2, que significa $1{,}0 \times 10^{-2}$
  - 0,001 → 10^{-3} ou 1E-3, que significa $1{,}0 \times 10^{-3}$
  - E assim em diante.

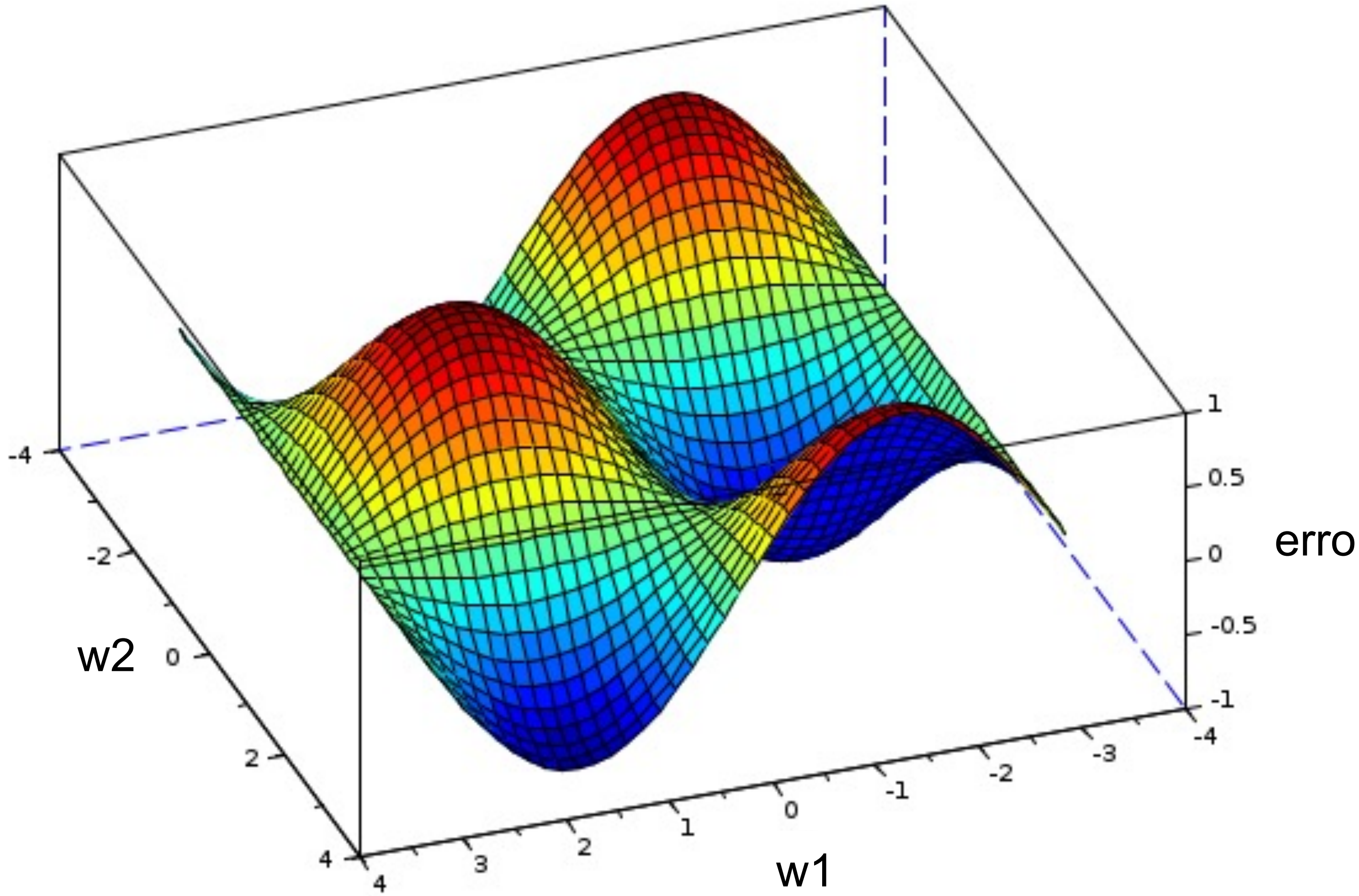
Superfícies de erro

- O conjunto dos $m+1$ pesos a serem ajustados em uma rede neural pode ser visto como um ponto em um espaço $(m+1)$-dimensional, conhecido como espaço de pesos;

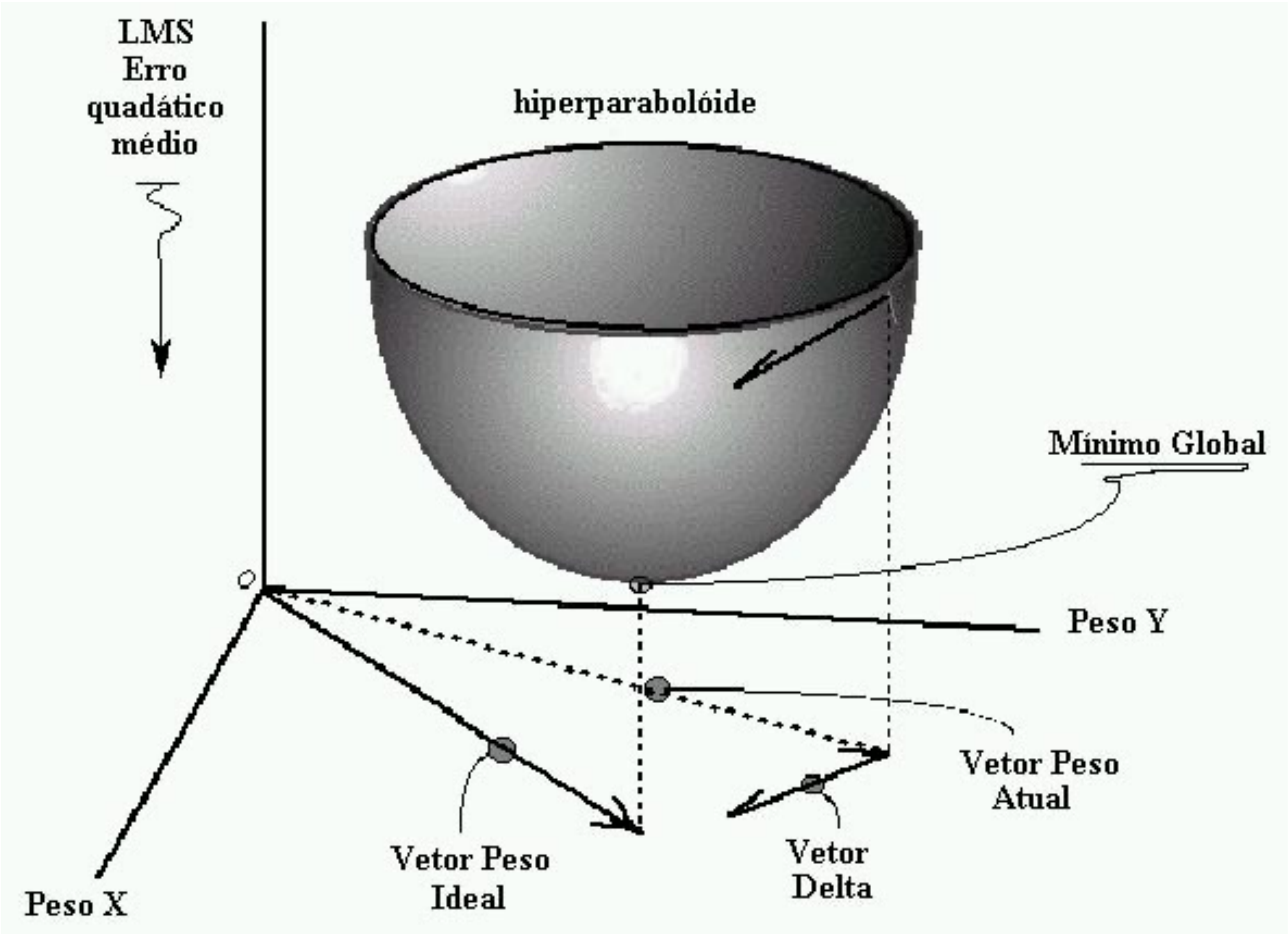
- Pode-se imaginar que cada conjunto de pesos apresenta um valor associado de erro para cada amostra de entrada e também para todo o conjunto de treinamento;

- Os valores de erro para todos os conjuntos possíveis de pesos definem uma superfície no espaço de pesos – a superfície de erro;

Superfícies de erro, cont.



Superfícies de erro, cont.

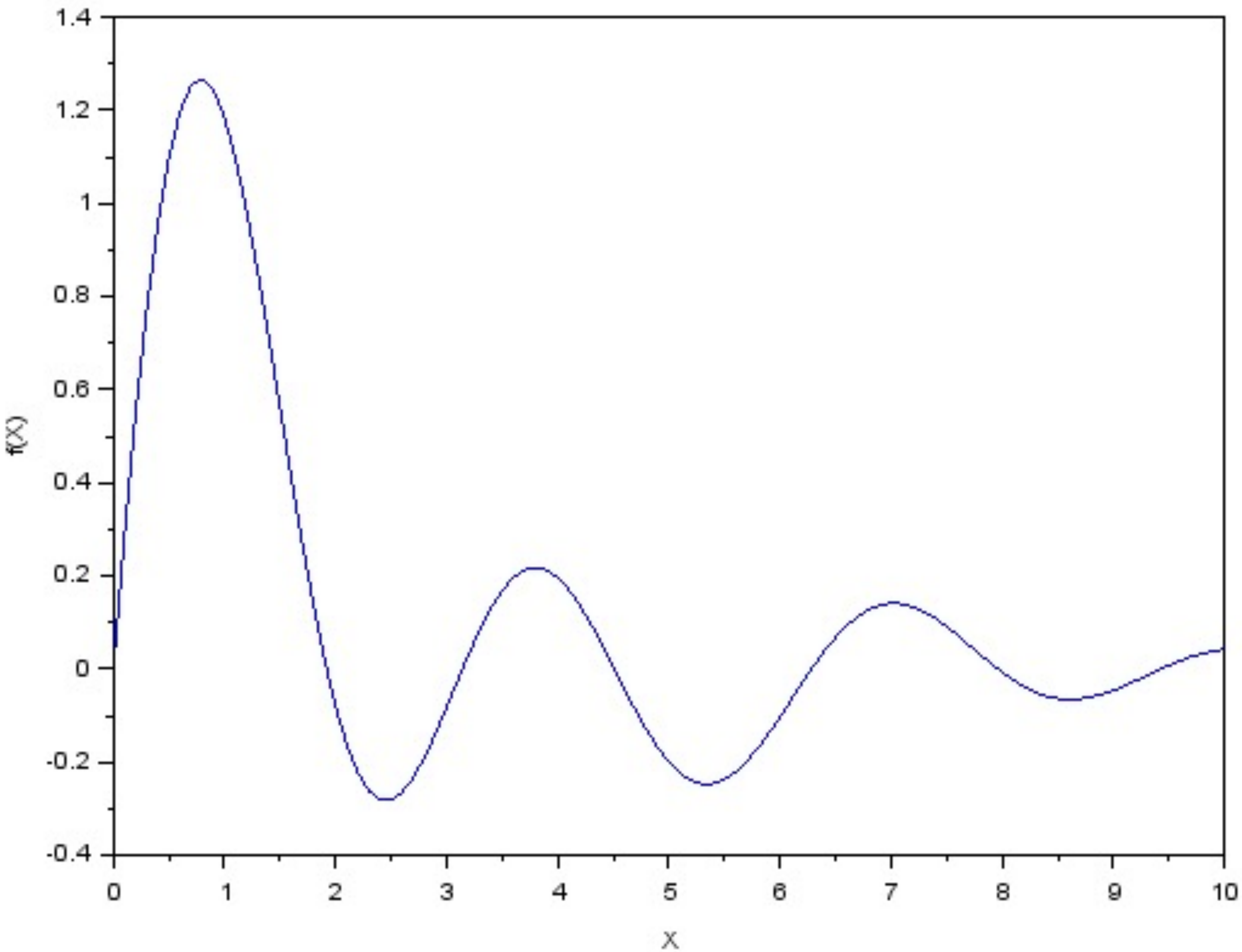


Superfícies de erro, cont.

- O algoritmo de treinamento baseado na regra Delta, que é um algoritmo supervisionado, opera com base na minimização de uma função de custo, que neste caso é baseada no erro entre o resultado do combinador linear e as saídas desejadas.
- Observações:
  - ✓ A superfície de erro possui uma grande quantidade de mínimos locais;
  - ✓ Pode-se encontrar situações onde o método do gradiente não tem solução (descontinuidades);
  - ✓ Outros métodos de busca podem ser utilizados.

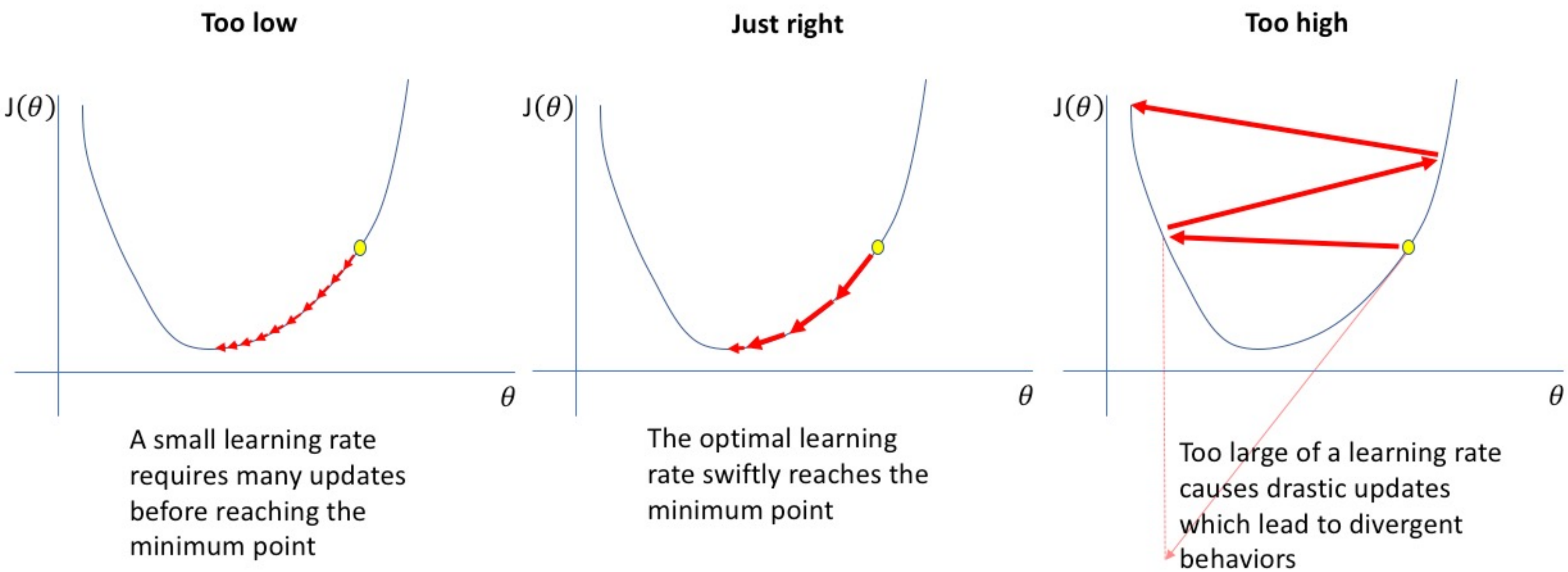
Sobre a Taxa de Aprendizagem

Como uma escolha errada da taxa de aprendizagem pode afetar o treinamento?



Sobre a Taxa de Aprendizagem

η muito baixa η correta η muito alta



Fonte: https://www.jeremyjordan.me/nn-learning-rate/

Algoritmo *Adaline* - treinamento

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

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

3) Iniciar o vetor w com valores aleatórios pequenos;

4) Especificar a taxa de aprendizagem $\{\eta\}$ e precisão requerida $\{\varepsilon\}$;

5) Iniciar o contador de número de épocas $\{\acute{e}pocas \leftarrow 0\}$;

6) Repetir as instruções:

6.1) $E_{qm}{}^{anterior} \leftarrow E_{qm}(w)$

6.2) Para todos pares de treinamento $\{x^{(k)}, d^{(k)}\}$, faça:

$$\begin{cases} \text{6.2.1) } v \leftarrow w^T * x^k; \\ \text{6.2.2) } w \leftarrow w + \eta * \left(d^{(k)} - v\right) * x^{(k)}; \end{cases}$$

6.3) $\acute{e}poca \leftarrow \acute{e}poca + 1$;

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

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

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 todos pares de treinamento $\{x^{(k)}, d^{(k)}\}$, faça:

3.1) $v \leftarrow w^T * x^k$;

3.2)$E_{qm} \leftarrow E_{qm} + \left(d^{(k)} - v\right)^2$;

4) $E_{qm} \leftarrow \dfrac{E_{qm}}{p}$;

Algoritmo *Adaline* - operação

1) Obter a amostra a ser classificada $\{x\}$;

2) Utilizar o vetor w ajustado durante o treinamento;

3) Executar as seguintes instruções:

3.1) $v \leftarrow w^T * x$;

3.2) $y \leftarrow$ degrau(v); (sign no Matlab)

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 $\boldsymbol{x} \in \{Classe\ B\}$

Exemplo de treinamento

- Supondo um problema a ser mapeado pelo *Adaline* com duas entradas $\{x_1, x_2\}$;
- Para um conjunto de quatro amostras de treinamento constituídas dos seguintes valores:

$$\Omega^{(x)} = \{[2.0\ 3.5]; [6.8\ 5.3]; [2.0\ 2.5]; [8.1\ 4.2]\}.$$

- Considerando-se ainda que os respectivos valores de saída para cada uma das amostras seja dado por:

$$\Omega^{(d)} = \{[-1]; [1]; [-1]; [1]\}.$$

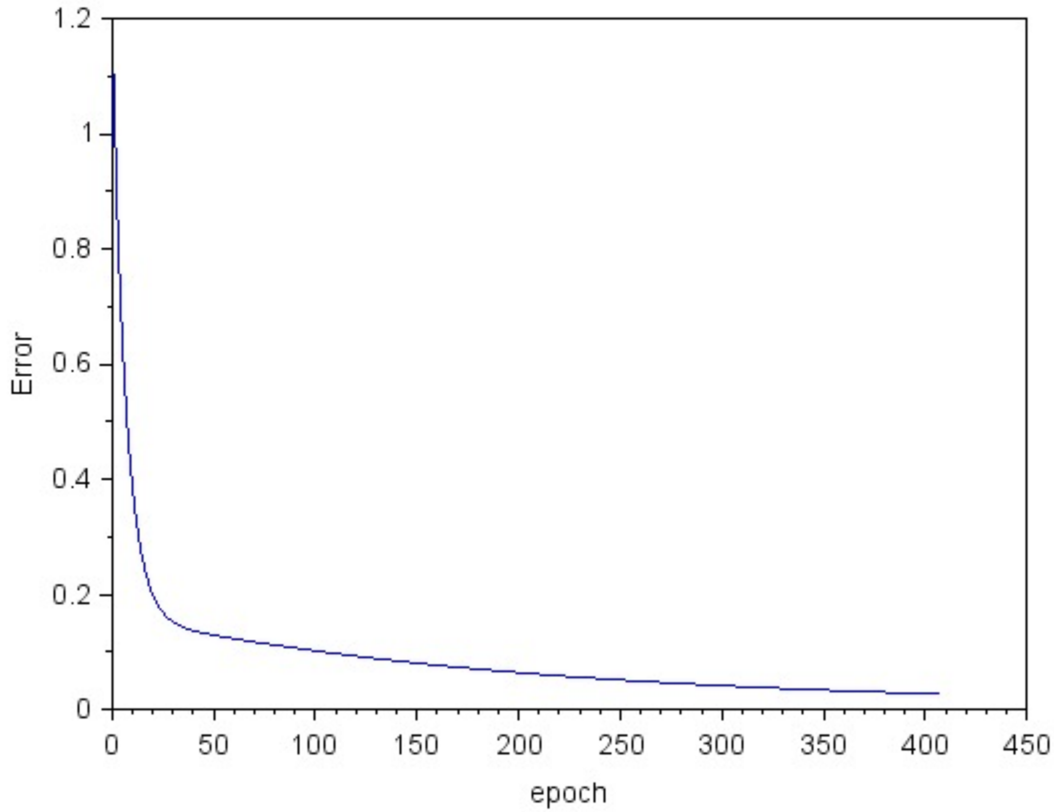
- Escolhendo aleatoriamente os pesos sinápticos iniciais:

$$w = \{0.36; 0.29; 0.57\}.$$

$$\Omega^{(x)} = \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{cccc} x^{(1)} & x^{(2)} & x^{(3)} & x^{(4)} \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 \\ 2.0 & 6.8 & 2.0 & 8.1 \\ 3.5 & 5.3 & 2.5 & 4.2 \end{bmatrix} \end{array} \qquad \Omega^{(d)} = \begin{array}{c} \begin{array}{cccc} d^{(1)} & d^{(2)} & d^{(3)} & d^{(4)} \end{array} \\ \begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix} \end{array}$$

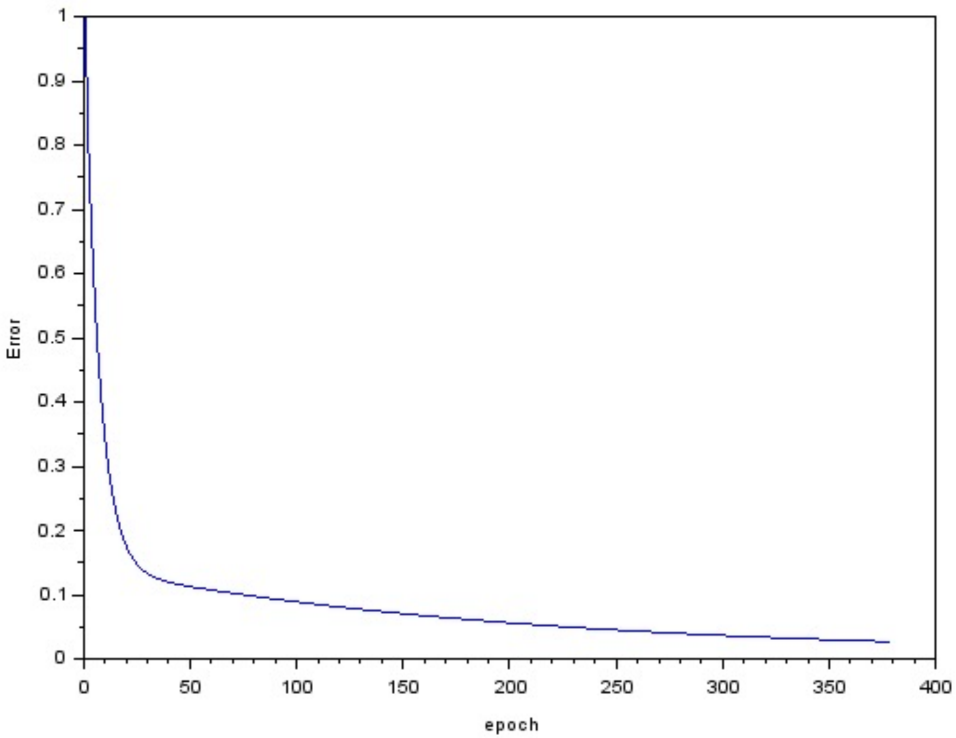
Exemplo de treinamento, cont.

- Após 407 épocas de treinamento:
  - ✓ $w = \{1.71; 0.30;\ 0.07\}$



Exemplo de treinamento, cont.

- Usando o mesmo problema anterior porém escolhendo outros valores, aleatoriamente, para os pesos sinápticos iniciais, obtém-se após 379 épocas de treinamento:
  - ✓ $w = \{1.71; 0.30; 0.07\}$



Aspectos práticos sobre o *Adaline*

- O processo de treinamento da *Adaline* tende a mover o vetor de pesos sinápticos visando diminuir o erro quadrático médio em relação a todas as amostras de treinamento;
- O processo de convergência da rede caminha o hiperplano de separação sempre em direção à fronteira de separabilidade ótima mesmo começando com vetores iniciais distintos;
- A curva do erro quadrático médio para o *Adaline* é sempre descendente, à medida que as épocas de treinamento são executadas, estabilizando-se num valor constante quando o ponto de mínimo da função de erro quadrático médio é alcançado;

Aspectos práticos sobre o *Adaline*, cont.

- No *Adaline,* o processo de treinamento busca o hiperplano de separabilidade ótimo enquanto no *Perceptron*, qualquer hiperplano dentro da faixa de separabilidade é considerado uma solução;

- O *Adaline* ajusta a inclinação do hiperplano através do método dos mínimos quadrados dos erros (*LMS – least mean squares*);

- Por atingir o hiperplano ótimo para quaisquer valores atribuídos inicialmente a seus pesos, o *Adaline* é uma rede neural com maior imunidade a ruídos.

Aspectos práticos sobre o *Adaline*, cont.

- O valor da taxa de aprendizagem $\{\eta\}$ deve ser cuidadosamente especificado para evitar instabilidades em torno do ponto mínimo e também evitar um processo de convergência extremamente lento;

- O número de épocas de treinamento depende dos pesos sinápticos iniciais atribuídos $\{w\}$ assim como do valor assumido para a taxa de aprendizagem $\{\eta\}$;

- O desempenho do treinamento do *Adaline* pode ser melhorado por intermédio da normalização dos sinais de entrada frente ao domínio apresentado;

Código no GitHub

- Implementação em Java (classe Adaline):

https://github.com/marcelovca90-inatel/EC017/tree/master/NeuralNetworks-java

- Implementação em Python (módulo Adaline):

https://github.com/marcelovca90-inatel/EC017/tree/master/NeuralNetworks-python

Fim

Marcelo Vinícius C. Aragão
marcelovca90@inatel.br