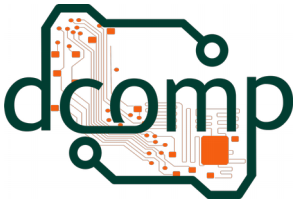
Universidade Federal do Espírito Santo – CCENS UFES

Universidade Federal do Espírito Santo
Centro de Ciências Agrárias – CCENS UFES
Departamento de Computação



Adaline

Redes Neurais Artificiais

Site: http://jeiks.net
E-mail: jacsonrcsilva@gmail.com

Adeline

- Adaline,
  - chamado inicialmente de:

    ADAptive LINear Element;
  - após popularização da regra de aprendizado:

    ADAptive LInear NEuron;
- O modelo Adaline surgiu quase simultaneamente ao Perceptron.
- Surgiram em áreas com enfoque diferente:
  - Frank Rosenblatt divulgou o Perceptron em uma revista de psicologia;
  - Bernard Widrow divulgou o Adaline em uma conferência do IRE (Institute of Radio Eletronics), atualmente IEEE (Institute of Electrical and Electronic Enginers).
- O algoritmo de treinamento de Widrow e Hoff é conhecido como Regra Delta e tem enorme importância em IA, também dando origem ao *backpropagation*.

Adaline

- O modelo Adaline:
  - Utilizado como aproximador de funções;
  - Tem seus pesos adaptados sobre a função de erro da linear do neurônio, ou seja, antes da aplicação da função de adaptação;
  - É gerada uma função de custo quadrática que permite utilização do método do gradiente.

Bernard Widrow

Marcian E. Hoff

Aprendizado do Adaline

- Somente há ajuste dos pesos quando o erro é diferente de zero.
- O algoritmo tenta minimizar o erro das saídas em relação aos valores desejados do treinamento:

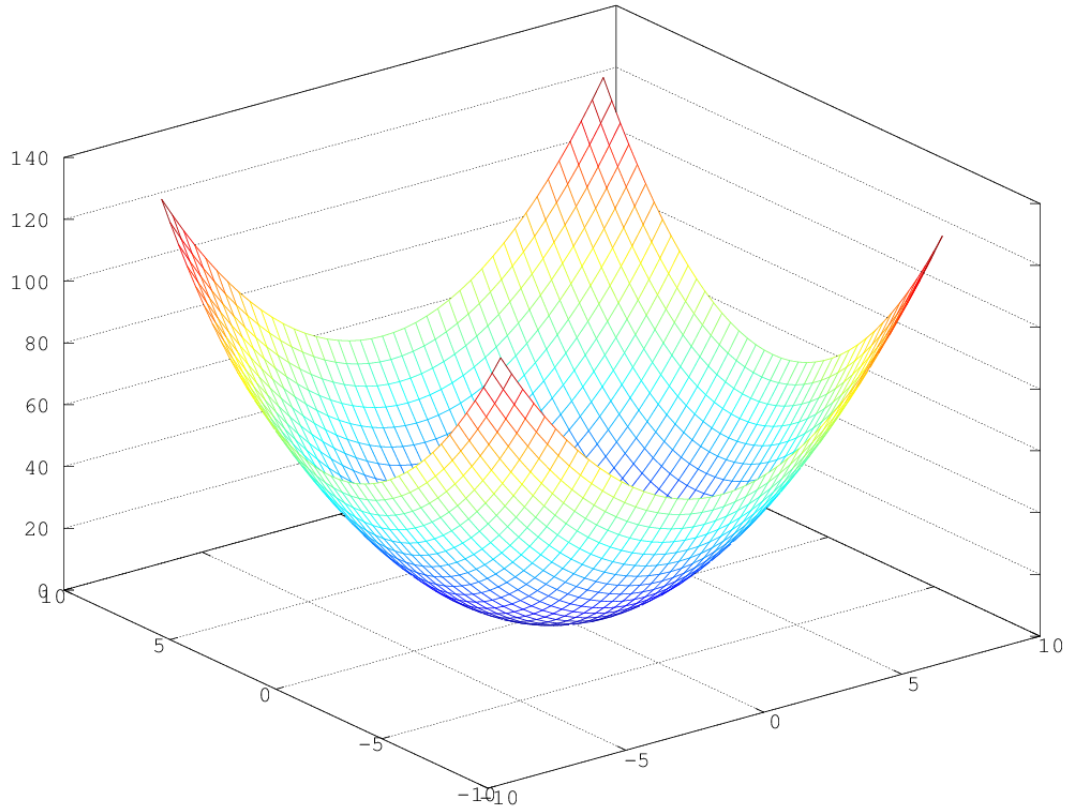
$$\Gamma = \{(x^i, d^i)\}_{i=1}^{p}$$

- A função de custo a ser minimizada é a soma dos erros quadráticos:

$$J = \frac{1}{2}\sum_{i=1}^{p}(d^i - y^i)^2$$

Aprendizado do Adaline

- Para uma condição inicial qualquer w(0):
  - Deseja-se obter a direção do ajuste;
  - O ajuste será aplicado no vetor de pesos;
  - Se a direção for definida corretamente, os pesos caminharão em direção à solução ótima.



Plotando 3D com Octave

```
% linspace(BASE, LIMITE, QNT. ITENS)
% retorna um vetor com QNT. ITENS, de BASE até LIMITE
tx = ty = linspace(-5, 5, 10);
% MESHGRID( X, Y )
% retorna uma malha
[ xx yy ] = meshgrid(tx, ty);
% Aqui calculamos o valor de tz
tz = xx.^2 + yy.^2;
% Agora a função mesh plotará o gráfico:
mesh(tx, ty, tz);
% Salvando:
print -dpng 'grafico.png'
```

Aprendizado do Adaline

- Então, o ajuste de peso deve ser feito em direção contrária ao vetor gradiente no ponto anterior.
- Os componentes do vetor gradiente são:

$$\frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial w_i}$$

$$\frac{\partial J}{\partial w_i} = -x_i\left(d-\left(w_0.1+w_1x_1+w_2x_2+...+w_jx_j+w_dx_d\right)\right)$$

$$\frac{\partial J}{\partial w_i} = -x_i\,erro$$

- Então, para ajustar o vetor de pesos em direção contrária ao vetor gradiente, tem-se:

$$\Delta w_i \propto e\,x_i, \text{onde } e = erro$$

$$\Delta w_i = \eta e\,.\,x_i$$

Regra Delta

- Então, a regra Delta pode ser definida como:

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

$$com \quad \Delta t = \eta e x(t), temos:$$

$$w(t+1) = w(t) + \eta e x(t)$$

Interpretação Gráfica

- No Octave,
  - Crie as entradas:

    $x^1 = 0,2;$

    $x^2 = -1,5;$
  - Crie as saídas desejadas:

    $y^1 = 0,9;$

    $y^2 = 0,3.$
  - Para:

    bias = peso = linspace(-2,2,50);
  - Crie um gráfico 3D para ver o gradiente de erro.