RNA

Redes Neurais Lineares

Prof. Frederico Coelho.

(Baseado no livro de Antônio de Pádua Braga)

Introdução

- Modelos abordados serão caracterizados pela equação

$$\mathbf{Xw = Y} \qquad \text{Ou de maneira mais geral} \qquad \mathbf{f(Xw) = Y}$$

Onde f(.) é a função de ativação do modelo:

- Função identidade $f(u)=u$;
- Função degrau;
- Função logística.

Modelos de única camada

Introdução

São modelos de única camada uma vez que as amostras de entrada, linhas de **X**, são multiplicadas pelo vetor **w** resultando em **Y**.

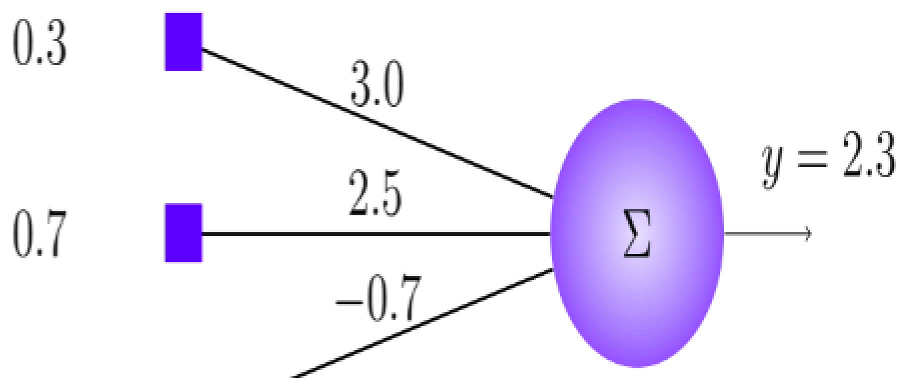
Para o caso de $f(u) = u$, o modelo de camada única será linear

$$\mathbf{X}\mathbf{w} = \mathbf{Y}$$

Introdução

$$\underbrace{\begin{pmatrix} \boxed{0.3 \quad 0.7 \quad 0.5} \\ -1.2 \quad 0.5 \quad 3 \\ 0.4 \quad -1.7 \quad -2.1 \end{pmatrix}}_{\mathbf{x}} \underbrace{\begin{pmatrix} \boxed{\begin{matrix} 3.0 \\ 2.5 \\ -0.7 \end{matrix}} \end{pmatrix}}_{\mathbf{w}} = \underbrace{\begin{pmatrix} \boxed{2.3} \\ -4.45 \\ -1.58 \end{pmatrix}}_{\mathbf{y}}$$

Representação Matricial



$$u = 0.3 \times 3 + 0.7 \times 2.5 + 0.5 \times -0.7$$

Representação Modelo linear de camada únical

Introdução

Dado

$$\tau = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}.$$

Quero encontrar w de forma que

$$f(\mathbf{x}_i, \mathbf{w}) \approx f_g(\mathbf{x}_i)\ \forall \mathbf{x}_i.$$

A aproximação é feita apenas com as amostras conhecidas, mas eu quero que esta função aproxime bem para quaisquer x.

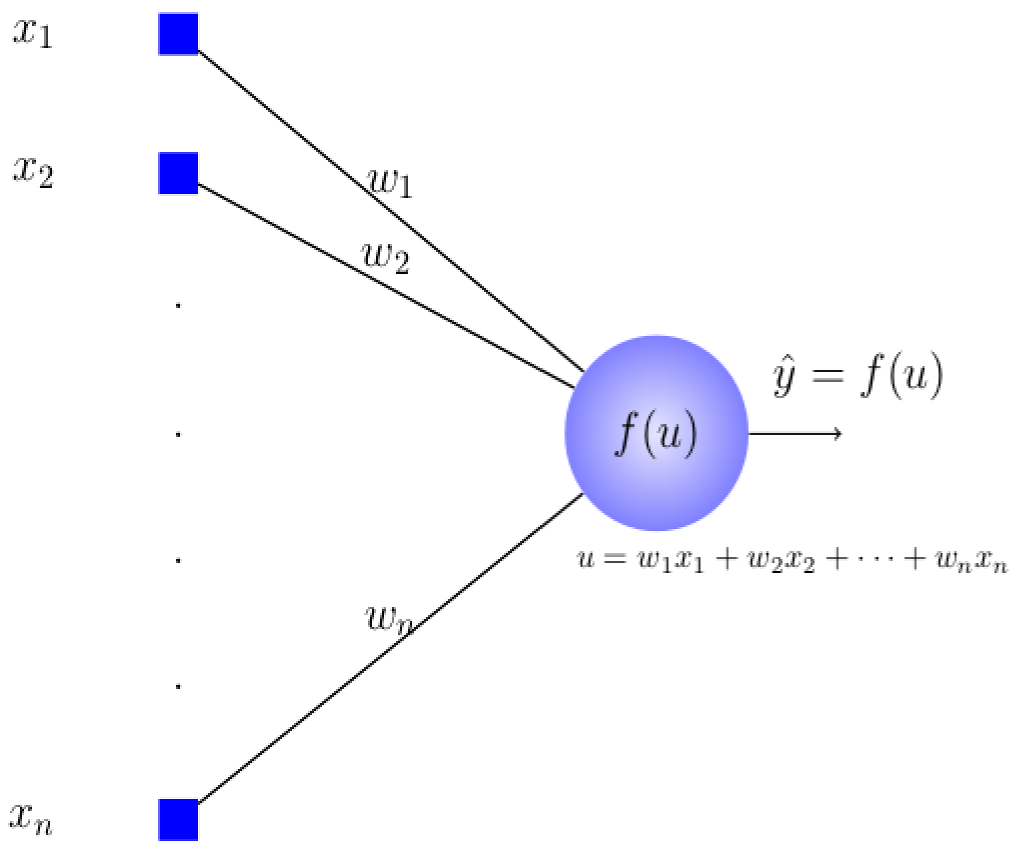
Aprendizado

- Hebbiano
  $W = X^T Y$

- Mínimos quadrados
  $W = H^+ Y$

- Adaline
  Regra Delta

Adaline



Modelo de McCulloch e Pitts:

- Aplicação de função de limiar à soma ponderada das entradas
- No modelo MCP a função de ativação é a degrau
- No Adaline a função de ativação é a função identidade:

f(u) = u

Então a saída corresponde exatamente à soma ponderada das entradas:

$$\hat{y} = \sum w_i x_i$$

Adaline

Treinar o neurônio minimizando uma função de custo:

$$J = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

É uma função quadrática – Facilita busca por mínimo global

Como $\hat{y}_i = \sum_{j=1}^{n} w_j x_{ij}$

A minimização de J é um problema quadrático nos pesos w_j já que a função de custo pode ser escrita como

$$J = \sum_{i=1}^{N} (y_i - \sum_{j=1}^{n} w_j x_{ij})^2$$

Adaline

Se considerarmos um problema de uma única variável o modelo Adaline será caracterizado por:

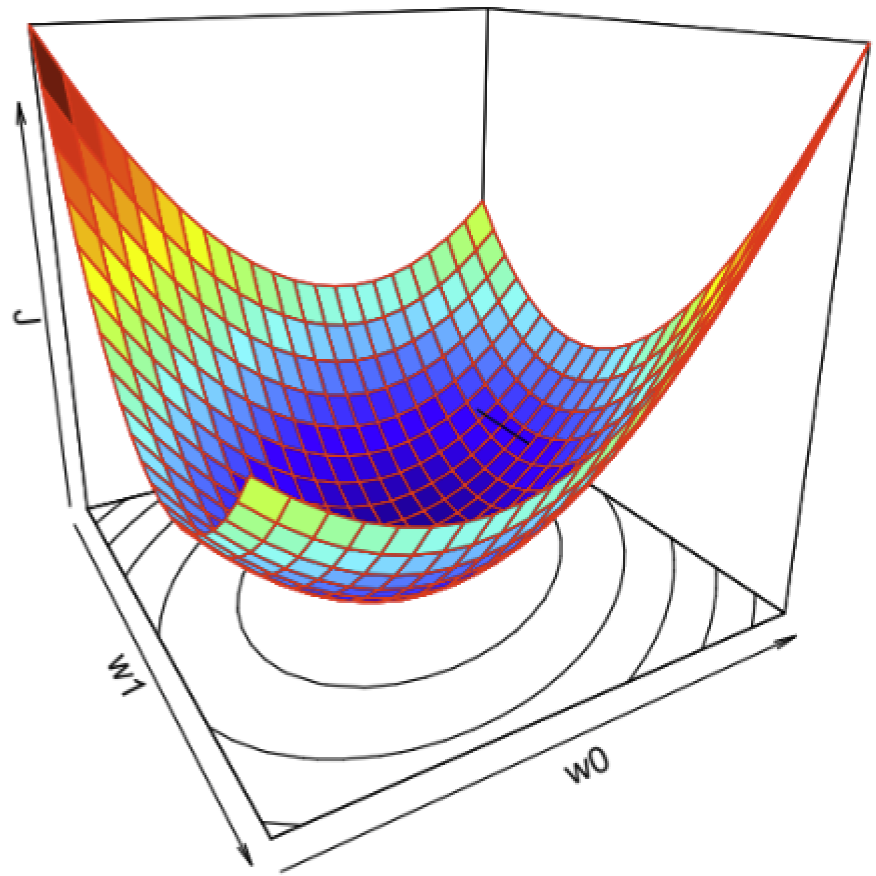
$$\hat{y}_i = w_1 x_i + w_0$$

E a função de custo será:

$$J = \sum_{i=1}^{N} (y_i - (w_1 x_i + w_0))^2$$

Que é quadrática em w_1 e w_0 que são os parâmetros que definem a função.

Adaline

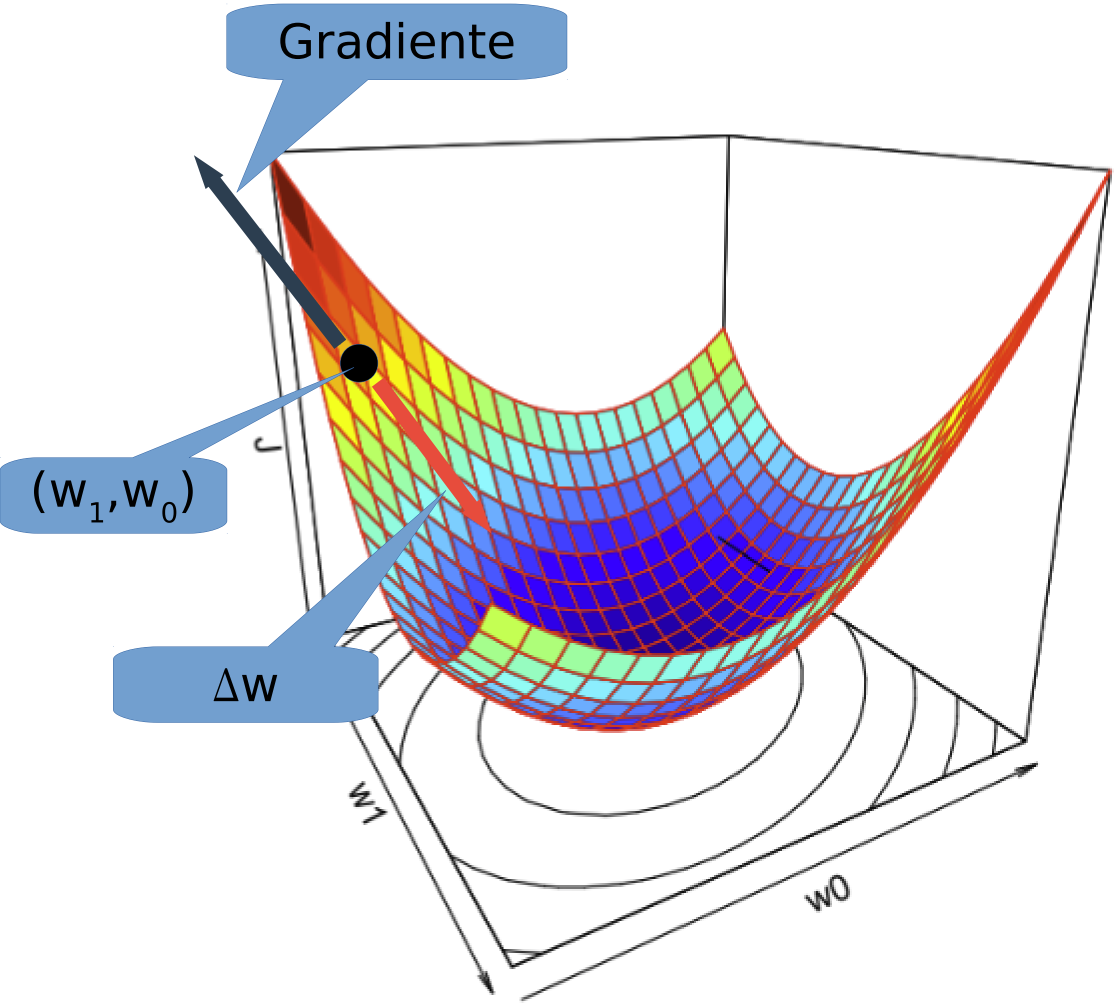


Superfície de erro quadrática correspondente à função de custo considerando:

$$\mathbf{x} = [0.1, 2, -1, 0.2]^T \text{ e } \mathbf{y} = [2.2, 6, 0, 2.4]^T$$

O mínimo da superfície ocorre para W = [2,2]

Adaline – Regra Delta



Deseja-se obter a direção de ajuste do vetor de pesos pelo gradiente descendente, ou seja, o ajuste deve ocorrer em direção contrária ao gradiente em cada ponto da supefície J.

Adaline – Regra Delta

$$\frac{\partial J}{\partial w_j} = \frac{1}{2}\frac{\partial J}{\partial w_j}\left(y_i - \sum_{k=1}^{n} w_k x_{ik}\right)^2$$

$$\frac{\partial J}{\partial w_j} = -\overbrace{\left(y_i - \sum_{k=1}^{n} w_k x_{ik}\right)}^{erro} x_{ij}$$

$$\Delta \mathbf{w}_j \propto -\frac{\partial J}{\partial w_j} \qquad \Delta \mathbf{w}_j = \eta e_i x_{ij}$$

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta e_i \mathbf{x}_i$$

Adaline – algoritmo de treinamento

Exemplo de ajuste de pesos para um neurônio linear simples de três entradas x_1, x_2 e x_3.

Dados:

$(x_1,y_1) = ([0.4, 1.2, 1]^T , 4,64)$

$(x_2,y_2) = ([1.35, 0.7, 1]^T , 2.26)$

$W_{atual} = [-0.21, 3.35, 0.5]^T$

H = 0.1

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta e_i \mathbf{x}_i$$

$$\begin{pmatrix} -0.20184 \\ 3.37448 \\ 0.52040 \end{pmatrix} = \begin{pmatrix} -0.21 \\ 3.35 \\ 0.5 \end{pmatrix} + 0.1(4.640 - 4.436) \begin{pmatrix} 0.4 \\ 1.2 \\ 1 \end{pmatrix}$$

Adaline – algoritmo de treinamento

Algoritmo 1 Algoritmo de treinamento do Adaline.

função TRAINADALINE
  Entrada: X,y,η,tol,maxepocas
  Saída: $\mathbf{w}$,erro por época
  $N \leftarrow$ Número de linhas de $\mathbf{X}$
  Inicializa vetor de pesos $\mathbf{w}$
  enquanto (erroepoca > tol) **e** (nepocas < maxepocas) **faça**
    vetor xseq = permutação de 1 a N
    para i $\leftarrow$ 1 **até** N **faça**
      iseq = xseq[i]
      $erro = (\text{y[iseq]} - \mathbf{w}^T \mathbf{x}[iseq,])$
      $\mathbf{w} = \mathbf{w} + \eta$ erro $\mathbf{x}$**[iseq,]**
      ei2 = ei2 + $erro^2$
    fim para
    erroepoca[nepocas] = ei2
    nepocas = nepocas + 1
  fim enquanto
fim função

Adaline – algoritmo de treinamento

```
trainadaline <- function(xin,yd,eta,tol,maxepocas,par)
{
  dimxin<-dim(xin)
  N<-dimxin[1]
  n<-dimxin[2]

  if (par==1){
    wt<-as.matrix(runif(n+1)-0.5)
    xin<-cbind(1,xin)
  }
  else wt<-as.matrix(runif(n)-0.5)

  nepocas<-0
  eepoca<-tol+1

  evec<-matrix(nrow=1,ncol=maxepocas)
  while ((nepocas < maxepocas) && (eepoca>tol))
  {
    ei2<-0
    xseq<-sample(N)
    for (i in 1:N)
    {
      irand<-xseq[i]
      yhati<-1.0*((xin[irand,] %*% wt))
      ei<-yd[irand]-yhati
      dw<-eta*ei*xin[irand,]
      wt<-wt+dw
      ei2<-ei2+ei*ei
    }
    nepocas<-nepocas+1
    evec[nepocas]<-ei2/N

    eepoca<-evec[nepocas]
  }
  retlist<-list(wt,evec[1:nepocas])
  return(retlist)
}
```

Adaline - exercício

- **Problema univariado**

Livro Braga