Aula 5: Redes Neurais Artificiais (RNA)

Prof. Sérgio Montazzolli Silva

smsilva@uel.br

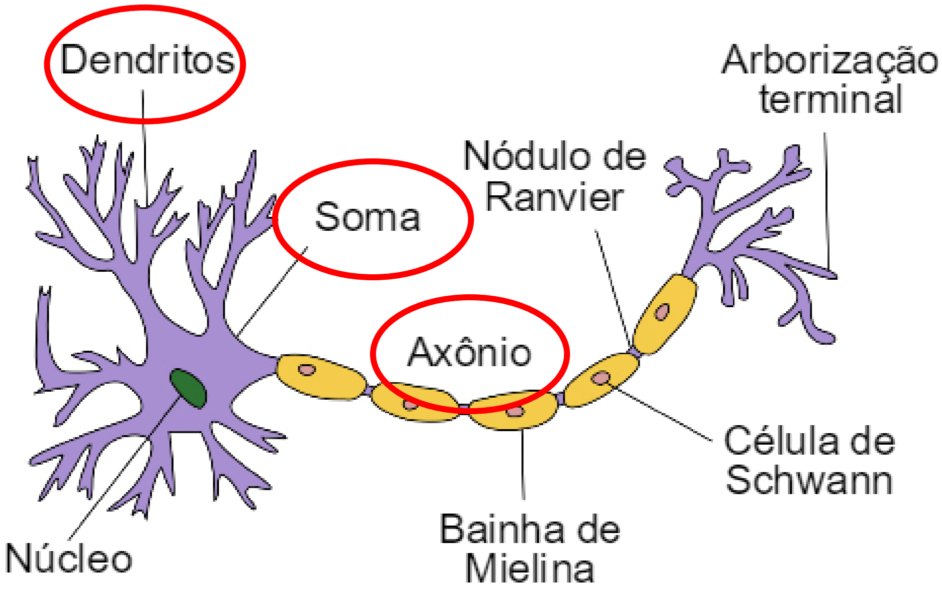
Introdução

- O sistema nervoso humano pode ser visto como um sistema de 3 fases:

- A parte central desse sistema é o cérebro
  - Ele recebe informações, percebe e toma decisões
- As setas para frente (direita) indicam a transmissão de informações através de sinais elétricos
- E as setas para trás (esquerda) indicam o *feedback* recebido pelo sistema

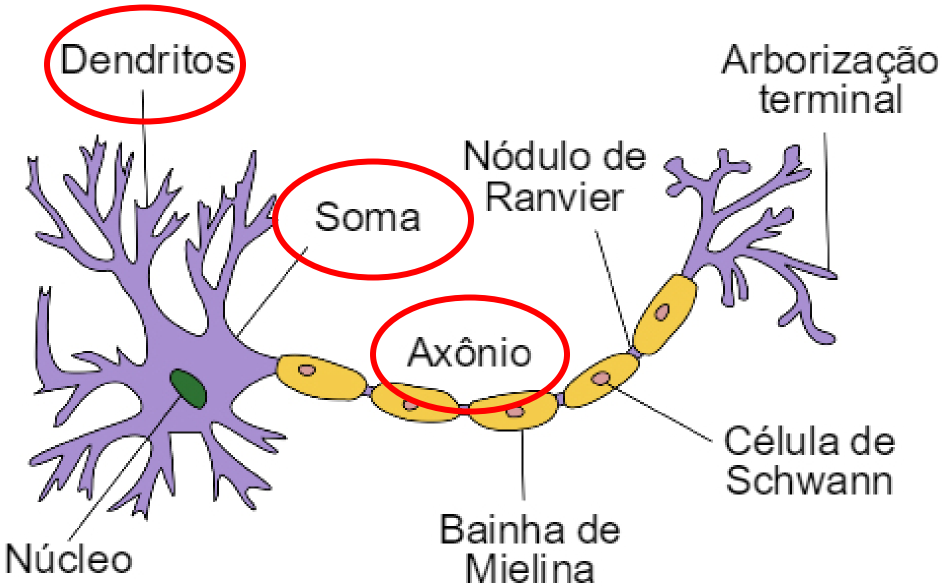
Introdução

- O cérebro é uma grande rede neural composta de aproximadamente 10 bilhões de neurônios que realizam 60 trilhões de conexões
- A figura abaixo mostra um neurônio biológico



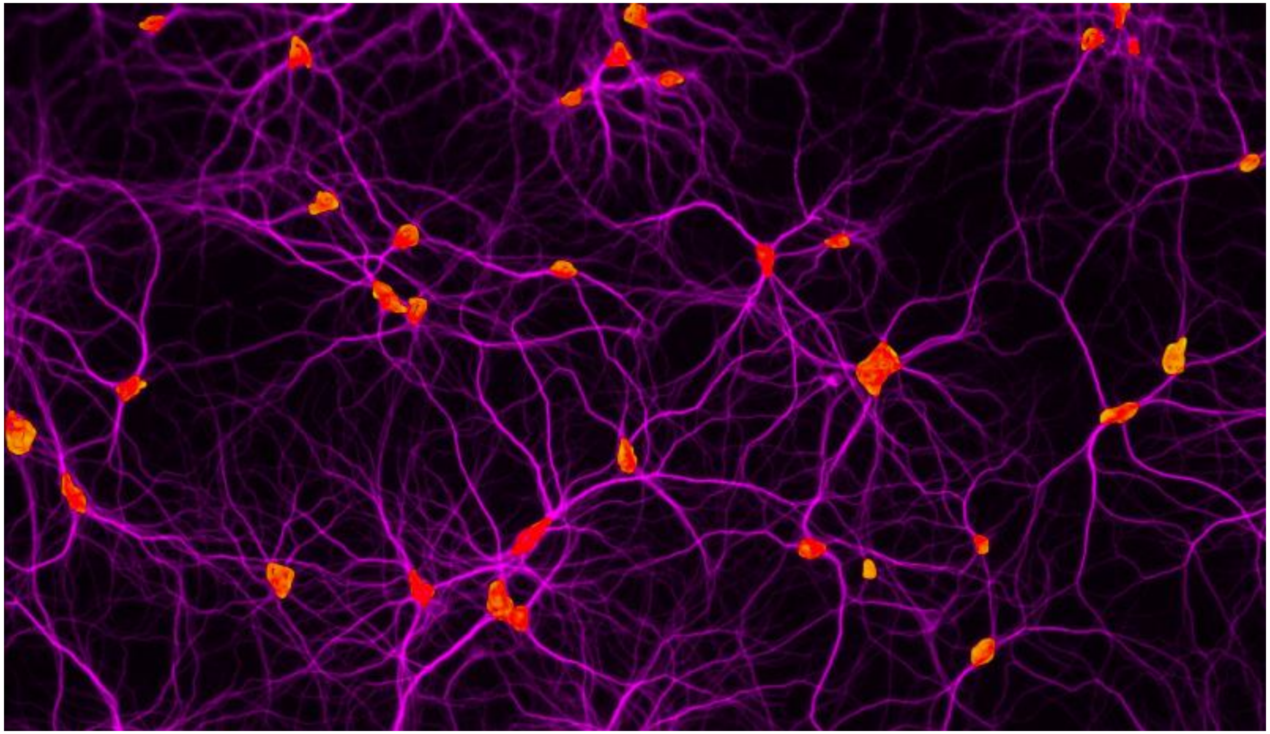
Introdução

- Os **dendritos** são responsáveis por receberem sinais elétricos
- A **Soma** combina de alguma forma estes sinais e os transmite ao axônio
- O **Axônio** por sua vez transmite o sinal de um neurônio para outro neurônio (ou músculo)



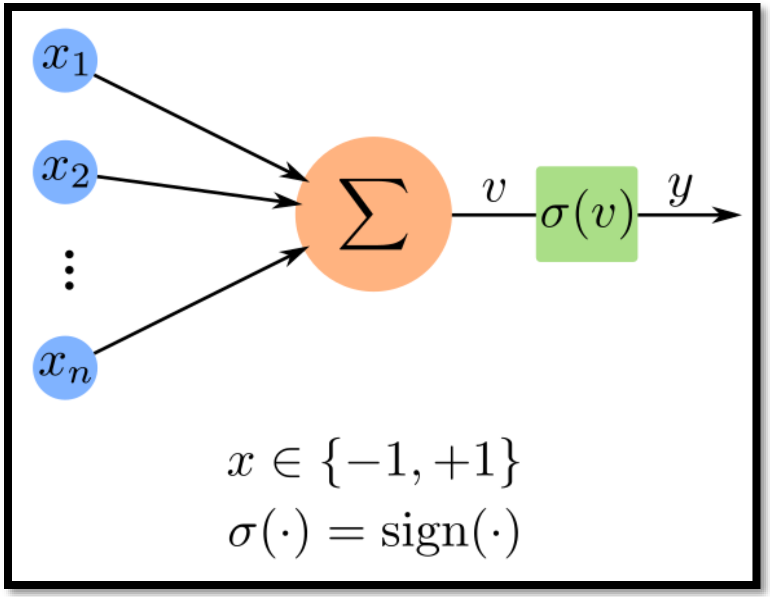
Introdução

- A ligação entre vários neurônios no sistema nerovoso é chamada de rede neural



Modelo Matemático 1

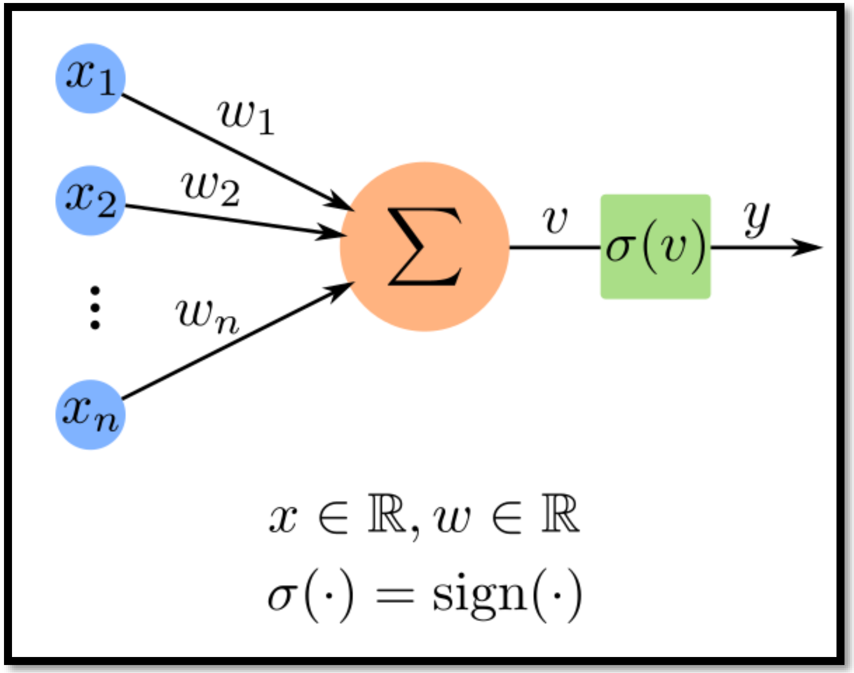
- Neurônio de McCulloch-Pitts (1943)
  - Combina entradas **binárias** através de uma soma
  - Resultado final passa por uma função sinal



$$y = \sigma(x_1 + x_2 + \cdots + x_n)$$

Modelo Matemático 2

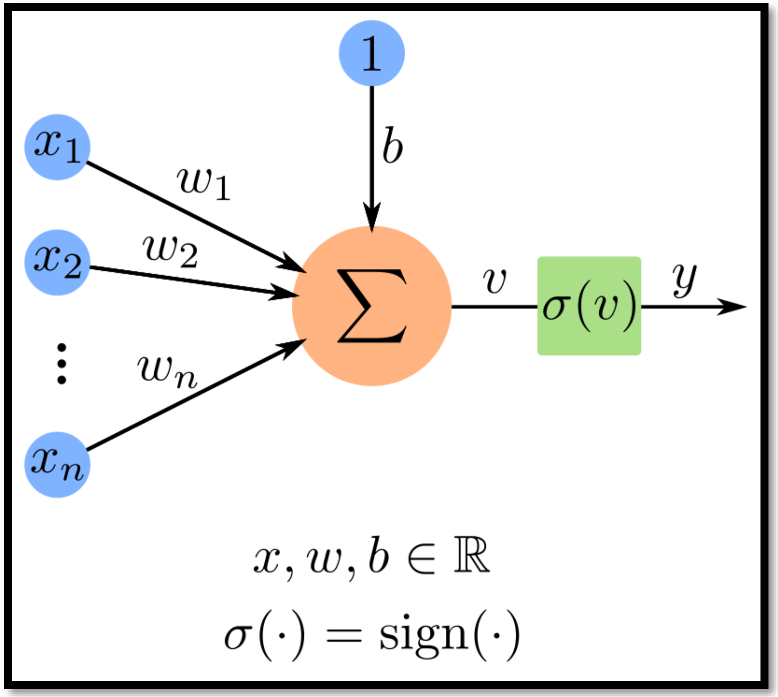
- Neurônio de Rosenblatt (1958)
  - Introdução de pesos $\vec{w}$ para entrada $\vec{x}$



$$y = \sigma(x_1 w_1 + x_2 w_2 + \cdots + x_n w_n)$$

Modelo Matemático 3 (Perceptron)

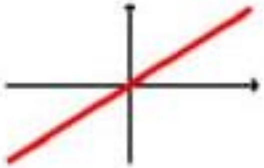
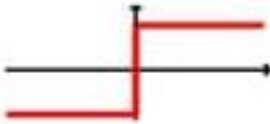
- Neurônio de Widrow-Hoff (1960)
  - Introdução do parâmetro de bias (b)



$$y = \sigma(x_1 w_1 + x_2 w_2 + \cdots + x_n w_n + b)$$

Ativação (sigma)

- Atualmente usa-se diversos tipos de função de ativação no modelo de Widrow-Hoff
  - Notação: na imagem abaixo $\sigma = \phi$ e $v = z$

Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	

Fonte: https://www.simplilearn.com/what-is-perceptron-tutorial

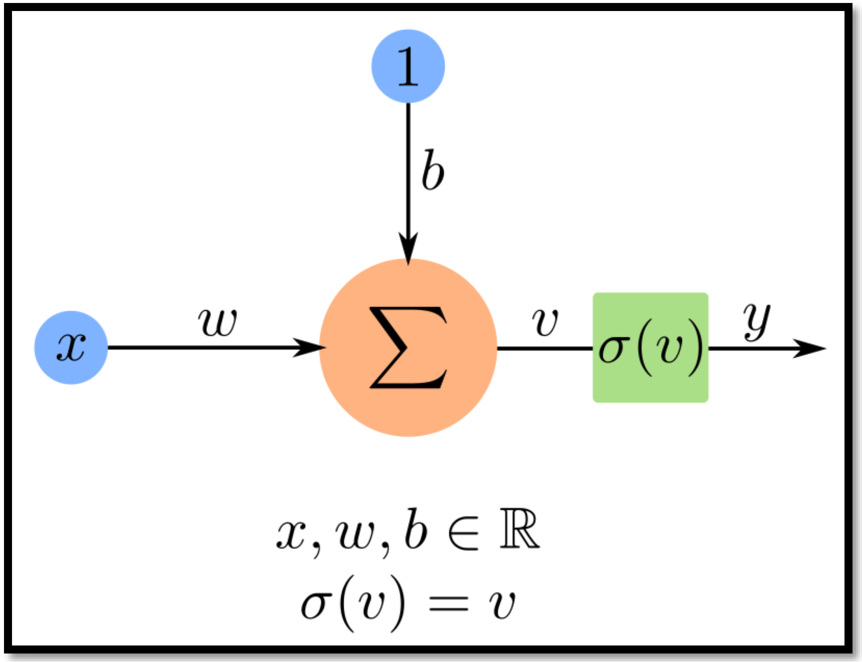
Ativação (sigma)

Piece-wise Linear	$\phi(z)=\begin{cases} 0 & z \le -\frac{1}{2} \\ z + \frac{1}{2} & -\frac{1}{2} \le z \le \frac{1}{2} \\ 1 & z \ge \frac{1}{2} \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z)= \dfrac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z)= \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multilayer NN, RNNs	

Fonte: https://www.simplilearn.com/what-is-perceptron-tutorial

Como representar a equação da reta por um neurônio?

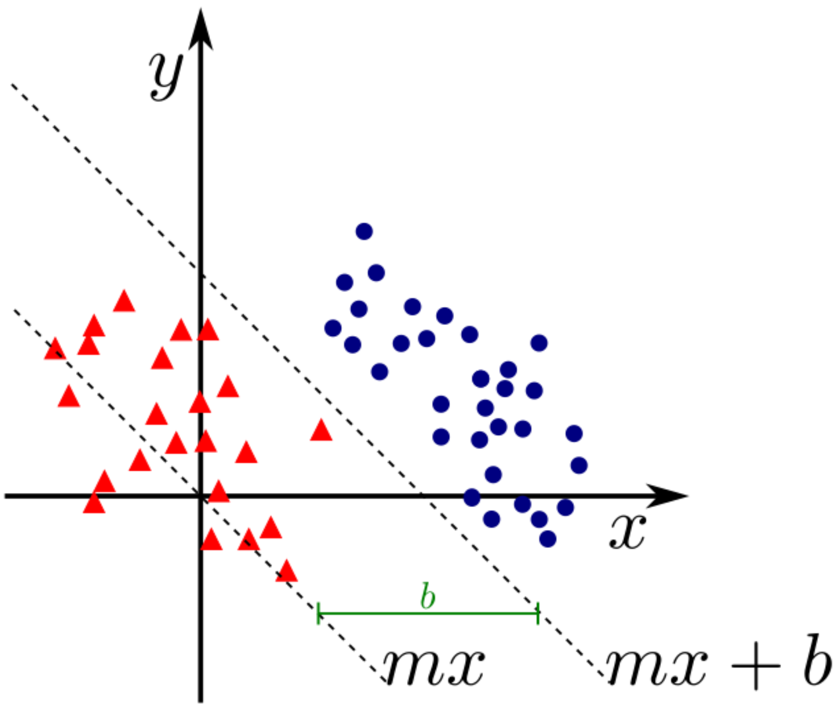
- Considerando $w = m$ na equação $y = xm + b$:



$$y = \sigma(xw + b) = xw + b$$

Parâmetro de bias

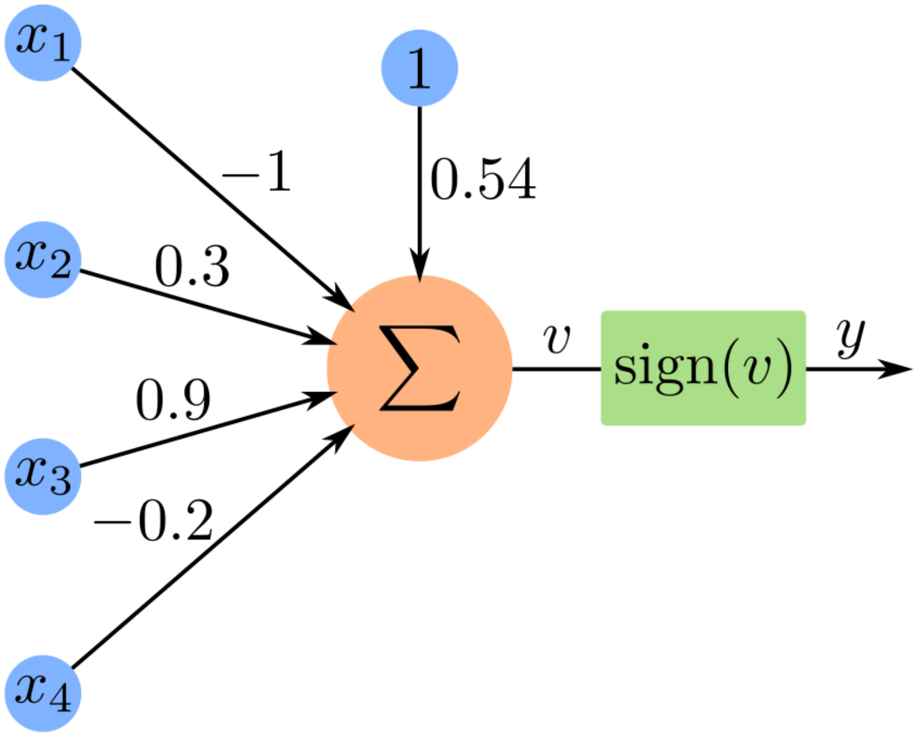
- Com base na equação da reta, podemos inferir que o **bias** tem relação com a **translação** da reta ou plano de separação



- Podemos tratar o bias como um **limiar** para o valor de saída

Exercício em sala

- Qual é a saída do seguinte neurônio?
  - Considere a entrada $\vec{x} = [0.2, 0.9, -1.0, -0.5]$?



Classificação de Cor

- Classificar uma cor entre azulada ou avermelhada
- Um pixel no formato RGB possui 3 valores referentes a intensidade de cada cor
  - A intesidade varia entre 0 e 1
    - 0: cor ausente
    - 1: totalmente presente

RGB
R = red (vemelho)
G = green (verde)
B = blue (azul)

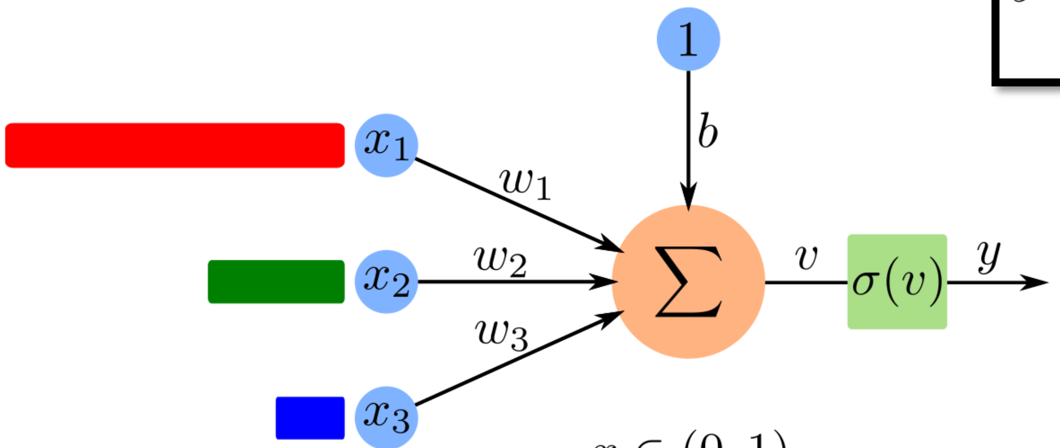
$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0.4 \\ 0.2 \end{bmatrix}$$

Classificação de Cor

- Neurônio:

Saída desejada:

$$y = \begin{cases} 1 & \text{avermelhado} \\ 0 & \text{indefinido} \\ -1 & \text{azulado} \end{cases}$$



$$x \in (0, 1)$$
$$w, b \in \mathbb{R}$$
$$\sigma(\cdot) = \text{sign}(\cdot)$$

Classificação binária

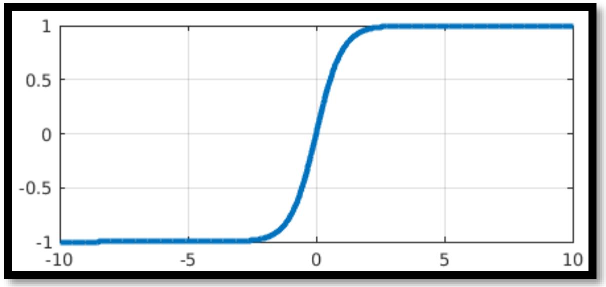
- Se restringirmos nosso estudo a problemas de classificação binária, onde as classes são dadas por valores fixos, como $\{-1, +1\}$ ou $\{0,1\}$
- Podemos forçar nosso modelo (neste tópico: um neurônio) a retornar valores delimitados pelos valores destas classes
- A *função sinal* e a *função de passo* unitário, por exemplo, retornariam exatamente $\{-1, +1\}$ e $\{0,1\}$
- Porém, estas funções possuem **mudanças de valor abruptas** e acabam não sendo adequadas para treinamento

Classificação binária

- As funções *Tangente Hiperbólica* e *Sigmoide*, por sua vez, são delimitadas por $(-1, +1)$ e $(0,1)$, respectivamente
- Elas são suaves e <u>possuem derivadas definidas</u>
- É comum utilizar este tipo de função para problemas de classificação binária
- O valor retornado pode ser entendido com sendo uma distribuição de probabilidade das classes, que diz o grau de confiança para determinado objetivo

Tangente hiperbólica

- Vamos considerar primeiro a Tangente Hiperbólica, que é dada por:



$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- E a sua derivada, que é dada por:

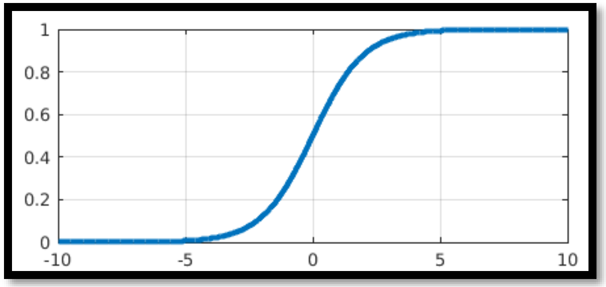
$$\frac{\partial \tanh(x)}{\partial x} = 1 - \tanh^{2}(x)$$

- Em problemas de classificação binária, podemos entender a sua saída como a probabilidade da classe -1, se o valor retornado for negativo, ou a probabilidade da classe $+1$, se o valor retornado for positivo:
  - Se $\tanh(x) < 0$ então $P(cl = -1) = |\tanh(x)|$
  - Se $\tanh(x) > 0$ então $P(cl = +1) = \tanh(x)$

Sigmoide

- Já o Sigmoide é dado por:

$$sigm(x) = \frac{1}{1 + e^{-x}}$$



- E a sua derivada, por:

$$\frac{\partial sigm(x)}{\partial x} = sigm(x)(1 - sigm(x))$$

- Em problemas de classificação binária, podemos entender a sua saída como:
  - Classe 0 então $P(cl = 0) = 1 - sigm(x)$
  - Classe 1 então $P(cl = 1) = sigm(x)$

Aprendizado por correção de erro

- A definição de uma métrica de erro é o ponto de partida para o treinamento de um modelo de classificação

- A partir de agora, considere:
  - $d =$ valor esperado
  - $y =$ saída do seu classificador

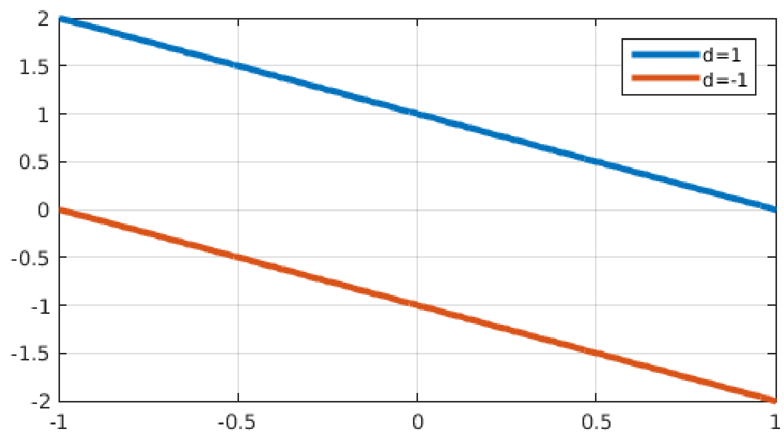
- Uma maneira de definir o erro de classificação poderia ser através da diferença entre o valor esperado e o valor obtido:
  - $e(d, y) = d - y$

Aprendizado por correção de erro

- O problema com $e(d, y) = d - y$:
  - Supondo um problema de classificação binária, onde $d \in \{-1, +1\}$ e representa as classes -1 e $+1$, e $y \in (-1, +1)$
  - Podemos perceber que $e()$ irá assumir valores no intervalo $(-2, +2)$, e $e(d, y)$ somente será zero quando $d = y$
  - Logo, $e()$ não é uma função convexa
  - Se aplicarmos um algoritmo de **minimização** para esta função, a solução que será encontrada tentará fazer com que sempre $\mathrm{e}(d, y) = -2$, ou seja, irá forçar y a ser sempre igual a $+1$

Aprendizado por correção de erro

- Abaixo é mostrado o gráfico $e()$, para $d = 1$ (azul) e $d = -1$ (vermelho)



$e(d = 1, y)$

$e(d = -1, y)$

- Veja que precisamos encontrar métricas de erro que tenham um comportamento convexo

Aprendizado por correção de erro

- Existem várias maneiras de se trabalhar valores de erro para classificação
- Por enquanto veremos duas delas:
  - **Erro quadrático**

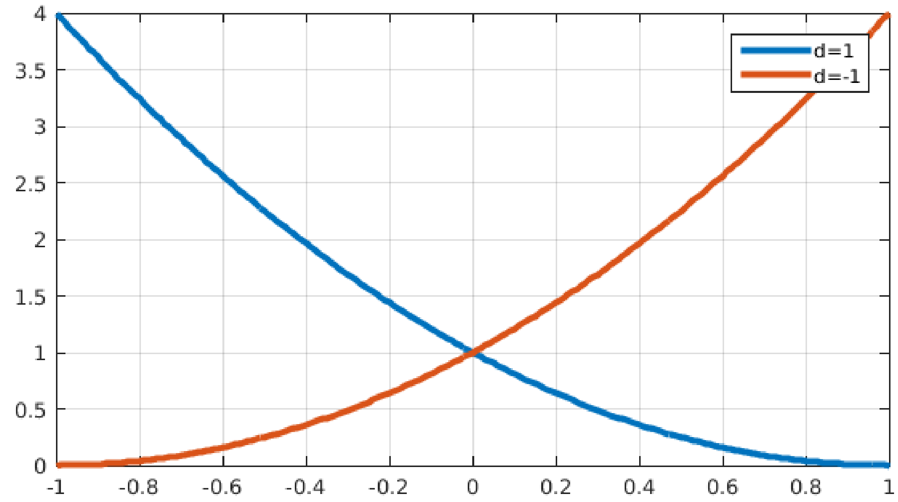
$$e(d, y) = (d - y)^2$$

  - **Erro absoluto**

$$e(d, y) = |d - y|$$

Erro quadrático

- $e(d, y) = (d - y)^2$
- Gráfico:



$e(d = 1, y)$

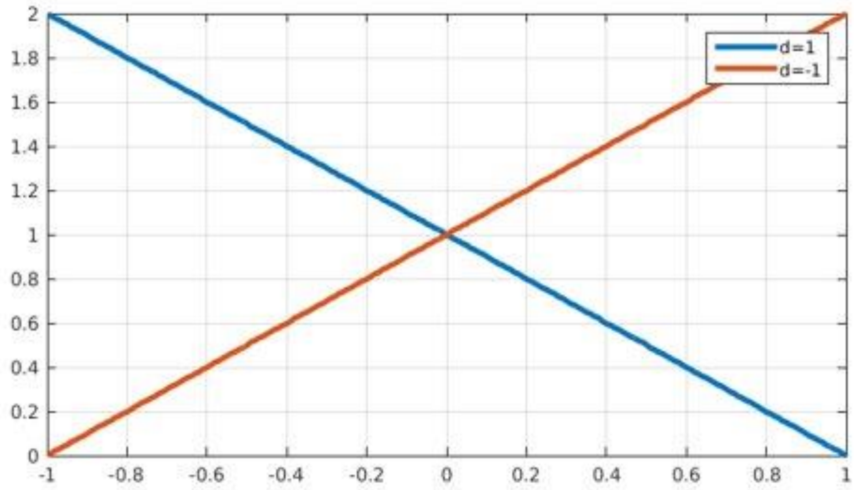
$e(d = -1, y)$

- Derivada:

$$\frac{\partial e(d, y)}{\partial y} = 2y - 2d$$

Erro absoluto

- $e(d, y) = |d - y|$
- Gráfico:



$e(d = 1, y)$

$e(d = -1, y)$

- Derivada:

$$\frac{\partial e(d,y)}{\partial y} = -sign(d - y) = -\left(\frac{d - y}{|d - y|}\right)$$

Erro quadrático e absoluto

- O erro quadrático da maior ênfase ao erro quando y está muito distante de d
- Por isso o erro quadrático, em problemas de minimização, tende a convergir mais rapidamente
- Por outro lado, ele é mais susceptível a *outliers*
- Já o erro absoluto é constante (a notar-se pela derivada) em qualquer parte, o que desacelera a convergência, porém é mais robusto a *outliers*

- *O que é um outlier?*

Definições matemáticas

- Neurônio
  - Entrada: $\vec{x} = \{x_1, x_2, \dots, x_N\}$
  - Parâmetros: $\Theta = \{w_1, w_2, \dots, w_N, b\}$
  - Combinação Linear:
  $$f(\vec{x}) = x_1 w_1 + x_2 w_2 + \cdots + x_N w_N + b$$
  - Ativação:
  $$\sigma(f(\vec{x}))$$
- Erro:
$$e\big(d, \sigma(f(\vec{x}))\big)$$

Derivadas parciais

- Neurônio
  - Derivadas parciais em relação aos parâmetros de Θ:

$$\frac{\partial e(d,\sigma(f(\vec{x})))}{\partial w_1}, \quad \ldots \quad \frac{\partial e(d,\sigma(f(\vec{x})))}{\partial w_N}, \quad \frac{\partial e(d,\sigma(f(\vec{x})))}{\partial b},$$

  - Regra da cadeia:

$$\frac{\partial e(d,\sigma(f(\vec{x})))}{\partial \theta} = \frac{\partial e(d,\sigma(f(\vec{x})))}{\cancel{\partial \sigma(f(\vec{x}))}} \frac{\cancel{\partial \sigma(f(\vec{x}))}}{\cancel{\partial f(\vec{x})}} \frac{\cancel{\partial f(\vec{x})}}{\partial \theta}, \forall \theta \in \Theta$$

Derivadas Parciais

Erro

Ativação

Combinação linear

$$\frac{\partial e(d, \sigma(f(\vec{x})))}{\partial \sigma(f(\vec{x}))} \frac{\partial \sigma(f(\vec{x}))}{\partial f(\vec{x})} \frac{\partial f(\vec{x})}{\partial \theta}$$

Derivadas parciais

- Vamos considerar as seguintes funções de erro e ativação:

$$e(d, y) = (d - y)^2 \qquad \sigma(x) = sigm(x)$$

- Inserindo-as na equação do slide anterior, temos:

$$\frac{\partial (d - sigm(f(\vec{x})))^2}{\partial sigm(f(\vec{x}))} \frac{\partial sigm(f(\vec{x}))}{\partial f(\vec{x})} \frac{\partial f(\vec{x})}{\partial \theta}$$

Derivadas parciais

- Sabendo que:

$$\frac{\partial (d-y)^2}{\partial y} = 2y - 2d$$

$$\frac{\partial sigm(x)}{\partial x} = sigm(x)\big(1 - sigm(x)\big)$$

- Podemos substituir:

$$\big(2sigm(f(\vec{x})) - 2d\big)\Big(sigm(f(\vec{x}))\big(1 - sigm(f(\vec{x}))\big)\Big)\frac{\partial f(\vec{x})}{\partial \theta}$$

Derivadas parciais

- Por fim, se:

$$\frac{\partial f(\vec{x})}{\partial w_n} = x_n \qquad \frac{\partial f(\vec{x})}{\partial b} = 1$$

- Então:

$$\frac{\partial e}{\partial w_n} = \left(2sigm\left(f(\vec{x})\right) - 2d\right)\left(sigm\left(f(\vec{x})\right)\left(1 - sigm\left(f(\vec{x})\right)\right)\right)x_n$$

$$\frac{\partial e}{\partial b} = \left(2sigm\left(f(\vec{x})\right) - 2d\right)\left(sigm\left(f(\vec{x})\right)\left(1 - sigm\left(f(\vec{x})\right)\right)\right)1$$

Derivadas parciais

- Observando que $sigm\big(f(\vec{x})\big) = y$, podemos reduzir:

$$\frac{\partial e}{\partial w_n} = (2y - 2d)\big(y(1 - y)\big)x_n$$

$$\frac{\partial e}{\partial b} = (2y - 2d)\big(y(1 - y)\big)1$$

- Agora podemos utilizar o algoritmo da descida do gradiente, visto anteriormente

Treinamento

- Algoritmo

```
função treinamento(X,D,Ep,λ):
    para cada n em [1,...,N]:
        w_n = numero_aleatório(-1,1)
    b = numero_aleatório(-1,1)

    para cada ep em [1,...,Ep]:
        para cada x,d em X,D :
            y = σ(x_1 w_1 + ⋯ + x_n w_n + b)
            para cada n em [1,...,N]:
                w_n = w_n − (1/|X|) λ · ∂e(d,y)/∂w_n
            b = b − (1/|X|) λ · ∂e(d,y)/∂b

    retorna w_1,…,w_N,b
```

$$y = \sigma(x_1 w_1 + \cdots + x_n w_n + b)$$

$$w_n = w_n - \frac{1}{|X|}\lambda \cdot \frac{\partial e(d,y)}{\partial w_n}$$

$$b = b - \frac{1}{|X|}\lambda \cdot \frac{\partial e(d,y)}{\partial b}$$

- X : conjunto de dados de tamanho $|X|$ e dimensão N
- D : conjunto de valores esperados para X
- Ep : número de épocas
- λ : taxa de aprendizado

Treinamento