Ciência de Dados

Introdução às redes neurais artificiais

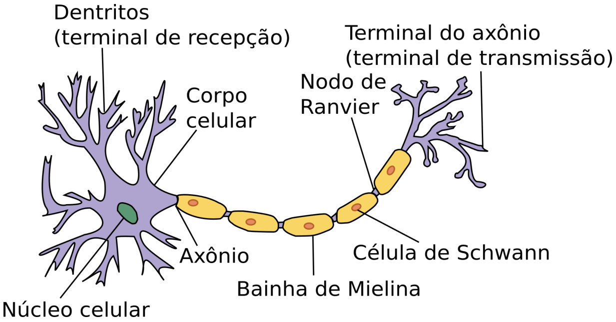
Renato Moraes Silva

renato.silva@facens.br

Facens

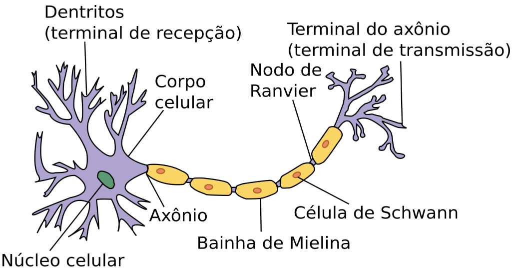
Neurônio

- Os **neurônios** são **células** que geram **sinais elétricos** que são transmitidos para outras células permitindo a comunicação e o controle de atividades corporais.



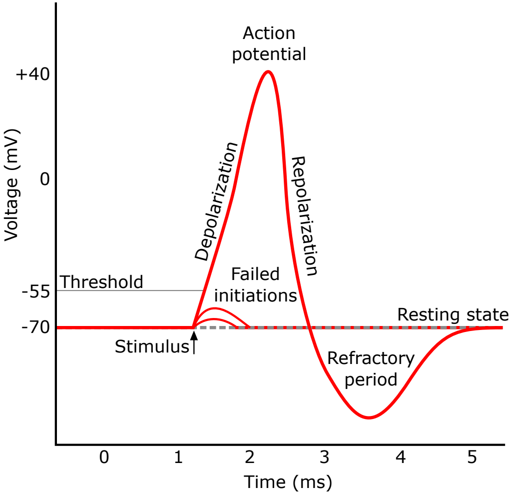
Neurônio

- O neurônio **recebe estímulos elétricos** a partir dos **dendritos**
- Esses estímulos são **integrados**
  - A carga no interior é mais negativa que no exterior
  - O acesso ao interior, durante o repouso, fica fechado
  - A concentração de íons de **sódio** ou **potássio** podem causar um desequilíbrio de cargas no interior da membrana do neurônio
- A estimulação pode levar à geração de um sinal elétrico (**potencial de ação**) que se propaga pelo **axônio**



Introdução

- Potencial de Ação.



Neurônio computacional

- O **neurônio computacional** possui entradas e saídas e processa informação, podendo ser resumido da seguinte forma:
  - Os neurônios **recebem estímulos elétricos**;
  - Esses estímulos são **integrados**;
  - Se a atividade **exceder certo limiar**, o neurônio **gera um pulso** (potencial de ação).

O Modelo de McCulloch e Pitts

- McCulloch e Walter Pitts (1943): “A Logical Calculus of Ideas Immanent in Nervous Activity“
  - Primeiro modelo “computacional“ de neurônio
- Premissas:
  - A atividade do neurônio é um processo binário
  - Certa quantidade de sinapses deve ser excitada para que o neurônio seja ativado
  - Possui uma sinapse inibitória: impede o disparo do neurônio
  - A estrutura do neurônio é fixa (não se altera com o tempo)

O Modelo de McCulloch e Pitts

A representação matemática desse modelo é:

$$y = \begin{cases} 0, & \text{se qualquer entrada } x_i \text{ é inibitória} \\ f\left(g(x)\right), & \text{caso contrário} \end{cases}$$

Na equação acima:

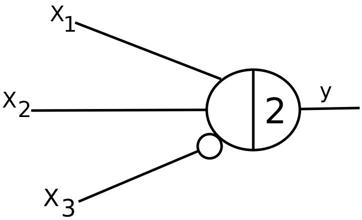
$$f\left(g(x)\right) = \begin{cases} 0, & \text{se } g(x) < b \\ 1, & \text{se } g(x) \geq b \end{cases}$$

Para calcular $g(x)$, faça:

$$g(x) = \sum_{i=1}^{n} x_i, \text{ onde } x_i \in \{0, 1\}$$

O Modelo de McCulloch e Pitts

- Exemplo de Modelo de McCulloch e Pitts
  - Note que o limiar de disparo é dois e que a terceira entrada se liga por meio de uma sinapse inibitória.

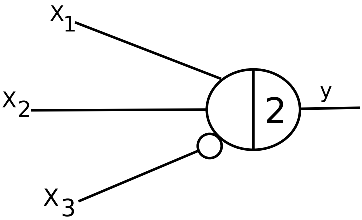


- Tabela-verdade que rege o funcionamento do neurônio:

x_1	x_2	x_3	y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

O Modelo de McCulloch e Pitts

- Exemplo de Modelo de McCulloch e Pitts
  - Note que o limiar de disparo é dois e que a terceira entrada se liga por meio de uma sinapse inibitória.



- Tabela-verdade que rege o funcionamento do neurônio:

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

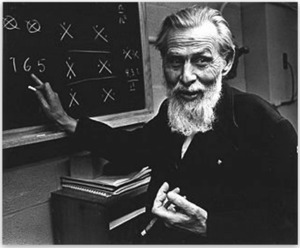
Exercício

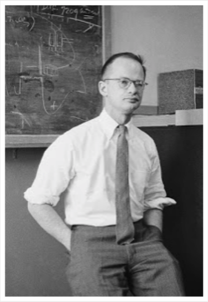
- Desenhe os neurônios de McCulloch e Pitts para as seguintes funções lógicas:
  - AND
  - OR
  - NOT: se a entrada é 1, a rede deve produzir 0, senão 1.

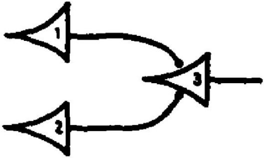
x_1	x_2	AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

História

- 1943: Warren McCulloch e Walter Pitts criam um modelo computacional para redes neurais (neurônio de McCulloch-Pitts)

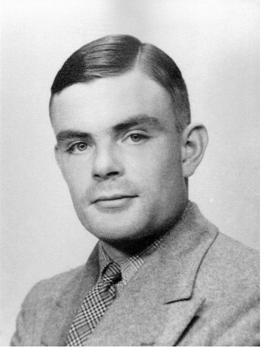


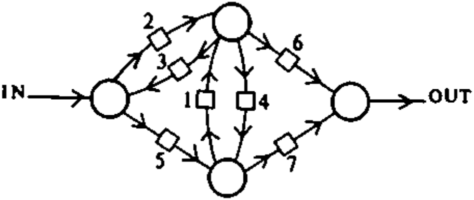




História

- 1948: máquinas desorganizadas (Alan Turing)





História

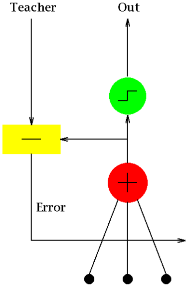
- 1956: campo pesquisa da inteligência artificial foi fundado em uma conferência no campus do Dartmouth College

História

- 1958: Frank Rosenblatt cria o Perceptron, um algoritmo para o reconhecimento de padrões baseado em uma rede neural computacional de camada única

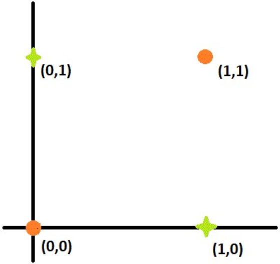
História

- 1960: Adaline (Bernard Widrow e Ted Hoff)
  - Baseado no Neurônio de McCulloch–Pitts
  - Ajusta os pesos usando a regra Delta que é baseada no método dos mínimos quadrados



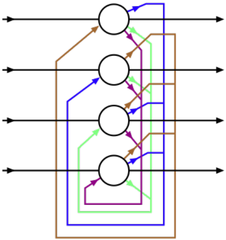
História

- 1969: Minsky demonstrou a limitação das redes neurais de única camada. Início do inverno das redes neurais
  - Problema XOR



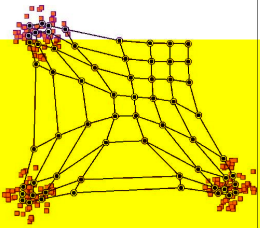
História

- 1982: Hopfield publicou uma série de artigos sobre as redes de Hopfield (uma forma de rede neural artificial recorrente)



História

- 1982: Mapas Auto-Organizáveis ou Mapas de Kohonen (redes não supervisionadas) (Teuvo Kohonen)



- 1986: O algoritmo de aprendizado Back-Propagation para as redes multicamadas foi redescoberto. Fim do inverno das redes neurais.

História

- Década de 90: as redes neurais de base radial foram desenvolvidas

- 1992: máquinas de vetores de suporte (SVM, do inglês, *support-vector machines*)

História

- 2012: deep learning começa a ganhar atenção
  - Primeiros modelos de redes neurais convolucionais a atingirem desempenho estado da arte nas tarefas de classificação de imagens e detecção de objetos
  - Uso de GPUs para o treinamento
  - Lançamento da série GeForce GTX 690

História

- 2017: adoção em massa das redes neurais profundas

Em 2019, os "padrinhos da IA", Yoshua Bengio, Geoffrey Hinton e Yann LeCun, ganharam o prêmio Turing (Nobel da Computação)

.

Classificadores

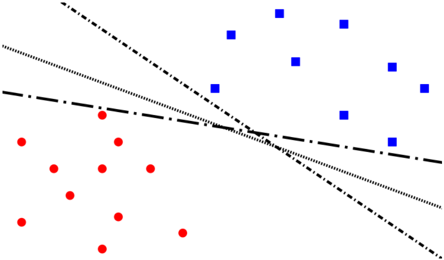
- Classificador:
  - Recebe vetores de **atributos** como entrada;
  - Atribui cada vetor a uma das classes $C_1, C_2, \ldots, C_q$;

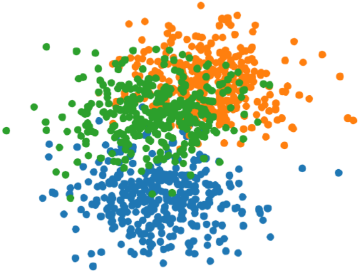
Entrada (vetor de atributos) Saída (classes)

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{bmatrix} \longrightarrow \boxed{\text{Classificador}} \longrightarrow \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \end{bmatrix}$$

Classificadores

- Em geral os classificadores de padrões particionam o espaço de entradas em volumes, chamados regiões de decisão;
- Todos os vetores de atributos dentro de uma mesma região são atribuídos a uma mesma categoria (classe);
- Regiões são separadas por superfícies chamadas **fronteiras de decisão (FD)**;

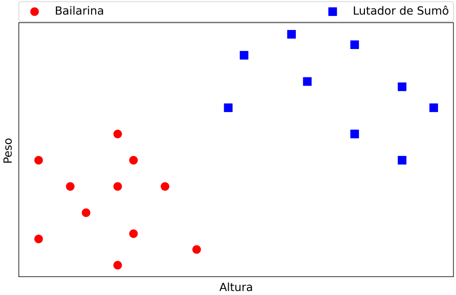




Classificadores

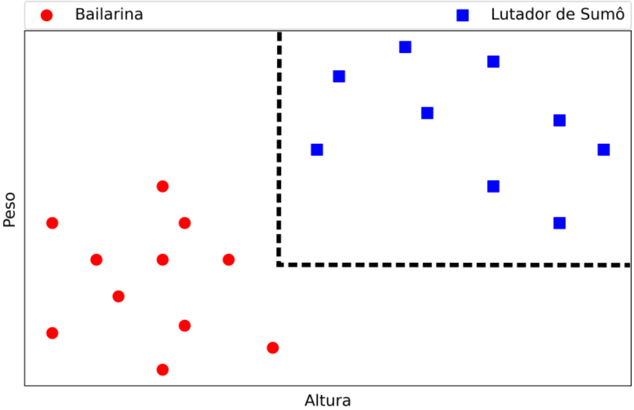
Exemplo: classificar entre lutadores de sumô e bailarinas

- atributos
  - peso
  - altura
- padrões
  - pontos no espaço (peso, altura)



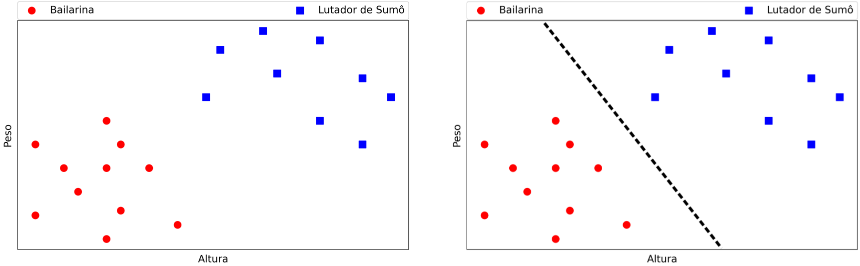
Classificadores

Exemplo da **fronteira de decisão** produzida por um classificador. Diferentes classificadores pode produzir diferentes fronteiras de decisão



Fronteira de decisão linear

Uma das formas de separar as classes do problema mostrado anteriormente é usar um **discriminante linear**



- **Discriminante linear**:
  - Utiliza uma combinação linear das entradas para produzir a FD;
  - Transformação linear de um problema multidimensional em um problema unidimensional.

Discriminante linear

Pode ser dado pela expressão:

$$u = \theta_1 x_1 + \theta_2 x_2 + ... + \theta_m x_m + \theta_0 = \theta_0 + \sum_{i=1}^{m} \theta_i x_i$$

onde, θ_i é o peso da i-ésima entrada x_i e θ_0 é o bias:

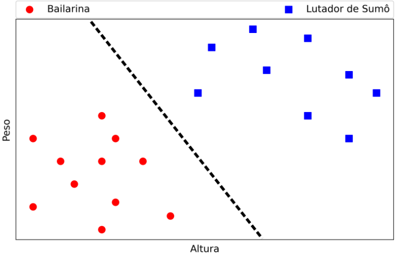
$$\begin{cases} u > 0, & \text{vetor de entrada é relacionado à classe } C_1 \\ u <= 0, & \text{vetor de entrada é relacionado à classe } C_2 \end{cases} \quad (1)$$

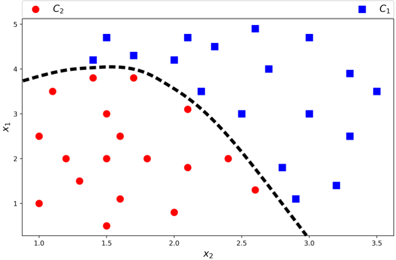
Desafio:

- encontrar o conjunto de pesos que separe adequadamente as classes (**fronteira**: $u = 0$).

Fronteira linear vs não-linear

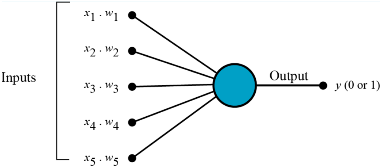
- Alguns problemas admitem fronteira de decisão linear, outros não
- No último problema, uma única reta não é capaz se separar adequadamente as classes



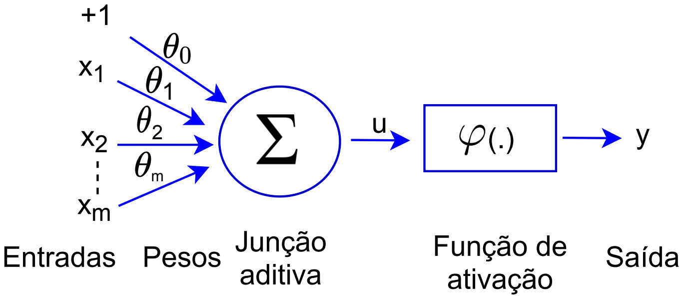


Perceptron

- Desenvolvido por Rosemblat (1958);
- Discriminante linear;
- Problema de reconhecimento de padrões visuais;
- Similar ao ADALINE (ADAptive LINear Element).



Perceptron – diagrama de blocos



- y: ativação (saída do neurônio)
- φ: ativação do neurônio
- θ_0: bias

$$y = \varphi(u) = \varphi\left(\theta_0 + \sum_{i=1}^{m} x_i\theta_i\right)$$

Perceptron

A saída do Perceptron é dada por:

$$y = \varphi(u) = \varphi\left(\theta_0 + \sum_{i=1}^{m} x_i\theta_i\right)$$

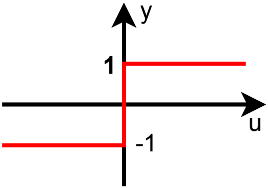
Considere:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \\ \theta_m \end{bmatrix}$$

- Podemos escrever a ativação do neurônio **de forma vetorial**: $u = \theta_0 + \theta^T x$
- E sua saída: $y = \varphi(u) = \varphi\left(\theta_0 + \theta^T x\right)$
  - x é o vetor de entradas, θ é o vetor de pesos (indica as forças da conexão das sinapses), θ_0 é o bias e $\varphi(.)$ é a função de ativação do neurônio.

Perceptron

- Costuma-se usar a função sinal como função de ativação no perceptron:



$$y = \varphi(u) = \begin{cases} +1 & \text{, se } u > 0 \\ -1 & \text{, se } u \leq 0 \end{cases}$$

- Assim, regra de decisão para a classificação de um ponto x é dada por:

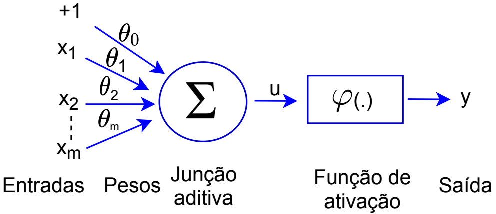
$$y = \varphi(u) = \begin{cases} +1 & \text{, se } u = \theta_0 + \theta^T \mathbf{x} > 0 \\ -1 & \text{, se } u = \theta_0 + \theta^T \mathbf{x} \leq 0 \end{cases}$$

$$y = \begin{cases} +1 & \text{, se } x \text{ pertencente à classe } C_1 \\ -1 & \text{, se } x \text{ pertencente à classe } C_2 \end{cases}$$

Exemplo

- Função lógica AND (tabela verdade):

x_1	x_2	Classe
0	0	C_2
0	1	C_2
1	0	C_2
1	1	C_1



Na tabela acima,
$C_1 = 1$ e $C_2 = 0$.

- Deve-se classificar os padrões de entrada em duas classes (C_1 e C_2);
- Cada padrão de entrada tem dois componentes (x_1 e x_2);
- Considere os pesos: $\theta_0 = -1.5, \theta_1 = 1, \theta_2 = 1$.

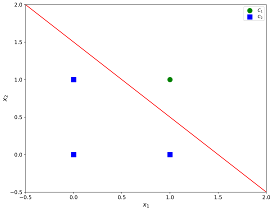
Exemplo

- **Função AND**: $\theta_0 = -1.5, \theta_1 = 1, \theta_2 = 1.$

$$\text{FD (hiperplano)} \rightarrow u = 0 = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = -1.5 + 1x_1 + 1x_2$$

$$y = \varphi(u) = \begin{cases} +1 & \text{, se } u = \theta_0 + \theta^T x > 0 \\ -1 & \text{, se } u = \theta_0 + \theta^T x + \leq 0 \end{cases}$$

$$y = \begin{cases} +1 & \text{, se } x \text{ pertencente à classe } C_1 \\ -1 & \text{, se } x \text{ pertencente à classe } C_2 \end{cases}$$

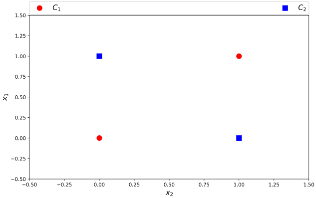


- Os **pontos do hiperplano** são calculados para $u = 0$;
  - Exemplos:
    - $x_1 = 1.0$ e $x_2 = 0.5$
    - $x_1 = 1.5$ e $x_2 = 0$
    - $x_1 = 0.5$ e $x_2 = 1.0$
    - $x_1 = 0$ e $x_2 = 1.5$

Exemplo

- **Função XOR**
  - O perceptron não é capaz de resolver

x_1	x_2	Classe
0	0	C_2
0	1	C_1
1	0	C_1
1	1	C_2



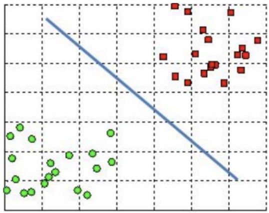
Na tabela acima, $C_1 = 1$ e $C_2 = 0$.

Perceptron – hiperplano

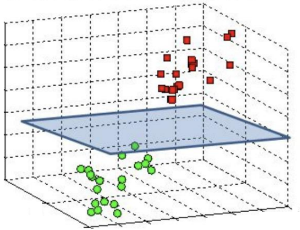
- Em um perceptron, a FD é dada por um **hiperplano** definido por:

$$\theta_0 + \theta^T \mathbf{x} = 0$$

- O hiperplano é:
  - uma reta no caso bidimensional (duas variáveis de entrada);
  - um plano no caso tridimensional;
  - Os pesos e o bias do perceptron definem a **localização do hiperplano** no espaço das entradas;
  - O bias (θ_0) **desloca** a fronteira de decisão em relação à **origem**.



(a) Um hiperplano em R^2 é uma linha.



(b) Um hiperplano em R^3 é um plano.

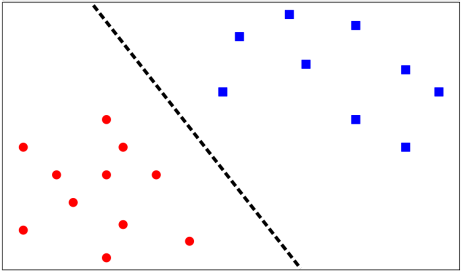
Treinamento

- Suponha que as variáveis de entrada se originem de duas classes C_1 e C_2 linearmente separáveis;
- O problema do aprendizado do perceptron (treinamento) consiste em encontrar um conjunto de pesos e bias que definem um hiperplano que separe linearmente os vetores de entrada das classes C_1 e C_2;
- Isto é, um vetor de pesos θ e um bias θ_0 tal que:

$$\begin{cases} \theta_0 + \theta^T x > 0, & \text{para todo } x \text{ pertencente à classe } C_1 \\ \theta_0 + \theta^T x \leq 0, & \text{para todo } x \text{ pertencente à classe } C_2 \end{cases}$$

Treinamento

- Para isso, um conjunto com padrões para o treinamento do perceptron é definido:
  - Conjunto de treinamento T;
  - Conjunto de pares: **entrada** $x(n)$ e **saída desejada** $d(n)$;
- O conjunto de treinamento T é formado por dois subconjuntos:
  - T_1 composto por vetores de entrada que pertencem à classe C_1;
  - T_2 composto por vetores de entrada que pertencem à classe C_2;



Treinamento

- Antes de apresentarmos a estratégia de treinamento, vamos definir o padrão de treinamento e o vetor de pesos na n- ésima iteração do algoritmo como:

$$x(n) = \begin{bmatrix} +1 \\ x_1(n) \\ x_2(n) \\ x_3(n) \\ \vdots \\ x_m(n) \end{bmatrix} \qquad \theta(n) = \begin{bmatrix} \theta_0(n) \\ \theta_1(n) \\ \theta_2(n) \\ \theta_3(n) \\ \vdots \\ \theta_m(n) \end{bmatrix}$$

- Note que o **bias** agora é visto como um **peso**;
- A saída será dada por:

$$y(n) = \varphi\left(\theta(n)^T x(n)\right)$$

Treinamento

- Usando-se a **função sinal como ativação** e considerando-se a taxa de aprendizagem fixa, a regra de ajuste de é:

$$\theta(n+1) = \theta(n) + \alpha\left[d(n) - y(n)\right]x(n)$$

onde,

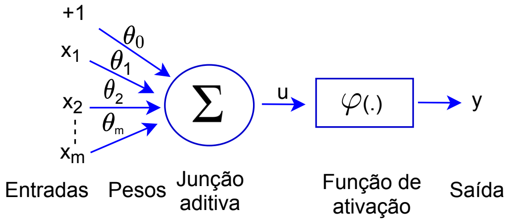
$$y(n) = \varphi\left(\theta(n)^T x(n)\right) = \begin{cases} +1 & ,\text{se } \theta(n)^T x(n) > 0 \\ -1 & ,\text{se } \theta(n)^T x(n) \leq 0 \end{cases}$$

e

$$d(n) = \begin{cases} +1 & ,\text{se } x(n) \text{ pertencente à classe } C_1 \\ -1 & ,\text{se } x(n) \text{ pertencente à classe } C_2 \end{cases}$$

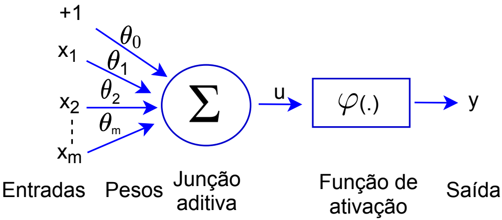
Treinamento do Perceptron

- Proposta por Rosenblatt em 1958;
- Regra de adaptação com incremento fixo:
- Guarda uma certa semelhança com a Regra Delta (ADALINE);
  - A Regra Delta utiliza o conceito do gradiente descendente do erro médio quadrático.



Treinamento do Perceptron

- Proposta por Rosenblatt em 1958;
- Regra de adaptação com incremento fixo:
- Guarda uma certa semelhança com a Regra Delta (ADALINE);
  - A Regra Delta utiliza o conceito do gradiente descendente do erro médio quadrático.



Treinamento do Perceptron

função perceptron($\mathbf{X}^T, \mathbf{d}_T, \alpha$)

$\theta \leftarrow 0$ // inicializa os pesos
$n \leftarrow 1$ // inicializa o número da iteração (época)

faça

$y(n) \leftarrow \varphi\left(\theta^T \mathbf{x}(n)\right)$ // ativação do neurônio (saída)

// calcula o erro
$e(n) = d(n) - y(n)$

// ajuste de pesos
$\theta \leftarrow \theta + \alpha * e(n) * \mathbf{x}(n)$

$n \leftarrow n + 1$

enquanto (erro na classificação não é aceitável)

fim função

Treinamento do Perceptron

- No algoritmo anterior, o conjunto de treinamento T tem N padrões, ou seja:

$$\mathbf{X}_T = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} \qquad \mathbf{d}_T = \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix}$$

- O vetor de entradas $x(n)$ e a saída desejada $d(n)$ podem ser obtidas a partir de $\mathbf{X}_T$ e $\mathbf{d}_T$ através de:

$$x(n) = x_t, \qquad d(n) = d_t$$

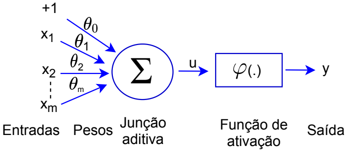
na qual $t = \text{mod}(n-1, N) + 1$

  - mod retorna o resto da divisão

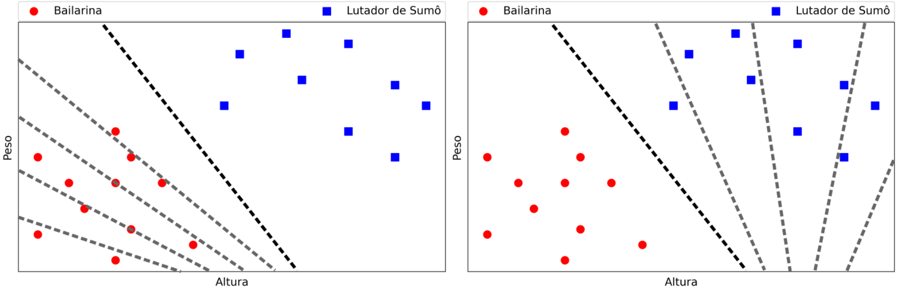
- O erro é dado por: $e(n) = d(n) - y(n)$

Escolha da taxa de aprendizagem

- Deve ser restrita ao intervalo $0 < \alpha \leq 1$
- α baixa:
  - adaptação mais lenta;
  - a mudança dos pesos é mais estável;
- α alta:
  - adaptação mais rápida;
  - a mudança dos pesos é menos estável;



Fronteira de decisão



Possíveis fronteiras de decisão geradas pelo Perceptron durante a convergência

Exercício 1

Conside que o Perceptron foi treinado e possui os seguintes pesos: $w = [0.7, -5, -2]$. Qual a saída da rede e a classe quando usada para classificar o seguinte dado: $x = [0.9, -0.5]$?

Exercício 1

Conside que o Perceptron foi treinado e possui os seguintes pesos: $w = [0.7, -5, -2]$. Qual a saída da rede e a classe quando usada para classificar o seguinte dado: $x = [0.9, -0.5]$?

---

função perceptron($\mathbf{X}^T, \mathbf{d}_T, \alpha$)

$\theta \leftarrow 0$ // inicializa os pesos
$n \leftarrow 1$ // inicializa o número da iteração (época)

faça

$y(n) \leftarrow \varphi\left(\theta^T \mathbf{x}(n)\right)$ // ativação do neurônio (saída)

// calcula o erro
$e(n) = d(n) - y(n)$

// ajuste de pesos
$\theta \leftarrow \theta + \alpha * e(n) * \mathbf{x}(n)$

$n \leftarrow n + 1$

enquanto (erro na classificação não é aceitável)

fim função

Exercício 2

Conside que o Perceptron está sendo treinado e possui os seguintes pesos na iteração atual: $w[0, 0, 0, 8]$. Considere também que iremos passar o seguinte dado para treinameto: $x = [0.5, 0.7]$. Suponha que ele possui a seguinte classe $d = -1$. Se a taxa de aprendizado do Perceptron é 0.5, quais serão os novos pesos w?

Exercício 2

Conside que o Perceptron está sendo treinado e possui os seguintes pesos na iteração atual: $w[0, 0, 0, 8]$. Considere também que iremos passar o seguinte dado para treinameto: $x = [0.5, 0.7]$. Suponha que ele possui a seguinte classe $d = -1$. Se a taxa de aprendizado do Perceptron é 0.5, quais serão os novos pesos w?

---

função perceptron($\mathbf{X}^T, \mathbf{d}_T, \alpha$)

$\theta \leftarrow 0$ // inicializa os pesos
$n \leftarrow 1$ // inicializa o número da iteração (época)

faça

$y(n) \leftarrow \varphi\left(\theta^T \mathbf{x}(n)\right)$ // ativação do neurônio (saída)

// calcula o erro
$e(n) = d(n) - y(n)$

// ajuste de pesos
$\theta \leftarrow \theta + \alpha * e(n) * \mathbf{x}(n)$

$n \leftarrow n + 1$

enquanto (erro na classificação não é aceitável)

fim função

Referências

- Aula dos professores Hugo Valadares Siqueira, Levy Boccato e Romis Attux
- CARVALHO, André Carlos Ponce de Leon Ferreira et al. Inteligência Artificial - Uma Abordagem de Aprendizado de Máquina. Disponível em: Minha Biblioteca, (2nd edição). Grupo GEN, 2021.