Universidade de Fortaleza - UNIFOR

Sistemas Inteligentes - T951

Msc. Prof. Paulo Cirillo Souza Barbosa
Centro de Ciências Tecnológicas - CCT
Universidade de Fortaleza
Fortaleza, Ceará, Brasil

4 de abril de 2023

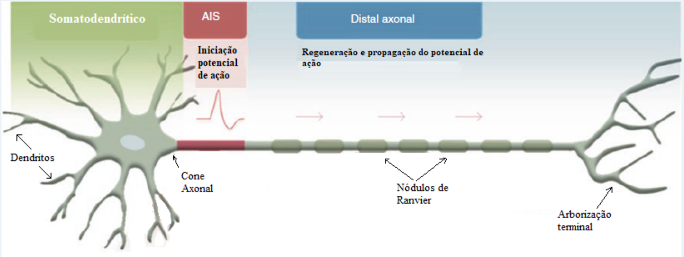
Introdução.

- São modelos computacionais inspirados no sistema nervoso de seres vivos.
- São definidas por um conjunto de unidades de processamento, caracterizadas por **neurônios artificiais** que são interconectados através de uma matriz de pesos (sinapses artificiais).

- Características principais:
  1. Adaptação por experiência.
  2. Capacidade de aprendizado.
  3. Habilidade de generalização.
  4. Organização de dados.
  5. Tolerância a falhas.
  6. Armazenamento distribuído.
  7. Facilidade de prototipagem.

- Aplicações:
  1. Aproximador universal de funções.
  2. Controle de processos.
  3. Reconhecimento/classificação de padrões.
  4. Agrupamento de dados.
  5. Sistemas de previsão.
  6. Otimização de sistemas.
  7. Memórias Associativas.

Neurônio Biológico.



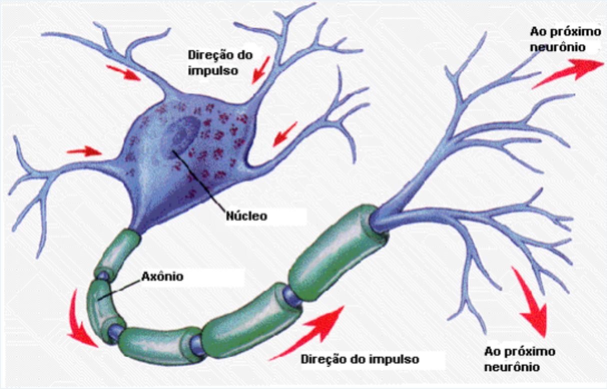
- O neurônio nada mais é do que uma célula que consegue conduzir estímulos elétricos advindos de reações físico-químicas.
  1. Dendritos.
  2. Soma ou Corpo Celular.
  3. Axônio.
  4. Sinapses.

Neurônio Biológico.

- **Dendritos** – Ramificações correspondentes aos canais de entrada de informação (sinais elétricos, escala mV).
- **Corpo Celular** – Local onde é feito o balanço energético da célula nervosa (soma das contribuições de energia).
- **Axônios** – Canal de saída do neurônio, ou seja, caminho de propagação dos impulsos nervosos em direção a outros neurônios ou músculos.
- **Sinapses** – Pontos de contato entre neurônios onde há passagem de neurotransmissores do axônio de um neurônio para os dendritos de outro neurônio.

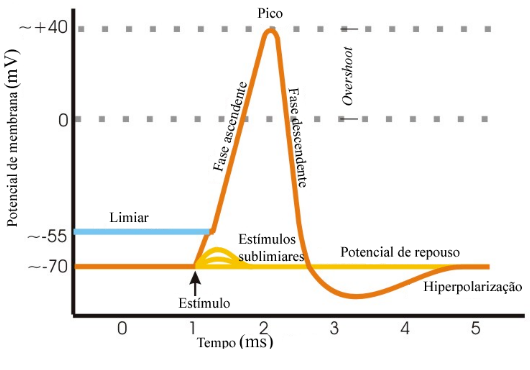
Neurônio Biológico.

- O Fluxo da informação ocorre sempre no sentido: **Dendritos** $\Longrightarrow$ **Corpo Celular** $\Longrightarrow$ **Axônio**.



Neurônio Biológico.

- O axônio emite um impulso elétrico (potencial de ação) apenas se o balanço energético realizado no corpo celular for maior que um certo limiar. Neste caso, diz-se que o neurônio disparou ou está ativado.



Neurônio Biológico.

- A chegada deste sinal de disparo no terminal do axônio, faz com que neurotransmissores sejam liberados na fenda sináptica.
- Sinapses podem ser excitatórias (facilitam a passagem do potencial de ação) ou inibitórias (inibem a passagem do potencial de ação).
- Neurônios podem fazer conexões:
  1. com outros neurônios.
  2. com os músculos diretamente.
  3. com os órgãos sensoriais.

Curiosidades!

- Um comparativo pode ser feito com relação às portas lógicas, que operam na ordem dos nanossegundos.
- Um neurônio biológico opera na ordem dos milissegundos.
- Como computadores tem características "inferiores"ao cérebro humano?

Curiosidades!

- Um comparativo pode ser feito com relação às portas lógicas, que operam na ordem dos nanossegundos.
- Um neurônio biológico opera na ordem dos milissegundos.
- Como computadores tem características "inferiores"ao cérebro humano?
- Há cerca de 10 Bilhões de neurônios no cortex cerebral (massa cinzenta).
- O córtex é a estrutura responsável pelas habilidades cognitivas superiores, tais como memória, raciocínio lógico, linguagem, consciência, dentre outras.

Neurônio de Mculloch - Pitts.

- Modelo proposto em: *MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, v. 5, n. 4, p. 115-133, 1943.*

Neurônio de Mculloch - Pitts.

- Modelo proposto em: *MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, v. 5, n. 4, p. 115-133, 1943.*
- Faz-se necessário **destacar** que se trata de um modelo, ou seja, é uma aproximação do neurônio natural.

Neurônio de Mculloch - Pitts.

- Modelo proposto em: *MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, v. 5, n. 4, p. 115-133, 1943.*
- Faz-se necessário **destacar** que se trata de um modelo, ou seja, é uma aproximação do neurônio natural.
- Portanto, o neurônio M-P é uma aproximação útil do neurônio real, pois, serve até hoje como bloco construtivo básico de algoritmos de RNA.

Neurônio de Mculloch - Pitts.

- Modelo proposto em: *MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, v. 5, n. 4, p. 115-133, 1943.*
- Faz-se necessário **destacar** que se trata de um modelo, ou seja, é uma aproximação do neurônio natural.
- Portanto, o neurônio M-P é uma aproximação útil do neurônio real, pois, serve até hoje como bloco construtivo básico de algoritmos de RNA.
- A modelagem realizada está ligada aos aspectos do **processamento da informação** em um neurônio biológico, ou seja, os caminhos e etapas pelas quais passam os potenciais de ação que trafegam:

Neurônio de Mculloch - Pitts.

- Modelo proposto em: *MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, v. 5, n. 4, p. 115-133, 1943.*
- Faz-se necessário **destacar** que se trata de um modelo, ou seja, é uma aproximação do neurônio natural.
- Portanto, o neurônio M-P é uma aproximação útil do neurônio real, pois, serve até hoje como bloco construtivo básico de algoritmos de RNA.
- A modelagem realizada está ligada aos aspectos do **processamento da informação** em um neurônio biológico, ou seja, os caminhos e etapas pelas quais passam os potenciais de ação que trafegam:
    1. de um neurônio a outro neurônio,

Neurônio de Mculloch - Pitts.

- Modelo proposto em: *MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, v. 5, n. 4, p. 115-133, 1943.*
- Faz-se necessário **destacar** que se trata de um modelo, ou seja, é uma aproximação do neurônio natural.
- Portanto, o neurônio M-P é uma aproximação útil do neurônio real, pois, serve até hoje como bloco construtivo básico de algoritmos de RNA.
- A modelagem realizada está ligada aos aspectos do **processamento da informação** em um neurônio biológico, ou seja, os caminhos e etapas pelas quais passam os potenciais de ação que trafegam:
    1. de um neurônio a outro neurônio,
    2. receptores sensoriais a um neurônio, ou

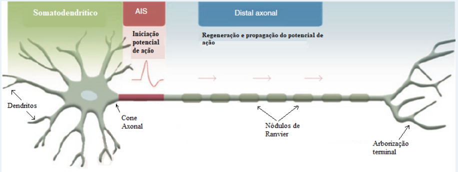
Neurônio de Mculloch - Pitts.

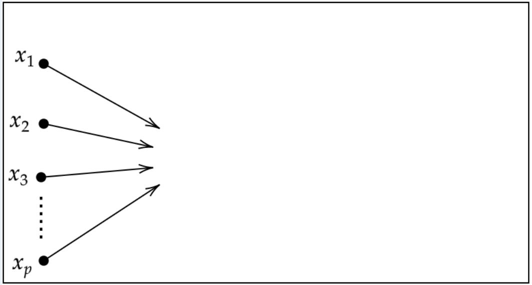
- Modelo proposto em: *MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, v. 5, n. 4, p. 115-133, 1943.*
- Faz-se necessário **destacar** que se trata de um modelo, ou seja, é uma aproximação do neurônio natural.
- Portanto, o neurônio M-P é uma aproximação útil do neurônio real, pois, serve até hoje como bloco construtivo básico de algoritmos de RNA.
- A modelagem realizada está ligada aos aspectos do **processamento da informação** em um neurônio biológico, ou seja, os caminhos e etapas pelas quais passam os potenciais de ação que trafegam:
  1. de um neurônio a outro neurônio,
  2. receptores sensoriais a um neurônio, ou
  3. de um neurônio a um atuador (e.g. músculo).

Neurônio de Mculloch - Pitts.

- Modelo proposto em: *MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, v. 5, n. 4, p. 115-133, 1943.*
- Faz-se necessário **destacar** que se trata de um modelo, ou seja, é uma aproximação do neurônio natural.
- Portanto, o neurônio M-P é uma aproximação útil do neurônio real, pois, serve até hoje como bloco construtivo básico de algoritmos de RNA.
- A modelagem realizada está ligada aos aspectos do **processamento da informação** em um neurônio biológico, ou seja, os caminhos e etapas pelas quais passam os potenciais de ação que trafegam:
  1. de um neurônio a outro neurônio,
  2. receptores sensoriais a um neurônio, ou
  3. de um neurônio a um atuador (e.g. músculo).
- Deseja-se portanto, desenvolver modelos matemáticos que representem os **dendritos**, as **sinapses**, o **corpo celular** e o **axônio**.

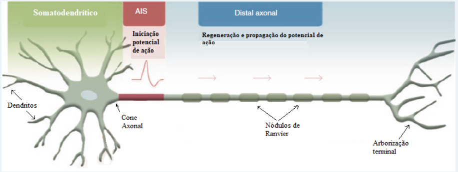
Neurônio de Mculloch - Pitts.

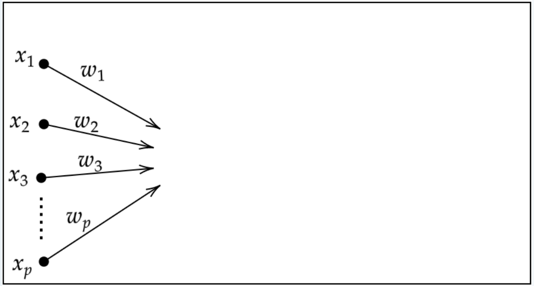




- Cada ramo dendrítico é modelado como um canal, pelo qual flui a informação de entrada ($x_j, j = 1, \cdots, p$).

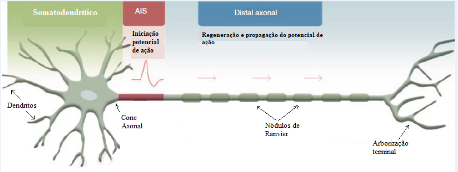
Neurônio de Mculloch - Pitts.

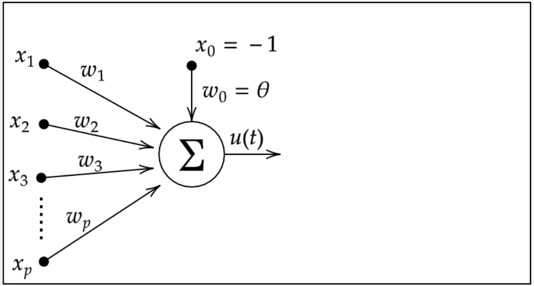




- A força (ou eficiência) das conexões sinápticas de uma certa árvore dendrítica é modelada como um fator (peso sináptico), cujo papel é modular o fluxo de sinais passando por uma certa árvore dendrítica.

Neurônio de Mculloch - Pitts.

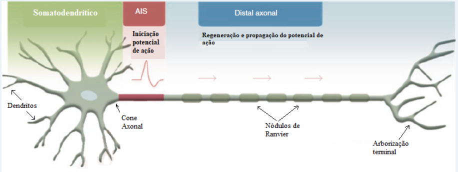


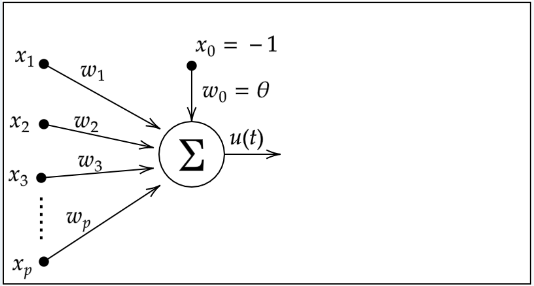


- A função do corpo celular de realizar o balanço ou acúmulo energético é modelada por uma operação de somatório sobre as entradas moduladas pelos pesos sinápticos.

$$u =$$

Neurônio de Mculloch - Pitts.

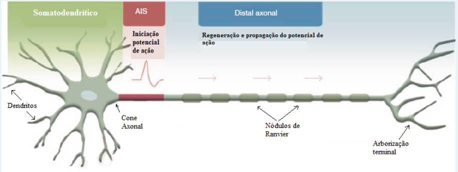


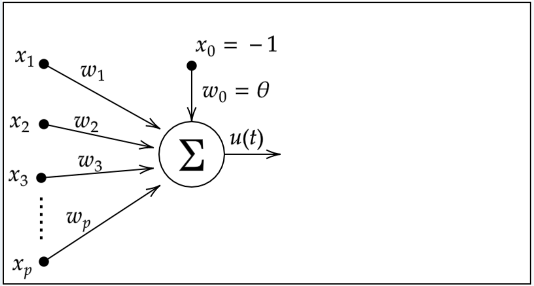


- A função do corpo celular de realizar o balanço ou acúmulo energético é modelada por uma operação de somatório sobre as entradas moduladas pelos pesos sinápticos.

$$u = w_1 x_1 + w_2 x_2 + \cdots w_p x_p - \theta$$

Neurônio de Mculloch - Pitts.

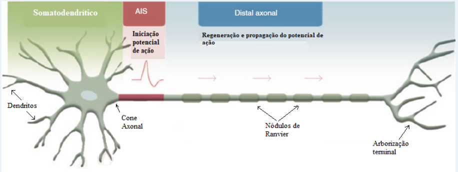


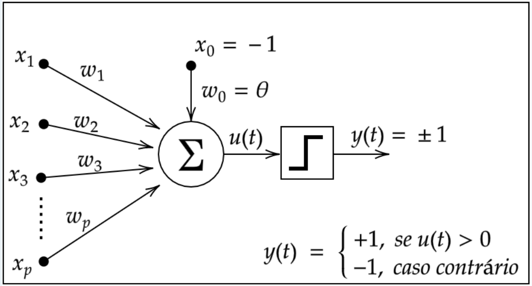


- x_1,x_2 são as entradas.
- w_1,w_2 os pesos sinápticos.
- θ o limiar (bias, viés, *threshold*)
- u ativação.

$$u = w_1x_1 + w_2x_2 + \cdots w_px_p - \theta$$

Neurônio de Mculloch - Pitts.





- x_1,x_2 são as entradas.
- w_1,w_2 os pesos sinápticos.
- θ o limiar (bias, viés, *threshold*)
- u ativação.

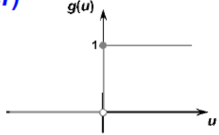
$$u = w_1x_1 + w_2x_2 + \cdots w_px_p - \theta$$

Neurônio de Mculloch - Pitts.

- A função de ativação $y(t)$ não se limita apenas ao degrau bipolar.
- Funções parcialmente diferenciáveis.

❑ Função degrau (*heavyside/ hard limiter*)

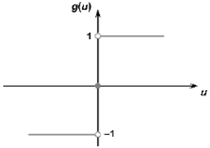
$$g(u) = \begin{cases} 1, & \text{se } u \geq 0 \\ 0, & \text{se } u < 0 \end{cases}$$



❑ Função degrau bipolar ou sinal (*symmetric hard limiter*)

$$g(u) = \begin{cases} 1, & \text{se } u > 0 \\ 0, & \text{se } u = 0 \\ -1, & \text{se } u < 0 \end{cases} \quad \text{ou} \quad g(u) = \begin{cases} 1, & \text{se } u \geq 0 \\ -1, & \text{se } u < 0 \end{cases}$$

P/ classificação

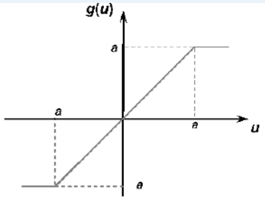


Neurônio de Mculloch - Pitts.

- A função de ativação $y(t)$ não se limita apenas ao degrau bipolar.
- Funções parcialmente diferenciáveis.

❑ Função rampa simétrica

$$g(u) = \begin{cases} a\,, & \text{se} \quad u > a \\ u\,, & \text{se} \quad -a \le u \le a \\ -a\,, & \text{se} \quad u < -a \end{cases}$$

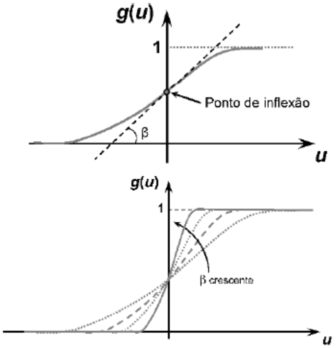


Neurônio de Mculloch - Pitts.

- A função de ativação $y(t)$ não se limita apenas ao degrau bipolar.
- Funções totalmente diferenciáveis.

❑ **Função logística**

$$g(u) = \frac{1}{1+e^{-\beta \cdot u}} \quad \beta > 0$$

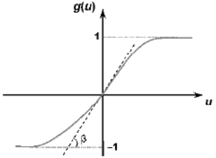


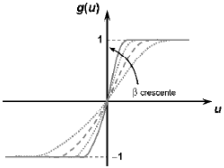
Neurônio de Mculloch - Pitts.

- A função de ativação $y(t)$ não se limita apenas ao degrau bipolar.
- Funções totalmente diferenciáveis.

❑ Função tangente hiperbólica

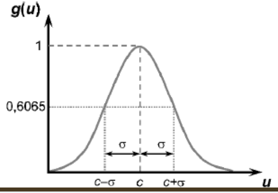
$$g(u) = \frac{1 - e^{-\beta \cdot u}}{1 + e^{-\beta \cdot u}} \qquad \beta > 0$$





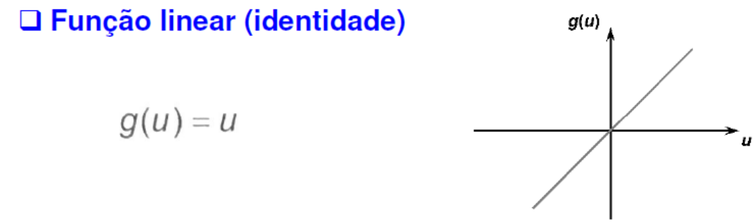
❑ Função gaussiana

$$g(u) = e^{-\frac{(u-c)^2}{2\sigma^2}} \qquad \sigma \neq 0$$



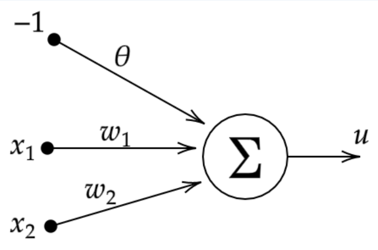
Neurônio de Mculloch - Pitts.

- A função de ativação $y(t)$ não se limita apenas ao degrau bipolar.
- Função Identidade.



Neurônio de Mculloch - Pitts.

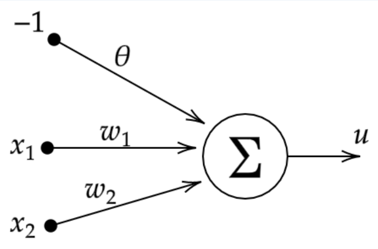
- Dado o seguinte neurônio, com duas entradas x_1 e x_2, o seu modelo pode ser escrito como:



- A combinação linear u é dada por:

Neurônio de Mculloch - Pitts.

- Dado o seguinte neurônio, com duas entradas x_1 e x_2, o seu modelo pode ser escrito como:



- A combinação linear u é dada por:

$$u = w_1x_1 + w_2x_2 - \theta$$

- Para fins de classificação, pode-se trabalhar no plano (x_1, x_2), ou seja, $u = 0$.

Neurônio de Mculloch - Pitts.

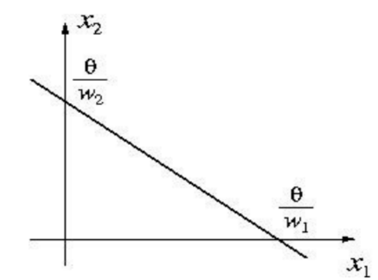
$$u = w_1 x_1 + w_2 x_2 - \theta = 0$$

Neurônio de Mculloch - Pitts.

$$u = w_1 x_1 + w_2 x_2 - \theta = 0$$

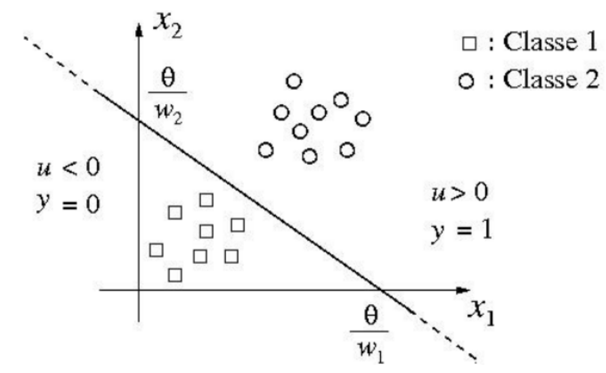
$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{\theta}{w_2}$$

- Esta equação define a seguinte reta em (x_1, x_2).



Neurônio de Mculloch - Pitts.

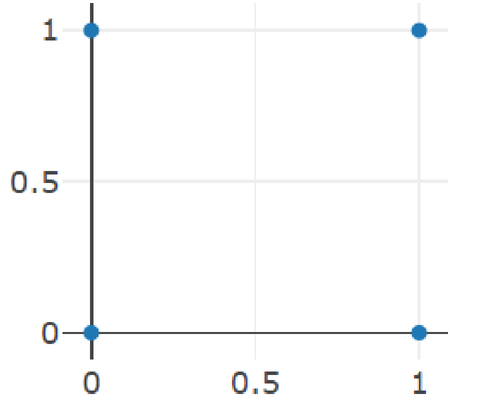
- Assim, um neurônio M-P pode ser usado para separar com eficiência, duas classes que estejam bem isoladas uma da outra.



Exemplo 1: Implementação das portas **OR**, **AND** e **NOT**.

- **OR**: É possível encontrar uma reta que separe os pontos da Classe **UM** (y=1) dos da Classe **DOIS** (y=0)?

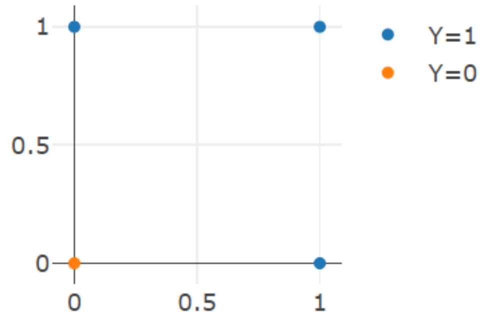
x_1	x_1	y
0	0	
0	1	
1	0	
1	1	



Exemplo 1: Implementação das portas **OR**, **AND** e **NOT**.

- **OR**: É possível encontrar uma reta que separe os pontos da Classe **UM** (y=1) dos da Classe **DOIS** (y=0)?
- É possível encontrar mais de uma reta?

x_1	x_1	y
0	0	0
0	1	1
1	0	1
1	1	1



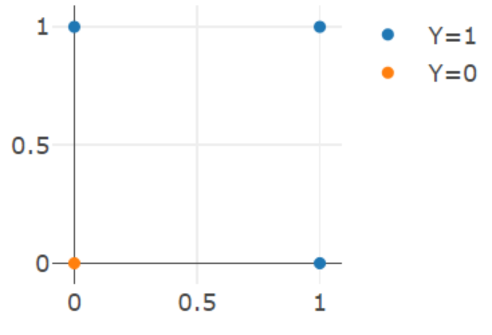
Exemplo 1: Implementação das portas **OR**, **AND** e **NOT**.

- **OR**: É possível encontrar uma reta que separe os pontos da Classe **UM** (y=1) dos da Classe **DOIS** (y=0)? SIM
- É possível encontrar mais de uma reta? Quantas?

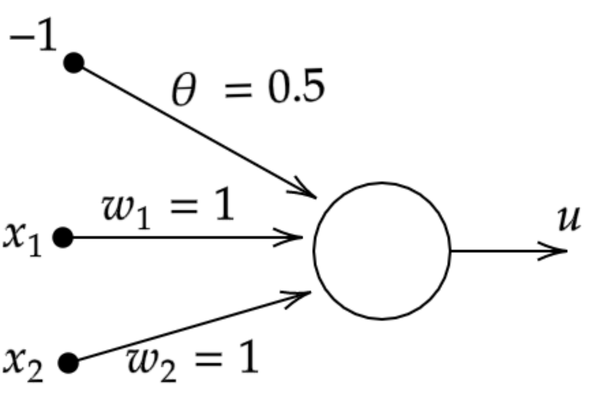
Exemplo 1: Implementação das portas **OR**, **AND** e **NOT**.

- **OR**: É possível encontrar uma reta que separe os pontos da Classe **UM** (y=1) dos da Classe **DOIS** (y=0)? SIM
- É possível encontrar mais de uma reta? Quantas? Infinitas.

x_1	x_1	y
0	0	0
0	1	1
1	0	1
1	1	1



Exemplo 1: neurônio MP das portas **OR**.

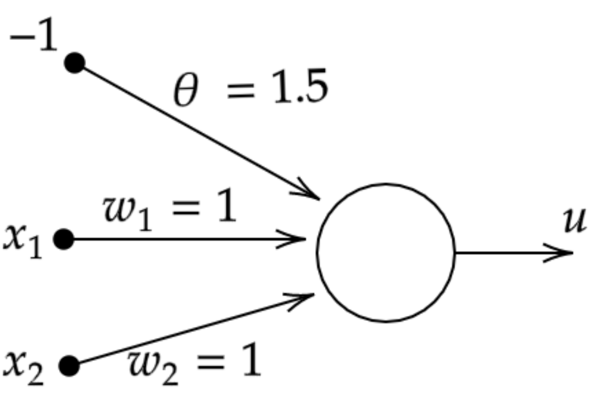


$$w_1 = w_2 = 1 \text{ e } \theta = 0.5$$

$$y = 1, \text{se } u \geq 0$$

$$y = 0, \text{se } u < 0$$

Exemplo 1: neurônio MP das portas **AND**.

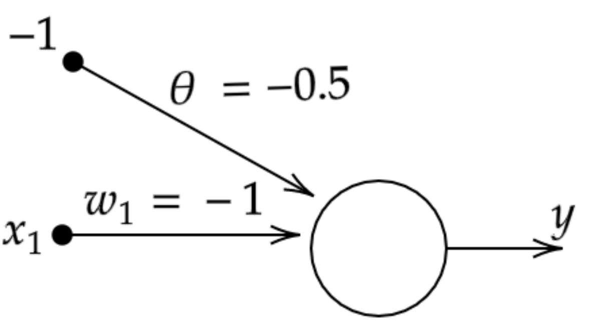


$$w_1 = w_2 = 1 \ \text{ e } \ \theta = 1.5$$

$$y = 1, \text{se } u \geq 0$$

$$y = 0, \text{se } u < 0$$

Exemplo 1: neurônio MP das portas NOT.



$$w_1 = -1 \text{ e } \theta = -0.5$$

$$y = 1, \text{se } u \geq 0$$

$$y = 0, \text{se } u < 0$$

Notas importantes.

- O neurônio MP pode ser usado para implementar as portas lógicas AND, OR e NOT porque estas, do ponto de vista geométrico, podem ser interpretadas como um problema de classificação binária (duas categorias).

Notas importantes.

- O neurônio MP pode ser usado para implementar as portas lógicas AND, OR e NOT porque estas, do ponto de vista geométrico, podem ser interpretadas como um problema de classificação binária (duas categorias).
- O neurônio MP, do ponto de vista geométrico, pode ser intepretado como uma reta (2D), ou um plano (3D) ou ainda um hiperplano ($> 3D$), que é usado para separar duas categorias de dados distintas.

Notas importantes.

- O neurônio MP pode ser usado para implementar as portas lógicas AND, OR e NOT porque estas, do ponto de vista geométrico, podem ser interpretadas como um problema de classificação binária (duas categorias).
- O neurônio MP, do ponto de vista geométrico, pode ser intepretado como uma reta (2D), ou um plano (3D) ou ainda um hiperplano ($> 3D$), que é usado para separar duas categorias de dados distintas.
- Na implementação das portas lógicas AND, OR e NOT, os valores dos pesos e do limiar foram determinados pelo projetista com base na análise geométrica do problema.

Notas importantes.

- O neurônio MP pode ser usado para implementar as portas lógicas AND, OR e NOT porque estas, do ponto de vista geométrico, podem ser interpretadas como um problema de classificação binária (duas categorias).
- O neurônio MP, do ponto de vista geométrico, pode ser intepretado como uma reta (2D), ou um plano (3D) ou ainda um hiperplano ($> 3D$), que é usado para separar duas categorias de dados distintas.
- Na implementação das portas lógicas AND, OR e NOT, os valores dos pesos e do limiar foram determinados pelo projetista com base na análise geométrica do problema.
- Como fazer com que o neurônio M-P determine de forma automática os valores dos pesos e do limiar para um problema específico?

Notas importantes.

- O neurônio MP pode ser usado para implementar as portas lógicas AND, OR e NOT porque estas, do ponto de vista geométrico, podem ser interpretadas como um problema de classificação binária (duas categorias).
- O neurônio MP, do ponto de vista geométrico, pode ser intepretado como uma reta (2D), ou um plano (3D) ou ainda um hiperplano ($> 3D$), que é usado para separar duas categorias de dados distintas.
- Na implementação das portas lógicas AND, OR e NOT, os valores dos pesos e do limiar foram determinados pelo projetista com base na análise geométrica do problema.
- Como fazer com que o neurônio M-P determine de forma automática os valores dos pesos e do limiar para um problema específico?
- Para que o neurônio M-P seja capaz de aprender sozinho a resolver um problema de classificação é necessário dotá-lo de uma **regra de aprendizagem**.

Notas importantes.

- O neurônio MP pode ser usado para implementar as portas lógicas AND, OR e NOT porque estas, do ponto de vista geométrico, podem ser interpretadas como um problema de classificação binária (duas categorias).
- O neurônio MP, do ponto de vista geométrico, pode ser intepretado como uma reta (2D), ou um plano (3D) ou ainda um hiperplano ($> 3D$), que é usado para separar duas categorias de dados distintas.
- Na implementação das portas lógicas AND, OR e NOT, os valores dos pesos e do limiar foram determinados pelo projetista com base na análise geométrica do problema.
- Como fazer com que o neurônio M-P determine de forma automática os valores dos pesos e do limiar para um problema específico?
- Para que o neurônio M-P seja capaz de aprender sozinho a resolver um problema de classificação é necessário dotá-lo de uma **regra de aprendizagem**.
- Uma regra de aprendizagem nada mais é do que uma equação que altera os valores dos pesos e do limiar em função dos erros cometidos durante a execução da tarefa de classificação.

Algoritmo do Perceptron Simples (Definições Preliminares).

- Vamos assumir que existe uma lei matemática, ou função $\mathbf{H}(\cdot)$. Tal função chamada de mapeamento, que relaciona um vetor de entrada $\mathbf{x} \in \mathbb{R}^{p+1}$ qualquer com um vetor de saída $\mathbf{y} \in \mathbb{R}^{c}$. Matematicamente essa relação pode ser descrita como $\mathbf{y} = \mathbf{H}(\mathbf{x})$.
- Porém, $\mathbf{H}(\cdot)$ é

Algoritmo do Perceptron Simples (Definições Preliminares).

- Vamos assumir que existe uma lei matemática, ou função $\mathbf{H}(\cdot)$. Tal função chamada de mapeamento, que relaciona um vetor de entrada $\mathbf{x} \in \mathbb{R}^{p+1}$ qualquer com um vetor de saída $\mathbf{y} \in \mathbb{R}^{c}$. Matematicamente essa relação pode ser descrita como $\mathbf{y} = \mathbf{H}(\mathbf{x})$.
- Porém, $\mathbf{H}(\cdot)$ é desconhecida.
- Esse mapeamento pode representar diversos problemas de interesse prático.
  1. Aproximação de Função.

Algoritmo do Perceptron Simples (Definições Preliminares).

- Vamos assumir que existe uma lei matemática, ou função $\mathbf{H}(\cdot)$. Tal função chamada de mapeamento, que relaciona um vetor de entrada $\mathbf{x} \in \mathbb{R}^{p+1}$ qualquer com um vetor de saída $\mathbf{y} \in \mathbb{R}^{c}$. Matematicamente essa relação pode ser descrita como $\mathbf{y} = \mathbf{H}(\mathbf{x})$.
- Porém, $\mathbf{H}(\cdot)$ é desconhecida.
- Esse mapeamento pode representar diversos problemas de interesse prático.
  1. Aproximação de Função.
  2. Classificação de padrões.

Algoritmo do Perceptron Simples (Definições Preliminares).

- Vamos assumir que existe uma lei matemática, ou função $\mathbf{H}(\cdot)$. Tal função chamada de mapeamento, que relaciona um vetor de entrada $\mathbf{x} \in \mathbb{R}^{p+1}$ qualquer com um vetor de saída $\mathbf{y} \in \mathbb{R}^{c}$. Matematicamente essa relação pode ser descrita como $\mathbf{y} = \mathbf{H}(\mathbf{x})$.
- Porém, $\mathbf{H}(\cdot)$ é desconhecida.
- Esse mapeamento pode representar diversos problemas de interesse prático.
  1. Aproximação de Função.
  2. Classificação de padrões.
- Aproximação de função, a saída é quantitativa e é normalmente dada por números reais.
- Para classificação, a saída é qualitativa, muitas vezes representadas por $+1$s e -1s.
- Independente da aplicação, é de desejo a construção de um modelo adaptativo que aproxime a função $\mathbf{H}$ a partir dos pares entrada-saída.

Algoritmo do Perceptron Simples.

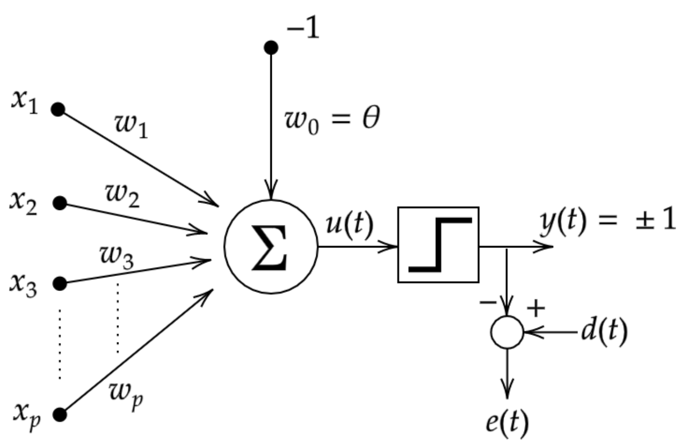
- A rede Perceptron Simples (PS) é considerada o primeiro algoritmo de redes neurais artificiais.
- Proposta em 1958 por Frank Rosenblatt em: *ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, v. 65, n. 6, p. 386, 1958.*
- Em sua versão mais simples, trata-se de um neurônio de M-P dotado de

Algoritmo do Perceptron Simples.

- A rede Perceptron Simples (PS) é considerada o primeiro algoritmo de redes neurais artificiais.
- Proposta em 1958 por Frank Rosenblatt em: *ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, v. 65, n. 6, p. 386, 1958.*
- Em sua versão mais simples, trata-se de um neurônio de M-P dotado de uma regra de aprendizagem ou algoritmo de aprendizagem.
- Tal regra é o mecanismo que torna a rede PS um dispositivo inteligente.

Algoritmo do Perceptron Simples.

- A arquitetura do neurônio da rede PS é dada por



Algoritmo do Perceptron Simples.

- Este modelo, significa que para um problema de classificação binário, tem-se:

$$\sum_{j=1}^{p} w_j x_j \geq \text{limiar} \longrightarrow Classe1.$$

$$\sum_{j=1}^{p} w_j x_j < \text{limiar} \longrightarrow Classe2.$$

- Ou então, este pode ser reescrito em uma única equação:

Algoritmo do Perceptron Simples.

- Este modelo, significa que para um problema de classificação binário, tem-se:

$$\sum_{j=1}^{p} w_j x_j \geq \text{limiar} \longrightarrow Classe1.$$

$$\sum_{j=1}^{p} w_j x_j < \text{limiar} \longrightarrow Classe2.$$

- Ou então, este pode ser reescrito em uma única equação:

$$\begin{aligned} y(t) &= sinal(u(t)) \\ &= sinal\left(\left(\sum_{j=1}^{p} w_j x_j\right) - limiar\right) \end{aligned}$$

Algoritmo do Perceptron Simples.

$$= sinal\left(\left(\sum_{j=1}^{p} w_j x_j\right) - limiar\right)$$

Algoritmo do Perceptron Simples.

$$= sinal\left(\left(\sum_{j=1}^{p} w_j x_j\right) - limiar\right)$$

- Pode-se reescrever $limiar = \theta = w_0$ e adicionar o artifício $x_0 = -1$.

$$\begin{aligned} y(t) &= sinal\left(\left(\sum_{j=1}^{p} w_j x_j\right) + \theta(-1)\right) \\ &= sinal\left(\left(\sum_{j=1}^{p} w_j x_j\right) + w_0 x_0\right) \\ &= sinal\left(\sum_{j=0}^{p} w_j x_j\right) = sinal(\mathbf{w}^T \mathbf{x}) = sinal(u(t)) \end{aligned}$$

Algoritmo do Perceptron Simples (Discriminante).

- Nesta, os vetores **x** e **w** são definidos como:

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} -1 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix} = \begin{bmatrix} \theta \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}$$

- Do ponto de vista geométrico, o que representa u(t)?

Algoritmo do Perceptron Simples (Discriminante).

- Nesta, os vetores **x** e **w** são definidos como:

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} -1 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix} = \begin{bmatrix} \theta \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}$$

- Do ponto de vista geométrico, o que representa u(t)?Exato!!!

Algoritmo do Perceptron Simples (Discriminante).

- Nesta, os vetores **x** e **w** são definidos como:

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} -1 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix} = \begin{bmatrix} \theta \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}$$

- Do ponto de vista geométrico, o que representa u(t)?Exato!!! Uma **SIMILARIDADE**.
- Sabendo disto, se o ângulo entre **x** e **w** for menor que 90^o, o que dizer sobre u(t)?

Algoritmo do Perceptron Simples (Discriminante).

- Nesta, os vetores **x** e **w** são definidos como:

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} -1 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix} = \begin{bmatrix} \theta \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}$$

- Do ponto de vista geométrico, o que representa u(t)?Exato!!! Uma **SIMILARIDADE**.
- Sabendo disto, se o ângulo entre **x** e **w** for menor que 90^o, o que dizer sobre u(t)?
- Se o ângulo entre **x** e **w** for maior que 90^o, o que dizer sobre u(t)?

Algoritmo do Perceptron Simples (Discriminante).

- Nesta, os vetores **x** e **w** são definidos como:

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} -1 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix} = \begin{bmatrix} \theta \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}$$

- Do ponto de vista geométrico, o que representa u(t)?Exato!!! Uma **SIMILARIDADE**.
- Sabendo disto, se o ângulo entre **x** e **w** for menor que 90^o, o que dizer sobre u(t)?
- Se o ângulo entre **x** e **w** for maior que 90^o, o que dizer sobre u(t)?

$$y(t) = sinal(u(t)) = \begin{cases} +1, & u(t) \geq 0 \\ -1, & u(t) < 0 \end{cases}$$

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

- O processo de aprendizagem consiste na modificação (ou ajuste) dos parâmetros do neurônio M-P, até que se consiga resolver o problema de interesse ou que se chegue ao final do período de aprendizagem. **Pergunta:** Quais são os parâmetros do modelo?

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

- O processo de aprendizagem consiste na modificação (ou ajuste) dos parâmetros do neurônio M-P, até que se consiga resolver o problema de interesse ou que se chegue ao final do período de aprendizagem. **Pergunta:** Quais são os parâmetros do modelo? **Pesos e o limiar**
- A regra de aprendizagem é uma função de dois fatores:
  1. Erro entre a saída desejada d(t) e a saída gerada pela rede y(t). Logo, $e(t) = d(t) - y(t)$.
  2. Informação fornecida pelo vetor de entrada **x**.
- O processo de aprendizagem, ou seja, o ajuste dos parâmetros do neurônio M-P, é guiado pelo erro $e(t)$ e pelo vetor de entrada **x**.

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

- Como projetar a regra de aprendizagem?
- Uma regra de aprendizagem pode ser projetada com base em:
  1. Argumentos geométricos ou empíricos.
  2. Critério de otimização de função-custo.
- Em geral, uma regra de aprendizagem tem a seguinte forma:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t)$$

- Em que:
  1. $\mathbf{w}(t)$ é o conhecimento atual (ou memória).
  2. $\Delta\mathbf{w}(t)$ informação adquirida (ou incremento na memória).
  3. $\mathbf{w}(t+1)$ memória é modificada com o acréscimo de nova informação.

$$\Delta\mathbf{w}(t) = \mathbf{F}(e(t), \mathbf{x}(t))$$

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

- Utilizaremos argumentos **geométricos** para obter a regra de aprendizagem:
- Portanto, quais os possíveis valores que a variável erro $e(t)$ pode assumir?

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

- Utilizaremos argumentos **geométricos** para obter a regra de aprendizagem:
- Portanto, quais os possíveis valores que a variável erro $e(t)$ pode assumir?
  1. $e(t) = d(t) - y(t) = 2$ ($d = +1$ e $y = -1$).

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

- Utilizaremos argumentos **geométricos** para obter a regra de aprendizagem:
- Portanto, quais os possíveis valores que a variável erro $e(t)$ pode assumir?
  1. $e(t) = d(t) - y(t) = 2$ ($d = +1$ e $y = -1$).
  2. $e(t) = d(t) - y(t) = -2$ ($d = -1$ e $y = +1$).

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

- Utilizaremos argumentos **geométricos** para obter a regra de aprendizagem:
- Portanto, quais os possíveis valores que a variável erro $e(t)$ pode assumir?
  1. $e(t) = d(t) - y(t) = 2$ ($d = +1$ e $y = -1$).
  2. $e(t) = d(t) - y(t) = -2$ ($d = -1$ e $y = +1$).
  3. $e(t) = d(t) - y(t) = 0$ ($d = -1$ e $y = -1$) ou ($d = +1$ e $y = +1$).
- Com valores iniciais para $\mathbf{w}$, o algoritmo deve testar: dado $\mathbf{x}(t)$ se $sinal(u(t)) \neq d(t)$ é verdadeiro.
- Então ajustar o vetor de pesos de acordo com:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t)$$

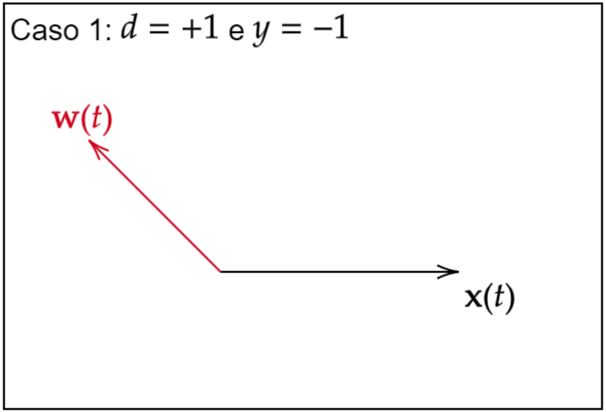
Algoritmo do Perceptron Simples (Regra de Aprendizagem).

- Utilizaremos argumentos **geométricos** para obter a regra de aprendizagem:
- Portanto, quais os possíveis valores que a variável erro $e(t)$ pode assumir?
  1. $e(t) = d(t) - y(t) = 2$ ($d = +1$ e $y = -1$).
  2. $e(t) = d(t) - y(t) = -2$ ($d = -1$ e $y = +1$).
  3. $e(t) = d(t) - y(t) = 0$ ($d = -1$ e $y = -1$) ou ($d = +1$ e $y = +1$).
- Com valores iniciais para $\mathbf{w}$, o algoritmo deve testar: dado $\mathbf{x}(t)$ se $sinal(u(t)) \neq d(t)$ é verdadeiro.
- Então ajustar o vetor de pesos de acordo com:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t)$$

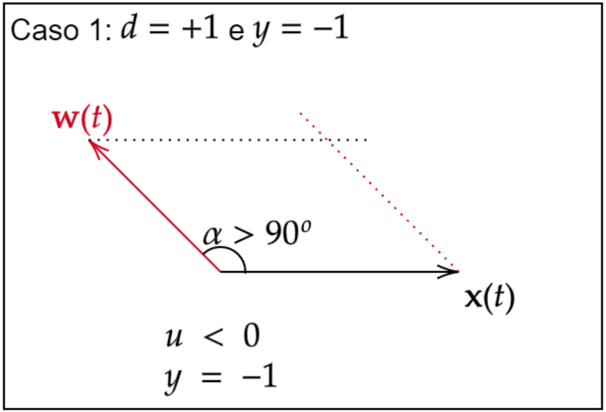
$$\color{blue}{\mathbf{w}(t+1)} = \color{red}{\mathbf{w}(t)} + \Delta\mathbf{w}(t)$$

Algoritmo do Perceptron Simples (Regra de Aprendizagem).



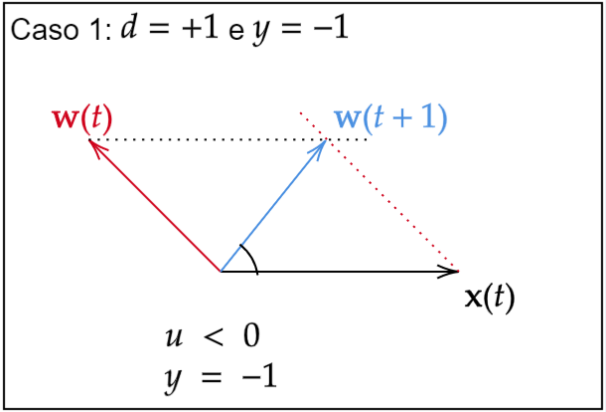
- O que pode ser feito para que y seja igual a d??

Algoritmo do Perceptron Simples (Regra de Aprendizagem).



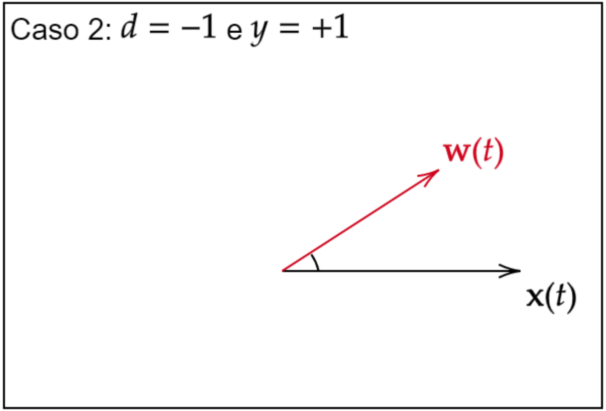
- O que pode ser feito para que y seja igual a d??

Algoritmo do Perceptron Simples (Regra de Aprendizagem).



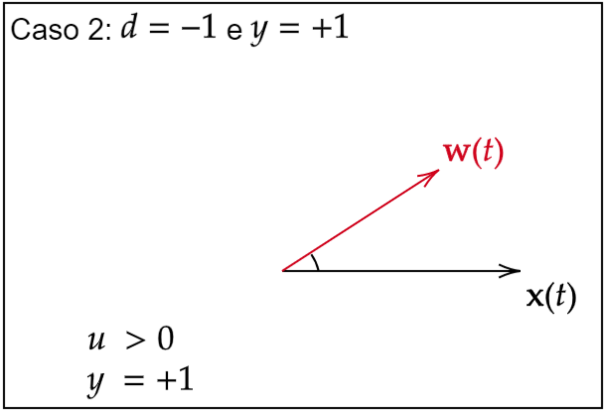
$$w(t+1) = w(t) + x(t)$$

Algoritmo do Perceptron Simples (Regra de Aprendizagem).



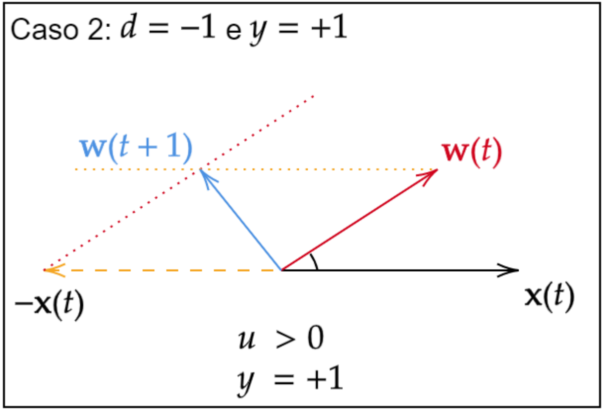
- O que pode ser feito para que y seja igual a d??

Algoritmo do Perceptron Simples (Regra de Aprendizagem).



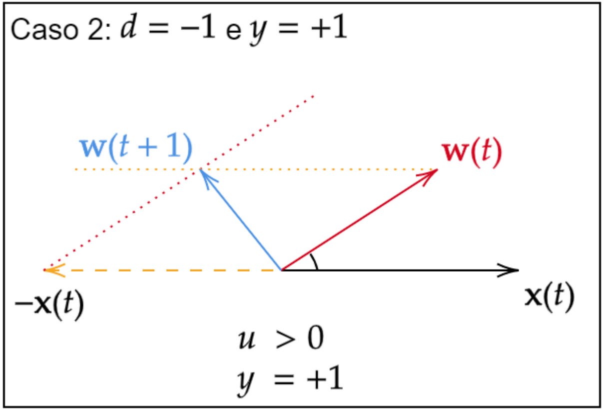
- O que pode ser feito para que y seja igual a d??

Algoritmo do Perceptron Simples (Regra de Aprendizagem).



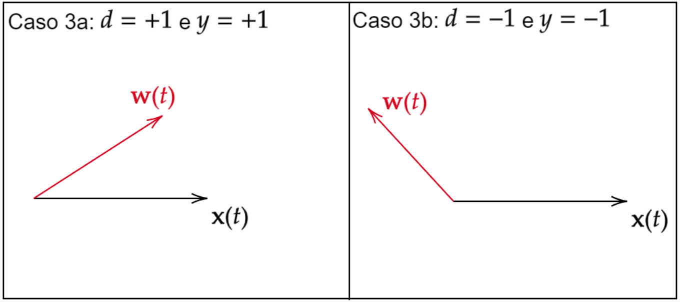
$$w(t+1) = w(t) - x(t)$$

Algoritmo do Perceptron Simples (Regra de Aprendizagem).



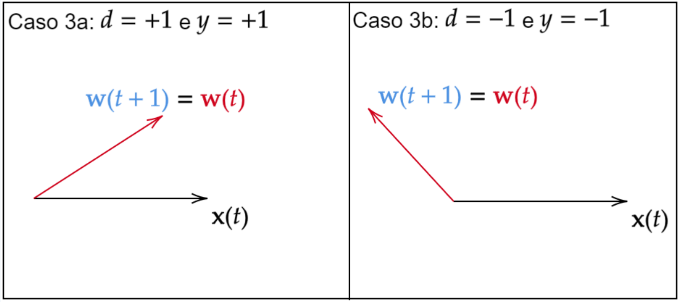
$$w(t+1) = w(t) - x(t)$$

Algoritmo do Perceptron Simples (Regra de Aprendizagem).



- O que deve ser feito nesses casos?

Algoritmo do Perceptron Simples (Regra de Aprendizagem).



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

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

- As equações mostradas na interpretação geométrica, podem ser combinadas em uma única equação dependente do erro e do vetor de entrada:

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

- As equações mostradas na interpretação geométrica, podem ser combinadas em uma única equação dependente do erro e do vetor de entrada:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + e(t)\mathbf{x}(t)$$

- Qual problemática desta abordagem?

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

- As equações mostradas na interpretação geométrica, podem ser combinadas em uma única equação dependente do erro e do vetor de entrada:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + e(t)\mathbf{x}(t)$$

- Qual problemática desta abordagem?
- Há, portanto, uma maneira de tornar o processo de ajuste do vetor **w** mais estável.
- Isto pode ser realizado, ao adicionar um fator de escala η, comumente conhecido como passo, ou taxa de aprendizagem.

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

- As equações mostradas na interpretação geométrica, podem ser combinadas em uma única equação dependente do erro e do vetor de entrada:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + e(t)\mathbf{x}(t)$$

- Qual problemática desta abordagem?
- Há, portanto, uma maneira de tornar o processo de ajuste do vetor **w** mais estável.
- Isto pode ser realizado, ao adicionar um fator de escala η, comumente conhecido como passo, ou taxa de aprendizagem.

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot e(t)\mathbf{x}(t)$$

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

- As equações mostradas na interpretação geométrica, podem ser combinadas em uma única equação dependente do erro e do vetor de entrada:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + e(t)\mathbf{x}(t)$$

- Qual problemática desta abordagem?
- Há, portanto, uma maneira de tornar o processo de ajuste do vetor **w** mais estável.
- Isto pode ser realizado, ao adicionar um fator de escala η, comumente conhecido como passo, ou taxa de aprendizagem.

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot e(t)\mathbf{x}(t)$$

- Em que $0 < \eta \leq 1$.

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

- As equações mostradas na interpretação geométrica, podem ser combinadas em uma única equação dependente do erro e do vetor de entrada:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + e(t)\mathbf{x}(t)$$

- Qual problemática desta abordagem?
- Há, portanto, uma maneira de tornar o processo de ajuste do vetor **w** mais estável.
- Isto pode ser realizado, ao adicionar um fator de escala η, comumente conhecido como passo, ou taxa de aprendizagem.

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot e(t)\mathbf{x}(t)$$

- Em que $0 < \eta \leq 1$.

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

Algorithm 1: Pseudocódigo para ajuste (fase de treinamento), do perceptron.

Início ($t = 0$)
Definir o valor de η entre 0 e 1.
Inicializar o vetor de pesos $\mathbf{w}(t)$ com valores nulos **ou** aleatórios.
ERRO $\longleftarrow$ EXISTE
while ERRO == 'EXISTENTE' **do**
    ERRO $\longleftarrow$ 'INEXISTE'.
    for Todas amostras em **x** **do**
        u(t) $\longleftarrow \mathbf{w}^T(t)\mathbf{x}(t)$
        $y(t) \longleftarrow signal(u(t))$
        $\mathbf{w}(t+1) \longleftarrow \mathbf{w}(t) + \eta(d(t) - y(t))\mathbf{x}(t)$
        if d(t)!=y(t) **then**
            ERRO $\longleftarrow$ 'EXISTENTE'
        end if
    end for
    t $\longleftarrow$ t +1
end while
FIM TREINAMENTO.

Algoritmo do Perceptron Simples (Regra de Aprendizagem).

Algorithm 2: Pseudocódigo para operação (fase de teste), do perceptron.

Obter uma amostra ($\mathbf{x}_{desconhecido}$) a ser classificada
Utilizar o vetor **w** já estimado
Realizar as seguintes operações:
u $\longleftarrow \mathbf{w}^T\mathbf{x}_{desconhecido}$
$y(t) \longleftarrow signal(u(t))$
if y==-1 **then**
    amostra percence a classe A
else
    amostra percence a classe B
end if

Exemplo

- Considere o conjunto de dados fornecido

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \\ x_{4,1} & x_{4,2} \\ x_{5,1} & x_{5,2} \\ x_{6,1} & x_{6,2} \\ x_{7,1} & x_{7,2} \\ x_{8,1} & x_{8,2} \\ x_{9,1} & x_{9,2} \\ x_{10,1} & x_{10,2} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 2 \\ 1 & 0 \\ 2 & 2 \\ 4 & 1.5 \\ 1.5 & 6 \\ 3 & 5 \\ 3 & 3 \\ 6 & 4 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

Modelo ADALINE.

- Trata-se de um tipo elementar de algoritmo adaptativo.
- Seu nome original é *ADAptive LINear Element* (ou em português, Elemento Linear Adaptativo).
- O presente modelo foi proposto por Widrow & Hoff (1960).
- O modelo ADALINE têm seus parâmetros ajustados por meio de uma regra de atualização recursiva:

Modelo ADALINE.

- Trata-se de um tipo elementar de algoritmo adaptativo.
- Seu nome original é *ADAptive LINear Element* (ou em português, Elemento Linear Adaptativo).
- O presente modelo foi proposto por Widrow & Hoff (1960).
- O modelo ADALINE têm seus parâmetros ajustados por meio de uma regra de atualização recursiva:
    1. Regra de Widrow-Hoff.

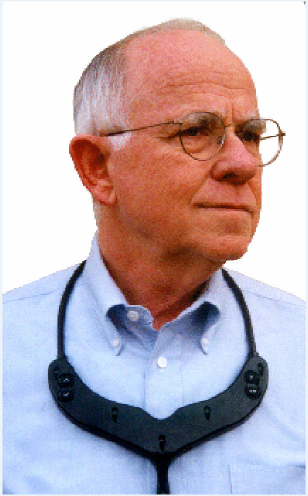
Modelo ADALINE.

- Trata-se de um tipo elementar de algoritmo adaptativo.
- Seu nome original é *ADAptive LINear Element* (ou em português, Elemento Linear Adaptativo).
- O presente modelo foi proposto por Widrow & Hoff (1960).
- O modelo ADALINE têm seus parâmetros ajustados por meio de uma regra de atualização recursiva:
  1. Regra de Widrow-Hoff.
  2. Regra Delta.

Modelo ADALINE.

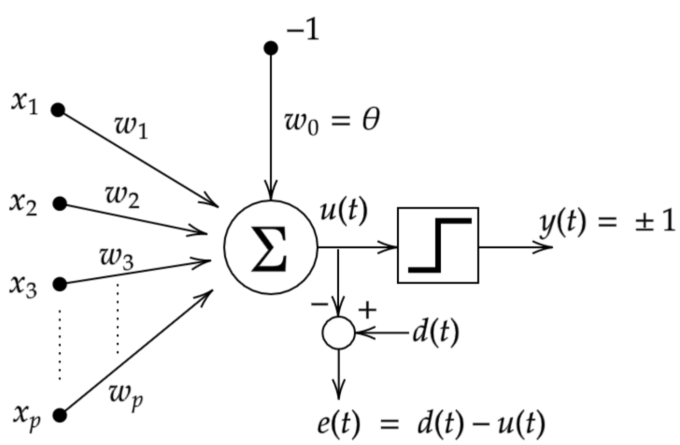
- Trata-se de um tipo elementar de algoritmo adaptativo.
- Seu nome original é *ADAptive LINear Element* (ou em português, Elemento Linear Adaptativo).
- O presente modelo foi proposto por Widrow & Hoff (1960).
- O modelo ADALINE têm seus parâmetros ajustados por meio de uma regra de atualização recursiva:
    1. Regra de Widrow-Hoff.
    2. Regra Delta.
    3. Algoritmos de adaptação LMS (Least Mean Squares).

Widrow-Hoff.




Federico Faggin
Marcian "Ted" Hoff
Stan Mazor
Masatoshi Shima

Modelo ADALINE.



- Há diferença entre o perceptron?

Modelo ADALINE.

- Em que o vetor de entradas e o de pesos, são definidos como:

$$\mathbf{x}(t) = \begin{bmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \\ \vdots \\ x_p(t) \end{bmatrix} = \begin{bmatrix} -1 \\ x_1(t) \\ x_2(t) \\ \vdots \\ x_p(t) \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix} = \begin{bmatrix} \theta \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}$$

- A saída desejada $d(t)$ do presente modelo tem qual ordem?

Modelo ADALINE.

- Em que o vetor de entradas e o de pesos, são definidos como:

$$\mathbf{x}(t) = \begin{bmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \\ \vdots \\ x_p(t) \end{bmatrix} = \begin{bmatrix} -1 \\ x_1(t) \\ x_2(t) \\ \vdots \\ x_p(t) \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix} = \begin{bmatrix} \theta \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}$$

- A saída desejada $d(t)$ do presente modelo tem qual ordem?
- Isto porque se trata apenas de **UM** único neurônio. Para uma rede com múltiplos neurônios,o modelo passa a se chamar MADALINE.
- Em princípio a saída desejada pode assumir qualquer valor real.
- Contudo, em problemas de classificação de padrões a saída desejada assume geralmente apenas dois valores $d \in \{-1, +1\}$

Modelo ADALINE.

- Quais são os parâmetros ajustáveis do modelo ADALINE?

Modelo ADALINE.

- Quais são os parâmetros ajustáveis do modelo ADALINE?
- O vetor de pesos $\mathbf{w}$.
- Qual diferença entre o PS e o ADALINE?

Modelo ADALINE.

- Quais são os parâmetros ajustáveis do modelo ADALINE?
- O vetor de pesos **w**.
- Qual diferença entre o PS e o ADALINE?

$$u(t) = \left(\sum_{j=1}^{p} w_j(t)x_j(t)\right) - \theta = \left(\sum_{j=1}^{p} w_j(t)x_j(t)\right) - w_0x_0 = \sum_{j=0}^{p} w_j(t)x_j(t) = \mathbf{w}^T(t)\mathbf{x}(t)$$

- Assim, $u(t)$ é simplesmente o produto escalar

Modelo ADALINE.

- Quais são os parâmetros ajustáveis do modelo ADALINE?
- O vetor de pesos **w**.
- Qual diferença entre o PS e o ADALINE?

$$u(t) = \left(\sum_{j=1}^{p} w_j(t) x_j(t) \right) - \theta = \left(\sum_{j=1}^{p} w_j(t) x_j(t) \right) - w_0 x_0 = \sum_{j=0}^{p} w_j(t) x_j(t) = \mathbf{w}^T(t)\mathbf{x}(t)$$

- Assim, $u(t)$ é simplesmente o produto escalar do vetor de entradas $\mathbf{x}(t)$ com o vetor de pesos $\mathbf{w}(t)$
- Como dito, $u(t) \in \mathbb{R}$, ou seja, pode assumir infinitos valores.
- Quantização escalar é o processo de transformar a saída contínua $u(t)$ em discreta $y(t) \in \{+1, -1\}$
- Já utilizamos alguma função quantizadora?

Modelo ADALINE.

- Uma função quantizadora bastante utilizada em reconhecimento de padrões é construída com a função sinal (*sign function)*
- **Dica importante:** A codificação das saídas desejadas, deve ser compatível com a saída quantizadora.

Regra de Aprendizagem.

- Definições iniciais:
  1. A precisão instantânea (ou seja, no instante t) do modelo ADALINE é medida com base no **Erro Quadrático**(EQ):

$$\varepsilon(t) = \frac{1}{2}e^2(t) = \frac{1}{2}(d(t) - u(t))^2$$

- Em que $e(t) = d(t) - u(t)$ é o erro associado à apresentação do par entrada-saída $(\mathbf{x}(t), d(t))$.

Regra de Aprendizagem.

- A regra de aprendizagem, é baseada na minimização de uma medida global do desempenho.
- Esta medida é chamada de **Erro Quadrático Médio**(EQM), que é produzida para **todos** os pares entrada-saída $(\mathbf{x}(t), d(t))$:

$$J[\mathbf{w}] = \frac{1}{N}\sum_{t=1}^{N}\varepsilon(t) = \frac{1}{2N}\sum_{t=1}^{N}e^2(t) = \frac{1}{2N}\sum_{t=1}^{N}[d(t) - u(t)]^2$$

- Em que $\mathbf{w}$ denota o conjunto de todos os parâmetros ajustáveis do modelo.
- Os parâmetros do modelo ADALINE devem ser especificados de modo que este produza uma saída bem próxima da esperada para um vetor de entrada $\mathbf{x}(t)$.
- Ou seja, identificar um $\mathbf{w}$ ótimo ($\mathbf{w}^*$) que minimize o EQM.

Regra de Aprendizagem.

- Um procedimento iterativo de se chegar aos parâmetros ótimos envolve o uso da equação recursiva:

$$w_j(t+1) = w_j(t) - \eta \frac{\partial \varepsilon(t)}{\partial w_j(t)}$$

- Em que η é a taxa de aprendizagem $0 < \eta < 1$.
- Utilizando a regra da cadeia, na derivada exibida, pode-se fazer:

Regra de Aprendizagem.

- Um procedimento iterativo de se chegar aos parâmetros ótimos envolve o uso da equação recursiva:

$$w_j(t+1) = w_j(t) - \eta \frac{\partial \varepsilon(t)}{\partial w_j(t)}$$

- Em que η é a taxa de aprendizagem $0 < \eta < 1$.
- Utilizando a regra da cadeia, na derivada exibida, pode-se fazer:

$$\frac{\partial \varepsilon(t)}{\partial w_j(t)} = \frac{\partial \varepsilon(t)}{\partial e(t)} \cdot \frac{\partial e(t)}{\partial u(t)} \cdot \frac{\partial u(t)}{\partial w_j(t)}$$

Regra de Aprendizagem.

$$\frac{\partial \varepsilon(t)}{\partial e(t)} = e(t)$$

$$\frac{\partial e(t)}{\partial u(t)} = -1$$

$$\frac{\partial u(t)}{\partial w_j(t)} = x_j(t)$$

- Assim, a regra recursiva de ajuste dos pesos é dada por:

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

$$w_j(t+1) = w_j(t) + \eta e(t) x_j(t)$$

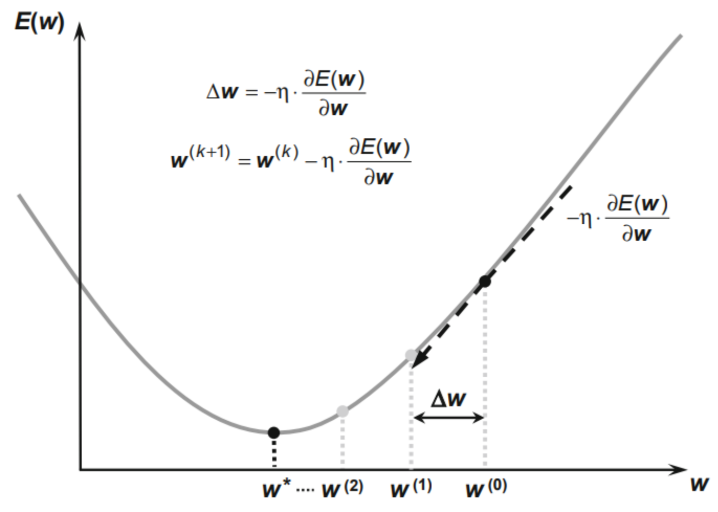
Regra de Aprendizagem.

- De maneira vetorial, a regra de ajuste do pesos pode ser escrita como:

$$\begin{aligned}\mathbf{w}(t+1) &= \mathbf{w}(t) + \eta \Delta J[\mathbf{w}] \\ &= \mathbf{w}(t) + \eta e(t)\mathbf{x}(t)\end{aligned}$$

- É possível verificar a interpretação geométrica deste ajuste de pesos da seguinte maneira:

Regra de Aprendizagem.



Regra de Aprendizagem.

- Pontos importantes:
  1. Em sua etapa de treinamento, o modelo ADALINE deve ser interrompido quando a convergência acontecer, ou seja: Quando a diferença dos EQM entre duas épocas sucessivas for suficientemente pequeno:

$$|EQM_{atual} - EQM_{anterior}| \leq \epsilon$$

  2. Onde ϵ é a precisão a ser definida pelo projetista da rede ADALINE.
- Outro critério de parada, envolve simplesmente a definição de um número máximo de épocas.
- É de interesse verificar a curva de aprendizagem do modelo, dado o problema proposto. Para tal feito, deve-se para cada época plotar valores dos EQM cálculados.

Algoritmo ADALINE (Treinamento).

Algorithm 3: Pseudocódigo para ajuste (fase de treinamento), do ADALINE.

Definir o valor de η, número máximo de épocas e precisão (ϵ).
Inicializar o vetor de pesos $\mathbf{w}(t)$ com valores nulos **ou** aleatórios.
Iniciar o contador de épocas ($epoch \longleftarrow 0$)
repeat
    $EQM_{anterior} \longleftarrow EQM(\mathbf{x}, d, \mathbf{w})$
    for todas as N amostras de treinamento **do**
        $u(t) \longleftarrow \mathbf{w}^T(t)\mathbf{x}(t)$
        $\mathbf{w}(t+1) \longleftarrow \mathbf{w}(t) + \eta(d(t) - u(t))\mathbf{x}(t)$
    end for
    $epoch \longleftarrow epoch + 1$
    $EQM_{atual} \longleftarrow EQM(\mathbf{x}, d, \mathbf{w})$
until $|EQM_{atua} - EQM_{anterior}| \leq \epsilon$ OU Número máximo de épocas atingido

Algoritmo ADALINE (Treinamento).

Algorithm 4: Algoritmo para cálculo do *EQM*.

$EQM \longleftarrow 0$
for todas as amostras de treinamento **do**
    $u(t) \longleftarrow \mathbf{w}^T(t)\mathbf{x}(t)$
    $EQM \longleftarrow EQM + (d(t) - u(t))^2$
end for
$EQM \longleftarrow \frac{EQM}{2N}$

Algoritmo do ADALINE (Regra de Aprendizagem).

Algorithm 5: Pseudocódigo para operação (fase de teste), do ADALINE.

Obter uma amostra ($\mathbf{x}_{desconhecido}$) a ser classificada
Utilizar o vetor **w** já estimado
Realizar as seguintes operações:
$\text{u} \longleftarrow \mathbf{w}^T \mathbf{x}_{desconhecido}$
$y(t) \longleftarrow signal(u(t))$
if y==-1 **then**
    amostra percence a classe A
else
    amostra percence a classe B
end if