REDES NEURAIS ARTIFICIAIS

AULA 2 – REDES *PERCEPTRON*

Prof. Rodrigo Palácios

rodrigopalacios@utfpr.edu.br

REDES *PERCEPTRON*

- Características:
  - É a forma mais simples de configuração de uma rede neural artificial (idealizada por Rosenblatt, 1958).
  - Constituída de apenas uma camada, tendo-se ainda somente um neurônio nesta única camada.
  - Seu propósito inicial era implementar um modelo computacional inspirado na retina, objetivando-se então um elemento de percepção eletrônica de sinais.
  - Suas aplicações consistiam de identificar padrões geométricos.

REDES *PERCEPTRON* - CONCEPÇÃO INICIAL

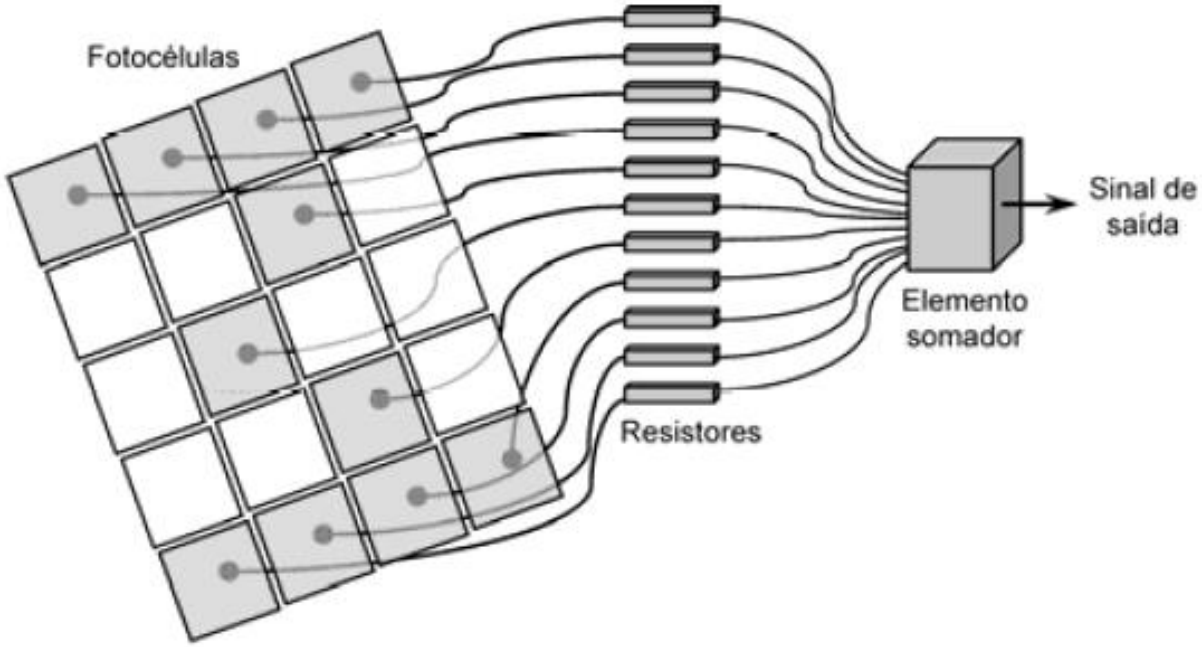
- Modelo ilustrativo do ***Perceptron*** para reconhecimento de padrões:

  1) Sinais elétricos advindos de fotocélulas mapeando padrões geométricos eram ponderados por resistores sintonizáveis.

  2) Os resistores eram ajustados durante o processo de treinamento.

  3) Um somador efetuava a composição de todos os sinais.

  4) Em consequência, o ***Perceptron*** poderia reconhecer diversos padrões geométricos, tais como letras e números.

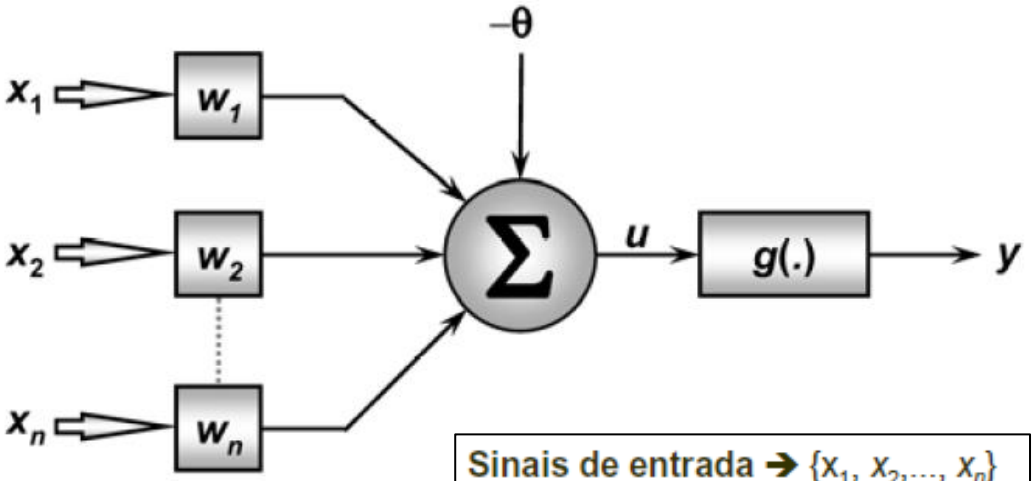


REDES *PERCEPTRON* - ASPECTOS TOPOLÓGICOS

1) Embora seja uma rede simples, o ***Perceptron*** teve potencial de atrair, quando de sua proposição, diversos pesquisadores que aspiravam investigar essa promissora área de pesquisa.

2) Recebeu ainda especial atenção da comunidade científica que também trabalhava com inteligência artificial.

3) O ***Perceptron*** é tipicamente utilizado em problemas de "Classificação de Padrões".



Sinais de entrada ➔ $\{x_1, x_2, \ldots, x_n\}$
Pesos sinápticos ➔ $\{w_1, w_2, \ldots, w_n\}$
Combinador linear ➔ $\{\Sigma\}$
Limiar de ativação ➔ $\{\theta\}$
Potencial de ativação ➔ $\{u\}$
Função de ativação ➔ $\{g\}$
Sinal de saída ➔ $\{y\}$

REDES *PERCEPTRON* - FUNCIONAMENTO

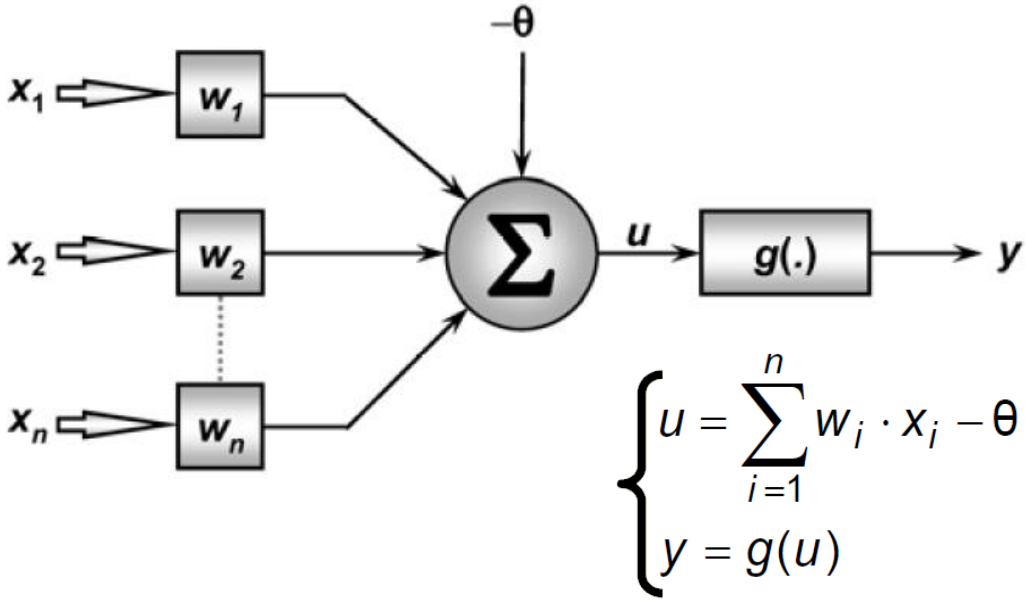
1) Apresentação de um conjunto de valores que representam as variáveis de entrada do neurônio.

2) Multiplicação de cada entrada do neurônio pelo seu respectivo peso sináptico.

3) Obtenção do potencial de ativação produzido pela soma ponderada dos sinais de entrada, subtraindo-se o limiar de ativação.

4) Aplicação de uma função de ativação apropriada, tendo-se como objetivo limitar a saída do neurônio.

5) Compilação da saída a partir da aplicação da função de ativação neural em relação ao seu potencial de ativação.

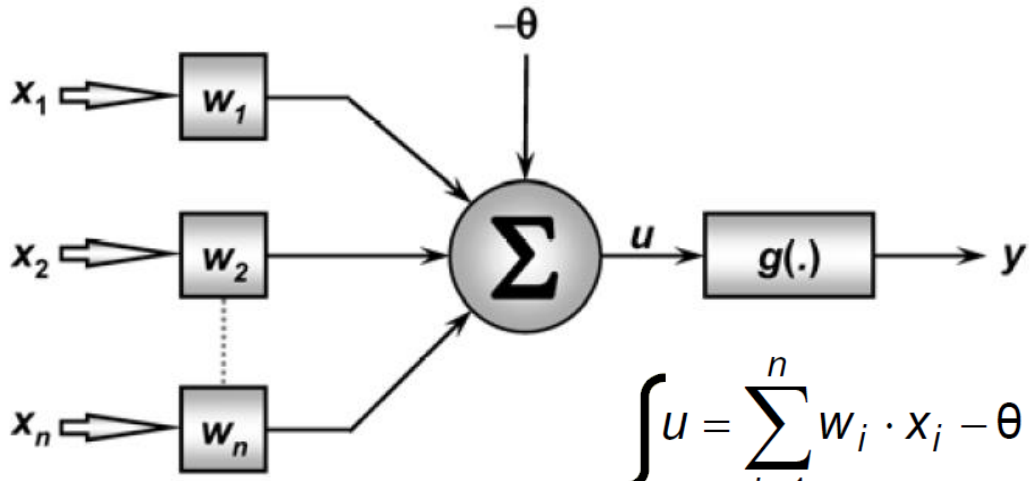


$$\begin{cases} u = \sum_{i=1}^{n} w_i \cdot x_i - \theta \\ y = g(u) \end{cases}$$

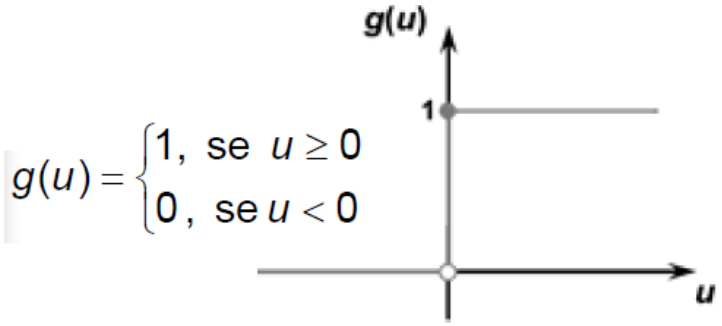
REDES *PERCEPTRON* - APLICABILIDADE

1) Tipicamente, devido às suas características estruturais, as funções de ativação usadas no ***Perceptron*** são a "degrau" ou "degrau bipolar".

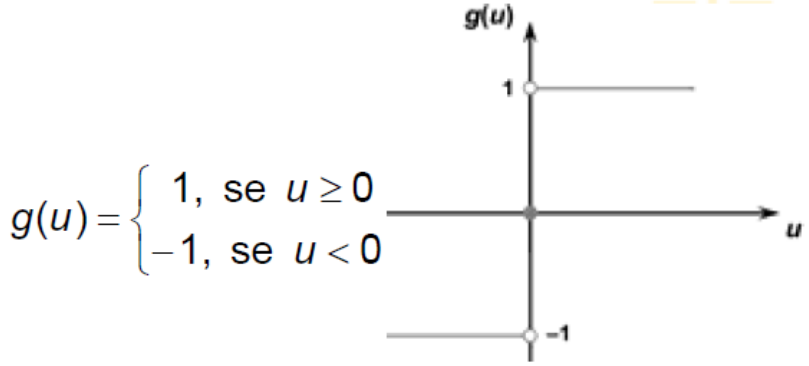
2) Assim, tem-se apenas "duas possibilidades" de valores a serem produzidos pela sua saída, ou seja, valor 0 ou 1 (para a função de ativação



$$\begin{cases} u = \sum_{i=1}^{n} w_i \cdot x_i - \theta \\ y = g(u) \end{cases}$$

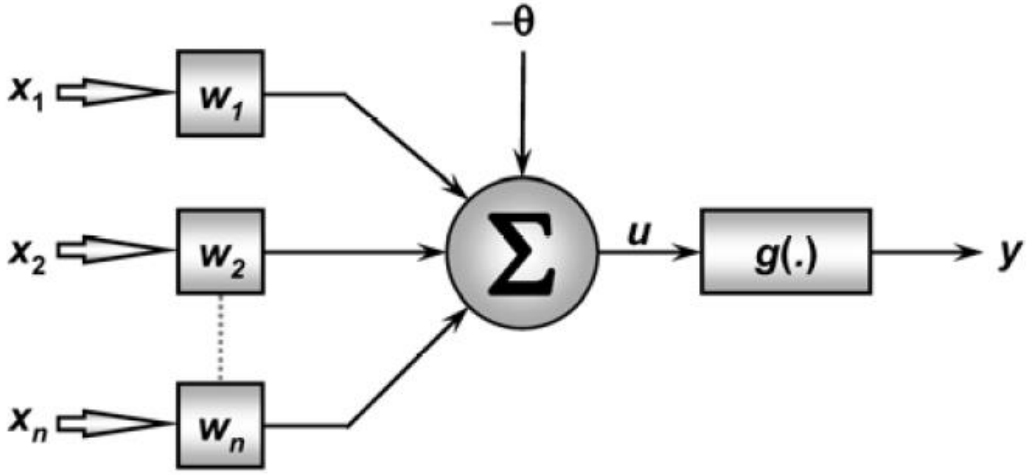


Degrau



Degrau Bipolar

REDES *PERCEPTRON* - APLICABILIDADE



Sinais de entrada ➔ $\{x_1, x_2, ..., x_n\}$
Pesos sinápticos ➔ $\{w_1, w_2, ..., w_n\}$
Combinador linear ➔ $\{\Sigma\}$
Limiar de ativação ➔ $\{\theta\}$
Potencial de ativação ➔ $\{u\}$
Função de ativação ➔ $\{g\}$
Sinal de saída ➔ $\{y\}$

Parâmetro	Variável Representativa	Tipo Característico
Entradas	x_i (i-ésima entrada)	Reais ou Binária (advindas externamente)
Pesos Sinápticos	w_i (associado a x_i)	Reais (iniciados aleatoriamente)
Limiar	θ	Real (iniciado aleatoriamente)
Saída	y	Binária
Função de Ativação	$g(.)$	Degrau ou Degrau Bipolar
Processo de Treinamento	-----	Supervisionado
Regra de Aprendizado	-----	Regra de Hebb

REDES *PERCEPTRON* - TREINAMENTO SUPERVISIONADO

Parâmetro	Variável Representativa	Tipo Característico
Entradas	x_i (i-ésima entrada)	Reais ou Binária (advindas externamente)
Pesos Sinápticos	w_i (associado a x_i)	Reais (iniciados aleatoriamente)
Limiar	θ	Real (iniciado aleatoriamente)
Saída	y	Binária
Função de Ativação	$g(.)$	**Degrau ou Degrau Bipolar**
Processo de Treinamento	**-----**	**Supervisionado**
Regra de Aprendizado	-----	Regra de Hebb

1) Conforme tabela apresentada, o ajuste dos pesos e limiar do ***Perceptron*** é efetuado utilizando processo de treinamento "Supervisionado".

2) Então, para cada amostra dos sinais de entrada se tem a respectiva saída (resposta) desejada.

3) Como o ***Perceptron*** é tipicamente usado em problemas de classificação de padrões, a sua saída pode assumir somente dois valores possíveis.

4) Assim, cada um de tais valores será associado a uma das "duas classes" que o ***Perceptron*** estará identificando.

REDES *PERCEPTRON* - MAPEAMENTO DE PROBLEMAS PARA CLASSIFICAÇÃO PADRÕES

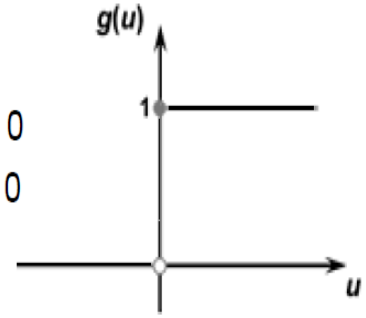
Aspectos de Aplicabilidade

1) Portanto, para problemas de classificação dos sinais de entrada, tem-se então duas classes possíveis, denominadas de *Classe A* e *Classe B*;

2) Paralelamente, como se tem também "duas possibilidades" de valores a serem produzidos na saída do ***Perceptron***, tem-se as seguintes associações:

Degrau

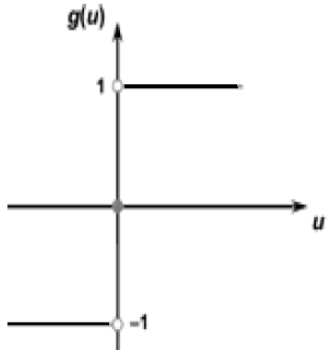
$$g(u) = \begin{cases} 1, \text{ se } u \geq 0 \\ 0, \text{ se } u < 0 \end{cases}$$



$$g(u) = \begin{cases} 1, \text{ se } u \geq 0 \Rightarrow \boldsymbol{x} \in \textit{Classe A} \\ 0, \text{ se } u < 0 \Rightarrow \boldsymbol{x} \in \textit{Classe B} \end{cases}$$

Degrau Bipolar

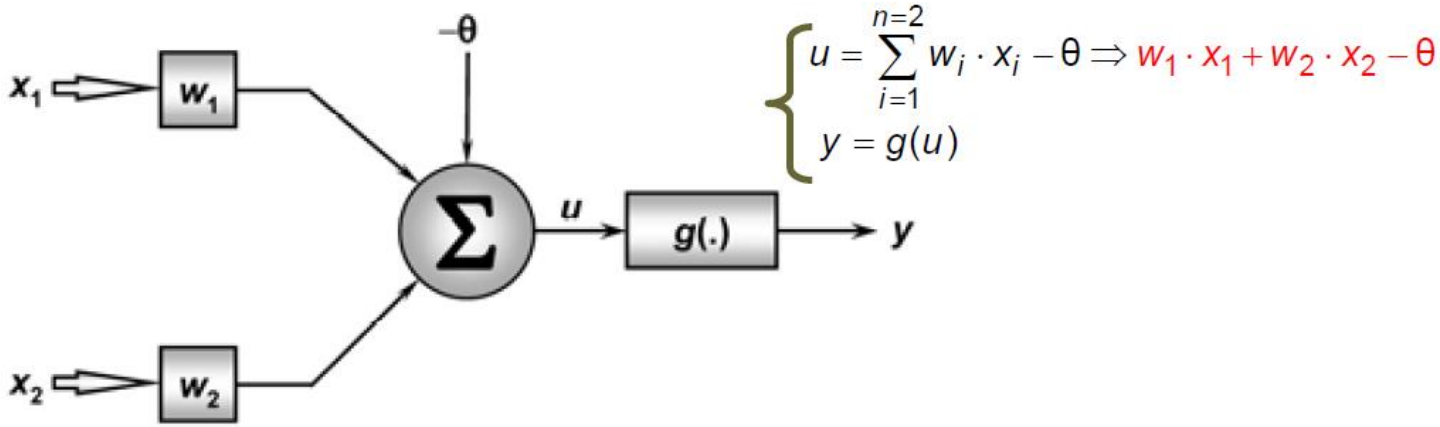
$$g(u) = \begin{cases} 1, \text{ se } u \geq 0 \\ -1, \text{ se } u < 0 \end{cases}$$



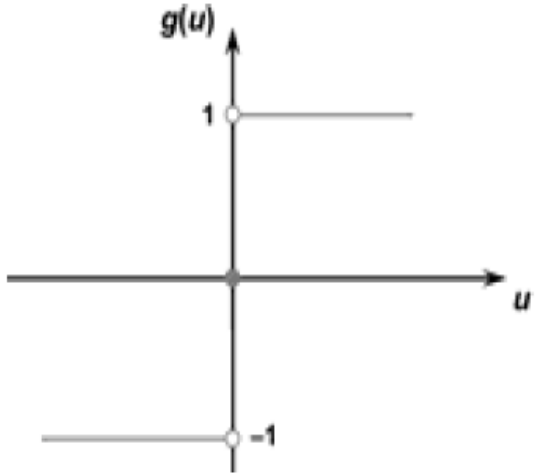
$$g(u) = \begin{cases} 1, \text{ se } u \geq 0 \Rightarrow \boldsymbol{x} \in \textit{Classe A} \\ -1, \text{ se } u < 0 \Rightarrow \boldsymbol{x} \in \textit{Classe B} \end{cases}$$

REDES *PERCEPTRON* - ANÁLISE MATEMÁTICA

- Para mostrar o quê ocorre internamente, assume-se aqui um ***Perceptron*** com apenas duas entradas:



- Usando como ativação a função sinal, a saída do ***Perceptron*** será dada por:



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

$$y = g(u) = \begin{cases} 1, & \text{se } \sum w_i \cdot x_i - \theta \geq 0 \Leftrightarrow w_1 \cdot x_1 + w_2 \cdot x_2 - \theta \geq 0 \\ -1, & \text{se } \sum w_i \cdot x_i - \theta < 0 \Leftrightarrow w_1 \cdot x_1 + w_2 \cdot x_2 - \theta < 0 \end{cases}$$

Conclusão: O ***Perceptron*** é um classificador linear que pode ser usado para classificar duas classes linearmente separáveis.

REDES *PERCEPTRON* - CLASSIFICAÇÃO DE PADRÕES

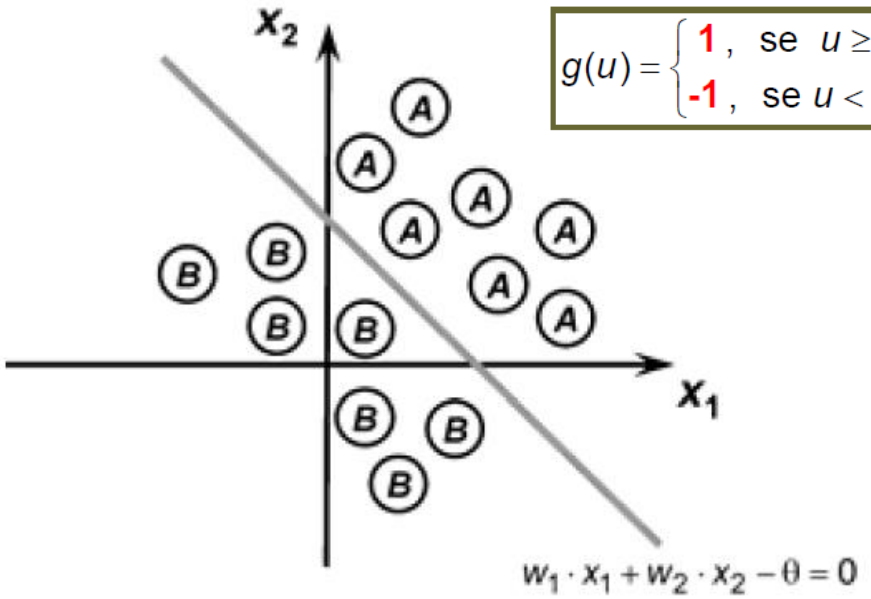
- Da conclusão do slide anterior, observam-se que as desigualdades são representadas por uma expressão de primeiro grau (linear).
- A fronteira de decisão para esta instância (***Perceptron*** de duas entradas) será então uma reta cuja equação é definida por:

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

$$y = g(u) = \begin{cases} 1, \text{ se } \sum w_i \cdot x_i - \theta \geq 0 \Leftrightarrow w_1 \cdot x_1 + w_2 \cdot x_2 - \theta \geq 0 \\ -1, \text{ se } \sum w_i \cdot x_i - \theta < 0 \Leftrightarrow w_1 \cdot x_1 + w_2 \cdot x_2 - \theta < 0 \end{cases}$$

- Assim, tem-se a seguinte representação gráfica para a saída do ***Perceptron***:

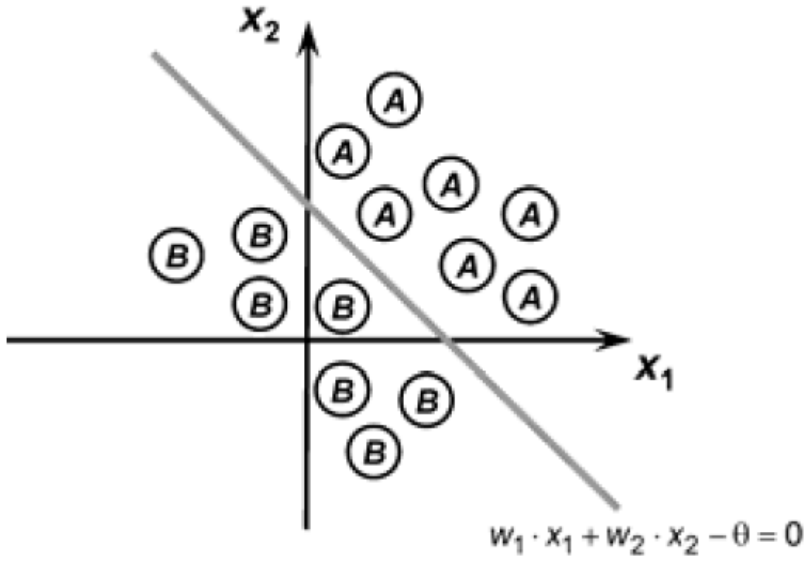
$$g(u) = \begin{cases} 1, \text{ se } u \geq 0 \Rightarrow \boldsymbol{x} \in \textit{Classe A} \\ -1, \text{ se } u < 0 \Rightarrow \boldsymbol{x} \in \textit{Classe B} \end{cases}$$



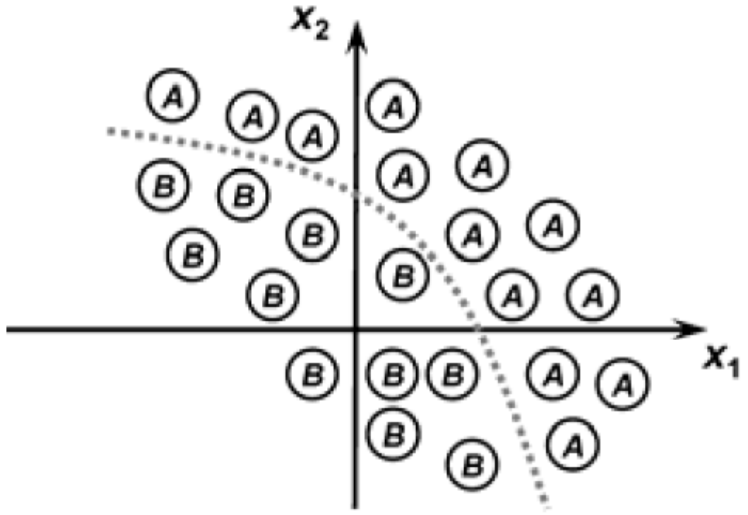
- Em suma, para a circunstância ao lado, o ***Perceptron*** consegue então dividir duas classes linearmente separáveis
- A saída do mesmo for 1 significa que os padrões (Classe A) estão localizados acima da fronteira (reta) de separação; caso contrário, quando a saída for -1 indica que os padrões (Classe B) estão abaixo desta fronteira.

REDES *PERCEPTRON* - CLASSIFICAÇÃO DE PADRÕES

- Se o ***Perceptron*** fosse constituído de três entradas (três dimensões), a fronteira de separação seria representada por um plano.
- Se o ***Perceptron*** fosse constituído de quatro ou mais entradas, suas fronteiras seriam hiperplanos.



Problema Linearmente Separável



Problema Não-Linearmente Separável

Conclusão: A condição necessária para que o ***Perceptron*** de camada simples possa ser utilizado como um classificador de padrões é que as classes do problema a ser mapeado sejam **linearmente separáveis**.

REDES *PERCEPTRON* - PROCESSO DE TREINAMENTO

- **Aspectos do Processo de Treinamento (Regra de Aprendizado de Hebb)**
- O ajuste dos pesos $\{w_i\}$ e limiar $\{\theta\}$ do ***Perceptron***, visando-se propósitos de classificação de padrões que podem pertencer a uma das duas únicas classes possíveis, é feito por meio da **regra de aprendizado de Hebb**.
- Se a saída produzida pelo ***Perceptron*** está não coincidente com a saída desejada, os seus pesos sinápticos e limiares são então incrementados (**ajuste excitatório**) proporcionalmente aos valores de suas entradas.
- Se a saída produzida pelo ***Perceptron*** é diferente do valor desejado, os pesos sinápticos e limiar serão então decrementados (**ajuste inibitório**).
- Este processo é repetido, sequencialmente, para todas as amostras de treinamento, até que a saída produzida pelo ***Perceptron*** seja similar à saída desejada de cada amostra.

- Em termos matemáticos, tem-se:

$$\begin{cases} w_i^{atual} = w_i^{anterior} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)} \\ \theta_i^{atual} = \theta_i^{anterior} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)} \end{cases}$$

- w_i são os pesos sinápticos.
- θ é limiar do neurônio.
- $\boldsymbol{x}^{(k)}$ é o vetor contendo a k-ésima amostra de treinamento
- $d^{(k)}$ é saída desejada para a k-ésima amostra de treinamento.
- y é a saída do ***Perceptron***.
- η é a taxa de aprendizagem da rede.

A taxa de aprendizagem $\{\eta\}$ exprime o quão rápido o processo de treinamento da rede estará sendo conduzido rumo à sua convergência.

REDES *PERCEPTRON* - PROCESSO DE TREINAMENTO

- **Aspectos de implementação computacional (adequações algorítmicas)**
  - Em termos de implementação, torna-se mais conveniente tratar as expressões anteriores em sua forma vetorial.
  - Como a mesma regra de ajuste é aplicada tanto para os pesos w_i como para o limiar θ, pode-se então inserir θ dentro do vetor de pesos:

$$\boldsymbol{w} = [\theta \quad w_1 \quad w_2 \ldots w_n]^T$$

- De fato, o valor do limiar é também uma variável a ser ajustada a fim de se realizar o treinamento do ***Perceptron***. Portanto, tem-se:

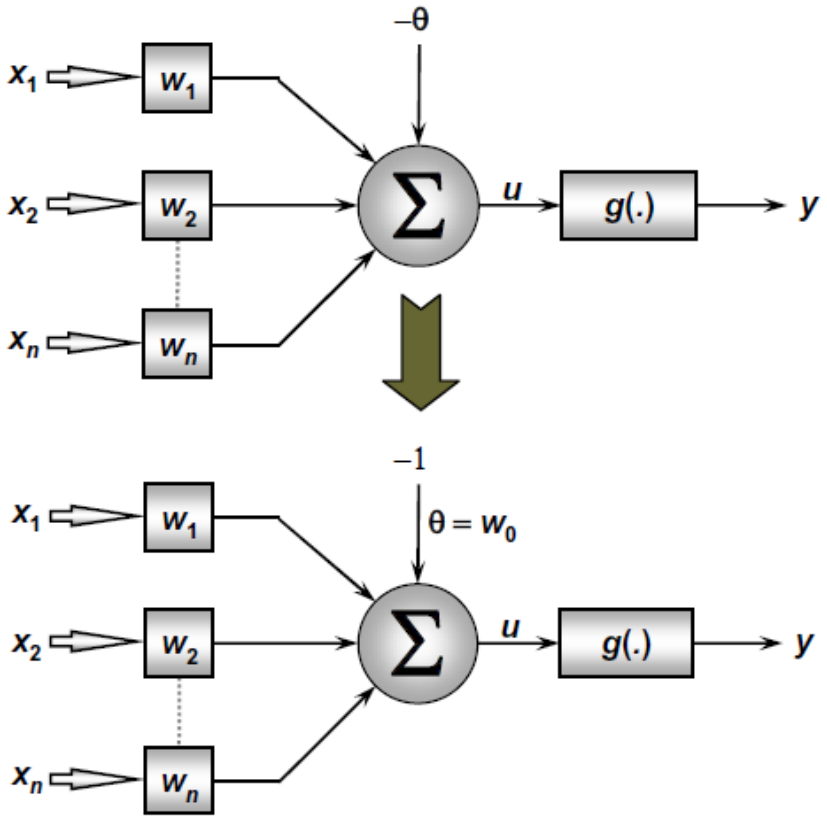
$$\begin{cases} w_i^{atual} = w_i^{anterior} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)} \\ \theta_i^{atual} = \theta_i^{anterior} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)} \end{cases}$$

notação vetorial

$$\boldsymbol{w}^{atual} = \boldsymbol{w}^{anterior} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)}$$

notação algorítmica

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)}$$



Formato de cada amostra de treinamento:

$$\boldsymbol{x}^{(k)} = [-1 \quad x_1^{(k)} \quad x_2^{(k)} \cdots x_n^{(k)}]^T$$

REDES *PERCEPTRON* - PROCESSO DE TREINAMENTO

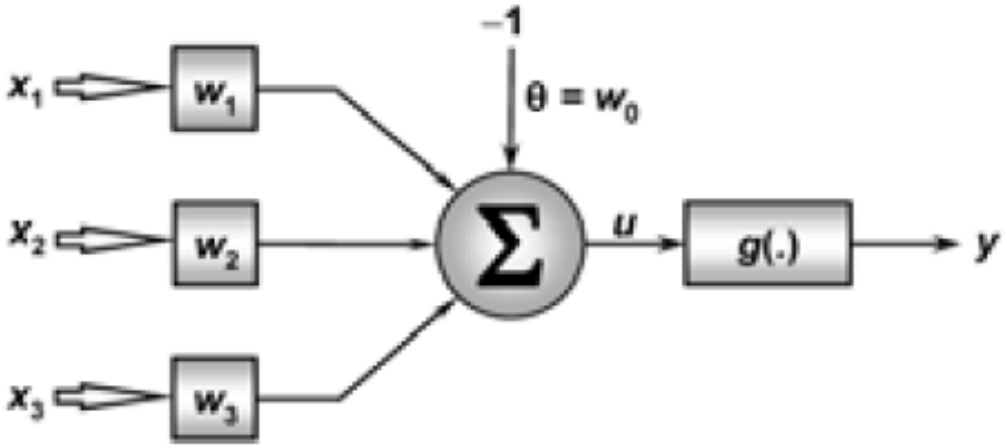
- **Aspectos de implementação computacional (montagem de conjuntos de treinamento)**
  - Supõe-se que um problema a ser mapeado pelo ***Perceptron*** tenha três entradas $\{x_1, x_2, x_3\}$, conforme a figura ao lado (abaixo).
  - Assume-se que se tem quatro amostras, constituída dos seguintes valores de entrada:

Amostra 1 → Entrada: [0,1 0,4 0,7] → Saída desejada: [1]

Amostra 2 → Entrada: [0,3 0,7 0,2] → Saída desejada: [-1]

Amostra 3 → Entrada: [0,6 0,9 0,8] → Saída desejada: [-1]

Amostra 4 → Entrada: [0,5 0,7 0,1] → Saída desejada: [1]



REDES *PERCEPTRON* - PROCESSO DE TREINAMENTO

- **Aspectos de implementação computacional (montagem de conjuntos de treinamento)**
  - Então, pode-se converter tais sinais para que os mesmos possam ser usados no treinamento do ***Perceptron***:

Conjunto de treinamento

$\boldsymbol{x}^{(1)} = [-1 \quad 0{,}1 \quad 0{,}4 \quad 0{,}7]^T \rightarrow \text{com } d^{(1)} = 1$

$\boldsymbol{x}^{(2)} = [-1 \quad 0{,}3 \quad 0{,}7 \quad 0{,}2]^T \rightarrow \text{com } d^{(2)} = -1$

$\boldsymbol{x}^{(3)} = [-1 \quad 0{,}6 \quad 0{,}9 \quad 0{,}8]^T \rightarrow \text{com } d^{(3)} = -1$

$\boldsymbol{x}^{(4)} = [-1 \quad 0{,}5 \quad 0{,}7 \quad 0{,}1]^T \rightarrow \text{com } d^{(4)} = 1$

forma matricial

$$\Omega^{(x)} = \begin{matrix} \\ x_0 \\ x_1 \\ x_2 \\ x_3 \end{matrix} \begin{matrix} \boldsymbol{x}^{(1)} & \boldsymbol{x}^{(2)} & \boldsymbol{x}^{(3)} & \boldsymbol{x}^{(4)} \\ \end{matrix} \begin{bmatrix} -1 & -1 & -1 & -1 \\ 0{,}1 & 0{,}3 & 0{,}6 & 0{,}5 \\ 0{,}4 & 0{,}7 & 0{,}9 & 0{,}7 \\ 0{,}7 & 0{,}2 & 0{,}8 & 0{,}1 \end{bmatrix};$$

$$\Omega^{(d)} = \begin{matrix} \boldsymbol{d}^{(1)} & \boldsymbol{d}^{(2)} & \boldsymbol{d}^{(3)} & \boldsymbol{d}^{(4)} \end{matrix} \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix}$$

- Geralmente, as amostras de treinamento são disponibilizadas em sua forma matricial (por meio de arquivo texto ou planilha).

REDES *PERCEPTRON* - PROCESSO DE TREINAMENTO

- **Algoritmo de Aprendizagem (fase de treinamento)**

Início {Algoritmo *Perceptron* – Fase de Treinamento}

<1> Obter o conjunto de amostras de treinamento $\{ \boldsymbol{x}^{(k)} \}$;

<2> Associar a saída desejada $\{ d^{(k)} \}$ para cada amostra obtida;

<3> Iniciar o vetor $\boldsymbol{w}$ com valores aleatórios pequenos;

<4> Especificar a taxa de aprendizagem $\{\eta\}$;

<5> Iniciar o contador de número de épocas $\{época \leftarrow 0\}$;

<6> Repetir as instruções:

<6.1> $erro \leftarrow$ "inexiste";

<6.2> Para todas as amostras de treinamento $\{ \boldsymbol{x}^{(k)}, d^{(k)} \}$, fazer:

<6.2.1> $u \leftarrow \boldsymbol{w}^T \cdot \boldsymbol{x}^{(k)} - (\theta)$;

<6.2.2> $y \leftarrow \text{sinal}(u)$;

<6.2.3> Se $y \neq d^{(k)}$

<6.2.3.1> Então
$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)}$
$erro \leftarrow$ "existe"

<6.3> $época \leftarrow época + 1$;

Até que: $erro \leftarrow$ "inexiste"

Fim {Algoritmo *Perceptron* – Fase de Treinamento}

Época de treinamento → É cada apresentação completa de todas as amostras pertencentes ao subconjunto de treinamento, visando, sobretudo, o ajuste dos pesos sinápticos e limiares de seus neurônios.

REDES *PERCEPTRON* - PROCESSO DE TREINAMENTO

- **Pseudocódigo para fase de operação**

Início {Algoritmo *Perceptron* – Fase de Operação}

<1> Obter uma amostra a ser classificada { $\boldsymbol{x}$ };

<2> Utilizar o vetor $\boldsymbol{w}$ ajustado durante o treinamento;

<3> Executar as seguintes instruções:

<3.1> $u \leftarrow \boldsymbol{w}^T \cdot \boldsymbol{x}$;

<3.2> $y \leftarrow \text{sinal}(u)$;

<3.3> Se $y = -1$

<3.3.1> Então: amostra $\boldsymbol{x} \in$ *{Classe A}*

<3.4> Se $y = 1$

<3.4.1> Então: amostra $\boldsymbol{x} \in$ *{Classe B}*

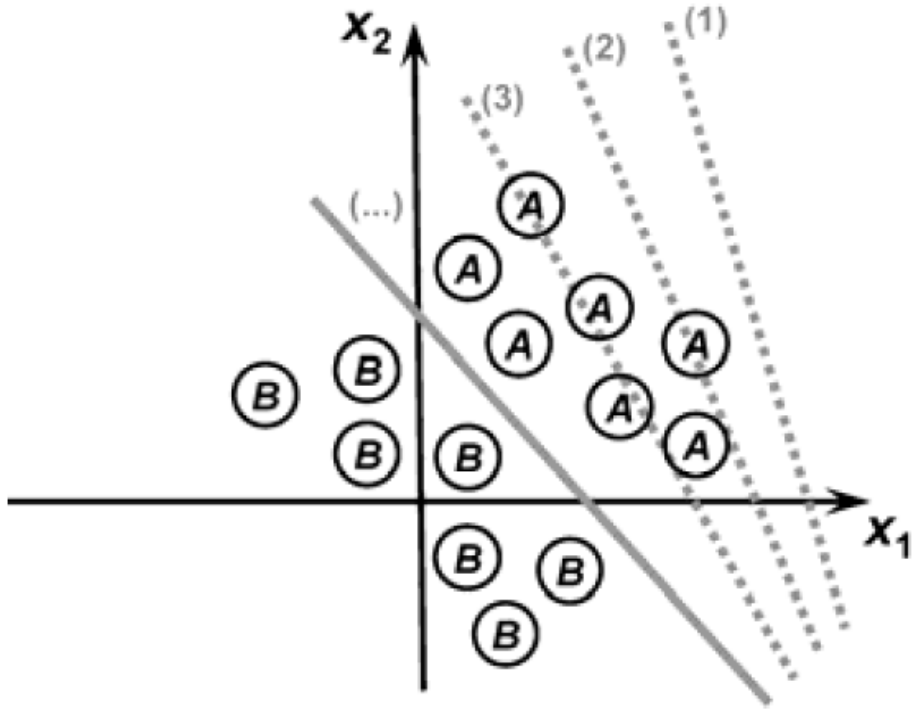
Fim {Algoritmo *Perceptron* – Fase de Operação}

Obs. 1: A "Fase de Operação" é usada somente após a fase de treinamento, pois aqui a rede já está apta para ser usada no processo.

Obs. 2: A "Fase de Operação" é então utilizada para realizar a tarefa de classificação de padrões frente às novas amostras que serão apresentadas em suas entradas.

REDES *PERCEPTRON* - PROCESSO DE TREINAMENTO

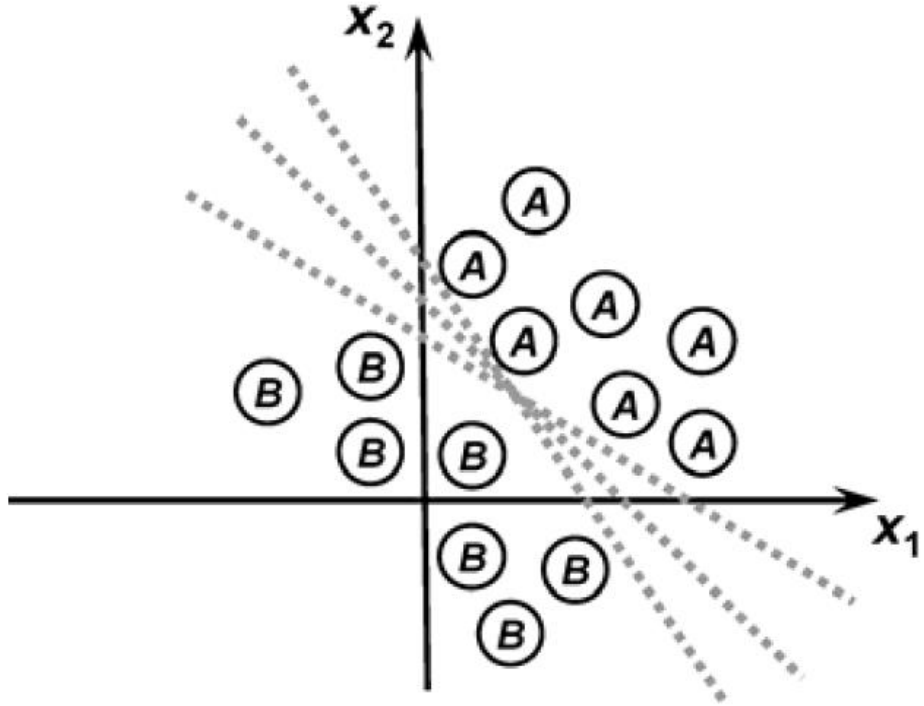
- **Ilustração do processo de convergência**
  - Processo de treinamento tende a mover continuamente o hiperplano de classificação até que seja alcançada uma fronteira de separação que permite dividir as duas classes.
  - Como exemplo, seja uma rede composta de apenas duas entradas { x_1 e x_2 }.



- Após a primeira época de treinamento (1), constata-se que o hiperplano está ainda bem longínquo da fronteira de separabilidade das classes.
- A distância tende ir cada vez mais decrescendo na medida em que se transcorre as épocas de treinamento.
- Quando o ***Perceptron*** já estiver convergido, isto estará então significando que tal fronteira foi finalmente alcançada.

REDES *PERCEPTRON* - PROCESSO DE TREINAMENTO

- **Região de separabilidade (aspectos de convergência)**
  - A reta de separabilidade a ser produzida após o treinamento do ***Perceptron*** não é única.
  - O número de épocas pode também variar de treinamento para treinamento.



REFERÊNCIA

- SILVA, Ivan Nunes da e SPATTI, Danilo Hernane e FLAUZINO, Rogério Andrade. Redes neurais artificiais para engenharia e ciências aplicadas. . São Paulo: Artliber Editora. . Acesso em: 23 dez. 2023. , 2010