# Inteligência Artificial

# Redes Neurais Artificiais
# Perceptron e Adaline

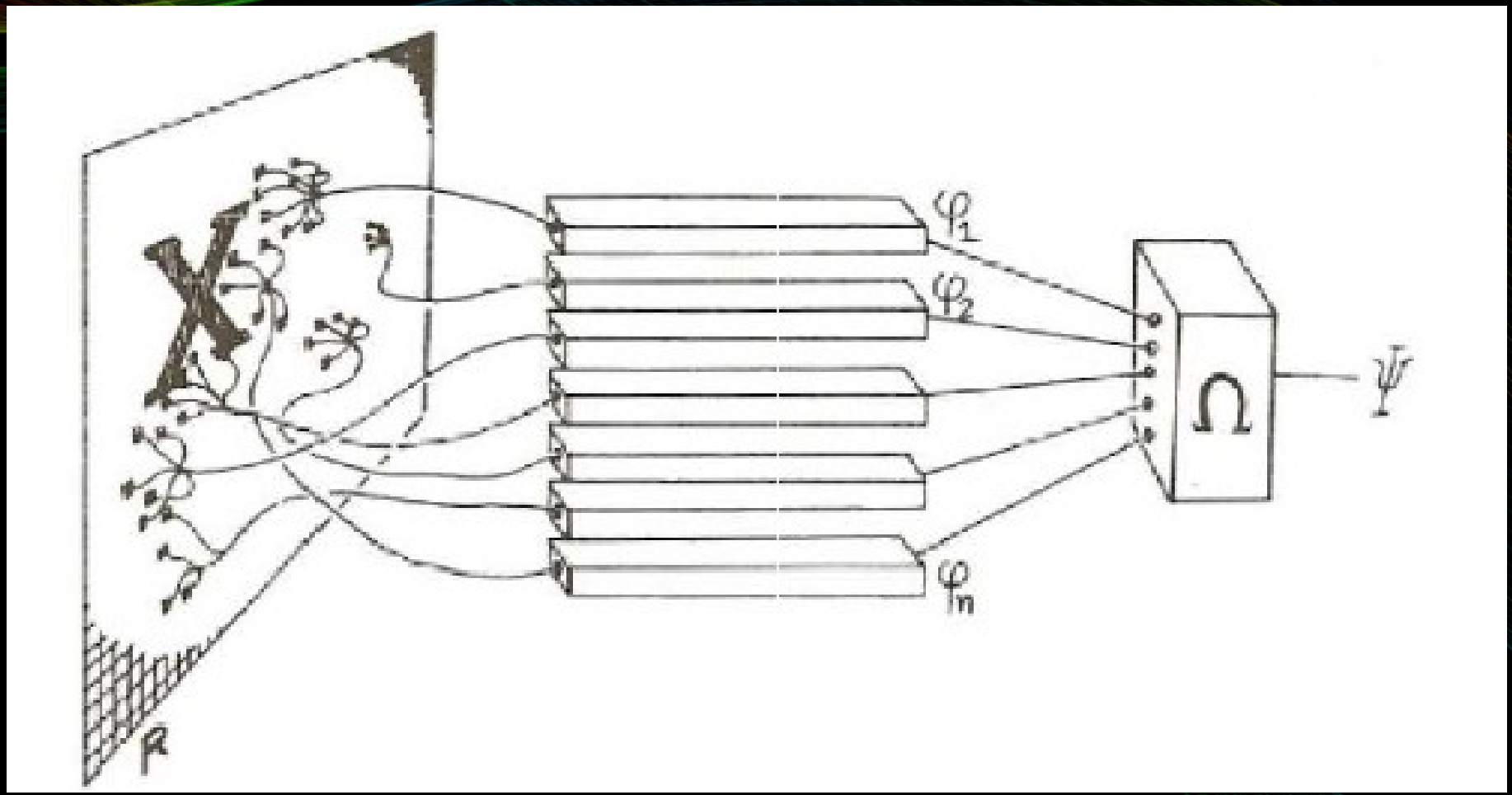
José Luis Seixas Junior

# Índice

- Perceptron.
- Algoritmo.
- Utilização.

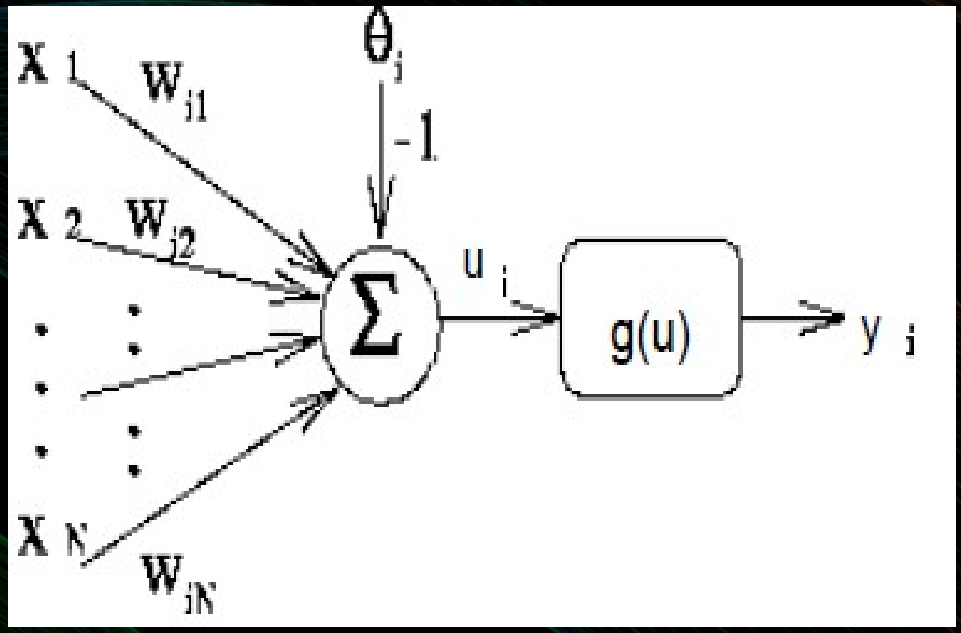
# Perceptron

- A Perceptron (1958), é a forma mais simples de configuração de uma rede neural artificial;



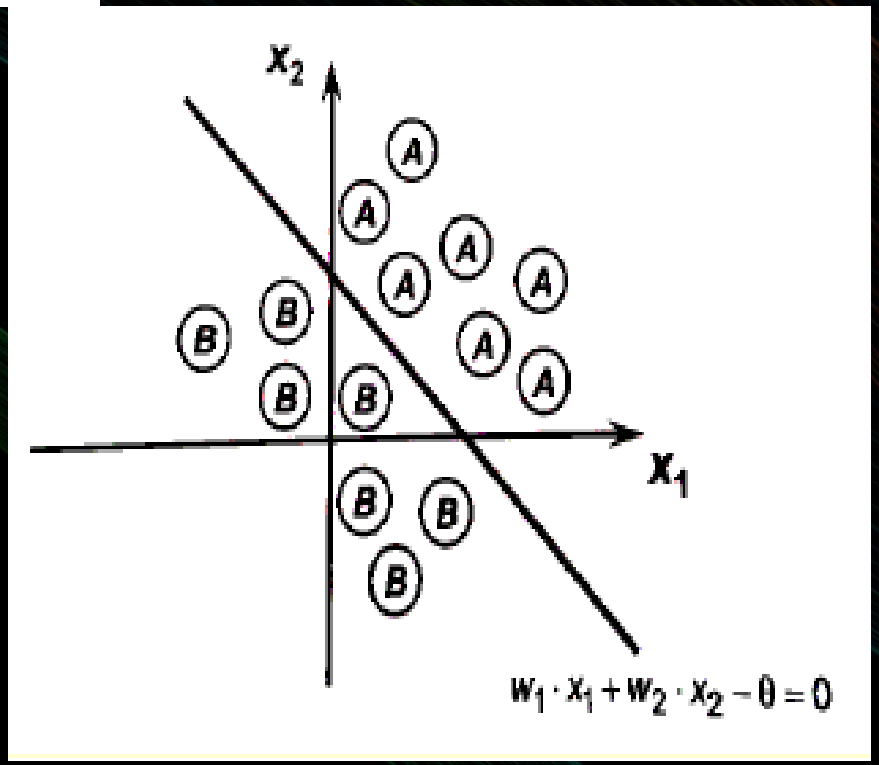
# Perceptron

- Características:
  - X: entradas;
  - W: pesos sinápticos;
  - n: quantidade de características;
  - Theta: limiar de ativação;
  - i: quantidade de neurônios;
  - u: potencial de ativação;
  - g(u): função de ativação;
  - y: saída;



# Perceptron

$$y = \begin{cases} 1, se \sum w_i x_i - \theta \geq 0 \\ -1, se \sum w_i x_i - \theta < 0 \end{cases}$$

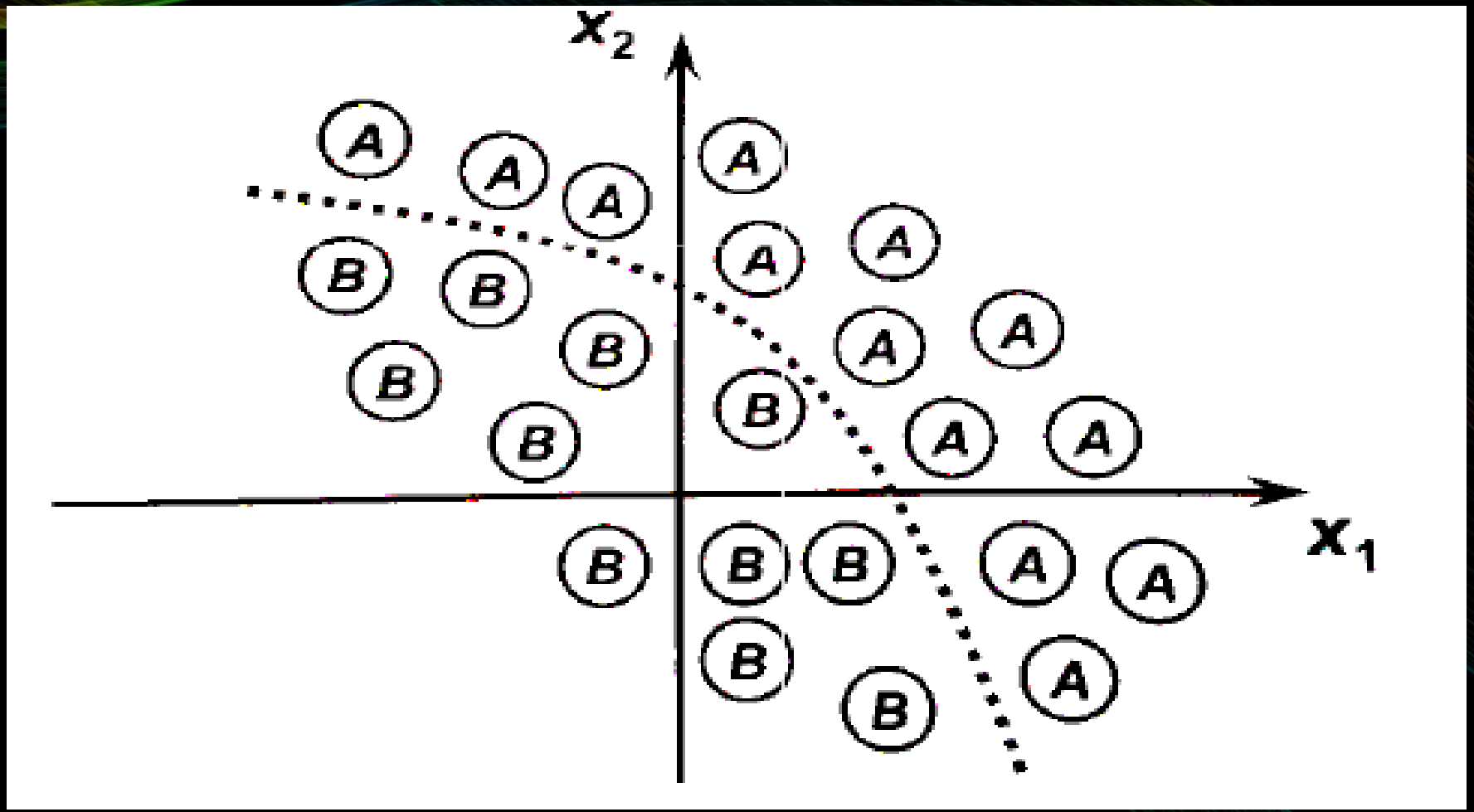


# Perceptron

- Uma Perceptron formada por três dimensões (três entradas), tem uma fronteira de separação por um plano;
- Para mais que três entradas, as fronteiras serão hiperplanos;
- Para a utilização da Perceptron como classificador preciso é necessário que a solução seja linear;
- A Perceptron pode aproximar, mas não solucionar problemas não linearmente separaráveis;

# Perceptron

- A *Single-Layer* Perceptron não soluciona este problema:



# Algoritmo

- Regra de Hebb:
  - Caso a saída esteja em conformidade com a saída esperada os pesos sináticos e limiares são incrementados – ajuste excitatório;
  - Caso a saída não esteja em conformidade, os pesos e limiares serão decrementados;

- Em termos matemáticos:

taxa de aprendizagem

valor desejado

Peso sináptico

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

limiar

# Algoritmo

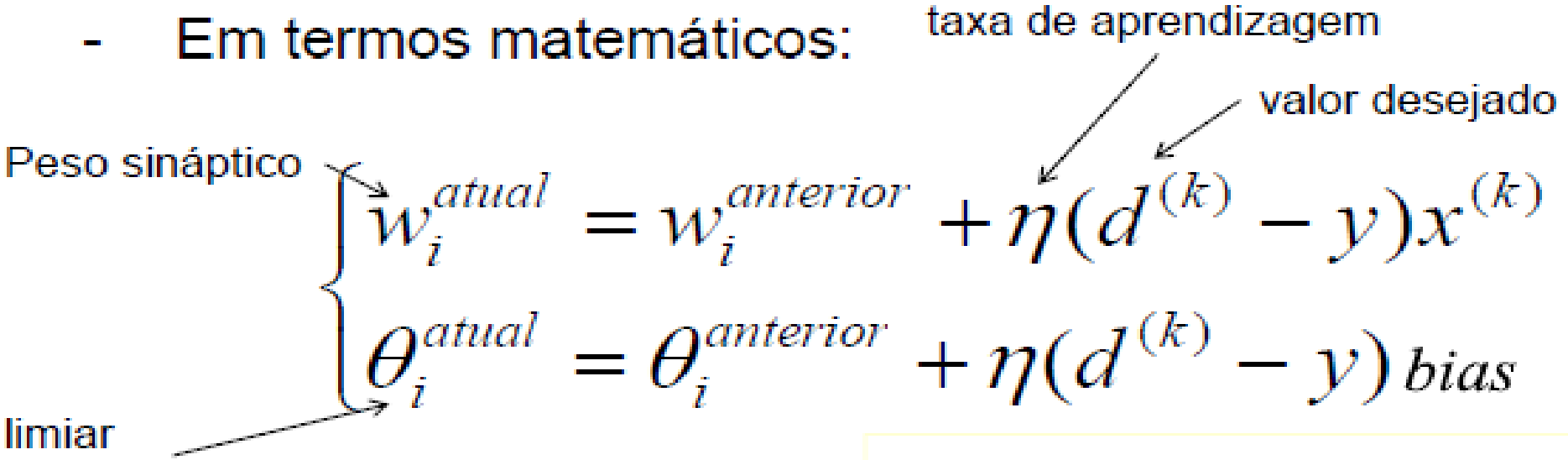
(1) Obter o conjunto de treinamento;
(2) Associar a saída desejada $\{d^k\}$;
(3) Iniciar *w* com valores aleatórios pequenos; [0..1]
(4) Especificar a taxa de aprendizado; n = 0.01
(5) Iniciar o contador de épocas com 0 e erro com 0;
(6) Repetir enquanto
    (1) Erro = inexiste;
    (2) Para todas as amostras de treinamento, fazer:
        (1) Encontrar d;
        (2) Encontrar y;
        (3) Se y ≠ d;
            (1) Então ajusta w e Erro = existe;
    (3) Época = Época + 1;
    (4) Até que erro = inexiste

# Algoritmo

(1) Obter amostra a ser classificada {x}
(2) Utilizar *w* ajustado durante fase de treinamento;
(3) Executar operação
    (1) Obter u;
    (2) Obter y;
    (3) Se y=-1
        (1) Então amostra x $\in$ {Classe A}
    (4) Se y=1
        (1) Então amostra x $\in$ {Classe B}

# Algoritmo

- Caso encontre Erro:
  - Em termos matemáticos:




- Caso não encontre Erro, Fim.

# Utilização

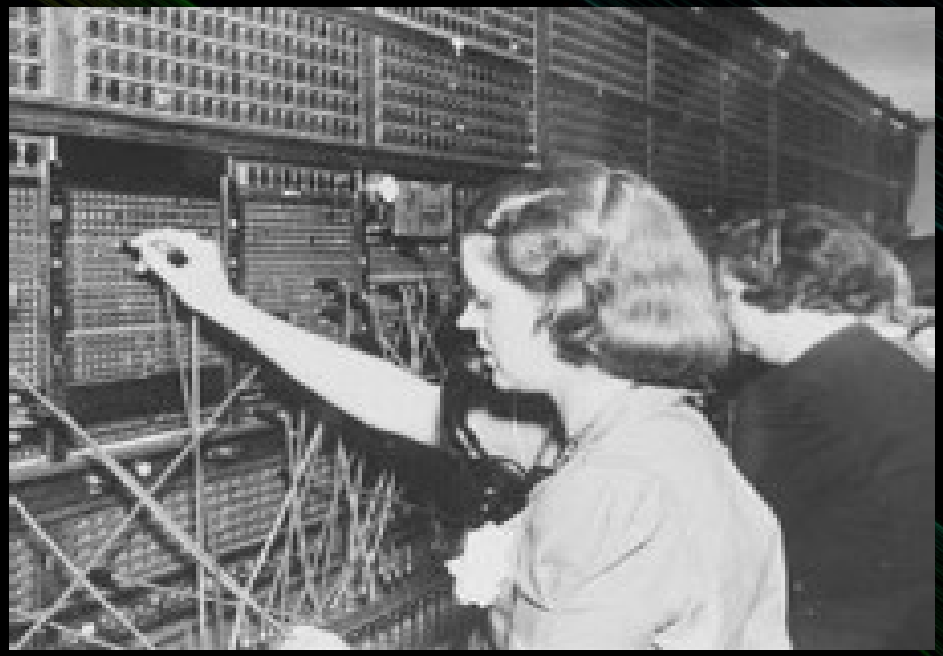
- A rede irá divergir se o problema não for linearmente separável.
- Podemos esperar um valor aproximado de execução limitando o número de épocas.
- Se a separabilidade entre as duas classes for muito estreita, o processo será instável.
  - Deve-se utilizar taxas de aprendizado pequena (n);

# Utilização

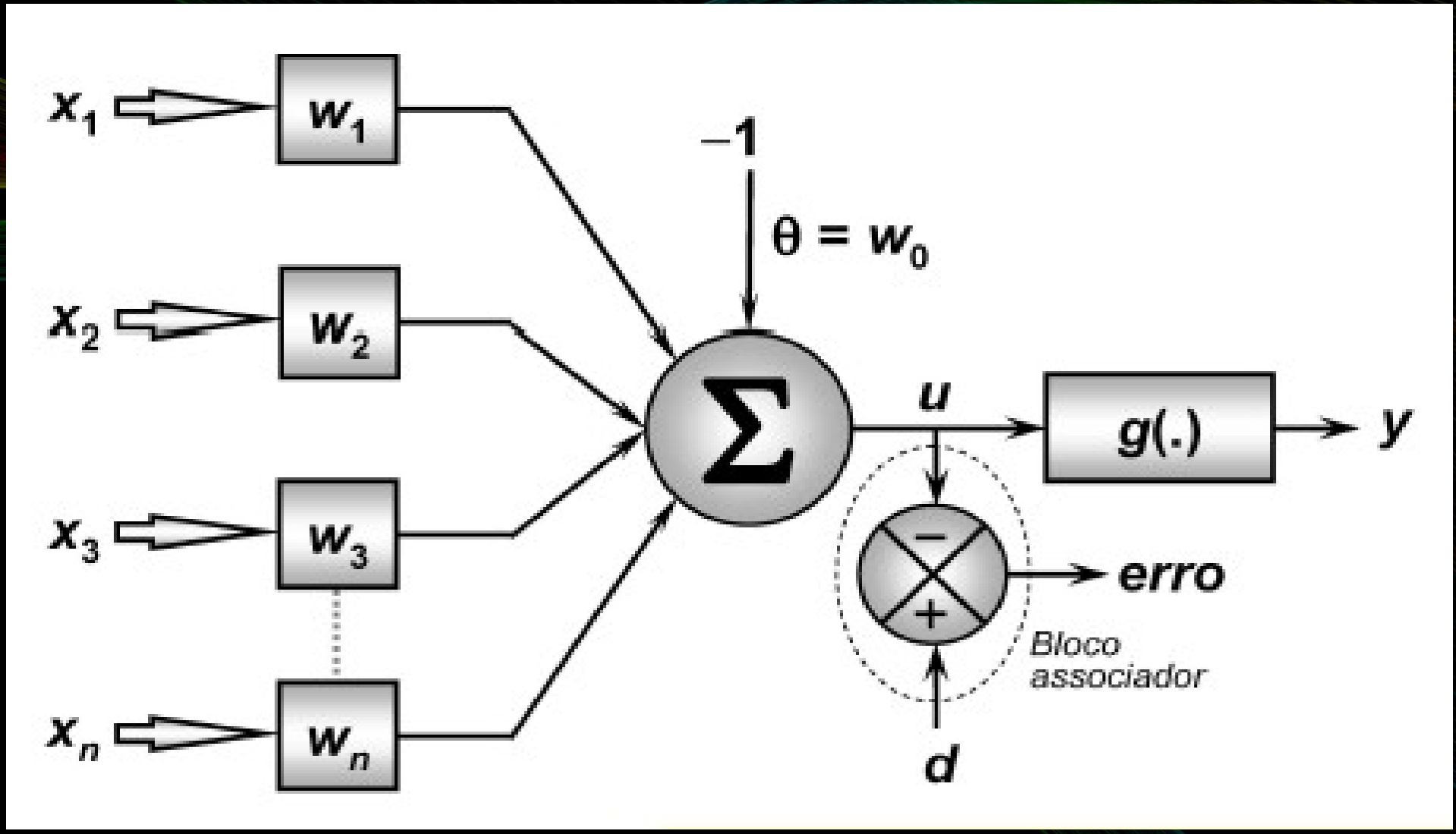
- Boas prática:
  - Normalização das entradas acelera o processo de treinamento;
  - Variar taxas de aprendizado pode inferir em um treinamento mais rápido;
  - Não havendo regra para quantidade de neurônios, varia-los também pode fornecer aumento na velocidade;
- Conhecer o problema pode resultar em melhor escolha de arquitetura.

# Adaline

- Criada em 1960 foi a primeira rede aplicada na indústria.
  - Aplicações em soluções de sinais analógicos;
  - Desenvolvimento da regra Delta;



# Adaline


$x_1$
$x_2$
$x_3$
$x_n$
$w_1$
$w_2$
$w_3$
$w_n$
−1
$\theta = w_0$
$\Sigma$
$u$
$g(.)$
$y$
−
+
erro
Bloco associador
$d$

# Adaline

- O erro (erro = d – u;) será utilizado no ajuste de pesos (Ws);
- A teoria matemática é a mesma da Perceptron, mas a atualização varia com a regra Delta;
  - E não a regra de Hebb;

$$w^{atual} = w^{anterior} + \eta.(d^{(k)} - u).x^{(k)}$$

# Adaline

- Critério de parada:
  - Função do erro quadrático médio:
  - O algoritmo para quando a diferença entre duas épocas for menor que a precisão imposta;

$$E_{qm}(w) = \frac{1}{p} \sum_{k=1}^{p} \left(d^{(k)} - u\right)^2$$

# Adaline

- Se erro < epsilon:
  - Fim;
- Senão:

$$w^{atual} = w^{anterior} + \eta.(d^{(k)} - u).x^{(k)}$$

# Perceptron x Adaline