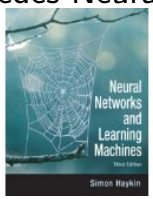
Aula 04 – Redes Neurais Artificiais

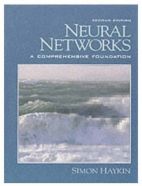
Clodoaldo A. M. Lima

17 de março de 2015

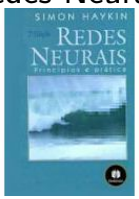
Redes Neurais



Redes Neurais

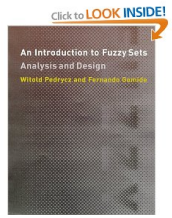


Redes Neurais



Algoritmo Genético

Lógica Fuzzy



Motivação

Neurônio: 2ms | Processador: 2ns

- Processador é 10^6 mais rápido que o neurônio
- Cérebro reage a um estímulo entre 0,2 e 1 seg.

- Constatação que o cérebro processa informações de forma diferente dos computadores convencionais

Cérebro

- velocidade 1 milhão de vezes mais lenta que qualquer gate digital

Computador

- processamento extremamente rápido e preciso na execução de seqüência de instruções

- Processamento altamento paralelo(10^11 neurônios com 10^4 conexões cada)

Motivação

Observações

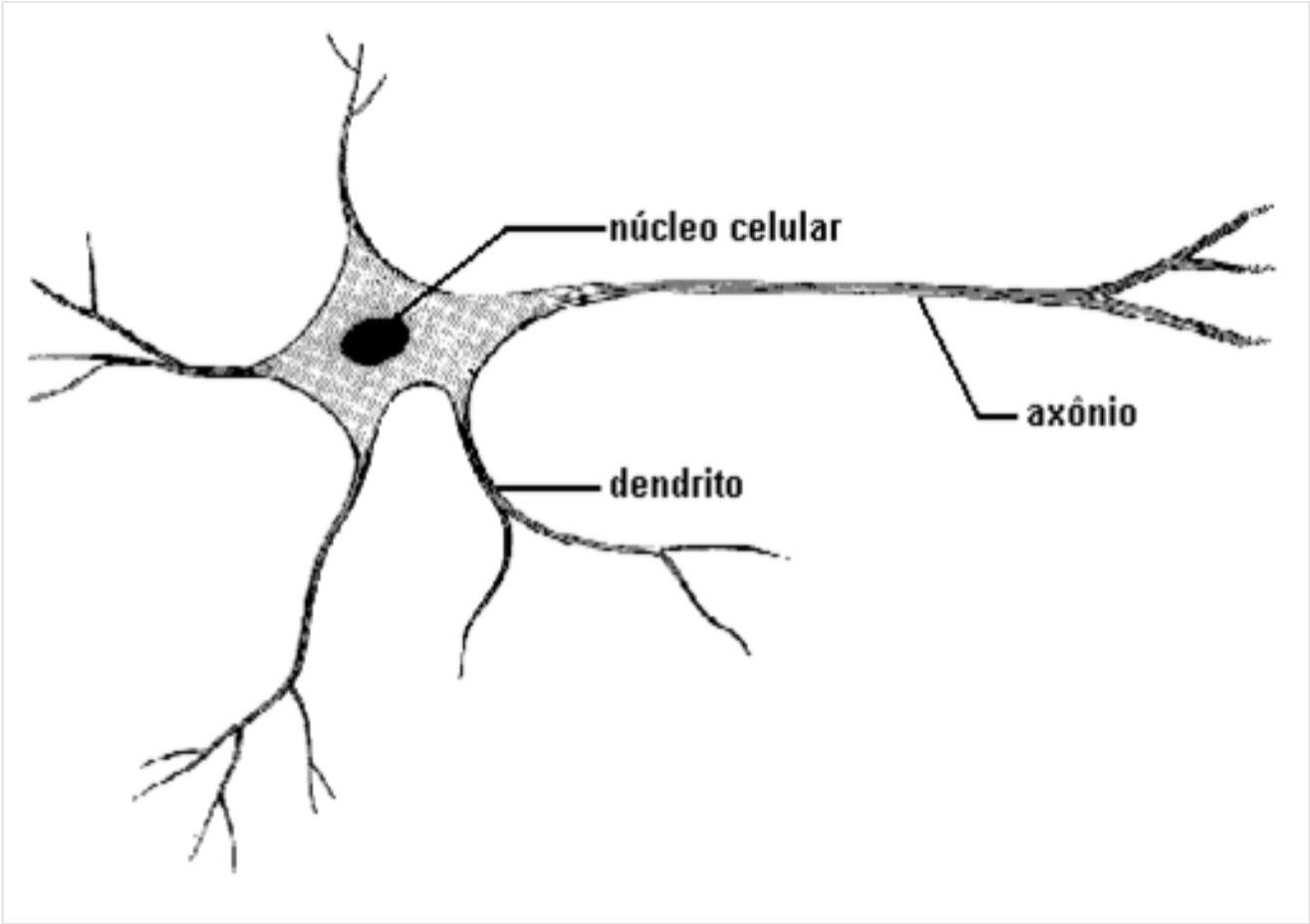
- O cérebro tem $\sim$ 10 bilhões de neurônios.
- Cada neurônio tem $\sim$ 1.000 a 10.000 conexões
- 60 trilhões de conexões – 10^{14} sinapses!
- Cada pessoa pode dedicar 100 conexões para armazenar cada segundo de experiência (65 anos $\Rightarrow$ 2.000.000.000 de segundos!)
- Durante os 2 primeiros anos de vida, 1.000.000 de sinapses são formadas por segundo!!

Modelo biológico

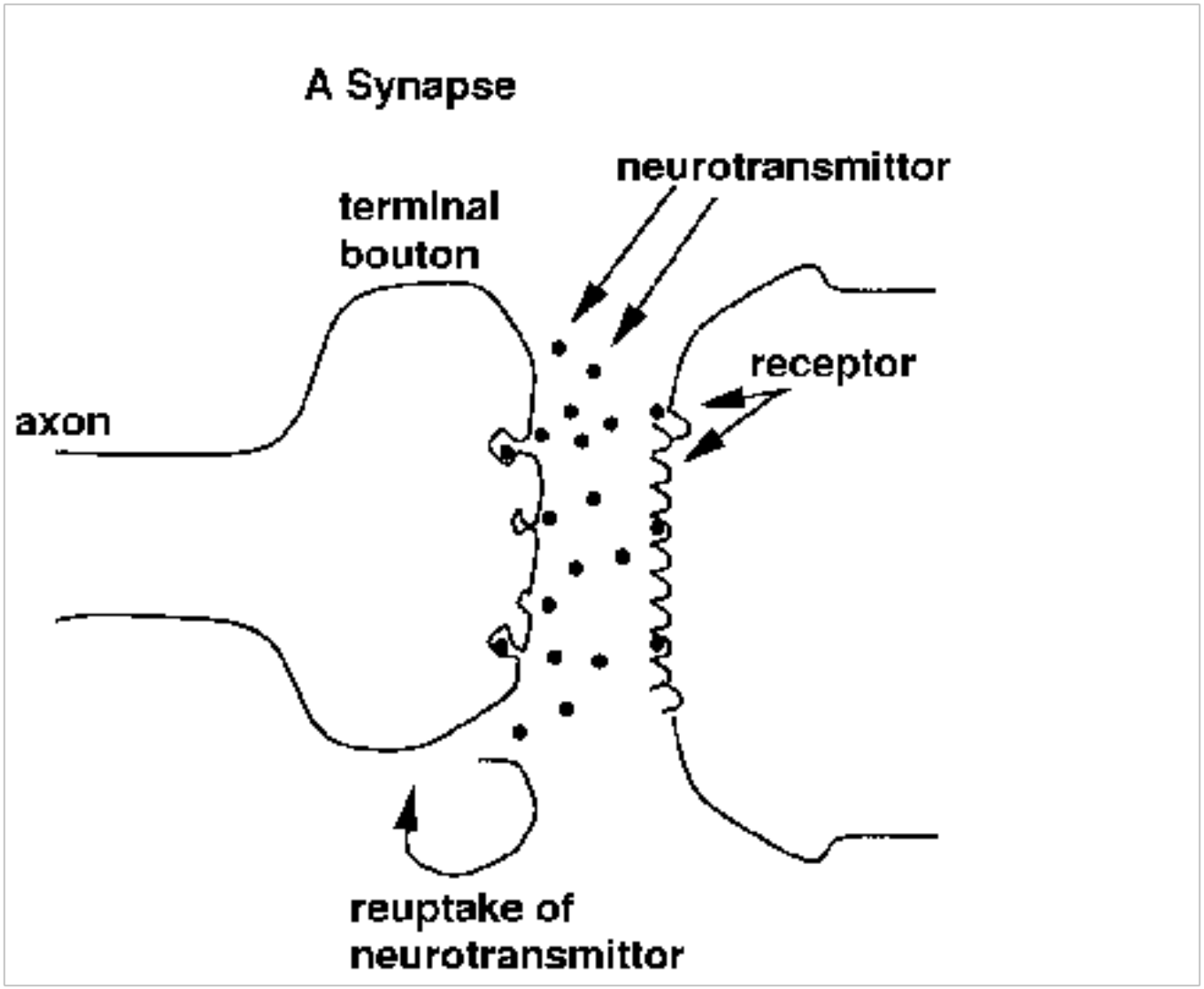
O neurônio é composto por um corpo celular chamado soma, ramificações chamadas dendritos (que recebem as entradas) e um prolongamento denominado axônio que tem como função transmitir o sinal do corpo celular para suas extremidades (é a saída do sinal).

As extremidades do axônio são conectadas com dendritos de outros neurônios pelas sinapses, formando grandes redes.

Modelo biológico



Sinapse


A Synapse
neurotransmittor
terminal
bouton
receptor
axon
reuptake of
neurotransmittor

Breve Histórico

- 1911 – Ramon y Cajal definiram a idéia de neurônio
- 1943 – McCullock e Pitts:
  - primeiro modelo matemático do neurônio
  - combinação de vários neurônios simples possui elevado poder computacional
  - qualquer função matemática ou lógica pode ser implementada
- 1949 – Donald Hebb no livro The Organization of Behavior definiu o conceito de atualização de pesos sinápticos
- 1958 – Implementação do primeiro modelo de neurônio artificial: o perceptron, por Franck Rosemblat

Breve Histórico

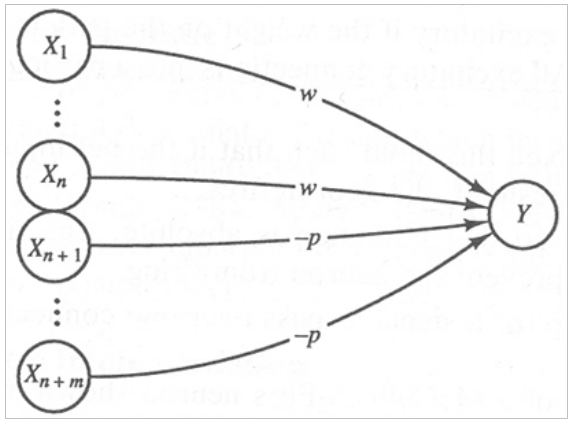
- 1969 – No livro Perceptrons: an Introduction to Computational Geometry, M. Minsky e S. Papert mostram que com um perceptron de uma camada não é possível representar problemas não linearmente separáveis, como o operador XOR.
- 1970 a 1980 – buraco negro
- 1980 a ...- desenvolvimento de novas arquiteturas de redes neurais e de novos algoritmos de aprendizagem.
- É o "renascimento" das redes neurais

Como funciona um neurônio artificial ?

O neurônio de McCulloc-Pitts:

- A ativação do neurônio McCulloc-Pitts é binária. Isto é, em um determinado momento, ou o neurônio dispara (tem a ativação $= 1$) ou o neurônio não dispara (tem a ativação $= 0$).
- Esses neurônios são conectados por caminhos direcionados e ponderados.
- O caminho de conexão é excitatório se o peso no caminho é positivo; caso contrário é inibitório. Todas as conexões excitatórias chegando em um neurônio tem os mesmos pesos.
- Cada neurônio tem um limiar fixo (threshold) tal que se a entrada do neurônio é maior que o tal limiar, então o neurônio dispara.
- O limiar é determinado tal que uma inibição é absoluta. Isto é, qualquer entrada inibitória não-zero impedirá o neurônio de disparar.
- É necessário um "passo de tempo" para um sinal passar por um link de conexão.

O neurônio de McCulloc-Pitts



- Temos que determinar o limiar de disparo do neurônio Y e então determinar os pesos das conexões, com o intuito de resolver um problema.
- Um neurônio McCulloc-Pitts Y deve receber sinais de qualquer número de outros neurônios.

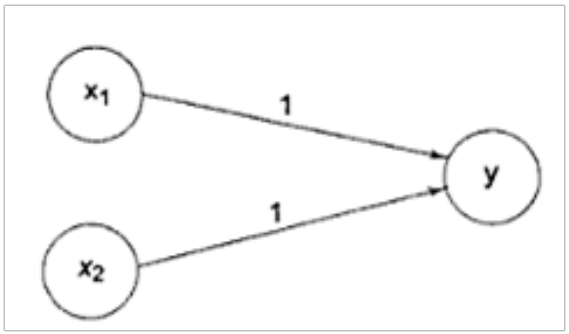
$$y_{in} = \sum_i x_i w_{ij}$$

O neurônio de McCulloc-Pitts

- Cada link de conexão é ou excitatório (w, com $w > 0$) ou inibitório ($-p$, com $p > 0$).
- Na figura temos n unidades $X1, \cdots, X_n$, as quais enviam sinais excitatórios para a Unidade Y, e m unidades, $Xn + 1, \cdots, X_{n+m}$, as quais enviam sinais inibitórios.
- A função de ativação para a unidade Y é: $f(y_i n) = \begin{cases} 1 & if \quad y_{in} \geq \theta \\ 0 & if \quad y_{in} < \theta \end{cases}$ onde y_{in} é o sinal total recebido e θ é o limiar.
- Y dispara se recebe k ou mais entradas excitatórias e nenhuma inibitória, onde $kw \geq \theta > (k-1)w$.
- Embora todos os pesos excitatórios que entram em uma unidade (Y_1) devam ser os mesmos, os pesos que entram em um outra unidade (Y_2) não precisam ter o mesmo valor dos que estão entrando na primeira unidade (Y_1).

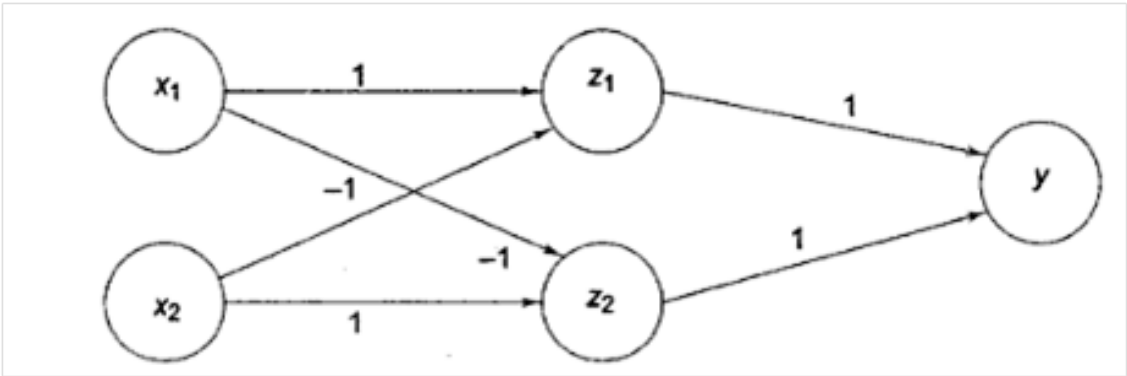
Exercícios sobre Neurônio McCulloch-Pitts

- Gere a saída da função lógica AND pelo modelo neurônio McCulloch Pits
- O neurônio de McCulloch-Pitts para implementar a função AND é mostrada na figura 1. O limiar sobre a unidade Y é 2



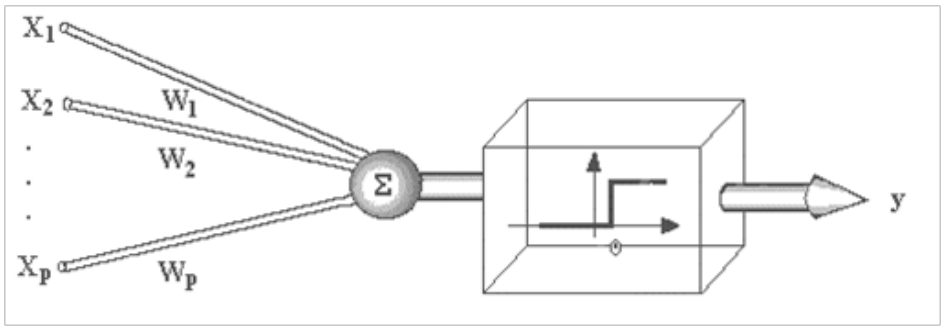
Exercícios sobre Neurônio McCulloch-Pitts

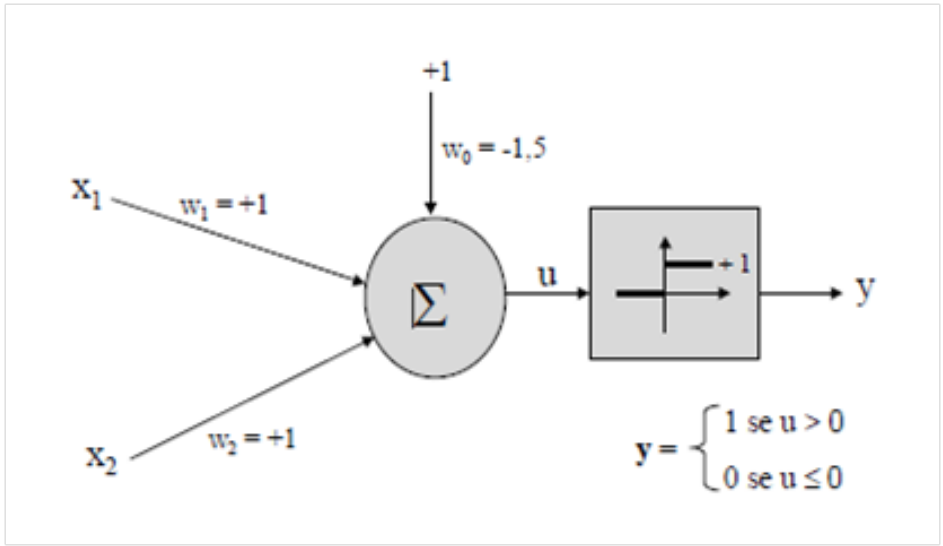
- Realize a função XOR usando neuronio McCulloch
- O modelo neuronio McCulloch-Pitts para a figura abaixo. O limiar da unidade Y é 1
- Interprete o resultado



Perceptron

- Primeiro neurônio artificial
- Modela um neurônio biológico realizando a soma ponderada de suas entradas e enviando o resultado 1 se a soma for maior que um valor inicial ajustável. Caso contrário o resultado é zero





Paradigmas de Aprendizagem

- A capacidade de "aprender" associada a uma rede neural é uma das mais importantes qualidades destas estruturas
- Trata-se da habilidade de adaptar-se, de acordo com regras pré-existentes, ao seu ambiente, alterando seu desempenho ao longo do tempo.

Sendo assim, considera-se "aprendizado" o processo que adapta o comportamento e conduz a uma melhoria de desempenho.

- No contexto de redes neurais artificiais, aprendizagem ou treinamento corresponde ao processo de ajuste dos parâmetros livres da rede através de um mecanismo de apresentação de estímulos ambientais, conhecidos como padrões (ou dados) de entrada ou de treinamento.

estímulo $\rightarrow$ adaptação $\rightarrow$ novo comportamento da rede

Paradigmas de Aprendizagem

- Nas RNAs mais simples e tradicionais, os parâmetros livres da rede correspondem apenas aos pesos sinápticos. Toda a estrutura da rede, incluindo os tipos de neurônios e suas funções de ativação, é pré-definida.
- O objetivo do aprendizado em redes neurais é a obtenção de um modelo implícito do sistema em estudo, por ajuste dos parâmetros da rede.
- Dada uma rede neural artificial, seja $w(t)$ um peso sináptico de um dado neurônio, no instante de tempo t. O ajuste $\Delta w(t)$ é aplicado ao peso sináptico $w(t)$ no instante t, gerando o valor corrigido $w(t+1)$, na forma:

Paradigmas de Aprendizagem

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

A obtenção de $\Delta w(t)$ pode ser feita de diversas formas. O tipo de aprendizado é determinado pela técnica empregada no processo de ajuste dos pesos sinápticos (parâmetros da rede neural).

- Um conjunto bem definido de regras para obtê-los é denominado um algoritmo de aprendizagem ou treinamento. Exemplos de alguns algoritmos: regra de Hebb, algoritmo de backpropagation, estratégias de competição, máquina de Boltzmann..
- A maneira pela qual o ambiente influencia a rede em seu aprendizado define o paradigma de aprendizagem. Exemplos de paradigmas: aprendizado supervisionado, aprendizado por reforço e aprendizado não-supervisionado (ou auto-organizado).

Paradigmas de Aprendizagem

- Seja qual for o algoritmo ou o paradigma utilizado, ao alcançarmos o objetivo obtemos uma representação de conhecimento que obedece a uma ou mais das quatro regras de bom senso descritas abaixo:
  - R1 entradas similares provenientes de classes similares de fenômenos ou eventos tendem a produzir representações similares dentro da rede, o que pode levar a classificá-las como pertencentes à mesma categoria
  - R2 itens que devem ser classificados ou processados distintamente devem provocar, de alguma forma, representações distintas dentro da rede.
  - R3 se uma característica é importante, então devem ser alocados recursos da rede neural (por exemplo, neurônios e conexões) para representá-la devidamente. Quanto mais complexa a representação, mais recursos devem ser alocados.
  - R4 a etapa de aprendizado pode ser simplificada caso as informações conhecidas a priori e invariâncias sejam embutidas diretamente no projeto da rede neural.

Paradigmas de Aprendizagem

As duas primeiras regras utilizam os conceitos de similaridade e/ou de distância. Estes conceitos podem ser expressos matematicamente a partir da definição formal de um critério de medida. Em particular, a Teoria de Medidas é uma das áreas mais bem formalizadas da Matemática, juntamente com a Teoria de Conjuntos, constituindo a base de todos os métodos matemáticos.

Regra de Aprendizado

- Uma rede neural aprende a respeito seu ambiente através um processo iterativo de ajuste aplicado seus pesos sinápticos e nível de bias.
- Aprendizado é o processo pelo qual os parâmetros livres de uma rede neural será adaptada através um processo de estimulação pela ambiente na qual a rede esta embutida.
- O tipo de aprendizado é determinado pela maneira na qual os parâmetros alterados toma lugar.

Paradigmas de Aprendizagem

Existem basicamente três paradigmas de aprendizado:

- Aprendizado supervisionado: é baseado em um conjunto de exemplos de estímulo resposta (ou entrada-saída), ou em algum outro tipo de informação, que represente o comportamento que deve ser apresentado pela rede neural;
- Aprendizado por reforço: o comportamento da rede é avaliado apenas com base em alguma critério numérico, fornecido em instantes espaçados de tempo;
- Aprendizado não-supervisionado: é baseado apenas nos estímulos recebidos pela rede neural. Basicamente, a rede deve aprender a "categorizar" os estímulos.

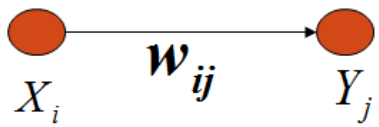
Este curso vai se ocupar com o desenvolvimento de técnicas para aprendizado supervisionado e não-supervisionado.

Regra de Hebb

- Após a publicação do trabalho de McCulloch & Pitts em 1943, Norbert Wiener publicou um livro famoso, em 1948, denominado Cybernetics, seguido pela publicação do livro The Organization of Behavior por Hebb.
- No livro de Hebb, foi proposta pela primeira vez uma regra de aprendizagem através da modulação (ou modificação) de pesos sinápticos
- Basicamente, Hebb propôs que a efetividade de uma sinapse aumenta devido a ativação repetida de um neurônio (por outro neurônio). Com suas próprias palavras:

"Quando o axônio de uma célula A está próximo o suficiente de excitar uma célula B ou persistentemente contribui para sua ativação, algum processo de crescimento ou variação metabólica ocorre em uma ou ambas as células, tal que a efetividade da célula A em ativar a célula B é aumentada."

Este postulado requer uma mudança no peso sináptico entre células quando as células pré- e pós-sinápticas estão ativas simultaneamente.



Regra de Hebb

- Hebb sugeriu que esta mudança era a base para a aprendizagem associativa, resultando em uma modificação duradoura no padrão de atividade de uma rede neural.

$$\Delta w_{ij}(t) = \alpha y_i(t) x_i(t)$$

- onde $\Delta w_{ij}(t)$ é a mudança a ser aplicada no neurônio i, α é um fator multiplicativo denominado de taxa de aprendizagem, y_i é a saída do neurônio i, x_j é a entrada do neurônio j, e t é o índice de tempo.
- Note que esta equação deixa clara a natureza correlacional ou associativa da regra de atualização de Hebb.
  - Sabe-se que boa parte da memória humana é associativa. Neste tipo de memória, um evento está ligado a outro evento, de forma que a ocorrência do primeiro evento resulta na ocorrência do evento ligado.
  - Em sua versão mais simples, um estímulo está ligado a uma resposta.

Regra de Hebb

- A diferença principal entre a proposta original e a regra generalizada é o fato de que no caso generalizado tanto os estímulos excitatórios quanto os inibitórios influenciam na atividade do neurônio.
- A equação acima pode ser expressa de forma genérica como sendo:

$$\Delta w_{ij}(t) = \alpha g(y_i(t), x_i(t))$$

- onde $g(.,.)$ é uma função de ambos os sinais, pré- e pós-sináptico.
- Portanto, o peso de um neurônio i é atualizado de acordo com a seguinte regra:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

Regra Aprendizado Delta

- A regra de aprendizado delta é também referida como regra Wildron-Hoff, chamada devido para os originadores (Widrow & Hoff, 1960).
- Seja $\{x_1, d_1\}, \{x_2, d_2\}, \cdots, \{x_N, d_N\}$, um conjunto de pares de entrada-saída, onde x_j é o vetor de entradas j, e d_j seu correspondente vetor de saídas (desejadas).
- A regra delta ajusta os pesos e o limiar da rede neural de forma a minimizar o erro (diferença) entre a saída da rede e a saída desejada para todos os padrões de treinamento
- A soma dos erros quadráticos (SSE) para um determinado padrão é dada por:.

$$J(W) = \sum_i (y_i - d_i)^2$$

- O gradiente de J, também denominado de índice de desempenho ou função custo, é o vetor que consiste das derivadas parciais de J em relação a cada um dos pesos.

Regra Aprendizado Delta

- Seja $y_i = f(\sum_j w_{ij} x_j)$ e $J(W) = \sum_i (y_i - d_i)^2$
- Como o peso w_{ij} influencia apenas a unidade i, o gradiente do erro é dado por:

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_i (y_i - d_i)^2$$

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} (y_i - d_i)^2$$

$$\frac{\partial J}{\partial w_{ij}} = 2(y_i - d_i) \frac{\partial y_i}{\partial w_{ij}}$$

- se mapeamento linear $y_i = \sum_j w_{ij} x_j \rightarrow \frac{\partial J}{\partial w_{ij}} = 2(y_i - d_i) x_j$
- se mapeamento não linear $y_i = f(\sum_j w_{ij} x_j) \rightarrow \frac{\partial J}{\partial w_{ij}} = 2(y_i - d_i) f'(\sum_j w_{ij} x_j) x_j$

Regra Aprendizado Delta

- Portanto a regra delta para atualizar o peso do neurônio w_{ij} é dada por:

$$w_{ij} = w_{ij} - \alpha(y_i - d_i)x_j$$

$$b_i = b_i - \alpha(y_i - d_i)$$

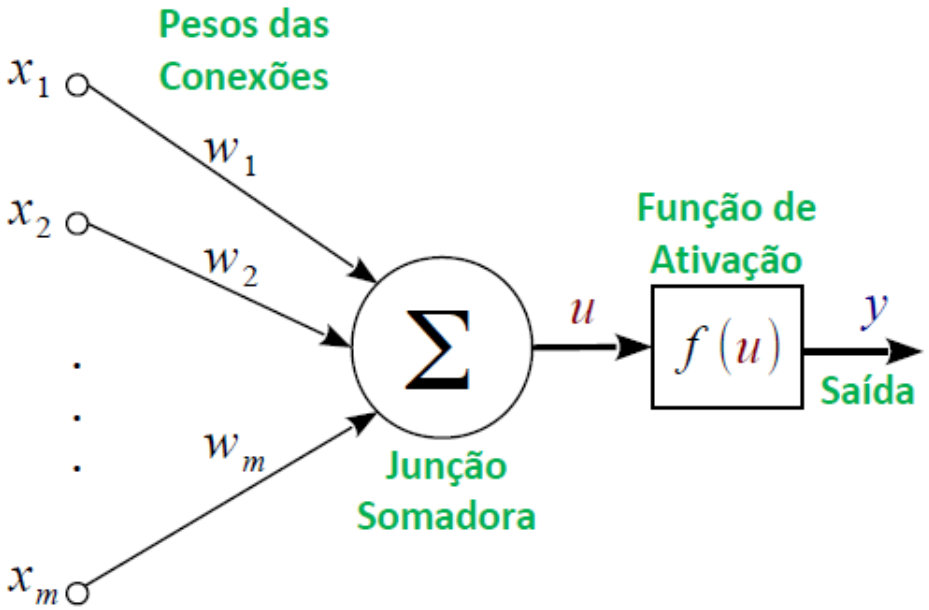
- Note que o parâmentro α incorpora a constante 2 do gradiente
- Notação Matricial

$$\mathbf{W} \leftarrow \mathbf{W} + \alpha\, \mathbf{e}_i\, \mathbf{x}_i^T$$

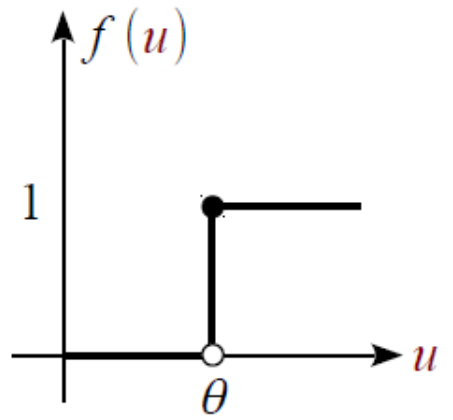
$$\mathbf{b} \leftarrow \mathbf{b} + \alpha\, \mathbf{e}_i$$

onde $\mathbf{W} \in \Re^{o \times m}$, $\mathbf{x}_i \in \Re^{m \times 1}$, $i = 1,\ldots,N$, $\mathbf{e}_i \in \Re^{o \times 1}$, e $\mathbf{b} \in \Re^{o \times 1}$.

Perceptron Simples



Arquitetura do Perceptron.



$$\begin{cases} u \rightarrow \textit{Saída Linear} \\ y \rightarrow \textit{Saída de Ativação} \end{cases}$$

$$y = f(u) = f\left(\sum_{j=1}^{m} w_j x_j\right) = \begin{cases} 1, & u \geq \theta \\ 0, & u < \theta \end{cases} \quad \rightarrow \quad \begin{cases} \mathbf{W} = \begin{bmatrix} w_1 & w_2 & \cdots & w_m \end{bmatrix}^T \\ \mathbf{X} = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix}^T \end{cases}$$

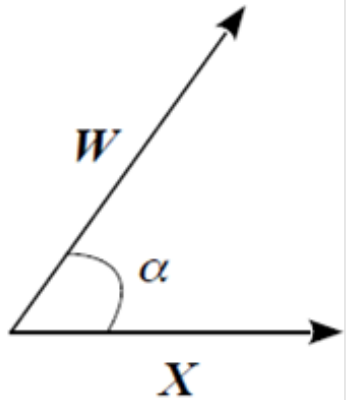
Perceptron Simples

$$y = f(u) = f\left(\sum_{j=0}^{m} w_j x_j\right) = f(\mathbf{W}\cdot\mathbf{X}) = f(\|\mathbf{W}\|\|\mathbf{X}\|\cos(\alpha)) = \begin{cases} 1, & u \geq 0 \\ 0, & u < 0 \end{cases}$$

$$\mathbf{W}\cdot\mathbf{X} = \|\mathbf{W}\|\|\mathbf{X}\|\cos(\alpha) \rightarrow \begin{cases} \|\mathbf{W}\| \geq 0 \\ \|\mathbf{X}\| \geq 0 \end{cases}$$

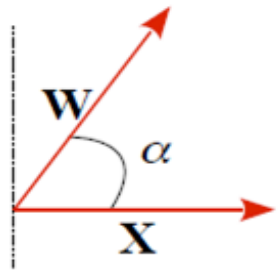
(α *é o menor ângulo entre* $\mathbf{W}$ *e* $\mathbf{X}$: $0° \leq \alpha \leq 180°$)

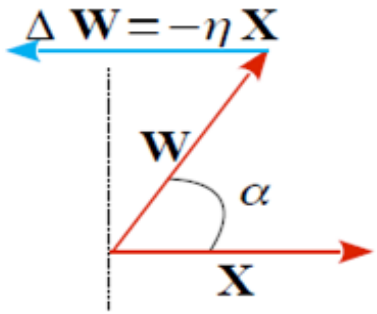
$$\begin{cases} u = 0 \text{ e } y = 1 \text{ se } \cos(\alpha) = 0 \rightarrow \alpha = 90° \\ u > 0 \text{ e } y = 1 \text{ se } \cos(\alpha) > 0 \rightarrow \alpha < 90° \\ u < 0 \text{ e } y = 0 \text{ se } \cos(\alpha) < 0 \rightarrow \alpha > 90° \end{cases}$$

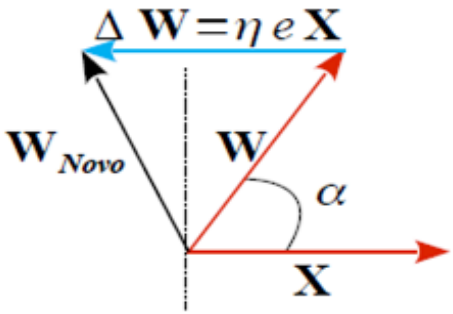


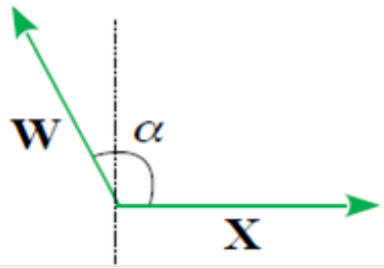
Perceptron Simples

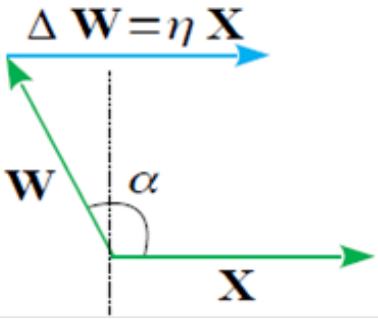
d	y	e	$\Delta \mathbf{W}$
0	0	0	0
0	1	-1	?
1	0	1	?
1	1	0	0

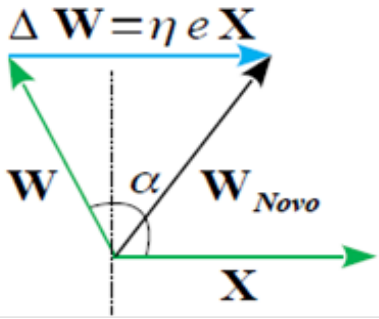












Perceptron Simples

- Rosenblatt introduziu o perceptron como a arquitetura mais simples de rede neural capaz de classificar padrões linearmente separáveis
- O algoritmo de treinamento do perceptron foi o primeiro modelo de treinamento supervisionado, embora alguns perceptrons fossem auto-organizados.
- Basicamente, o perceptron consiste em uma única camada de neurônios com pesos sinápticos e bias ajustáveis.
- Se os padrões de entrada forem linearmente separáveis, o algoritmo de treinamento do perceptron possui convergência garantida, ou seja, é capaz de encontrar um conjunto de pesos que classifica corretamente os dados
- Os pesos dos neurônios que compõem o perceptron serão tais que as superfícies de decisão produzidas pela rede neural estarão apropriadamente posicionadas no espaço.
- Os neurônios do perceptron são similares ao neurônio de McCulloch & Pitts (função de ativação tipo degrau), mas possuem pesos associados, incluindo o bias.

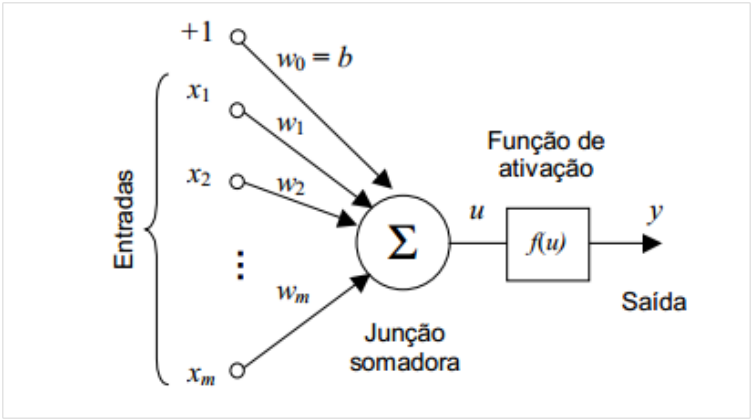
Perceptron Simples para Classificação de Padrões

- O algoritmo do perceptron funciona como a seguir.
- Para cada padrão de treinamento (dado de entrada) x_i, a saída da rede y_i é calculada.
- Em seguida, é determinado o erro ei entre a saída desejada para este padrão d_i e a saída da rede y_i , $e_i = d_i - y_i$.
- O vetor de pesos conectando as entradas (neurônios pré-sinápticos) a cada saída (neurônios pós-sinápticos) e o bias do neurônio são atualizados de acordo com as seguintes regras:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha\, e_i\, \mathbf{x}_i,$$

$$b(t+1) = b(t) + \alpha\, e_i,$$

onde $\mathbf{w} \in \Re^{1\times m}$, $\mathbf{x} \in \Re^{1\times m}$, e $b \in \Re^{1\times 1}$.



Perceptron Simples para Classificação de Padrões

- O objetivo desta rede, mais especificamente deste neurônio, é classificar alguns padrões de entrada como pertencentes ou não pertencentes a uma dada classe.
- Considere o conjunto de dados de entrada como sendo formado por N amostras $\{x_1, d_1\}, \{x_2, d_2\}, \cdots, \{x_N, d_N\}$, onde x_j é o vetor j de entradas, e d_j sua saída desejada (classe) correspondente.
- Seja $X \in \mathcal{R}^{N \times m}$, a matriz de dados de entradas com N padrões de dimensão m cada (colunas de X), e $d \in \mathcal{R}^{N \times 1}$ o vetor de saídas desejadas.

Perceptron Simples para Classificação de Padrões

- O algoritmo abaixo pode ser utilizado para treinar o perceptron de um único neurônio:

```
procedure [w] = perceptron(max_it, E, α, X, d)
  initialize w      // por simplicidade, inicialize com 0
  initialize b      //por simplicidade, inicialize com 0
  t ← 1
  while t < max_it & E > 0 do,
    for i from 1 to N do,  //para cada padrão de entrada
      y_i ← f(w x_i + b)          //determine a saída para x_i
      e_i ← d_i − y_i             //determine o erro para x_i
      w ← w + α e_i x_i           //atualize o vetor de pesos
      b ← b + α e_i               //atualize o bias
    end for
    E ← sum(e_i)
    t ← t + 1
  end while
end procedure
```

Adaline

Adaline = ***Ada**ptive **Li**near **Ne**uron* ou ***Ada**ptive **Line**-**ar** **E**lement*.

Surgiu na literatura **quase que simultaneamente** com o **Perceptron** ao final da década de 50.

Assim como o **Perceptron**, é um modelo baseado em **elementos** que **executam operações** sobre a **soma ponderada** de suas **entradas**.

- Operações **não-lineares**, no caso do **Perceptron**, e **puramente lineares**, no cado do **Adaline**.

Adaline

Apesar das **semelhanças**, os **trabalhos** que descreveram o **Perceptron** e o Adaline surgiram em **áreas diferentes** e com **enfoques diferentes**:

- Frank Rosenblatt, que era **psicólogo**, **enfocou** a descrição do **Perceptron** em **aspectos cognitivos do armazenamento da informação** e da **organização cerebral**, enquanto Bernard Widrow e Marcian Hoff enfocaram a descrição do **Adaline** na **construção de filtros lineares**.

Perceptron = Separador Linear

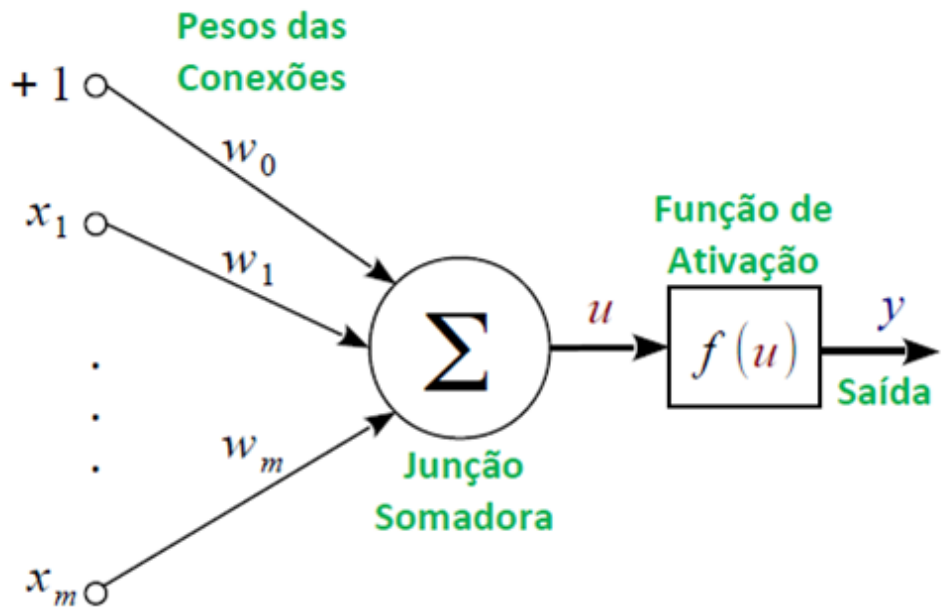
Adaline = Aproximador Linear de Funções

Adaline

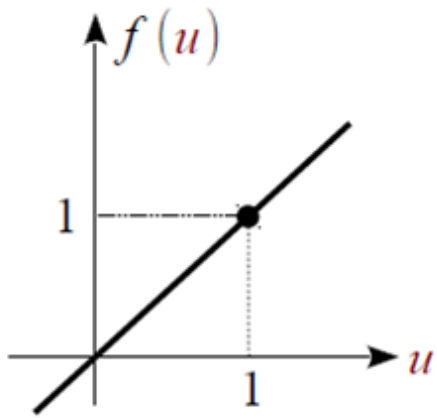
O algoritmo de **treinamento** do **Adaline** utiliza a **informação** contida no **gradiente do erro** para obter **calcular** o **ajuste** $\Delta \mathbf{W}$ a ser aplicado ao **vetor de pesos**.

Esse algoritmo, conhecido como **Regra Delta**, **deu origem**, anos mais tarde, ao **primeiro algoritmo** de **treinamento** de redes **Perceptron** de **múltiplas camadas**, o ***Backpropagation***.

Adaline



Arquitetura do Adaline.



$$\begin{cases} u \rightarrow \textit{Saída Linear} \\ y \rightarrow \textit{Saída de Ativação} \end{cases}$$

$$y = f(u) = f\left(\sum_{j=0}^{m} w_j x_j\right) = u$$

Adaline

Análise Matemática do Adaline

$$y = f(u) = f\left(\sum_{j=0}^{m} w_j x_j\right) = f(\mathbf{W}\cdot\mathbf{X}) = u = \underbrace{w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m}$$

Combinação Linear das Entradas

Adaline com Uma Entrada $\longrightarrow$ $y = w_0 + w_1 x_1$

Reta!

Função Quadrática de Erro

$$J(\mathbf{W}) = \frac{1}{2}\sum_{i=1}^{N}\left(d_i - y_i\right)^2 = \frac{1}{2}\sum_{i=1}^{N}\left(d_i - \left(\mathbf{W}\cdot\mathbf{X_i}\right)\right)^2 = \frac{1}{2}\sum_{i=1}^{N}\left(d_i - \left(\mathbf{W}^T\mathbf{X_i}\right)\right)^2 = \frac{1}{2}\sum_{i=1}^{N}\left(e_i\right)^2$$

- O **objetivo** do treinamento será **minimizar a função de custo** $J(\mathbf{W})$: