Introdução às Redes Neurais Artificiais

Prof. Matheus Giovanni Pires
mgpires@ecomp.uefs.br
Laboratório de Sistemas Inteligentes e Cognitivos – LASIC
http://sites.ecomp.uefs.br/lasic

erbase 2014
XIV ESCOLA REGIONAL DE COMPUTAÇÃO
BAHIA - ALAGOAS - SERGIPE

O que é inteligência?

O que é inteligência?

- Alguém inteligente:
  - Aprende por experiência
  - Usa conhecimento adquirido por experiência
  - Soluciona problemas na ausência de alguma informação
  - Reage rapidamente perante uma nova situação
  - Determina o que é importante
  - Raciocina e pensa
  - Entende imagens visuais
  - Processa e manipula símbolos
  - É criativo e imaginativo
  - Usa heurísticas

Inteligência vs. Aprendizado

- Aprendizado é a chave da superioridade da Inteligência Humana
  - Aprendizado é a essência da Inteligência
- Para que uma máquina tenha comportamento inteligente, deve-se aumentar sua capacidade de aprendizado
- O ser humano está *pré-programado* para o aprendizado
  - Paradigmas e técnicas de aprendizado de máquina possuem um alvo bem mais limitado do que o aprendizado humano.

Abordagens da Inteligência Artificial

- Abordagem Simbólica
  - Segundo a *IA simbólica* é preciso:
    - Identificar o ***conhecimento*** do domínio (modelo do problema)
    - Representá-lo utilizando uma ***linguagem*** formal de representação
    - Implementar um mecanismo de ***inferência*** para utilizar esse conhecimento
- Abordagem Não-Simbólica
  - Na abordagem Não-Simbólica, o conhecimento não é representado explicitamente por meio de símbolos, e sim, construído a partir de um processo de aprendizado, adaptação ou inferência.
    - Inteligência Computacional $\Rightarrow$ nova tendência!
      - Redes Neurais Artificiais, Computação Evolutiva, Sistemas Nebulosos, Inteligência de Enxames, Sistemas Imunológicos Artificiais, Nuvem de Partículas, etc.

Abordagens da Inteligência Artificial

- Abordagem Simbólica
  - Representa o conhecimento por sentenças declarativas
  - Deduz consequências por métodos de raciocínio lógico
    - Exemplo:

$\forall x \; \forall y$ irmão(x,y) $\Rightarrow$ parente(x,y)
$\forall x \; \forall y \; \forall z$ pai(z,x) $\wedge$ pai(z,y) $\Rightarrow$ irmão(x,y)

pai(joão,maria).
pai(joão,eduardo).

Maria e Eduardo são parentes.

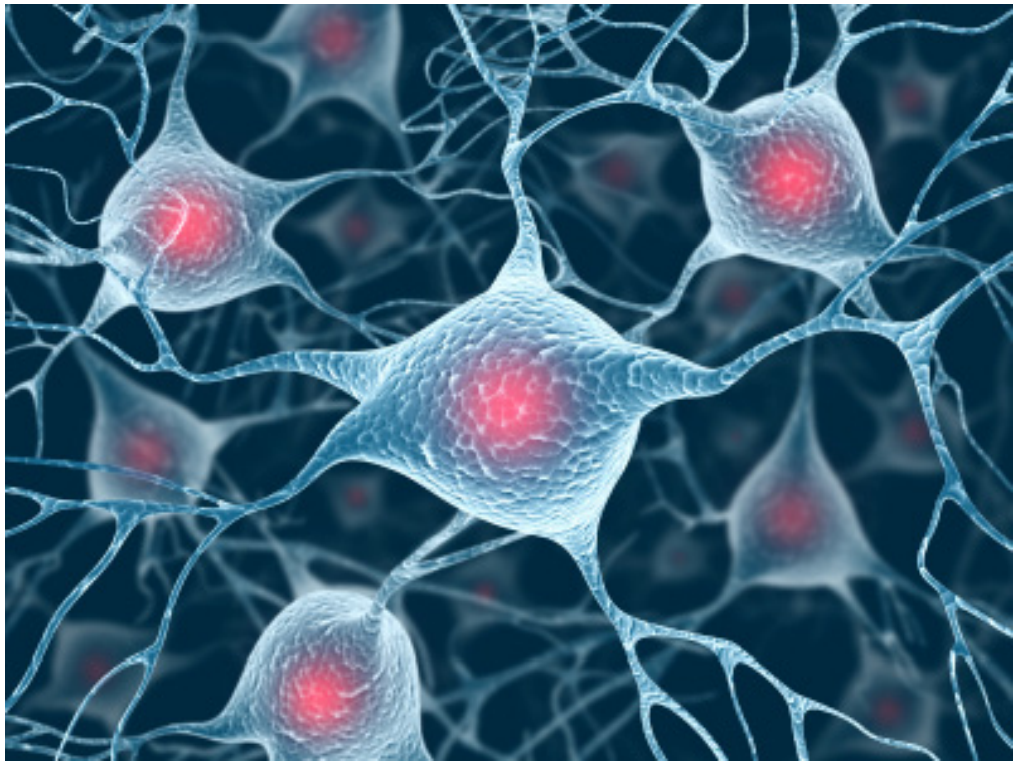
Introdução às Redes Neurais Artificiais

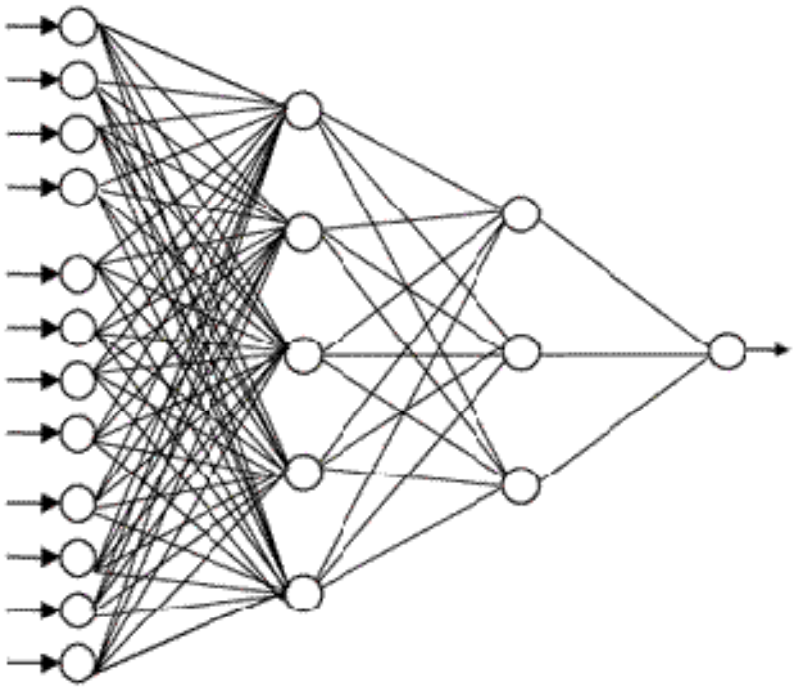
- Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados no cérebro humano e que possuem a capacidade de aquisição, manutenção e uso do conhecimento
- Modelos computacionais distribuídos, compostos por um conjunto de unidades de processamento (“neurônios”), dispostas em uma ou mais camadas, e que são interligadas por diversas conexões (“sinapses”)

Introdução às Redes Neurais Artificiais

- Essas conexões são associadas a pesos, os quais armazenam o conhecimento representado no modelo e servem para ponderar a entrada recebida por cada neurônio da rede
- Em RNAs, o processo usual na solução de problemas passa inicialmente por uma fase de aprendizagem (treinamento) e outra fase de teste

Estrutura neural





Principais características

- Aprendem através de exemplos (padrões de treinamento)
- Capacidade de se adaptar ou aprender
- Capacidade de generalização
- Agrupa ou organiza dados
- Tolerância a falhas
- Auto-organização
- Facilidade de implementação em *hardware* ou *software*

Quando usar RNA?

- Problema não-linear
- Dados disponíveis de forma quantitativa
- Quando os modelos matemáticos existentes não atendem ao problema
- Quando os modelos existentes são muito particularizados, resolvendo apenas parte do problema
- Quando o esforço computacional com modelos existentes é muito grande
- Quando a precisão obtida pelos modelos existentes não é satisfatória

Áreas de aplicação

- Reconhecimento de padrões
  - Atribuir um padrão de entrada a uma das várias classes pré-definidas
    - Reconhecimento de faces, impressões digitais, voz.
- Categorização ou *Clustering*
  - Explorar semelhanças entre padrões e agrupá-los
    - *Mineração de dados.*
- Aproximação de funções
  - Encontrar uma estimativa *y* de uma função desconhecida *f*, a partir de um conjunto de valores representativos
    - Problemas de modelagem científica e de engenharia.

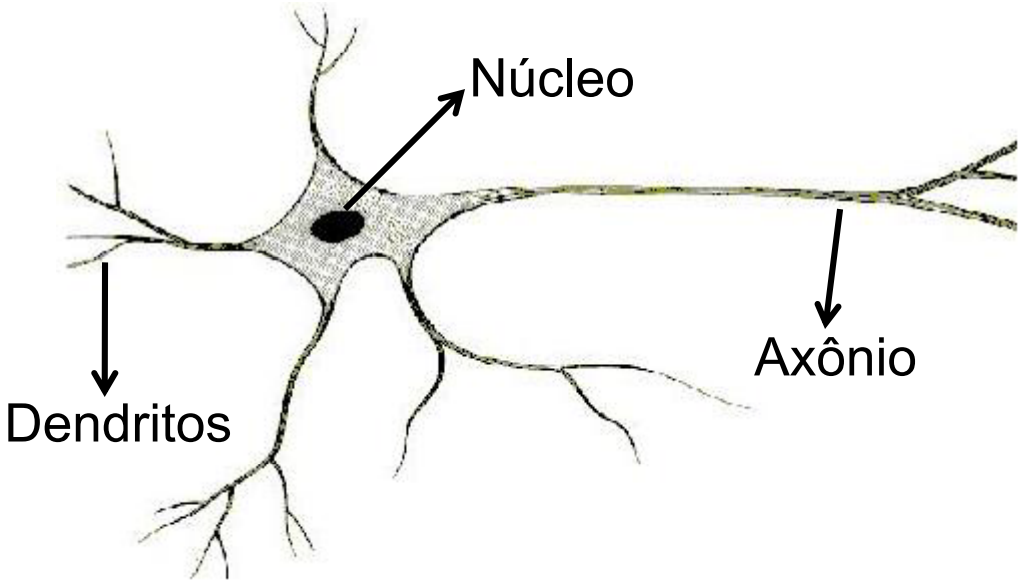
Áreas de aplicação

- Sistemas de Controle
  - Identificar ações de controle que permitam o alcance dos requisitos de qualidade, eficiência e segurança do processo.
    - Controle de robôs, elevadores, eletrodomésticos.
- Previsão/Estimação
  - Dado um conjunto de exemplos $y(t_1)$, $y(t_2)$, ..., $y(t_n)$, prever o valor $y(t_{n+1})$, no instante t_{n+1}
    - Mercado financeiro, previsões climáticas.
- Otimização
  - Minimizar ou maximizar uma função, sujeita ou não a restrições
    - Otimização não-linear, otimização combinatorial.

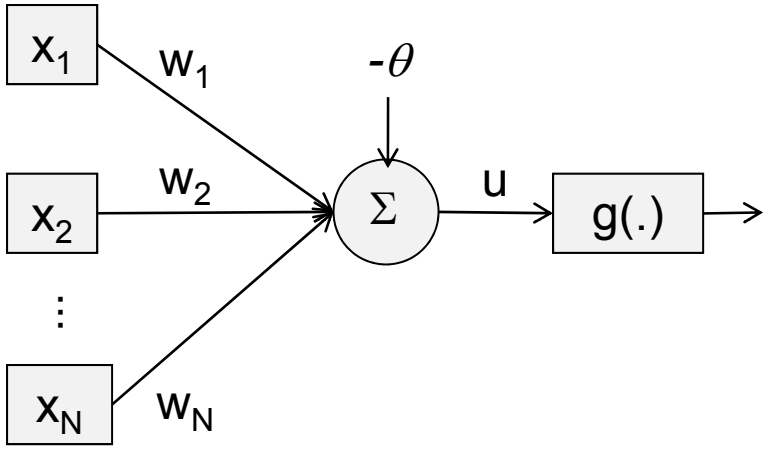
Áreas de aplicação

- Memórias Associativas
  - Recuperar item correto mesmo que a entrada seja parcial ou distorcida.
    - Processamento de imagens, transmissão de sinais, identificação de caracteres.

Modelo biológico vs artificial



Neurônio biológico



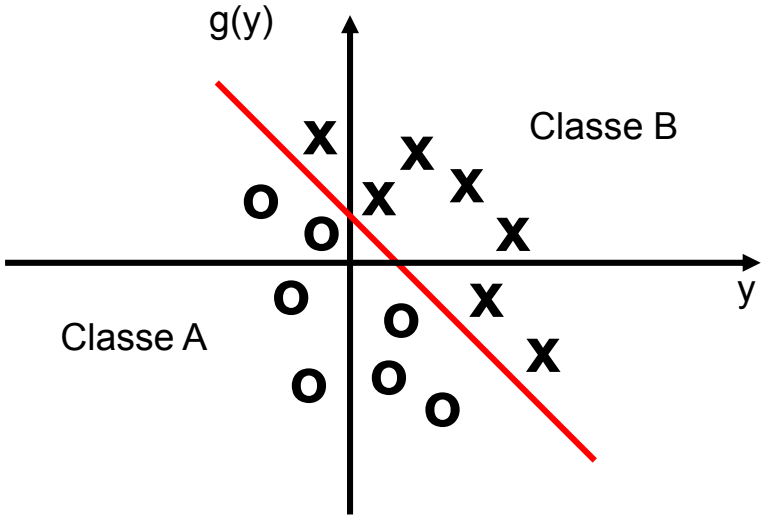
$$u = \sum_{i=1}^{N} W_i \cdot x_i - \theta$$

$$y = g(u)$$

Neurônio artificial

Análise do neurônio

- O neurônio definido por McCulloch e Pitts, comporta-se como um classificador de padrões que separa duas regiões linearmente separáveis, dividindo o espaço de entrada através de uma reta (equação linear)



$$y = ax + b$$

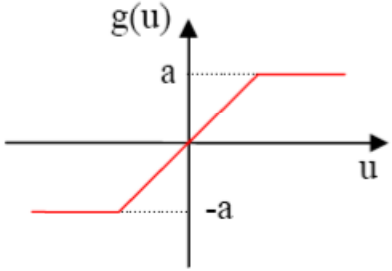
$$y = \sum_{i=1}^{N} W_i \cdot x_i - \theta$$

Função de ativação

- A função de ativação processa o conjunto de entradas recebidas e o transforma em um estado de ativação
- Normalmente, o estado de ativação dos neurônios pode assumir os seguintes valores:
  - Binários (0 e 1)
  - Bipolares (-1 e 1)
  - Reais ($-1 \leq f(.) \leq 1$) ou ($0 \leq f(.) \leq 1$)

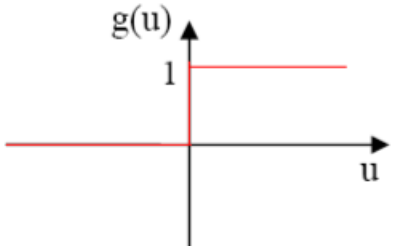
Tipos de funções de ativação

- Rampa



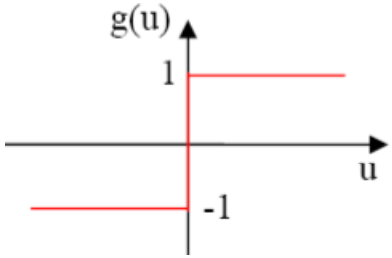
$$g(u) = \begin{cases} a, \text{ se } u > a \\ u, \text{ se } -a \le u \le a \\ -a, \text{ se } u < a \end{cases}$$

- Degrau ou Limiar



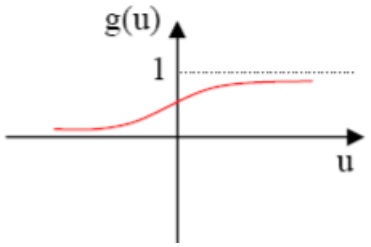
$$g(u) = \begin{cases} 1, u \ge 0 \\ 0, u < 0 \end{cases}$$

- Sinal ou Degrau Bipolar



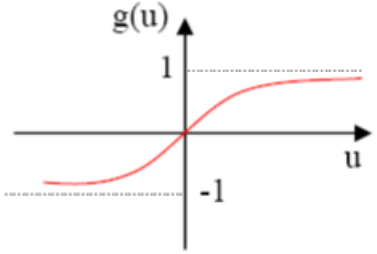
$$g(u) = \begin{cases} 1, \text{ se } u > 0 \\ 0, \text{ se } u = 0 \\ -1, \text{ se } u < 0 \end{cases}$$

- Logística ou Sigmóide



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

- Tangente Hiperbólica



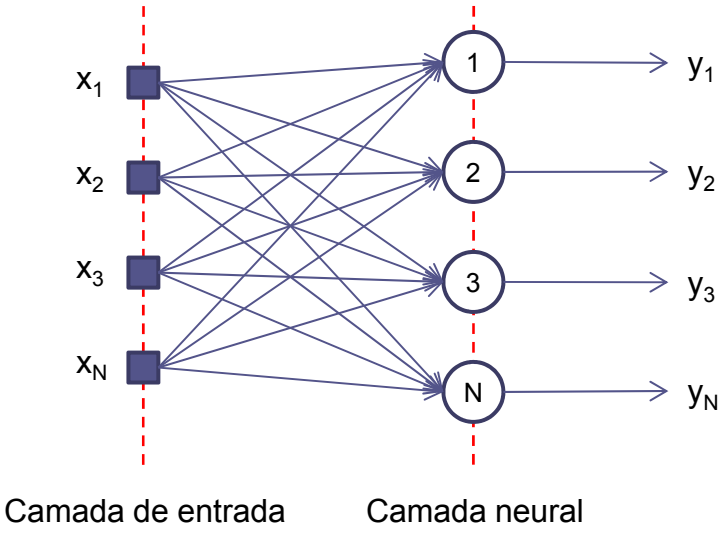
$$g(u) = \frac{1 - e^{-\beta u}}{1 + e^{\beta u}}$$

Arquiteturas de RNAs

- A arquitetura de uma RNA está relacionada com a maneira em que os neurônios estão arranjados
- A topologia de uma RNA define as características intrínsecas que diferenciam as redes dentro de uma mesma arquitetura.
  - Exemplo: número de neurônios na primeira camada neural, número de neurônios na camada de saída, número de camadas, etc.

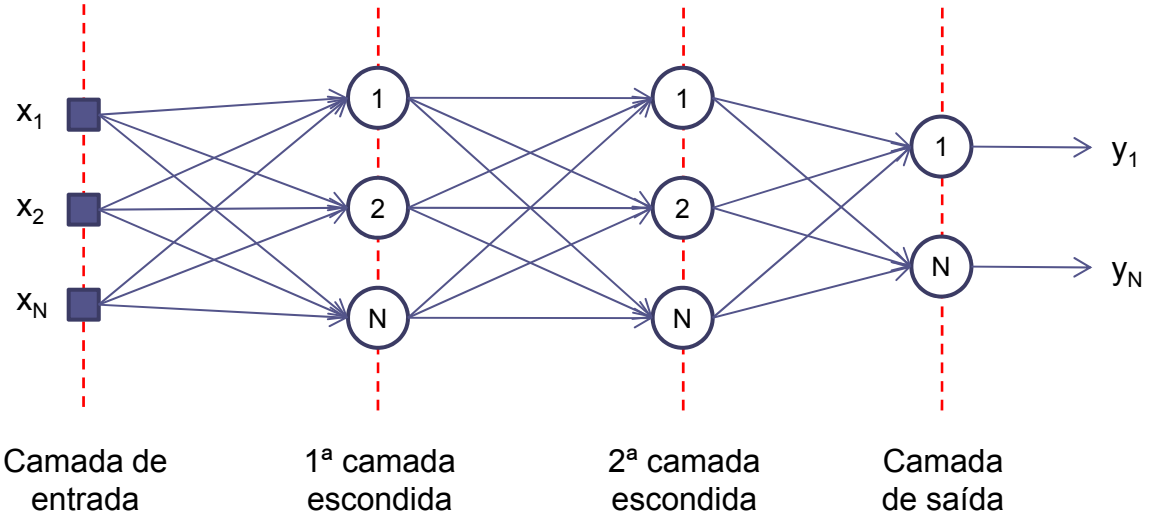
Arquiteturas de RNAs

- Rede *Feedforward* de Camada Única
  - Neste tipo de rede tem-se uma camada de entrada e uma única camada de neurônios, que é a própria camada de saída
    - Tipos: Perceptron e Adaline
    - Aplicações: Memória associativa, reconhecimento de padrões



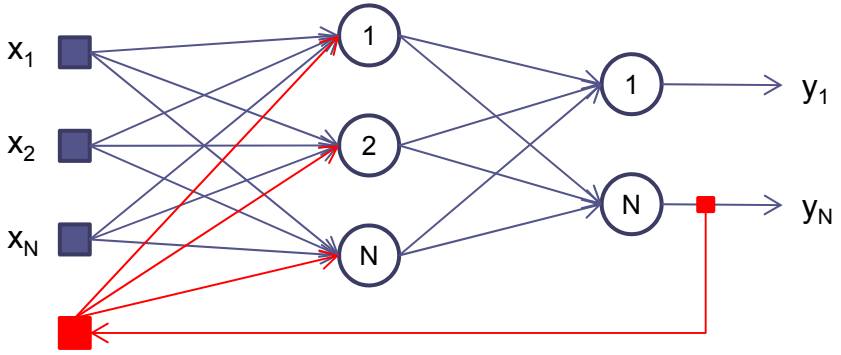
Arquiteturas de RNAs

- Rede *Feedforward* Multicamadas
  - Esta rede difere da anterior pela presença de uma ou mais camadas escondidas de neurônios
    - Tipos: Perceptron Multicamadas e Funções de Base Radial
    - Aplicações: Reconhecimento de padrões, aproximação funcional, identificação e controle



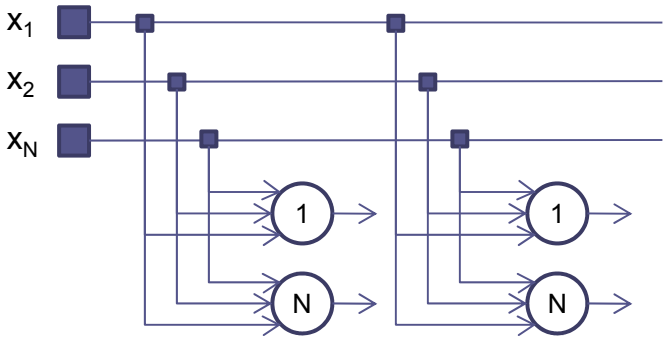
Arquiteturas de RNAs

- Redes Recorrentes
  - São redes que contem retroalimentação entre neurônios de camadas diferentes
    - Tipos: Perceptron com Realimentação e Redes de Hopfield
    - Aplicações: Controle, previsão, séries temporais, memórias associativas, otimização



Arquiteturas de RNAs

- Redes *Lattice* ou Reticulada
  - Consiste em um vetor de neurônios de uma ou mais dimensões. Os sinais de entrada são os mesmos de todos os neurônios
  - É uma rede *feedforward* cujos neurônios são arranjados em linhas e colunas
    - Tipos: Redes de Kohonen
    - Aplicações: Grafos, *clustering*



Treinamento e Teste

- Fase de Treinamento
  - Um conjunto de exemplos é apresentado à rede, a qual extrai automaticamente as características necessárias para representar a informação fornecida
- Fase de Teste
  - Fase em que as características extraídas no treinamento são utilizadas para gerar respostas para o problema

Treinamento de RNAs

- Conjunto de procedimentos bem definidos que consiste em ajustar os pesos sinápticos e limiares de uma RNA de forma que a aplicação de um conjunto de entradas produza um conjunto de saídas desejadas
  - Este ajuste é feito através de uma forma continuada de estímulo pelo ambiente no qual a rede está operando
  - O tipo específico de aprendizagem é definido pela maneira particular de como ocorrem os ajustes nos pesos

Tipos de treinamento

- Supervisionado
  - A rede é treinada para fornecer a saída desejada a um estímulo de entrada específico
- Não Supervisionado
  - Não há uma saída específica em relação aos estímulos de entrada
  - A rede se auto-organiza em relação às particularidades dos padrões de entrada

Principais periódicos

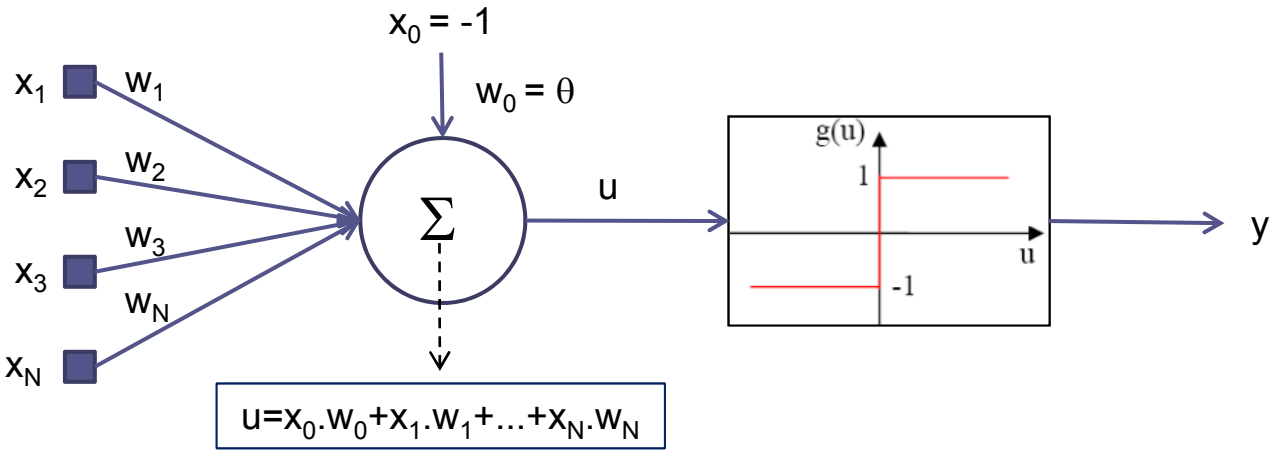
- IEEE Transactions on Neural Networks
- International Journal of Neural Systems
- Journal of Neural Networks
- Journal of Neural Computation
- Journal of Neurocomputing
- IEEE Transactions on Systems, Man and Cybernetics
- Journal of Cognitive Neuroscience

Perceptron

- O Perceptron foi proposto por Rosenblatt em 1958
- É composto pelo neurônio de McCulloch-Pitts, com função de ativação limiar ou sinal e aprendizado supervisionado
- É a forma mais simples de uma rede neural artificial
  - Possui uma única camada neural, com apenas 1 neurônio
- Princípio de Aprendizado de Hebb

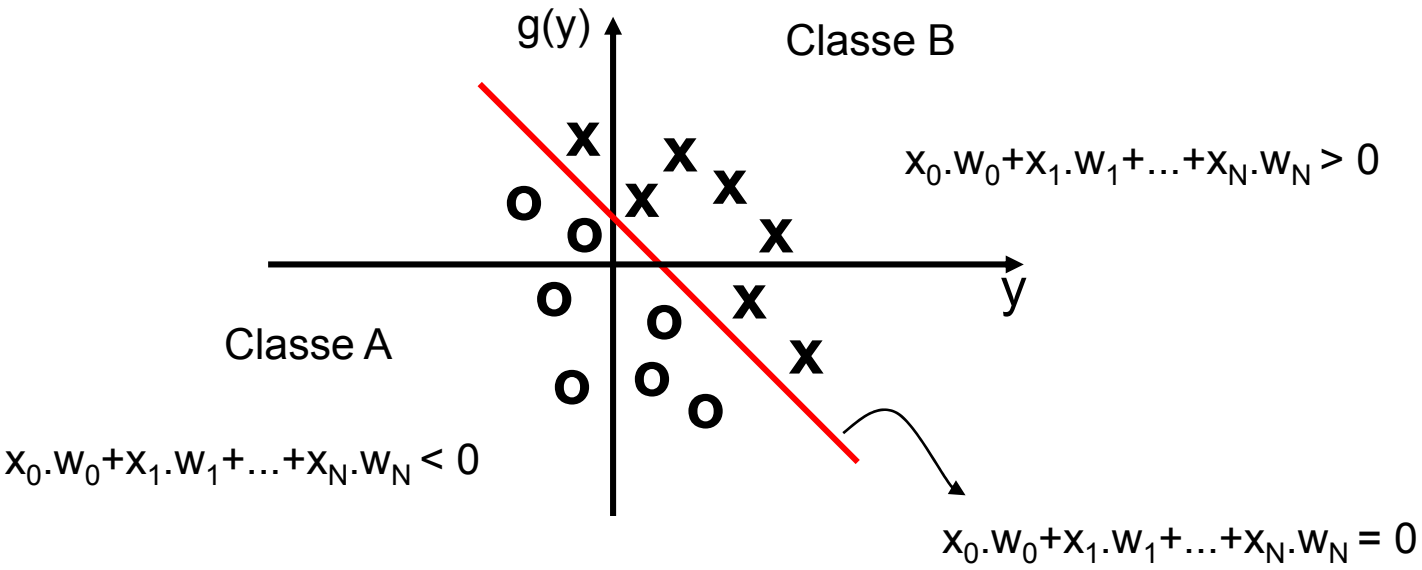
Perceptron

- Exemplo de um Perceptron com *n* entradas com função de ativação sinal



Perceptron

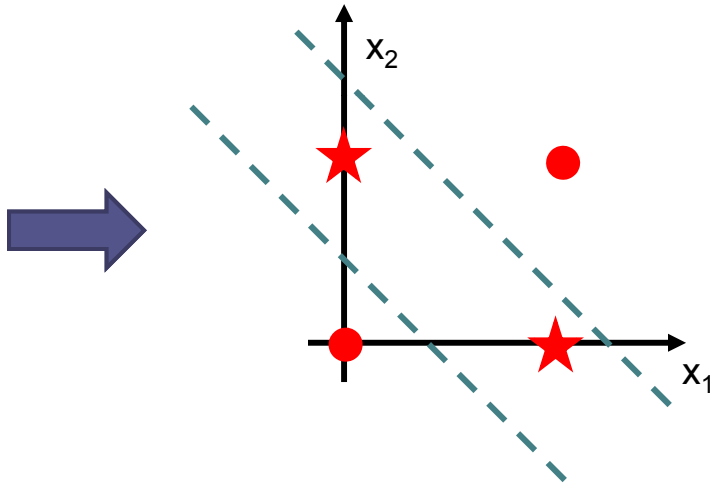
- A limitação desta rede neural se encontra na reduzida quantidade de problemas que consegue tratar
  - Classificação de conjuntos linearmente separáveis



Perceptron: Problema XOR

- Ou exclusivo não é linearmente separável

X_1	X_2	Y	Simb
0	0	0	●
0	1	1	★
1	0	1	★
1	1	0	●



- Com apenas uma reta não é possível separar as classes. Neste caso é necessário duas retas para separá-las

Perceptron: Treinamento

- Princípio de Hebb:
  - “Quando um axônio de uma célula A está próximo o suficiente para excitar uma célula B e repetidamente ou persistentemente participa da ativação desta, um processo de crescimento ou mudança metabólica ocorre em uma ou ambas as células, de tal forma que a eficiência de A em ativar B é aumentada.”
- Esta afirmação foi feita em um contexto neurobiológico
- Pode-se reescrevê-la como uma regra em duas partes:

Perceptron: Treinamento

1. Se dois neurônios em ambos os lados de uma sinapse são ativados simultaneamente, então a força daquela sinapse é seletivamente aumentada
2. Se dois neurônios em ambos os lados de uma sinapse são ativados assincronamente, então aquela sinapse é seletivamente enfraquecida ou eliminada

- Portanto, a cada apresentação do padrão, a saída fica mais reforçada ou enfraquecida!

Perceptron: Treinamento

- Durante o processo de treinamento do Perceptron, busca-se encontrar um conjunto de pesos que defina uma reta que separe as duas classes, de forma que a rede classifique corretamente as entradas apresentadas
  - Parâmetros:
    - x(k) = vetor de entrada do padrão k
    - w = vetor de pesos
    - y = saída do neurônio
    - d(k) = saída desejada em relação ao padrão k
    - η = taxa de aprendizagem ($0 < \eta < 1$)

Perceptron: Treinamento

- O ajuste de pesos é feito utilizando a seguinte equação:

$$w_i \leftarrow w_{i-1} + \eta.(d(k) - y).x(k)$$

- Quando não houver alteração da matriz *w* entre duas respostas sucessivas, a rede está treinada!

Perceptron: Treinamento

- Algoritmo de treinamento

Inicializar o vetor de pesos com valores aleatórios;
Inicializar a taxa de aprendizado;
Repita
    Erro ← “não existe”;
    Para cada par de treinamento {x(k),d(k)} faça
        $u \leftarrow x(k)^T . w$;
        y ← Sinal(u);
        Se (d(k) ≠ y) então
            $w_i \leftarrow w_{i-1} + \eta.(d(k) - y).x(k)$
            Erro ← “existe”;
        fim_se;
    fim_para;
Até Erro = “não existe”;

Perceptron: Treinamento

- Neste algoritmo, o ajuste na matriz de pesos w é realizado apenas em função de informações locais à sinapse
  - Se d(k) = 1 e y = -1 ou d(k) = -1 e y = 1
  - Altera o vetor de pesos

Perceptron: Teste

- Algoritmo de teste

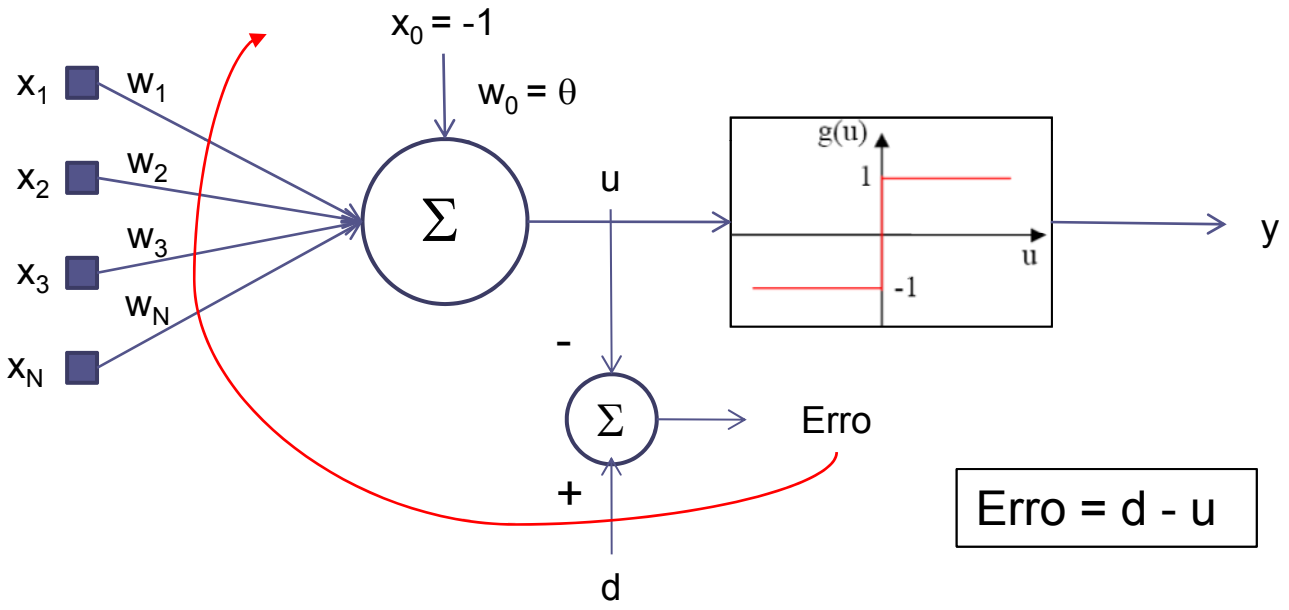
Apresentar padrão x a ser reconhecido;
u $\leftarrow$ x(k)^T * w;
y $\leftarrow$ Sinal(u);
    Se (y = 1) então
        x $\in$ “Classe 1”;
    Senão
        x $\in$ “Classe 2”;
    fim_se;

Adaline

- O Adaline (*Adaptative Linear Element*), idealizado por Widrow e Hoff, é uma rede neural que utiliza um algoritmo supervisionado para minimizar o erro entre as entradas e saídas
  - Também é composto apenas por uma camada neural com um único neurônio
  - Funciona como um filtro que separa duas classes linearmente separáveis
- Principais contribuições
  - Invenção do algoritmo de treinamento conhecido como Regra Delta
  - Aplicações em processamento de sinais desde 1960

Adaline

- A arquitetura básica do Adaline é constituída por:



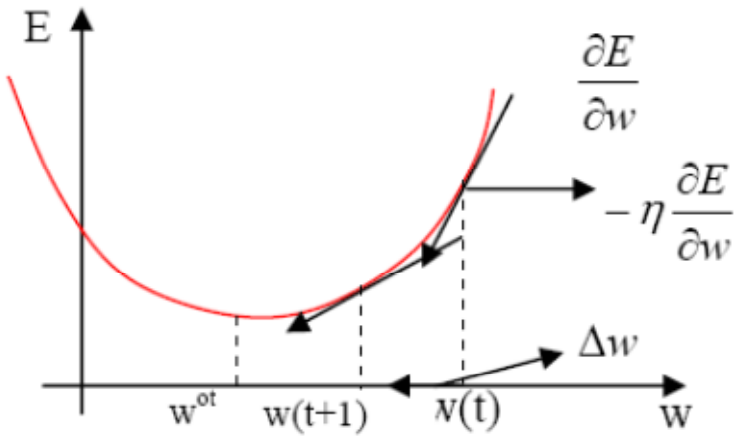
Regra Delta

- O ajuste dos pesos no algoritmo de aprendizagem (Regra Delta) do Adaline é feita minimizando o *erro quadrático* entre a saída desejada (*d*) e a saída do combinador linear (*u*)
  - A minimização do erro é feita com a utilização do vetor gradiente (∇) do erro em relação a *w*
  - Esta metodologia torna o sistema mais robusto (tolerante) em relação às flutuações dos ruídos nos sinais de entradas

Regra Delta

- Interpretação geométrica
  - Como o erro quadrático possui um termo quadrático em w, este possui forma de parábola.
    - Para p padrões de entrada, a função erro quadrático é definida por:

$$E(w) = \frac{1}{2}\sum_{k=1}^{p}(d(k) - u(k))^2$$



O η não pode ser grande por causa da estabilidade em convergir para o ótimo.

Adaline: Treinamento

- O ajuste de pesos é feito utilizando a seguinte equação:

$$w_t \leftarrow w_{t-1} + \eta \,.\, (d(k) - u(k)) \,.\, x(k)$$

- O critério de parada é estipulado em função do erro quadrático médio, definido por:

$$EQM(t) = \frac{1}{p}\sum_{k=1}^{p}(d(k) - u(k))^2$$

- O algoritmo converge quando o EQM entre duas épocas sucessivas for suficientemente pequeno, ou seja:

$$| EQM(t) - EQM(t-1) | \leq Erro$$

Adaline: Treinamento

- Algoritmo de treinamento

Inicializar o vetor de pesos com valores aleatórios;
Inicializar a taxa de aprendizado;
EQM_ant $\leftarrow$ INF;
EQM_atual $\leftarrow$ 1;
Enquanto | EQM_atual - EQM_ant | > Erro
    EQM_ant $\leftarrow$ EQM_atual;
    Para cada par de treinamento {x(k),d(k)} Faça
        u $\leftarrow$ $x(k)^T$ * w;
        w $\leftarrow$ w + η . (d(k) - u) . x(k);
    Fim_para;
    EQM_atual $\leftarrow$ EQM;
Fim_enquanto;

Adaline: Teste

- Algoritmo de teste

Apresentar padrão ‘x’ a ser reconhecido;
$u \leftarrow x(k)^T * w$;
$y \leftarrow$ Sinal(u);
Se (y = 1) então
    x $\in$ “Classe 1”;
Senão
    x $\in$ “Classe 2”;
Fim_se;

Perceptron VS Adaline

- O Perceptron pode alterar ou não os pesos, uma ou mais vezes por padrão de entrada
- O Adaline altera os pesos com todos os padrões de entrada, e novamente para cada vez que EQM_atual - EQM_ant > Erro
- O Perceptron se caracteriza como separador linear e o Adaline como aproximador linear de funções