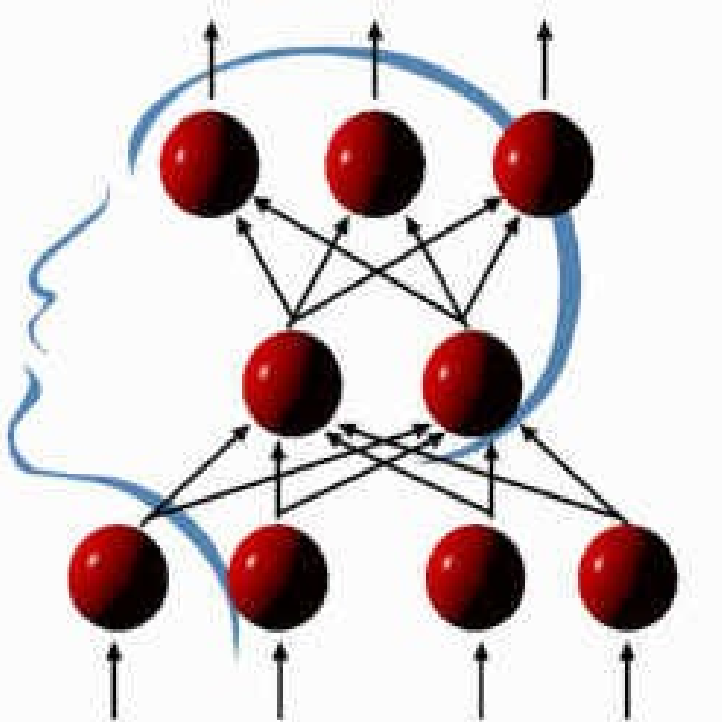




Redes Neurais Artificiais

Inteligência Artificial

Prof. Cedric Luiz de Carvalho
Instituto de Informática
UFG 2006





Tópicos

- Introdução
- Redes Neurais Humanas
- O que são Redes Neurais Artificiais
- Características das Redes Neurais Artificiais
- *Perceptrons*
- *Perceptrons* Multi Camadas
- Modelo de *Hopfield*



Motivação

- Criar máquinas capazes de operar independentemente do homem
 - aprender sozinhas
 - interagir com ambientes desconhecidos
 - possa ser chamada de autônoma, inteligente ou cognitiva
 - capacidade de lidar com eventos inesperados
- Obs.: **cognição** → **aquisição de um conhecimento**

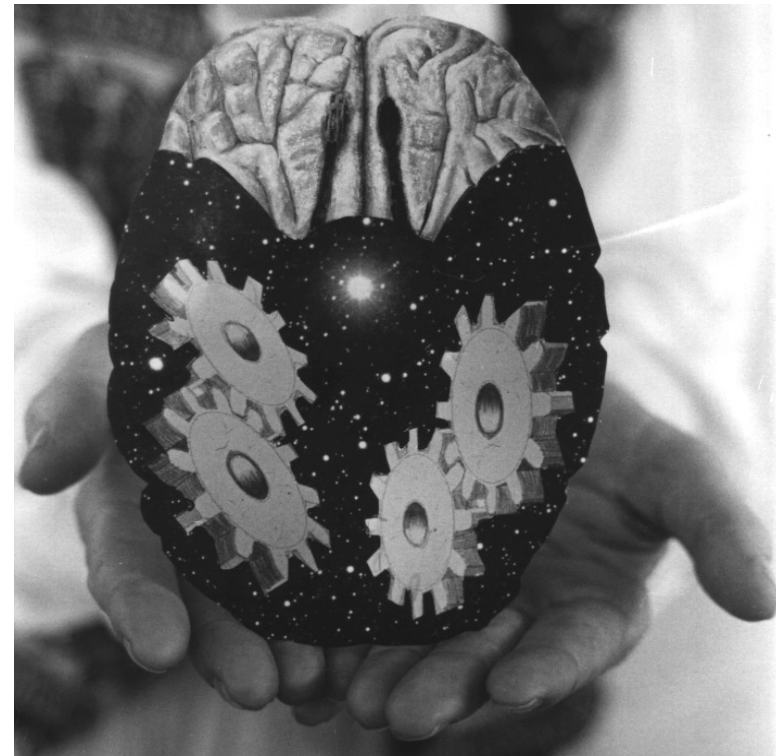


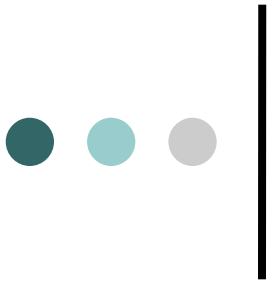
Utilidade

- Teriam maior capacidade de **aprender tarefas de alto nível cognitivo** que não são facilmente manipuladas por máquinas atuais
- Seriam úteis onde a presença humana é perigosa, tediosa ou impossível, como em:
 - reatores nucleares
 - combate ao fogo
 - operações militares
 - exploração ao espaço...

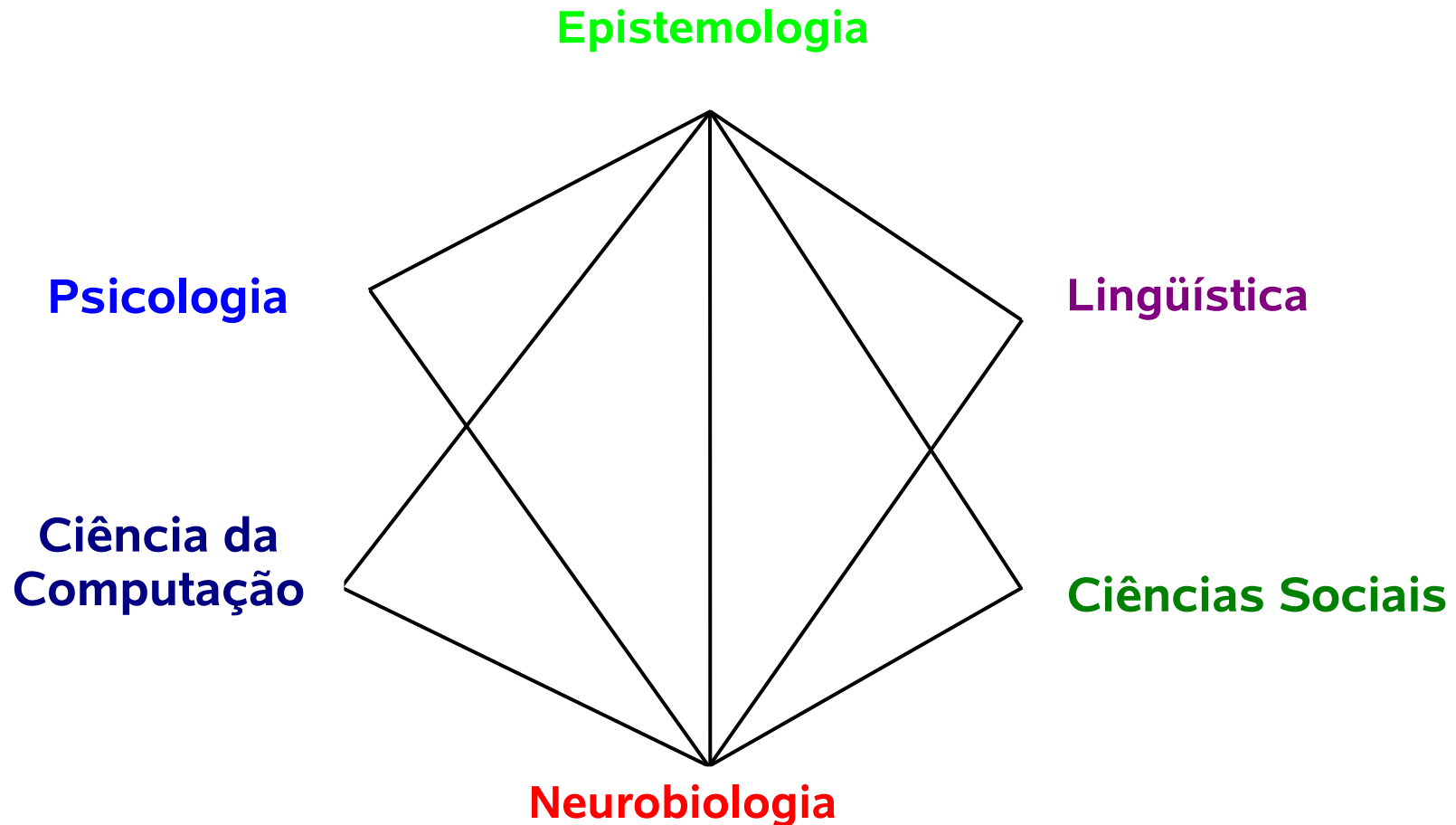
Redes Neurais Artificiais (RNA)

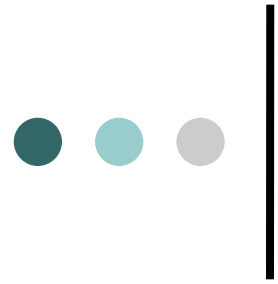
- Investigam aspectos naturais e adaptativos do sistema cognitivo
- Aprendem a reconhecer padrões de informações estabelecidos no meio ambiente
- Abordagem *bottom-up*: a partir de elementos básicos busca o entendimento de conceitos gerais





Ciências Cognitivas



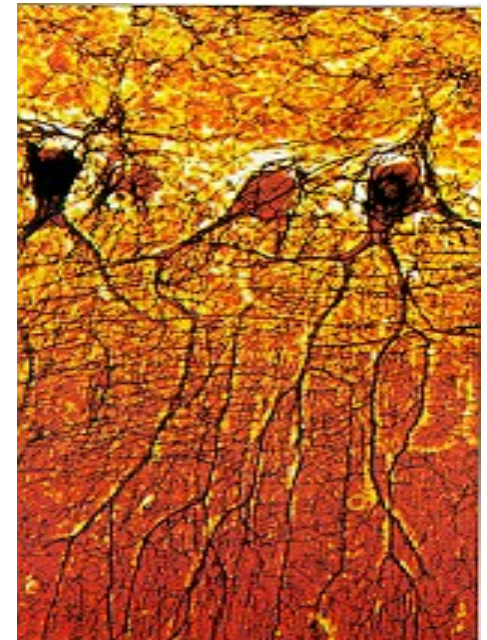
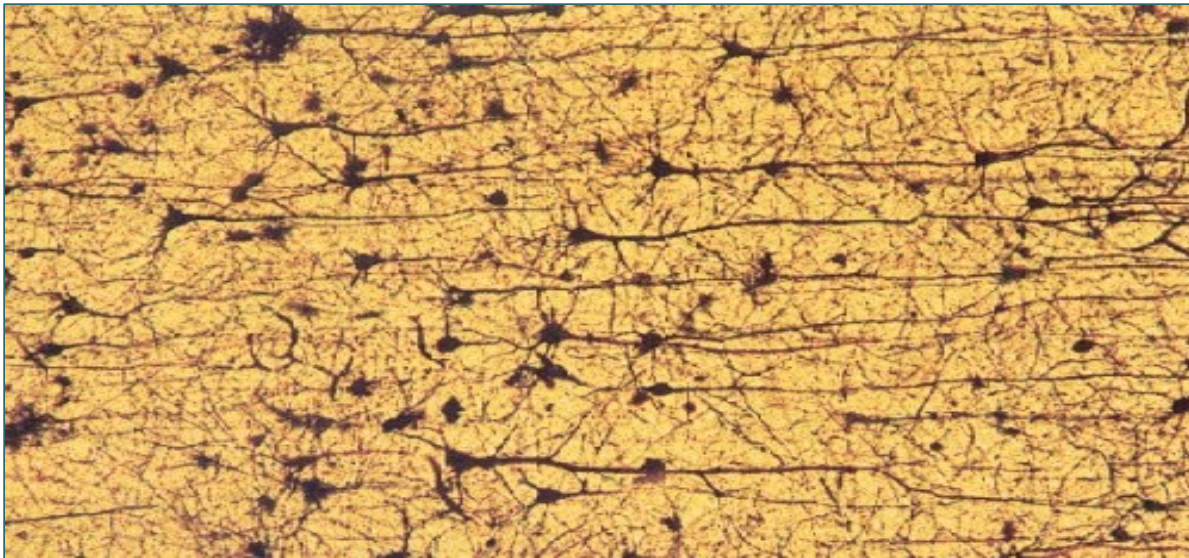


Cérebro humano

- Mais fascinante processador baseado em **carbono**
- O **neurônio** é um célula no cérebro cuja principal função é colecionar, processar e disseminar sinais elétricos
- **10 bilhões** de neurônios
 - todos movimentos do organismo
 - são conectados através de sinapses
 - processam e armazenam informações

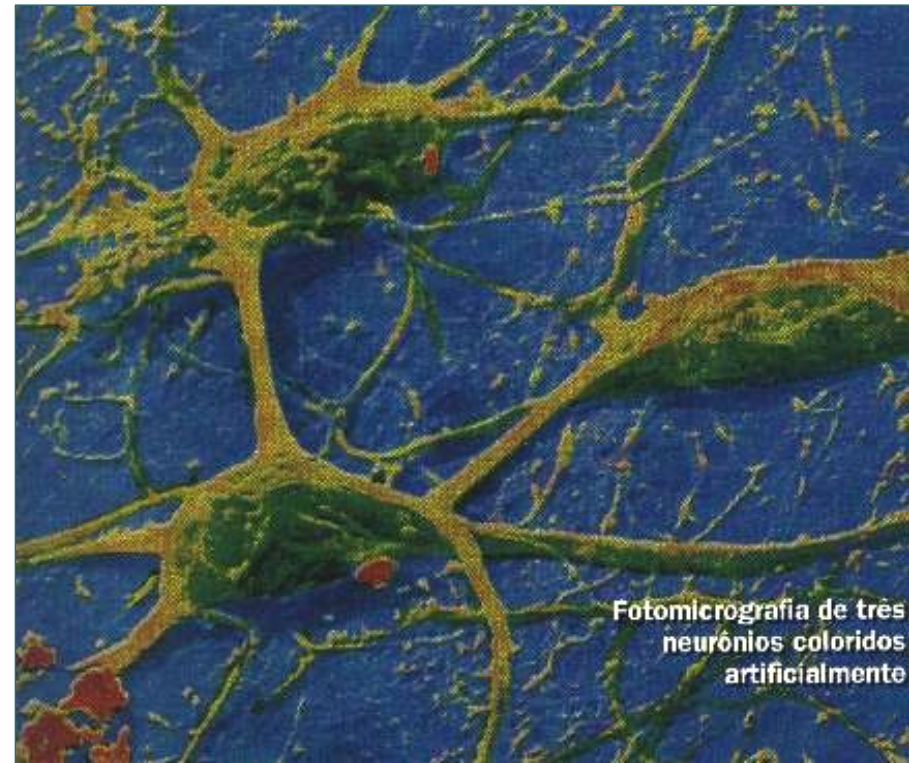
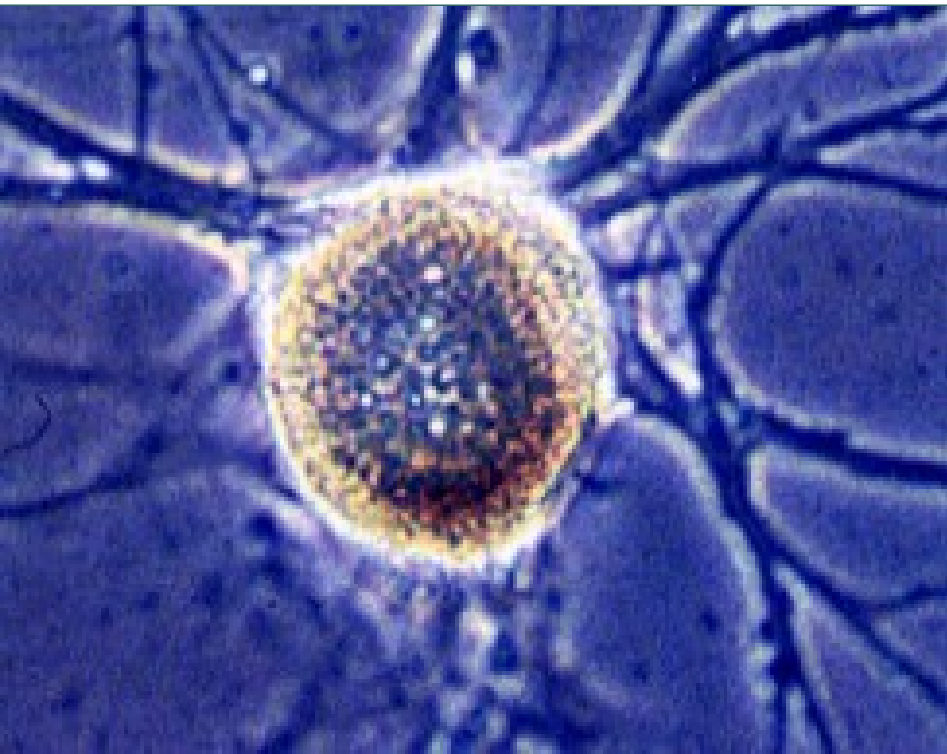
Redes neurais naturais

- O sistema nervoso é formado por um conjunto extremamente complexo de células, os **neurônios**
- O cérebro humano possui cerca de 10^{11} **neurônios** e mais de 10^{14} *sinapses*, possibilitando a formação de redes muito complexas



Redes neurais naturais

- Neurônios têm papel essencial na determinação do funcionamento e comportamento do corpo humano e do raciocínio



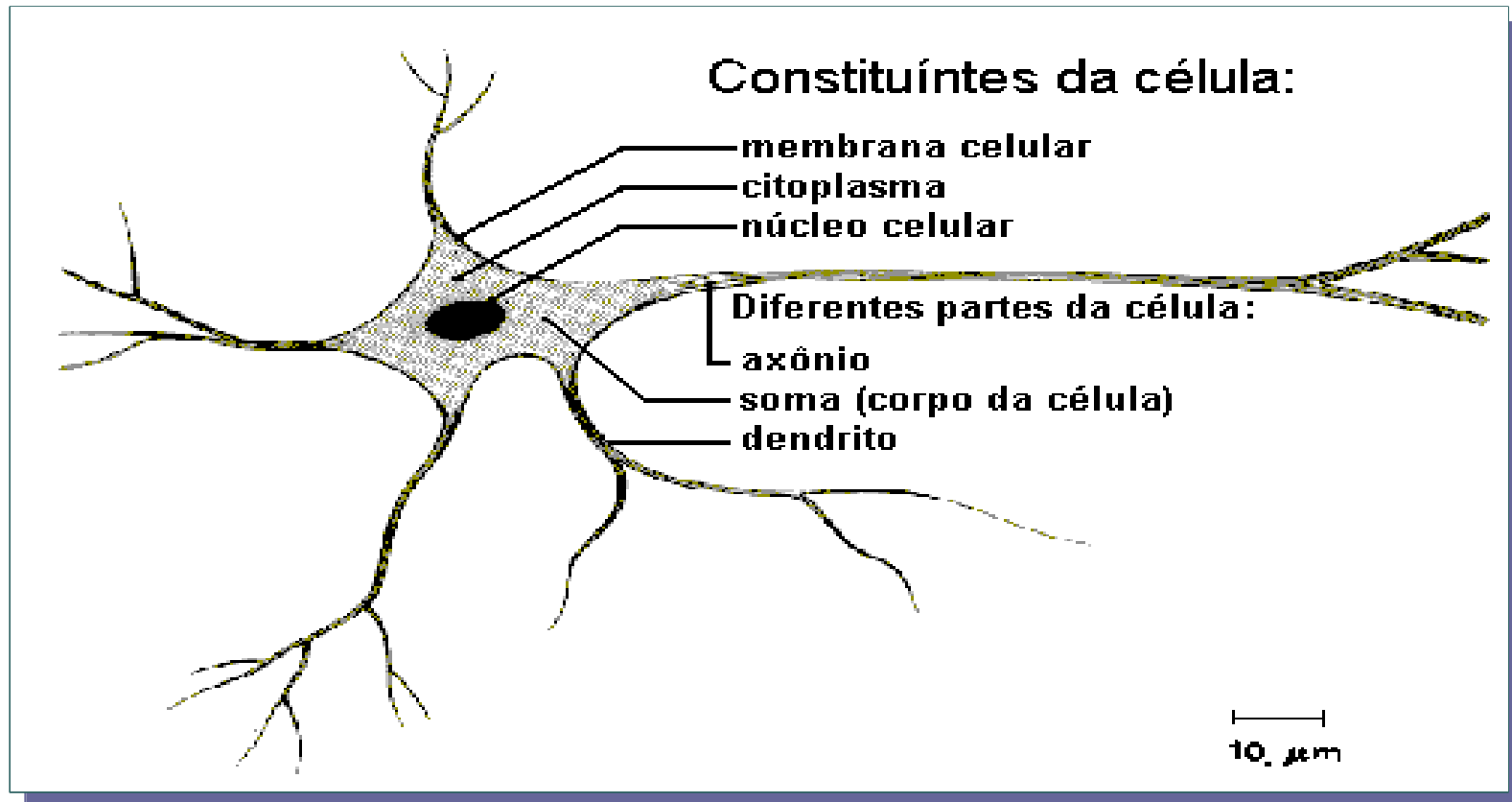
Fotomicrografia de três
neurônios coloridos
artificialmente



Neurônios naturais

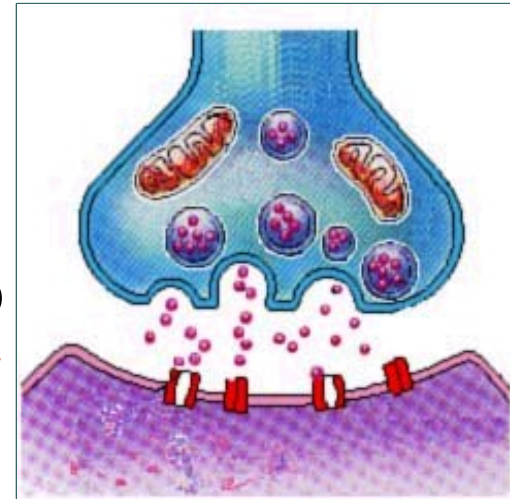
- Neurônios são formados:
 - pelos **dendritos**, que são um conjunto de terminais de entrada
 - recebem estímulos
 - pelo **corpo central** (soma)
 - coleta, combina e processa informações
 - pelos **axônios** que são longos terminais de saída
 - transmitem os estímulos
- Neurônios se comunicam através de ***sinapses***

Neurônios naturais



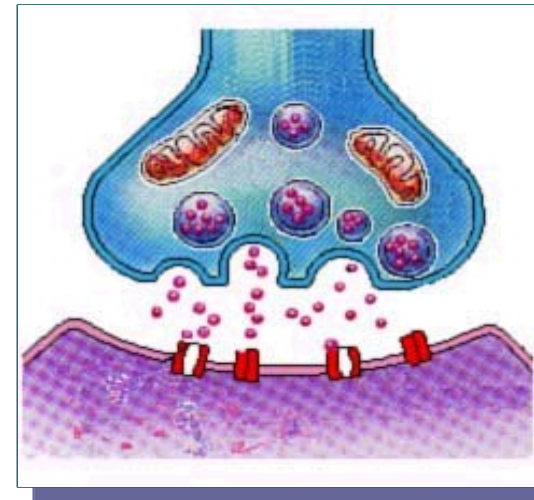
Neurônios naturais: Sinapse

- É o contato entre dois neurônios através da qual os **impulsos nervosos** são transmitidos entre eles
- Os impulsos recebidos por um neurônio são processados e, atingindo um **limiar de ação**, dispara, produzindo uma substância **neurotransmissora** que flui para o **axônio**, que pode estar conectado a um **dendrito** de outro neurônio



Neurônios naturais: Sinapse

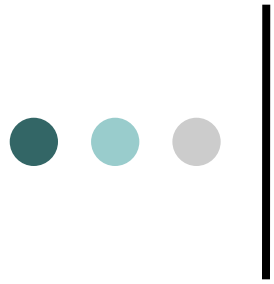
- O **neurotransmissor** pode diminuir ou aumentar a polaridade da membrana **pós-sináptica**, inibindo ou excitando a geração dos pulsos no outro neurônio
- Este processo depende de fatores como **geometria da sinapse** e o tipo de **neurotransmissor**





Neurônios naturais

- Constatação que o cérebro processa informações de forma diferente dos computadores convencionais
- **Cérebro**
 - velocidade 1 milhão de vezes mais lenta
 - processamento altamente paralelo
 - 10^{11} neurônios com 10^4 conexões cada
- **Computador**
 - processamento extremamente rápido e preciso na execução de sequência de instruções



Problema dos 100 Passos

- **Neurônio**: 2ms
- **Processador**: 2ns
- **Processador** é 10^6 mais rápido que o neurônio
- **Cérebro** reage a um estímulo entre 0,2 e 1 segundo
- **Cérebro** responde em 100 passos



Neurônios naturais

- O **cérebro** tem **10 bilhões** de neurônios
- Cada neurônio tem **1.000** a **10.000** conexões
- **60 trilhões** de conexões
 - **10^{14} sinapses**
- Cada pessoa pode dedicar **100.000 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



Cérebro X Computador

Parâmetro	Cérebro	Computador
Material	Orgânico	Metal e plástico
Velocidade	Milisegundos	Nanosegundos
Tipo de Processamento	Paralelo	Seqüencial
Armazenamento	Adaptativo	Estático
Controle de Processos	Distribuído	Centralizado
Número de elementos processados	10 e 11 à 10 e 14	10 e 5 à 10 e 6
Ligações entre elementos processados	10.000	<10



Computador X RNAs

Computadores	Neurocomputadores
Executa programas	Aprende
Executa operações lógicas	Executa operações não lógicas, transformações, comparações
Depende do modelo ou do programador	Descobre as relações ou regras dos dados e exemplos
Testa uma hipótese por vez	Testa todas as possibilidades em paralelo



Redes Neurais Artificiais (RNAs)

- RNAs são técnicas computacionais que apresentam um **modelo matemático** inspirado na **estrutura neural de organismos inteligentes** e que adquirem conhecimento através da **experiência**
- Método de solucionar problemas de IA, que utiliza um sistema que **possui circuitos que simulam o cérebro humano**, inclusive seu comportamento, ou seja, **aprendendo, errando e fazendo descobertas**
- Uma grande RNA pode ter centenas ou milhares de **unidades de processamento**



Redes Neurais Artificiais (RNAs)

- Redes Neurais Artificiais são sistemas inspirados nos **neurônios biológicos** e na **estrutura massivamente paralela do cérebro**, com capacidade de adquirir, armazenar e utilizar conhecimento experimental
- Devido à similaridade com a estrutura do cérebro, as Redes Neurais exibem características similares ao do comportamento humano, tais como:



Redes Neurais Artificiais (RNAs)

- Procura Paralela e Endereçamento pelo Conteúdo
 - o cérebro não possui endereço de memória e não procura a informação seqüencialmente
- Aprendizado
 - a rede **aprende por experiência**, não necessitando explicitar os algoritmos para executar uma determinada tarefa



Redes Neurais Artificiais (RNAs)

○ Associação

- a rede é capaz de fazer associações entre padrões diferentes. Ex.
 - Pessoa → Nome
 - Perfume → Pessoa

○ Generalização

- são capazes de **generalizar o seu conhecimento** a partir de exemplos anteriores



Redes Neurais Artificiais (RNAs)

- Abstração

- capacidade de **abstrair a essência de um conjunto de entradas**, isto é, a partir de padrões ruidosos, extrair a informação do padrão sem ruído

- Robustez e Degradação Gradual

- a perda de um conjunto de elementos processadores não causa o mal funcionamento da rede neural



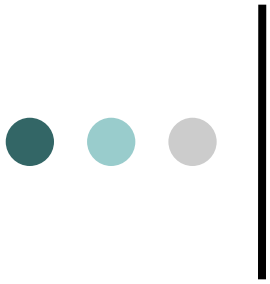
O que as RNAs não são

- RNN não são circuitos digitais
 - o modelo MCP usava sinais binários, o neurônio biológico expressa sua ativação pela **frequência que emite pulsos** e esta frequência tem uma **variação contínua** entre dois valores positivos
- RNN não podem ter excitação negativa
 - alguns modelos usam valores de **excitação negativa**



O que as RNAs não são

- RNN não são homogêneas
 - as RNN não possuem todos os seus **neurônios de mesmo tipo** como nas RNA, apenas em algumas regiões existe uma certa uniformidade no tipo de neurônios
- RNN não são circuitos síncronos ou assíncronos
 - as RNN são **sistemas de tempo contínuo**, logo não cabe a classificação de síncrono ou assíncrono



O que as RNAs não são

- Nem neurônios nem sinapses tem dois valores
- Circuitos cerebrais não são capazes de cálculos recursivos
 - isto é consequência dos neurônios não serem sistemas discretos, levando a rede a não ser um autômato
 - logo, equivalência com problemas solúveis por funções recursivas não tem sentido biológico
 - entretanto, os neurônios das RNAs são capazes de resolver funções recursivas



Fases da história da IA

- Época pré-histórica

- nesta época nada se conhecia sobre os mecanismos da mente, nem sob o prisma fisiológico nem psicológico e por esta razão vai até 1875 quando Camillo Golgi visualizou o neurônio

- Época Antiga (1875-1943)

- época em que a Lógica formal apareceu (Russel, etc)



Fases da história da IA

- Época Romântica (1943-1956)
 - o otimismo desordenado, tal um jovem rapaz romântico, **crê que tudo é possível**
 - acaba com a reunião no *Darhmouth College*
- Época Barroca (1956-1969)
 - tudo é fácil e será conseguido
 - fracasso do projeto de tradução automática de línguas pelo MIT devido a dimensionalidade
 - inglês-russo
 - o livro *Perceptrons* mostra que **nem tudo é possível**



Fases da história da IA

- **Época das Trevas (1969-1981)**
 - paralisação de quase todas as pesquisas em IA por **falta de verbas**
- **Renascimento (1981-1987)**
 - começou a corrida para IA
 - os resultados obtidos nas épocas anteriores atingiram o público em geral
 - **sistemas especialistas se popularizaram**
- **Época Contemporânea (1987 - atual)**
 - logo no início do período Gallant publica seu célebre artigo sobre **sistemas especialistas conexionalistas**
 - foi o ponto de partida para a união das duas abordagens de IA



Acontecimentos

- 1943 – Primeiras informações da neuro computação (McCulloch e Pitts)
 - Psychon –
 - Excitação / Inibição - Sem aprendizado
- 1949 - Regra de Hebb - D. O. Hebb
 - Aprendizado / Adaptação
- 1951 – Snark, por Mavin Minsky
 - operava com sucesso
 - mas, não executava funções de processamento interessantes, porém serviu de inspiração



Acontecimentos

- 1956 – “Darthmouth College” surgiram os paradigmas da Inteligência Artificial:
 - **simbólica**: simular o comportamento humano desconsiderando os mecanismos responsáveis
 - **conexionista**: simular a estrutura cerebral, acreditando-se que seria capaz de apresentar inteligência



Acontecimentos

- 1957 – Mark I Perceptron, por Frank Rosenblatt, Charles Wightman e outros
 - interesse: reconhecimento de padrões
 - Bernard Widrow desenvolveu um novo tipo de processamento de redes neurais: ADALINE
 - grande capacidade de aprendizado
 - valores discretos / binários / Regra Delta
- 1962 - Perceptron - Frank Rosenblatt
 - valores contínuos
- 1962 - Madaline - Bernard Widrow
 - combinando Adalines / Combinação manual



Acontecimentos

- 1969 - Problema do XOR - Minsky & Papert (Livro “Perceptrons”)
- 1970 - 1980 → Década perdida...
- 1980 – Campo de pesquisas explodiu:
 - Parallel Distributed Processing
- 1982 - Modelo de Hopfield
 - Redes recorrentes - Memórias Auto-Associativas



Acontecimentos

- 1986 - MLP Back-Propagation - Rumelhart, Hinton & Williams (Multi-nível)
- 1987 – Primeira conferência de redes neurais
- 1980-1990 → década das aplicações
 - Jogos, Robótica, Visão, Reconhecimento de Imagens e Padrões (OCR, Digitais, Assinaturas), Reconhecimento de Voz (Comandos e Fonemas), Previsão de séries temporais (Ações, tempo, consumo, etc)
- 1990-2005 → revendo conceitos e limitações
 - propondo novos modelos



O Modelo MCP (McCulloch e Pitts)

- Uma RNA é composta por várias **unidades de processamento (nós)**, cujo funcionamento é bastante simples
- Essas unidades geralmente são **ligadas por conexões (links)** que estão associados a um determinado **peso**
- As unidades fazem operações apenas sobre seus dados locais, que são entradas recebidas pelas suas conexões
- O comportamento inteligente de uma RNA vem das **interações** entre as unidades de processamento da rede



O Modelo MCP

- Uma ligação de uma unidade j para unidade i serve para propagar a ativação a_j de j para i
- Cada ligação possui um peso $w_{j,i}$ associado, que determinar a força e o sinal da conexão
- Cada unidade i primeiro computa a soma dos pesos de suas entradas:

$$in_i = \sum_{j=0}^n w_{j,i} a_j$$



O Modelo MCP

Função de ativação

- Então se aplica uma função de ativação ***g*** nesta soma para derivar a saída:

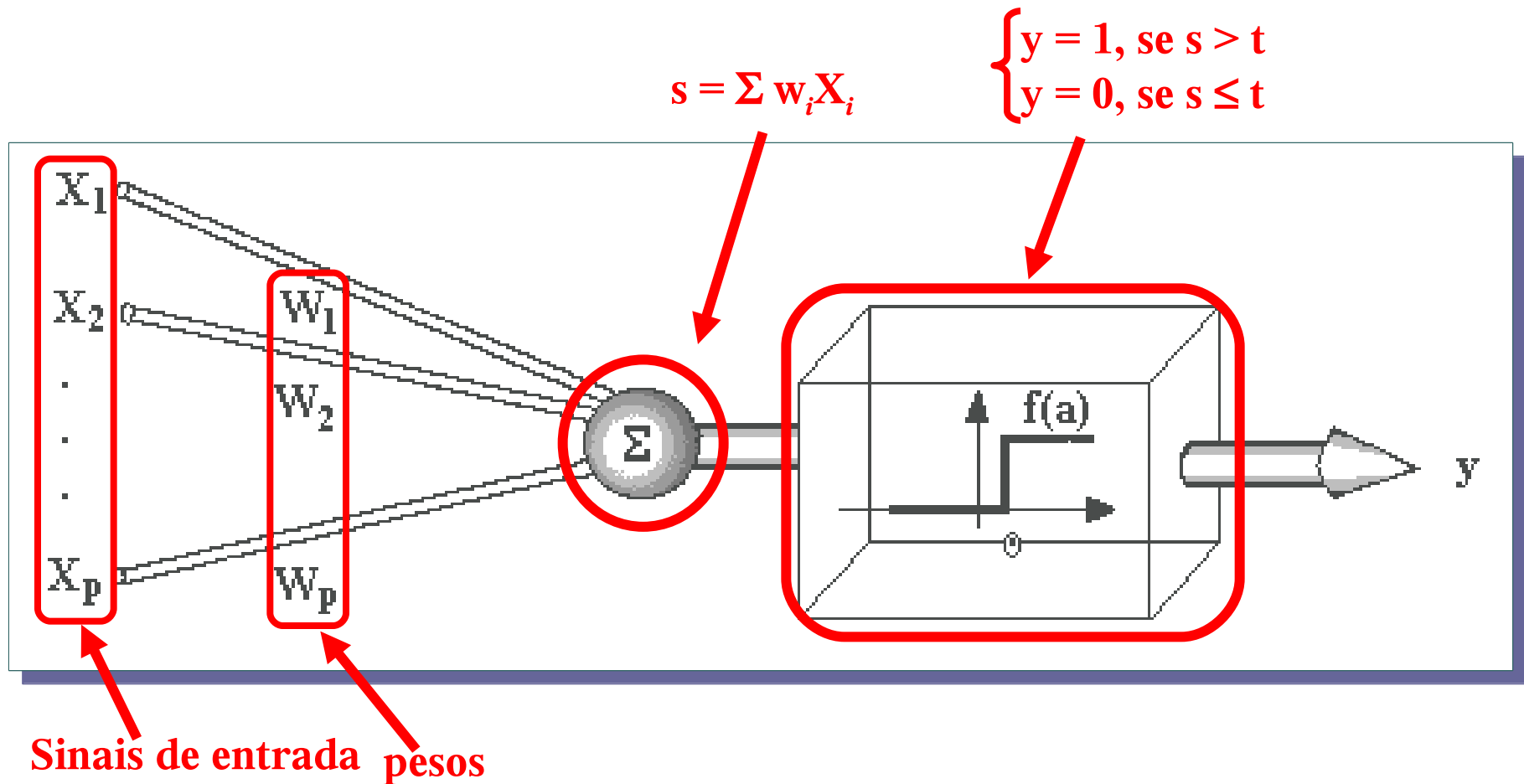
$$a_i = g(in_i) = g\left(\sum_{j=0}^n W_{j,i} a_j\right)$$

- Se este nível de atividade exceder um certo **limite ou limiar** (*threshold*) a unidade produz uma determinada resposta de saída

$$a_i \geq \theta$$

O Modelo MCP

Operação de uma unidade de processamento



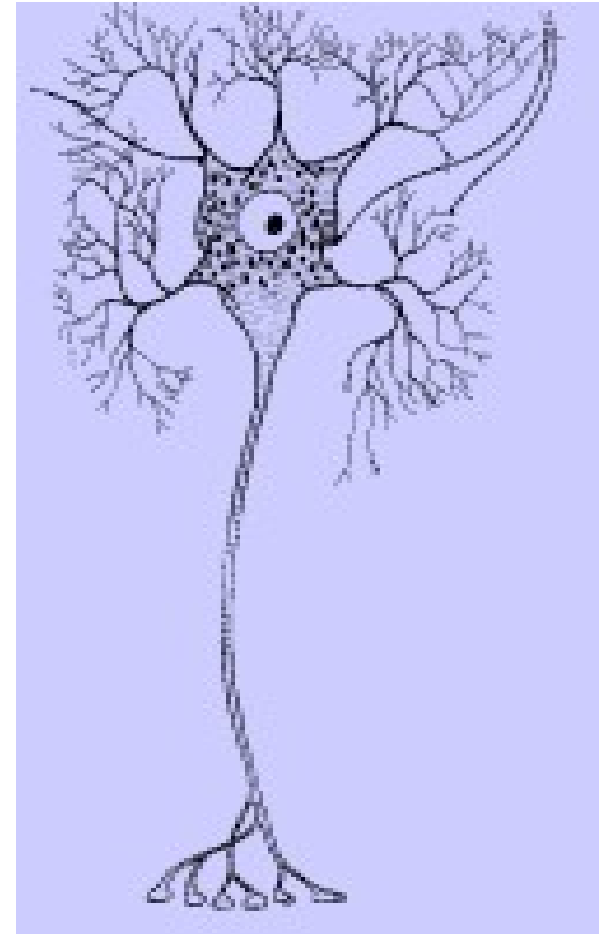
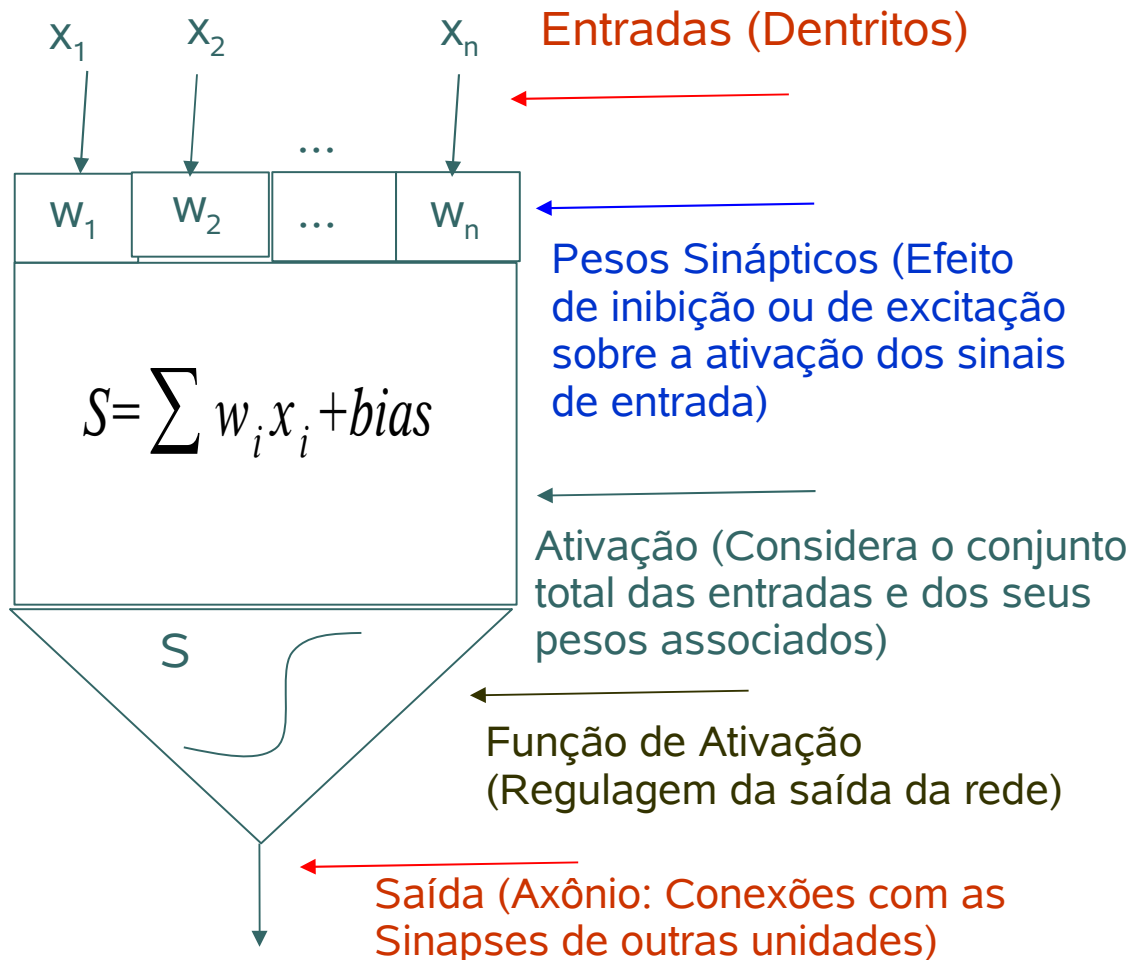


O Modelo MCP

Função de ativação

- Este modelo foi simplificado
 - os nós em cada camada da rede disparam **sincronicamente**
 - as entradas em um instante t produzem a sua saída no tempo $t+1$
 - diferente do biológico, onde não existe sincronismo e nem ativação em tempo discreto
- Além disso, possuem outras limitações
 - com apenas uma camada só conseguem implementar **funções linearmente separáveis**
 - **pesos fixos**, não ajustáveis, não há aprendizado

Características das RNAs



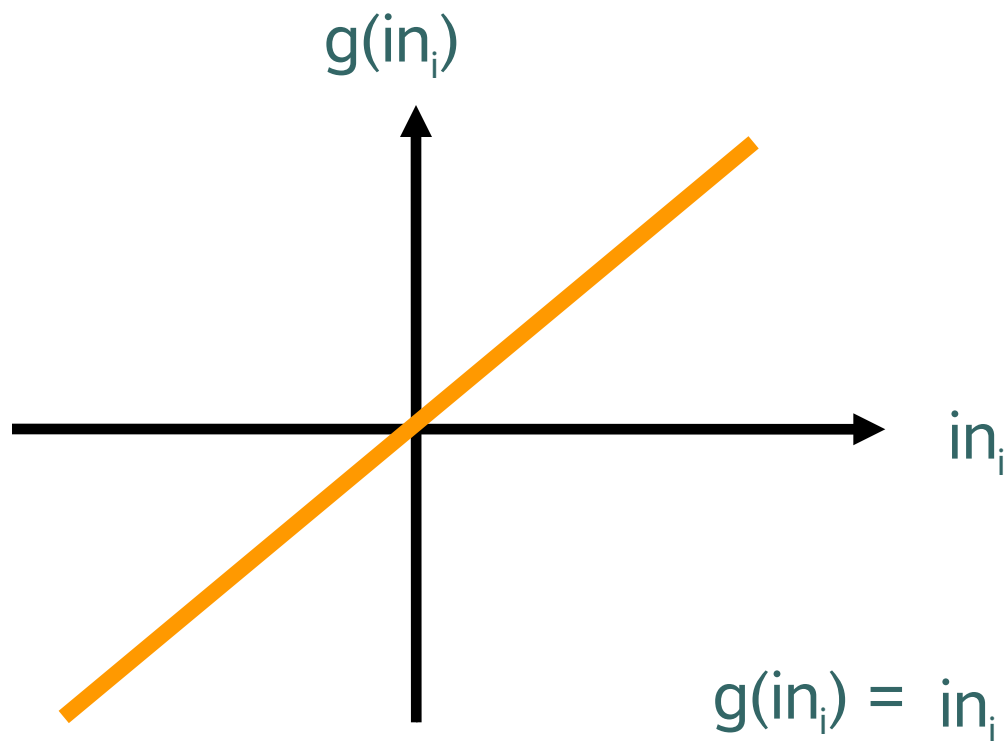


Função de ativação

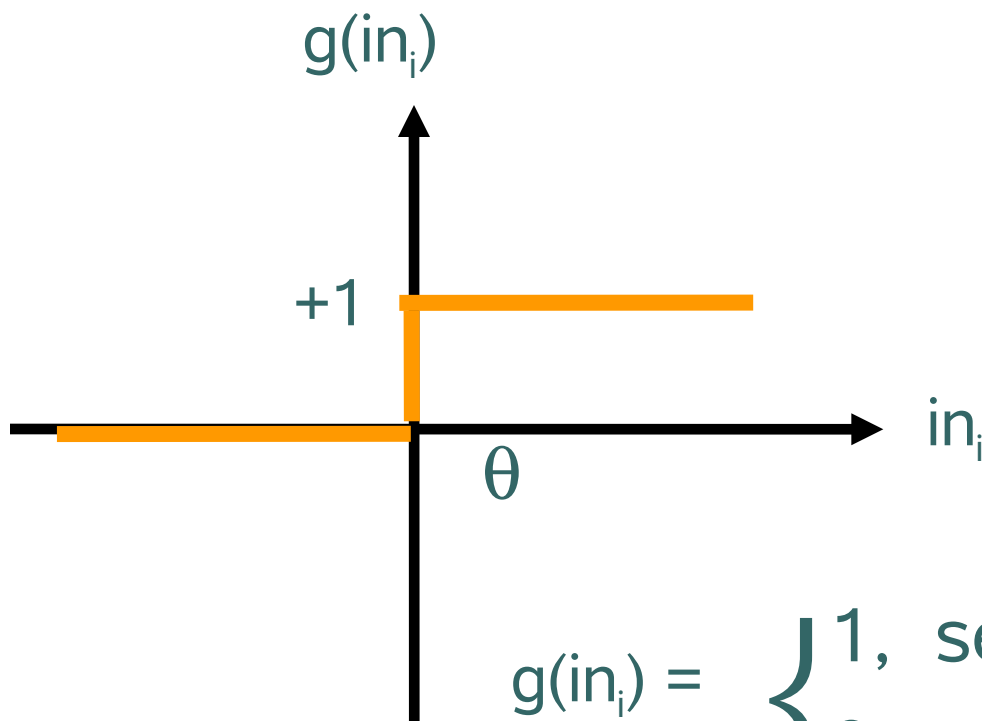
- A função de ativação é projetada com dois desejos
 - a unidade deve ser **ativa** (perto de 1+) quando a entrada correta for dada e **inativa** (perto de 0) quando uma saída errada é dada
 - a ativação necessita ser não linear, caso contrário a rede neural completa desmoronaria em uma simples função linear
- Escolha para a função de ativação **g** são mostradas a seguir:

Função de ativação

Função linear



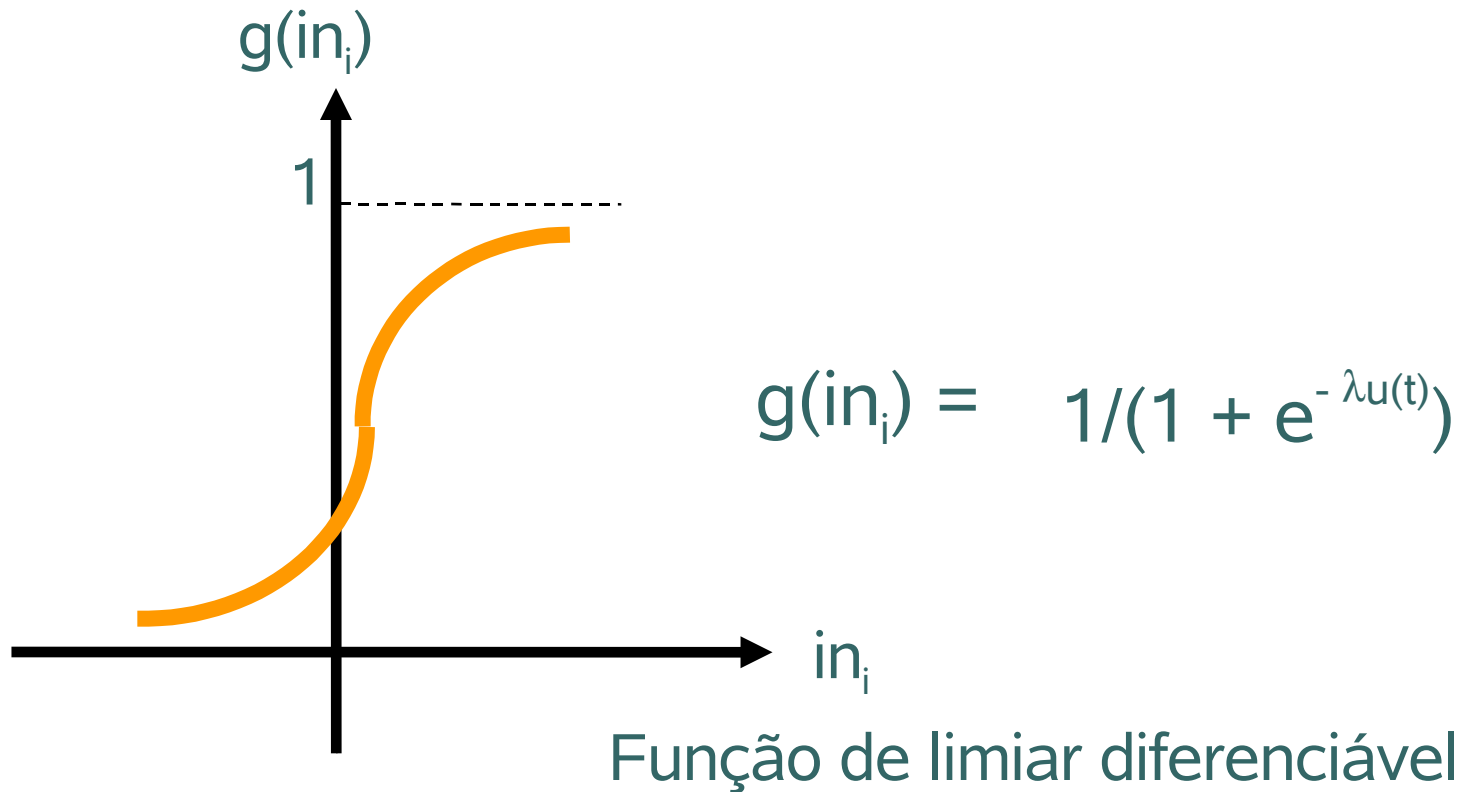
Função de ativação Função limiar (*threshold*)



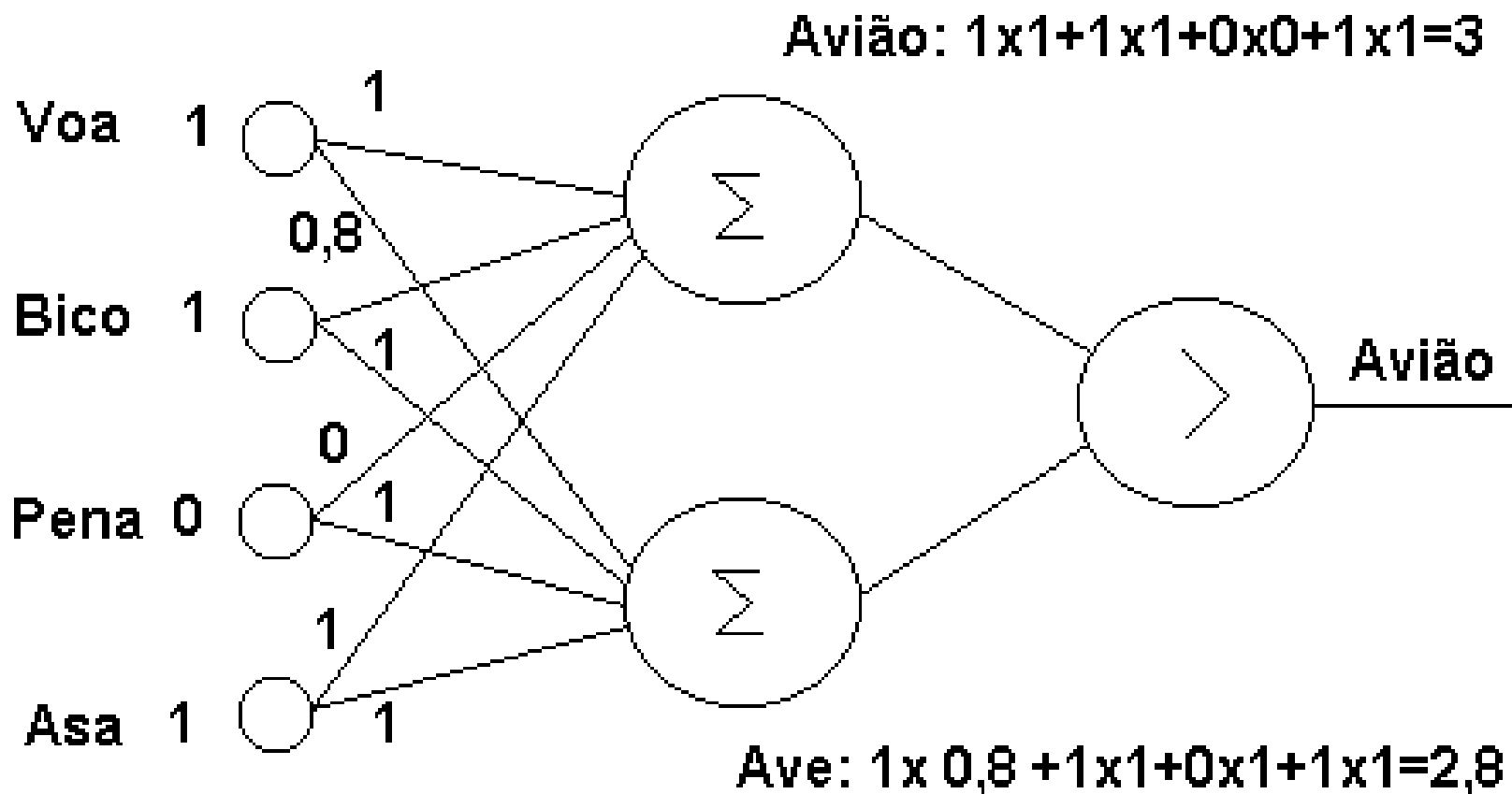
$$g(in_i) = \begin{cases} 1, & \text{se } in_i \geq \theta \\ 0, & \text{se } in_i < \theta \end{cases}$$

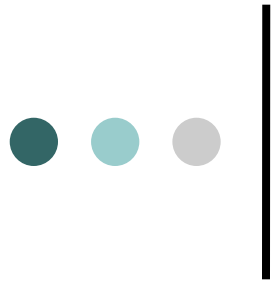
Função de ativação

Função sigmoidal logística (bipolar)



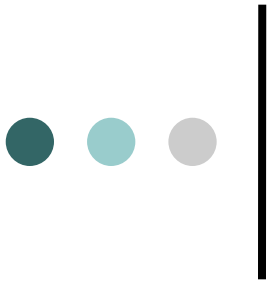
Exemplo de funcionamento





Características das RNAs

- O comportamento inteligente vem das interações entre as unidades de processamento da rede
- Elas aprendem através de exemplos
- Processo de treinamento a partir dos casos reais
- Capaz de extrair regras básicas a partir de dados reais, diferindo da computação programada



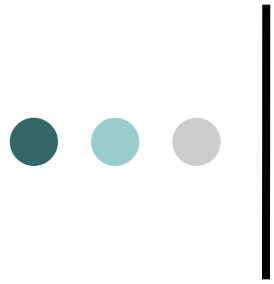
Aprendizagem

- Ajustes de seus pesos
- Aprendizado ocorre quando atinge uma solução generalizada para uma classe de problemas
- 50 a 90% do total de dados são escolhidos aleatoriamente afim que a rede aprenda
 - o restante só é apresentado na fase de testes



Processamento Neural

- O processamento de uma Rede Neural pode ser dividido em duas fases:
 - processo de cálculo da saída da rede, dado um certo padrão de entrada
 - **Recuperação da Informação**
 - processo de atualização dos pesos sinápticos para a aquisição do conhecimento
 - **Aquisição da Informação**



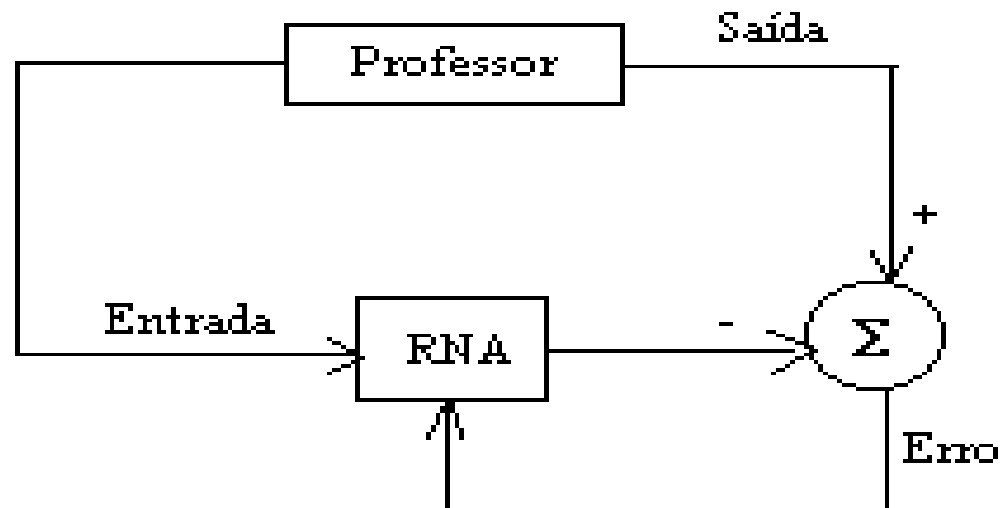
Aprendizagem das RNAs

- A maioria dos modelos de redes neurais possui alguma **regra de treinamento**
 - onde os **pesos** de suas conexões são **ajustados** de acordo com os padrões apresentados
 - elas **aprendem através de exemplos**

Formas de aprendizado: RNAs

○ **Aprendizagem supervisionada**

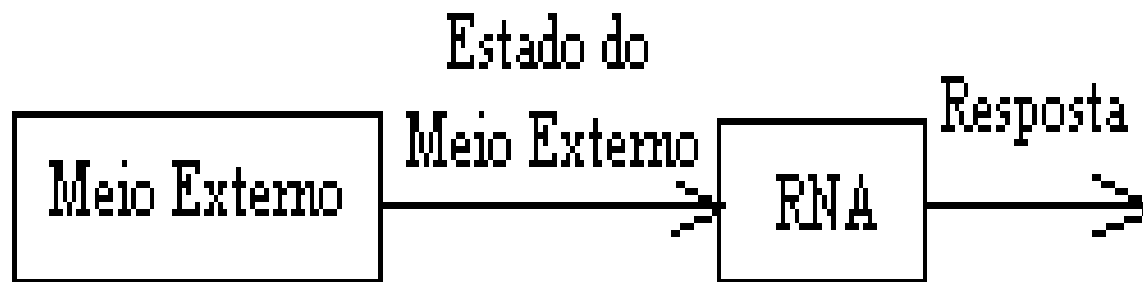
- a saída desejada é conhecida e informada para que a rede compare com a saída processada
- se houver erro, a rede tenta corrigir este erro até que a mesma forneça uma saída igual a saída desejada



Formas de aprendizado: RNAs

○ **Aprendizagem não supervisionada**

- a saída desejada é obtida através de entradas repetitivas até a rede reter o conhecimento
- não existe saída informada para comparação
 - deve existir redundâncias nos dados para a rede encontrar padrões ou características dos dados

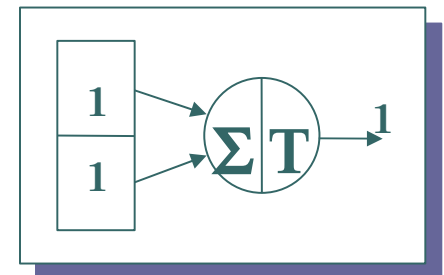
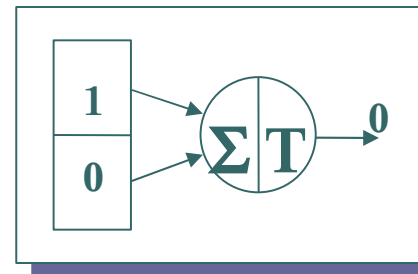
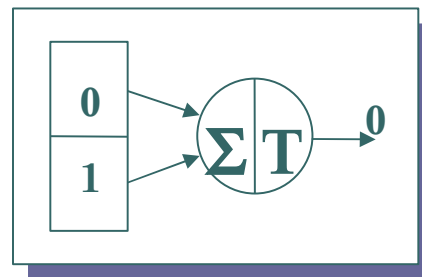
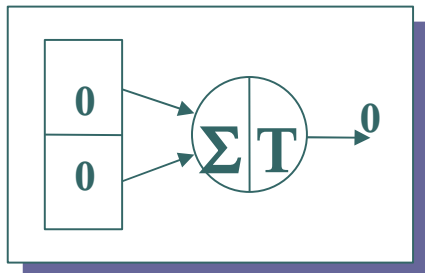


Exemplo: Aprendizagem

Implementação da porta lógica AND

- Certificar-se de que todas as respostas estão corretas para cada conjunto de entradas pela tabela-verdade
- A RNA possui um único neurônio de duas entradas e uma saída

Tabela Verdade - AND		
Entrada 1	Entrada 2	Saída
1	1	1
1	0	0
0	1	0
0	0	0



Exemplo: Aprendizagem

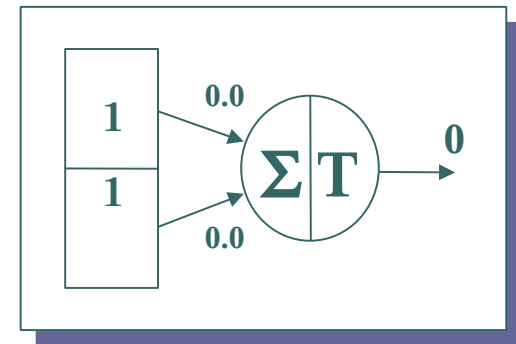
Implementação da porta lógica AND

- Para treinar a rede vamos seguir alguns passos:

para as entradas [1, 1]

→

pesos iniciais [0, 0]



$$\theta = 0,5$$

Função Soma

$$\sum_{i=1}^n x_i w_i \geq \theta$$

$$y = 1, \text{ se } s > \theta$$

$$y = 0, \text{ se } s \leq \theta$$

limiar

Exemplo: Aprendizagem

Implementação da porta lógica AND

- Para treinar a rede vamos seguir alguns passos:

- Passo 1: Aplicar a função *Soma*

$$\sum_{i=1}^n x_i w_i \geq \theta$$

$$\text{Soma} = 1*0 + 1*0 = 0$$

- Passo 2: Aplicar a função de *Transferência*

$$\text{Soma} \leq 0,5 \rightarrow y = 0$$

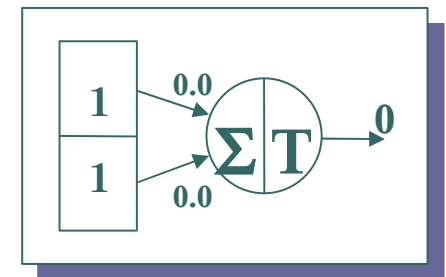
$$\text{Soma} > 0,5 \rightarrow y = 1$$

- Transferido 0 para a saída.
Erro!!!!

$$\theta = 0,5$$

Função Soma

$$\sum_{i=1}^n x_i w_i \geq \theta$$



$$y = 1, \text{ se } s > \theta$$

$$y = 0, \text{ se } s \leq \theta$$

limiar



Exemplo: Aprendizagem

Implementação da porta lógica AND

- Passo 3: Ajuste do peso

Equação do erro:

$$E = S_d - S_o$$

onde

S_d é a saída desejada

S_o é a saída obtida

Fator de correção:

$$F = c * x * E$$

onde

$c = 0,5$ (constante)

x é a entrada

E é o erro

Equação do ajuste:

$$w_{novo} = w + F$$

Exemplo: Aprendizagem

Implementação da porta lógica AND

○ Passo 3: Ajuste do peso

Calcular o erro: $E = 1 - 0 = 1$

Calcular o fator de correção:

$$F_1 = c * E * x_1$$

$$F_2 = c * E * x_2$$

$$F_1 = 0,5 * 1 * 1$$

$$F_2 = 0,5 * 1 * 1$$

$$F_1 = 0,5$$

$$F_2 = 0,5$$

Calcular o novo peso:

$$w_{1novo} = w_1 + F_1$$

$$w_{2novo} = w_1 + F_2$$

$$w_{1novo} = 0 + 0,5$$

$$w_{2novo} = 0 + 0,5$$

$$w_{1novo} = 0,5$$

$$w_{2novo} = 0,5$$

Equação do erro:

$$E = S_d - S_o$$

onde

S_d é a saída desejada

S_o é a saída obtida

Fator de correção:

$$F = c * x * E$$

onde

$c = 0,5$ (constante)

x é a entrada

E é o erro

Equação do ajuste:

$$w_{novo} = w + F$$

Exemplo: Aprendizagem

Implementação da porta lógica AND

- Para treinar a rede vamos seguir alguns passos:
para as entradas [1, 1] ...

pesos iniciais [0,5, 0,5]

- Passo 1: Aplicar a função *Soma*

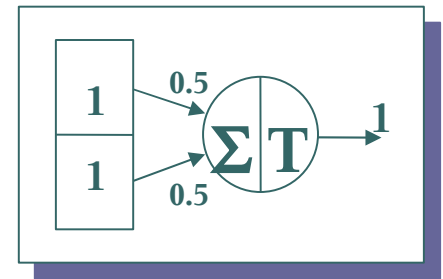
$$Soma = 1 * 0,5 + 1 * 0,5 = 1$$

- Passo 2: Aplicar a função de *Transferência*

$$Soma \leq 0,5 \rightarrow y = 0$$

$$Soma > 0,5 \rightarrow y = 1$$

- Transferido 1 para a saída. **Correto!!!!**





EXERCÍCIO: Aprendizagem Implementação da porta lógica AND

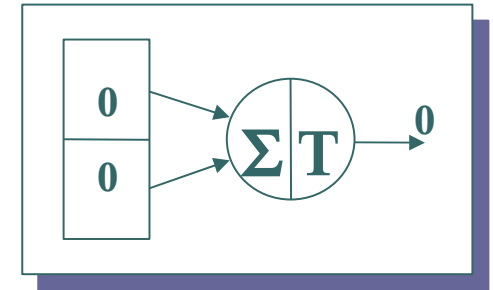
EXERCÍCIO

- Continuar treinando a rede
 - para as entradas $[0, 0]$ e pesos $[0,5, 0,5]$
- Testar a rede
 - Para as entradas $[0, 1]$ e $[1, 0]$

EXERCÍCIO: Aprendizagem

Implementação da porta lógica AND

- Para treinar a rede vamos seguir alguns passos:
para as entradas [0, 0] e
pesos [0,5, 0,5]



- Passo 1: Aplicar a função *Soma*

$$Soma = 0 * 0,5 + 0 * 0,5 = 0$$

- Passo 2: Aplicar a função de *Transferência*

$$Soma \leq 0,5 \rightarrow y = 0$$

$$Soma > 0,5 \rightarrow y = 1$$

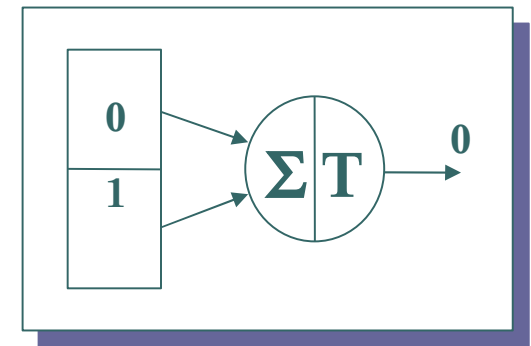
- Transferido 0 para a saída. **Correto!!!!**

EXERCÍCIO: Aprendizagem

Implementação da porta lógica AND

- Testar a rede: para as entradas [0, 1] e pesos [0,5, 0,5]

- Passo 1: Aplicar a função *Soma*
 $Soma = 0*0,5 + 1*0,5 = 0,5$



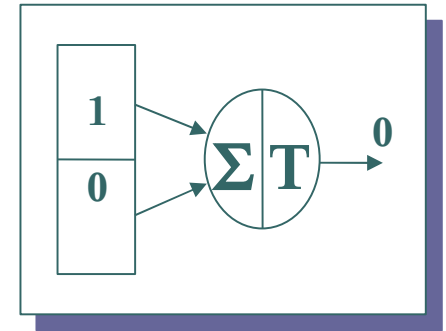
- Passo 2: Aplicar a função de *Transferência*
 $Soma \leq 0,5 \rightarrow y = 0$
 $Soma > 0,5 \rightarrow y = 1$
- Transferido 0 para a saída. **Correto!!!!**

EXERCÍCIO: Aprendizagem

Implementação da porta lógica AND

- Testar a rede: para as entradas [1, 0] e pesos [0,5, 0,5]

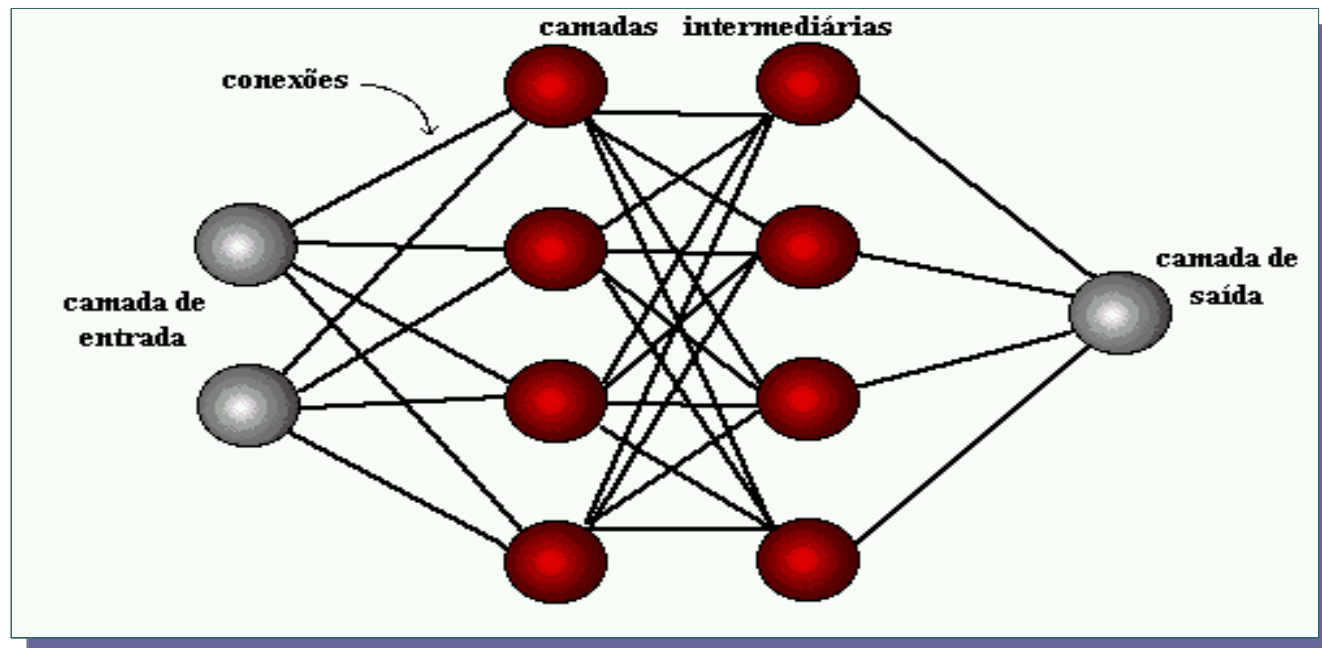
- Passo 1: Aplicar a função *Soma*
 $Soma = 1*0,5 + 0*0,5 = 0,5$



- Passo 2: Aplicar a função de *Transferência*
 $Soma \leq 0,5 \rightarrow y = 0$
 $Soma > 0,5 \rightarrow y = 1$
- Transferido 0 para a saída. **Correto!!!!**

Arquiteturas de RNA

- Arquiteturas neurais são tipicamente organizadas em camadas, com unidades que podem estar conectadas às unidades da camada posterior



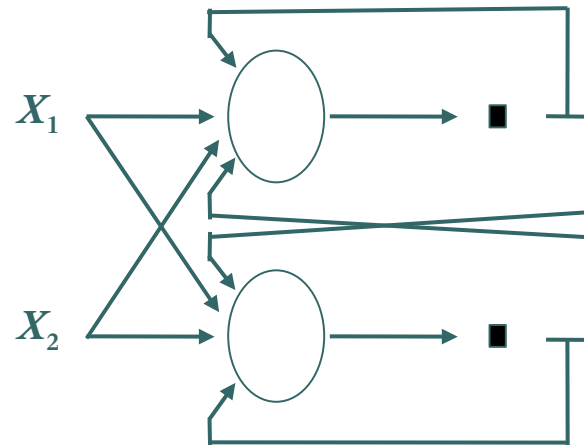
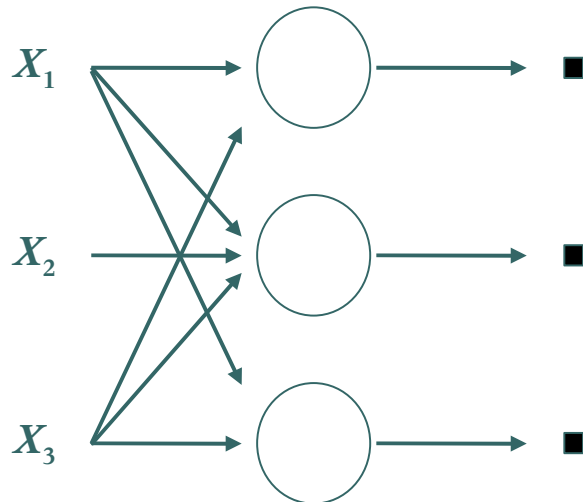


Arquiteturas de RNA

- Usualmente as camadas são classificadas em três grupos:
 - **Camada de Entrada:** onde os padrões são apresentados à rede
 - **Camadas Intermediárias ou Escondidas:** onde é feita a maior parte do processamento, através das conexões ponderadas
 - podem ser consideradas como extratoras de características
 - **Camada de Saída:** onde o resultado final é concluído e apresentado

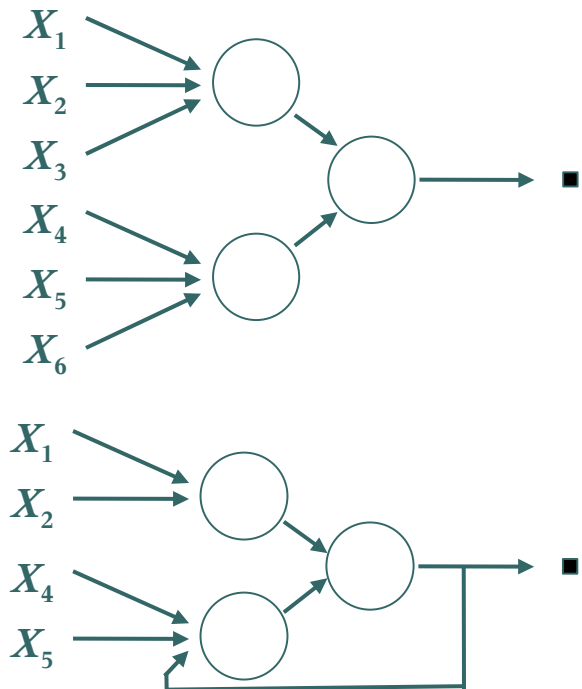
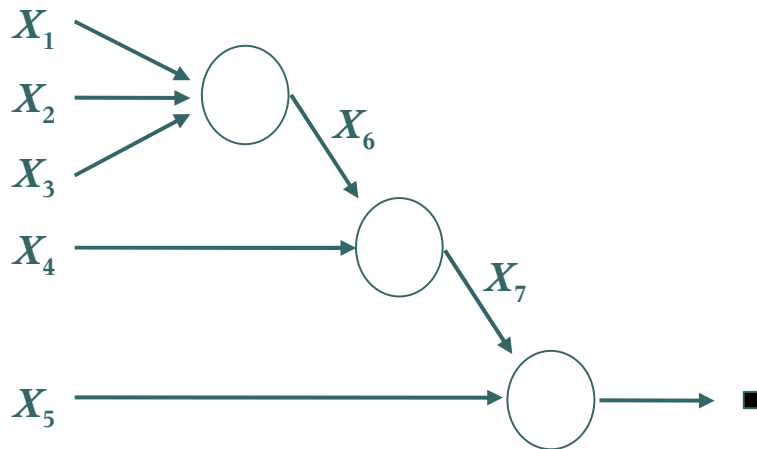
Arquiteturas de RNA

- Número de camadas
 - redes de camada única
 - só existe um nó entre qualquer entrada e qualquer saída da rede



Arquiteturas de RNA

- Número de camadas
 - redes de múltiplas camadas
 - existe mais de um neurônio entre alguma entrada e alguma saída da rede

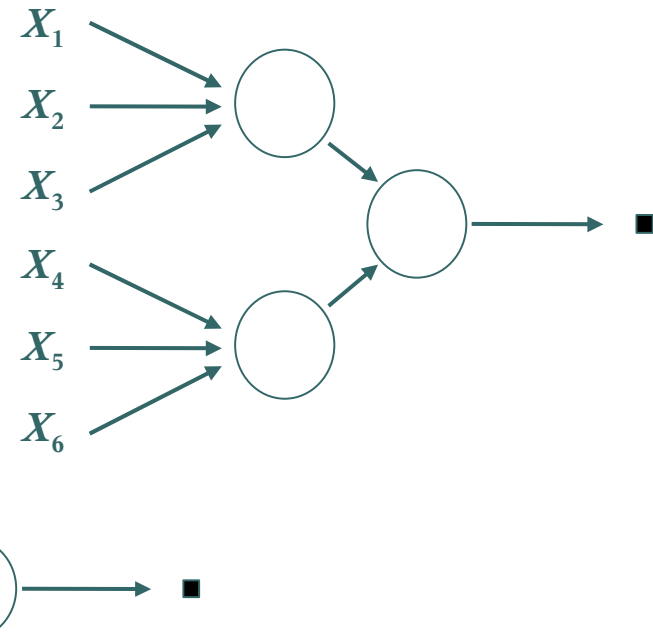
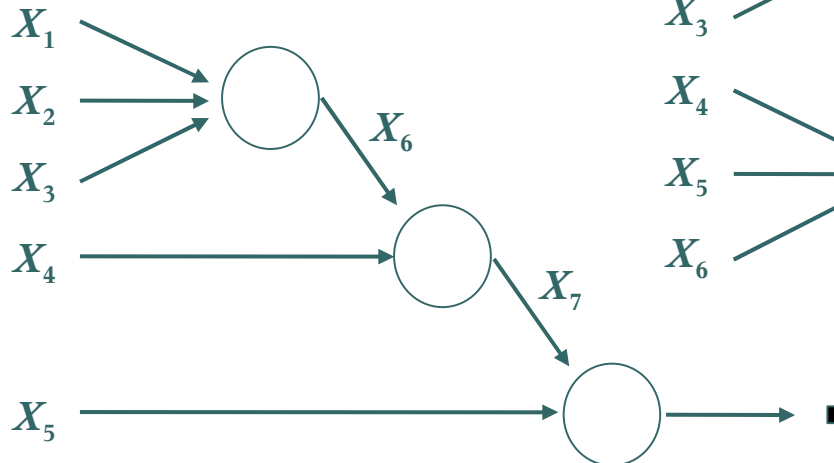
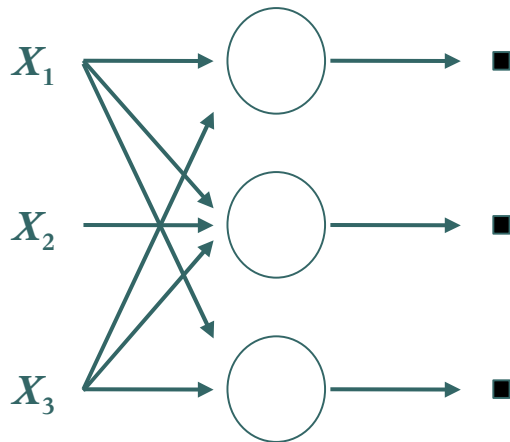


Arquiteturas de RNA

Tipos de conexões dos nós

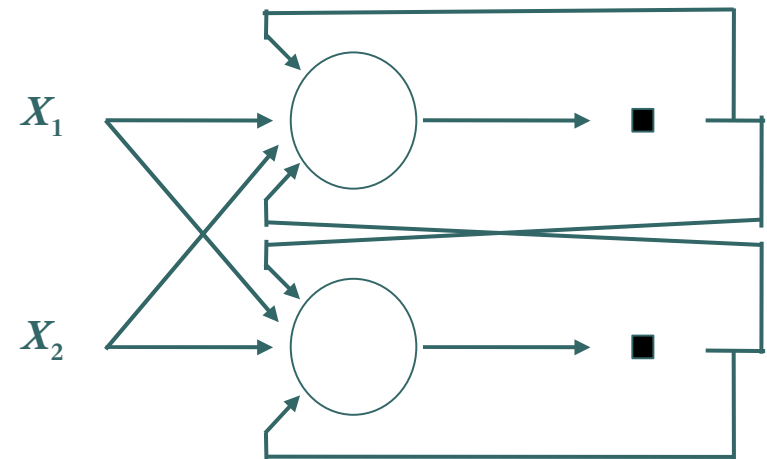
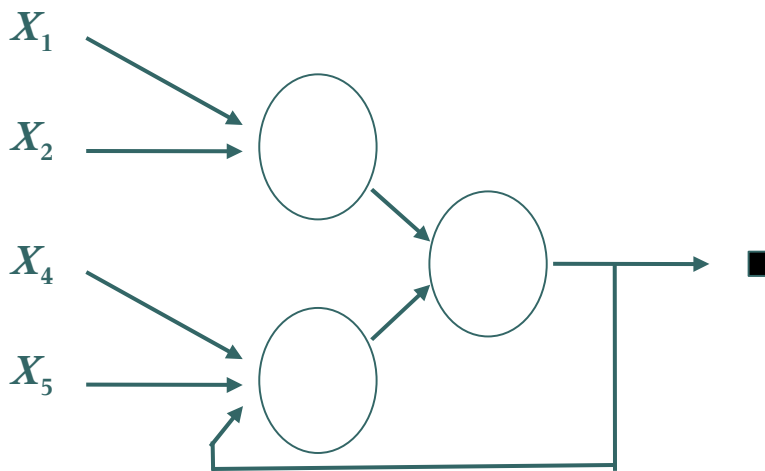
feedforward ou acíclica

- a saída do neurônio na ***i*-ésima camada** da rede não pode ser usada como entrada de nodos em **camadas de índice menor ou igual a *i***



Arquiteturas de RNA

- Tipos de conexões dos nós
 - **feedback ou cíclica ou recorrente**
 - a saída do neurônio na **i -ésima camada** da rede é usada como entrada de nodos em camadas de índice **menor ou igual a i**





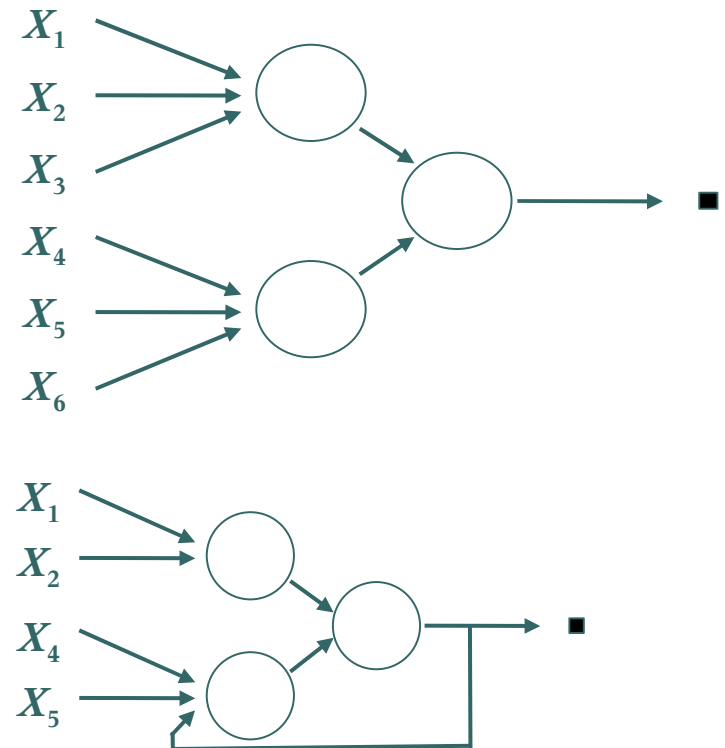
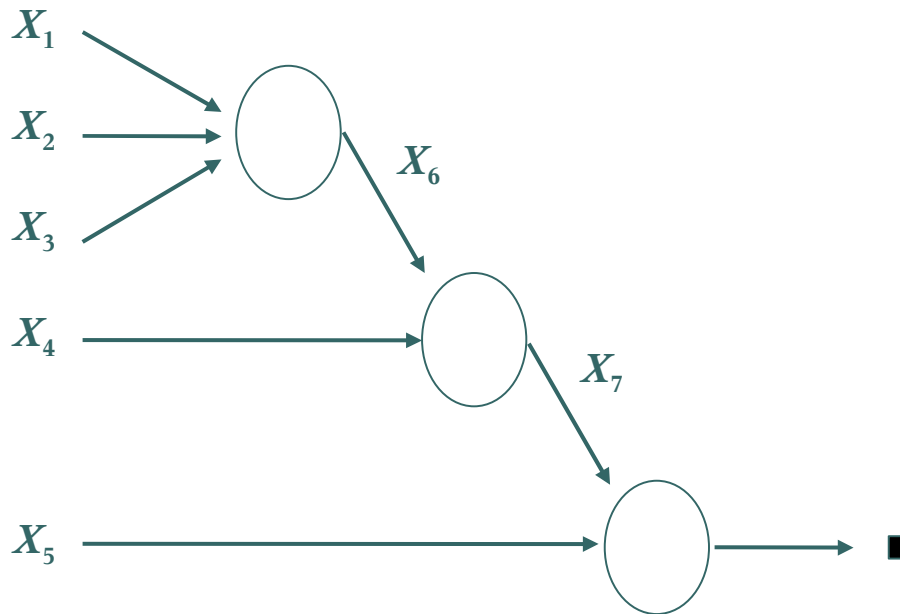
Arquiteturas de RNA

- **feedback ou cíclica ou recorrente**

- o nível de ativação da rede forma um sistema dinâmico que pode alcançar um estado estável ou exibir oscilações ou comportamento caótico
- além disso, a resposta da rede para uma dada entrada depende de seu estado inicial, que pode depender das entradas anteriores
- por isso, elas podem suportar **memória** (em partes)
 - elas são mais parecidas com o cérebro humano (diferente das acíclicas), mas são mais complicadas

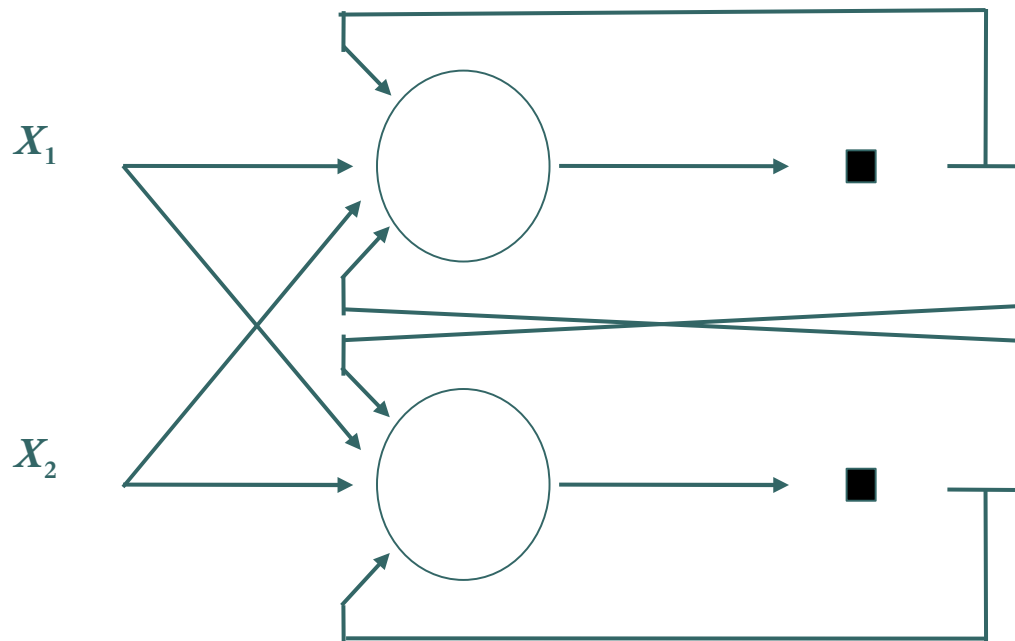
Arquiteturas de RNA

- Conectividade
 - fracamente (ou parcialmente) conectada



Arquiteturas de RNA

- Conectividade
 - completamente conectada





Arquiteturas de RNA

- Uma rede neural é caracterizada, principalmente
 - pela sua **topologia**
 - *feedforward, feedback*
 - pelas **características dos nós**
 - booleano, fuzzy, híbrido
 - pelas **regras de treinamento**
 - Hebb, *backpropagation*, ...



Modelos de RNA

- *Perceptrons* (acíclica)
- *Perceptrons* de Múltiplas Camadas (acíclica)
- Rede *Hopfield* (cíclica)
- ...



Aprendizagem em Redes Neurais Artificiais

- No aprendizado cognitivo, não se procura obter regras como na abordagem simbólica da IA
 - mas sim determinar a intensidade de conexões entre neurônios
- Em outras palavras, aprendizagem em RNA é o processo de modificar os valores de pesos e do limiar



Aprendizagem em Redes Neurais Artificiais

○ Definição

“Aprendizagem é o processo pelo qual os parâmetros de uma RNA são ajustados através de uma forma continuada de estímulo pelo ambiente no qual a rede está operando, sendo o tipo específico de aprendizagem realizada definido pela maneira particular como ocorrem os ajustes realizados nos parâmetros”



Aprendizagem em Redes Neurais Artificiais

○ Regras de aprendizagem em RNA

- estabelecer um **conjunto de pesos** para suas conexões
- ativar um conjunto de unidades que correspondam a um **padrão de entrada**
- observar o padrão para o qual a **rede converge** e em que se **estabiliza**



Aprendizagem em Redes Neurais Artificiais

○ Regras de aprendizagem em RNA

- se o padrão final não corresponder ao que se deseja associar como resposta ao de entrada, é preciso fazer ajustes nos pesos e ativar novamente o padrão de entrada
- por causa de sua **semelhança** com o aprendizado humano, esse processo de ajustes sucessivos das RNA é **chamado de aprendizagem**



Aprendizagem em Redes Neurais Artificiais

○ Regra de Hebb

- Desenvolvida por Donald Hebb em 1949
- **Princípio:**
 - a força da conexão entre dois neurônios é aumentada se os neurônios estão simultaneamente excitados

$$\Delta w_{ij} = \mu \cdot y_i \cdot x_j$$

- μ = taxa de aprendizado
- y_i e x_j = ativações das unidades y_i e x_j



Aprendizagem em Redes Neurais Artificiais

○ Lei de aprendizagem de Hebb

“Se um **neurônio A** é repetidamente estimulado por um outro **neurônio B**, ao mesmo tempo em que ele está ativo, ele ficará mais sensível ao estímulo de **B**, e a conexão sináptica de **B** para **A** será mais forte. Deste modo, **B** achará mais fácil estimular **A** para produzir uma saída.”

- O conhecimento fica retido nos neurônios
- Para reter conhecimento, toda RNA passa por um processo de aprendizagem



Aprendizagem em Redes Neurais Artificiais

○ Regra Delta (Widrow-Hoff)

- A regra Delta é uma variação da regra de Hebb
- Foi desenvolvida por Bernard Widrow e Ted Hoff (1982), conhecida também como *least mean square* (LMS), por minimizar o erro médio quadrático



Gradiente Descendente e a Regra Delta

- Uso do gradiente descendente para reduzir o erro
- Uso do gradiente descendente para buscar o vetor de pesos que melhor se ajuste ao conjunto de treinamento
 - independentemente do conjunto de treinamento ser linearmente separável



Função de Custo

- Especifique uma medida de erro (função de custo) para o treinamento
- $E(w) = 1/2 \sum_{d \in D} (t_d - o_d)^2$, em que
 - D é o conjunto de treinamento
 - t_d é a saída desejada para o exemplo d
 - o_d é a saída para o exemplo d



Minimização do custo

- A função E é uma medida objetiva do erro preditivo para uma escolha específica de vetor de pesos
- **Objetivo:** encontrar um vetor \mathbf{w} que minimize E
- **Solução:** usar técnicas de gradiente descendente



Minimização do custo

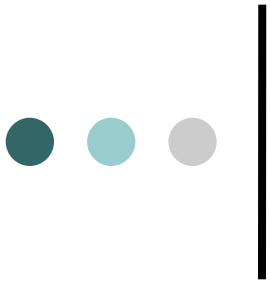
1. Escolha valores iniciais arbitrários para os pesos
2. Calcule o gradiente da função de custo com respeito a cada peso
3. Mude os pesos tal que haja um deslocamento pequeno na direção de $-G$
-G => Maior taxa de diminuição do erro
5. Repita passos 2 e 3 até que erro se aproxime de zero

Como este algoritmo funciona?



Algoritmo de Widrow e Hoff

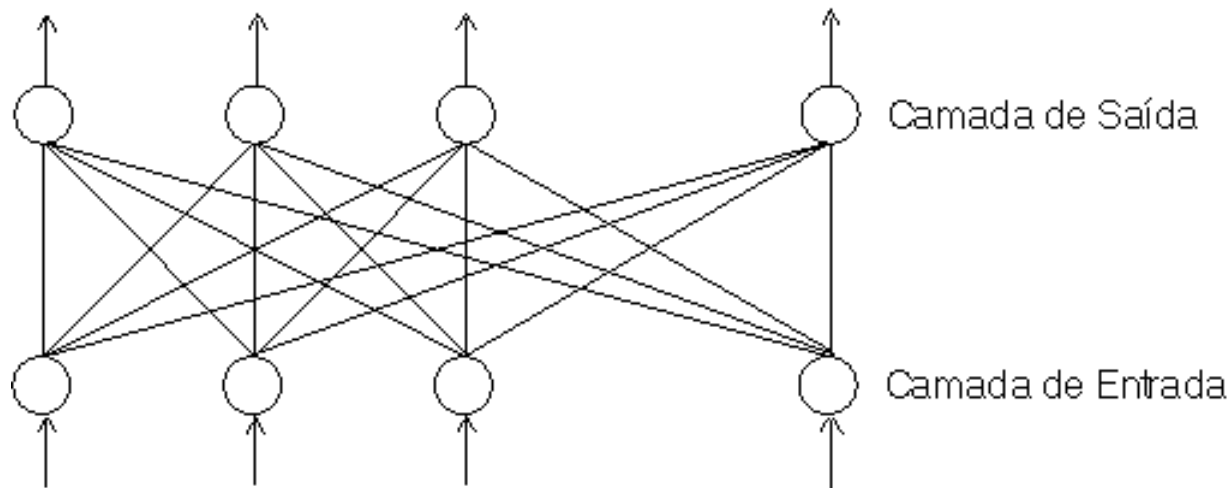
- Inicializar η e o vetor de pesos w
- Repetir
- Inicializar Δw_i com zero
- Para cada par do conjunto de treinamento (x, t)
 - calcule a saída o
 - para cada peso w_i
 - calcule $\Delta w_i \leftarrow \Delta w_i + \eta (t - o)x_i$
- Para cada peso w_i
 - $w_i \leftarrow w_i + \Delta w_i$
- Até que uma condição de término seja satisfeita



Perceptrons

- Desenvolvido por Rosenblat (1958)
- Utiliza modelo de McCulloch-Pitts
- Todas as entradas conectadas diretamente nas saídas
 - Rede neural camada simples
 - *Perceptron*

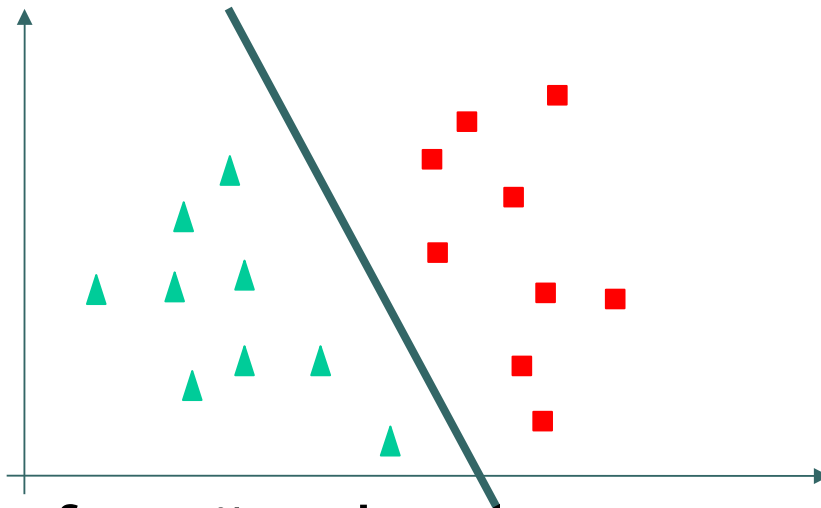
Perceptrons



- Estrutura mais simples de RNAs

Perceptrons

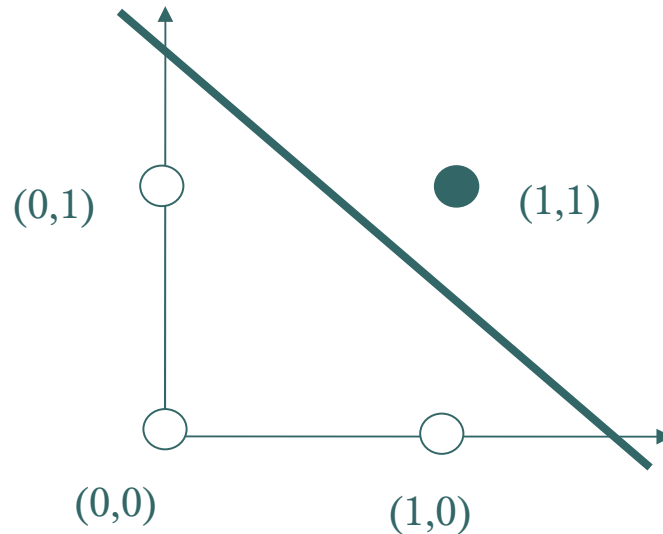
- Rede mais simples que pode ser utilizada para classificação de padrões linearmente separáveis



- Há muitas funções booleanas que o limiar (*threshold*) *perceptron* não pode representar

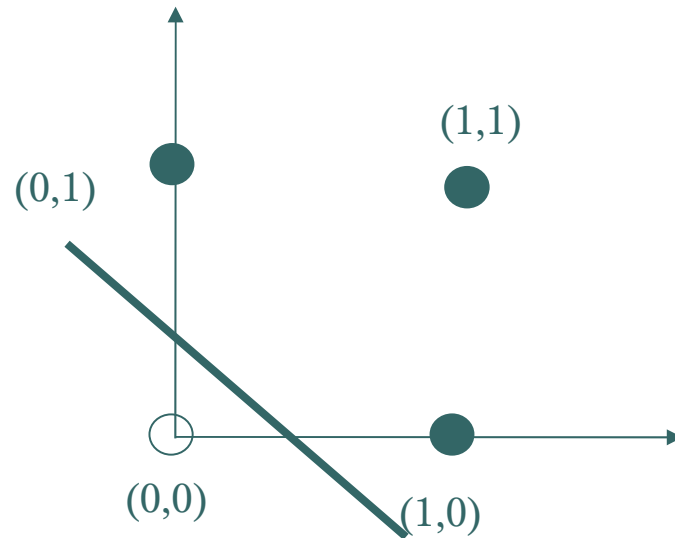
Portas de limiar (*threshold*)

- Linearmente separáveis – I_1 AND I_2



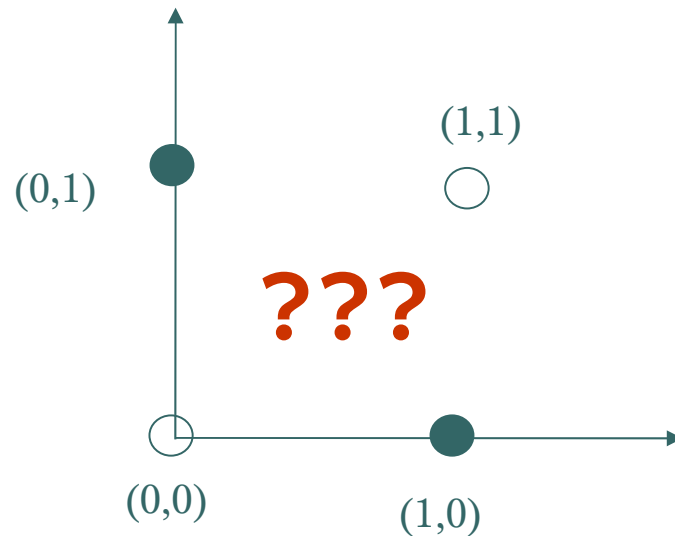
Portas de limiar (*threshold*)

- Linearmente separáveis – I_1 OR I_2



Portas de limiar (*threshold*)

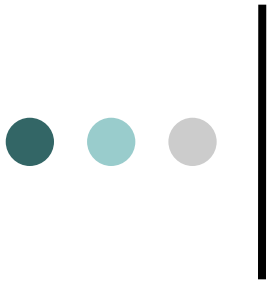
- Linearmente separáveis – I_1 XOR I_2





Perceptrons

- “Há um algoritmo de aprendizado simples que ajustará (encontrará) um limiar para qualquer conjunto de treinamento linearmente separável”
- A idéia deste algoritmo, e da maioria dos algoritmos de aprendizado de redes neurais, é ajustar os pesos da rede para minimizar a medida de erro em um conjunto de treinamento



Perceptrons

- O aprendizado é formulado como uma busca otimizada em um espaço de busca
- O algoritmo de aprendizagem do *perceptron* utiliza o algoritmo de correções de erros como base



Perceptrons

- Treinamento Supervisionado

- Correção de erro: $\Delta W_{ij} = \mu * e * x_i$

∀ μ = taxa de aprendizagem

- x_i = valor de entrada
- $e = (d_j - y_i)$ = erro (valor calculado – valor desejado)

- Teorema da convergência:

“Se é possível classificar um conjunto de entradas, uma rede *Perceptron* fará a classificação”



Perceptrons

○ **Treinamento:** Algoritmo

Iniciar todas as conexões com $w_i = 0$ (ou aleatórios)

Repita

Para cada padrão de treinamento (X, d)
faça

Calcular a saída y

Se $(d \neq y)$

então atualizar pesos

$$w_i(t+1) = w_i + \eta * e * x_i$$

até o erro ser aceitável



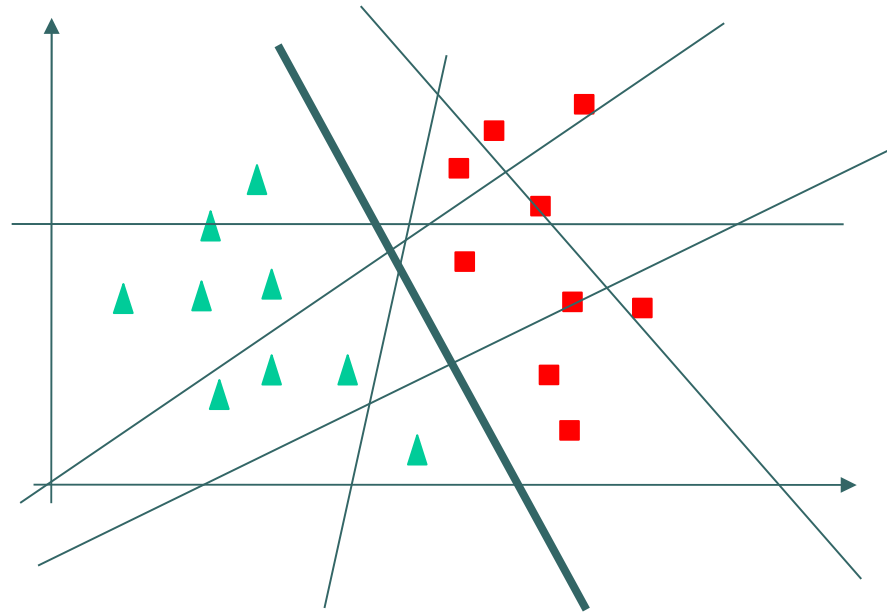
Algoritmo de aprendizado do *Perceptron*

- Teorema da convergência (Minsky e Papert, 1969)
 - O algoritmo converge dentro de um número finito de passos para um vetor de pesos que classifica corretamente todo o conjunto de treinamento
 - dado que o conjunto de treinamento é linearmente separável

Perceptrons

- **Treinamento**

- Algoritmo:





Perceptrons

- Algoritmo de teste:

Para cada padrão de 1 a p
faça

Apresentar X_p à entrada da rede

Calcular a saída y

Se $y \geq \theta$

então $X_p \in \text{Classe 1}$

senão $X_p \in \text{Classe 2}$

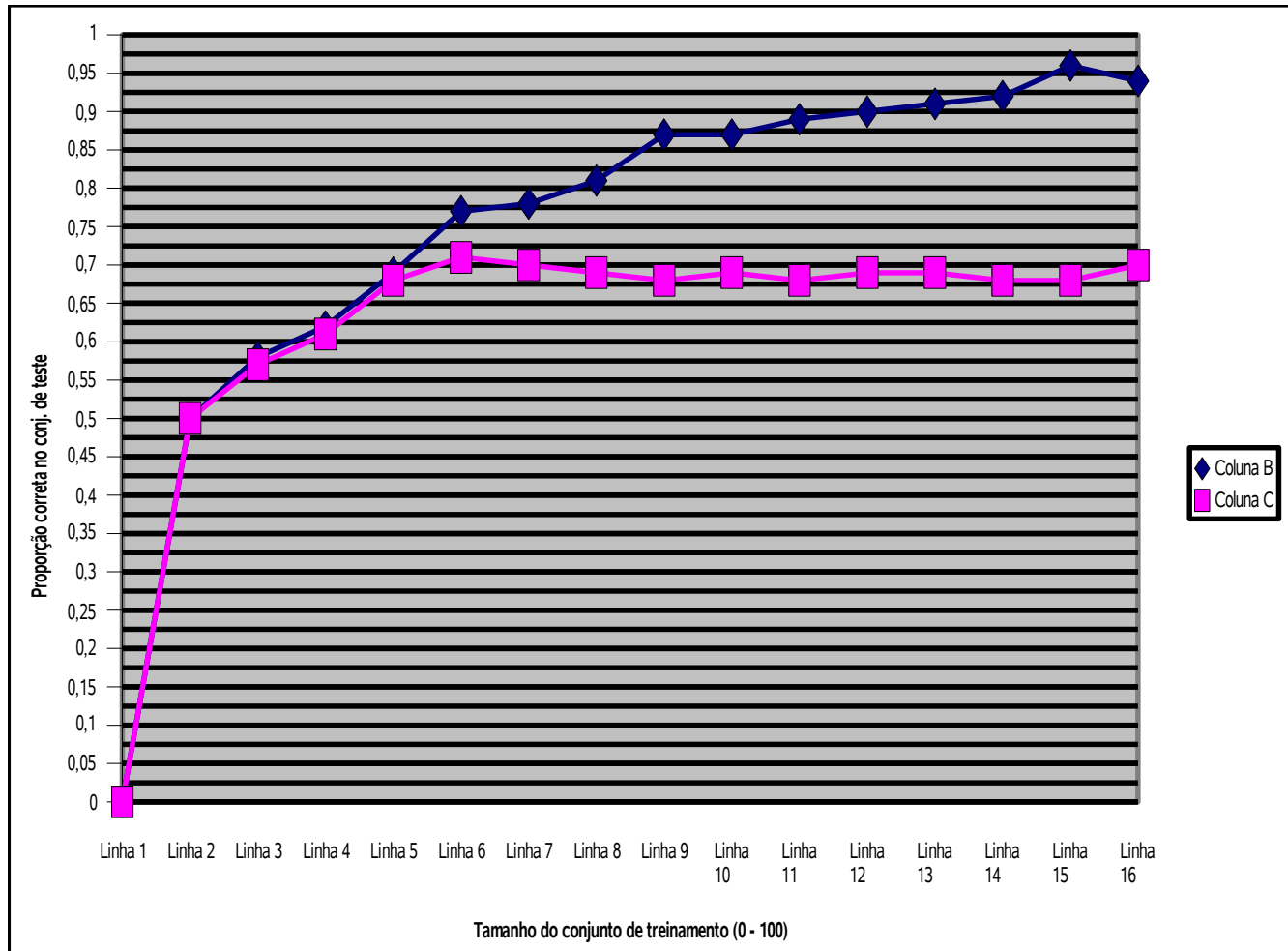


Perceptrons

- Apesar de causar grande euforia na comunidade científica, não teve vida longa
 - sofreu duras críticas de Minsky e Papert sobre sua capacidade computacional
 - causando grande desinteresse na área na década de 70 e início dos anos 80
- Esta visão pessimista sobre a capacidade do *perceptron* e das RNAs mudou devido às descrições de **Hopfield em 1982** e do algoritmo *back-propagation* (*retropropagação*)
 - a área ganhou novo impulso

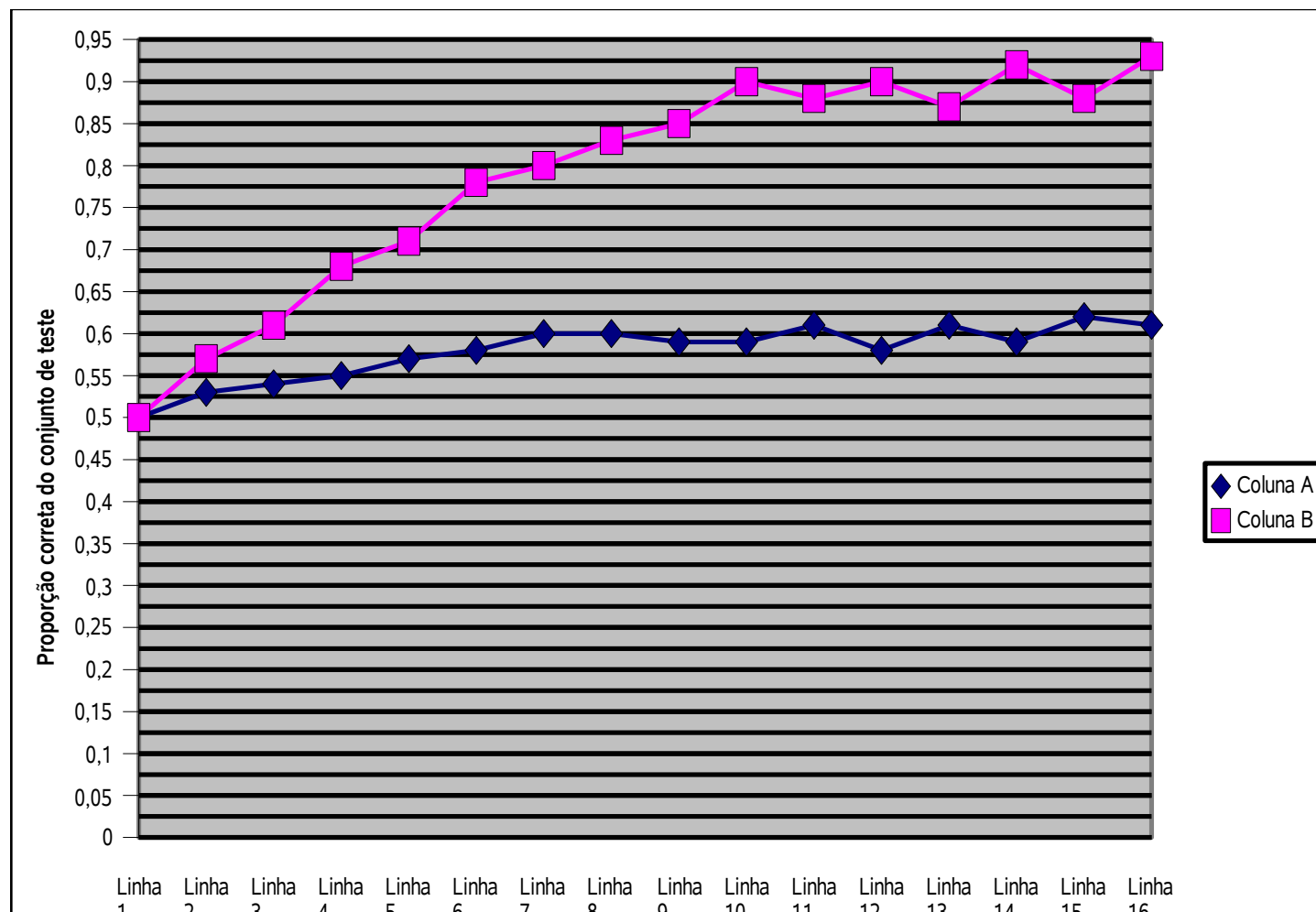
Perceptrons

- Função linearmente separável com 11 entradas booleanas



Perceptrons

Exemplo do restaurante, não é linearmente separável





Aprendizagem em RNAs

Algoritmo de retropropagação

- Desenvolvido por Paul Werbos (1974) e redescoberto independentemente por Parker (1982) e Rumelhart (1986)
- Aplicado para RNA cíclica (**feedforward**) com uma ou mais camadas intermediárias
- Utiliza um método do gradiente descendente por correção de erro: o algoritmo de codificação executa um mapeamento entrada-saída através da minimização de uma função de custo qualquer



Aprendizagem em RNAs

Algoritmo de retropropagação

- A função de custo é minimizada realizando-se iterativamente ajustes nos pesos sinápticos de acordo com o **erro quadrático** acumulado para todos os padrões do conjunto de treinamento
- Outras funções de custo podem ser utilizadas, mas independentemente disto, o procedimento de ajuste de pesos é realizado através do cálculo da mudança da função de custo com respeito à mudança em cada peso (método do delta)



Aprendizagem em RNAs

Algoritmo de retropropagação

- O processo de redução gradativa de erro que acompanha a minimização se denomina **convergência**
- A medida que a rede aprende, o valor do erro converge para um **valor estável**, normalmente **irredutível**
- O processo de aprendizagem prossegue até que algum critério seja estabelecido, como por exemplo, uma diferença sucessiva mínima entre erros calculados para cada iteração



Aprendizagem em RNAs

Algoritmo de retropropagação

- **Regra Delta generalizada**
 - **Cálculo do erro na saída**
 - *LMS*

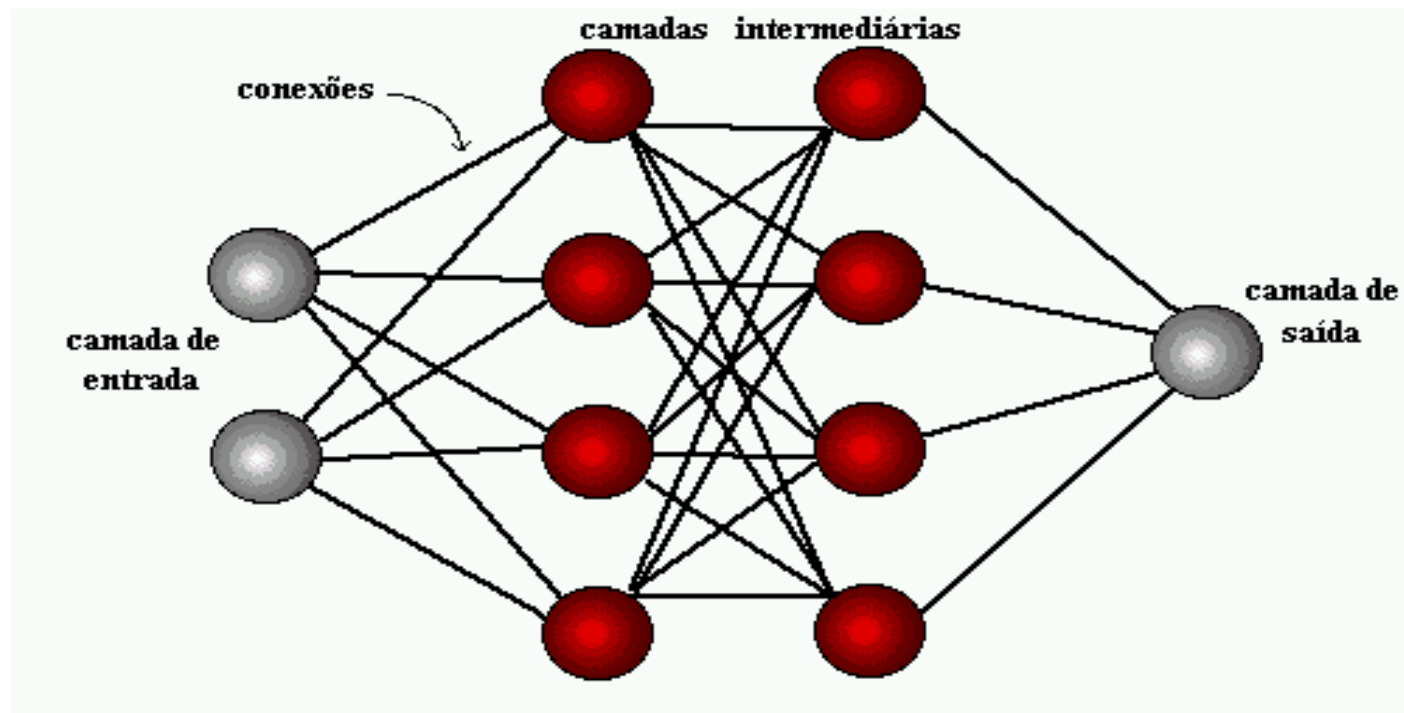
$$E = \frac{1}{2} \sum_i (d_i - y_i)^2$$

- *RMS*

$$E = \sqrt{\sum_i (d_i - y_i)^2}$$

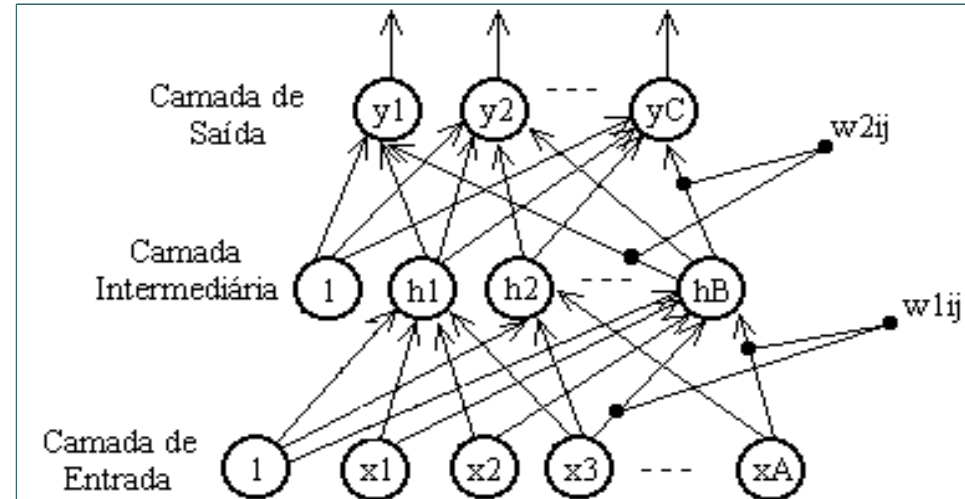
Perceptrons com múltiplas camadas

- Redes neurais acíclicas multicamadas
 - redes com unidades intermediárias ou escondidas



Perceptrons com múltiplas camadas

- Apresentam um poder computacional muito maior do que aquele apresentado pelas redes sem camadas intermediárias
- Tratam com dados que **não são linearmente separáveis**
- Teoricamente, redes com mais de uma camada intermediária podem implementar qualquer função, seja ela linearmente separável ou não





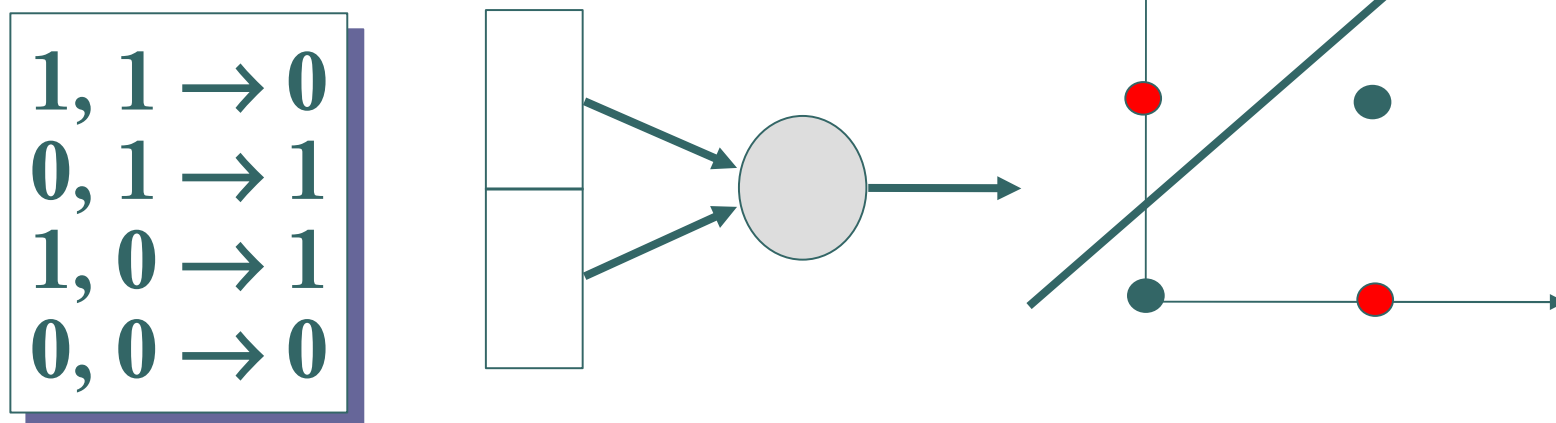
Perceptrons com múltiplas camadas

- A vantagem de adicionar camadas escondidas é que elas **aumentam o espaço de hipóteses** que a rede pode representar
 - exemplificando
 - cada unidade escondida é vista como um *perceptron* que representa uma função limiar no espaço de entradas
 - então, uma unidade de saída é vista como uma combinação de várias funções

Perceptrons com múltiplas camadas

○ Problema: *Perceptron*

- redes com uma camada resolvem apenas problemas linearmente separáveis

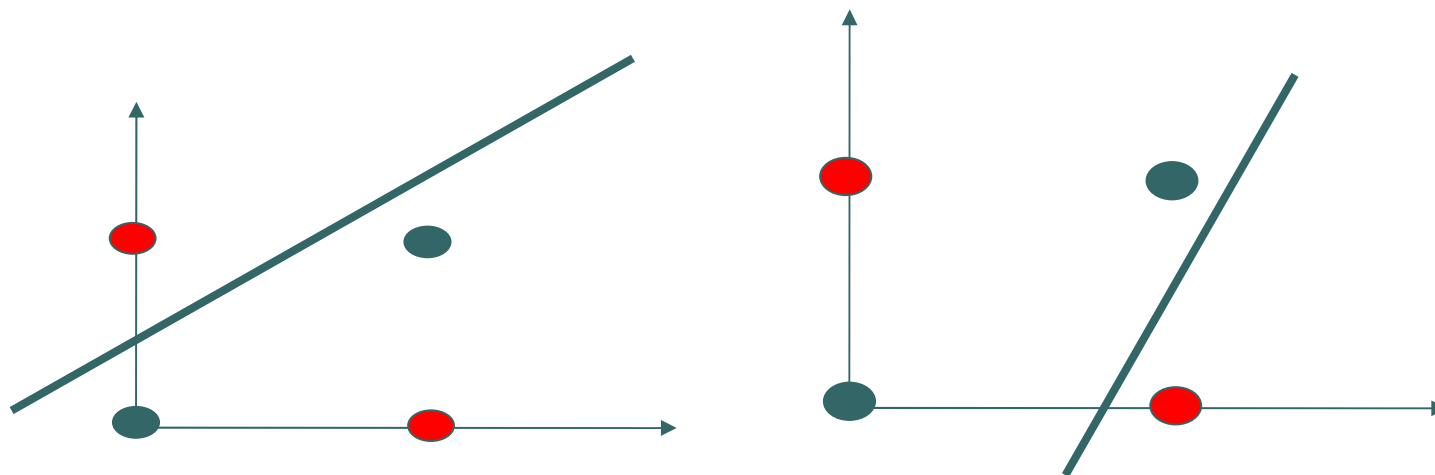


● ● ● | *Perceptrons* com múltiplas camadas

○ **Solução:** Utilizar mais de uma camada

● **Camada 1:**

- uma rede *Perceptron* para cada grupo de entradas linearmente separáveis

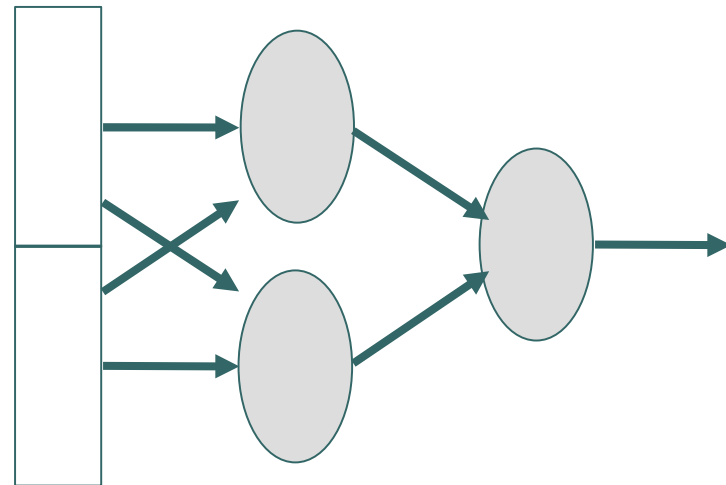
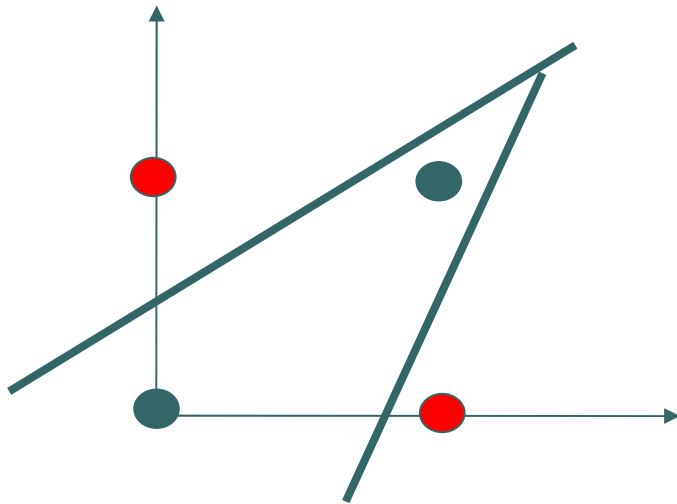


Perceptrons com múltiplas camadas

- **Solução:** Utilizar mais de uma camada

- **Camada 2:**

- uma rede combinando as saídas das redes da 1ª camada, produzindo a classificação final





Perceptrons com múltiplas camadas

○ Problema

- nem sempre se conhece a saída desejada dos nós da camada intermediária
- Suponha que queremos construir uma rede com camada escondida para o problema do restaurante
 - 10 atributos no exemplo, então 10 entradas na rede
 - **Quantas unidades escondidas?**
 - **foi usada 4**
 - o problema de escolher o número correto de unidades escondidas ainda não é bem entendido



Perceptrons com múltiplas camadas

○ **Algoritmo de aprendizado: retropropagação**

- **Número de neurônios da camada intermediária**

NH = Número de neurônios da camada intermediária

NS = Número de neurônios da camada de saída

NE = Número de neurônios da camada de entrada

$$NH = NS + \sqrt{NE}$$

$$NH = 1 + \sqrt{10} = 4$$

- **Outros:**

$$NH = NS \cdot NE$$

$$NH = 1 \cdot 10 = 10$$



Perceptrons com múltiplas camadas

- **Algoritmo de aprendizado: retropropagação**
 - antes de se iniciar o processo de aprendizado por retropropagação, é necessário que se tenha:
 - o conjunto de padrões de treinamento, entrada e saída desejada
 - um valor para a taxa de aprendizado
 - um critério que finalize o algoritmo (por nº de ciclos - ou épocas - ou por erro)



Perceptrons com múltiplas camadas

- **Algoritmo de aprendizado: retropropagação**
 - antes de se iniciar o processo de aprendizado por retropropagação, é necessário que se tenha:
 - uma metodologia para atualizar os pesos (Δw)
 - a função de transferência não-linear
 - valores de pesos iniciais



Perceptrons com múltiplas camadas

- **Algoritmo de aprendizado: retropropagação**
 - basicamente a rede aprende um **conjunto pré-definido de pares de exemplos** de entrada/saída em ciclos de propagação/adaptação
 - depois que um padrão de entrada foi aplicado como um estímulo aos elementos da primeira camada da rede, ele é **propagado por cada uma das outras camadas** até que a saída seja gerada



Perceptrons com múltiplas camadas

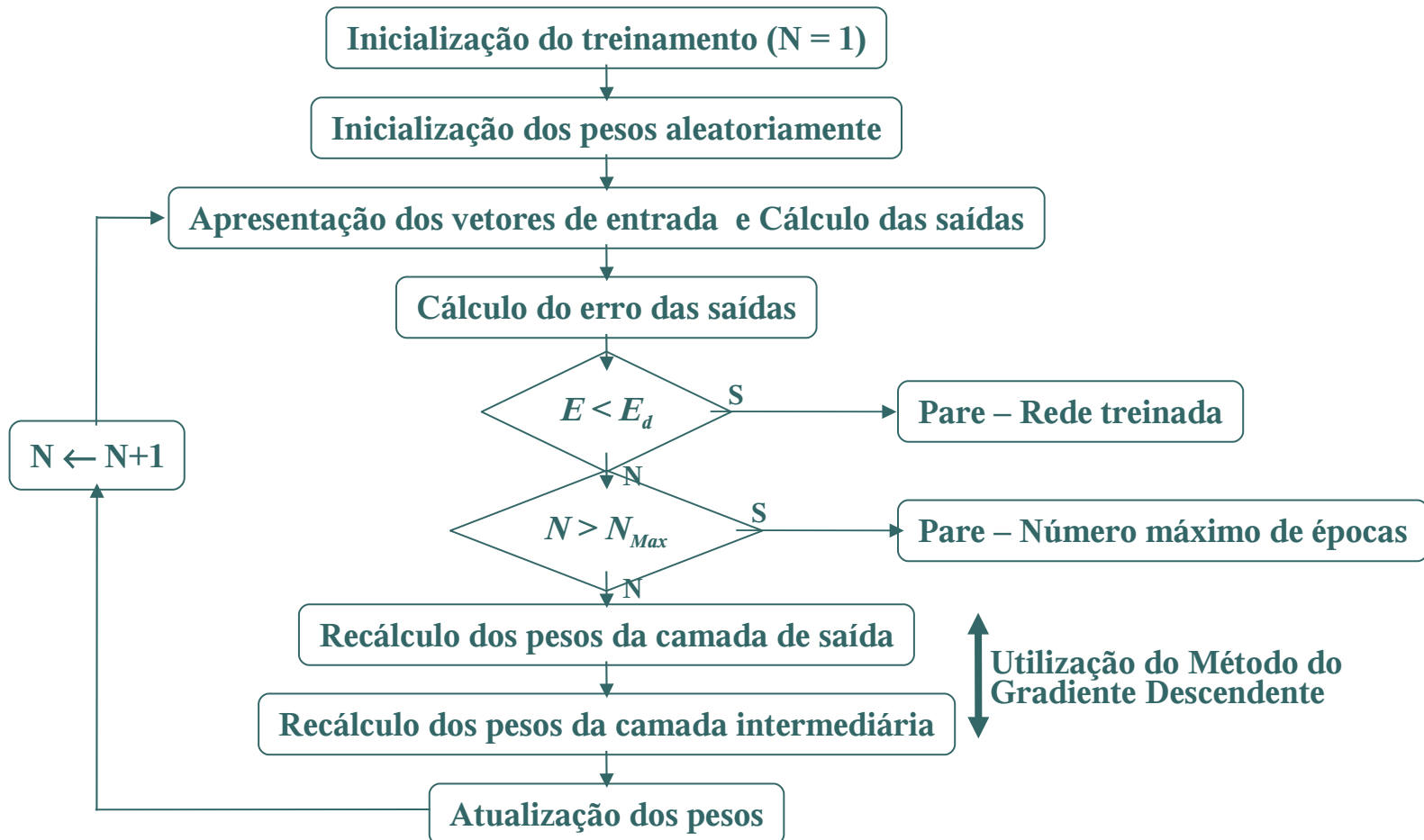
- **Algoritmo de aprendizado: retropropagação**
 - este padrão de saída é então comparado com a saída desejada e **um sinal de erro** é calculado para cada elemento de saída
 - o sinal de erro é então **retro-propagado da camada de saída para cada elemento da camada intermediária anterior** que contribui diretamente para a formação da saída
 - entretanto, cada elemento da camada intermediária recebe **apenas uma porção do sinal de erro total**, proporcional apenas à contribuição relativa de cada elemento na formação da saída original

● ● ● | *Perceptrons* com múltiplas camadas

- **Algoritmo de aprendizado: retropropagação**
 - este processo se repete, camada por camada, até que cada elemento da rede receba um sinal de erro que descreva sua **contribuição relativa para o erro total**
 - baseado no sinal de erro recebido, os pesos das conexões são, então, **atualizados para cada elemento** de modo a fazer a rede **convergir** para um estado que permita a codificação de todos os padrões do conjunto de treinamento

Perceptrons com múltiplas camadas

○ Algoritmo de aprendizado: retropropagação





Perceptrons com múltiplas camadas

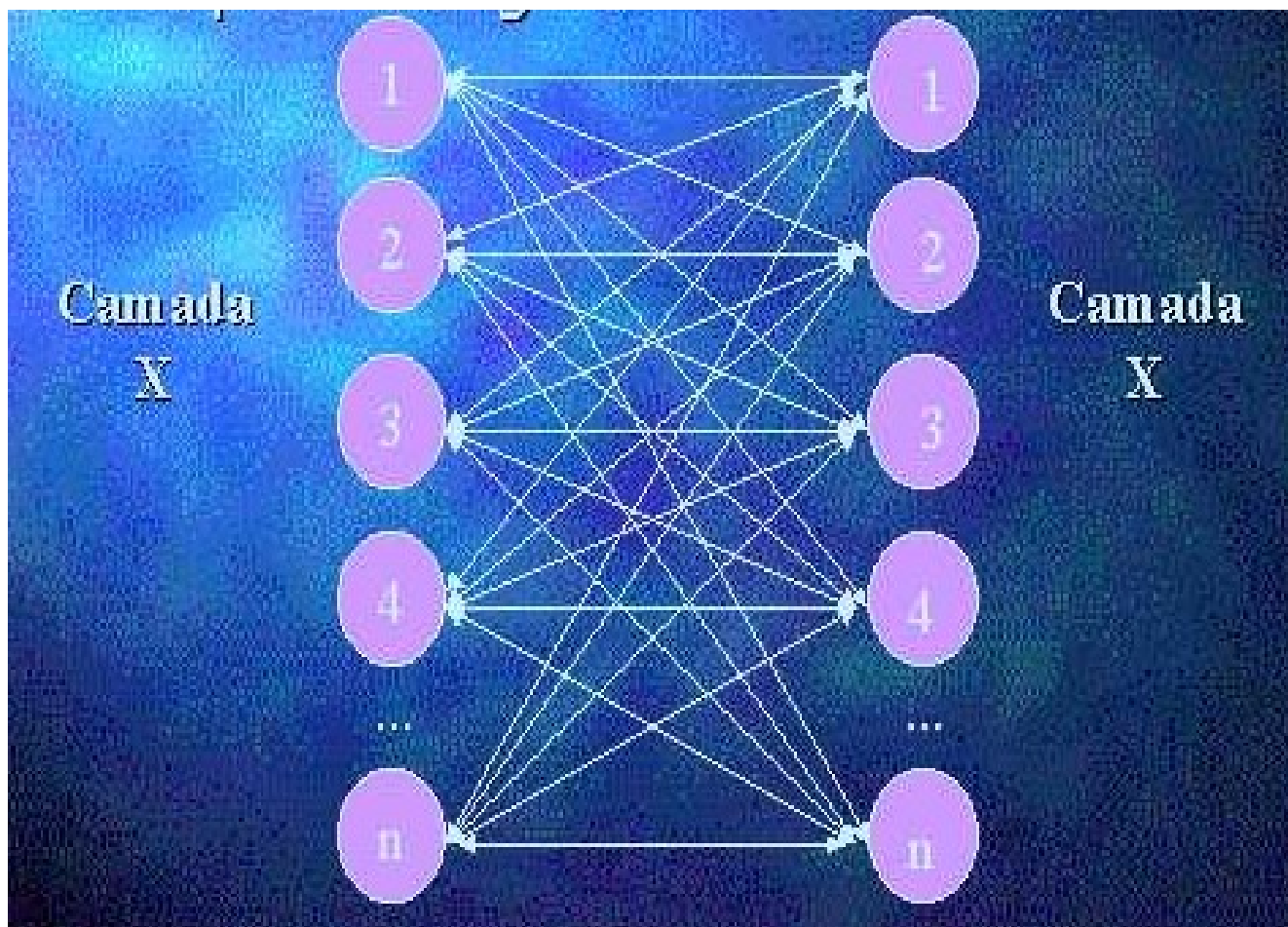
- **Algoritmo de aprendizado: retropropagação**
 - **Critérios de parada**
 - finalizar o treinamento após *n* ciclos
 - finalizar o treinamento após o erro quadrático médio ficar abaixo de uma constante α
 - finalizar o treinamento quando a porcentagem de classificações corretas estiver acima de uma constante α (mais indicado para saídas binárias)
 - combinação dos métodos acima



Rede de *Hopfield*

- Rede cíclica ou recorrente
 - alimenta a sua saída nas suas próprias entradas
- As unidades assumem um estado binário
 - **ativo** ou inativo
 - estas unidades estão conectadas entre si por arestas simétricas com pesos
 - arestas com **pesos positivos** indicam que as duas unidades tendem a **ativar** uma a outra
 - arestas com **pesos negativos** indica que uma unidade **ativa** pode **desativar** outra unidade

Rede de *Hopfield*





Rede de *Hopfield*

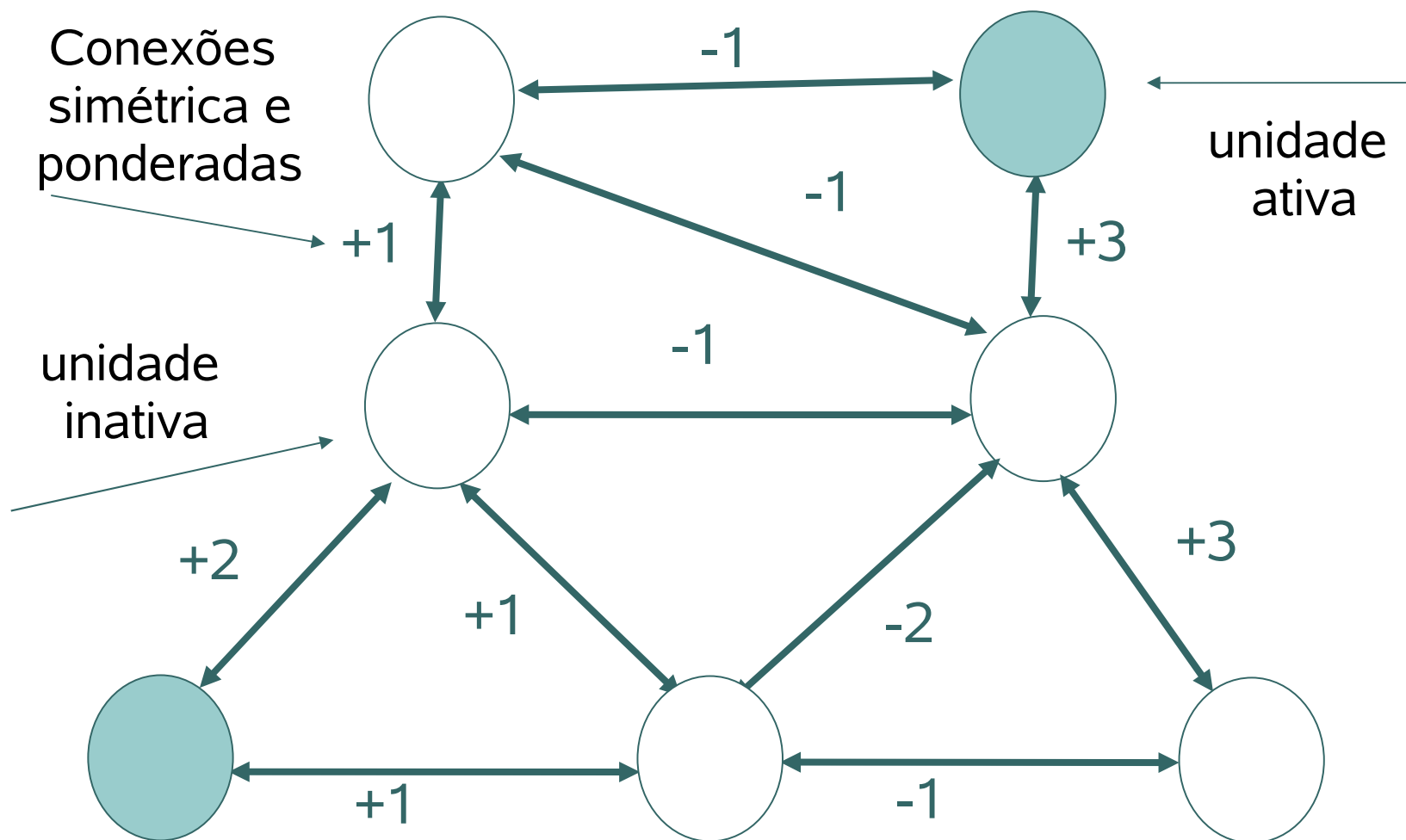
- Dado um padrão que procuramos, podemos encontrar um que se aproxime, sem precisar ser exato
 - O que é um elefante?
 - um mamífero **grande** e cinza
- O aprendizado é não supervisionado e baseado no conceito de energia da rede



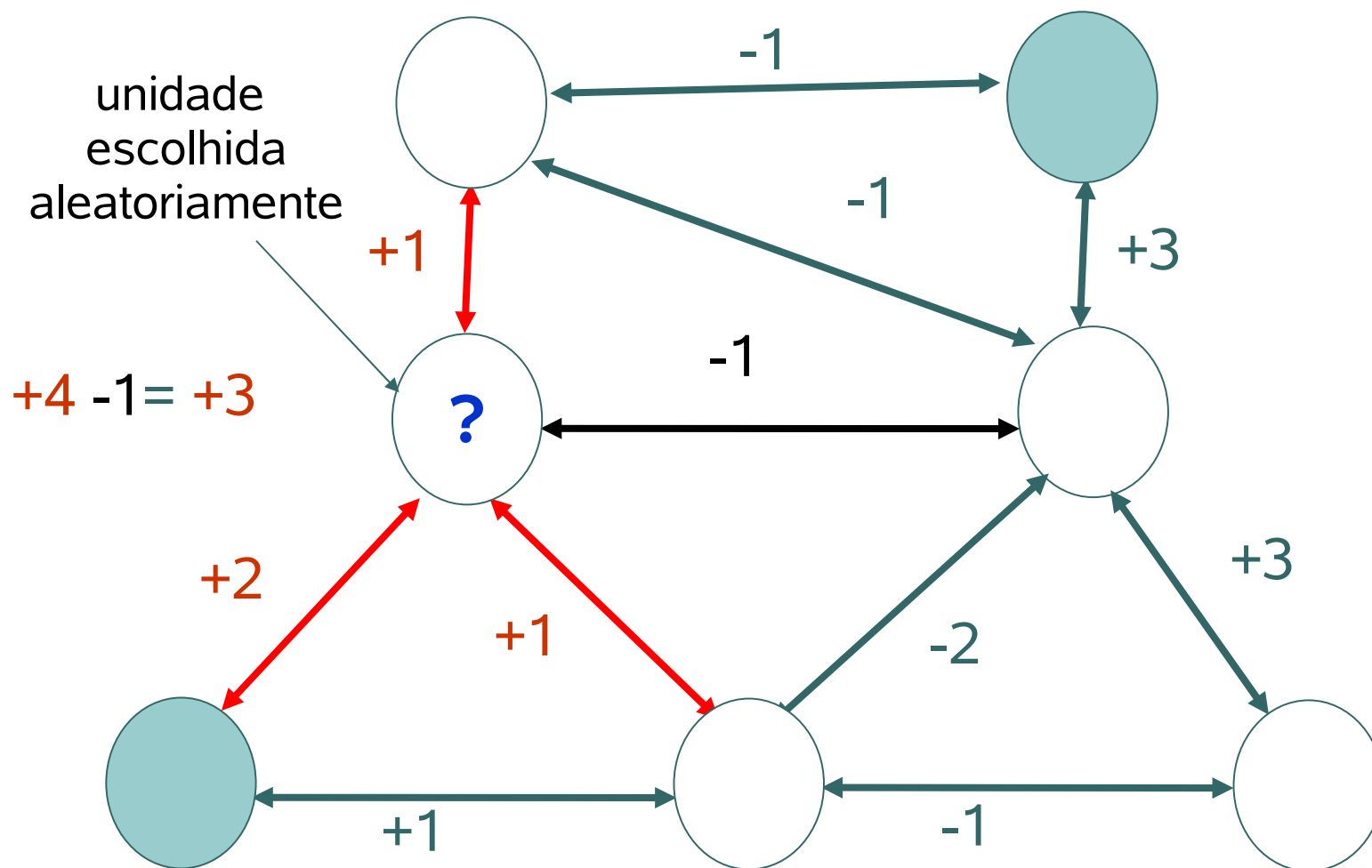
Rede de *Hopfield* Algoritmo

- Uma unidade aleatória é escolhida
- Se qualquer um dos vizinhos estiver **ativo**, a unidade computará a soma dos pesos das conexões com os neurônios vizinhos **ativos**
 - se a soma for **positiva**, a unidade ficará **ativa**
 - se a soma for **negativa**, a unidade ficará **inativa**
- Uma outra unidade aleatória é escolhida e o processo se repete
 - até que toda rede atinja um estado estável
 - **relaxamento paralelo**

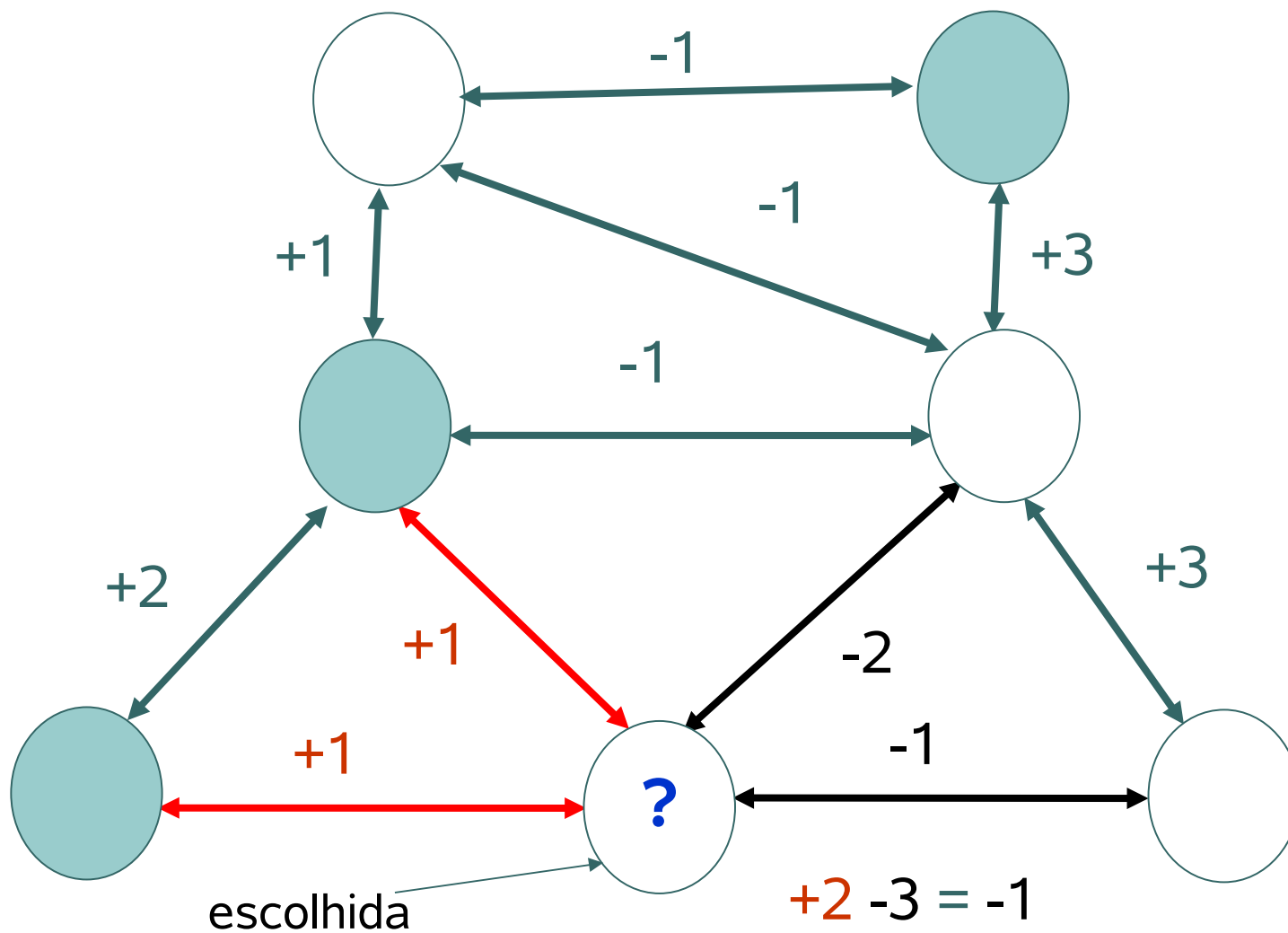
Rede de *Hopfield*



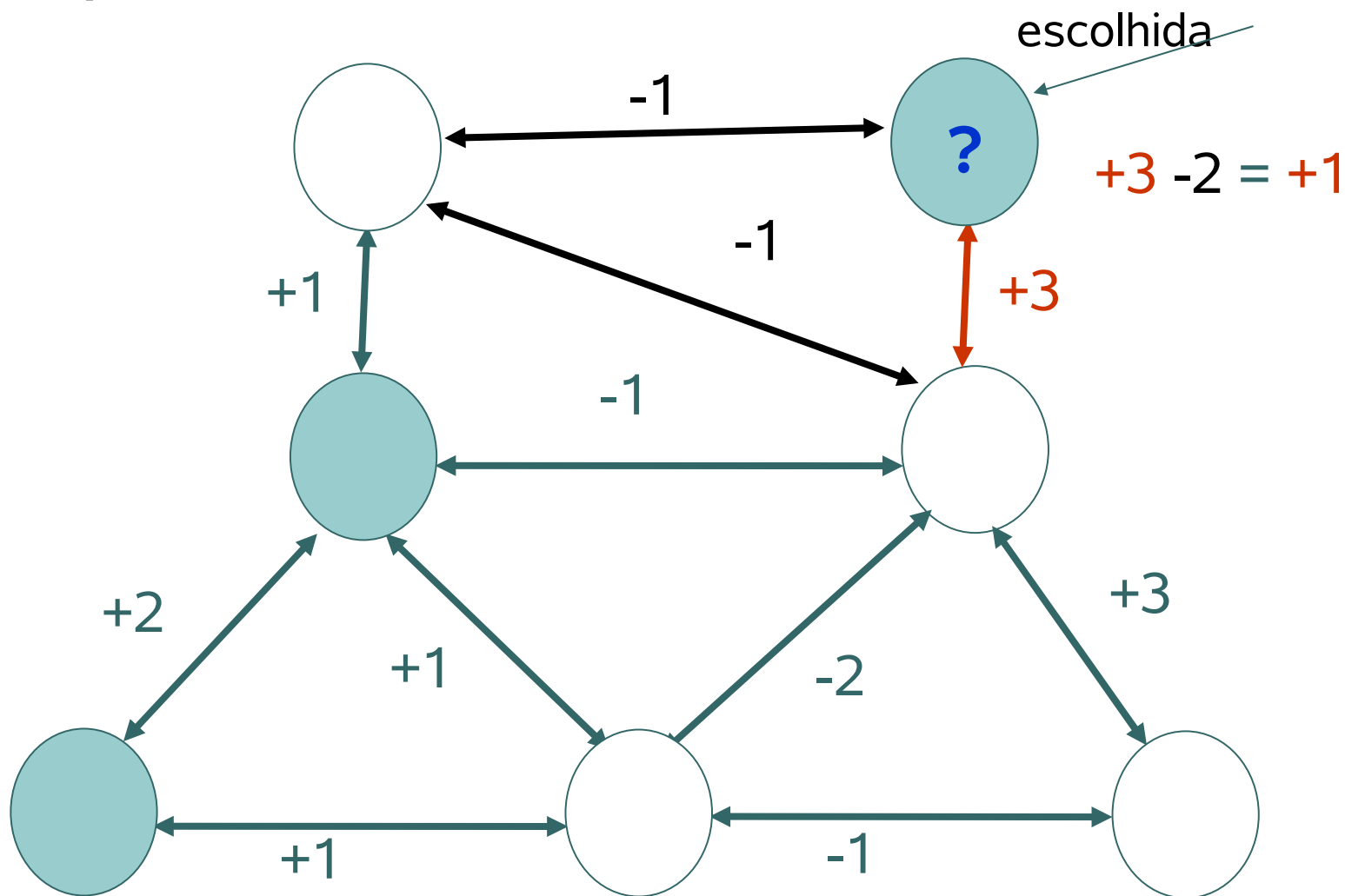
Rede de *Hopfield*



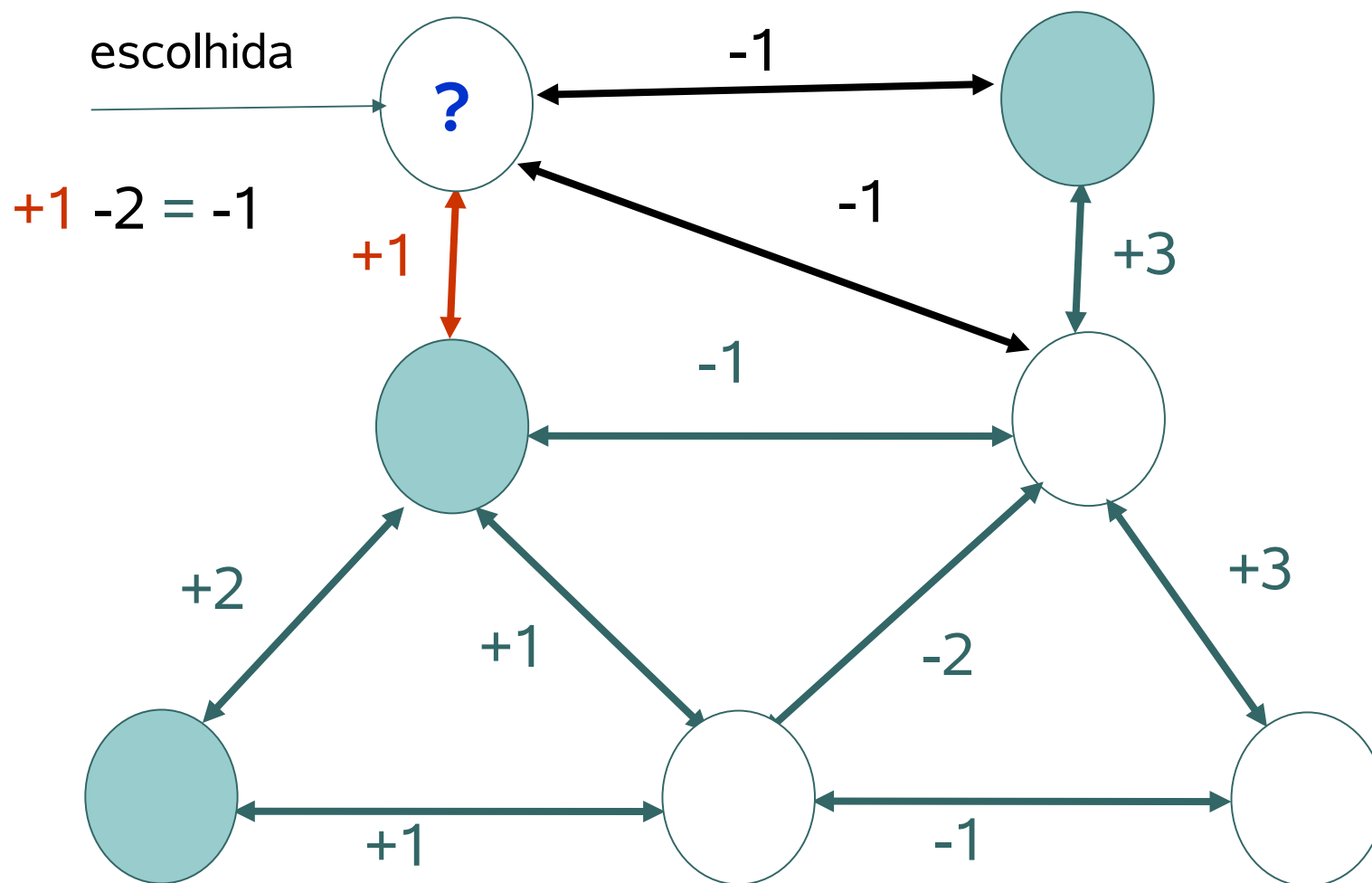
Rede de *Hopfield*



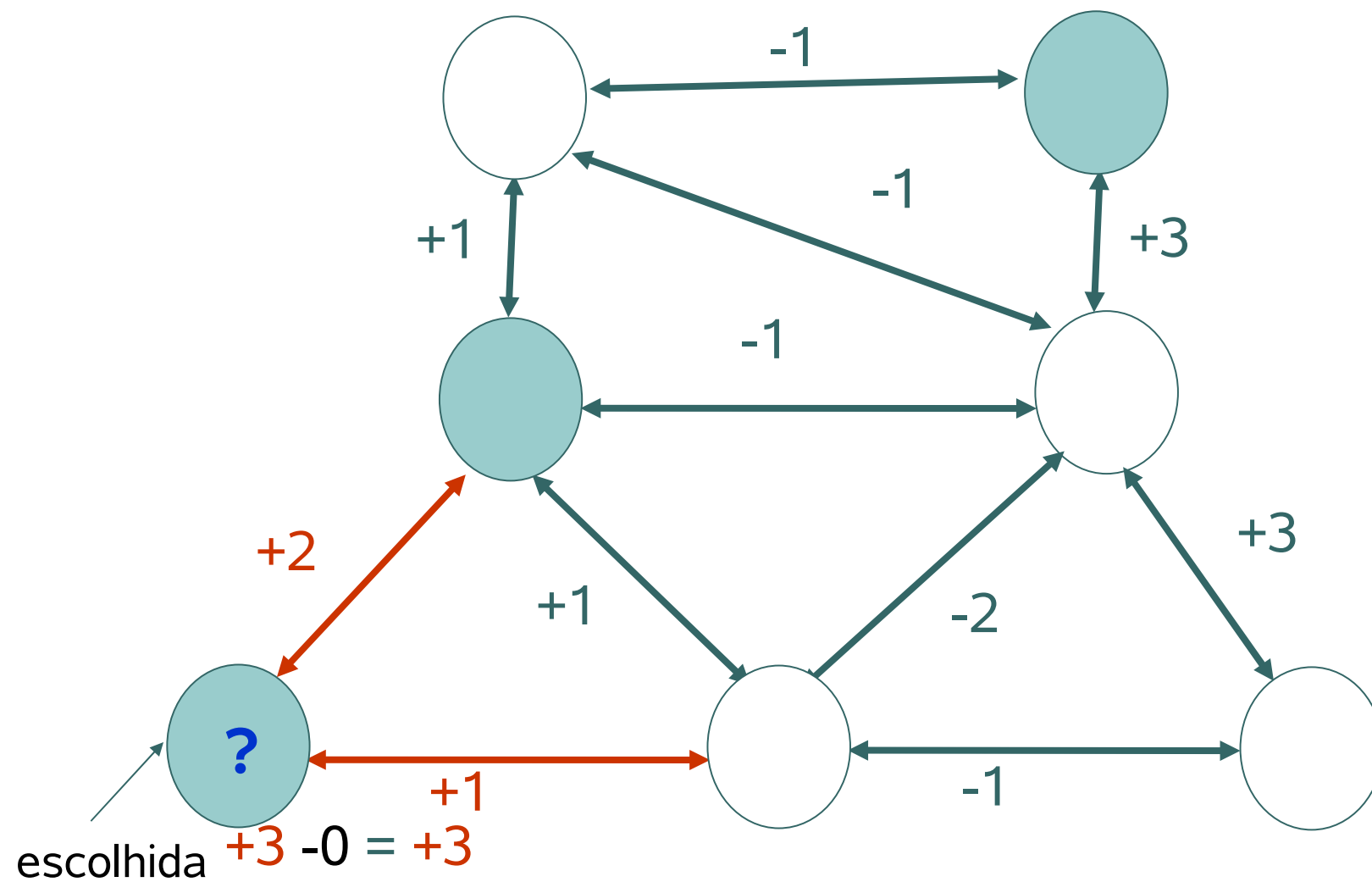
Rede de *Hopfield*



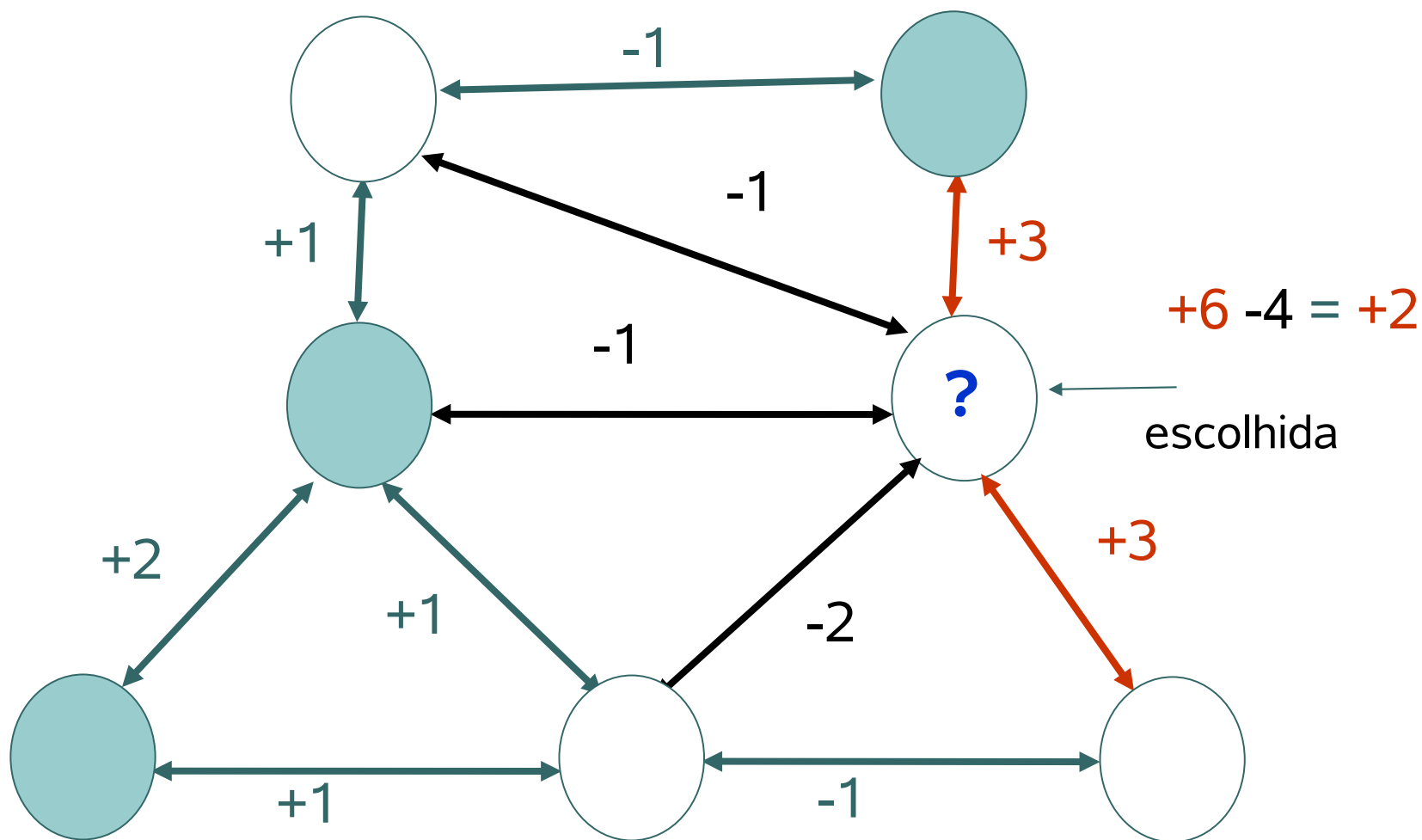
Rede de *Hopfield*



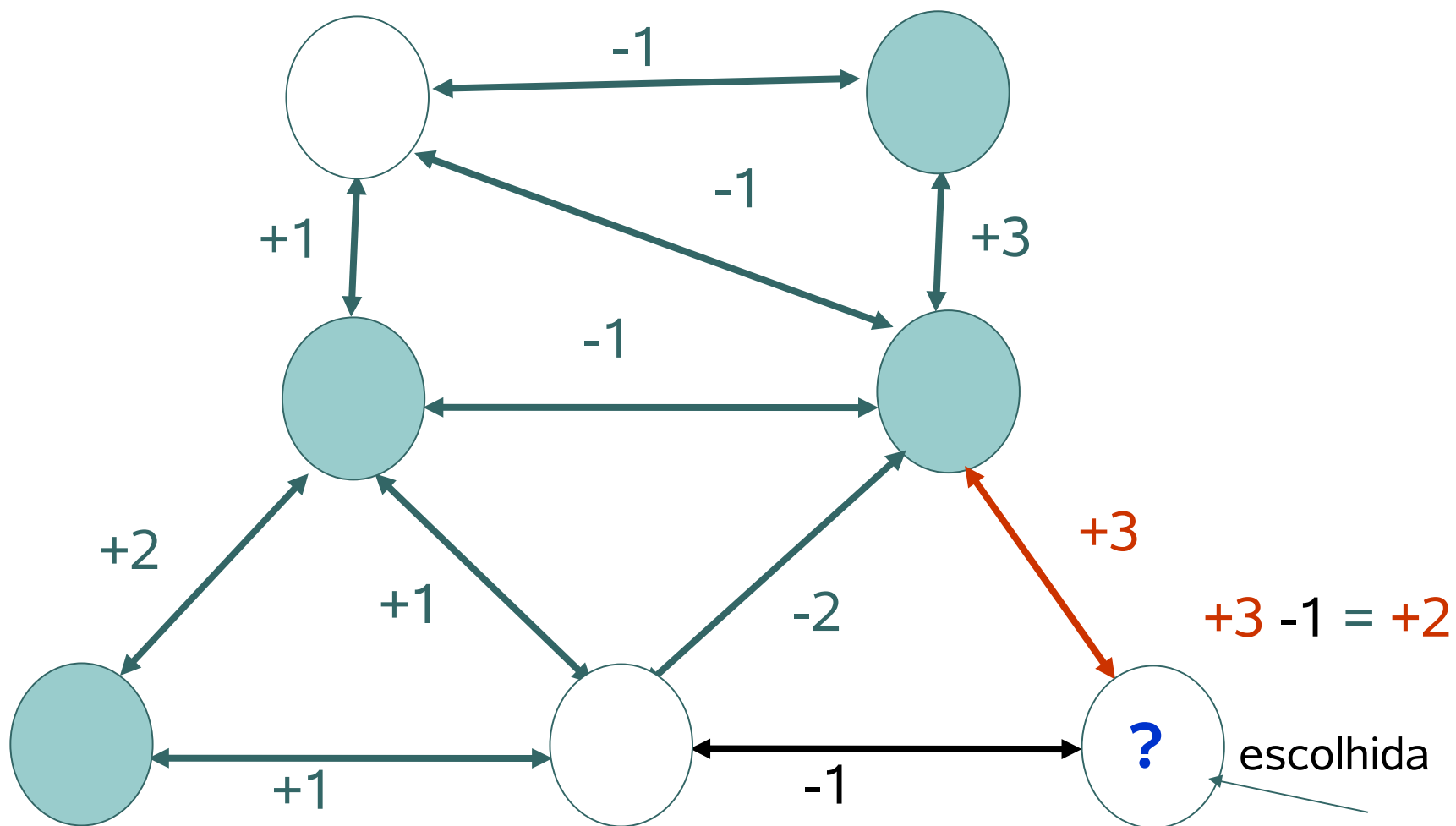
Rede de *Hopfield*



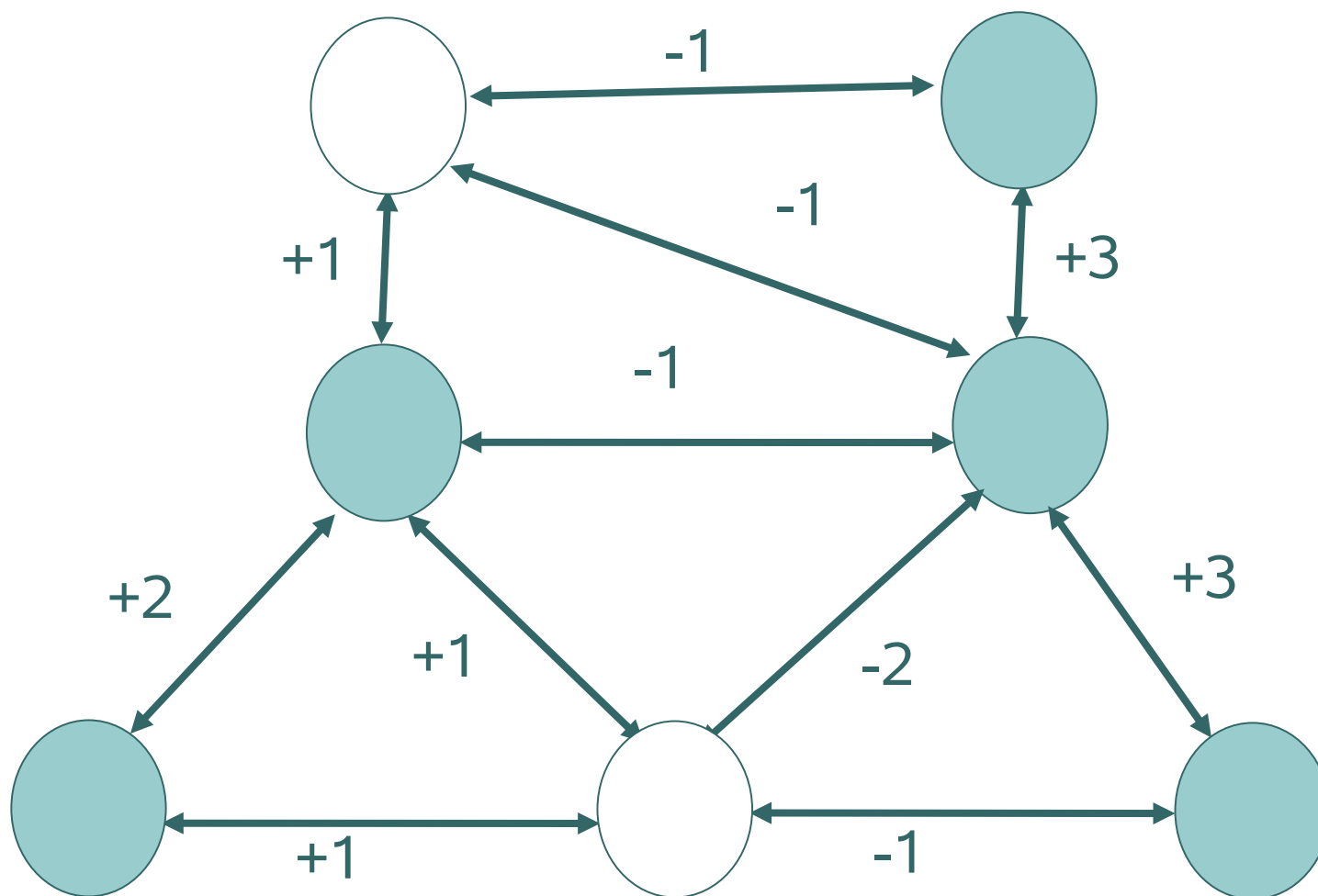
Rede de *Hopfield*



Rede de *Hopfield*



Rede de *Hopfield*



Rede de *Hopfield*

Características interessantes

- Controle distribuído e assíncrono
 - cada elemento de processamento toma decisões baseadas apenas em sua própria situação local
- Memória acessível pelo conteúdo
 - vários padrões podem ser armazenados em uma mesma rede

Rede de *Hopfield*

Características interessantes

- Tolerâncias a falhas
 - se algum dos elementos de processamento se comportarem mal ou falharem totalmente, a rede ainda funcionará bem
- Fator de aproximação
 - por exemplo:
 - “**peixe, grande, cinza e come placton**”???
 - **Baleia**
 - Baleia não é peixe e sim mamífero



Desenvolvimento de aplicações

1. 2. Coleta de dados e separação em conjuntos

- esta tarefa requer uma análise cuidadosa sobre o problema para minimizar **ambigüidades** e **erros nos dados**
- os dados coletados devem ser significativos e cobrir amplamente o domínio do problema
 - não devem cobrir apenas as operações normais ou rotineiras, mas também as **exceções** e as condições nos **limites do domínio** do problema



Desenvolvimento de aplicações

1. 2. Coleta de dados e separação em conjuntos

- os dados coletados são separados em duas categorias:
 - **dados de treinamento**
 - **dados de teste**



Desenvolvimento de aplicações

3. Configuração da rede: pode ser dividido em três etapas:

3.1 Seleção do paradigma neural apropriado à aplicação

- acíclica ou recorrente (cíclica)

3.2 Determinação da topologia da rede a ser utilizada

- número de camadas, número de unidades em cada camada, etc.

3.3 Determinação de parâmetros do algoritmo de treinamento e funções de ativação

- este passo tem um grande impacto na performance do sistema resultante



Desenvolvimento de aplicações

4. Treinamento

- seguindo o algoritmo de treinamento escolhido, serão ajustados os pesos das conexões
- é importante considerar, nesta fase, alguns aspectos
 - a inicialização da rede
 - o modo de treinamento
 - o tempo de treinamento
- o treinamento deve ser interrompido quando
 - a rede apresentar uma boa capacidade de generalização
 - quando a taxa de erro for suficientemente pequena



Desenvolvimento de aplicações

5. Teste

- durante esta fase o conjunto de teste é utilizado para determinar a performance da rede com dados que não foram previamente utilizados
- a performance da rede, medida nesta fase, é uma boa indicação de sua performance real

6. Implantação



Aplicações

- Sistemas de auxílio ao Diagnóstico: Médico, Falhas de Sistemas etc.
- Previsão de Séries Temporais: Cotações da Bolsa de Valores, Dados Econômicos, Consumo de Energia Elétrica, Meteorologia etc.
- Processamento de Linguagem Natural - PLN (Textos e *Web*)



Aplicações

- *Data Mining & KDD (Knowledge Data Discovery)*
- Robótica Inteligente
- Sistemas de Controle e Automação
- Reconhecimento e Síntese de Voz
- Processamento de Sinais e Imagens: Radar, Sensores, Imagens de satélite etc.



Aplicações

- Prognósticos de mercados financeiros
- Reconhecimento ótico de caracteres (OCR)
- Controle de processos industriais
- Análise de jogadores e times (*NBA*)
- Reconhecimento da fala
- Piloto automático
- Reprodução da fala
- SE



Educação

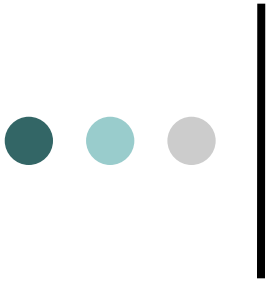
- Em *software* educacionais:
 - identificar deficiências
 - auxiliando
 - avaliar desempenhos
 - prevendo problemas
 - aprendendo



Considerações finais

Vantagens das RNAs

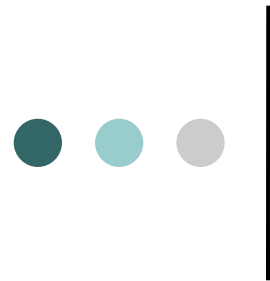
- Aplicações de *Machine Learning* e Sistemas Adaptativos
- Aplicadas em tarefas onde temos bases de exemplos disponíveis sobre um determinado problema, realizando a aquisição automática de conhecimentos
- Associação de padrões de entradas e saída
- Classificação de padrões de forma supervisionada ou não



Considerações finais

Vantagens das RNAs

- Aproximação de funções desconhecidas através de amostras destas funções
- Trabalhar com dados aproximados, incompletos e inexatos
- Paralelismo, generalização, robustez
- “Tarefas complexas realizadas por seres humanos”



Considerações finais

Limitações das RNAs

- Composição e construção de conhecimentos estruturados
- Dificuldade de explicitação dos conhecimentos adquiridos
- Dificuldade para definir a estrutura da rede, seus parâmetros e a base de dados
- Falta de garantia de uma convergência do algoritmo para uma solução ótima



Considerações finais

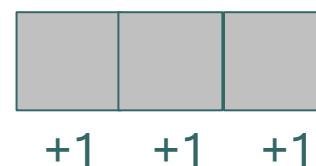
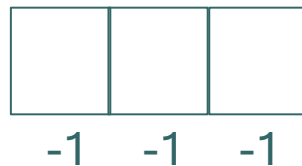
- Apesar da neurocomputação ter nascido praticamente junto com a computação programada (décadas de 40 à década de 50), era inviável que se desenvolvesse
- E o seu futuro???
 - comparável à invenção do avião

Perceptrons

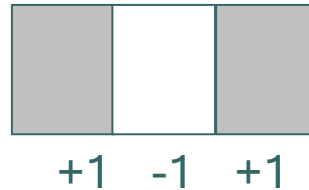
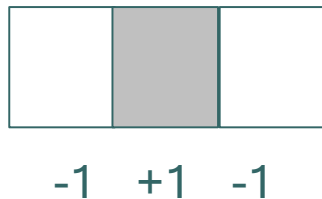
EXERCÍCIO

○ Treinamento

- Ensinar uma rede *Perceptron* a classificar os seguintes padrões:



- Utilizar a rede treinada para classificar os padrões



Perceptrons

EXERCÍCIO

Codificar as entradas

Saída desejada



= 1



= -1

Entrada



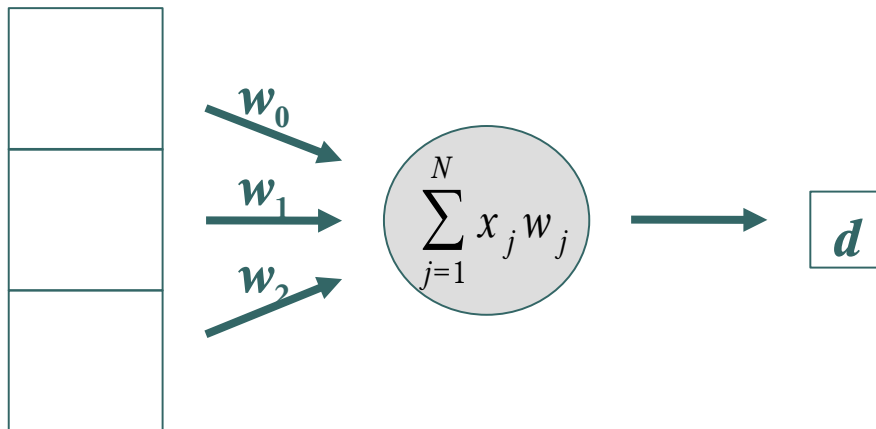
= 111



= -1-1-1

Supor $\eta = 0.2$, $\theta = 0$,

$w_0 = 0.4$, $w_1 = -0.8$, $w_2 = 0.3$



$$\sum_{i=1}^n x_i w_i \geq \theta$$

$$S \geq \theta = 1$$

$$S < \theta = -1$$

Equação do erro:

$$E = S_d - S_o$$

onde

S_d é a saída desejada

S_o é a saída obtida

Fator de correção:

$$F = \eta * x * E$$

onde

η = taxa de aprendizagem

x é a entrada

E é o erro

Equação do ajuste:

$$w_{novo} = w + F$$



Perceptrons EXERCÍCIO

- **Treinar a rede**

- para o padrão

--	--	--

 $\Leftrightarrow -1-1-1$ (**d = -1**)

- **Passo 1:** definir a saída da rede

- Supor $\eta = 0.2$, $\theta = 0$, $w_0 = 0.4$, $w_1 = -0.8$, $w_2 = 0.3$

$$\sum_{i=1}^n x_i w_i \geq \theta$$

- $S = (-1)(0.4) + (-1)(-0.8) + (-1)(0.3) = \mathbf{0.1}$
- **y = +1 (uma vez que $0.1 \geq 0$)**
- **como ($d \neq y$), atualizar pesos**



Perceptrons EXERCÍCIO

- **Treinar a rede**

- para o padrão



$\Leftrightarrow -1-1-1$ (**d = -1**)

- **Passo 2:** atualizar pesos

- Supor $\eta = 0.2$, $\theta = 0$, $w_0 = 0.4$, $w_1 = -0.8$, $w_2 = 0.3$

$$w_i(t+1) = w_i + \eta * e * x_i$$

- $w_0 = 0.4 + 0,2(-1)(-1 - (+1)) = \mathbf{0.8}$
- $w_1 = -0.8 + 0,2(-1)(-1 - (+1)) = \mathbf{-0.4}$
- $w_2 = 0.3 + 0,2(-1)(-1 - (+1)) = \mathbf{0.7}$

Perceptrons

EXERCÍCIO

- **Treinar a rede**

- para o padrão



$\Leftrightarrow -1-1-1$ (**d = -1**)

- **Passo 1:** definir a saída da rede

- Supor $\eta = 0.2$, $\theta = 0$, $w_0 = 0.8$, $w_1 = -0.4$, $w_2 = 0.7$

$$\sum_{i=1}^n x_i w_i \geq \theta$$

- $S = (-1)(0.8) + (-1)(-0.4) + (-1)(0.7) = -1$
- **y = +1** (uma vez que $+1 \geq 0$)
- **como (d = y), não necessita atualizar pesos**



Perceptrons

EXERCÍCIO

- *treinar a rede*

- para o padrão  $\Leftrightarrow 111$ (**d = 1**)

- **Passo 1:** definir a saída da rede

$$S = (1)(0.8) + (1)(-0.4) + (1)(0.7) = 1.1$$

$y = +1$ (uma vez que $1.1 \geq 0$)

como ($d = y$), não precisa atualizar pesos

Agora fazer os testes



Perceptrons EXERCÍCIO

- Validação: testar a rede

- Para o padrão  $\Leftrightarrow -11-1$

- Para o padrão  $\Leftrightarrow 1-11$

- Para o padrão  $\Leftrightarrow 1-11$

Perceptrons

EXERCÍCIO

Validação: testar a rede

- Para o padrão  $\Leftrightarrow -11-1$

$$s = (-1)(0.8) + (1)(-0,4) + (-1)(0.7) = -1.9 \quad \textbf{(classe 1)}$$



- Para o padrão  $\Leftrightarrow 1-11$

$$s = (1)(0.8) + (-1)(-0,4) + (1)(0.7) = 1.9 \quad \textbf{(classe 2)}$$



- Para o padrão  $\Leftrightarrow 1-11$

$$s = (1)(0.8) + (-1)(-0,4) + (-1)(0.7) = 0,5 \quad \textbf{(classe 2)}$$

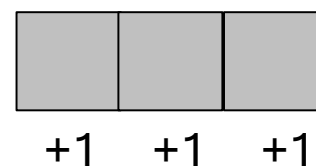
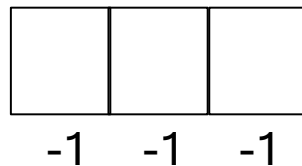


Perceptrons

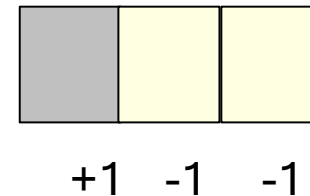
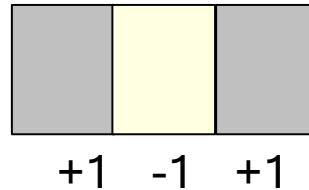
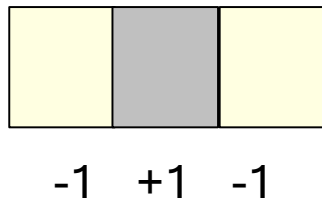
EXERCÍCIO

Treinamento

- Ensinar uma rede *Perceptron* a classificar os seguintes padrões:



- Utilizar a rede treinada para classificar os padrões





Perceptrons

EXERCÍCIO

Codificar as entradas

Saída desejada

 = 1

 = -1

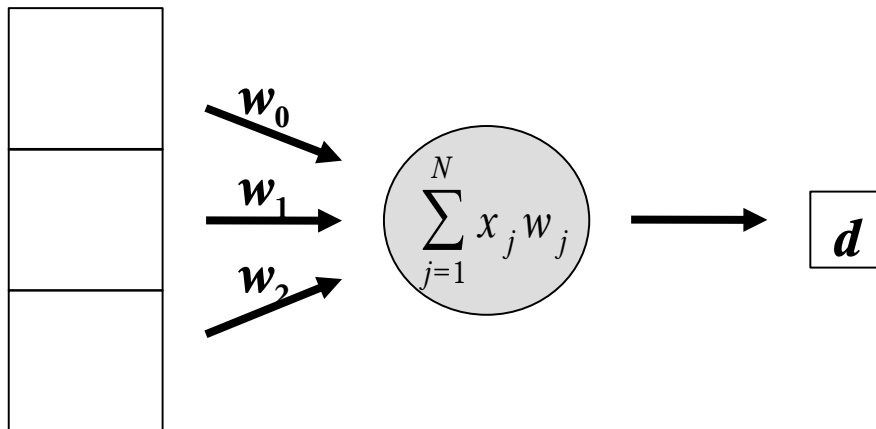
Entrada

 = 111

 = -1-1-1

Supor $\eta = 0.2$, $\theta = 0$,

$w_0 = 0.4$, $w_1 = -0.8$, $w_2 = 0.3$



$$\sum_{i=1}^n x_i w_i \geq \theta$$

$$S \geq \theta = 1$$

$$S < \theta = -1$$

Equação do erro:

$$E = S_d - S_o$$

onde

S_d é a saída desejada

S_o é a saída obtida

Fator de correção:

$$F = \eta * x * E$$

onde

η = taxa de aprendizagem

x é a entrada

E é o erro

Equação do ajuste:

$$w_{novo} = w + F$$



Perceptrons

EXERCÍCIO

■ **Treinar a rede**

- para o padrão

--	--	--

 $\Leftrightarrow -1-1-1$ (**d = -1**)

■ **Passo 1:** definir a saída da rede

- Supor $\eta = 0.2$, $\theta = 0$, $w_0 = 0.4$, $w_1 = -0.8$, $w_2 = 0.3$

$$\sum_{i=1}^n x_i w_i \geq \theta$$

- $S = (-1)(0.4) + (-1)(-0.8) + (-1)(0.3) = \mathbf{0.1}$
- **y = +1** (uma vez que $0.1 \geq 0$)
- **como ($d \neq y$), atualizar pesos**



Perceptrons

EXERCÍCIO

■ *Treinar a rede*

- para o padrão

--	--	--

 $\Leftrightarrow -1-1-1$ (**d = -1**)

■ **Passo 2:** atualizar pesos

- *Supor* $\eta = 0.2$, $\theta = 0$, $w_0 = 0.4$, $w_1 = -0.8$, $w_2 = 0.3$

$$w_i(t+1) = w_i + \eta * e * x_i$$

- $w_0 = 0.4 + 0,2(-1)(-1 - (+1)) = \mathbf{0.8}$
- $w_1 = -0.8 + 0,2(-1)(-1 - (+1)) = \mathbf{-0.4}$
- $w_2 = 0.3 + 0,2(-1)(-1 - (+1)) = \mathbf{0.7}$



Perceptrons

EXERCÍCIO

■ **Treinar a rede**

- para o padrão

--	--	--

 $\Leftrightarrow -1-1-1$ (**d = -1**)

■ **Passo 1:** definir a saída da rede

- Supor $\eta = 0.2$, $\theta = 0$, $w_0 = 0.8$, $w_1 = -0.4$, $w_2 = 0.7$

$$\sum_{i=1}^n x_i w_i \geq \theta$$

- $S = (-1)(0.8) + (-1)(-0,4) + (-1)(0.7) = \mathbf{-1}$
- **y = +1 (uma vez que +1 \geq 0)**
- **como (d = y), não necessita atualizar pesos**



Perceptrons EXERCÍCIO

■ *treinar a rede*

■ para o padrão  $\Leftrightarrow 111$ ($d = 1$)

■ **Passo 1:** definir a saída da rede

$$S = (1)(0.8) + (1)(-0.4) + (1)(0.7) = 1.1$$

$y = +1$ (uma vez que $1.1 \geq 0$)

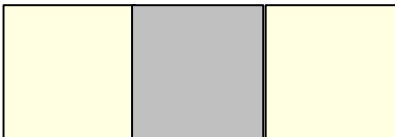
como ($d = y$), não precisa atualizar pesos

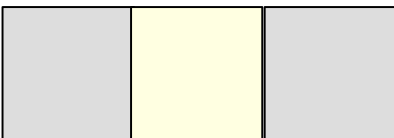
Agora fazer os testes

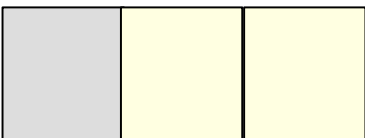


Perceptrons EXERCÍCIO

■ Validação: testar a rede

■ Para o padrão  $\Leftrightarrow -11-1$


■ Para o padrão  $\Leftrightarrow 1-11$

■ Para o padrão  $\Leftrightarrow 1-11$

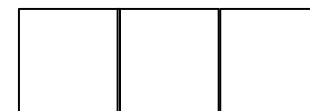
Perceptrons

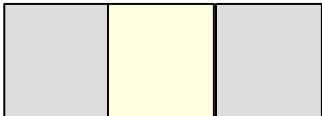
EXERCÍCIO

■ Validação: testar a rede

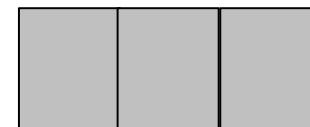
■ Para o padrão  $\Leftrightarrow -11-1$ (classe 1)

$$s = (-1)(0.8) + (1)(-0,4) + (-1)(0.7) = -1.9$$



■ Para o padrão  $\Leftrightarrow 1-11$ (classe 2)

$$s = (1)(0.8) + (-1)(-0,4) + (1)(0.7) = 1.9$$



■ Para o padrão  $\Leftrightarrow 1-11$ (classe 2)

$$s = (1)(0.8) + (-1)(-0,4) + (-1)(0.7) = 0,5$$





Referências

- Russel, S, & Norvig, P. (1995). Artificial Intelligence: a Modern Approach Prentice-Hall. Cap.20.
- Pearl, J. (1988) Probabilistic Reasoning in Intelligent Systems