

# Tópico 9 - Redes Neurais Artificiais

João B. Florindo

Instituto de Matemática, Estatística e Computação Científica  
Universidade Estadual de Campinas - Brasil  
jbflorindo@ime.unicamp.br

# Outline

- 1 Redes Neurais Artificiais
- 2 Redes de Hopfield
- 3 Formulação Algébrica
- 4 Memória Associativa Birecional
- 5 Formulação Algébrica
- 6 Perceptron de Camada Única
- 7 Formulação Algébrica
- 8 Perceptron Multi-Camadas
- 9 Formulação Algébrica

# Redes Neurais Artificiais

Neurônio animal recebe estímulo (impulso elétrico) de órgãos sensoriais e podem repassar ou não para que outros neurônios tratem.

Presença ou ausência de conexões dependem da experiência passada (memória/aprendizado).

RNAs possuem um grande número de elementos simples de processamento (*neurônios*) interconectados (*conexões sinápticas/axônios*).

Caracterizadas pelo padrão de conectividade, pesos das conexões, características de cada elemento de processamento e por regras de treinamento e aprendizado.

Estas regras referem-se a um conjunto de pesos iniciais e estratégias que adaptam estes pesos ao longo do aprendizado.

# Redes Neurais Artificiais

Em geral, toda RNA possui os seguintes elementos:

- 1 Conjunto finito de neurônios  $a(1), a(2), \dots, a(n)$ , sendo que cada neurônio  $a(i)$  vai possuir um valor (estado) associado  $a_t(i)$  no tempo  $t$ ;
- 2 Conjunto finito de conexões  $W = w_{ij}$ , em que  $w_{ij}$  denota o peso da conexão entre os neurônios  $a(i)$  e  $a(j)$ ;
- 3 Regra de propagação  $\tau_t(i) = \sum_{j=1}^n a_t(j) \cdot w_{ij}$ ;
- 4 Função de ativação  $f$  que determina o próximo estado do neurônio a partir de  $\tau$  e de um limiar  $\theta$ :

$$a_{t+1}(i) = f(\tau_t(i) - \theta).$$

Esta função introduz não-linearidade no modelo e costuma ser uma função de limiarização simples ou uma sigmoide.

# Outline

- 1 Redes Neurais Artificiais
- 2 Redes de Hopfield**
- 3 Formulação Algébrica
- 4 Memória Associativa Birecional
- 5 Formulação Algébrica
- 6 Perceptron de Camada Única
- 7 Formulação Algébrica
- 8 Perceptron Multi-Camadas
- 9 Formulação Algébrica

## Rede de Hopfield

Na rede de Hopfield, um padrão é representado por um vetor  $N$ -dimensional assumindo valores  $-1$  ou  $1$ , ou seja:

$$\mathbf{p} = (p_1, p_2, \dots, p_N), \text{ tal que } \mathbf{p} \in \mathbf{P} = \{-1, 1\}^N.$$

O aprendizado é *supervisionado*, de modo que a rede deve ser apresentada a um subconjunto especial  $\mathbf{E} \subset \mathbf{P}$  contendo  $K$  padrões de exemplo:

$$\mathbf{E} = \{e^k : 1 \leq k \leq K\}, \text{ sendo que } e^k = (e_1^k, e_2^k, \dots, e_N^k).$$

Para cada novo padrão  $\mathbf{p} \in \mathbf{P}$  que possa então ser apresentado à rede, esta vai associar com algum padrão de exemplo em  $\mathbf{E}$  por similaridade.

Temos assim um processo de classificação baseada em aprendizado.

Vamos analisar agora cada elemento da rede de Hopfield.

# Rede de Hopfield

## Neurônio

Conjunto finito de neurônios  $\mathbf{a}(i)$ ,  $1 \leq i \leq N$ . No tempo  $t$ , temos  $\mathbf{a}_t(i) \in \{-1, +1\}$ .

## Conexões

Exige-se que  $w_{ij} = w_{ji}$  e  $w_{ii} = 0$ .

## Regra de propagação

$$\tau_t(i) = \sum_{j=1}^N \mathbf{a}_t(j) w_{ij}.$$

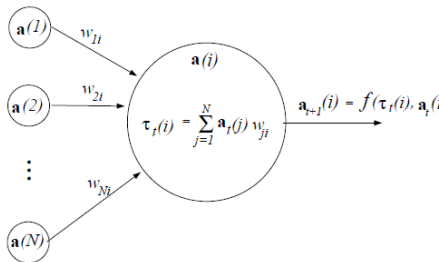
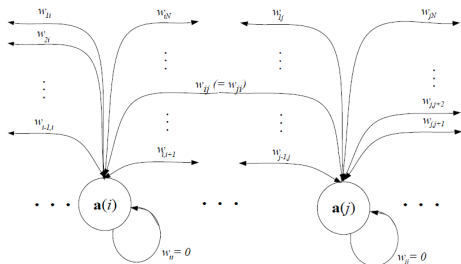
# Rede de Hopfield

## Função de ativação

Limiar simples (“duro”):

$$\mathbf{a}_{t+1}(i) = f(\tau_t(i), \mathbf{a}_t(i)) = \begin{cases} \mathbf{a}_t(i) & \text{se } \tau_t(i) = 0 \\ 1 & \text{se } \tau_t(i) > 0 \\ -1 & \text{se } \tau_t(i) < 0. \end{cases}$$

A rede vai ter  $N$  nós no total.





# Rede de Hopfield

Passos do método:

- 1 Atribuir pesos a partir dos padrões de exemplo:

$$w_{ij} = \begin{cases} \sum_{k=1}^N e_i^k e_j^k & \text{se } i \neq j \\ 0 & \text{se } i = j. \end{cases}$$

Como  $w_{ij} = w_{ji}$ , este cálculo só precisa ser feito para  $i < j$ .

- 2 Para cada novo padrão a ser classificado, inicializam-se os neurônios com este padrão:

$$\mathbf{a}_0(i) = p_i, 1 \leq i \leq N.$$

- 3 Calcular próximo estado:

$$\mathbf{a}_{t+1}(i) = f \left( \sum_{j=1}^N \mathbf{a}_t(j) w_{ij}, \mathbf{a}_t(i) \right).$$

Repete-se esta iteração até convergir (os estados dos neurônios não mudarem mais). Os estados finais são os mesmos do padrão de exemplo que melhor corresponde ao padrão introduzido.

## Rede de Hopfield - Exemplo

Seja um meio que transmite 6 tipos de símbolos (1, 2, 3, 4, 9 e X) por imagens binárias  $12 \times 10$ . Padrões de exemplo abaixo.



Uma imagem binária  $\mathbf{b} \in \{0, 1\}^{\mathbf{X}}$ , em que  $\mathbf{X} = \mathbb{Z}_{12} \times \mathbb{Z}_{10}$  pode ser mapeada para o padrão  $\mathbf{p} = (p_1, p_2, \dots, p_{120})$  por:

$$p_{10(i-1)+j} = \begin{cases} -1 & \text{se } \mathbf{b}(i, j) = 0 \\ 1 & \text{se } \mathbf{b}(i, j) = 1 \end{cases}$$

e vice-versa:

$$\mathbf{b}(i, j) = \begin{cases} 0 & \text{se } p_{10(i-1)+j} = -1 \\ 1 & \text{se } p_{10(i-1)+j} = 1. \end{cases}$$

## Rede de Hopfield - Exemplo

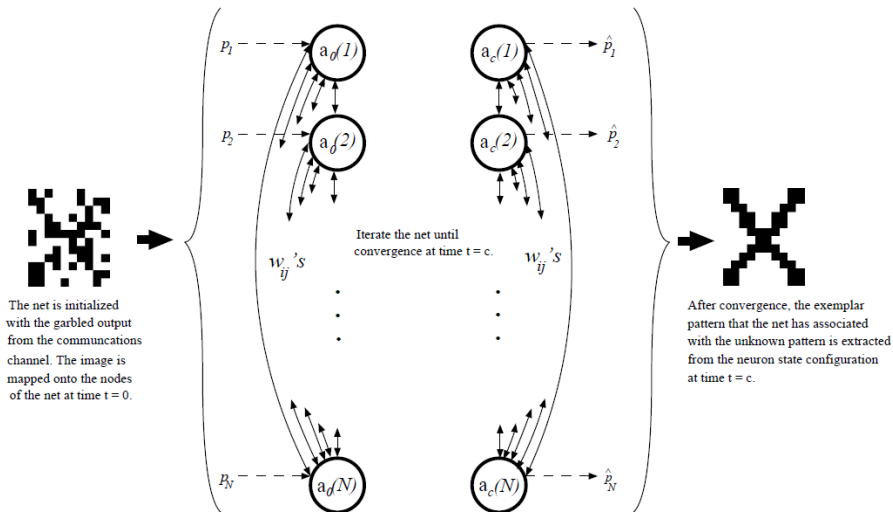
A rede terá portanto 120 nós.

Pode-se receber um símbolo em uma imagem binária com ruído.

Se há convergência para algum padrão de exemplo, este é considerado como sendo o símbolo correto (sem ruído) representado pela imagem de entrada.

A figura seguinte resume o processo.

# Rede de Hopfield - Exemplo



# Outline

- 1 Redes Neurais Artificiais
- 2 Redes de Hopfield
- 3 Formulação Algébrica**
- 4 Memória Associativa Birecional
- 5 Formulação Algébrica
- 6 Perceptron de Camada Única
- 7 Formulação Algébrica
- 8 Perceptron Multi-Camadas
- 9 Formulação Algébrica

# Rede de Hopfield-Álgebra

Os estados dos neurônios podem ser armazenados em uma imagem  $\mathbf{a} \in \{-1, 1\}^{\mathbb{Z}_N}$ .

$\mathbf{a}$  é inicializada com o padrão desconhecido a ser classificado.

Os pesos das conexões são expressos por um *template*  $\mathbf{t} \in (\mathbb{R}^{\mathbb{Z}_N})^{\mathbb{Z}_N}$ :

$$\mathbf{t}_i(j) = \begin{cases} \sum_{k=1}^K e_j^k e_i^k & \text{se } i \neq j \\ 0 & \text{se } i = j. \end{cases}$$

Temos então o seguinte algoritmo geral:

---

```

b ← 0;
enquanto a ≠ b faça
  |   b ← a;
  |   a ←  $f(\mathbf{a} \oplus \mathbf{t}, \mathbf{a})$ ;
fim
  
```

# Rede de Hopfield-Álgebra

O algoritmo anterior permite paralelismo mas não atualiza os neurônios assincronamente, o que é um requisito para que se garanta convergência.

Uma implementação assíncrona seria a seguinte:

---



---

```

b ← 0;
enquanto a ≠ b faça
  | Y ← domain(a);
  | enquanto Y ≠ ∅ faça
  | | i ← choice(Y);
  | | a(i) ← f(a ⊕ ti, a(i));
  | | Y ← Y \ {i};
  | fim
fim
  
```

---

## Rede de Hopfield-Observações

A convergência é garantida se os neurônios forem atualizados assincronamente e  $w_{ij} = w_{ji}$ .

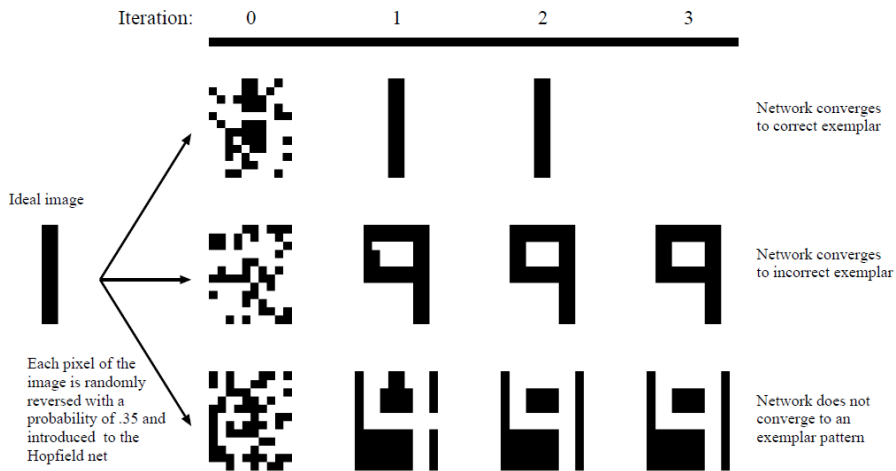
Porém pode convergir para padrão incorreto ou que não seja exemplo (figura com imagens corrompidas com probabilidade 0.35).

A convergência para um padrão que não é exemplo geralmente é associada a um número de padrões exemplares muito grande em relação ao número de nós.

Já a convergência para padrão incorreto costuma estar associada à presença de padrões de exemplo muito similares.



# Rede de Hopfield-Observações



# Outline

- 1 Redes Neurais Artificiais
- 2 Redes de Hopfield
- 3 Formulação Algébrica
- 4 Memória Associativa Birecional**
- 5 Formulação Algébrica
- 6 Perceptron de Camada Única
- 7 Formulação Algébrica
- 8 Perceptron Multi-Camadas
- 9 Formulação Algébrica

# Memória Associativa Birecional

Uma *memória associativa* é uma transformação, não necessariamente linear, entre espaços vetoriais

$$\mathcal{T} : \mathbb{R}^M \rightarrow \mathbb{R}^N.$$

Uma rede de Hopfield é um caso especial de memória associativa  $\mathcal{H} : \mathbb{R}^M \rightarrow \mathbb{R}^M$  que idealmente se comporta como a identidade quando restrita aos padrões de exemplo:

$$\mathcal{H}|_{\mathbf{E}} = \mathcal{I}_{\mathbf{E}}$$

e devolve o padrão de exemplo mais similar ao de entrada quando este último não faz parte de  $\mathbf{E}$ .

# Memória Associativa Birecional

Uma *memória associativa birecional (MAB)* é uma generalização da rede de Hopfield em que a temos um conjunto de associações

$$\mathbf{A} = \{(\mathbf{a}^k, \mathbf{b}^k) : \mathbf{a}^k \in \mathbb{R}^M \text{ e } \mathbf{b}^k \in \mathbb{R}^N, 1 \leq k \leq K\}$$

e a transformação  $\mathcal{B}$  satisfaz

$$\mathcal{B}|_{\{\mathbf{a}^k : 1 \leq k \leq K\}} \equiv \mathbf{A},$$

ou seja,

$$\mathcal{B}(\mathbf{a}^k) = \mathbf{b}^k, 1 \leq k \leq K.$$

Se o padrão de entrada  $\mathbf{a}$  não fizer parte de  $\{\mathbf{a}^k : 1 \leq k \leq K\}$ , a MAB deve convergir para o par  $(\mathbf{a}^k, \mathbf{b}^k)$  que possua  $\mathbf{a}^k$  mais similar a  $\mathbf{a}$ .

Veremos os componentes a seguir.

# Memória Associativa Birecional

## Neurônios

Como domínio e contradomínio podem ter dimensões diferentes, temos agora um conjunto  $\mathbf{S}_a$  para as entradas e outro  $\mathbf{S}_b$  para as saídas:

$$\begin{aligned}\mathbf{S}_a &= \{\mathbf{a}(m) : 1 \leq m \leq M\} \\ \mathbf{S}_b &= \{\mathbf{b}(n) : 1 \leq n \leq N\}.\end{aligned}$$

Os estados no tempo  $t$  são denotados  $\mathbf{a}_t(m)$  e  $\mathbf{b}_t(m)$  e aqui só poderão assumir valor 1 ou  $-1$ .

# Memória Associativa Birecional

## Conexões

As associações  $(\mathbf{a}^k, \mathbf{b}^k)$ , tais que  $\mathbf{a}^k = (a_1^k, a_2^k, \dots, a_M^k)$  e  $\mathbf{b}^k = (b_1^k, b_2^k, \dots, b_N^k)$ , são armazenados na memória permanente da MAB, representada por uma matriz  $M \times N$  em cada elemento é

$$w_{mn} = \sum_{k=1}^K a_m^k b_n^k.$$

## Função de Ativação

Novamente, usa-se limiarização “dura”:

$$f(x, y) = \begin{cases} y & \text{se } x = 0 \\ 1 & \text{se } x > 0 \\ -1 & \text{se } x < 0. \end{cases}$$

# Memória Associativa Birecional

O próximo estado de um neurônio em  $\mathbf{S}_b$  é dado por

$$\mathbf{b}_{t+1}(n) = f(\mathbf{a}_t w_{.n}, \mathbf{b}_t(n)) = \begin{cases} \mathbf{b}_t(n) & \text{se } \mathbf{a}_t w_{.n} = 0 \\ 1 & \text{se } \mathbf{a}_t w_{.n} > 0 \\ -1 & \text{se } \mathbf{a}_t w_{.n} < 0, \end{cases}$$

considerando-se que  $\mathbf{a}_t = (\mathbf{a}_t(1), \mathbf{a}_t(2), \dots, \mathbf{a}_t(M))$  e  $w_{.n}$  é a  $n$ -ésima coluna de  $w$ .

Usando notação similar, o próximo estado de um neurônio em  $\mathbf{S}_a$  é dado por

$$\mathbf{a}_{t+1}(m) = f(\mathbf{b}_t w'_{.m}, \mathbf{a}_t(m)) = \begin{cases} \mathbf{a}_t(m) & \text{se } \mathbf{a}_t w'_{.m} = 0 \\ 1 & \text{se } \mathbf{a}_t w'_{.m} > 0 \\ -1 & \text{se } \mathbf{a}_t w'_{.m} < 0, \end{cases}$$

em que  $w'$  denota transposta de  $w$ .

# Memória Associativa Birecional

Passos do algoritmo:

- 1 Gerar matriz de conexões:  $w_{ij} = \sum_{k=1}^K a_m^k b_n^k$ .
- 2 Dado o padrão desconhecido de entrada  $\mathbf{p} = (p_1, p_2, \dots, p_M)$ , iniciar os neurônios de  $\mathbf{S}_a$  por

$$\mathbf{a}_0(m) = p_m, 1 \leq m \leq M$$

e os de  $\mathbf{S}_b$  aleatoriamente com valores 1 ou  $-1$ :

$$\mathbf{b}_0(n) = \text{choice}(\{-1, 1\}), 1 \leq n \leq N.$$

- 3 Calcular próximo estado de neurônio em  $\mathbf{S}_a$  por

$$\mathbf{b}_{t+1}(n) = f(\mathbf{a}_t w_{.n}, \mathbf{b}_t(n)), 1 \leq n \leq N$$

e em  $\mathbf{S}_a$  por

$$\mathbf{a}_{t+1}(m) = f(\mathbf{b}_t w'_{.m}, \mathbf{a}_t), 1 \leq m \leq M.$$



# Memória Associativa Birecional

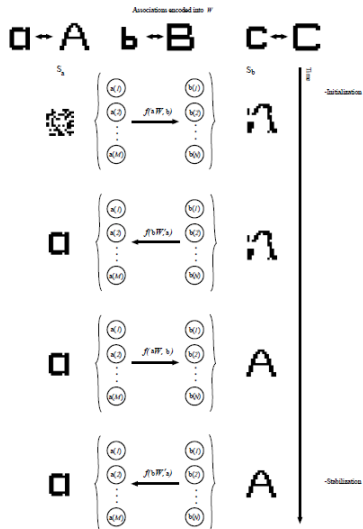
O nome *bidirecional* vem desta retro-alimentação na atualização dos estados em  $\mathbf{S}_a$  e  $\mathbf{S}_b$ .

A iteração se repete até que nenhum neurônio mude mais de estado (convergência). A figura seguinte ilustra o processo.

MABs possuem limitações semelhantes às redes de Hopfield: número limitado de associações armazenadas, tendência de convergir para uma associação incorreta se existem pares de associação que compartilham muitos *pixels* em comum, etc. (figura a seguir)

Pode haver convergência para o complemento de um padrão já que a associação complementar também é armazenada na memória (bloco 5 da figura de convergência).

# Memória Associativa Birecional



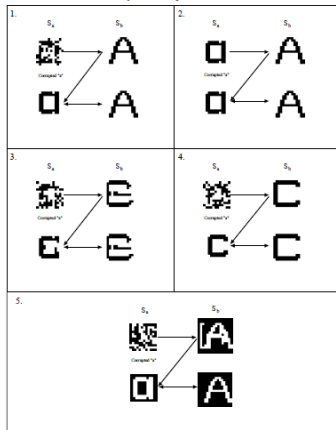
# Memória Associativa Birecional

Associations encoded into the BAM

$S_1$     $S_2$     $S_1$     $S_2$     $S_1$     $S_2$

**a** → **A**   **b** → **B**   **c** → **C**

Five examples of the convergence behavior of the BAM



# Outline

- 1 Redes Neurais Artificiais
- 2 Redes de Hopfield
- 3 Formulação Algébrica
- 4 Memória Associativa Birecional
- 5 Formulação Algébrica**
- 6 Perceptron de Camada Única
- 7 Formulação Algébrica
- 8 Perceptron Multi-Camadas
- 9 Formulação Algébrica

# Memória Associativa Birecional - Álgebra

Os estados dos neurônios em  $\mathbf{S}_a$  e  $\mathbf{S}_b$  são armazenados em variáveis de imagens  $\mathbf{a} \in \{-1, 1\}^{\mathbb{Z}_M}$  e  $\mathbf{b} \in \{-1, 1\}^{\mathbb{Z}_N}$ .

$\mathbf{a}$  é iniciado com o padrão desconhecido e  $\mathbf{S}_b$  randomicamente com  $-1$  ou  $1$ .

Matriz de pesos  $W$  representada pelo *template*  $\mathbf{t} \in (\mathbb{R}^{\mathbb{Z}_N})^{\mathbb{Z}_M}$ :

$$\mathbf{t}_m(n) = \sum_{k=1}^K a_m^k b_n^k$$

em que  $a_m^k$  representa o  $m$ -ésimo componente de  $\mathbf{a}^k$  na associação  $(\mathbf{a}^k, \mathbf{b}^k)$ .

# Memória Associativa Birecional

---

---

**c**  $\leftarrow$  **0**;

**enquanto** **a**  $\neq$  **c** **faça**

**c**  $\leftarrow$  **a**;

**b**  $\leftarrow f(\mathbf{a} \oplus \mathbf{t}, \mathbf{b})$ ;

**a**  $\leftarrow f(\mathbf{b} \oplus \mathbf{t}', \mathbf{a})$ ;

**fim**

---

# Outline

- 1 Redes Neurais Artificiais
- 2 Redes de Hopfield
- 3 Formulação Algébrica
- 4 Memória Associativa Birecional
- 5 Formulação Algébrica
- 6 Perceptron de Camada Única**
- 7 Formulação Algébrica
- 8 Perceptron Multi-Camadas
- 9 Formulação Algébrica

# Perceptron Camada Única

Um *perceptron de camada única* (SLP em inglês) classifica um padrão  $\mathbf{p} = (p_1, p_2, \dots, p_m) \in \mathbf{P} \subset \mathbb{R}^m$  em duas classes.

Um SLP deve conter um conjunto de pesos

$$\{w_0, w_1, \dots, w_m\}, w_i \in \mathbb{R}, 0 \leq i \leq m$$

e uma função limitante  $f : \mathbb{R} \setminus \{0\} \rightarrow \{0, 1\}$ :

$$f(x) = \begin{cases} 1 & \text{se } x > 0 \\ 0 & \text{se } x \leq 0. \end{cases}$$

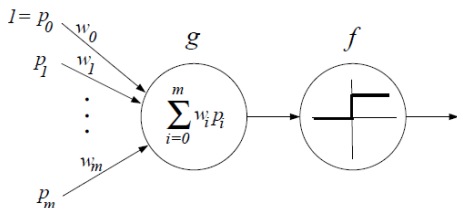
Em geral, dado  $\mathbf{p}$ , calcula-se

$$g(\mathbf{p}) = w_0 + w_1 p_1 + \dots + w_m p_m$$

seguido da aplicação da função limitante. Se  $f \circ g(\mathbf{p}) < 0$ , atribui-se  $\mathbf{p}$  à classe  $C_0$  e se  $f \circ g(\mathbf{p}) > 0$ , atribui-se à classe  $C_1$ .



# Perceptron Camada Única



O gráfico de  $g(\mathbf{x}) = 0$  forma um hiperplano, chamado de *superfície de decisão* que divide  $\mathbb{R}^m$  em duas partes.

Dois modos de operação: classificação (acima) e aprendizado (ajuste dos pesos e da superfície de decisão consequentemente).

Vamos ver um exemplo de algoritmo de aprendizado.

# Perceptron Camada Única

Seja  $\{(\mathbf{p}_k, y_k)\}_{k=1}^n$  um conjunto de treinamento, tal que

$\mathbf{p}_k = (p_{k_1}, p_{k_2}, \dots, p_{k_m}) \in \mathbf{P}$  e  $y_k \in \{0, 1\}$  é a classe associada ao padrão.

Seja ainda  $w_i(t)$  o  $i$ -ésimo peso após o  $t$ -ésimo padrão de treinamento ter sido introduzido no SLP.

- 1 Iniciar cada  $w_i(0)$ ,  $0 \leq i \leq m$  randomicamente;
- 2 Apresentar  $\mathbf{p}_k$  ao SLP e calcular sua classe  $y'_k$ :

$$y'_k(t) = f \left( w_0(t-1) + \sum_{i=1}^m p_{k_i} \cdot w_i(t-1) \right).$$

- 3 Ajustar os pesos

$$w_i(t) = w_i(t-1) + \eta \cdot (y_k - y'_k(t)) \cdot p_{k_i}.$$

- 4 Repetir passos 2 e 3 até convergência ou um número máximo de iterações.

# Perceptron Camada Única

A constante  $0 < \eta \leq 1$  regula a taxa de ajuste de pesos. Se for muito pequena o aprendizado fica lento, se for muito grande os pesos não se ajustam adequadamente.

Se  $\mathbf{p}_k$  é classificado corretamente,  $y_k - y'_k(t) = 0$  e não há mais nenhum ajuste de peso. Se  $y_k - y'_k(t) \neq 0$  a alteração no peso é proporcional ao padrão de entrada.

# Outline

- 1 Redes Neurais Artificiais
- 2 Redes de Hopfield
- 3 Formulação Algébrica
- 4 Memória Associativa Birecional
- 5 Formulação Algébrica
- 6 Perceptron de Camada Única
- 7 Formulação Algébrica**
- 8 Perceptron Multi-Camadas
- 9 Formulação Algébrica

# Perceptron Camada Única - Álgebra

Guardamos os pesos na variável imagem  $\mathbf{w} \in \mathbb{R}^{m+1}$ .

O padrão a ser classificado é  $\mathbf{p} \in \mathbb{R}^m$ .

Definimos também  $\tilde{\mathbf{p}}$  aumentado com um 1:

$$\tilde{\mathbf{p}} = (1, p_1, \dots, p_m) \in \{1\} \times \mathbb{R}^m.$$

A classe de  $\mathbf{p}$  é obtida por

$$y := f\left(\sum(\mathbf{w} \cdot \mathbf{p})\right).$$

O conjunto de treinamento será denotado por

$$\{(\mathbf{p}_k, y_k)\}_{k=1}^n$$

# Perceptron Camada Única

---

```

v ← 0;
enquanto v ≠ w faça
  | v ← w;
  | para  $k \in 1..n$  faça
  | |  $y'_k \leftarrow f(\sum(\tilde{\mathbf{p}}_k \cdot \mathbf{w}))$ ;
  | |  $\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot (y_k - y'_k) \cdot \tilde{\mathbf{p}}_k$ ;
  | fim
fim

```

---

# Outline

- 1 Redes Neurais Artificiais
- 2 Redes de Hopfield
- 3 Formulação Algébrica
- 4 Memória Associativa Birecional
- 5 Formulação Algébrica
- 6 Perceptron de Camada Única
- 7 Formulação Algébrica
- 8 Perceptron Multi-Camadas**
- 9 Formulação Algébrica

# Perceptron Multi-Camadas

Muitos problemas não são linearmente separáveis (por hiperplanos).

Perceptron retroalimentado contém uma camada de entrada e outra de saída de nós, além de uma ou mais camadas escondidas.

Vamos focar no caso de duas camadas (uma escondida).

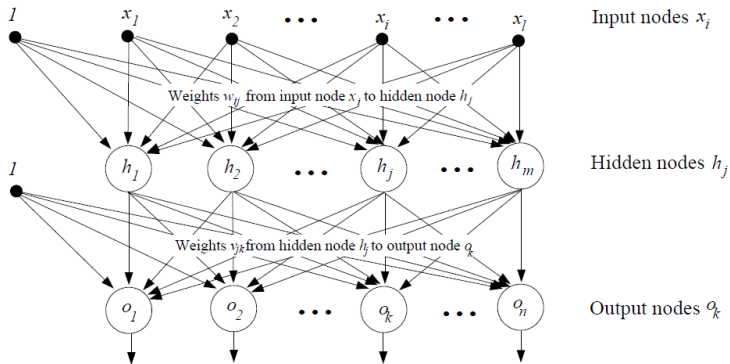
Um nó  $h_j$  na camada escondida se conecta a um nó  $x_i$  na camada de entrada com peso  $w_{ij}$  e a um nó  $o_k$  na camada de saída com peso  $v_{jk}$ .

O padrão a ser classificado é introduzido aos nós de entrada  $x_i, 1 \leq i \leq l$  e de lá os dados fluem em sentido único até os nós de saída  $o_k, 1 \leq k \leq n$ .

Nós de saída podem ter valor 0 ou 1 e portanto  $2^n$  classes podem ser expressas.



# Perceptron Multi-Camadas



# Perceptron Multi-Camadas

Algoritmo de classificação:

- 1 Apresentar o padrão  $\mathbf{p} = (p_1, p_2, \dots, p_l) \in \mathbb{R}^l$  ao perceptron:

$$x_i = p_i, 1 \leq i \leq l.$$

- 2 Calcular valores de nós na camada escondida por

$$h_j = \frac{1}{1 + e^{-(w_{0j} + \sum_{i=1}^l w_{ij} x_i)}}, 1 \leq j \leq m.$$

- 3 Calcular nós na camada de saída:

$$o_k = \frac{1}{1 + e^{-(v_{0k} + \sum_{j=1}^m v_{jk} h_j)}}, 1 \leq k \leq n.$$

- 4 Cada  $o_k$  é limiarizado por algum  $\tau$  apropriado à aplicação de modo que a classe do objeto de entrada seja representada por um vetor binário  $\mathbf{c} = (c_1, c_2, \dots, c_n)$ , em que

$$c_k = \chi_{\geq \tau}(o_k).$$

# Perceptron Multi-Camadas

Novamente, temos uma fase de aprendizado chamada de *aprendizado retropropagado*.

Usa um conjunto de treinamento  $\{(\mathbf{p}^t, \mathbf{c}^t)\}_{t=1}^s$ , em que  $\mathbf{p}^t \in \mathbb{R}^l$  é um padrão e  $\mathbf{c}^t \in \{0, 1\}^n$  é a classe verdadeira daquele padrão.

A ida (*forward*) processa o padrão de treinamento e o classifica. A volta (*backward*) faz correções nos pesos com base em medidas de erro.

# Perceptron Multi-Camadas

- 1 Inicia pesos com valores aleatórios entre  $-0.1$  e  $0.1$ :

$$w_{ij} = \text{choice}([-0.1, 0.1]), 0 \leq i \leq l, 1 \leq j \leq m$$

$$v_{jk} = \text{choice}([-0.1, 0.1]), 0 \leq j \leq m, 1 \leq k \leq n.$$

- 2 Apresenta  $\mathbf{p}^t = (p_1^t, p_2^t, \dots, p_l^t)$  do par de treinamento  $(\mathbf{p}^t, \mathbf{c}^t)$  e aplica passos 1, 2 e 3 do algoritmo de classificação.
- 3 Calcula erros  $\delta_{o_k}, 1 \leq k \leq n$  na camada de saída:

$$\delta_{o_k} = o_k(1 - o_k)(c_k^t - o_k),$$

em que  $\mathbf{c}^t = (c_1^t, \dots, c_n^t)$  é a classe correta de  $\mathbf{p}^t$  e  $(o_1, \dots, o_n)$  é a saída do perceptron.

- 4 Calcula erros  $\delta_{hj}, 1 \leq l \leq m$  na camada escondida:

$$\delta_{hj} = h_j(1 - h_j) \sum_{k=1}^n \delta_{o_k} \cdot v_{jk}.$$

# Perceptron Multi-Camadas

- 5 Seja  $v_{jk}(t)$  o valor do peso  $v_{jk}$  após apresentação do  $t$ -ésimo padrão de treinamento. Os pesos entre a camada escondida e de saída são ajustados por

$$v_{jk}(t) = v_{jk}(t - 1) + \eta \cdot \delta_{o_k} \cdot h_j,$$

em que  $0 < \eta \leq 1$  determina a taxa de aprendizado.

- 6 Ajusta os pesos entre a camada escondida e a de entrada:

$$w_{ij}(t) = w_{ij}(t - 1) + \eta \cdot \delta_{h_j} \cdot p_i^t.$$

- 7 Repete passos 2-6 para cada elemento do conjunto de treinamento. Cada ciclo é chamado de *época*.

# Outline

- 1 Redes Neurais Artificiais
- 2 Redes de Hopfield
- 3 Formulação Algébrica
- 4 Memória Associativa Birecional
- 5 Formulação Algébrica
- 6 Perceptron de Camada Única
- 7 Formulação Algébrica
- 8 Perceptron Multi-Camadas
- 9 Formulação Algébrica**

# Perceptron Multi-Camadas - Álgebra

Os valores na camada de entrada e saída são armazenados nas variáveis-imagem  $\mathbf{h} \in \{1\} \times \mathbb{R}^m$  e  $\mathbf{o} \in \mathbb{R}^n$ . Já os pesos entre a camada de entrada e a escondida são expressos no *template*  $\mathbf{w} \in (\mathbb{R}^{\mathbb{Z}_{l+1}})^{\mathbb{Z}_{m+1}}$ . Tal *template* é iniciado por

$$\mathbf{w}_0(0) = 1$$

$$\mathbf{w}_0(i) = 0, 1 \leq i \leq l$$

$$\mathbf{w}_j(i) = \text{choice}([-0.1, 0.1]), 0 \leq i \leq l, 1 \leq j \leq m.$$

O *template*  $\mathbf{v} \in (\mathbb{R}^{\mathbb{Z}_{m+1}})^{\mathbb{Z}_n}$  contém os pesos entre a camada escondida e a de saída e é iniciado por

$$\mathbf{v}_j(k) = \text{choice}([-0.1, 0.1]), 0 \leq j \leq m, 1 \leq k \leq n.$$

A função de ativação será

$$f(x) = \frac{1}{1 + e^{-x}}.$$

# Perceptron Multi-Camadas - Álgebra

A partir do conjunto de treinamento  $\{(\mathbf{p}^t, \mathbf{c}^t)\}_{t=1}^s$ , define-se

$$\tilde{\mathbf{p}}^t = (1, p_1^t, \dots, p_l^t)$$

e os *templates* parametrizados  $\mathbf{t}(\mathbf{d}, \mathbf{h}) \in (\mathbb{R}^{\mathbb{Z}_{m+1}})^{\mathbb{Z}_n}$  e  $\mathbf{u}(\mathbf{d}, \mathbf{p}) \in (\mathbb{R}^{\mathbb{Z}_{l+1}})^{\mathbb{Z}_{m+1}}$ :

$$\mathbf{t}(\mathbf{d}, \mathbf{h}) = \eta \cdot \mathbf{d}(k) \cdot \mathbf{h}(j), 0 \leq j \leq m, 1 \leq k \leq n$$

e

$$\mathbf{u}(\mathbf{d}, \mathbf{p})_0(i) = 0, 0 \leq i \leq l$$

$$\mathbf{u}(\mathbf{d}, \mathbf{p})_j(i) = \eta \cdot \mathbf{d}(j) \cdot \mathbf{p}(i), 0 \leq i \leq l, 1 \leq j \leq m.$$



# Perceptron Multi-Camadas - Álgebra

O algoritmo de aprendizado em cada época é o seguinte.

---



---

**para**  $t \in 1..s$  **faça**

$$\mathbf{h} \leftarrow f(\tilde{\mathbf{p}}^t \oplus \mathbf{w});$$

$$\mathbf{o} \leftarrow f(\mathbf{h} \oplus \mathbf{v});$$

$$\mathbf{d}_v \leftarrow \mathbf{o}(\mathbf{1} - \mathbf{o})(\mathbf{c}^t - \mathbf{o});$$

$$\mathbf{d}_w \leftarrow \mathbf{h}(\mathbf{1} - \mathbf{h})(\mathbf{d}_v \oplus \mathbf{v}');$$

$$\mathbf{v} \leftarrow \mathbf{v} + \mathbf{t}(\mathbf{d}_v, \mathbf{h});$$

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{u}(\mathbf{d}_w, \tilde{\mathbf{p}}^t);$$

**fim**

---

# Perceptron Multi-Camadas - Álgebra

Para classificar um padrão  $\mathbf{p} = (p_1, \dots, p_l)$ , definimos  $\tilde{\mathbf{p}} = (1, p_1, \dots, p_l)$  e a classe obtida será representada na imagem  $\mathbf{c}$  gerada pelo seguinte algoritmo:

---

$$\mathbf{h} \leftarrow f(\tilde{\mathbf{p}} \oplus \mathbf{w});$$

$$\mathbf{o} \leftarrow f(\mathbf{h} \oplus \mathbf{v});$$

$$\mathbf{c} \leftarrow \chi_{\geq \tau}(\mathbf{o});$$

---