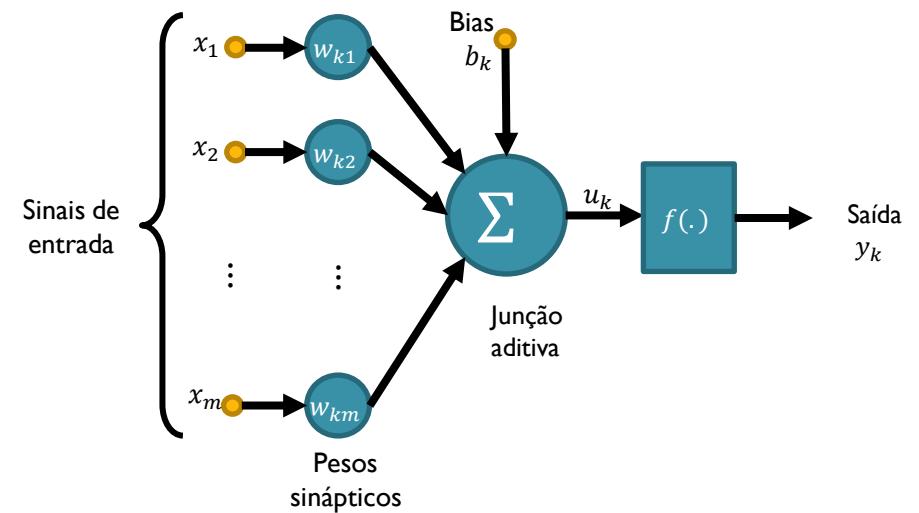
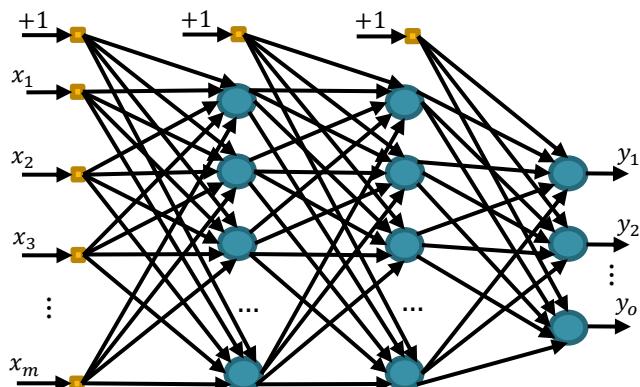
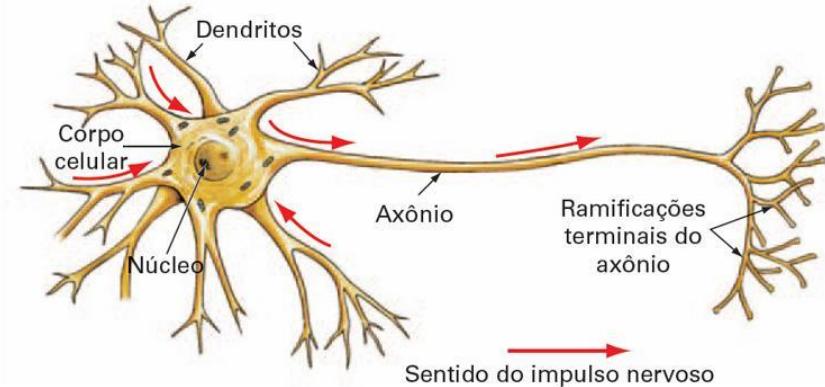
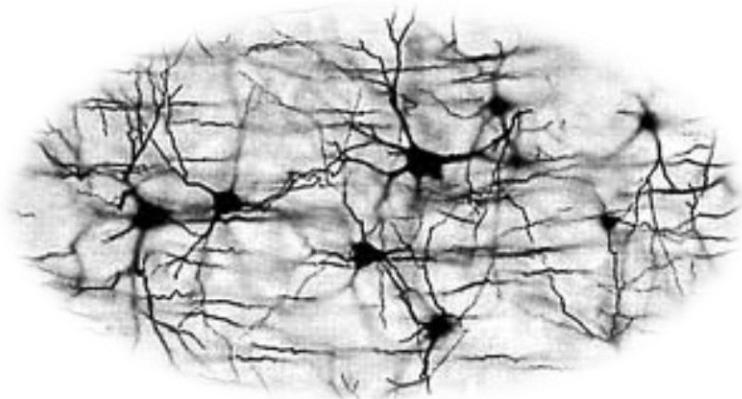


Redes Neurais Artificiais

Parte 2



Fabricio Breve – fabricio.breve@unesp.br



Redes Neurais Artificiais

PERCEPTRON DE UMA ÚNICA CAMADA

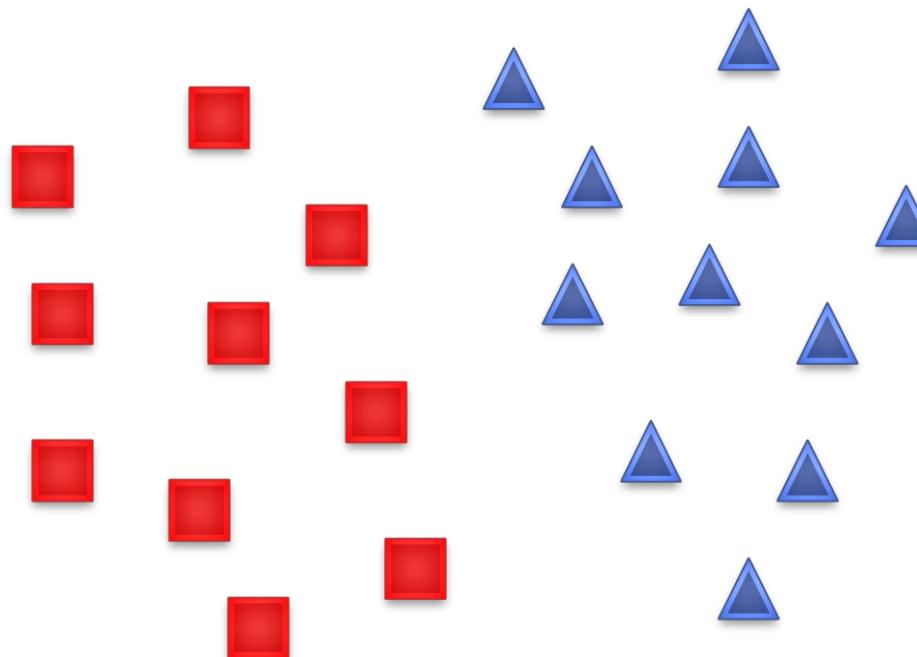
Perceptron

- Desenvolvida por Frank Rosenblatt em 1958.
 - Psicólogo americano notável por seu trabalho com inteligência artificial.
- Utiliza modelo de McCulloch-Pitts como neurônio.
- Rede mais simples para classificação de padrões linearmente separáveis.
- Primeiro modelo de aprendizado supervisionado.

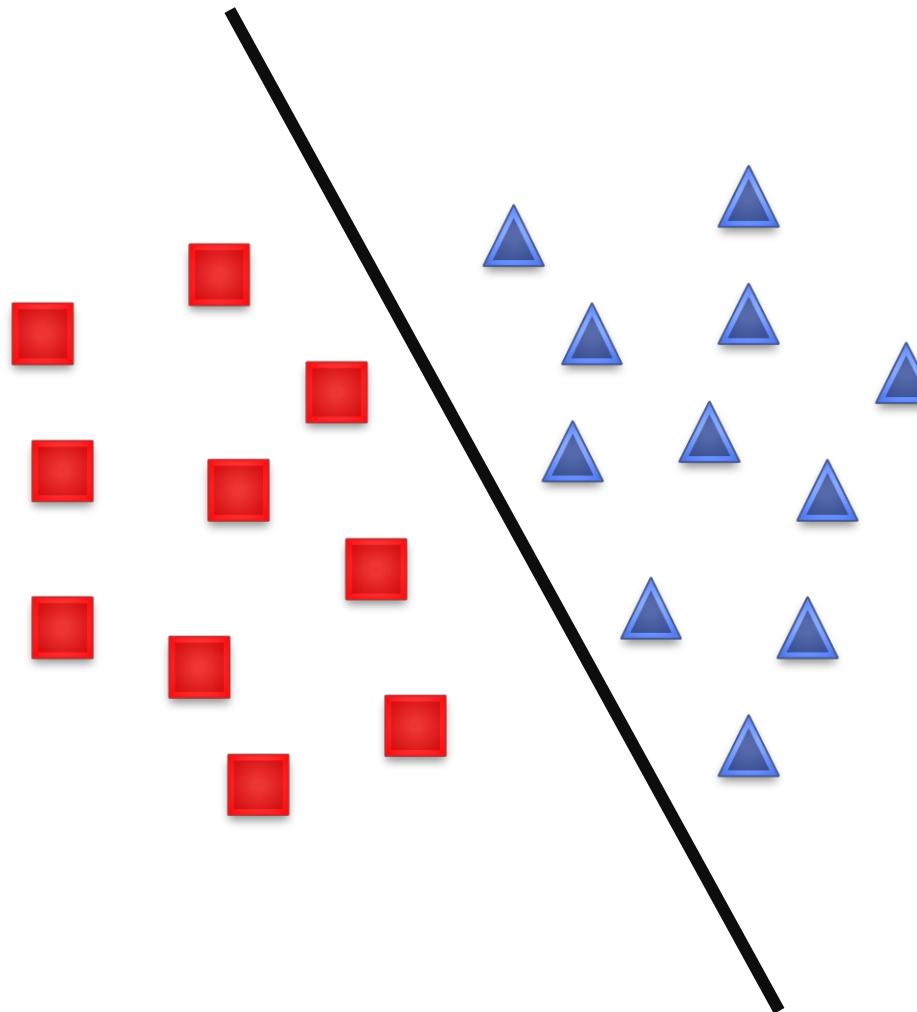


Frank Rosenblatt [*1928 – †1971]

Separabilidade Linear



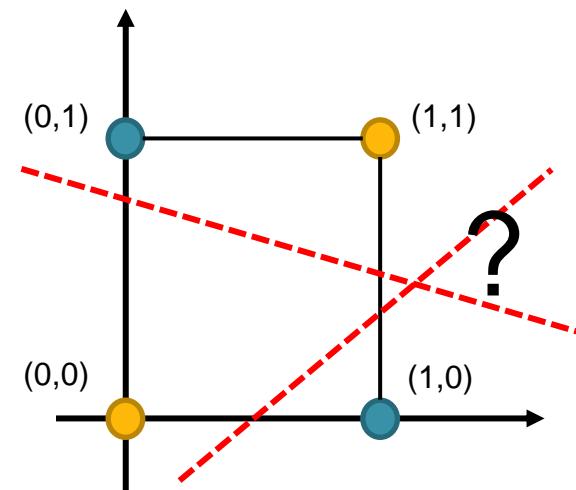
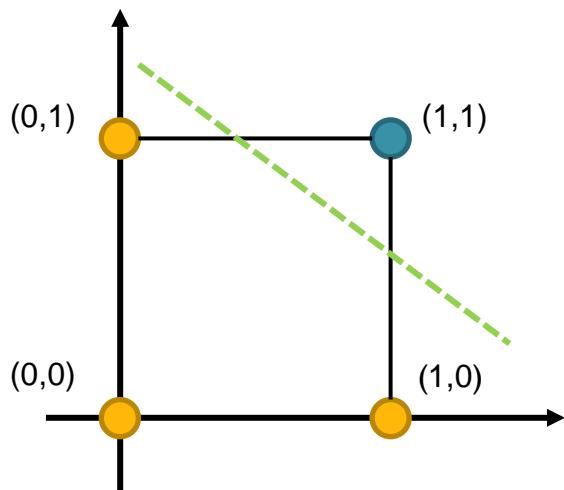
Separabilidade Linear



Separabilidade Linear

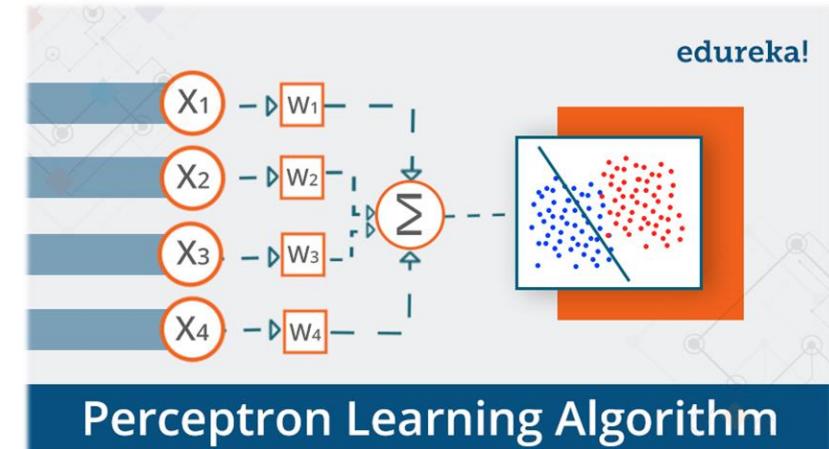
Entradas		Saída
x_1	x_2	$x_1 \text{ AND } x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Entradas		Saída
x_1	x_2	$x_1 \text{ XOR } x_2$
0	0	0
0	1	1
1	0	1
1	1	0



Perceptron Simples para Classificação de Padrões

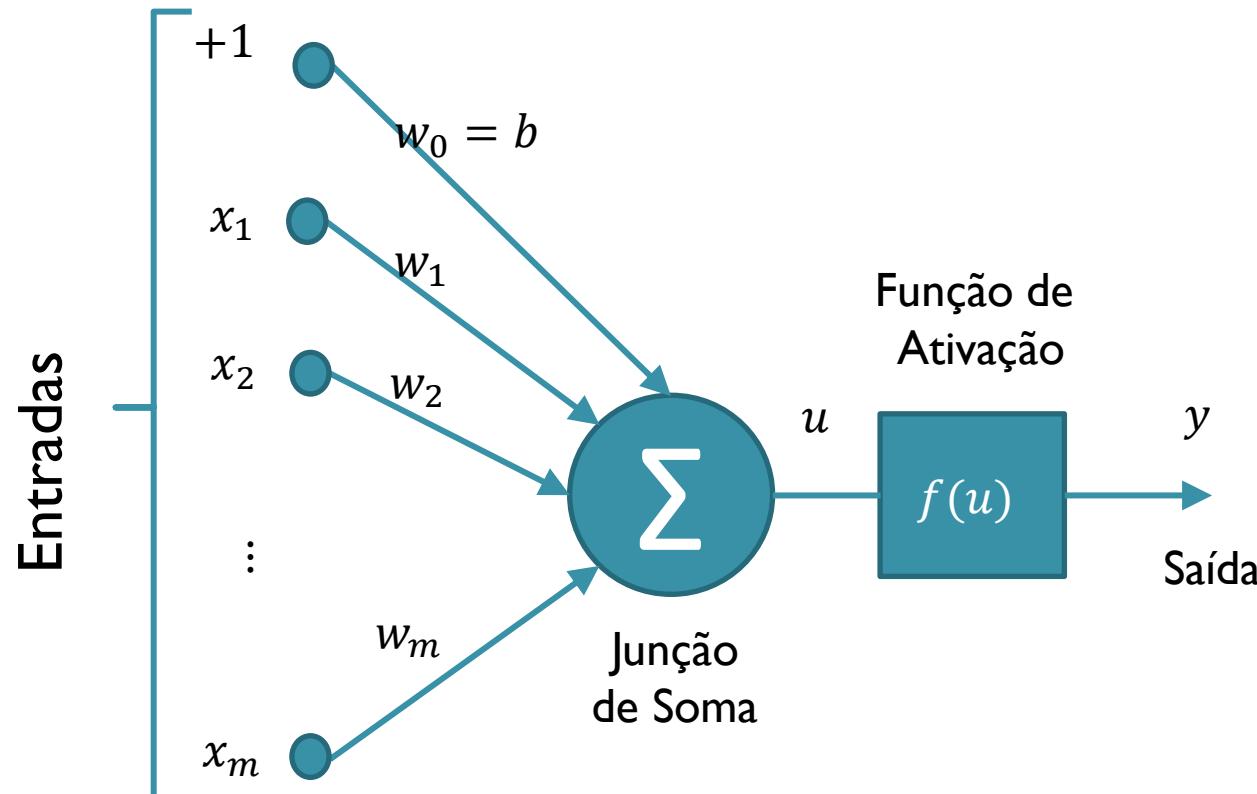
- Uma única camada de neurônios com pesos sinápticos ajustáveis e *bias*.
- Sob condições adequadas converge para o conjunto de pesos adequado.
 - Particularmente se padrões de treinamento pertencem a classes linearmente separáveis.
 - Prova de convergência: *Teorema de Convergência do Perceptron*.



Perceptron Simples para Classificação de Padrões

- Neurônios similares aos de McCulloch e Pitts com função de ativação de limiar
 - Mas incluindo um *bias*
- Principal contribuição de Rosenblatt
 - Regra de aprendizado para treinar os perceptrons para resolver problemas de reconhecimento e classificação de padrões

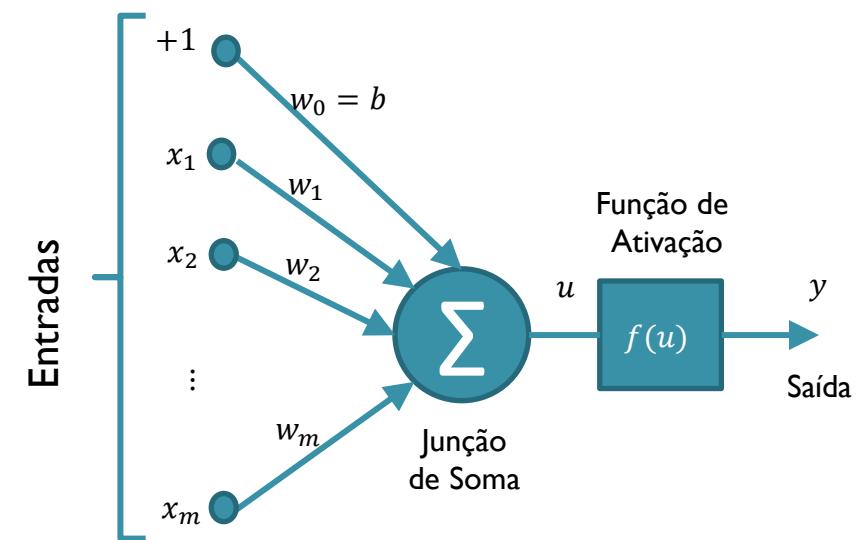
Perceptron Simples para Classificação de Padrões



Perceptron mais simples para realizar classificação de padrões

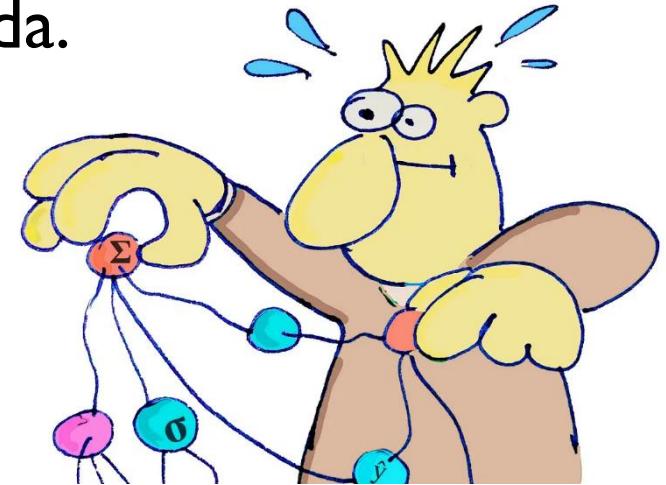
Treinamento

- Para cada padrão de entrada \mathbf{x}_i calcule a saída da rede y_i
 - Se não for a resposta esperada calcule o erro:
 - A saída esperada é conhecida (treinamento supervisionado).
 - $e_i = d_i - y_i$
 - Atualizar pesos de acordo com as regras:
 - $\mathbf{w}(t + 1) = \mathbf{w}(t) + \alpha e_i \mathbf{x}_i$
 - $b(t + 1) = b(t) + \alpha e_i$
 - onde $\mathbf{w} \in \mathbb{R}^{1 \times m}$, $\mathbf{x}_i \in \mathbb{R}^{1 \times m}$, e $b \in \mathbb{R}^{1 \times 1}$
 - α é a taxa de aprendizado.



Algoritmo de treinamento

- $\mathbf{X} \in \Re^{N \times m}$ é a matriz de N padrões de entrada.
 - Cada padrão tem m dimensões.
 - Cada padrão é uma linha de \mathbf{X} .
- \mathbf{d} é o vetor de saídas desejadas.
- $f()$ é uma função de limiar.
 - -1 e 1 , 0 e 1 , etc...
- Critério de parada:
 - Número fixo de iterações max_it
 - Soma E dos erros quadráticos, de cada padrão de entrada, igual a zero ($E = 0$).
- Algoritmo retorna o vetor de pesos \mathbf{w}
- α é a taxa de aprendizado.



Algoritmo de treinamento

```
procedimento [w] = perceptron(max_it, $\alpha$ ,x,d)
    inicializar w    // ajuste para zero ou pequenos valores aleatórios
    inicializar b    // ajuste para zero ou um pequeno valor aleatório
    t  $\leftarrow$  1;
    E  $\leftarrow$  1;
    enquanto t  $\leq$  max_it & E  $>$  0 faça
        E  $\leftarrow$  0;
        para i de 1 até N faça                                // para cada padrão de treinamento
             $y_i \leftarrow f(\mathbf{w}\mathbf{x}_i^T + b)$                       // saída da rede para  $\mathbf{x}_i$ 
             $e_i \leftarrow d_i - y_i$                                     // determinar o erro para  $\mathbf{x}_i$ 
             $\mathbf{w} \leftarrow \mathbf{w} + \alpha e_i \mathbf{x}_i$                   // atualizar o vetor de pesos
            b  $\leftarrow$  b +  $\alpha e_i$                                 // atualizar o termo de bias
            E  $\leftarrow$  E +  $e_i^2$                                     // acumular o erro
        fim-para
        t  $\leftarrow$  t + 1
    fim-enquanto
fim-procedimento
```

Algoritmo de aprendizado de um perceptron simples. A função $f(\cdot)$ é uma função de limiar, e a saída desejada é ‘1’ se um padrão pertence à classe ou ‘−1’ (ou ‘0’) se ele não pertence à classe.

Perceptron de Múltiplas Saídas para Classificação de Padrões

- A regra de aprendizado pode ser facilmente estendida para lidar com mais de um neurônio de saída
- Neste caso o vetor y com a saída de cada neurônio é dado por:

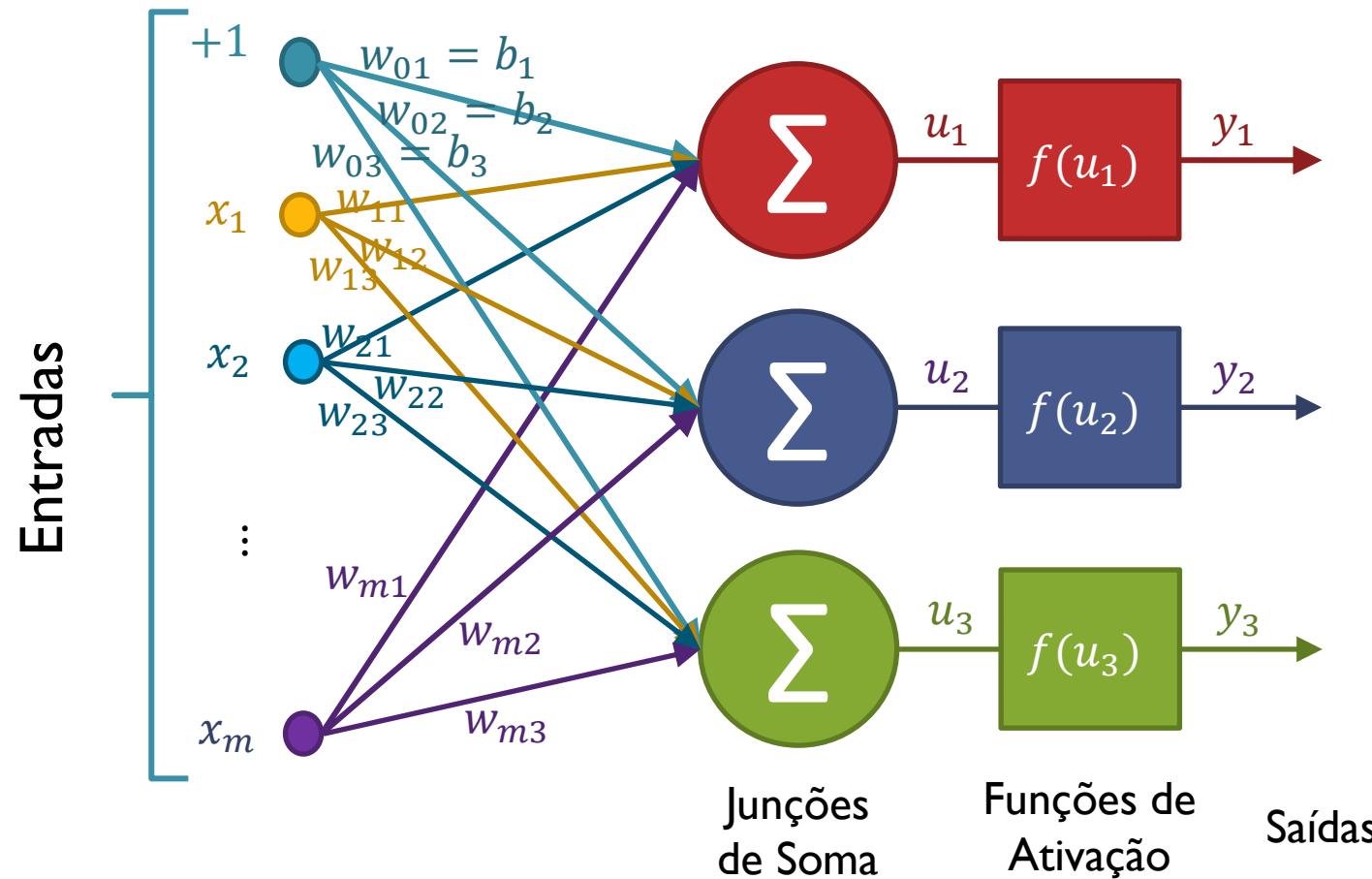
- $$y = f(Wx_i^T + b)$$

- onde $W \in \mathbb{R}^{o \times m}$, $x_i \in \mathbb{R}^{1 \times m}$, $y_i \in \mathbb{R}^{o \times 1}$, e $b \in \mathbb{R}^{o \times 1}$

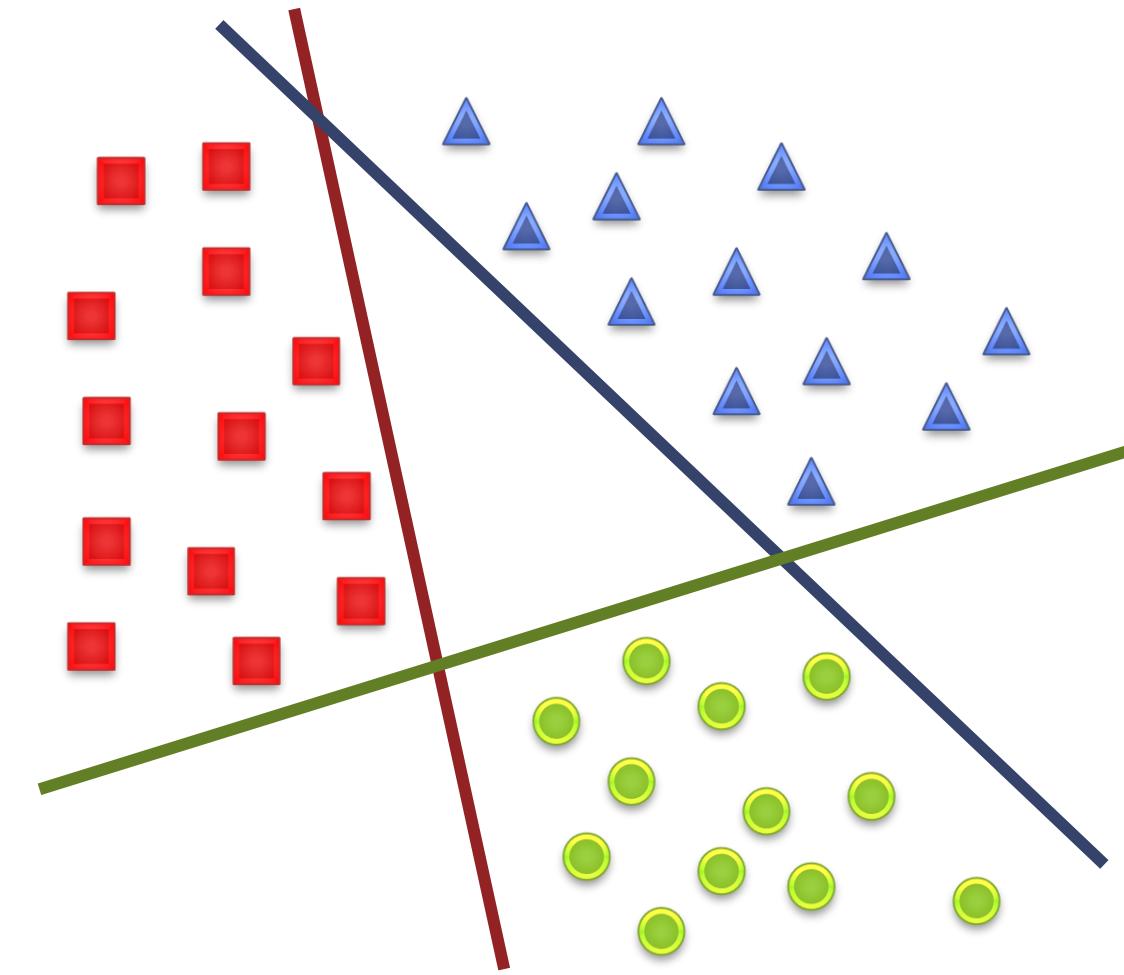
Perceptron de Múltiplas Saídas para Classificação de Padrões

$$\mathbf{y} = f(\mathbf{W}\mathbf{x}_i^T + \mathbf{b})$$

onde $\mathbf{W} \in \mathbb{R}^{o \times m}$, $\mathbf{x}_i \in \mathbb{R}^{1 \times m}$, $y_i \in \mathbb{R}^{o \times 1}$, e $\mathbf{b} \in \mathbb{R}^{o \times 1}$



Perceptron de Múltiplas Saídas para Classificação de Padrões



Algoritmo de Treinamento (Perceptron com múltiplas saídas)

- D é a matriz de saídas desejadas
 - Cada coluna é a saída para um dos padrões de entrada
- Vetor de erro é calculado subtraindo o vetor de saída do vetor de saída esperada
 - $e_i = d_i - y_i$
- Se os padrões forem linearmente separáveis o algoritmo converge

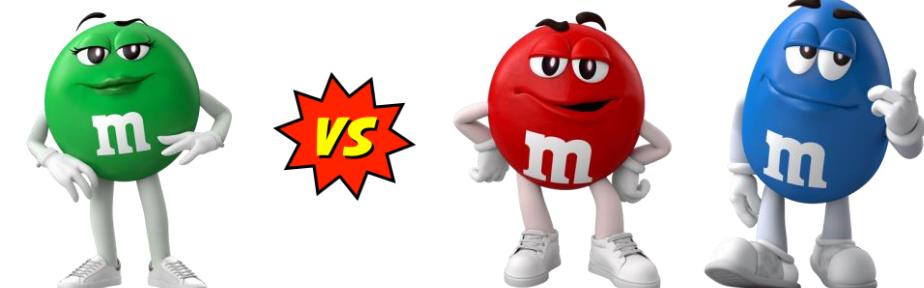
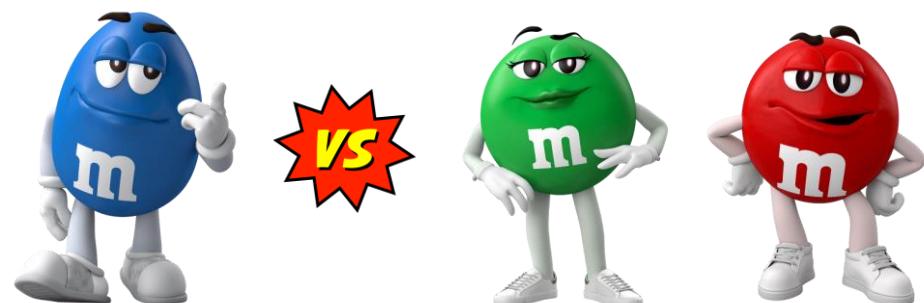
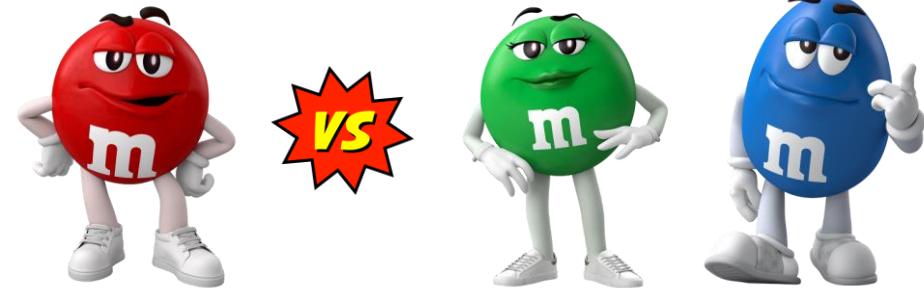
Algoritmo de Treinamento (Perceptron com múltiplas saídas)

```
procedimento [W] = perceptron(max_it, $\alpha$ ,X,D)
    inicializar W // ajuste para zero ou pequenos valores aleatórios
    inicializar b // ajuste para zero ou pequenos valores aleatórios
    t  $\leftarrow$  1;
    E  $\leftarrow$  1;
    enquanto t  $\leq$  max_it & E  $>$  0 faça
        E  $\leftarrow$  0;
        para i de 1 até N faça // para cada padrão de treinamento
            yi  $\leftarrow$  f(WxiT + b) // saída da rede para xi
            ei  $\leftarrow$  di - yi // determinar o erro para xi
            W  $\leftarrow$  W +  $\alpha \mathbf{e}_i \mathbf{x}_i$  // atualizar o vetor de pesos
            b  $\leftarrow$  b +  $\alpha \mathbf{e}_i$  // atualizar o termo de bias
            E  $\leftarrow$  E + soma(eji2) // j = 1, ..., o
        fim-para
        t  $\leftarrow$  t + 1
    fim-enquanto
fim-procedimento
```

Algoritmo de aprendizado de um perceptron de múltiplas saídas. A função $f(\cdot)$ é uma função de limiar, e a saída desejada é ‘1’ se um padrão pertence à classe ou ‘−1’ (ou ‘0’) se ele não pertence à classe. Note que \mathbf{e}_i , \mathbf{d}_i , \mathbf{y}_i , e \mathbf{b} são vetores e \mathbf{W} é uma matriz. e_{ji} corresponde ao erro do neurônio j quando recebe o padrão de entrada i .

Um-versus-resto

- Em problemas com múltiplas saídas temos neurônios que classificam *um-versus-resto*.
 - Exemplo:
 - vermelho X verde e azul
 - azul X verde e vermelho
 - verde X vermelho e azul
 - Nesse caso, usar função de limiar pode levar à ambiguidades, com mais de um neurônio disparando ao mesmo tempo.
 - Como resolver este problema?



Função Softmax

- Usar valores contínuos de saída pode ser uma solução
 - Podem ser interpretados como um índice de confiança na resposta.
- Uma estratégia comum é usar uma função de ativação do tipo *softmax* na camada de saída.
 - $y_j = f(u_j) = \frac{\exp u_j}{\sum_{k=1}^K \exp u_k}$ onde $j = 1, \dots, K$
 - u_j é o somatório das entradas em y_j antes da função de ativação.
 - Dessa forma a saída dos neurônios fica contínua, no intervalo $[0, 1]$, com soma 1.
 - Cada exemplo é classificado pelo neurônio que apresentou maior valor de saída.

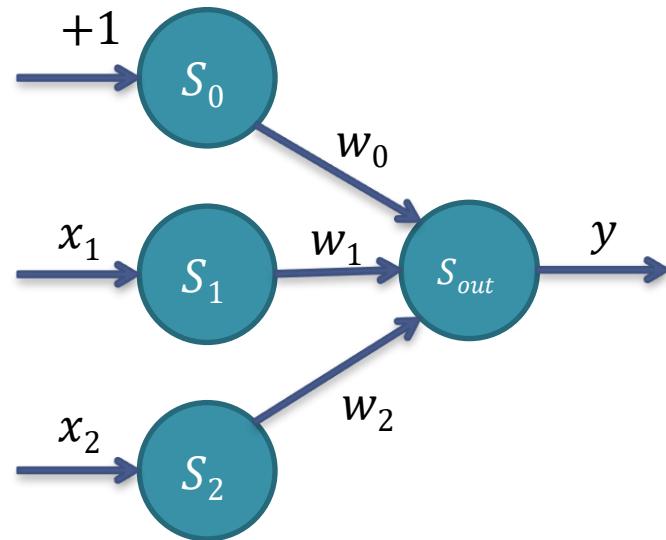
Exemplo de Aplicação

AND	x_0	x_1	x_2	d
Entrada 1:	1	0	0	0
Entrada 2:	1	0	1	0
Entrada 3:	1	1	0	0
Entrada 4:	1	1	1	1

Peso inicial: $w_0 = 0, w_1 = 0, w_2 = 0$
Taxa de aprendizado: $\alpha = 0,5$

Função de ativação de limiar:

$$\begin{aligned}s_{out} > 0 &\Rightarrow y = 1 \\ s_{out} \leq 0 &\Rightarrow y = 0\end{aligned}$$

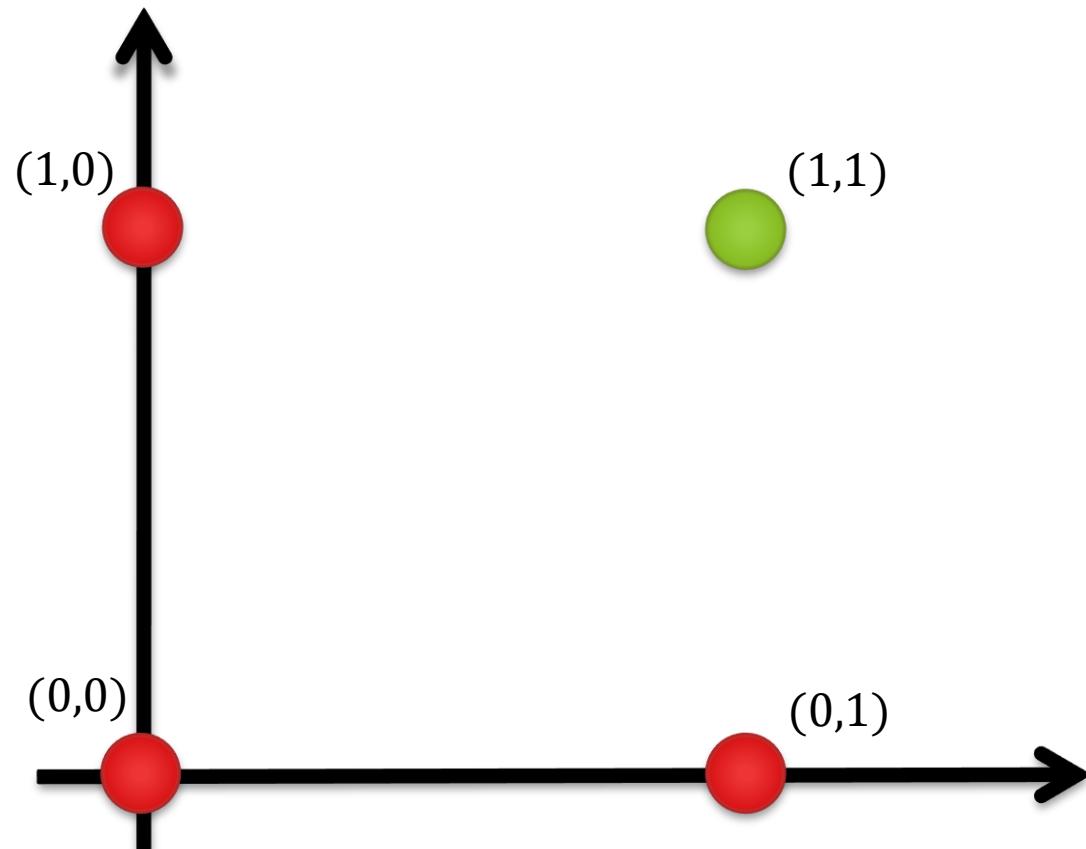


Lembre-se:
 $x_0 = b$

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Saída Desejada



Iº Ciclo

- Entrada 1:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(0 \times 1 + 0 \times 0 + 0 \times 0) = f(0) = 0 \Rightarrow y = d$$

- Entrada 2:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(0 \times 1 + 0 \times 0 + 0 \times 1) = f(0) = 0 \Rightarrow y = d$$

- Entrada 3:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(0 \times 1 + 0 \times 1 + 0 \times 0) = f(0) = 0 \Rightarrow y = d$$

- Entrada 4:

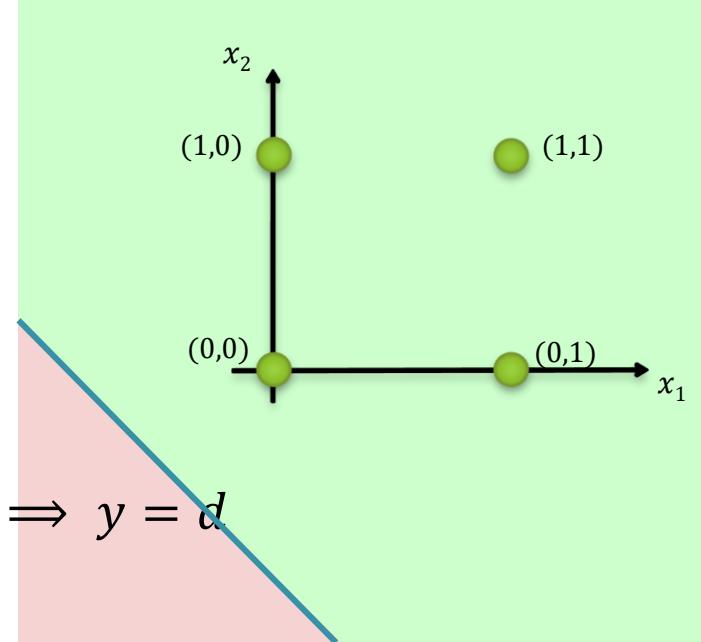
$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(0 \times 1 + 0 \times 1 + 0 \times 1) = f(0) = 0 \Rightarrow y \neq d$$

$$w_0 = w_0 + \alpha(d - y)x_0 = 0 + 0,5 \times (1 - 0) \times 1 = 0,5$$

$$w_1 = w_1 + \alpha(d - y)x_1 = 0 + 0,5 \times (1 - 0) \times 1 = 0,5$$

$$w_2 = w_2 + \alpha(d - y)x_2 = 0 + 0,5 \times (1 - 0) \times 1 = 0,5$$



2º Ciclo

- Entrada 1:

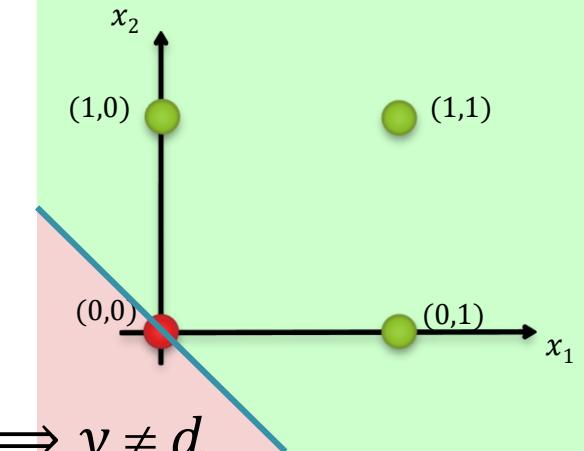
$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(0,5 \times 1 + 0,5 \times 0 + 0,5 \times 0) = f(0,5) = 1 \Rightarrow y \neq d$$

$$w_0 = w_0 + \alpha(d - y)x_0 = 0,5 + 0,5 \times (0 - 1) \times 1 = 0$$

$$w_1 = w_1 + \alpha(d - y)x_1 = 0,5 + 0,5 \times (0 - 1) \times 0 = 0,5$$

$$w_2 = w_2 + \alpha(d - y)x_2 = 0,5 + 0,5 \times (0 - 1) \times 0 = 0,5$$



- Entrada 2:

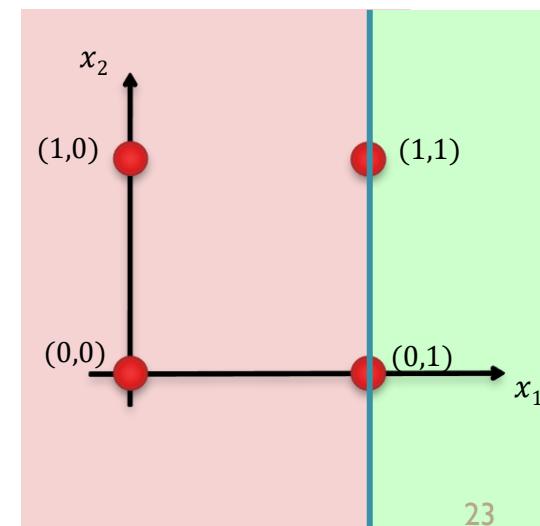
$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(0 \times 1 + 0,5 \times 0 + 0,5 \times 1) = f(0,5) = 1 \Rightarrow y \neq d$$

$$w_0 = w_0 + \alpha(d - y)x_0 = 0 + 0,5 \times (0 - 1) \times 1 = -0,5$$

$$w_1 = w_1 + \alpha(d - y)x_1 = 0,5 + 0,5 \times (0 - 1) \times 0 = 0,5$$

$$w_2 = w_2 + \alpha(d - y)x_2 = 0,5 + 0,5 \times (0 - 1) \times 0 = 0$$



2º Ciclo

- Entrada 3:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-0,5 \times 1 + 0,5 \times 1 + 0 \times 0) = f(0) = 0 \Rightarrow y = d$$

- Entrada 4:

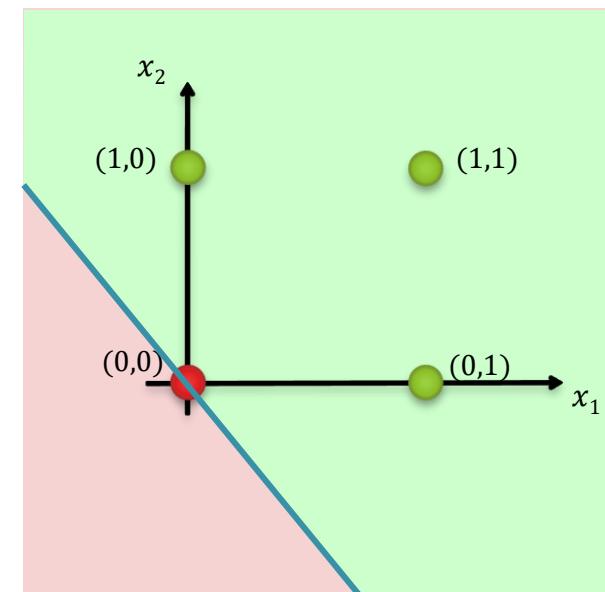
$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-0,5 \times 1 + 0,5 \times 1 + 0 \times 1) = f(0) = 0 \Rightarrow y \neq d$$

$$w_0 = w_0 + \alpha(d - y)x_0 = -0,5 + 0,5 \times (1 - 0) \times 1 = 0$$

$$w_1 = w_1 + \alpha(d - y)x_1 = 0,5 + 0,5 \times (1 - 0) \times 1 = 1$$

$$w_2 = w_2 + \alpha(d - y)x_2 = 0 + 0,5 \times (1 - 0) \times 1 = 0,5$$



3º Ciclo

- Entrada 1:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(0 \times 1 + 1 \times 0 + 0,5 \times 0) = f(0) = 0 \Rightarrow y = d$$

- Entrada 2:

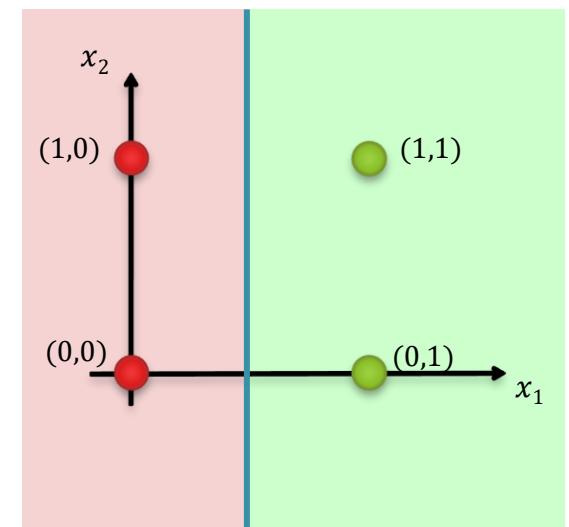
$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(0 \times 1 + 1 \times 0 + 0,5 \times 1) = f(0,5) = 1 \Rightarrow y \neq d,$$

$$w_0 = w_0 + \alpha(d - y) \quad x_0 = 0 + 0,5 \times (0 - 1) \times 1 = -0,5$$

$$w_1 = w_1 + \alpha(d - y) \quad x_1 = 1 + 0,5 \times (0 - 1) \times 0 = 1$$

$$w_2 = w_2 + \alpha(d - y) \quad x_2 = 0,5 + 0,5 \times (0 - 1) \times 1 = 0$$



3º Ciclo

- Entrada 3:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-0,5 \times 1 + 1 \times 1 + 0 \times 0) = f(0,5) = 1 \Rightarrow y \neq d$$

$$w_0 = w_0 + \alpha(d - y)x_0 = -0,5 + 0,5 \times (0 - 1) \times 1 = -1$$

$$w_1 = w_1 + \alpha(d - y)x_1 = 1 + 0,5 \times (0 - 1) \times 1 = 0,5$$

$$w_2 = w_2 + \alpha(d - y)x_2 = 0 + 0,5 \times (0 - 1) \times 0 = 0$$

- Entrada 4:

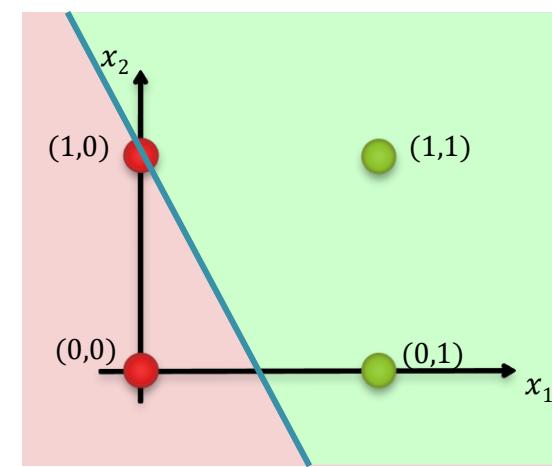
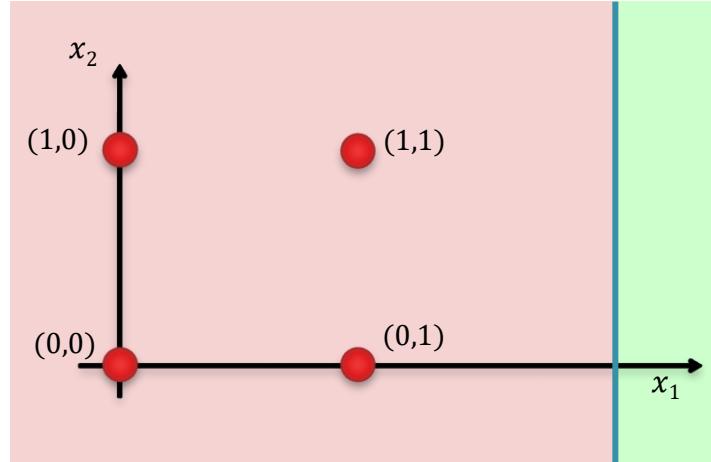
$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-1 \times 1 + 0,5 \times 1 + 0 \times 1) = f(-0,5) = 0 \Rightarrow y \neq d$$

$$w_0 = w_0 + \alpha(d - y)x_0 = -1 + 0,5 \times (1 - 0) \times 1 = -0,5$$

$$w_1 = w_1 + \alpha(d - y)x_1 = 0,5 + 0,5 \times (1 - 0) \times 1 = 1$$

$$w_2 = w_2 + \alpha(d - y)x_2 = 0 + 0,5 \times (1 - 0) \times 1 = 0,5$$



4º Ciclo

- Entrada 1:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-0,5 \times 1 + 1 \times 0 + 0,5 \times 0) = f(-0,5) = 0 \Rightarrow y = d$$

- Entrada 2:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-0,5 \times 1 + 1 \times 0 + 0,5 \times 1) = f(0) = 0 \Rightarrow y = d$$

4º Ciclo

- Entrada 3:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-0,5 \times 1 + 1 \times 1 + 0,5 \times 0) = f(0,5) = 1 \Rightarrow y \neq d$$

$$w_0 = w_0 + \alpha(d - y)x_0 = -0,5 + 0,5 \times (0 - 1) \times 1 = -1$$

$$w_1 = w_1 + \alpha(d - y)x_1 = 1 + 0,5 \times (0 - 1) \times 1 = 0,5$$

$$w_2 = w_2 + \alpha(d - y)x_2 = 0,5 + 0,5 \times (0 - 1) \times 0 = 0,5$$

- Entrada 4:

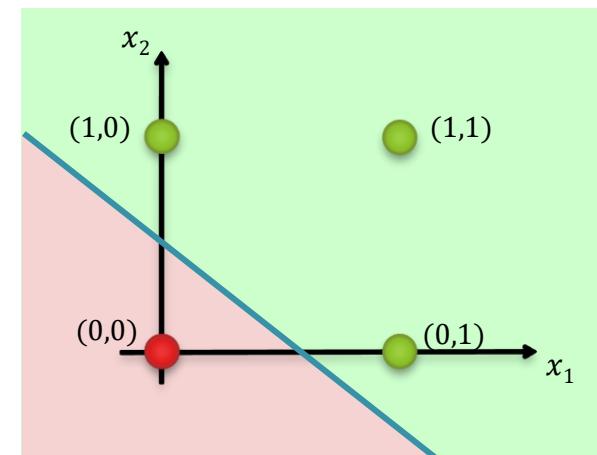
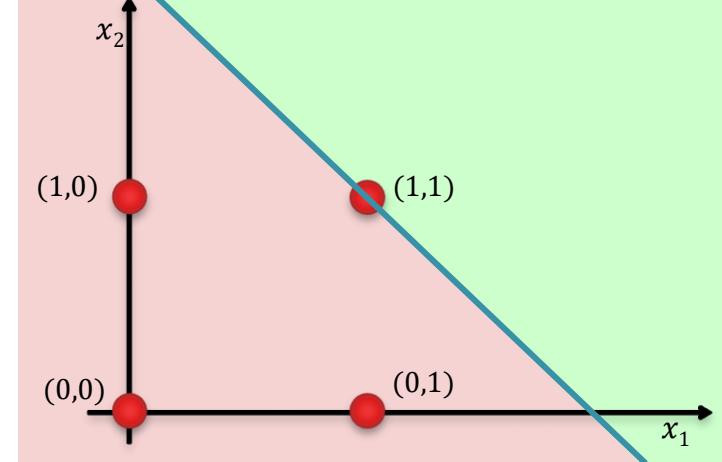
$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-1 \times 1 + 0,5 \times 1 + 0,5 \times 1) = f(0) = 0 \Rightarrow y \neq d$$

$$w_0 = w_0 + \alpha(d - y)x_0 = -1 + 0,5 \times (1 - 0) \times 1 = -0,5$$

$$w_1 = w_1 + \alpha(d - y)x_1 = 0,5 + 0,5 \times (1 - 0) \times 1 = 1$$

$$w_2 = w_2 + \alpha(d - y)x_2 = 0,5 + 0,5 \times (1 - 0) \times 1 = 1$$



5º Ciclo

- Entrada 1:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-0,5 \times 1 + 1 \times 0 + 1 \times 0) = f(-0,5) = 0 \Rightarrow y = d$$

- Entrada 2:

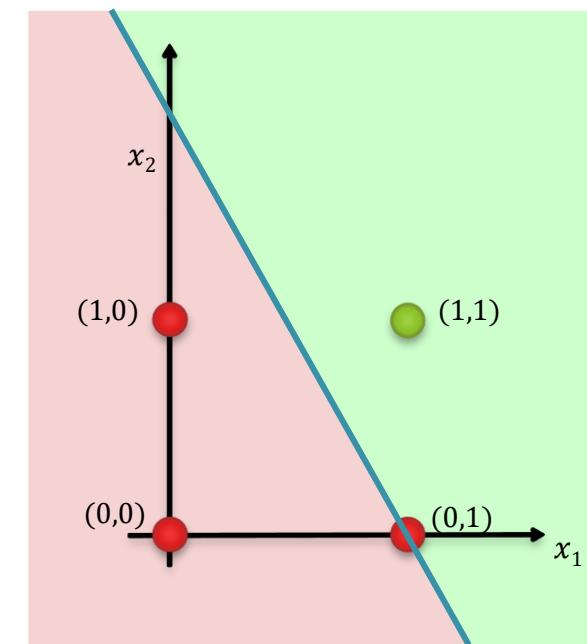
$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-0,5 \times 1 + 1 \times 0 + 1 \times 1) = f(0,5) = 1 \Rightarrow y \neq d$$

$$w_0 = w_0 + \alpha(d - y) \quad x_0 = -0,5 + 0,5 \times (0 - 1) \times 1 = -1$$

$$w_1 = w_1 + \alpha(d - y) \quad x_1 = 1 + 0,5 \times (0 - 1) \times 0 = 1$$

$$w_2 = w_2 + \alpha(d - y) \quad x_2 = 1 + 0,5 \times (0 - 1) \times 1 = 0,5$$



5º Ciclo

- Entrada 3:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-1 \times 1 + 1 \times 1 + 0,5 \times 0) = f(0) = 0 \Rightarrow y = d$$

- Entrada 4:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-1 \times 1 + 1 \times 1 + 0,5 \times 1) = f(0,5) = 1 \Rightarrow y = d$$

6º Ciclo

- Entrada 1:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-1 \times 1 + 1 \times 0 + 0,5 \times 0) = f(-1) = 0 \Rightarrow y = d$$

- Entrada 2:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-1 \times 1 + 1 \times 0 + 0,5 \times 1) = f(-0,5) = 0 \Rightarrow y = d$$

- Entrada 3:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

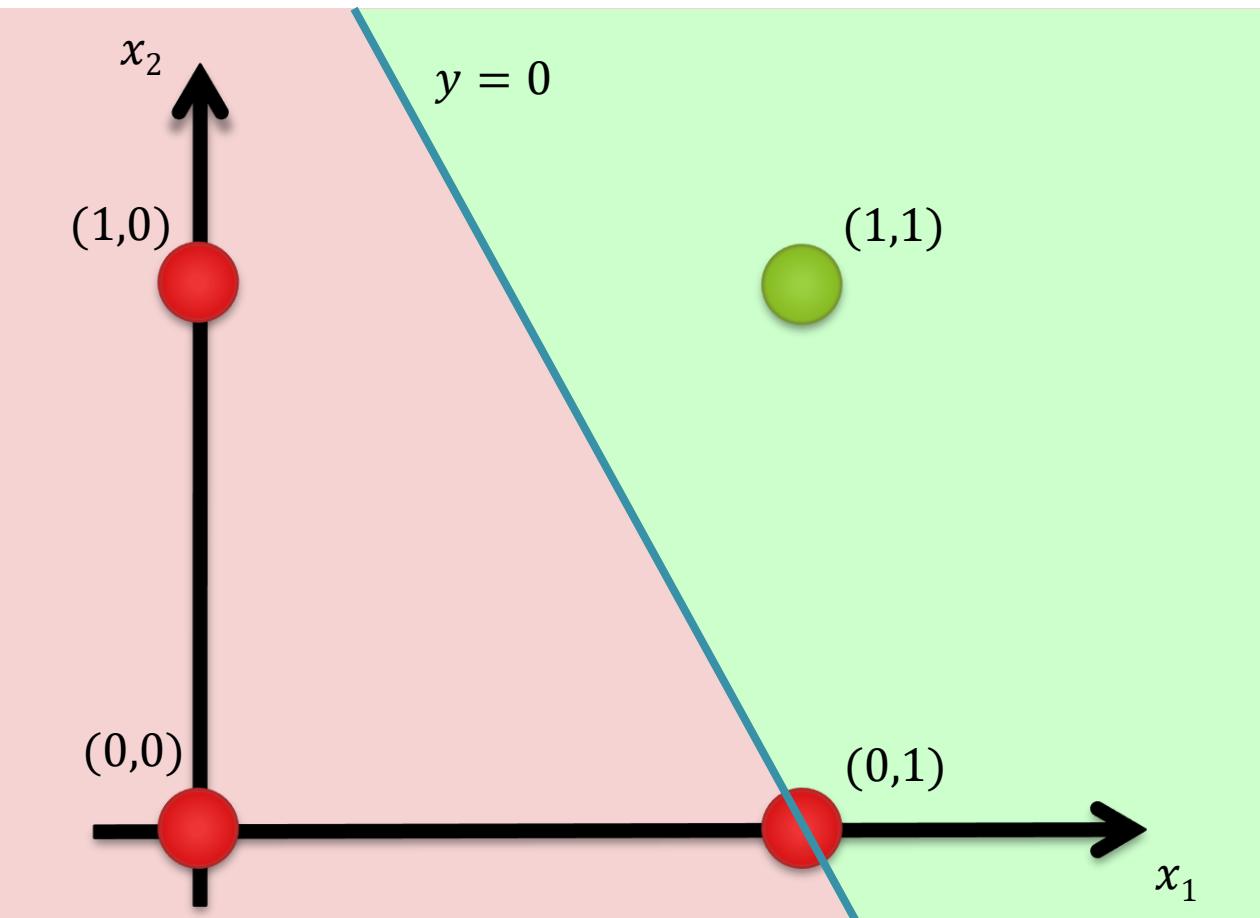
$$y = f(-1 \times 1 + 1 \times 1 + 0,5 \times 0) = f(0) = 0 \Rightarrow y = d$$

- Entrada 4:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(-1 \times 1 + 1 \times 1 + 0,5 \times 1) = f(0,5) = 1 \Rightarrow y = d$$

Pesos Finais



$$y = f(w_1 x_1 + w_2 x_2 + b)$$
$$y = f(1x_1 + 0,5x_2 - 1)$$

Exercício

- Dado o seguinte cadastro de pacientes:



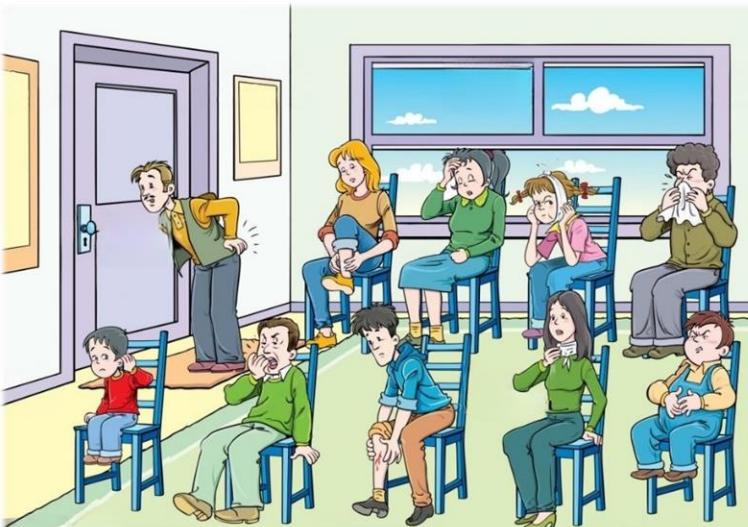
Nome	Febre	Enjoo	Manchas	Dores	Diagnóstico
João	sim	sim	pequenas	sim	doente
Pedro	não	não	grandes	não	saudável
Maria	sim	sim	pequenas	não	saudável
José	sim	não	grandes	sim	doente
Ana	sim	não	pequenas	sim	saudável
Leila	não	não	grandes	sim	doente

- Ensine uma rede Perceptron a distinguir:
 - Paciente saudáveis
 - Pacientes doentes

Exercício

- Testar a rede para novos casos:

Nome	Febre	Enjoo	Manchas	Dores	Diagnóstico
Luis	não	não	pequenas	sim	?
Laura	sim	sim	grandes	sim	?



Generalização

- Classificação correta de padrões não utilizados no treinamento ou com ruído.
 - Ocorre através da detecção de características relevantes do padrão de entrada durante o treinamento.
 - Padrões desconhecidos são atribuídos a classes cujos padrões apresentam características semelhantes.
- Garante tolerância a falhas.



Dados de Treinamento



Dados de Teste

<https://www.kdnuggets.com/2019/11/generalization-neural-networks.html>



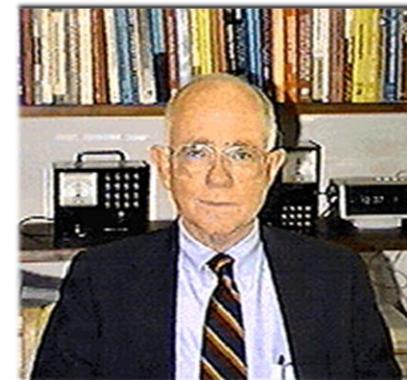
Redes Neurais Artificiais



ADALINE

ADALINE

- Widrow e Hoff, 1960
 - Introduzido quase ao mesmo tempo que o Perceptron.
- ADAptive LInear NEuron
 - Função de ativação linear em vez de limiar.
- Também restrito a problemas linearmente separáveis.



Bernard Widrow [*1929]

Bernard Widrow é um engenheiro americano e professor de engenharia elétrica na Universidade de Stanford.

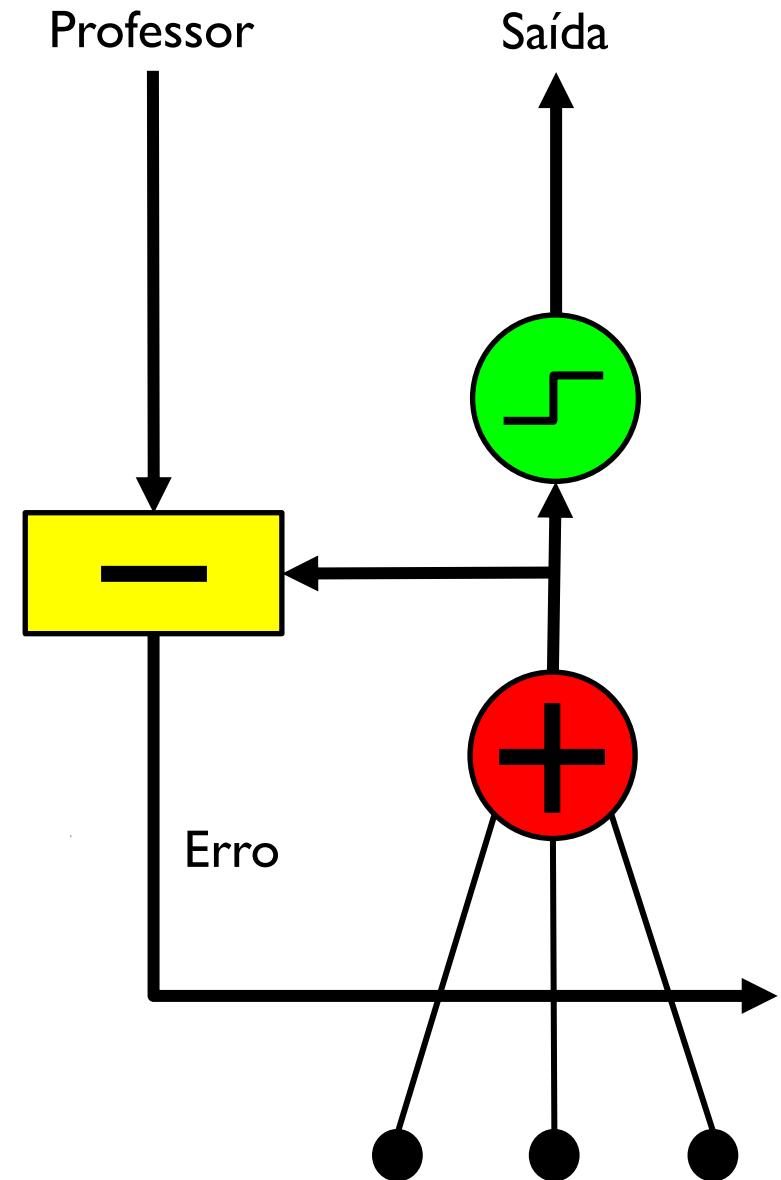


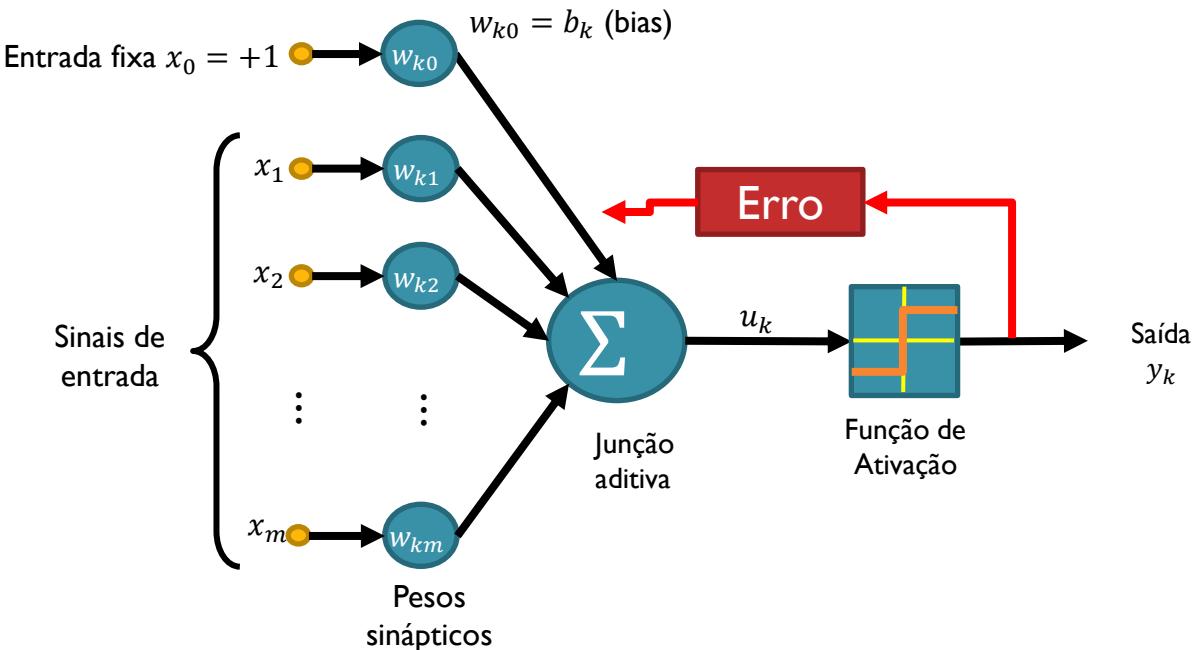
Marcian Hoff [*1937]

Marcian Edward "Ted" Hoff, Jr. também é um engenheiro americano e foi o primeiro aluno de doutorado de Widrow. Trabalhou na Intel onde foi um dos inventores do microprocessador.

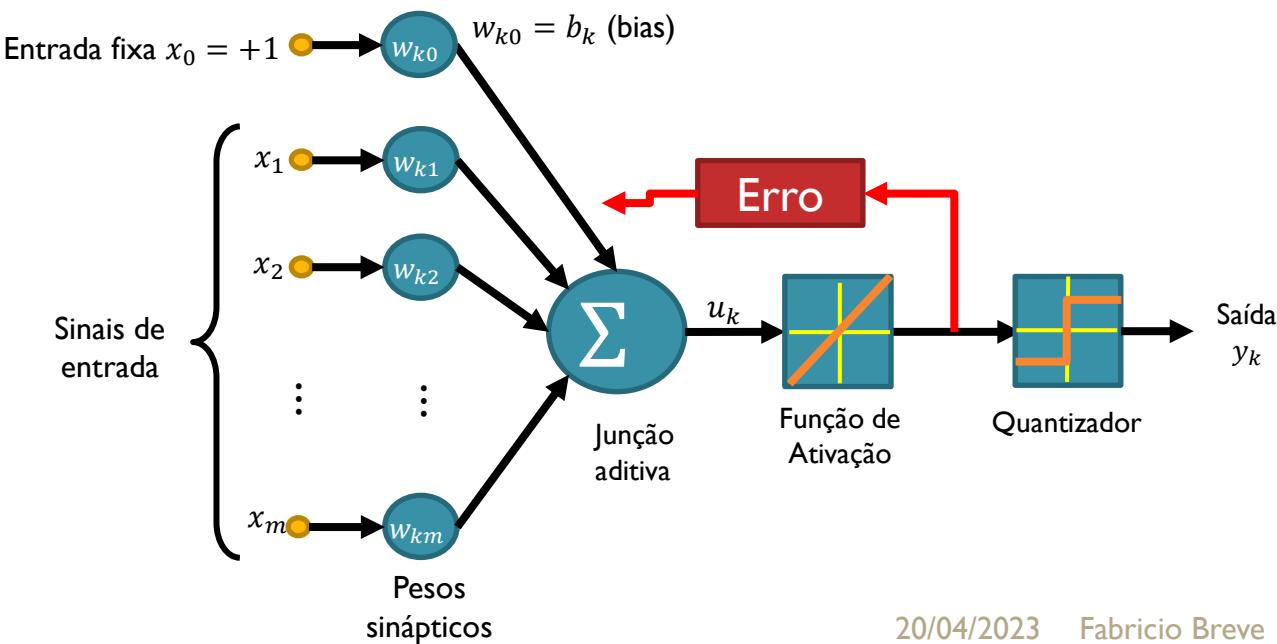
ADALINE

- Algoritmo de treinamento LMS (*Least Mean Squared*).
 - Mais poderoso que a regra de aprendizado do Perceptron.
 - Aprendizado continua mesmo após aprender o padrão de treinamento.
 - Mais robusto a ruído.





Perceptron



ADALINE

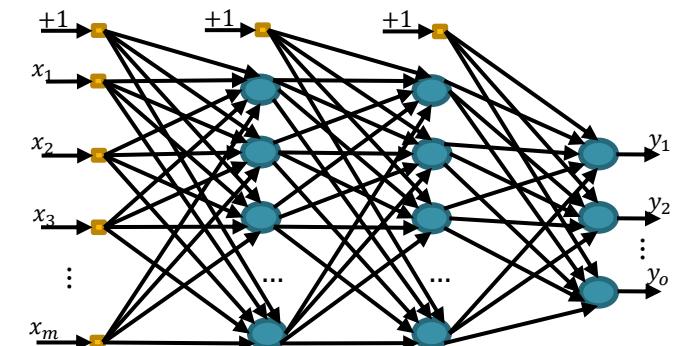


Redes Neurais Artificiais

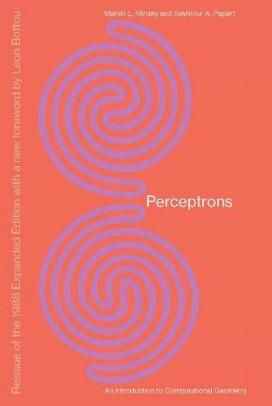
PERCEPTRON DE MÚLTIPLAS CAMADAS

Perceptron de Múltiplas Camadas

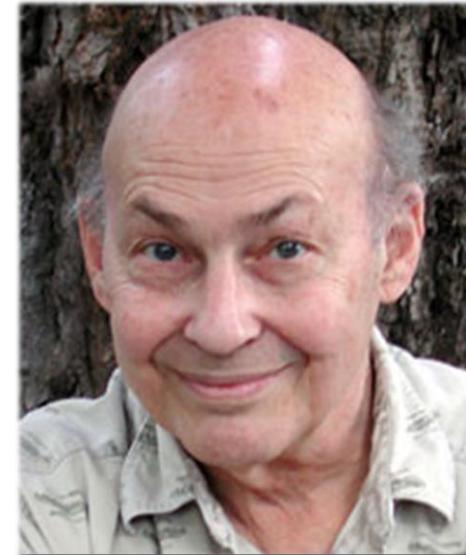
- Possui uma ou mais camadas intermediárias de nós
- Grande Funcionalidade
 - Uma camada intermediária: qualquer função contínua ou Booleana
 - Duas camadas intermediárias: qualquer função
- Treinada com o algoritmo *Backpropagation* (Retropropagação)



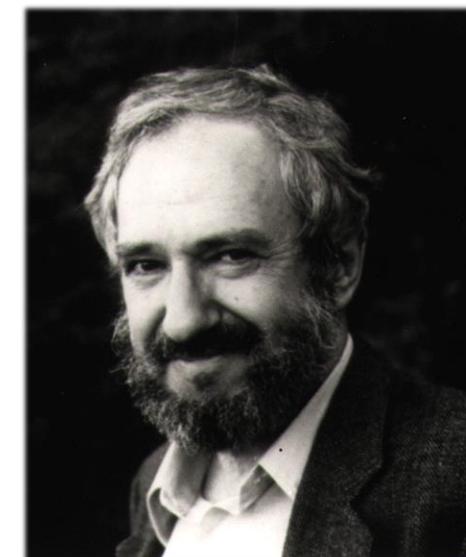
Histórico



- Em 1969 Minsky e Papert lançaram um livro mostrando as limitações do Perceptron.
 - Basicamente sua incapacidade de lidar com problemas não linearmente separáveis.
- Rosenblatt e Widrow sabiam dessas limitações e propuseram o uso de múltiplas camadas.
 - Mas não conseguiram generalizar seus algoritmos para treinar essas redes mais poderosas.
 - Isto diminuiu bastante o interesse em redes neurais nos anos 70.



Marvin Minsky [*1927 – †2016]



Seymour Papert [*1928 – †2016]

Marvin Lee Minsky foi um cientista cognitivo americano. Atuou principalmente nos estudos cognitivos no campo da inteligência artificial.

Seymour Papert foi um matemático e educador americano nascido na África do Sul. Lecionava no MIT. Graduou-se na Universidade de Witwatersrand, e obteve um PhD em matemática em 1952.

Algoritmo de Retropropagação de Erro

- Já existia nos anos 1970, mas ganhou destaque apenas nos 1980.
 - Popularizado por Rumelhart, Hinton, e Williams (1986).
 - É uma generalização do LMS.
 - Fez ressurgir o interesse em redes neurais.



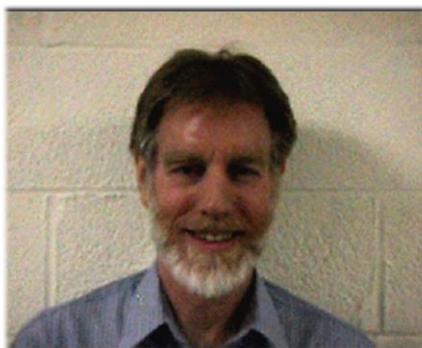
David Everett Rumelhart
[*1942 – †2011]

David Everett Rumelhart foi um psicólogo americano que fez muitas contribuições para a análise formal da cognição humana, trabalhando principalmente nas estruturas da psicologia matemática, inteligência artificial simbólica e processamento paralelo distribuído.



Geoffrey Everest Hinton
[*1947]

Geoffrey Everest Hinton é um psicólogo cognitivo e cientista da computação anglo-canadense, conhecido por seu trabalho sobre redes neurais artificiais. Desde 2013 divide seu tempo trabalhando para o Google e a Universidade de Toronto.



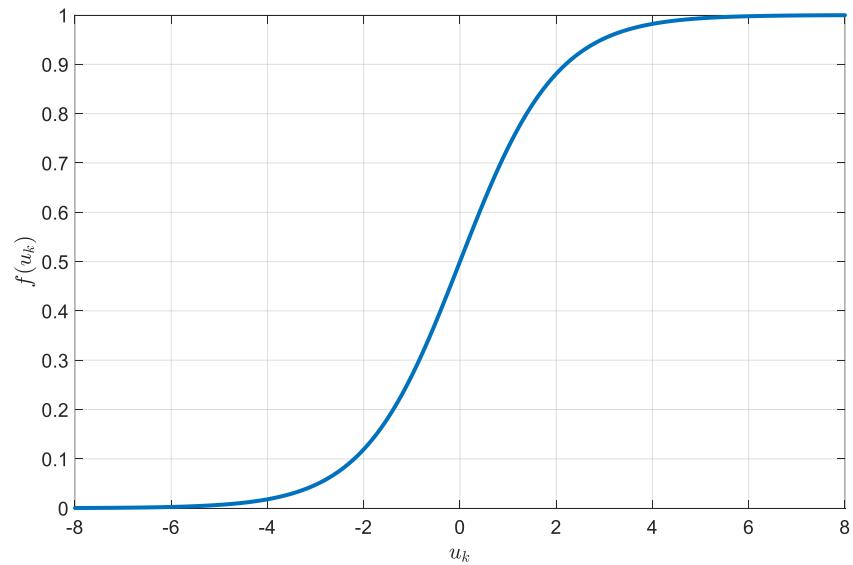
Ronald J. Williams

Ronald J. Williams é professor de ciência da computação na Northeastern University e um dos pioneiros das redes neurais.

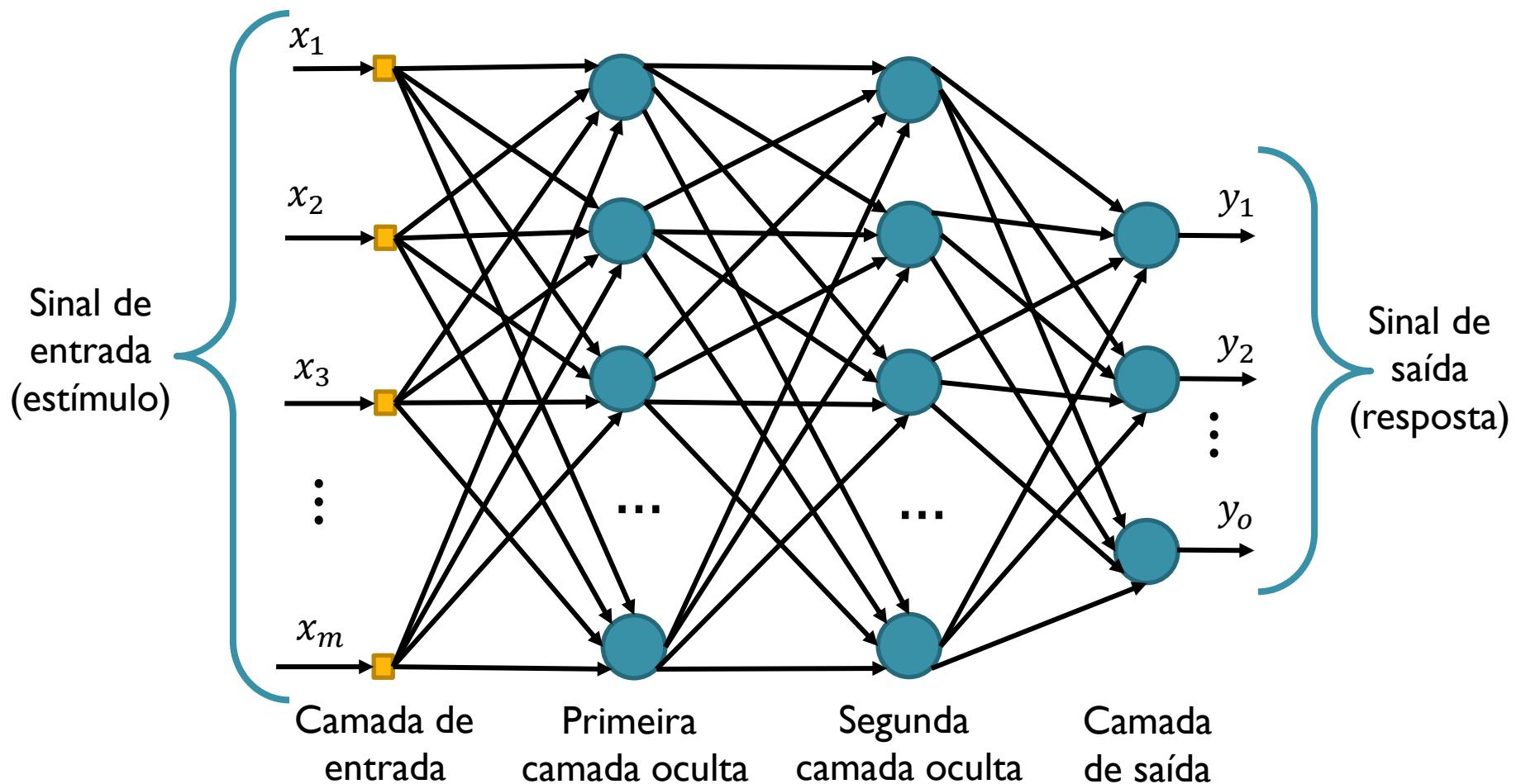
Perceptron de Múltiplas Camadas

- Uma camada de entrada, uma ou mais camadas intermediárias e uma camada de saída
- Camadas intermediárias e ocultas tipicamente usam funções sigmoides

$$g(a) \equiv \frac{1}{1 + \exp(-a)}$$



Perceptron de Múltiplas Camadas

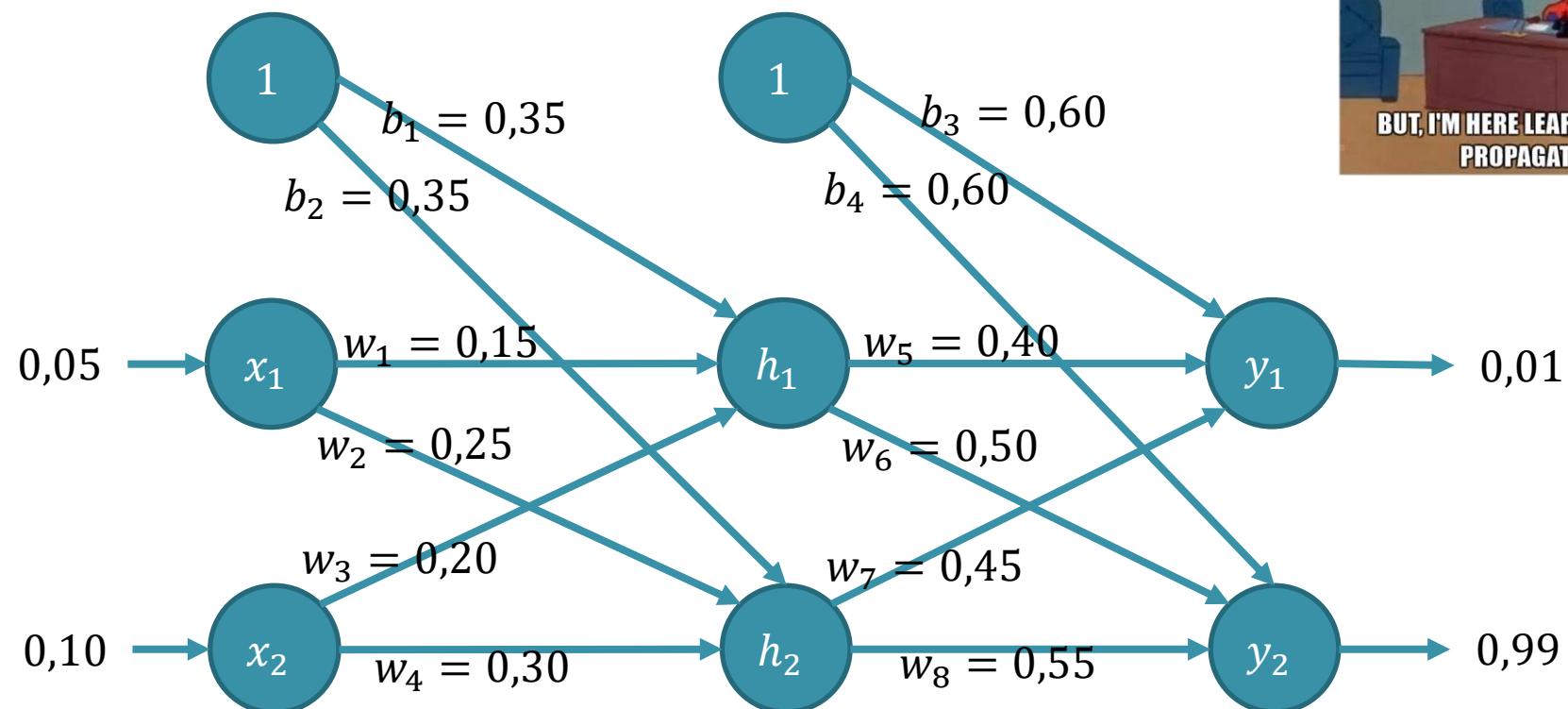


Algoritmo de Retropropagação de Erro

- Consiste basicamente em dois passos:
 - *Passo para frente*
 - Sinal aplicado à entrada vai se propagando pelos nós computacionais da rede até chegar aos nós de saída
 - *Passo para trás*
 - Todos os pesos sinápticos são ajustados de acordo com uma regra de correção de erro
 - Utiliza as derivadas parciais do erro com relação à cada peso.
 - Erro dos neurônios das camadas intermediárias é inferido à partir do erro dos neurônios da camada seguinte.

Algoritmo de Retropropagação: Exemplo

- Considera a seguinte rede neural:



Adaptado de:

<https://www.edureka.co/blog/backpropagation/>

Passo para a frente: camada intermediária

- Primeiro a camada intermediária, começando com h_1 :

- $h_1 = f(w_1x_1 + w_3x_2 + b_1)$

- $h_1 = f(0,15 \times 0,05 + 0,2 \times 0,1 + 0,35)$

- $h_1 = f(0,3775)$

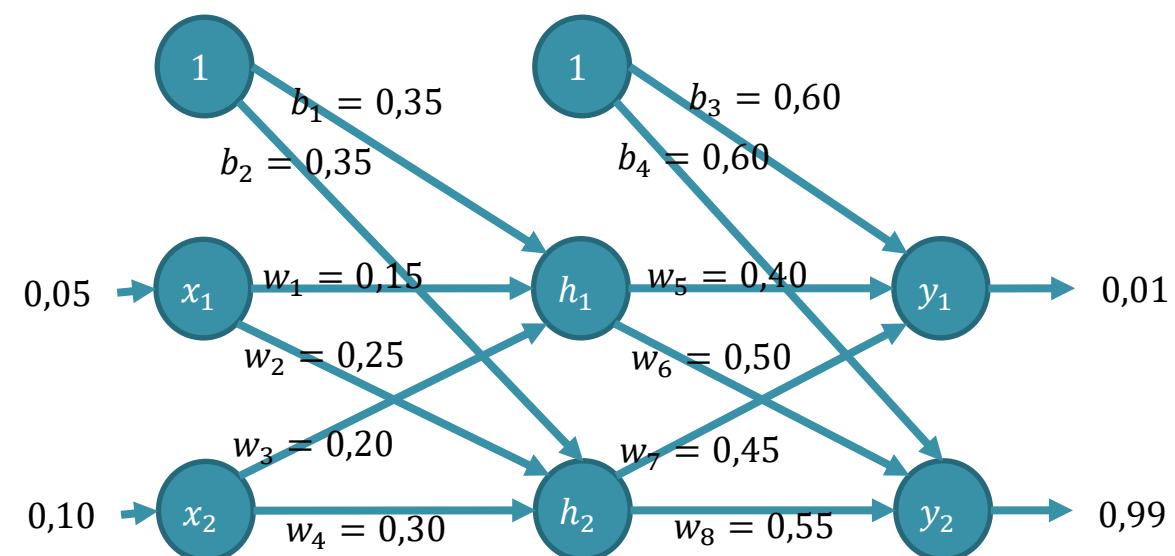
- $h_1 = \frac{1}{1+\exp(-0,3775)}$

- $h_1 = 0,593269992$

- De maneira similar:

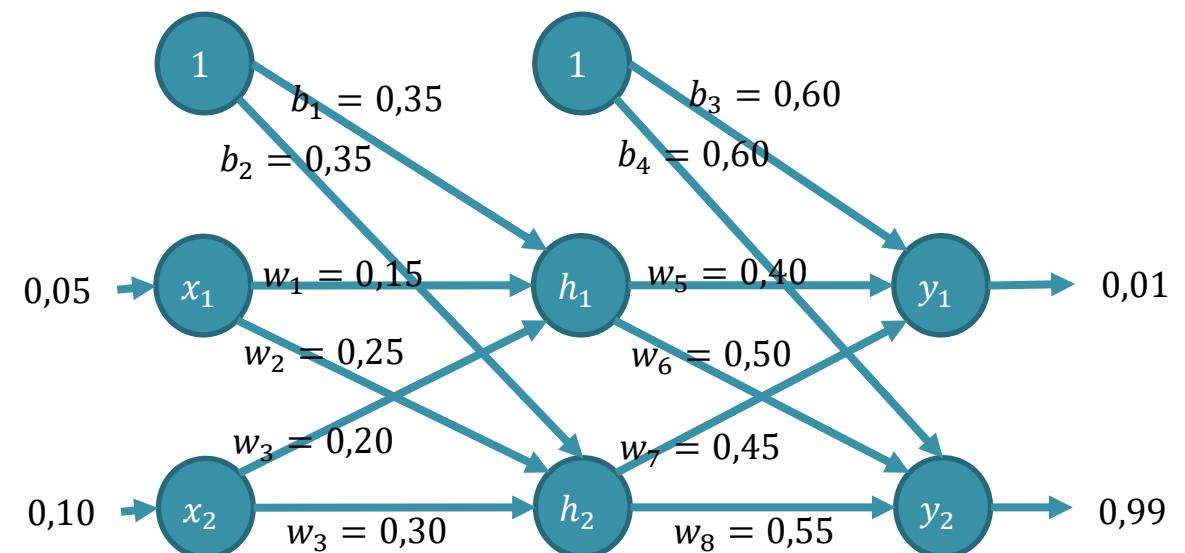
- $h_2 = 0,596884378$

Lembre-se que a função de ativação é a sigmoide



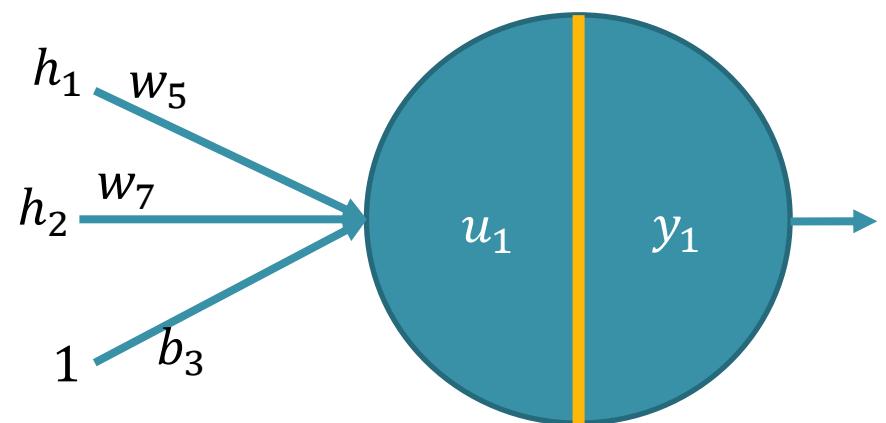
Passo para a frente: camada de saída

- Agora a camada de saída, começando por y_1 :
 - $y_1 = f(w_5 h_1 + w_7 h_2 + b_3)$
 - $y_1 = f(0,4 \times 0,593269992 + 0,45 \times 0,596884378 + 0,6)$
 - $y_1 = f(1,105905967)$
 - $y_1 = \frac{1}{1+\exp(-1,105905967)}$
 - $y_1 = 0,75136507$
- De maneira similar:
 - $y_2 = 0,772928465$



Erro na Camada de Saída

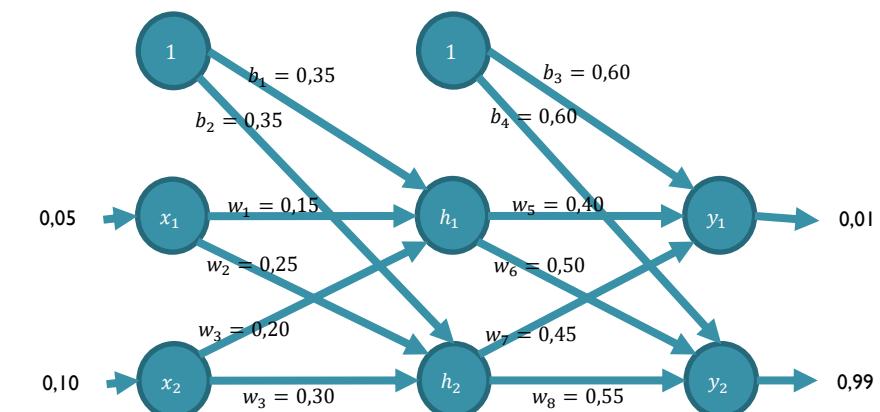
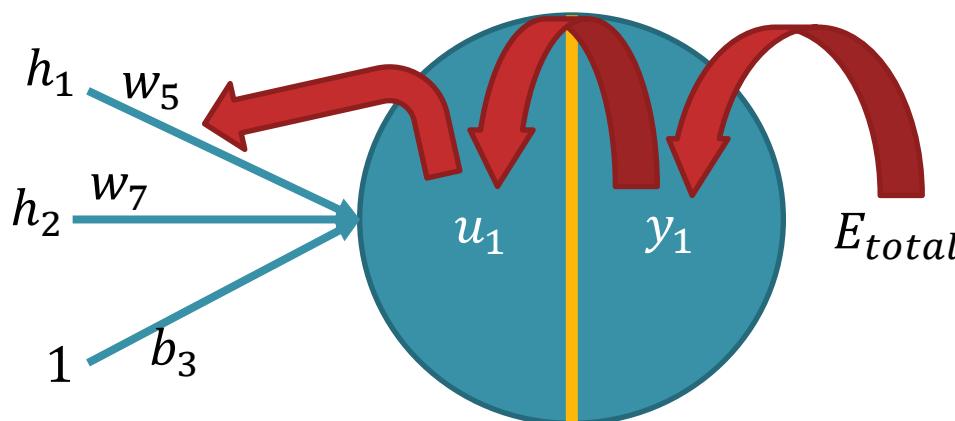
- Calculando o erro na camada de saída:
 - $E_1 = \frac{1}{2}(d_1 - y_1)^2$
 - $E_1 = \frac{1}{2}(0,01 - 0,75136507)^2$
 - $E_1 = 0,274811083$
 - $E_2 = 0,023560026$
 - $E_{total} = E_1 + E_2$
 - $E_{total} = 0,298371109$
- Atenção para a notação que usaremos nos próximos passos:
 - u_1 : somatório das entradas de y_1 , antes da função de ativação.
 - y_1 : saída de y_1 após a aplicação da função de ativação.



Passo para trás: a retropropagação

- Agora vamos retropropagar o erro.
 - Tentar reduzir o erro ajustando valores de pesos e bias.
- Considere w_5 , calcularemos a taxa de variação do erro em relação à alteração no peso w_5 :

- $$\frac{\delta E_{total}}{\delta w_5} = \frac{\delta E_{total}}{\delta y_1} \times \frac{\delta y_1}{\delta u_1} \times \frac{\delta u_1}{\delta w_5}$$



Passo para trás: erro em relação às saídas

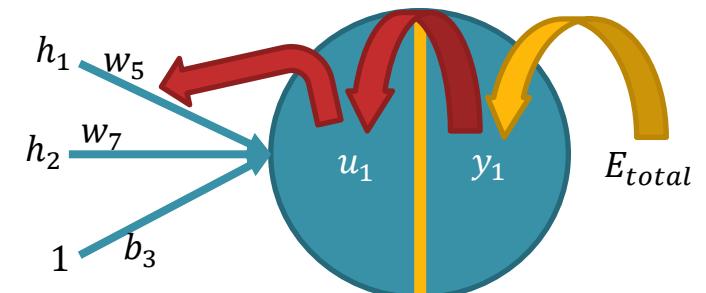
- Como estamos propagando para trás, a primeira coisa a fazer é calcular a mudança nos erros totais em relação às saídas y_1 e y_2 :

- $\circ E_{total} = \frac{1}{2}(d_1 - y_1)^2 + \frac{1}{2}(d_2 - y_2)^2$

- $\circ \frac{\delta E_{total}}{\delta y_1} = -(d_1 - y_1)$

- $\circ \frac{\delta E_{total}}{\delta y_1} = -(0,01 - 0,75136507)$

- $\circ \frac{\delta E_{total}}{\delta y_1} = 0,74136507$



Passo para trás: erro da saída em relação ao somatório das entradas

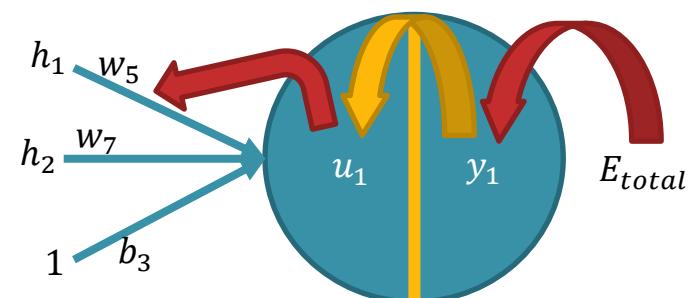
- Agora, vamos propagar mais para trás e calcular a mudança na saída y_1 em relação ao total de suas entradas:

- $y_1 = \frac{1}{1+\exp(-u_1)}$

- $\frac{\delta y_1}{\delta u_1} = y_1(1 - y_1)$

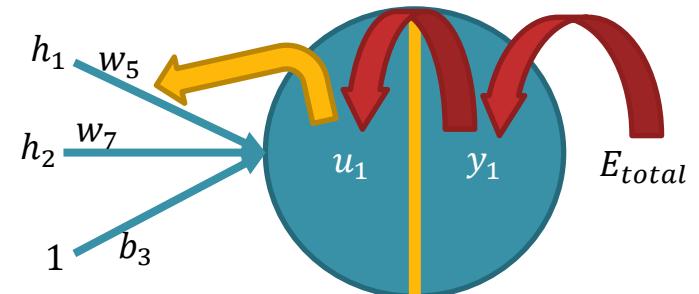
- $\frac{\delta y_1}{\delta u_1} = 0,75136507(1 - 0,75136507)$

- $\frac{\delta y_1}{\delta u_1} = 0,186815602$



Passo para trás:

- Vamos ver agora quanto a entrada líquida total de y_1 , isto é, o u_1 , muda em relação a w_5 :
 - $u_1 = w_5 \times h_1 + w_7 \times h_2 + b_3$
 - $\frac{\delta u_1}{\delta w_5} = 1 \times h_1 w_5^{(1-1)} + 0 + 0$
 - $\frac{\delta u_1}{\delta w_5} = 0,593269992$



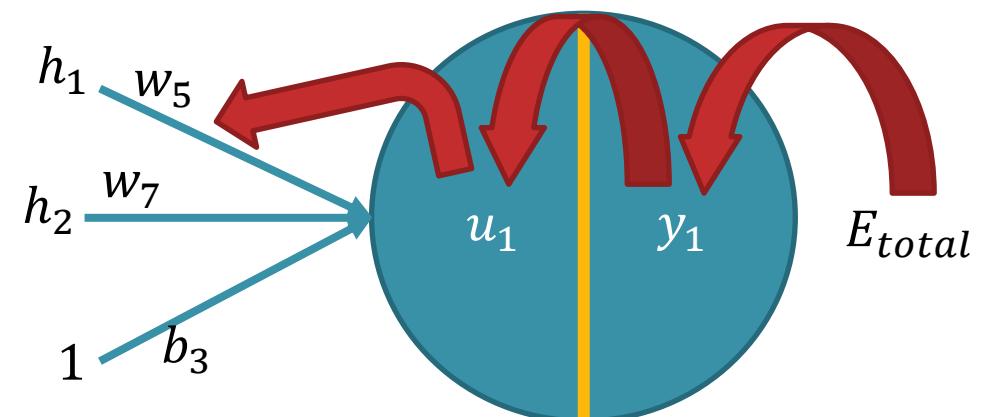
Calculando o peso atualizado

- Agora vamos juntar todos os valores:

- $\frac{\delta E_{total}}{\delta w_5} = \frac{\delta E_{total}}{\delta y_1} \times \frac{\delta y_1}{\delta u_1} \times \frac{\delta u_1}{\delta w_5}$

- $\frac{\delta E_{total}}{\delta w_5} = 0,74136507 \times 0,186815602 \times 0,593269992$

- $\frac{\delta E_{total}}{\delta w_5} = 0,082167041$



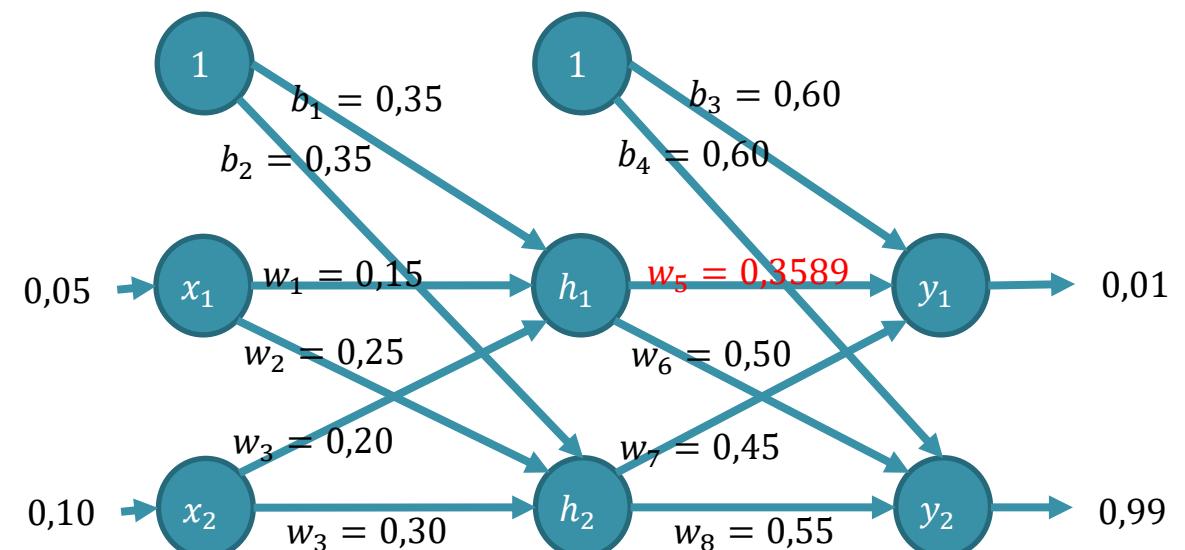
Calculando o peso atualizado

- Finalmente, vamos calcular o valor atualizado de w_5 :

- $w_5^+ = w_5 - \alpha \frac{\delta E_{total}}{\delta w_5}$

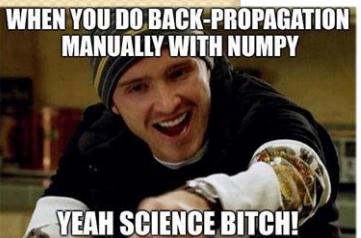
- $w_5^+ = 0,4 - 0,5 \times 0,082167041$

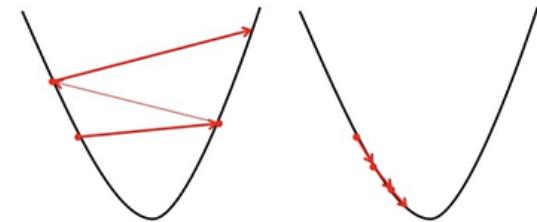
- $w_5^+ = 0,35891648$



Algoritmo de Retropropagação: Exemplo

- De forma similar, podemos calcular todos os demais valores de peso.
- Após atualizar todos os pesos, iremos novamente propagar para frente e calcular a saída.
- Novamente, vamos calcular o erro.
- Se o erro atingiu o valor alvo, vamos parar por aí, caso contrário, vamos novamente propagar para trás e atualizar os valores de peso.
- Este processo continuará se repetindo até que um critério de parada seja atingido.
- Nota: o aprendizado de retropropagação não requer a **normalização** de vetores de entrada; no entanto, a normalização pode melhorar o desempenho.





Taxa de Aprendizado

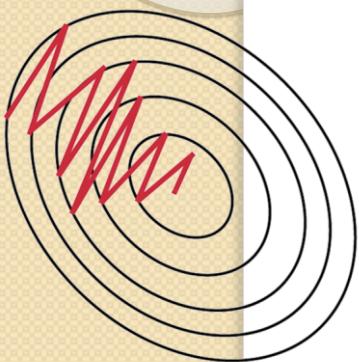
- Em nossos exemplos e exercícios de aula usamos $\alpha = 0,5$ para acelerar a convergência.
- O valor ideal depende do problema.
 - Não existe uma taxa de aprendizado que seja adequada para todos os casos.
 - Se muito baixas, podem levar a um treinamento muito lento.
 - Se muito altas, podem fazer com que a rede neural oscile em torno do mínimo global do erro e nunca atinja a convergência.
- A taxa de aprendizado pode ser variável / adaptativa.
 - Exemplo: Podemos iniciar com $\alpha = 0,001$ ou $\alpha = 0,01$, reduzindo ao longo do tempo, de acordo com o desempenho obtido em um conjunto de validação.

Variações do Algoritmo de Retropropagação

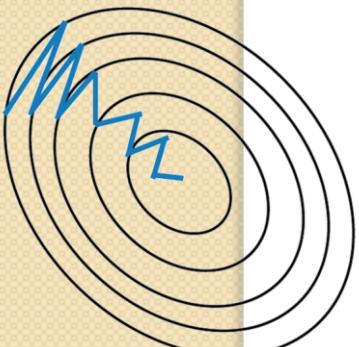
- Momentum
- Quickprop
- Newton
- Levenberg Marquardt
- Super Self-Adjusting Backpropagation (superSAB)
- Métodos de gradiente conjugado

Exemplo: Momentum

- Visa acelerar a convergência do processo de treinamento da rede neural.
- É uma medida do “impulso” do processo de treinamento.
 - Leva em consideração a direção e magnitude dos gradientes de erro que foram calculados durante o treinamento anterior.
 - É uma forma de “lembra” da direção em que os parâmetros da rede neural estavam sendo atualizados nas iterações anteriores e continuar atualizando nessa direção com uma certa inércia.
- O momentum γ é adicionado ao cálculo do gradiente descendente, multiplicando-se a taxa de aprendizado pelo momentum da iteração anterior e somando-se ao gradiente atual:
 - $$\Delta w_{ij} = \left(\alpha \frac{\delta E}{\delta w_{ij}} \right) + (\gamma \Delta w_{ij}^{t-1})$$
 - Isso permite que a rede neural "pule" sobre mínimos locais e evite ficar presa em platôs.
- O valor do momentum é geralmente um hiperparâmetro ajustável que deve ser escolhido com cuidado para evitar o efeito oposto, de oscilação.
 - Um valor típico de momentum varia de 0,1 a 0,9,
 - Mas pode variar dependendo do conjunto de dados e do problema em questão.

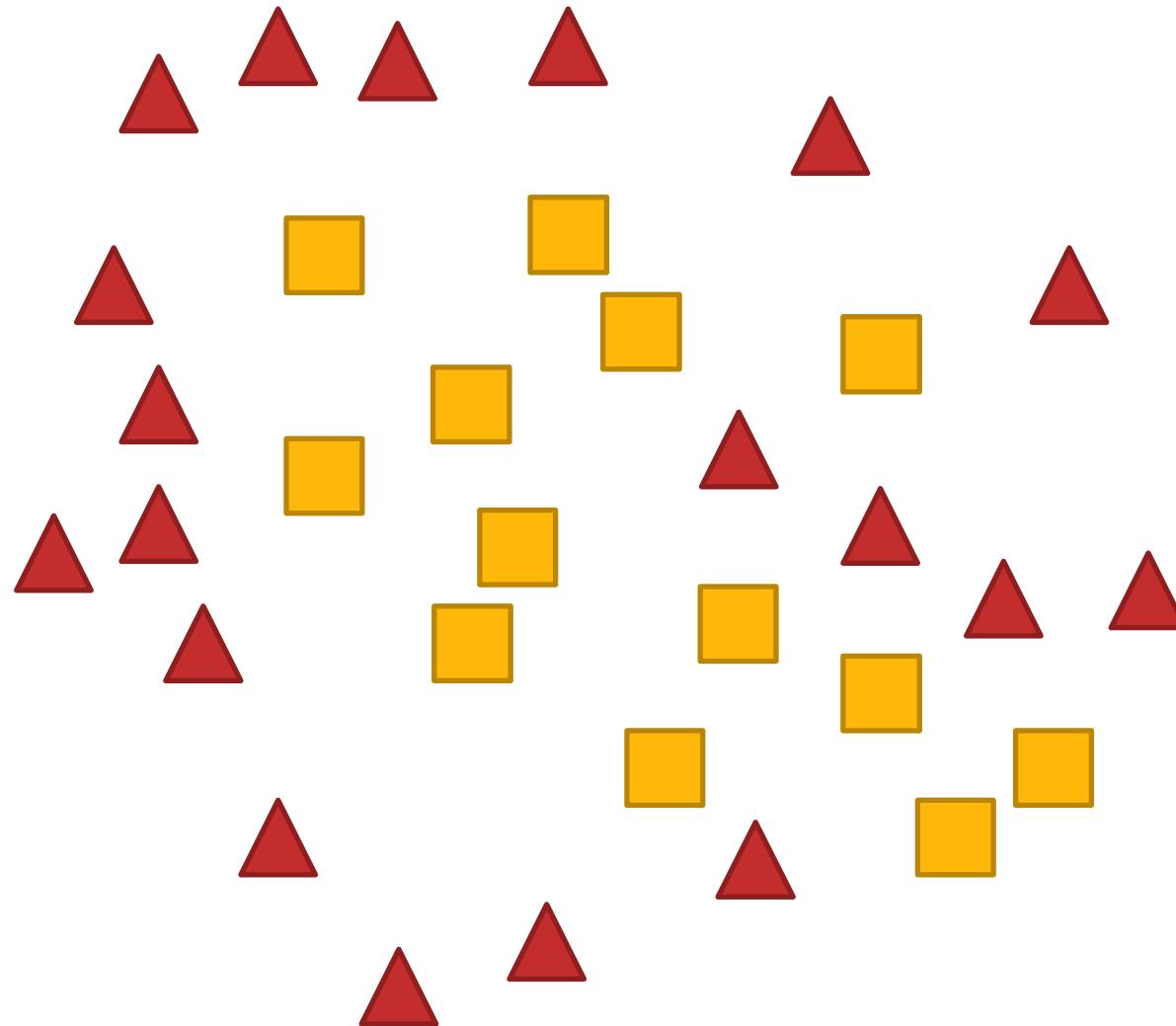


sem momentum

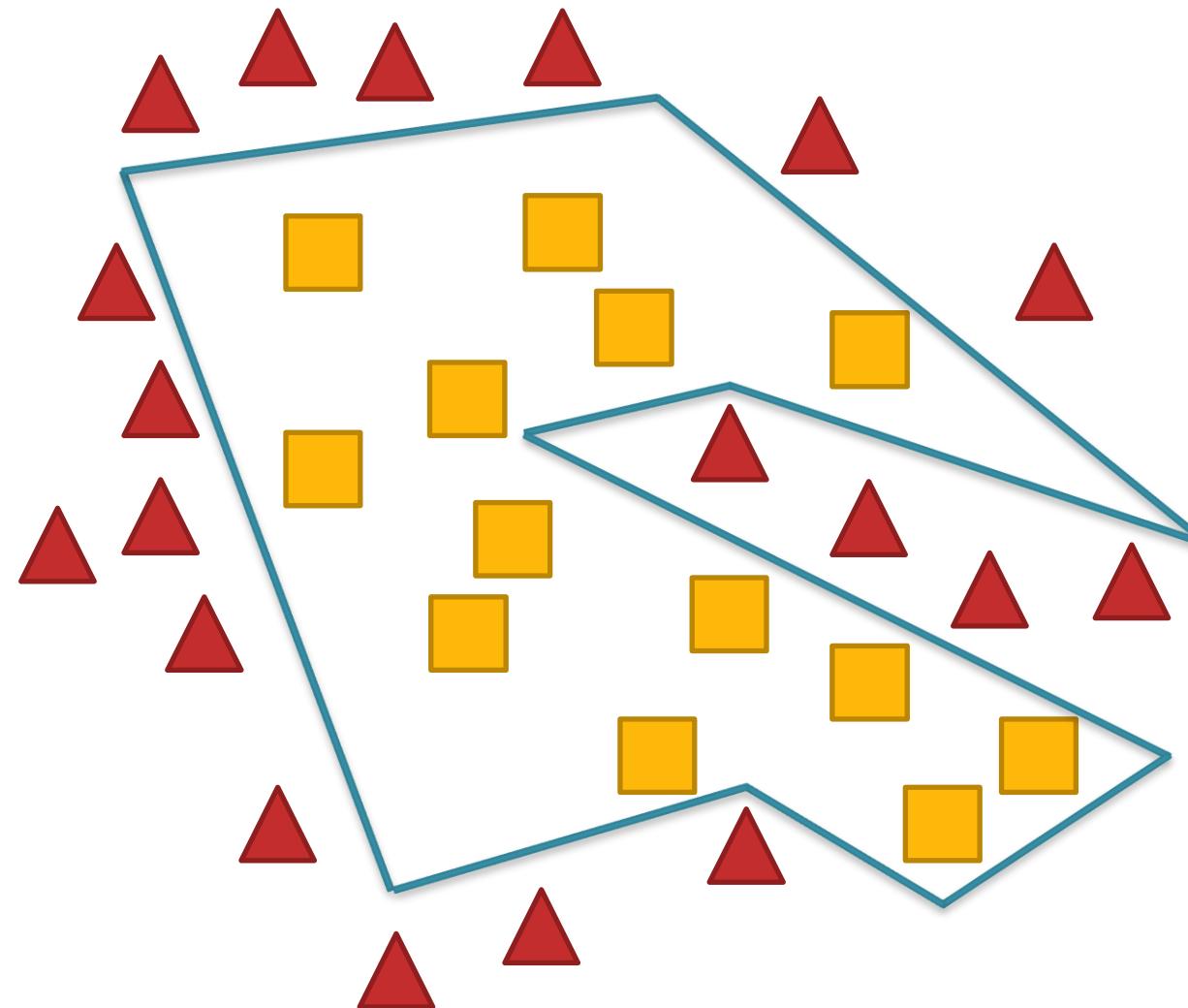


com momentum

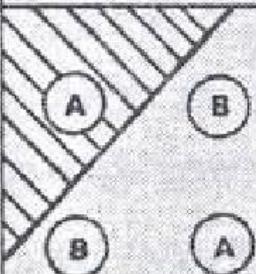
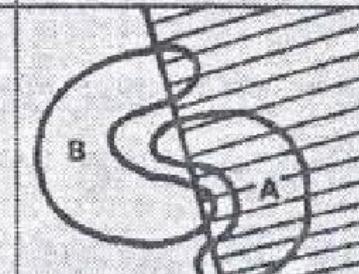
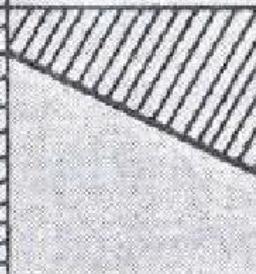
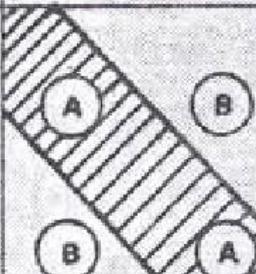
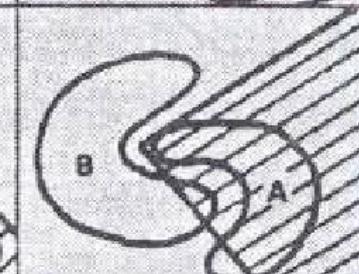
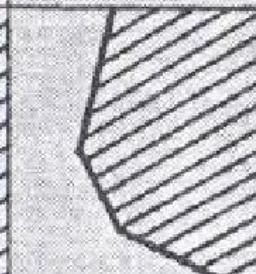
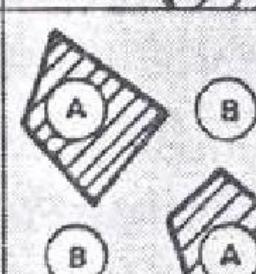
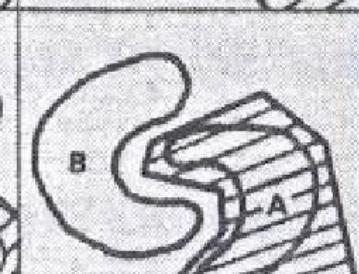
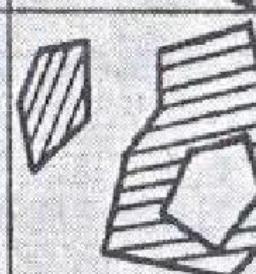
Por que múltiplas camadas?



Por que múltiplas camadas?



Por que múltiplas camadas?

STRUCTURE	TYPES OF DECISION REGIONS	EXCLUSIVE OR PROBLEM	CLASSES WITH MESHED REGIONS	MOST GENERAL REGION SHAPES
SINGLE-LAYER	HALF PLANE BOUNDED BY HYPERPLANE			
TWO-LAYER	CONVEX OPEN OR CLOSED REGIONS			
THREE-LAYER	ARBITRARY (Complexity Limited By Number of Nodes)			

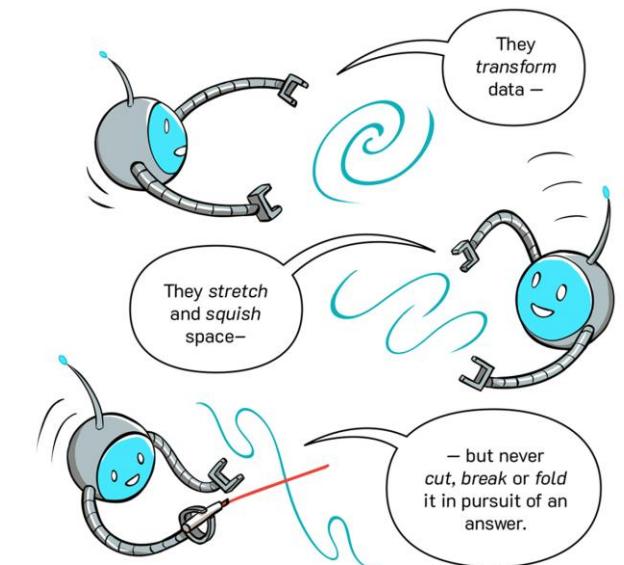
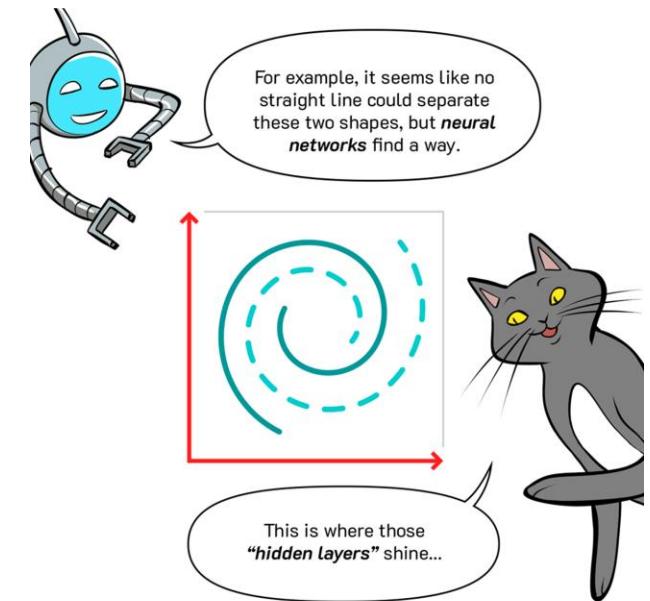
Lippmann, R.P., "An introduction to computing with neural nets," ASSP Magazine, IEEE , vol.4, no.2, pp.4-22, Apr 1987

doi: 10.1109/MASSP.1987.1165576

URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1165576&isnumber=26240>

Por que múltiplas camadas?

- Unidades intermediárias
 - Detectores de características
 - Geram uma codificação interna da entrada
 - Dado um número grande o suficiente de unidades intermediárias é possível formar representações internas para qualquer distribuição dos padrões de entrada



<https://cloud.google.com/products/ai/ml-comic-2>

Quantas camadas intermediárias utilizar?

- **Uma camada:** suficiente para aproximar qualquer função contínua ou Booleana.
- **Duas camadas:** suficiente para aproximar qualquer função.
- **Três ou mais camadas:** pode facilitar o treinamento da rede.
 - **Cuidado:** cada vez que o erro é propagado para uma camada anterior, ele se torna menos útil.
 - Neurônios demais podem causar *overfitting* (à seguir).

Dificuldades de aprendizado

- *Overfitting*

- Depois de um certo ponto do treinamento, a rede piora ao invés de melhorar
- Memoriza padrões de treinamento, incluindo suas peculiaridades (piora generalização)
- Alternativas
 - Encerrar treinamento mais cedo (*early stop*)
 - Reduzir pesos (*weight decay*)
 - Ignorar parte dos neurônios aleatoriamente durante o treinamento (*dropout*)



<https://cloud.google.com/products/ai/ml-comic-2>



Overfitting

- Pode ser utilizado um subconjunto de validação para testar a capacidade de generalização
 - Parando o treinamento quando a capacidade de generalização deixa de melhorar



Treinamento

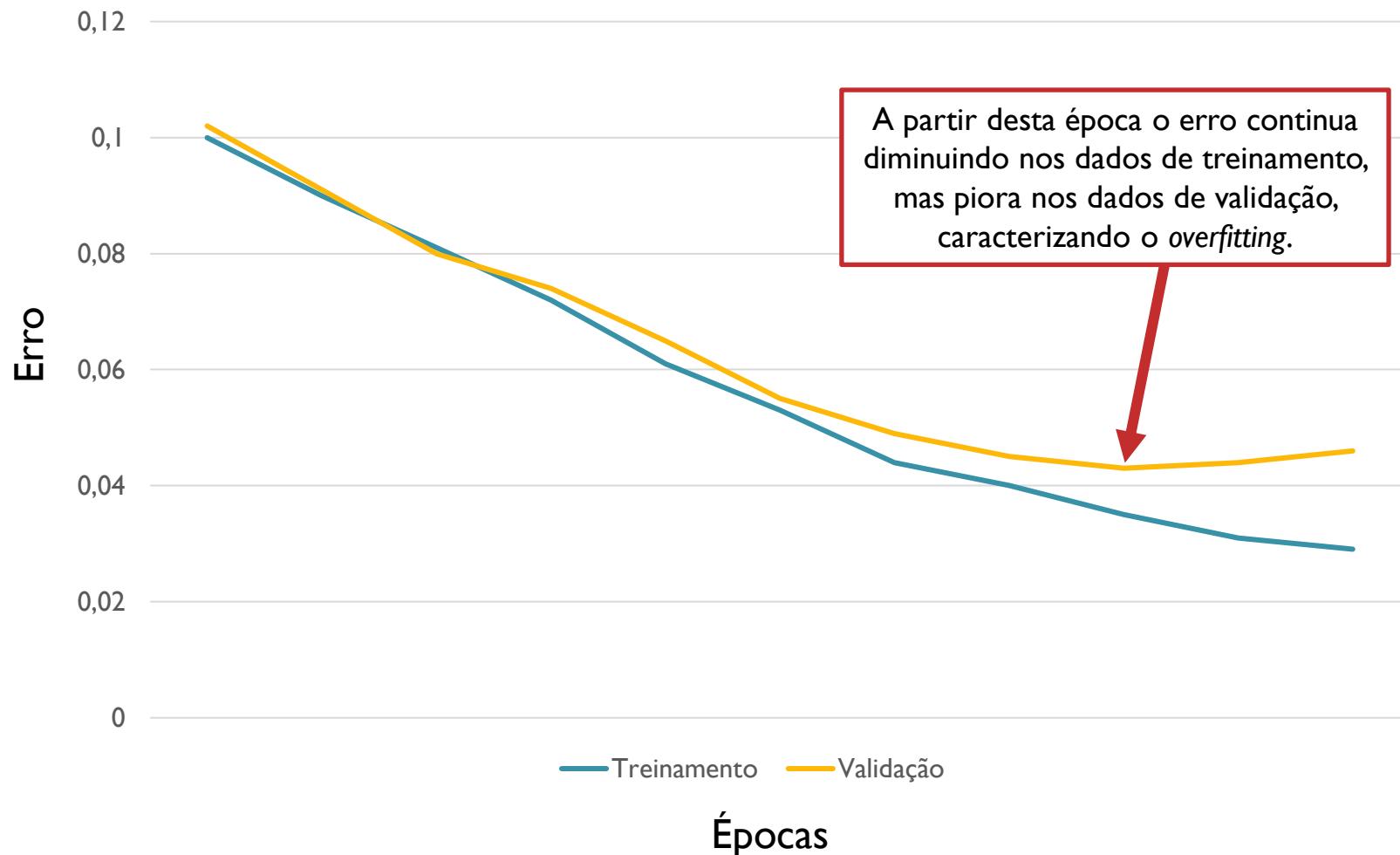
Validação

Overfitting

- Exemplo: dividindo o conjunto de dados em:
 - Subconjunto de Treinamento
 - Usados para treinar a rede, sinal de erro usado para ajustar pesos.
 - Subconjunto de Validação
 - Usado para testar a capacidade de generalização da rede a cada época, ou para testar diferentes arquiteturas/algoritmos de rede candidatas.
 - Subconjunto de Testes
 - Usado apenas após o treinamento, para testar a capacidade da rede escolhida e treinada em classificar dados que ainda não foram vistos.



Overfitting





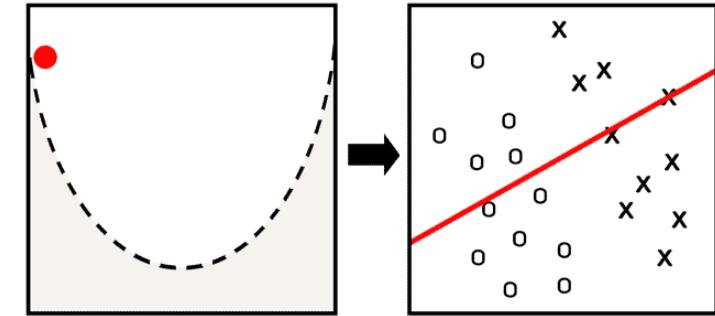
Critério de Parada



- Nem sempre é viável treinar uma rede até atingir um erro zero.
 - Pode até mesmo ser **impossível** dada a arquitetura da rede.
 - Exemplo: problema não linearmente separável em rede Perceptron com única camada.
 - A descida do gradiente com retropropagação **não garante** encontrar o mínimo global da função de erro, mas apenas um mínimo local
 - E muitas vezes nem é desejável: **overfitting**.
- É comum parar o treinamento quando o erro no conjunto de validação não teve redução (ou redução significativa) por um número determinado de iterações.
 - Nesse ponto você pode optar por manter os pesos finais ou por restaurar os pesos da iteração com menor erro.

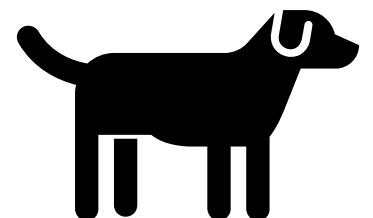
Atualização dos pesos

- Ciclo (ou Época)
 - Apresentação de todos os exemplos de treinamento durante o aprendizado.
 - Exemplos devem ser apresentados em ordem aleatória.
- Abordagens para atualização dos pesos.
 - Por padrão (online).
 - Por ciclo.
 - Por lotes (*batches*).
- Melhor método depende da aplicação.



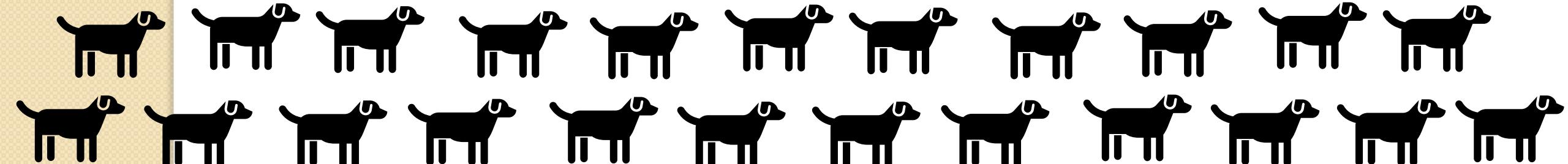
Atualização de Pesos por Padrão

- Pesos são atualizados após apresentação de cada padrão.
- Estável se taxas de aprendizado forem pequenas.
 - Taxas elevadas \Rightarrow rede instável.
 - Reduzir progressivamente as taxas.
- Mais rápido, principalmente se o conjunto de treinamento for grande e redundante .
- Requer menos memória.



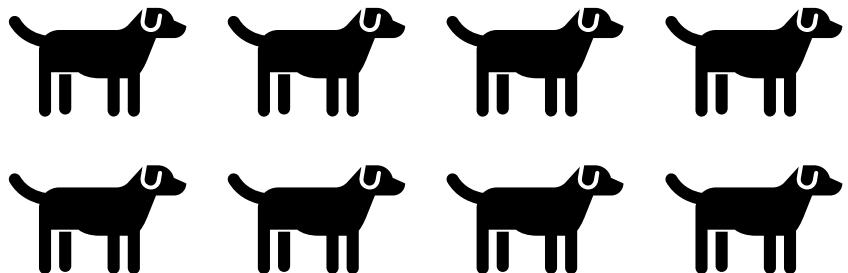
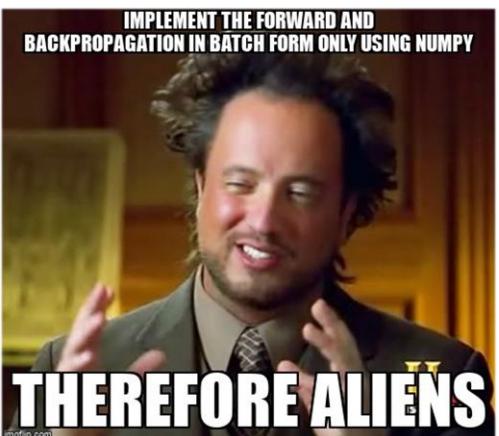
Atualização dos Pesos por Ciclo

- Pesos atualizados depois que todos os padrões de treinamento forem apresentados.
- Geralmente mais estável.
- Pode ser lento se o conjunto de treinamento for grande e redundante.
- Requer mais memória.



Atualização dos Pesos por Lotes (Batches)

- Quantidade específica de padrões por lote pré-definida.
- Pesos são atualizados após a apresentação de cada lote.
- A alocação dos padrões em cada lote deve ser feita aleatoriamente.
 - E refeita em cada ciclo.





Redes Neurais Artificiais

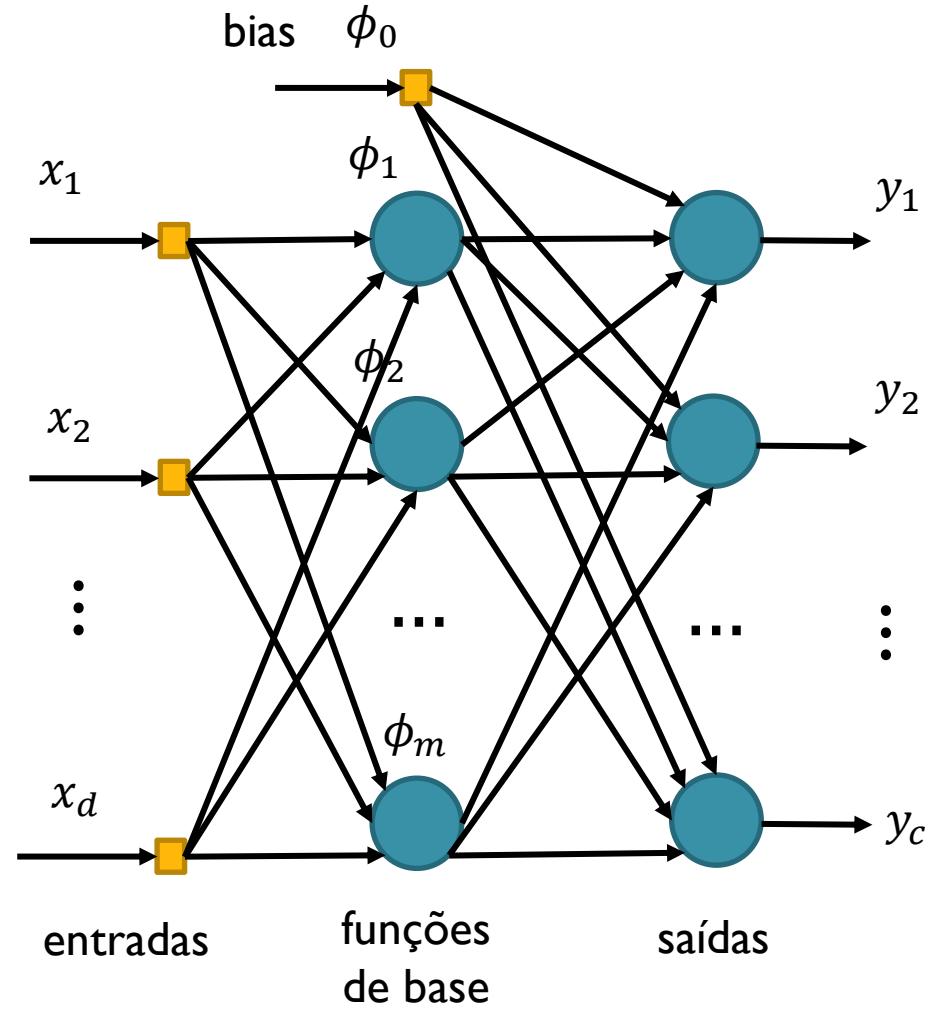
REDES DE FUNÇÃO DE BASE RADIAL

Redes de Função de Base Radial

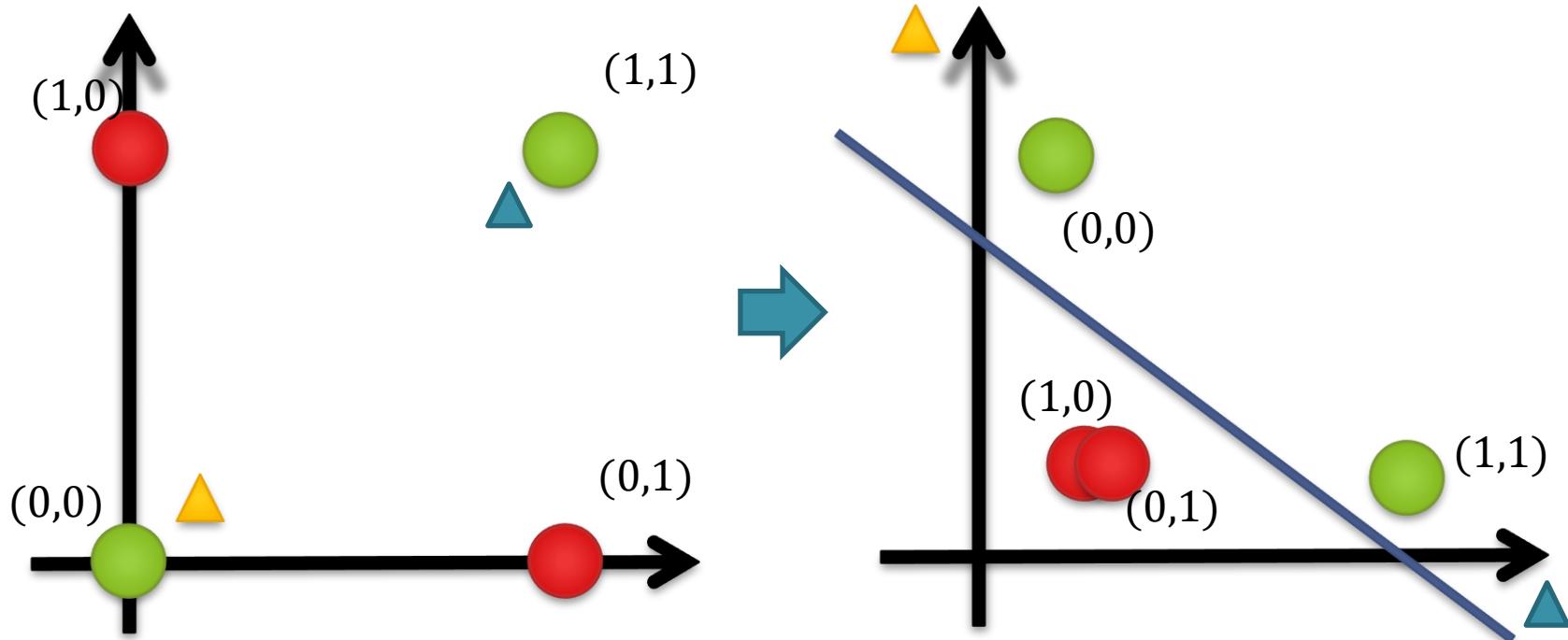
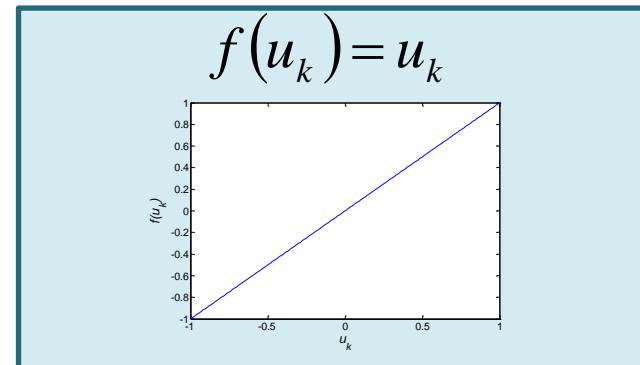
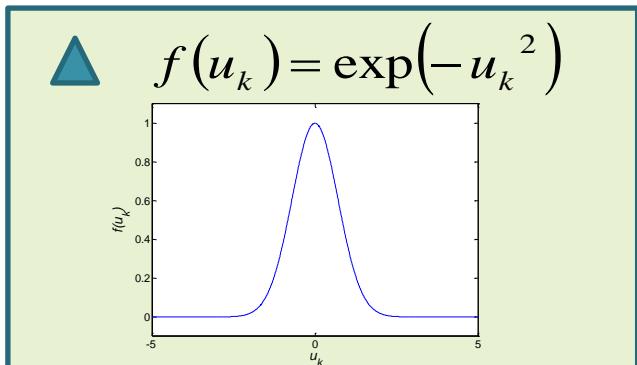
- *Radial Basis Function*
- Vê a rede neural como um *problema de ajuste de curva* em um espaço de alta dimensionalidade.
- Aprender equivale a encontrar uma superfície num espaço multidimensional que forneça o melhor ajuste para os dados de treinamento do ponto de vista estatístico.

Redes de Função de Base Radial

- Possui 3 camadas:
 - **Entrada:** nós sensoriais que recebem os dados do ambiente.
 - **Intermediária (oculta):** faz uma transformação não-linear do espaço de entrada para um espaço oculto, normalmente de alta dimensionalidade.
 - **Saída:** linear, e fornece a resposta da rede ao sinal de entrada.



Redes de Função de Base Radial

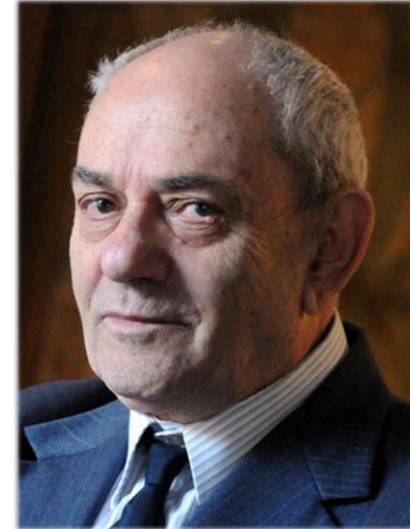




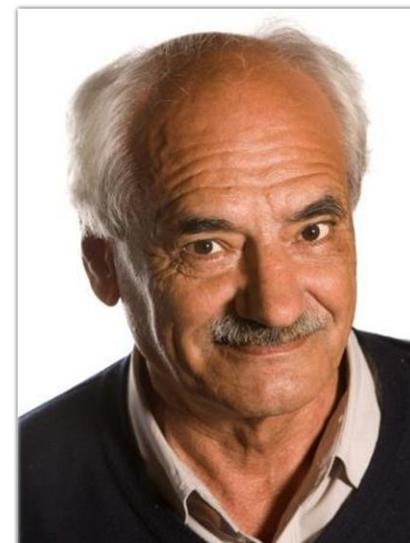
MÁQUINAS DE VETOR DE SUPORTE

Máquinas de Vetor de Suporte

- *Support Vector Machines*
- Máquina linear.
- Constrói um hiperplano em que a margem de separação entre classes seja máxima.
- Baseadas na Teoria do Aprendizado Estatístico.
 - Vapnik e Chervonenkis em 1968
- Reduz erro de generalização.



Vladimir Naumovich Vapnik
[*1936]

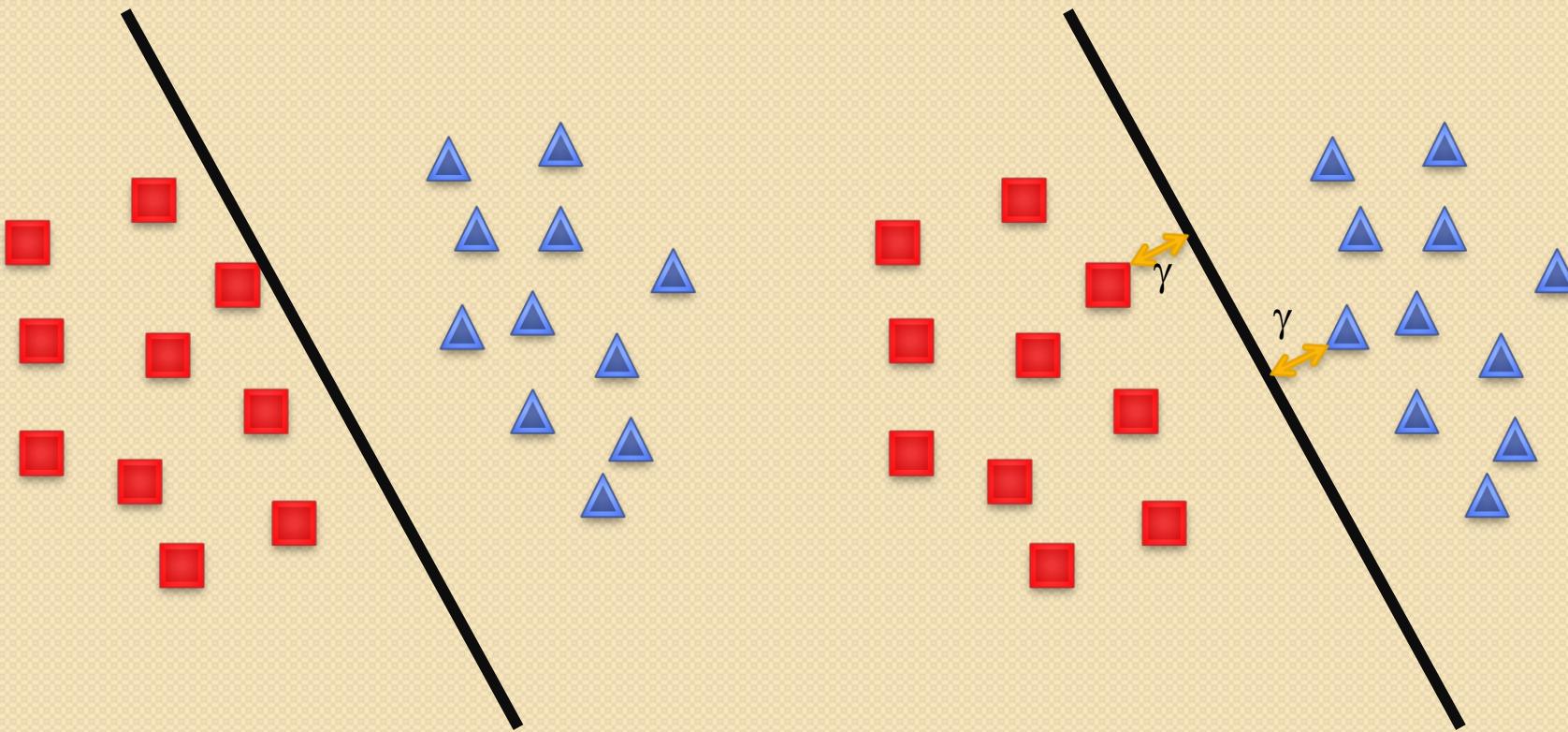


Alexey Yakovlevich
Chervonenkis [*1938 –
†2014]

Vapnik, Vladimir Naumovich, and Aleksei Yakovlevich Chervonenkis. "The uniform convergence of frequencies of the appearance of events to their probabilities." *Doklady Akademii Nauk.* Vol. 181. No. 4. Russian Academy of Sciences, 1968.

Vladimir Naumovich Vapnik é um matemático, cientista da computação, pesquisador e acadêmico soviético-americano.

Alexey Yakovlevich Chervonenkis foi um matemático soviético e russo



Hiperplano: Perceptron X Máquinas de Vetor de Suporte

Redes Neurais Artificiais



MAPAS AUTO-ORGANIZÁVEIS

Mapas Auto-Organizáveis

- Do inglês *Self Organizing Map* (SOM).
- Também chamados Mapas de Kohonen.
- Modelos não supervisionados.
- Baseado em aprendizagem competitiva.
 - Os neurônios competem entre si para serem ativados ou disparados.
 - Apenas um será ativado em dado instante de tempo
 - Winner-Takes-All.
 - O vencedor leva tudo.
 - Somente o peso do neurônio vencedor e de seus vizinhos é atualizado.
 - Os pesos aproximam o padrão de entrada.

Teuvo Kalevi Kohonen
foi um cientista da
computação finlandês.
Ele foi professor
emerito da Academia
da Finlândia.



Teuvo Kohonen [*1934 –
†2021]

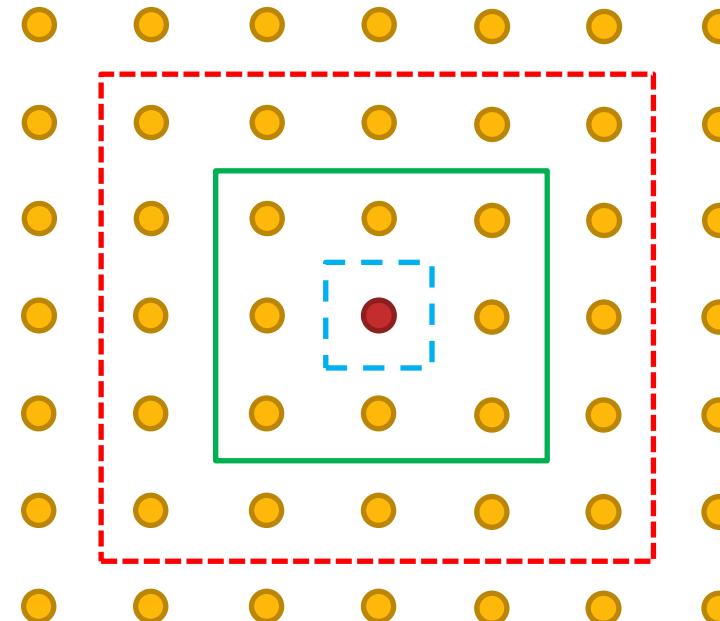
Mapas Auto-Organizáveis

- Competição
 - Para cada padrão de entrada os neurônios da rede competem entre si para determinar o vencedor.
- Cooperação
 - O neurônio vencedor determina sua vizinhança espacial, os neurônios vizinhos também serão estimulados.
- Adaptação
 - O neurônio vencedor e seus vizinhos terão seus vetores de peso atualizados.
 - Atualização dos vizinhos proporcional a distância.

Mapas Auto-Organizáveis: vizinhanças

Função de
Vizinhança

- $R = 0$
- $R = 1$
- $R = 2$

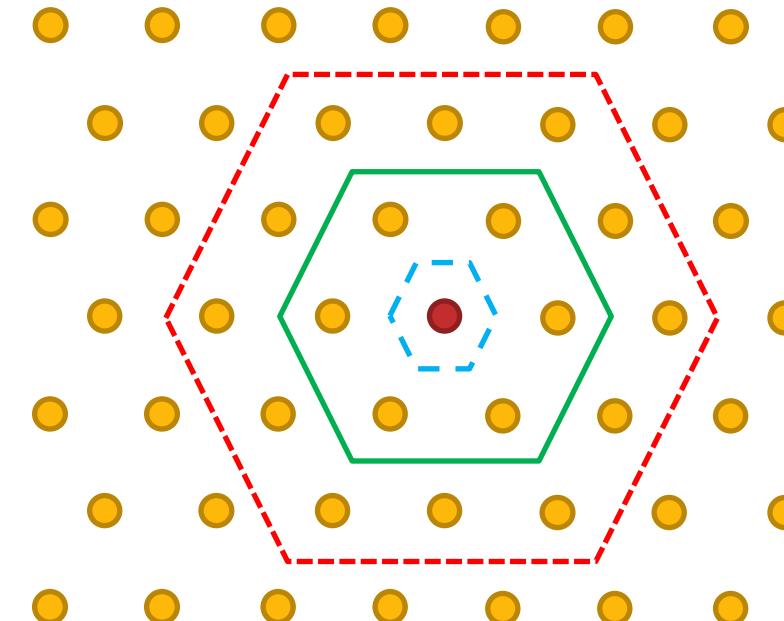


Vizinhança de grade retangular

Mapas Auto-Organizáveis: vizinhanças

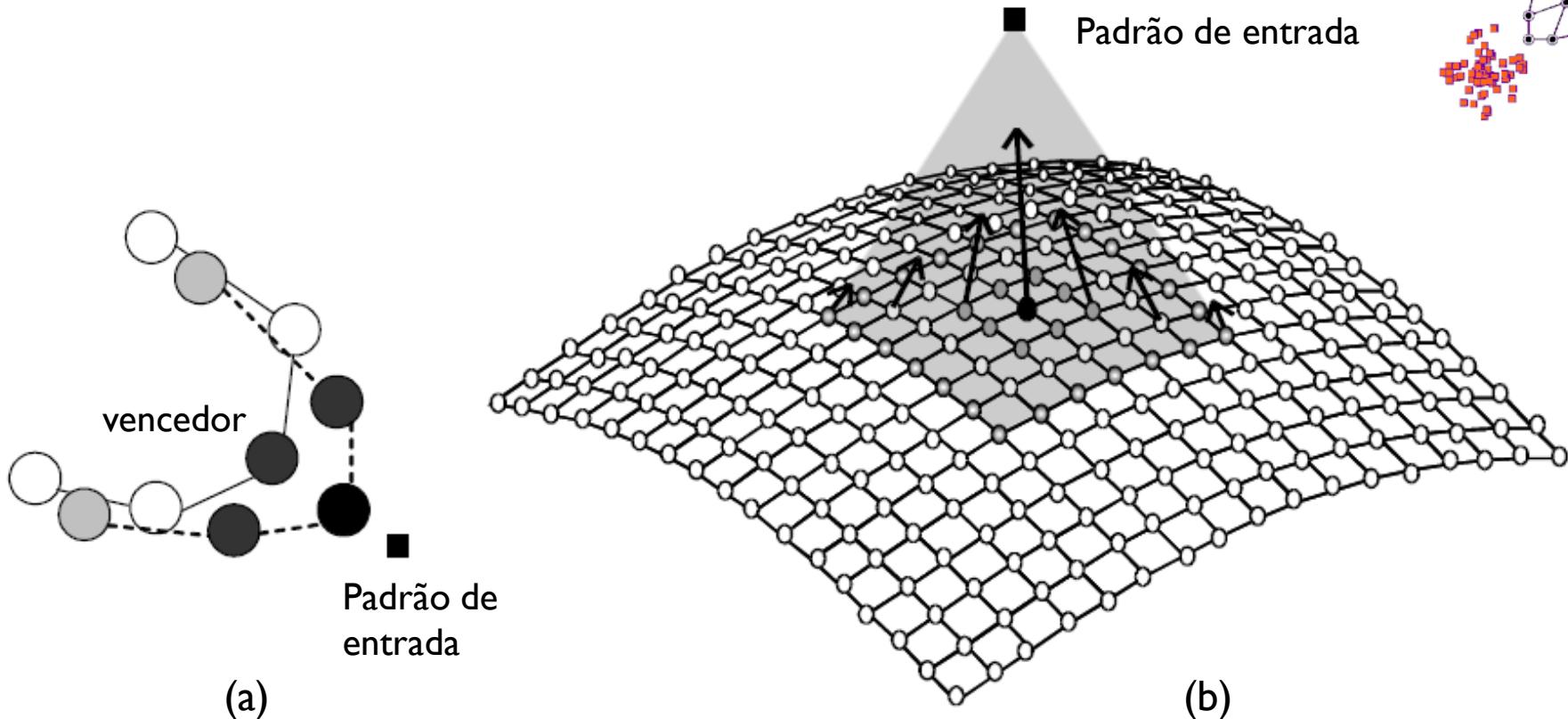
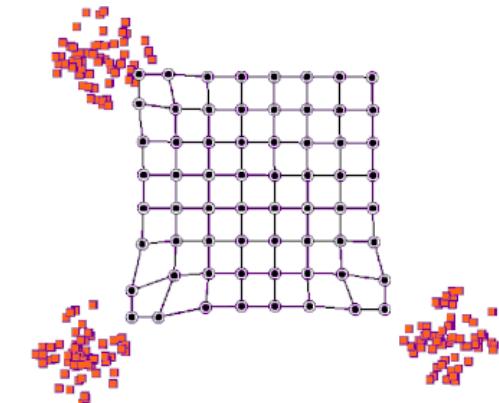
Função de
Vizinhança

- $R = 0$
- $R = 1$
- $R = 2$



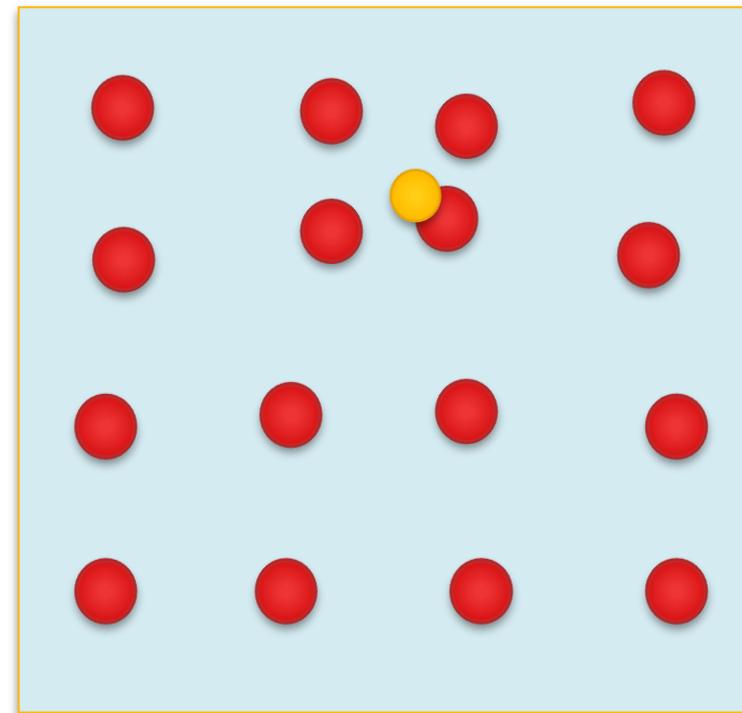
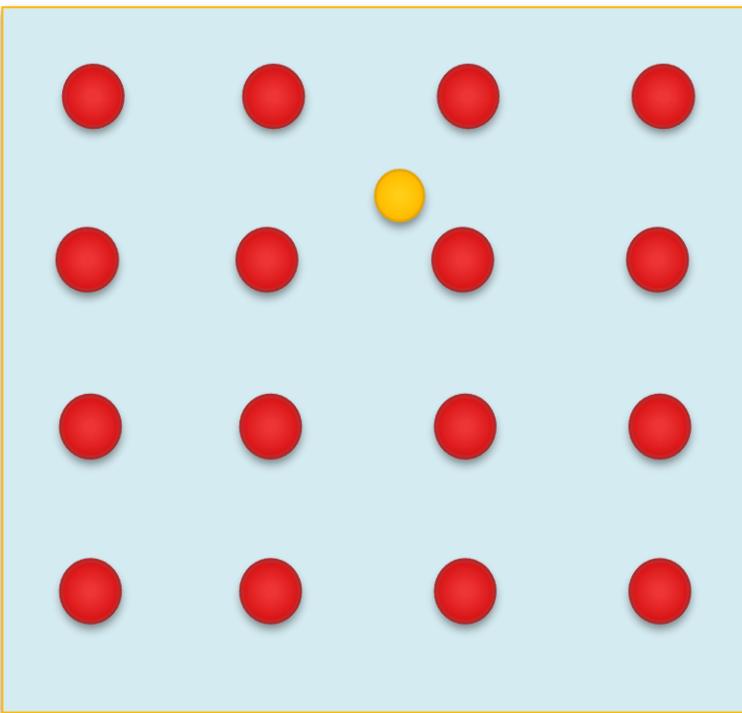
Vizinhança de grade hexagonal

Mapas Auto-Organizáveis



Procedimento adaptativo. O vencedor e seus vizinhos são movidos em direção ao padrão de entrada. A unidade vencedora realiza a maior atualização em direção ao padrão de entrada, seguida de suas vizinhas imediatas. (a) Vizinhança uni-direcional. (b) Vizinhança bi-direcional.

Mapas Auto-Organizáveis



Mapas Auto-Organizáveis

Nomes de Animais e seus Atributos

Animal		Pombo	Galinha	Pato	Ganso	Coruja	Falcão	Águia	Raposa	Cão	Lobo	Gato	Tigre	Leão	Cavalo	Zebra	Vaca
é	pequeno	-	-	-	-	-	-	0	0	0	0	-	0	0	0	0	0
	médio	0	0	0	0	0	0	-	-	-	-	0	0	0	0	0	0
	grande	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-
tem	2 patas	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0
	4 patas	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-
	pelos	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-
	cascos	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-
	crina/juba	0	0	0	0	0	0	0	0	0	-	0	0	-	-	-	0
	penas	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0
gosta de	caçar	0	0	0	0	-	-	-	0	-	-	-	-	-	0	0	0
	correr	0	0	0	0	0	0	0	0	-	-	0	-	-	-	-	0
	voar	-	0	0	-	-	-	-	0	0	0	0	0	0	0	0	0
	nadar	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0

Mapas Auto-Organizáveis



cão	.	.	.	raposa	.	.	gato	.	.	águia
.
.	coruja
.	tigre	.	.	.
lobo	falcão
.	.	.	.	leão
.	pombo
cavalo	galinha	.	.
.	.	.	.	vaca	ganso
zebra	pato	.	.

Mapa de características contendo neurônios rotulados com respostas mais fortes e suas respectivas entradas



Mapas Auto-Organizáveis



cão	cão	raposa	raposa	raposa	gato	gato	gato	água	água
cão	cão	raposa	raposa	raposa	gato	gato	gato	água	água
lobo	lobo	lobo	raposa	gato	tigre	tigre	tigre	coruja	coruja
lobo	lobo	leão	leão	leão	tigre	tigre	tigre	falcão	falcão
lobo	lobo	leão	leão	leão	tigre	tigre	tigre	falcão	falcão
lobo	lobo	leão	leão	leão	coruja	pombo	falcão	pombo	pombo
cavalo	cavalo	leão	leão	leão	pombo	galinha	galinha	pombo	pombo
cavalo	cavalo	zebra	vaca	vaca	vaca	galinha	galinha	pombo	pombo
zebra	zebra	zebra	vaca	vaca	vaca	galinha	galinha	pato	ganso
zebra	zebra	zebra	vaca	vaca	vaca	pato	pato	pato	ganso



Mapa semântico obtido através do uso de mapeamento simulado de penetração de eletrodo. O mapa é dividido em três regiões apresentando: pássaros, espécies pacíficas e caçadores

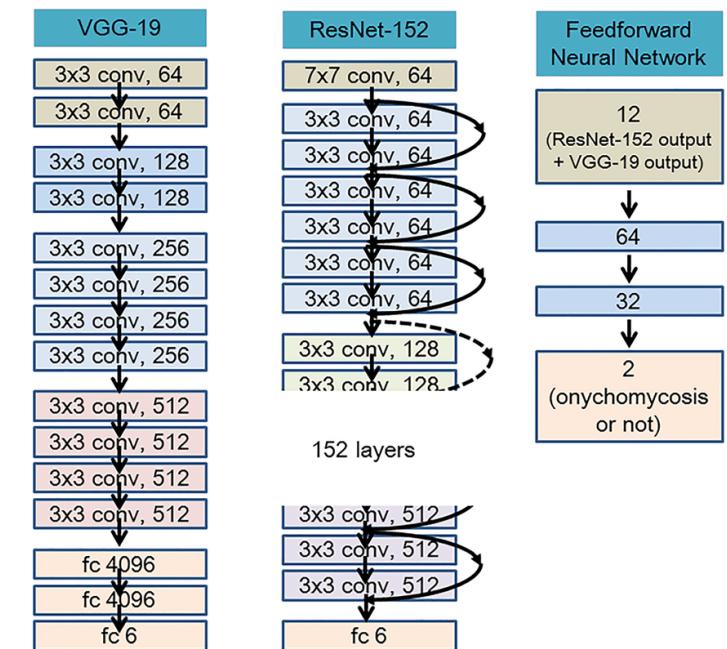
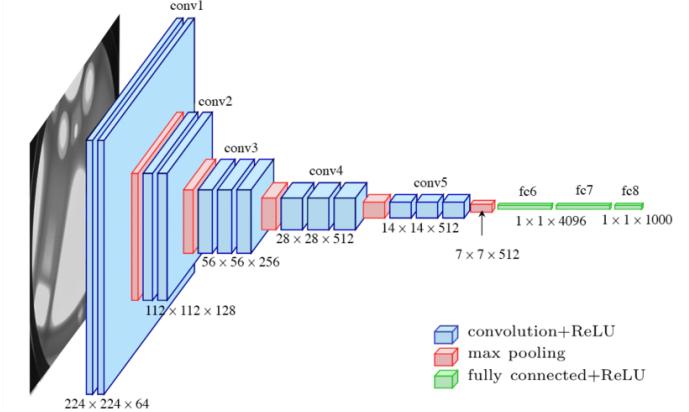
Redes Neurais Artificiais



REDES NEURAIS PROFUNDAS E OUTRAS REDES NEURAIS

Redes Neurais Profundas

- Deep Neural Networks; Deep Learning
- Redes Neurais com mais de três camadas (incluindo entrada e saída).
 - Redes Neurais Convolucionais com grande quantidade de camadas.
 - VGG (2014): 19 camadas, 144 milhões de parâmetros.
 - ResNet (2015): 152 camadas, 60 milhões de parâmetros.



Outras Redes Neurais

- Máquinas de Comitê
- Redes ART (Adaptive Resonance Theory)
- Redes de Hopfield
- Máquina de Boltzmann
- Redes de Crença Sigmóide
- Máquina de Helmholtz
- Echo States Networks
- Long / Short Term Memory (LSTM)
- Gated Recurrent Unit (GRU)
- Auto Encoder (AE)
- Variational AE (VAE)
- Denoising AE (DAE)
- Sparse AE (SAE)
- Deep Belief Network (DBN)
- Convolutional Neural Networks (CNN)
- Deconvolutional Networks (DN)
- Deep Convolutional Inverse Graphics Networks (DCIGN)
- Generative Adversarial Network (GAN)
- Liquid State Machine (LSM)
- Extreme Learning Machine (ELM)
- Deep Residual Network (DRN)
- Neural Turing Machine (NTM)

A mostly complete chart of

Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

○ Backfed Input Cell

○ Input Cell

△ Noisy Input Cell

○ Hidden Cell

○ Probabilistic Hidden Cell

△ Spiking Hidden Cell

○ Output Cell

○ Match Input Output Cell

○ Recurrent Cell

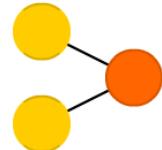
○ Memory Cell

△ Different Memory Cell

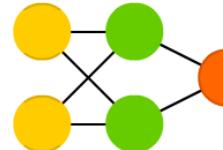
○ Kernel

○ Convolution or Pool

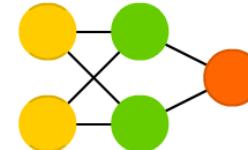
Perceptron (P)



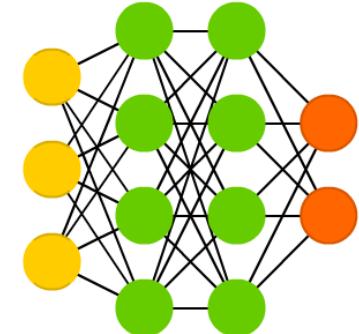
Feed Forward (FF)



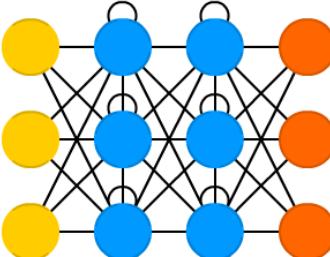
Radial Basis Network (RBF)



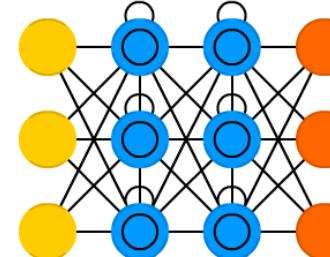
Deep Feed Forward (DFF)



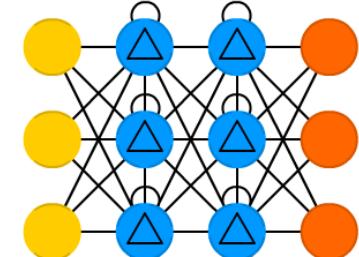
Recurrent Neural Network (RNN)



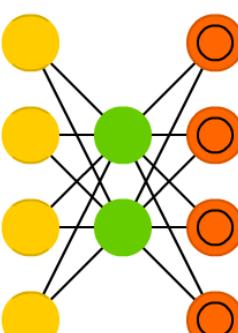
Long / Short Term Memory (LSTM)



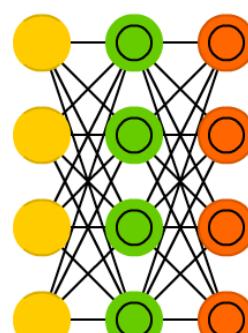
Gated Recurrent Unit (GRU)



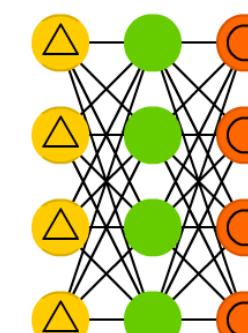
Auto Encoder (AE)



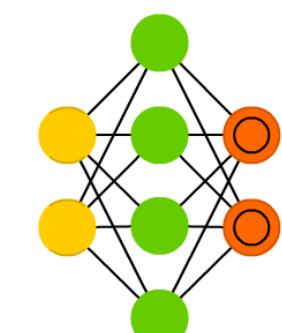
Variational AE (VAE)



Denoising AE (DAE)



Sparse AE (SAE)



○ Backfed Input Cell

○ Input Cell

△ Noisy Input Cell

● Hidden Cell

○ Probabilistic Hidden Cell

△ Spiking Hidden Cell

○ Output Cell

○ Match Input Output Cell

● Recurrent Cell

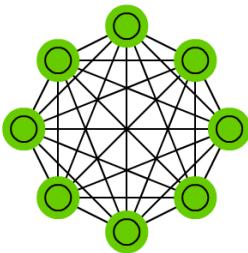
○ Memory Cell

△ Different Memory Cell

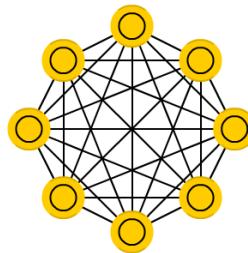
● Kernel

○ Convolution or Pool

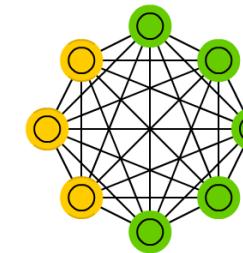
Markov Chain (MC)



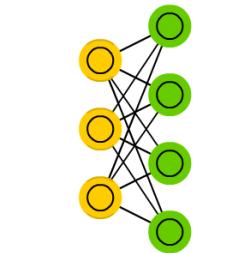
Hopfield Network (HN)



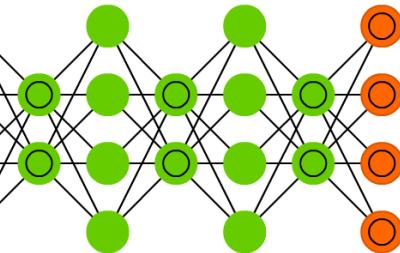
Boltzmann Machine (BM)



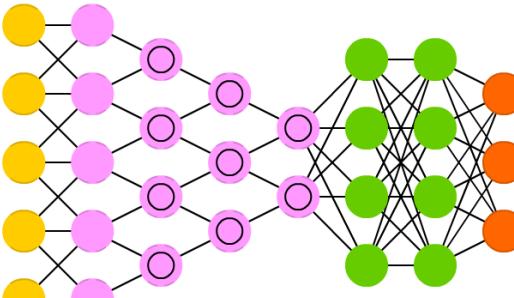
Restricted BM (RBM)



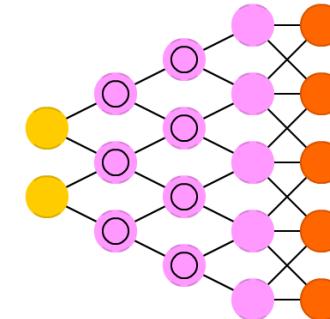
Deep Belief Network (DBN)



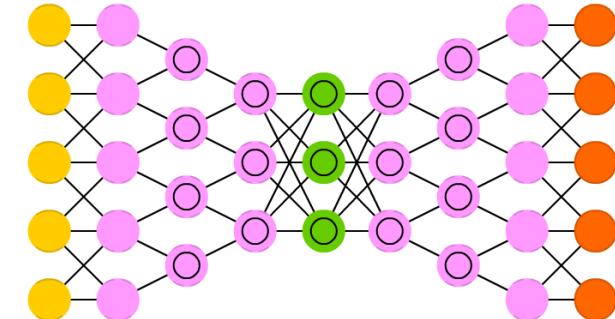
Deep Convolutional Network (DCN)



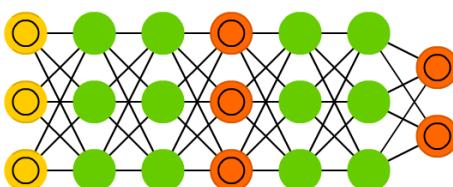
Deconvolutional Network (DN)



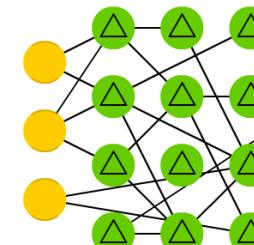
Deep Convolutional Inverse Graphics Network (DCIGN)



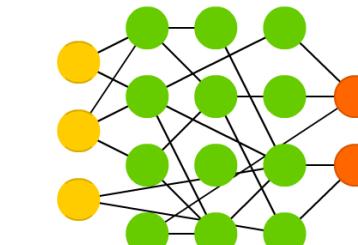
Generative Adversarial Network (GAN)



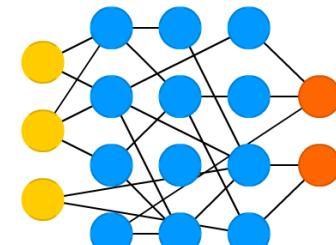
Liquid State Machine (LSM)



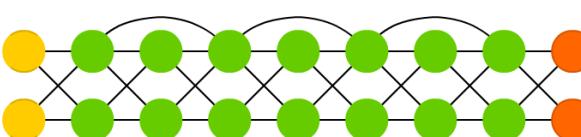
Extreme Learning Machine (ELM)



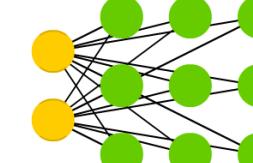
Echo State Network (ESN)



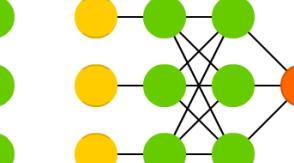
Deep Residual Network (DRN)



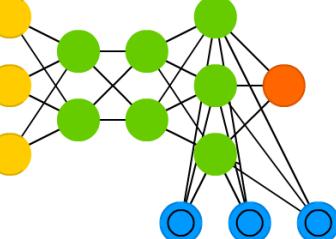
Kohonen Network (KN)



Support Vector Machine (SVM)



Neural Turing Machine (NTM)



Redes Neurais Artificiais



APLICAÇÕES DE REDES NEURAIS ARTIFICIAIS

Aplicações de Redes Neurais Artificiais

- Reconhecimento de voz e imagem
- Diagnóstico médico
- Controle de robôs
- Controle de processos químicos
- Otimização
- Finanças
- Fusão de sensores
- Bioinformática
- Carros Autônomos
- Predição de Próximas Palavras (auto-completar)
- Classificação e Categorização de Texto



<https://cheekymonkey.co.uk/voice-activated-computers-future/>

RNAs na Indústria

- RNAs são muito utilizadas em indústrias no exterior.
 - Especialmente quando instalação rápida ou grande velocidade de operação é requerida.
 - Ex.: Caixas de peixes congelados da *Birds Eye* são inspecionados visualmente por uma RNA.
 - Caixas imperfeitas são rejeitadas para exportação.



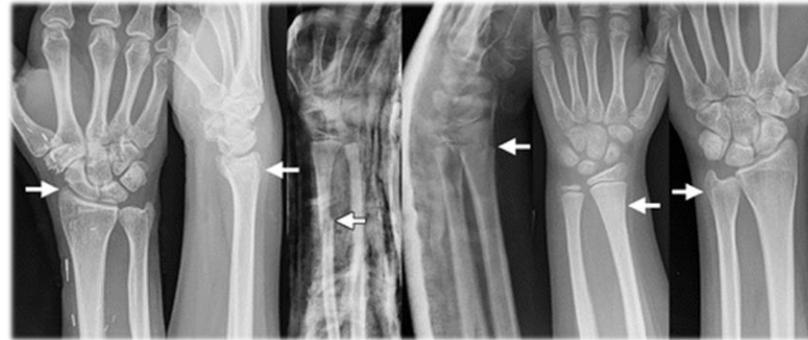
RNAs no Setor de Serviços

- Grande quantidade de pesquisas sobre a utilização de RNAs para previsão financeira.
 - Tendências sugerem expansão destas aplicações.
 - Ex.: Cartão de crédito VISA utiliza RNAs para liberação de cartões e detecção de fraudes.

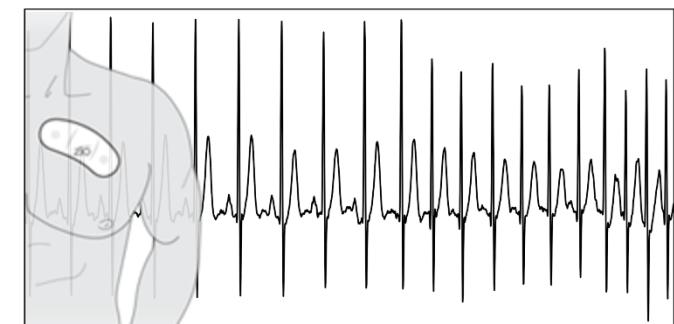


RNAs na Medicina

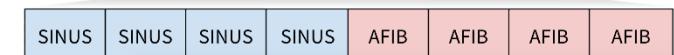
- RNAs têm sido utilizadas para:
 - Diagnóstico de doenças
 - Rede treinada para, dado um conjunto de sintomas, diagnosticar a doença e sugerir melhor tratamento.
 - Câncer
 - Diabetes
 - Reconhecimento de imagens médicas
 - Reconhecimento de fraturas.
 - Reconhecimento de anomalias no coração.



<https://pubs.rsna.org/doi/10.1148/rjai.2019180001>



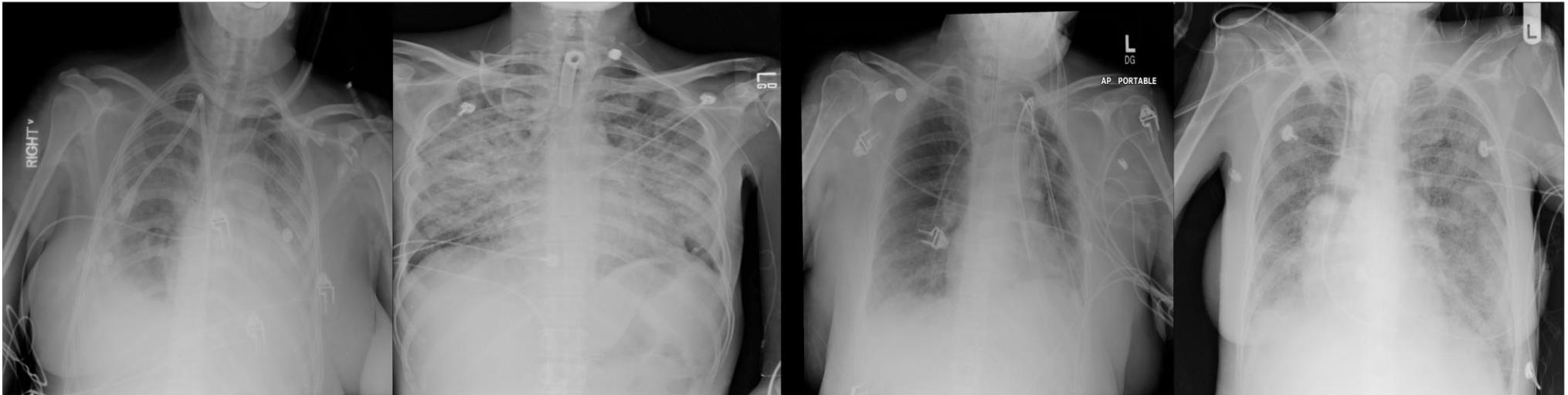
34-layer Convolutional Neural Network



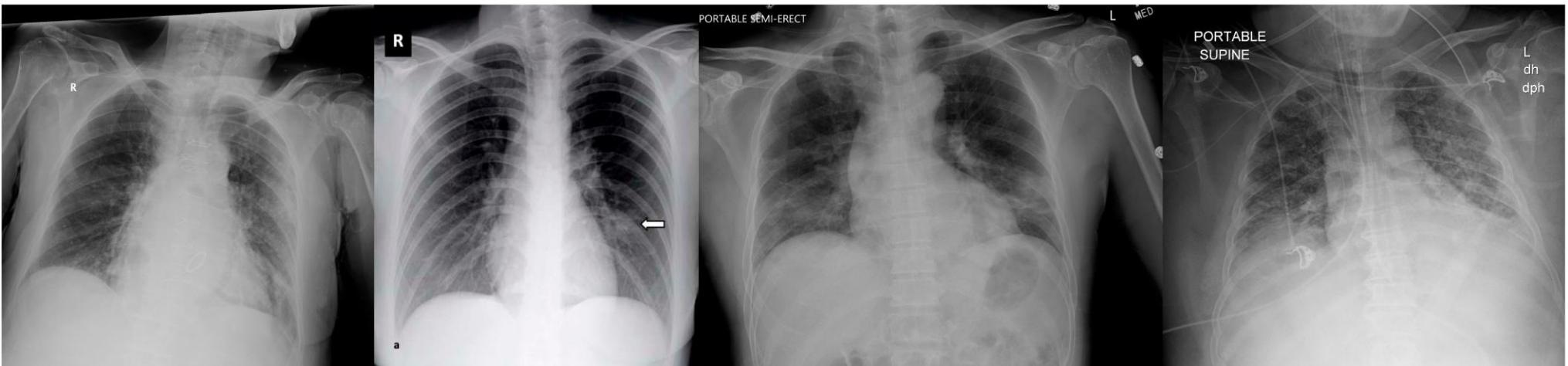
<https://www.arxiv-vanity.com/papers/1707.01836/>

RNAs na Detecção de COVID-19 em Imagens de Radiografia de Tórax

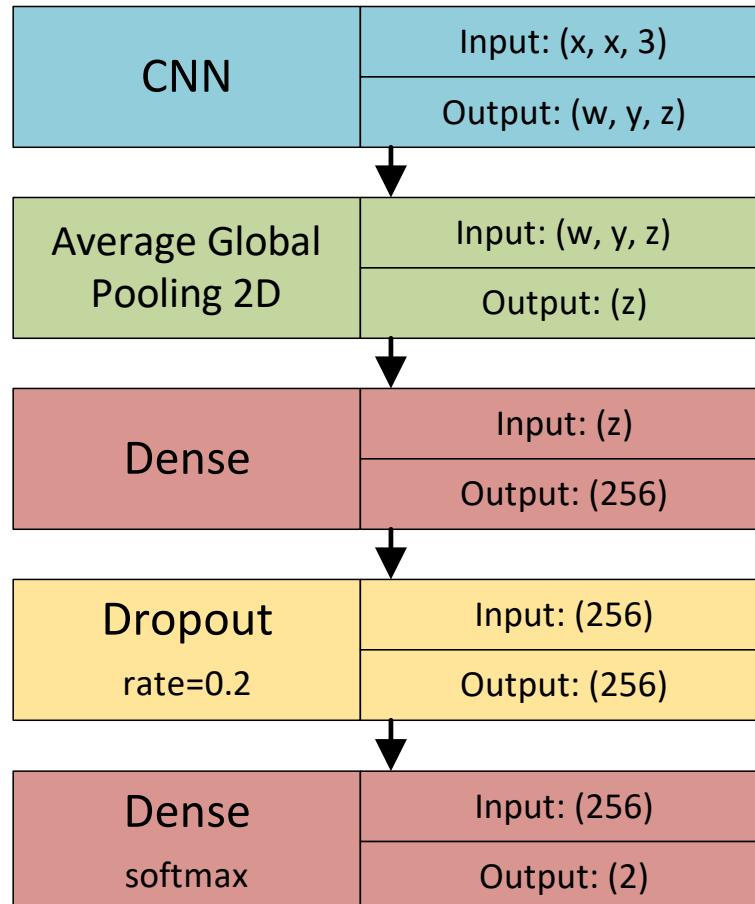
Negativo



Positivo



RNAs na Detecção de COVID-19 em Imagens de Radiografia de Tórax

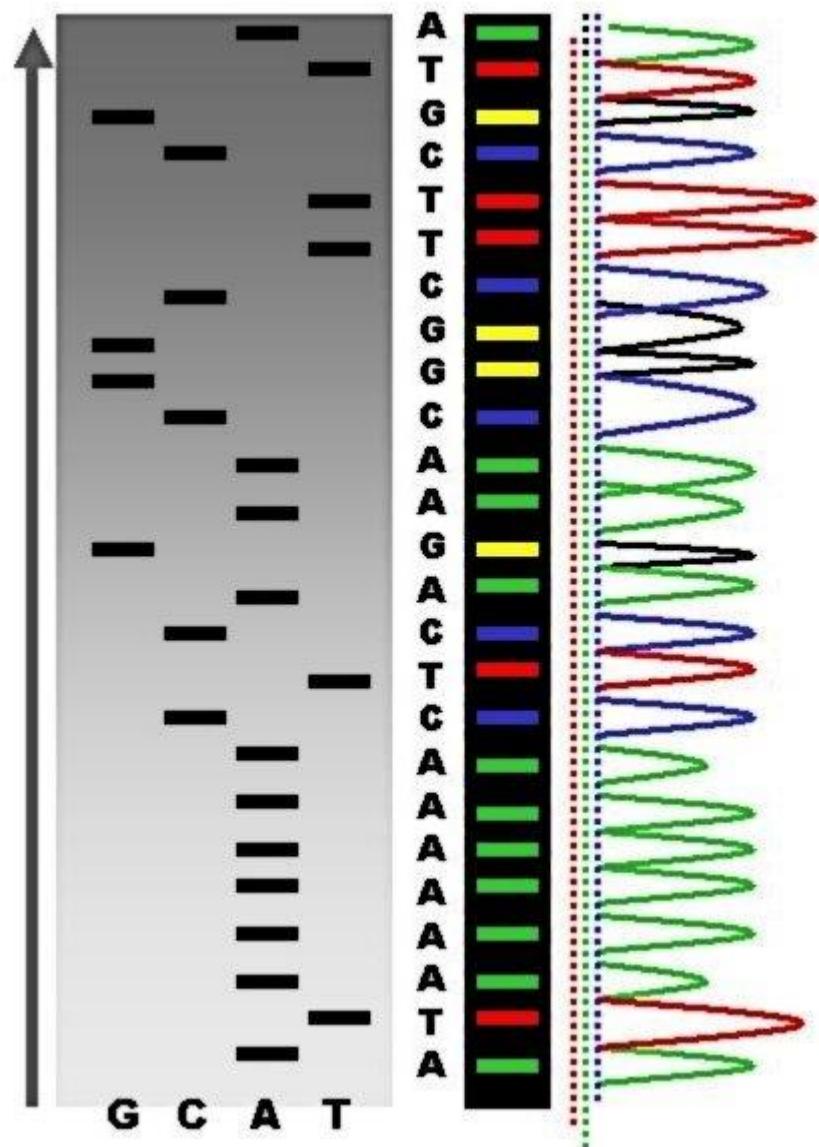


Model	ACC		TPR		PPV		F1	
	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
DenseNet169	0,9815	0,0056	0,9700	0,0138	0,9930	0,0075	0,9812	0,0058
EfficientNetB2	0,9760	0,0049	0,9600	0,0141	0,9918	0,0051	0,9756	0,0052
InceptionResNetV2	0,9755	0,0099	0,9590	0,0246	0,9919	0,0051	0,9749	0,0106
InceptionV3	0,9750	0,0065	0,9520	0,0144	0,9979	0,0041	0,9744	0,0069
MobileNet	0,9710	0,0060	0,9430	0,0136	0,9990	0,0021	0,9701	0,0064
EfficientNetB0	0,9705	0,0051	0,9510	0,0086	0,9896	0,0033	0,9699	0,0053
EfficientNetB3	0,9700	0,0163	0,9470	0,0337	0,9927	0,0051	0,9690	0,0177
DenseNet201	0,9695	0,0176	0,9400	0,0342	0,9989	0,0022	0,9683	0,0186
ResNet152V2	0,9695	0,0244	0,9420	0,0510	0,9970	0,0040	0,9679	0,0268
ResNet152	0,9660	0,0223	0,9370	0,0443	0,9947	0,0033	0,9644	0,0243
DenseNet121	0,9630	0,0053	0,9270	0,0103	0,9989	0,0022	0,9616	0,0057
Xception	0,9615	0,0077	0,9230	0,0154	1,0000	0,0000	0,9599	0,0083
VGG19	0,9580	0,0198	0,9170	0,0385	0,9989	0,0023	0,9558	0,0216
EfficientNetB1	0,9570	0,0224	0,9240	0,0413	0,9892	0,0075	0,9551	0,0242
ResNet50	0,9545	0,0172	0,9090	0,0344	1,0000	0,0000	0,9520	0,0192
VGG16	0,9525	0,0123	0,9090	0,0282	0,9958	0,0052	0,9501	0,0138
ResNet101V2	0,9530	0,0302	0,9100	0,0643	0,9959	0,0050	0,9497	0,0342
MobileNetV2	0,9485	0,0172	0,9030	0,0359	0,9935	0,0019	0,9457	0,0190
ResNet101	0,9410	0,0170	0,8830	0,0333	0,9988	0,0023	0,9370	0,0190
ResNet50V2	0,9280	0,0075	0,8590	0,0153	0,9966	0,0046	0,9226	0,0087
NASNetMobile	0,8530	0,0653	0,7090	0,1317	0,9960	0,0034	0,8212	0,0918
Average	0,9569	0,0162	0,9178	0,0334	0,9957	0,0036	0,9536	0,0187

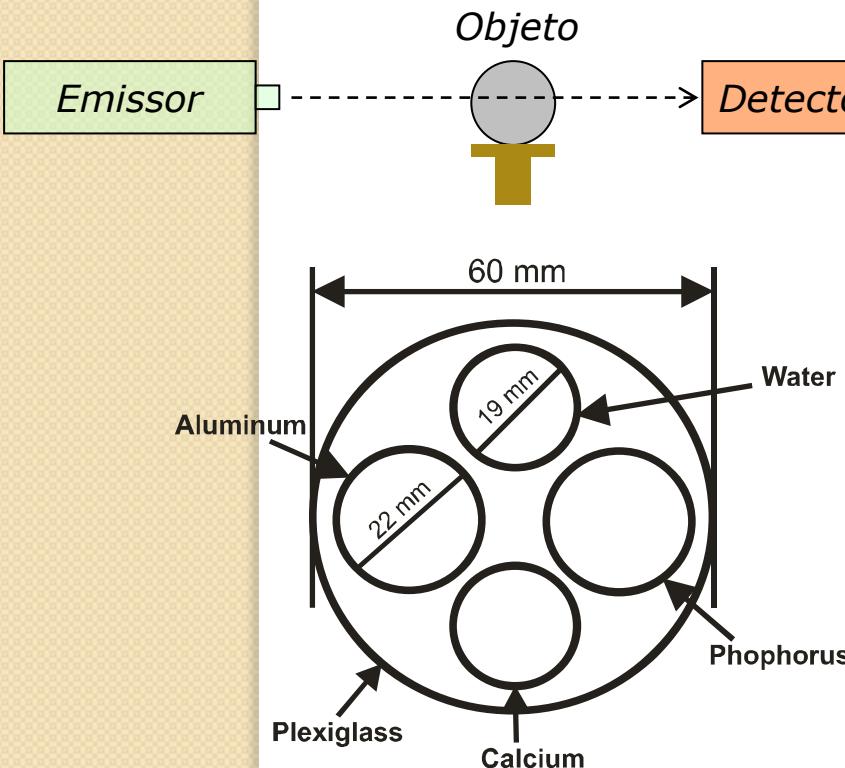
BREVE, Fabricio Aparecido. **COVID-19 detection on Chest X-ray images: A comparison of CNN architectures and ensembles.** *Expert Systems With Applications*, v. 204, p.117549, 2022.
<https://doi.org/10.1016/j.eswa.2022.117549>

RNAs na Bioinformática

- Existem trabalhos que utilizam RNAs para:
 - Reconhecimento de genes em sequências de DNA.
 - Análise de expressão gênica.
 - Previsão da estrutura de proteínas.

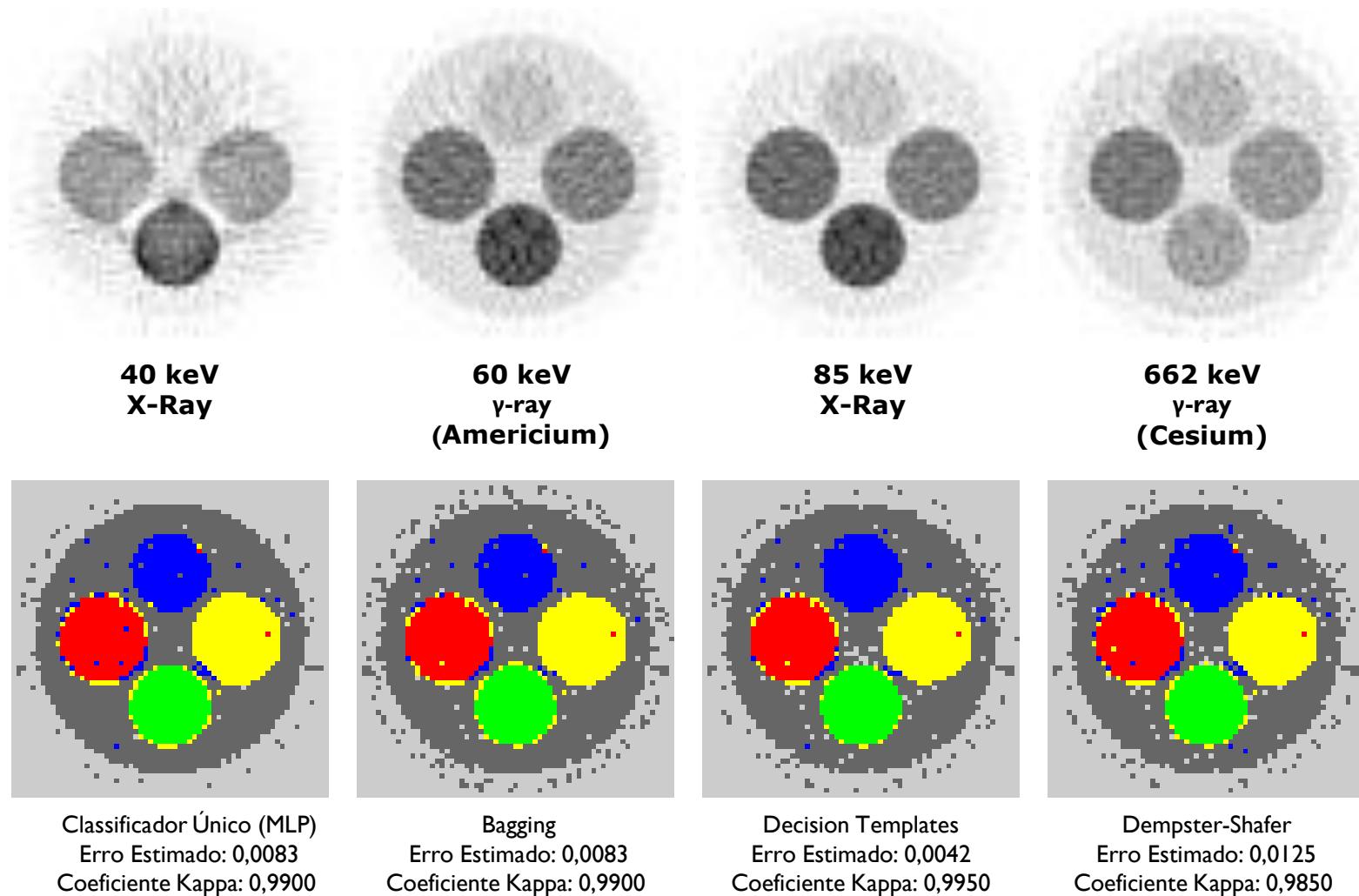


RNAs nas Ciências dos Solos



BREVE, Fabricio Aparecido ; PONTI JUNIOR, Moacir Pereira ; MASCARENHAS, Nelson Delfino Dávila. *Combining Methods to Stabilize and Increase Performance of Neural Network-Based Classifiers*. In: SIBGRAPI – Brazilian Symposium on Computer Graphics and Image Processing, 2005, Natal. SIBGRAPI '05: Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing. Washington, DC, USA : IEEE Computer Society, 2005. p. 105-111.
<https://dx.doi.org/10.1109/SIBGRAPI.2005.19>

BREVE, Fabricio Aparecido ; PONTI JUNIOR, Moacir Pereira ; MASCARENHAS, Nelson Delfino Dávila. *Multilayer Perceptron Classifier Combination for Identification of Materials on Noisy Soil Science Multispectral /images*. In: Brazilian Symposium on Computer Graphics and Image Processing, 2007, Belo Horizonte. *Proceedings of XX Brazilian Symposium on Computer Graphics and Image Processing*. Los Alamitos, CA : IEEE Computer Society, 2007. p. 239-244.
<https://dx.doi.org/10.1109/SIBGRA.2007.4368190>



RNAs no Auxílio à Deficientes Visuais



BREVE, Fabricio Aparecido;
FISCHER, Carlos
Norberto.

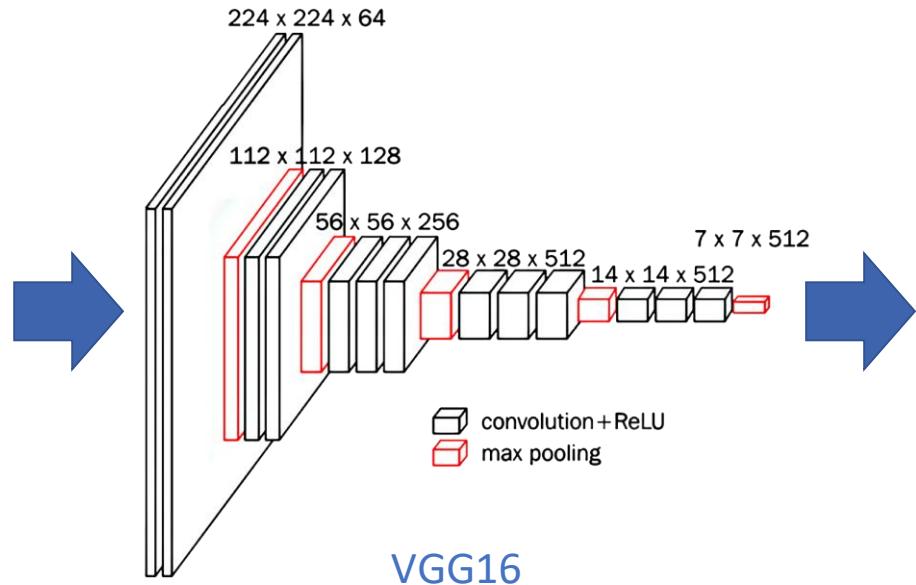
Visually Impaired Aid using Convolutional Neural Networks, Transfer Learning, and Particle Competition and Cooperation In: 2020

International Joint Conference on Neural Networks (IJCNN 2020), 2020, Glasgow, UK. *Proceedings of 2020 International Joint Conference on Neural Networks (IJCNN 2020)*, 2020.

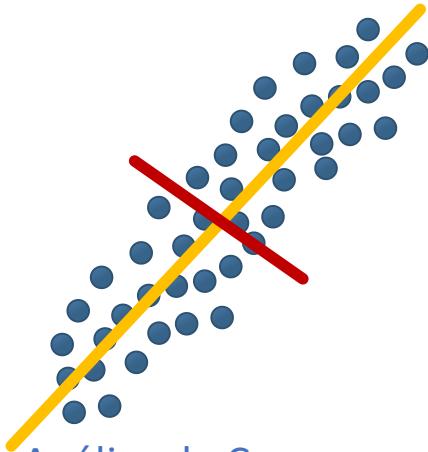
<https://doi.org/10.1109/IJCNN48605.2020.9207606>



Imagens
 $224 \times 224 \times 3$



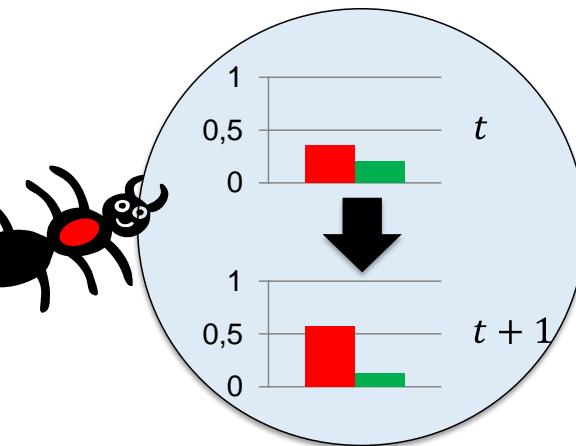
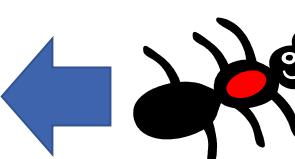
VGG16



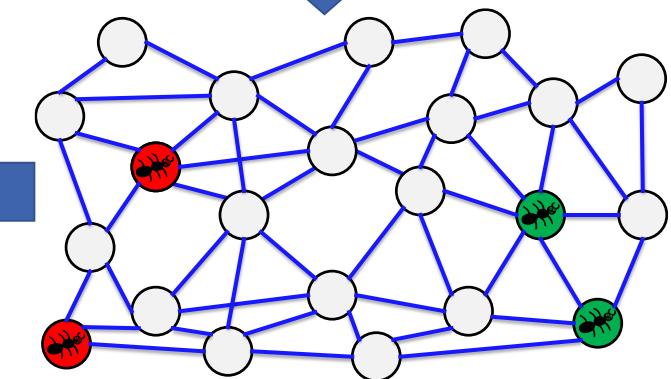
Análise de Componentes Principais



Imagens
Classificadas



Competição e Cooperação
entre Partículas

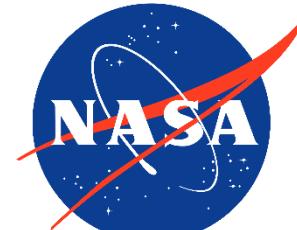


Grafo Sem Peso Não
Direcionado

Framework proposto para o problema de detecção de obstáculos para auxílio à deficientes visuais, usando o modelo de Competição e Cooperação entre Partículas para a classificação semi-supervisionada com o VGG16 como extrator de atributos

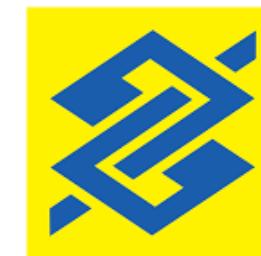
Empresas pioneiras no uso de RNAs

- General Motors
- Mercedes Benz
- Nasa
- AT&T
- British Telecom
- Lucent Technology
- Citibank
- Glaxo
- British Aerospace
- Rhodia (Grupo Solvay)



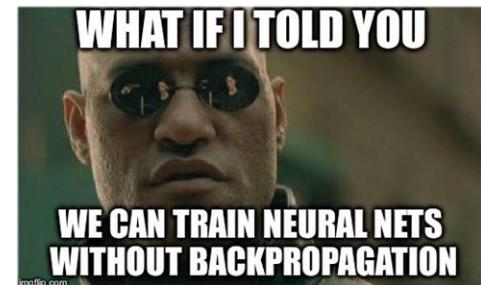
Empresas pioneiras no uso de RNAs no Brasil

- Usiminas
- Marinha do Brasil
- Petrobrás
- CEPEL
- Rhodia
- Belgo-Mineira
- Itaipu
- Embraer
- Furnas
- CHESF
- Banco Itaú
- Banco do Brasil



Conclusão

- Redes Neurais têm sido bem sucedidas em problemas como Reconhecimento de Padrões, dentre muitos outros.
- Alguns problemas:
 - Como definir valores de seus hiperparâmetros?
 - Otimização.
 - Algoritmos Genéticos.
 - Como explicar as decisões tomadas pela rede?
 - Extração de conhecimento.



Bibliografia

- CASTRO, Leandro Nunes. *Fundamentals of Natural Computing: Basic Concepts, Algorithms, And Applications.* CRC Press, 2006.
- CARVALHO, André Ponce de Leon F. de. *Notas de Aula,* 2007.
- BROWNLEE, Jason. *Clever Algorithms: Nature-Inspired Programming Recipes.* Jason Brownlee, 2011.
- HAYKIN, Simon. *Neural Networks and Learning Machines, 3rd Edition.* Prentice Hall, 2008.
- KOVACS, Zsolt L. *Redes Neurais Artificiais: Fundamentos e Aplicações.* Livraria da Física, 2006.
- BISHOP, Christopher M. *Pattern Recognition and Machine Learning.* Springer, 2007.

