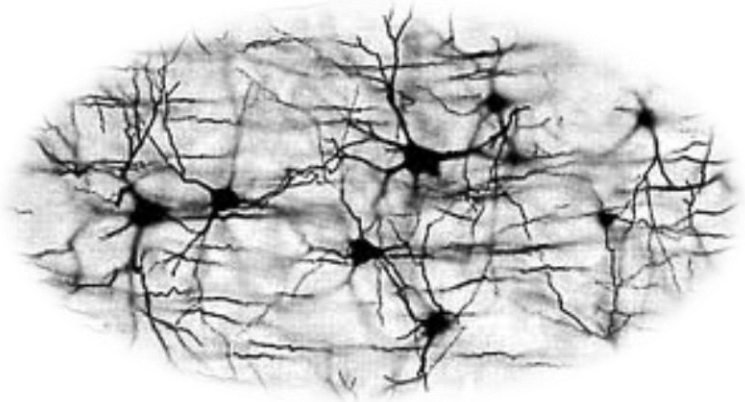
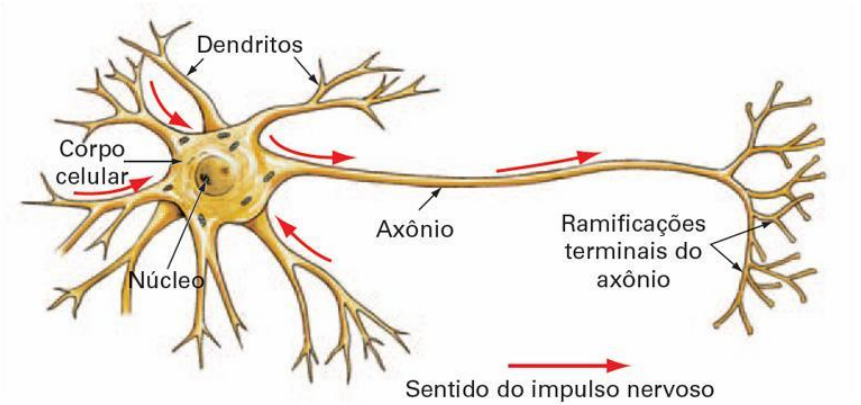
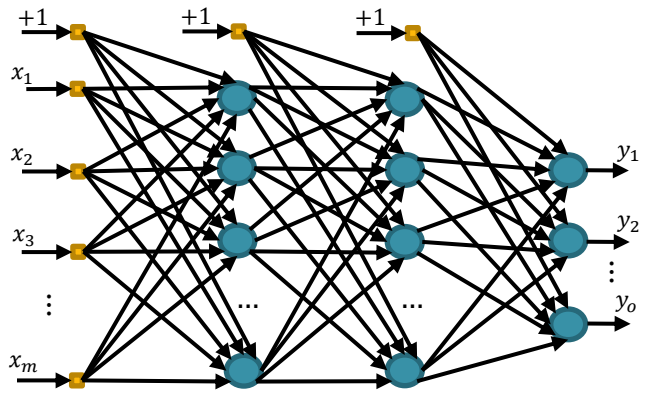
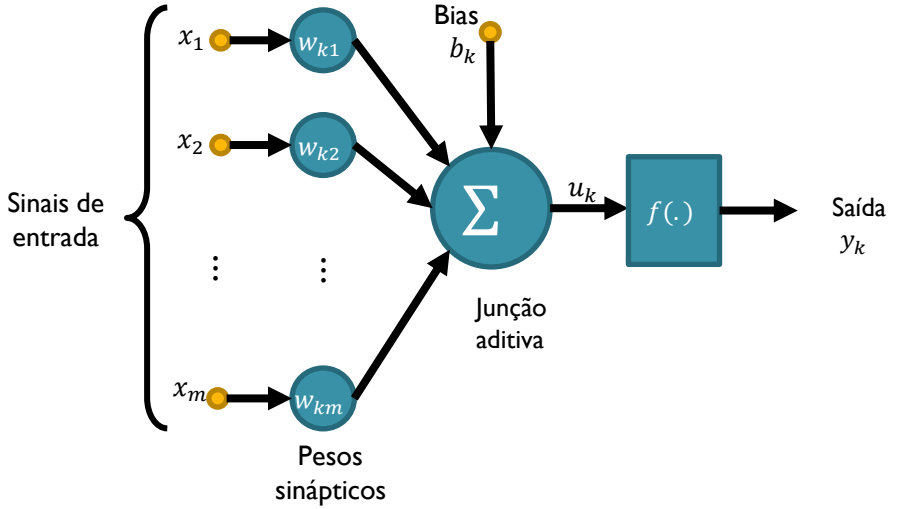
Redes Neurais Artificiais

Parte 2





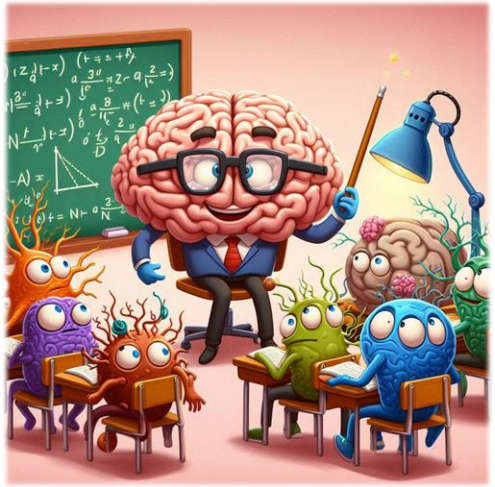


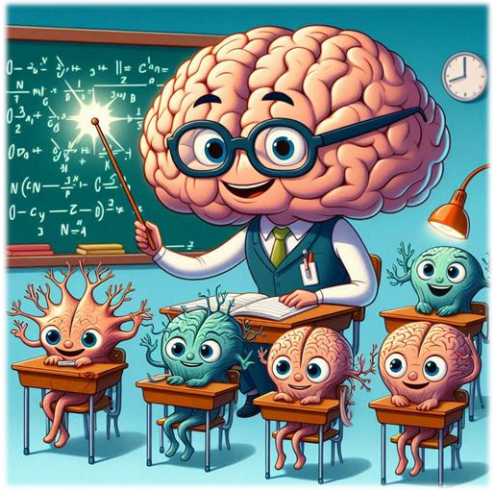


Fabricio Breve – www.fabriciobreve.com

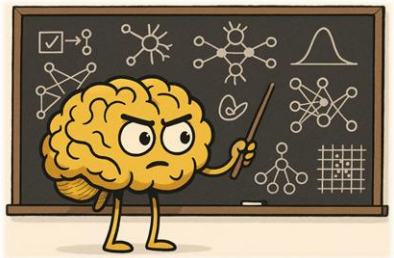
Aula Anterior

- Por que Redes Neurais?
- Redes Neurais Artificiais
- Neurônios
  - Neurônio Natural
  - Neurônios Artificiais
  - Neurônio de McCulloch e Pitts
  - Neurônio Genérico
  - Neurônio Genérico Matematicamente
- Bias
  - Finalidade do Termo Bias
- Funções de Ativação
  - Função Linear
  - Função de Limiar (*Threshold*, *Step*)
  - Função Sigmóide / Logística
  - Função Tangente Hiperbólica
  - Função de Base Radial
  - ReLU (*Rectified Linear Unit*)
  - Outras Funções de Ativação
- Arquiteturas de Redes Neurais
  - Redes *Feedforward* de Uma Única Camada
  - Redes *Feedforward* de Múltiplas Camadas
  - Redes Recorrentes
- Abordagens de Aprendizado
  - Aprendizado Supervisionado
    - Exemplo: Problema de Classificação - Detecção de SPAM
    - Exemplo: Problema de Regressão - Como calcular o preço de uma casa?
    - Treinando uma Rede Neural
    - Função de Custo
    - Capacidade de Generalização
    - *Underfitting* e *Overfitting*
    - Estimando o Desempenho da Rede
    - Validação Cruzada
    - Matriz de Confusão
  - Aprendizado Não Supervisionado
    - Aprendizado Competitivo
    - Exemplo: Separação de Balões
    - Exemplo: Onde abrir pizzarias?
  - Aprendizado por Reforço





Agenda



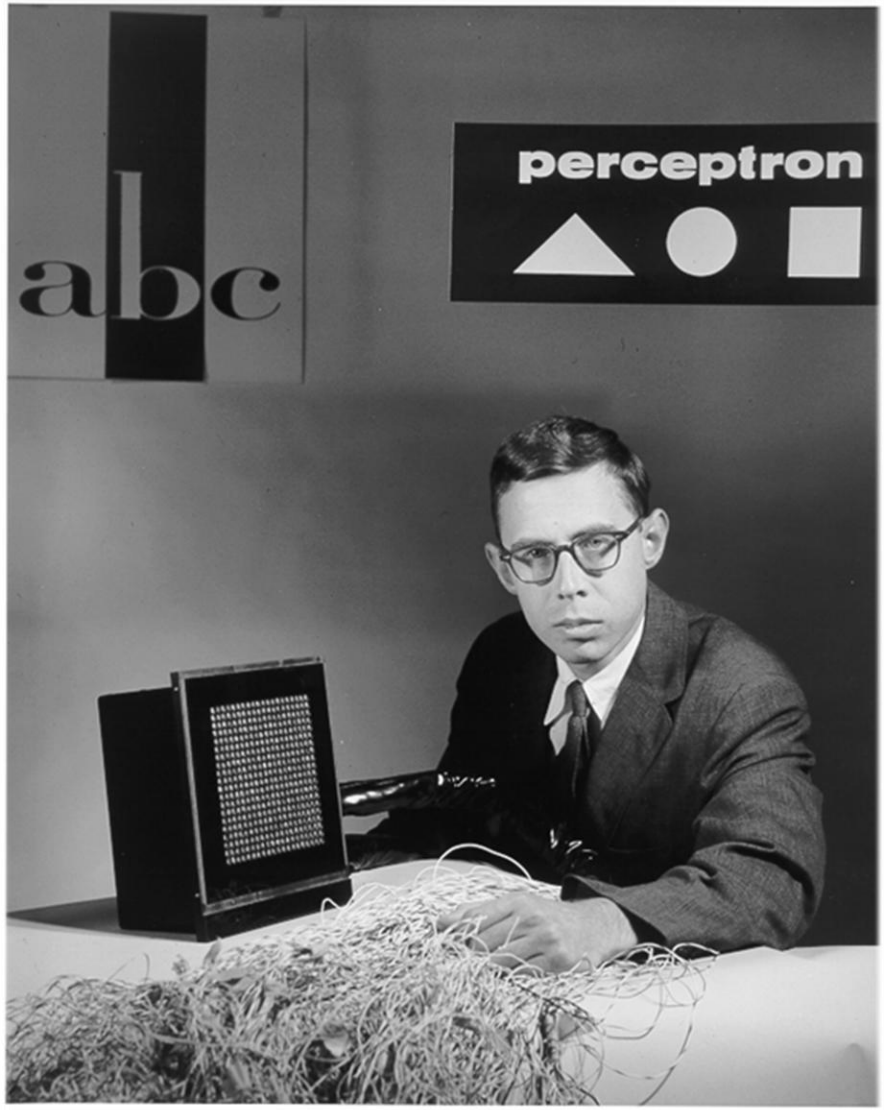
- Perceptron de Uma Única Camada
- Separabilidade Linear
- Perceptron Simples para Classificação de Padrões
- Treinamento
- Perceptron de Múltiplas Saídas para Classificação de Padrões
- Um-versus-Resto
- Função Softmax
- Exemplo de Aplicação
- Saída Desejada
- Ciclos (Épocas)
- Pesos Finais
- Exercício
- Generalização
- Adaline
- Perceptron de Múltiplas Camadas
- Algoritmo de Retropagação de Erro
- Por que Múltiplas Camadas?
- Dificuldades de Aprendizado
- Atualização de Pesos
- Redes de Função de Base Radial
- Mapas Auto-Organizáveis
- Mapa de Kohonen
- Redes Neurais Profundas
- Outras Redes Neurais
- Aplicações de Redes Neurais Artificiais
- Empresas pioneiras no uso de RNAs
- Conclusão

Redes Neurais Artificiais

PERCEPTRON DE UMA ÚNICA CAMADA

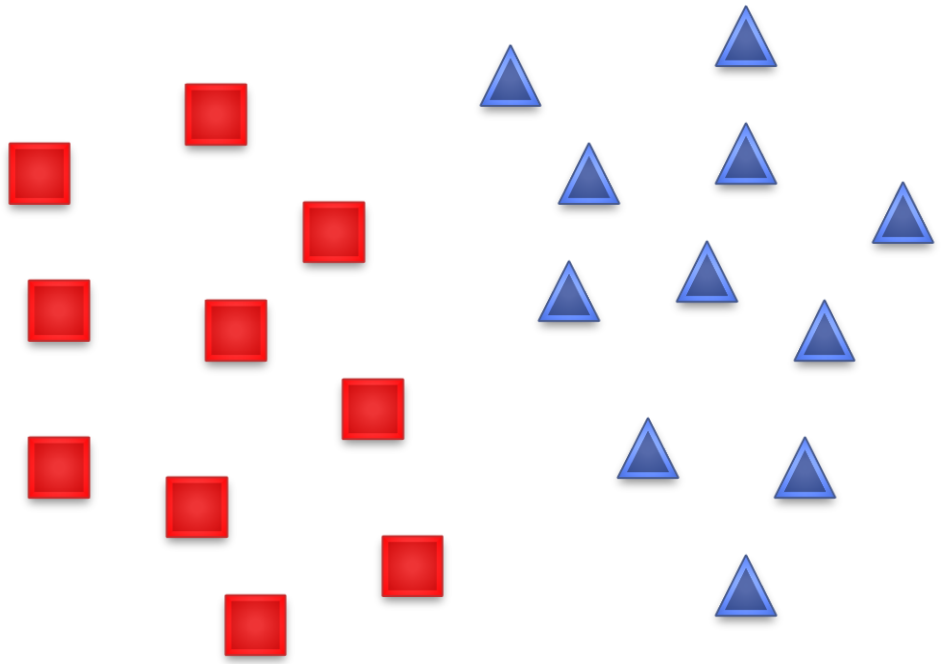
Perceptron

- 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.

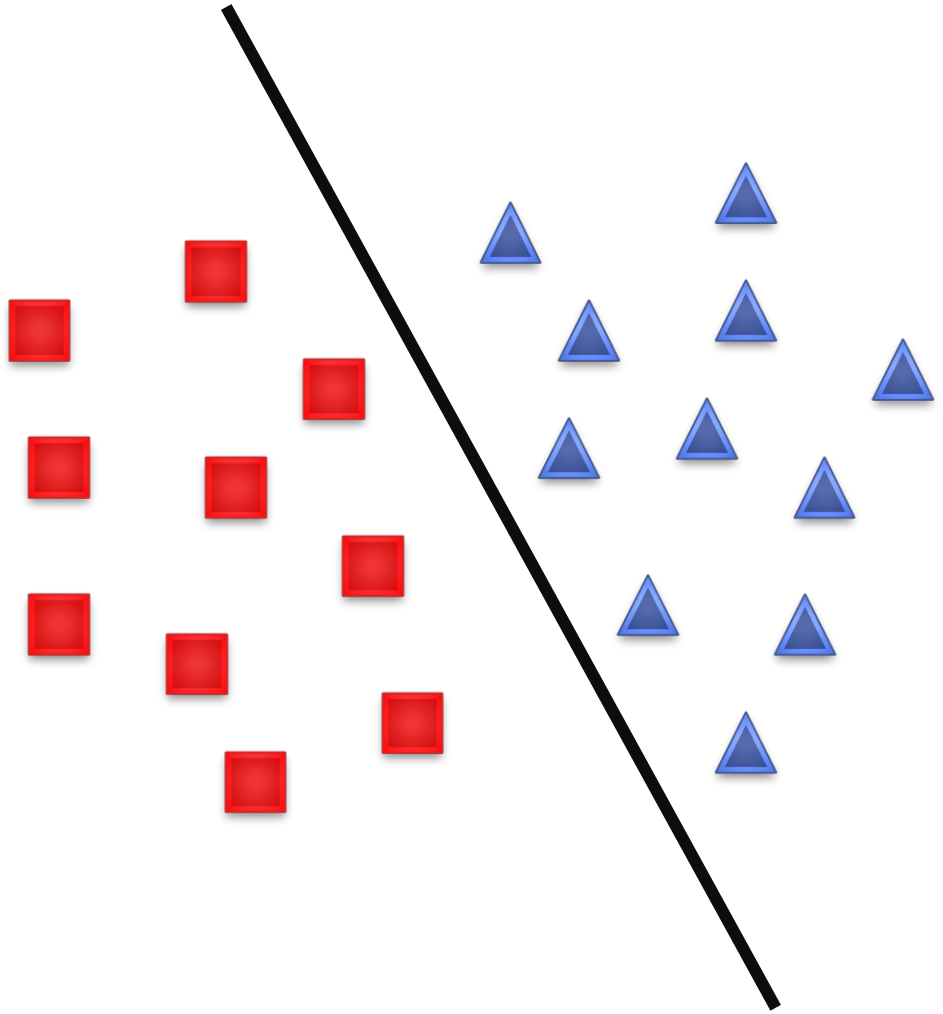


Frank Rosenblatt [*1928 – †1971]

Separabilidade Linear

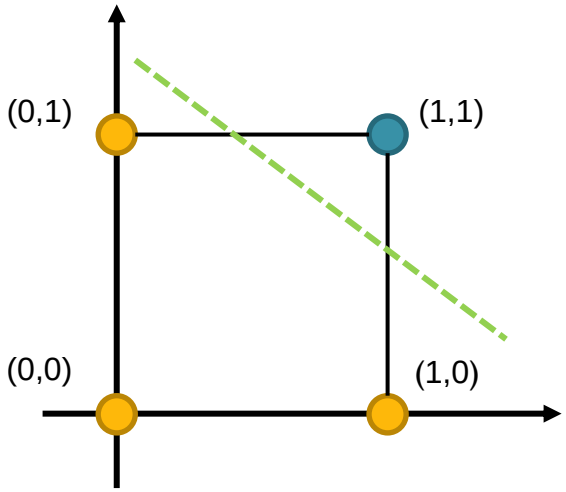


Separabilidade Linear

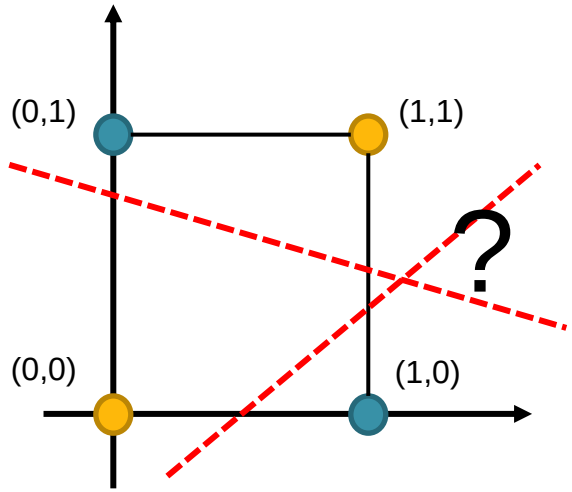


Separabilidade Linear

Entradas		Saída
x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

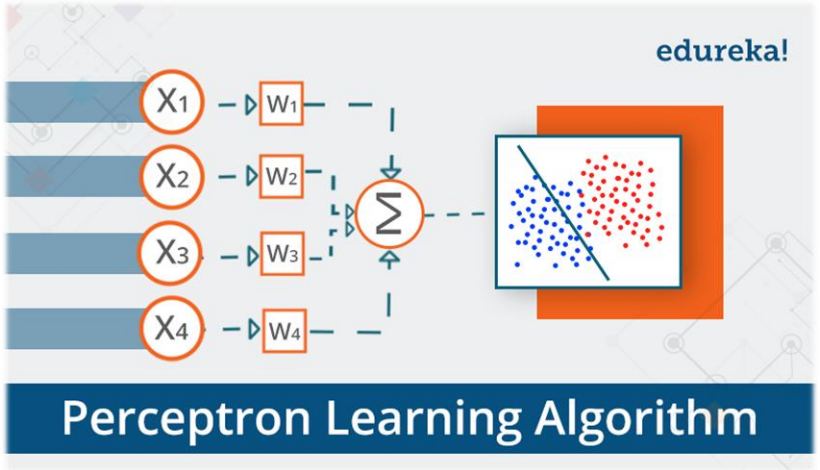


Entradas		Saída
x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0



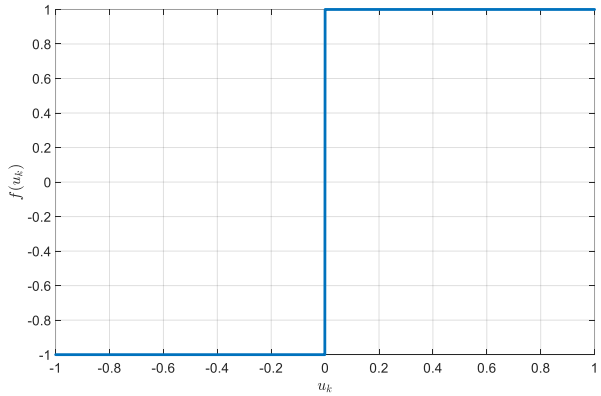
Perceptron Simples para Classificação de Padrões

- 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*.



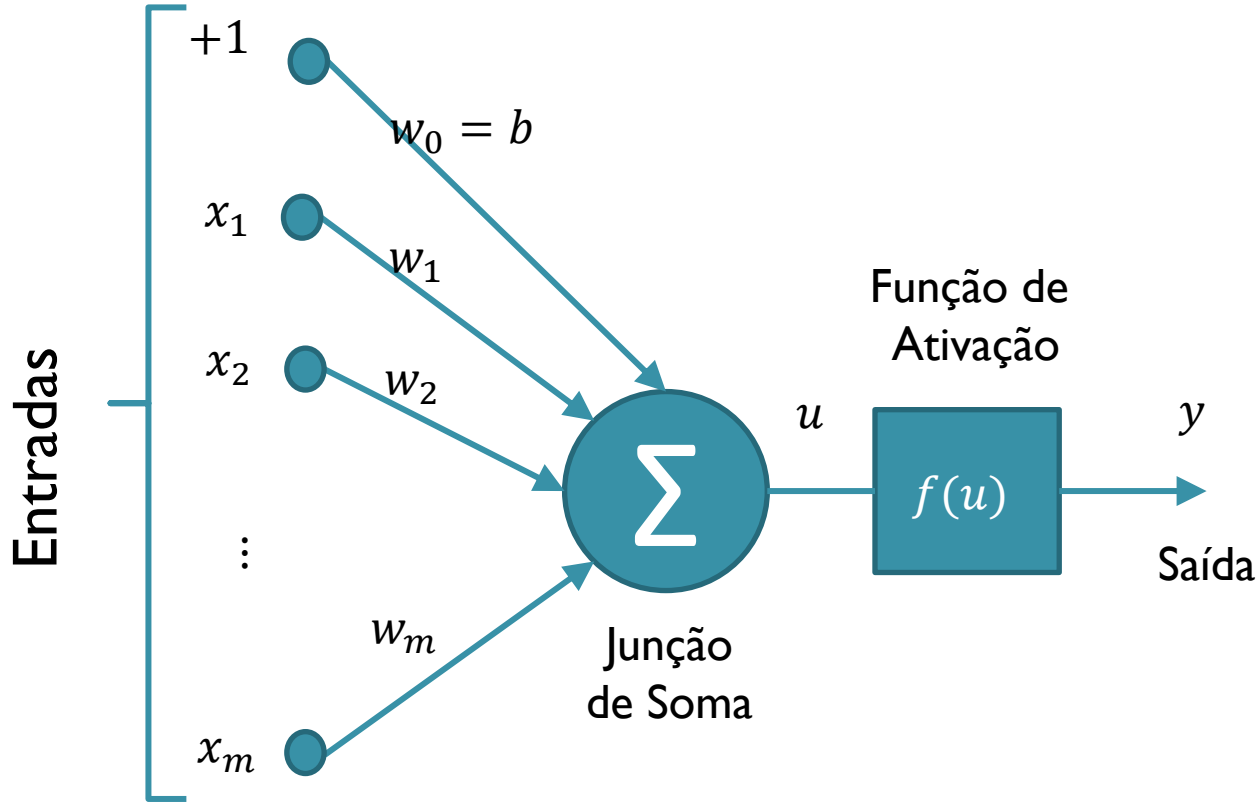
Perceptron Simples para Classificação de Padrões

- 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.

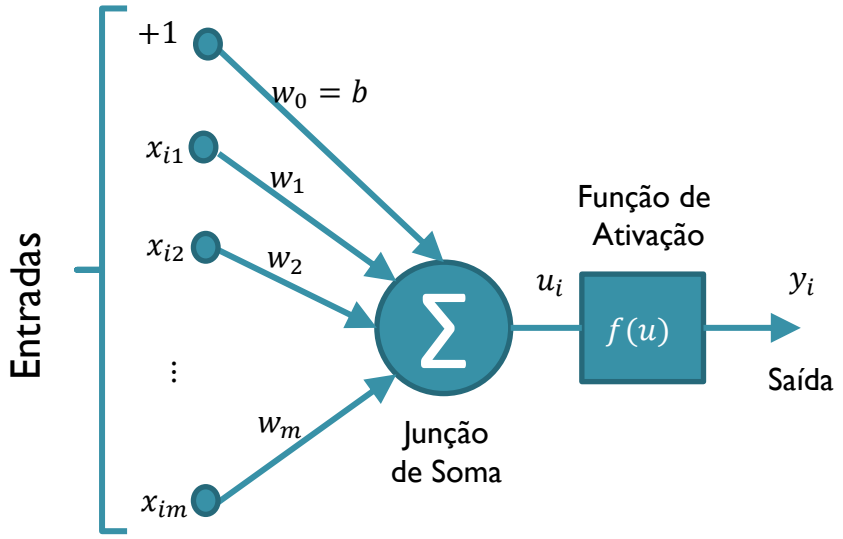
Perceptron Simples para Classificação de Padrões



Perceptron mais simples para realizar classificação de padrões

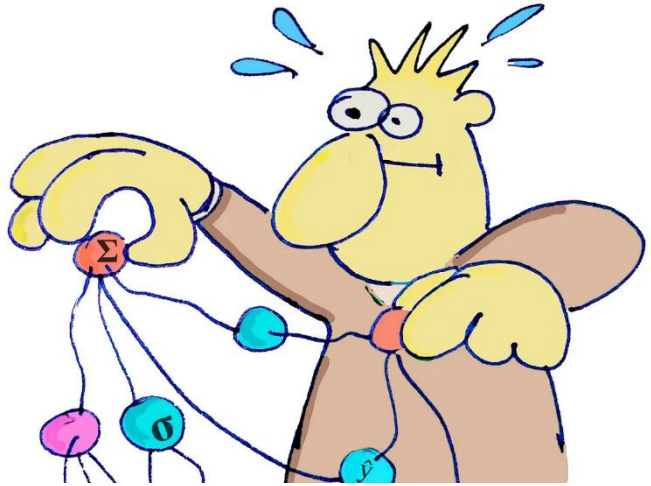
Treinamento do Perceptron

- 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 $\mathbf{w} \in \Re^{1\times m}$, $\mathbf{x}_i \in \Re^{1\times m}$, e $b \in \Re^{1\times 1}$
        - m é a quantidade de dimensões (atributos) de cada padrão.
        - α é a taxa de aprendizado.



Algoritmo de Treinamento do Perceptron

- $\mathbf{X} \in \Re^{N \times m}$ é a matriz de N padrões (exemplos) 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.



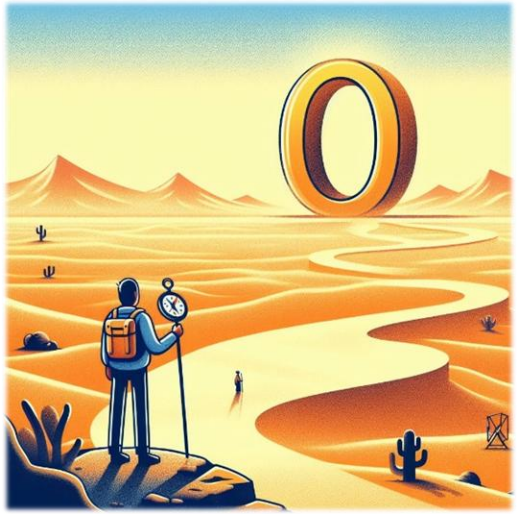
Algoritmo de Treinamento do Perceptron

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

Algoritmo de aprendizado de um perceptron simples. A função $f(.)$ 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.

Algoritmo de Treinamento do Perceptron

- Importante:
  - Nem sempre o erro chegará a zero!
    - Problema não linearmente separável.
    - Taxa de aprendizado muito alta.
      - Algoritmo oscila.
    - Etc.
  - Considere usar um critério de parada alternativo.
    - Ex.: Calcular o erro médio quadrático (E/N) no final de cada época (iteração).
      - MSE (*Mean Squared Error*) na sigla em inglês.
      - Guardar o menor valor obtido e parar quando ele não diminuiu nas últimas p épocas.



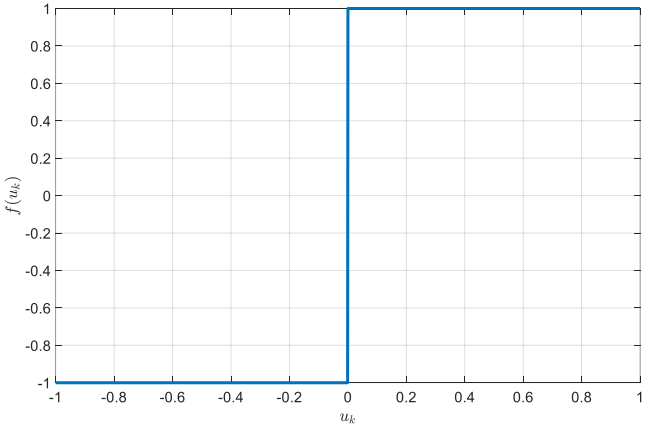
Exemplo de Aplicação

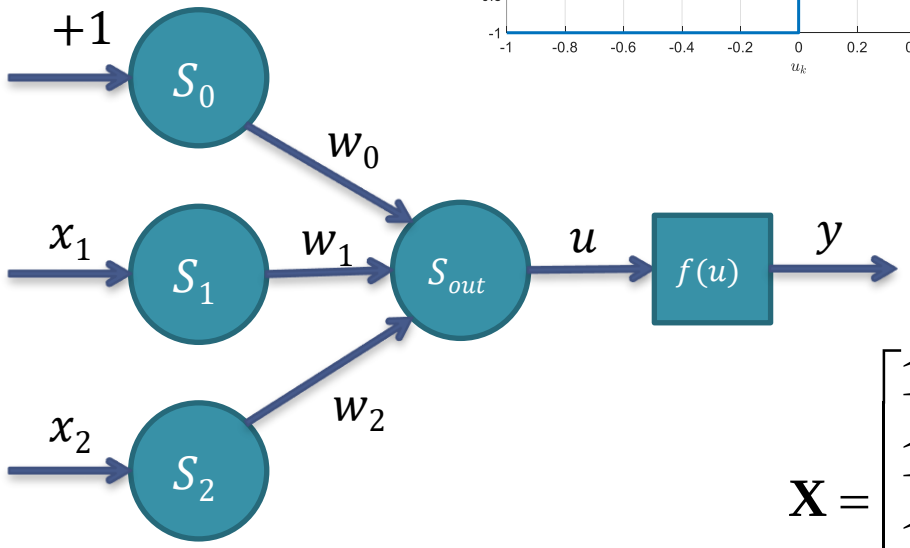
AND	x_0	x_1	x_2	d
Entrada 1:	1	0	0	0
Entrada 2:	1	0	1	0
Entrada 3:	1	1	0	0
Entrada 4:	1	1	1	1

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

Função de ativação de limiar:

$$u > 0 \Longrightarrow y = 1$$
$$u \leq 0 \Longrightarrow y = 0$$



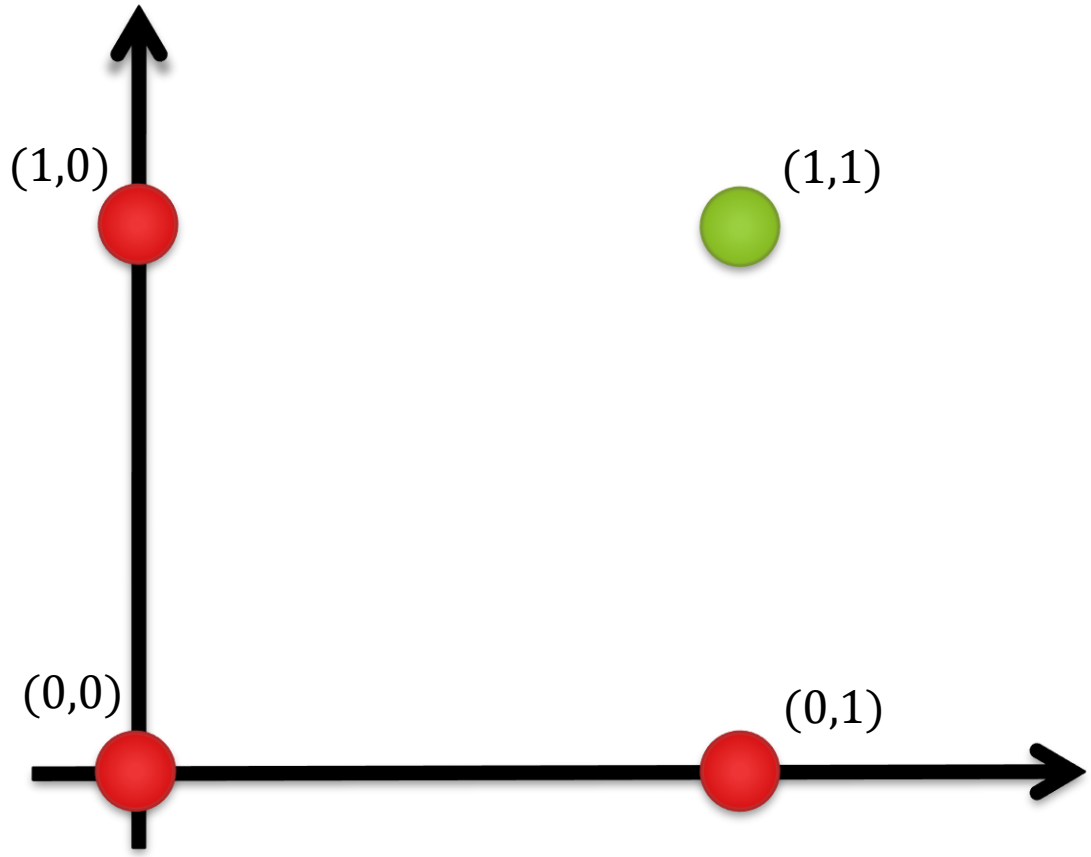


Lembre-se:
$x_0 = b$

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

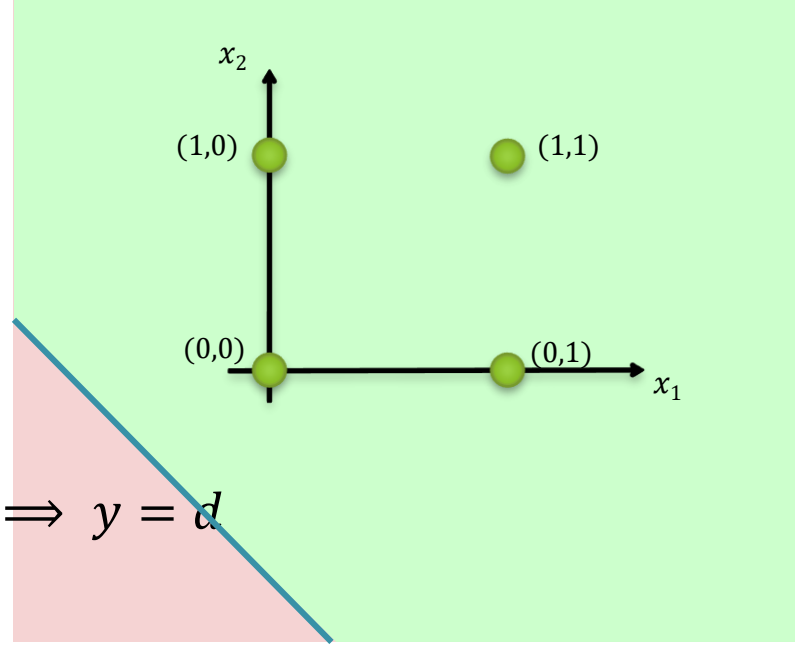
$$\mathbf{d} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Saída Desejada



1° Ciclo

Obs: neste exemplo a notação está simplificada, omitindo o índice da entrada. Ex.: para entrada 1, x_1 é na verdade x_{11} , x_2 é x_{12}, y é y_1, e d é d_1.



- Entrada 1:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$
$$y = f(0 \times 1 + 0 \times 0 + 0 \times 0) = f(0) = 0 \implies y = d$$

- Entrada 2:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$
$$y = f(0 \times 1 + 0 \times 0 + 0 \times 1) = f(0) = 0 \implies y = d$$

- Entrada 3:

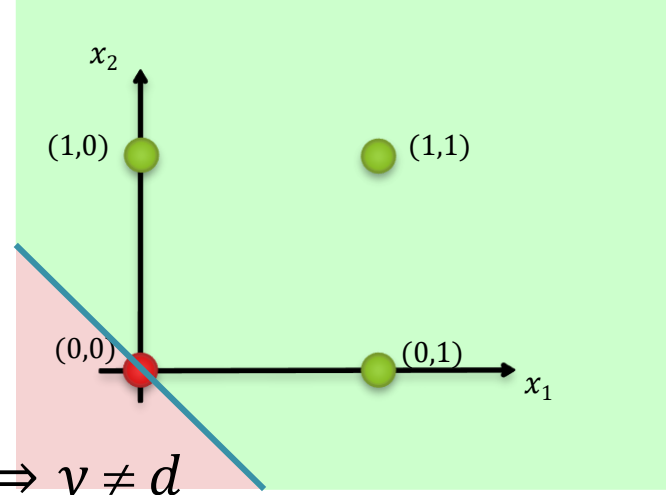
$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$
$$y = f(0 \times 1 + 0 \times 1 + 0 \times 0) = f(0) = 0 \implies y = d$$

- Entrada 4:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$
$$y = f(0 \times 1 + 0 \times 1 + 0 \times 1) = f(0) = 0 \implies y \neq d$$

$$w_0 = w_0 + \alpha(d - y)\, x_0 = 0 + 0{,}5 \times (1 - 0) \times 1 = 0{,}5$$
$$w_1 = w_1 + \alpha(d - y)\, x_1 = 0 + 0{,}5 \times (1 - 0) \times 1 = 0{,}5$$
$$w_2 = w_2 + \alpha(d - y)\, x_2 = 0 + 0{,}5 \times (1 - 0) \times 1 = 0{,}5$$

2° Ciclo

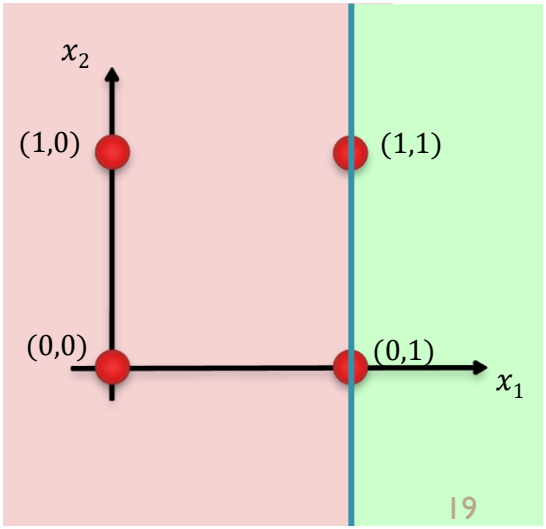


- Entrada 1:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$
$$y = f(0{,}5 \times 1 + 0{,}5 \times 0 + 0{,}5 \times 0) = f(0{,}5) = 1 \Longrightarrow y \neq d$$
$$w_0 = w_0 + \alpha(d - y)\, x_0 = 0{,}5 + 0{,}5 \times (0 - 1) \times 1 = 0$$
$$w_1 = w_1 + \alpha(d - y)\, x_1 = 0{,}5 + 0{,}5 \times (0 - 1) \times 0 = 0{,}5$$
$$w_2 = w_2 + \alpha(d - y)\, x_2 = 0{,}5 + 0{,}5 \times (0 - 1) \times 0 = 0{,}5$$

- Entrada 2:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$
$$y = f(0 \times 1 + 0{,}5 \times 0 + 0{,}5 \times 1) = f(0{,}5) = 1 \Longrightarrow y \neq d$$
$$w_0 = w_0 + \alpha(d - y)\, x_0 = 0 + 0{,}5 \times (0 - 1) \times 1 = -0{,}5$$
$$w_1 = w_1 + \alpha(d - y)\, x_1 = 0{,}5 + 0{,}5 \times (0 - 1) \times 0 = 0{,}5$$
$$w_2 = w_2 + \alpha(d - y)\, x_2 = 0{,}5 + 0{,}5 \times (0 - 1) \times 1 = 0$$



2° Ciclo

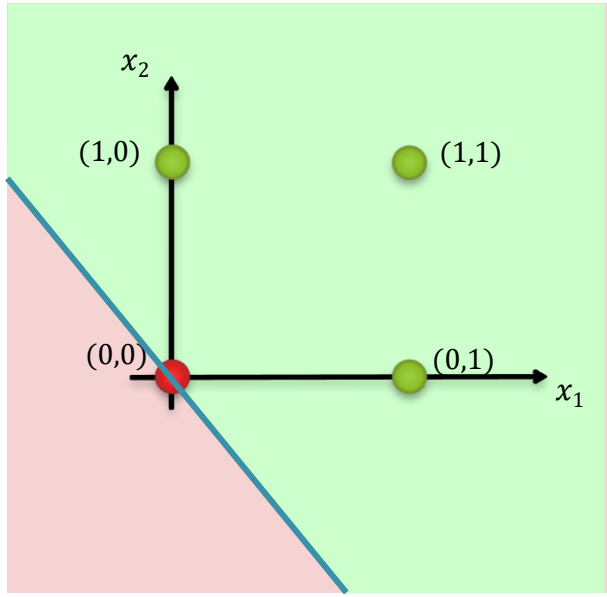
- Entrada 3:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$
$$y = f(-0{,}5 \times 1 + 0{,}5 \times 1 + 0 \times 0) = f(0) = 0 \Longrightarrow y = d$$

- Entrada 4:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$
$$y = f(-0{,}5 \times 1 + 0{,}5 \times 1 + 0 \times 1) = f(0) = 0 \Longrightarrow y \neq d$$

$$w_0 = w_0 + \alpha(d - y)\, x_0 = -0{,}5 + 0{,}5 \times (1 - 0) \times 1 = 0$$
$$w_1 = w_1 + \alpha(d - y)\, x_1 = 0{,}5 + 0{,}5 \times (1 - 0) \times 1 = 1$$
$$w_2 = w_2 + \alpha(d - y)\, x_2 = 0 + 0{,}5 \times (1 - 0) \times 1 = 0{,}5$$



3° Ciclo

- Entrada 1:

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

$$y = f(0 \times 1 + 1 \times 0 + 0{,}5 \times 0) = f(0) = 0 \Longrightarrow y = d$$

- Entrada 2:

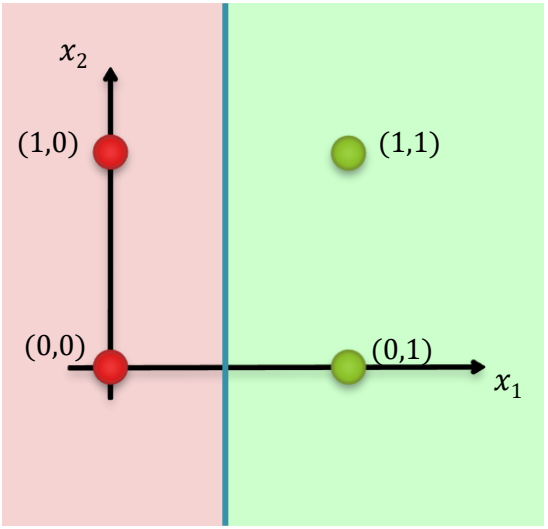
$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

$$y = f(0 \times 1 + 1 \times 0 + 0{,}5 \times 1) = f(0{,}5) = 1 \Longrightarrow y \neq d,$$

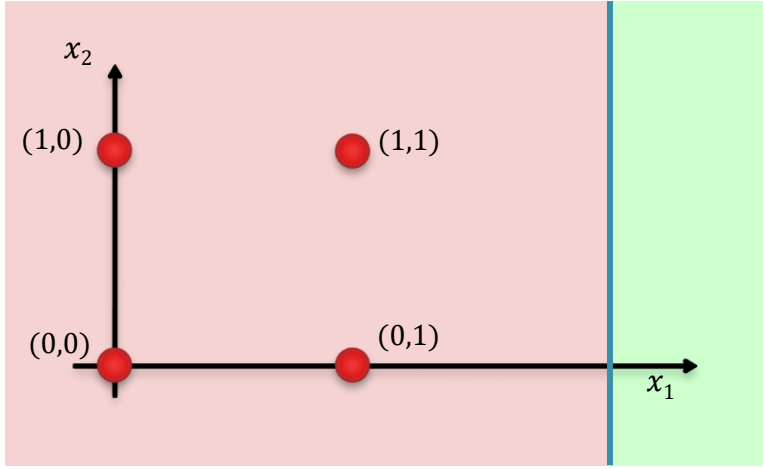
$$w_0 = w_0 + \alpha(d - y)\, x_0 = 0 + 0{,}5 \times (0 - 1) \times 1 = -0{,}5$$

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

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



3° Ciclo



- Entrada 3:

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

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

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

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

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

- Entrada 4:

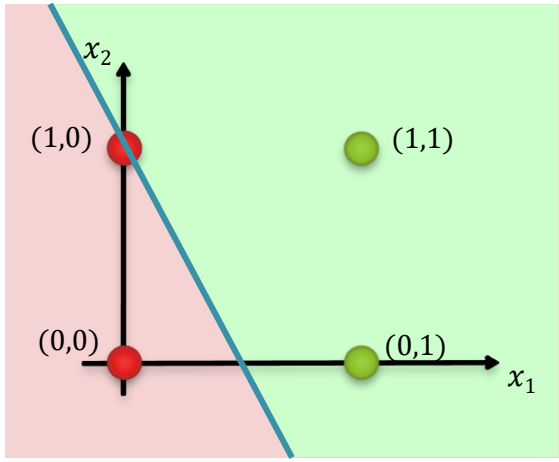
$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

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

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

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

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



4º Ciclo

- Entrada 1:

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

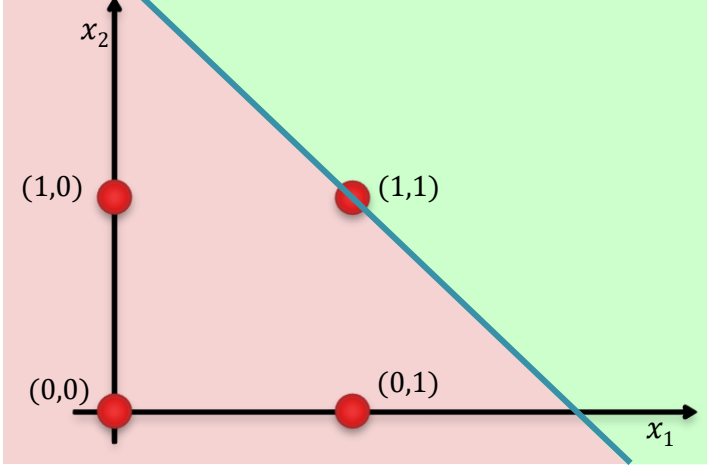
$$y = f(-0{,}5 \times 1 + 1 \times 0 + 0{,}5 \times 0) = f(-0{,}5) = 0 \Longrightarrow y = d$$

- Entrada 2:

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

$$y = f(-0{,}5 \times 1 + 1 \times 0 + 0{,}5 \times 1) = f(0) = 0 \Longrightarrow y = d$$

4° Ciclo



- Entrada 3:

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

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

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

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

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

- Entrada 4:

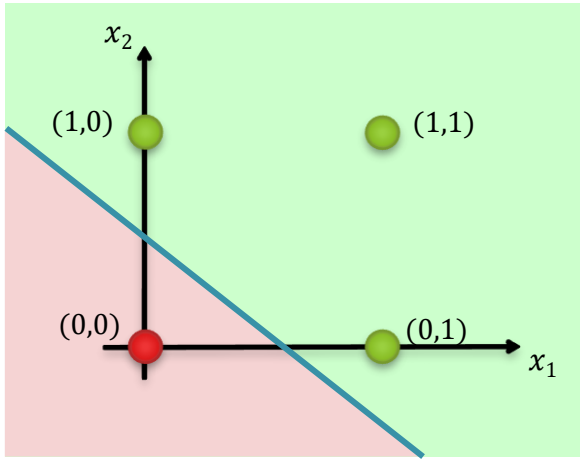
$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

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

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

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

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



5° Ciclo

- Entrada 1:

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

$$y = f(-0{,}5 \times 1 + 1 \times 0 + 1 \times 0) = f(-0{,}5) = 0 \Longrightarrow y = d$$

- Entrada 2:

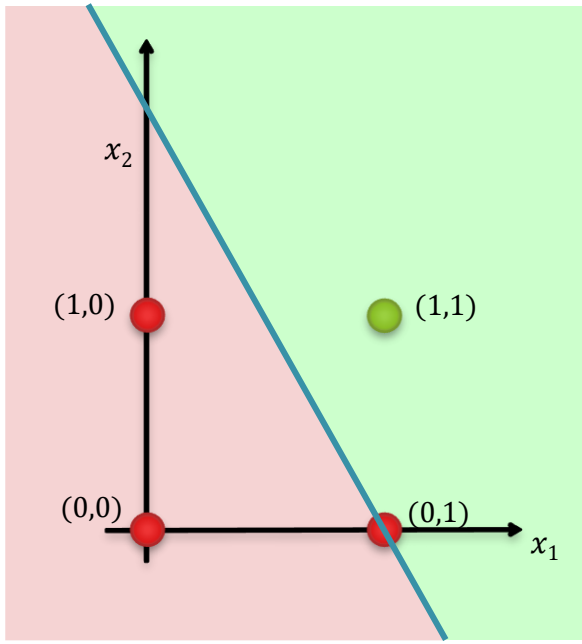
$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$

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

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

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

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



5° Ciclo

- Entrada 3:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$
$$y = f(-1 \times 1 + 1 \times 1 + 0{,}5 \times 0) = f(0) = 0 \Longrightarrow y = d$$

- Entrada 4:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$
$$y = f(-1 \times 1 + 1 \times 1 + 0{,}5 \times 1) = f(0{,}5) = 1 \Longrightarrow y = d$$

6° Ciclo

- Entrada 1:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$
$$y = f(-1 \times 1 + 1 \times 0 + 0{,}5 \times 0) = f(-1) = 0 \Longrightarrow y = d$$

- Entrada 2:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$
$$y = f(-1 \times 1 + 1 \times 0 + 0{,}5 \times 1) = f(-0{,}5) = 0 \Longrightarrow y = d$$

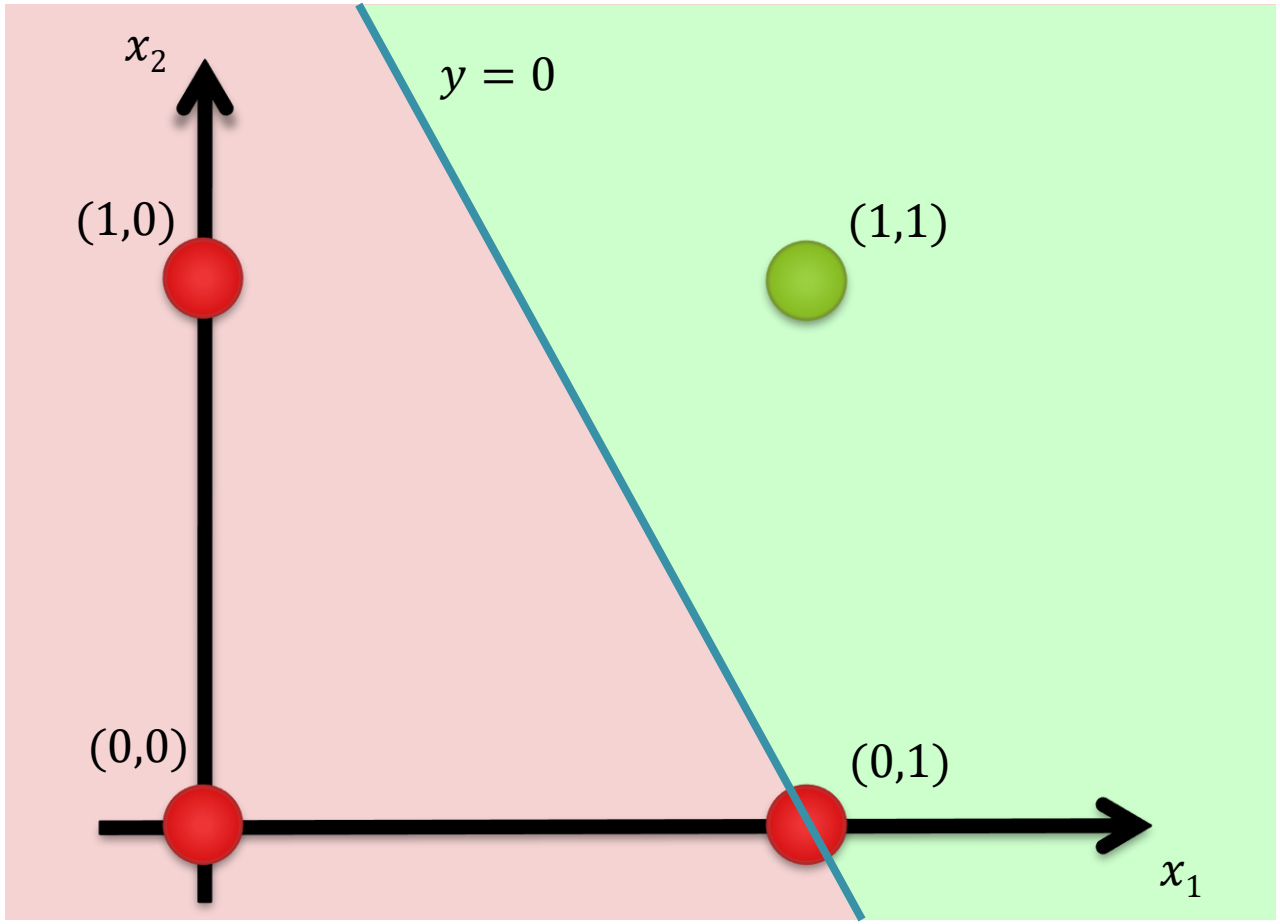
- Entrada 3:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$
$$y = f(-1 \times 1 + 1 \times 1 + 0{,}5 \times 0) = f(0) = 0 \Longrightarrow y = d$$

- Entrada 4:

$$y = f(w_0x_0 + w_1x_1 + w_2x_2)$$
$$y = f(-1 \times 1 + 1 \times 1 + 0{,}5 \times 1) = f(0{,}5) = 1 \Longrightarrow y = d$$

Pesos Finais



$$y = f\left(w_1 x_1 + w_2 x_2 + b\right)$$
$$y = f\left(1x_1 + 0{,}5x_2 - 1\right)$$

Exercício

- Dado o seguinte cadastro de pacientes:

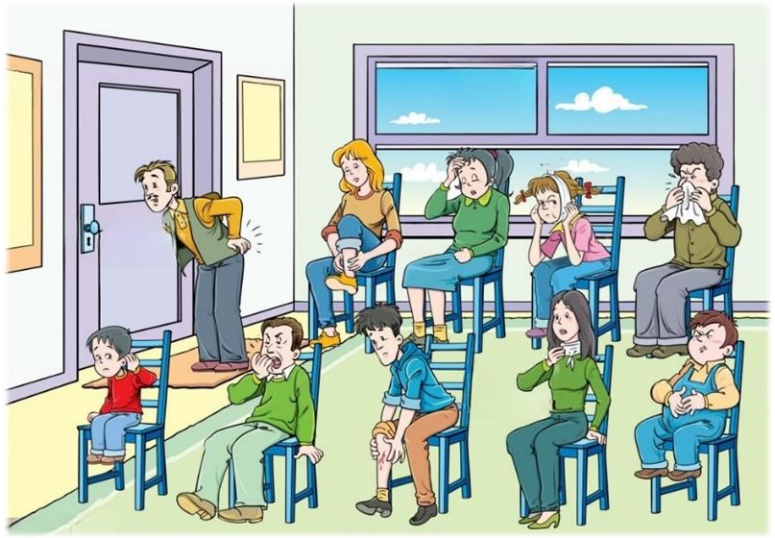
Nome	Febre	Enjoo	Manchas	Dores	Diagnóstico
João	sim	sim	pequenas	sim	**doente**
Pedro	não	não	grandes	não	**saudável**
Maria	sim	sim	pequenas	não	**saudável**
José	sim	não	grandes	sim	**doente**
Ana	sim	não	pequenas	sim	**saudável**
Leila	não	não	grandes	sim	**doente**

- 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

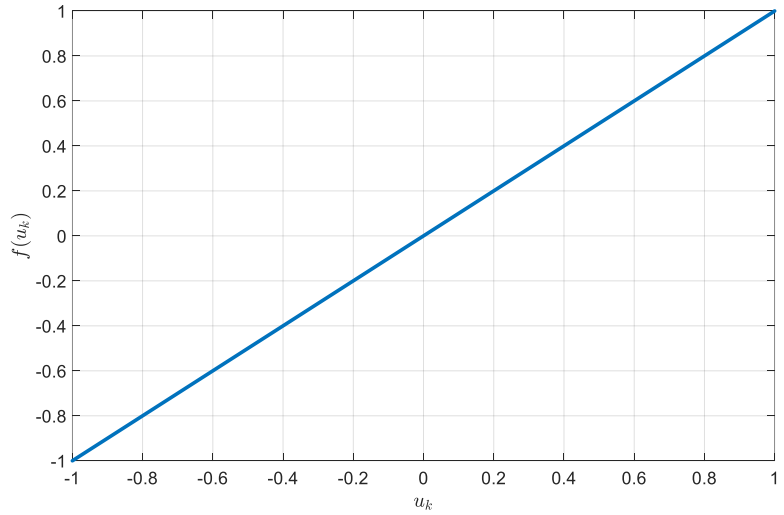
Perceptron Simples para Regressão

- Trocando a função de ativação de limiar por uma função de ativação linear podemos resolver problemas de regressão.

R$ 350.000

R$ 800.000

?



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 $\mathbf{y}_i$ com a saída de cada neurônio para a entrada $\mathbf{x}_i$ é dado por:
  - $\mathbf{y}_i = f\left(\mathbf{W}\mathbf{x}_i^{\mathrm{T}} + \mathbf{b}\right)$
    - onde $\mathbf{W} \in \Re^{o \times m}$, $\mathbf{x}_i \in \Re^{1 \times \mathrm{m}}$, $\mathbf{y}_i \in \Re^{o \times 1}$, e $\mathbf{b} \in \Re^{o \times 1}$
      - o é a quantidade de saídas (quantidade de classes)
      - m é a quantidade de dimensões (atributos) de cada padrão.

Perceptron de Múltiplas Saídas para Classificação de Padrões

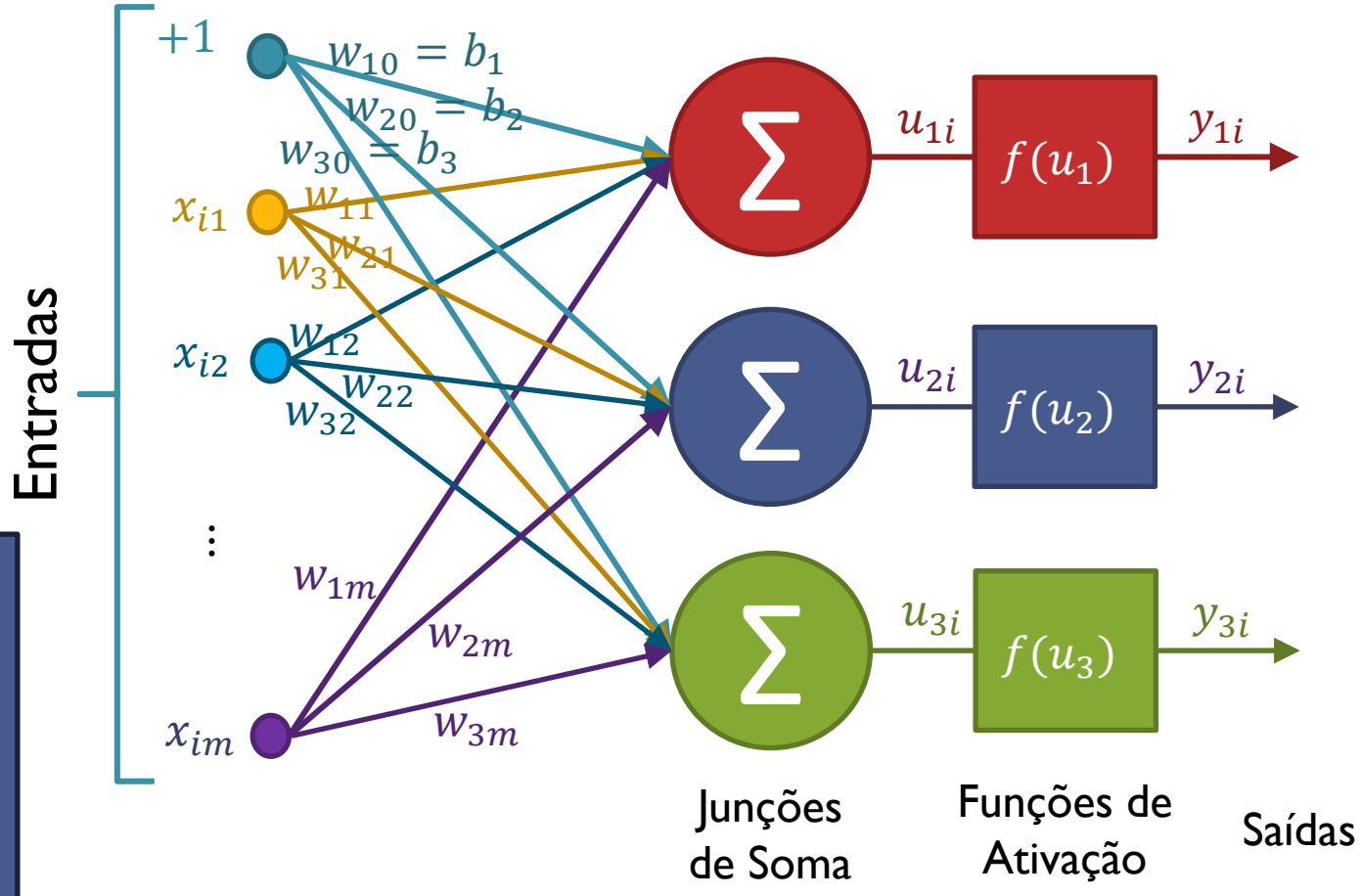
$$\mathbf{y_i} = f\left(\mathbf{W}\mathbf{x_i}^{\mathrm{T}} + \mathbf{b}\right)$$

onde $\mathbf{W} \in \Re^{o \times m}$, $\mathbf{x_i} \in \Re^{1 \times m}$, $\mathbf{y_i} \in \Re^{o \times 1}$, e $\mathbf{b} \in \Re^{o \times 1}$

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1m} \\ w_{21} & w_{22} & \dots & \vdots \\ w_{31} & w_{32} & \dots & w_{om} \end{bmatrix}$$

$$\boldsymbol{x_i} = [x_{i1} \quad x_{i2} \quad \dots \quad x_{im}]$$

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \qquad \mathbf{y_i} = \begin{bmatrix} y_{1i} \\ y_{2i} \\ y_{3i} \end{bmatrix}$$

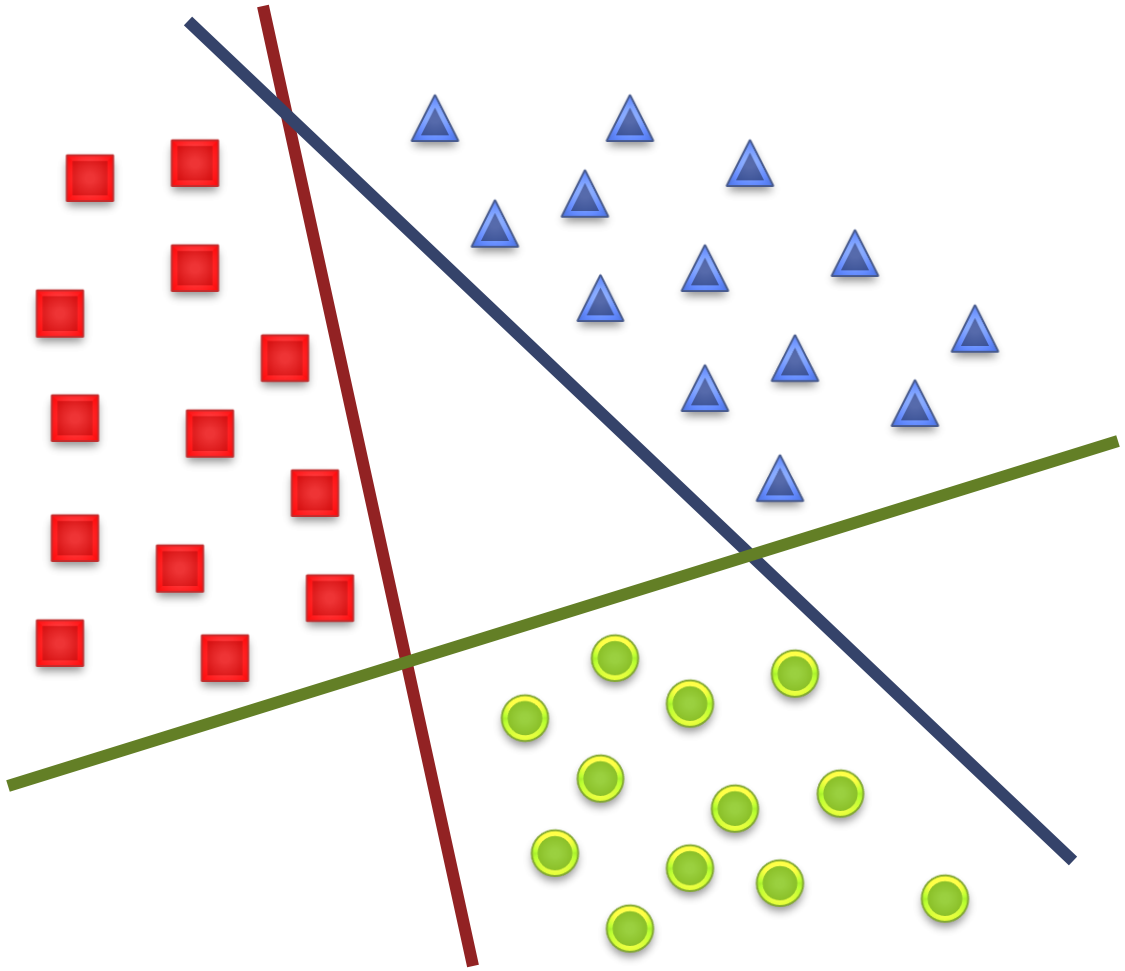


Dica: lembre-se que você pode inserir o vetor de bias como sendo a primeira coluna da matriz de pesos (coluna 0), e considerar $x_{i0} = 1$. Nesse caso:

$$\mathbf{y_i} = f\left(\mathbf{W}\mathbf{x_i}^{\mathrm{T}}\right)$$

$$y_{ji} = f(\mathbf{w_j}\mathbf{x_i}^{\mathrm{T}}) = f\left(\sum_{k=0}^{m} w_{jk}x_{ik}\right)$$

Perceptron de Múltiplas Saídas para Classificação de Padrões



Algoritmo de Treinamento do Perceptron com múltiplas saídas

- $\mathbf{D}$ é a matriz de saídas desejadas.
  - Cada coluna é a saída para um dos padrões de entrada.
  - Cada linha é a saída de um neurônio.
- Vetor de erro é calculado subtraindo o vetor de saídas desejadas do vetor de saídas obtidas.
  - $\mathbf{e}_i = \mathbf{d}_i - \mathbf{y}_i$
- Se os padrões forem linearmente separáveis o algoritmo converge.

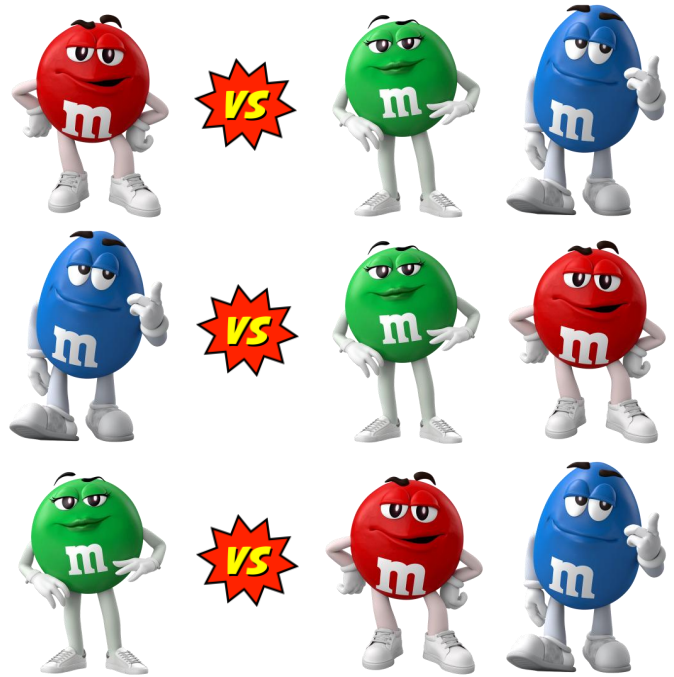
Algoritmo de Treinamento do Perceptron com múltiplas saídas

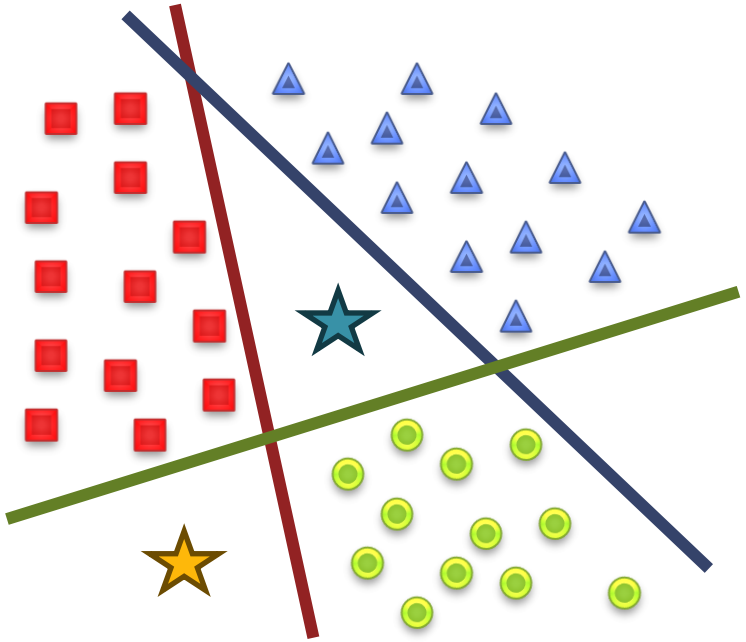
```
procedimento [W] = perceptron(max_it,α,X,D)
  inicializar W   // ajuste para zero ou pequenos valores aleatórios
  inicializar b   // ajuste para zero ou pequenos valores aleatórios
  t ← 1;
  E ← 1;
  enquanto t ≤ max_it & E > 0 faça
    E ← 0;
    para i de 1 até N faça          // para cada padrão de treinamento
      y_i ← f(W x_i^T + b)          // vetor de saída da rede para x_i
      e_i ← d_i - y_i               // determinar o vetor de erros para x_i
      W ← W + α e_i x_i             // atualizar a matriz de pesos
      b ← b + α e_i                 // atualizar o vetor de bias
      E ← E + soma(e_ji^2)          // j = 1,...,o
    fim-para
    t ← t + 1
  fim-enquanto
fim-procedimento
```

Algoritmo de aprendizado de um perceptron de múltiplas saídas. A função $f(.)$ 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

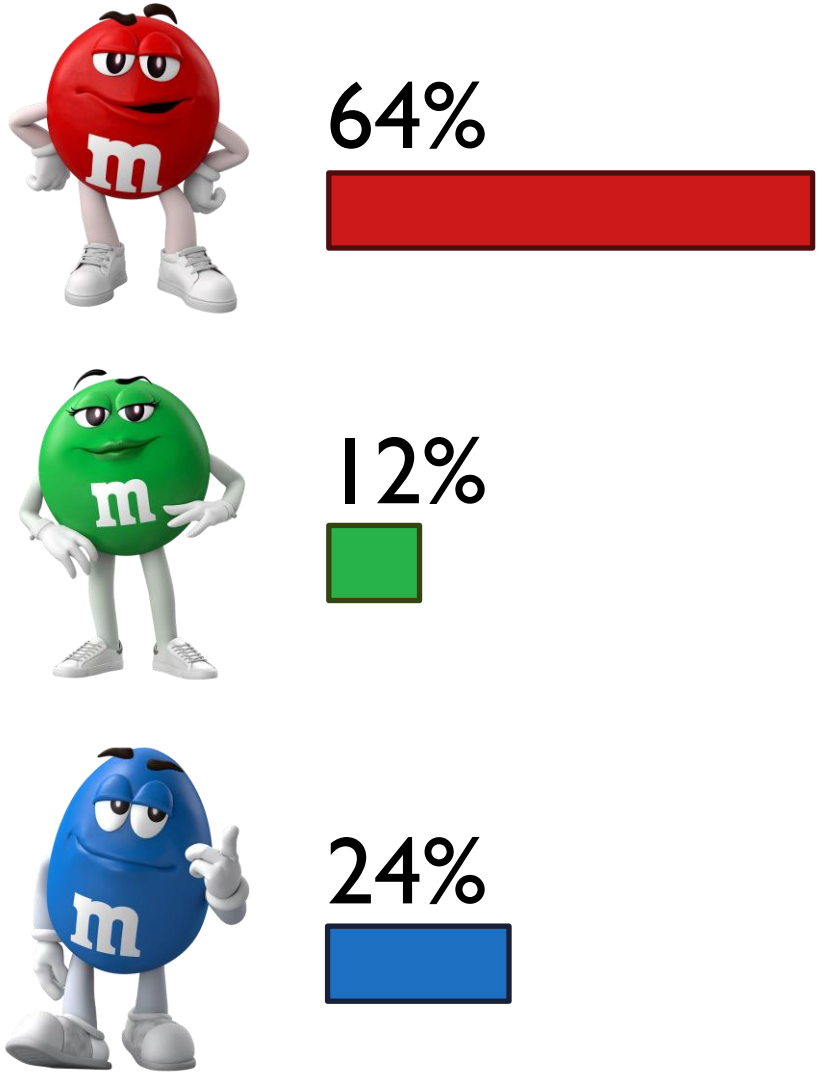
- Em problemas com múltiplas saídas temos neurônios que classificam *um-versus-resto*.
  - Exemplo:
    - vermelho X verde e azul
    - azul X verde e vermelho
    - verde X vermelho e azul
  - Nesse caso, usar função de limiar pode levar à ambiguidades, com mais de um neurônio disparando ao mesmo tempo.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





Um-versus-resto

- Usar valores contínuos de saída pode ser uma solução.
  - Podem ser interpretados como um índice de confiança na resposta.
  - Mas não dá para usar diretamente o somatório do neurônio antes da função de ativação (u_{ij}).
    - Pode apresentar valores negativos.



Função Softmax

- Uma estratégia comum é usar uma função de ativação do tipo *softmax* na camada de saída.
  - $y_{ji} = f(u_{ji}) = \frac{\exp u_{ji}}{\sum_{c=1}^{o} \exp u_{ci}}$ onde $j = 1, \ldots, o$
    - u_{ji} é o resultado do somatório do neurônio j, antes da função de ativação, o é a quantidade de neurônios de saída, ou seja, a quantidade de classes do problema.
  - Dessa forma a saída dos neurônios fica contínua, no intervalo $[0\ 1]$, com soma 1.
    - Cada saída y_{ji} corresponde à probabilidade de que o padrão x_i pertença à classe correspondente ao neurônio j.
    - No treinamento, utilize codificação *one-hot* para rotular os padrões.
      - Ex.: $d_i = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, onde 1 indica a classe à qual o padrão pertence e 0 indica as demais classes.
    - Cada exemplo é classificado pelo neurônio que apresentou maior valor de saída:
      - $\arg\max_j y_{ij}$

Função Softmax – Exemplo

$$y_{ji} = f(u_{ji}) = \frac{\exp u_{ji}}{\sum_{c=1}^{o} \exp u_{ci}} \text{ onde } j = 1, \ldots, o$$

- $u_{1i} = 2{,}33$

- $u_{2i} = -1{,}46$

- $u_{3i} = 0{,}56$

- $\exp u_{1i} = 10{,}29$

- $\exp u_{2i} = 0{,}23$

- $\exp u_{3i} = 1{,}75$

- $\sum_{c=1}^{o} \exp u_{ci} = 12{,}27$

- $y_{1i} = 0{,}84$

- $y_{2i} = 0{,}02$

- $y_{3i} = 0{,}14$

$$\mathbf{y}_i = \begin{bmatrix} 0{,}84 \\ 0{,}02 \\ 0{,}14 \end{bmatrix}$$

Função Softmax – Propagando o Erro

- Convenientemente, o vetor de erro continua sendo calculado subtraindo o vetor de saída obtido do vetor de saída desejado:
  - $\mathbf{e}_i = \mathbf{d}_i - \mathbf{y}_i$

Exemplo:

$$\mathbf{e}_i = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0{,}84 \\ 0{,}02 \\ 0{,}14 \end{bmatrix} = \begin{bmatrix} 0{,}16 \\ -0{,}02 \\ -0{,}14 \end{bmatrix}$$

  - Diferença entre as probabilidades atribuída pelo nosso modelo (softmax) e o rótulo real (codificação *one-hot*).
    - Esta é a derivada da saída com respeito a qualquer u_{ij}
      - Por conta dos gradientes da probabilidade logarítmica.
  - Desse modo continuamos ajustando os pesos como vimos anteriormente.

Função de Perda: Entropia Cruzada Categórica (*Categorical Cross-Entropy*)

- Lembre-se: a saída desejada em uma *softmax* é uma codificação *one-hot*.
  - Exemplo: $d_i = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$
- Tipicamente usamos a entropia cruzada categórica para calcular a perda em cada exemplo:
  - $Loss = -\sum_j^o d_{ji} \log(y_{ji})$
  - Exemplo:
    - Para a saída $y_i = \begin{bmatrix} 0{,}84 \\ 0{,}02 \\ 0{,}14 \end{bmatrix}$
    - $Loss = -(1 \times \log(0{,}84) + 0 \times \log(0{,}02) + 0 \times \log(0{,}14))$
    - $Loss = -(\log(0{,}84))$
    - $Loss = -(-0{,}173)$
    - $Loss = 0{,}173$

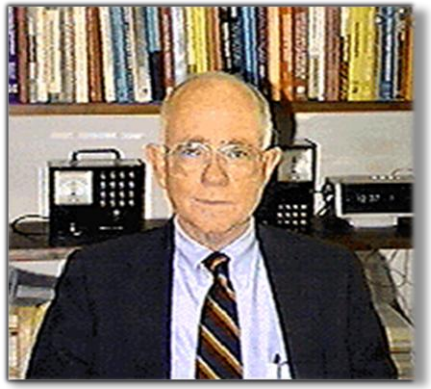
Queremos minimizar a perda!

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]

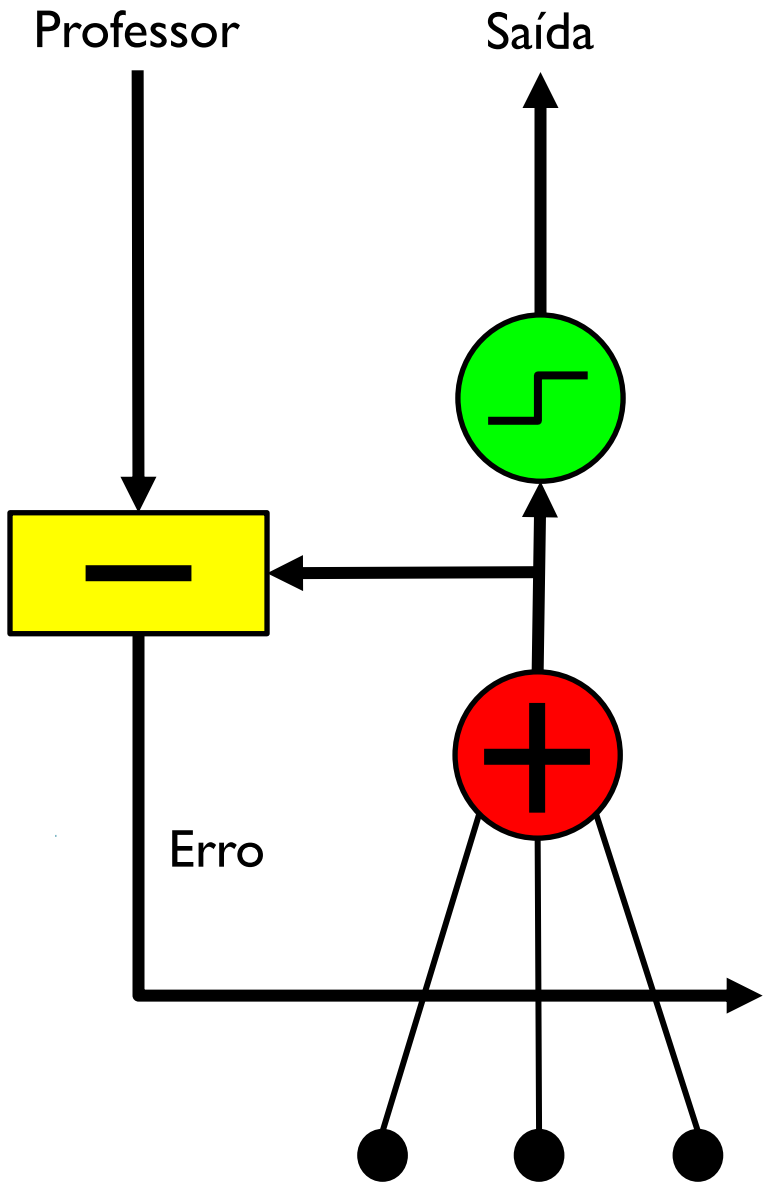
Bernard Widrow é um engenheiro americano e professor de engenharia elétrica na Universidade de Stanford.

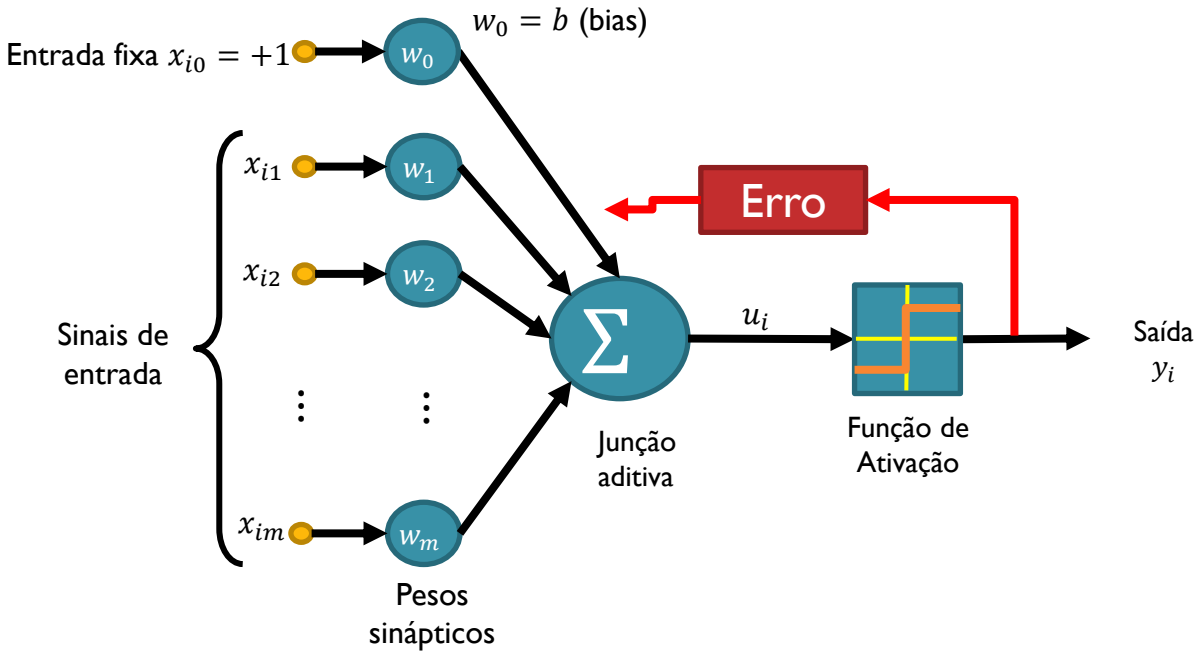
Marcian Hoff [*1937]

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.

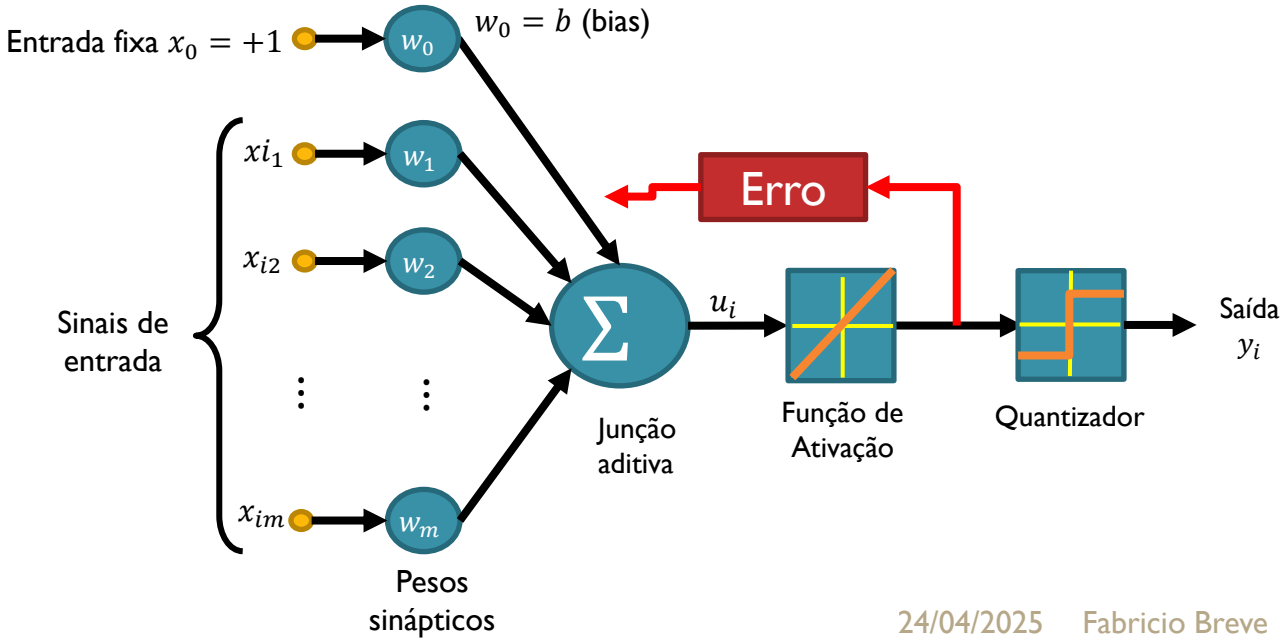
ADALINE

- 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



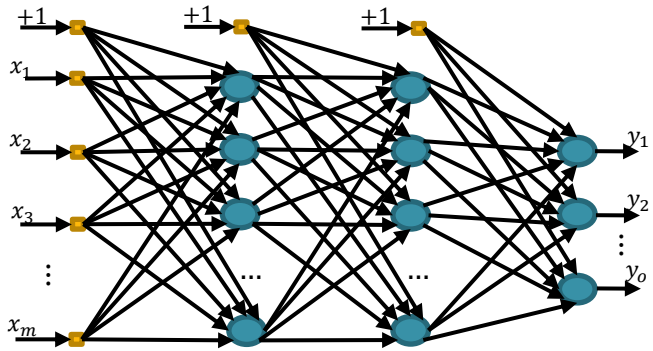
ADALINE

Redes Neurais Artificiais

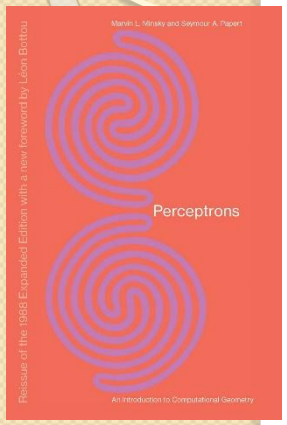
PERCEPTRON DE MÚLTIPLAS CAMADAS

Perceptron de Múltiplas Camadas

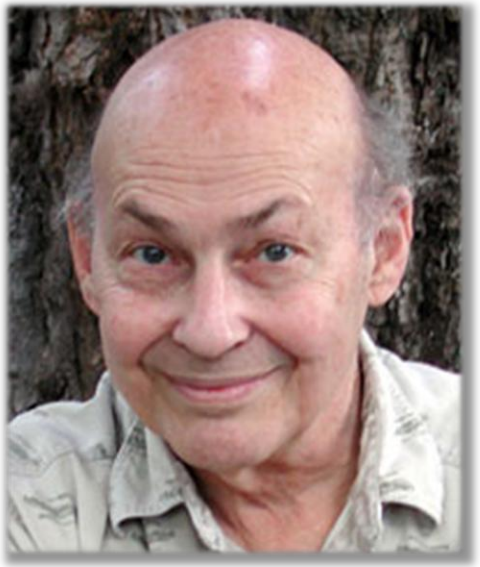
- Possui uma ou mais camadas intermediárias de nós.
- Grande Funcionalidade.
  - Uma camada intermediária: qualquer função contínua ou Booleana.
  - Duas camadas intermediárias: qualquer função.
- Treinada com o algoritmo *Backpropagation* (Retropropagação).



Histórico

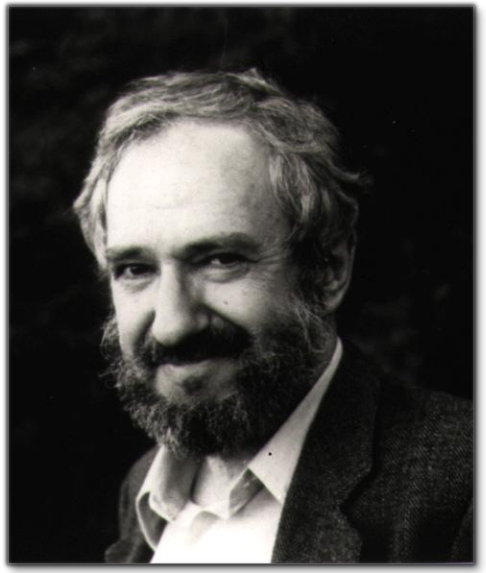


- Em 1969 Minsky e Papert lançaram um livro mostrando as limitações do Perceptron.
  - Basicamente sua incapacidade de lidar com problemas não linearmente separáveis.
- Rosenblatt e Widrow sabiam dessas limitações e propuseram o uso de múltiplas camadas.
  - Mas não conseguiram generalizar seus algoritmos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



Marvin Minsky [*1927 – †2016]

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

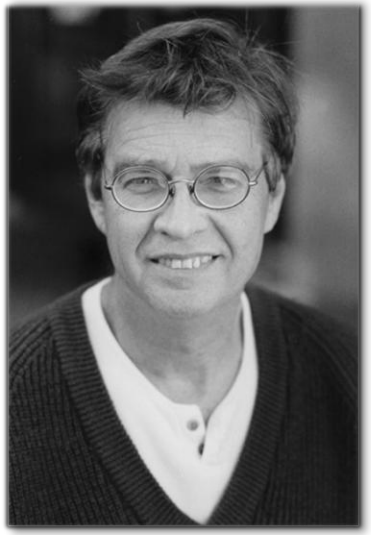


Seymour Papert [*1928 – †2016]

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

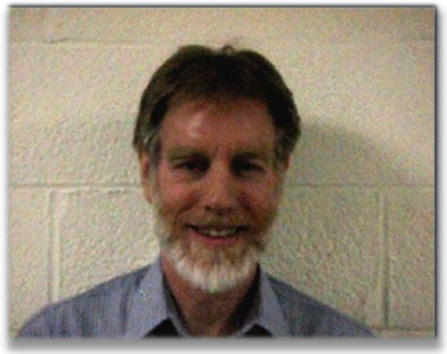
- Já existia nos anos 1970, mas ganhou destaque apenas nos 1980.
  - Popularizado por Rumelhart, Hinton, e Williams (1986).
  - É uma generalização do LMS.
  - Fez ressurgir o interesse em redes neurais.



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]



Ronald J. Williams
[*1945 – †2024]

Geoffrey Everest Hinton é um psicólogo cognitivo e cientista da computação anglo-canadense, conhecido por seu trabalho sobre redes neurais artificiais. Desde 2013 dividiu seu tempo trabalhando para o Google e a Universidade de Toronto. Deixou o Google em 2024.

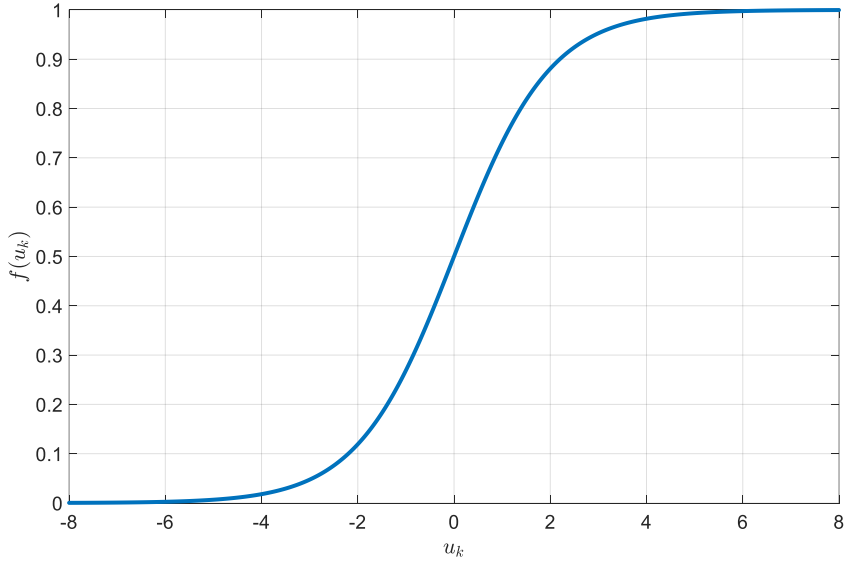
Ronald J. Williams foi professor de ciência da computação na Northeastern University e um dos pioneiros das redes neurais.

Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986). https://doi.org/10.1038/323533a0

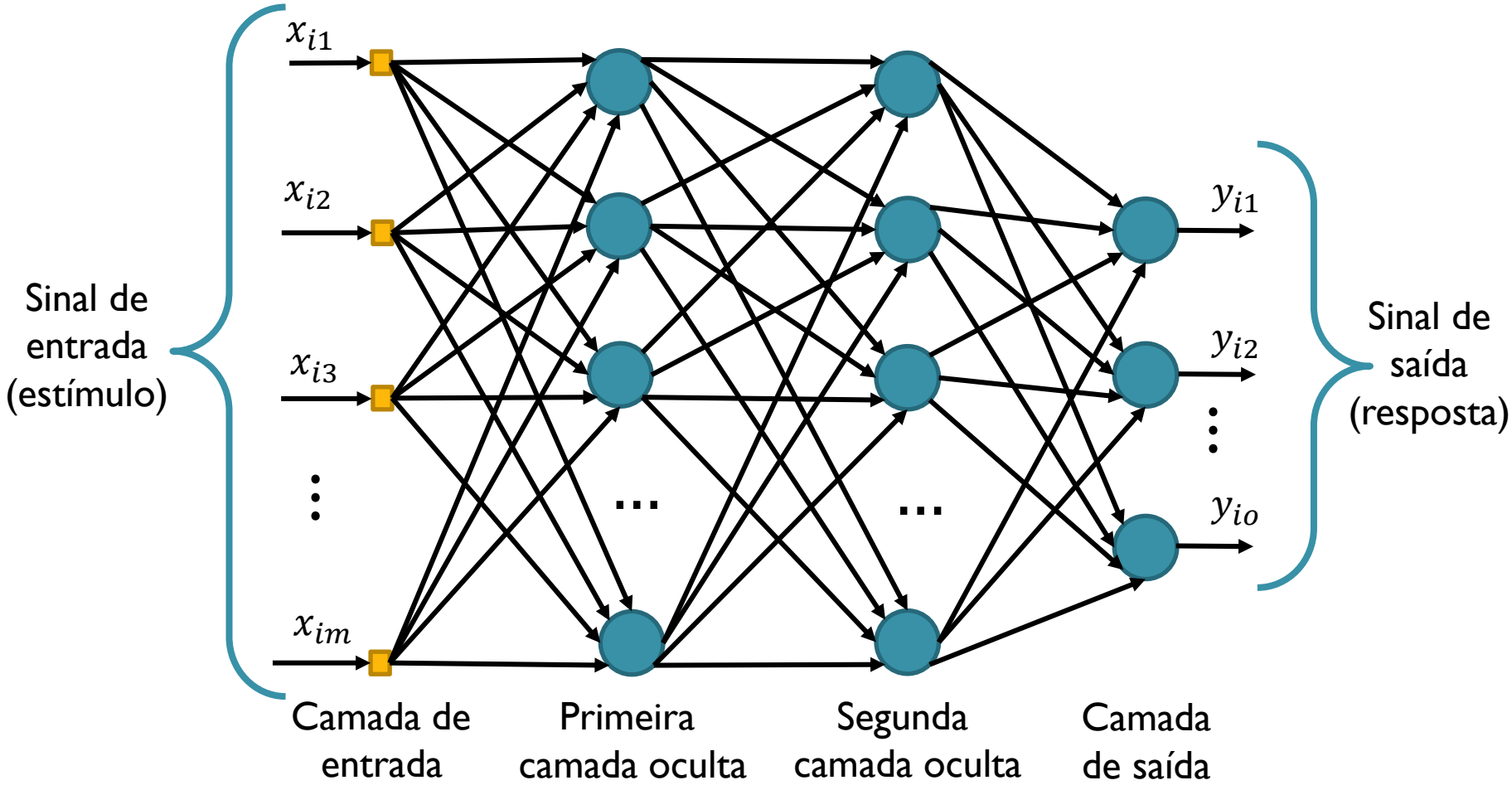
Perceptron de Múltiplas Camadas

- 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

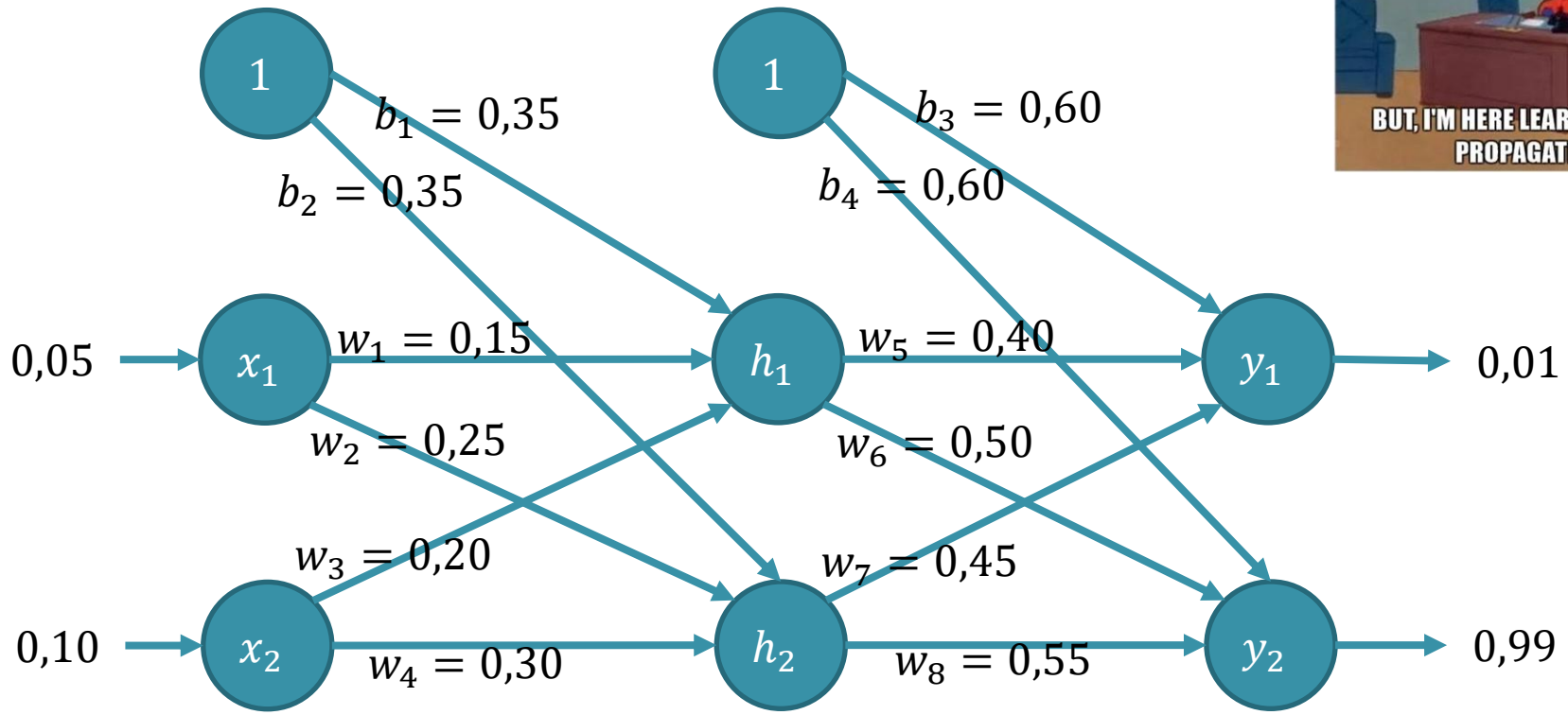


Algoritmo de Retropropagação de Erro

- 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

- Considere a seguinte rede neural:

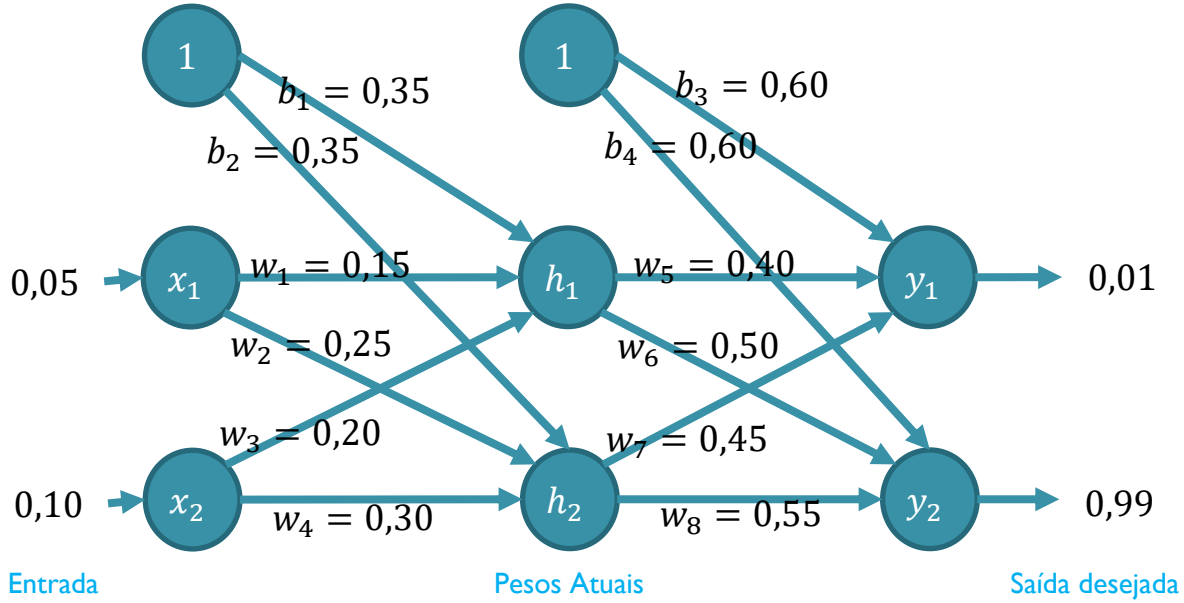


Adaptado de:
https://www.edureka.co/blog/backpropagation/

Passo para a frente: camada intermediária

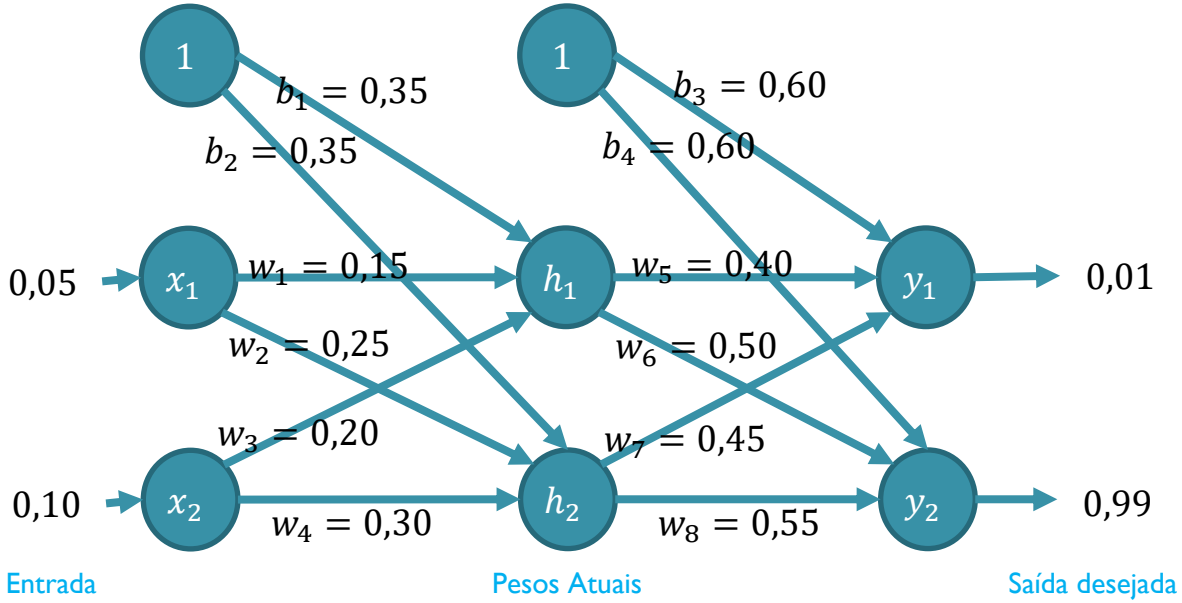
- Primeiro a camada intermediária, começando com h_1 :
  - $h_1 = f(w_1 x_1 + w_3 x_2 + b_1)$
  - $h_1 = f(0{,}15 \times 0{,}05 + 0{,}2 \times 0{,}1 + 0{,}35)$
  - $h_1 = f(0{,}3775)$
  - $h_1 = \frac{1}{1+\exp(-0{,}3775)}$
  - $h_1 = 0{,}593269992$
- De maneira similar:
  - $h_2 = 0{,}596884378$

Lembre-se que a função de ativação é a sigmoide



Passo para a frente: camada de saída

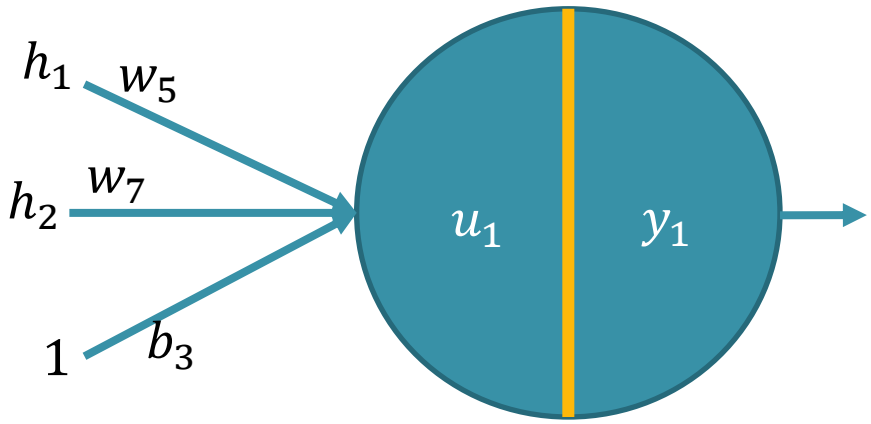
- Agora a camada de saída, começando por y_1 :
  - $y_1 = f(w_5 h_1 + w_7 h_2 + b_3)$
  - $y_1 = f(0{,}4 \times 0{,}593269992 + 0{,}45 \times 0{,}596884378 + 0{,}6)$
  - $y_1 = f(1{,}105905967)$
  - $y_1 = \frac{1}{1+\exp(-1{,}105905967)}$
  - $y_1 = 0{,}75136507$
- De maneira similar:
  - $y_2 = 0{,}772928465$



Erro na Camada de Saída

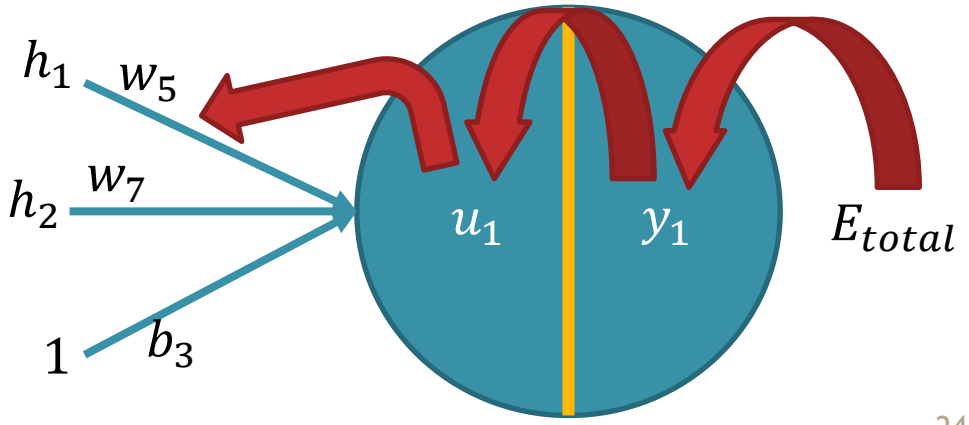
- Calculando o erro quadrático na camada de saída:
  - $E_1 = \frac{1}{2}(d_1 - y_1)^2$
  - $E_1 = \frac{1}{2}(0{,}01 - 0{,}75136507)^2$
  - $E_1 = 0{,}274811083$
  - $E_2 = 0{,}023560026$
  - $E_{total} = E_1 + E_2$
  - $E_{total} = 0{,}298371109$

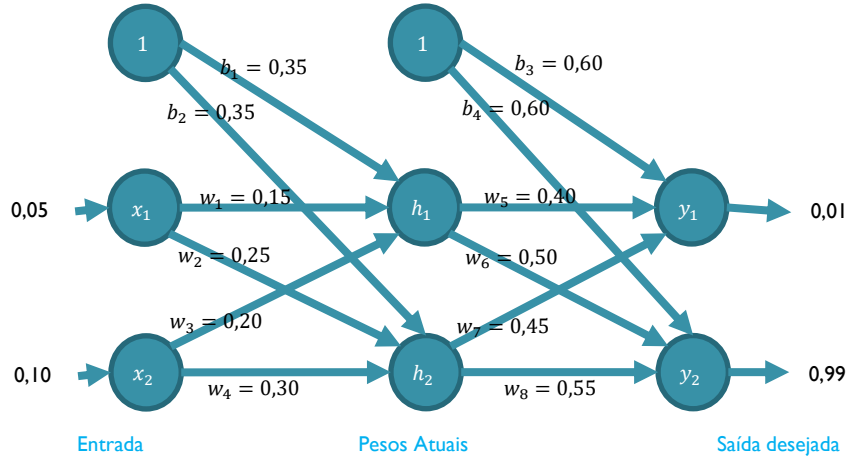
- Atenção para a notação que usaremos nos próximos passos:
  - u_1: somatório das entradas de y_1, antes da função de ativação.
  - y_1: saída de y_1 após a aplicação da função de ativação.



Passo para trás: a retroprogação

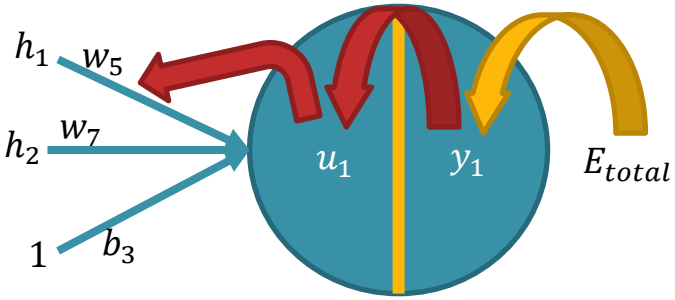
- Agora vamos retropropagar o erro.
  - Tentar reduzir o erro ajustando valores de pesos e bias.
- Considere w_5 , calcularemos a taxa de variação do erro em relação à alteração no peso w_5 :
  - $$\frac{\delta E_{total}}{\delta w_5} = \frac{\delta E_{total}}{\delta y_1} \times \frac{\delta y_1}{\delta u_1} \times \frac{\delta u_1}{\delta w_5}$$





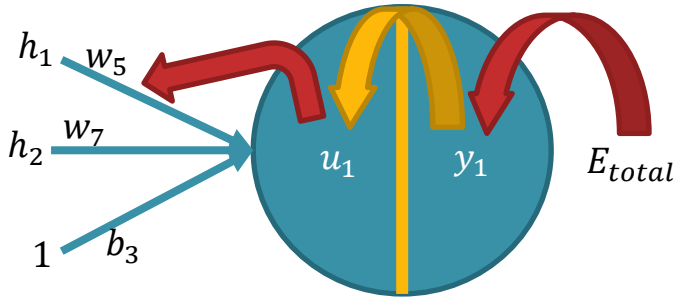
Passo para trás: derivada do erro total em relação à saída y_1

- 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 :
  - $E_{total} = \frac{1}{2}(d_1 - y_1)^2 + \frac{1}{2}(d_2 - y_2)^2$
  - $\frac{\delta E_{total}}{\delta y_1} = -(d_1 - y_1)$
  - $\frac{\delta E_{total}}{\delta y_1} = -(0{,}01 - 0{,}75136507)$
  - $\frac{\delta E_{total}}{\delta y_1} = 0{,}74136507$



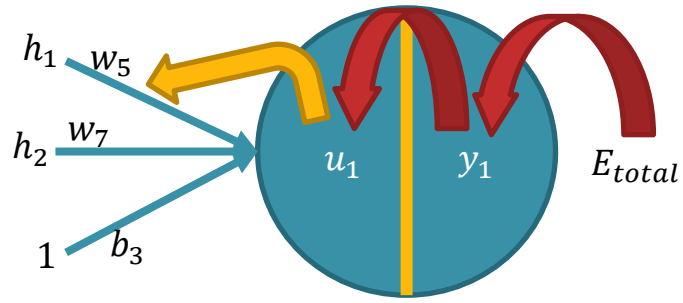
Passo para trás: derivada da saída y_1 em relação ao somatório u_1

- Agora, vamos propagar mais para trás e calcular a mudança na saída y_1 em relação ao total de suas entradas:
  - $y_1 = \frac{1}{1+\exp(-u_1)}$
  - $\frac{\delta y_1}{\delta u_1} = y_1(1 - y_1)$
  - $\frac{\delta y_1}{\delta u_1} = 0{,}75136507(1 - 0{,}75136507)$
  - $\frac{\delta y_1}{\delta u_1} = 0{,}186815602$



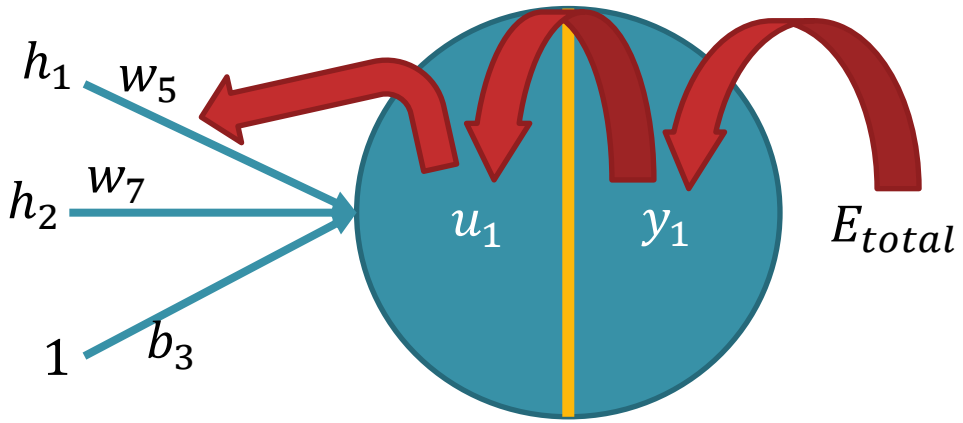
Passo para trás: derivada do somatário u_1 em relação ao peso w_5

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



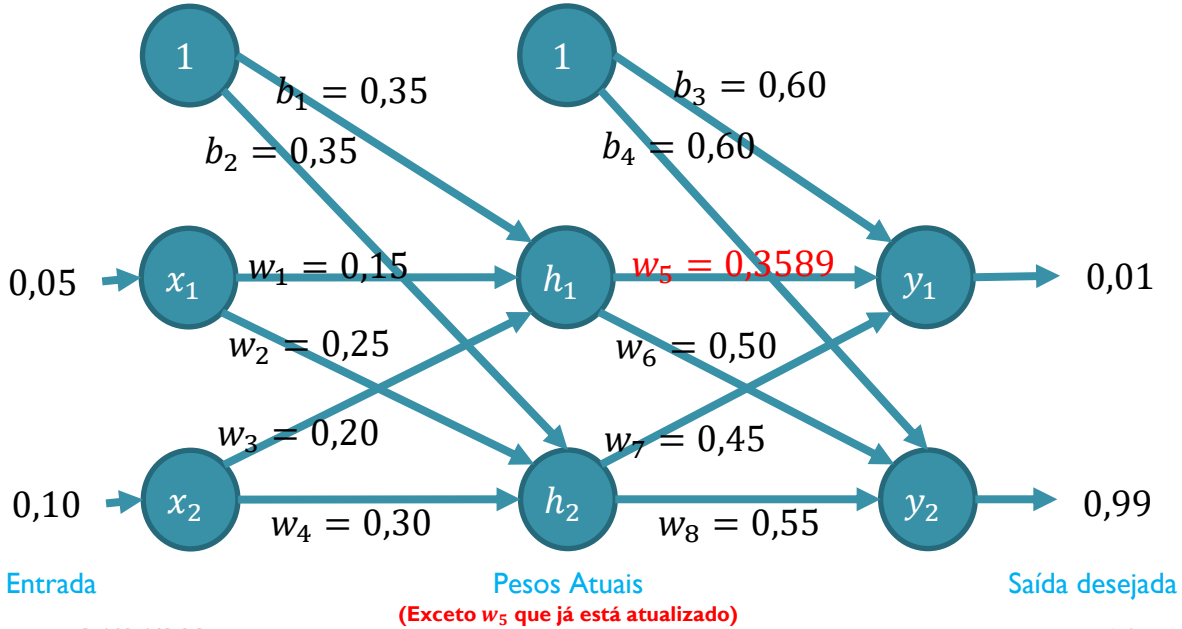
Calculando a atualização do peso w_5

- Agora vamos multiplicar todas as derivadas parciais para obter a derivada do erro total em relação ao peso w_5:
  - $\frac{\delta E_{total}}{\delta w_5} = \frac{\delta E_{total}}{\delta y_1} \times \frac{\delta y_1}{\delta u_1} \times \frac{\delta u_1}{\delta w_5}$
  - $\frac{\delta E_{total}}{\delta w_5} = 0{,}74136507 \times 0{,}186815602 \times 0{,}593269992$
  - $\frac{\delta E_{total}}{\delta w_5} = 0{,}082167041$



Atualizando o peso de w_5

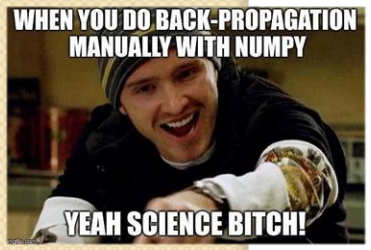
- Finalmente, vamos calcular o valor atualizado de w_5:
  - $w_5^+ = w_5 - \alpha \frac{\delta E_{total}}{\delta w_5}$
  - $w_5^+ = 0{,}4 - 0{,}5 \times 0{,}082167041$
  - $w_5^+ = 0{,}35891648$



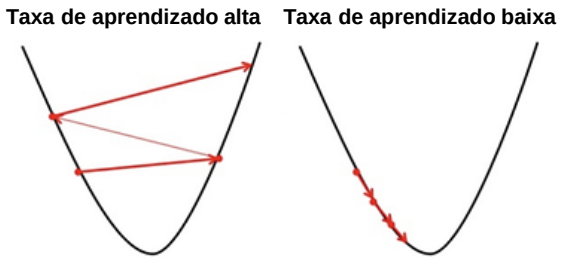
Nota: Neste exemplo, definimos a taxa de aprendizado $\alpha = 0{,}5$

Algoritmo de Retropropagação: Exemplo

- De forma similar, calculamos todos os demais pesos atualizados.
- Após atualizar todos os pesos, iremos novamente propagar para frente e calcular a saída (o próximo exemplo do conjunto de dados).
- E novamente, vamos calcular o erro.
- No final da época:
  - Se o erro atingiu o valor alvo ou algum outro critério de parada foi atingido, vamos parar.
  - Caso contrário, vamos novamente propagar para frente e para trás, atualizando os peso.
- 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.

Imagem: https://www.kdnuggets.com/2020/12/optimization-algorithms-neural-networks.html

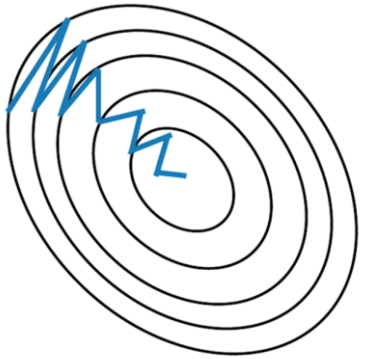
Variações do Algoritmo de Retropropagação

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

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 "lembrar" da direção em que os parâmetros da rede neural estavam sendo atualizados nas iterações anteriores e continuar atualizando nessa direção com uma certa inércia.

sem *momentum*



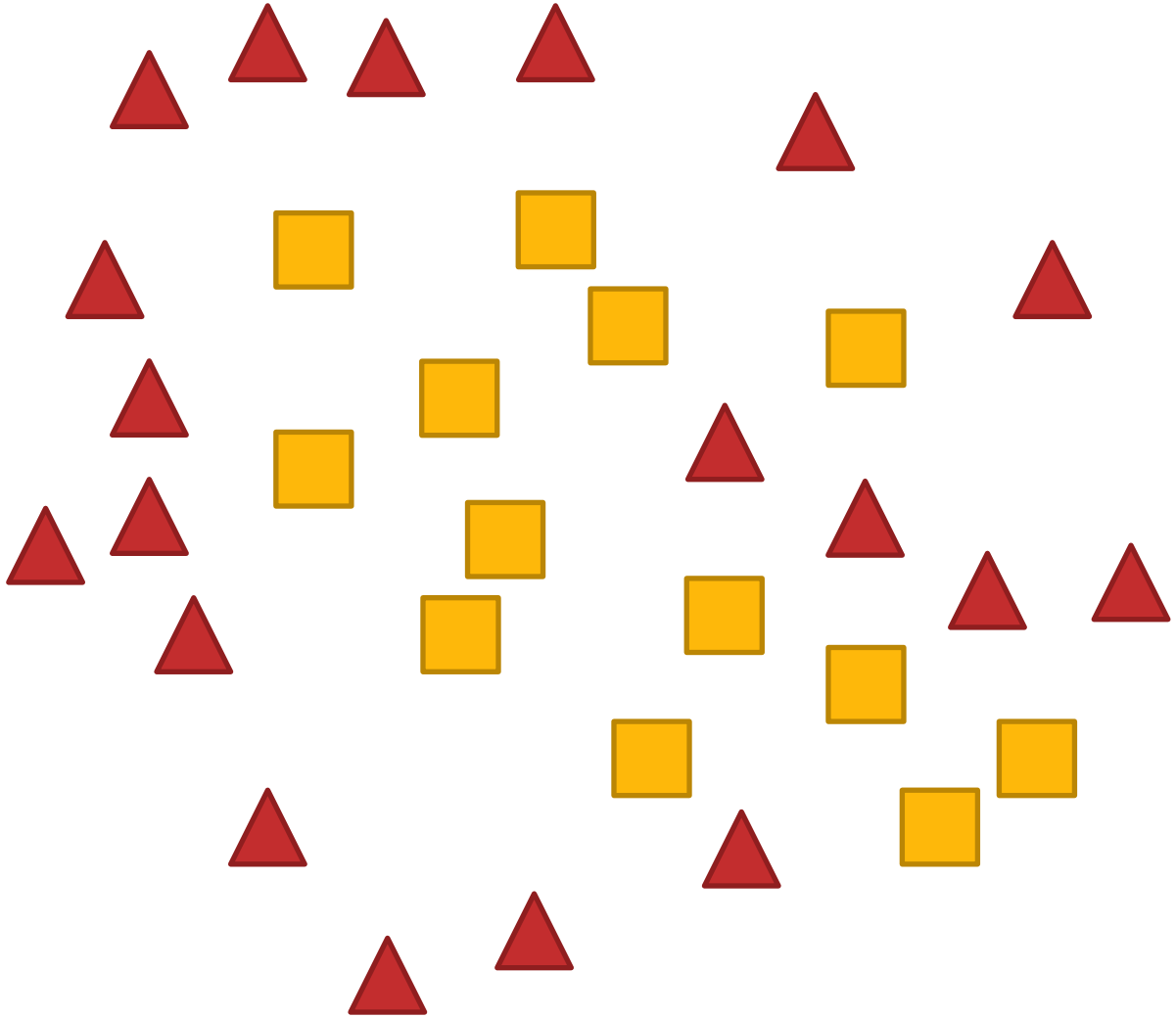
com *momentum*

Imagem: https://eloquentarduino.github.io/2020/04/stochastic-gradient-descent-on-your-microcontroller/

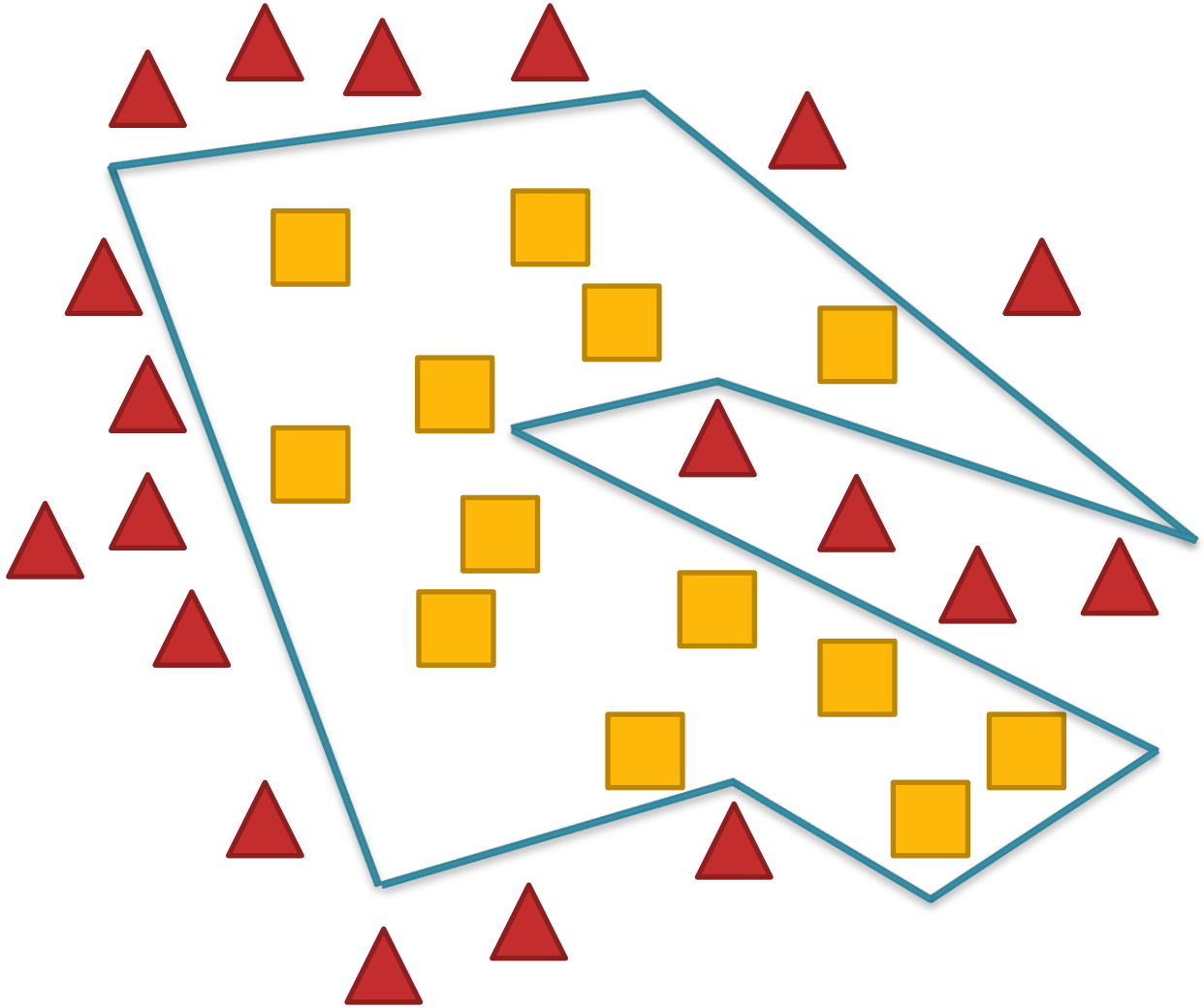
Momentum

- 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:
  - $\Delta w_{jk} = \left(\alpha \frac{\delta E}{\delta w_{jk}}\right) + \left(\gamma \Delta\, w_{jk}^{t-1}\right)$
  - 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.

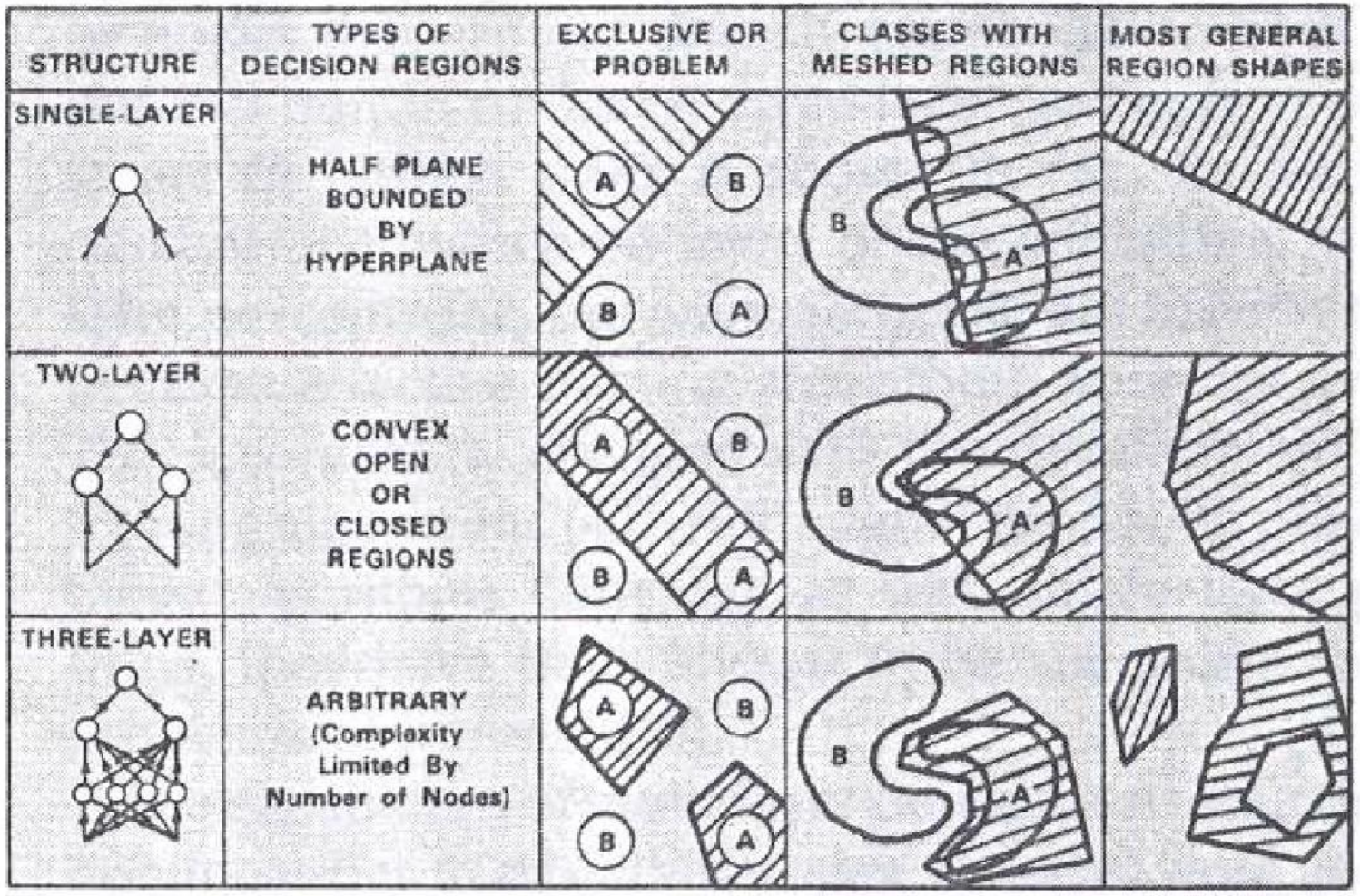
Por que múltiplas camadas?



Por que múltiplas camadas?



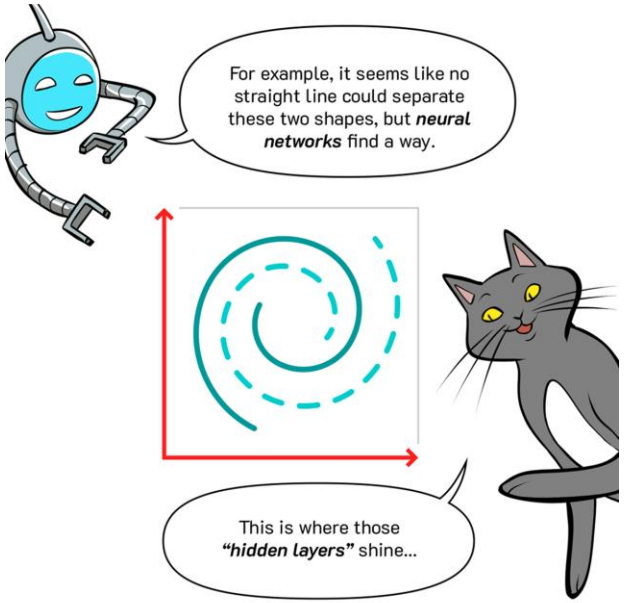
Por que múltiplas camadas?

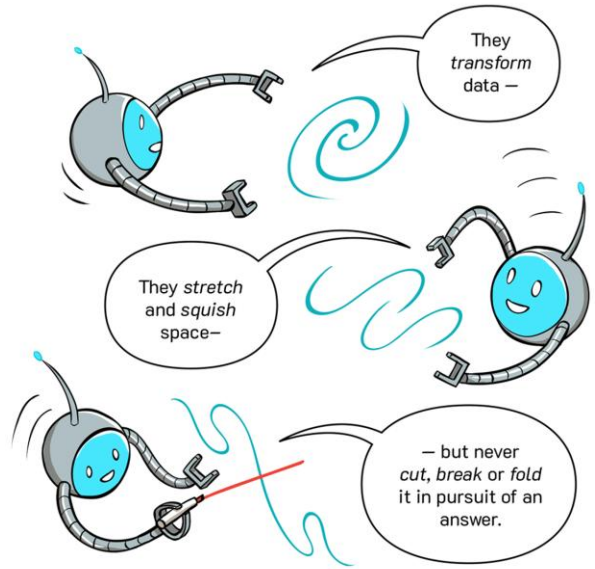


Lippmann, R.P., "An introduction to computing with neural nets," *ASSP Magazine, IEEE* , vol.4, no.2, pp.4,22, Apr 1987
doi: 10.1109/MASSP.1987.1165576
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1165576&isnumber=26240

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.





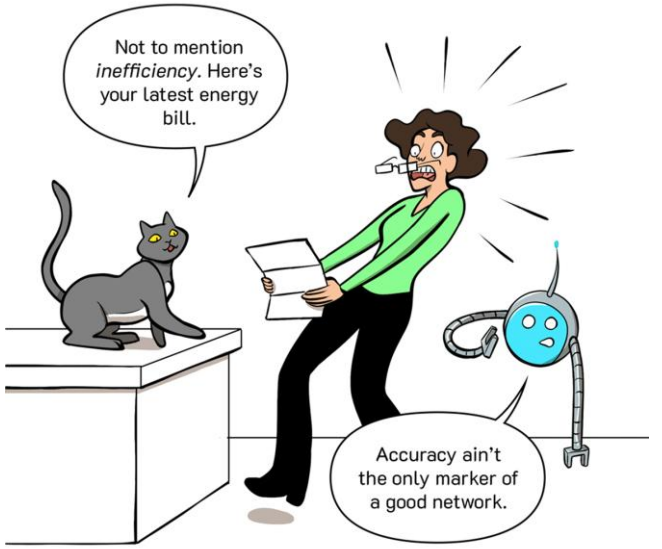
https://cloud.google.com/products/ai/ml-comic-2

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*).
      - Monitorar os erros no conjunto de validação, como visto na aula anterior.
    - Reduzir pesos (*weight decay*).
    - Ignorar parte dos neurônios aleatoriamente durante o treinamento (*dropout*).



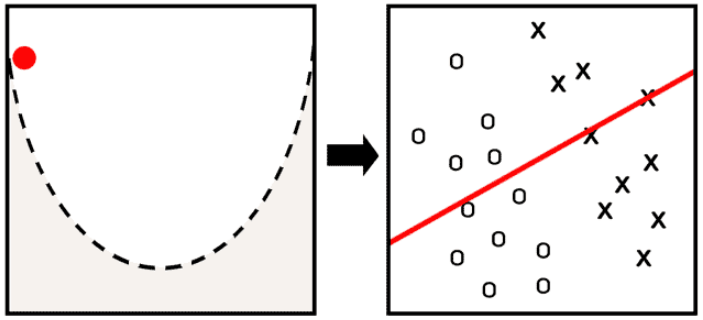
https://cloud.google.com/products/ai/ml-comic-2

https://sudonull.com/post/69729-Dropout-a-method-for-solving-the-problem-of-retraining-in-neural-networks

Critério de Parada

- Nem sempre é viável treinar uma rede até zerar a função de perda.
  - Erro Médio Quadrático, Entropia Cruzada Categórica, etc.
  - 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 perda, mas apenas um mínimo local.
  - E muitas vezes nem é desejável: ***overfitting***.
- É comum parar o treinamento quando a perda 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 perda.

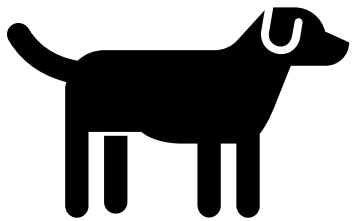
Atualização dos pesos



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

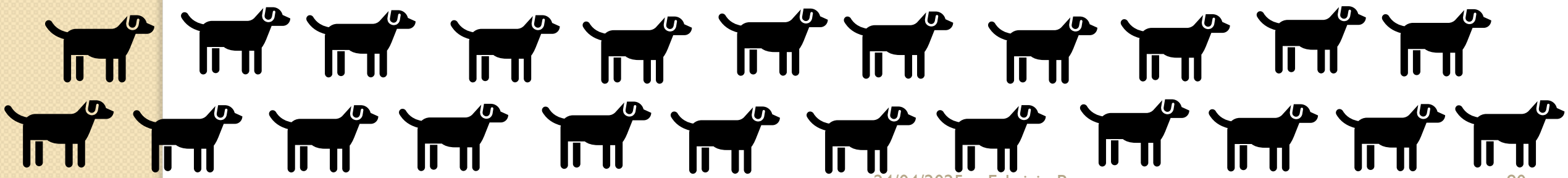
Atualização de Pesos por Padrão

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



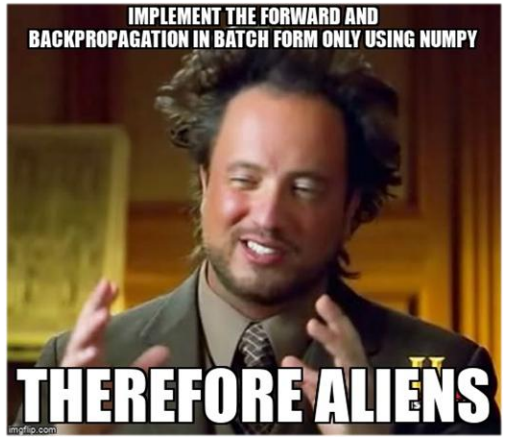
Atualização dos Pesos por Época

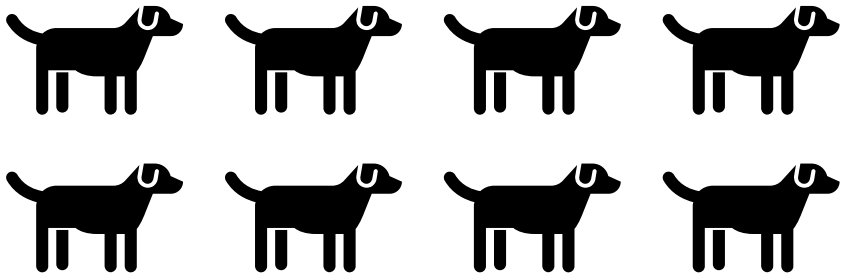
- Pesos atualizados depois que todos os padrões de treinamento forem apresentados.
- Geralmente mais estável.
- Pode ser lento se o conjunto de treinamento for grande e redundante.
- Requer mais memória.



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 época.





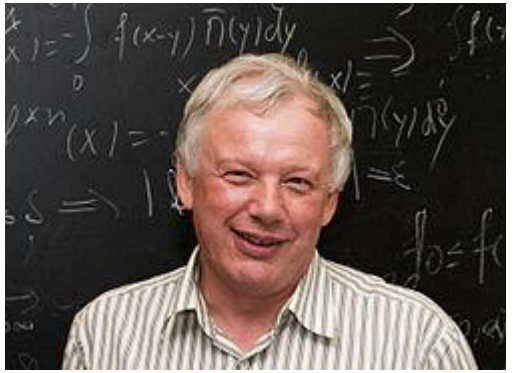
Redes Neurais Artificiais

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.

David S. Broomhead foi um matemático britânico especializado em sistemas dinâmicos e professor de matemática aplicada na Escola de Matemática da Universidade de Manchester.



David S. Broomhead
[*1950 – †2014]

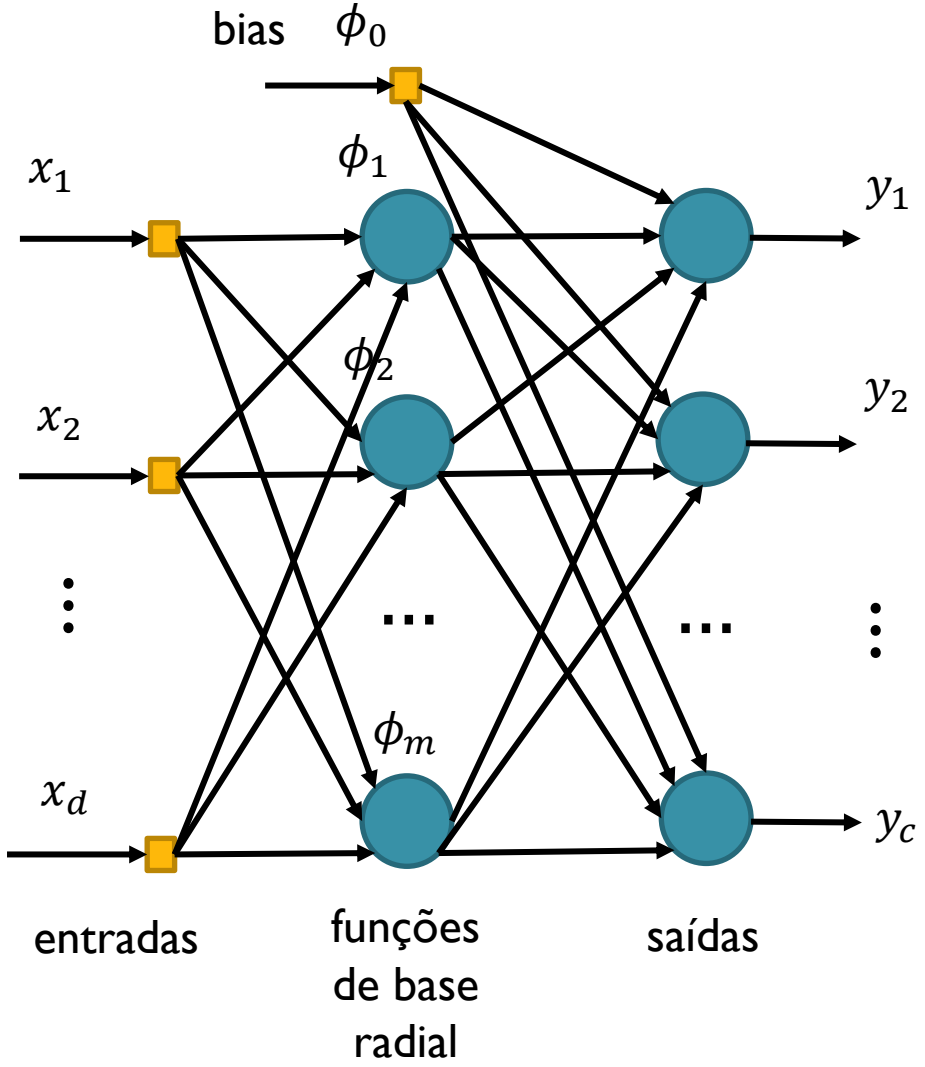
David G. Lowe

David Lowe é Professor Emérito no Departamento de Ciência da Computação da University of British Columbia (UBC). Foi pesquisador sênior no Google (2015-2018) e cofundador da Cloudburst Research, adquirida pelo Google em 2015. Atuou como professor de ciência da computação na UBC de 1987 a 2015.

Broomhead, David, and David Lowe. **Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks**. RSRE-Memo-4148, *Royal Signals and Radar Establishment, Malvern*, 1988.

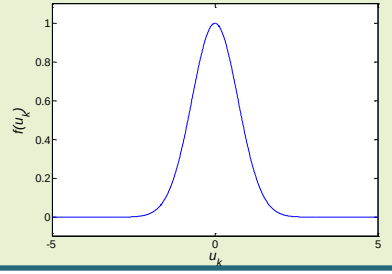
Redes de Função de Base Radial

- 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.

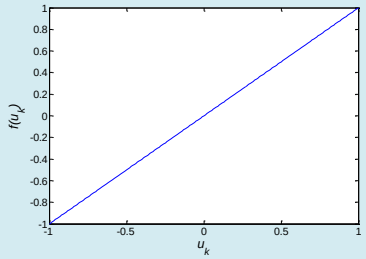


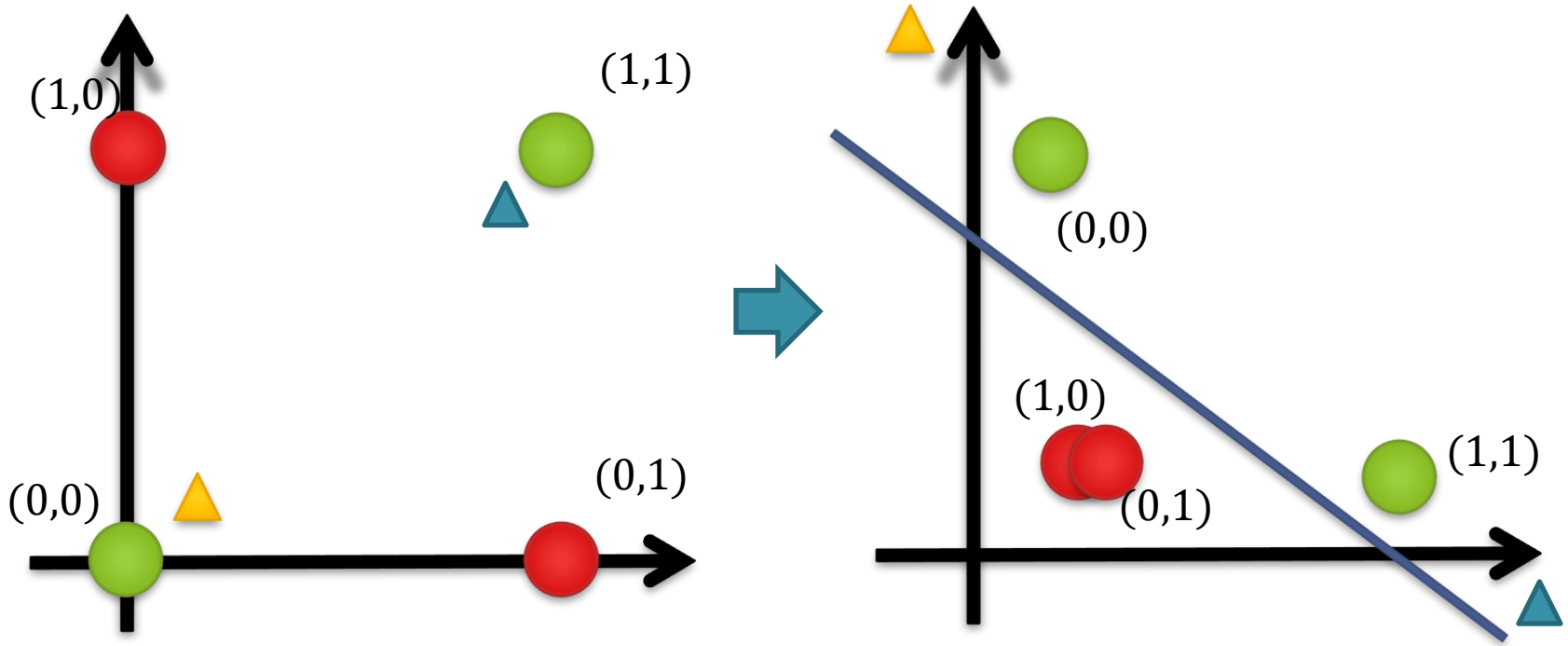
Redes de Função de Base Radial

$f(u_i) = \exp(-u_i{}^2)$



$f(u_i) = u_i$





Redes Neurais Artificiais

MAPAS AUTO-ORGANIZÁVEIS

Mapas Auto-Organizáveis

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

Teuvo Kohonen [*1934 – †2021]

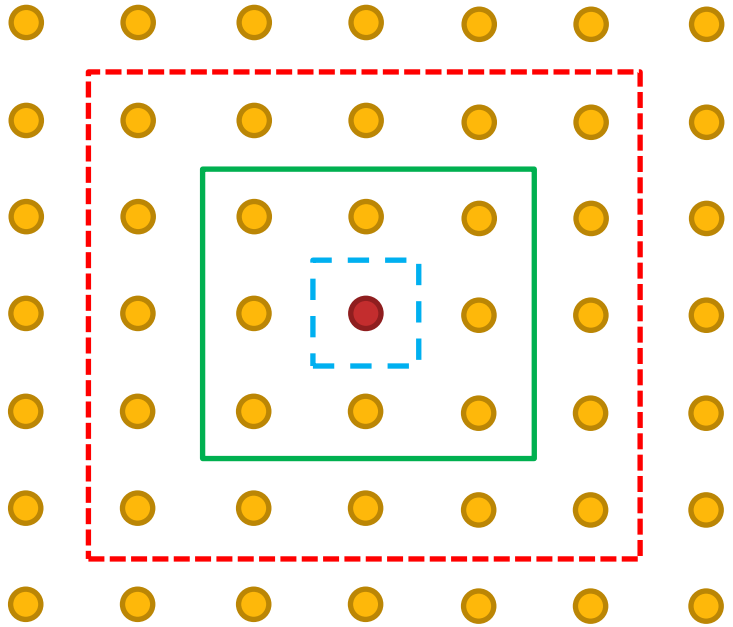
- Do inglês *Self Organizing Map* (SOM).
- Também chamados Mapas de Kohonen.
- Modelos não supervisionados.
- Baseado em aprendizagem competitiva.
  - Os neurônios competem entre si para serem ativados ou disparados.
    - Apenas um será ativado em dado instante de tempo.
      - Winner-Takes-All.
 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

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

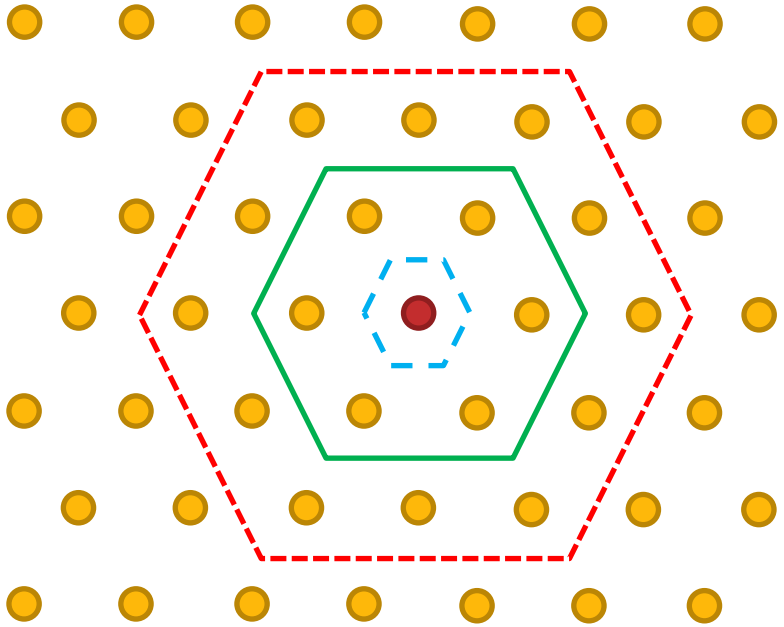
Mapas Auto-Organizáveis: vizinhanças

Função de Vizinhança

$R = 0$

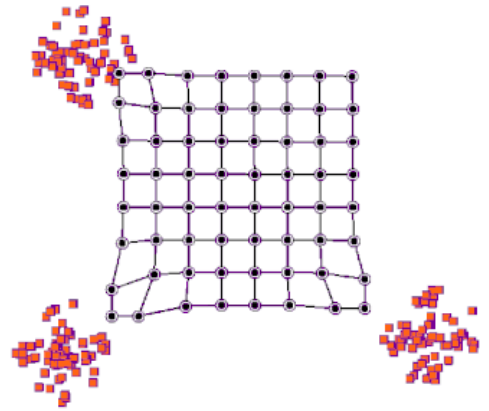
$R = 1$

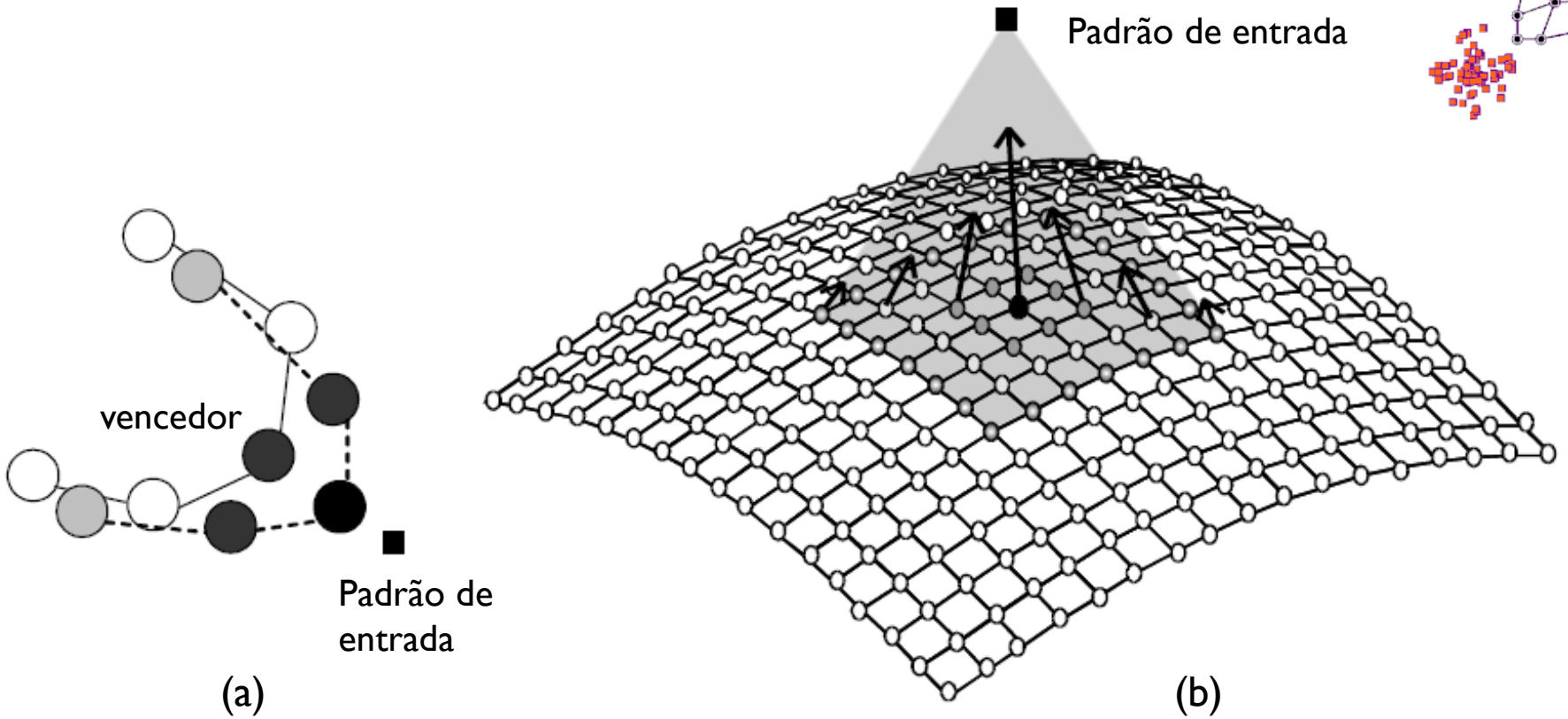
$R = 2$



Vizinhança de grade hexagonal

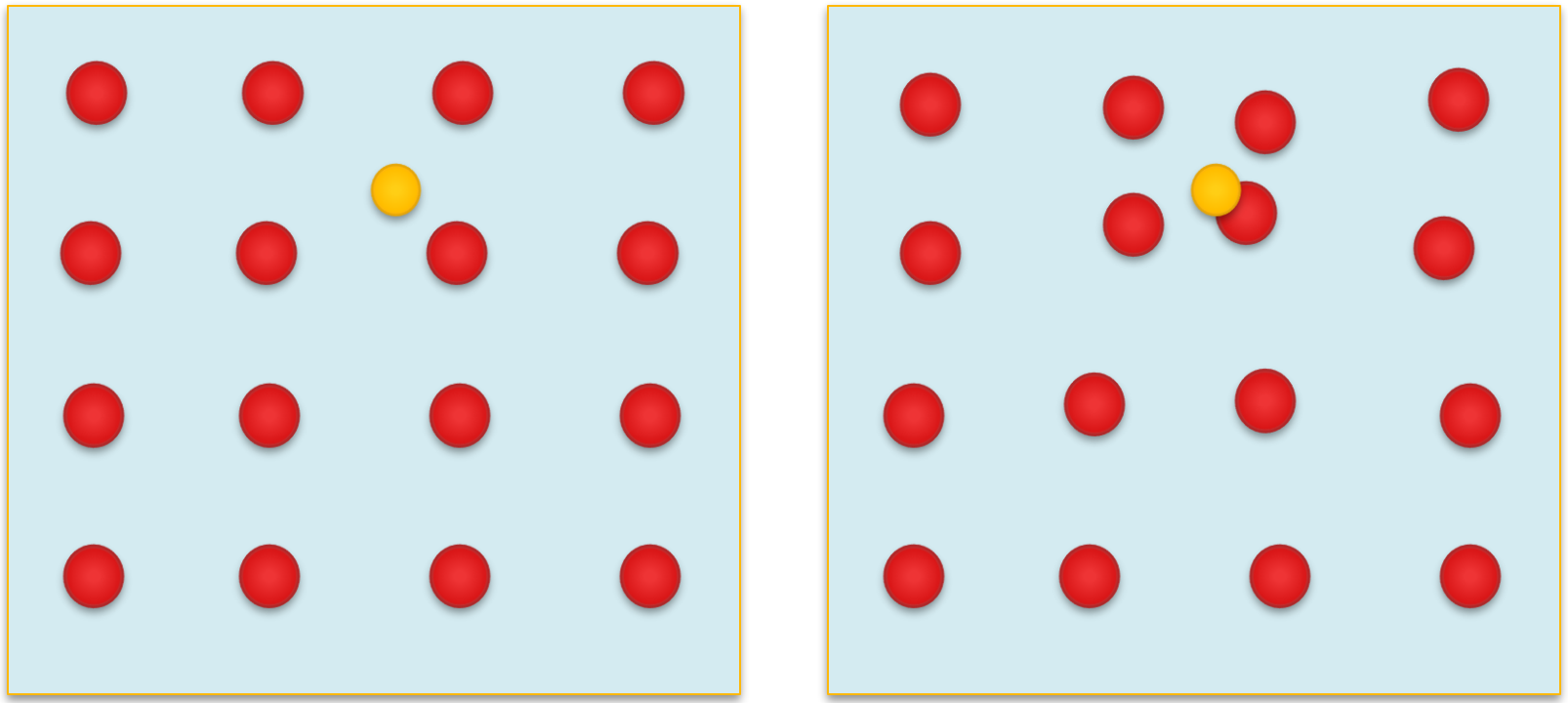
Mapas Auto-Organizáveis





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	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	médio	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	grande	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
tem	2 patas	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	4 patas	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	pelos	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	cascos	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	crina/juba	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	penas	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
gosta de	caçar	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	correr	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	voar	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	nadar	0	0	1	1	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	águia	águia
cão	cão	raposa	raposa	raposa	gato	gato	gato	águia	águia
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

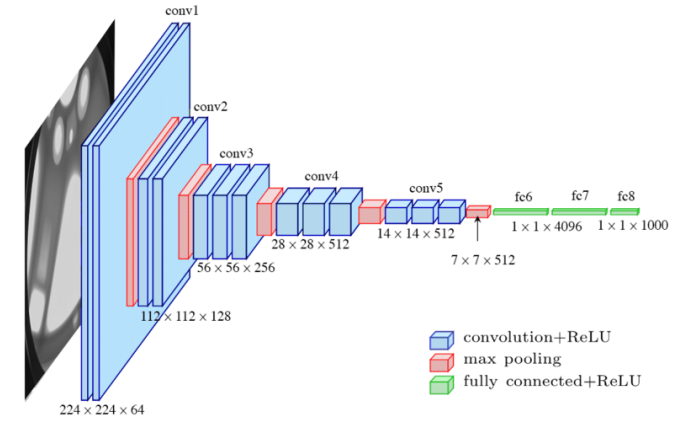
Mapa semântico obtido através do uso de mapeamento simulado de penetração de eletrodo. O mapa é dividido em três regiões apresentando: pássaros, espécies pacíficas e caçadores

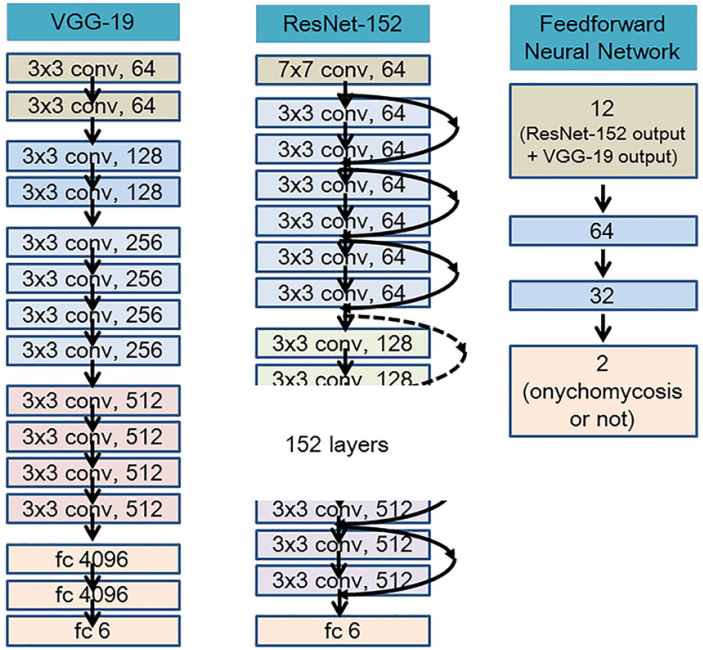
Redes Neurais Artificiais

REDES NEURAIS PROFUNDAS E OUTRAS REDES NEURAIS

Redes Neurais Profundas

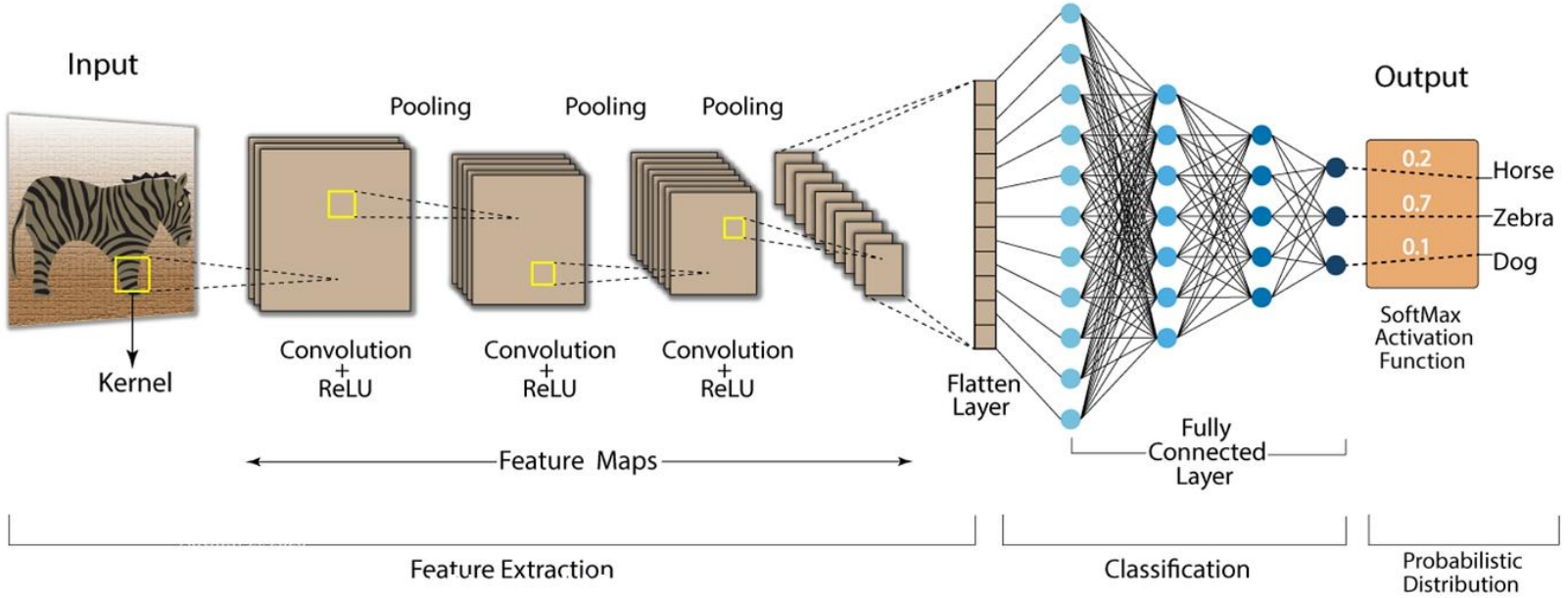
- *Deep Neural Networks*; *Deep Learning*
- Redes Neurais com mais de três camadas (incluindo entrada e saída).
  - Exemplo: Redes Neurais Convolucionais tem grande quantidade de camadas.
    - VGG (2014): 19 camadas, 144 milhões de parâmetros.
    - ResNet (2015): 152 camadas, 60 milhões de parâmetros.
    - ConvNeXtXLarge (2022): 350 milhões de parâmetros.





Redes Neurais Convolucionais

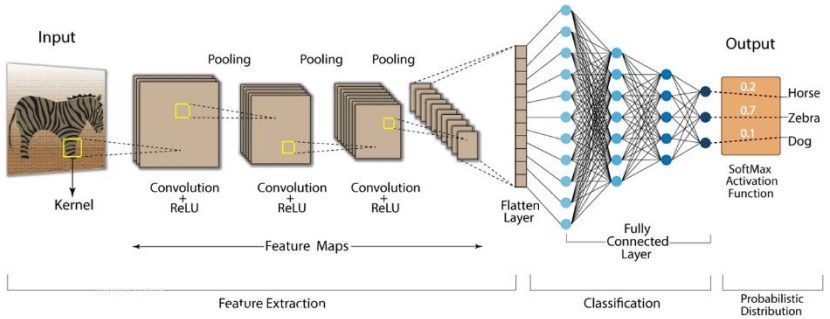
- Amplamente usadas em tarefas de visão computacional.
- Capazes de extrair características diretamente de imagens, dispensando uma etapa separada de extração de atributos.



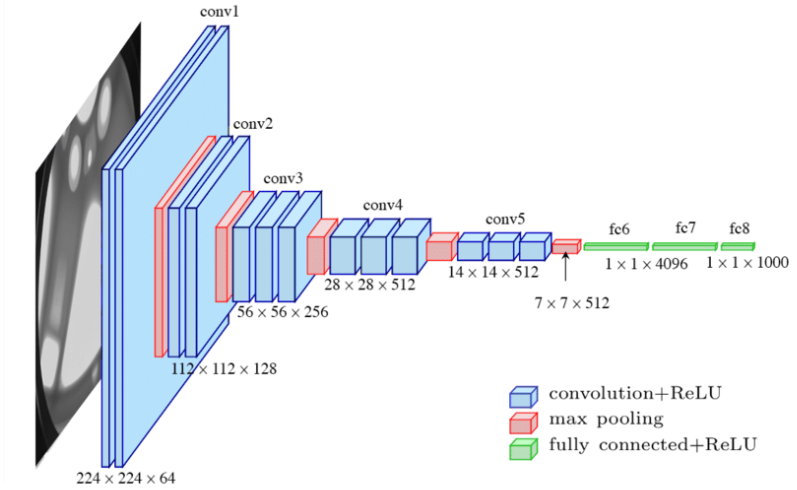
https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5

Camadas das Redes Neurais Convolucionais

- **Camadas Convolucionais**
  - Aplicam filtros (kernels) para extrair características locais dos dados.
    - Ex.: bordas, texturas e formas em imagens).
  - Preservam relações espaciais, permitindo que as CNNs entendam a estrutura do dado.
- **Pooling**
  - Reduz a dimensionalidade das características extraídas, mantendo as informações mais relevantes.
    - Ex.: Max pooling (escolhe o valor máximo) e Average pooling (calcula a média).



- **Camadas Completamente Conectadas**
  - Ao final do processo, conectam todas as características extraídas para realizar tarefas como classificação.
- **Função de Ativação**
  - Como em outras redes neurais, as CNNs utilizam funções como ReLU para introduzir não linearidade.



Outras Redes Neurais

- Máquinas de Comitê
- Redes ART (Adaptive Resonance Theory)
- Redes de Hopfield (HN)
- Máquinas de Boltzmann (BM)
- Redes de Crença Sigmóide
- Máquina de Helmholtz
- Echo States Networks (ESN)
- Long / Short Term Memory (LSTM)
- Gated Recurrent Unit (GRU)
- Auto Encoders (AE)
- Variational Auto Encoders (VAE)
- Denoising Auto Encoders (DAE)
- Sparse Auto Encoders (SAE)
- Markov Chains (MC)
- 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)
- Differentiable Neural Computer (DNC)
- Capsule Networks (CN)
- Attention Networks (AN)
- Graph Convolutional Network (GCN)

https://www.asimovinstitute.org/neural-network-zoo/

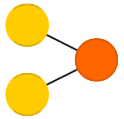
A mostly complete chart of

Neural Networks

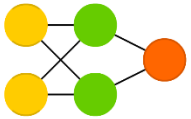
 asimovinstitute.org

- Input Cell
- Backfed Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Capsule Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Gated Memory Cell
- Kernel
- Convolution or Pool

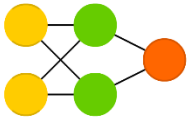
Perceptron (P)



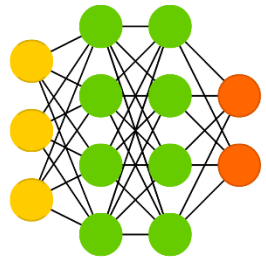
Feed Forward (FF)



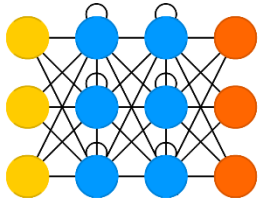
Radial Basis Network (RBF)



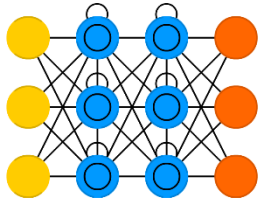
Deep Feed Forward (DFF)



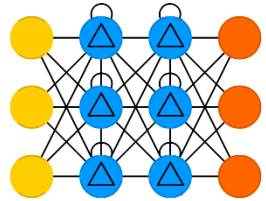
Recurrent Neural Network (RNN)



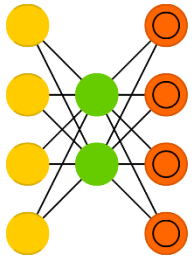
Long / Short Term Memory (LSTM)



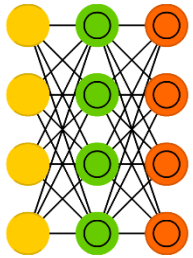
Gated Recurrent Unit (GRU)



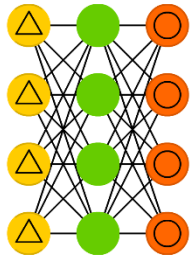
Auto Encoder (AE)



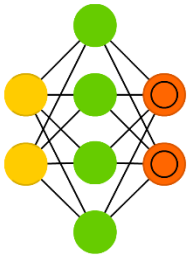
Variational AE (VAE)



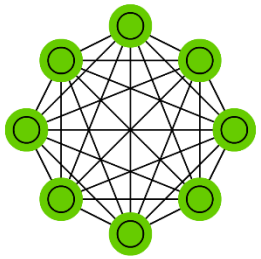
Denoising AE (DAE)



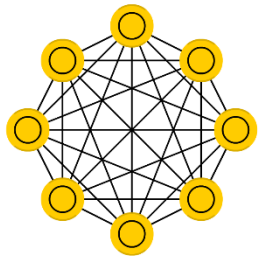
Sparse AE (SAE)



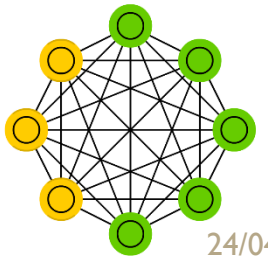
Markov Chain (MC)



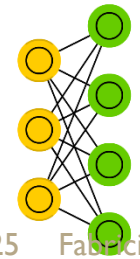
Hopfield Network (HN)



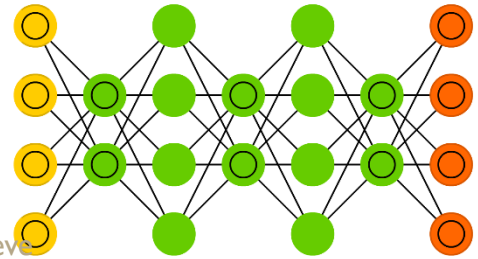
Boltzmann Machine (BM)



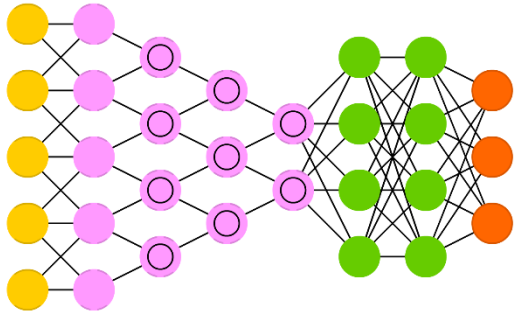
Restricted BM (RBM)



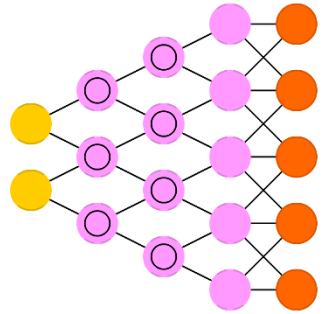
Deep Belief Network (DBN)



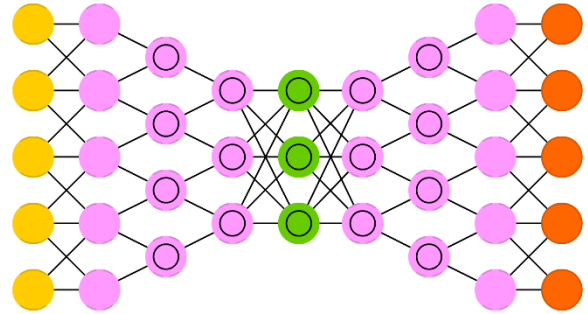
Deep Convolutional Network (DCN)



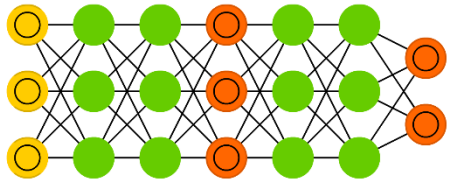
Deconvolutional Network (DN)



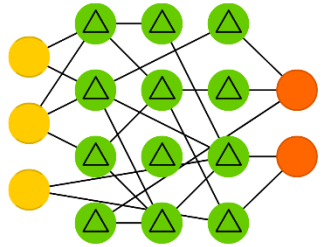
Deep Convolutional Inverse Graphics Network (DCIGN)



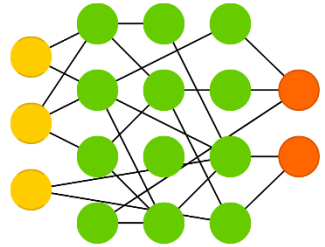
Generative Adversarial Network (GAN)



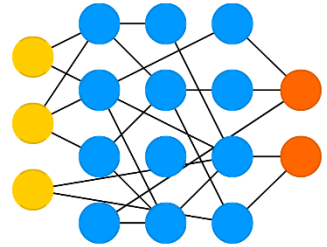
Liquid State Machine (LSM)



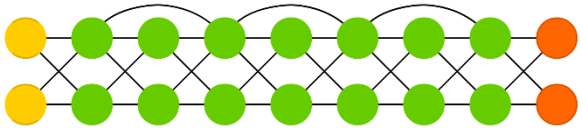
Extreme Learning Machine (ELM)



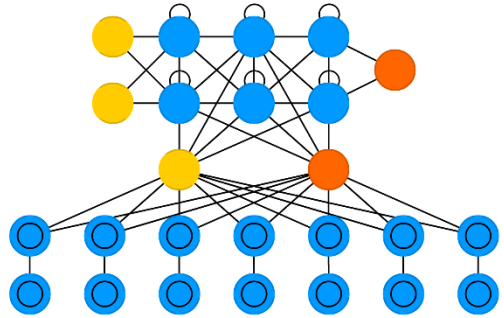
Echo State Network (ESN)



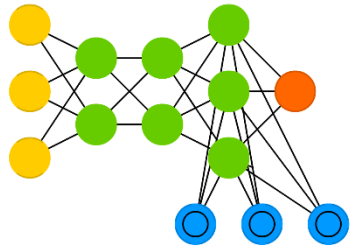
Deep Residual Network (DRN)



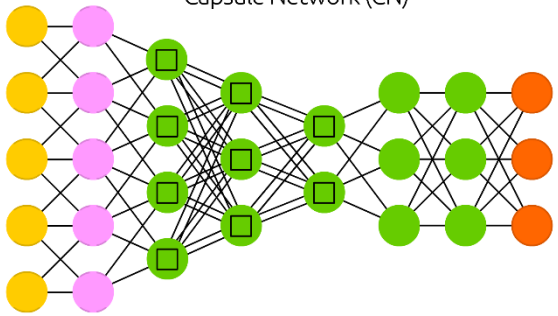
Differentiable Neural Computer (DNC)



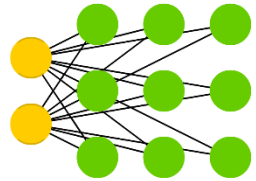
Neural Turing Machine (NTM)



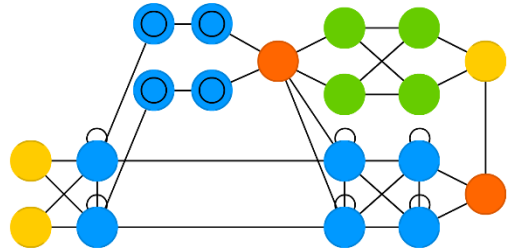
Capsule Network (CN)



Kohonen Network (KN)



Attention Network (AN)



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
- Geração de Texto, Imagens, Sons (fala, música, etc...)

https://cheekymunkey.co.uk/voice-activated-computers-future/

RNAs na Indústria

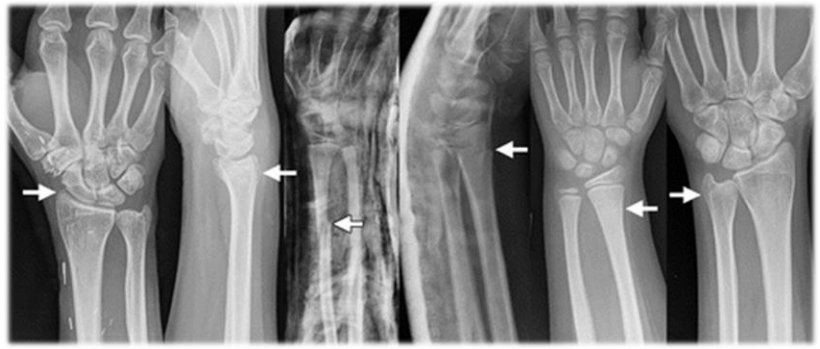
- RNAs são muito utilizadas em indústrias no exterior.
  - Especialmente quando instalação rápida ou grande velocidade de operação é requerida.
  - Ex.: Caixas de peixes congelados da *Birds Eye* são inspecionados visualmente por uma RNA.
    - Caixas imperfeitas são rejeitadas para exportação.

RNAs no Setor de Serviços

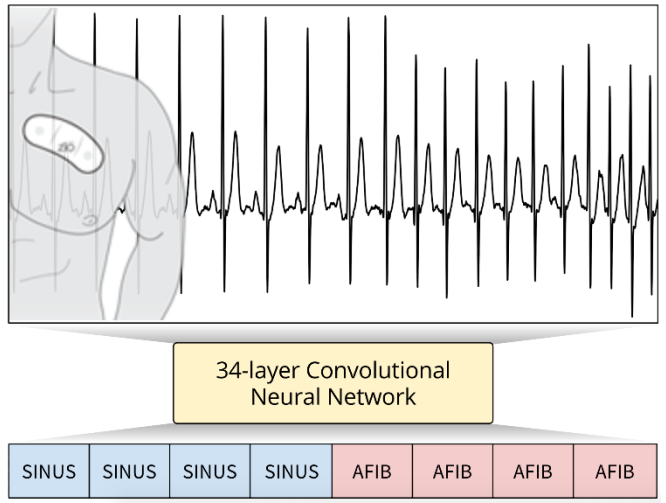
- Grande quantidade de pesquisas sobre a utilização de RNAs para previsão financeira.
  - Tendências sugerem expansão destas aplicações.
  - Ex.: Cartão de crédito VISA utiliza RNAs para liberação de cartões e detecção de fraudes.

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.



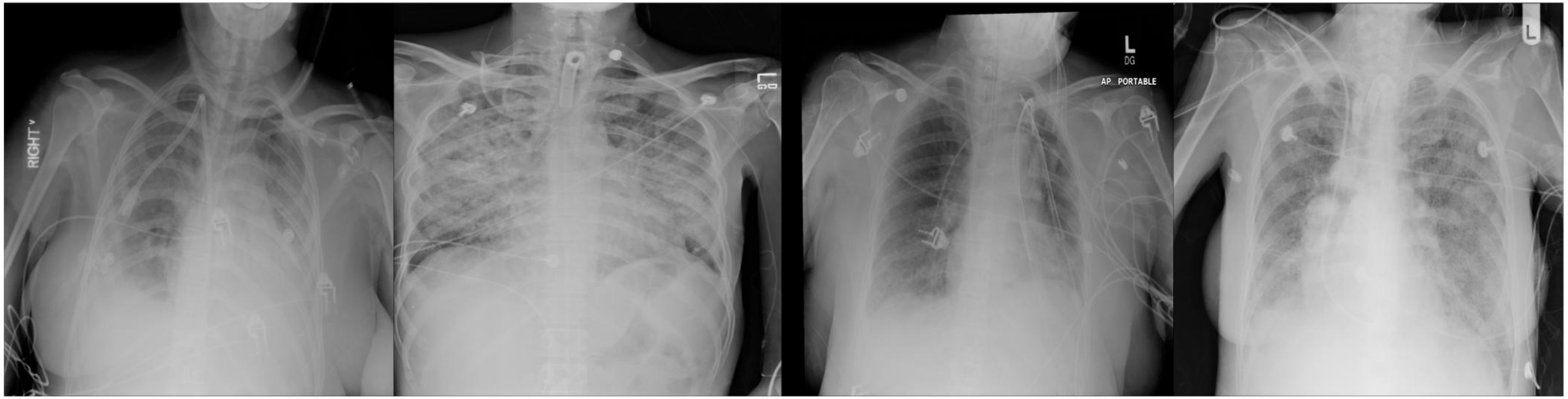
https://pubs.rsna.org/doi/10.1148/ryai.2019180001



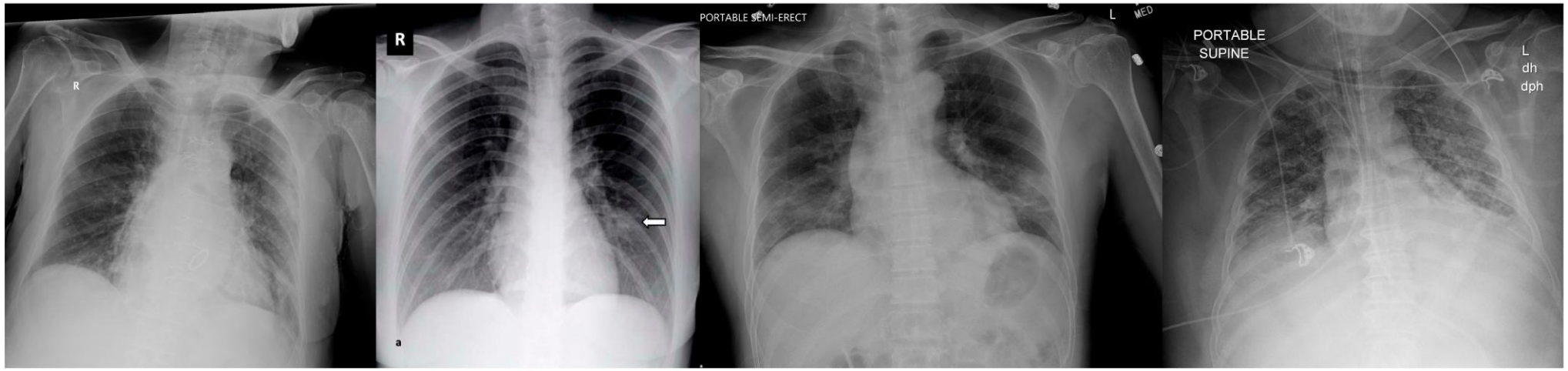
https://www.arxiv-vanity.com/papers/1707.01836/

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

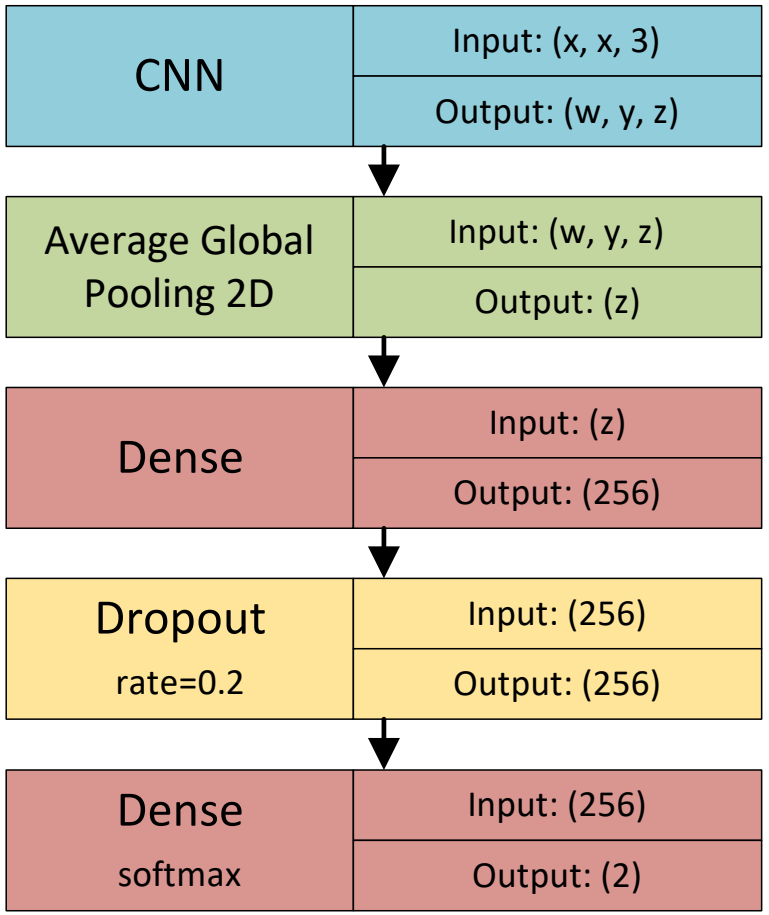
Negativo



Positivo



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

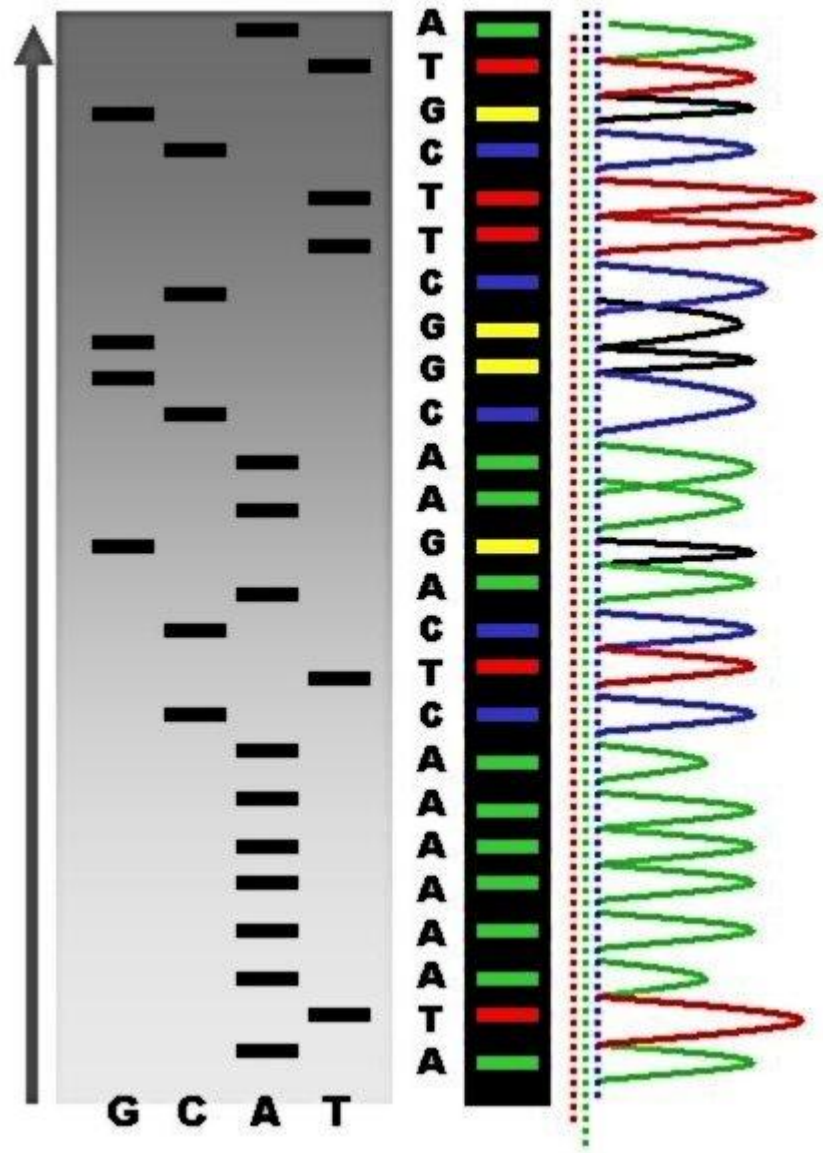


Model	ACC Mean	ACC S.D.	TPR Mean	TPR S.D.	PPV Mean	PPV S.D.	F1 Mean	F1 S.D.
DenseNet169	**0,9815**	0,0056	**0,9700**	0,0138	0,9930	0,0075	**0,9812**	0,0058
EfficientNetB2	0,9760	0,0049	0,9600	0,0141	0,9918	0,0051	0,9756	0,0052
InceptionResNetV2	0,9755	0,0099	0,9590	0,0246	0,9919	0,0051	0,9749	0,0106
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

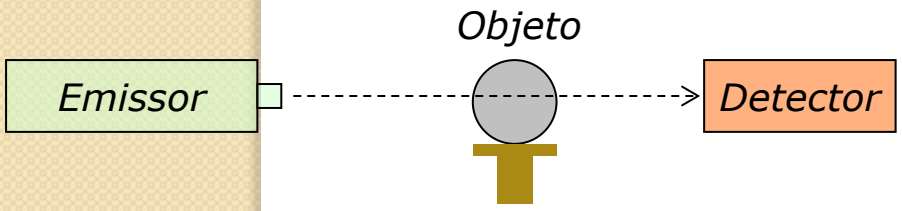
BREVE, Fabricio Aparecido. **COVID-19 detection on Chest X-ray images: A comparison of CNN architectures and ensembles**. *Expert Systems With Applications*, v. 204, p.117549, 2022. https://doi.org/10.1016/j.eswa.2022.117549

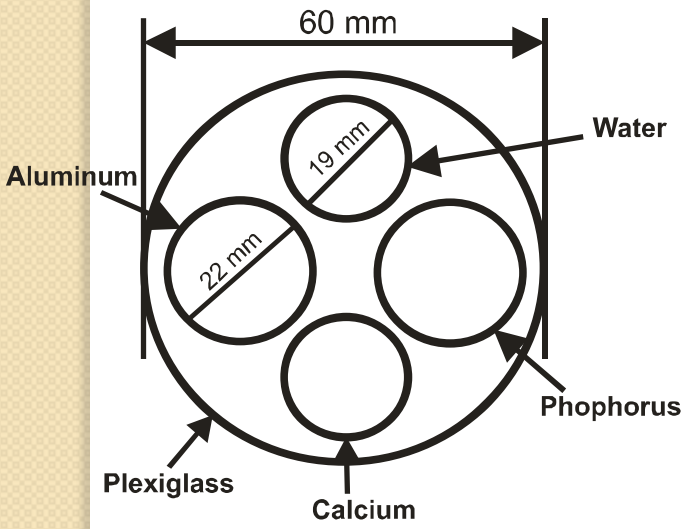
RNAs na Bioinformática

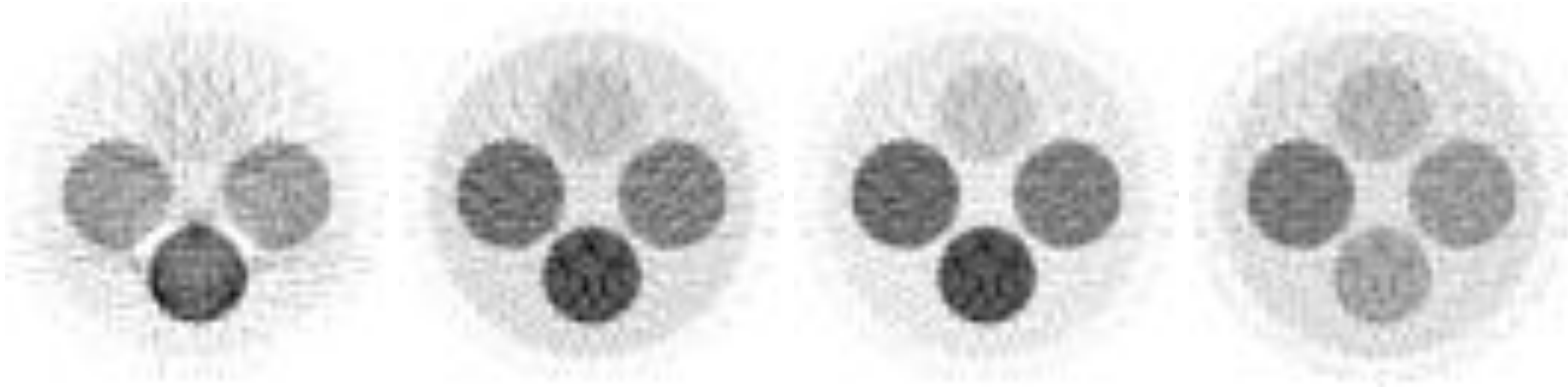
- Existem trabalhos que utilizam RNAs para:
  - Reconhecimento de genes em sequências de DNA.
  - Análise de expressão gênica.
  - Previsão da estrutura de proteínas.



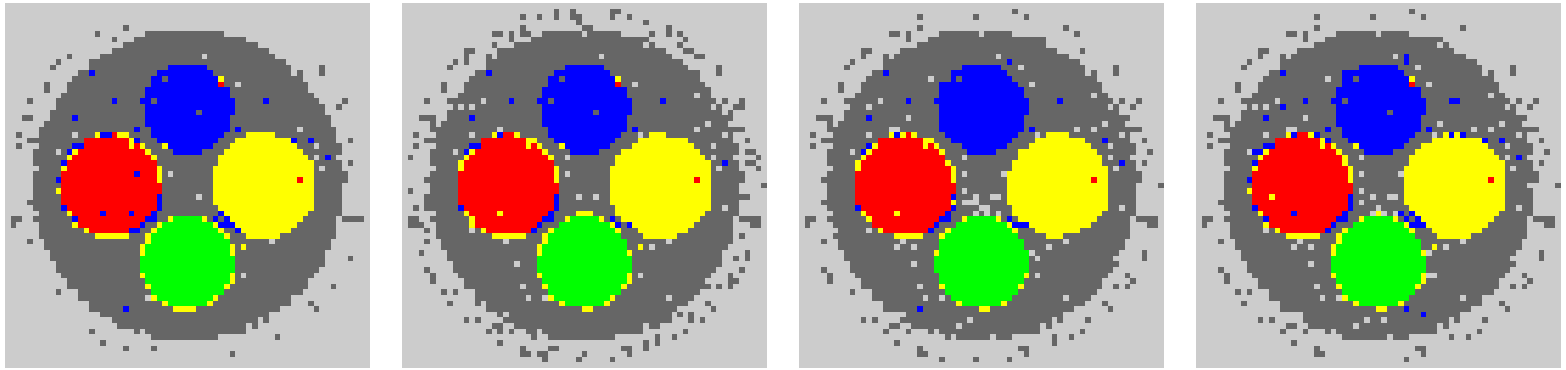
RNAs nas Ciências dos Solos







40 keV X-Ray | **60 keV γ-ray (Americium)** | **85 keV X-Ray** | **662 keV γ-ray (Cesium)**



Classificador Único (MLP)
Erro Estimado: 0,0083
Coeficiente Kappa: 0,9900

Bagging
Erro Estimado: 0,0083
Coeficiente Kappa: 0,9900

Decision Templates
Erro Estimado: 0,0042
Coeficiente Kappa: 0,9950

Dempster-Shafer
Erro Estimado: 0,0125
Coeficiente Kappa: 0,9850

BREVE, Fabricio Aparecido ; PONTI JUNIOR, Moacir Pereira ; MASCARENHAS, Nelson Delfino D'ávila. **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 D'ávila. **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







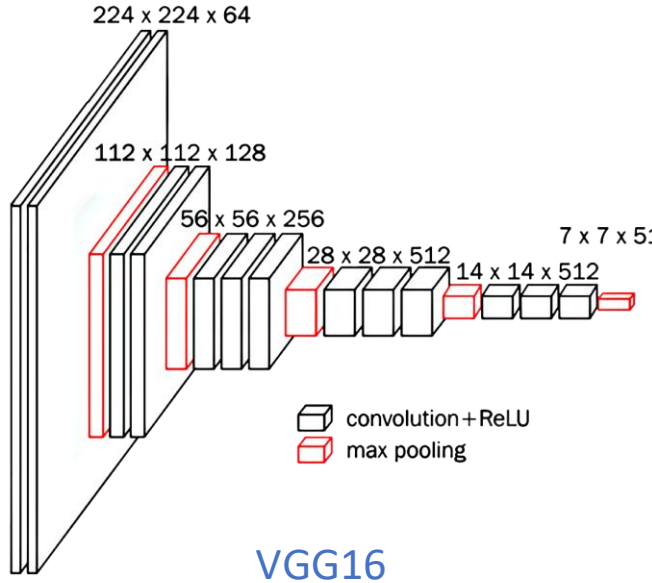

BREVE, Fabricio Aparecido; FISCHER, Carlos Norberto. **Visually Impaired Aid using Convolutional Neural Networks, Transfer Learning, and Particle Competition and Cooperation** In: 2020 International Joint Conference on Neural Networks (IJCNN 2020), 2020, Glasgow, UK. *Proceedings of 2020 International Joint Conference on Neural Networks (IJCNN 2020)*, 2020.

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

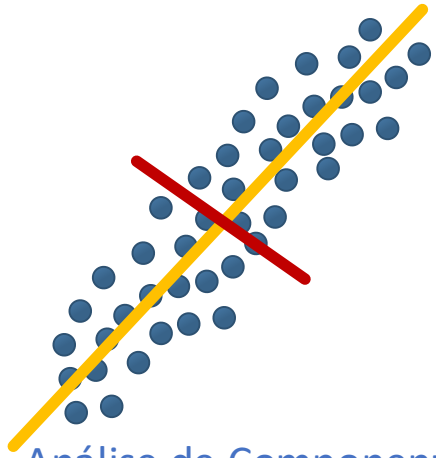
BREVE, Fabricio Aparecido. **Convolutional Neural Networks and Ensembles for Visually Impaired Aid** In: The 23rd International Conference on Computational Science and Its Applications, 2023, Atenas, Grécia. *Computational Science and Its Applications — ICCSA 2023 – Lecture Notes in Computer Science*. Cham: Springer Nature Switzerland, 2023. v.13956. p.520 – 534.

https://doi.org/10.1007/978-3-031-36805-9_34

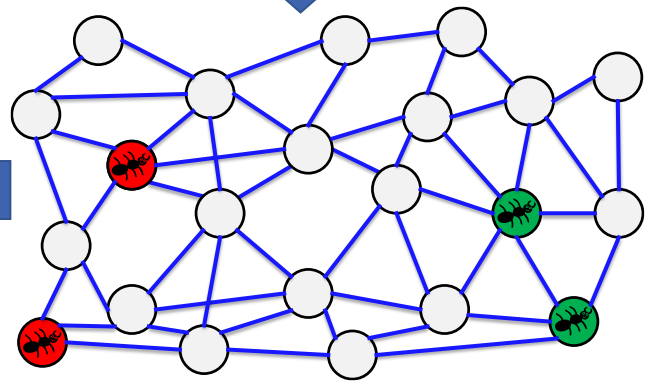
Imagens
224 x 224 x 3



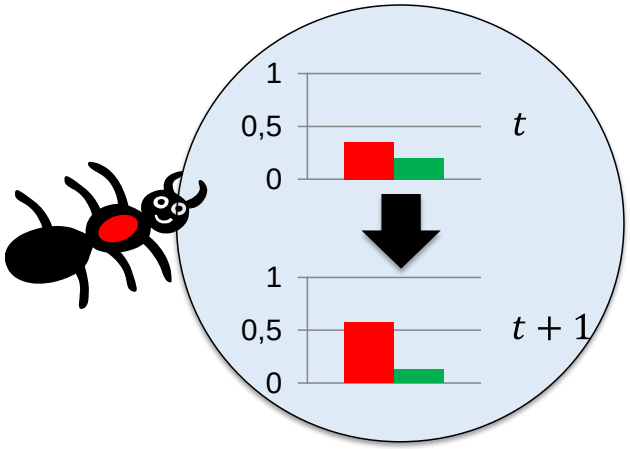
VGG16



Análise de Componentes Principais



Grafo Sem Peso Não Direcionado



Competição e Cooperação entre Partículas

Imagens Classificadas

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 extractor de atributos

Identificação de Jogos de Videogame à partir de *screenshots* com redes neurais convolucionais


Enduro - Atari 2600


Super Mario Bros. - NES


The Legend of Zelda - NES


Alex Kidd in Miracle World - Master System


Bonk's Adventure - PC Engine


Sonic the Hedgehog - Genesis


Super Mario World - Super Nintendo


Virtua Fighter 2 - Sega Saturn


Gran Turismo - PlayStation


Final Fantasy VII - PlayStation


Super Mario 64 - Nintendo 64


Sonic Adventure - Dreamcast


Grand Theft Auto: San Andreas - PlayStation 2


Super Smash Bros.: Melee - GameCube


Halo: Combat Evolved - Xbox


Halo 3 - Xbox 360


The Last of Us - PlayStation 3


Wii Sports - Wii


Mario Kart 8 - Wii U


The Witcher 3: Wild Hunt - PlayStation 4


Horizon: Zero Dawn - PlayStation 4


Gears of War 4 - Xbox One


The Legend of Zelda: Breath of the Wild - Nintendo Switch


Forza Horizon 5 - Xbox Series X/S

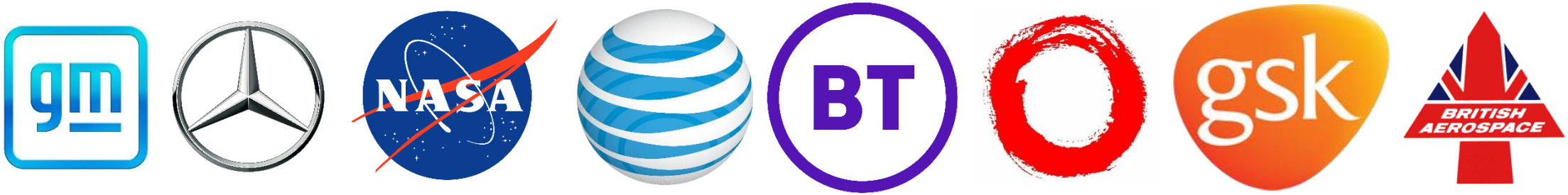

Ratchet & Clank: Rift Apart - PlayStation 5

System	Architecture	Weights	Accuracy
Atari 2600	EfficientNetB3	Arcade	90,64%
NES	EfficientNetV2S	ImageNet	72,51%
Master System	EfficientNetV2S	ImageNet	76,20%
PC Engine	EfficientNetB3	Arcade	81,08%
Mega Drive	EfficientNetV2S	ImageNet	74,56%
Super Nintendo	DenseNet201	ImageNet	74,51%
Sega Saturn	DenseNet201 EfficientNetV2S	Arcade	83,14%
PlayStation	EfficientNetV2S	ImageNet	74,77%
Nintendo 64	EfficientNetV2S	Arcade	85,28%
Dreamcast	EfficientNetV2S	Arcade	78,86%
PlayStation 2	EfficientNetV2S	Arcade	76,73%
GameCube	EfficientNetB3	Arcade	83,96%
Xbox	EfficientNetV2S	Arcade	86,42%
Xbox 360	EfficientNetV2S	ImageNet	80,25%
PlayStation 3	EfficientNetV2S	Arcade	72,47%
Wii	EfficientNetV2S	ImageNet	83,96%
Wii U	EfficientNetB3	Arcade	76,52%
PlayStation 4	EfficientNetB3	ImageNet	64,77%
Xbox One	EfficientNetV2S	ImageNet	66,25%
Nintendo Switch	EfficientNetB3	Arcade	81,80%
Xbox Series	EfficientNetV2S	ImageNet	100,00%
PlayStation 5	EfficientNetV2S	Arcade	68,62%
Average			78,79%

BREVE, Fabricio Aparecido. **From Pixels to Titles: Video Game Identification by Screenshots using Convolutional Neural Networks**. *IEEE Transactions on Games*, 2025.

Empresas pioneiras no uso de RNAs

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

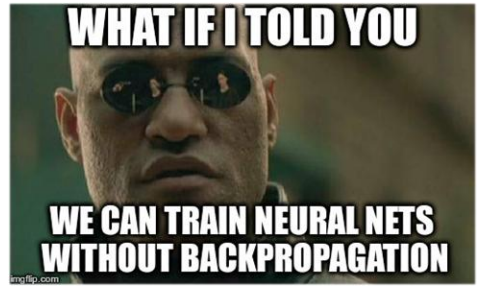


Empresas pioneiras no uso de RNAs no Brasil

- 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 de Reconhecimento de Padrões e muitos outros.
- Alguns problemas:
  - Como definir valores de seus hiperparâmetros?
    - Otimização.
      - Algoritmos Genéticos.
  - Como explicar as decisões tomadas pela rede?
    - Extração de conhecimento.



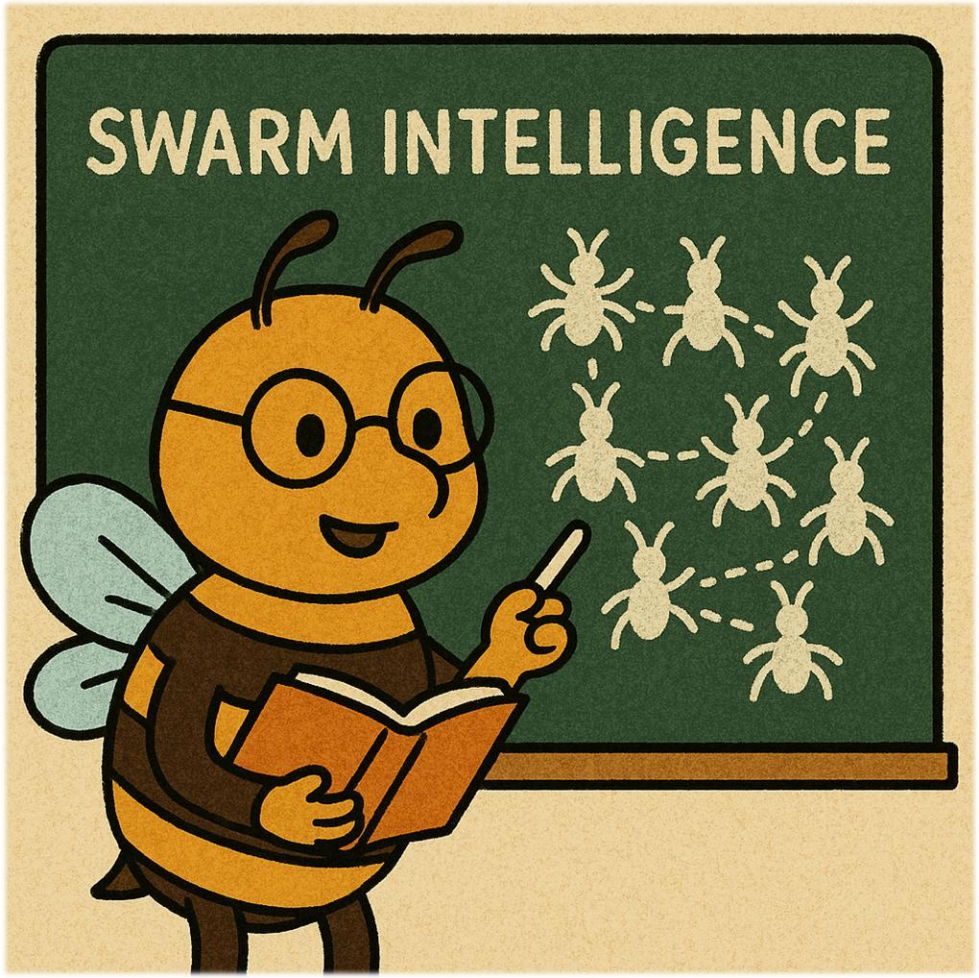
https://youtu.be/kaBwAfQ1-zo

Redes Neurais

(Verso 1) Nas entranhas do código, um labirinto de conexões, Redes neurais se entrelaçam, buscando revelações. Camadas ocultas, sinapses em dança, Aprendizado profundo, como uma eterna bonança.
(Refrão) Redes neurais, teço sonhos em bits, Classificando o mundo, como um poeta que grita. Pesos e ativações, segredos em cascata, A inteligência emerge, como uma estrela prateada.
(Verso 2) Nós e bias, como fios invisíveis, Aprendendo com os dados, como sábios incríveis. Backpropagation, como um vento que sopra, Ajustando os pesos, como um artista que molda.
(Refrão) Redes neurais, teço sonhos em bits, Classificando o mundo, como um poeta que grita. Pesos e ativações, segredos em cascata, A inteligência emerge, como uma estrela prateada.
(Ponte) Em cada neurônio, uma centelha de luz, Aprendendo padrões, como um rio que conduz. Convoluções e recorrências, como versos que rimam, Redes neurais, dançando no abismo do algoritmo.
(Refrão) Redes neurais, teço sonhos em bits, Classificando o mundo, como um poeta que grita. Pesos e ativações, segredos em cascata, A inteligência emerge, como uma estrela prateada.
(Final) E quando o treinamento cessa, a rede se liberta, Uma mente artificial, criada pela descoberta. Redes neurais, um eco do nosso desejo, A busca pela verdade, em cada byte que vejo.

- Letra feita pelo Copilot (GPT-4)
- Capa feita pelo DALL-E
- Música feita pelo Udio

Próxima Aula: Inteligência de Enxames



- Agentes
- Princípios Básicos
- Enxames
- Colônias de Formigas
- Bandos de Pássaros
- Rebanhos de Animais
- Enxames de Abelhas
- Cardumes de Peixes
- Tráfego de Veículos
- Multidão de Pessoas
- Partículas Humanas Inteligentes
- Algoritmos Baseados em Enxames
- Bibliografia

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.