

IA APLICADA AO MELHORAMENTO GENÉTICO

Redes Neurais Artificiais

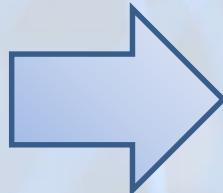
Vinícius Quintão Carneiro
Vinicius.carneiro@ufla.br



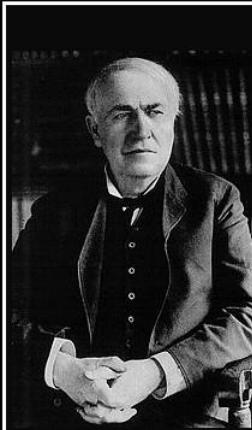
Redes Neurais Artificiais

Sistemas matemáticos e computacionais não lineares, inspirados na estrutura e operação do cérebro humano.

- *Aprendizado*
- *Associação*
- *Generalização*
- *Abstração*

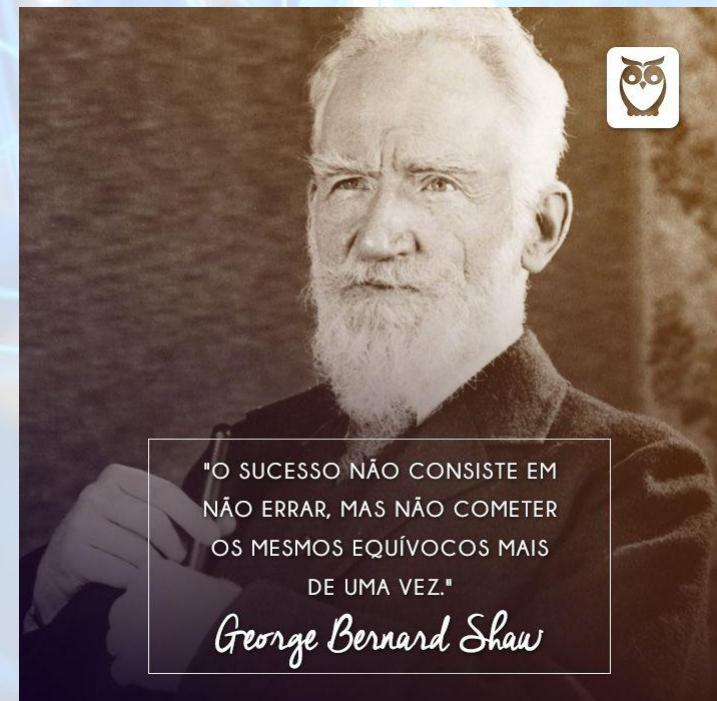


- *Análises Classificatórias*
 - *Análises Discriminantes*
- *Análises de Predição*
 - *Rregressão*

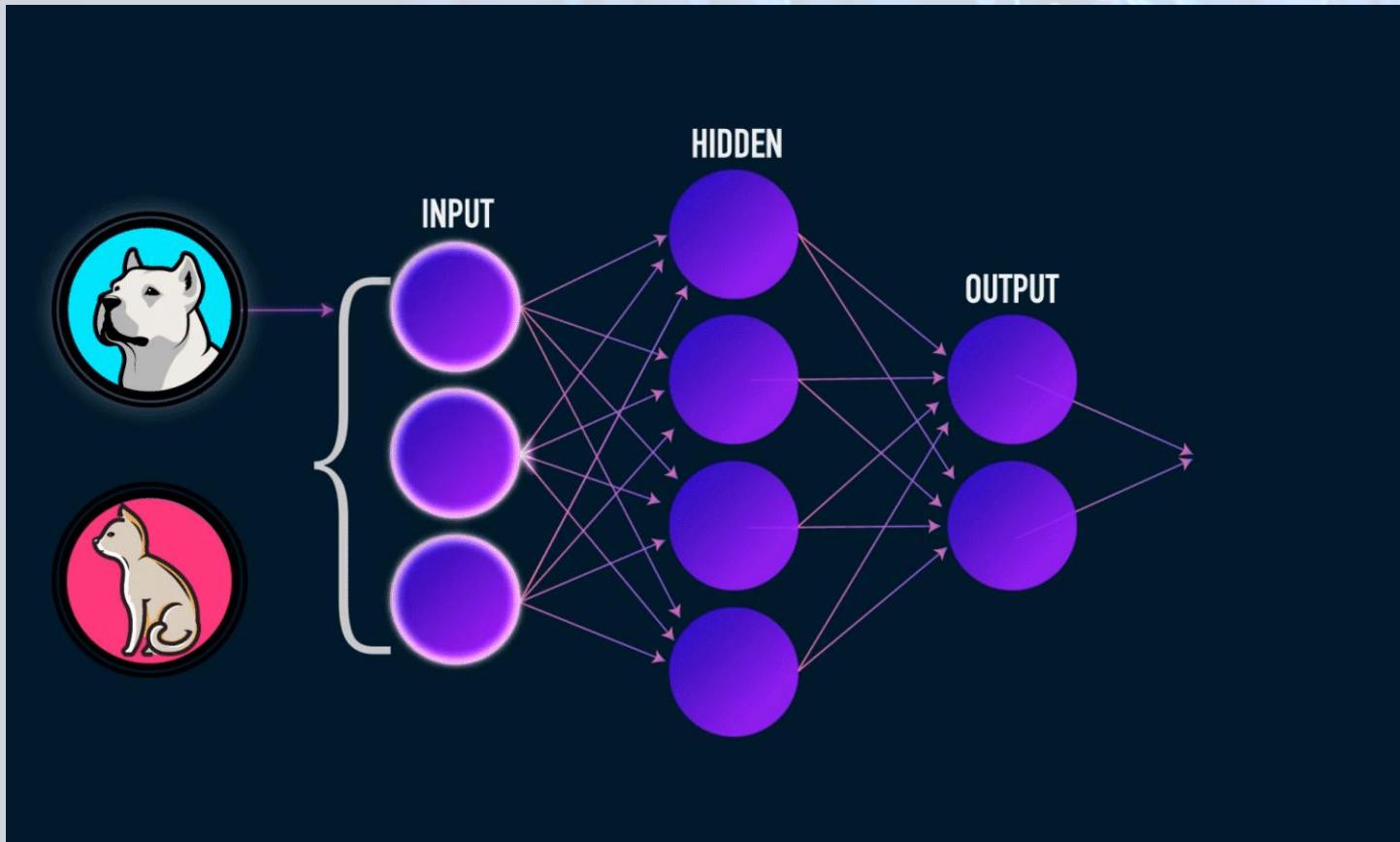


Eu aprendi muito mais com os meus erros do que com meus acertos.

(Thomas Edison)



Redes Neurais Artificiais



Redes Neurais Artificiais

Características Principais:

- Adaptação por experiência
- Capacidade de aprendizado
- Habilidade de generalização
- Suporta ruídos e perda de informações

Memória Associativa



Redes Neurais Artificiais

Utilizações:

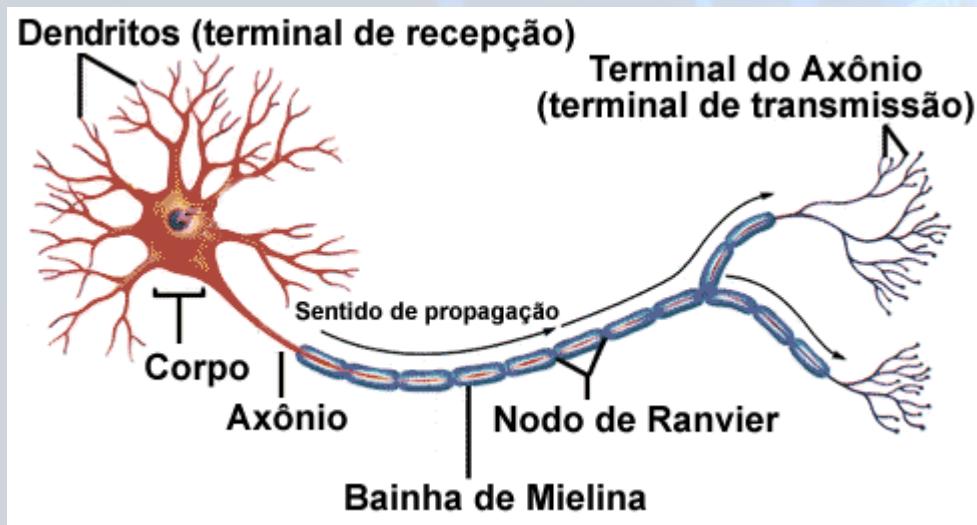
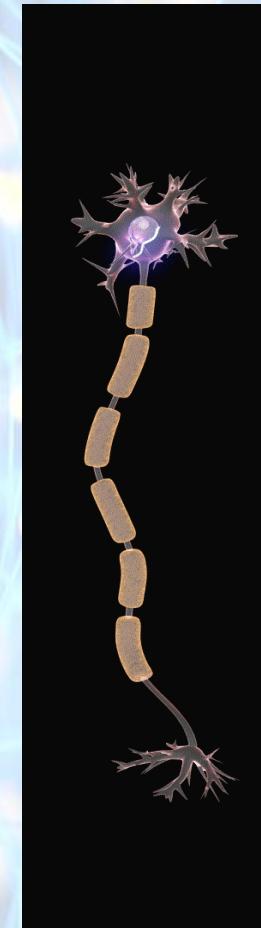
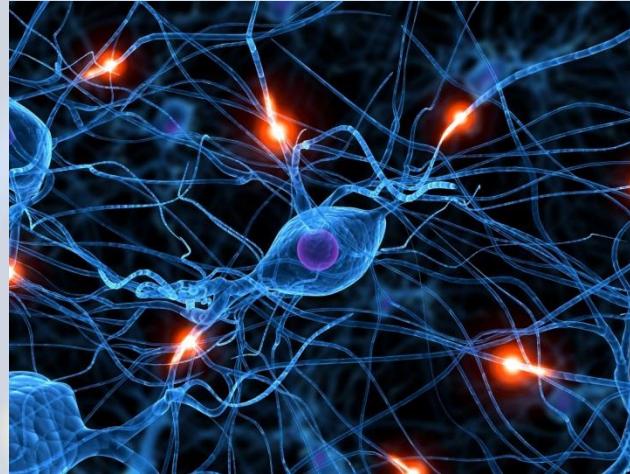
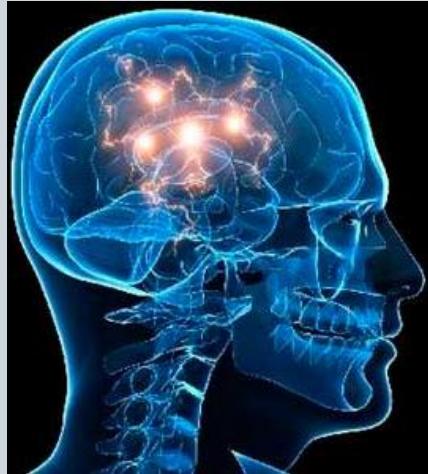
- Aproximador universal de funções;

x	$f(x) = X^2+3X$
0	0
1	4
2	16

- Reconhecimento de padrões;
- Agrupamento de dados;
- Sistemas de previsão;
- Análise e processamento de dados sem a necessidade de uma modelagem previa.

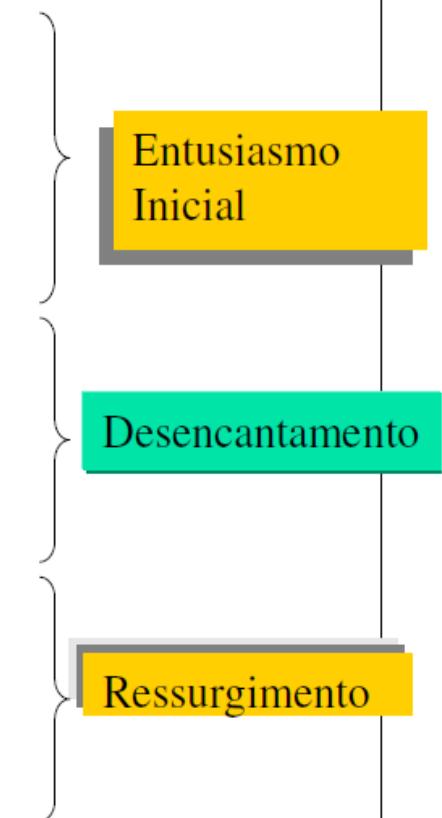
Redes Neurais Artificiais

Estrutura e Operação -> Cérebro Humano.



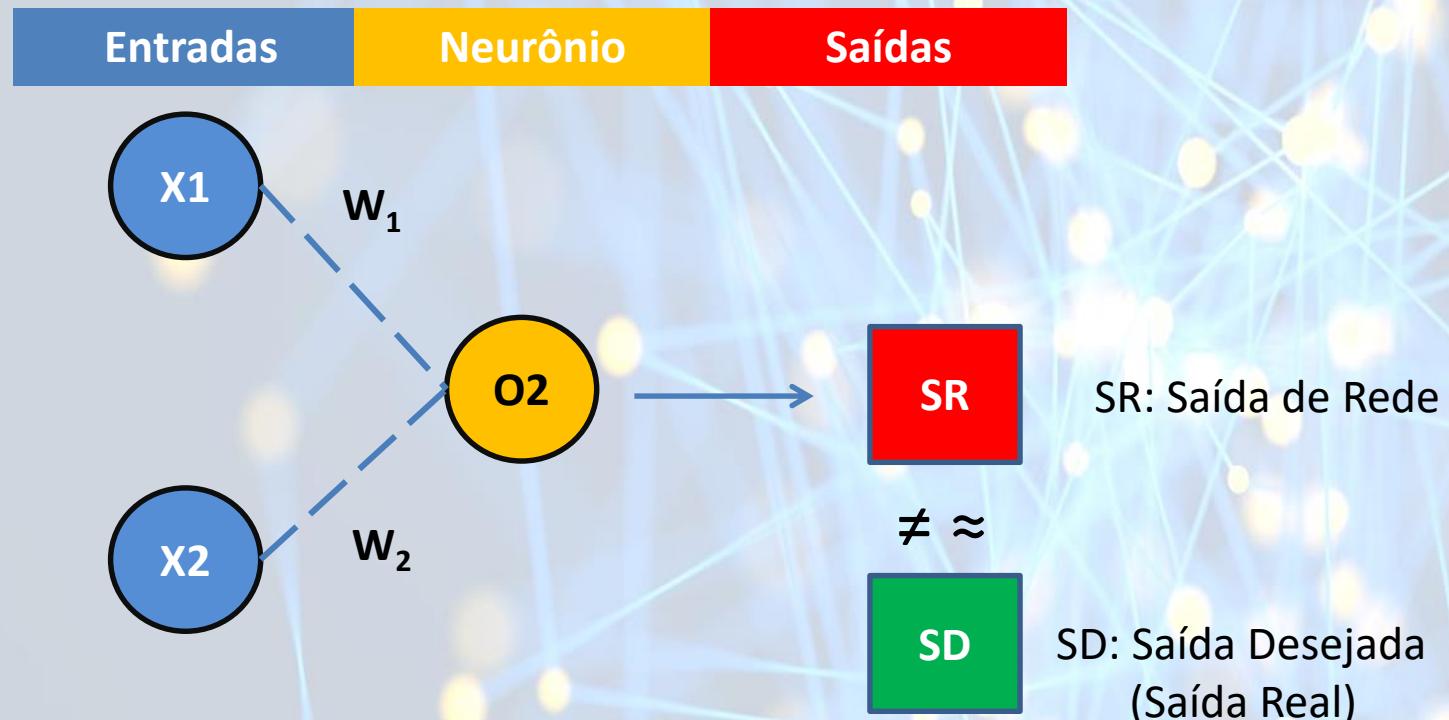
Histórico

- 1943 McCulloch-Pitts Neurônio Booleano
- 1957 Rosenblatt Perceptron
- 1960 Widrow-Hoff ADALINE
- 1969 Minsky-Papert *Perceptrons*
- 1986 Rumelhart, Hinton & Williams (MIT)
Backpropagation p/ Perceptron Multicamadas (MLP)



Redes Neurais Artificiais

Modelo de McCulloch e Pitts (1943)

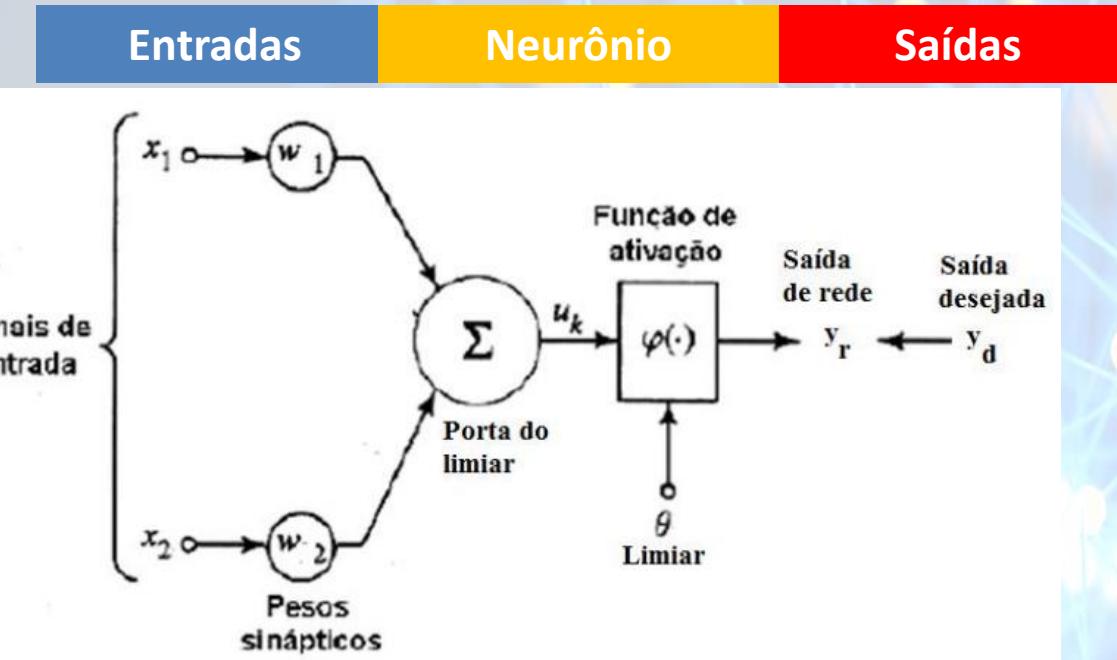


Sinais de Entrada: Medidas advindas do meio externo (variáveis de um problema);

Neurônios: Unidade básica de processamento;

Saídas: Respostas emitidas pelos neurônios.

Modelo de McCulloch e Pitts (1943)



Sinais de Entrada (X_i): Medidas advindas do meio externo (variáveis de um problema)

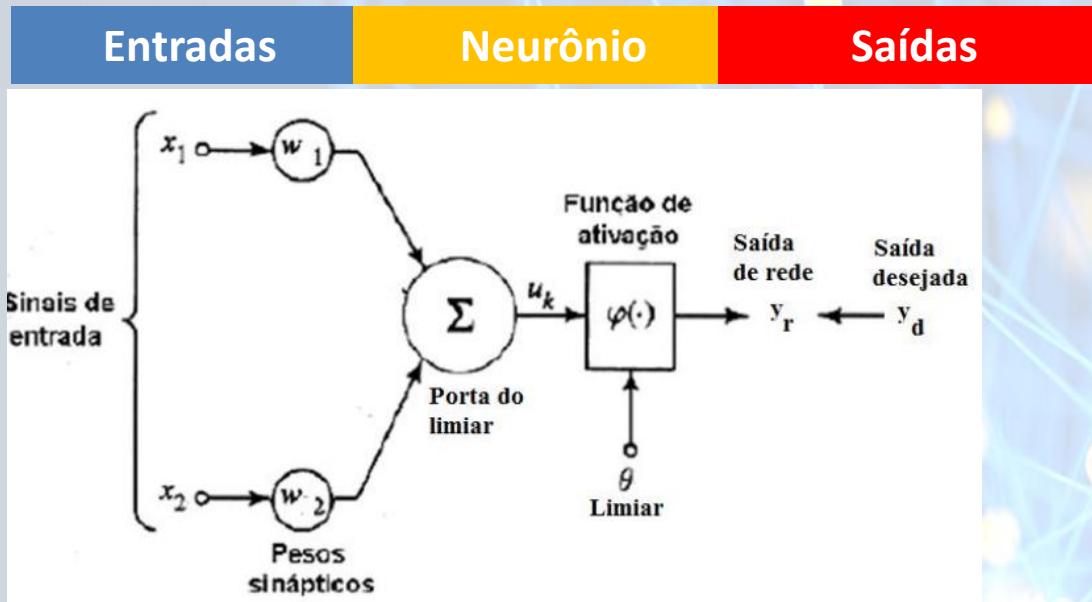
$$X_i = [X_1, X_2]$$

Pesos (W_i): São os valores que irão ponderar cada uma das variáveis de entrada;

$$W_i = [W_1, W_2]$$

Porta do limiar (L): função para agregar todos os sinais de entrada que foram ponderados;

$$L = \sum_{i=1}^p w_i x_i = W_1 X_1 + W_2 X_2$$



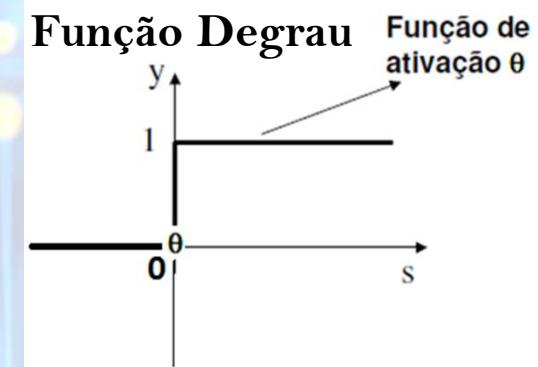
Porta do Limiar

$$L = \sum_{i=1}^p w_i x_i = w_1 x_1 + w_2 x_2$$

Limiar de ativação (θ): Variável (valor) que regula a resposta do neurônio (“Patamar”), segundo uma **função de ativação**.

Função de Ativação ($f(L)$)

- $Y = f(L)$
- 1, se $L \geq \theta$ → Limiar
 - 0, se $L < \theta$



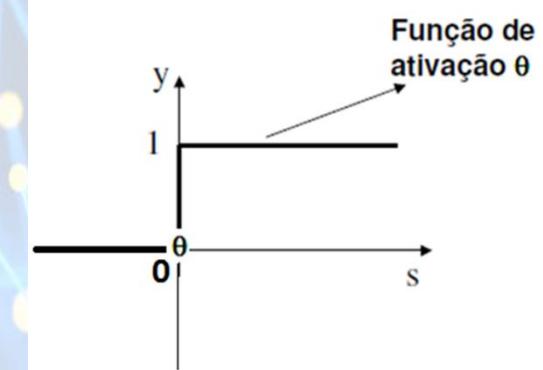
Modelo de McCulloch e Pitts

Porta do Limiar (L)

$$L = \sum_{i=1}^p w_i x_i = w_1 x_1 + w_2 x_2$$

Função de Ativação ($f(L)$)

- $Y = f(L)$
- 1, se $L \geq \theta$ → Limiar
 - 0, se $L < \theta$



Função de ativação (f): Função que regula a saída do neurônio, conforme o **limiar de ativação**.

$$L = W_1 X_1 + W_2 X_2 \geq ou < \theta$$

$$u = W_1 X_1 + W_2 X_2 - \theta \geq ou < 0$$

Potencial de ativação (u): Resultado da diferença entre os valores de **porta do limiar** e **limiar de ativação**.

$$u = \sum_{i=1}^p w_i x_i - \theta$$

Modelo de McCulloch e Pitts

Potencial de ativação (u): Resultado da diferença entre os valores da **porta do limiar** e **limiar de ativação**.

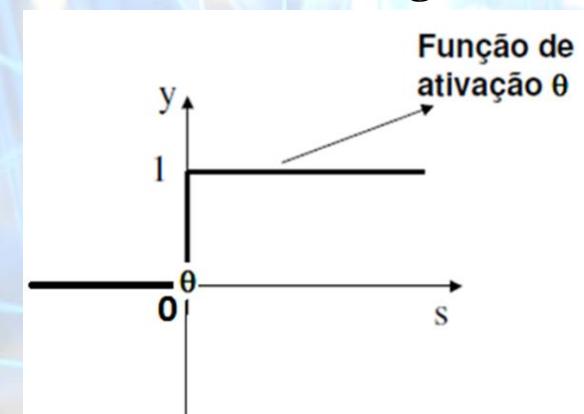
$$u = \sum_i^p W_i X_i - \Theta$$

Função de ativação ($f(u)$): Função que regula a saída do neurônio, conforme o **limiar de ativação**.

Função de Ativação ($f(u)$)

- $$Y = f(u) \quad \left\{ \begin{array}{l} \bullet \quad 1, \text{ se } u \geq 0 \\ \bullet \quad 0, \text{ se } u < 0 \end{array} \right.$$

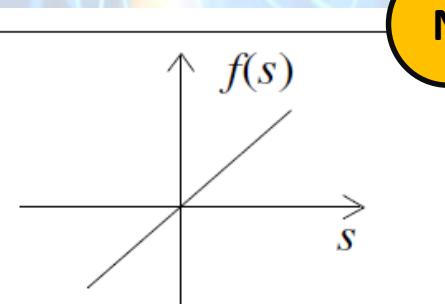
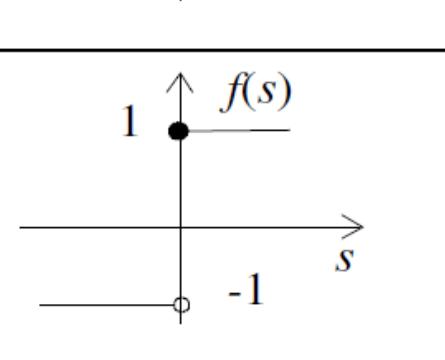
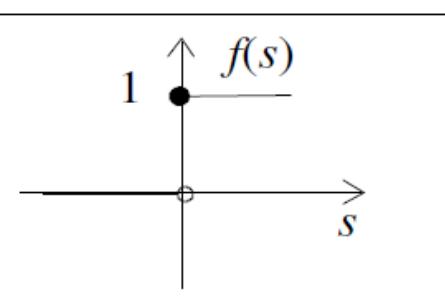
Função Degrau



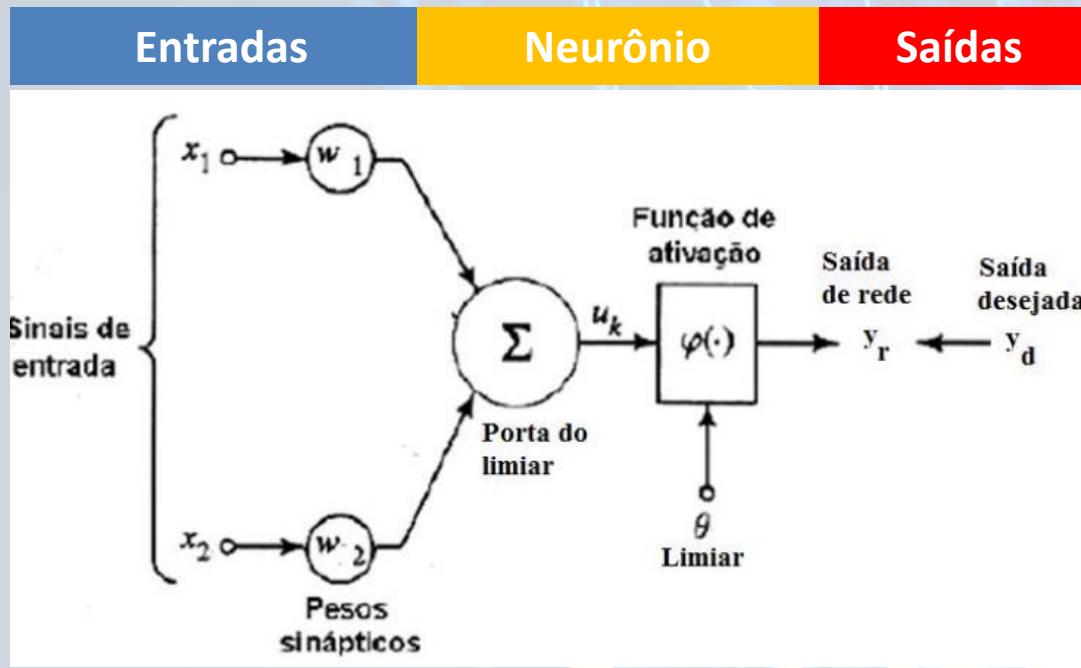
Sinal de Saída (y): Valor final produzido pelo neurônio em relação a um determinado conjunto de sinais de entrada.

Função de Ativação

A função de ativação incorporada numa estrutura de rede dá a cada neurônio que a constitui a capacidade de extrair informações e a potencialidade do aprendizado da rede.

Linear	$f(s) = s$ $u = s = \sum_i^p W_i X_i - \theta$	
Sinal	$f(s) = \begin{cases} +1 & \text{se } s \geq 0 \\ -1 & \text{se } s < 0 \end{cases}$	
Degrau	$f(s) = \begin{cases} +1 & \text{se } s \geq 0 \\ 0 & \text{se } s < 0 \end{cases}$	





Sinais de Entrada (X_i): Medidas advindas do meio externo (variáveis de um problema)

Pesos (W_i): São os valores que irão ponderar cada uma das variáveis de entrada;

Porta do limiar (L): função para agregar todos os sinais de entrada que foram ponderados;

Limiar de ativação (θ): Variável (valor) que regula a resposta do neurônio (“Patamar”).

Potencial de ativação (u): Resultado da diferença entre os valores de **porta do limiar** e **limiar de ativação**.

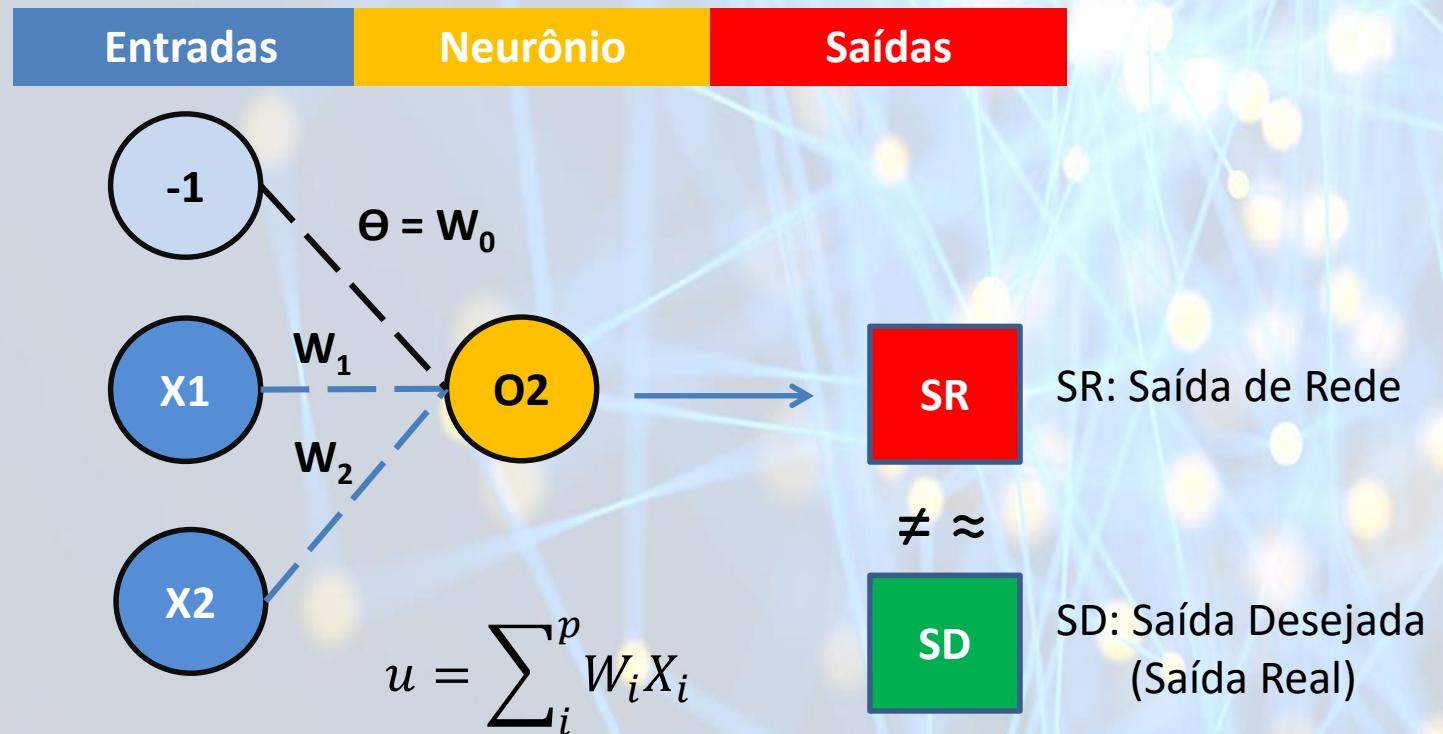
Função de ativação ($f(u)$): Função que limita a saída do neurônio, conforme o **limiar de ativação**.

Sinal de Saída (y): Valor final produzido pelo neurônio em relação a um determinado conjunto de sinais de entrada.

Modelo de McCulloch e Pitts (1943)

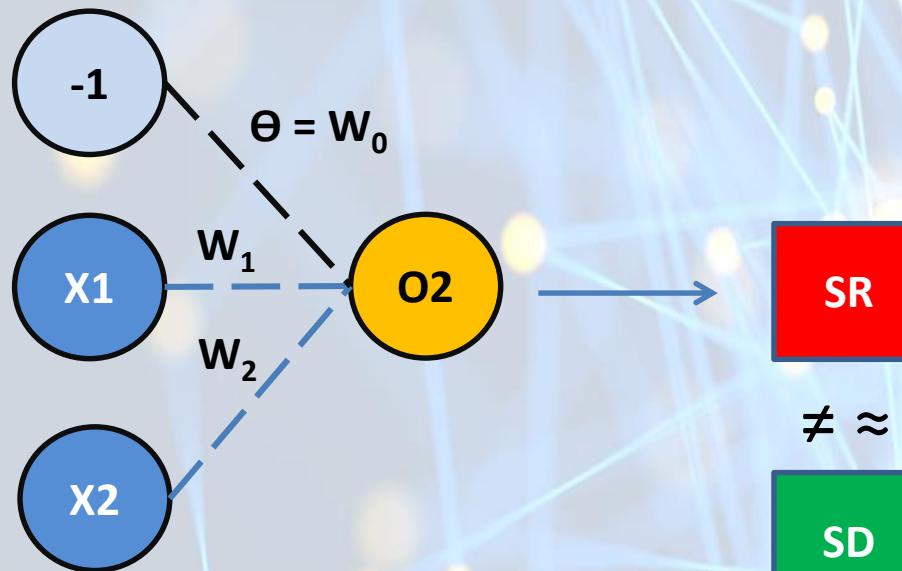
$$u = W_1X_1 + W_2X_2 - 1(\Theta)$$

$$u = \sum_i^p W_i X_i - \Theta$$





COMO ESTIMAR OS PESOS?



SR: Saída de Rede

SD

SD: Saída Desejada
(Saída Real)

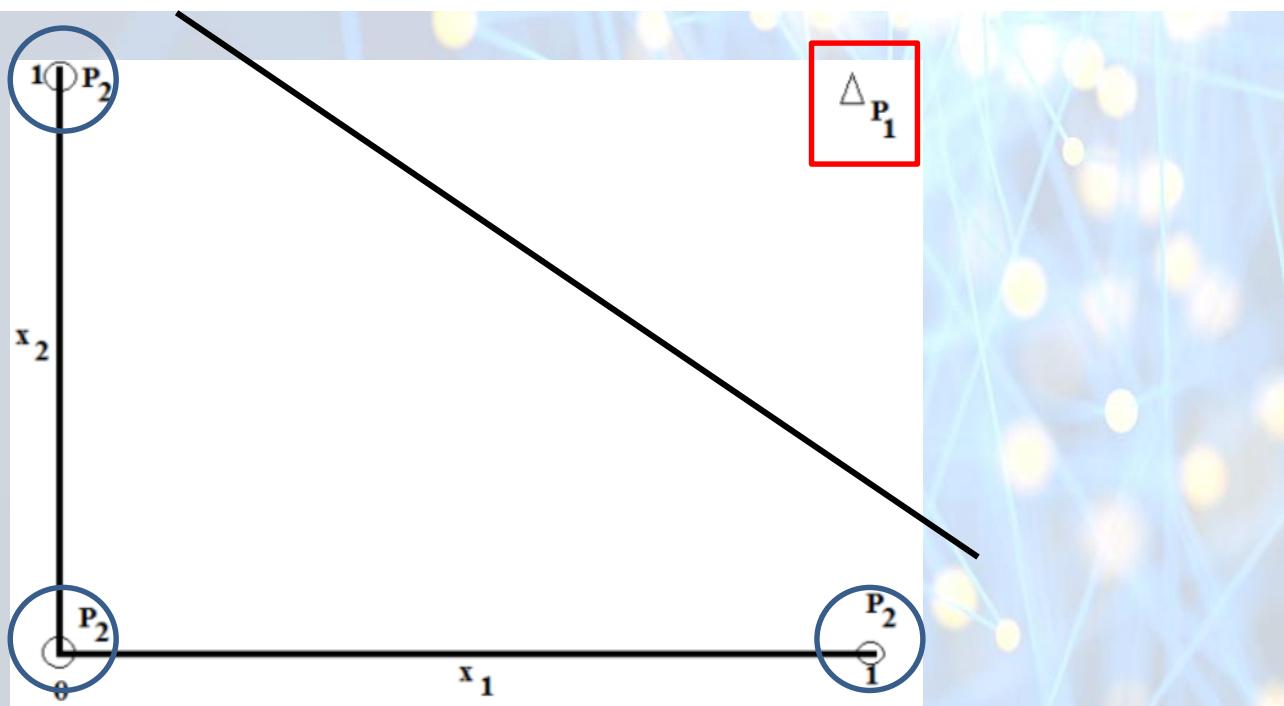
Modelo de McCulloch e Pitts (1943)

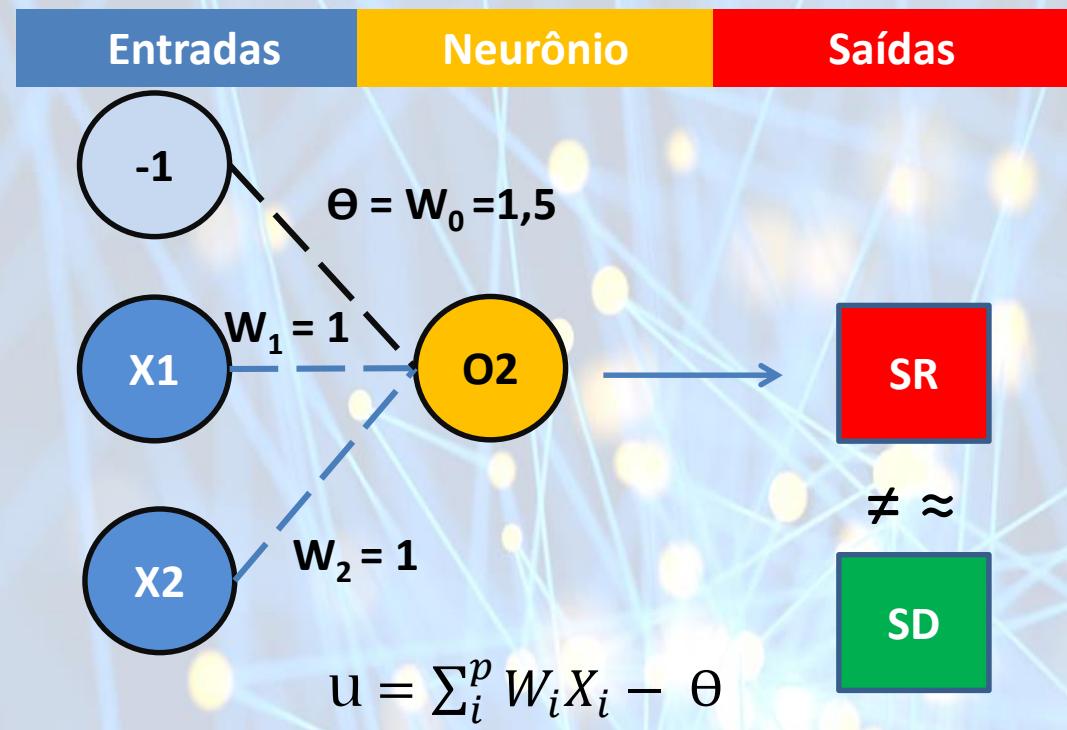
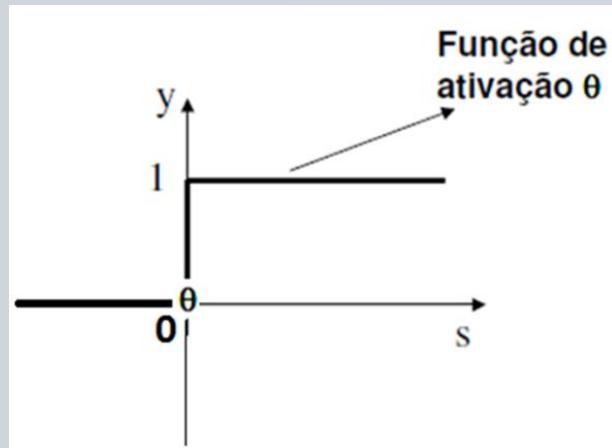
Treinamento: Etapa que consiste em estimar pesos

Época (iteração): Cada vez em que são estimados novos pesos



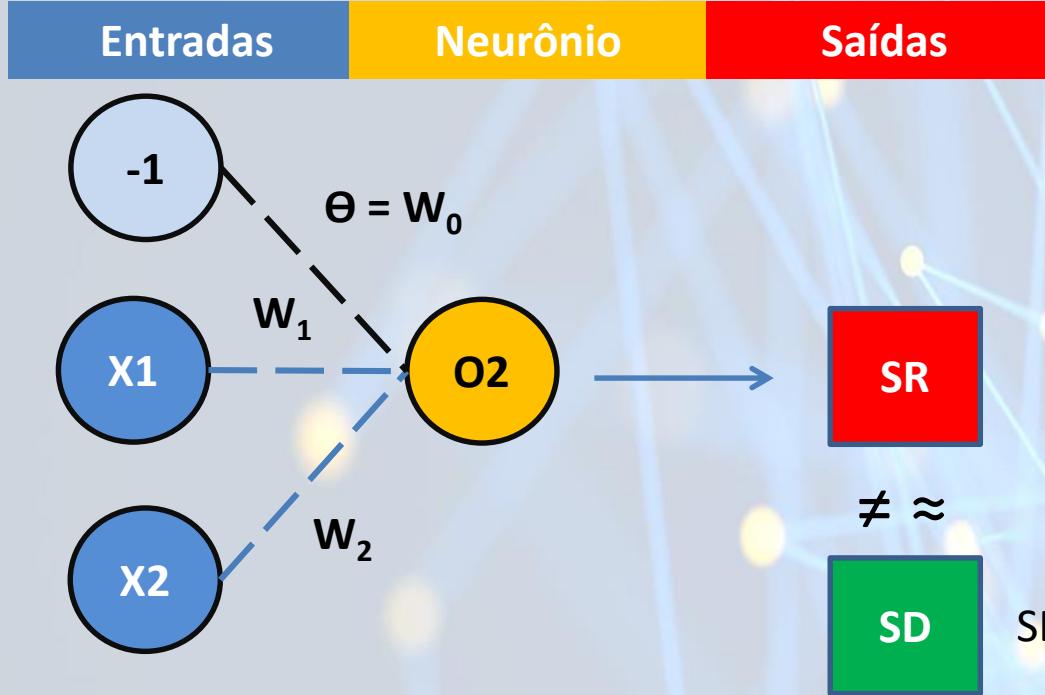
Entradas	Obs1	Obs2	Obs3	Obs4
x_1	1	1	0	0
x_2	1	0	1	0
Saída desejada				
y_d	1 (P_1)	0 (P_2)	0 (P_2)	0 (P_2)





x_1	1	1	0	0	Entrada
x_2	1	0	1	0	Entrada
$L = \sum w_i x_i$	2	1	1	0	Porta do limiar
y_r	1	0	0	0	Saída de Rede
y_d	1	0	0	0	Saída desejada
$\varepsilon = y - y_r$	0	0	0	0	Erros

Modelo Perceptron (1958)



Incorporação de Regras de Aprendizado

Processos iterativos

- Pesos iniciais - aleatórios

$$\vec{w}(t+1) = \vec{w}(t) + \eta \cdot e \cdot \vec{x}$$

$$\vec{X} = [-1; X_1; X_2 \dots X_n]$$

$$\vec{W} = [W_0; W_1; W_2 \dots W_n]$$

$$e = y_d - y_r$$

Taxa de aprendizado: Proporção do erro que contribuirá para o ajuste dos pesos.

Modelo Perceptron (1958)

1. Pesos Aleatórios $\vec{w}(t)$

2. Uso dos Pesos Ajustados

$$\vec{w}(t + 1)$$

- Processamento das Entradas

- Potencial de ativação (u)

$$u = \sum_i^p W_i X_i - \Theta$$

Entradas

Execução

Erro
 $(y_d - y_r)$

Saída de
Rede (y_r)

- Ajuste dos pesos

$$\vec{w}(t + 1) = \vec{w}(t) + \eta \cdot e \cdot \vec{x}$$

- Obtenção da saída de Rede ($y_r = f(u)$)

Erro: $e = y_d - y_r$

Treinamento: Etapa que consiste em estimar os pesos

Época (iteração): Cada vez em que são estimados novos pesos

Medidas de Performance

Erro Quadrático Médio (EQM)

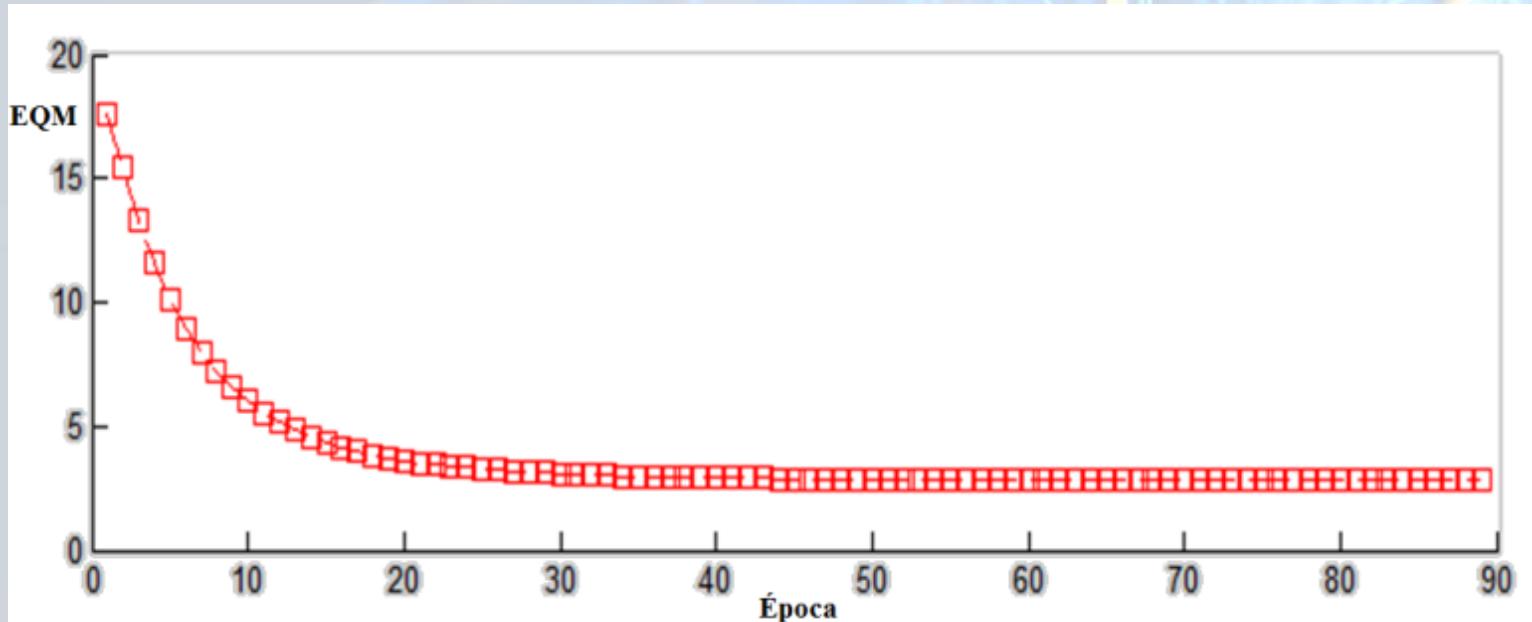
$$EQM = \frac{1}{p} \sum_{i=1}^p (y_{d(i)} - y_{r(i)})^2$$

Taxa de Erro Aparente (TEA)

$$TEA = \frac{\sum \text{erros}}{\text{Total}}$$

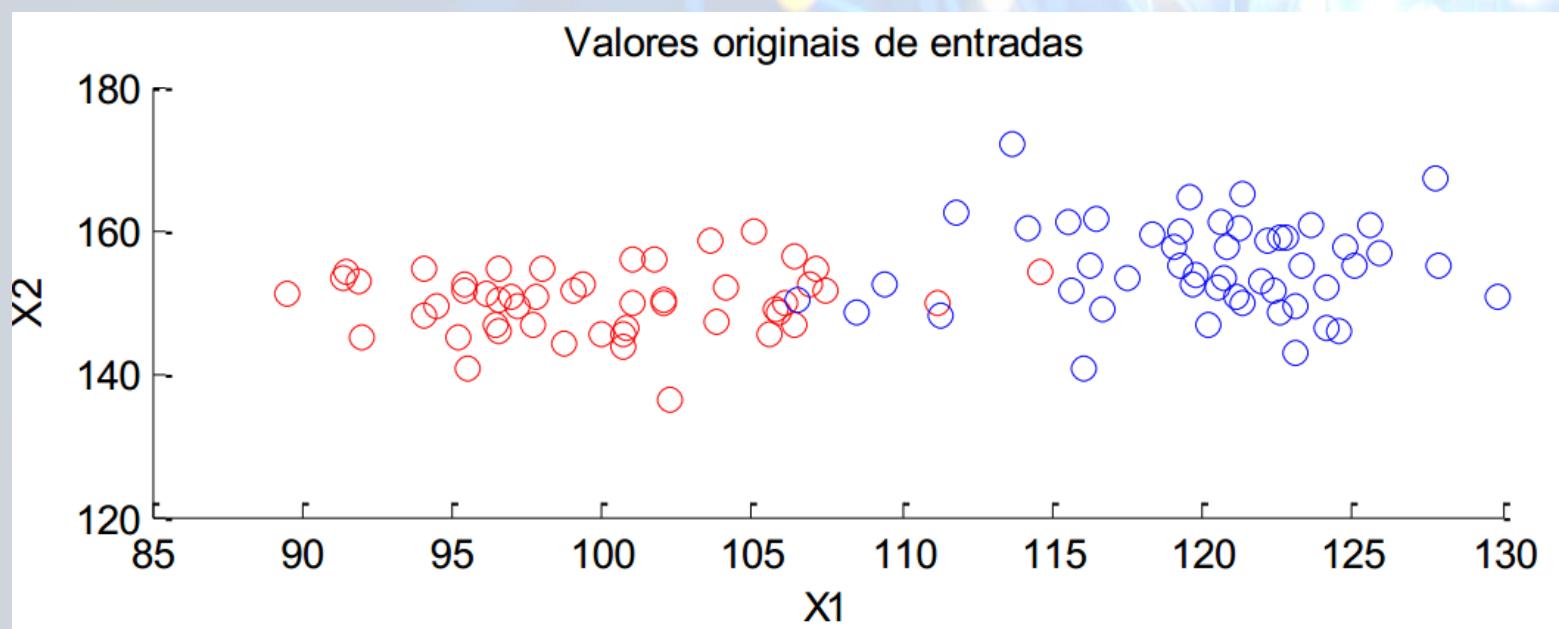
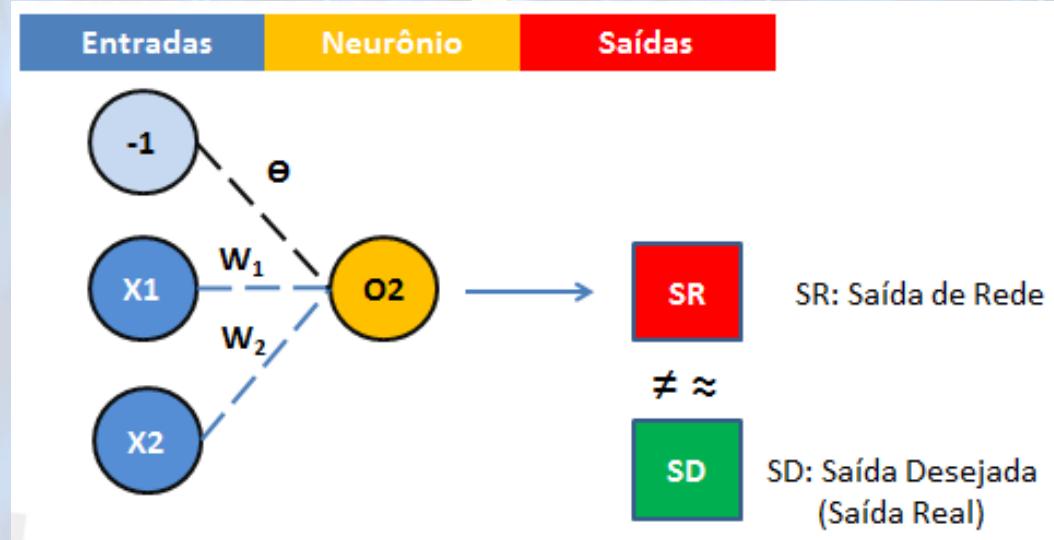
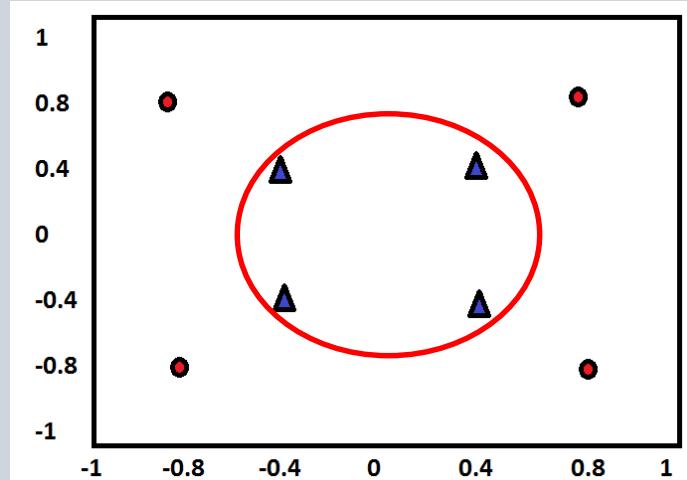
Coeficiente de determinação (R^2)

$$R^2 = [\text{corr}(Y_d, Y_r)]^2$$



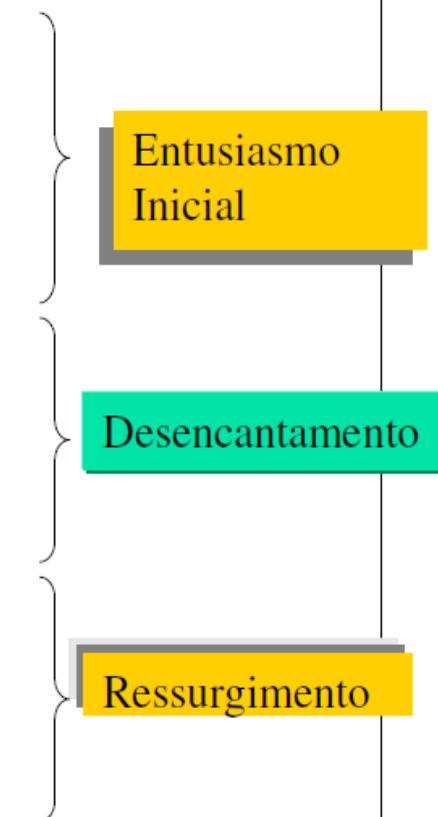
Desempenho de uma rede neural, quantificado pelo erro quadrático médio (EQM), após 89 iterações.

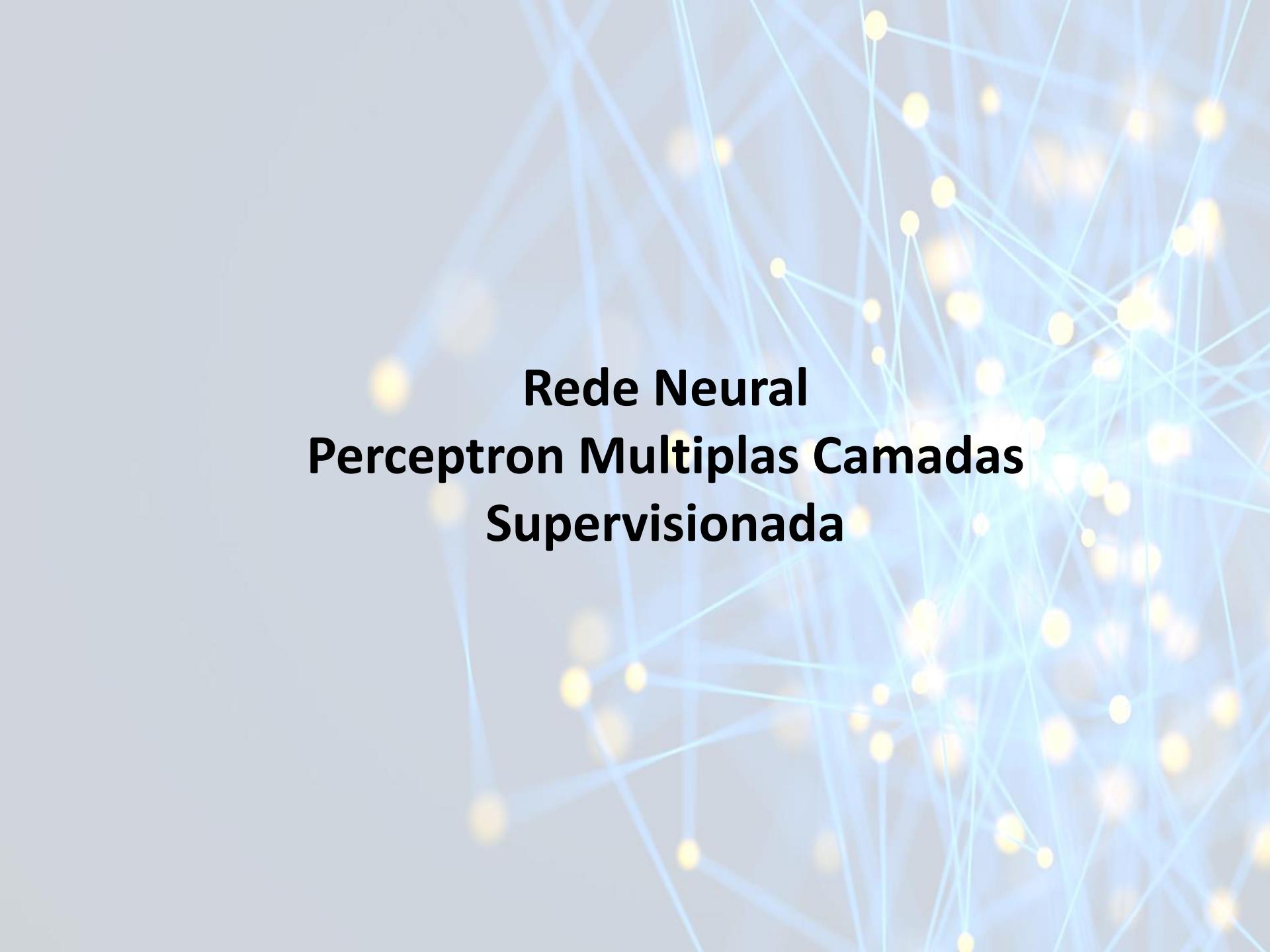
Exemplos



Histórico

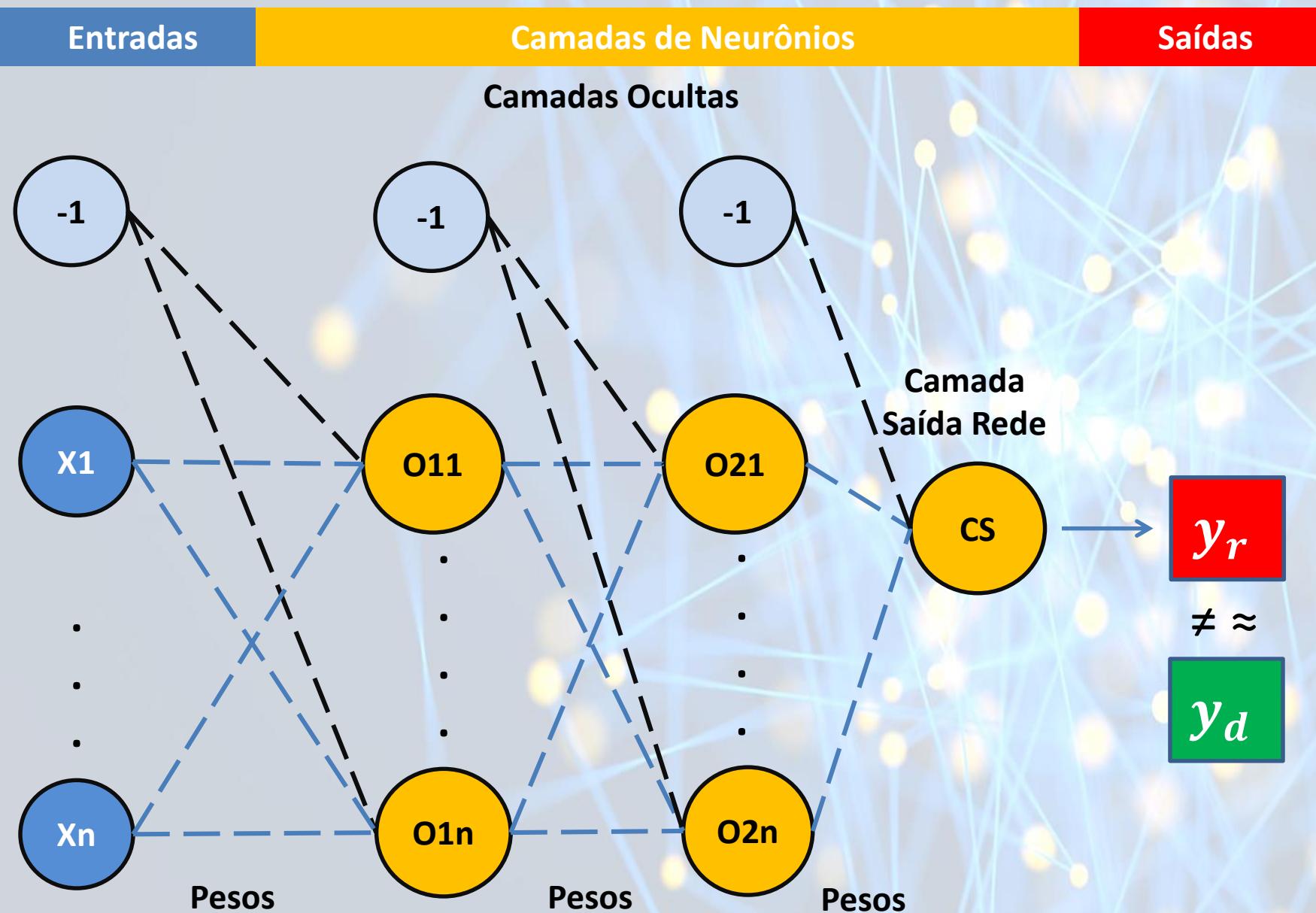
- 1943 McCulloch-Pitts Neurônio Booleano
- 1957 Rosenblatt Perceptron
- 1960 Widrow-Hoff ADALINE
- 1969 Minsky-Papert *Perceptrons*
- 1986 Rumelhart, Hinton & Williams (MIT)
Backpropagation p/ Perceptron Multicamadas (MLP)



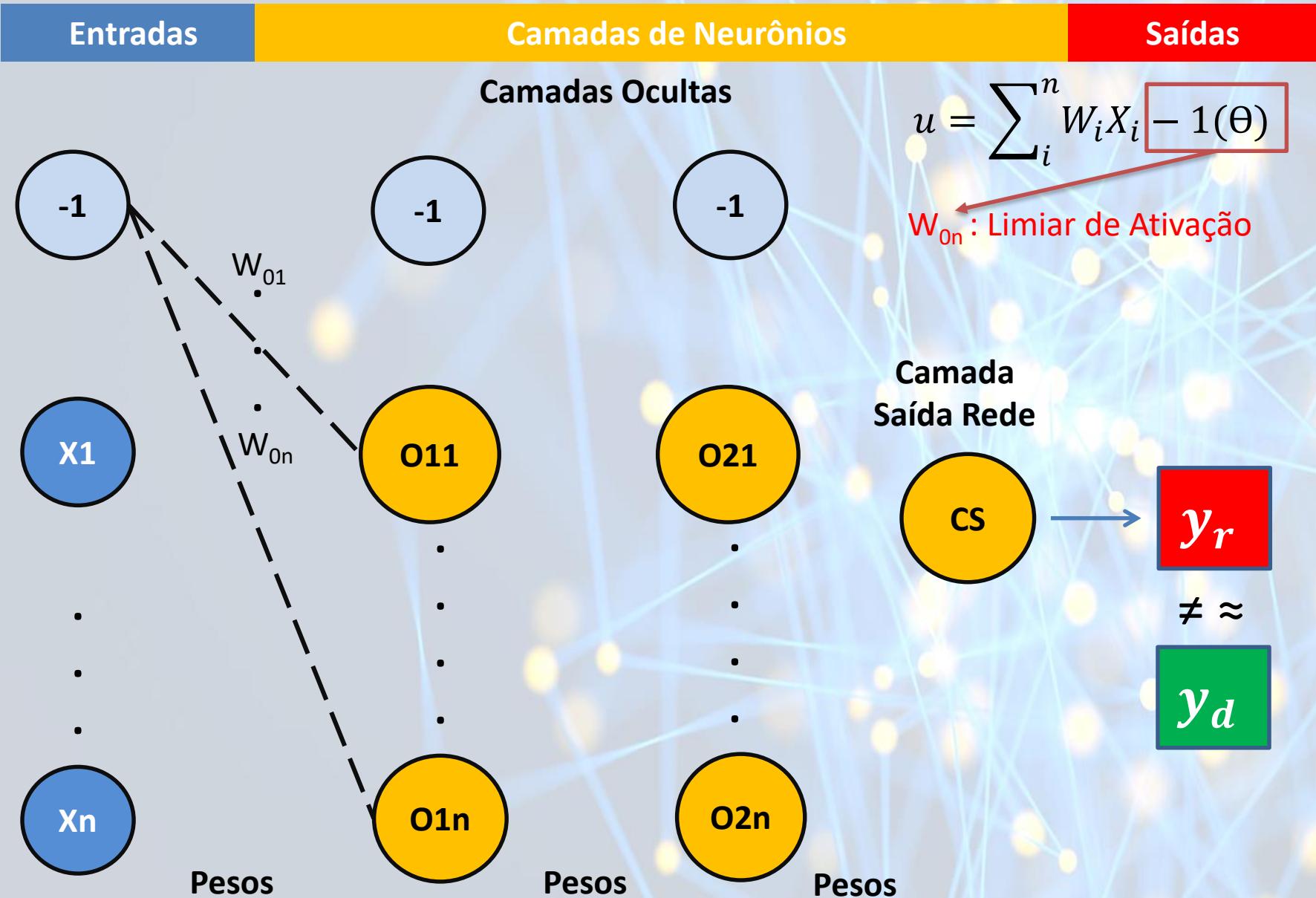


Rede Neural Perceptron Multiplas Camadas Supervisionada

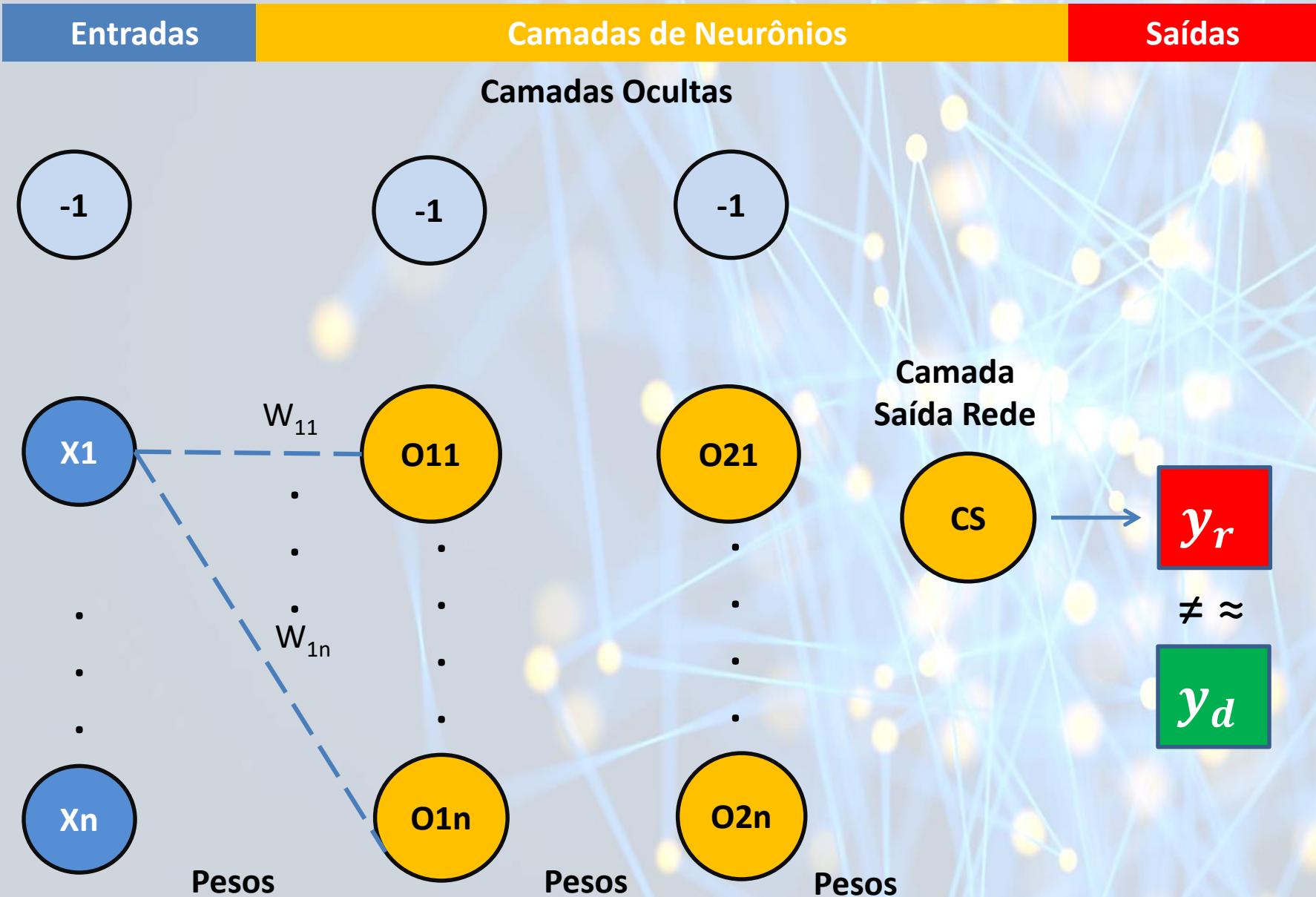
Multilayer Perceptron (MLP)



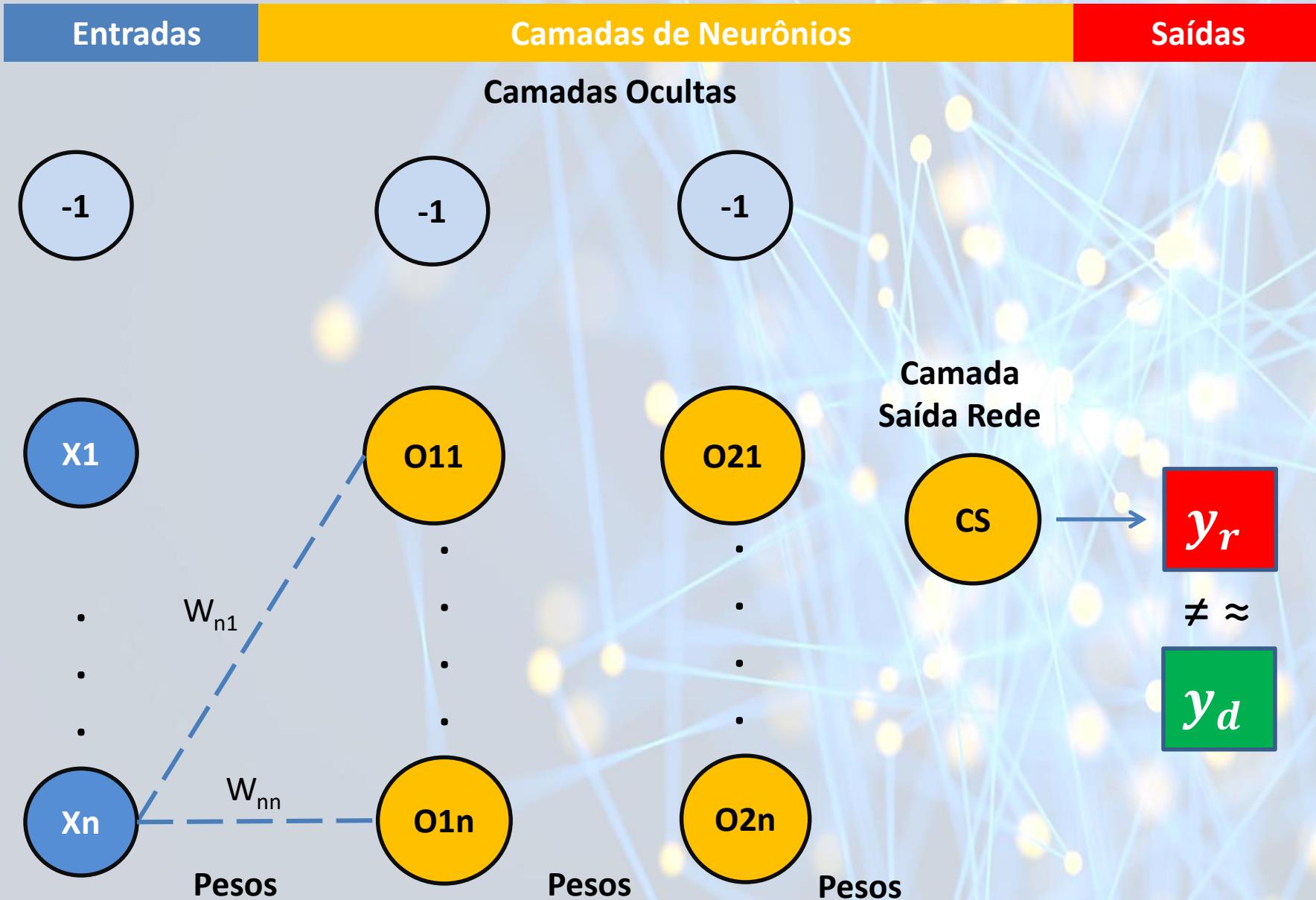
Multilayer Perceptron (MLP)



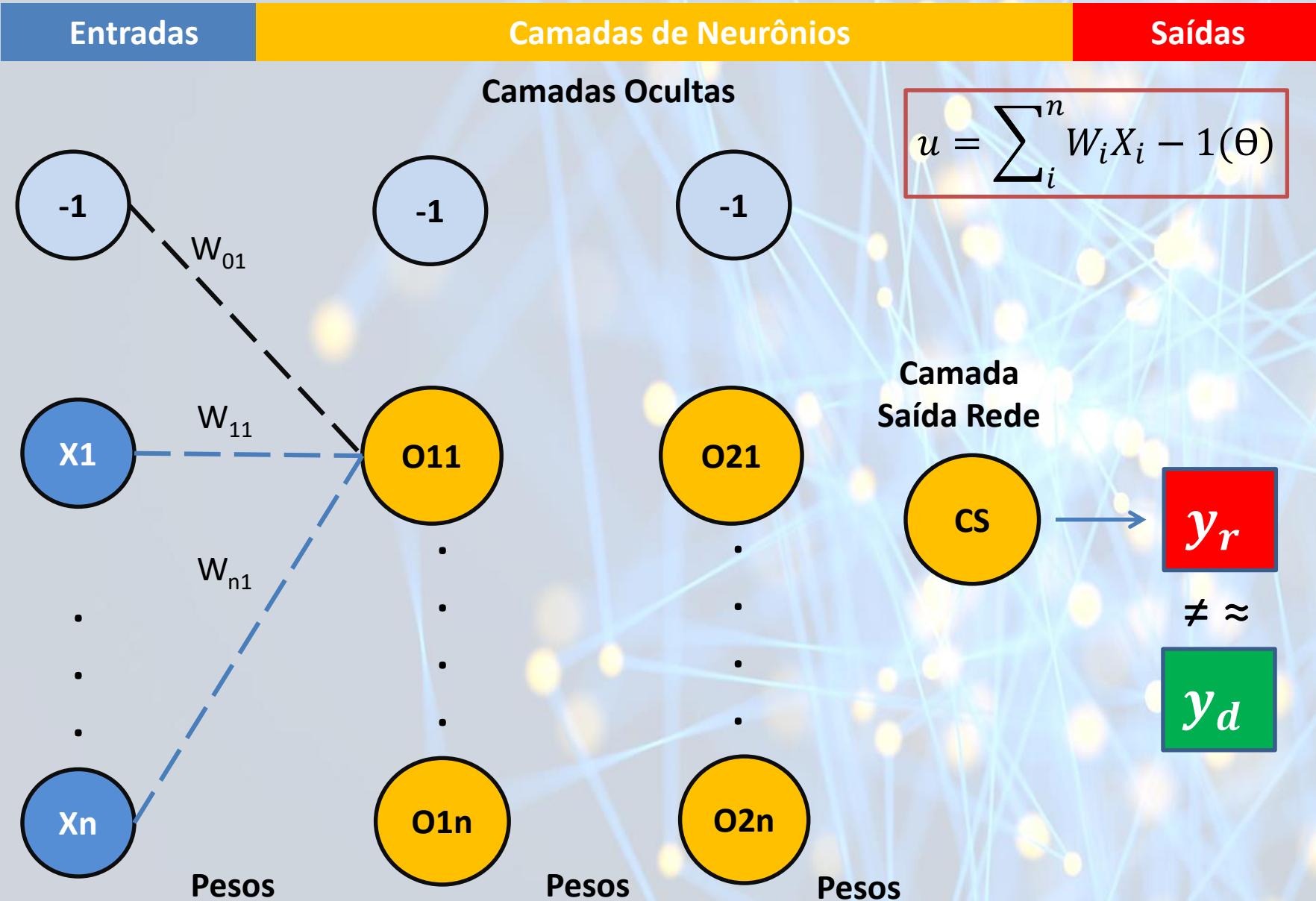
Multilayer Perceptron (MLP)



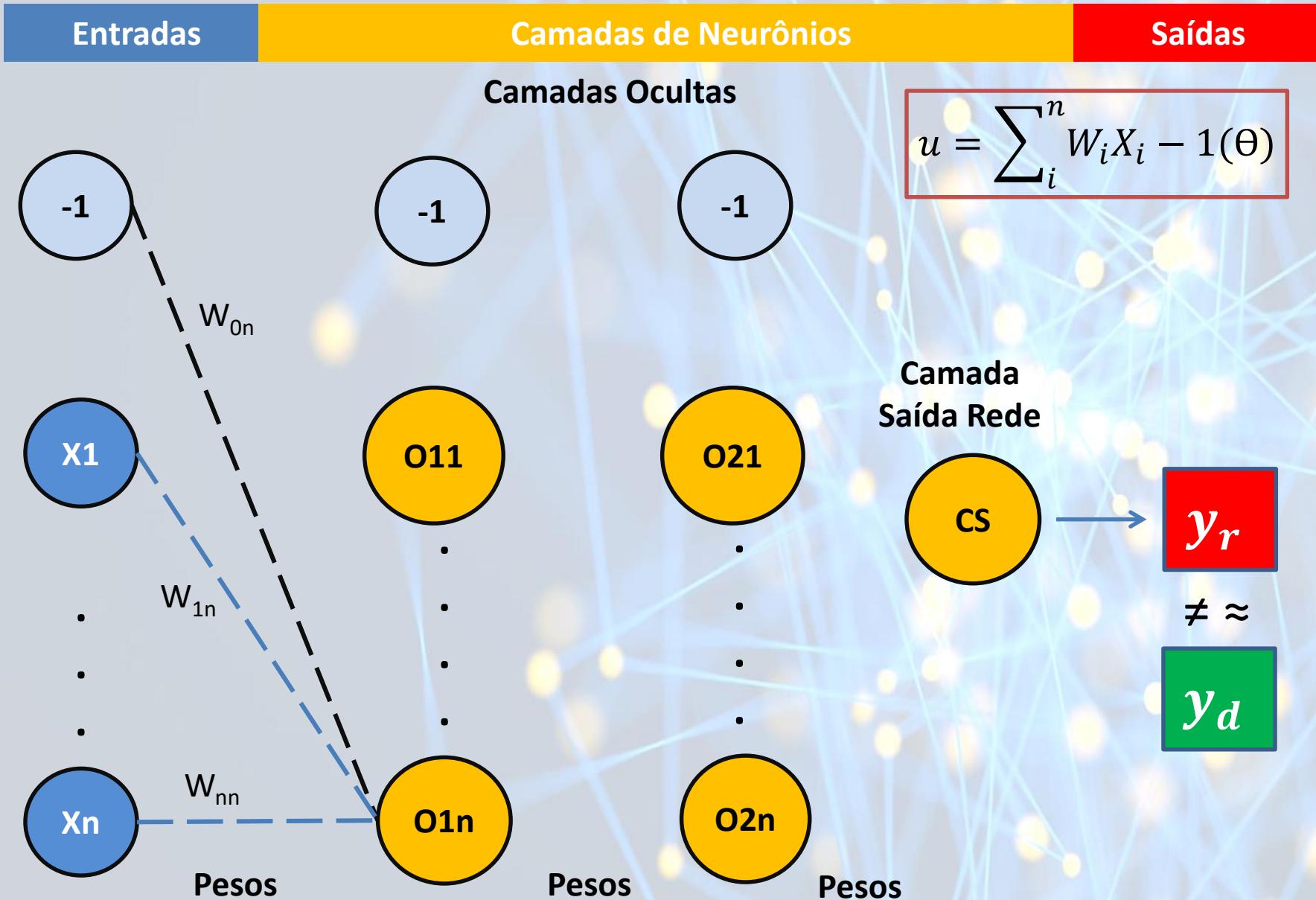
Multilayer Perceptron (MLP)



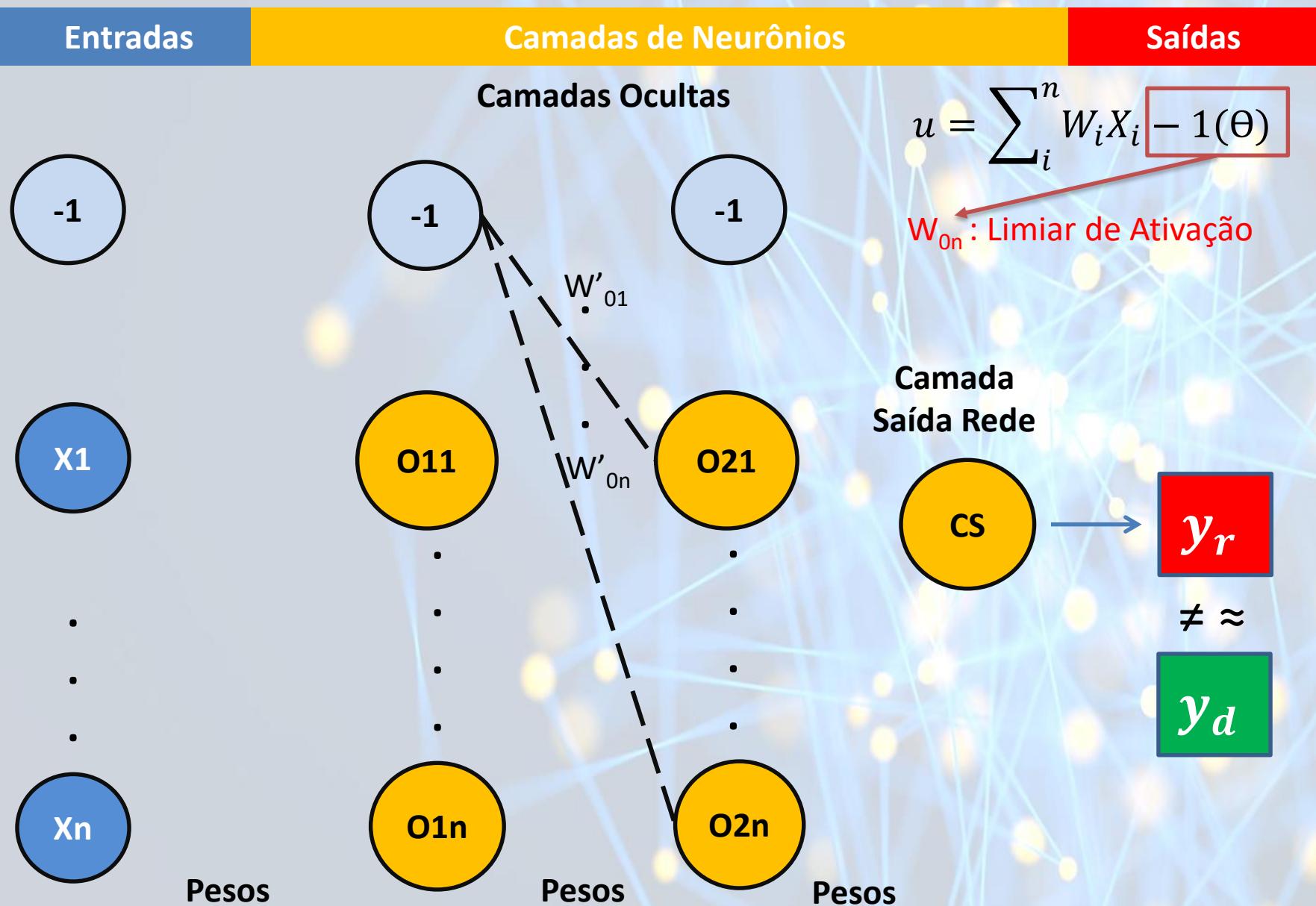
Multilayer Perceptron (MLP)



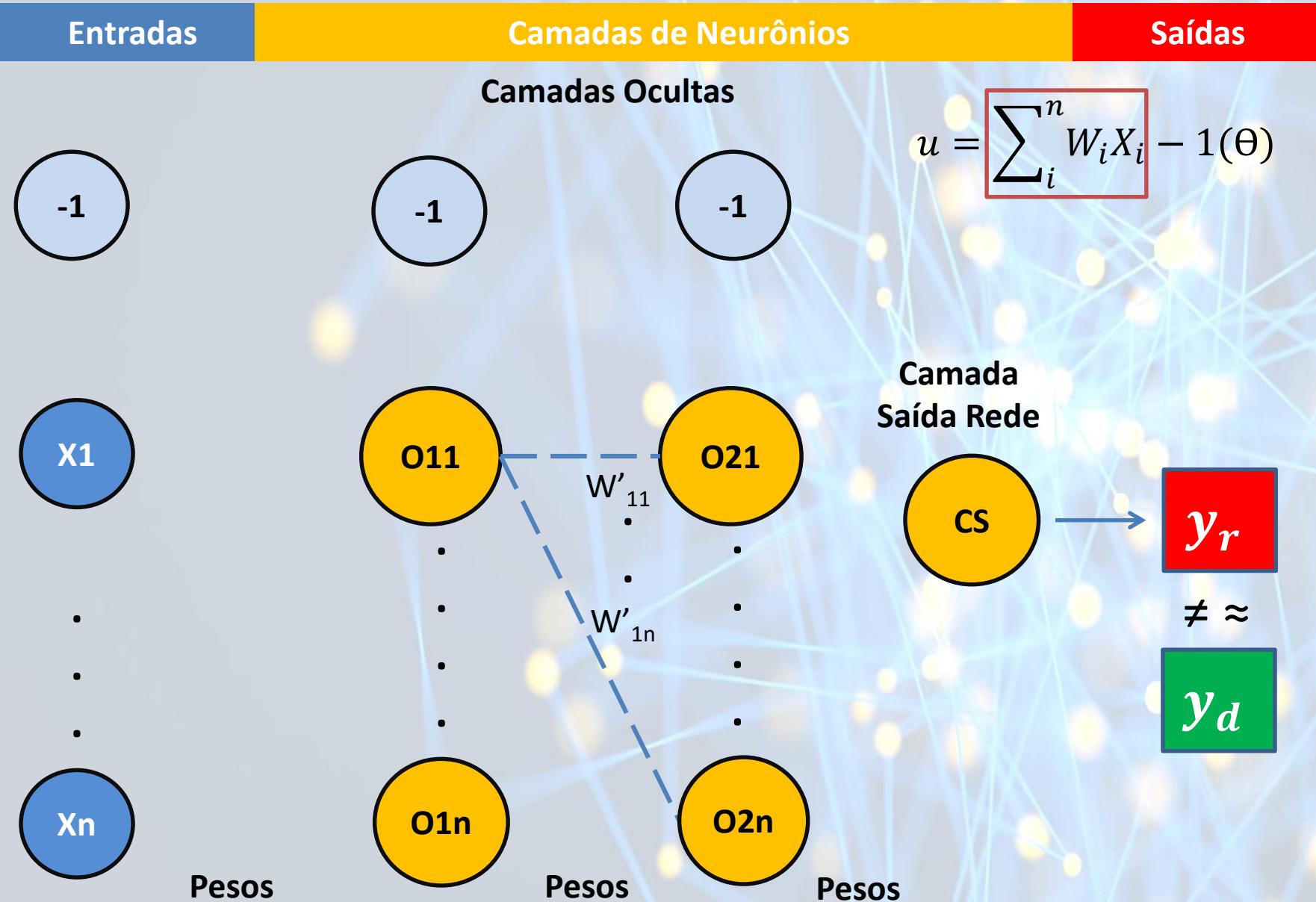
Multilayer Perceptron (MLP)



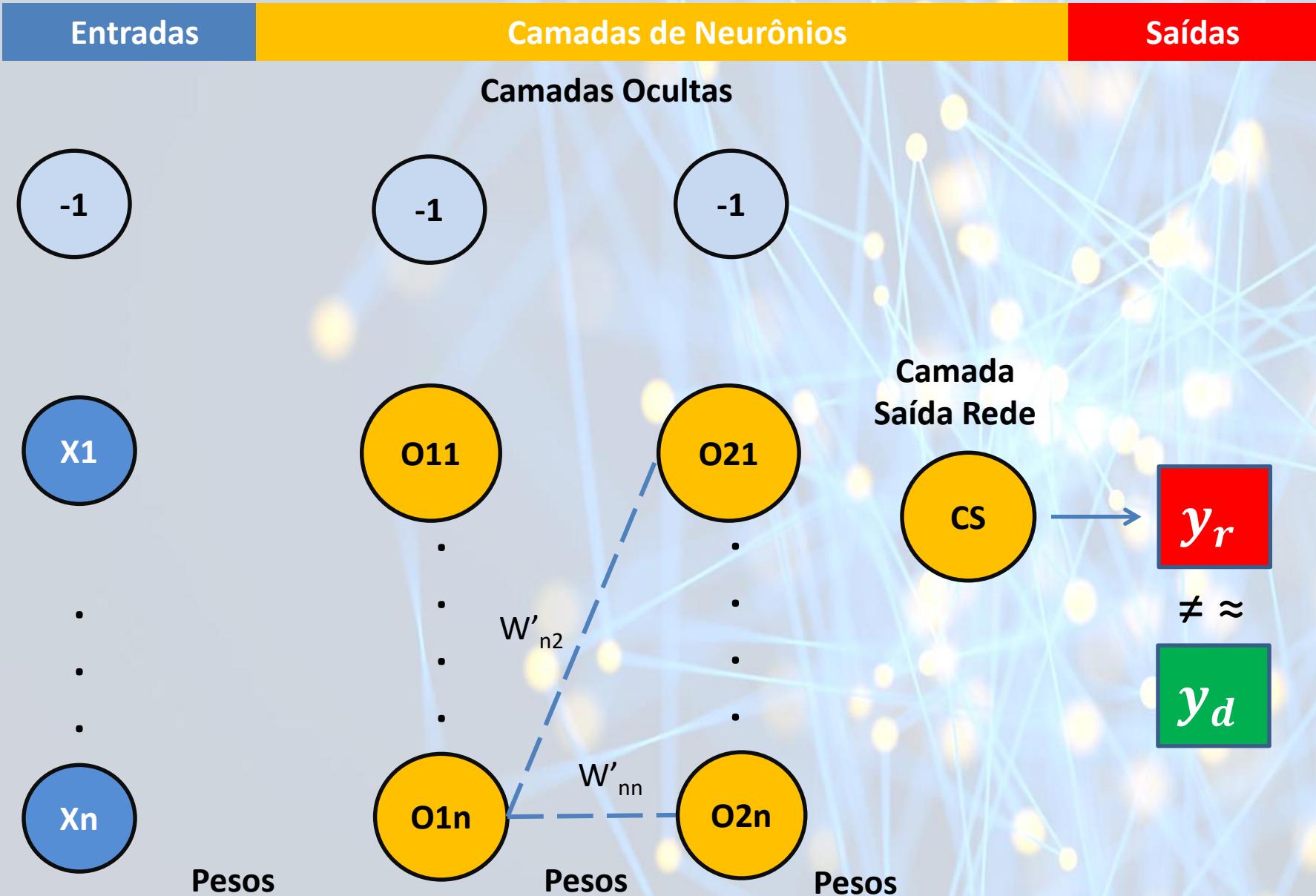
Multilayer Perceptron (MLP)



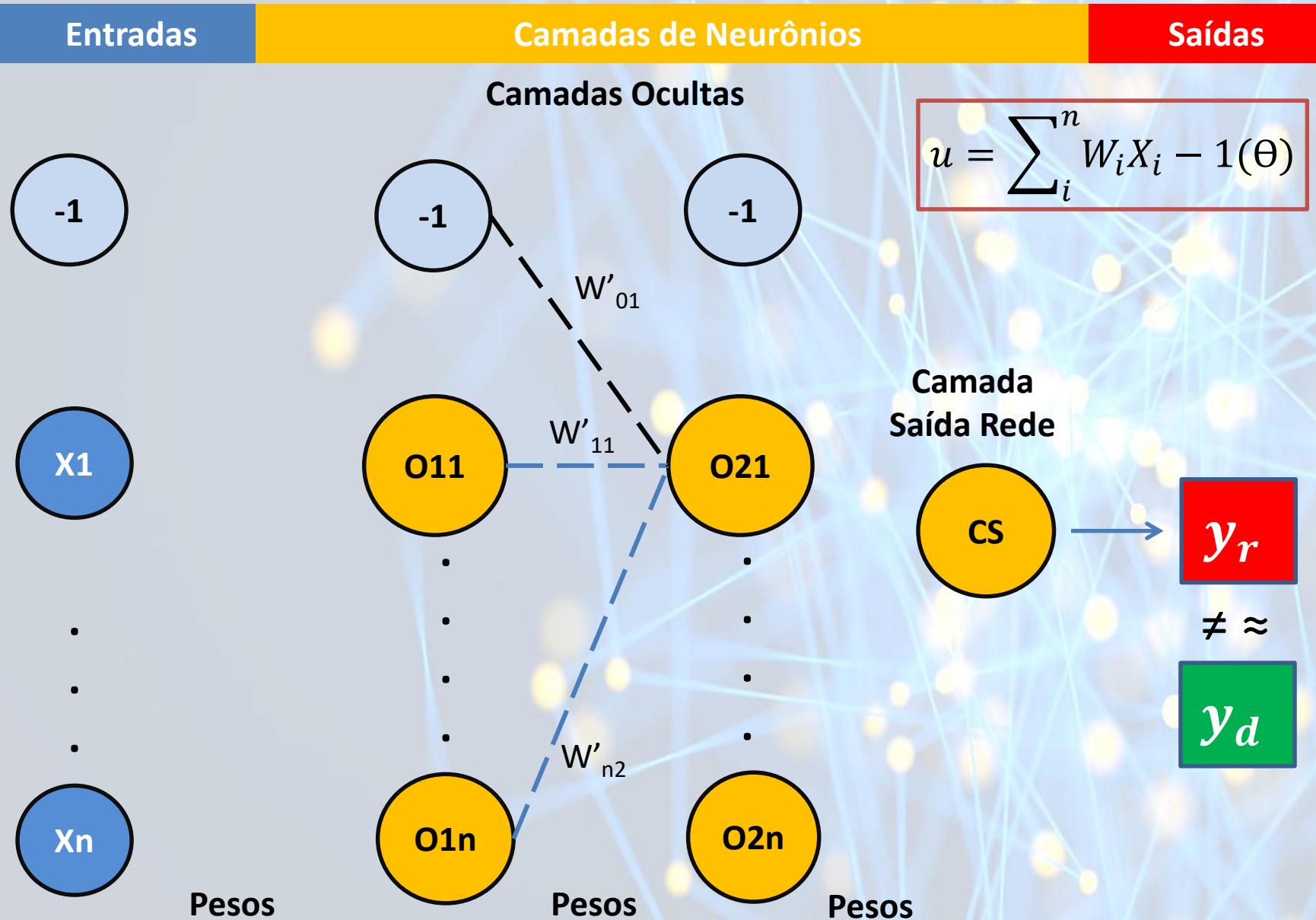
Multilayer Perceptron (MLP)



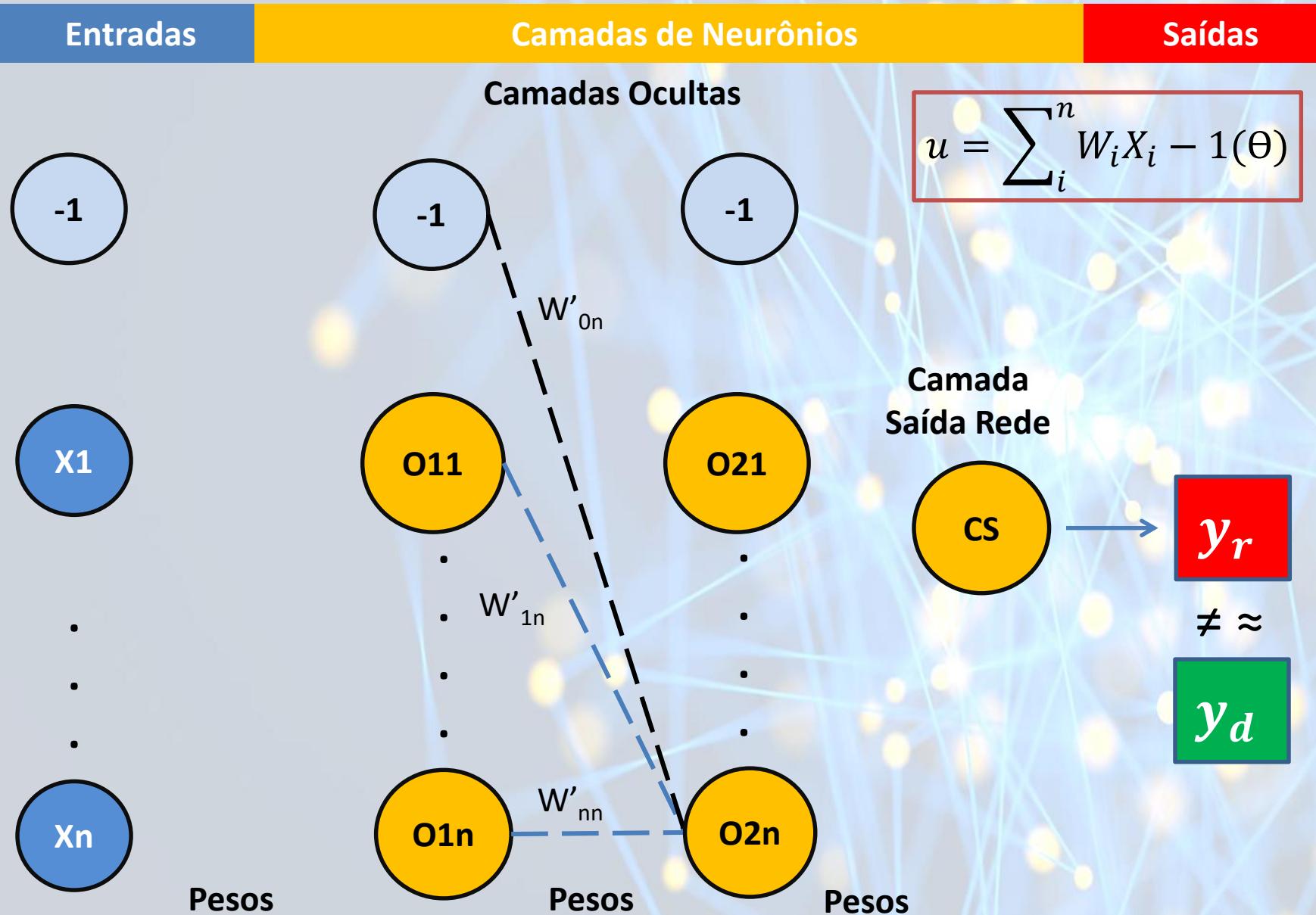
Multilayer Perceptron (MLP)



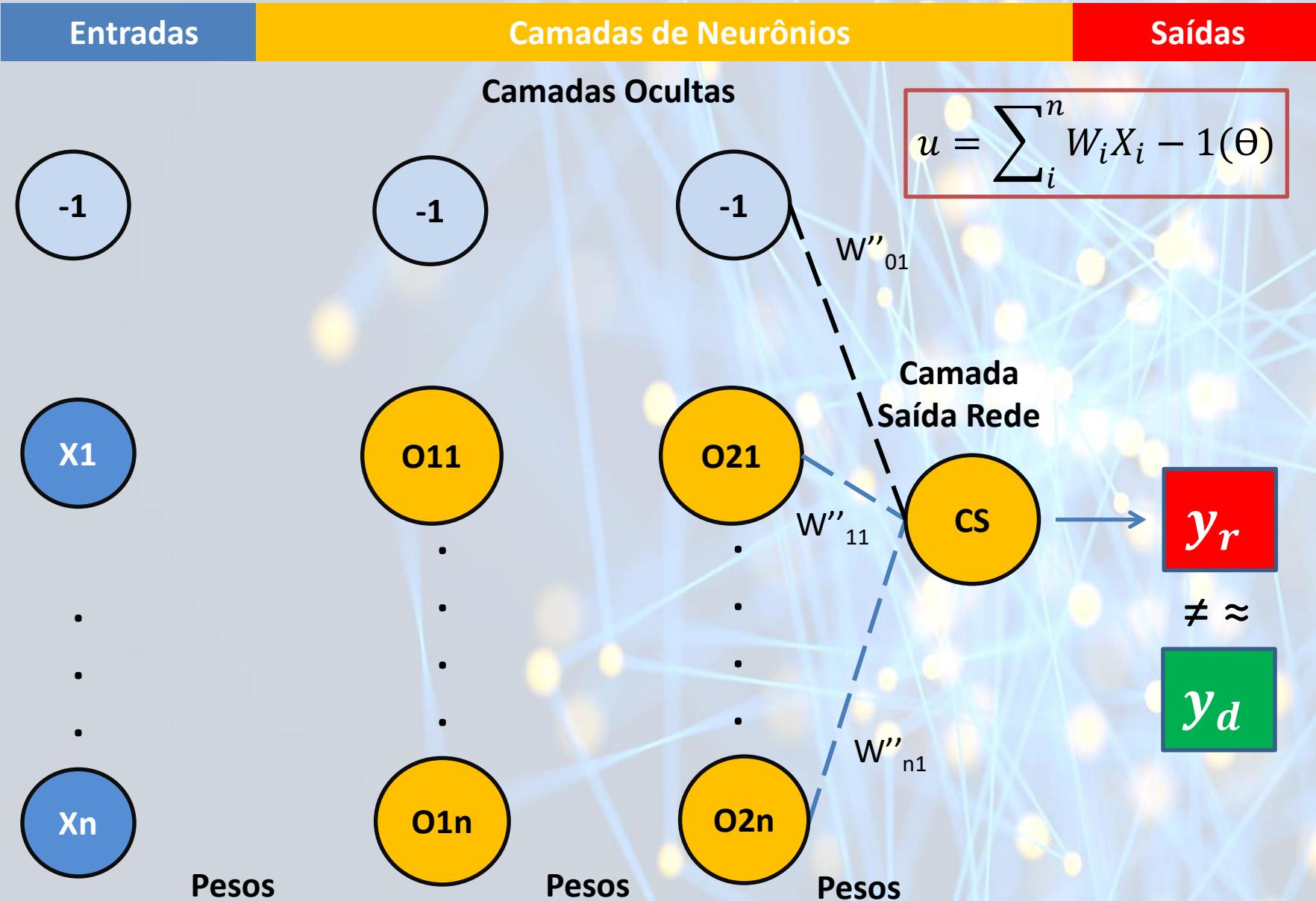
Multilayer Perceptron (MLP)



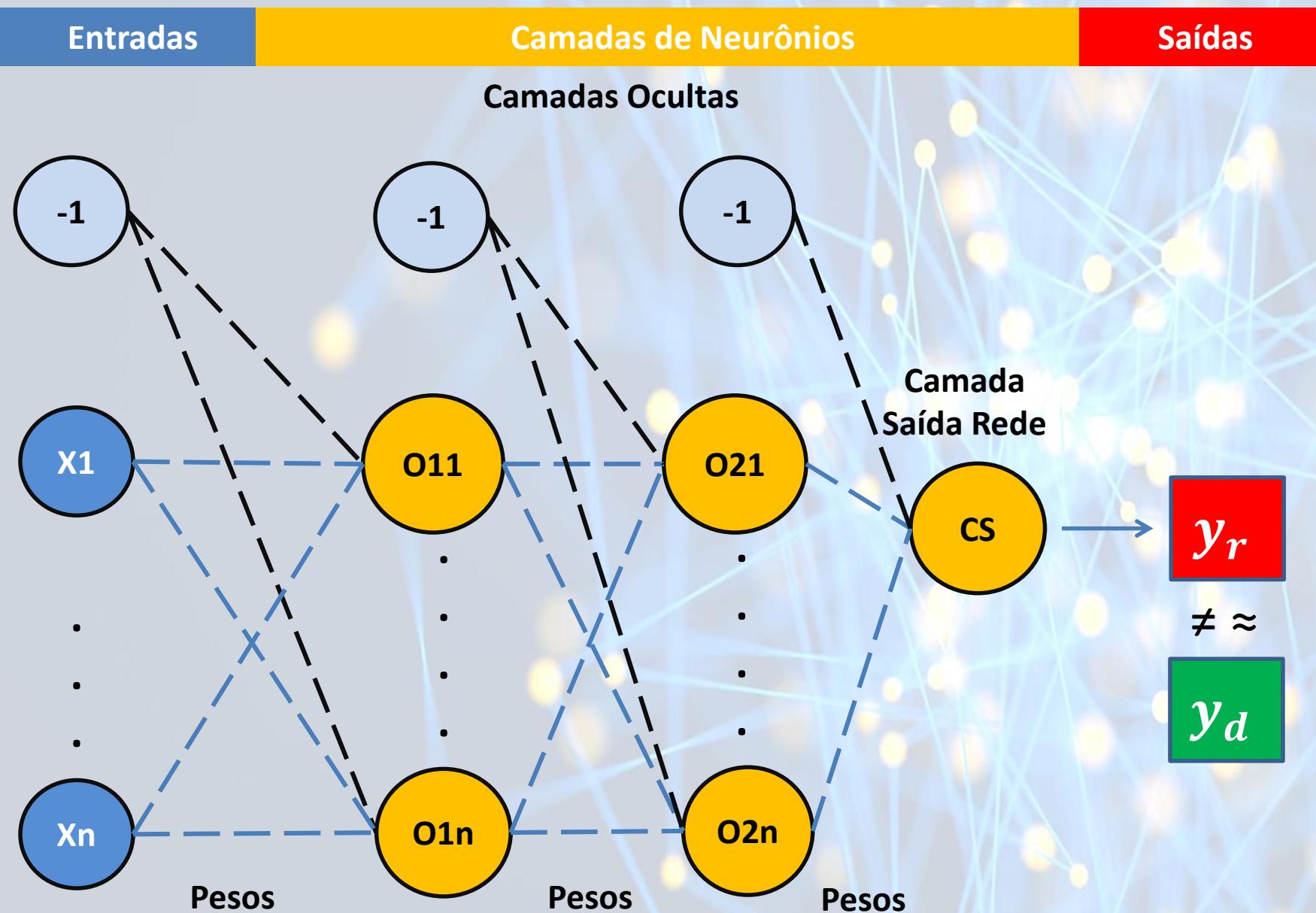
Multilayer Perceptron (MLP)



Multilayer Perceptron (MLP)

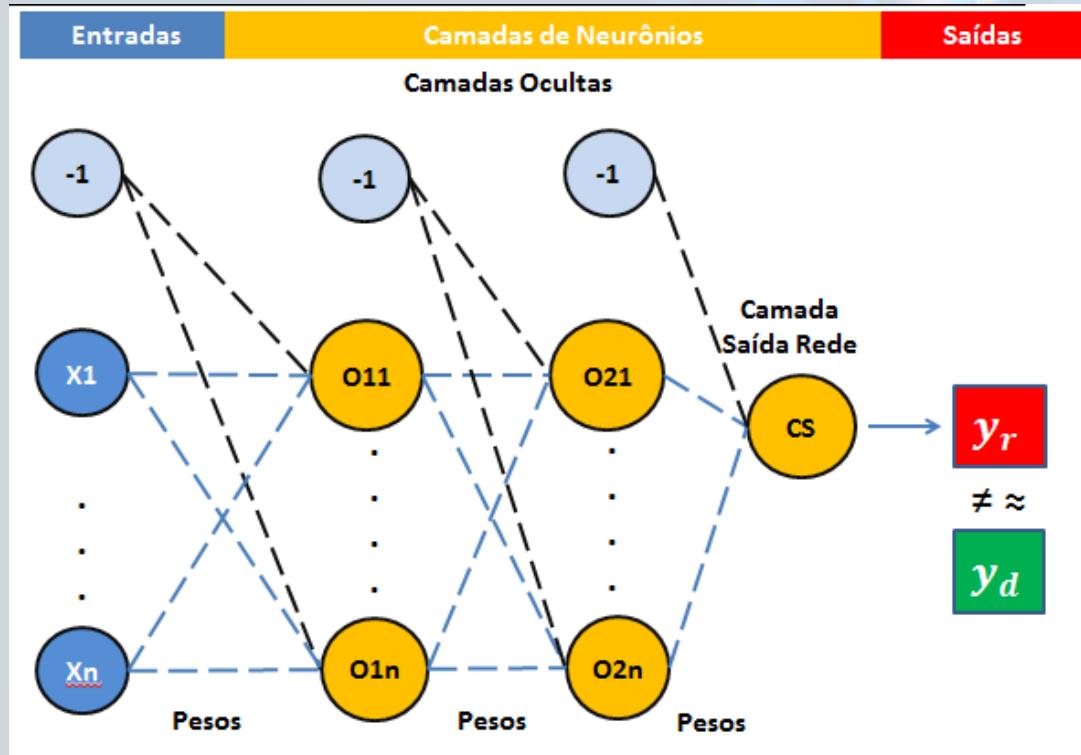


Multilayer Perceptron (MLP)



Algoritmo Backpropagation

Fase Forward



Fase Backward

$$\vec{w}(t+1) = \vec{w}(t) + \eta \cdot e \cdot \vec{x}$$

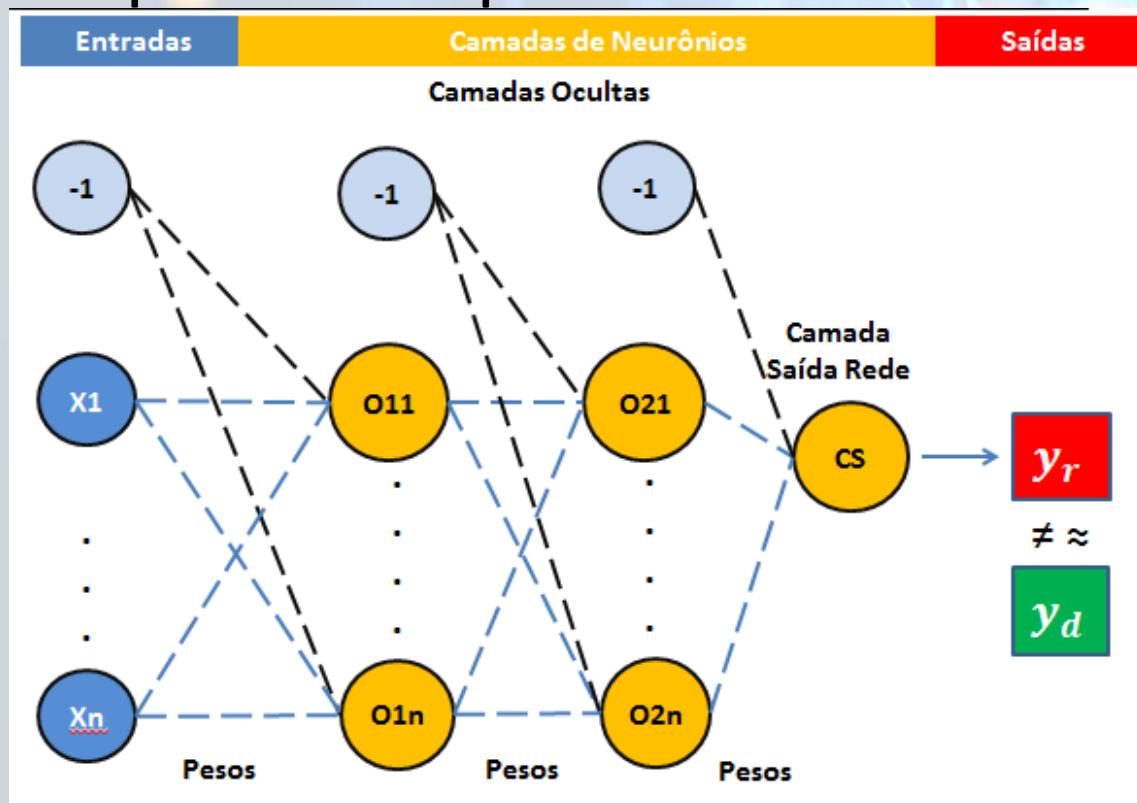
➤ Treinamento

Obs	Entradas	Saída Real
1	[20;500]	1
...
1000	[40;1000]	4



Etapas de Desenvolvimento de uma RNA (MLP)

- ✓ Estudo do problema -> Dados
- ✓ Escolha da arquitetura
 - ❖ Número de camadas ocultas
 - ❖ Número de neurônios
 - ❖ Tipos de funções de ativação
- ✓ Escolha do algoritmo de aprendizado
- ✓ Escolha dos parâmetros de qualidade da RNA



Estudo do Problema - DADOS

➤ Treinamento

Observação	Entradas	Saída Real
1	[20;500]	1
...
800	[40;1000]	4

➤ Validação

Observação	Entradas	Saída Real
1	[10;600]	2
...
200	[60;300]	3

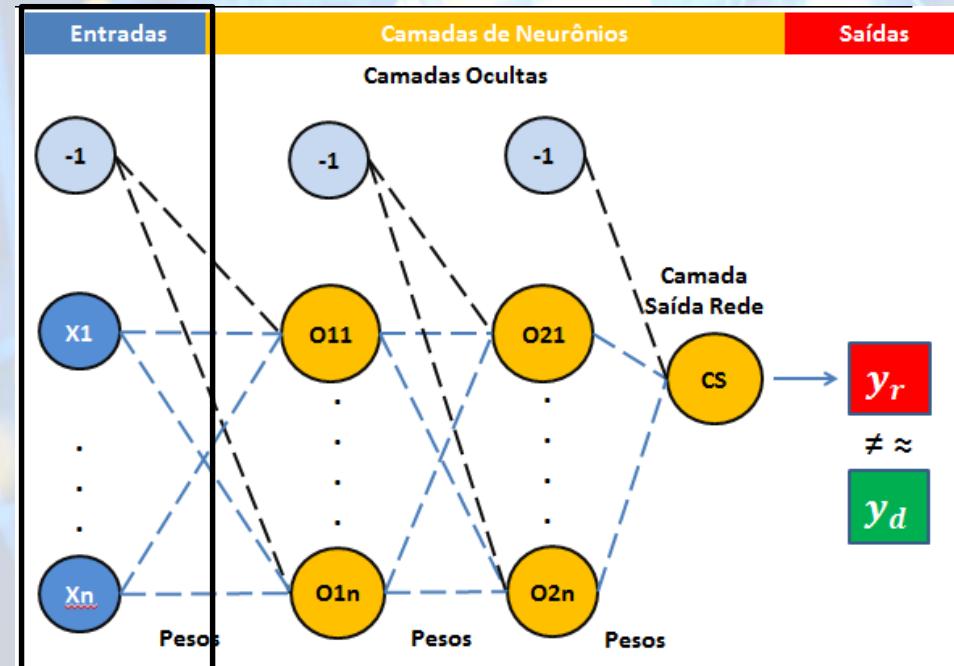
❖ Conjuntos de Dados

❖ Treinamento: 80 %

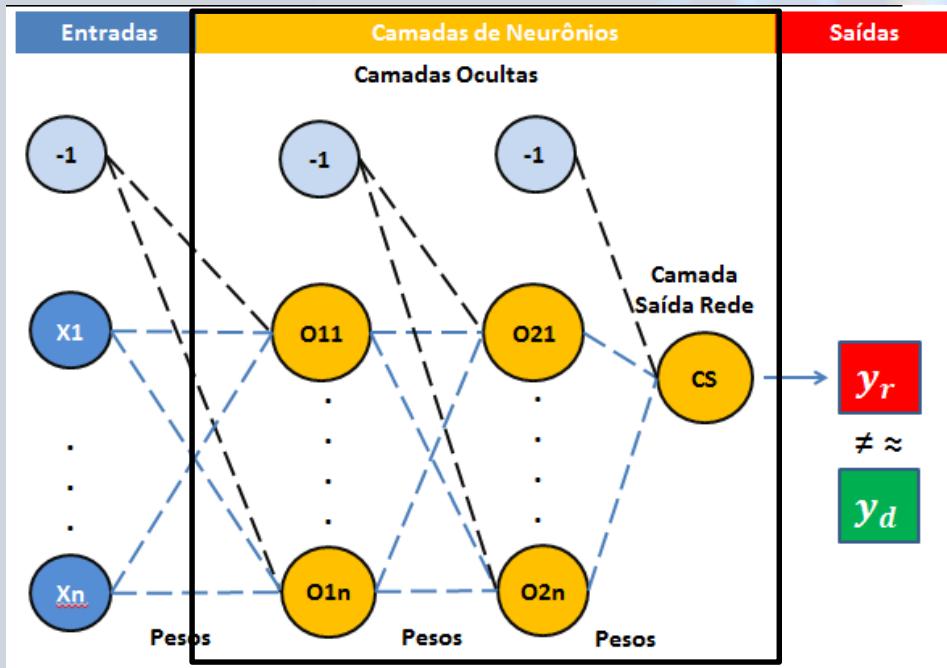
❖ Validação: 20%

➤ Teste

Observação	Entradas	Saída Real
1	[35;1000]	?
...	...	?
100	[20;300]	?



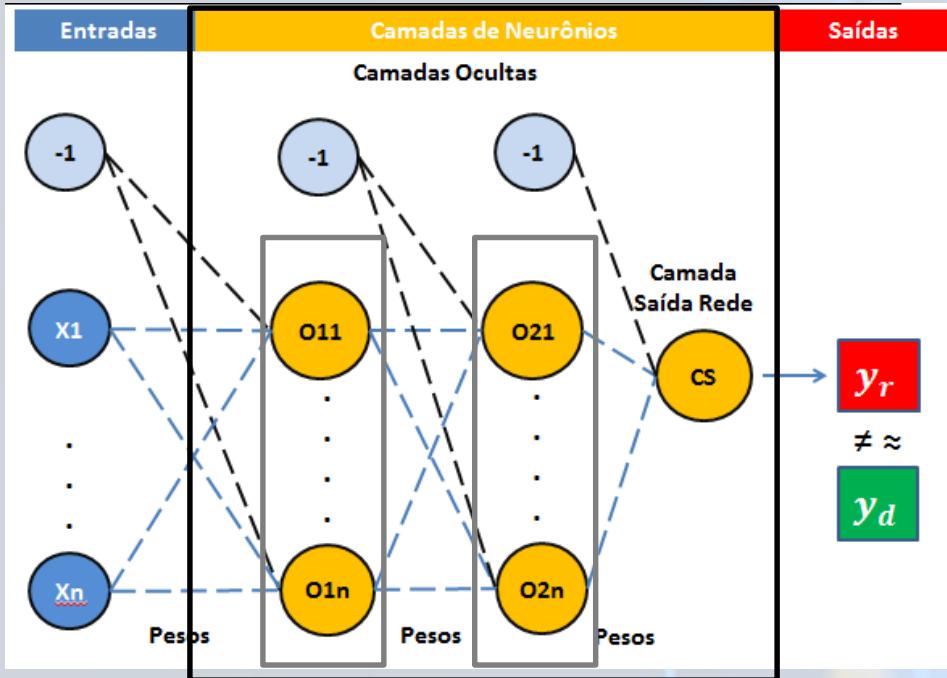
Arquitetura das RNA's - Camadas



- **OBSERVAÇÕES**
- Maior número de camadas ocultas acarreta em:
 - Maior o número de pesos
 - Maior demanda computacional
- **OBSERVAÇÕES**
- Uma única camada de saída da rede neural

- Número de camadas depende do tipo de problema
 - ✓ Problemas classificatórios
 - ❖ Linearmente Separáveis: 1 camada
 - ❖ Não Linearmente Separáveis: pelo menos uma
 - ✓ Problemas de Ajuste de modelos
 - ❖ Simples: 1 Camada
 - ❖ Complexos: 2 ou mais camadas

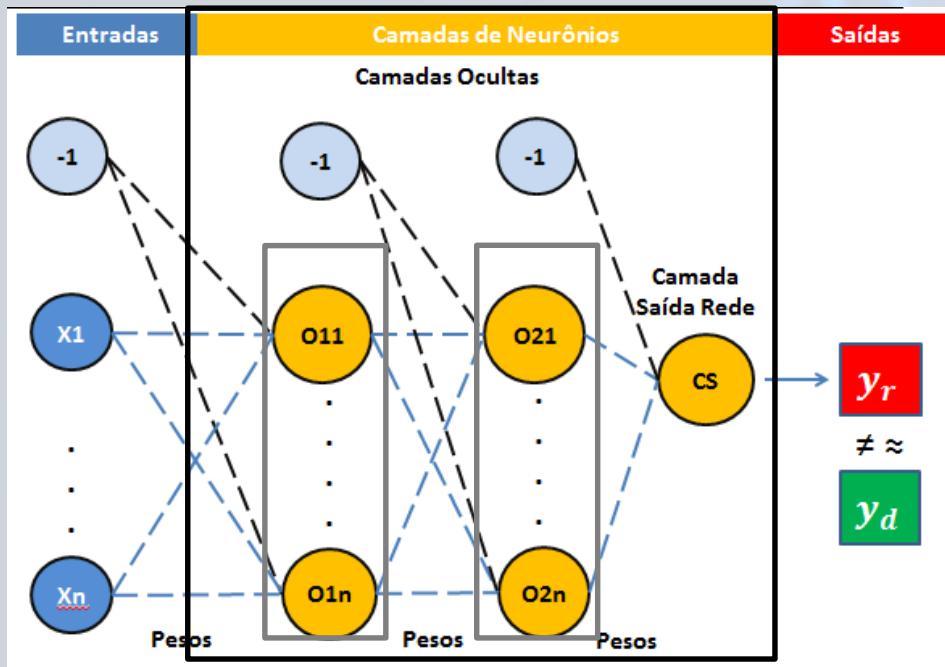
Arquitetura das RNA's - Neurônios



- **OBSERVAÇÕES**
- Maior número de neurônios acarreta em:
 - Maior o número de pesos
 - Maior demanda computacional
- **OBSERVAÇÕES**
- Número de neurônios na camada de saída:
 - 1 ou mais (dependendo do problema)

- O número de neurônios nas camadas ocultas são definidos de forma específica, ou seja, este número pode variar de uma camada para outra.
- Definição do número de neurônios das camadas ocultas:
 - ✓ Forma empírica
 - ❖ Teste utilizando várias combinações (GENES)

Funções de Ativação

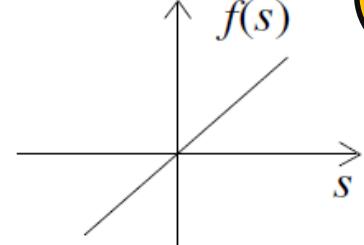
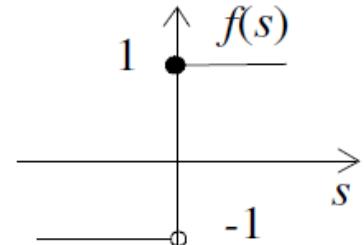
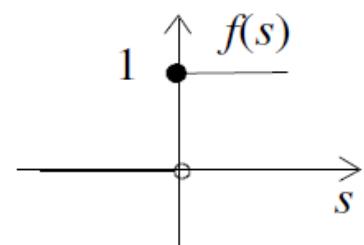


- Todos os neurônios de uma camada oculta apresentam a mesma função de ativação (FA). Porém, o tipo de FA pode variar de uma camada oculta para outra.
- Definição do tipo de função de ativação:
 - ✓ Forma empírica
 - ❖ Teste utilizando várias combinações (GENES)
 - ✓ Consenso
 - ✓ Problemas simples: linear (*purelin*)
 - ✓ Problemas Complexos: logística (*logsig*) ou tangente hiperbólica (*tansig*)

- OBSERVAÇÕES
- O tipo de função de ativação determina:
 - a demanda computacional
- OBSERVAÇÕES
- Função de ativação na camada de saída:
 - Linear (*Purelin*)

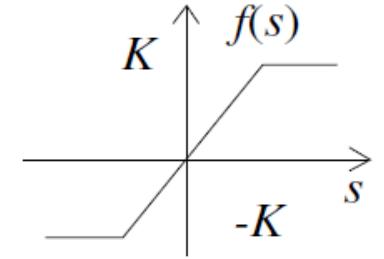
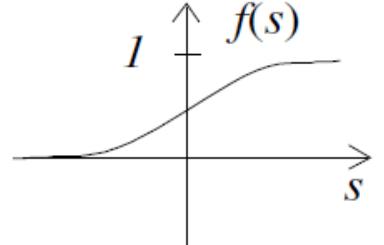
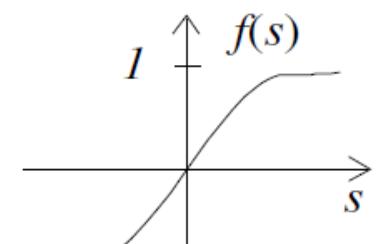
Funções de Ativação

A função de ativação incorporada numa estrutura de RNA dá a cada neurônio que a constitui a capacidade de extrair informações e a potencialidade do aprendizado da rede. O seu papel é determinar a forma e a intensidade de alteração dos valores transmitidos de um neurônio a outro.

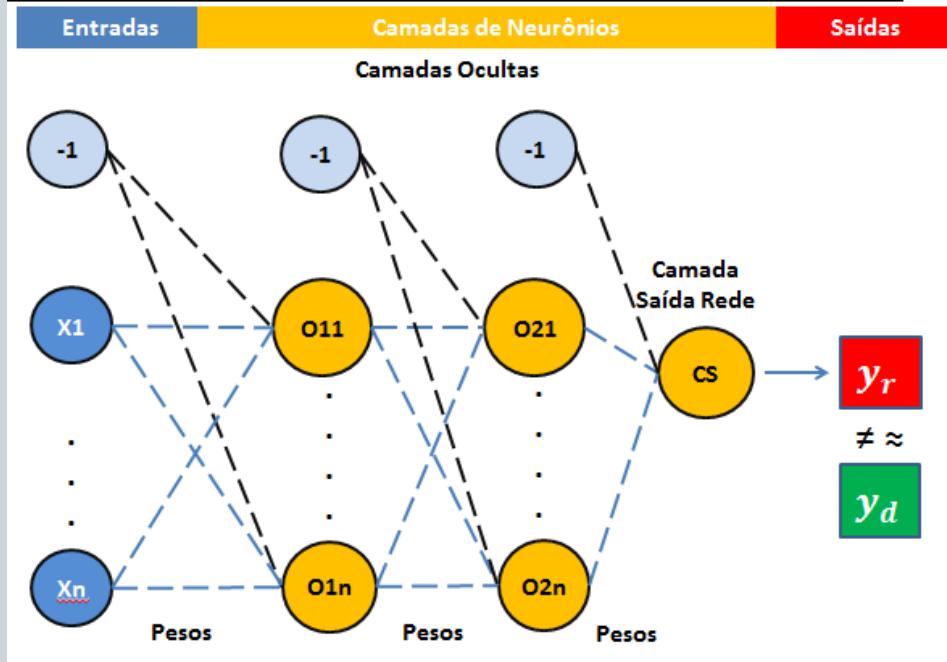
Linear	$f(s) = s$ $u = s = \sum_i^p W_i X_i - \theta$	purelin	
Sinal	$f(s) = \begin{cases} +1 & \text{se } s \geq 0 \\ -1 & \text{se } s < 0 \end{cases}$	hardlims	
Degrau	$f(s) = \begin{cases} +1 & \text{se } s \geq 0 \\ 0 & \text{se } s < 0 \end{cases}$	hardlim	

N

Funções de Ativação

BSB Limiar Lógico	ou $f(s) = \begin{cases} -K & \text{se } s \leq -K \\ s & \text{se } -K < s < +K \\ +K & \text{se } s \geq +K \end{cases}$	satlin satlins	
Logística	$f(s) = \frac{1}{1 + e^{-s}}$	logsig	
Tangente Hiperbólica	$f(s) = \tanh(s) = \frac{1 - e^{-2s}}{1 + e^{-2s}}$	tansig	

Algoritmo de Aprendizado



- OBSERVAÇÕES
- Existem diversos algoritmos baseados no princípio Backpropagation

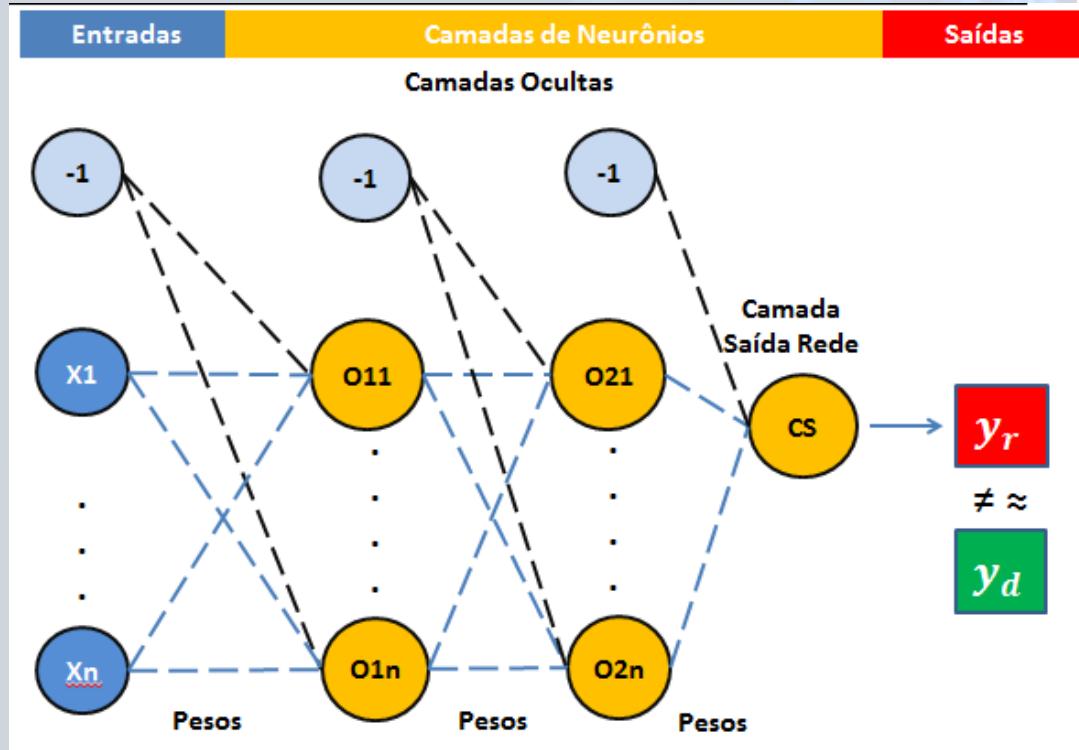
- OBSERVAÇÕES
- O tipo de algoritmo de aprendizado determina:
 - a demanda computacional

- A cada treinamento de uma RNA somente um algoritmo de aprendizado pode ser utilizado.

- Definição do algoritmo de aprendizado:
 - ✓ Forma empírica
 - ✓ Consenso
 - ✓ Problemas simples: Levenberg–Marquardt (*trainlm*)
 - ✓ Problemas Complexos: Regularização Bayesiana (*trainbr*)

Algoritmo Backpropagation

Fase Forward



Fase Backward

➤ Treinamento

Obs	Entradas	Saída Real
1	[20;500]	1
...
800	[40;1000]	4

Entradas

Execução

Erro
($y_d - y_r$)

Saída de
Rede (y_r)

Parâmetros das RNA's - Treinamento

1. **Taxa de aprendizado:** Proporção do erro que contribuirá para o ajuste dos pesos a cada iteração.

$$\vec{w}(t+1) = \vec{w}(t) + \eta \cdot e \cdot \vec{x}$$

- ❖ Uma taxa de aprendizado **muito baixa** torna o aprendizado da rede muito lento
- ❖ Uma taxa de aprendizado **muito alta** provoca oscilações no treinamento e impede a convergência do processo de aprendizado.
- ❖ Geralmente seu valor varia de 0.1 a 1.0.
- ❖ Sugestão: valor igual a 0.4.

Algoritmo de Aprendizado

1. Iterações: número de vezes que os pesos serão reajustados durante o treinamento de uma RNA.

- Número máximo de iterações (épocas ou ciclos) é determinado pela complexidade do problema.
- Maior número de iterações requer maior demanda computacional.
- Definição do número de iterações:
 - ✓ Forma empírica
 - ✓ Consenso
 - ✓ 500 a 3000 iterações

➤ OBSERVAÇÕES

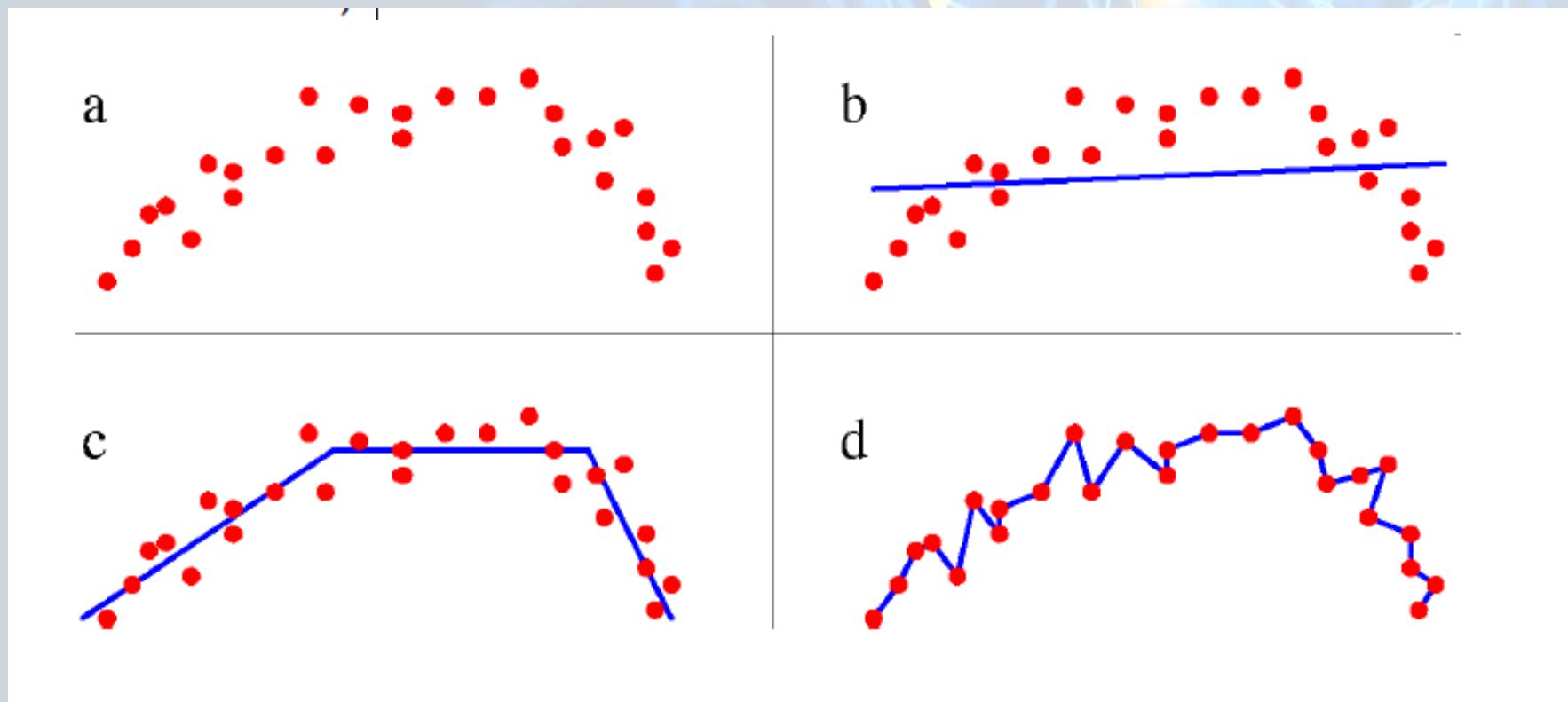
- Cuidados para determinar número de iterações:
 - Overfitting – Excesso de número de iterações
 - Underfitting – Poucas iterações

2. Erro Quadrático Médio: Medida que expressa em média a qualidade do treinamento e validação de uma RNA.

$$EQM = \frac{1}{p} \sum_{i=1}^p (y_{d(i)} - y_{r(i)})^2$$

- Sugestão é estabelecer um valor de 0.01 no primeiro treinamento e depois ajustá-lo em função do resultado

Overfitting Vs Underfitting



Resumo – RNA MLP

1. Pesos Aleatórios $\vec{w}(t)$

2. Uso dos Pesos Ajustados

$$\vec{w}(t + 1)$$



- Processamento das Entradas

- Potencial de ativação (u)

$$u = \sum_i^p W_i X_i - \Theta$$

- Ajuste dos pesos

$$\vec{w}(t + 1) = \vec{w}(t) + \eta \cdot e \cdot \vec{x}$$

- Obtenção da saída de Rede ($y_r = f(u)$)

Erro: $e = y_d - y_r$

Aplicações

<https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle®Dataset=reg-plane&learningRate=0.03®ularizationRate=0&noise=0&networkShape=4,2&seed=0.23530&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false>

Aplicações

Crop Breeding and Applied Biotechnology 13: 152-156, 2013
Brazilian Society of Plant Breeding. Printed in Brazil



ARTICLE

Artificial neural networks for adaptability and stability evaluation in alfalfa genotypes

Moysés Nascimento^{1*}, Luiz Alexandre Peternelli¹, Cosme Damião Cruz², Ana Carolina Campana Nascimento¹, Reinaldo de Paula Ferreira³, Leonardo Lopes Bhering² and Caio Césio Salgado²

Pesq. agropec. bras., Brasília, v.50, n.11, p.1054-1060, nov. 2015

DOI: 10.1590/S0100-204X2015001100008

Redes neurais artificiais para identificar genótipos de feijão-caupi semiprostrado com alta adaptabilidade e estabilidade fenotípicas

Paulo Eduardo Teodoro⁽¹⁾, Laís Mayara Azevedo Barroso⁽²⁾, Moysés Nascimento⁽²⁾, Francisco Eduardo Torres⁽¹⁾, Edvaldo Sagrilo⁽³⁾, Adriano dos Santos⁽⁴⁾ e Larissa Pereira Ribeiro⁽¹⁾

⁽¹⁾Universidade Estadual de Mato Grosso do Sul, Campus Aquidauana, Rodovia Aquidauana, Km 12, s/nº, CEP 79200-000 Aquidauana, MS, Brasil. E-mail: eduteodoro@hotmail.com, feduardo@uemt.br, larissa.uems@gmail.com ⁽²⁾Universidade Federal de Viçosa, Departamento de Estatística, Avenida Peter Henry Rolfs, s/nº, Campus Universitário, CEP 36571-000 Viçosa, MG, Brasil. E-mail: lais.azevedobarroso@gmail.com, moysesnascim@ufv.br ⁽³⁾Embrapa Meio-Norte, Avenida Duque de Caxias, nº 5.650, Buenos Aires, Caixa Postal 001, CEP 64006-220 Teresina, PI, Brasil. E-mail: edvaldo.sagrilo@embrapa.br ⁽⁴⁾Universidade Estadual do Norte Fluminense Darcy Ribeiro, Avenida Alberto Lamego, nº 2.000, Parque Califórnia, CEP 28013-602 Campos dos Goytacazes, RJ, Brasil. E-mail: adriano.agro84@yahoo.com.br

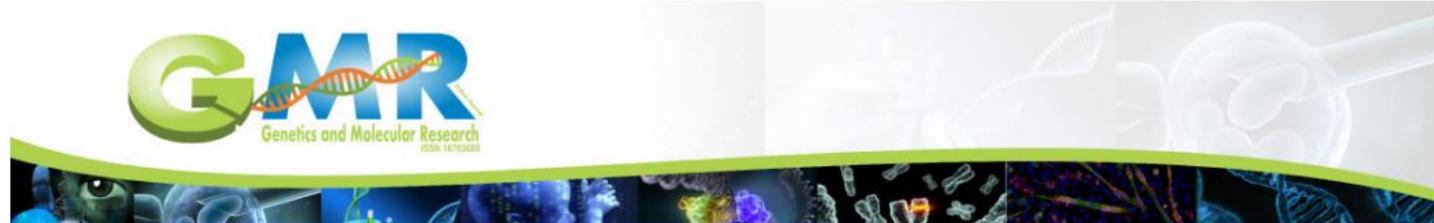
Aplicações

Scientia Agricola

<http://dx.doi.org/10.1590/0103-9016-2014-0057>

Neural networks for predicting breeding values and genetic gains

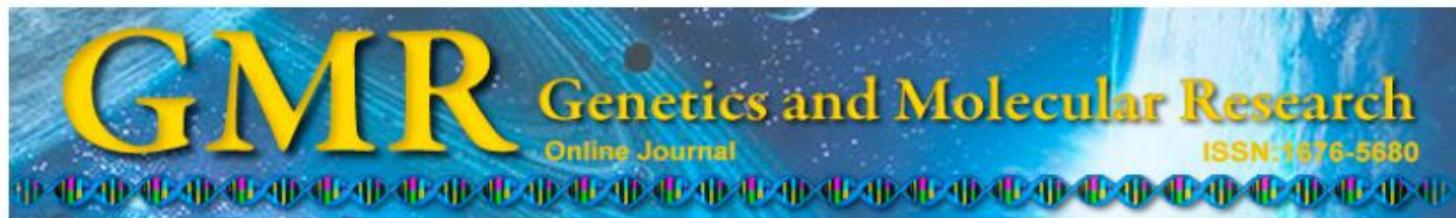
Gabi Nunes Silva^{1*}, Rafael Simões Tomaz², Isabela de Castro Sant'Anna², Moysés Nascimento¹, Leonardo Lopes Bhering², Cosme Damião Cruz^{1,2}



Evaluation of the efficiency of artificial neural networks for genetic value prediction

G.N. Silva^{1,4}, R.S. Tomaz², I.C. Sant'Anna^{3,4}, V.Q. Carneiro^{3,4}, C.D. Cruz^{3,4} and
M. Nascimento^{1,4}

Aplicações



Superiority of artificial neural networks for a genetic classification procedure

I.C. Sant'Anna^{1,4}, R.S. Tomaz³, G.N. Silva^{2,4}, M. Nascimento^{2,4},
L.L. Bhering¹ and C.D. Cruz^{1,2,4}

Cenários	Similaridade	Taxa de Erro Aparente (%)	
		Análise Discriminante	RNA
1	50%	27%	0%
2	75%	66%	0%
3	87,5%	78%	0%

Aplicações



Artificial neural networks as auxiliary tools for the improvement of bean plant architecture

V.Q. Carneiro^{1,4*}, G.N. Silva^{2,4*}, C.D. Cruz^{1,2,4}, P.C.S Carneiro^{1,5},
M. Nascimento^{2,4} and J.E.S. Carneiro^{3,5}

Taxas de erro aparente para as RNA's utilizando o diâmetro do hipocótilo individualmente ou em conjunto com altura de plantas.

Procedimentos	Ano de Predição	Taxa de Erro Aparente – TEA (%)		
		Treinamento	Validação	Predição
RNA (DH)	2009	10,33	15,79	15,79
RNA (DH)	2007	12,83	14,04	15,79
RNA (DH + ALT)	2009	6,83	12,28	5,26
RNA (DH + ALT)	2007	1,83	3,51	0,00

Aplicações

Pesq. agropec. bras. vol.52 no.3 Brasília Mar. 2017

<http://dx.doi.org/10.1590/s0100-204x2017000300009>

Artificial neural networks compared with Bayesian generalized linear regression for leaf rust resistance prediction in Arabica coffee

Gabi Nunes Silva⁽¹⁾, Moysés Nascimento⁽¹⁾, Isabela de Castro Sant'Anna⁽²⁾, Cosme Damião Cruz⁽²⁾, Eveline Teixeira Caixeta⁽³⁾, Pedro Crescêncio Souza Carneiro⁽²⁾, Renato Domiciano Silva Rosado⁽²⁾, Kátia Nogueira Pestana⁽⁴⁾, Dênia Pires de Almeida⁽⁵⁾ and Marciane da Silva Oliveira⁽²⁾

Table 1. Prediction error rates obtained with artificial neural network and Bayesian generalized linear regression (BGLR) in training and validation populations, and percentage of agreement between the two methodologies.

Methodology	Classification error (%)	
	Training	Validation
Artificial neural network	0.00	1.600
Bayesian generalized linear regression	0.00	2.400
Concordance (%)	-	81.37

Aplicações

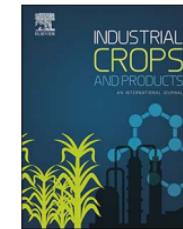
Industrial Crops & Products 108 (2017) 806–813



Contents lists available at ScienceDirect

Industrial Crops & Products

journal homepage: www.elsevier.com/locate/indcrop



Research Paper

High-performance prediction of macauba fruit biomass for agricultural and industrial purposes using Artificial Neural Networks



Carla Aparecida de O. Castro^a, Rafael T. Resende^{a,*}, Kacilda N. Kuki^c, Vinícius Q. Carneiro^b, Gustavo E. Marcatti^d, Cosme Damião Cruz^b, Sérgio Y. Motoike^c

Table 5
Results of prediction accuracy (r) and bias estimation (Bias) of the semi-destructive and non-destructive procedures.

Procedure	Fruit Variable	RS		G1 → G2		G2 → G1				
		ANN		MLM		ANN		MLM		
		r	Bias (%)	r	Bias (%)	r	Bias (%)	r		
Semi-destructive	KDW	0.93	8.73	0.56	0.97	6.56	0.50	0.92	10.08	0.28
	PDW	0.98	4.46	0.73	0.98	4.64	0.80	0.97	5.05	0.77
	HDW	0.97	6.00	0.81	0.98	4.82	0.79	0.96	5.12	0.67
	EDW	0.95	5.80	0.66	0.98	5.07	0.62	0.96	5.34	0.56
	KOC	0.96	0.94	0.16	0.96	0.74	0.02	0.95	1.05	0.00
	POC	0.89	7.23	0.41	0.90	3.97	0.59	0.89	6.67	0.19
Non-destructive	KDW	0.96	8.17	0.50	0.97	6.53	0.50	0.90	11.12	0.25
	PDW	0.97	5.63	0.78	0.94	7.58	0.76	0.94	6.88	0.76
	HDW	0.96	6.37	0.82	0.95	5.79	0.78	0.94	6.52	0.68
	EDW	0.95	6.56	0.63	0.97	5.38	0.59	0.90	8.64	0.56
	KOC	0.86	1.83	0.19	0.94	1.43	0.02	0.79	2.10	0.06
	POC	0.75	10.70	0.45	0.76	7.95	0.62	0.73	10.02	0.30

RS: Random Sampling; G1 → G2: Training on G1 group and validation on G2 group; G2 → G1: Training on G2 group and validation on G1 group; ANN: Artificial Neural Network; MLM: Multivariate Linear Model. Fruit variables acronyms, KDW: Kernel or seed dry weight; EDW: Endocarp Dry Weight; PDW: Pulp or Mesocarp dry Weight; HDW: Husk or Exocarp Dry Weight; KOC: Oil Content in the Kernel; POC: Oil Content in the Pulp.

Aplicações

Gianola et al. BMC Genetics 2011, 12:87
<http://www.biomedcentral.com/1471-2156/12/87>



METHODOLOGY ARTICLE

Open Access

Predicting complex quantitative traits with Bayesian neural networks: a case study with Jersey cows and wheat

Daniel Gianola^{1,2,3}, Hayrettin Okut^{1,4*}, Kent A Weigel² and Guilherme JM Rosa^{1,3}

González-Camacho et al. BMC Genomics (2016) 17:208
DOI 10.1186/s12864-016-2553-1

BMC Genomics

RESEARCH ARTICLE

Open Access

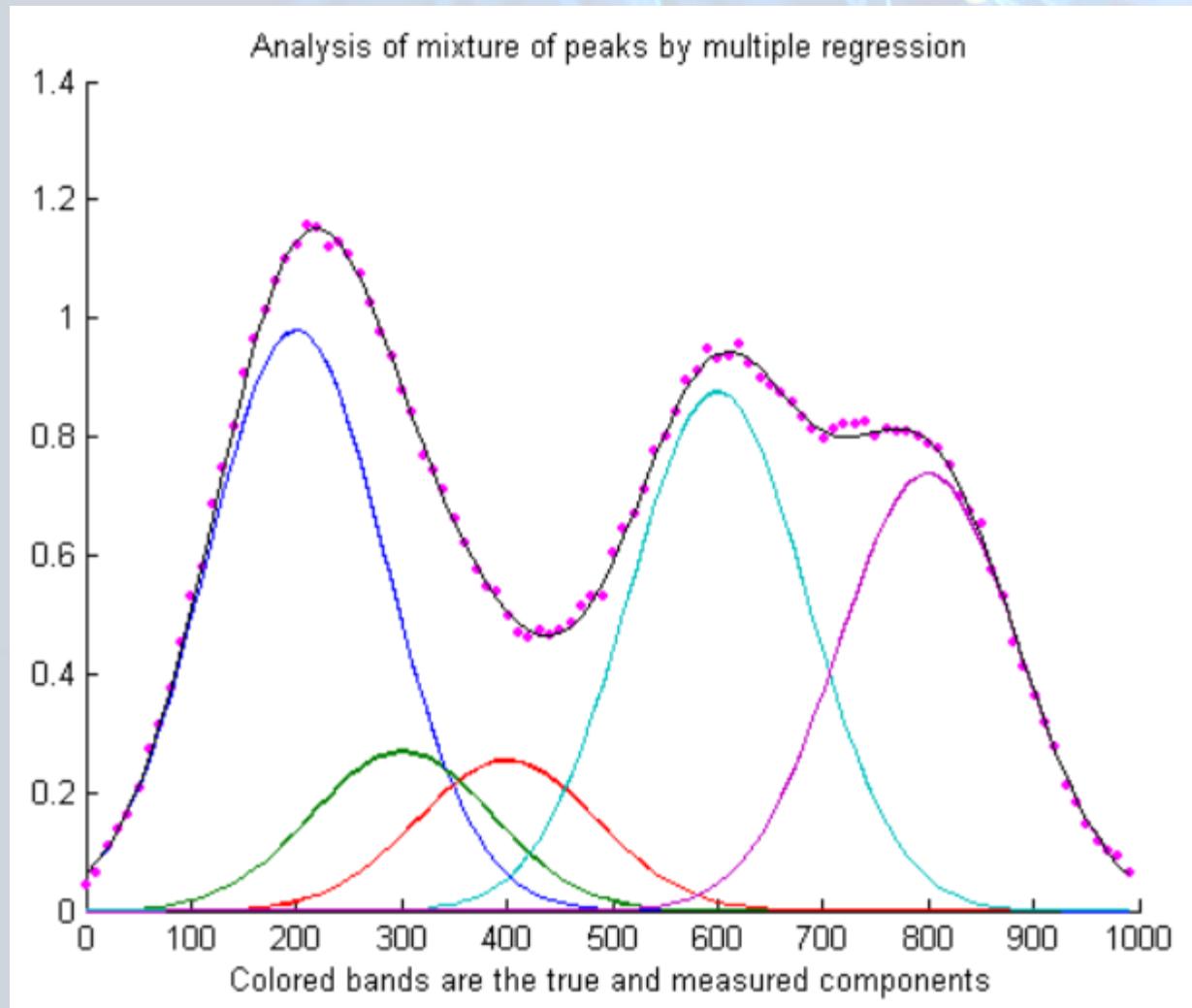
Genome-enabled prediction using probabilistic neural network classifiers



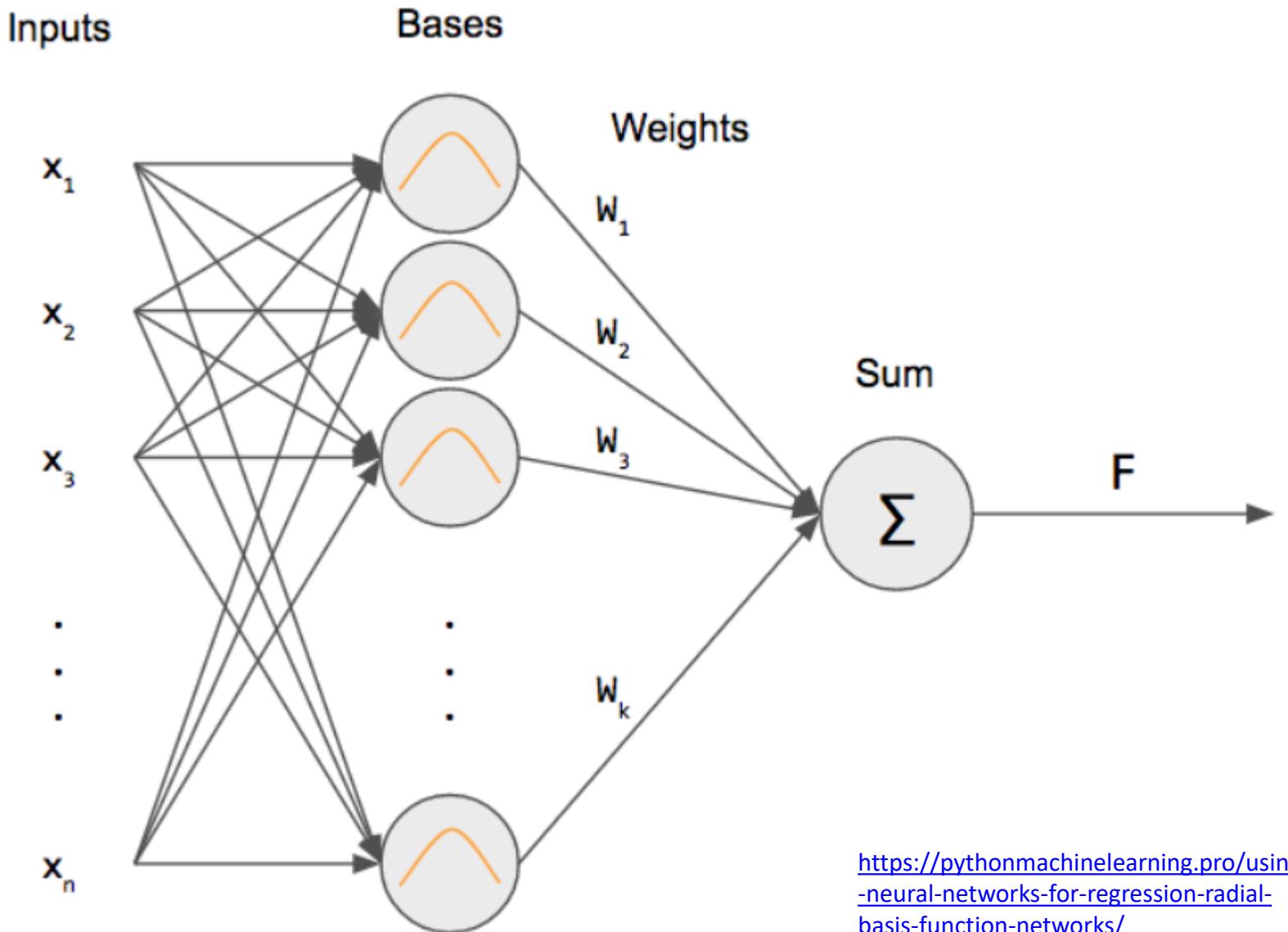
Juan Manuel González-Camacho¹, José Crossa^{2*}, Paulino Pérez-Rodríguez¹, Leonardo Ornella³ and Daniel Gianola⁴

Rede Neural de Base Radial Aprendizado Supervisionado

Rede Neural de Base Radial Regressão



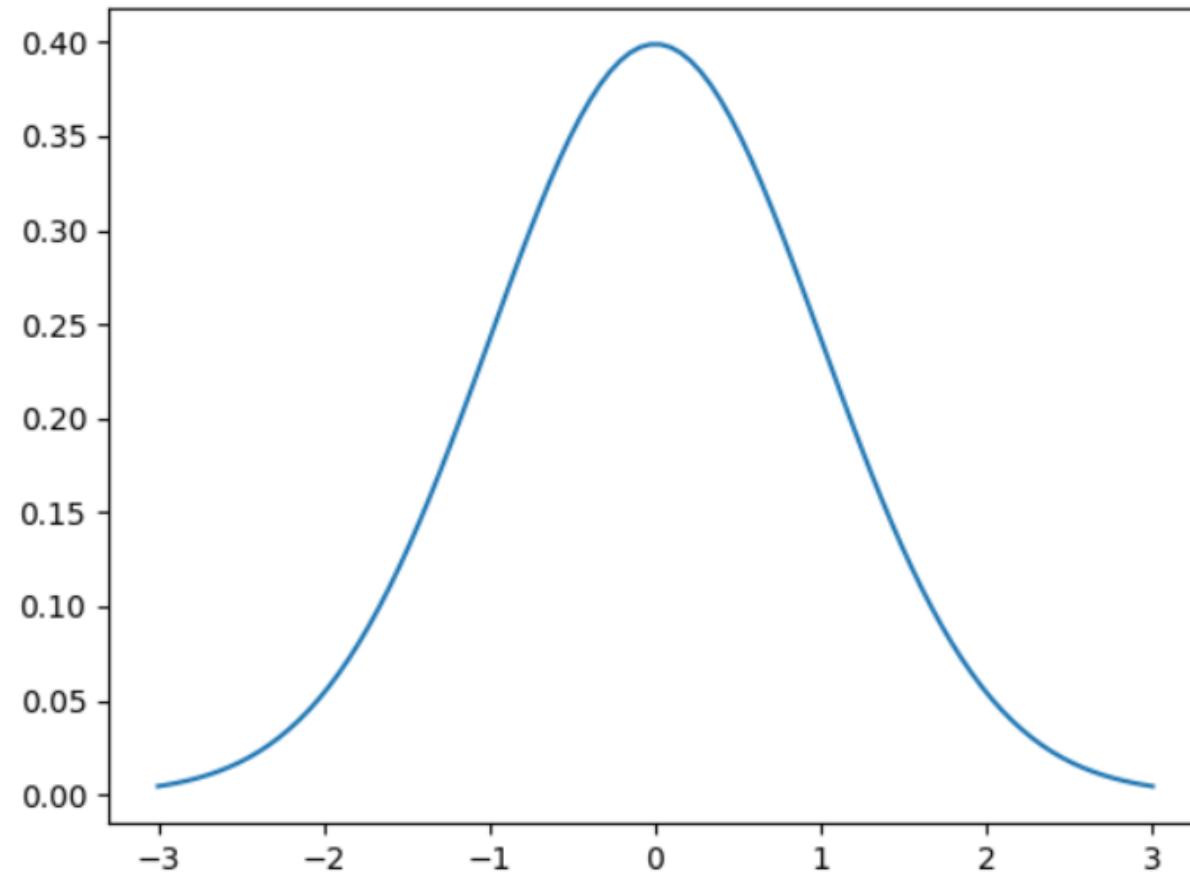
Rede Neural de Base Radial



<https://pythonmachinelearning.pro/using-neural-networks-for-regression-radial-basis-function-networks/>

Função Gaussiana

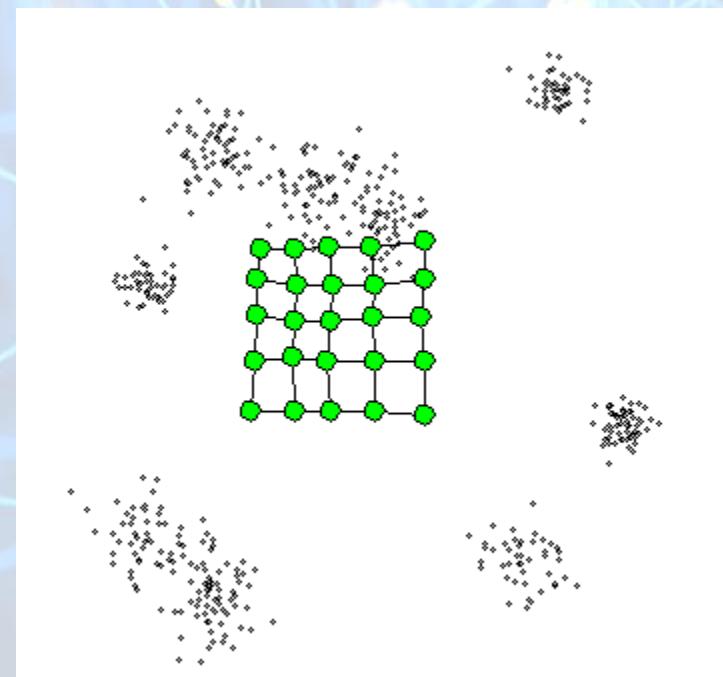
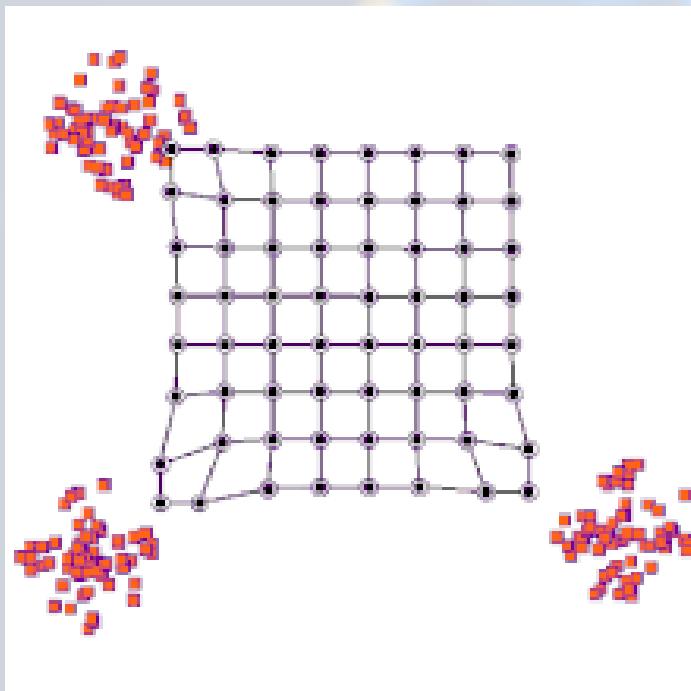
$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x - \mu)^2}{2\sigma^2}}$$



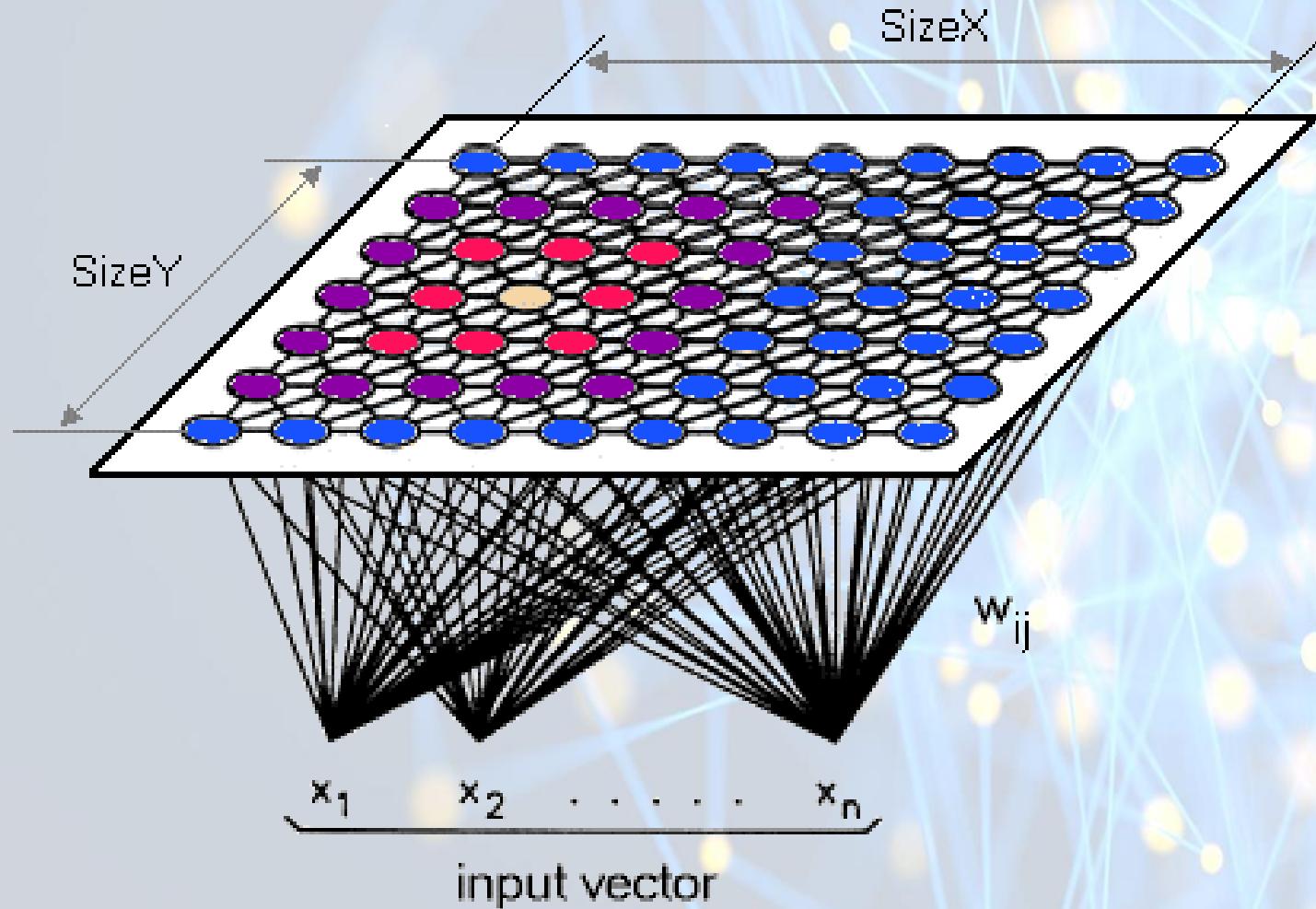
Mapas Auto Organizáveis – Kohonen

Aprendizado Não Supervisionado

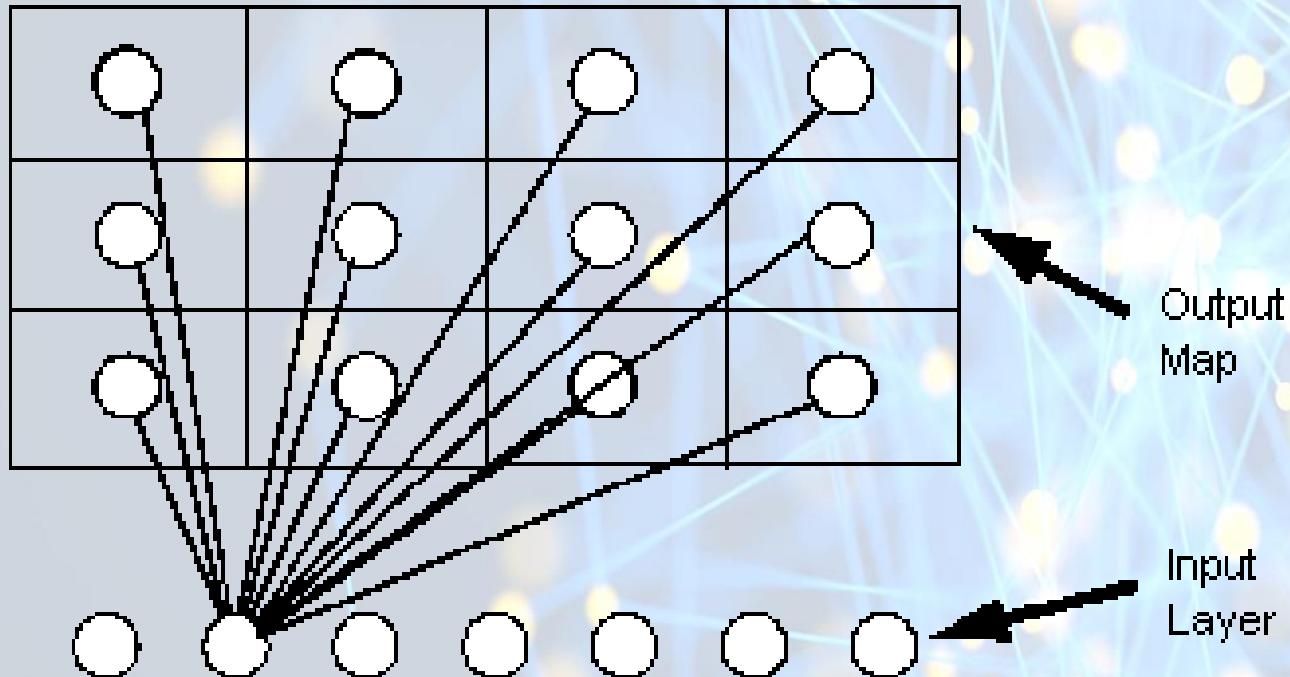
Mapas Auto-Organizáveis



Mapas Auto-Organizáveis

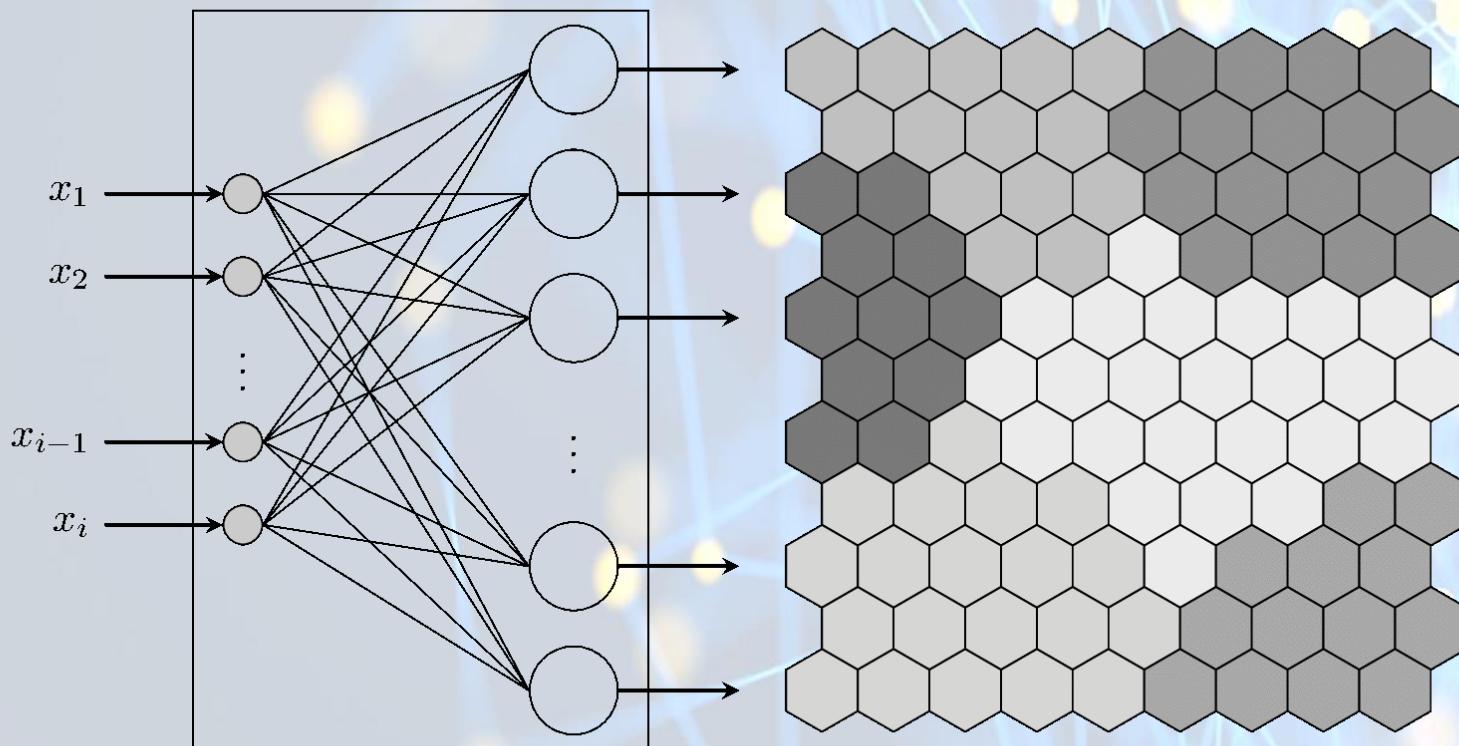


Mapas Auto-Organizáveis



All input neurons are connected to all neurons
in the output map. (Not shown for clarity)

Mapas Auto-Organizáveis



Mapas Auto-Organizáveis

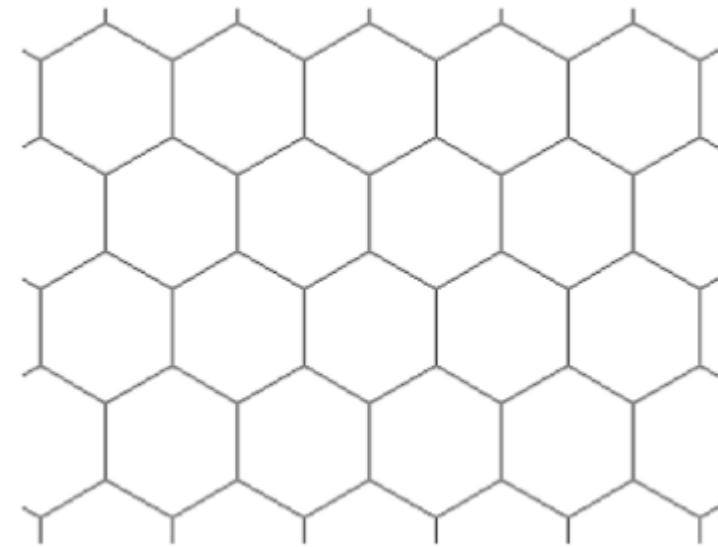
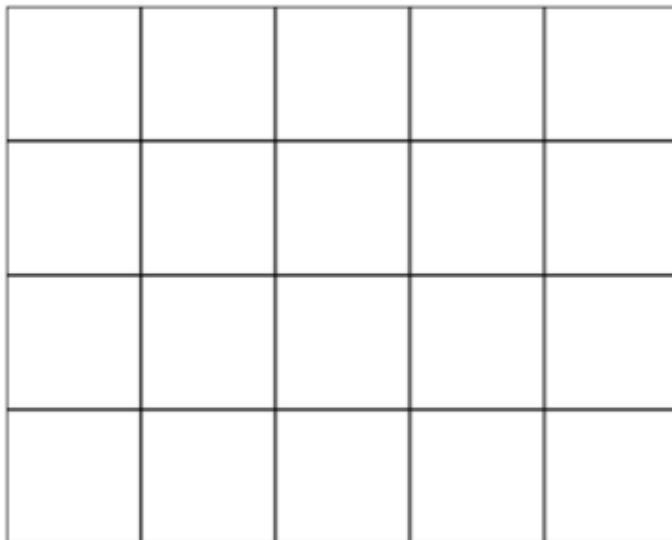
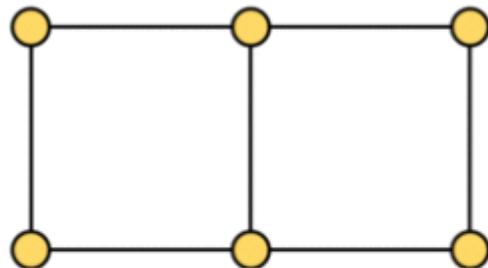


Figure 1. SOM grids can either be rectangular or hexagonal. Each square or hexagon is a neuron.

Mapas Auto-Organizáveis



Step 0: Position neurons (orange) in data space.

Mapas Auto-Organizáveis



Mapas Auto-Organizáveis

