



Introdução às Redes Neurais Artificiais

Universidade Federal Fluminense
Ygor Moreira Lima

Conteúdo

01 Introdução ao Machine Learning

Conceitos Básicos de Aprendizado de Máquina (Machine Learning) | Tipos de Aprendizado de Máquina | O papel das Redes Neurais no Aprendizado de Máquina

02 Neurônios Artificiais e Perceptrons

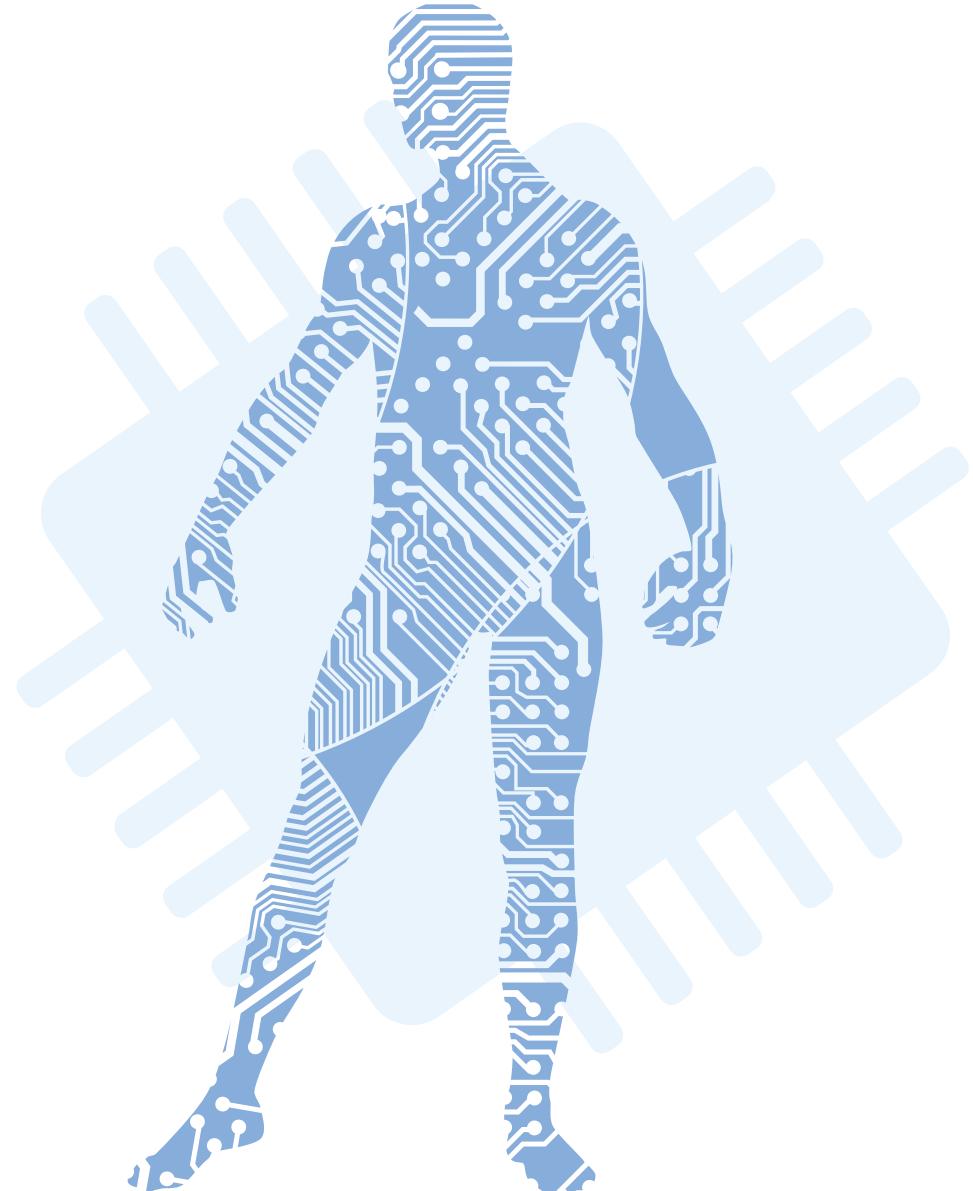
O Modelo de Neurônio Artificial | Perceptrons | Funções de Ativação e suas Aplicações

03 Aplicações das Redes Neurais

Reconhecimento de padrões | Visão Computacional | Outros campos de aplicação

04 Redes Neurais Artificiais

Arquitetura de redes neurais artificiais | Camadas ocultas e aprofundamento de redes | Feedforwarding | Backpropagation | Ajuste dos Pesos (gradient descent, delta, taxa de aprendizagem, momento)



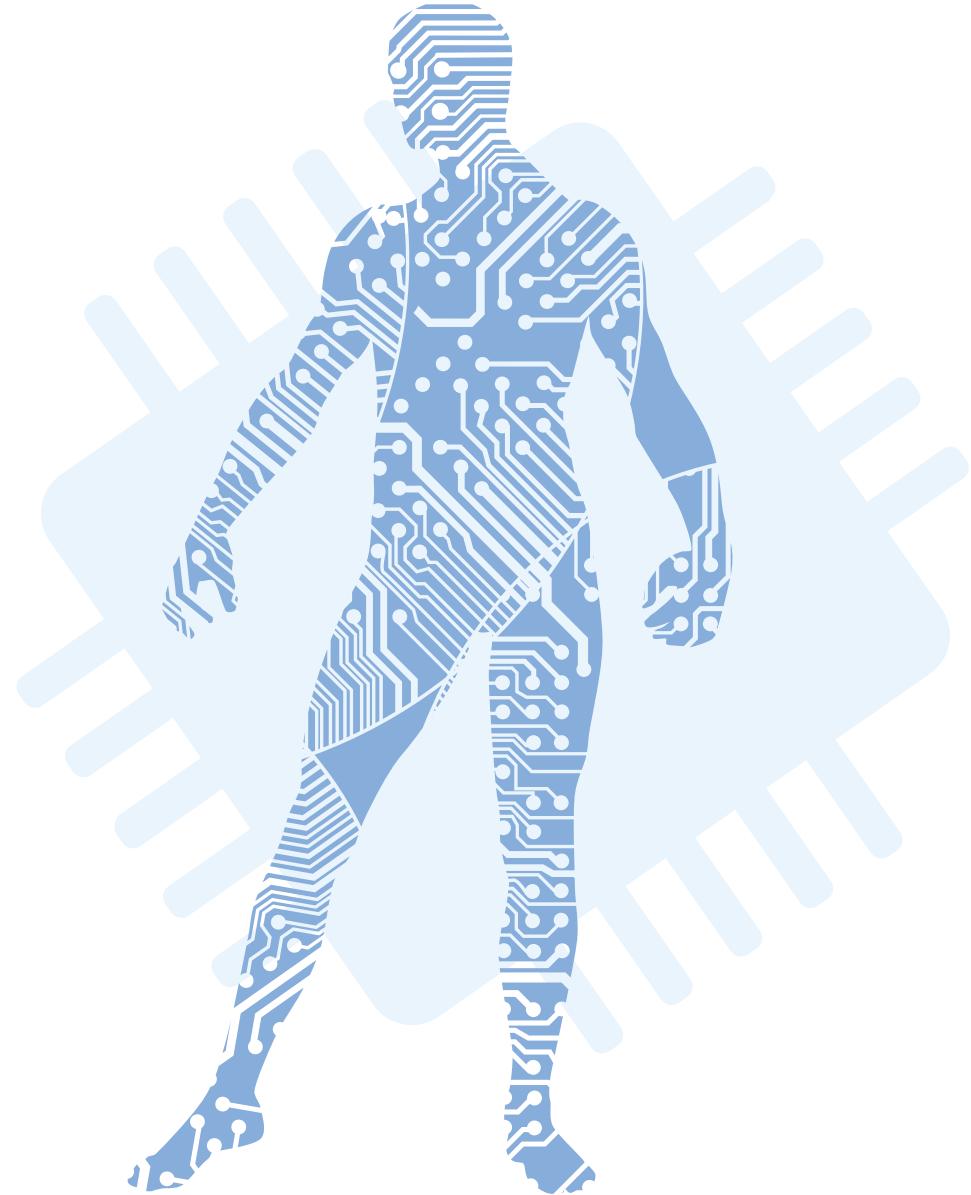
Conteúdo

05 Bibliotecas e Ferramentas

Configuração do ambiente de desenvolvimento | Introdução ao TensorFlow e Keras | Definindo a arquitetura da rede | Treinamento e avaliação do modelo | Transfer learning

06 Encerramento

Índice Remissivo | Contato





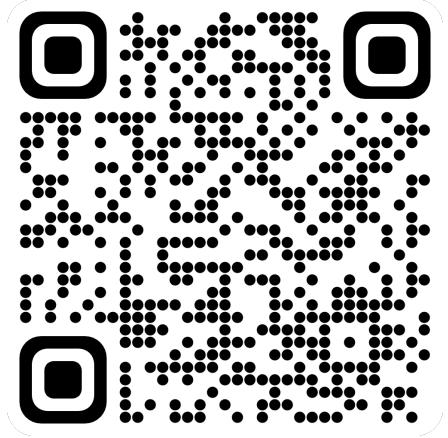
Sobre o curso

Sobre o Curso

- Material disponibilizado em <https://github.com/ygordev/uff-introducao-as-redes-neurais-artificiais>
- Duração: 4h, intervalos de 10 minutos
- Abordagem Teórica + Prática
- Composição da Turma

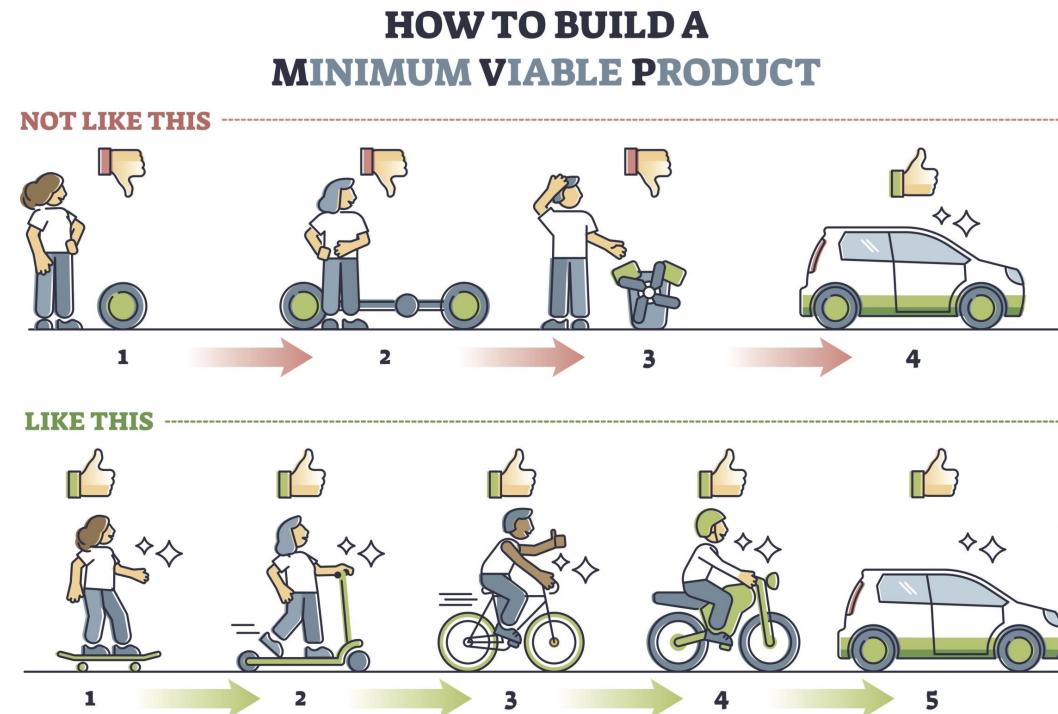
CONTATO:

Instagram: <https://Instagram.com/ygorml>
Linkedin: <https://www.linkedin.com/in/ygormoreiralima/>
Email: admin@datawarfare.org
Twitter/X: <https://twitter.com/colt7r>



Sobre o Curso

O que o curso é / o que não é





Introdução ao Machine Learning

Introdução ao Machine Learning

Conceitos Básicos de Aprendizado de Máquina

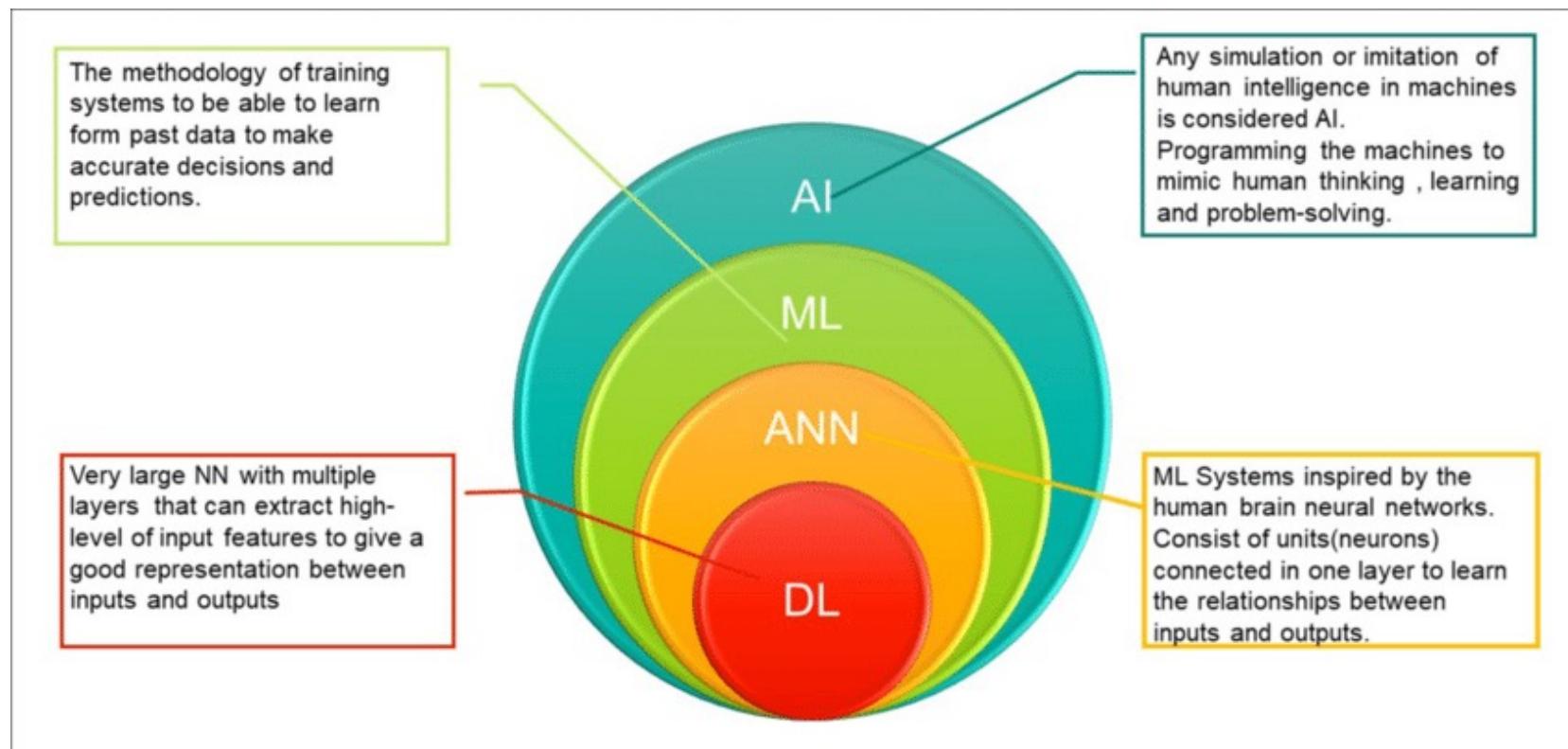
Objetivos do Módulo:

- Compreender os conceitos básicos do aprendizado de máquina.
- Identificar os diferentes tipos de aprendizado de máquina.
- Explorar o papel das redes neurais no aprendizado de máquina.



Introdução ao Machine Learning

Conceitos Básicos de Aprendizado de Máquina



Introdução ao Machine Learning

Conceitos Básicos de Aprendizado de Máquina

Definição Básica de “Aprendizado de Máquina”:

- Machine Learning is the science (and art) of programming computers **so they can learn from data**.
- [Machine Learning is the] field of study that gives computers the ability to learn **without being explicitly programmed** (Arthur Samuel, 1959)

Fonte: Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, de Aurélien Géron, pg 2.

Introdução ao Machine Learning

Conceitos Básicos de Aprendizado de Máquina

Por que Aprendizado de Máquina?

- **Detecção de Padrões**
 - Segurança Cibernética: Detecção de anomalias em logs de redes, por exemplo
 - Medicina: Detecção de padrões em imagens médicas → doenças em estágios iniciais
- **Tomadas de Decisões baseadas em Dados**
 - Algoritmos de Navegação Autônoma → Tesla
 - Marketing personalizado, baseado no comportamento do cliente → Instagram e Facebook Ads
- **Automatização de Tarefas**
 - Robôs em Linhas de Montagem (indústria) → Ford utiliza na montagem de veículos



Introdução ao Machine Learning

Tipos de Aprendizagem de Máquina

- **Supervisionado**
 - Quando apresentamos ao algoritmo dados de entrada e as respectivas saídas.
 - Treinamento realizado sobre dados rotulados
 - Principais Tarefas: Classificação e Regressão
- **Não-supervisionado**
 - Quando apresentamos somente os dados de entrada e o algoritmo descobre as saídas.
 - Extração de Características → Descoberta de estrutura, por exemplo.
 - Principais Tarefas: Clustering, Hierarchical Clustering, Segmentação de Imagens
- **Reforço**
 - O agente aprende a atingir uma meta em um ambiente incerto e potencialmente complexo.
 - No aprendizado por reforço, o sistema de inteligência artificial enfrenta uma situação.
 - O computador utiliza tentativa e erro para encontrar uma solução para o problema.
 - A rede gera seus próprios dados
 - É definida uma política de recompensa. **Cabe ao modelo descobrir como executar a tarefa para maximizar a recompensa**, começando com testes totalmente aleatórios e terminando com táticas sofisticadas.
 - Principais Tarefas: Controle de Veículos Autônomos, Jogos e Simulações (jogar xadrez)

Three main types of
Machine Learning Algorithms



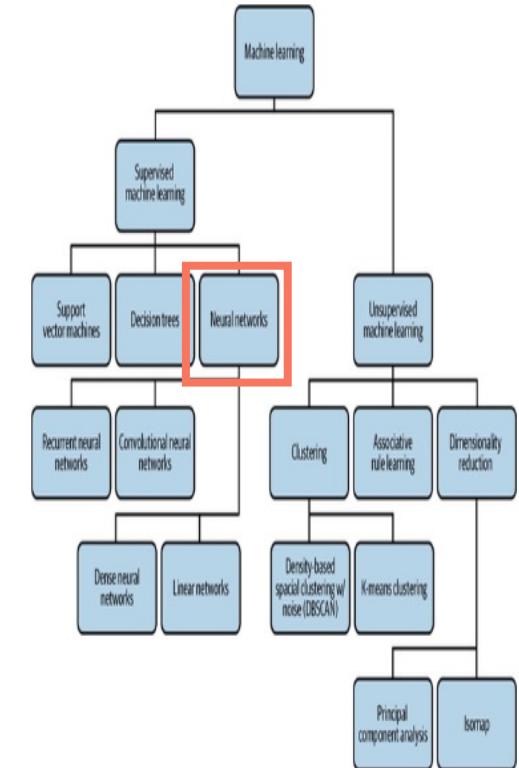
Fonte: DeepLearningBook.com.br

Introdução ao Machine Learning

Tipos de Aprendizagem de Máquina

Tipos de Aprendizagem de Máquina – Supervisionado

- Nosso objeto de estudo do curso (RNAs) se concentra aqui, na aprendizagem supervisionada.
- Classificação
- Regressão



Fonte: Machine Learning Design Patterns Solutions to Common Challenges in Data Preparation, Model Building, and MLOps, de Valliappa Lakshmanan, Sara Robinson

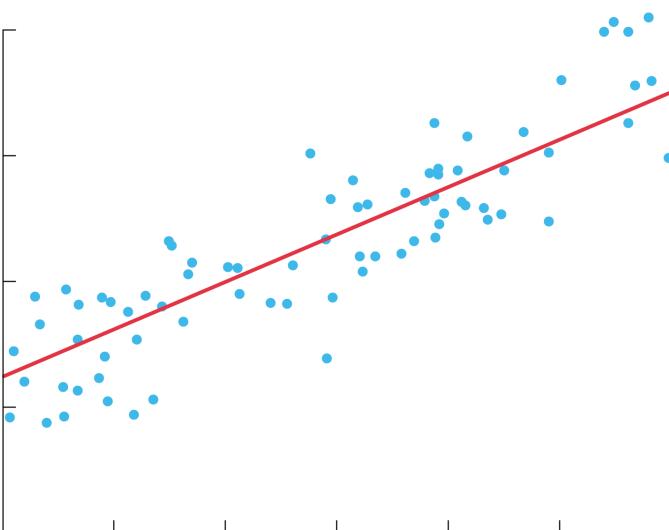
Introdução ao Machine Learning

Tipos de Aprendizagem de Máquina

Tipos de Aprendizagem de Máquina – Supervisionado

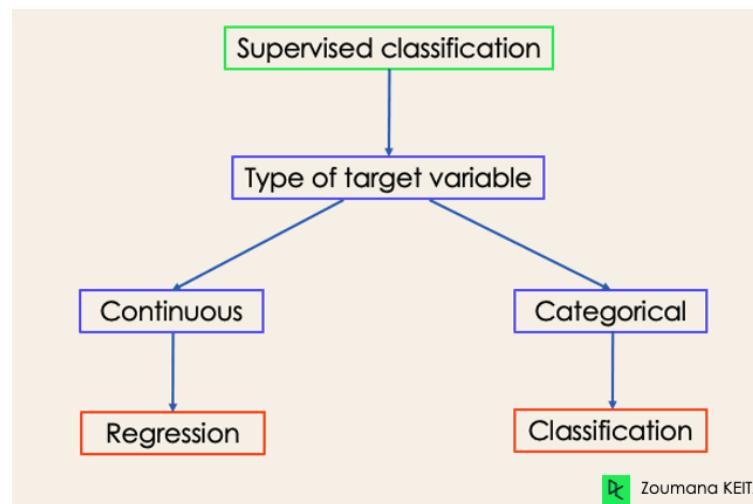
Building a Regression Model

The line summarizes the relationship between x and y.



Source: HBR.org

HBR



Fonte: Machine Learning Design Patterns Solutions to Common Challenges in Data Preparation, Model Building, and MLOps, de Valliappa Lakshmanan, Sara Robinson

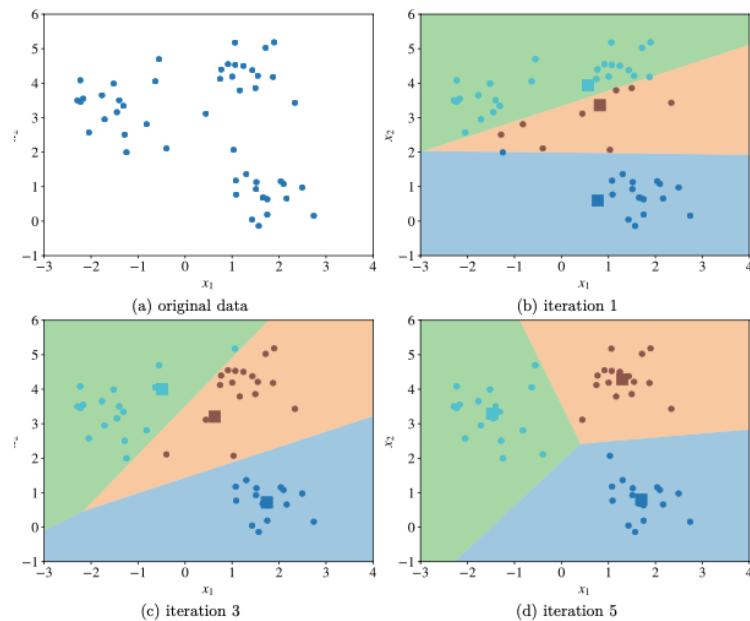


Introdução ao Machine Learning

Tipos de Aprendizagem de Máquina

Tipos de Aprendizagem de Máquina – Não Supervisionado

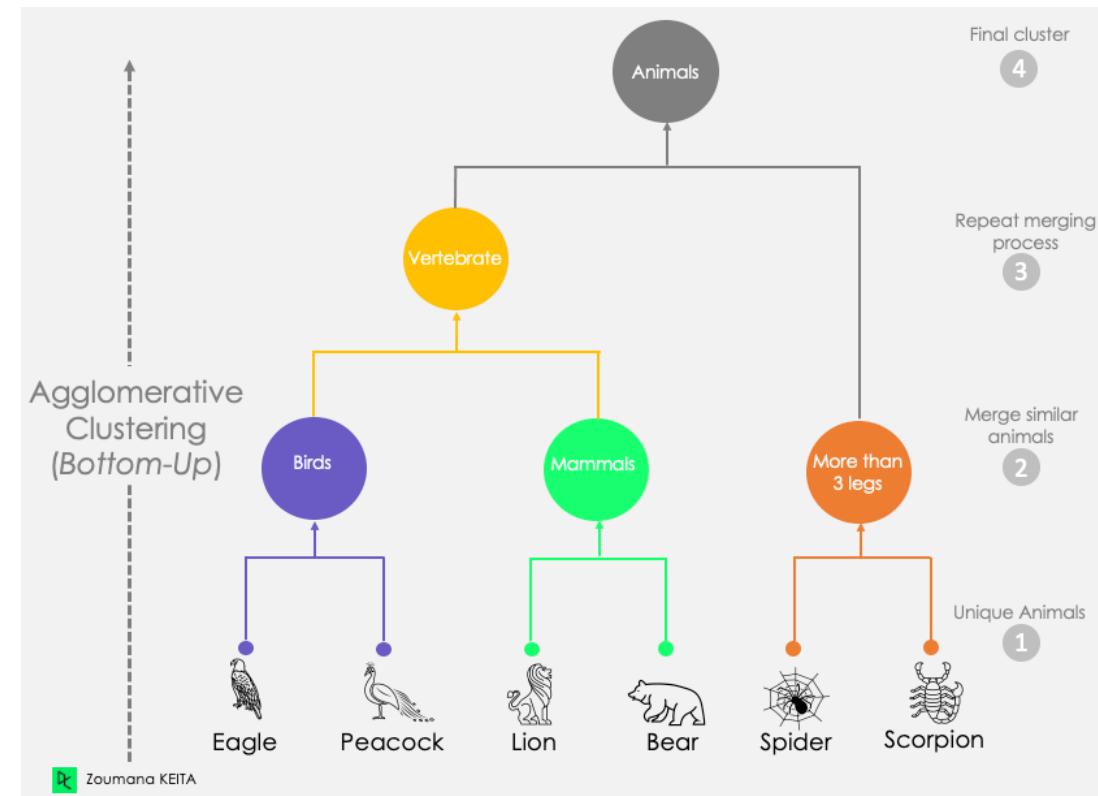
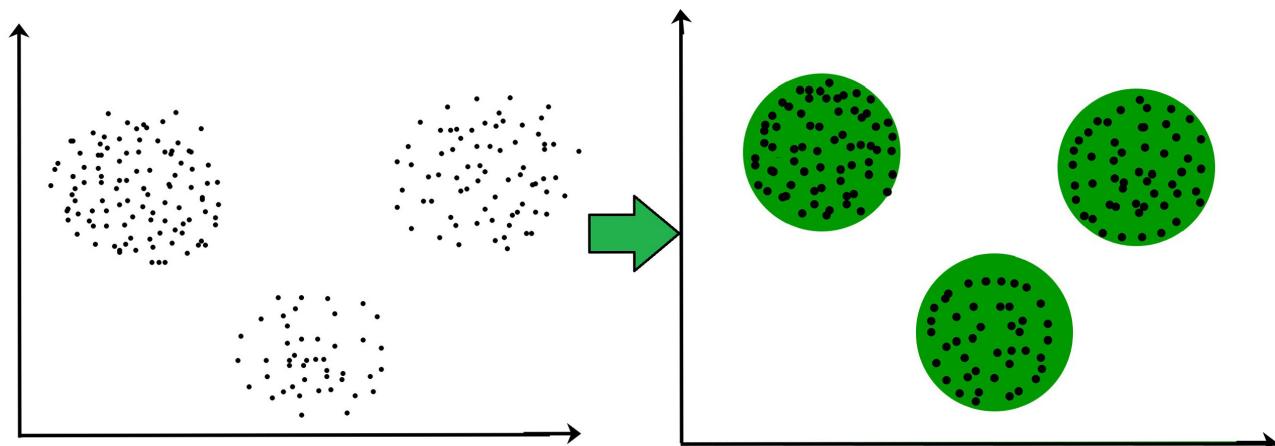
- Aqui lidamos com datasets onde não possuímos “rótulos”. Ou seja: perdemos qualquer ponto de referência para julgar a qualidade dos modelos → Estatística “hardcore”
- Clustering K-means



Fonte: DeepLearningBook.com.br

Introdução ao Machine Learning

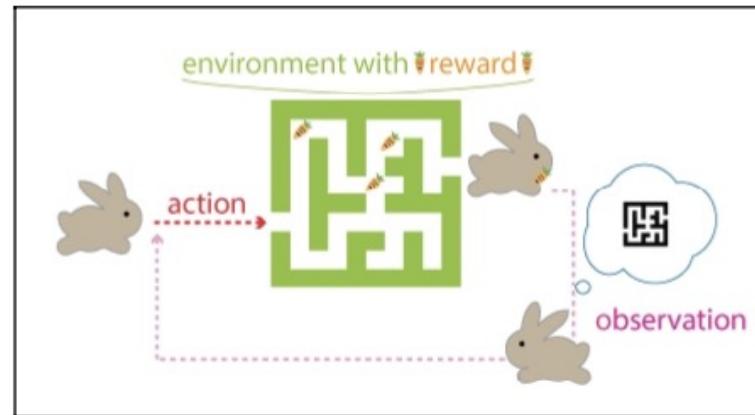
Tipos de Aprendizagem de Máquina



Introdução ao Machine Learning

Tipos de Aprendizagem de Máquina

Tipos de Aprendizagem de Máquina - Reforço



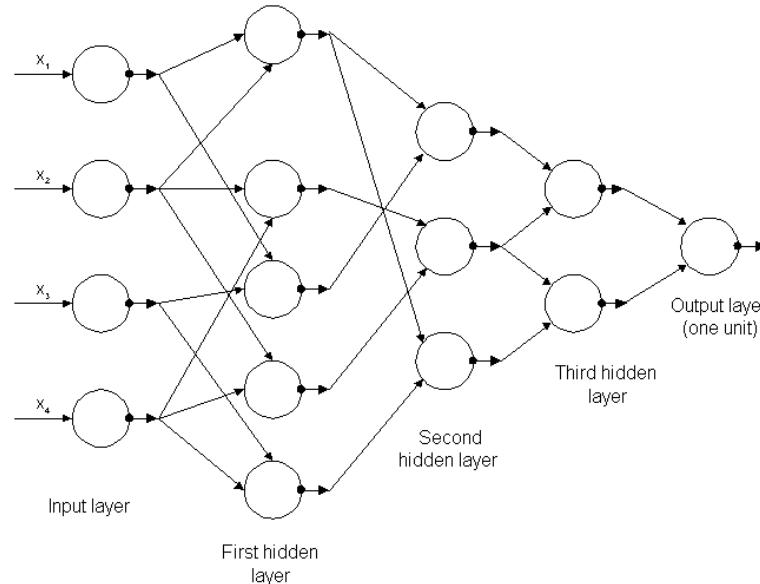
Fonte: Python Deep Learning - Exploring Deep Learning techniques, neural network architectures and GANs with PyTorch, Keras de Ivan Vasilev, Daniel Slater

Introdução ao Machine Learning

Aquecendo os Motores - Papel das Redes Neurais na Aprendizagem Supervisionada

O que torna possível o aprendizado por meio das redes neurais?

- RNAs modelam funções complexas, relacionando entradas (inputs) à saídas (labels).
- O processo de treinamento é, à grosso modo, encontrar os pesos a serem utilizados.





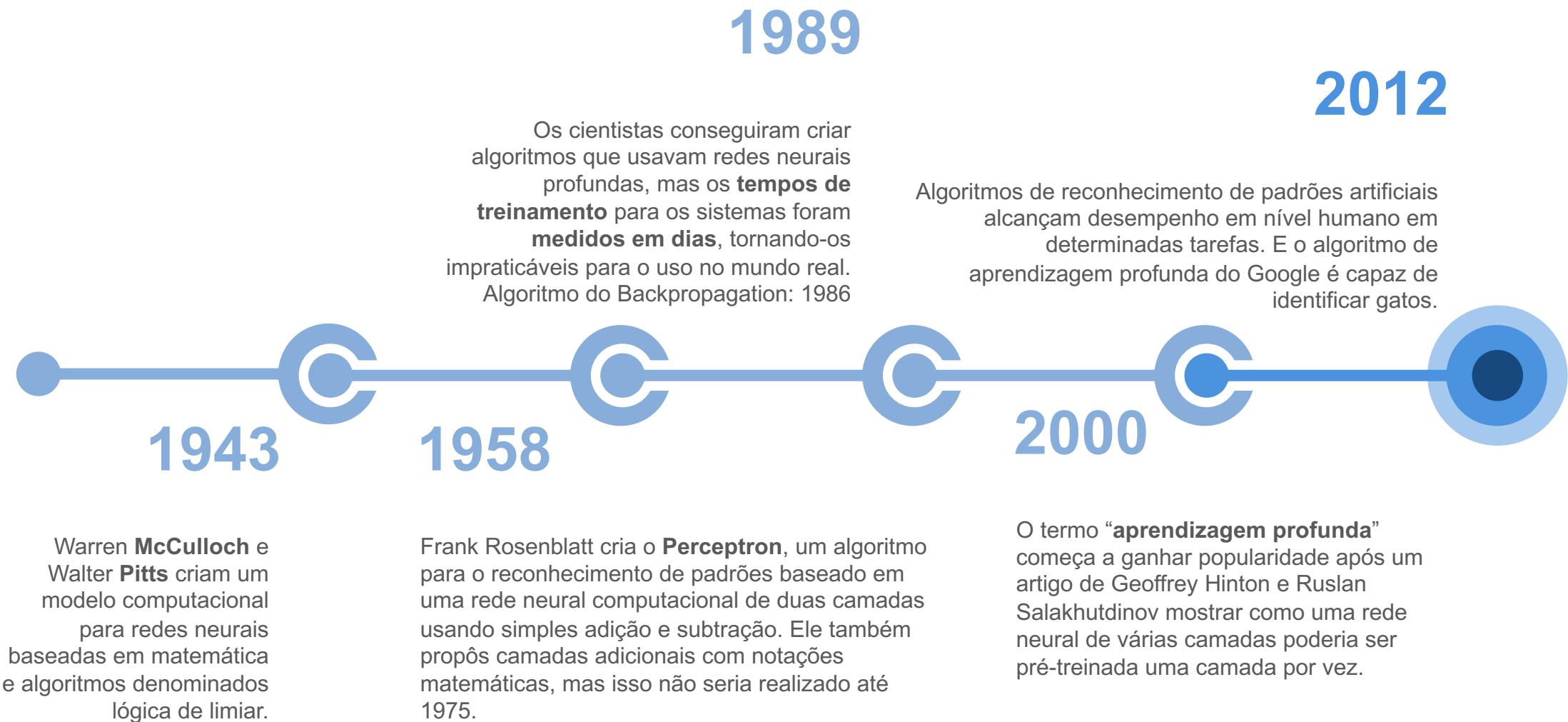
Neurônios Artificiais e Perceptrons

Neurônios Artificiais e Perceptrons

Objetivos do Módulo:

- Explicar sucintamente o funcionamento de um neurônio biológico
- Compreender o modelo de neurônio artificial.
- Explorar os conceitos e funcionalidades dos perceptrons.
- Aprender sobre funções de ativação e suas aplicações em redes neurais.

Histórico das Redes Neurais Artificiais



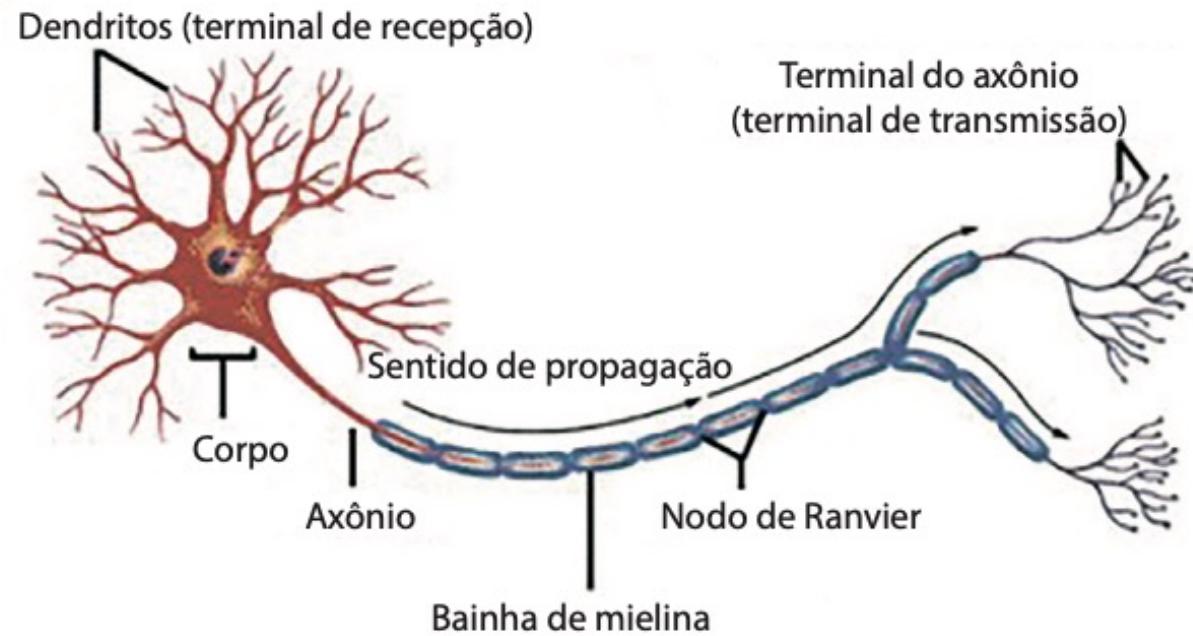
Neurônios Artificiais e Perceptrons

Neurônios

- Sequência Lógica: Neurônio Biológico → Neurônio Artificial de McCulloch-Pitts
→ Neurônio Artificial (Perceptron) de Rosenblatt → Redes Neurais Artificiais mais complexas, como MLP, CNN, etc.

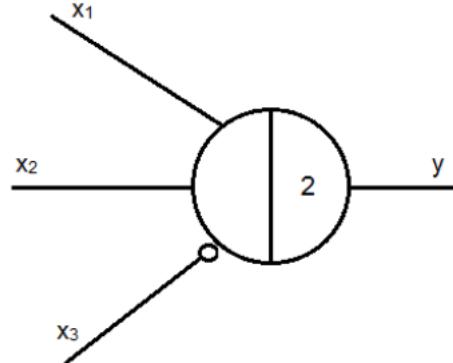
Neurônios Artificiais e Perceptrons

Neurônios Biológicos



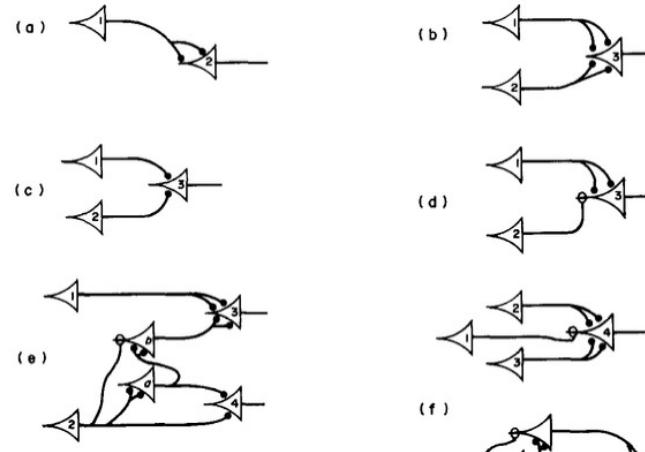
Neurônios Artificiais e Perceptrons

Neurônio Artificial de McCulloch-Pitts (1943)



x ₁	x ₂	x ₃	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

LOGICAL CALCULUS FOR NERVOUS ACTIVITY 105



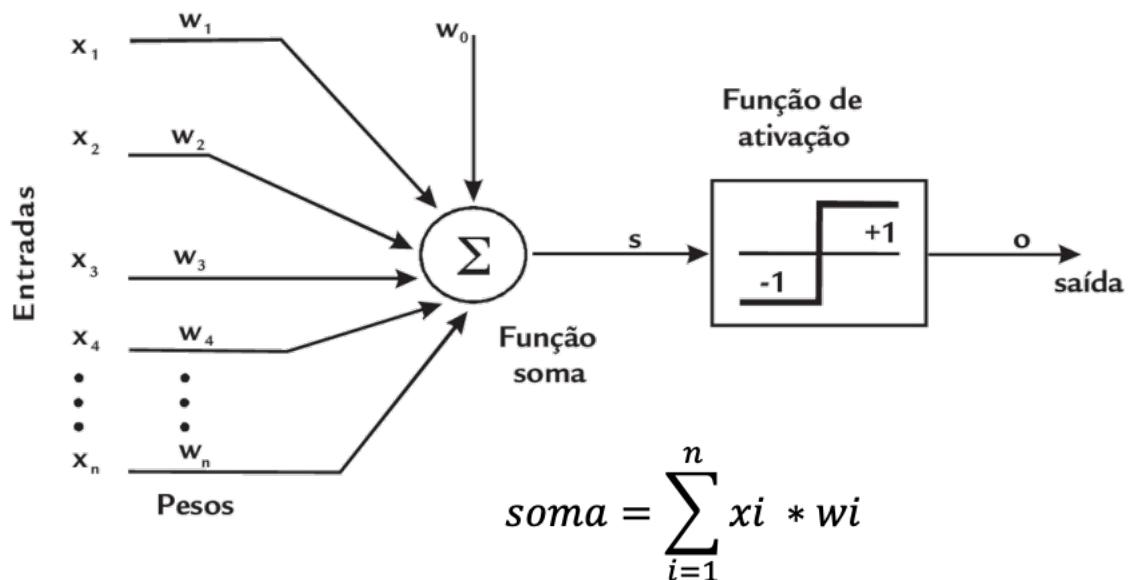
Definição de Neurônio Artificial: Um neurônio artificial é uma unidade computacional que simula o funcionamento de um neurônio biológico.

- O modelo de neurônio artificial de McCulloch-Pitts foi um dos primeiros modelos formais de neurônio artificial;
- Simplificação do neurônio biológico;
- Não há backpropagation;
- Uma combinação de neurônios pode simular uma porta lógica.

Fonte: Notas de Aula do Curso “Tópicos em Sistemas Inteligentes II”, Prof. Levy Boccato (UNICAMP)
MCCULLOCH, W., PITTS, W., “A Logical Calculus of the Ideas Immanent in Nervous Activity”

Neurônios Artificiais e Perceptrons

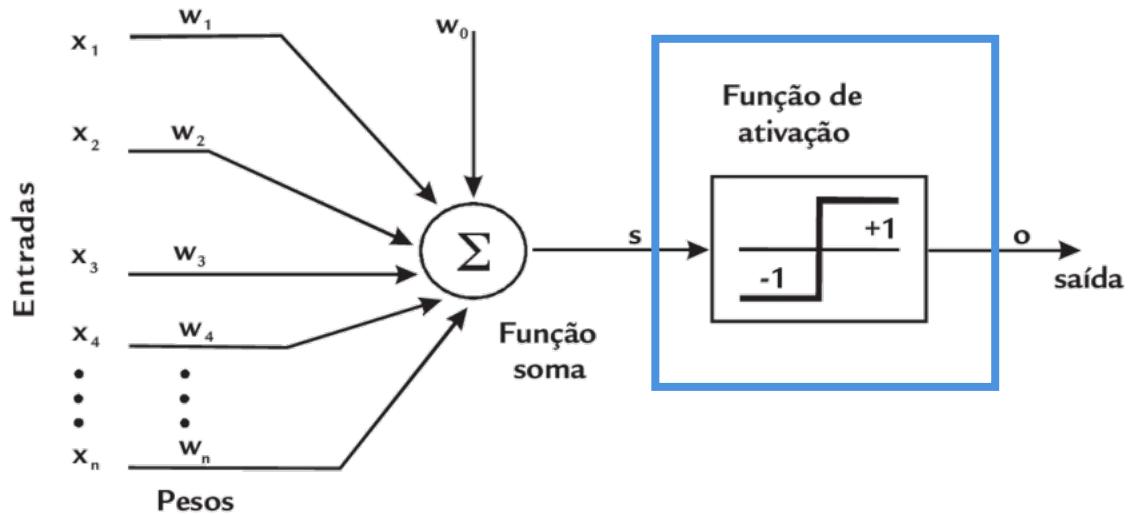
Neurônio Artificial (Perceptron) de Rosenblatt (1958)



- Modelo de neurônio mais avançado;
- Possui entradas ponderadas por pesos e uma função de ativação;
- Capacidade de aprendizado através do ajuste dos pesos;
- Perceptron → Rede Neural de Camada Única. Limitado a resolver problemas linearmente separáveis.
- Perceptron de várias camadas → Rede Neural Artificial. Para problemas mais complexos (não-linearmente separáveis)
- É a unidade básica de processamento em uma Rede Neural

Neurônios Artificiais e Perceptrons

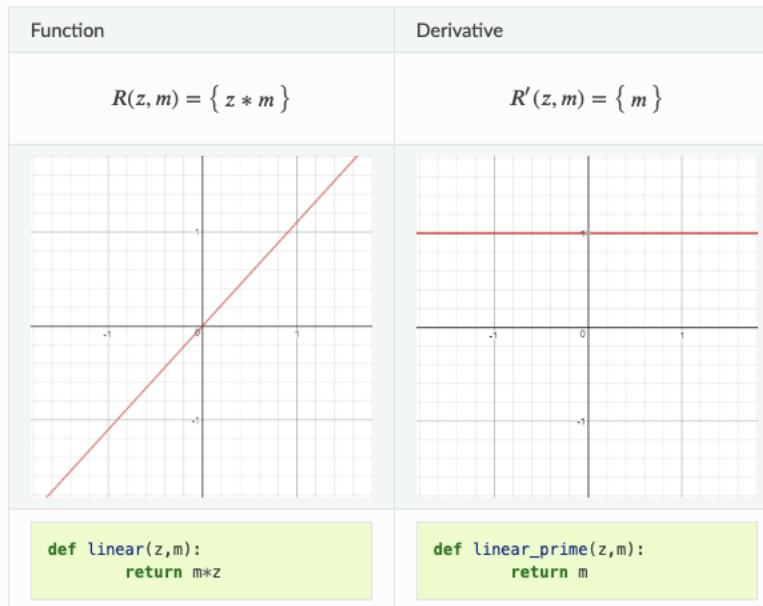
Função de Ativação



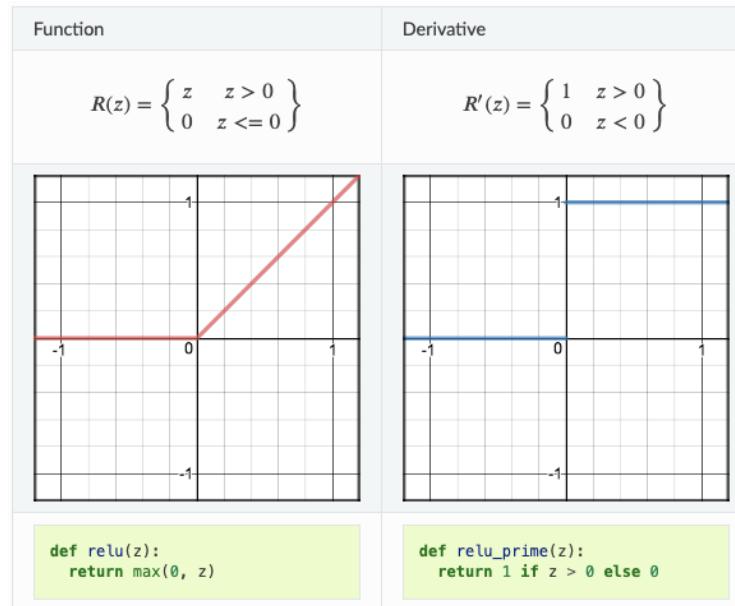
- A Função de Ativação é responsável pela decisão de ativar ou não um neurônio;
- Recebe $x_1*w_1+...+x_n*w_n+w_0$ (bias) como entrada;
- Principais Funções de Ativação: Step function, ReLU, Sigmóide, Tangente Hiperbólica, Softmax;
- Podem ser lineares ou não-lineares;

Neurônios Artificiais e Perceptrons

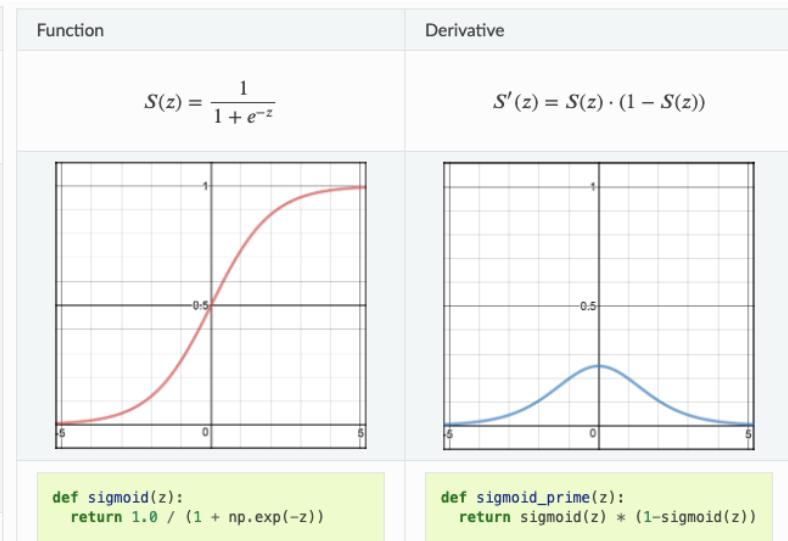
Função de Ativação



Função de Ativação: **Linear**



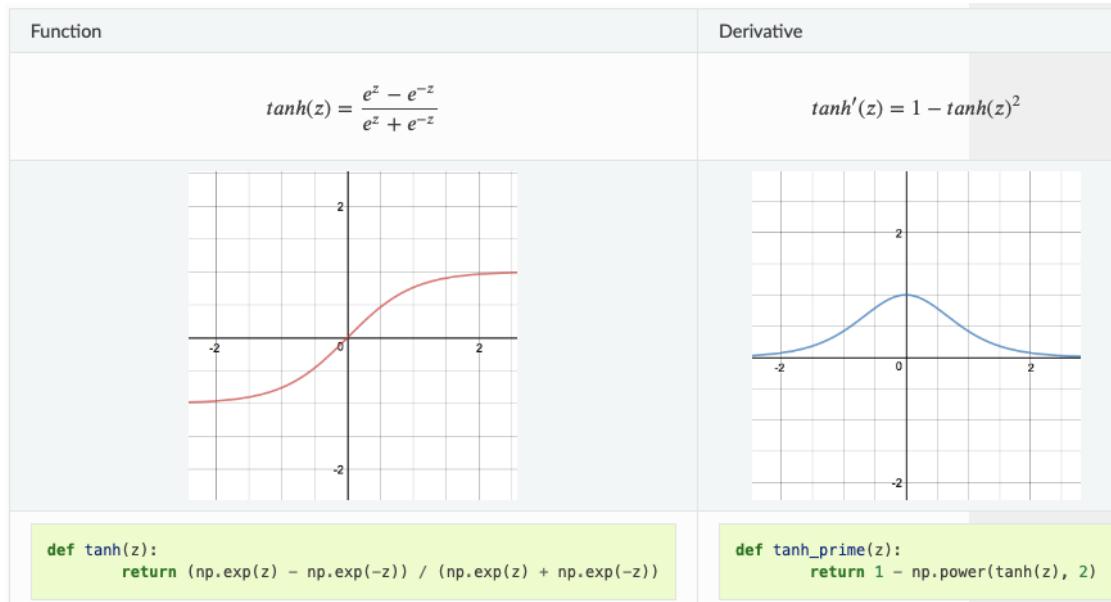
Função de Ativação: **ReLU**



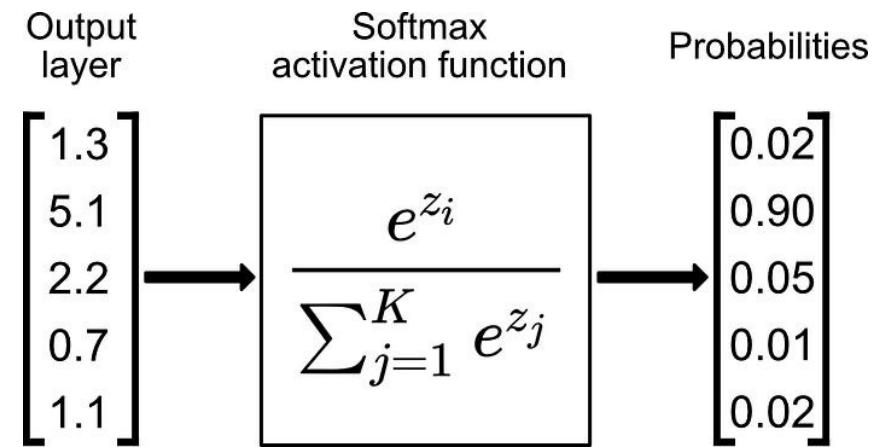
Função de Ativação: **Sigmóide**

Neurônios Artificiais e Perceptrons

Função de Ativação



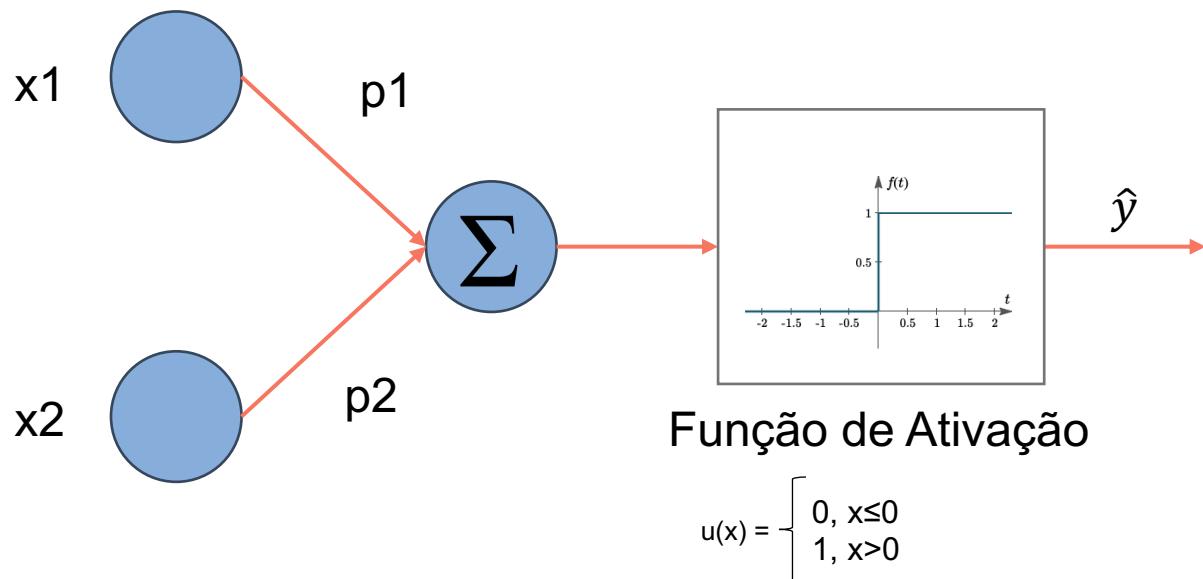
Função de Ativação: **Tangente Hiperbólica**



Função de Ativação: **Softmax**
- $e^{1.3} + e^{5.1} + \dots + e^{1.1} \approx 181.73$
- $\frac{e^{1.3}}{181.73} = 3.67 / 181.73 = 0.02$

Neurônios Artificiais e Perceptrons

Funcionamento de um Perceptron



Arquitetura:
2 inputs
Perceptron único

Função de Ativação:
Step function/Heaviside

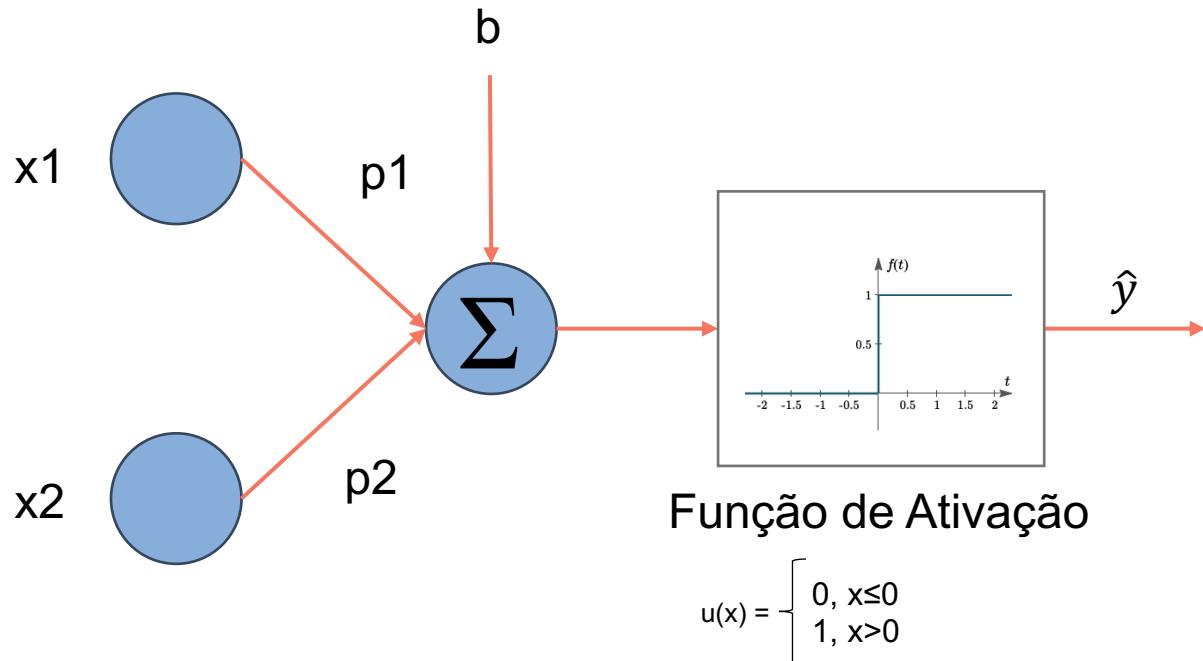
Inicialização dos pesos:
1) $p_1 = 0.0, p_2 = 0.0$
2) $p_1 = 0.5, p_2 = 0.5$

Porta Lógica OR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Neurônios Artificiais e Perceptrons

Funcionamento de um Perceptron – Inserção do Bias



Arquitetura:
2 inputs
Perceptron único
Bias

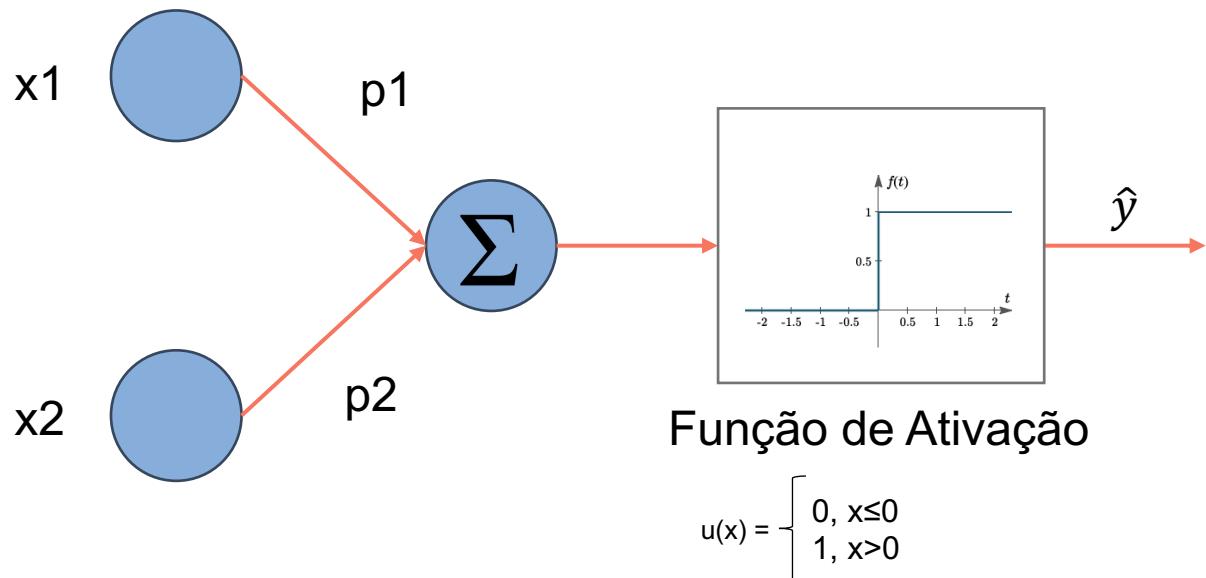
Função de Ativação:
Step function/Heaviside

Inicialização dos pesos e bias:
1) $p_1 = 0.0$, $p_2 = 0.0$, $b=1$
2) $p_1 = 0.5$, $p_2 = 0.5$, $b=0$

Porta Lógica OR		
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Neurônios Artificiais e Perceptrons

Funcionamento de um Perceptron – Atualização de Pesos (sem bias)



Arquitetura:
2 inputs
Perceptron único

Função de Ativação:
Step function/Heaviside

Inicialização dos pesos:
1) $p_1 = 0.0$, $p_2 = 0.0$
2) $p_1 = 0.5$, $p_2 = 0.5$

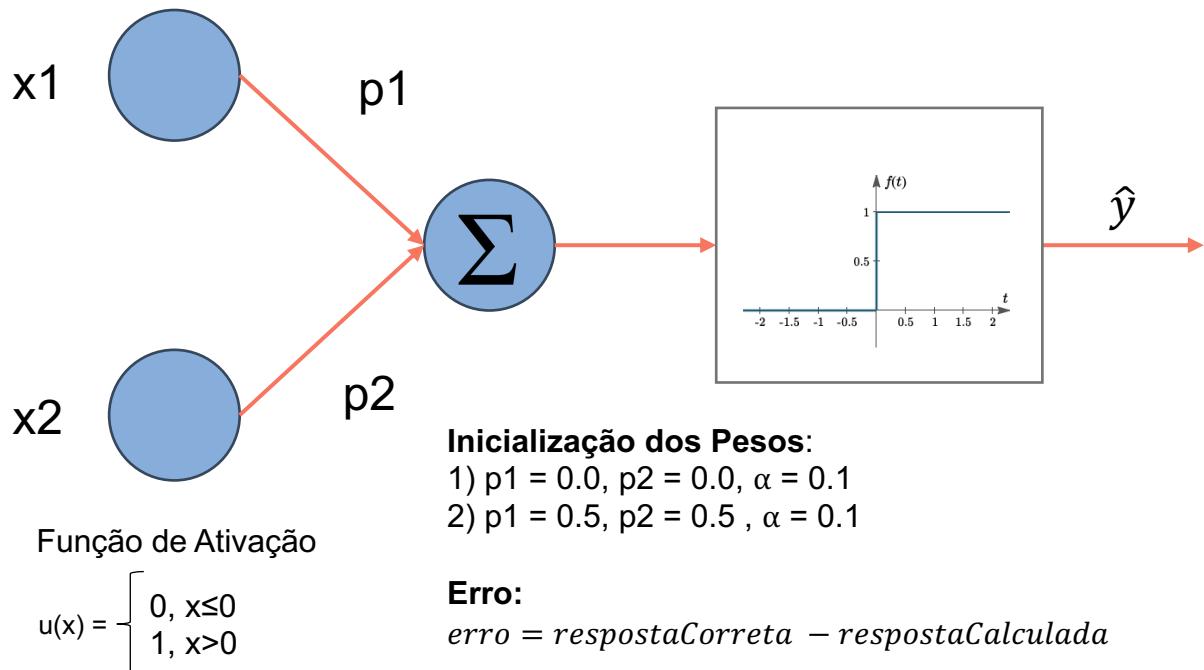
Erro:
 $erro = respostaCorreta - respostaCalculada$

Atualização dos Pesos:
 $peso_{n+1} = peso_n + taxaAprendizagem * entrada * erro$

Porta Lógica OR		
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Neurônios Artificiais e Perceptrons

Funcionamento de um Perceptron – Atualização de Pesos



Inicialização dos Pesos:

- 1) $p_1 = 0.0, p_2 = 0.0, \alpha = 0.1$
- 2) $p_1 = 0.5, p_2 = 0.5, \alpha = 0.1$

Erro:

$$\text{erro} = \text{respostaCorreta} - \text{respostaCalculada}$$

Atualização dos Pesos:

$$\text{peso}_{n+1} = \text{peso}_n + (\text{taxaAprendizagem} * \text{entrada} * \text{erro})$$

ALGORITMO

1. Inicializar perceptron: pesos, bias, taxa de aprendizagem
2. Proceder com o cálculo da rede
3. Calcular o erro
4. Erro é menor que o critério de parada?
 1. Não: Ajustar os pesos
 2. Sim: Fim

Porta Lógica OR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Neurônios Artificiais e Perceptrons

Funcionamento de um Perceptron – Atualização de Pesos

1) $p_1 = 0.0, p_2 = 0.0, \alpha = 0.1$

x1	x2	x1*p1	x2*p2	Σ	y	f(Σ) = \hat{y}	e = y - \hat{y}
0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	1
1	0	0	0	0	1	0	1
1	1	0	0	0	1	0	1

Ajuste dos Pesos para $x_1=1$ e $x_2=1$

p1:

$$\begin{aligned} peso_{n+1} &= peso_n + (taxaAprendizagem * entrada * erro) \\ peso_{n+1} &= 0.0 + (0.1 * 1 * 1) = 0.1 \end{aligned}$$

p2:

$$\begin{aligned} peso_{n+1} &= peso_n + (taxaAprendizagem * entrada * erro) \\ peso_{n+1} &= 0.0 + (0.1 * 1 * 1) = 0.1 \end{aligned}$$

Porta Lógica OR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Neurônios Artificiais e Perceptrons

Funcionamento de um Perceptron – Atualização de Pesos

2) $p_1 = 0.1, p_2 = 0.1, \alpha = 0.1$

x1	x2	x1*p1	x2*p2	Σ	y	f(Σ) = \hat{y}	e = y - \hat{y}
0	0	0	0	0	0	0	0
0	1	0	0.1	0.1	1	1	0
1	0	0.1	0	0.1	1	1	0
1	1	0.1	0.1	0.2	1	1	0

Para $p_1 = 0.0$ e $p_2 = 0.0$, tínhamos um acerto de 25% (1 classe prevista corretamente)

Para $p_1 = 0.1$ e $p_2 = 0.1$, temos um acerto de 100% (todas as classes previstas corretamente) → overfitting?

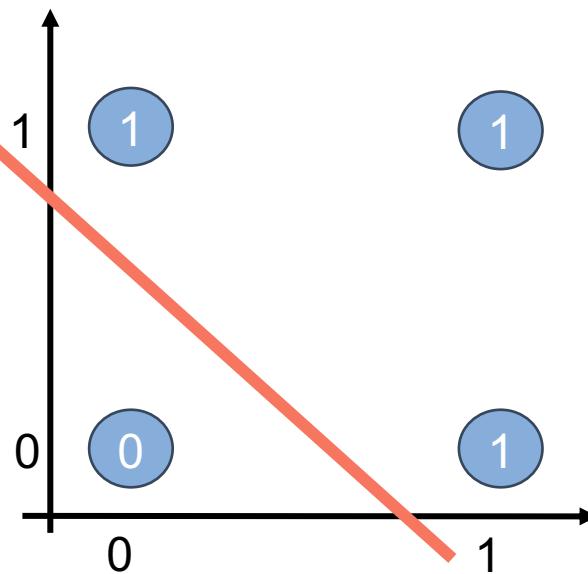
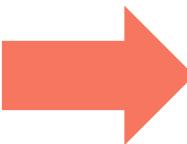
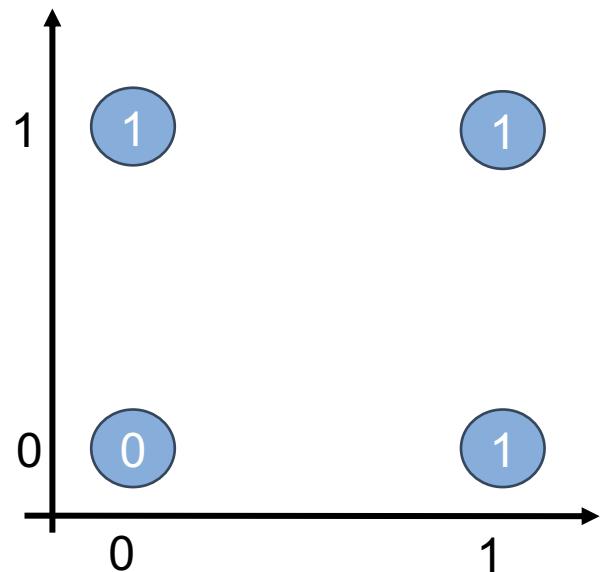
Porta Lógica OR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Neurônios Artificiais e Perceptrons

Aplicações de um Perceptron de Camada Única

Solução de Problemas Linearmente Separáveis



Porta Lógica OR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Neurônios Artificiais e Perceptrons

Aplicações de um Perceptron de Camada Única

Outras aplicações:

- Modelagem de Portas Lógicas (AND, OR, NOT). XOR é um problema não-linearmente separável
- Classificação Binária → É spam ou não é spam? Doente ou Sadio?
- Entre outras.

Neurônios Artificiais e Perceptrons

Problema Proposto

Suponha que você está trabalhando em um projeto de automação em um banco e precisa classificar automaticamente moedas com base em seu diâmetro (cm) e espessura (mm) para separar as moedas de prata, das moedas de ouro. Você coletou dados de treinamento de moedas de prata e ouro com as seguintes características:

- Diâmetro (em centímetros)
- Espessura (em milímetros)

Seu objetivo é criar um perceptron simples que possa determinar se uma moeda é de ouro (classe 0) ou uma prata (classe 1) com base nessas duas características.

Neurônios Artificiais e Perceptrons

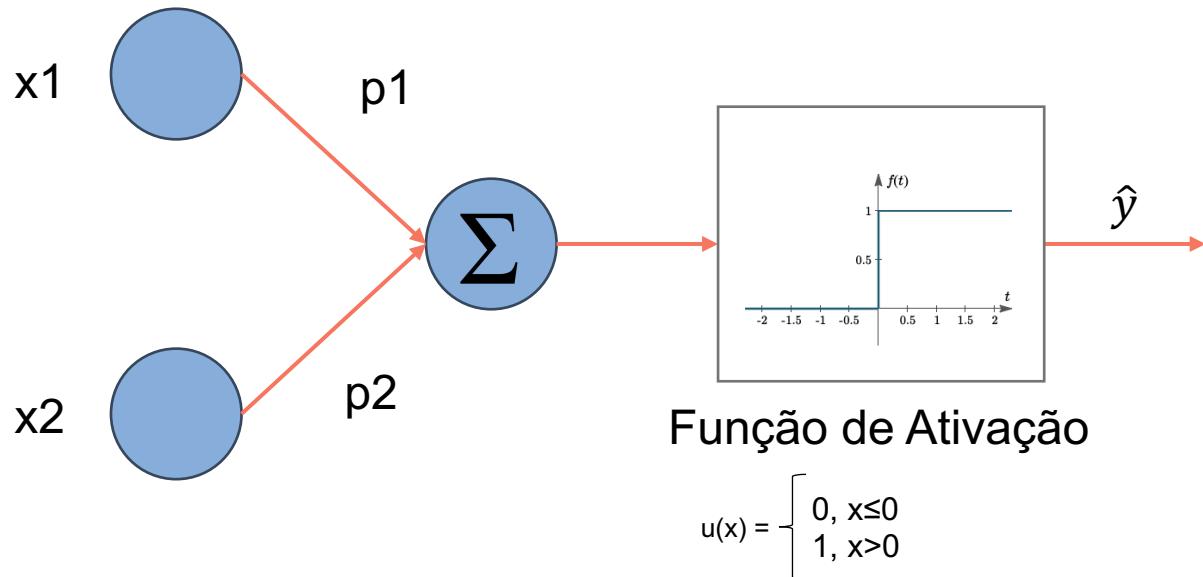
Problema Proposto

FIRST THINGS, FIRST



Neurônios Artificiais e Perceptrons

Problema Proposto



Arquitetura:

2 inputs → diâmetro e espessura
Perceptron único → classificador binário

Função de Ativação:

Step function/Heaviside

Inicialização dos pesos:

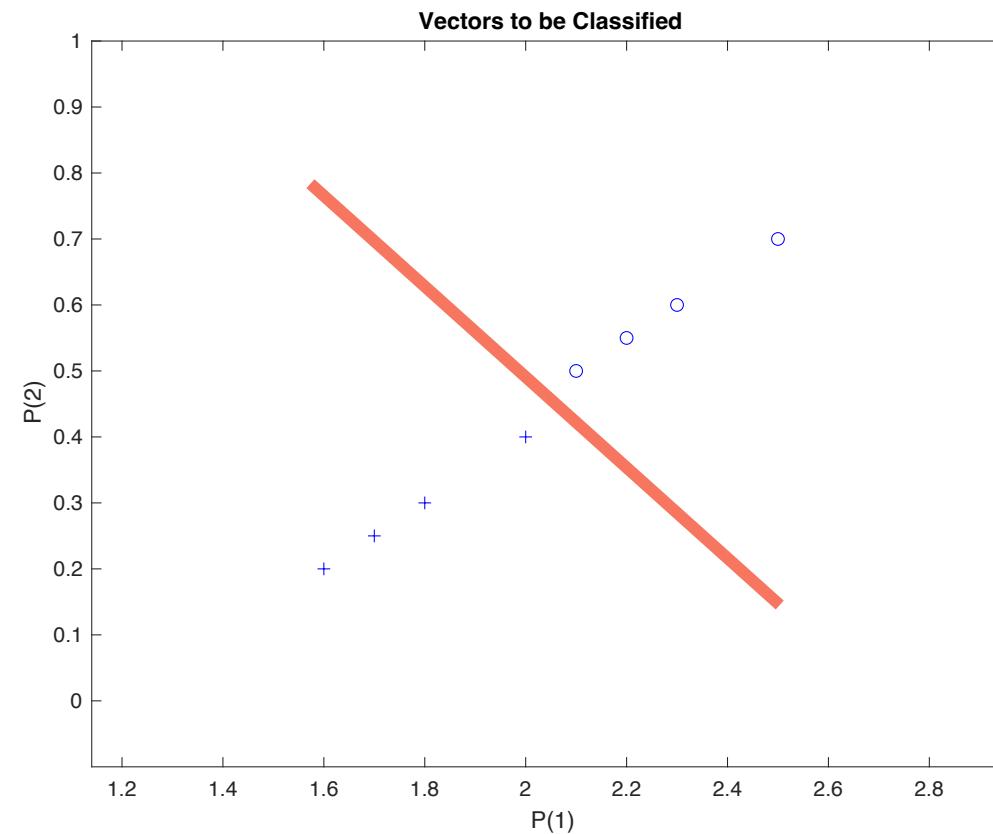
1) $p_1 = 1$, $p_2 = 1$, $\alpha = 0.1$

Neurônios Artificiais e Perceptrons

Problema Proposto – Coleta dos Dados

Diâmetro (cm)	Espessura (mm)	Classe
2.1	0.5	0
1.8	0.3	1
2.5	0.7	0
1.6	0.2	1
2.3	0.6	0
2.0	0.4	1
2.2	0.55	0
1.7	0.25	1

Classe 0: Moeda de Ouro
Classe 1: Moeda de Prata



Neurônios Artificiais e Perceptrons

$p_1 = 1$
 $p_2 = 1$
 $\alpha = 0.1$

Problema Proposto – 1^a iteração

x1	x2	x1*p1	x2*p2	Σ	y	f(Σ) = \hat{y}	e (y - \hat{y})
2.1	0.5	2.1	0.5	2.6	0	1	-1
1.8	0.3	1.8	0.3	2.1	1	1	0
2.5	0.7	2.5	0.7	3.2	0	1	-1
1.6	0.2	1.6	0.2	1.8	1	1	0
2.3	0.6	2.3	0.6	2.9	0	1	-1
2.0	0.4	2.0	0.4	2.4	1	1	0
2.2	0.55	2.2	0.55	2.75	0	1	-1
1.7	0.25	1.7	0.25	1.95	1	1	0

Diâmetro (cm)	Espessura (mm)	Classe
2.1	0.5	0
1.8	0.3	1
2.5	0.7	0
1.6	0.2	1
2.3	0.6	0
2.0	0.4	1
2.2	0.55	0
1.7	0.25	1



Neurônios Artificiais e Perceptrons

Problema Proposto – 1^a iteração – Ajuste dos Pesos

x1	x2	x1*p1	x2*p2	Σ	y	f(Σ) = \hat{y}	e (y - \hat{y})
2.1	0.5	2.1	0.5	2.6	0	1	-1
1.8	0.3	1.8	0.3	2.1	1	1	0
2.5	0.7	2.5	0.7	3.2	0	1	-1
1.6	0.2	1.6	0.2	1.8	1	1	0
2.3	0.6	2.3	0.6	2.9	0	1	-1
2.0	0.4	2.0	0.4	2.4	1	1	0
2.2	0.55	2.2	0.55	2.75	0	1	-1
1.7	0.25	1.7	0.25	1.95	1	1	0

Ajuste dos Pesos para x1= e x2= p1:

$$\begin{aligned} peso_{n+1} &= peso_n + (taxaAprendizagem * entrada * erro) \\ peso_{n+1} &= 1 + (0.1 * 2.5 * -1) = -0.75 \end{aligned}$$

p2:

$$\begin{aligned} peso_{n+1} &= peso_n + (taxaAprendizagem * entrada * erro) \\ peso_{n+1} &= 1 + (0.1 * 0.7 * -1) = 0.93 \end{aligned}$$

Neurônios Artificiais e Perceptrons

Problema Proposto

TREINAMENTO NO MATLAB



Neurônios Artificiais e Perceptrons

Problema Proposto – Código Fonte MATLAB

```
x = [2.1 1.8 2.5 1.6 2.3 2.0 2.2 1.7; 0.5 0.3 0.7 0.2 0.6 0.4 0.55 0.25];
t = [0 1 0 1 0 1 0 1];

net = perceptron;

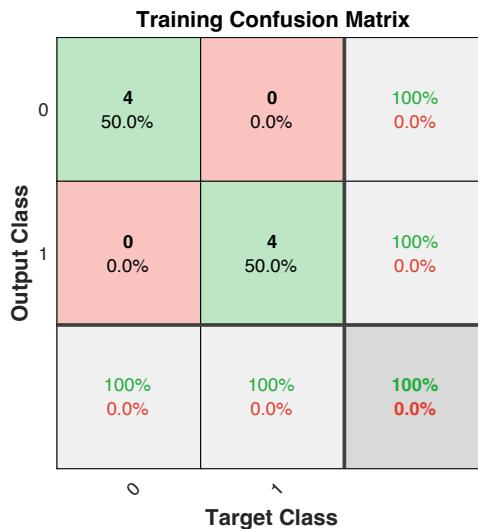
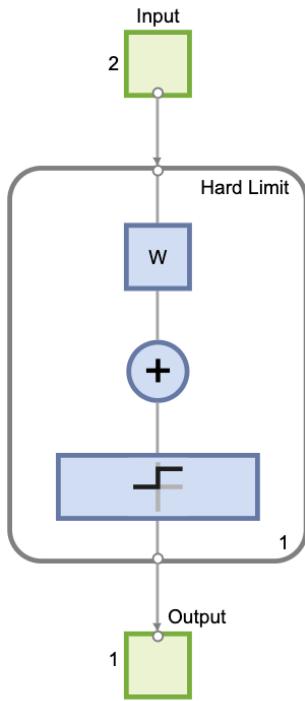
net.trainParam.epochs=1500;
net.trainParam.lr=0.1;
net.trainParam.showCommandLine = true;
net.biasConnect(1) = 0;

net_train = train(net,x,t);

pesos = net_train.iw{1,1}; % Pesos da camada de entrada para a camada de saída
```

Neurônios Artificiais e Perceptrons

Problema Proposto – Resultados MATLAB

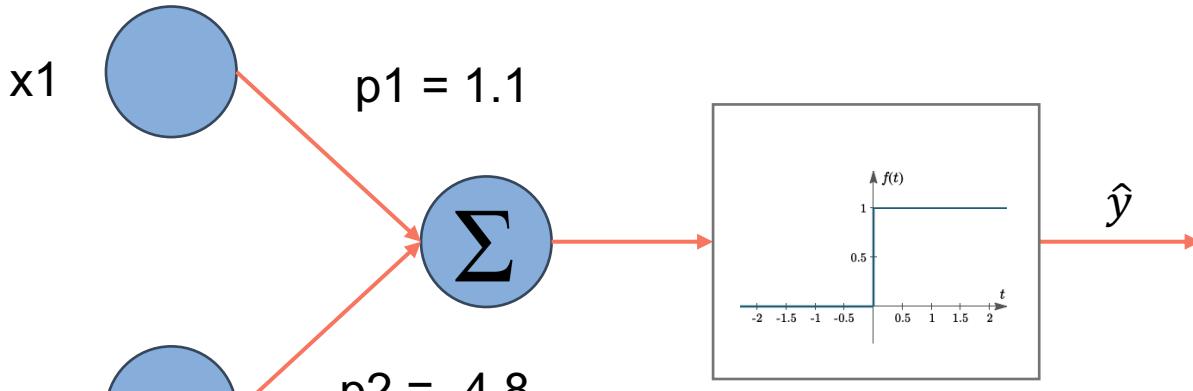


	net
	net_train
	pesos
	t
	x

1x1 network
1x1 network
[1.1000, -4.8000]
[0, 1, 0, 1, 0, 1]
2x8 double

Neurônios Artificiais e Perceptrons

Problema Proposto – Teste da Solução Encontrada



Função de Ativação

$$u(x) = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

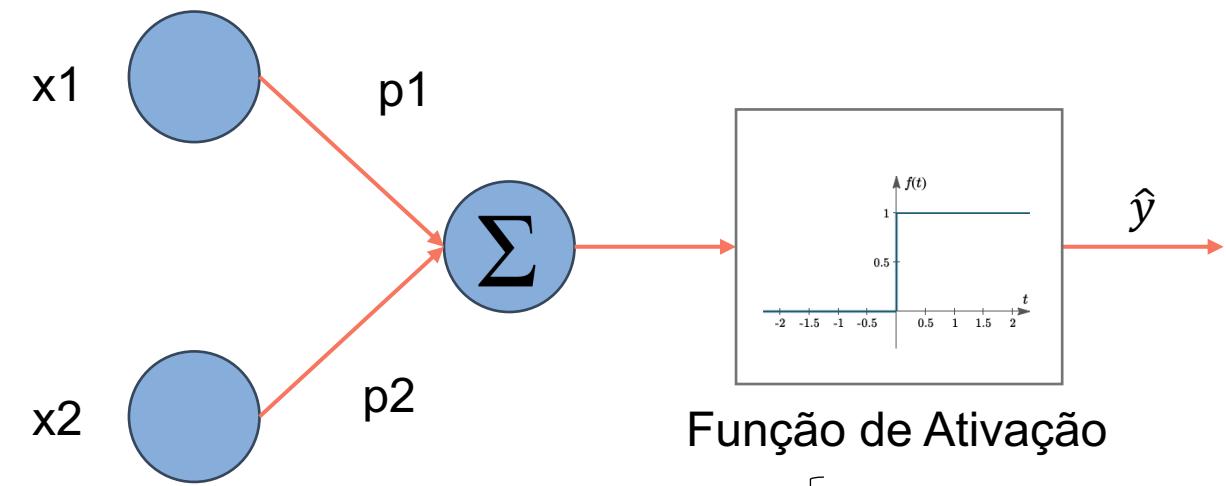
Diâmetro (cm)	Espessura (mm)	Classe
2.1	0.5	0
1.8	0.3	1
2.5	0.7	0
1.6	0.2	1
2.3	0.6	0
2.0	0.4	1
2.2	0.55	0
1.7	0.25	1

Neurônios Artificiais e Perceptrons

Problema Proposto – Invente um Problema Linearmente Separável

Agora é sua vez! Pense em um problema linearmente separável, e iremos codificar um perceptron para identificar automaticamente.

x1	x2	y



Função de Ativação

$$u(x) = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$



Aplicações das Redes Neurais

Aplicações das Redes Neurais

Objetivos do Módulo:

- Explorar as diversas aplicações práticas das redes neurais.
- Compreender o reconhecimento de padrões como uma aplicação-chave.
- Analisar o papel das redes neurais em visão computacional e outros campos de aplicação.

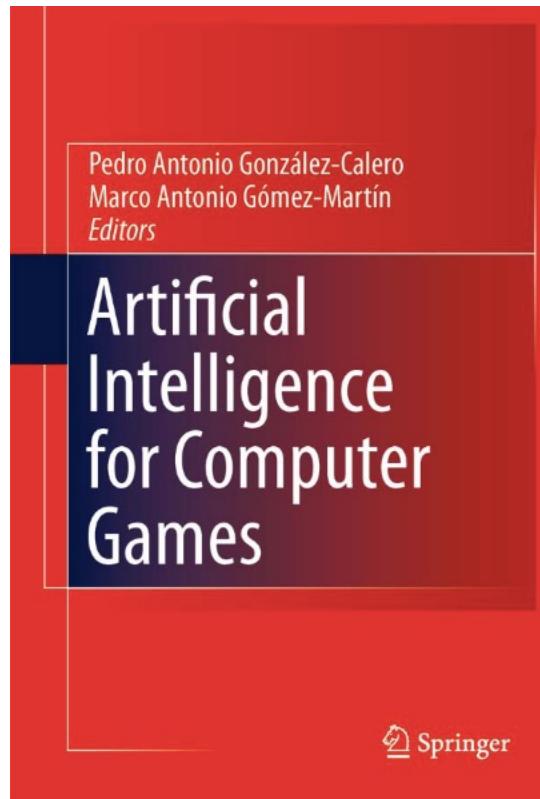
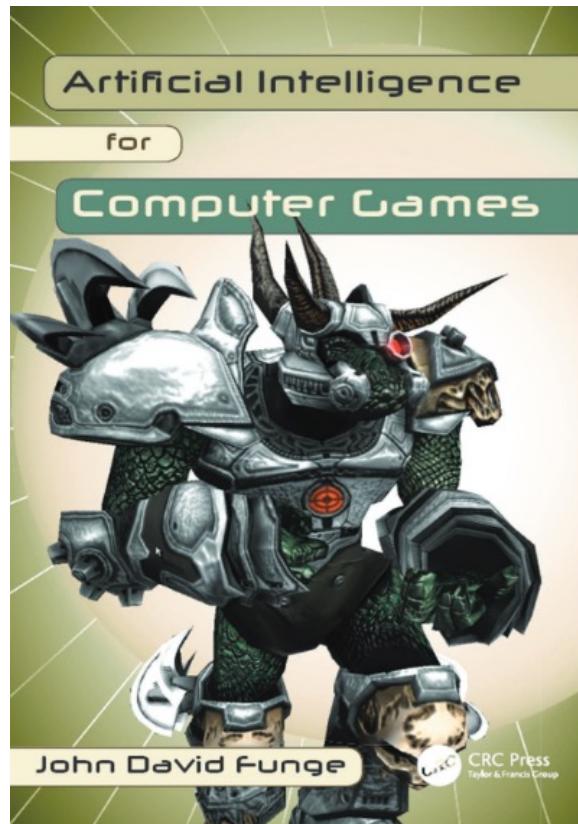
Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais

- Como visto e explicado anteriormente, as Redes Neurais permitem a solução de diversos problemas, destaco:

Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Jogos



- Inteligência Artificial de NPCs → Capacidade de Aprender com as Ações dos Jogadores
- Geração de Conteúdo → Mapas, cenários, personagens, etc → Radiant AI (The Elder Scrolls: Skyrim)
- Detecção de Cheats (reconhecimento de padrões) → BattlEye (Fortnite, Tibia)
- Sistemas de Recomendação em lojas de jogos online → Steam

Aplicações das Redes Neurais

PROCESSAMENTO DE LINGUAGEM NATURAL



Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Processamento de Linguagem Natural



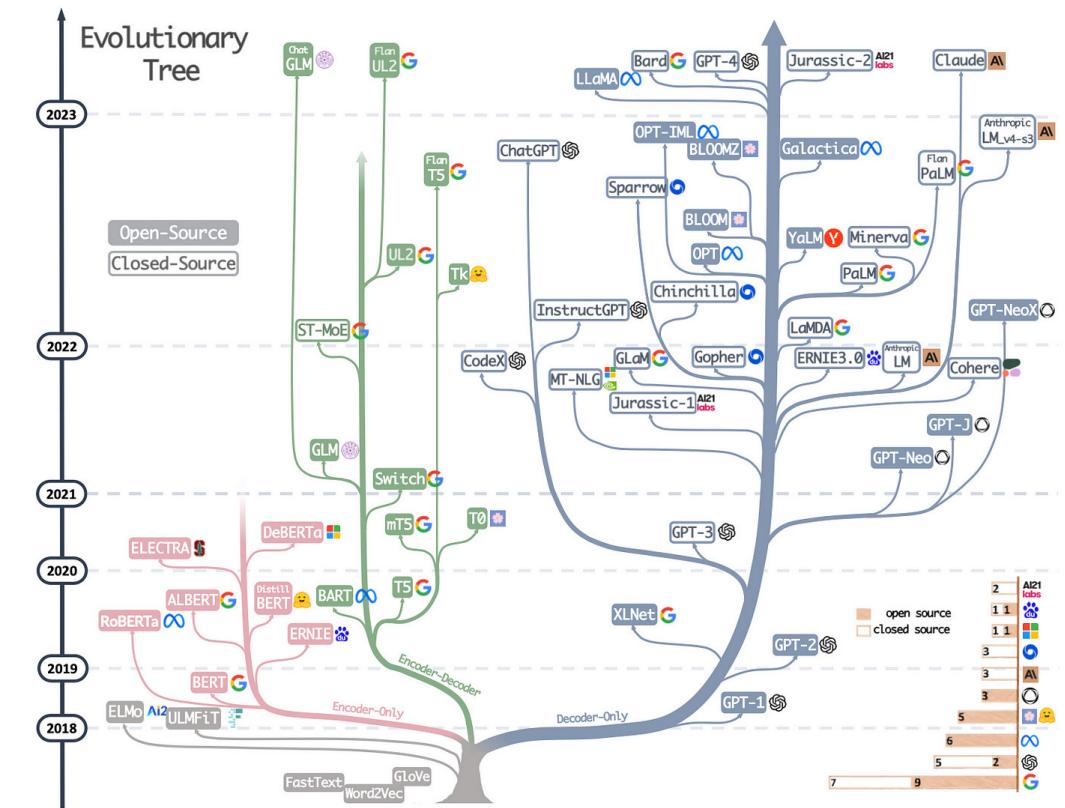
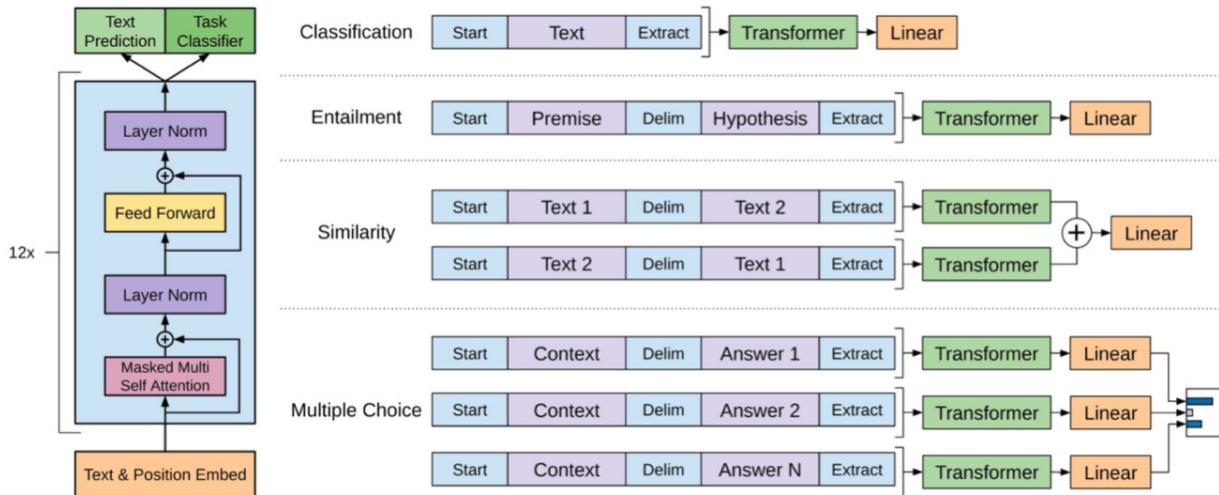
BloombergGPT	March 2023	Bloomberg L.P.	50 billion	Bloomberg's data sources, plus 345 billion tokens from general purpose datasets ^[147]		Proprietary	that "outperforms existing models on financial tasks by significant margins without sacrificing performance on general LLM benchmarks"
PanGu-Σ	March 2023	Huawei	1.085 trillion	329 billion tokens ^[148]		Proprietary	
OpenAssistant ^[149]	March 2023	LAION	17 billion	1.5 trillion tokens		Apache 2.0	Trained on crowdsourced open data
Jurassic-2 ^[150]	March 2023	AI21 Labs	Exact size unknown	Unknown		Proprietary	Multilingual ^[151]
PaLM 2 (Pathways Language Model 2)	May 2023	Google	340 billion ^[152]	3.6 trillion tokens ^[152]	85000 ^[139]	Proprietary	Used in Bard chatbot. ^[153]
Llama 2	July 2023	Meta	70 billion ^[154]	2 trillion tokens ^[154]		Llama 2 license	Successor of LLaMA.
Falcon 180B	September 2023	Technology Innovation Institute	180 billion ^[155]	3.5 trillion tokens ^[155]		Falcon 180B TII license	
Mistral 7B	September 2023	Mistral	7.3 billion ^[156]	Unknown		Apache 2.0	

Exemplos de aplicações dos LLMs: classificação de textos, tradução, geração de texto, resumos automáticos, etc

Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Processamento de Linguagem Natural

Decoder-style GPT model (originally for predictive modeling)



Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Processamento de Linguagem Natural

Decoder-style GPT model (originally for predictive modeling)

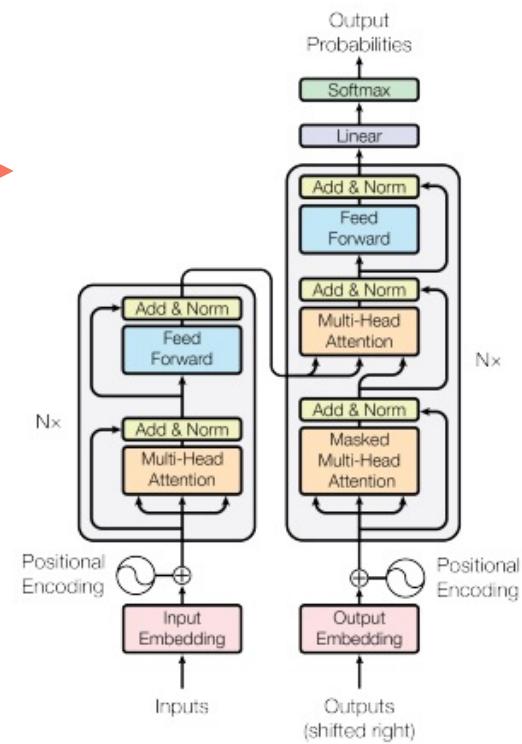
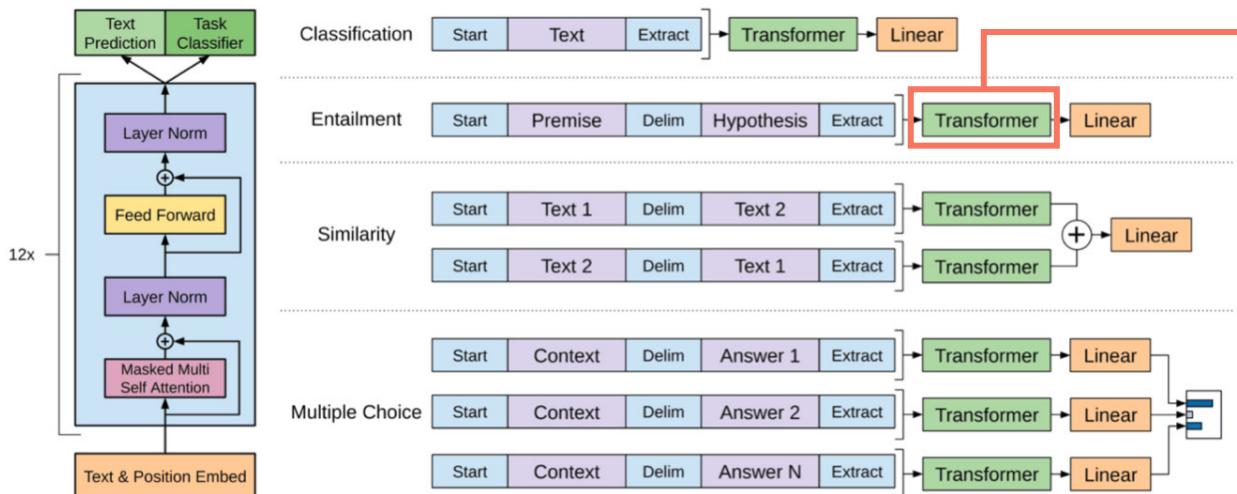
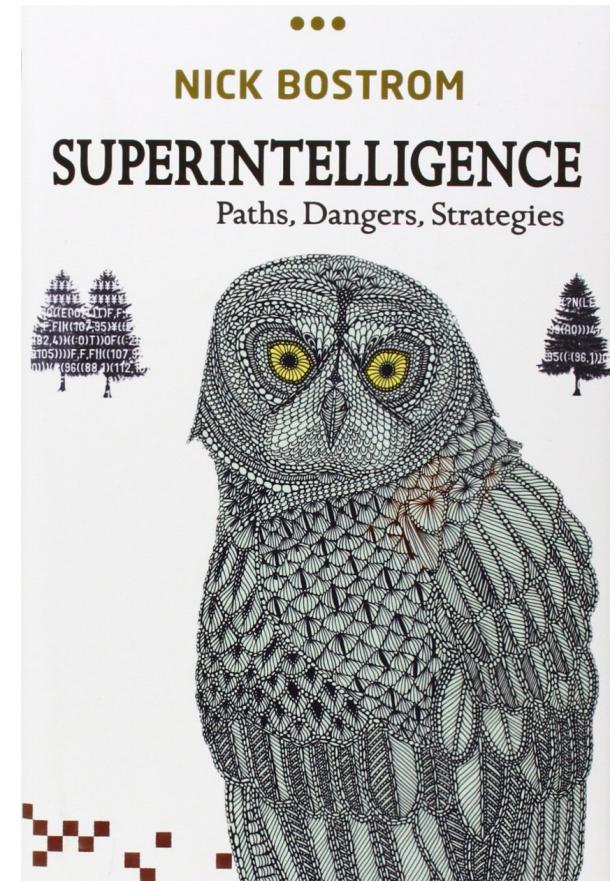


Figure 1: The Transformer - model architecture.

Aplicações das Redes Neurais

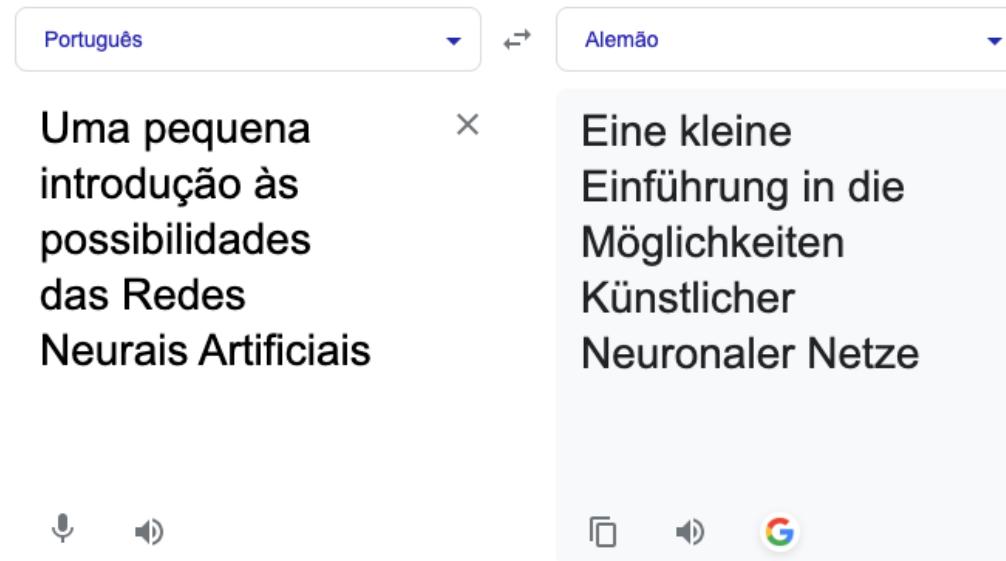
Aplicações das Redes Neurais Artificiais – Processamento de Linguagem Natural

O termo superinteligência, por sua vez, foi definido pelo filósofo sueco Nick Bostrom como “um intelecto que é muito mais inteligente do que o melhor cérebro humano em praticamente todas as áreas, incluindo criatividade científica, conhecimentos gerais e habilidades sociais” (BOSTROM, 2003, p. 12–17).



Aplicações das Redes Neurais

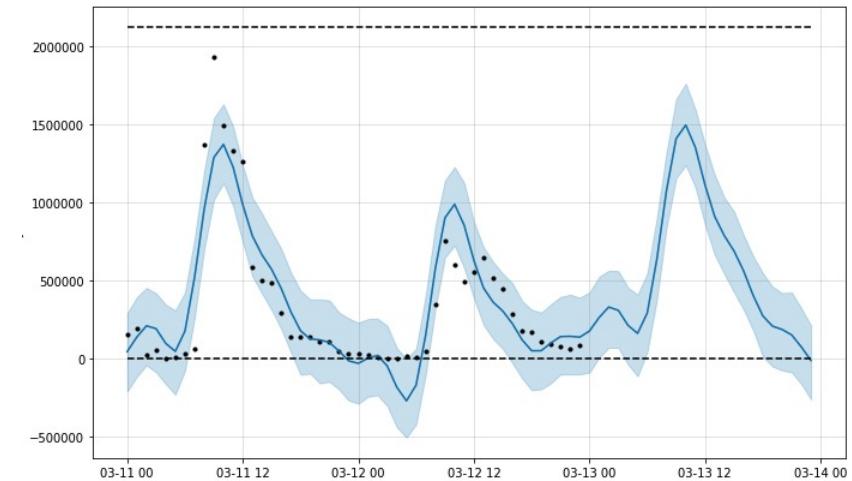
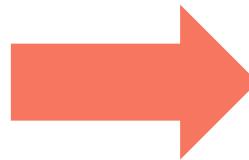
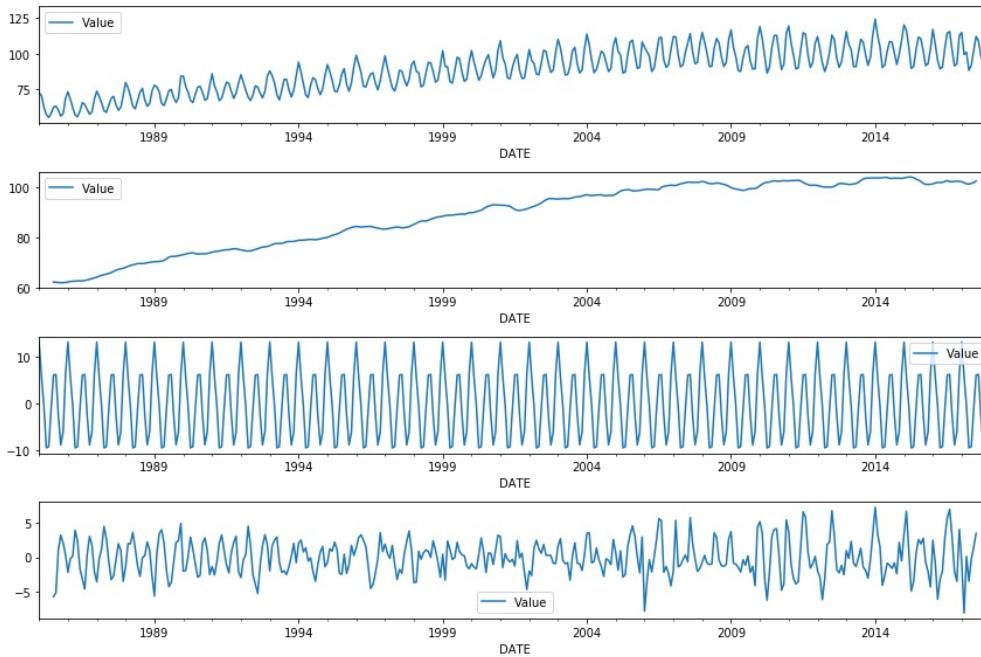
Aplicações das Redes Neurais Artificiais – Tradução Automática



Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Finanças

- Previsão de Séries Temporais (Time Series Forecasting)
 - Série de observações registrada em intervalos de tempo regulares
 - São diferentes em relação aos outros problemas pois há uma dependência em relação ao tempo
 - Componentes: Tendência, Sazonalidade, Padrões Cíclicos

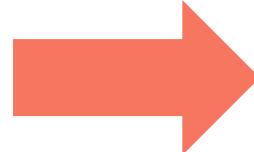


Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Finanças

- Classificação de Crédito

História do crédito	Dívida	Garantias	Renda anual	Risco
Ruim	Alta	Nenhuma	< 15.000	Alto
Desconhecida	Alta	Nenhuma	>= 15.000 a <= 35.000	Alto
Desconhecida	Baixa	Nenhuma	>= 15.000 a <= 35.000	Moderado
Desconhecida	Baixa	Nenhuma	> 35.000	Alto
Desconhecida	Baixa	Nenhuma	> 35.000	Baixo
Desconhecida	Baixa	Adequada	> 35.000	Baixo
Ruim	Baixa	Nenhuma	< 15.000	Alto
Ruim	Baixa	Adequada	> 35.000	Moderado
Boa	Baixa	Nenhuma	> 35.000	Baixo
Boa	Alta	Adequada	> 35.000	Baixo
Boa	Alta	Nenhuma	< 15.000	Alto
Boa	Alta	Nenhuma	>= 15.000 a <= 35.000	Moderado
Boa	Alta	Nenhuma	> 35.000	Baixo
Ruim	Alta	Nenhuma	>= 15.000 a <= 35.000	Alto

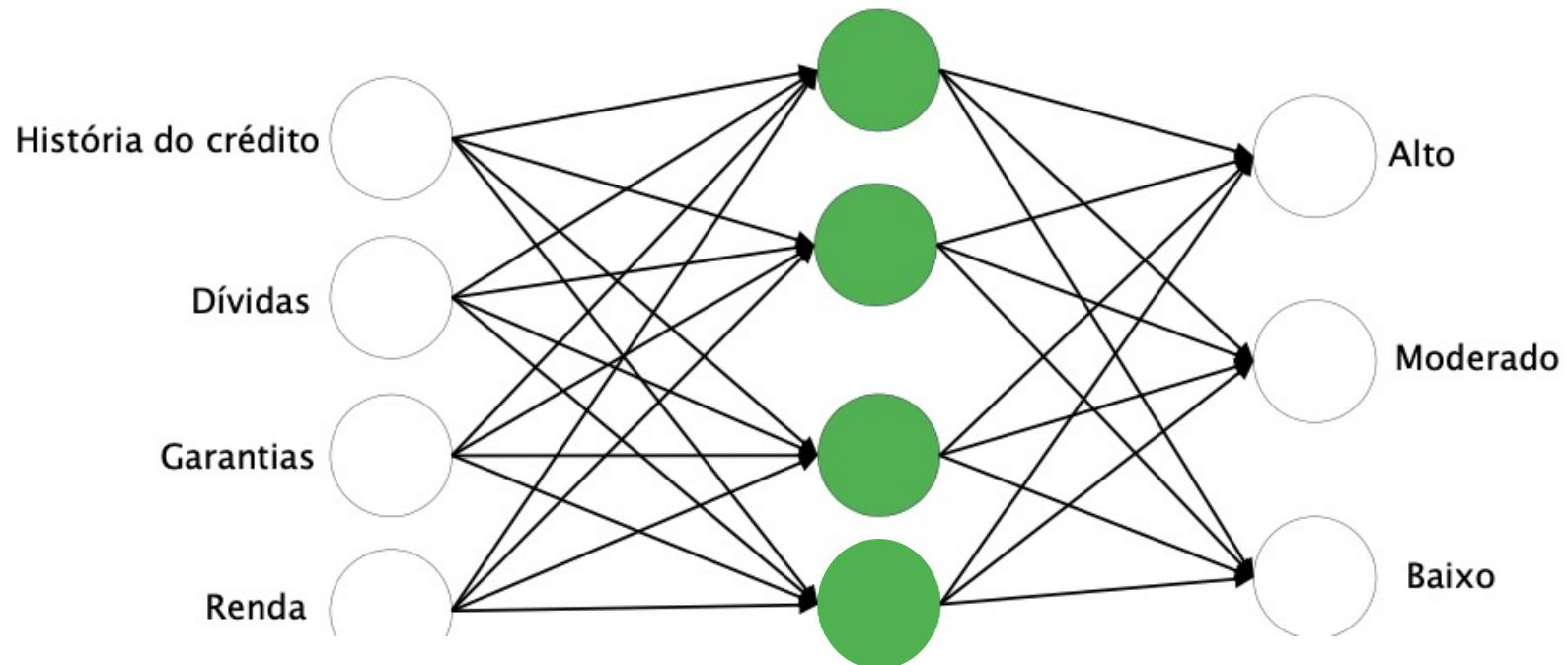


História do crédito	Dívida	Garantias	Renda anual	Risco
3	1	1	1	Alto
2	1	1	2	Alto
2	2	1	2	Moderado
2	2	1	3	Alto
2	2	1	3	Baixo
2	2	2	3	Baixo
3	2	1	1	Alto
3	2	2	3	Moderado
1	2	1	3	Baixo
1	1	2	3	Baixo
1	1	1	1	Alto
1	1	1	2	Moderado
1	1	1	3	Baixo
3	1	1	2	Alto

Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Finanças

- Classificação de Crédito



Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Finanças

- Otimização de Portfólio



Fonte: BARRA, Daniel Sousa. **Otimização de Portfólio para uma Carteira de Criptomoedas: Uma abordagem em Reinforcement Learning.**

Extrato das Conclusões deste TCC:

- O trabalho demonstra a utilidade da utilização de técnicas computacionais como o machine learning na resolução de problemas econômicos
- Observou-se que a carteira inteligente que utiliza reinforcement learning possui um índice de risco retorno muito melhor do que as outras carteiras

Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Finanças

- Otimização de Portfólio

Ação	Porcentagem do recurso
DVN	0 %
DWDP	0 %
DXC	0.00000057843725547906%
EA	0.0000000000000735686007059297%
EBAY	87.4036672635803%
ECL	0.0000000000000286799040242216%
ED	0.0001486208850856%
EFX	0.0000000000000574876676082195%
EIX	0.0000000000000327285440871356%
EL	0%
EMN	12.5961567350221%
EMR	0.0000000000000152674459227347%
EOG	0.0000000000000533872440097732%
EQIX	0.0000224935876819509%
EQR	0.0000248876773559999%
EQT	0.00000069819906173266%
ES	0.00000115812679355857%
ESRX	0%
ESS	0.000000000000100096466540419%
ETFC	0%

Extrato das Conclusões deste Artigo:

- Utilizado LSTM (Long-Short Term Memory)
- Os resultados usando as bases de dados dos índices S&P 500 e Dow Jones mostrou que a rede LSTM alcançou previsões com baixo erro;

Fonte: PORTELA, Elaine Pinto; CORTES, Omar Andres Carmona. **Otimização de Portfólio Futuro baseado em Aprendizagem Profunda e Algoritmo Evolutivo Multiobjetivo**



Aplicações das Redes Neurais

VISÃO COMPUTACIONAL



Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Reconhecimento de Imagens

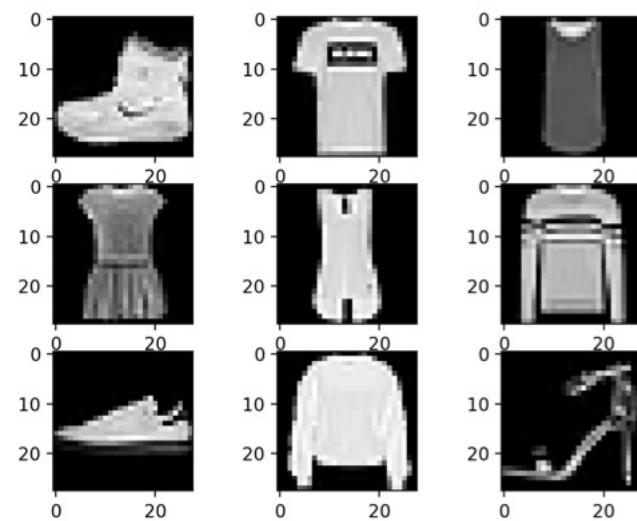
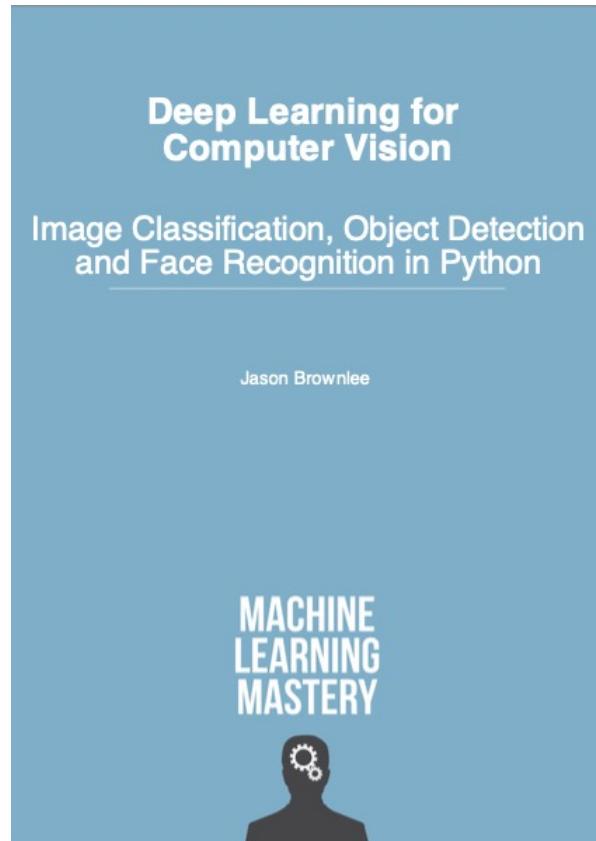


Figure 19.1: Plot of a Subset of Images From the Fashion-MNIST Dataset.



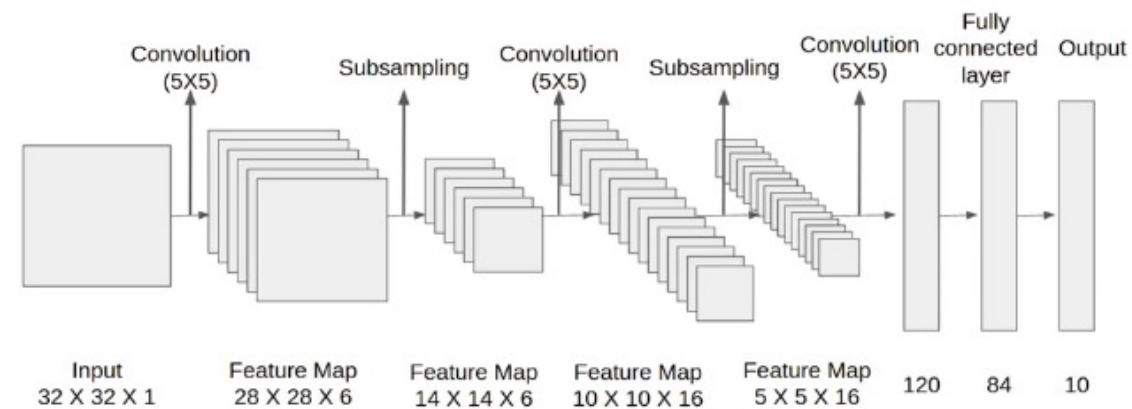
Figure 19.7: Sample Clothing (Pullover).

Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Reconhecimento de Imagens

- A LeNet-5 foi a primeira CNN a ser conhecida largamente e bem sucedida;
- Desenvolvida inicialmente para reconhecimento de caracteres escritos à mão;
- Operação de Convolução (A^*B) → extremamente importante em Processamento Digital de Sinais (DSP);
- Mais informações: <https://hackernoon.com/-understanding-convolution-neural-networks-cnn-the-eli5-way-photo-by-efe-kurnaz-on-unplash-u-pa1i327j>

Layer	# filters / neurons	Filter size	Stride	Size of feature map	Activation function
Input	-	-	-	32 X 32 X 1	
Conv 1	6	5 * 5	1	28 X 28 X 6	tanh
Avg. pooling 1		2 * 2	2	14 X 14 X 6	
Conv 2	16	5 * 5	1	10 X 10 X 16	tanh
Avg. pooling 2		2 * 2	2	5 X 5 X 16	
Conv 3	120	5 * 5	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully Connected 2	-	-	-	10	Softmax



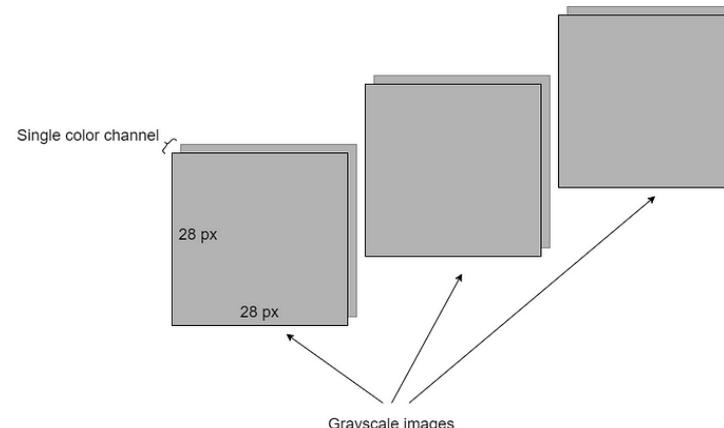
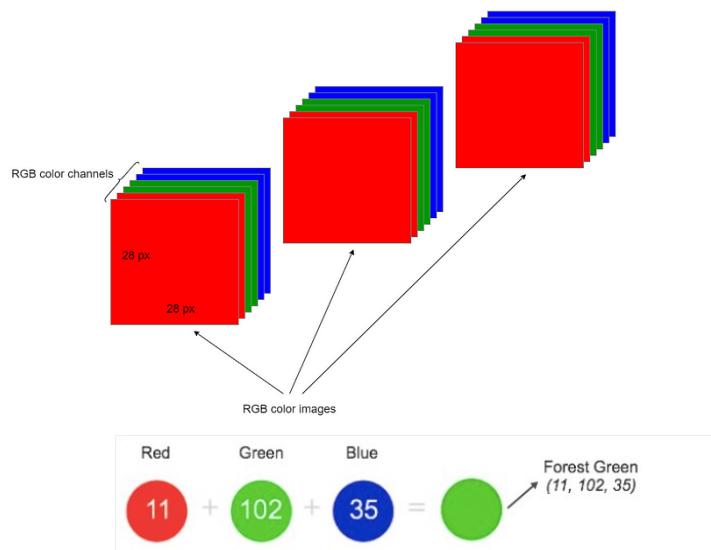
Arquitetura da LeNet-5 – Convolutional Neural Network

Fonte: LeCun, Yann et al.; Gradient-Based Learning Applied to Document Recognition (1998).

Analytics Vidhya, The Architecture of Lenet-5 (<https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/>)

Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Reconhecimento de Imagens



25 43 11 04 70 87 12 31 43 10 05 77 12 06 45 09 29 30 02
56 22 75 03 22 96 45 12 23 03 77 67 81 45 22 04 90 22 21
32 45 41 91 87 62 35 02 00 11 62 25 43 11 04 70 87 12 61
31 43 10 05 77 12 06 45 09 29 30 56 22 75 03 22 96 45 05
12 23 03 77 67 81 45 22 04 90 22 32 45 41 91 87 62 35 44
02 00 11 62 25 43 11 04 70 87 12 31 43 10 05 77 12 06 10
45 09 29 30 56 22 75 03 22 96 45 12 23 03 77 67 81 45 55
22 04 90 22 32 45 41 91 87 62 35 02 00 11 62 25 43 11 80
04 70 87 12 31 43 10 05 77 12 06 45 09 29 30 56 22 75 08
03 22 96 45 12 23 03 77 67 81 45 22 04 90 22 32 45 41 99
91 87 62 35 02 00 11 62 22 01 00 72 65 23 01 00 22 04 30
90 22 32 45 41 91 87 62 35 02 00 11 62 25 43 11 04 70 42
87 12 31 43 10 05 77 12 06 45 09 29 30 56 22 75 03 22 91
96 45 12 23 03 77 67 81 45 22 04 90 22 32 45 41 91 87 40
62 35 02 00 11 62 22 01 00 72 65 23 01 00 56 22 75 03 67
22 96 45 12 23 03 77 67 81 45 22 04 90 22 32 45 41 91 22

What we see

What computers see

A 5x5 input matrix is multiplied by a 3x3 kernel matrix. The result is a blurred 3x3 output matrix.

35	40	41	45	50
40	40	42	46	52
42	46	50	55	55
48	52	56	58	60
56	60	65	70	75

$$\times \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & 42 & \\ \hline & & \\ \hline \end{array}$$



$$* \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} =$$

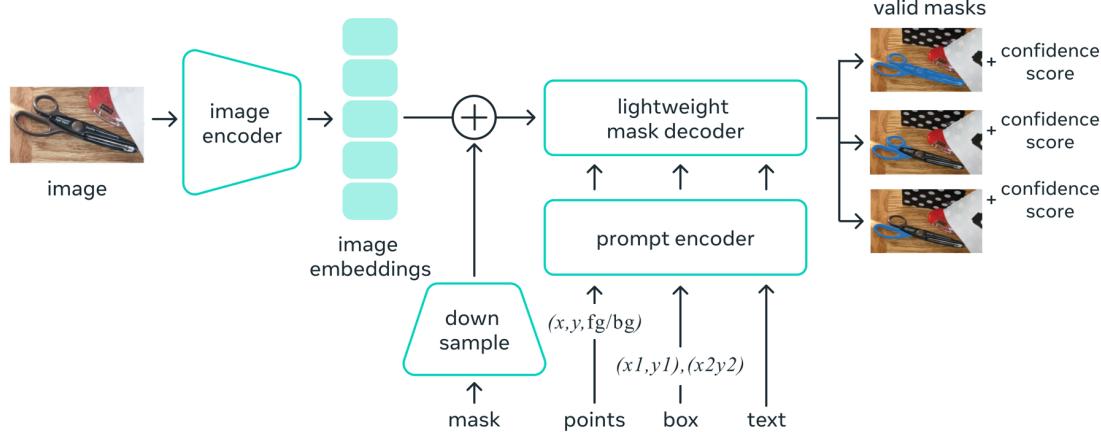


blurs the image

Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Segmentação de Imagens

Universal segmentation model

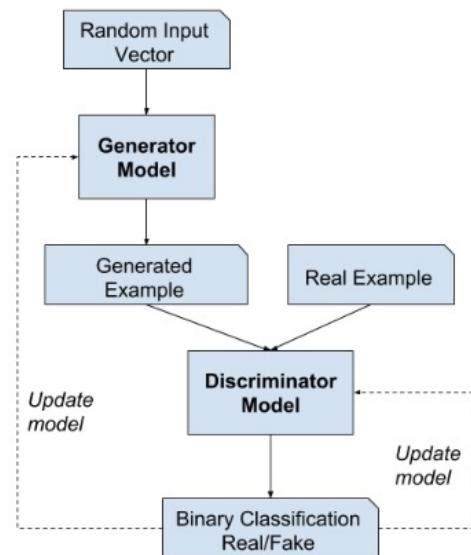
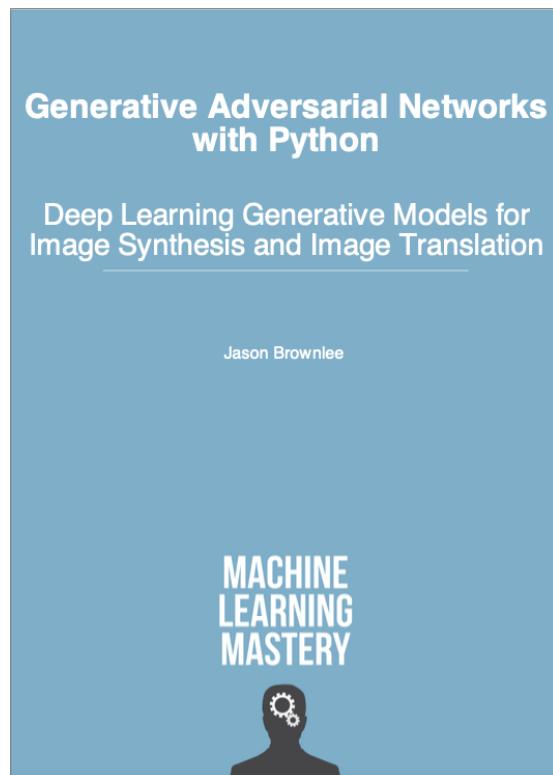


Arquitetura do Segment Anything by Meta Research AI

Fonte: Meta AI (<https://ai.meta.com/blog/segment-anything-foundation-model-image-segmentation/>), 5 de Abril de 2023.

Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – GANs



- GANs (Generative Adversarial Networks)
→ Generative + Adversarial
- Arquitetura descrita pela primeira vez em 2014, em um artigo de Ian Goodfellow → *Generative Adversarial Networks*
 - **Generator:** Gera novos exemplos plausíveis, considerando o domínio do problema. Tenta enganar o discriminador (adversarial)
 - **Discriminator:** Classifica os exemplos como verdadeiros ou falsos (real/fake). Busca não ser enganado pelo generator.

Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Navegação Autônoma

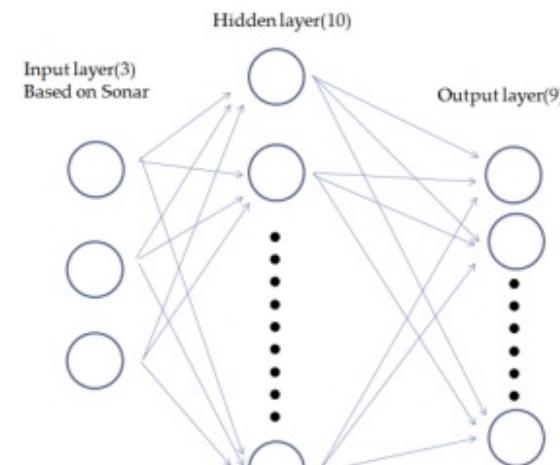
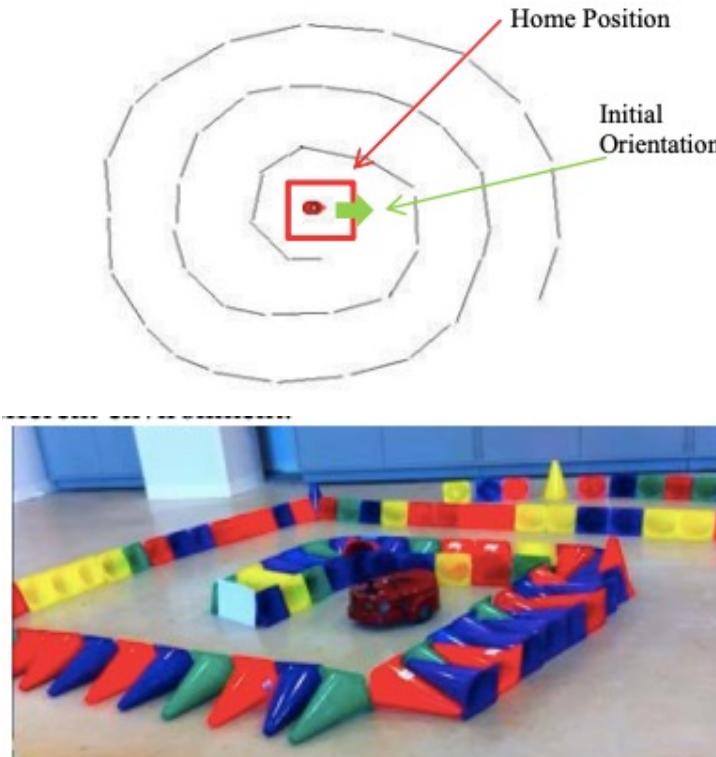
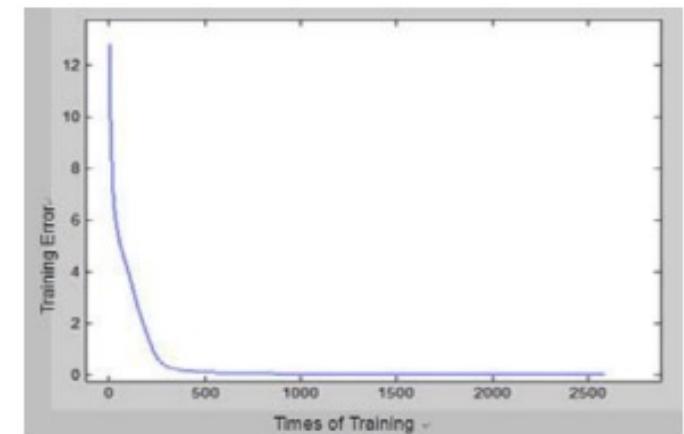


Fig. 2. The architecture of the BP Neural Network.



Fonte: SONG, Xiyang; FANG, Huangwei; JIAO, Xiong. *Autonomous Mobile Robot Navigation using Machine Learning* (2013). DOI: 10.1109/ICIAFS.2012.6419894

Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Navegação Autônoma

VI. CONCLUSIONS

In this paper, an autonomous navigation approach using neural networks was developed so that a mobile robot could employ its on-board sonar sensors to autonomously navigate through an unknown environment. First, the architecture of the BP Neural Network was established for autonomous navigation. Second, by training the network with 27 samples, the robot learned correct navigation skills in the unknown environment. Finally, the simulation and experimental results were presented to validate the proposed approach. This paper is thought to be a good effort to improve autonomous navigation capabilities of mobile robots using machine learning. The experimental results validated the effectiveness of the approach.

Fonte: SONG, Xiyang; FANG, Huangwei; JIAO, Xiong. **Autonomous Mobile Robot Navigation using Machine Learning** (2013).
DOI: 10.1109/ICIAFS.2012.6419894

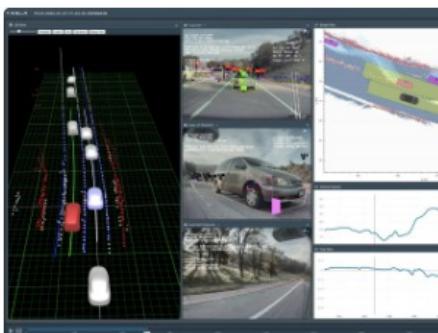
Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Navegação Autônoma



Neural Networks

Apply cutting-edge research to train deep neural networks on problems ranging from perception to control. Our per-camera networks analyze raw images to perform semantic segmentation, object detection and monocular depth estimation. Our birds-eye-view networks take video from all cameras to output the road layout, static infrastructure and 3D objects directly in the top-down view. Our networks learn from the most complicated and diverse scenarios in the world, iteratively sourced from our fleet of millions of vehicles in real time. A full build of Autopilot neural networks involves 48 networks that take 70,000 GPU hours to train. Together, they output 1,000 distinct tensors (predictions) at each timestep.



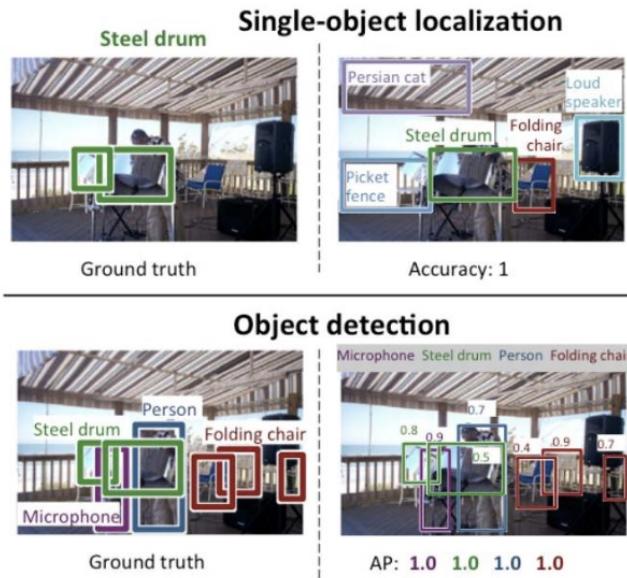
Autonomy Algorithms

Develop the core algorithms that drive the car by creating a high-fidelity representation of the world and planning trajectories in that space. In order to train the neural networks to predict such representations, algorithmically create accurate and large-scale ground truth data by combining information from the car's sensors across space and time. Use state-of-the-art techniques to build a robust planning and decision-making system that operates in complicated real-world situations under uncertainty. Evaluate your algorithms at the scale of the entire Tesla fleet.

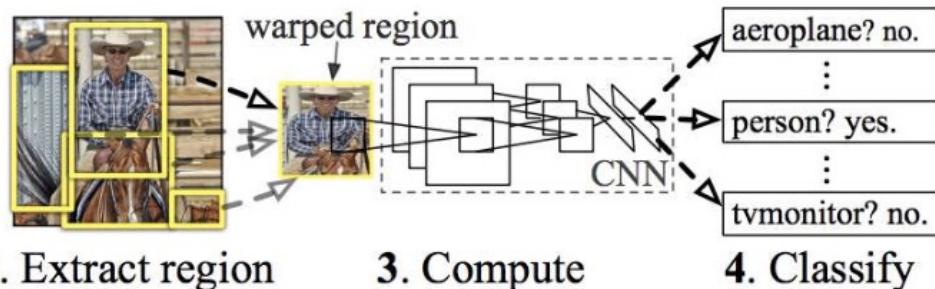
Fonte: <https://www.tesla.com/AI>

Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Reconhecimento de Objetos



R-CNN: *Regions with CNN features*



1. Input image

2. Extract region proposals (~2k)

3. Compute CNN features

4. Classify regions

- A arquitetura R-CNN foi descrita em 2014 em um artigo chamado “Rich Feature Hierarchies For Accurate Object Detection And Semantic Segmentation” → Localização, Detecção e Segmentação de Objetos
- Módulo 1 (it. 2): Proposta de Região → Gera propostas de regiões (2000/imagem)
- Módulo 2 (it. 3): Extração de Características → Trabalha em cada região proposta, buscando detectar objetos
- Módulo 3 (it. 4) Classificador → Atribui rótulos para os objetos que forem detectados

Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Reconstrução de Imagens

High-resolution image reconstruction with latent diffusion models from human brain activity

Yu Takagi^{1,2*} Shinji Nishimoto^{1,2}

¹Graduate School of Frontier Biosciences, Osaka University, Japan

²CiNet, NICT, Japan

{takagi.yuu.fbs,nishimoto.shinji.fbs}@osaka-u.ac.jp

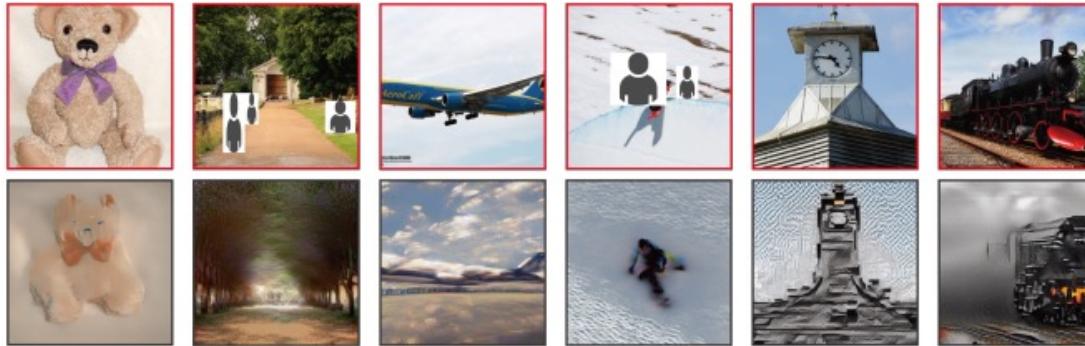
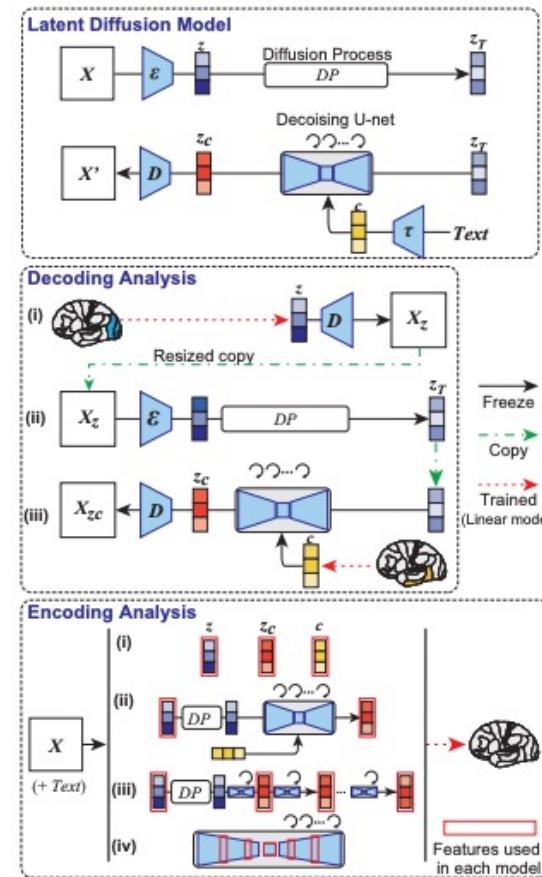


Figure 1. Presented images (red box, top row) and images reconstructed from fMRI signals (gray box, bottom row) for one subject (subj01).

Abstract

Reconstructing visual experiences from human brain ac-

forward fashion, without the need for any additional training and fine-tuning of complex deep-learning models. We also provide a quantitative interpretation of different LDM



Fonte: TAKAGI, Yu; NISHIMOTO, Shinji. **High-resolution image reconstruction with latent diffusion models from human brain activity.** Disponível em: <https://www.biorxiv.org/content/10.1101/2022.11.18.517004v2.full.pdf>

Aplicações das Redes Neurais

RECONHECIMENTO DE PADRÕES



Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Detecção de Fraudes



Detecção de Fraudes em Cartão de Crédito: Construindo Modelos de Machine Learning

Neste artigo, irei discorrer sobre as principais ideias e resultados obtidos através do meu projeto de Detecção de Fraudes em Cartão de...



Ygor Moreira Lima

Feb 6, 2022 · 10 min read

- A não-detecção de fraudes acarreta em prejuízos consideráveis, tanto para o consumidor, quanto para a instituição financeira.
- Geralmente os datasets fornecidos são desbalanceados → 0.1% de transações fraudulentas contra 99.9% de transações válidas → undersampling ou oversampling?



Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Detecção de Fraudes

CONJUNTO
ORIGINAL



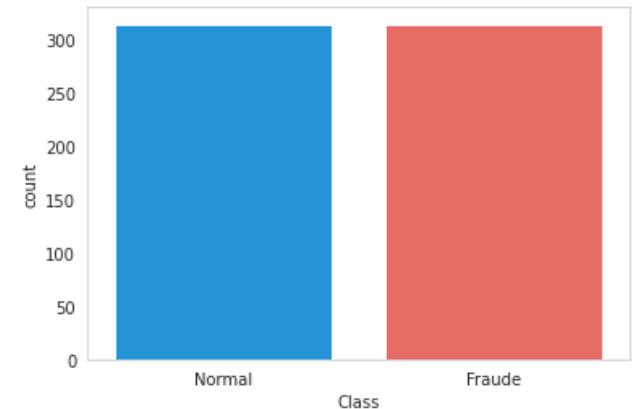
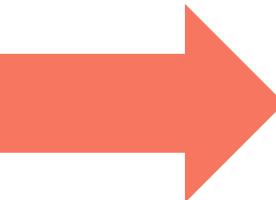
Por Ygor Lima
medium.com/ygormoreiralima

OVERSAMPLING



FRAUDE
LEGÍTIMA

UNDERSAMPLING



Fonte: LIMA, Ygor W S M. Detecção de Fraudes em Cartão de Crédito: Construindo Modelos de Machine Learning.

<https://laboratoriodebits.com.br/detect%C3%A7%C3%A3o-de-fraudes-em-cart%C3%A3o-de-cr%C3%A9dito-construindo-modelos-de-machine-learning-3484a39afee6>

Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Detecção de Fraudes

		CLASSE VERDADEIRA	
		POSITIVE	NEGATIVE
CLASSE PREVISTA	POSITIVE	TP	FP
	NEGATIVE	FN	TN

Por Ygor Lima
medium.com/ygormoreiralima

- TP — **True Positive**: Quando a transação é legítima e o modelo classifica como legítima.
- FP — **False Positive**: Quando a transação é legítima e o modelo classifica como fraudulenta.
- FN — **False Negative**: Quando a transação é fraudulenta e o modelo classifica como transação legítima.
- TN — **True Negative**: Quando a transação é fraudulenta e o modelo classifica como transação fraudulenta

Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Reconhecimento de Voz

- WaveNet: A generative model for Raw audio → “Deep Voice”
→ Um dos marcos significativos na síntese de voz que usou redes neurais profundas para gerar áudio de alta qualidade e naturalidade (DeepMind, 2016).

Top Gun: Maverick's Val Kilmer Has Had His Voice Reconstructed Through An Official AI, And The Results Are Amazing

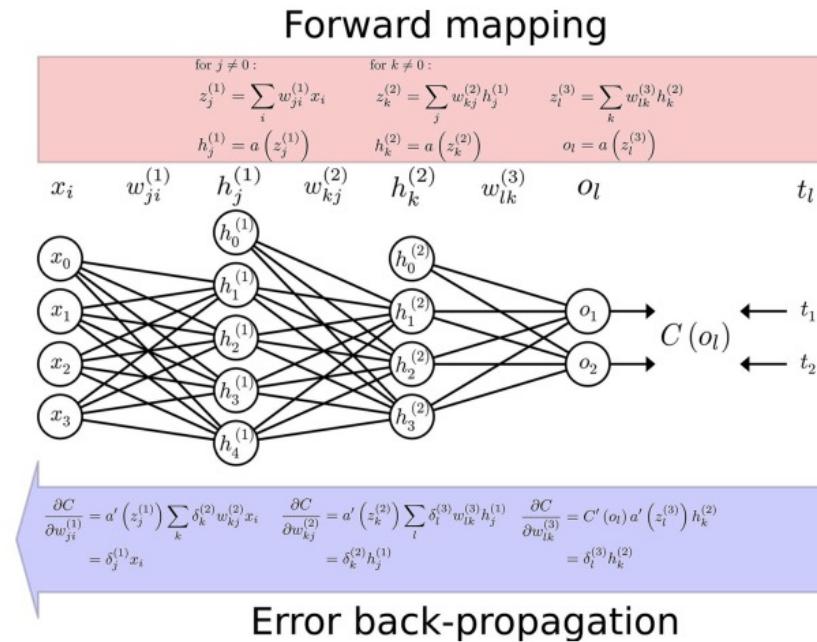
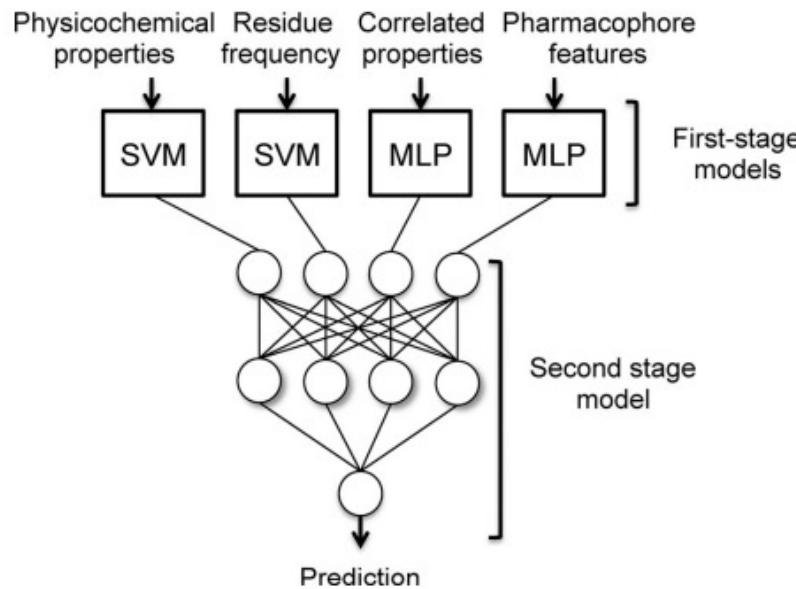
By Adreon Patterson published August 30, 2021



Aplicações das Redes Neurais

Aplicações das Redes Neurais Artificiais – Medicina

- Descoberta de novas medicações



Fonte: GAWEHN, Erik; HISS, Jan A.; SCHNEIDER, Gisbert. Deep Learning in Drug Discovery
DOI: 10.1002/minf.201501008





Redes Neurais Artificiais

Redes Neurais Artificiais

Objetivos do Módulo:

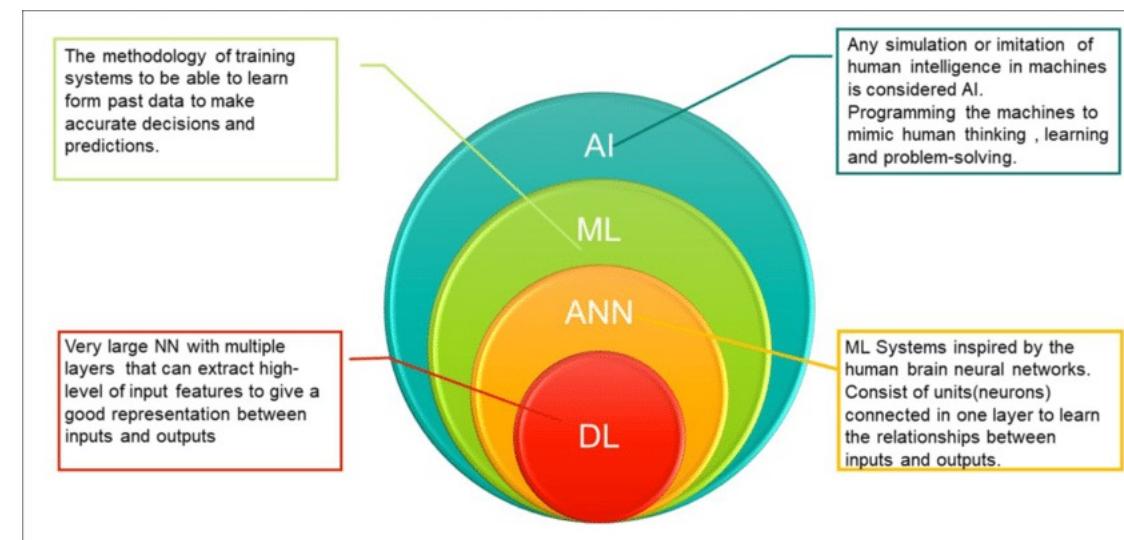
- Compreender a arquitetura de redes neurais artificiais.
- Explorar a importância das camadas ocultas e o aprofundamento de redes.
- Revisitar o feedforward
- Aprender sobre o algoritmo de treinamento Backpropagation.
- Compreender o ajuste de pesos com gradient descent, delta, taxa de aprendizagem e momento.



Redes Neurais Artificiais

Redes Neurais Artificiais – MLP - Arquitetura

- Vimos o funcionamento de um perceptron de camada única, que são uma maneira bastante interessante de introduzir um assunto mais complexo: os perceptrons multicamada – multi-layer perceptron (MLP)
- Até 2 camadas ocultas → Machine Learning
- Além de 2 camadas ocultas → Deep Learning



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

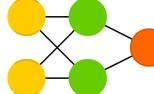
A mostly complete chart of
Neural Networks

©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org

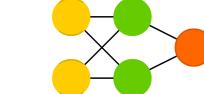
Perceptron (P)



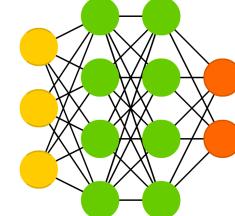
Feed Forward (FF)



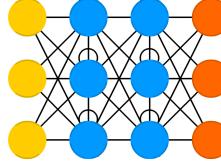
Radial Basis Network (RBF)



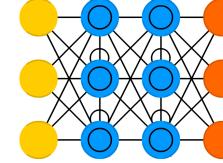
Deep Feed Forward (DFF)



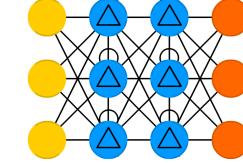
Recurrent Neural Network (RNN)



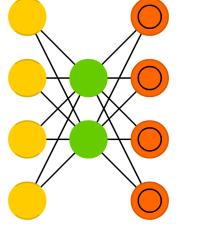
Long / Short Term Memory (LSTM)



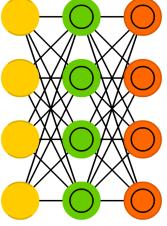
Gated Recurrent Unit (GRU)



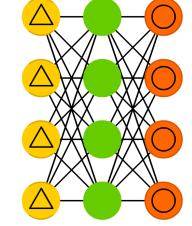
Auto Encoder (AE)



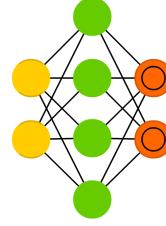
Variational AE (VAE)



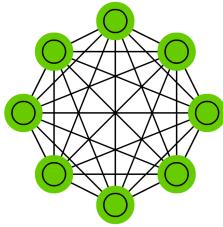
Denoising AE (DAE)



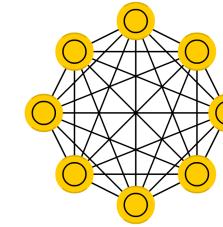
Sparse AE (SAE)



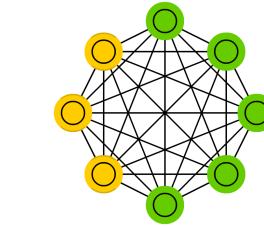
Markov Chain (MC)



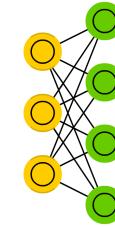
Hopfield Network (HN)



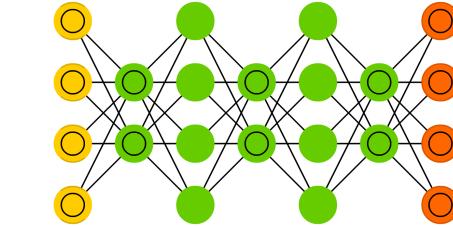
Boltzmann Machine (BM)



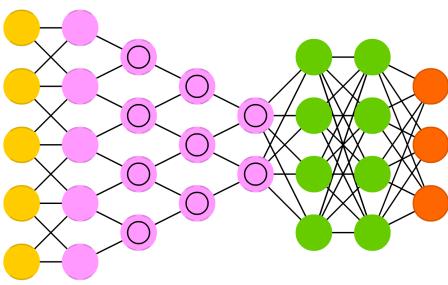
Restricted BM (RBM)



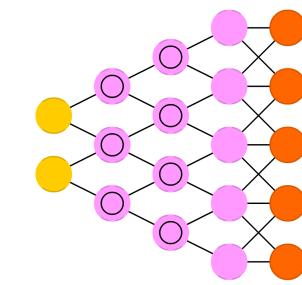
Deep Belief Network (DBN)



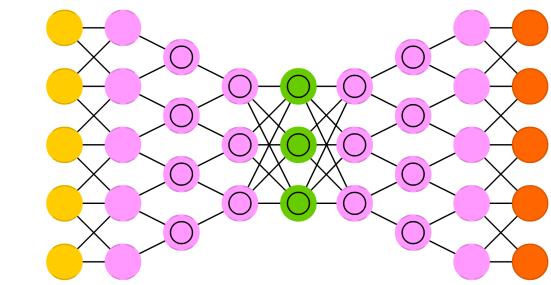
Deep Convolutional Network (DCN)



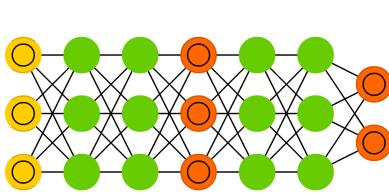
Deconvolutional Network (DN)



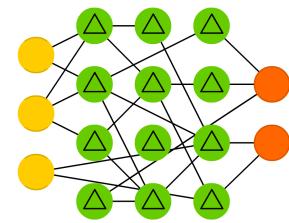
Deep Convolutional Inverse Graphics Network (DCIGN)



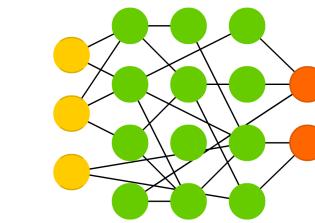
Generative Adversarial Network (GAN)



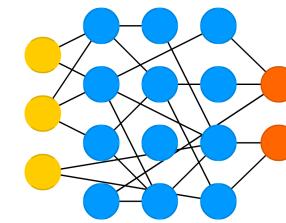
Liquid State Machine (LSM)



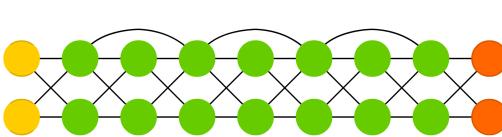
Extreme Learning Machine (ELM)



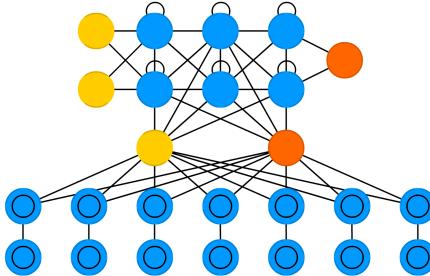
Echo State Network (ESN)



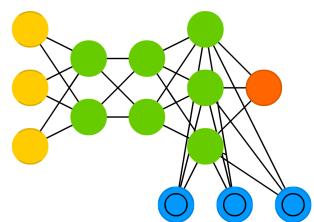
Deep Residual Network (DRN)



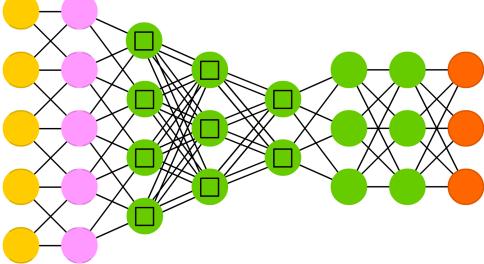
Differentiable Neural Computer (DNC)



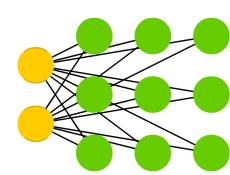
Neural Turing Machine (NTM)



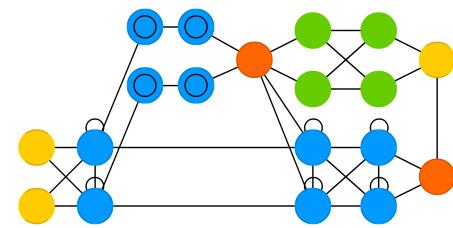
Capsule Network (CN)



Kohonen Network (KN)

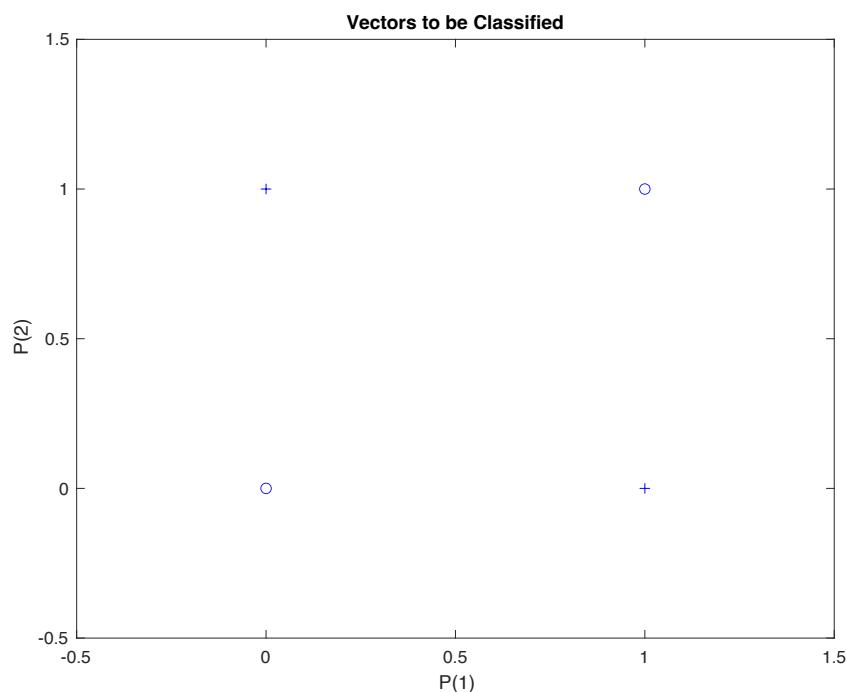


Attention Network (AN)



Redes Neurais Artificiais

Redes Neurais Artificiais – MLP - Arquitetura



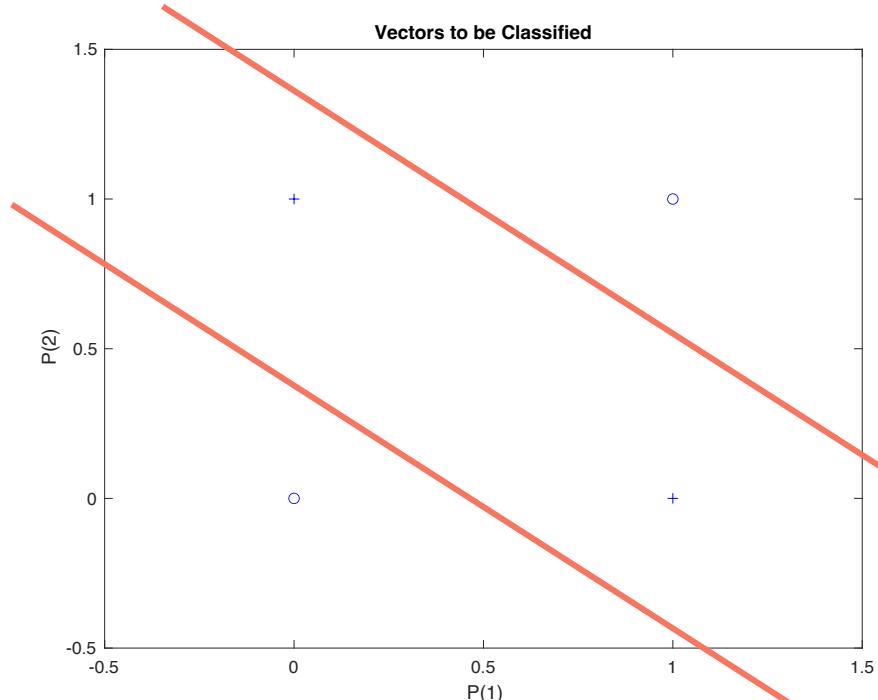
Porta Lógica XOR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Este problema é solucionável por perceptron único?

Redes Neurais Artificiais

Redes Neurais Artificiais – MLP - Arquitetura



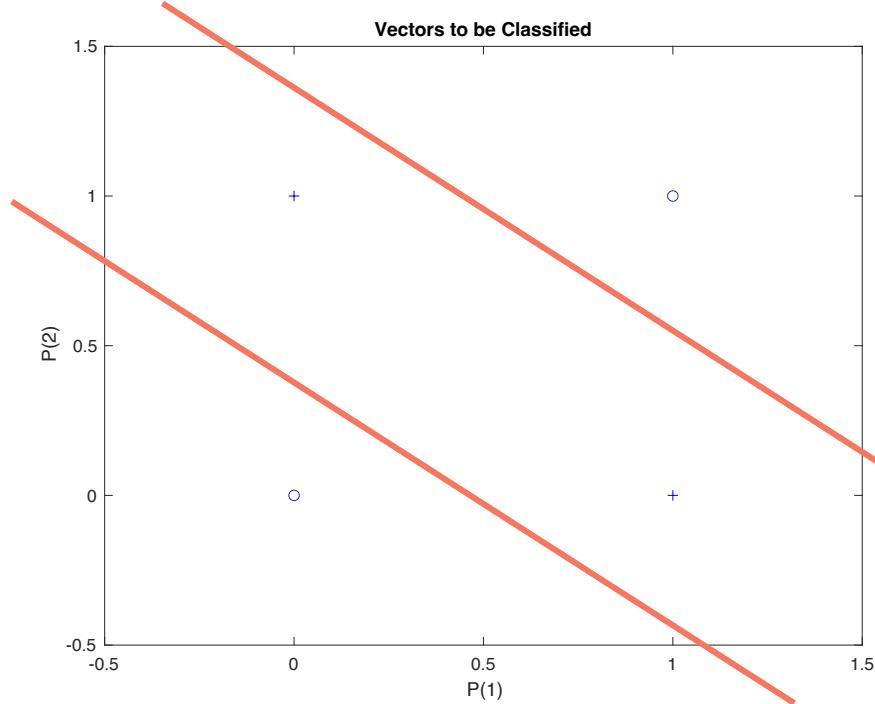
Não

Porta Lógica XOR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Redes Neurais Artificiais

Redes Neurais Artificiais – MLP - Arquitetura



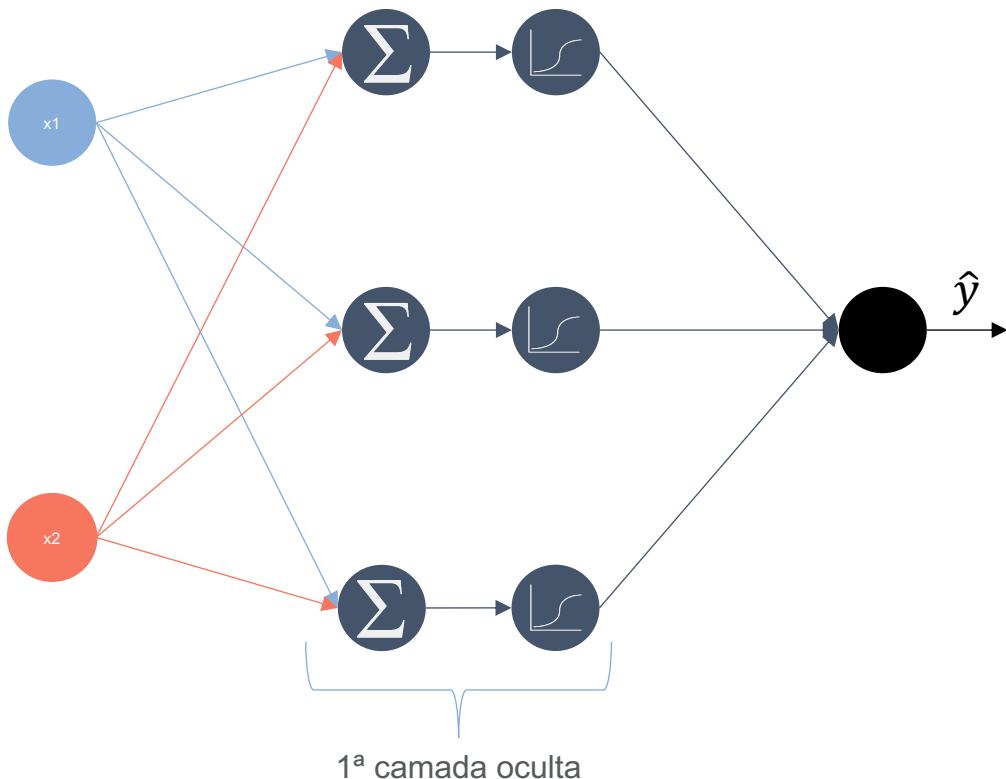
Porta Lógica XOR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Como fazemos? Arquiteturas mais complexas

Redes Neurais Artificiais

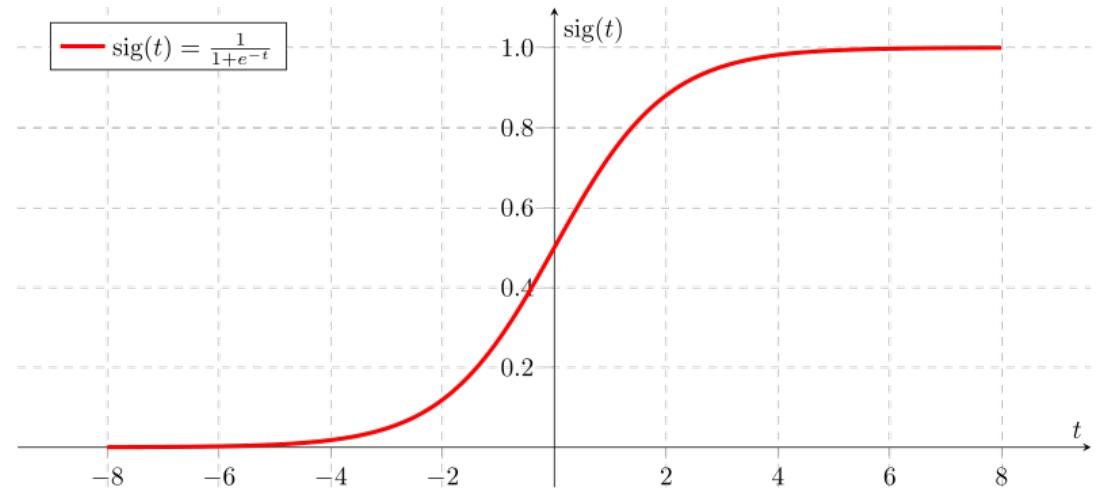
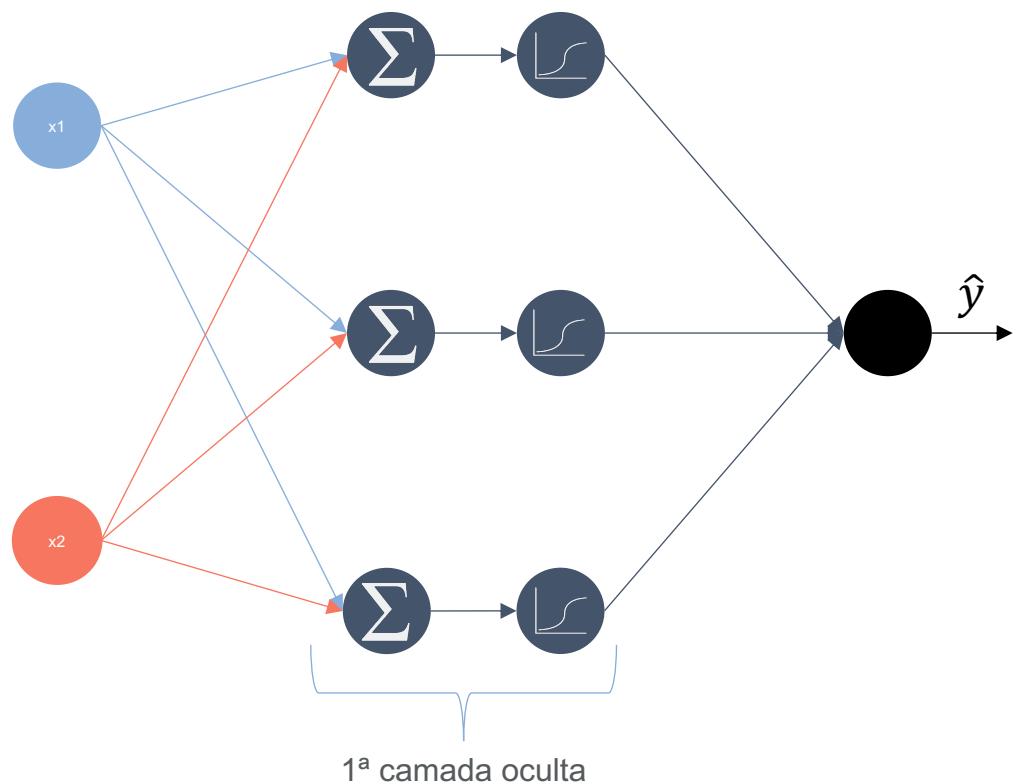
Redes Neurais Artificiais – MLP - Arquitetura

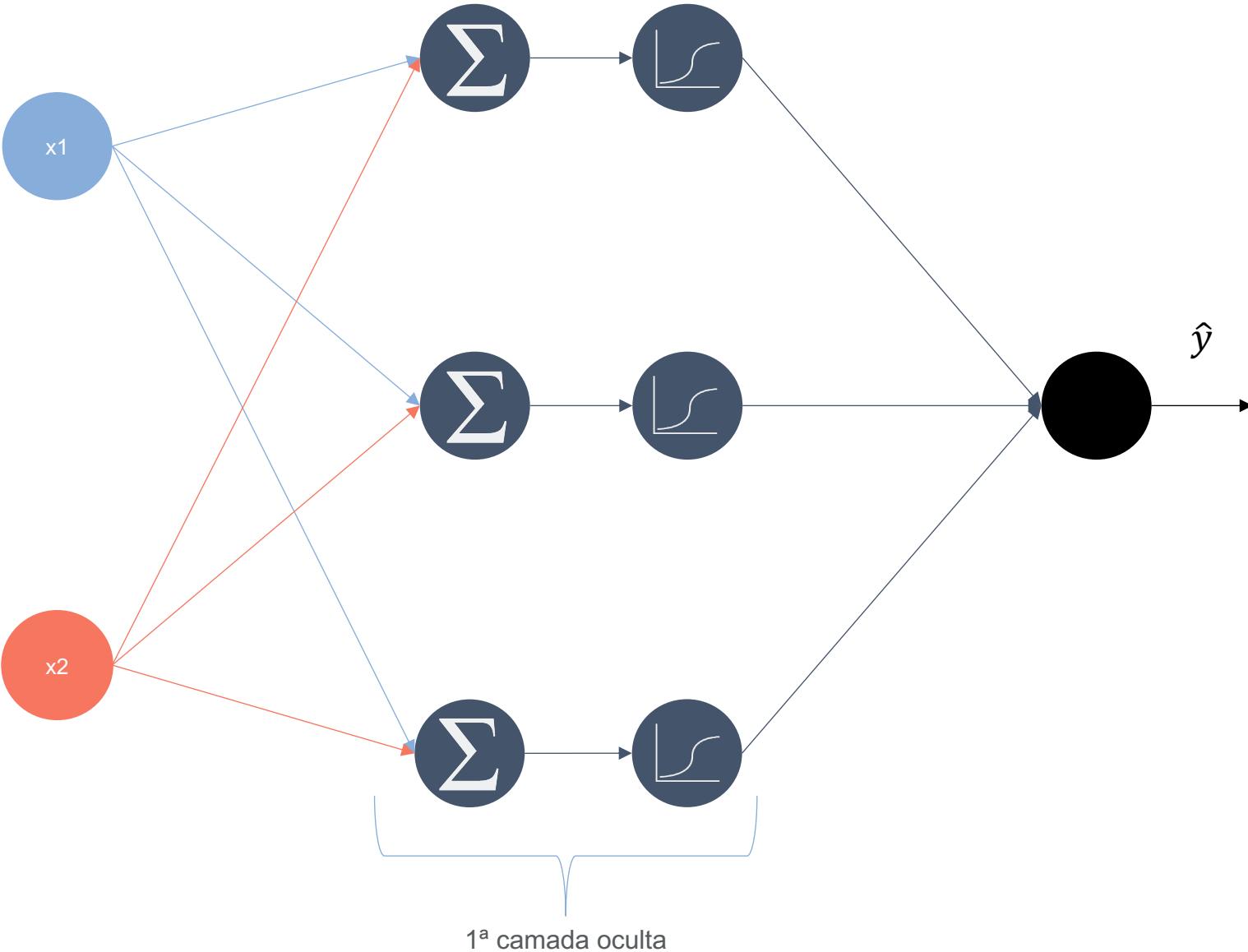


- Nesta arquitetura, há a possibilidade de solução de problemas mais complexos, conforme visto na seção “Aplicações de Redes Neurais”
- São fornecidos dois valores de entrada, x_1 e x_2 , a rede processa e retorna um valor estimado (\hat{y}) através do processo FEED-FORWARD
- O erro é calculado, propagado pela rede no sentido inverso (BACKPROPAGATION) e os pesos são ajustados

Redes Neurais Artificiais

Redes Neurais Artificiais – MLP - Arquitetura

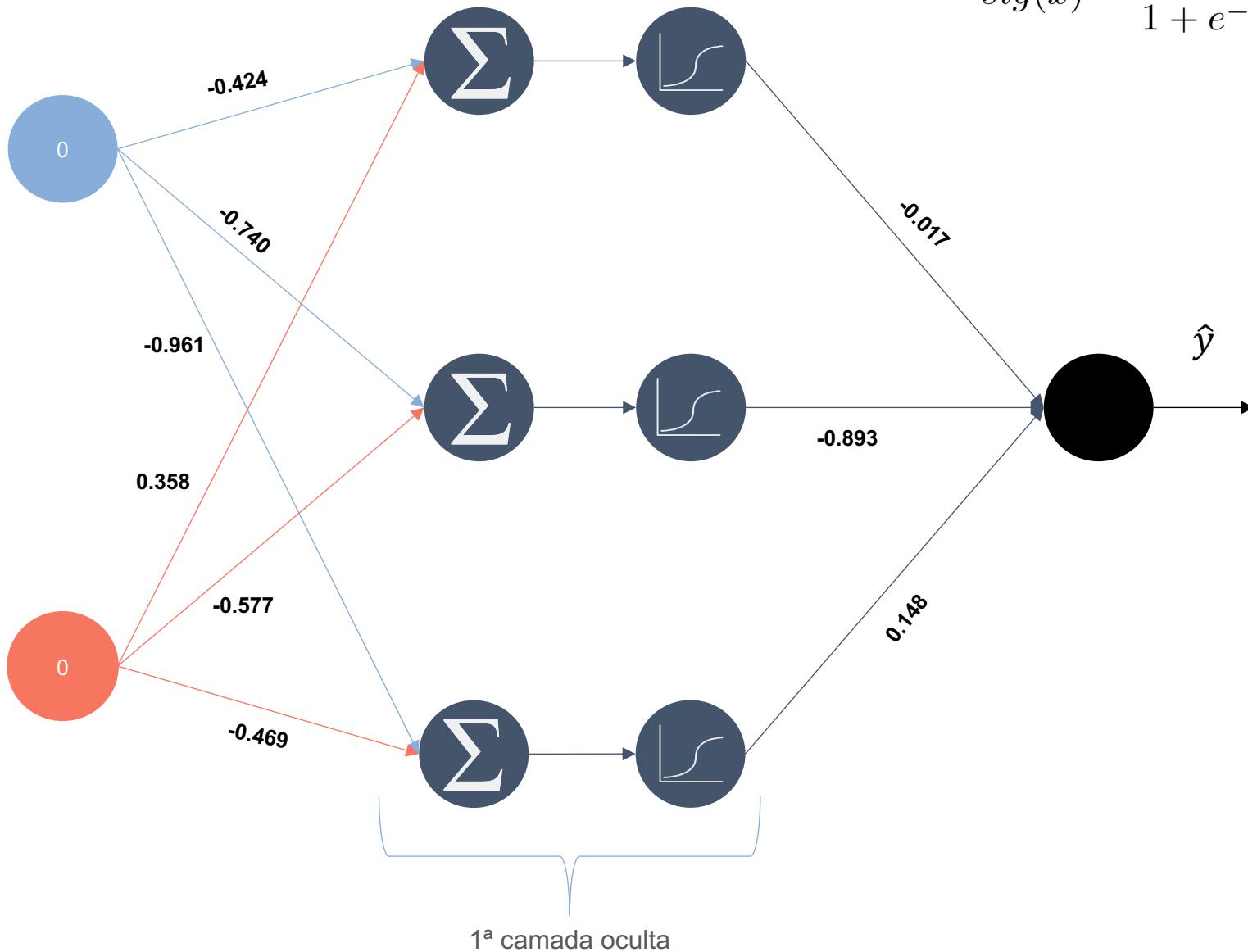




1º PASSO – INICIALIZAR OS PESOS

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

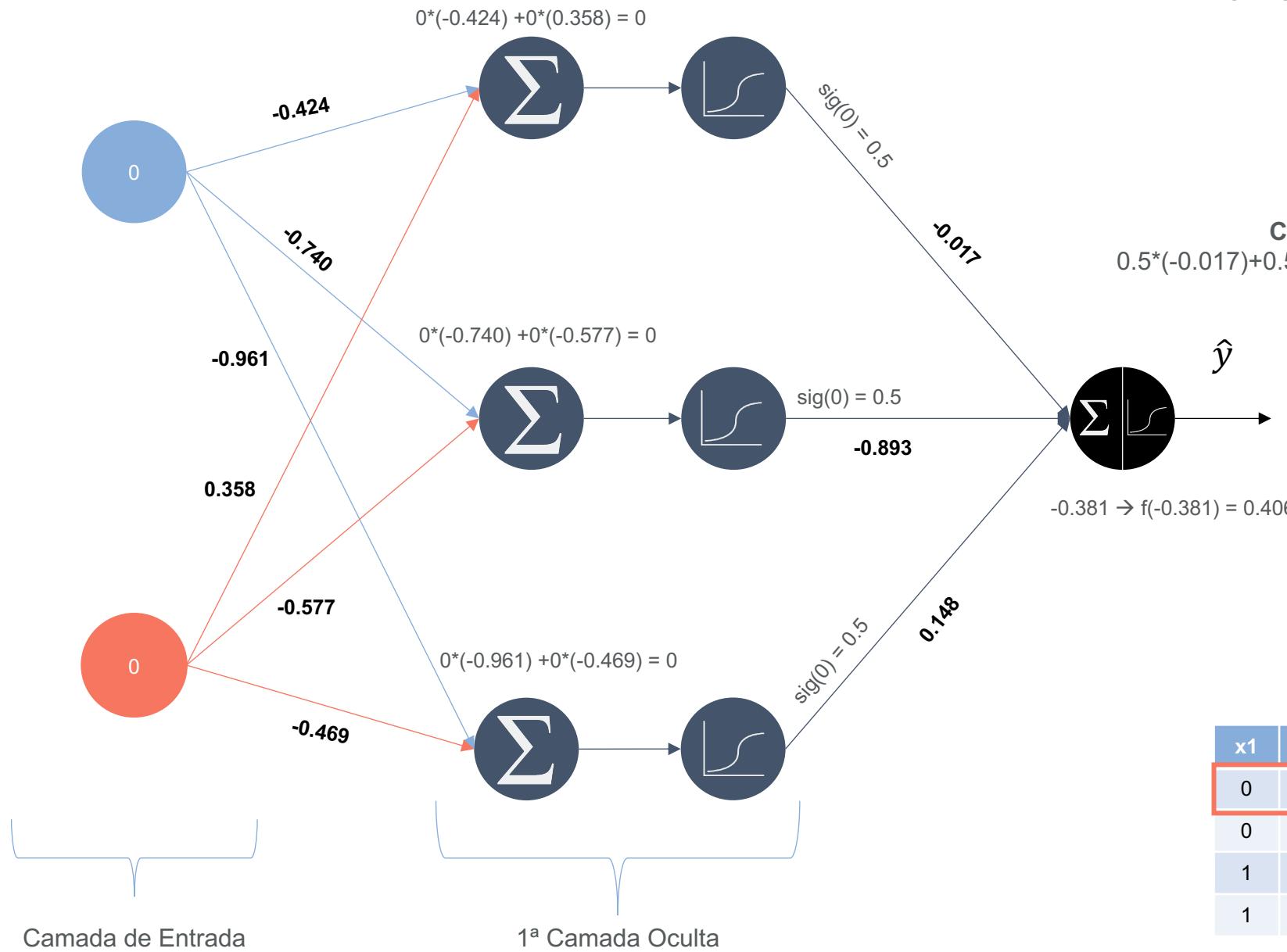
$$sig(x) = \frac{1}{1 + e^{-x}}$$



2º PASSO – INICIAR AS ITERAÇÕES (FEED-FORWARD)

CÁLCULOS ENTRADA → OCULTA

$$\begin{aligned} 0 * (-0.424) &= 0 \\ 0 * (-0.740) &= 0 \\ 0 * (-0.961) &= 0 \\ 0 * (0.358) &= 0 \\ 0 * (-0.577) &= 0 \\ 0 * (-0.469) &= 0 \end{aligned}$$



$$\text{sig}(x) = \frac{1}{1 + e^{-x}}$$

$$\text{sig}(0) = \frac{1}{1 + e^0}$$

$$\text{sig}(0) = \frac{1}{1 + 1}$$

$$\text{sig}(0) = \frac{1}{2} = 0.5$$

CÁLCULOS OCULTA → SAÍDA

$$\text{sig}(x) = \frac{1}{1 + e^{-x}}$$

$$\text{sig}(-0.381) = \frac{1}{1 + e^{+0.381}} = 0.406$$

x1	x2	y	\hat{y}	e
0	0	0	0.406	-0.406
0	1	1		
1	0	1		
1	1	0		



CÁLCULOS ENTRADA → OCULTA

$$0 * (-0.424) = 0$$

$$0 * (-0.740) = 0$$

$$0 * (-0.961) = 0$$

$$1 * (0.358) = 0.358$$

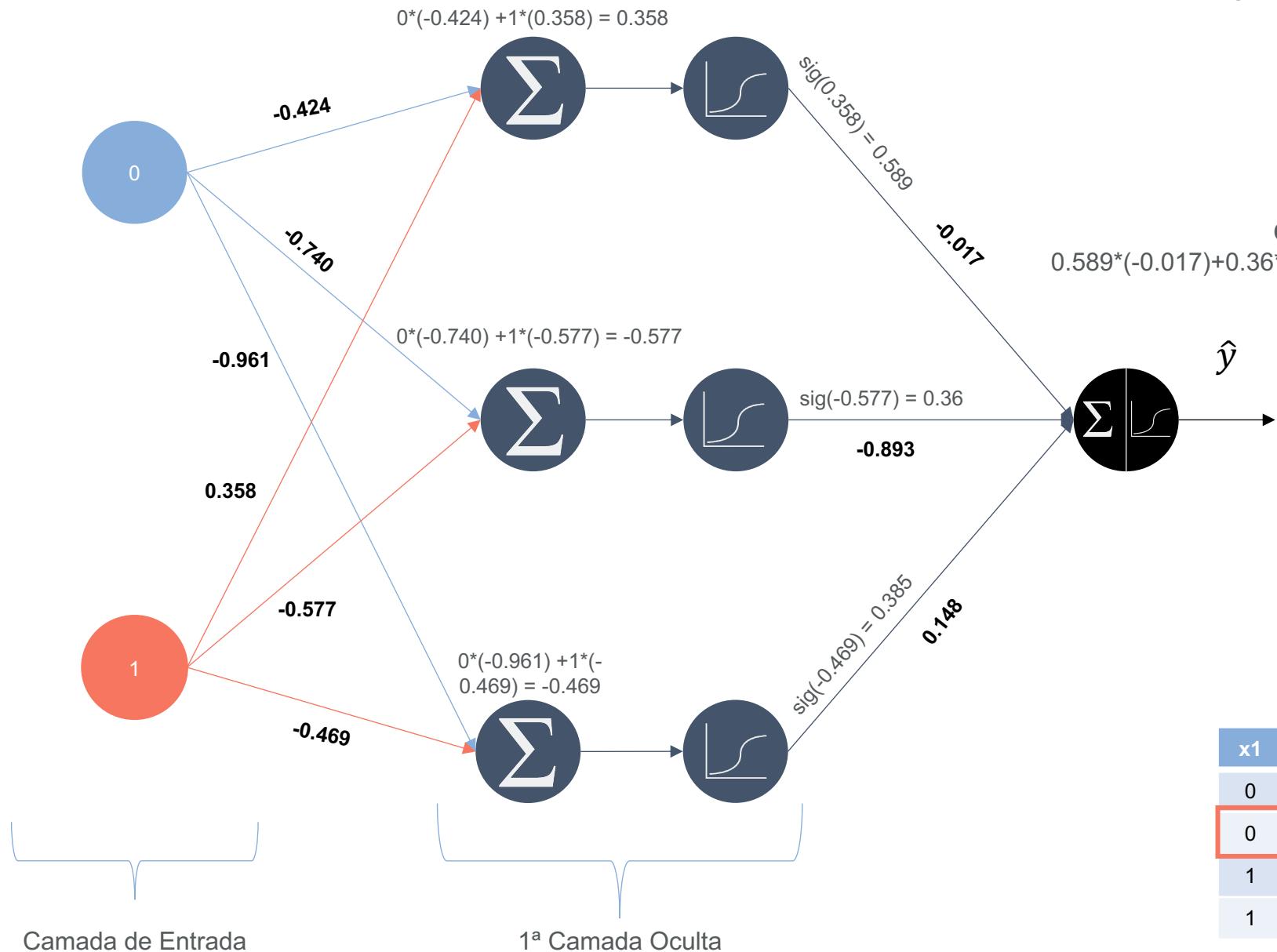
$$1 * (-0.577) = -0.577$$

$$1 * (-0.469) = -0.469$$

CÁLCULOS OCULTA → SAÍDA

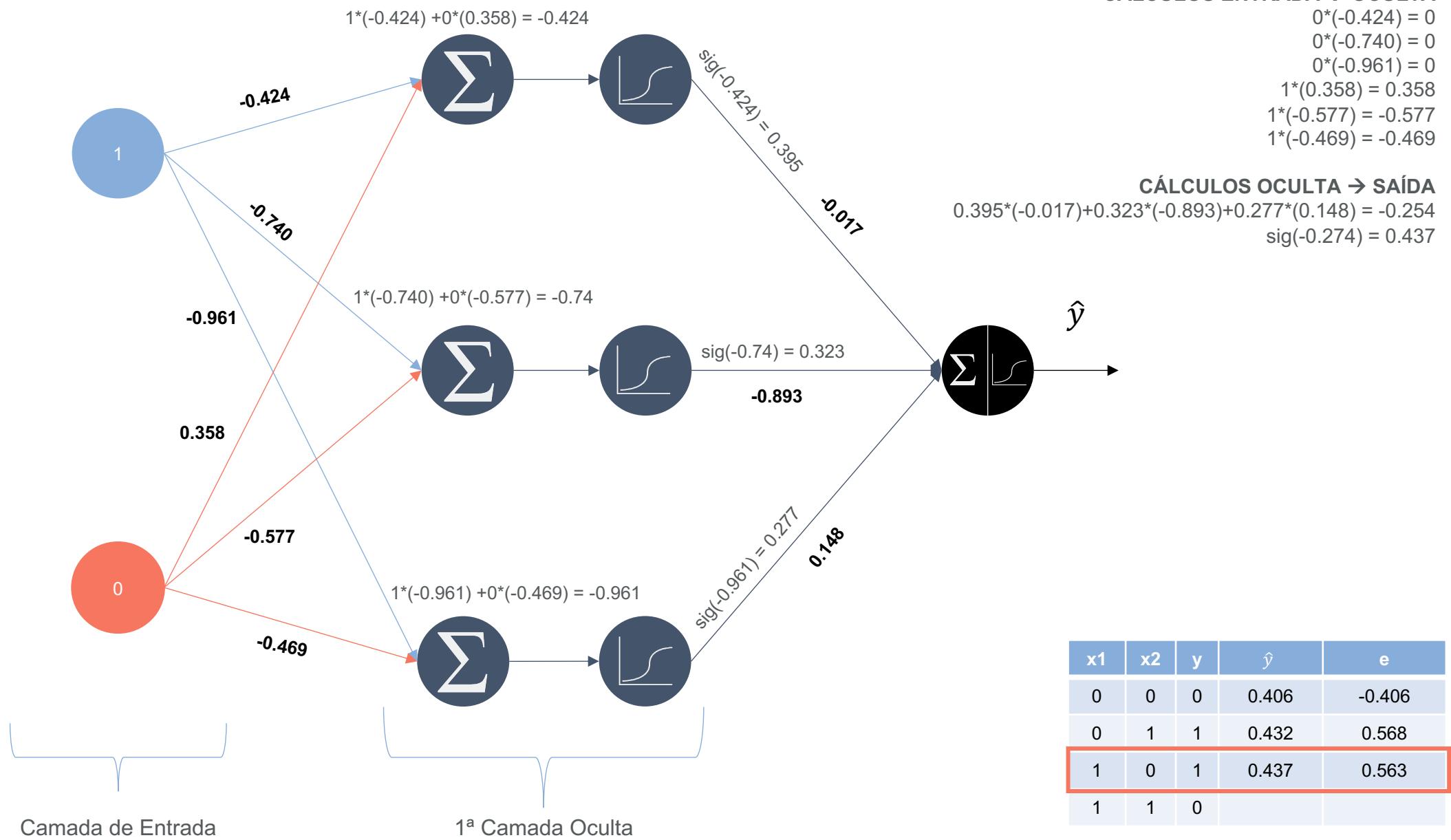
$$0.589 * (-0.017) + 0.36 * (-0.893) + 0.385 * (0.148) = -0.274$$

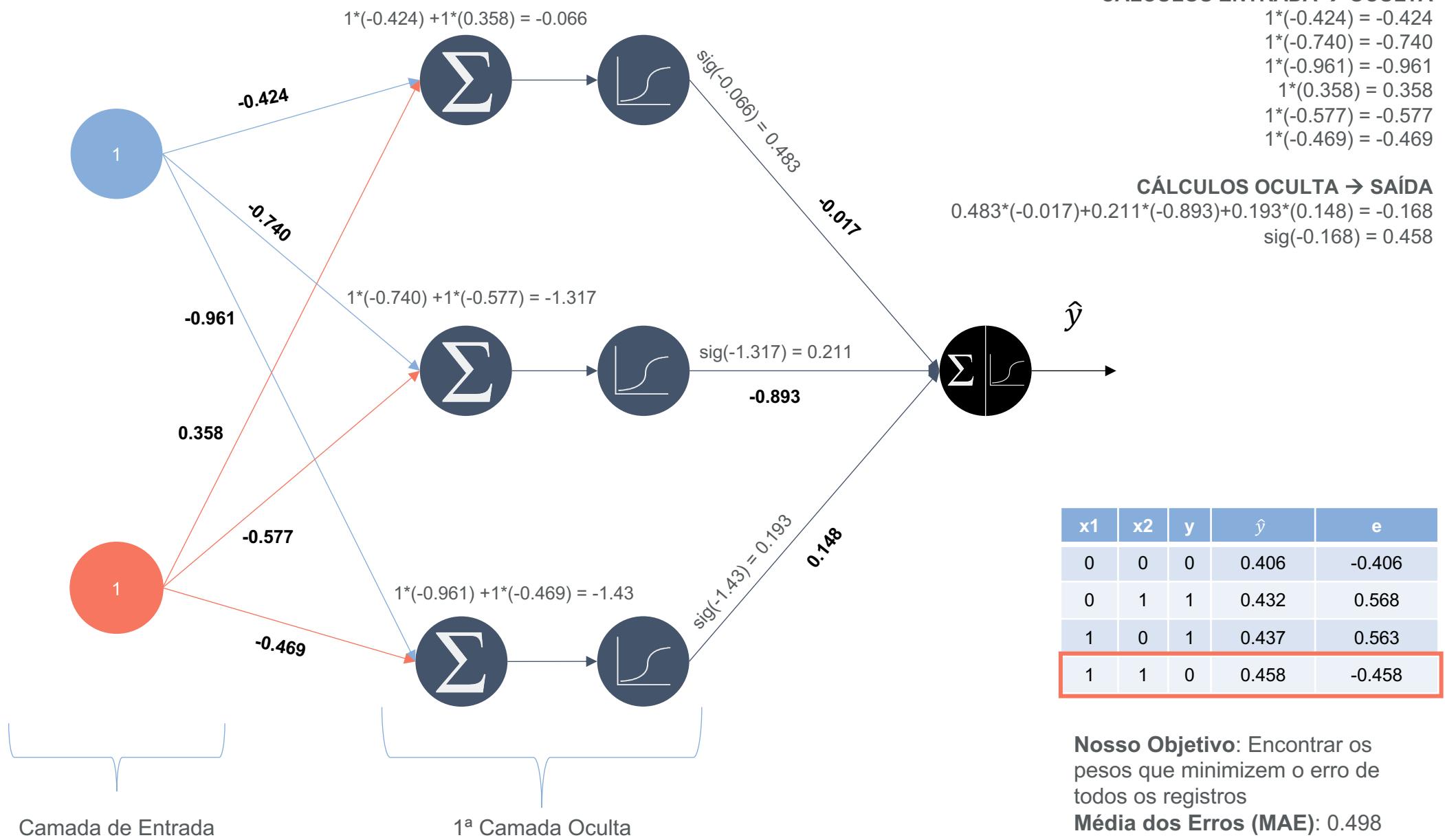
$$\text{sig}(-0.274) = 0.431$$



x1	x2	y	\hat{y}	e
0	0	0	0.406	-0.406
0	1	1	0.432	0.568
1	0	1		
1	1	0		



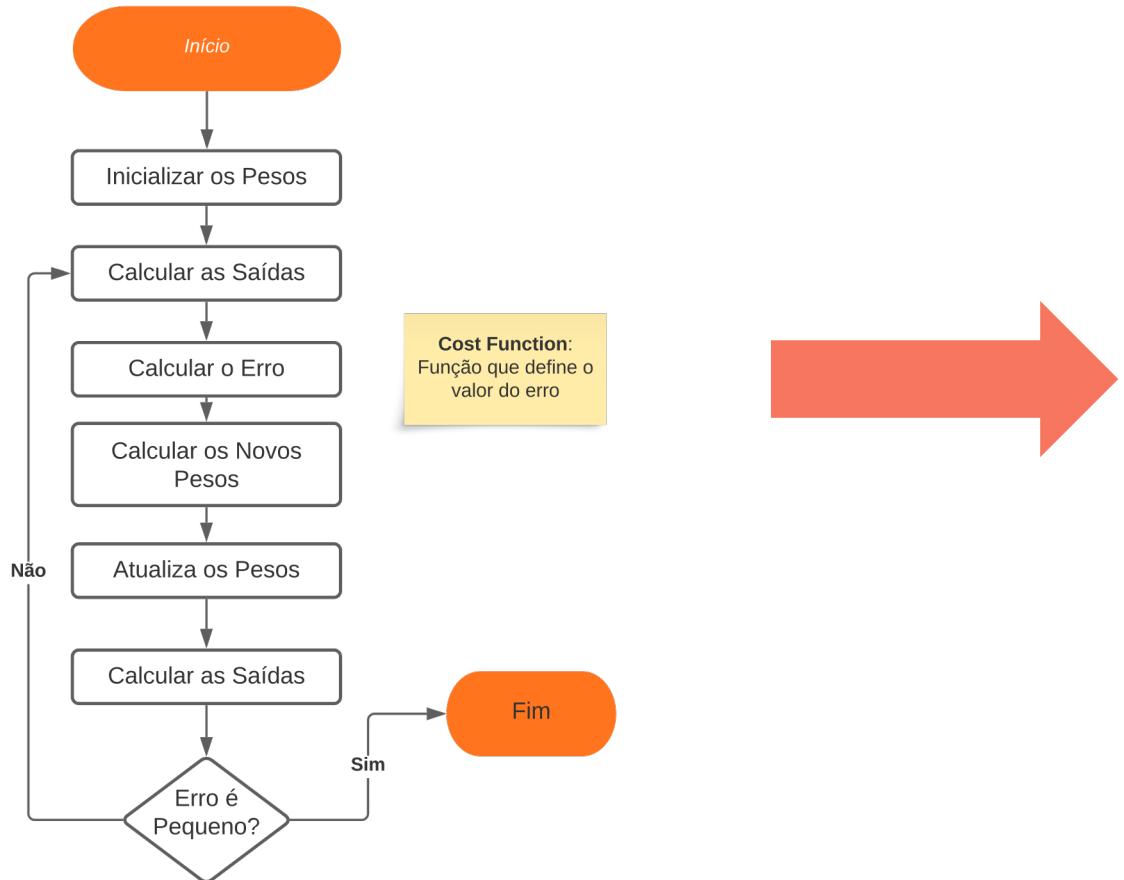




3º PASSO – INICIAR A RETROPROPAGAÇÃO (BACKPROPAGATION)

Redes Neurais Artificiais

Redes Neurais Artificiais – Gradiente

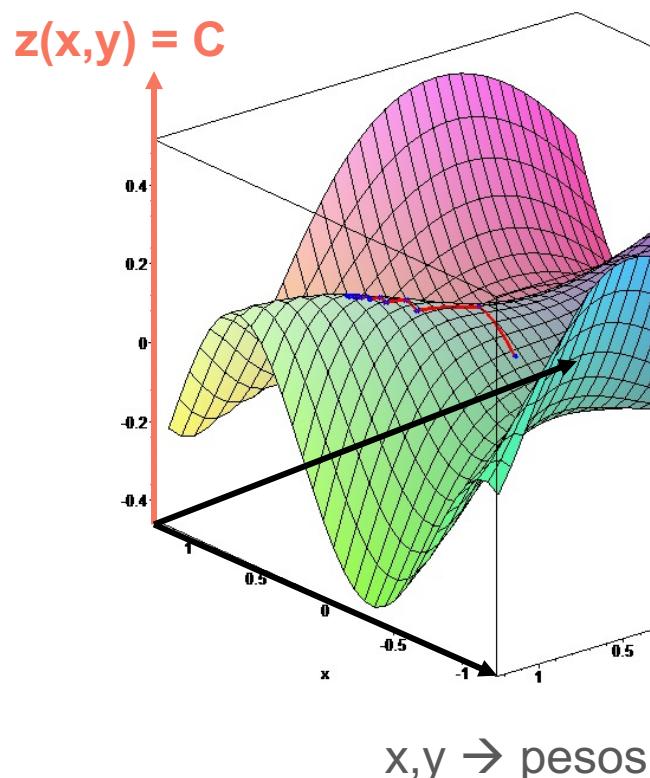


PARA O CÁLCULO DOS NOVOS PESOS:

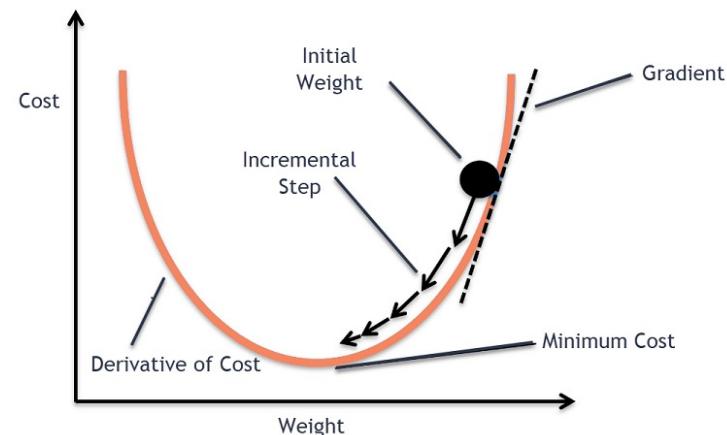
- Gradiente (Gradient Descent)**
- Derivadas e Derivadas Parciais**
- Cálculo do Delta**
- Backpropagation
- Learning Rate (α)
- Momentum**

Redes Neurais Artificiais

Redes Neurais Artificiais – Gradiente



$x, y \rightarrow$ pesos



Nosso objetivo é, portanto:

$$\min C(w_1, w_2, \dots, w_n)$$

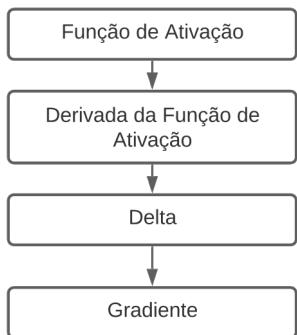
Como?

- Calculando a derivada parcial para nos movermos na direção da descida do gradiente (minimizando o erro)
- Força Bruta
- Algoritmos Genéticos (AGs)

2 pesos $\rightarrow \mathbb{R}^3$; n pesos $\rightarrow \mathbb{R}^{n+1}$

Redes Neurais Artificiais

Redes Neurais Artificiais – Backpropagation



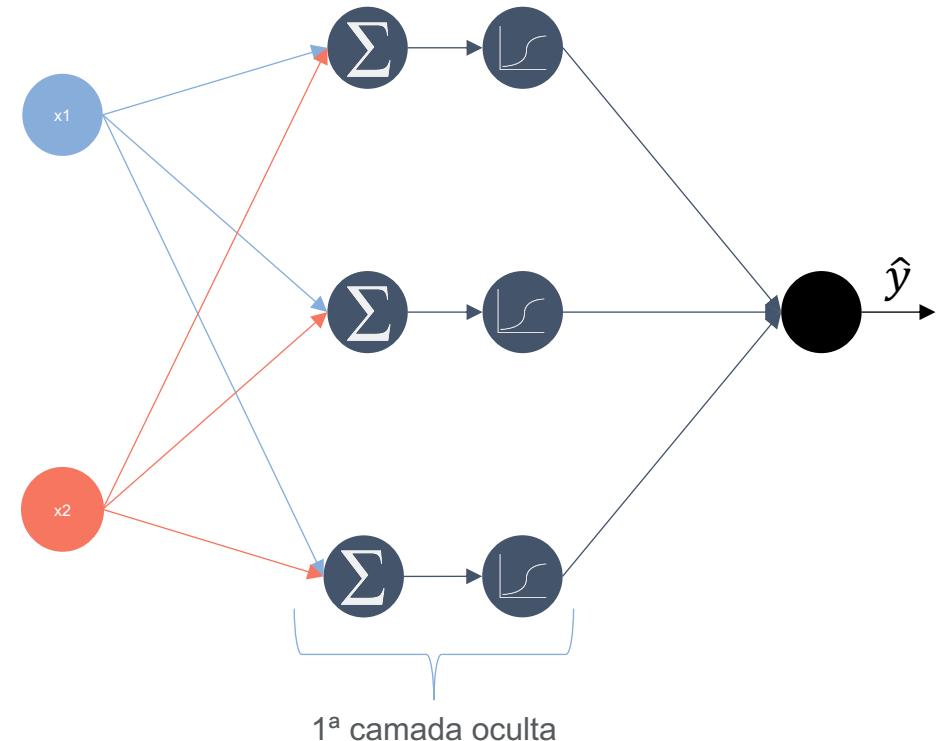
$$sig(x) = \frac{1}{1 + e^{-x}}$$

$$y'(x) = y(1 - y)$$

1º - DELTA CAMADA SAÍDA
 $\delta_{saída} = erro * derivadaSigmoide$
OBS: Calculado para todos os registros (data points)

2º - DELTA CAMADA OCULTA
 $\delta_{oculta} = derivadaSigmoide * peso * \delta_{saída}$

3º - ATUALIZAÇÃO DOS PESOS
 $peso_{n+1} = (peso_n * momento) + (entrada * \delta * taxaAprendizagem)$



1ª camada oculta

MOMENTO

- Permite escapar dos mínimos locais
- Define a confiabilidade da última alteração → inércia
- Valor alto: aumenta a velocidade de convergência
- Valor baixo: pode evitar mínimos locais

TAXA DE APRENDIZAGEM

- Define a velocidade de aprendizagem dos pesos
- Valor alto: convergência rápida → pode perder o mínimo global
- Valor baixo: será mais lento → pode chegar ao mínimo global

CÁLCULO DO $\delta_{saída}$

CÁLCULOS OCULTA → SAÍDA

$$0.5*(-0.017) + 0.5*(-0.893) + 0.5*(0.148) = -0.381$$

$$\text{sig}(-0.381) = \color{blue}{0.406}$$

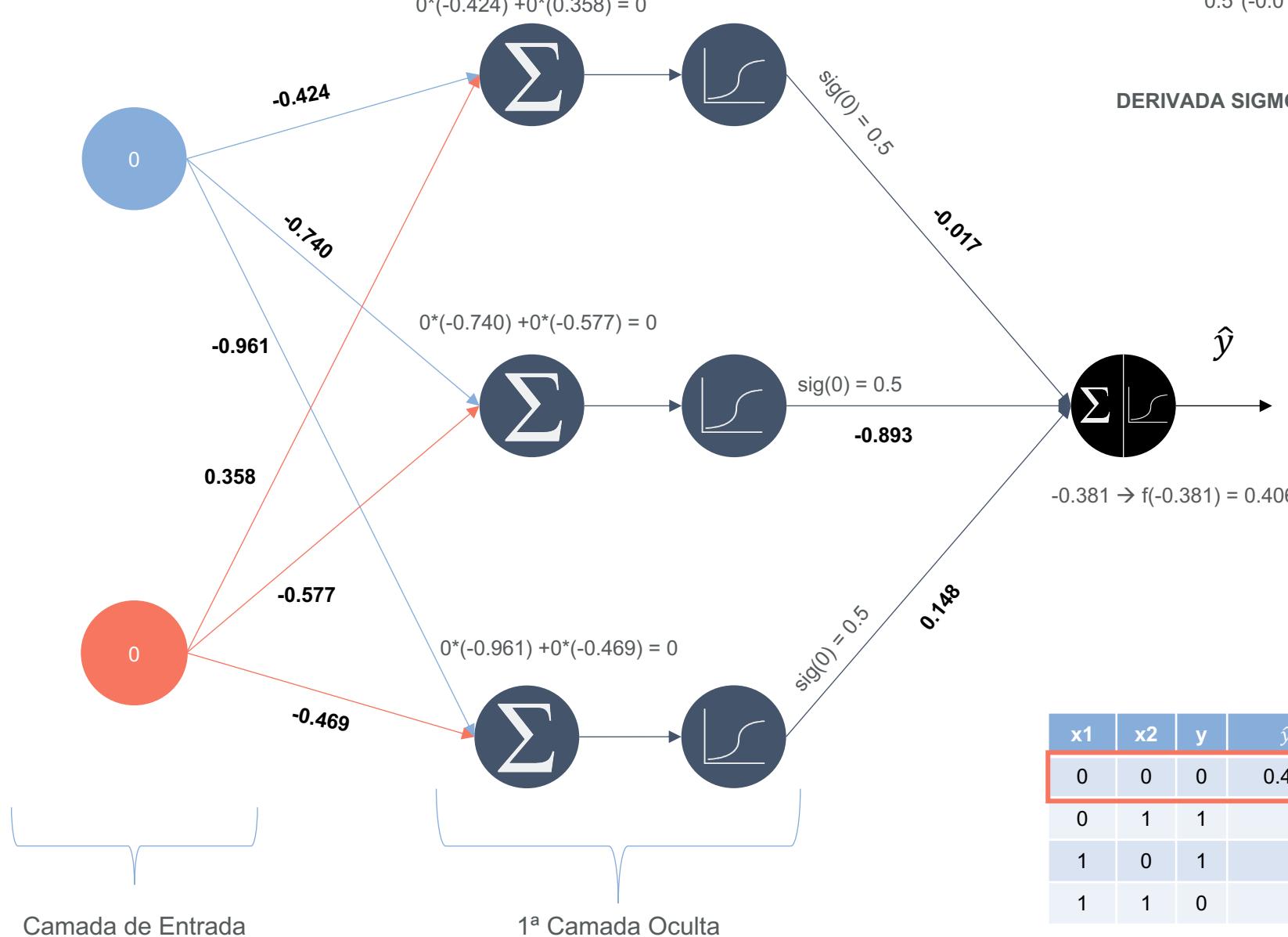
$$\text{ERRO: } 0 - 0.406 = \color{red}{-0.406}$$

$$\text{DERIVADA SIGMOIDE: } y(1-y) = \color{blue}{0.406 * (1-0.406)} = \color{green}{0.241}$$

DELTA CAMADA SAÍDA:

$$\delta_{\text{saída}} = \text{erro} * \text{derivadaSigmoide}$$

$$\delta_{\text{saída}} = -0.406 * 0.241 = -0.098$$



x1	x2	y	\hat{y}	e	$\delta_{\text{saída}}$
0	0	0	0.406	-0.406	-0.098
0	1	1			
1	0	1			
1	1	0			



CÁLCULOS OCULTA → SAÍDA

$$0.589*(-0.017)+0.36*(-0.893)+0.385*(0.148) = -0.274$$

$$\text{sig}(-0.274) = \color{blue}{0.431}$$

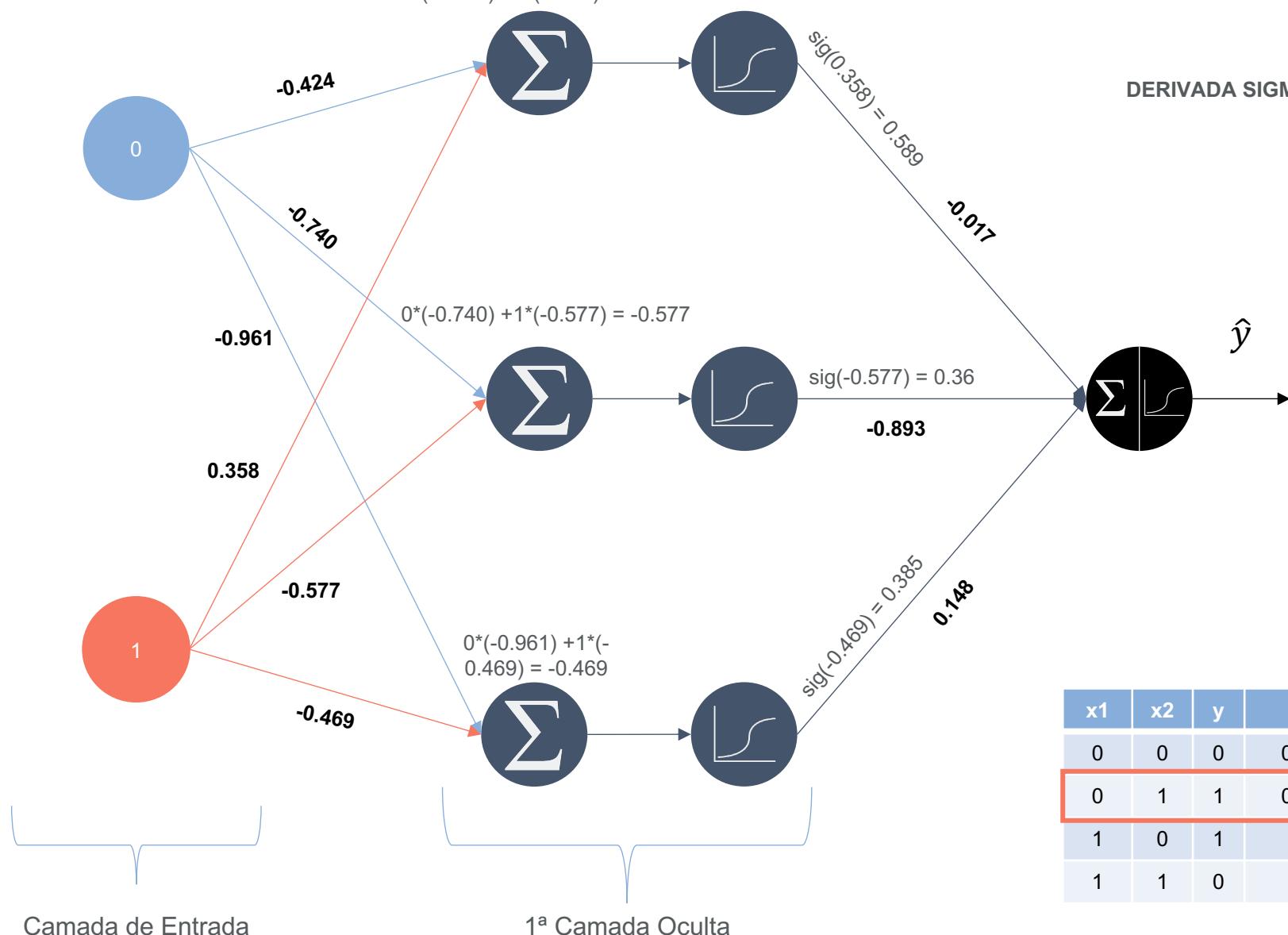
ERRO: $1 - 0.432 = \color{red}{0.568}$

DERIVADA SIGMOIDE: $y(1-y) = \color{blue}{0.431*(1-0.431)} = \color{green}{0.245}$

DELTA CAMADA SAÍDA:

$$\delta_{\text{saída}} = \text{erro} * \text{derivadaSigmoid}$$

$$\delta_{\text{saída}} = 0.568 * 0.245 = 0.139$$



x1	x2	y	\hat{y}	e	$\delta_{\text{saída}}$
0	0	0	0.406	-0.406	-0.098
0	1	1	0.432	0.568	0.139
1	0	1			
1	1	0			



CÁLCULOS OCULTA → SAÍDA

$$0.395*(-0.017)+0.323*(-0.893)+0.277*(0.148) = -0.254$$

$$\text{sig}(-0.274) = \color{blue}{\underline{\underline{0.437}}}$$

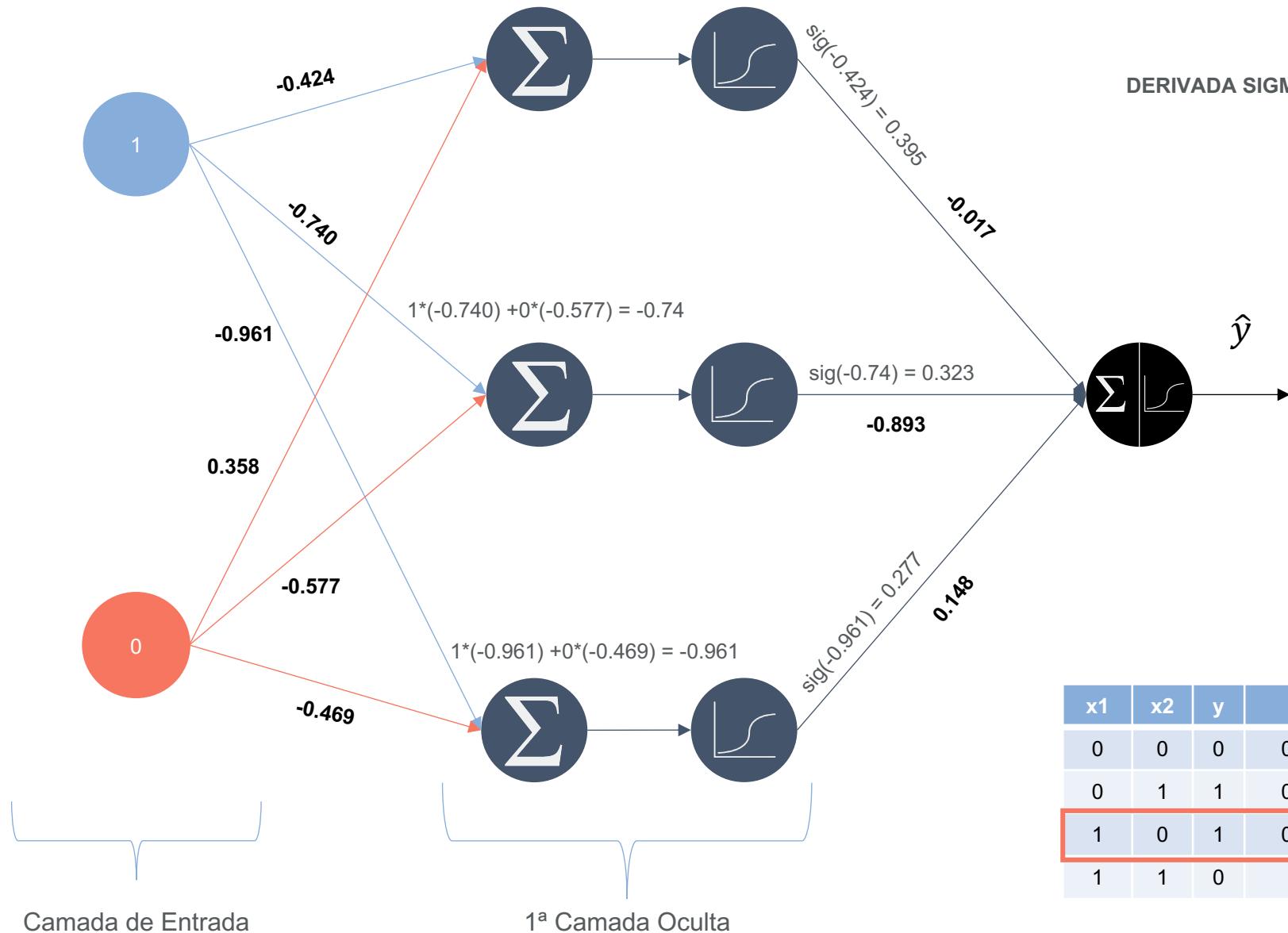
ERRO: $1 - 0.437 = \color{red}{\underline{\underline{0.563}}}$

DERIVADA SIGMOIDE: $y(1-y) = \color{blue}{\underline{\underline{0.437}} * (1 - \underline{\underline{0.437}})} = \color{green}{\underline{\underline{0.246}}}$

DELTA CAMADA SAÍDA:

$$\delta_{\text{saída}} = \text{erro} * \text{derivadaSigmoid}$$

$$\delta_{\text{saída}} = 0.563 * 0.246 = 0.138$$



x_1	x_2	y	\hat{y}	e	$\delta_{\text{saída}}$
0	0	0	0.406	-0.406	-0.098
0	1	1	0.432	0.568	0.139
1	0	1	0.437	0.563	0.138
1	1	0			



CÁLCULOS OCULTA → SAÍDA

$$0.483*(-0.017)+0.211*(-0.893)+0.193*(0.148) = -0.168$$

$$\text{sig}(-0.168) = \color{blue}{0.458}$$

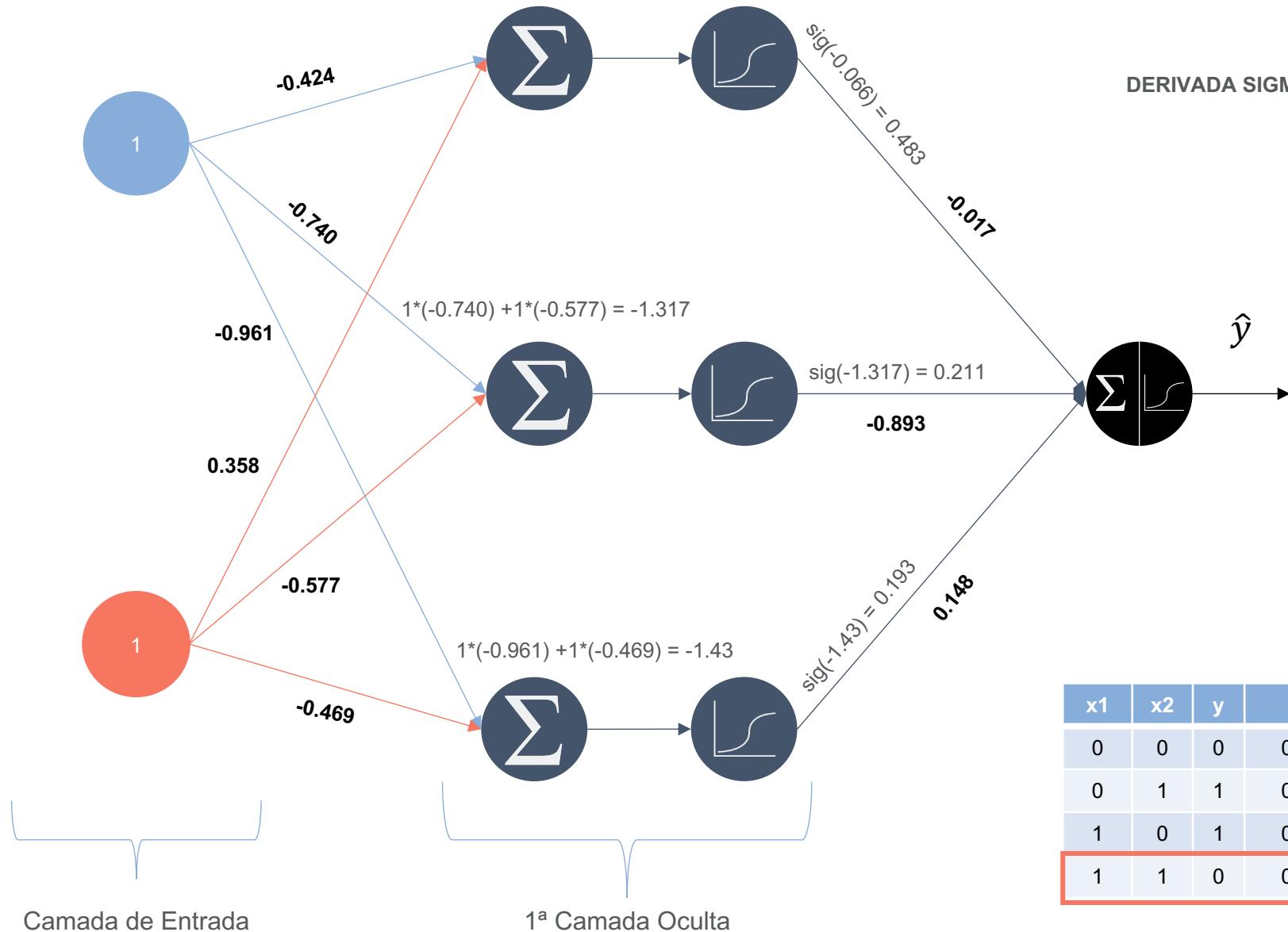
ERRO: $0 - 0.458 = \color{red}{-0.458}$

DERIVADA SIGMOIDE: $y(1-y) = \color{blue}{0.458 * (1 - 0.458)} = \color{green}{0.248}$

DELTA CAMADA SAÍDA:

$$\delta_{\text{saída}} = \text{erro} * \text{derivadaSigmoid}$$

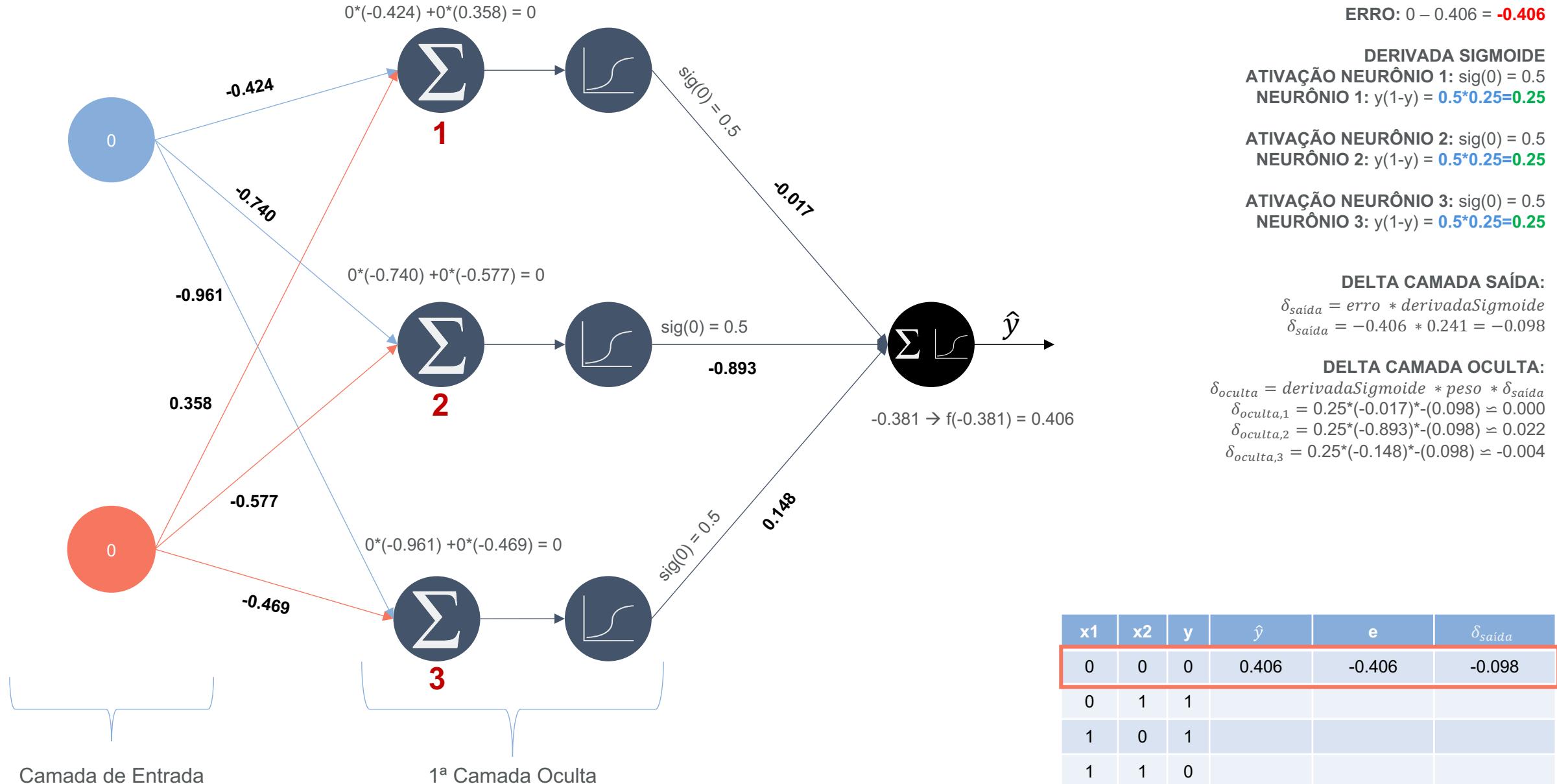
$$\delta_{\text{saída}} = -0.458 * 0.248 = -0.113$$



x1	x2	y	\hat{y}	e	$\delta_{\text{saída}}$
0	0	0	0.406	-0.406	-0.098
0	1	1	0.432	0.568	0.139
1	0	1	0.437	0.563	0.138
1	1	0	0.458	-0.458	-0.113



CÁLCULO DO δ_{oculta}



ERRO: $1 - 0.432 = \textcolor{red}{-0.568}$

DERIVADA SIGMOIDE

ATIVAÇÃO NEURÔNIO 1: $\text{sig}(0.358) = 0.589$
NEURÔNIO 1: $y(1-y) = \textcolor{blue}{0.589 * (1-0.589)} = \textcolor{green}{0.242}$

ATIVAÇÃO NEURÔNIO 2: $\text{sig}(-0.577) = 0.64$
NEURÔNIO 2: $y(1-y) = \textcolor{blue}{0.64 * (1-0.64)} = \textcolor{green}{0.23}$

ATIVAÇÃO NEURÔNIO 3: $\text{sig}(-0.469) = 0.384$
NEURÔNIO 3: $y(1-y) = \textcolor{blue}{0.384 * (1-0.384)} = \textcolor{green}{0.236}$

DELTA CAMADA SAÍDA:

$$\delta_{\text{saída}} = \text{erro} * \text{derivadaSigmoide}$$

$$\delta_{\text{saída}} = 0.568 * 0.245 = \textcolor{purple}{0.139}$$

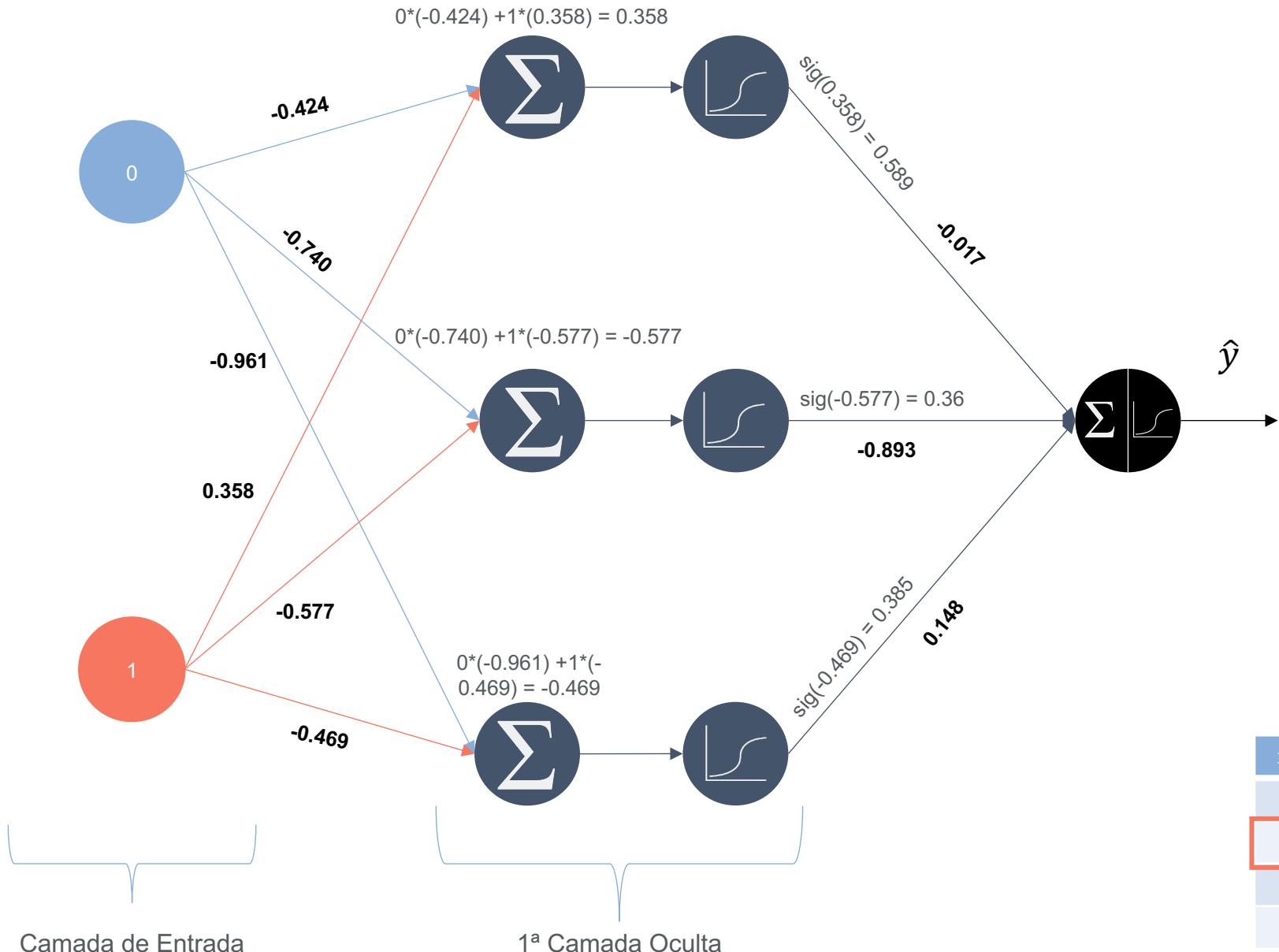
DELTA CAMADA OCULTA:

$$\delta_{\text{oculta}} = \text{derivadaSigmoide} * \text{peso} * \delta_{\text{saída}}$$

$$\delta_{\text{oculta},1} = \textcolor{blue}{0.242} * (-0.017) * (\textcolor{purple}{0.139}) \doteq -0.001$$

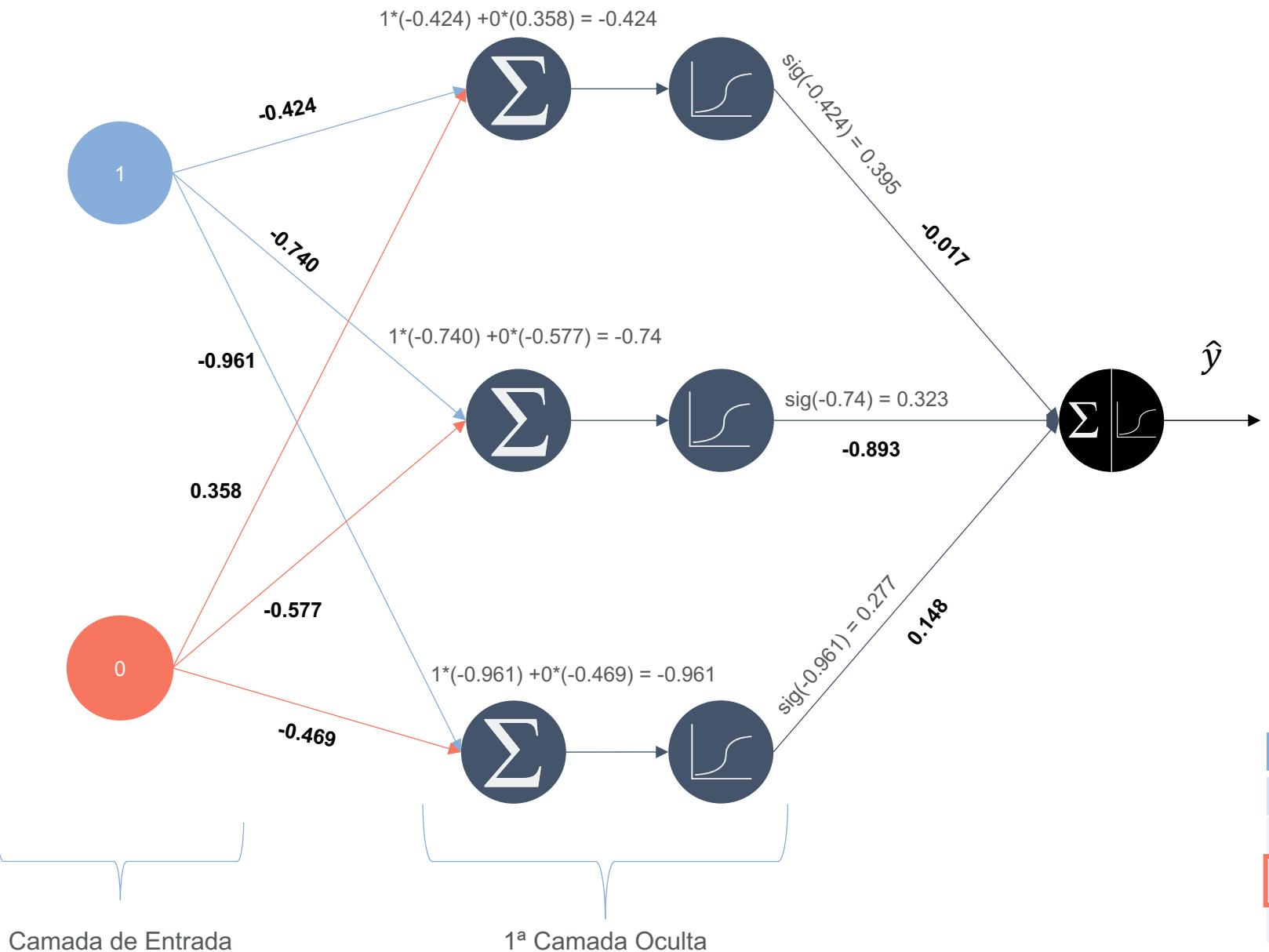
$$\delta_{\text{oculta},2} = \textcolor{blue}{0.23} * (-0.893) * (\textcolor{purple}{0.139}) \doteq -0.029$$

$$\delta_{\text{oculta},3} = \textcolor{blue}{0.236} * (-0.148) * (\textcolor{purple}{0.139}) \doteq 0.005$$



x1	x2	y	\hat{y}	e	$\delta_{\text{saída}}$
0	0	0	0.406	-0.406	-0.098
0	1	1	0.432	0.568	0.139
1	0	1			
1	1	0			





$$\text{ERRO: } 1 - 0.437 = \textcolor{red}{0.563}$$

DERIVADA SIGMOIDE

$$\begin{aligned}\text{ATIVAÇÃO NEURÔNIO 1: } & \text{sig}(-0.424) = 0.396 \\ \text{NEURÔNIO 1: } & y(1-y) = \textcolor{blue}{0.396 * (1-0.396)} = 0.239\end{aligned}$$

$$\begin{aligned}\text{ATIVAÇÃO NEURÔNIO 2: } & \text{sig}(-0.74) = 0.323 \\ \text{NEURÔNIO 2: } & y(1-y) = \textcolor{blue}{0.323 * (1-0.323)} = 0.219\end{aligned}$$

$$\begin{aligned}\text{ATIVAÇÃO NEURÔNIO 3: } & \text{sig}(-0.961) = 0.277 \\ \text{NEURÔNIO 3: } & y(1-y) = \textcolor{blue}{0.277 * (1-0.277)} = 0.200\end{aligned}$$

DELTA CAMADA SAÍDA:

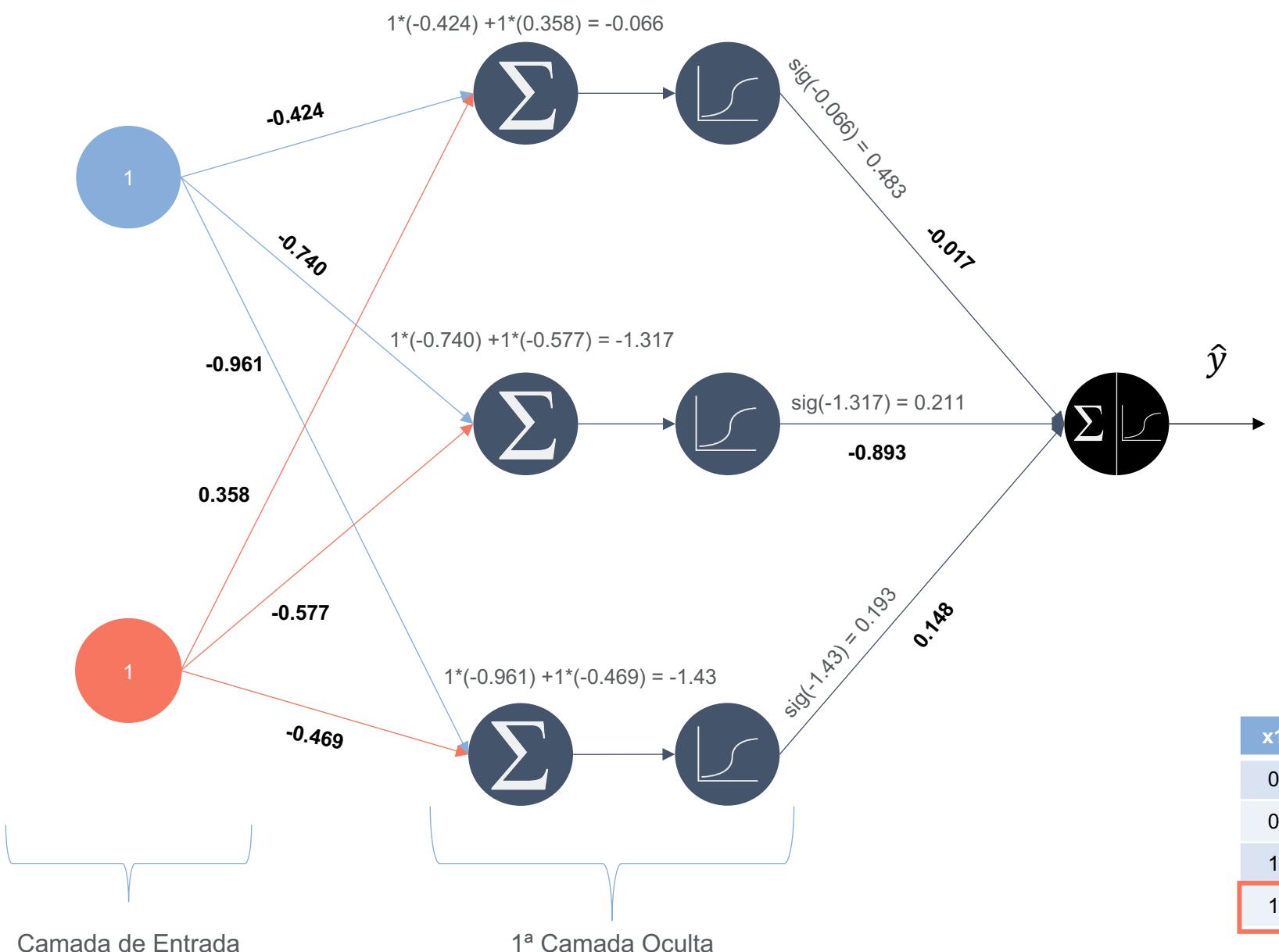
$$\begin{aligned}\delta_{\text{saída}} &= \text{erro} * \text{derivadaSigmoid} \\ \delta_{\text{saída}} &= 0.563 * 0.246 = \textcolor{blue}{0.138}\end{aligned}$$

DELTA CAMADA OCULTA:

$$\begin{aligned}\delta_{\text{oculta}} &= \text{derivadaSigmoid} * \text{peso} * \delta_{\text{saída}} \\ \delta_{\text{oculta},1} &= \textcolor{green}{0.239} * (-0.017) * \textcolor{blue}{(0.138)} \doteq -0.001 \\ \delta_{\text{oculta},2} &= \textcolor{green}{0.219} * (-0.893) * \textcolor{blue}{(0.138)} \doteq -0.027 \\ \delta_{\text{oculta},3} &= \textcolor{green}{0.2} * (-0.148) * \textcolor{blue}{(0.138)} \doteq 0.004\end{aligned}$$

x1	x2	y	\hat{y}	e	$\delta_{\text{saída}}$
0	0	0	0.406	-0.406	-0.098
0	1	1	0.432	0.568	0.139
1	0	1	0.437	0.563	0.138
1	1	0			





ERRO: $0 - 0.458 = \textcolor{red}{-0.458}$

DERIVADA SIGMOIDE

ATIVAÇÃO NEURÔNIO 1: $\text{sig}(-0.066) = 0.484$
NEURÔNIO 1: $y(1-y) = \textcolor{blue}{0.484 * (1-0.484)} = \textcolor{green}{0.249}$

ATIVAÇÃO NEURÔNIO 2: $\text{sig}(-1.317) = 0.211$
NEURÔNIO 2: $y(1-y) = \textcolor{blue}{0.211 * (1-0.211)} = \textcolor{green}{0.167}$

ATIVAÇÃO NEURÔNIO 3: $\text{sig}(-1.43) = 0.193$
NEURÔNIO 3: $y(1-y) = \textcolor{blue}{0.193 * (1-0.193)} = \textcolor{green}{0.156}$

DELTA CAMADA SAÍDA:

$$\delta_{\text{saída}} = \text{erro} * \text{derivadaSigmoid}$$

$$\delta_{\text{saída}} = -0.458 * 0.248 = \textcolor{purple}{-0.113}$$

DELTA CAMADA OCULTA:

$$\delta_{\text{oculta}} = \text{derivadaSigmoid} * \text{peso} * \delta_{\text{saída}}$$

$$\delta_{\text{oculta},1} = \textcolor{green}{0.249} * (-0.017) * (\textcolor{purple}{-0.113}) \approx -0.000$$

$$\delta_{\text{oculta},2} = \textcolor{green}{0.167} * (-0.893) * (\textcolor{purple}{-0.113}) \approx 0.017$$

$$\delta_{\text{oculta},3} = \textcolor{green}{0.156} * (-0.148) * (\textcolor{purple}{-0.113}) \approx -0.003$$

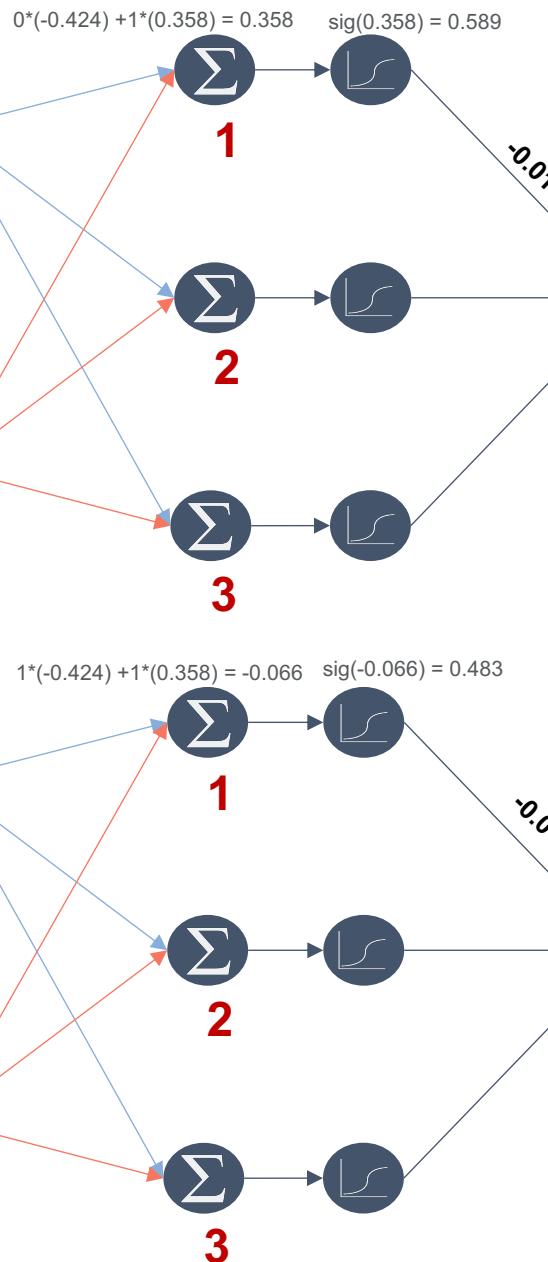
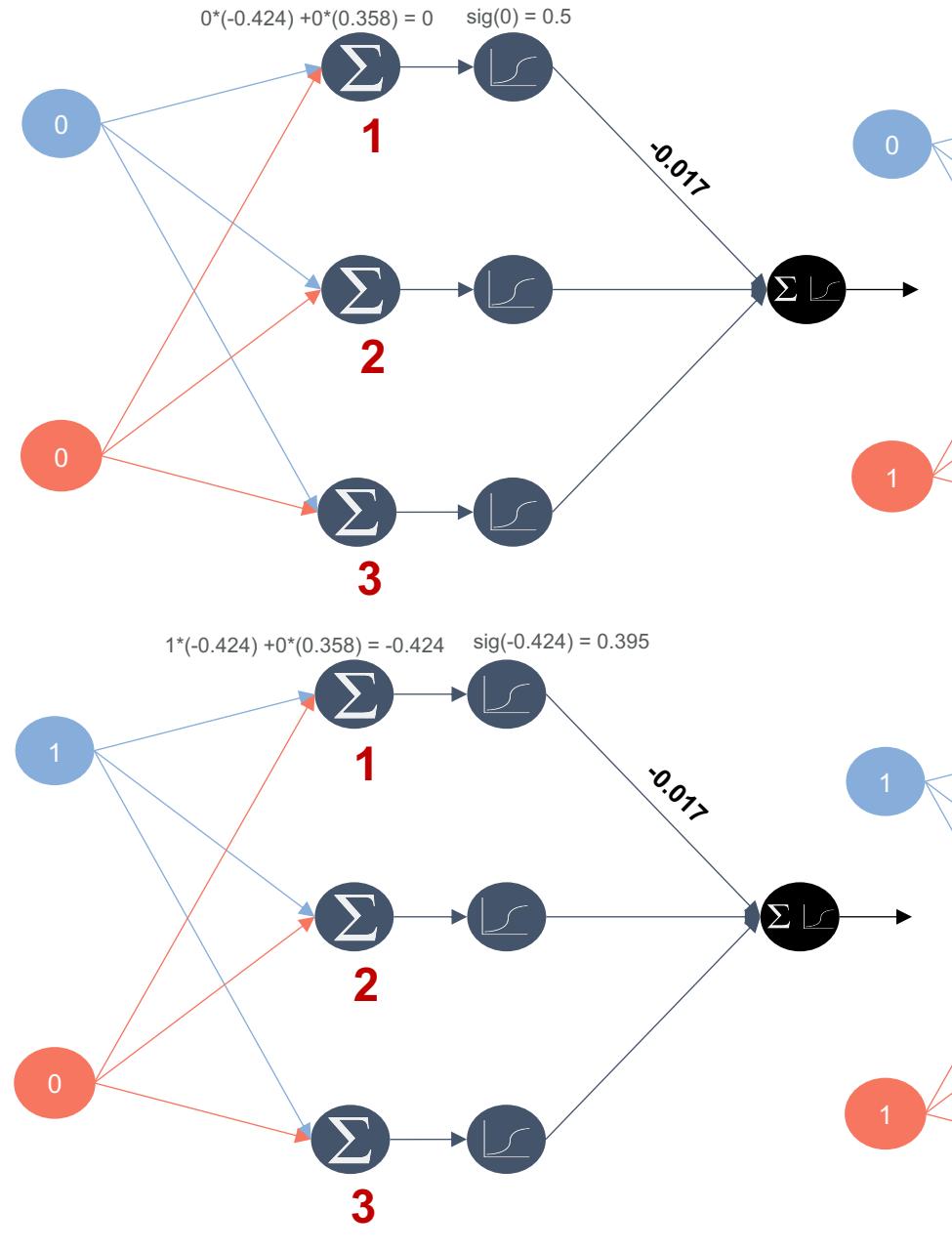
x1	x2	y	\hat{y}	e	$\delta_{\text{saída}}$
0	0	0	0.406	-0.406	-0.098
0	1	1	0.432	0.568	0.139
1	0	1	0.437	0.563	0.138
1	1	0	0.458	-0.458	-0.113



AJUSTE DOS PESOS

CAMADA OCULTA PARA CAMADA DE SAÍDA
NEURÔNIO 1





$peso_{n+1} = (peso_n * momento) + (entrada * \delta * taxaAprendizagem)$
 $\alpha = 0.3$
 $memento = 1$

Para o registro $x1=0, x2=0$:
Entrada = 0.5
 $\delta_{saída} = -0.406 * 0.241 = -0.098$

Para o registro $x1=1, x2=0$:
Entrada = 0.395
 $\delta_{saída} = 0.563 * 0.246 = 0.138$

Para o registro $x1=0, x2=1$:
Entrada = 0.589
 $\delta_{saída} = 0.568 * 0.245 = 0.139$

Para o registro $x1=1, x2=1$:
Entrada = 0.483
 $\delta_{saída} = -0.458 * 0.248 = -0.113$

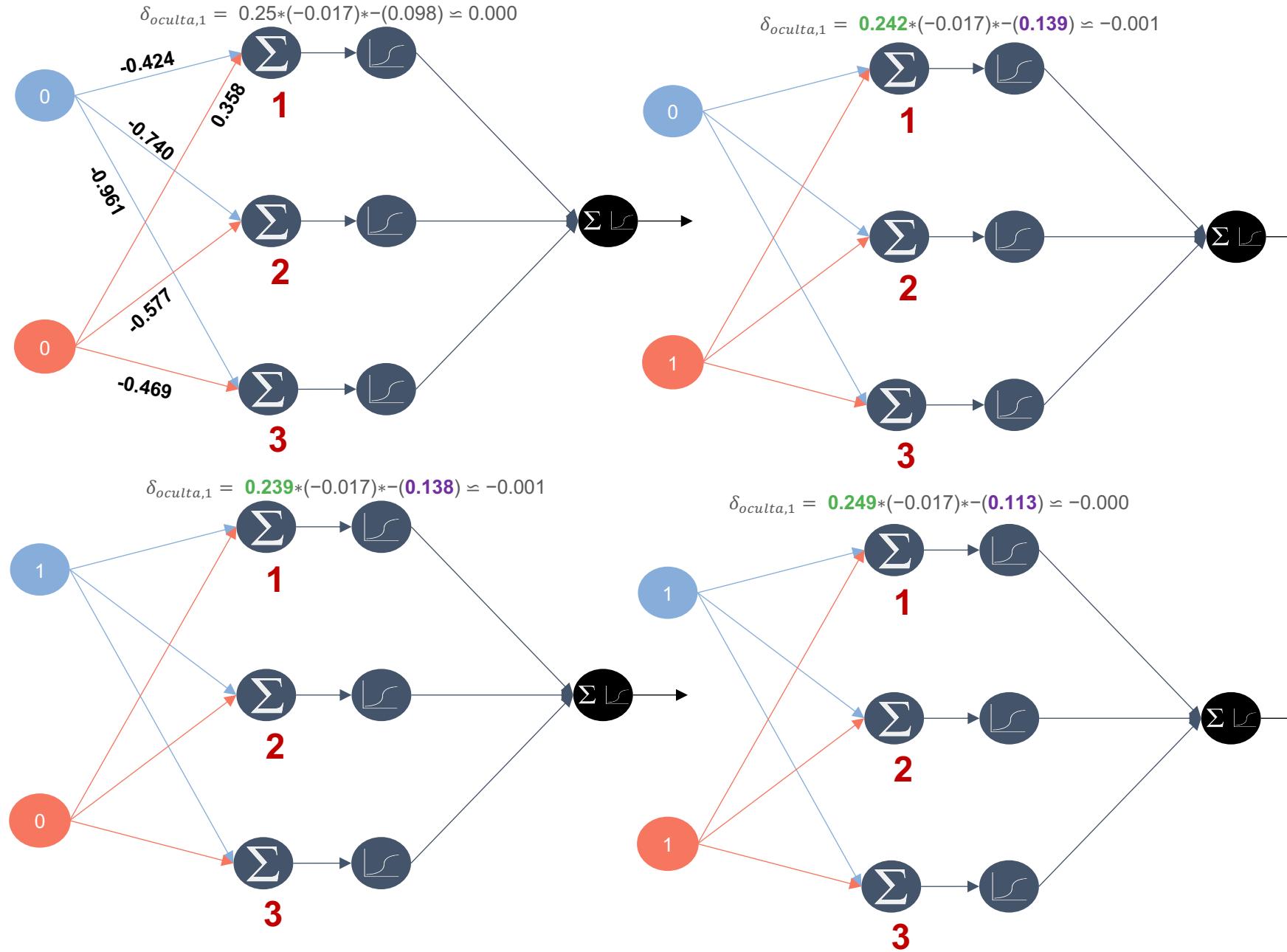
$\sum entrada_i * \delta_{saída,i} =$
 $0.5 * (-0.098) +$
 $0.395 * (0.138) +$
 $0.589 * (0.139) +$
 $0.483 * (-0.113) = 0.032$

$peso_{n+1} = (peso_n * momento) + (entrada * \delta * taxaAprendizagem)$
 $peso_{n+1} = (-0.017 * 1) + (0.032 * 0.3)$
 $peso_{n+1} = 0.0074$

AJUSTE DOS PESOS

CAMADA DE ENTRADA PARA CAMADA OCULTA
NEURÔNIO 1





UFA...



Bibliotecas e Ferramentas

Bibliotecas e Ferramentas

Conceitos Básicos de Aprendizado de Máquina

Objetivos do Módulo:

- Demonstrar como configurar um ambiente virtual de desenvolvimento para o aprendizado de máquina.
- Introduzir as bibliotecas TensorFlow e Keras para a solução do problema XOR;



Bibliotecas e Ferramentas

Bibliotecas e Ferramentas – Configuração do Ambiente

- Nosso curso será conduzido no **Google Colab**, através da criação de notebooks facilmente compartilháveis e pela facilidade em realizar a configuração.
- Google Colab é um Jupyter Notebook hospedado na nuvem, que não requer configuração para utilizar e entrega recursos gratuitos como utilização de GPUs e TPUs
- Google disponibiliza uma curadoria de notebooks em <https://colab.google/notebooks/>
- Para iniciar um novo notebook → <https://colab.new>
- Comandos são realizados com a utilização do sinalizador “!”. Exemplo: !pip install tensorflow
- É possível inserir textos, executar células individualmente ou em conjunto



Bibliotecas e Ferramentas

Bibliotecas e Ferramentas – TensorFlow

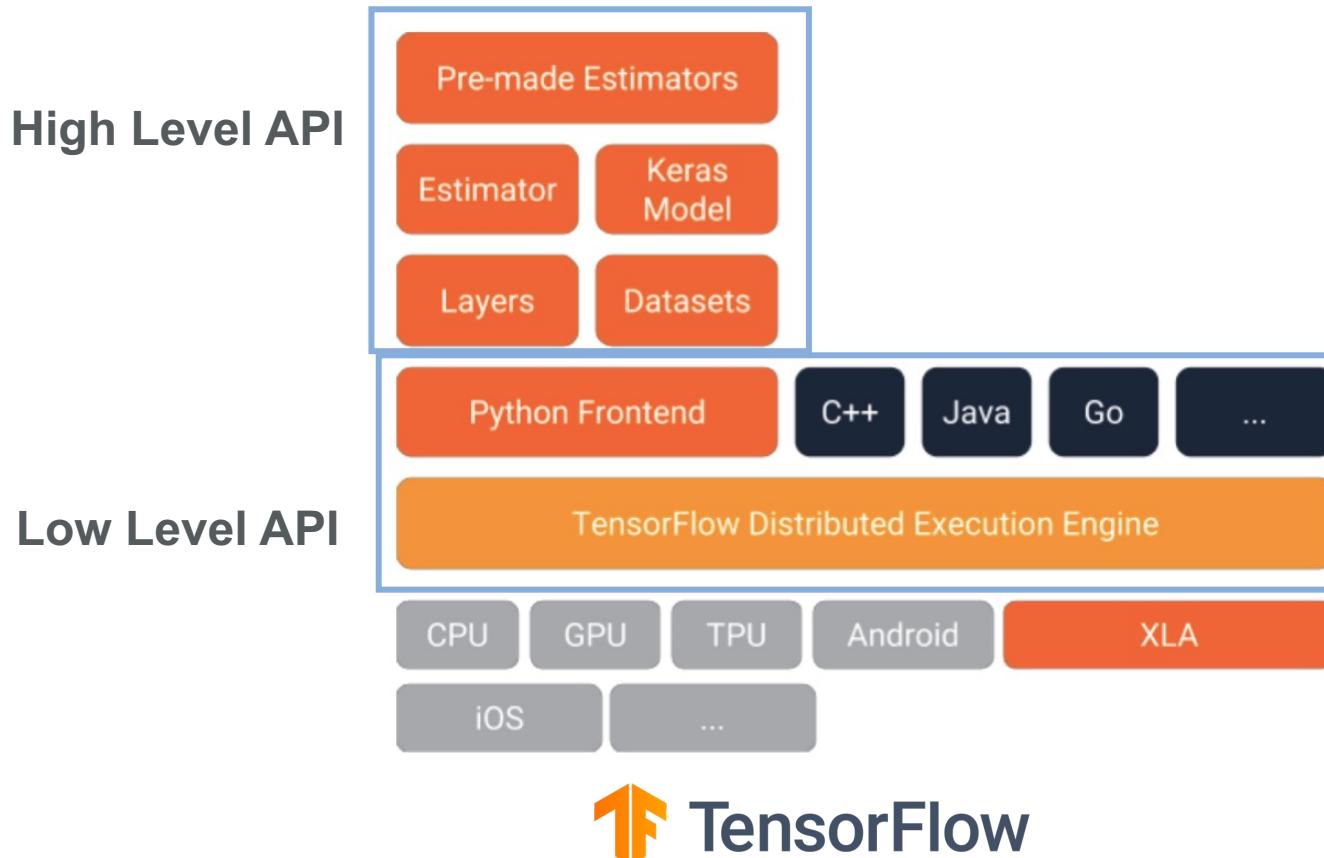
O que é o TensorFlow?

- É uma biblioteca open-source para Machine Learning, aplicável a uma vasta gama de tarefas.
- Permite a criação e treinamento de Redes Neurais Artificiais, sendo otimizada para computação numérica. Algumas partes do código rodam em C++
- Biblioteca mais eficiente para Deep Learning
- Vasta comunidade de apoio
- Inicialmente desenvolvido pela Google Brain (equipe de IA e Deep Learning do Google) para uso interno, sendo disponibilizado para o público em 2015. Mais informações sobre o Google Brain em <https://ai.google/>
- Página oficial da biblioteca: <https://www.tensorflow.org/>
- Documentação disponível em: <https://www.tensorflow.org/guide>
- Pré-instalada no Google Colab



Bibliotecas e Ferramentas

Bibliotecas e Ferramentas – TensorFlow



Bibliotecas e Ferramentas

Bibliotecas e Ferramentas – Keras

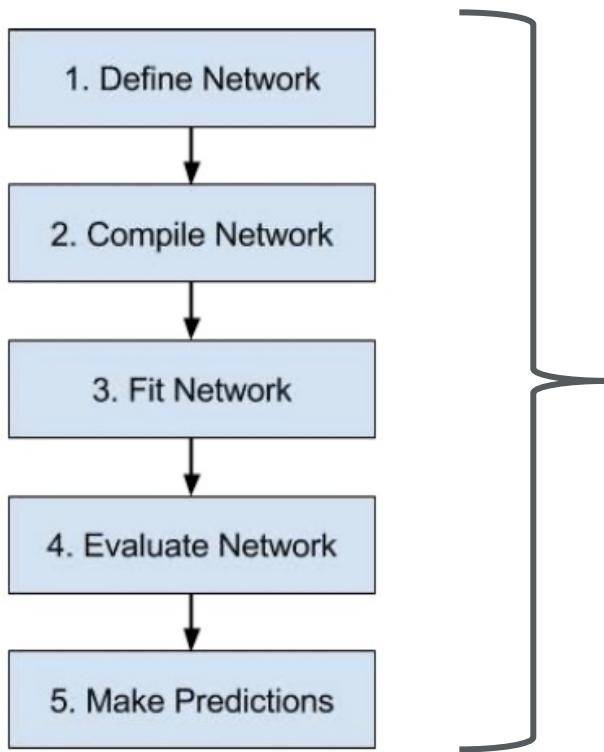
O que é o Keras?

- É uma biblioteca open-source para Machine Learning criada para prototipagem rápida de Redes Neurais Artificiais profundas. É considerada fácil de usar, modular e extensível.
- Capaz de trabalhar em cima do TensorFlow, atuando como uma interface. Ou seja: não é uma biblioteca para ML independente, ao contrário do TensorFlow.
- Página oficial da biblioteca: <https://keras.io/>
- Documentação disponível em: <https://keras.io/guides/>
- Pré-instalada no Google Colab



Bibliotecas e Ferramentas

Bibliotecas e Ferramentas – Keras

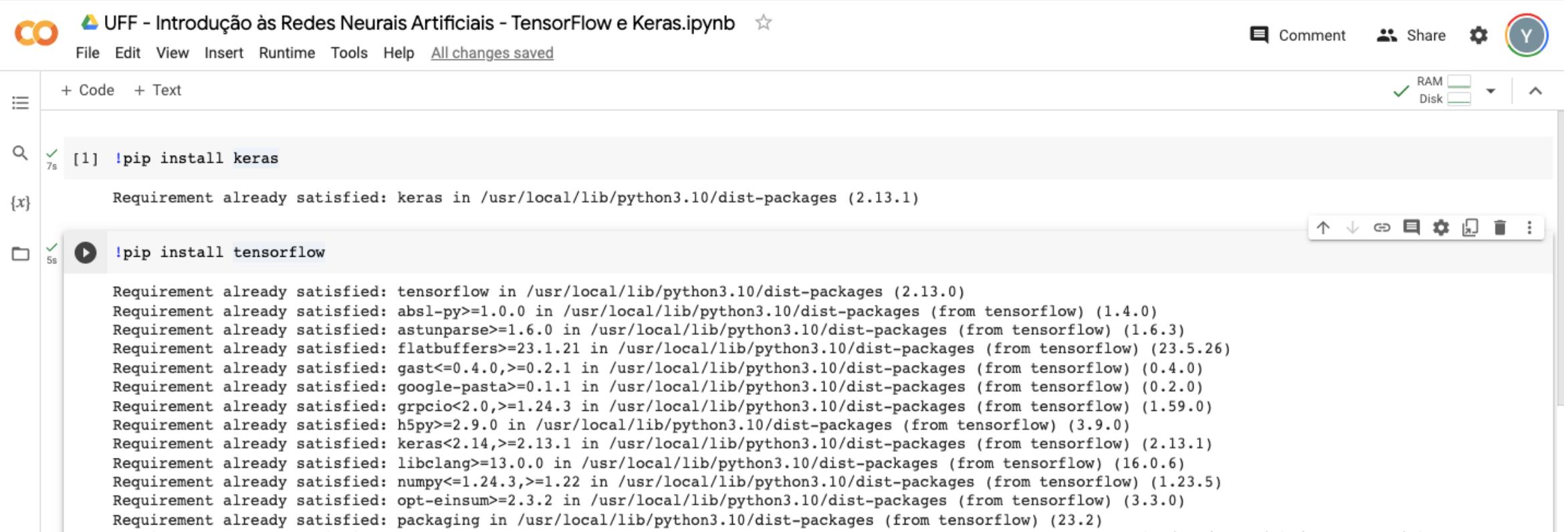


Ciclo de Vida para Modelos de Redes Neurais no Keras

K Keras

Bibliotecas e Ferramentas

Bibliotecas e Ferramentas – Configuração do Ambiente: TensorFlow e Keras



The screenshot shows a Google Colab notebook titled "UFF - Introdução às Redes Neurais Artificiais - TensorFlow e Keras.ipynb". The notebook interface includes a toolbar with File, Edit, View, Insert, Runtime, Tools, Help, and a status bar indicating "All changes saved". The code cell contains two commands: "!pip install keras" and "!pip install tensorflow". The output of the first command shows that keras is already satisfied at version 2.13.1. The output of the second command lists numerous dependencies that are already satisfied, including tensorflow, absl-py, astunparse, flatbuffers, gast, google-pasta, grpcio, h5py, keras, libclang, numpy, opt-einsum, and packaging, all at specific versions.

```
+ Code + Text
```

[1] `!pip install keras`
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages (2.13.1)

{x} [2] `!pip install tensorflow`
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.13.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.1.21 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.59.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: keras<2.14,>=2.13.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.13.1)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)
Requirement already satisfied: numpy<=1.24.3,>=1.22 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)

Bibliotecas e Ferramentas

Bibliotecas e Ferramentas – Conceitos Básicos para TensorFlow/Keras

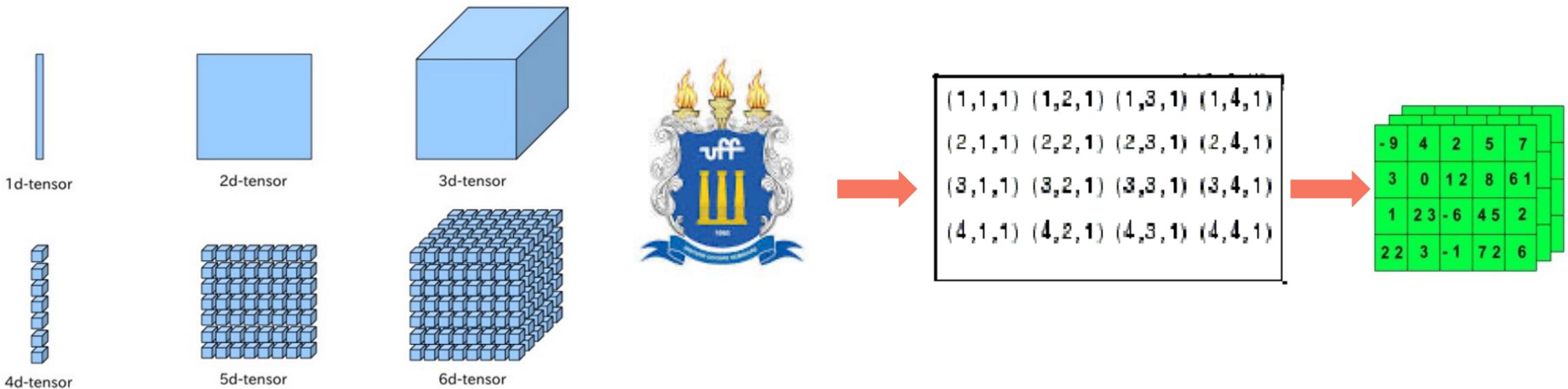
- Escalar: 2
- Vetor: [1, 3, 7, 8]
- Matriz 3x3:
 - Representação normal:
[1 3 7]
[3 1 2]
[2 0 1]
 - Matriz em Python: matriz = [[1, 3, 7], [3, 1, 2], [2, 0, 1]].
 - Exemplo para acessar o elemento da segunda linha e terceira coluna: matriz[1][2]



Bibliotecas e Ferramentas

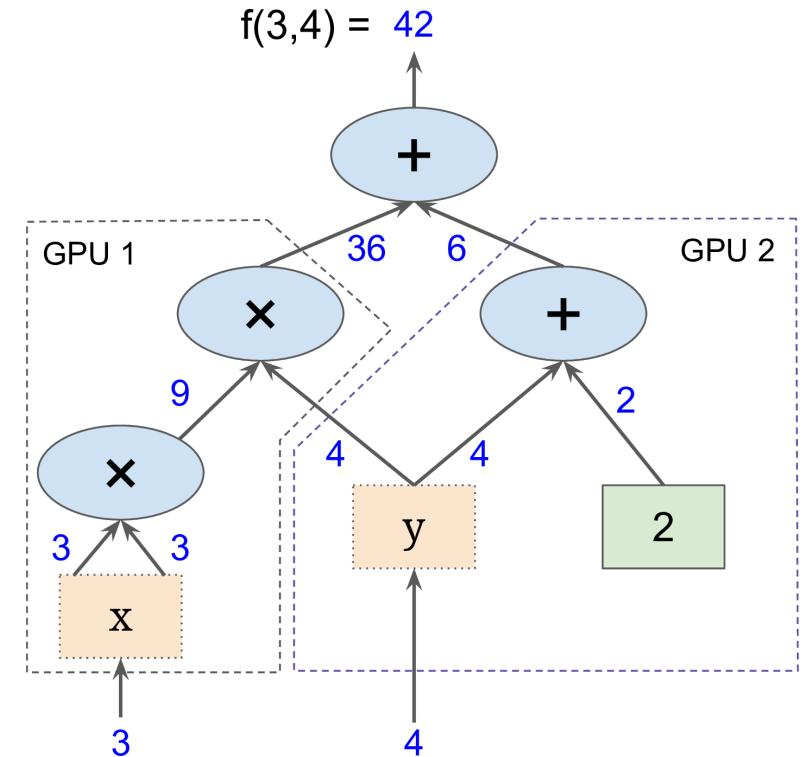
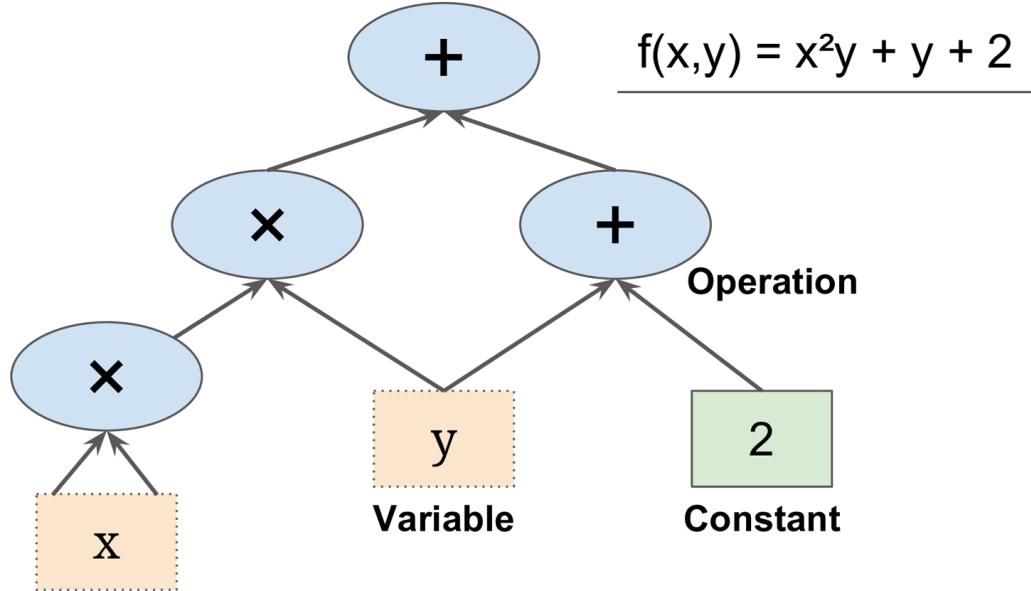
Bibliotecas e Ferramentas – Conceitos Básicos para TensorFlow/Keras

- Tensor → Generalizar escalares, vetores e matrizes (independente de suas dimensões)



Bibliotecas e Ferramentas

Bibliotecas e Ferramentas – Conceitos Básicos para TensorFlow/Keras



TensorFlow Keras

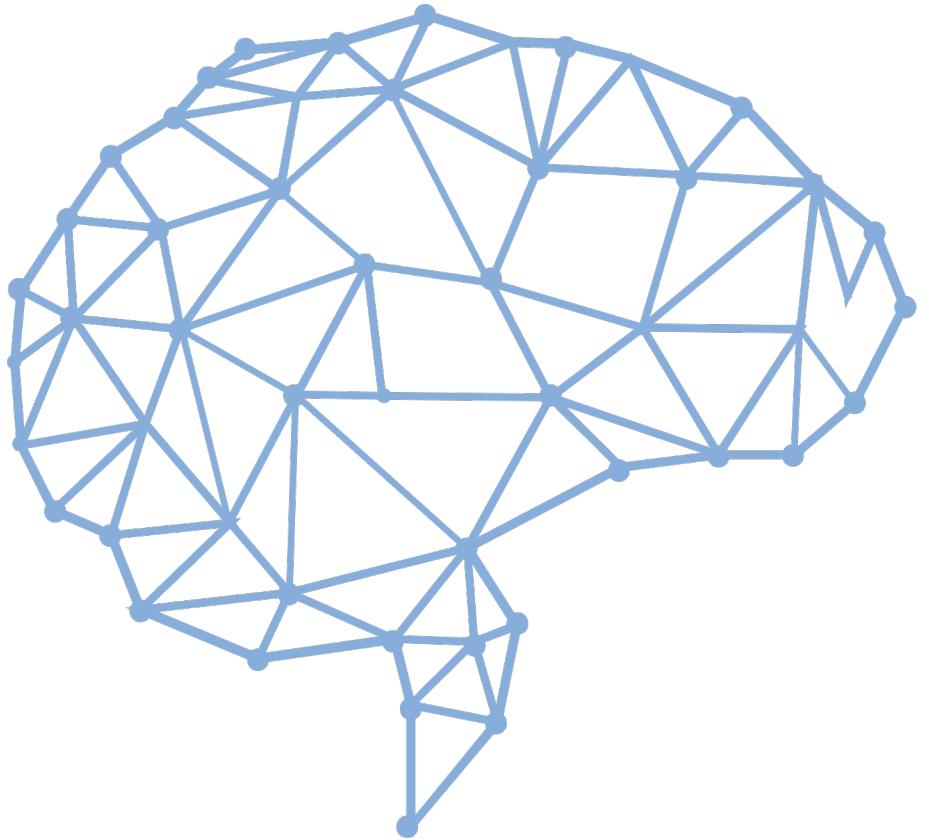


Bibliotecas e Ferramentas

Bibliotecas e Ferramentas – Conceitos Básicos para TensorFlow/Keras

Google Colab → File → Upload Notebook → UFF – Introdução às Redes Neurais Artificiais.ipynb





Encerramento

Encerramento

Encerramento – Índice Remissivo

INTRODUÇÃO AO MACHINE LEARNING

Conceitos Básicos de Aprendizado de Máquina (Machine Learning)

Tipos de Aprendizado de Máquina

O papel das Redes Neurais no Aprendizado de Máquina

NEURÔNIOS ARTIFICIAIS E PERCEPTRONS

O Modelo de Neurônio Artificial

Perceptrons

Funções de Ativação e suas Aplicações

APLICAÇÕES DAS REDES NEURAIS ARTIFICIAIS

Reconhecimento de padrões

Visão Computacional

Outros campos de aplicação

Arquitetura de redes neurais artificiais

REDES NEURAIS ARTIFICIAIS

Camadas ocultas e aprofundamento de redes

Feedforwarding

Backpropagation



Encerramento

Encerramento – Índice Remissivo

Ajuste dos Pesos (gradient descent, delta, taxa de aprendizagem, momento)

BIBLIOTECAS E FERRAMENTAS

Configuração do ambiente de desenvolvimento

Introdução ao TensorFlow e Keras

Definindo a arquitetura da rede

Treinamento e avaliação do modelo

Transfer learning



Obrigado