Introdução ao aprendizado de máquina

Aprendizado de máquina e aprendizado profundo - Lux.AI

INSTITUIÇÃO EXECUTORA

Centro de Informática UFPE

FADE UFPE

COORDENADORA

MCTI FUTURO

Softex

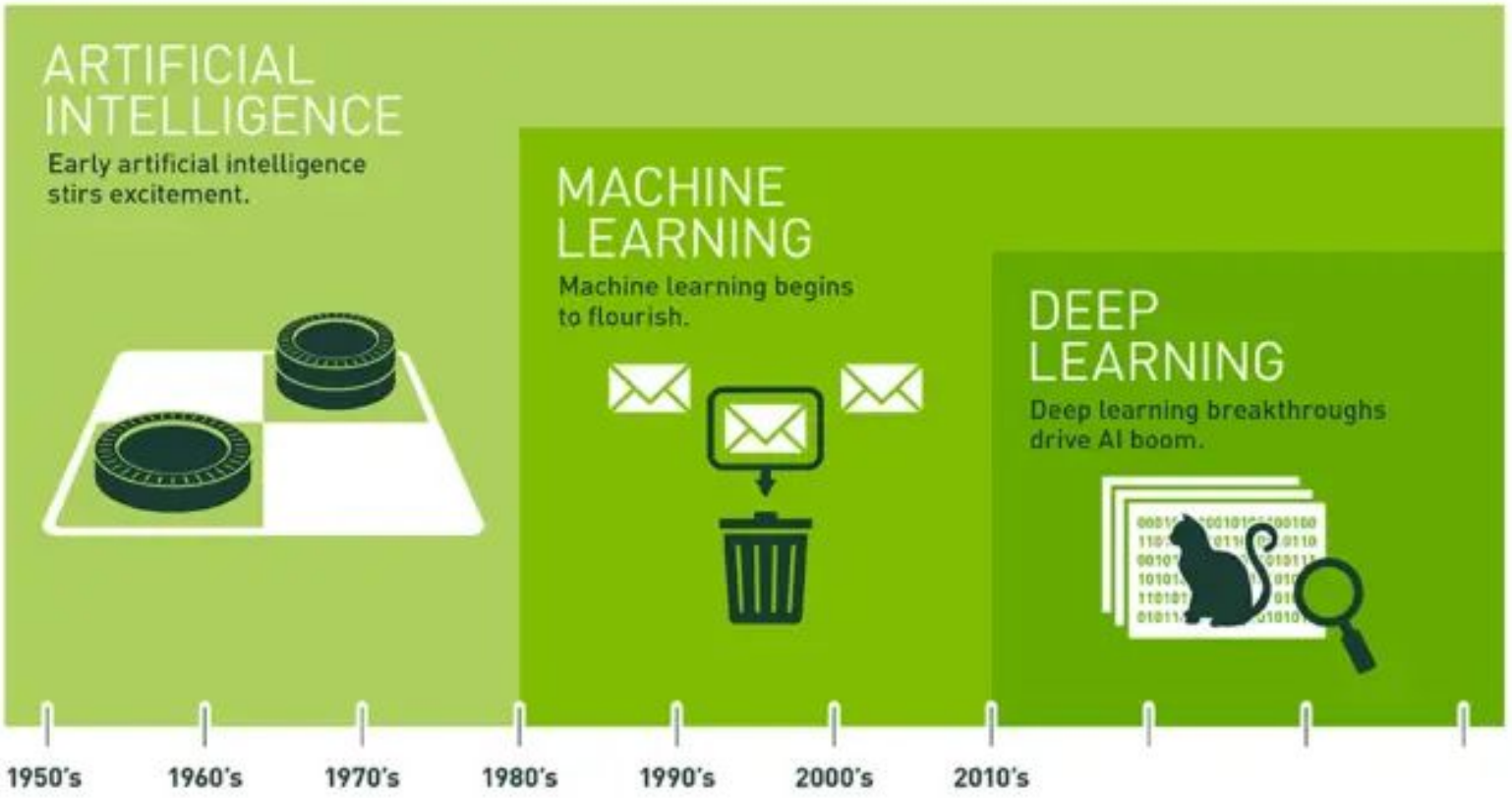
APOIO

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

Introdução - Aprendizado de máquina e profundo

Aprendizado de máquina e profundo



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Fonte: https://blogs.nvidia.com/blog/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

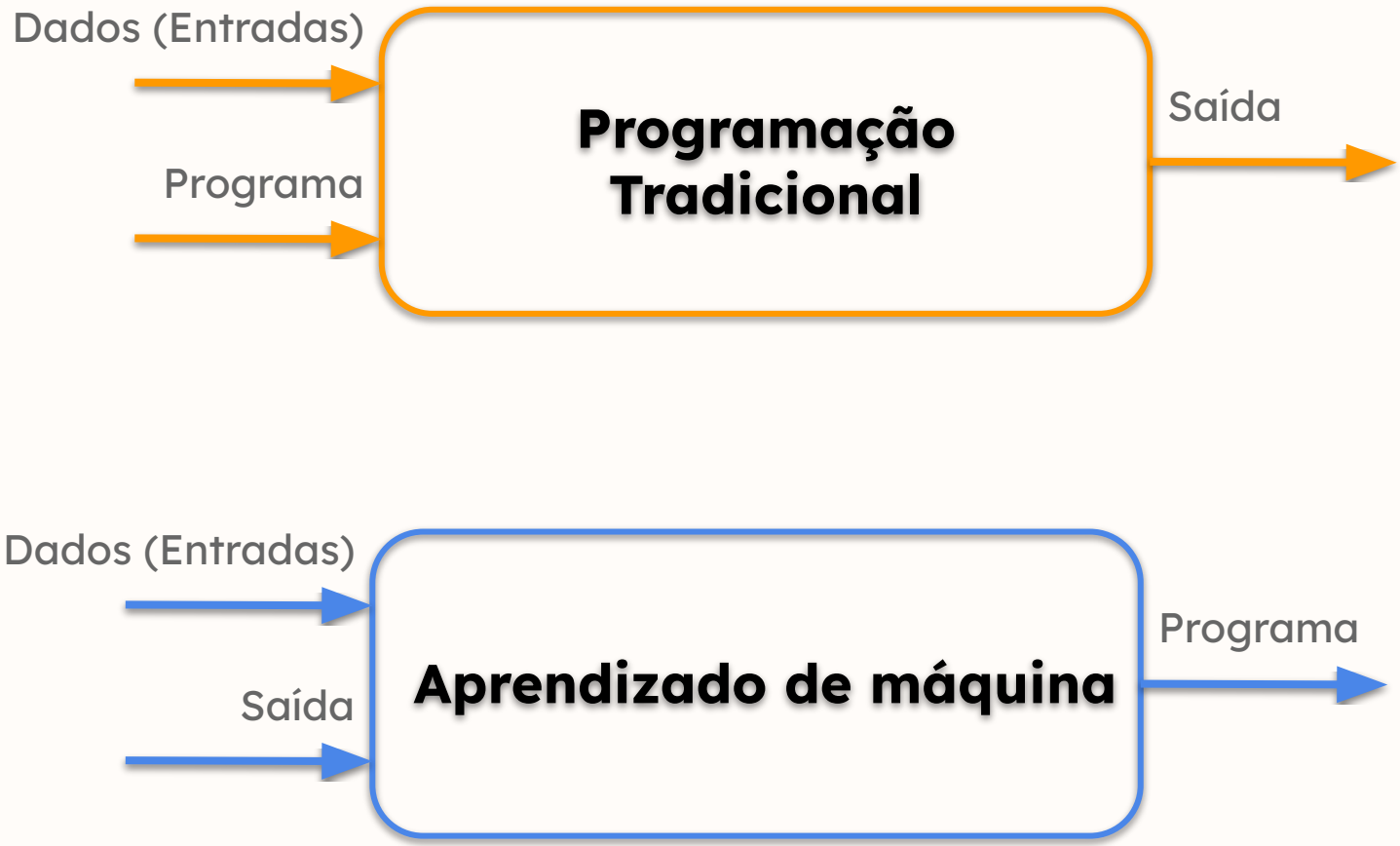
Aprendizado de máquina e profundo

➔ O que é aprendizado de máquina?

- “The field of study that gives computers the ability to **learn without being explicitly programmed**” (Arthur Samuel, 1959)

- “A computer program is said to **learn from experience E with respect to some class of tasks T** and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.” (Tom Mitchell, 1997)

Aprendizado de máquina e profundo



Aprendizado de máquina e profundo

- Modelos de aprendizado aprendem informações sobre o domínio a partir de dados relacionados, realizando estimações para novos casos não vistos anteriormente

# price Price of the Houses	# area Area of a House	# bedrooms Number of House Bedrooms	# bathrooms Number if Bathrooms	# stories Number of House Stories	✓ mainroa Weather c Main Road
1.75m 13.3m	1650 16.2k	1 6	1 4	1 4	
13300000	7420	4	2	3	yes
12250000	8960	4	4	4	yes
12250000	9960	3	2	2	yes
12215000	7500	4	2	2	yes
11410000	7420	4	1	2	yes
10850000	7500	3	3	1	yes
10150000	8580	4	3	4	yes
10150000	16200	5	3	2	yes
9870000	8100	4	1	2	yes

Fonte das imagens: https://www.kaggle.com/datasets/yasserh/housing-prices-dataset

Aprendizado de máquina e profundo

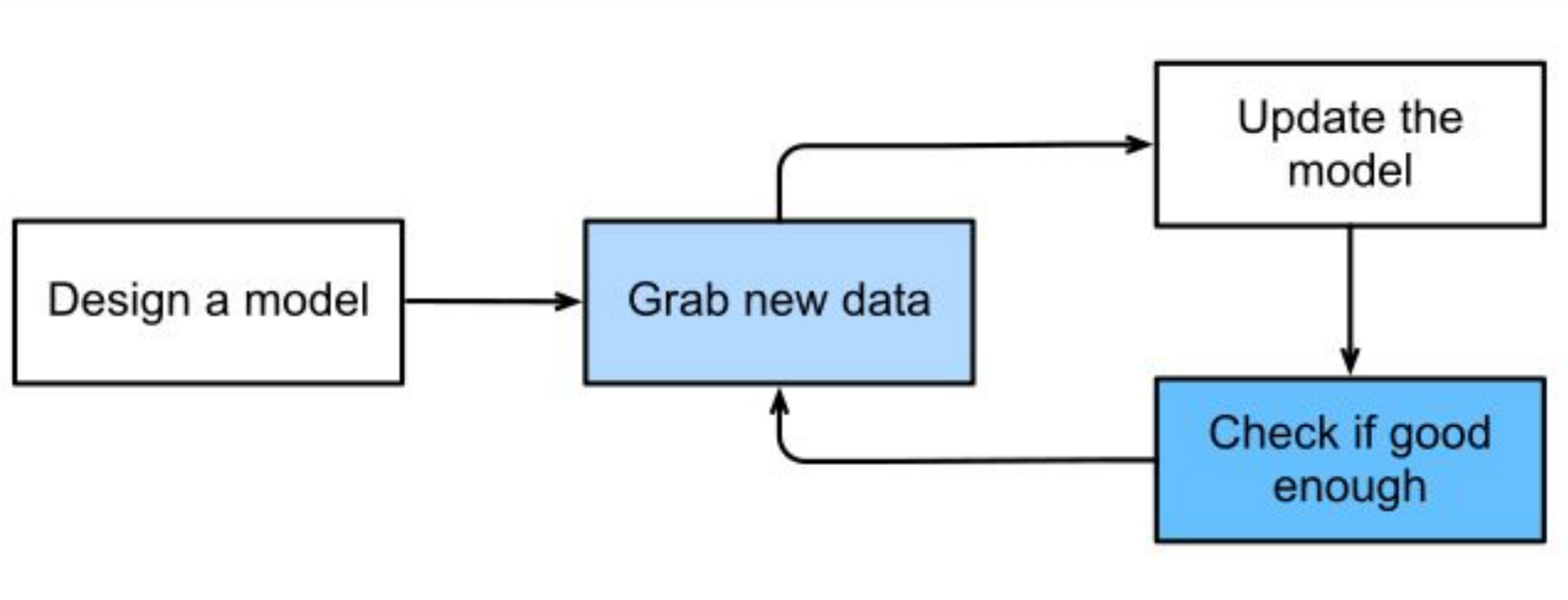
- Modelos de aprendizado aprendem informações sobre o domínio a partir de dados relacionados.

bedrooms	# bathrooms	# stories	✓ mainroa
umber of House edrooms	Number if Bathrooms	Number of House Stories	Weather c Main Road
6	1 4	1 4	
	2	3	yes
	4	4	yes
	2	2	yes
	2	2	yes
	1	2	yes
	3	1	yes
	3	4	yes
5	3	2	yes
4	1	2	yes

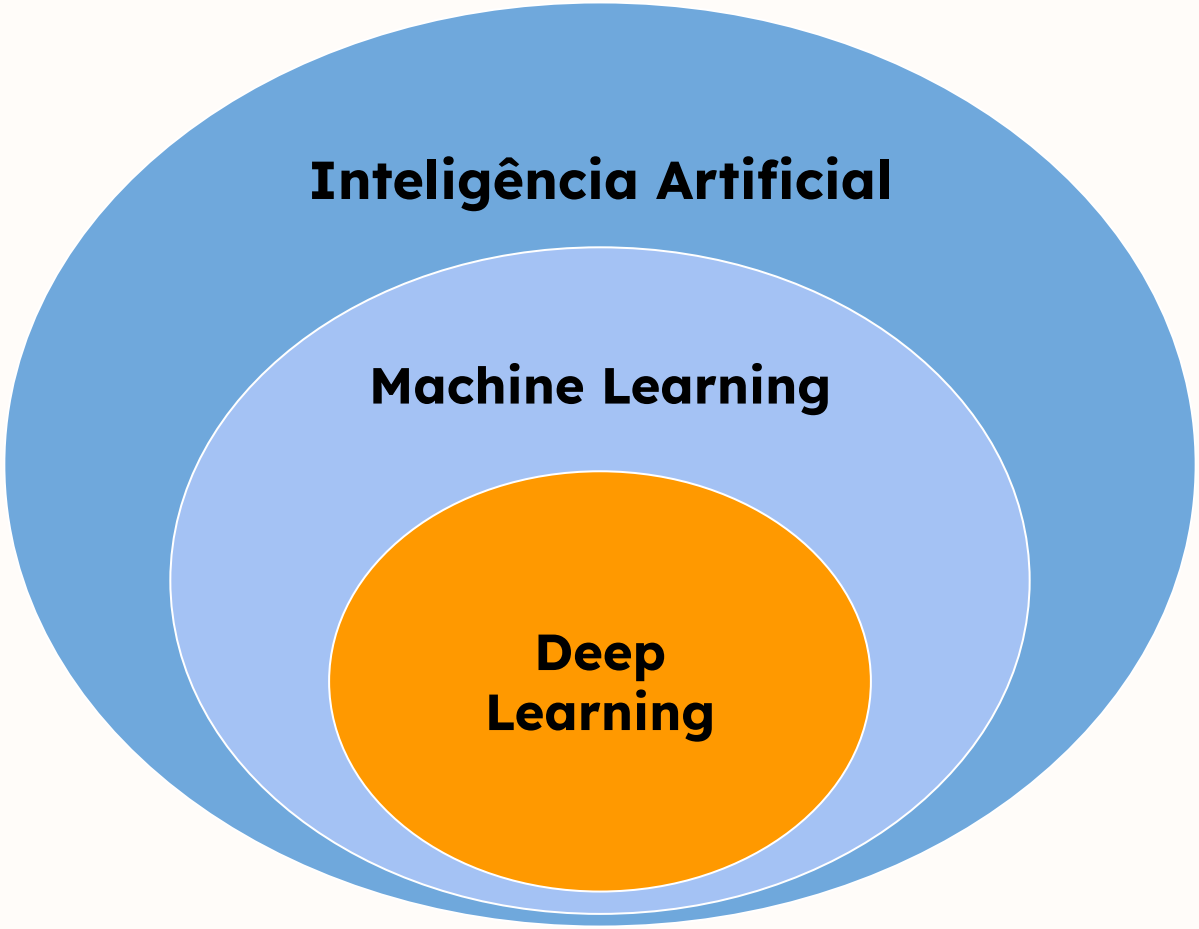
Fonte das imagens: https://www.kaggle.com/datasets/yasserh/housing-prices-dataset

Aprendizado de máquina e profundo

➔ Componentes chaves
- Dados
- Modelo
- Função Objetivo
- Otimizador

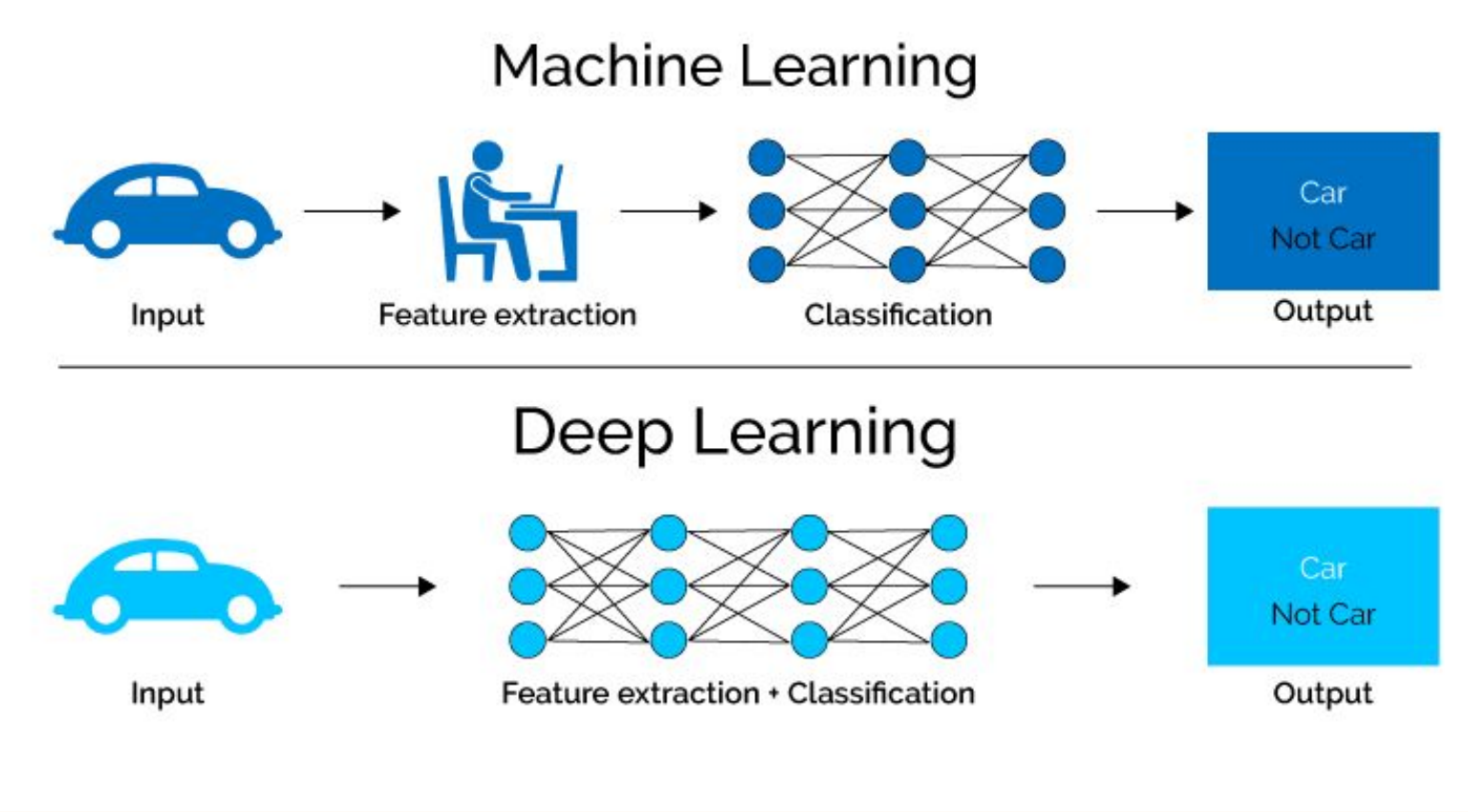


Aprendizado de máquina e profundo - ML vs DL



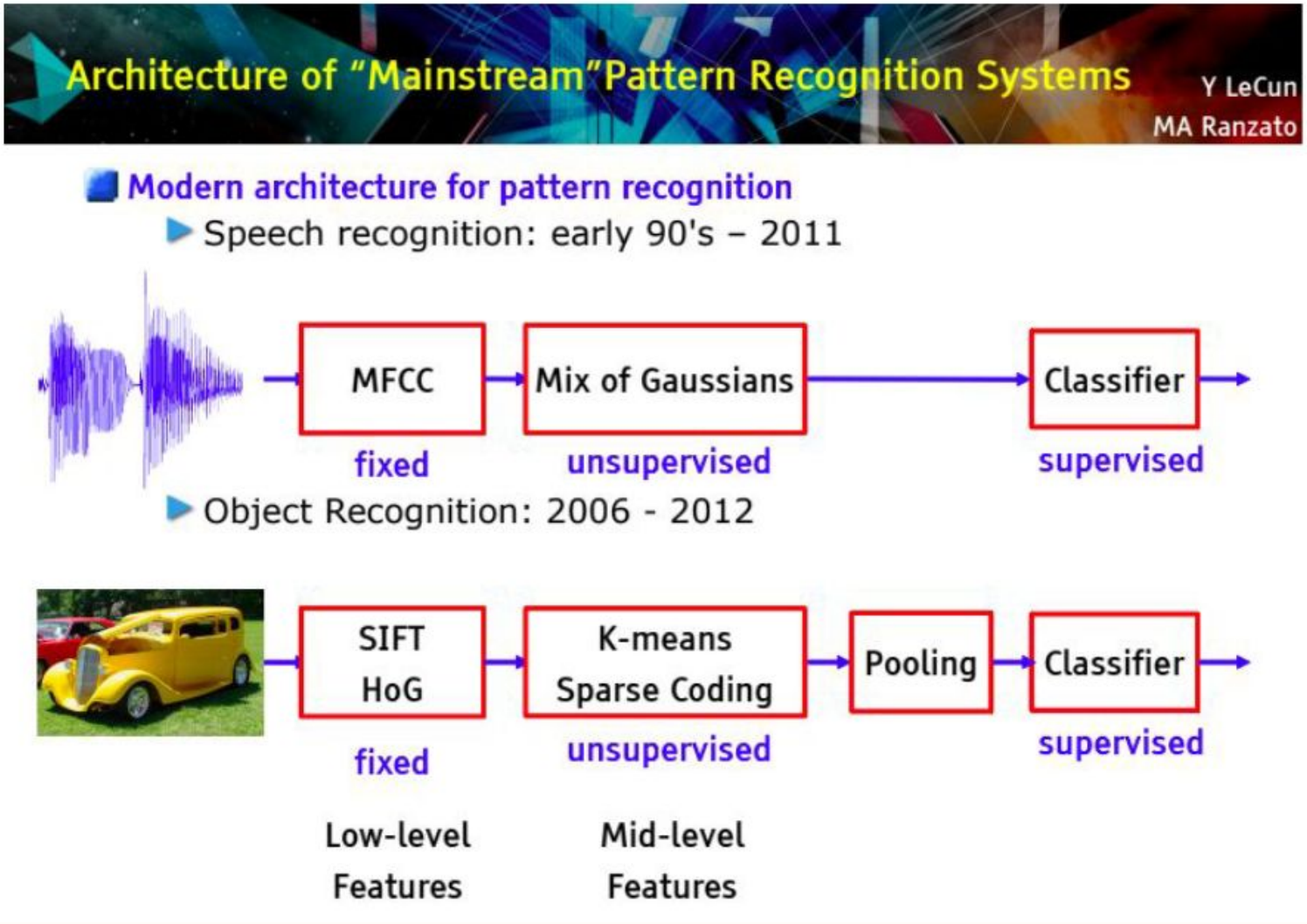
https://semiengineering.com/deep-learning-spreads/

Aprendizado de máquina e profundo - ML vs DL



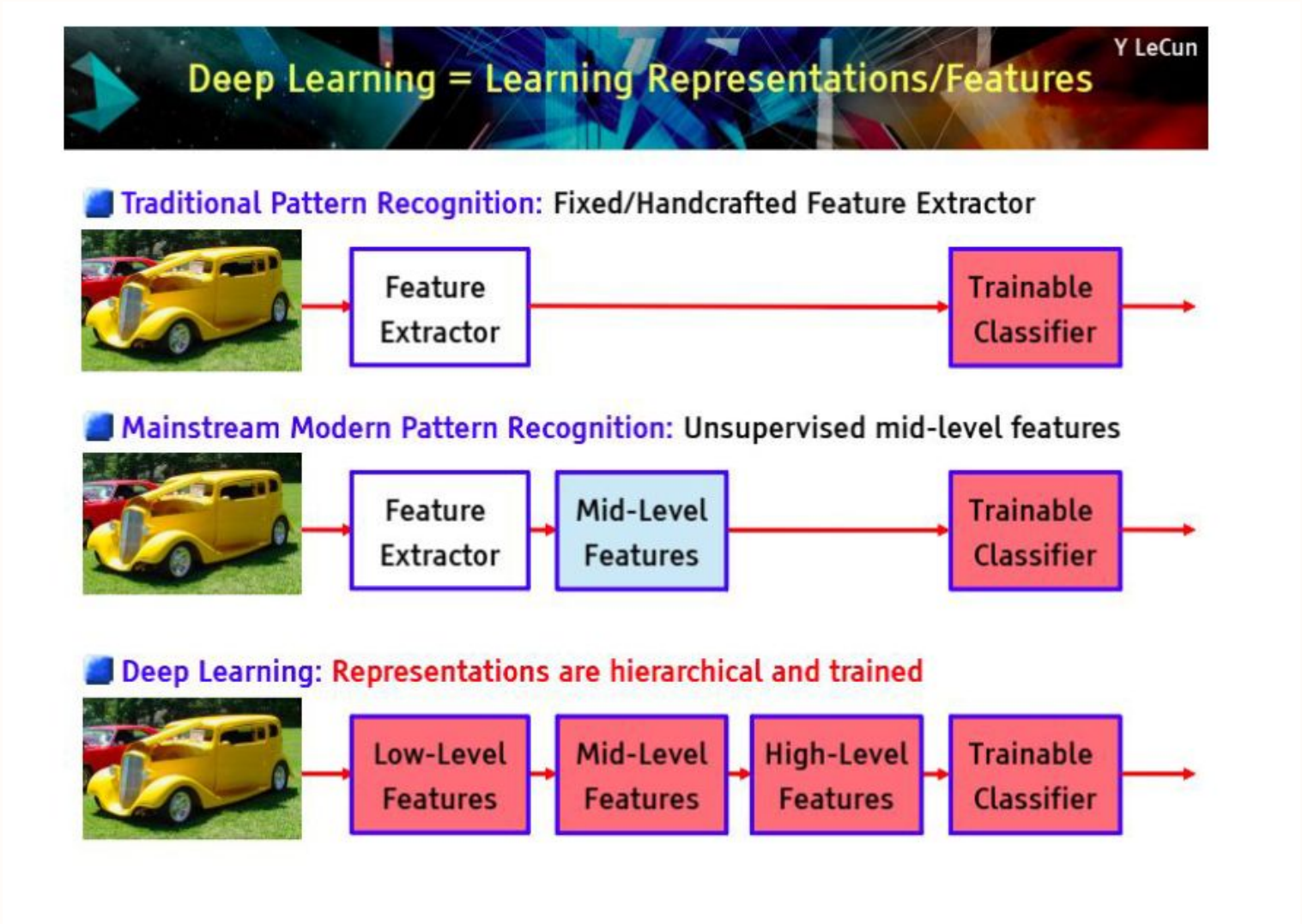
Fonte: https://semiengineering.com/deep-learning-spreads/

Aprendizado de máquina e profundo - ML vs DL



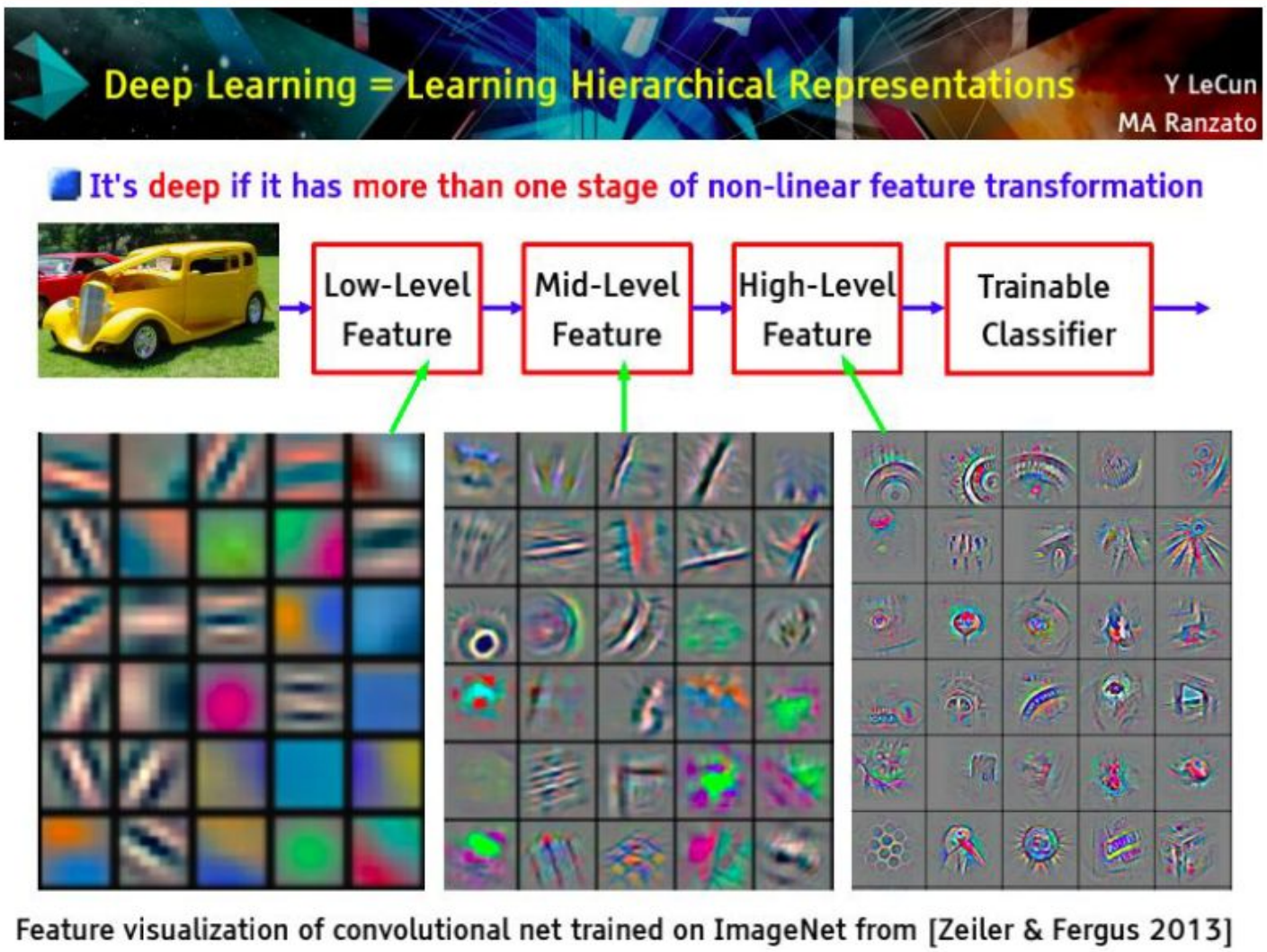
Fonte: yann.lecun.com

Aprendizado de máquina e profundo - ML vs DL



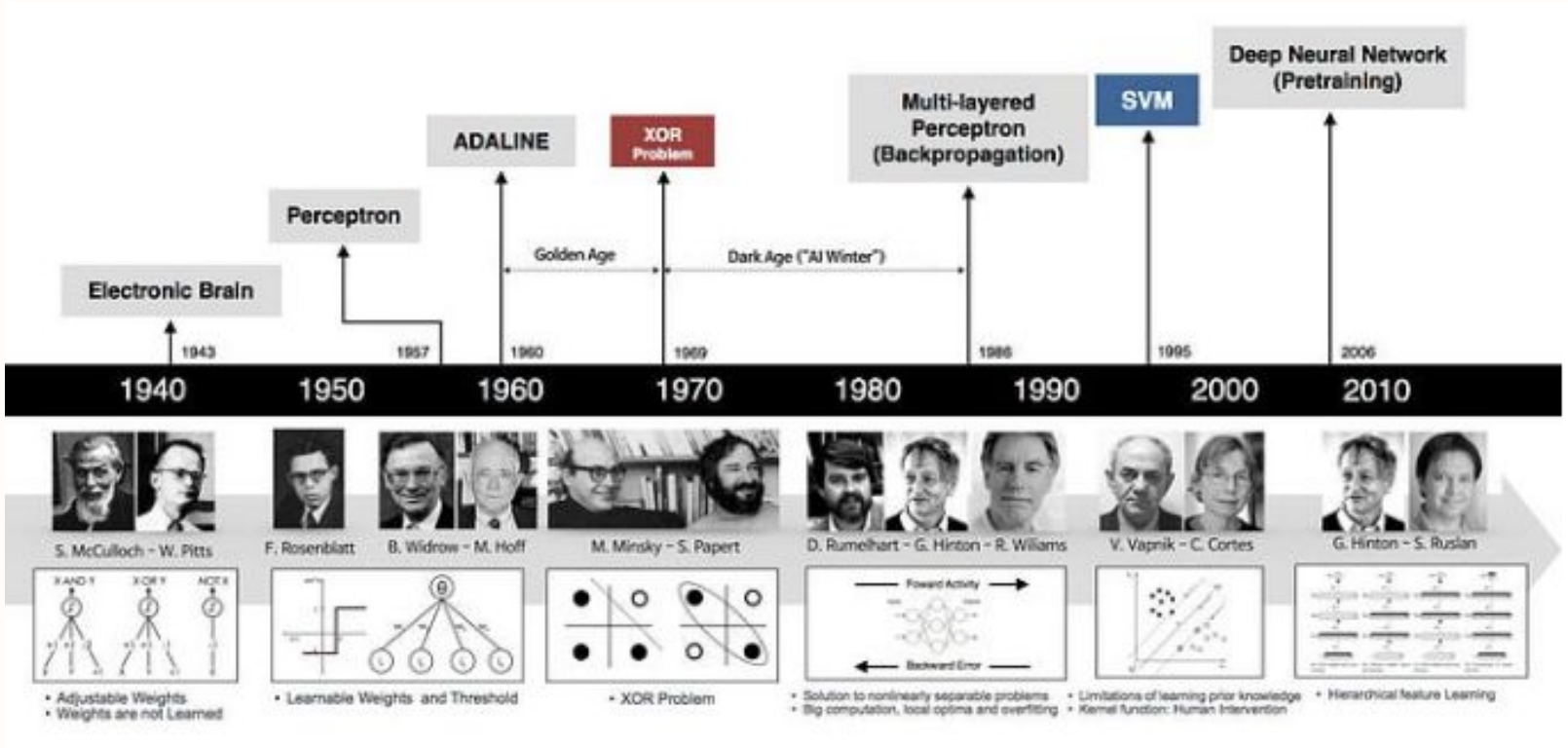
Fonte: yann.lecun.com

Aprendizagem profunda (Deep Learning)



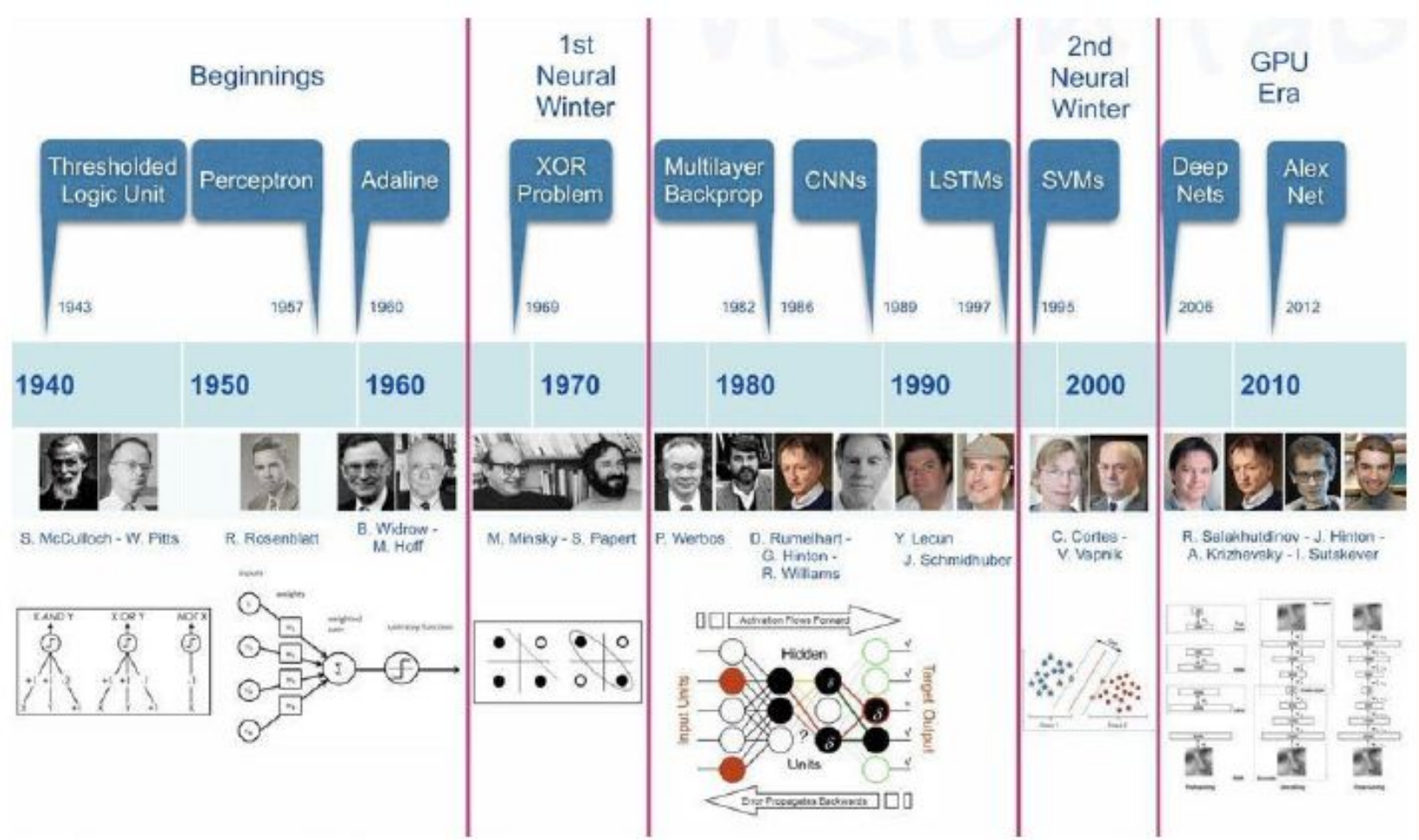
Fonte: yann.lecun.com

Aprendizado de máquina e profundo - Evolução redes neurais



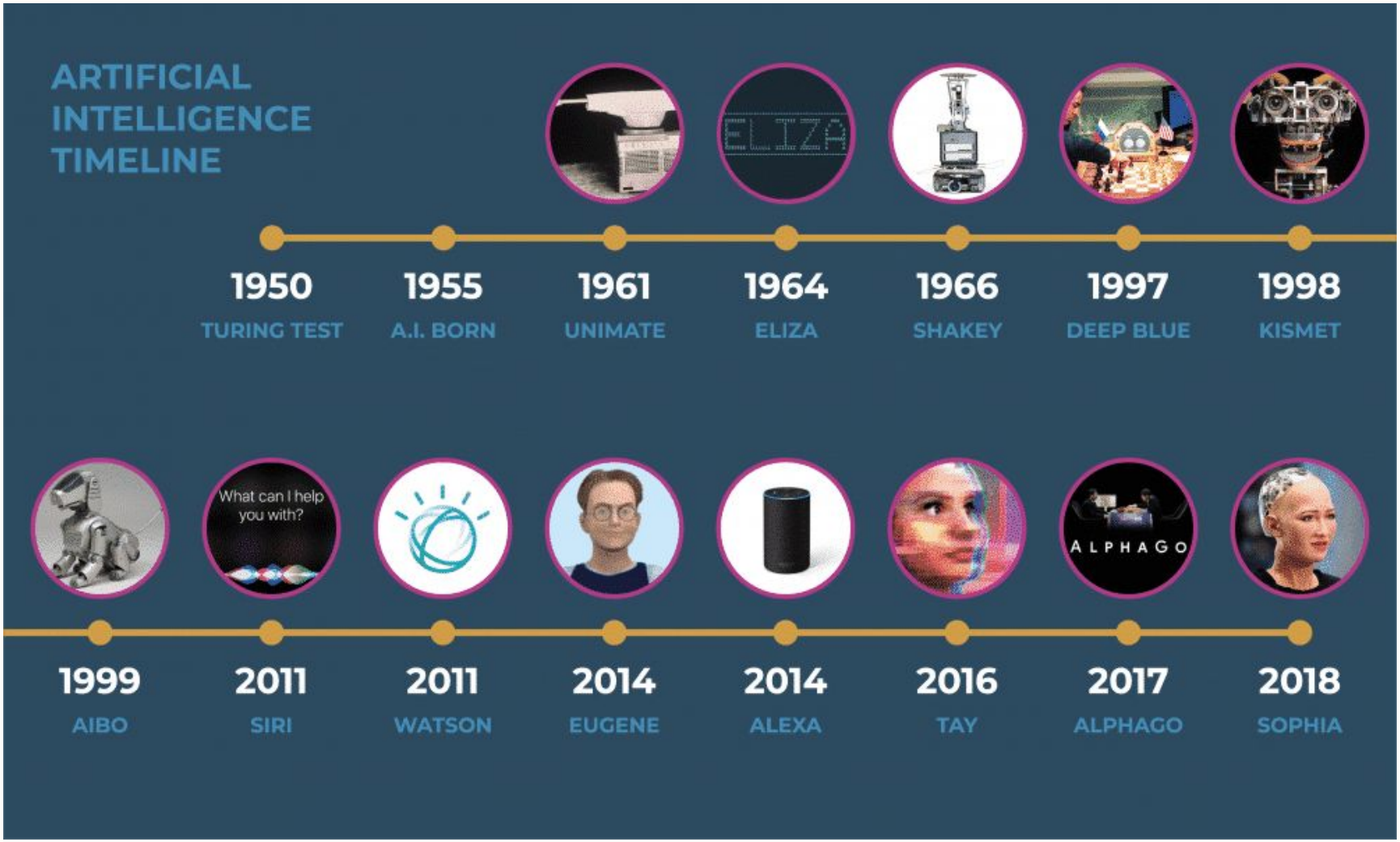
Fonte: https://medium.com/@augusto_Pinheiro/redes-neurais-artificiais-133de77c7240

Aprendizagem profunda (Deep Learning)



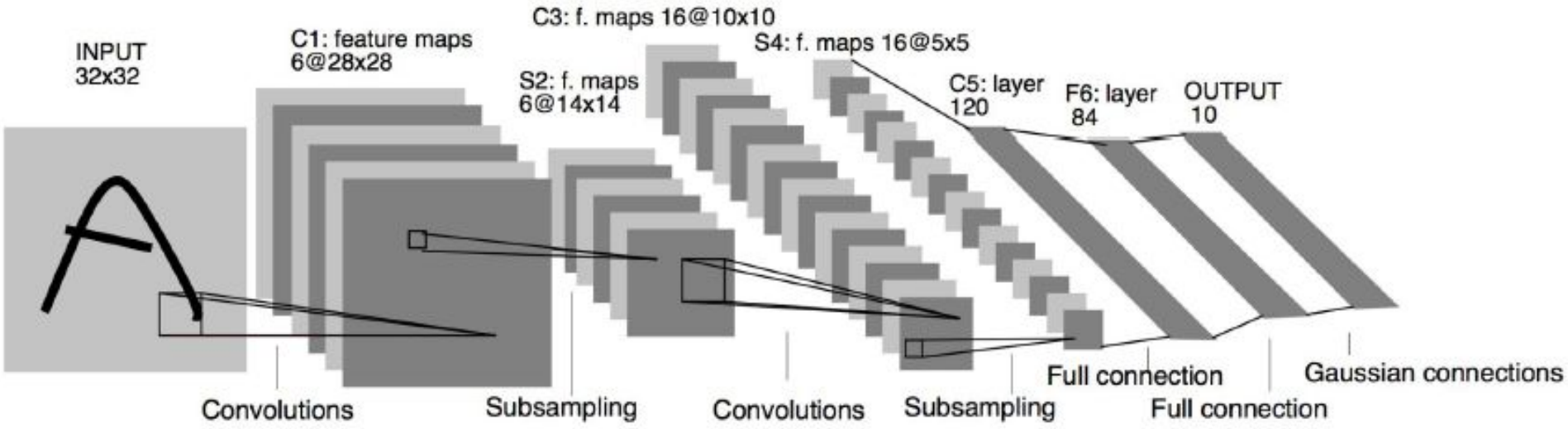
Fonte: https://twitter.com/jesparent/status/1173618397046804481

Aprendizado de máquina e profundo - Evolução da IA



Fonte: https://cafetalk.com/column/read/?id=279370&lang=it

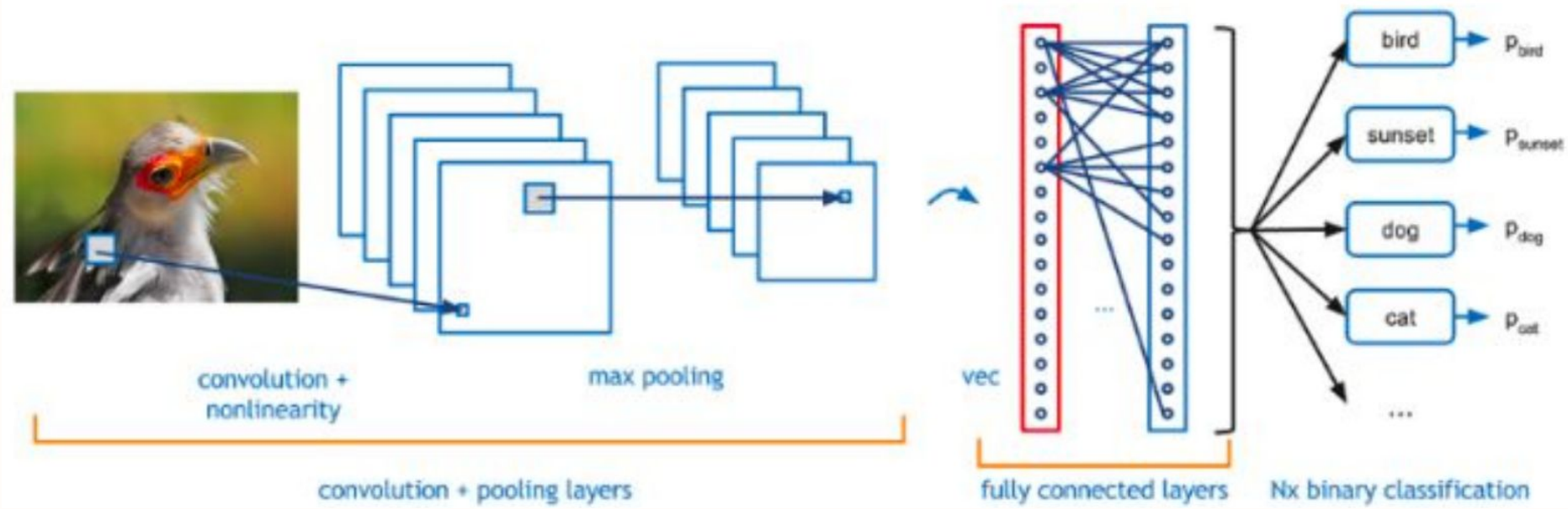
Aprendizagem profunda (Deep Learning)



Key Deep Learning Architecture: LeNet-5

LeCun et al., Gradient-based learning applied to document recognition (https://ieeexplore.ieee.org/document/726791)

Aprendizagem profunda (Deep Learning)



Aprendizagem profunda (Deep Learning)

- ImageNet (2010)
  - 22000 classes, 15 milhões de imagens

https://www.image-net.org/

Aprendizagem profunda (Deep Learning)

IMAGENET

14,197,122 images, 21841 synsets indexed

Home Download Challenges About

Not logged in. Login | Signup

ImageNet is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. The project has been instrumental in advancing computer vision and deep learning research. The data is available for free to researchers for non-commercial use.

Mar 11 2021. ImageNet website update.

FEI-FEI LI
(publishes under L.Fei-Fei)

Sequoia Professor of
Computer Science
Denning Co-Director
Stanford HAI

Aprendizagem profunda (Deep Learning)

➔ AlexNet (2012)

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky University of Toronto kriz@cs.utoronto.ca
Ilya Sutskever University of Toronto ilya@cs.utoronto.ca
Geoffrey E. Hinton University of Toronto hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

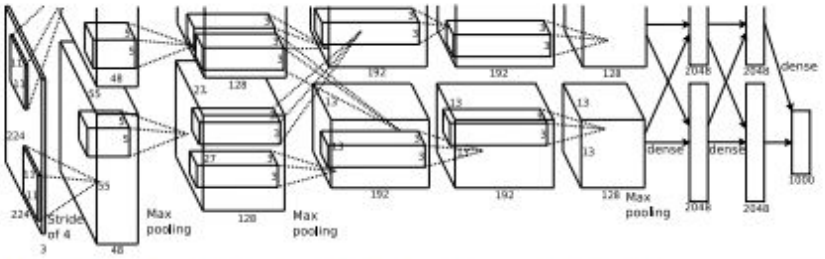
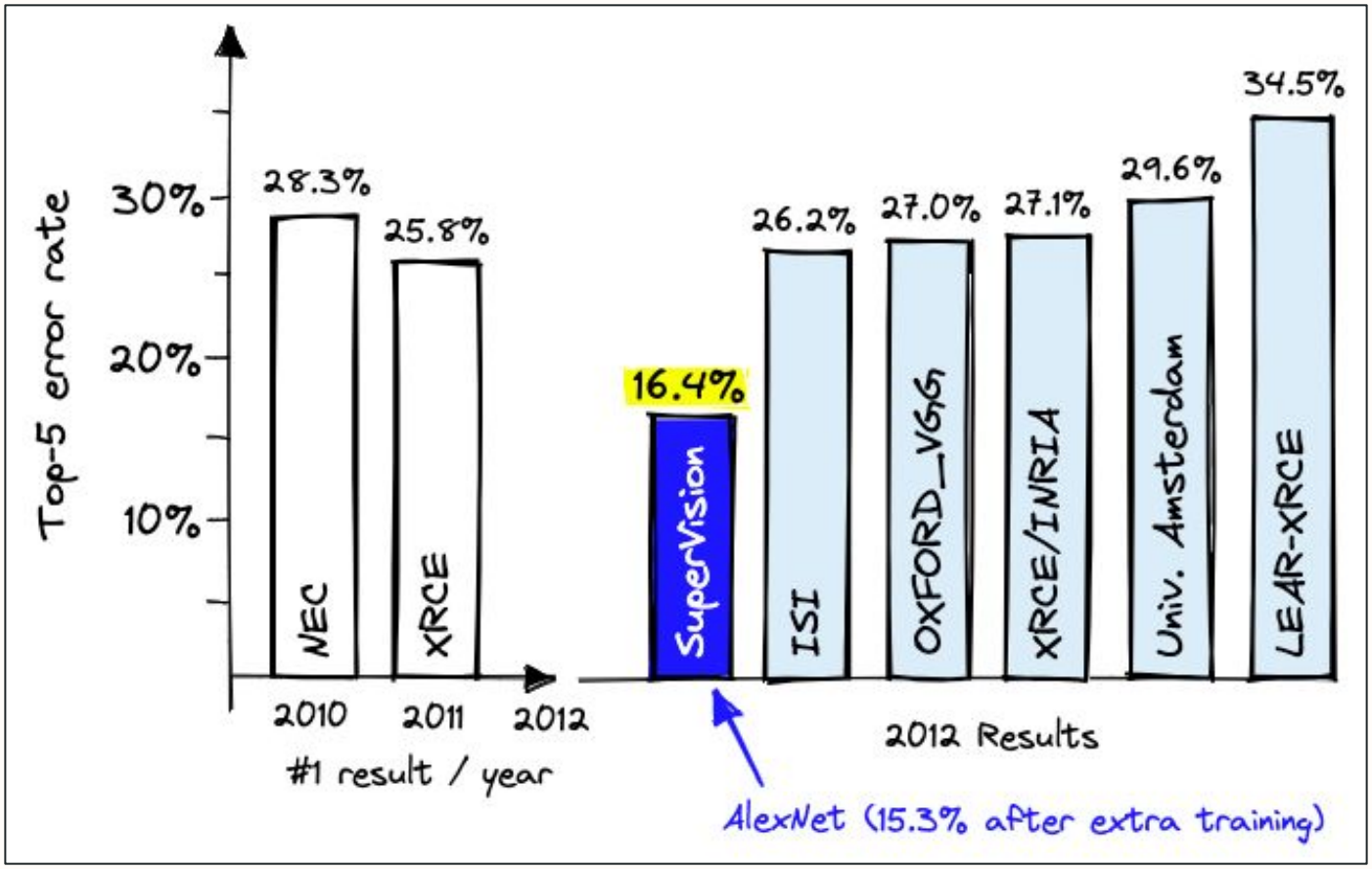


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Nvidia GTX 580 (https://www.nvidia.com/zh-tw/geforce/graphics-cards/geforce-gtx-580/product-images/)

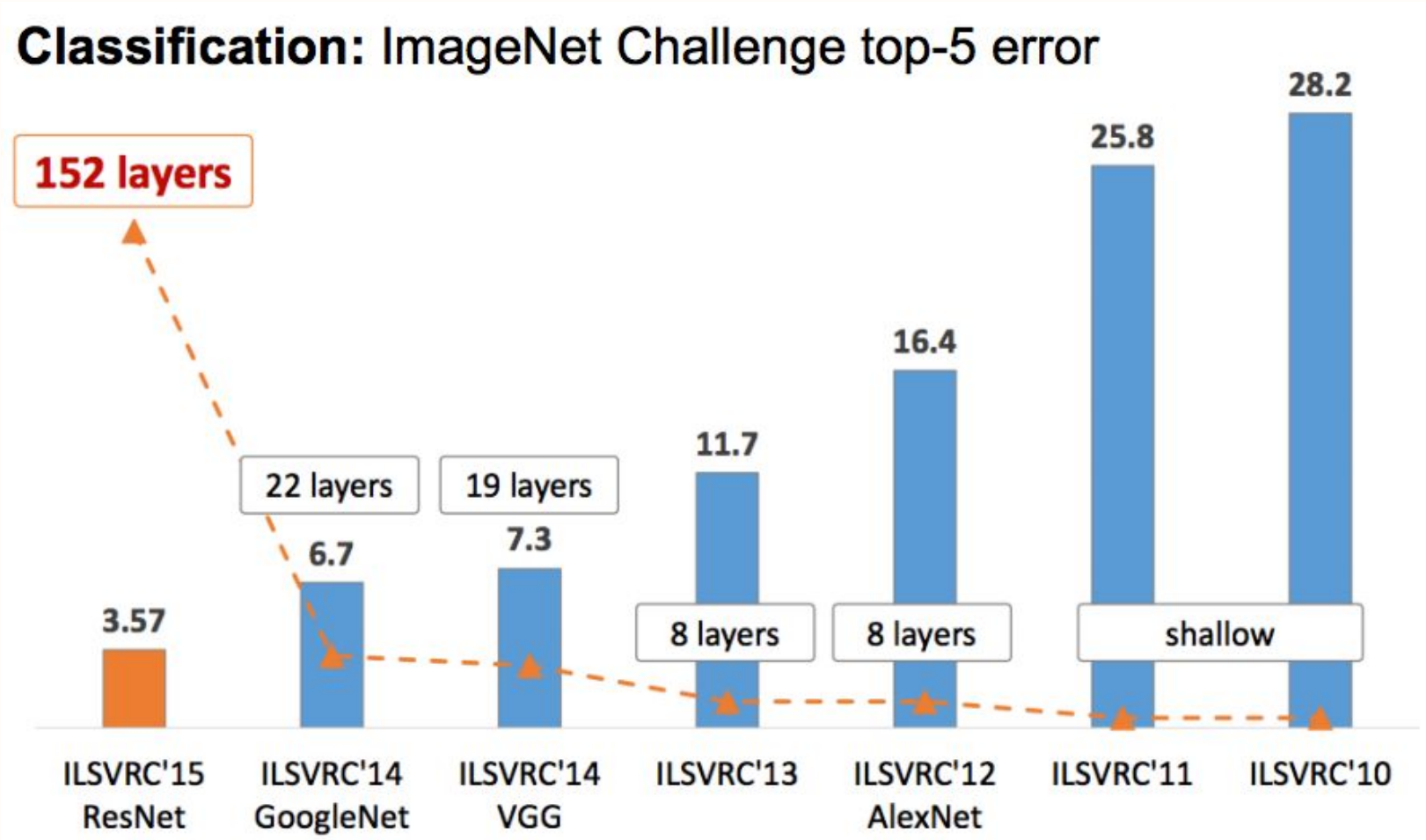
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012).

Aprendizagem profunda (Deep Learning)



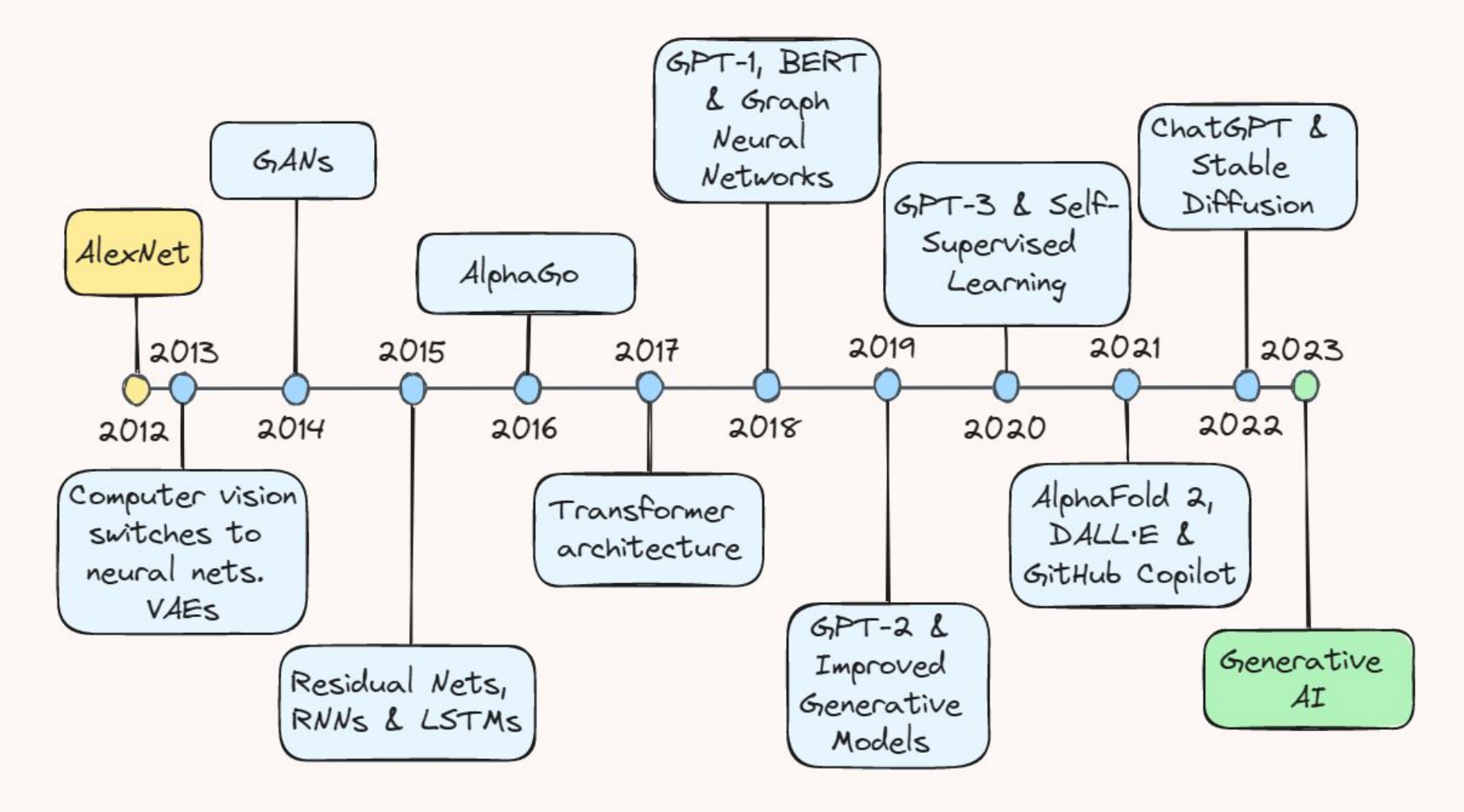
Fonte da imagem: https://www.pinecone.io/learn/series/image-search/imagenet/

Aprendizagem profunda (Deep Learning)



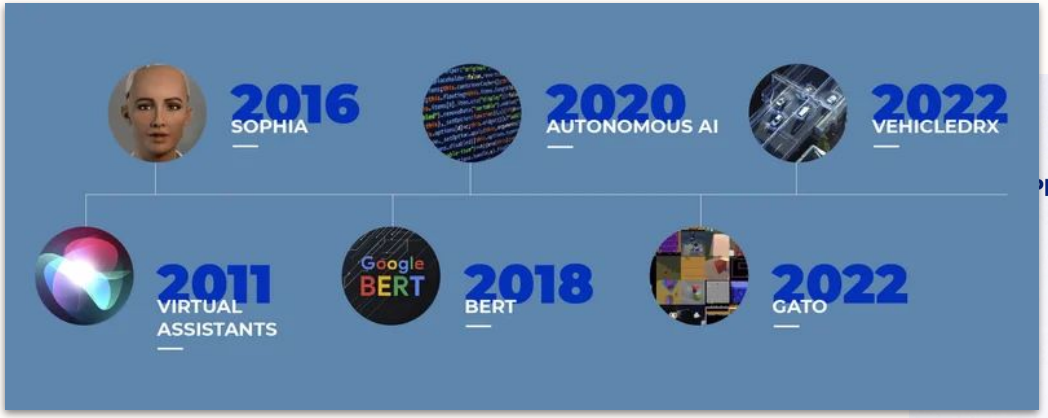
Fonte: https://medium.com/@RaghavPrabhu/cnn-architectures-lenet-alexnet-vgg-googlenet-and-resnet-7c81c017b848

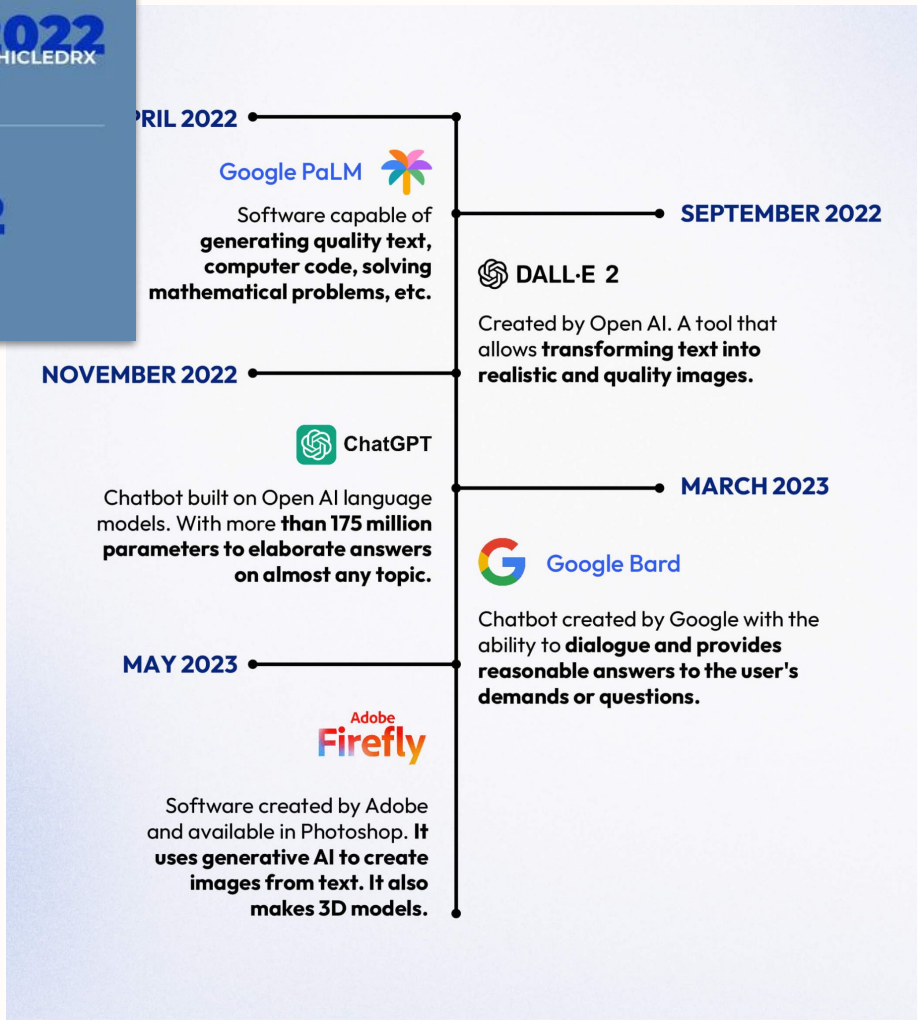
Aprendizagem profunda (Deep Learning)



Fonte: https://towardsdatascience.com/ten-years-of-ai-in-review-85decdb2a540

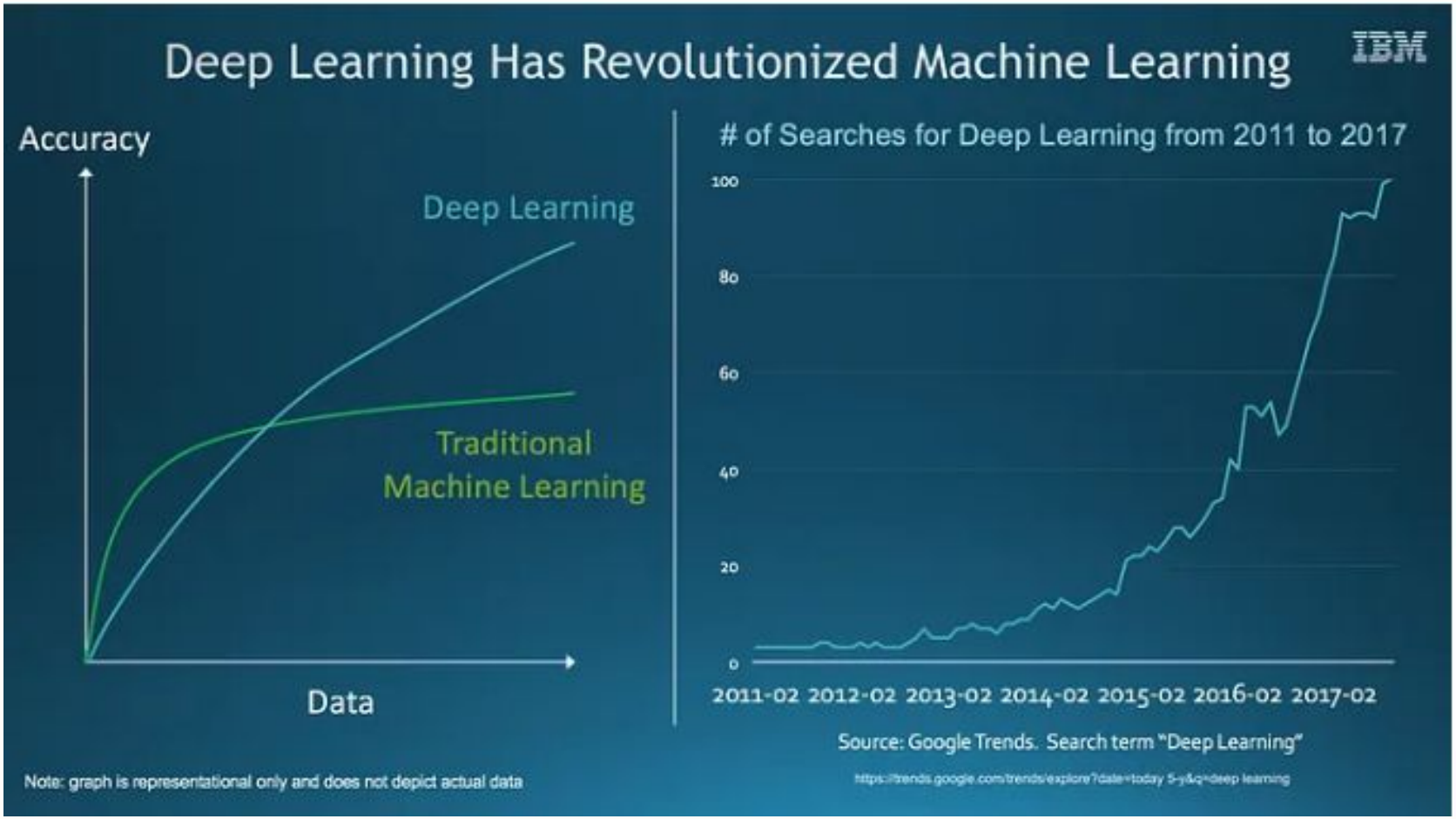
Aprendizado de máquina e profundo - Evolução da IA (2010 - 2023)





Fonte: https://www.algotive.ai/blog/the-history-of-artificial-intelligence-machine-learning-and-deep-learning

Aprendizado de máquina e profundo

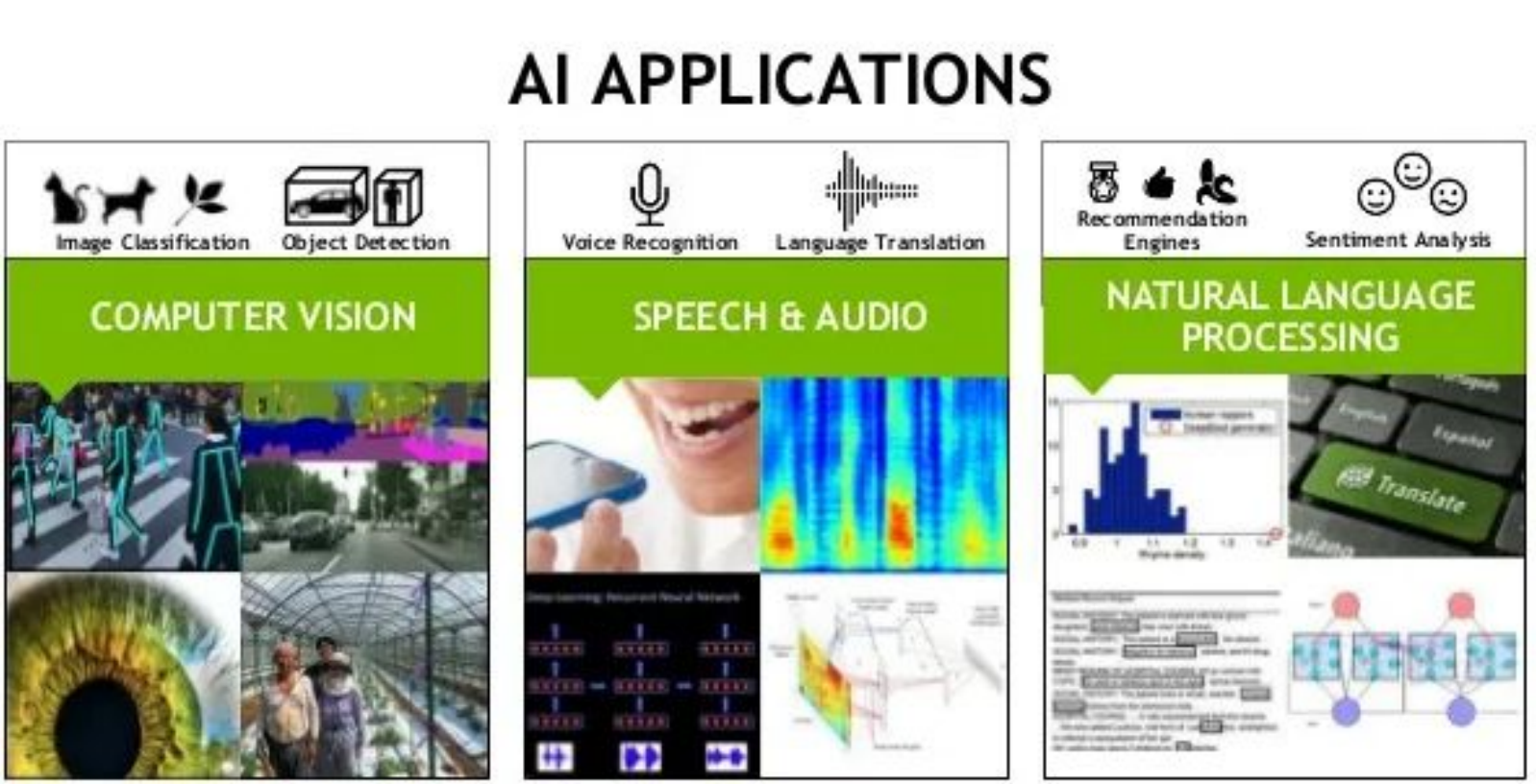


Fonte: https://sanchittanwar75.medium.com/introduction-to-machine-learning-and-deep-learning-bd25b792e488

Aprendizagem profunda (Deep Learning) - Aplicações

➔ Diferentes tarefas necessitam de diferentes abordagens.

- Imagens
- Sons
- Texto



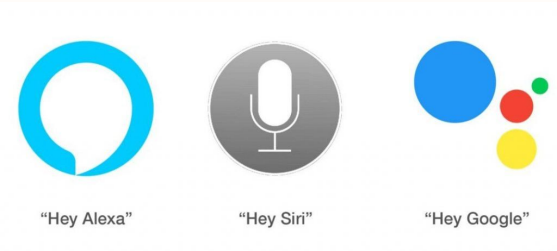
Fonte: https://sanchittanwar75.medium.com/introduction-to-machine-learning-and-deep-learning-bd25b792e488

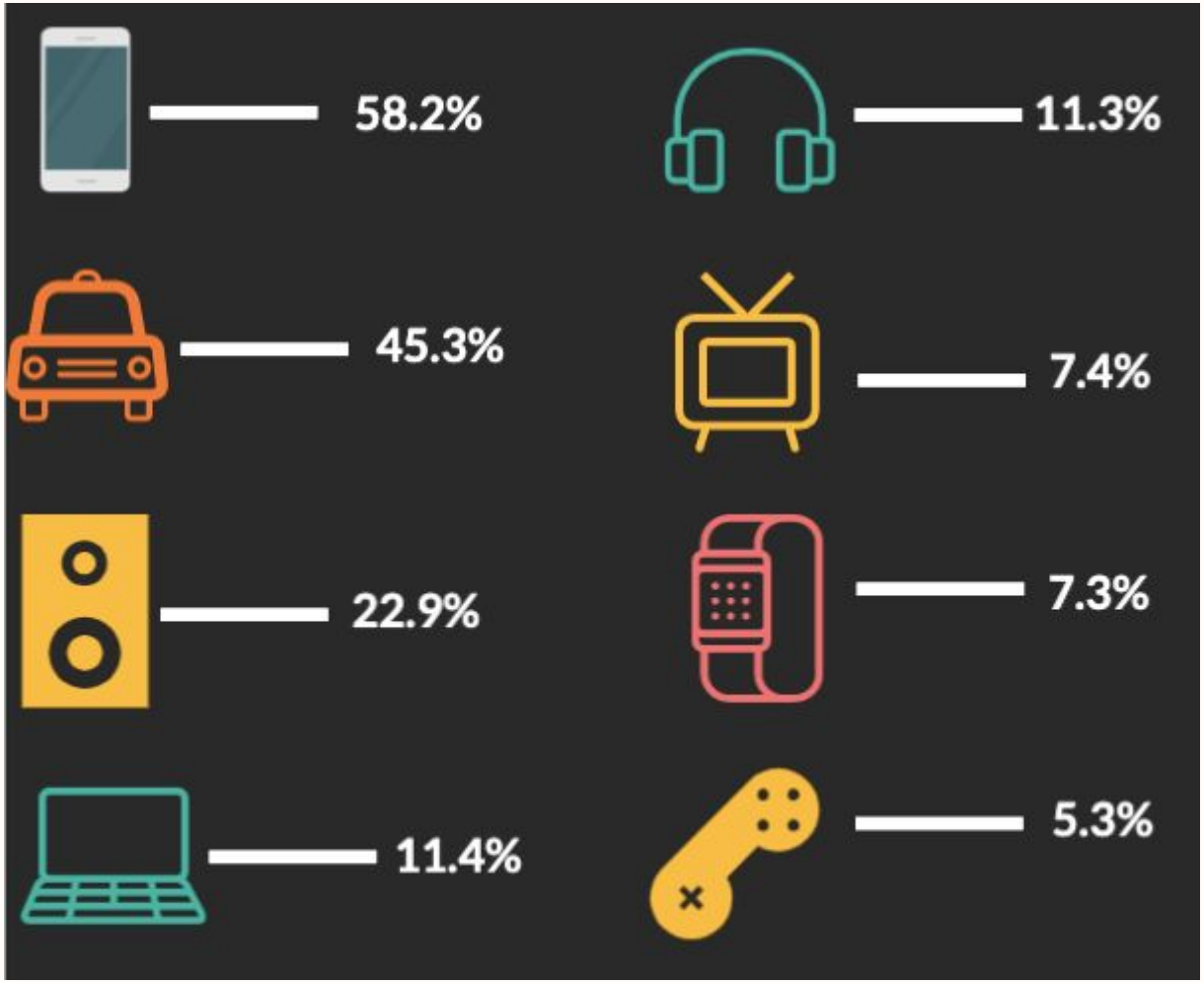
Aprendizagem profunda (Deep Learning) - Aplicações

➔ Uso doméstico com assistentes de voz

Fonte das imagens: https://cyfuture.com/blog/domineering-power-of-voice-technology-the-dawn-of-voice-assistants-is-here/

Aprendizagem profunda (Deep Learning) - Aplicações

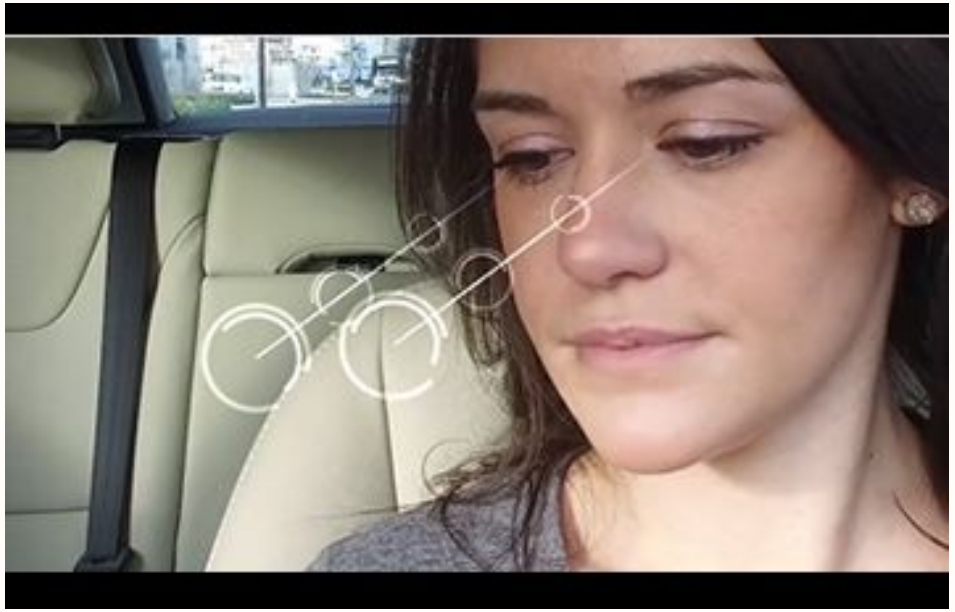




Fonte das imagens: https://cyfuture.com/blog/domineering-power-of-voice-technology-the-dawn-of-voice-assistants-is-here/

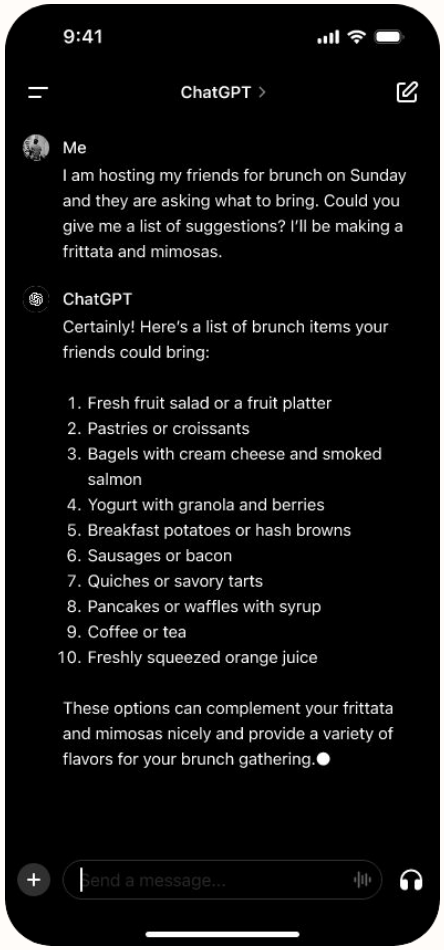
Aprendizagem profunda (Deep Learning) - Aplicações

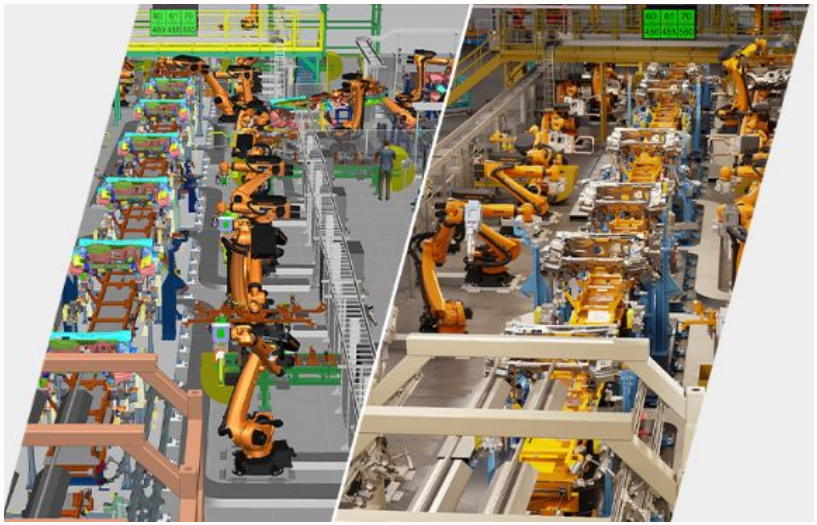
➔ Aplicações em segurança e indústria



Fonte: https://developer.nvidia.com/blog/share-your-science-the-future-of-facial-recognition-for-video-surveillance/
https://developer.nvidia.com/blog/ai-co-pilot-rnn-dynamic-facial-analysis/
https://developer.nvidia.com/blog/nvidia-drive-10-0-now-available/

Aprendizagem profunda (Deep Learning) - Aplicações

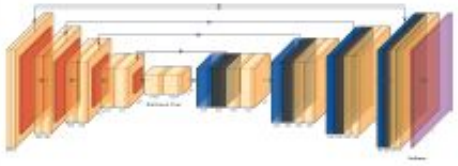




Fonte das imagens: https://thedirect.com/article/star-wars-luke-skywalker-deepfakes-cgi
https://openai.com/chatgpt
https://www.nvidia.com/content/dam/en-zz/Solutions/omniverse/siemens/omniverse-enterprise-siemens-partnership-hero-bb460_420-d.jpg

Aprendizagem profunda (Deep Learning) - Aplicações em Fotografia

➔ Learning to see in the dark



Learning to See in the Dark. Chen Chen; Qifeng Chen; Jia Xu; Vladlen Koltun. IEEE Conference on Computer Vision and Pattern Recognition, 2018

Aprendizagem profunda (Deep Learning) - Aplicações em Fotografia

➔ Multi-Image Deblurring

Burst Image Deblurring Using Permutation Invariant Convolutional Neural Networks. Miika Aittala; Fredo Durand. European Conference on Computer Vision (ECCV), 2018

Aprendizagem profunda (Deep Learning) - Aplicações em Fotografia

➔ Restauração e Colorização

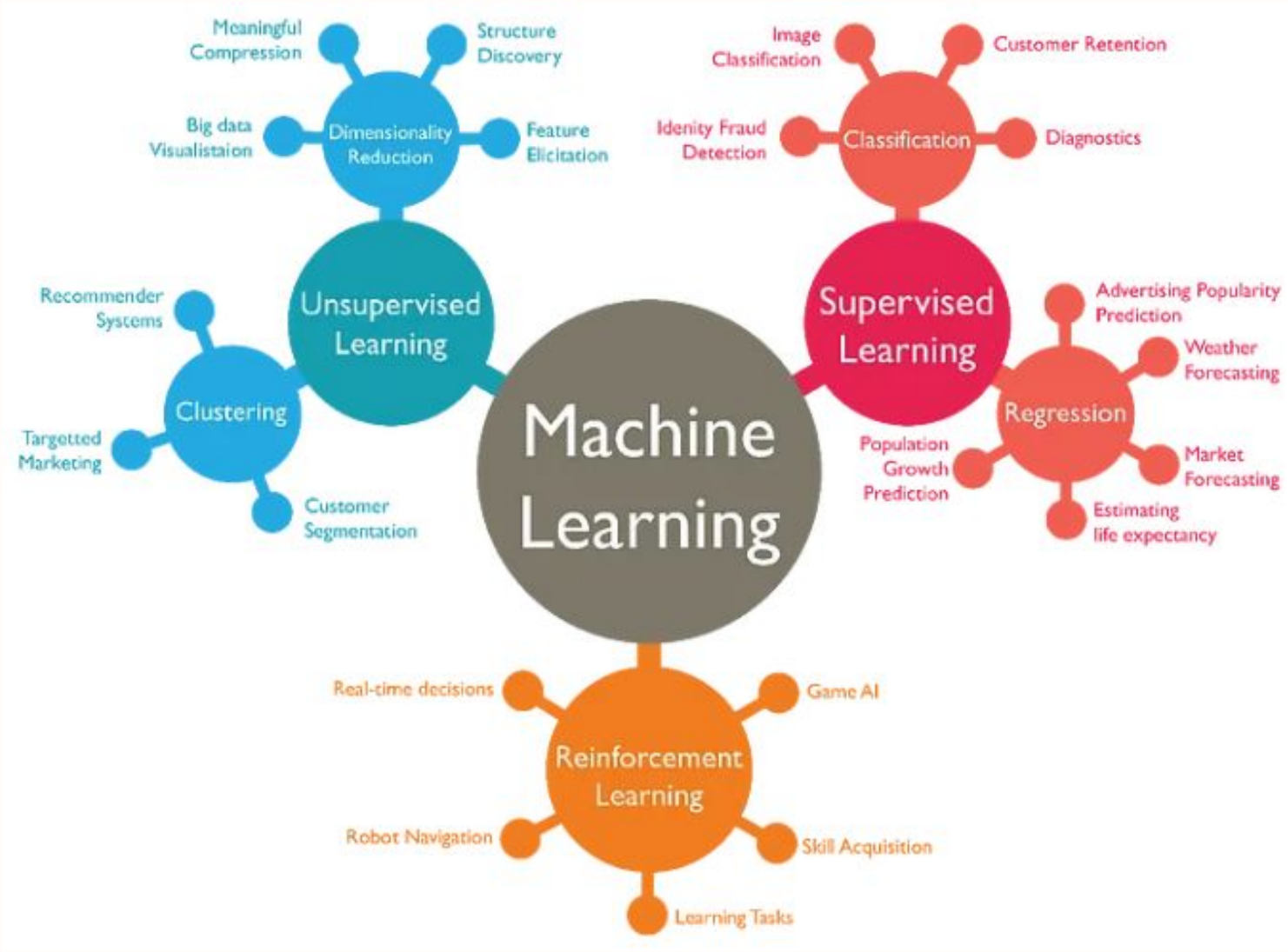
DeepRemaster: Temporal Source-Reference Attention Networks for Comprehensive Video Enhancement. SATOSHI IIZUKA; EDGAR SIMO-SERRA. SIGGRAPH, 2019

Aprendizagem profunda (Deep Learning) - Aplicações em Fotografia

➔ Inpainting

Image Inpainting for Irregular Holes Using Partial Convolutions. Guilin Liu; Fitsum A. Reda; Kevin J. Shih; Ting-Chun Wang; Andrew Tao; Bryan Catanzaro. European Conference on Computer Vision (ECCV), 2018

Aprendizado de máquina e profundo - Tipos de Aprendizagem

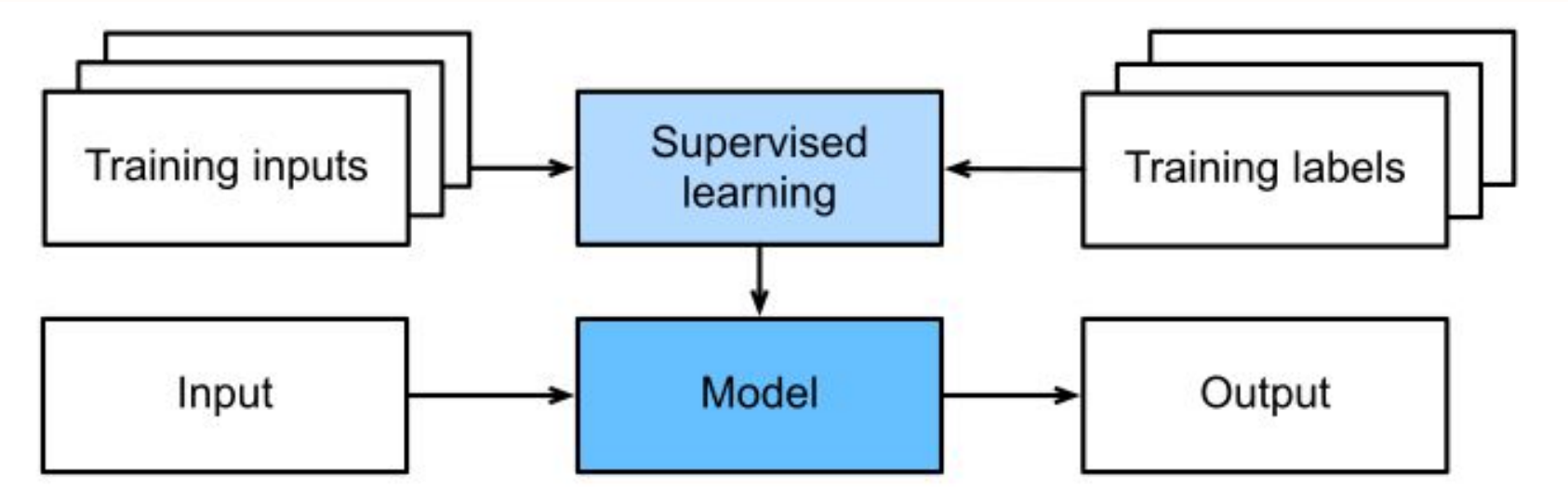


Fonte da imagem: https://sanchittanwar75.medium.com/introduction-to-machine-learning-and-deep-learning-bd25b792e488

Aprendizado de máquina e profundo - Tipos de Aprendizagem

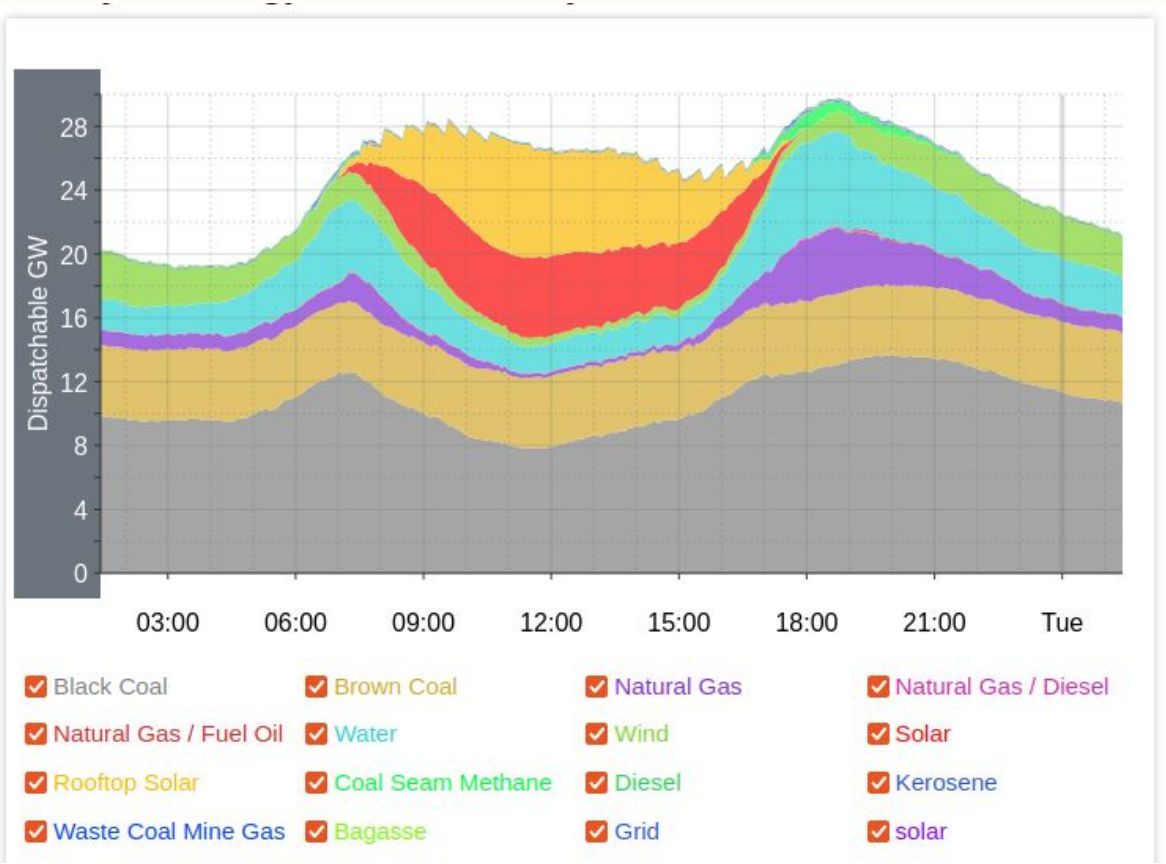
- Aprendizado Supervisionado:
  - Regressão
  - Classificação
  - Tagging (Multiclasses)
  - Busca
  - Sistema de Recomendação
  - Aprendizagem de Sequência
- Aprendizado Não-Supervisionado e Auto-Supervisionado:
  - Clustering
- Aprendizado por reforço

Aprendizado de máquina e profundo - Tipos de Aprendizagem (Supervisionado)



Aprendizado de máquina e profundo - Tipos de Aprendizagem (Supervisionado)

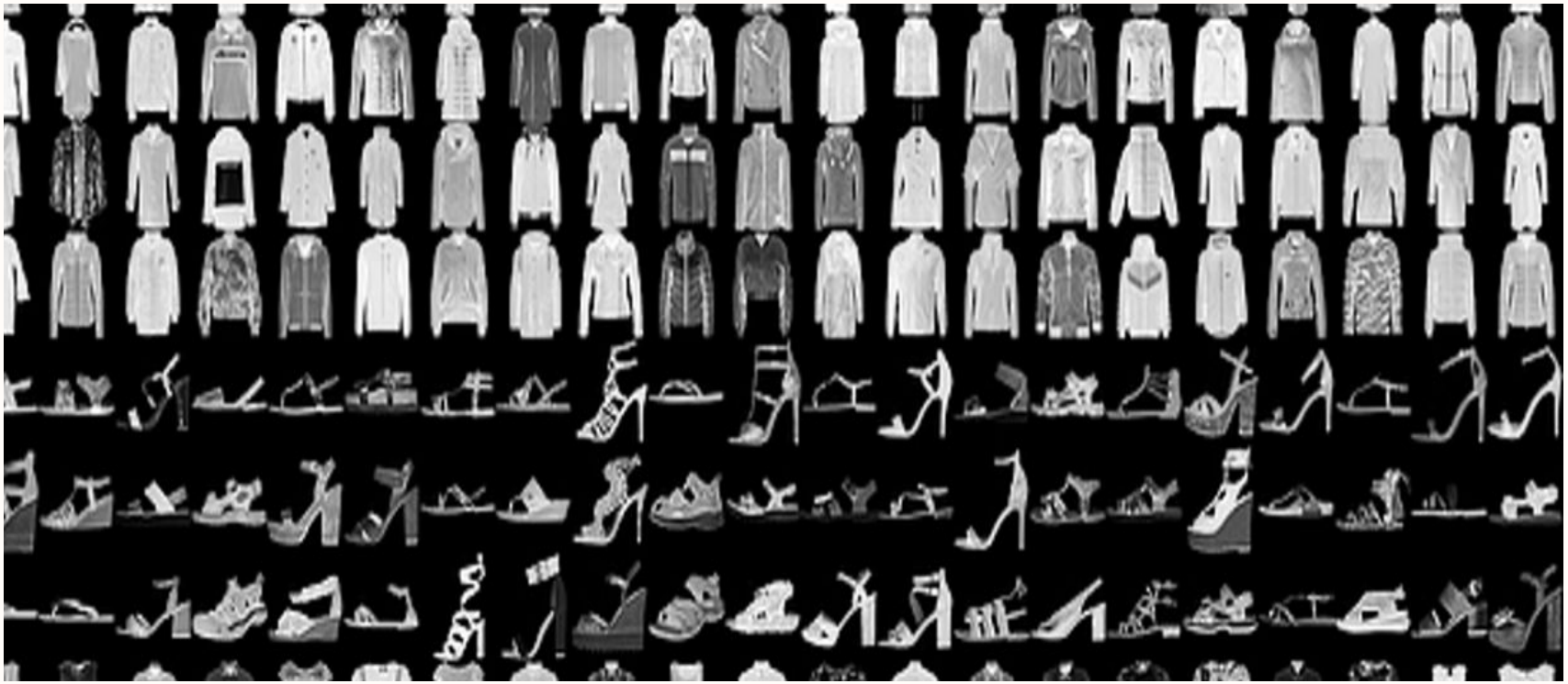
➔ Regressão



https://anero.id/energy

Aprendizado de máquina e profundo - Tipos de Aprendizagem (Supervisionado)

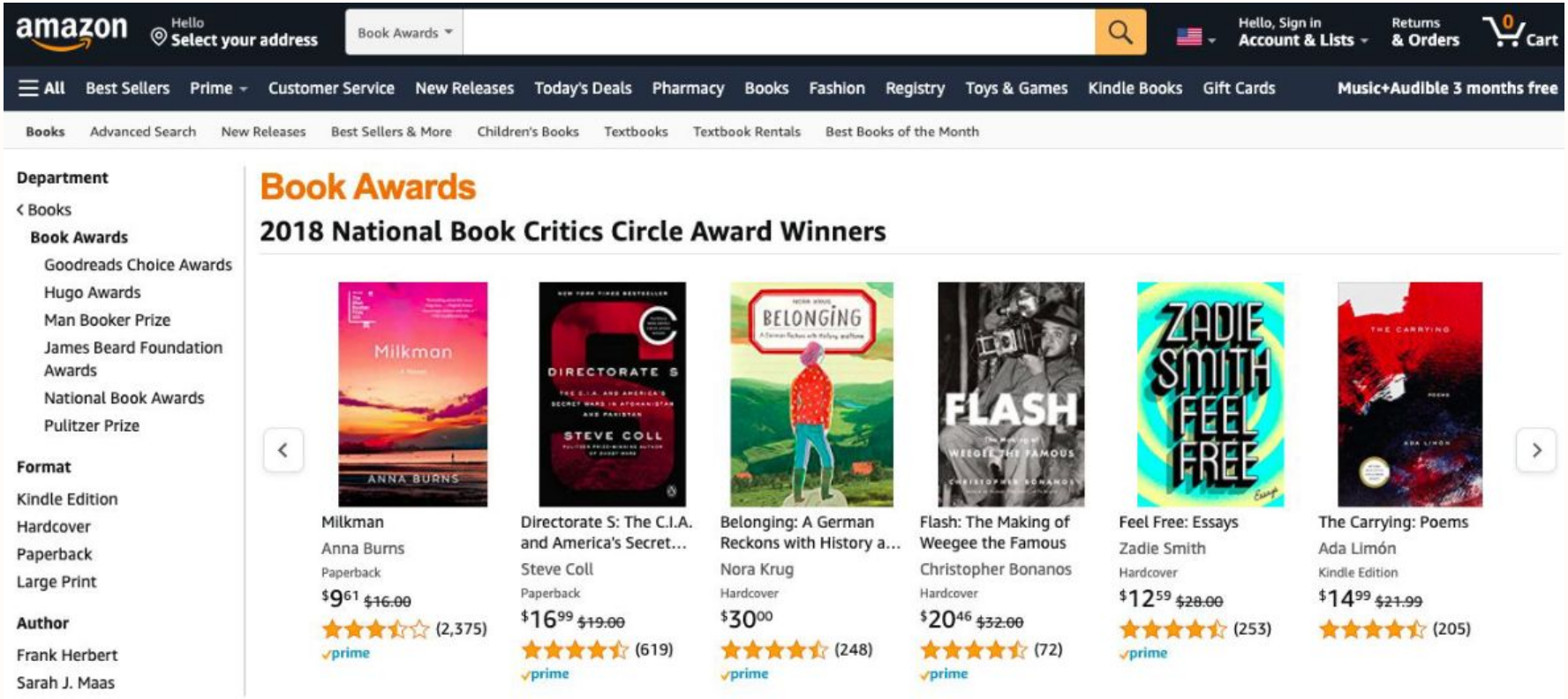
- ➔ Classificação
- ➔ Tagging



Fashion MNIST

Aprendizado de máquina e profundo - Tipos de Aprendizagem (Supervisionado)

- ➔ Busca
- ➔ Sistema de Recomendação



Fonte: www.amazon.com

Aprendizado de máquina e profundo - Tipos de Aprendizagem (Supervisionado)

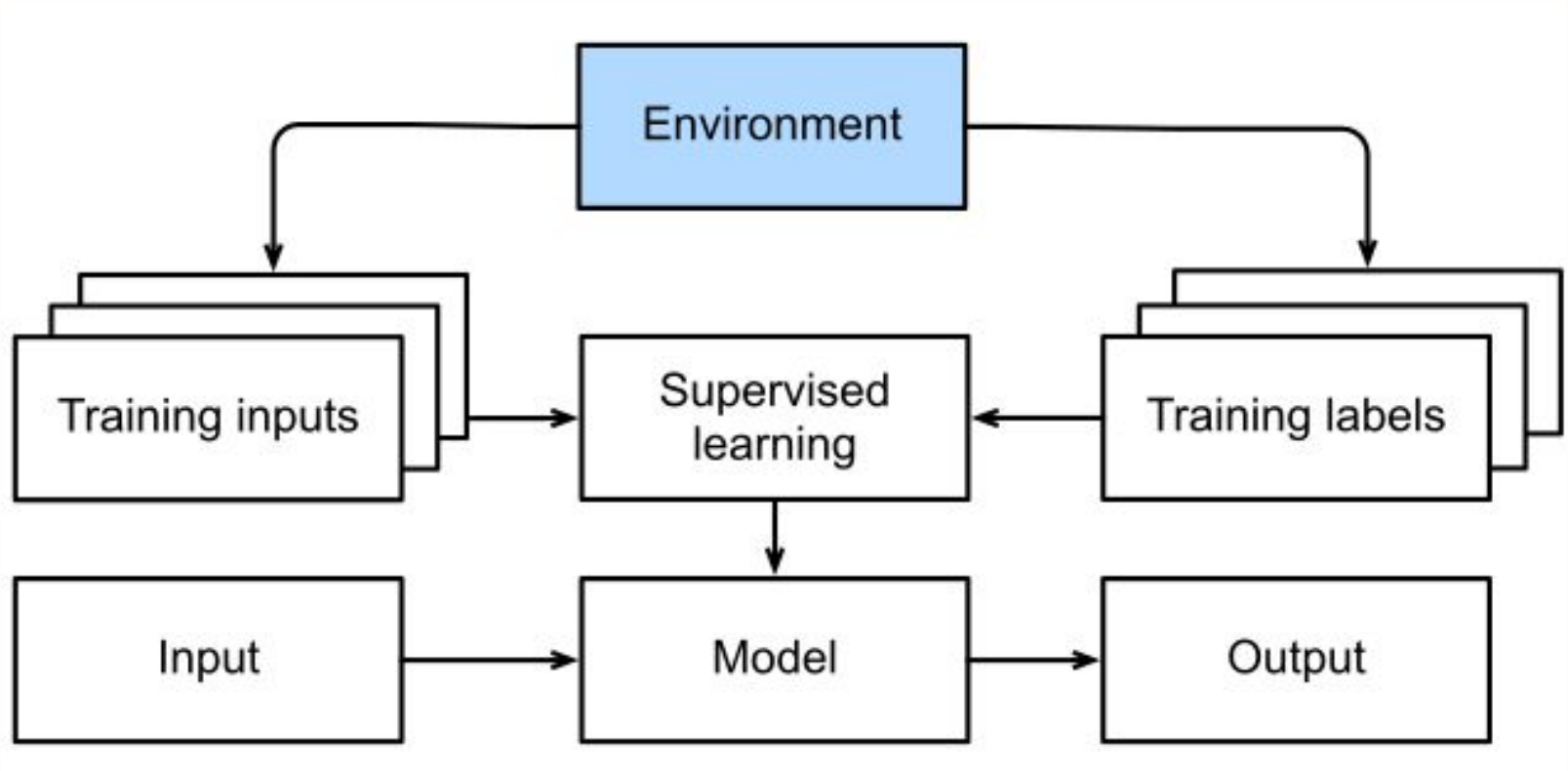
➔ Aprendizagem de Sequência

```
Tom has dinner in Washington with Sally
Ent  -    -    -     Ent      -    Ent
```

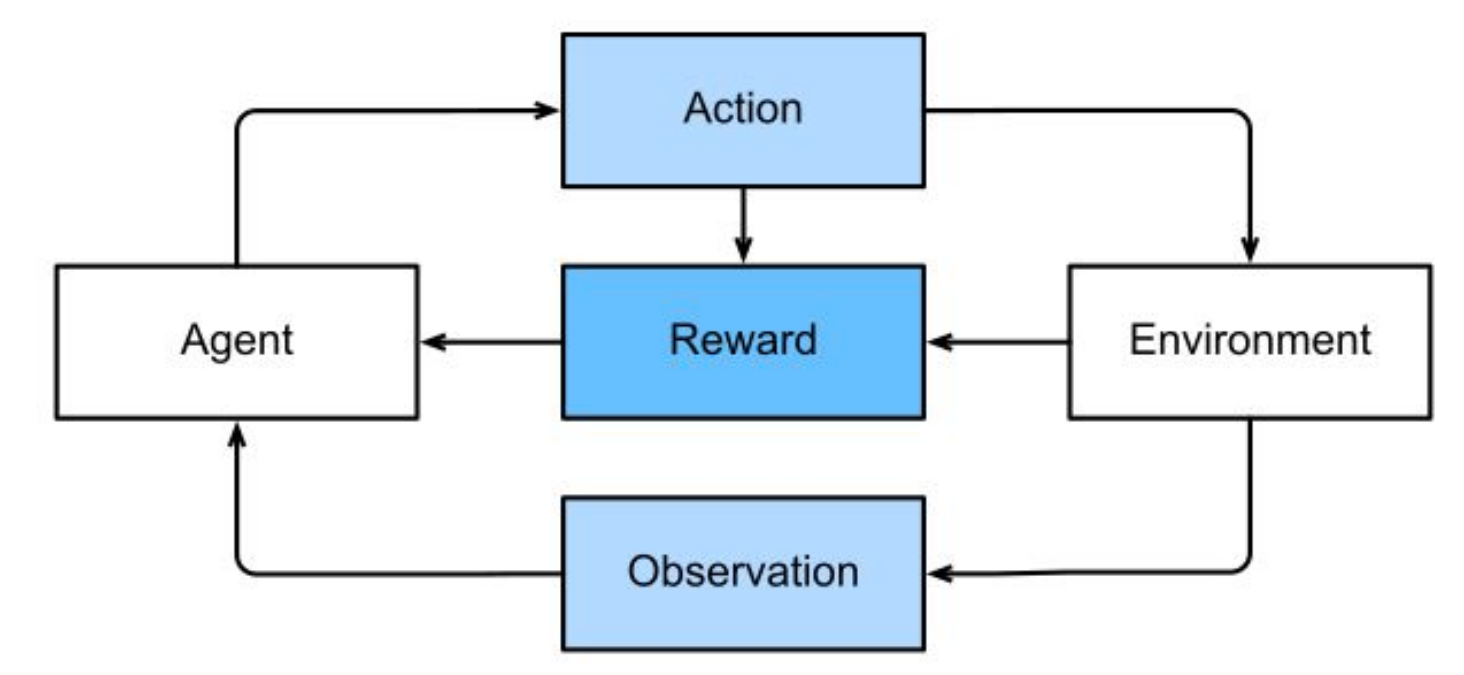
-D-e-e-p- L-ea-r-ni-ng- in an audio recording.

```
German:           Haben Sie sich schon dieses grossartige Lehrwerk angeschaut?
English:          Did you already check out this excellent tutorial?
Wrong alignment:  Did you yourself already this excellent tutorial looked-at?
```

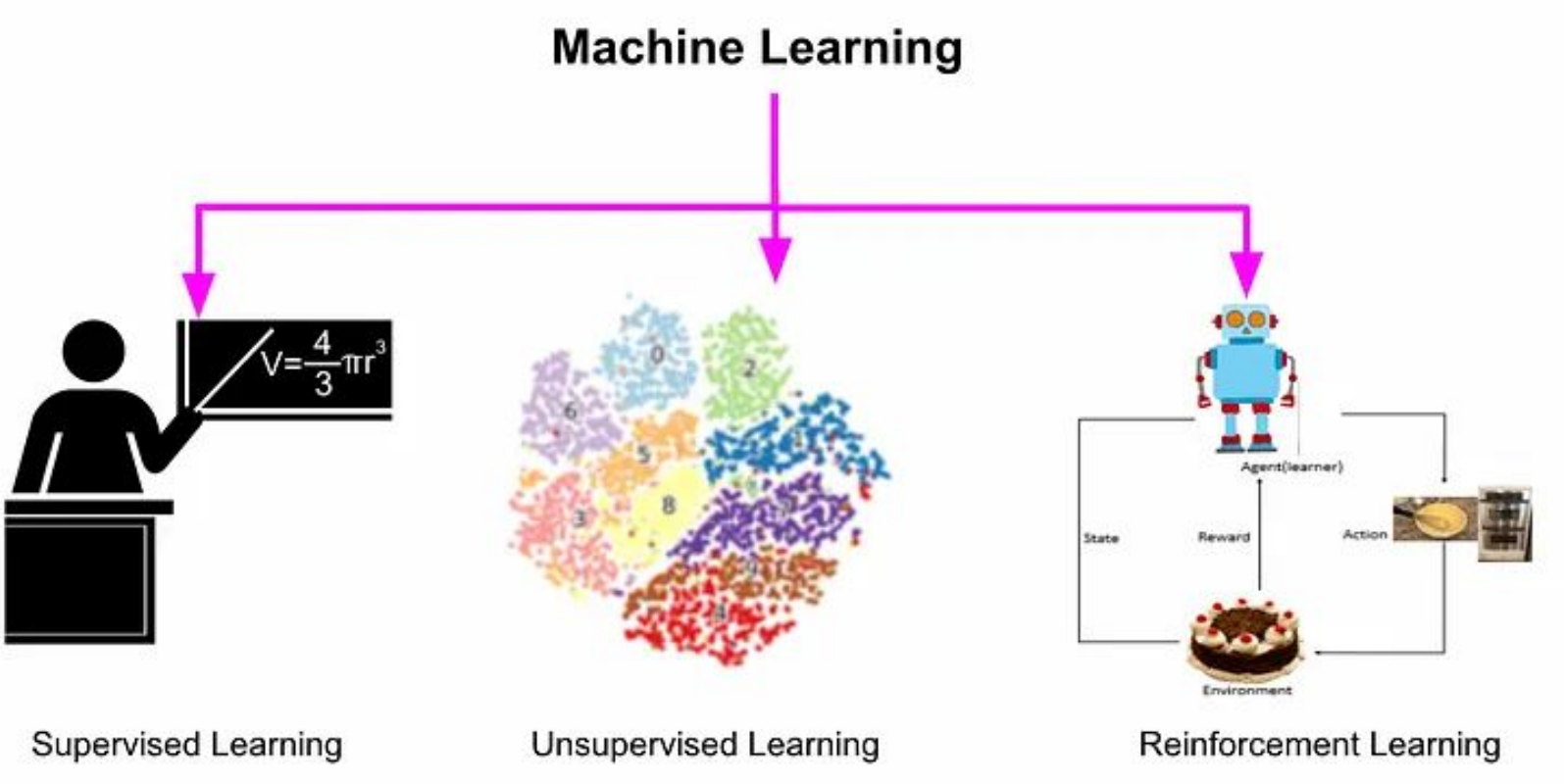
Aprendizado de máquina e profundo - Tipos de Aprendizagem (Não Supervisionado)



Aprendizado de máquina e profundo - Tipos de Aprendizagem (Aprendizado por reforço)



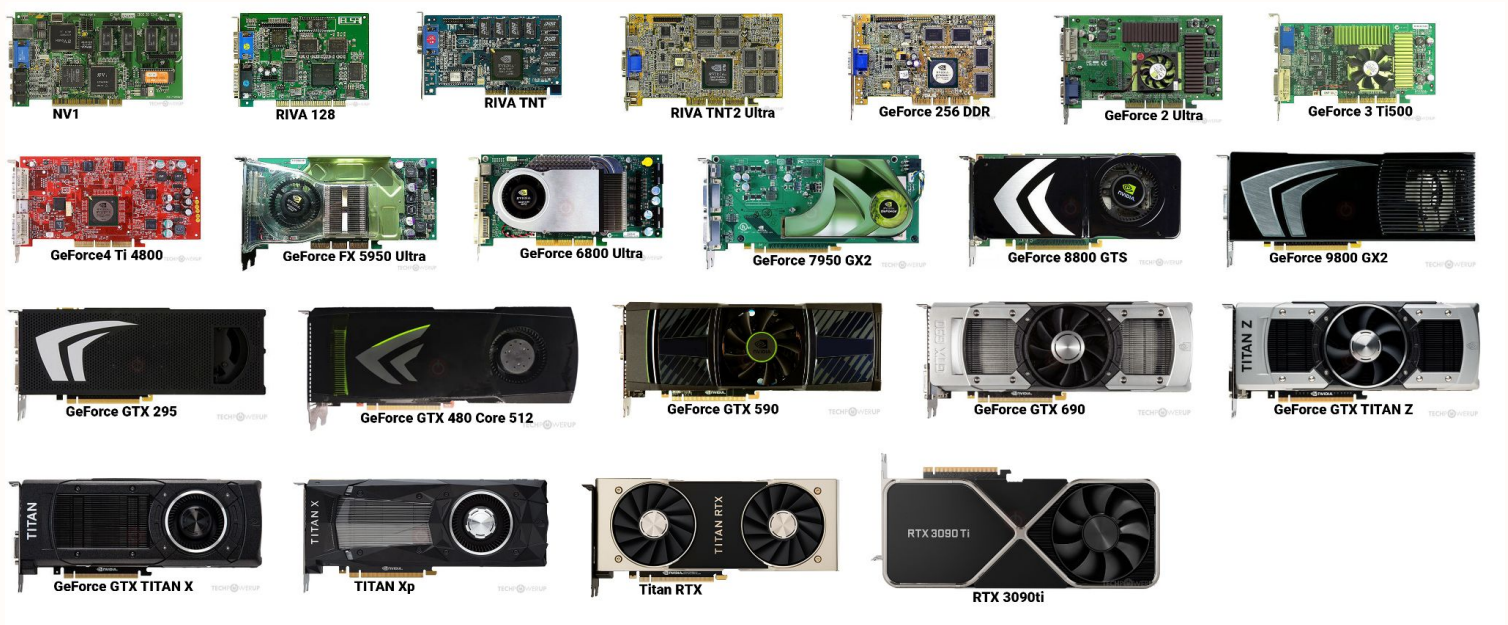
Aprendizado de máquina e profundo - Tipos de Aprendizagem



Fonte: https://arshren.medium.com/supervised-unsupervised-and-reinforcement-learning-245b59709f68

Aprendizado de máquina e profundo - Evolução dos hardwares

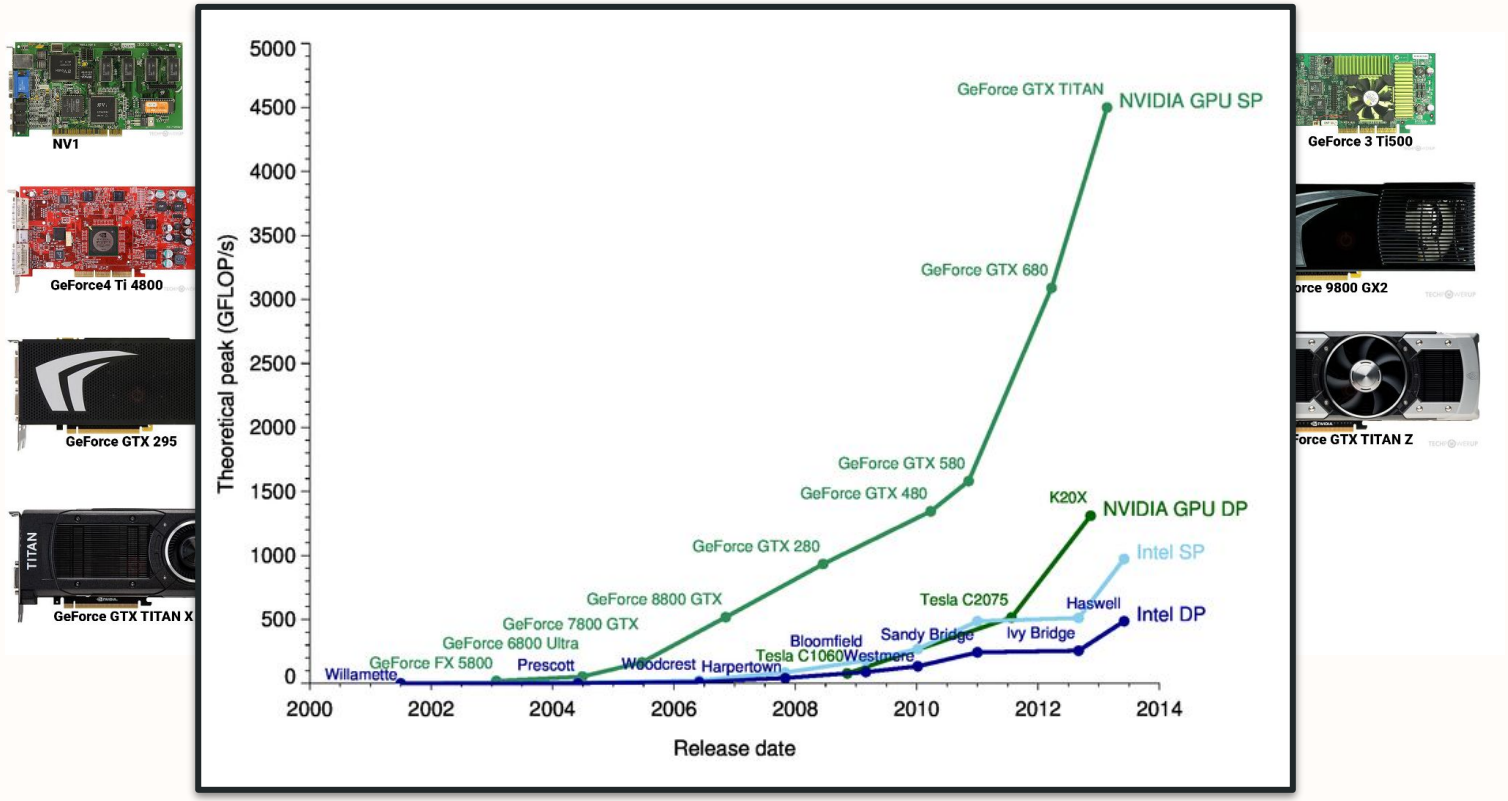
➔ Novos modelos de deep learning também possibilitaram a rápida a evolução de hardwares especializados



Fonte das imagens: https://www.reddit.com/r/pcmasterrace/comments/ufv6oc/nvidia_gpus_over_time_5k_comparison/

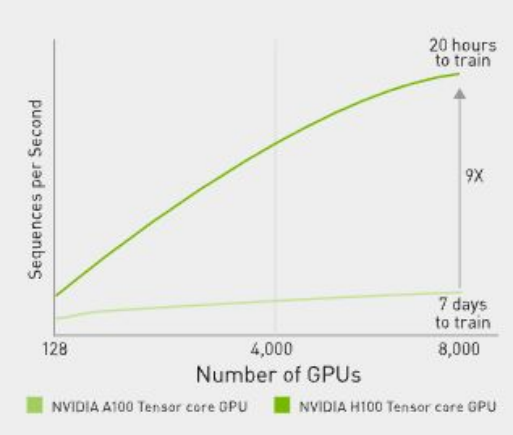
Aprendizado de máquina e profundo - Evolução dos hardwares

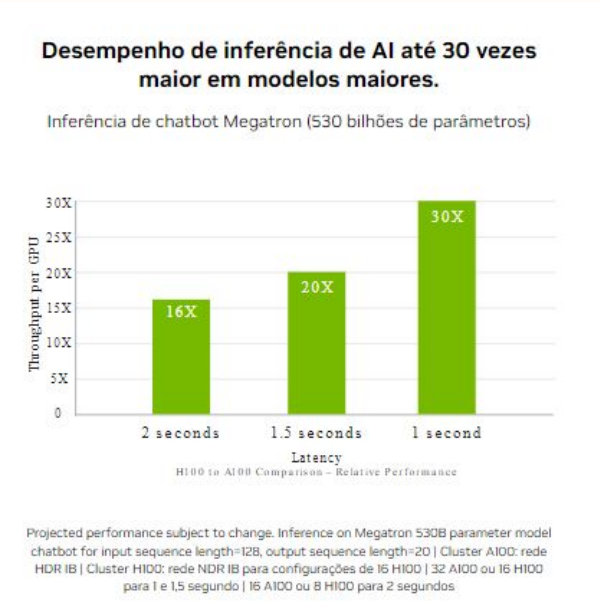
➔ Novos modelos de deep learning também possibilitaram a rápida a evolução de hardwares especializados



Fonte das imagens: https://www.reddit.com/r/pcmasterrace/comments/ufv6oc/nvidia_gpus_over_time_5k_comparison/
https://hemprasad.wordpress.com/2013/07/18/cpu-vs-gpu-performance/

Aprendizado de máquina e profundo - Evolução dos hardwares

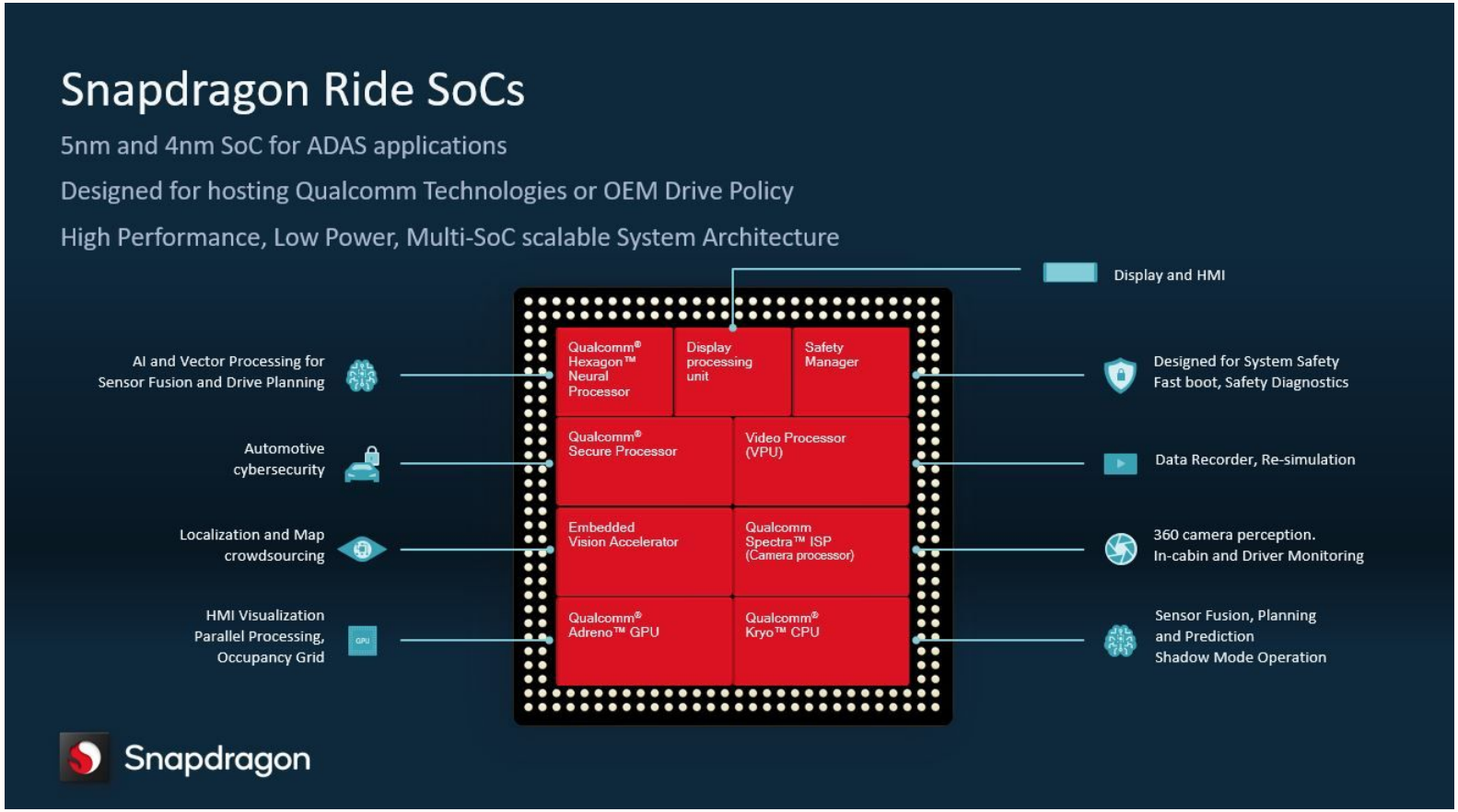




https://www.nvidia.com/pt-br/data-center/h100/

Aprendizado de máquina e profundo - Evolução dos hardwares

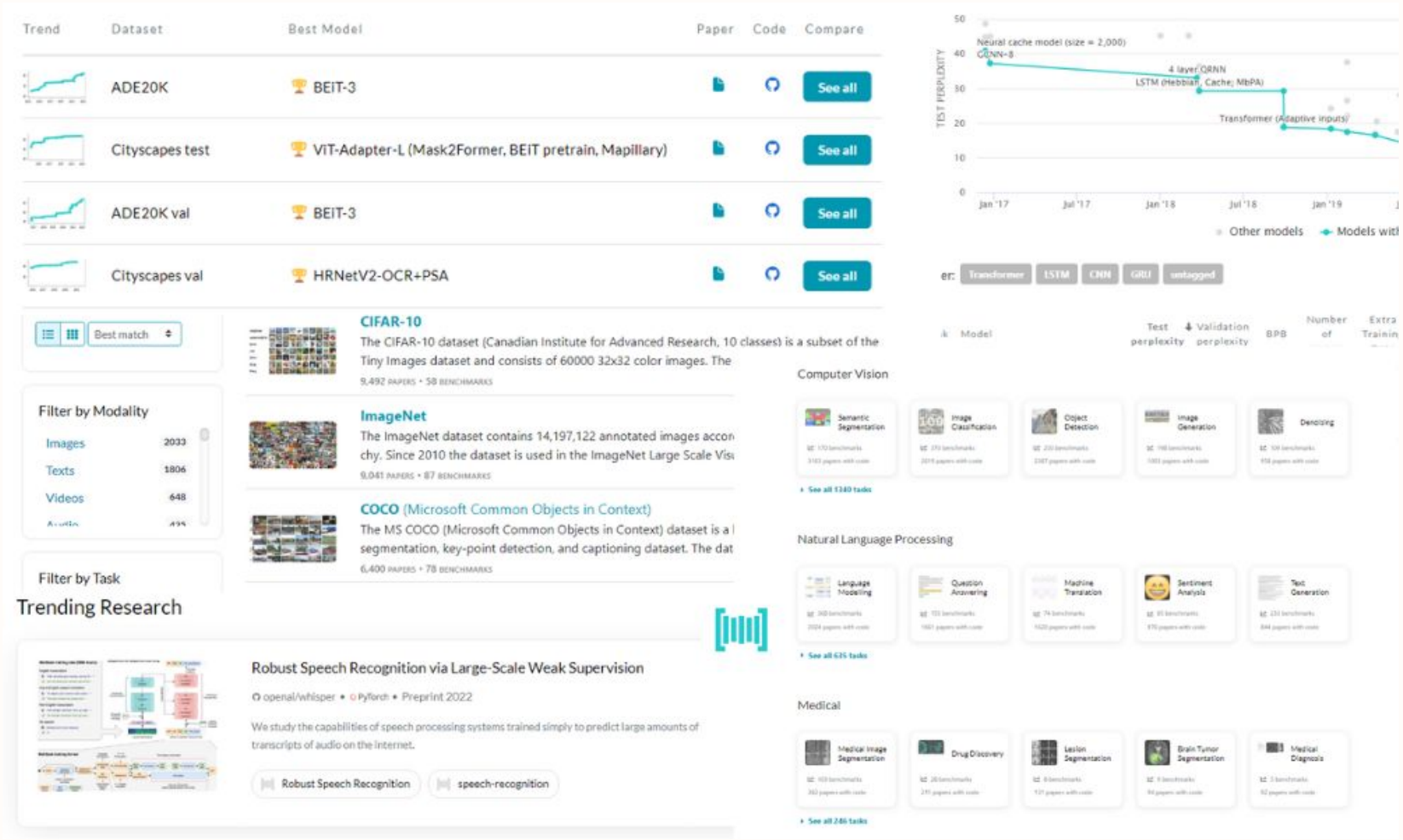
- ➔ Desenvolvimento de chips especializados para processar operações em domínios específicos
  - ◆ GPUs, Sistemas embarcados, SoCs



Fonte: https://www.qualcomm.com/news/onq/2022/01/snapdragon-ride-sdk-premium-solution-developing-customizable-adas-and-autonomous

Aprendizado de máquina e profundo - Datasets e códigos

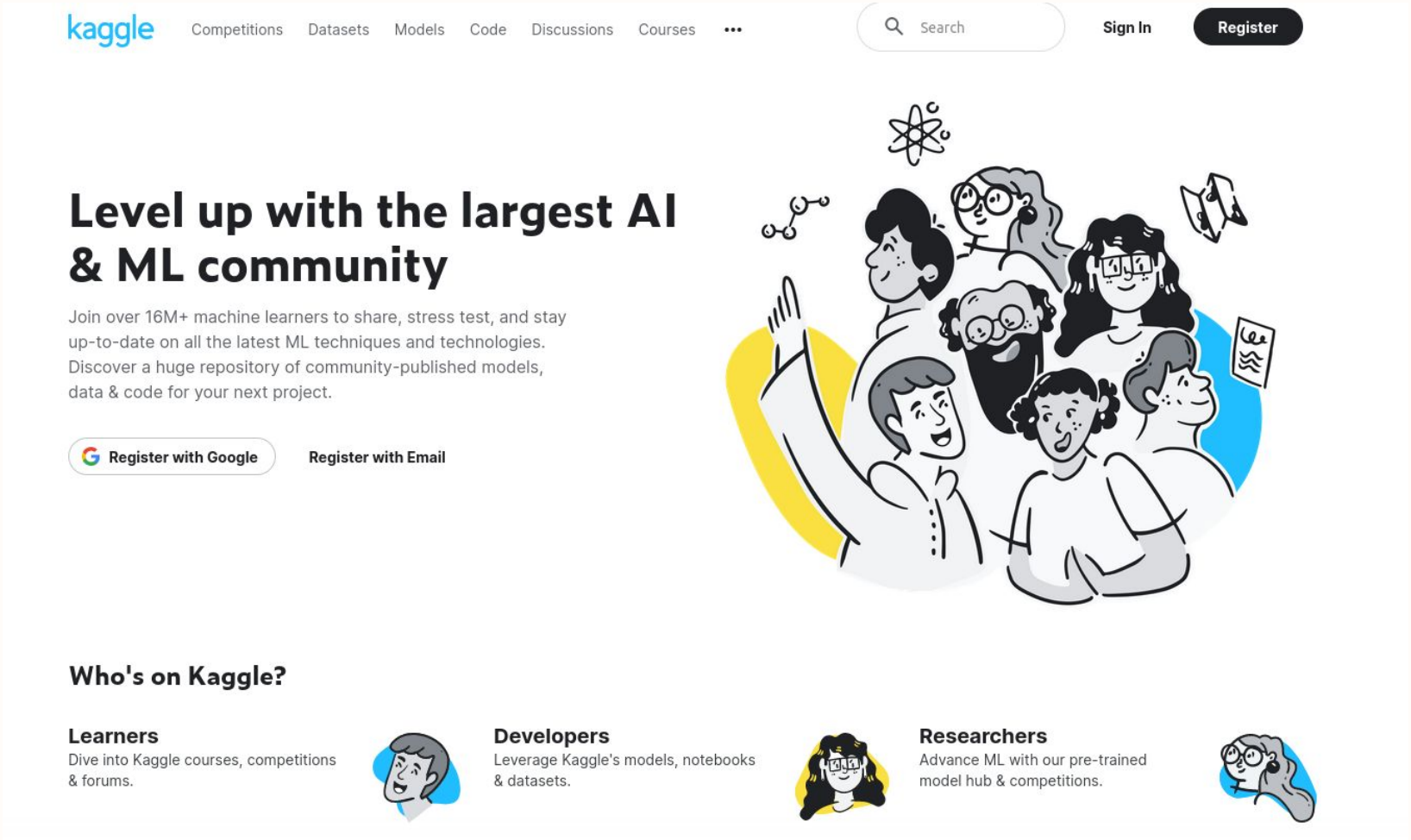
➔ Repositórios: UCI Repository, Kaggle, Papers with Code, etc



Fonte: https://paperswithcode.com

Aprendizado de máquina e profundo - Datasets e códigos

➔ Repositórios: UCI Repository, Kaggle, Papers with Code, etc



Fonte: https://www.kaggle.com/

Aprendizado de máquina e profundo - Datasets e códigos - áreas e aplicações

ADVANCED NEW

Data Parallelism: How to Train Deep Learning Models on Multiple GPUs

8 hours | PyTorch, PyTorch Distributed Data Parallel, NCCL

Certificate Available

View Course ›

FUNDAMENTALS NEW

Building Transformer-Based Natural Language Processing Applications

8 hours | PyTorch, Pandas, NVIDIA NeMo ™, NVIDIA Triton ™ Inference Server

Certificate Available

View Course ›

FUNDAMENTALS NEW

Applications of AI Anomaly Detection

8 hours | cuDF, CuPy, TensorFlow 2, NVIDIA Triton Inference Server

Certificate Available

View Course ›

FUNDAMENTALS POPULAR

Fundamentals of Accelerated Computing with CUDA Python

8 hours | CUDA, Python, Numba, NumPy

Certificate Available

View Course ›

FUNDAMENTALS NEW

Model Parallelism: Building and Deploying Large Neural Networks

8 hours | PyTorch, Megatron-LM, DeepSpeed, Slurm, Triton Inference Server, NVIDIA Nsight™

Certificate Available

View Course ›

FUNDAMENTALS POPULAR

Fundamentals of Deep Learning

8 hours | TensorFlow 2, Keras, pandas

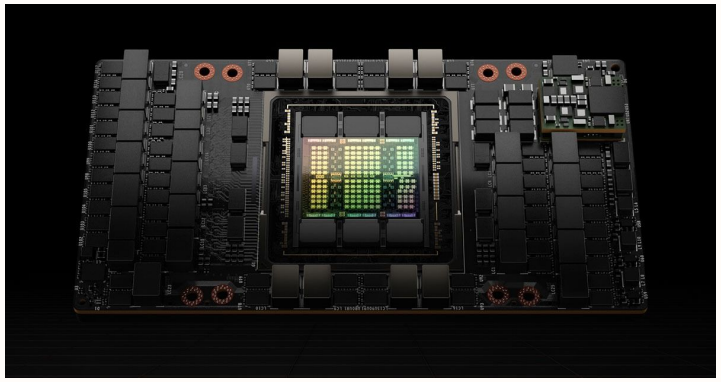
Certificate Available

View Course ›

Fonte: https://www.nvidia.com/en-us/training/enterprise-solutions/

Limitações

- ➔ Computação de alto desempenho
- ➔ Base de dados
- ➔ Interpretabilidade

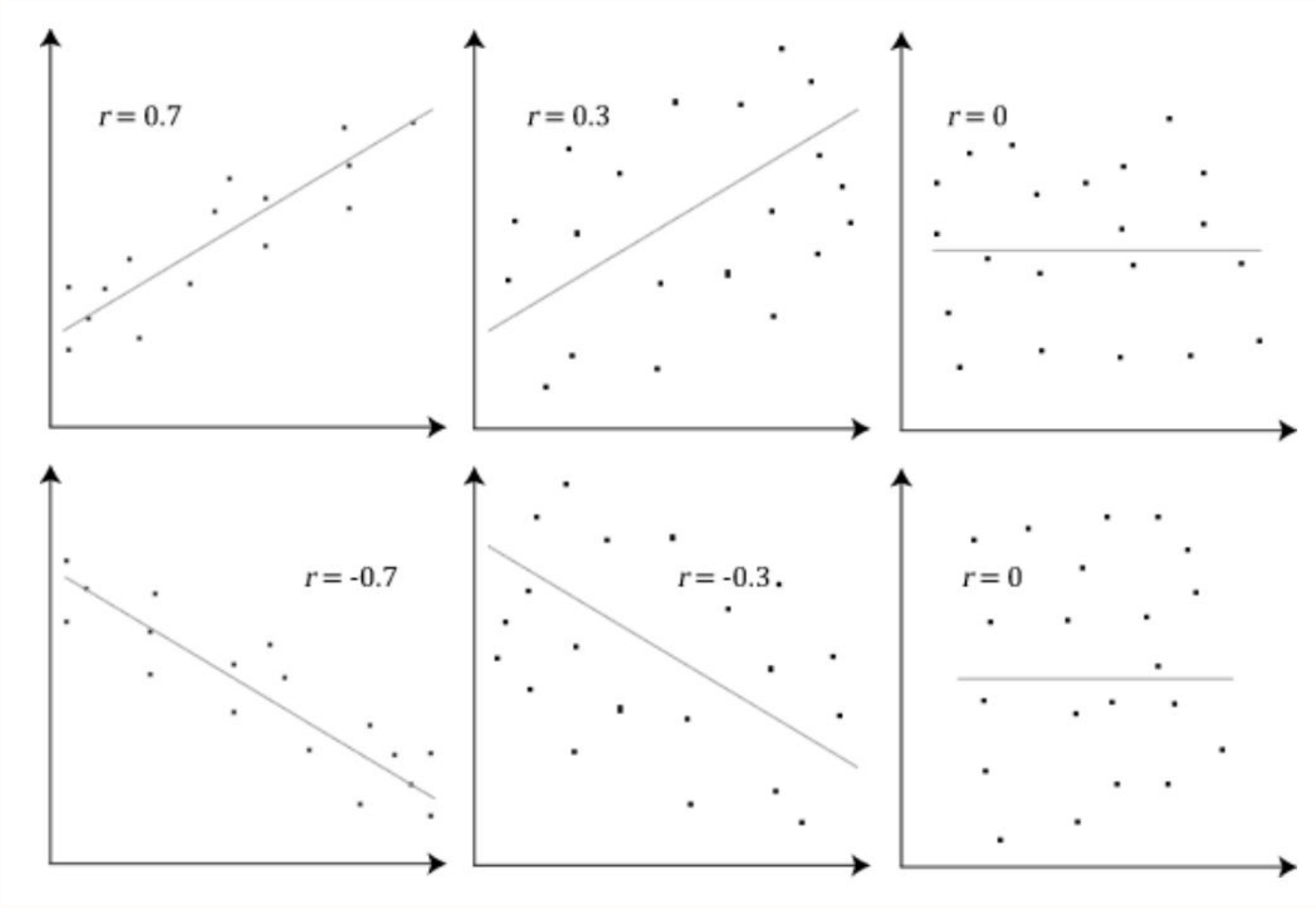


Fonte: https://data-science.llnl.gov/latest/news/new-partnership-unleash-us-supercomputing-resources-fight-against-covid-19
https://www.nvidia.com/pt-br/data-center/h100/

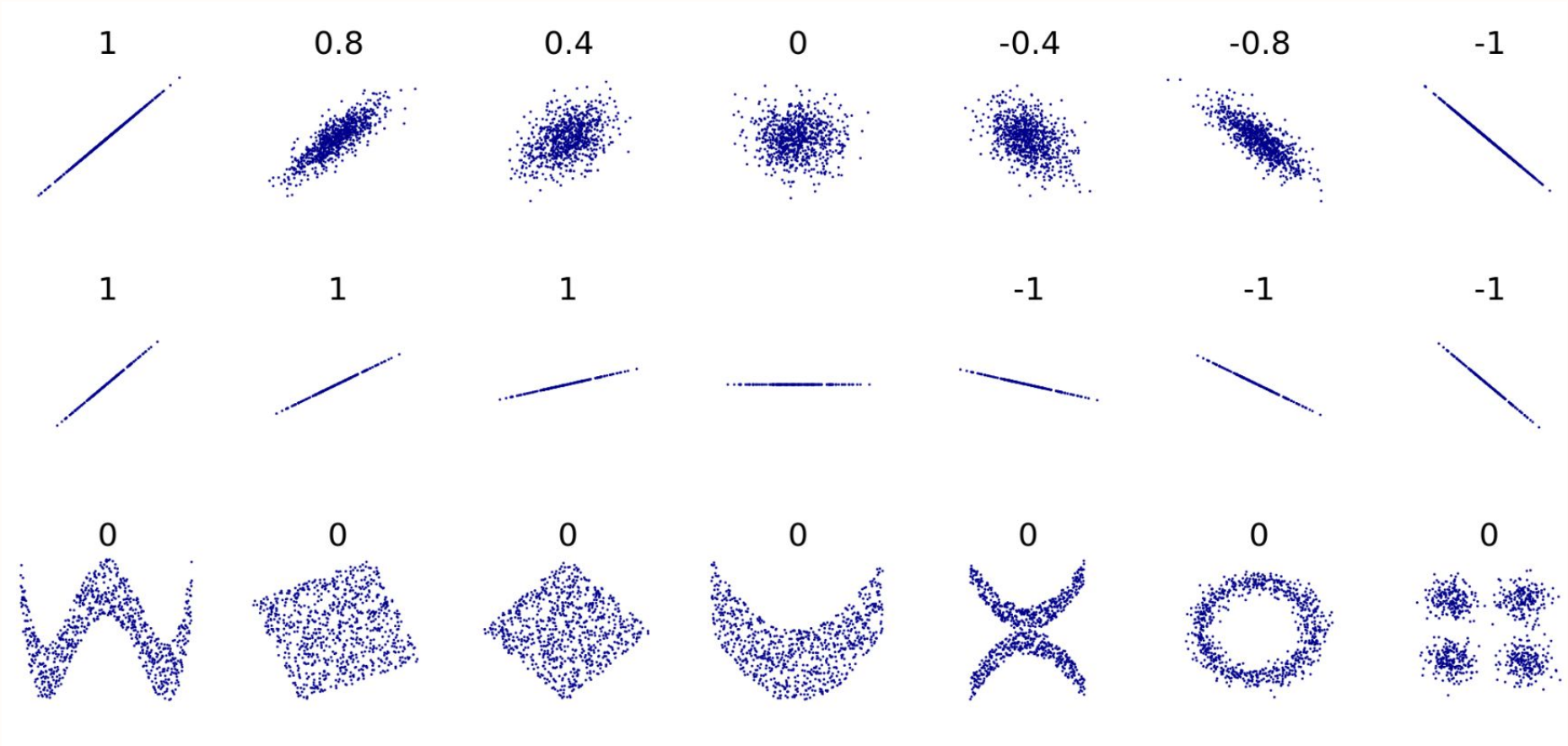
Modelos Lineares

Correlação linear

➔ Reflete a relação linear entre um conjunto de dados



Correlação linear



Correlação linear - Pearson

➔ Para um conjunto de dados {(x1, y1), (x2, y2), ..., (xn, yn)} a correlação de Pearson entre as variáveis x,y é dada por:

$$\rho_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \tilde{x})(y_i - \tilde{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \tilde{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \tilde{y})^2}}$$

➔ Com:

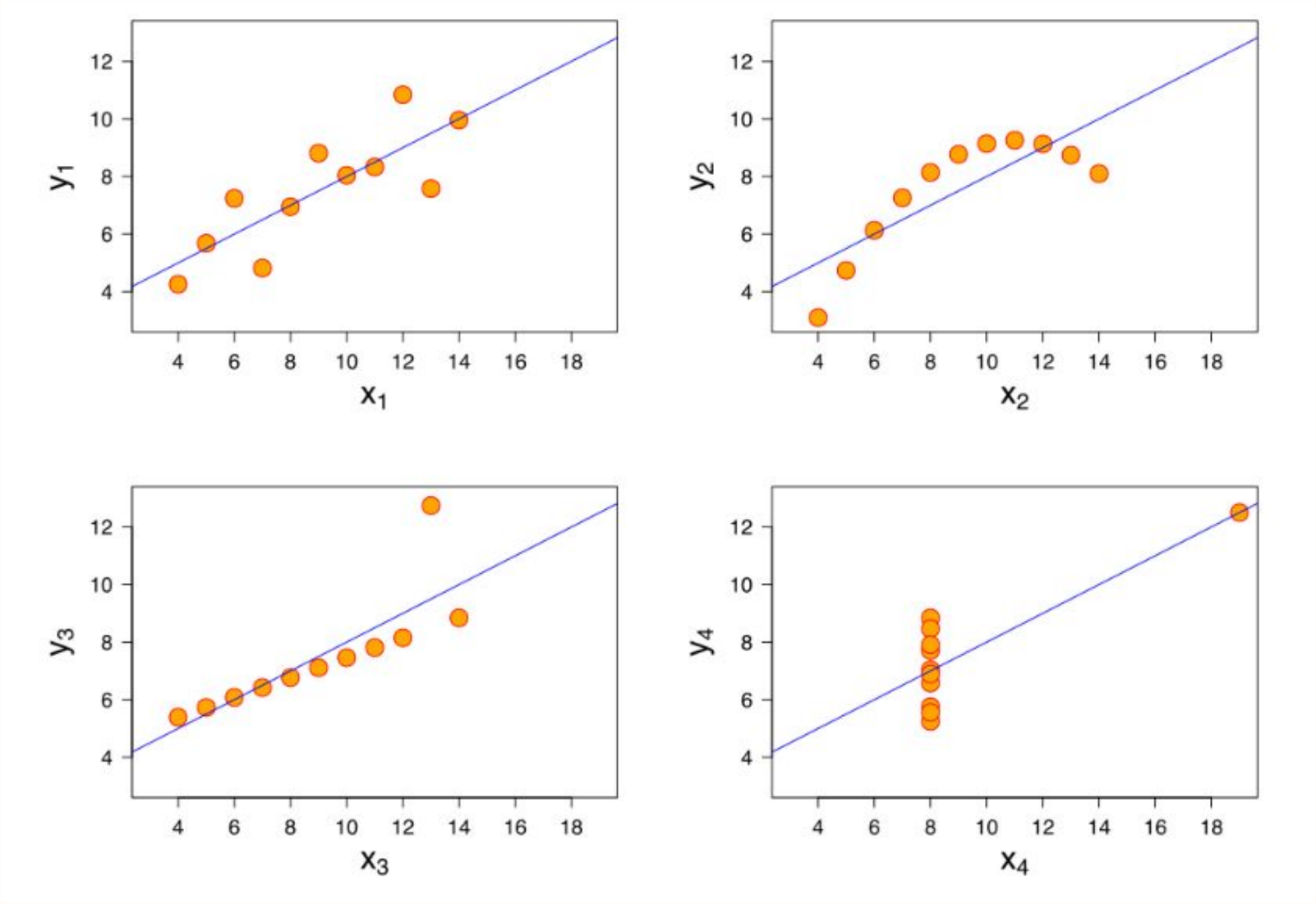
$$\tilde{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Correlação linear - Pearson

➔ Seu valor varia entre o intervalo [-1, 1], onde:

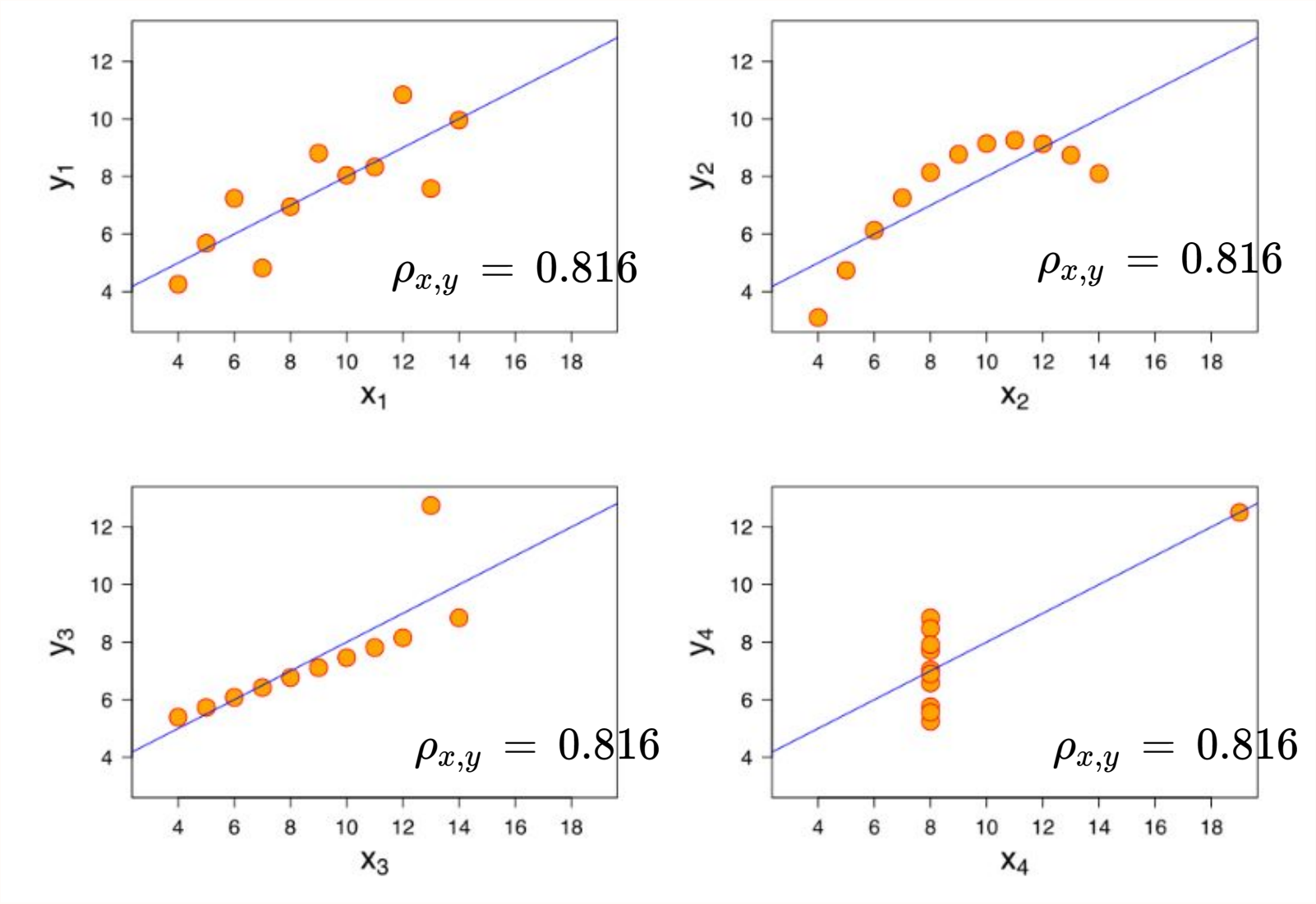
- **-1** - quando uma variável cresce, a outra diminui proporcionalmente.
- **1** - quando uma variável cresce, a outra cresce proporcionalmente.
- **0** - não há nenhuma relação entre as variáveis

Correlação linear - Pearson



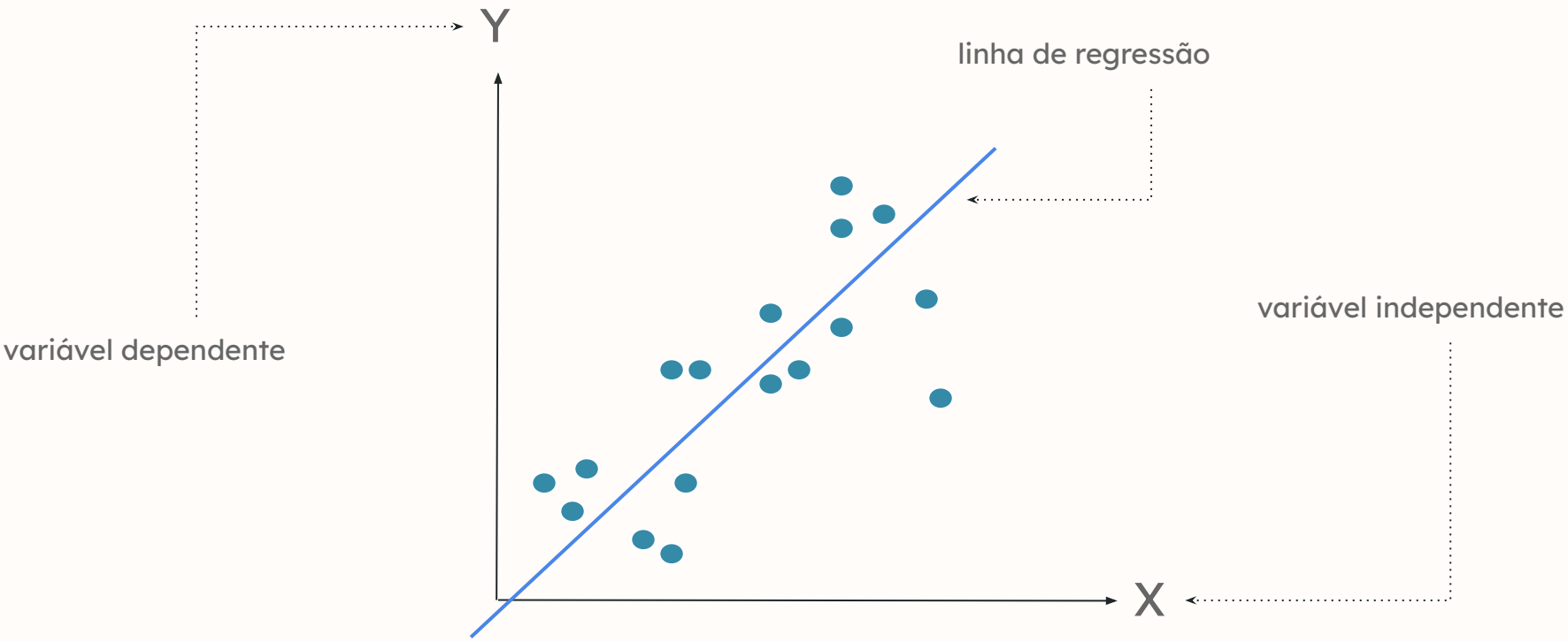
Correlação linear - Pearson

Mesmo valor de correlação para os 4 conjuntos



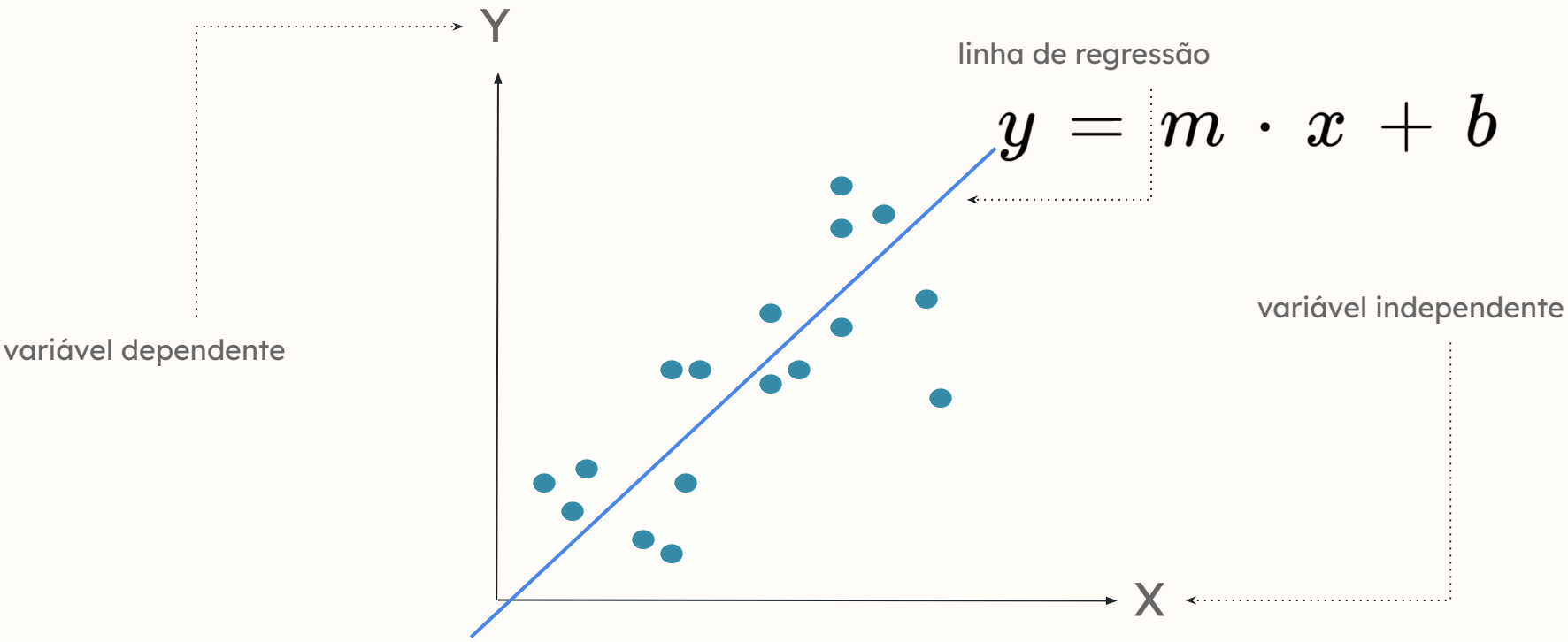
Regressão linear

- **Regressão:** Tem como objetivo realizar predições numéricas
  - **Regressão linear**: prediz a partir da relação entre as entradas do modelo (variáveis independentes) e a saída a ser predita (variável dependente) utilizando modelos lineares



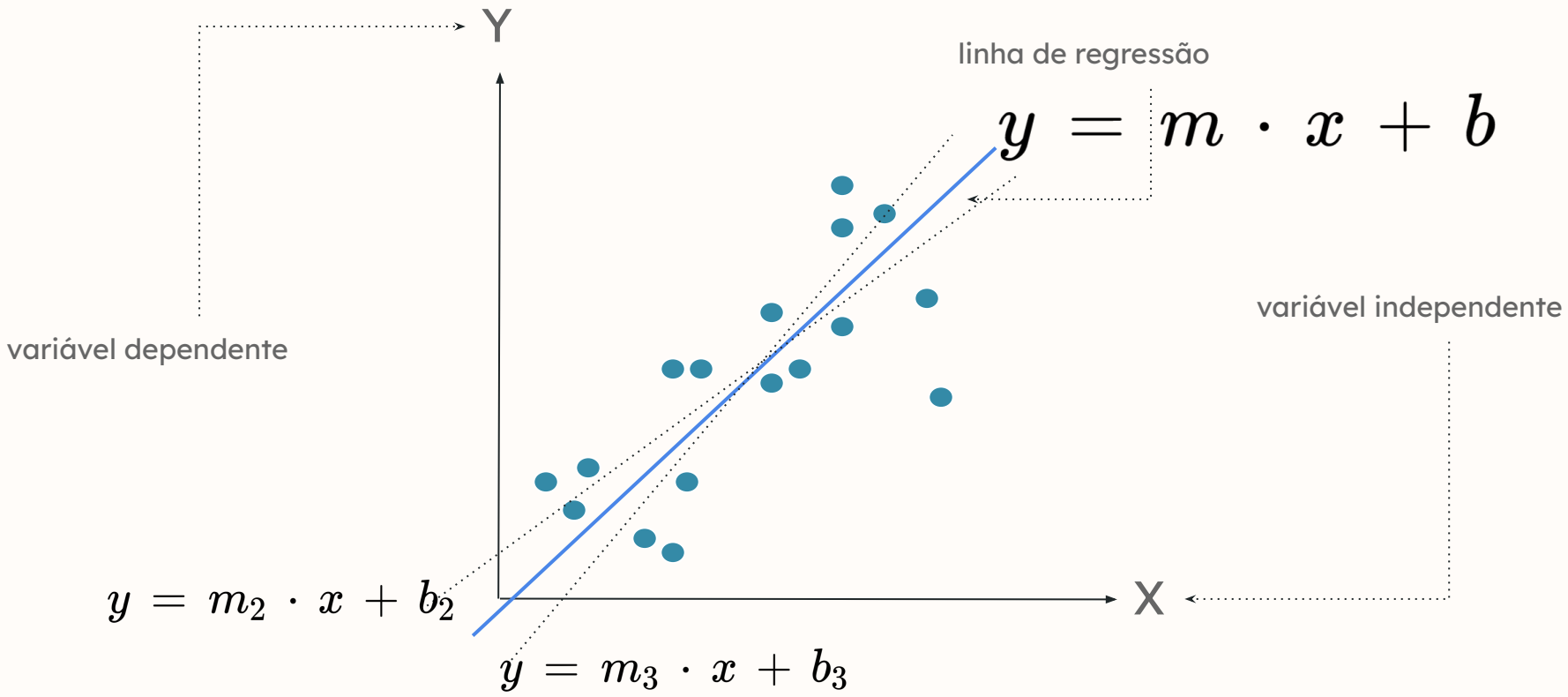
Regressão linear

- **Regressão:** Tem como objetivo realizar predições numéricas
  - **Regressão linear**: prediz a partir da relação entre as entradas do modelo (variáveis independentes) e a saída a ser predita (variável dependente) utilizando modelos lineares



Regressão linear

- **Regressão:** Tem como objetivo realizar predições numéricas
  - **Regressão linear**: prediz a partir da relação entre as entradas do modelo (variáveis independentes) e a saída a ser predita (variável dependente) utilizando modelos lineares



Regressão linear

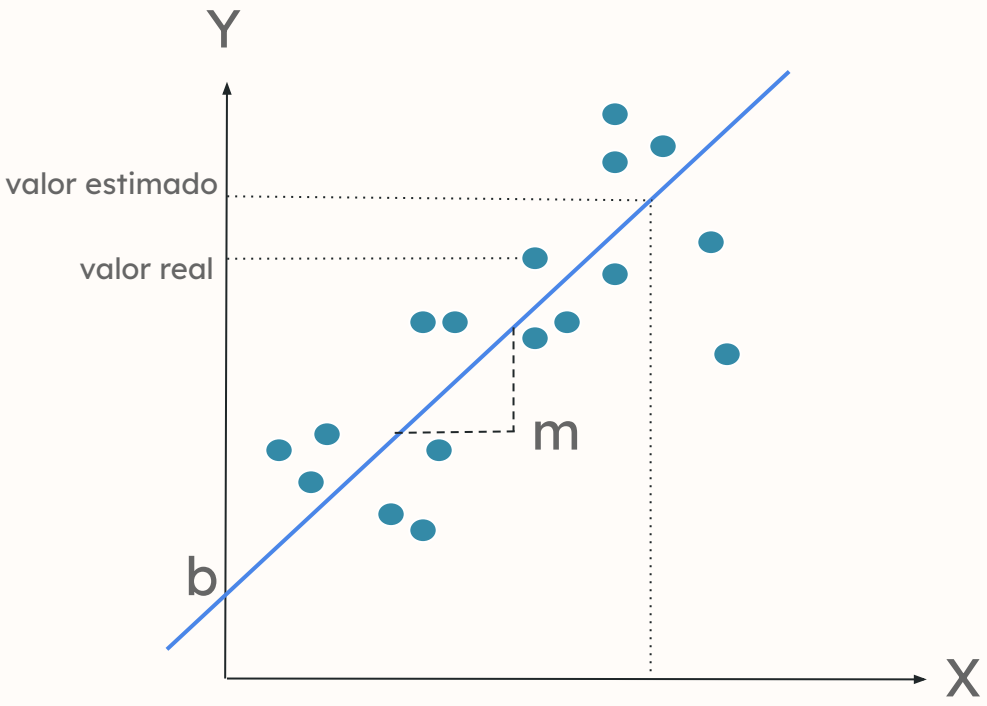
variável dependente / features de entrada

$$y = m \cdot x + b$$

variável dependente / variável de saída

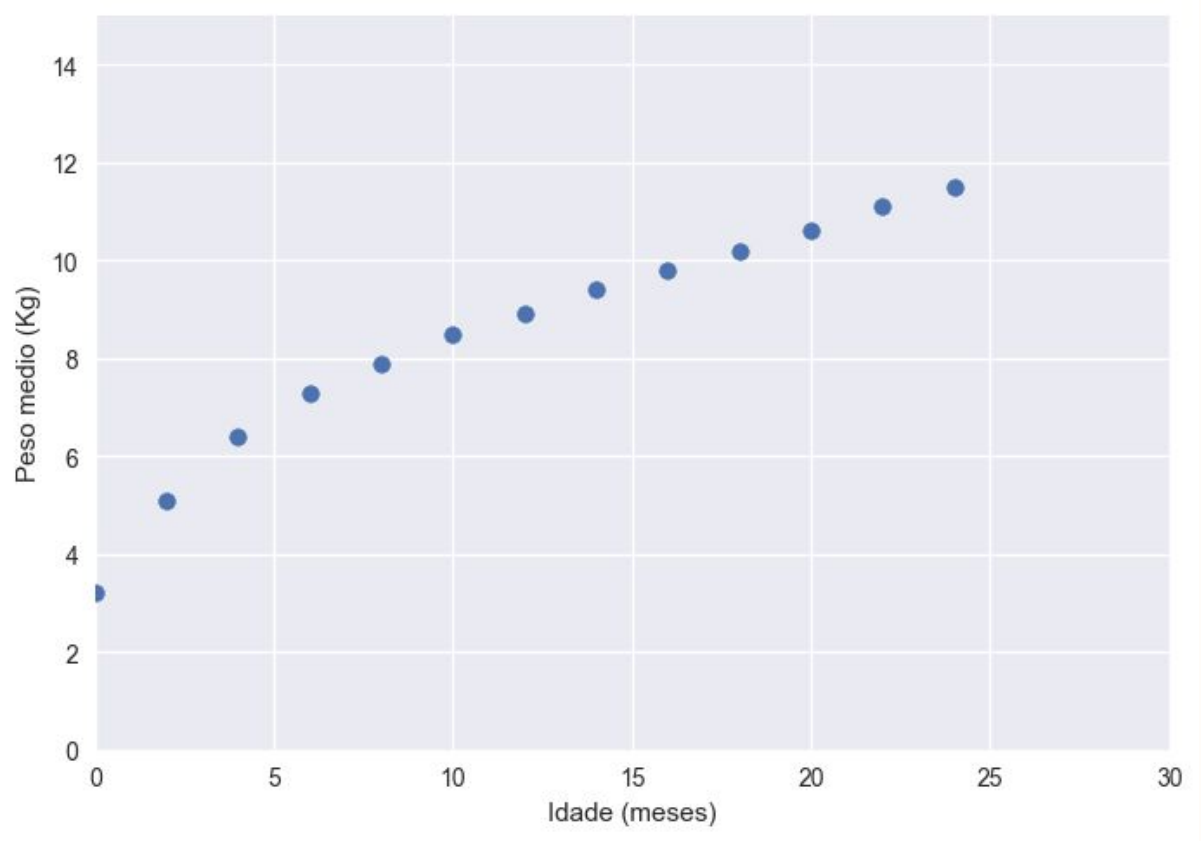
inclinação da curva

intersecção no eixo da variável dependente (bias)



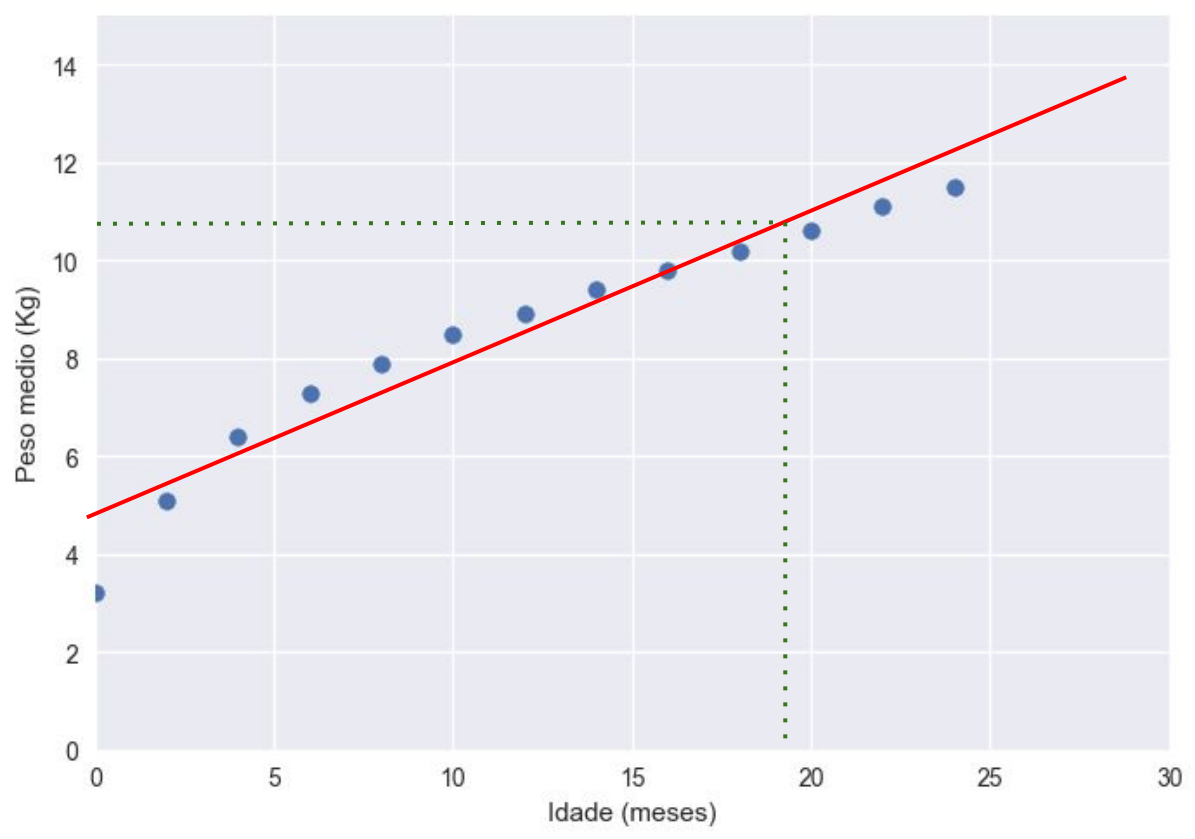
Regressão linear - Estimando peso

Idade (meses)	Peso medio (kg)
0	3.2
2	5.1
4	6.4
6	7.3
8	7.9
10	8.5
12	8.9
14	9.4
16	9.8
18	10.2
20	10.6
22	11.1
24	11.5



➔ Estimar o peso para 19 meses

Regressão linear - Estimando peso



➔ Estimar o peso para 19 meses

Regressão linear - Estimando peso

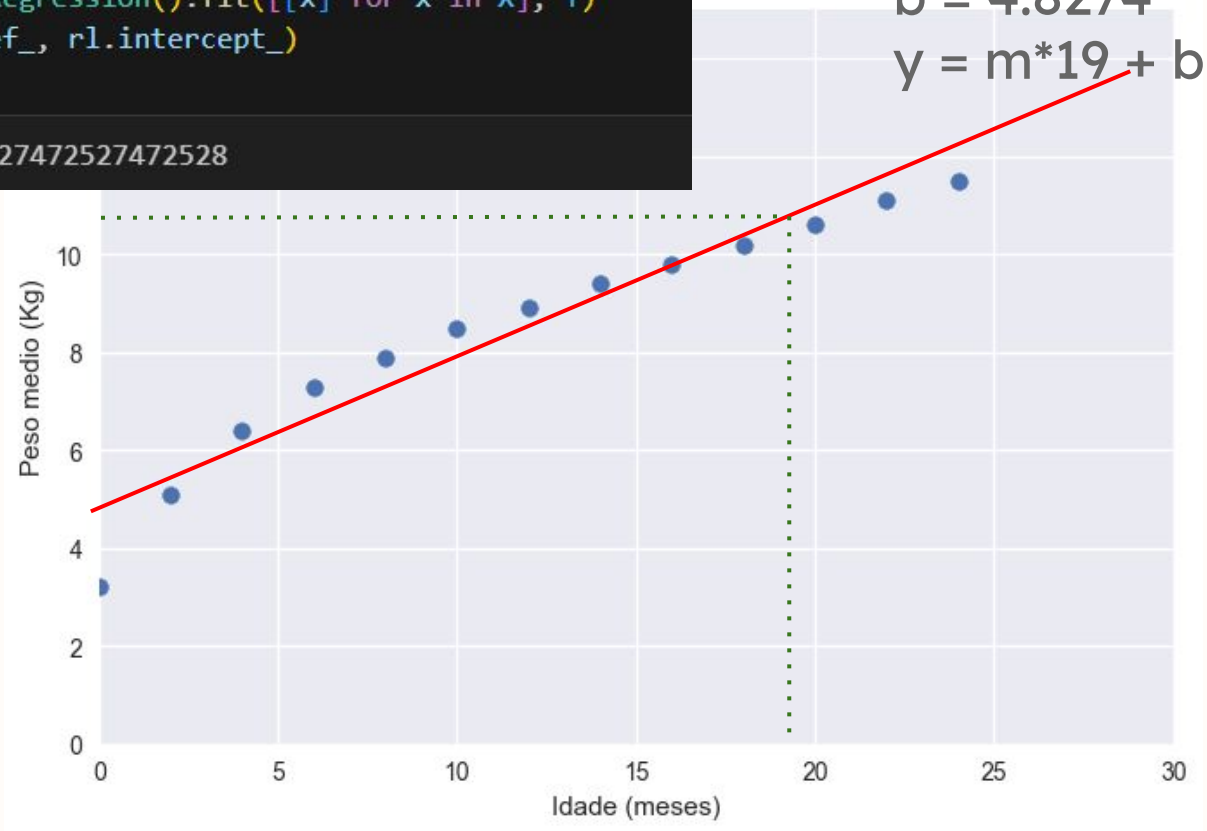
```
from sklearn.linear_model import LinearRegression

rl = LinearRegression().fit([[x] for x in X], Y)
print(rl.coef_, rl.intercept_)
```

✓ 0.0s

```
[0.3021978] 4.827472527472528
```

m = 0.30219
b = 4.8274
y = m*19 + b = 10.56



➔ Estimar o peso para 19 meses

Regressão linear

➔ No caso geral, a variável dependente pode ser calculada com base em diferentes características de entradas

$$\hat{y} = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

$$\hat{y} = \mathbf{w}^\top \mathbf{x} + b$$

Regressão linear

➔ No caso geral, a variável dependente pode ser calculada com base em diferentes características de entradas

$$\hat{y} = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

$$\hat{y} = \mathbf{w}^\top \mathbf{x} + b$$

y x1 x2 ...

# price	# area	# bedrooms	# bathrooms	# stories	✓ mainroa
Price of the Houses	Area of a House	Number of House Bedrooms	Number if Bathrooms	Number of House Stories	Weather c… Main Road
1.75m – 13.3m	1650 – 16.2k	1 – 6	1 – 4	1 – 4	
13300000	7420	4	2	3	yes
12250000	8960	4	4	4	yes
12250000	9960	3	2	2	yes

Correlação vs Regressão linear

- ➔ **Correlação linear:** análise a relação e conexão entre duas variáveis
- ➔ **Regressão linear:** Mostra como uma variável afeta outra e como podemos realizar predições/estimações baseado nesse comportamento
  - ◆ Para a correlação não faz diferença a ordem entre x e y (queremos identificar apenas a relação entre as variáveis)
  - ◆ Alterna a ordem entre x e y altera o resultado da regressão

Regressão linear - Função de custo

➔ Como calcular se a reta/função se adequa aos dados?

- Devemos encontrar o melhor valor para w e b
- Usamos os valores de X e y
- Precisamos quantificar o quanto a curva se adequa as informações do dataset

Função de Perda

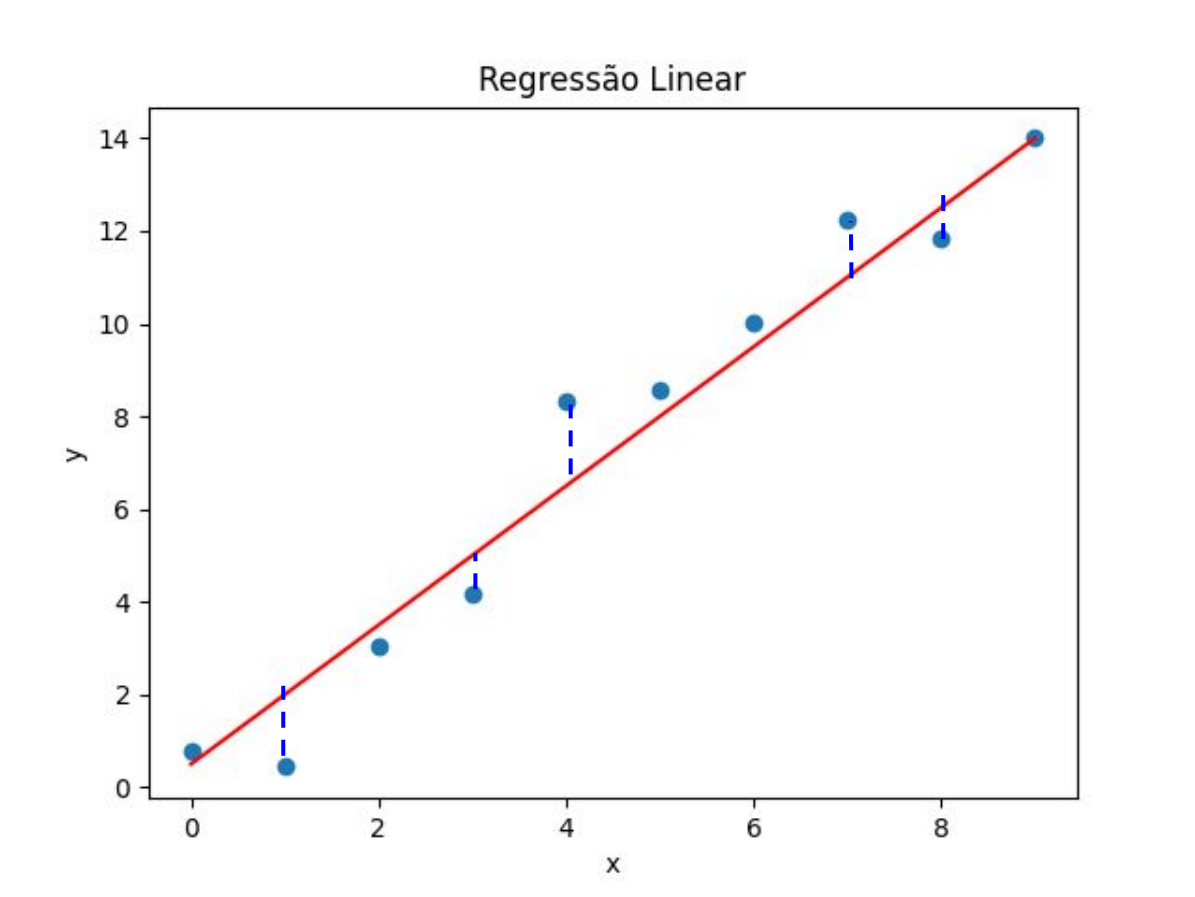
➔ Erro Médio Quadrático (MSE - *Mean Squared Error*) .

$$l^{(i)}(w, b) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

Função de Perda

➔ Erro Médio Quadrático (MSE - *Mean Squared Error*) .

$$l^{(i)}(w, b) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$



Função de Perda

➔ Para obter os erro associado às previsões, tira-se a média dos erros de cada instância.

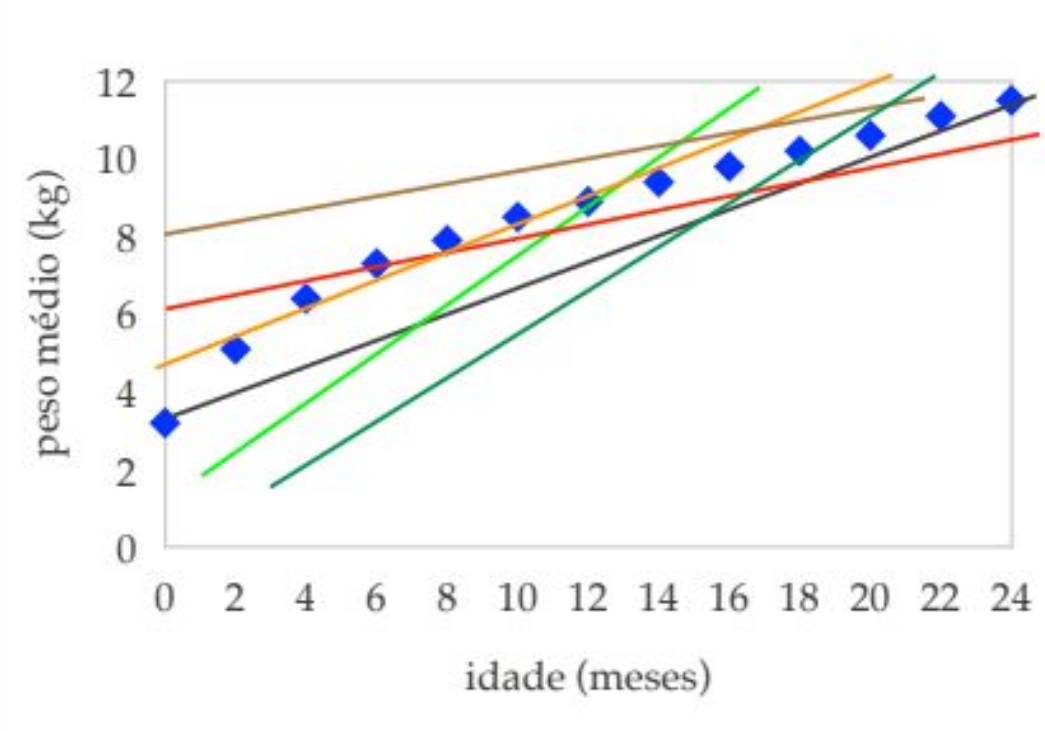
$$L(w,\ b) = \frac{1}{n}\sum_{i=1}^{n} l^{(i)}(w,\ b) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}\left(w^{\top} x^{(i)} + b - y^{(i)}\right)$$

➔ Tentamos minimizar o erro para encontrar o melhor ‘fit’

- ◆ Atualizamos o vetor de parâmetros (w) e os bias (b)

$$w^*,\ b^* = \underset{w,\ b}{\operatorname{argmin}}\ L(w,\ b)$$

Gradiente Descendente



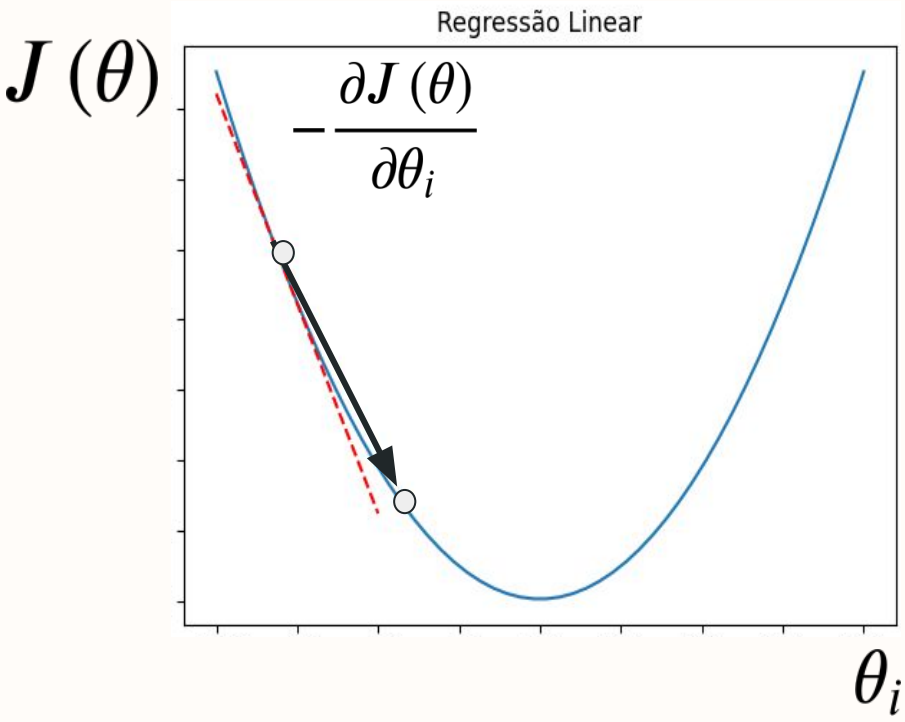
$$J(\theta) = \frac{1}{2}\sum_{i=0}^{n}(h_\theta(x^i) - y^i)^2$$

$$\theta^* = \arg\min_{\theta} J(\theta)$$

Gradiente Descendente

➔ O treinamento dos pesos dos modelos é atualizado iterativamente

- ◆ As derivadas parciais de uma função geram um vetor gradiente



$$\theta_i \leftarrow \theta_i - \eta \frac{\partial J(\theta)}{\partial \theta_i}$$

$\eta \rightarrow$ Taxa de aprendizagem

Gradiente Descendente

➔ Nesse caso, os parâmetros podem ser ajustados a partir da a função de perda MSE:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (h_\theta(x_i) - y_i)^2$$

➔ Para uma instância

$$\frac{\partial J(\theta)}{\partial \theta_i} = \frac{1}{2} \frac{\partial}{\partial \theta_i} (h_\theta(x_i) - y_i)^2$$

$$\frac{\partial J(\theta)}{\partial \theta_i} = (h_\theta(x_i) - y_i) \frac{\partial}{\partial \theta_i} (\theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m)$$

$$\boxed{\frac{\partial J(\theta)}{\partial \theta_i} = (h_\theta(x_i) - y_i) x_i}$$

$$\theta_i \leftarrow \theta_i - \eta \frac{\partial J(\theta)}{\partial \theta_i}$$

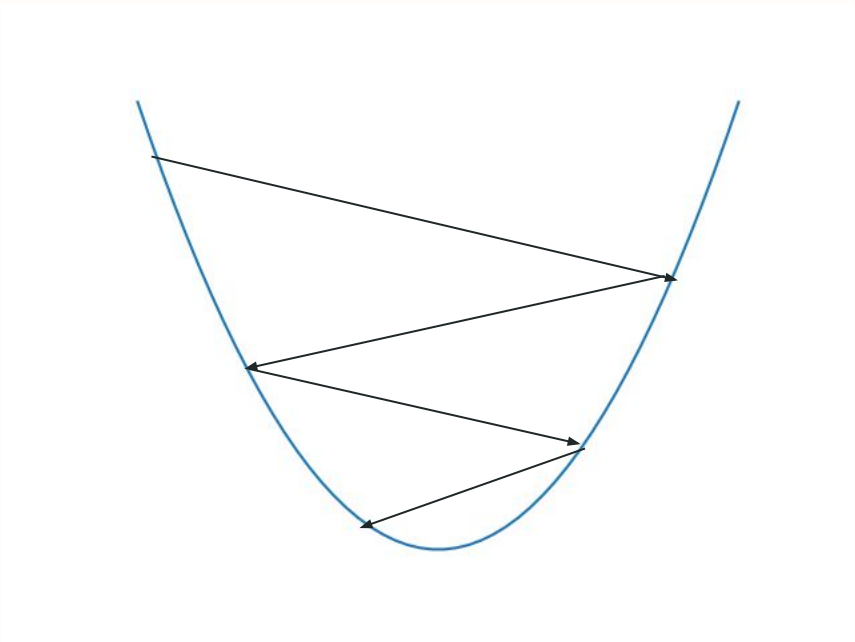
$$\theta_i \leftarrow \theta_i - \eta (h_\theta(x) - y) x_i$$

Gradiente Descendente - Taxa de Aprendizado

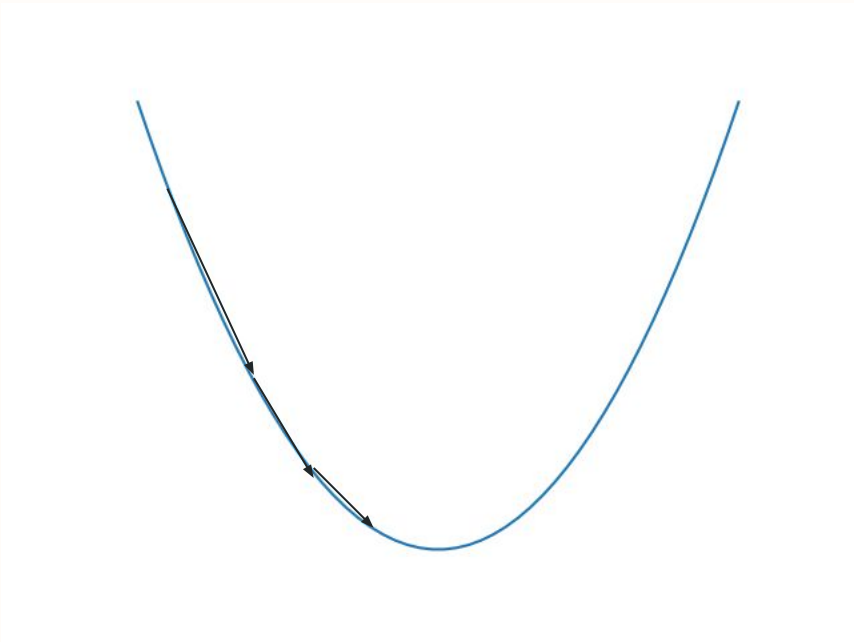
➔ É um parâmetro que vai indicar a velocidade de convergência dos parâmetros da rede.

$$\eta \rightarrow \text{Taxa de aprendizagem}$$

Alta



Baixa



Solução Analítica

➔ Solução Analítica: deriva-se em função de cada peso ***w*** e iguala-se a zero.

$$\partial_w \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 = 2\mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{y}) = 0$$

$$\mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \mathbf{X}\mathbf{w}$$

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$$

Solução Analítica

➔ Solução Analítica: deriva-se em função de cada peso ***w*** e iguala-se a zero.

$$\partial_w \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 = 2\mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{y}) = 0$$

$$\mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \mathbf{X}\mathbf{w}$$

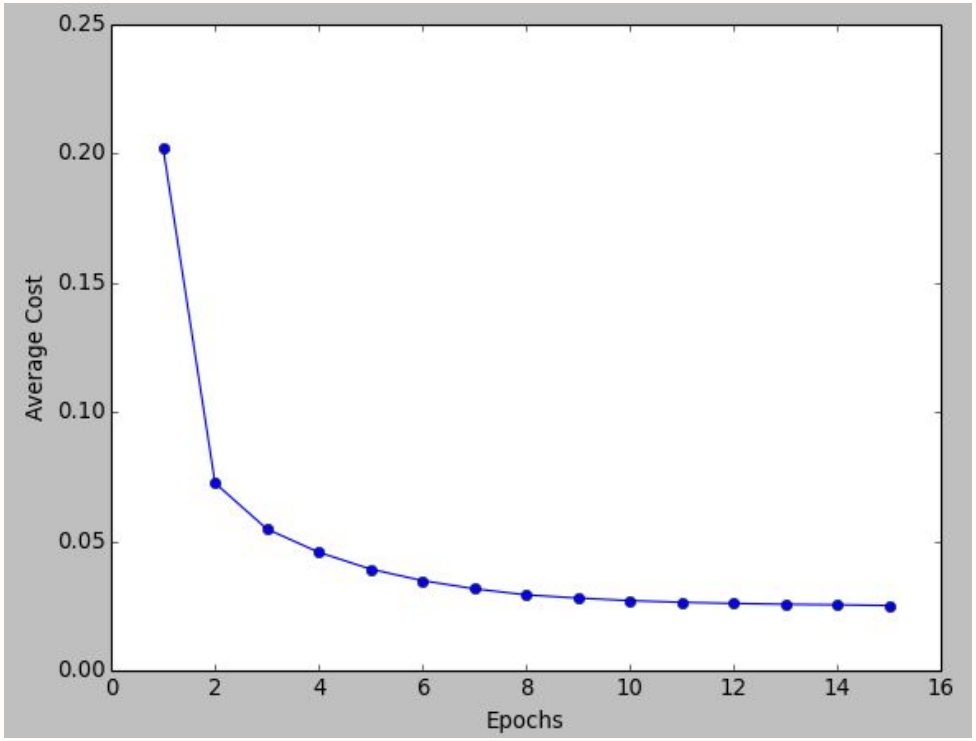
$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$$

➔ Contudo, essa solução depende da inversibilidade do termo abaixo, ou seja, quando suas colunas são linearmente independentes.

$$\mathbf{X}^\top \mathbf{X}$$

Batch Gradiente Descendente

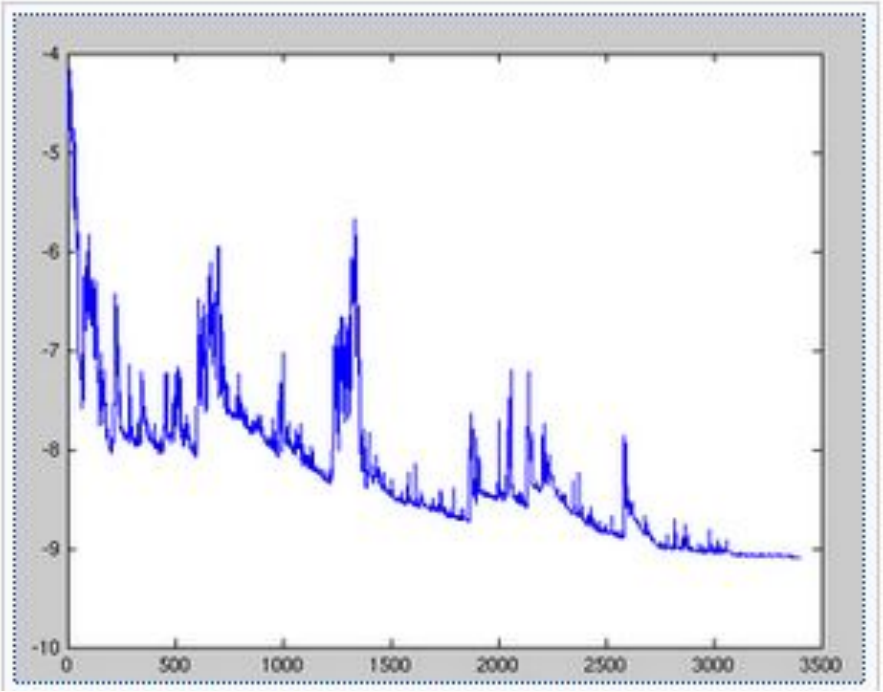
1. Calcula-se o erro para cada exemplo de treino
2. Os erros são acumulados
3. O somatório dos erros é usado para ajuste dos parâmetros
4. Repete os passos 1-3 até convergência ou critério de parada



Fonte: https://www.bogotobogo.com/python/scikit-learn/scikit-learn_batch-gradient-descent-versus-stochastic-gradient-descent.php#google_vignette

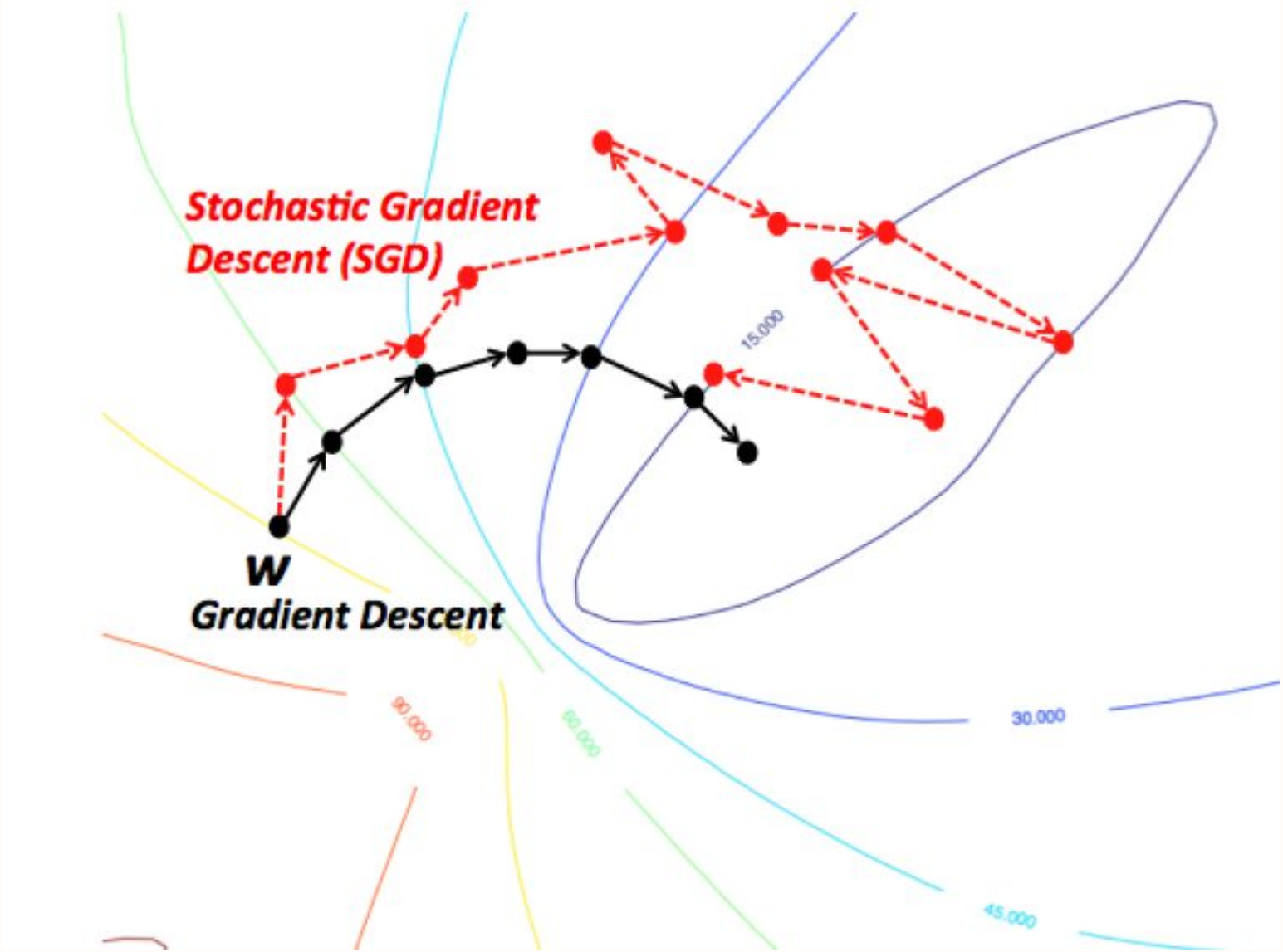
Mini Batch Gradiente Descendente

1. Seleciona-se uma amostra do conjunto de treino (mini batch)
2. Calcula-se o erro para cada exemplo de treino do batch
3. Os erros são acumulados
4. O somatório dos erros é usado para ajuste dos parâmetros
5. Repete os passos 1-4 até convergência ou critério de parada



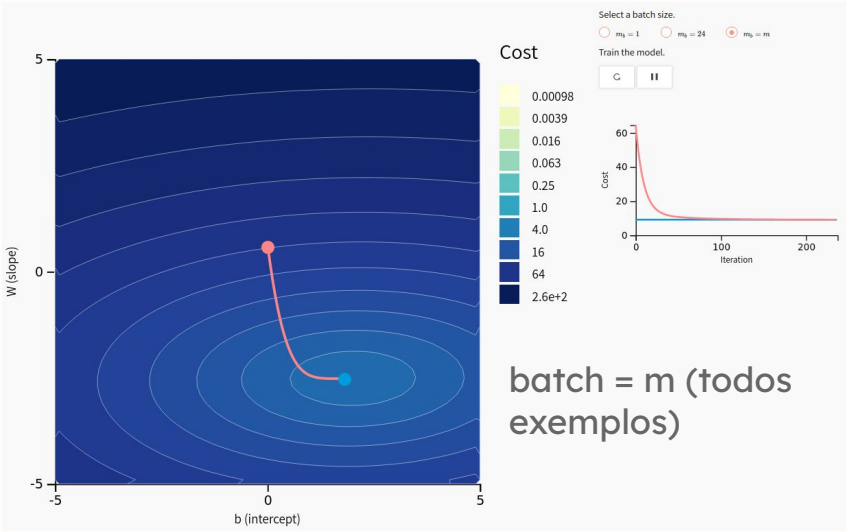
Fonte: https://www.bogotobogo.com/python/scikit-learn/scikit-learn_batch-gradient-descent-versus-stochastic-gradient-descent.php#google_vignette

Gradiente Descendente

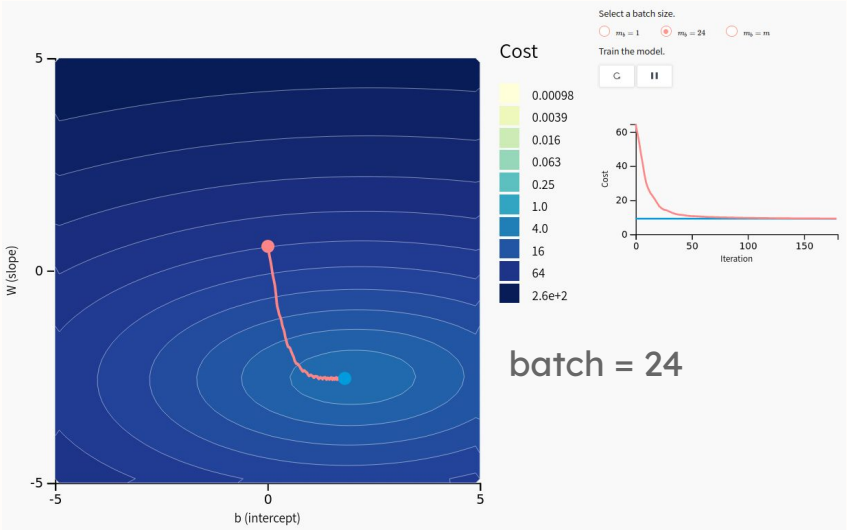


Fonte: https://www.bogotobogo.com/python/scikit-learn/scikit-learn_batch-gradient-descent-versus-stochastic-gradient-descent.php#google_vignette

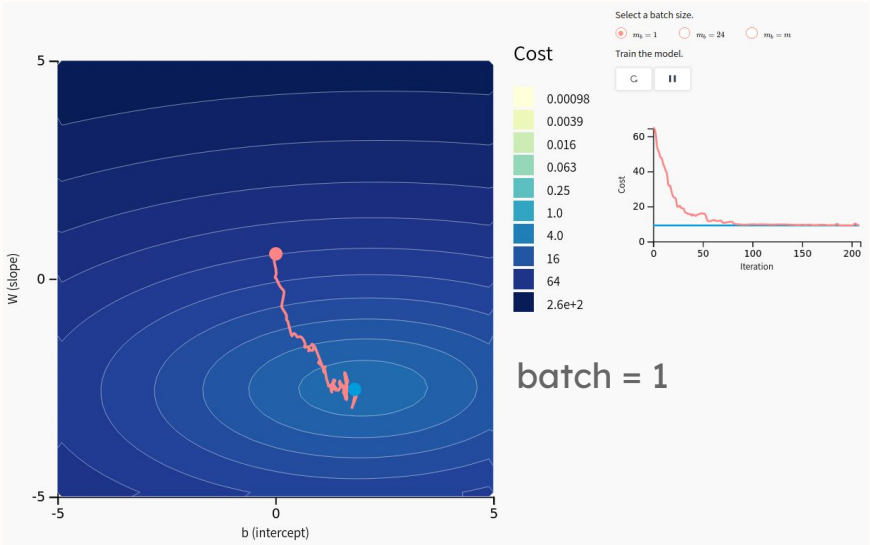
Gradiente Descendente



batch = m (todos exemplos)



batch = 24

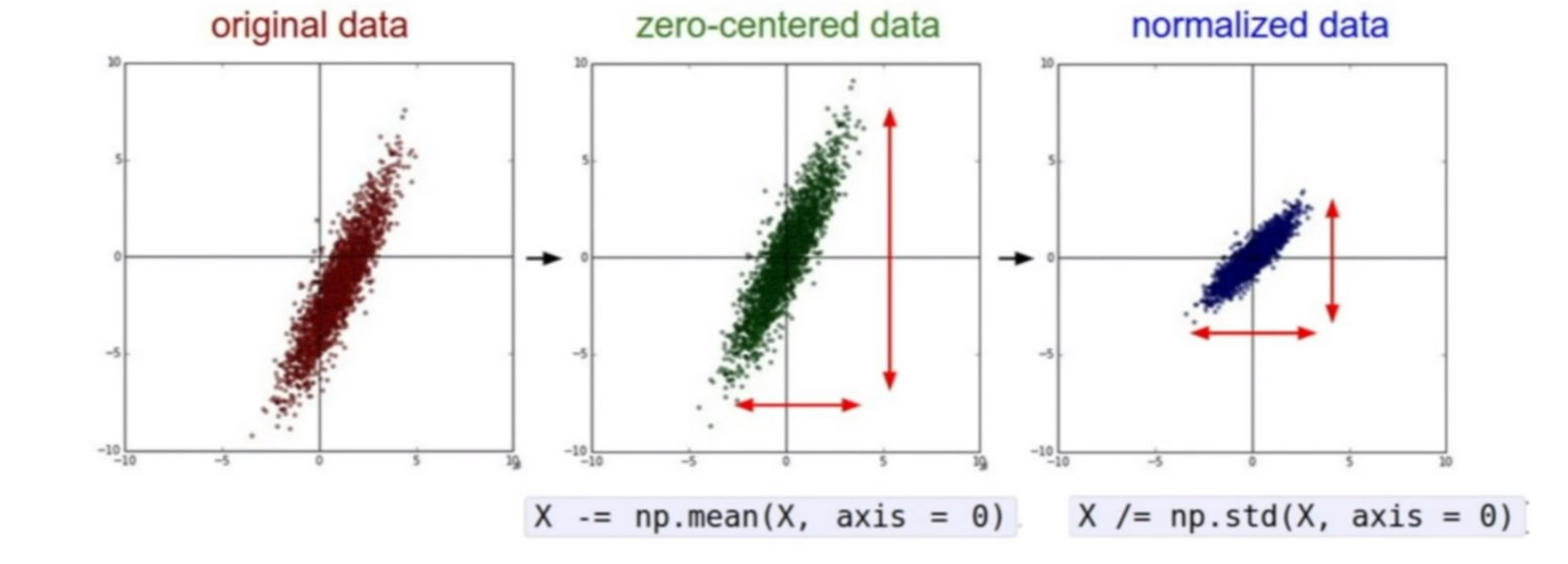


batch = 1

https://www.deeplearning.ai/ai-notes/optimization/index.html

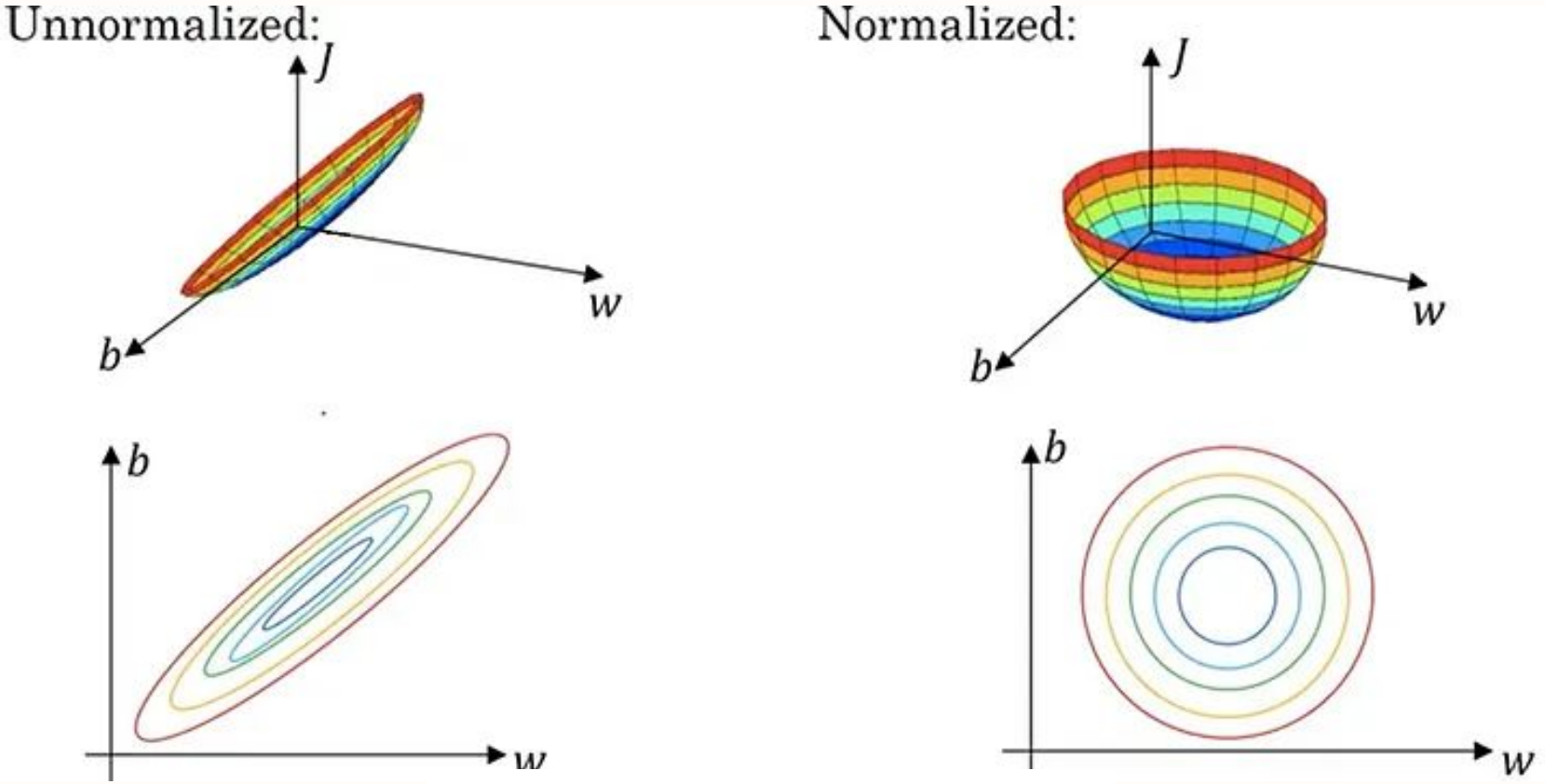
Gradiente Descendente - pré processamento (normalização e feature scaling)

- ➔ Acelera convergência do gradiente
- ➔ Necessário para ajustar a escala de features com intervalos diferentes (ex. peso: [10, 100], idade: [2, 50], altura: [0.3, 2.])



Fonte da Imagem: https://stackoverflow.com/questions/59540276/why-in-preprocessing-image-data-we-need-to-do-zero-centered-data

Gradiente Descendente - pré processamento (normalização e feature scaling)



$$x' = \frac{x - \mu}{\max(x) - \min(x)}$$

Fonte: deeplearning.ai | Andrew Ng
https://medium.com/analytics-vidhya/mean-normalization-and-feature-scaling-a-simple-explanation-3b9be7bfd3e8

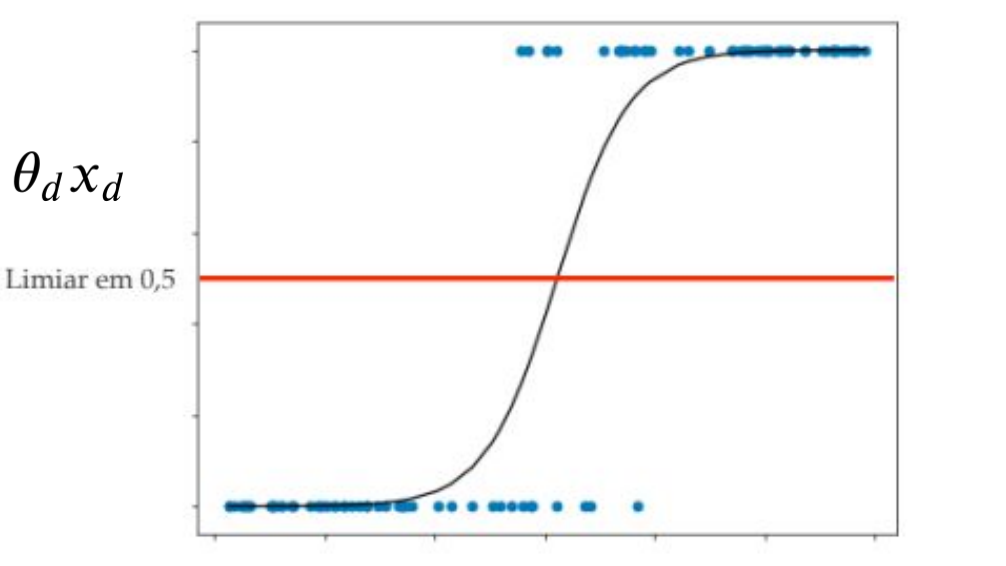
Regressão Logística

Regressão Logística

- ➔ Aplicado em classificação binária dos dados ao invés de valores reais e contínuos. Isto é, as saídas dos modelos podem indicar apenas dois valores 0 (zero) ou 1 (um).
  - ◆ Esses valores indicam a classe ao qual pertencem, por exemplo, se uma pessoa adquiriu determinada doença; se deve comprar um aparelho de celular etc.

$$\theta^\top x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$



Regressão Logística

➔ Interpretação probabilística

$$p(y = 1|x; \theta) = h_\theta(x)$$
$$p(y = 0|x; \theta) = 1 - h_\theta(x)$$
$$\longrightarrow p(y \mid x; \theta) = h_\theta(x)^y (1 - h_\theta(x))^{1-y}$$

➔ Função de Perda

$$L(\theta) = \prod_{i=1}^{n} p(y_i \mid x_i; \theta)$$

$$L(\theta) = \sum_{i=1}^{n} \log(p(y_i \mid x_i; \theta))$$

$$L(\theta) = \sum_{i=1}^{n} \left[y_i \log(h_\theta(x_i)) + (1 - y_i) \log(1 - h_\theta(x_i)) \right]$$

Regressão Logística

- Maximização da verossimilhança
  - Gradiente Ascendente

$$\frac{\partial l(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j}(y \log(h_\theta(x)) + (1-y)\log(1-h_\theta(x)))$$

$$\frac{\partial l(\theta)}{\partial \theta_j} = (y - h_\theta(x))\, x_j$$

$$\theta_i \leftarrow \theta_i + \eta \frac{\partial L(\theta)}{\partial \theta_i}$$

Regressão Logística

The Iris Dataset

This data sets consists of 3 different types of irises' (Setosa, Versicolour, and Virginica) petal and sepal length, stored in a 150x4 numpy.ndarray

The rows being the samples and the columns being: Sepal Length, Sepal Width, Petal Length and Petal Width.

X = (x1, x2, x3, x4)   y = [0,1,2]

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html

Regressão Logística

```
rlog = LogisticRegression(random_state=0).fit(X,y)
```
✓ 0.0s

```
rlog.predict(X[:2,:])
```
✓ 0.0s

```
array([0, 0])
```

```
rlog.predict_proba(X[:2,:])
```
✓ 0.0s

```
array([[9.81810819e-01, 1.81891663e-02, 1.44226885e-08],
       [9.71750652e-01, 2.82493178e-02, 3.01583887e-08]])
```

```
print(rlog.coef_, rlog.intercept_)
```
✓ 0.0s

```
[[-0.41795722  0.96618456 -2.52142059 -1.08400771]
 [ 0.53074362 -0.31438087 -0.19910389 -0.94894462]
 [-0.1127864  -0.65180369  2.72052448  2.03295233]] [  9.83828594   2.21506219 -12.05334813]
```

Regressão Logística

```
rlog = LogisticRegression(random_state=0).fit(X,y)
```
✓ 0.0s

```
rlog.predict(X[:2,:])
```
✓ 0.0s

```
array([0, 0])
```

```
rlog.predict_proba(X[:2,:])
```
✓ 0.0s

```
array([[9.81810819e-01, 1.81891663e-02, 1.44226885e-08],
       [9.71750652e-01, 2.82493178e-02, 3.01583887e-08]])
```

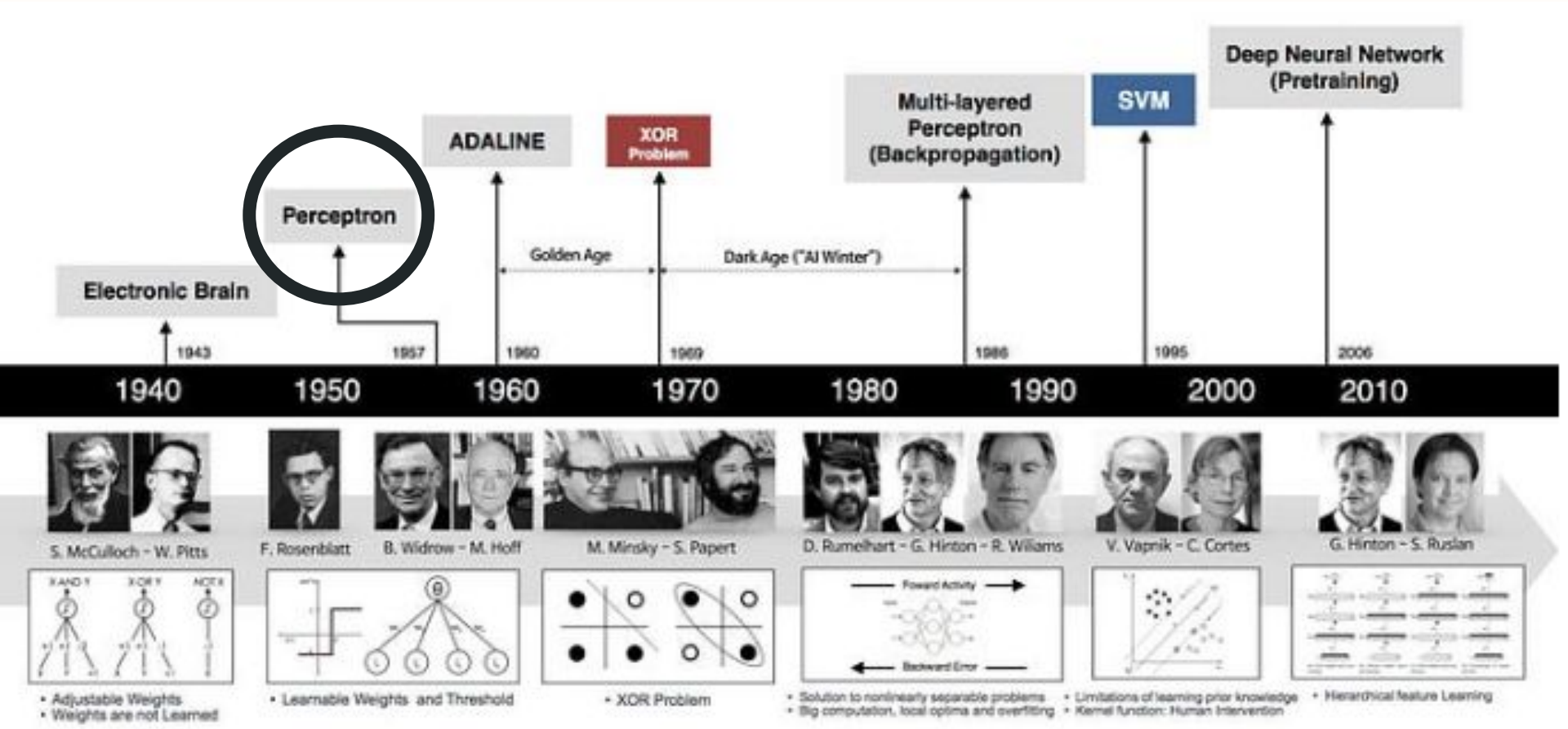
1 modelo para cada classe

```
print(rlog.coef_, rlog.intercept_)
```
✓ 0.0s

```
[[-0.41795722  0.96618456 -2.52142059 -1.08400771]
 [ 0.53074362 -0.31438087 -0.19910389 -0.94894462]
 [-0.1127864  -0.65180369  2.72052448  2.03295233]] [  9.83828594   2.21506219 -12.05334813]
```

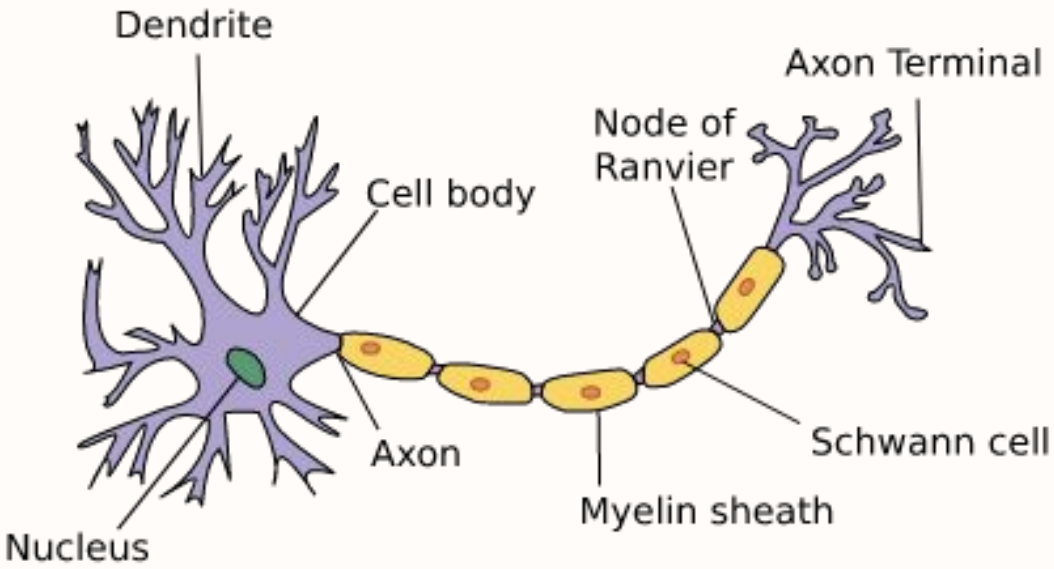
Perceptron

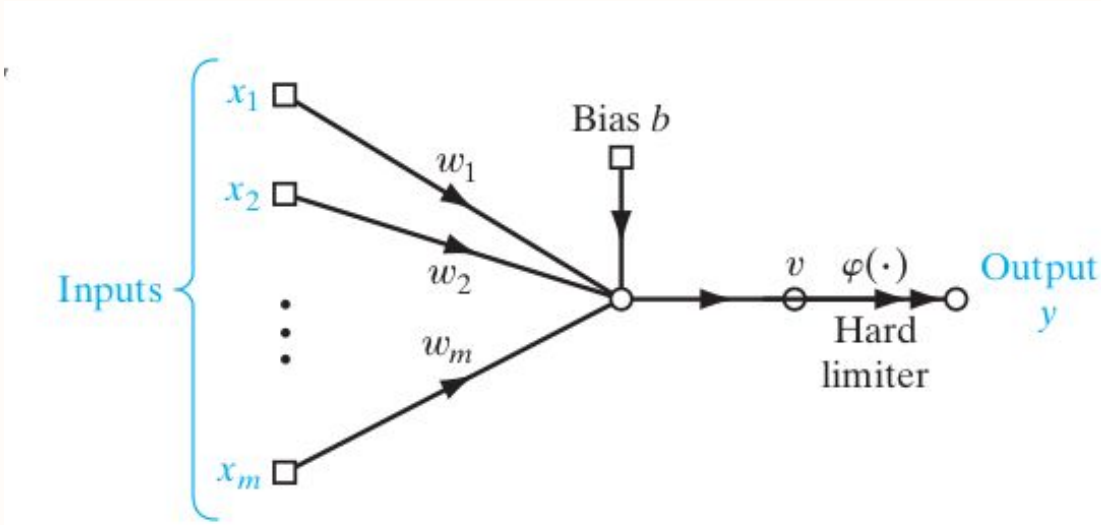
Histórico



https://medium.com/@augusto_Pinheiro/redes-neurais-artificiais-133de77c7240

Perceptron





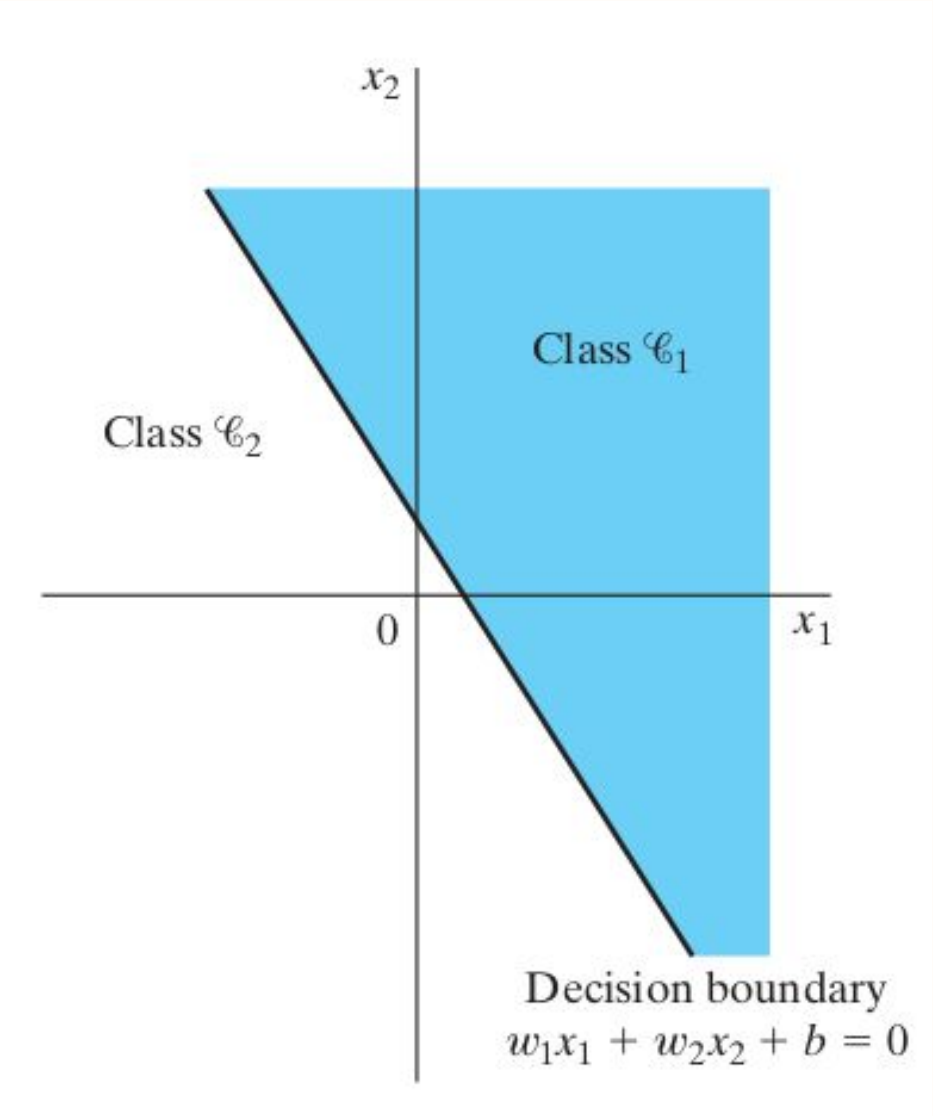
Fonte: “Anatomy and Physiology” by the US National Cancer Institute’s Surveillance, Epidemiology and End Results (SEER) Program).
Simon Haykins (2009)

Perceptron

$$v = \sum_{i=1}^{m} w_i x_i + b$$

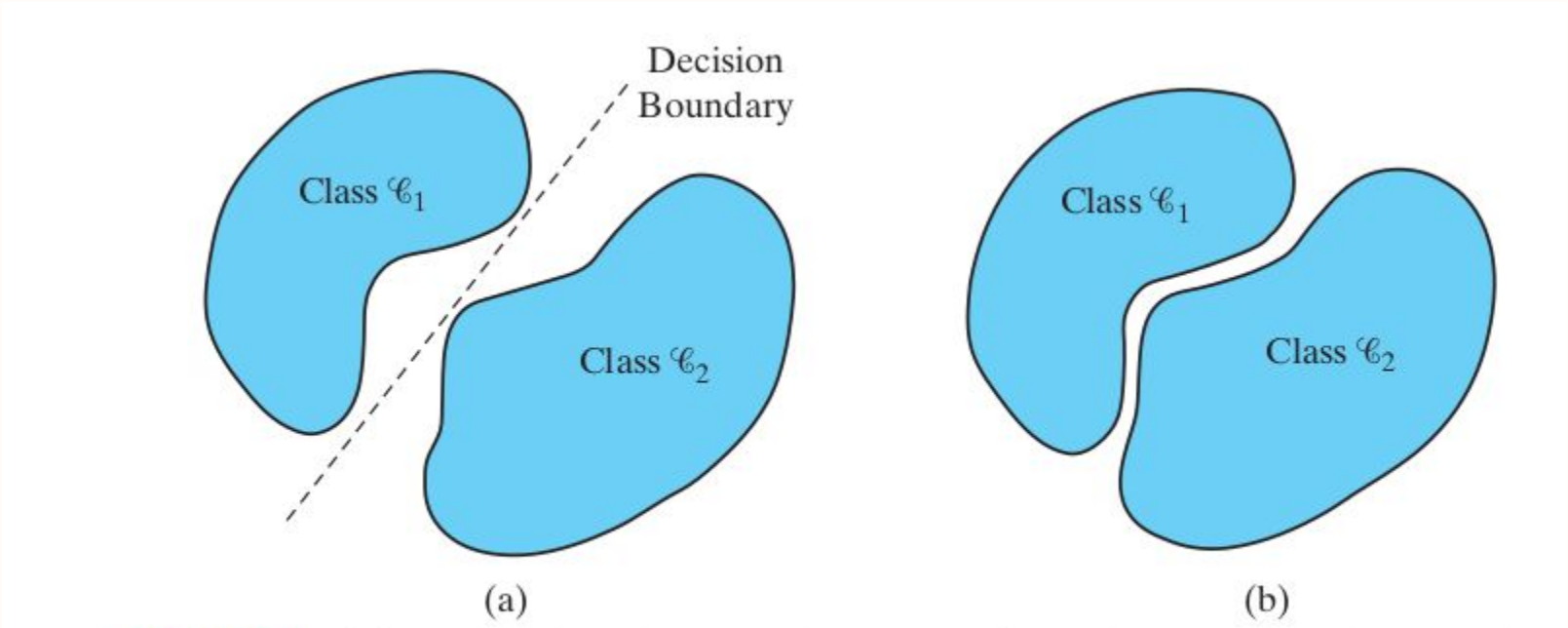
$$\text{sgn}(v) = \begin{cases} +1 & \text{if } v > 0 \\ -1 & \text{if } v < 0 \end{cases}$$

$$h(x) = sign\left(\left(\sum_{i=0}^{d} w_i x_i\right) + w_0\right)$$



Simon Haykins (2009)

Perceptron



a) Linearmente separável
b) Não-linearmente separável

Simon Haykins (2009)

Algoritmo de aprendizado do Perceptron

- O Perceptron implementa

$$h(\boldsymbol{x}) = sign(w^T \boldsymbol{x})$$

- Dado o conjunto de treinamento

$$(x_1, y_1), (x_2, y_2), \ldots, (x_d, y_d),$$

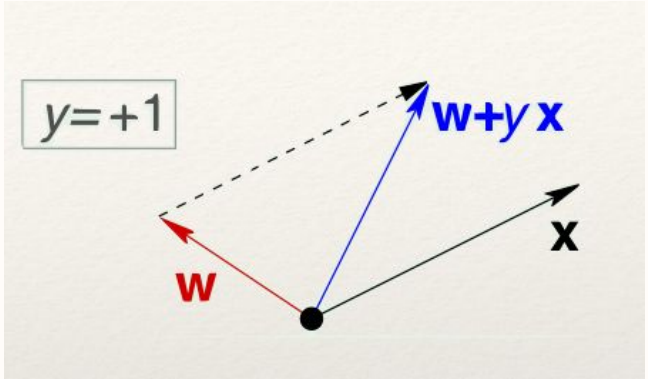
- Selecione um padrão classificado incorretamente

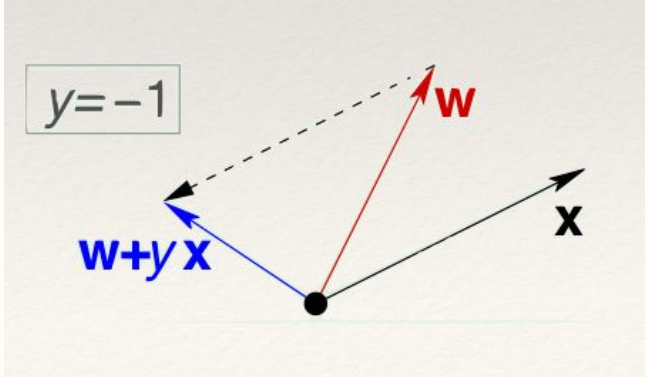
$$sign(w^T \boldsymbol{x}_n) \neq y_n$$

- Atualiza os pesos

$$w = w + y_n \boldsymbol{x}_n$$

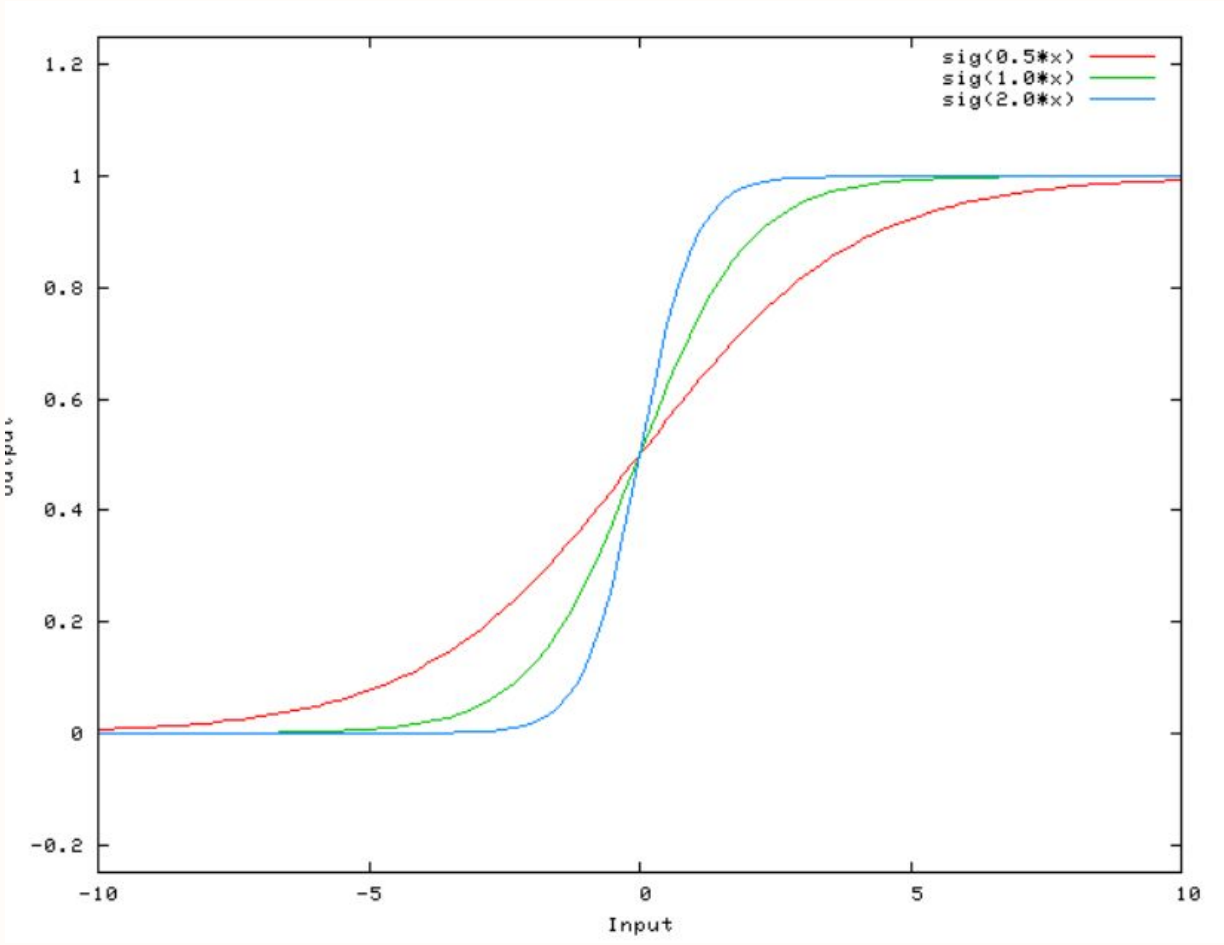
- Repetir os passos até convergir.





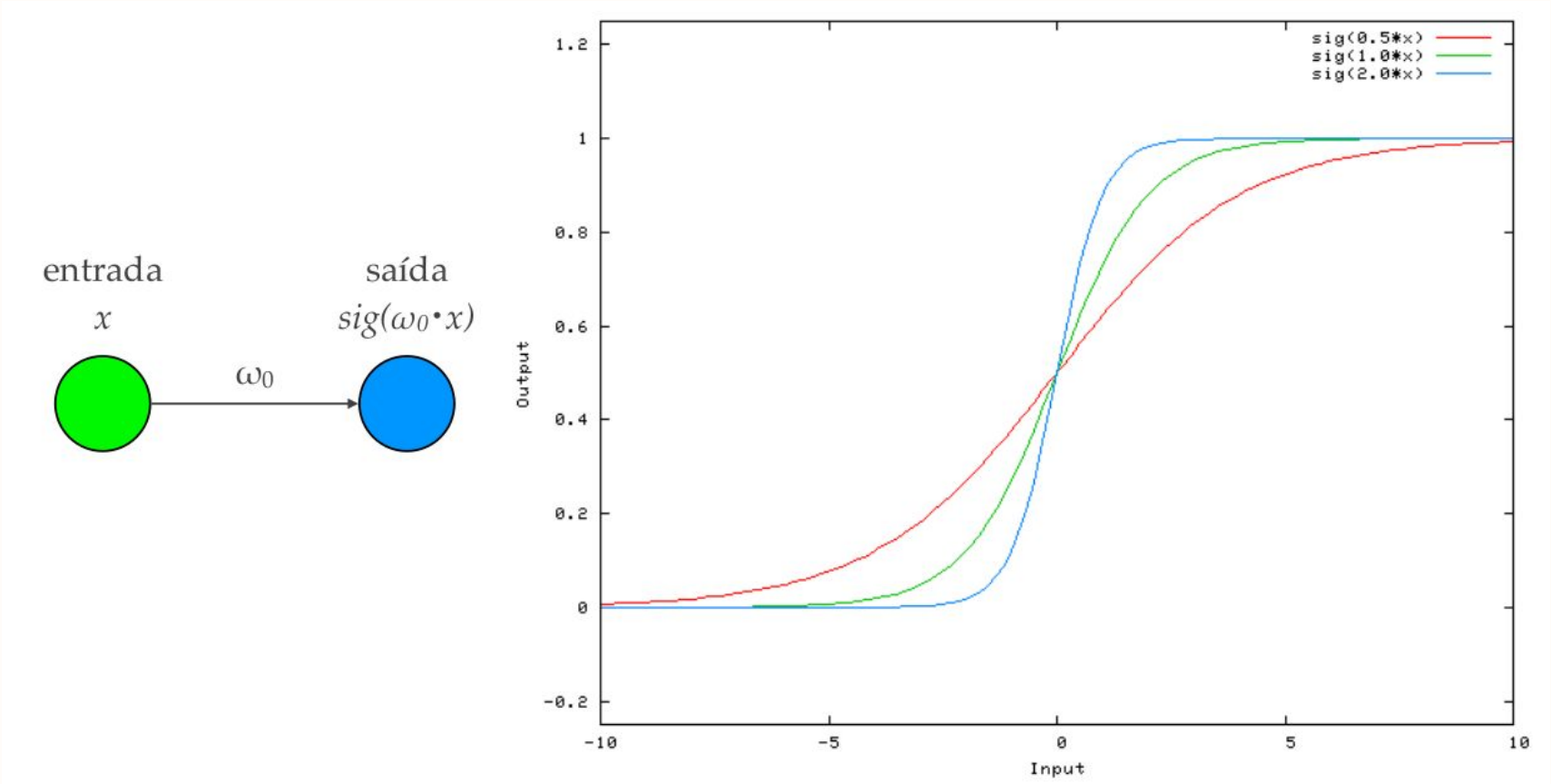
Função do bias

- A não utilização do bias, faz com que apenas o passo da sigmóide mude.



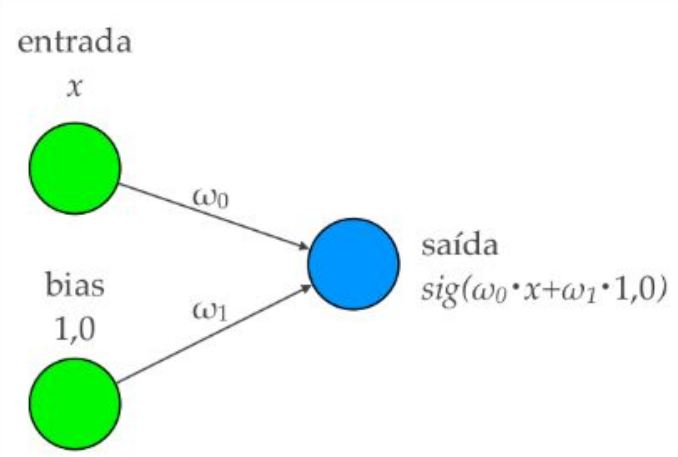
Função do bias

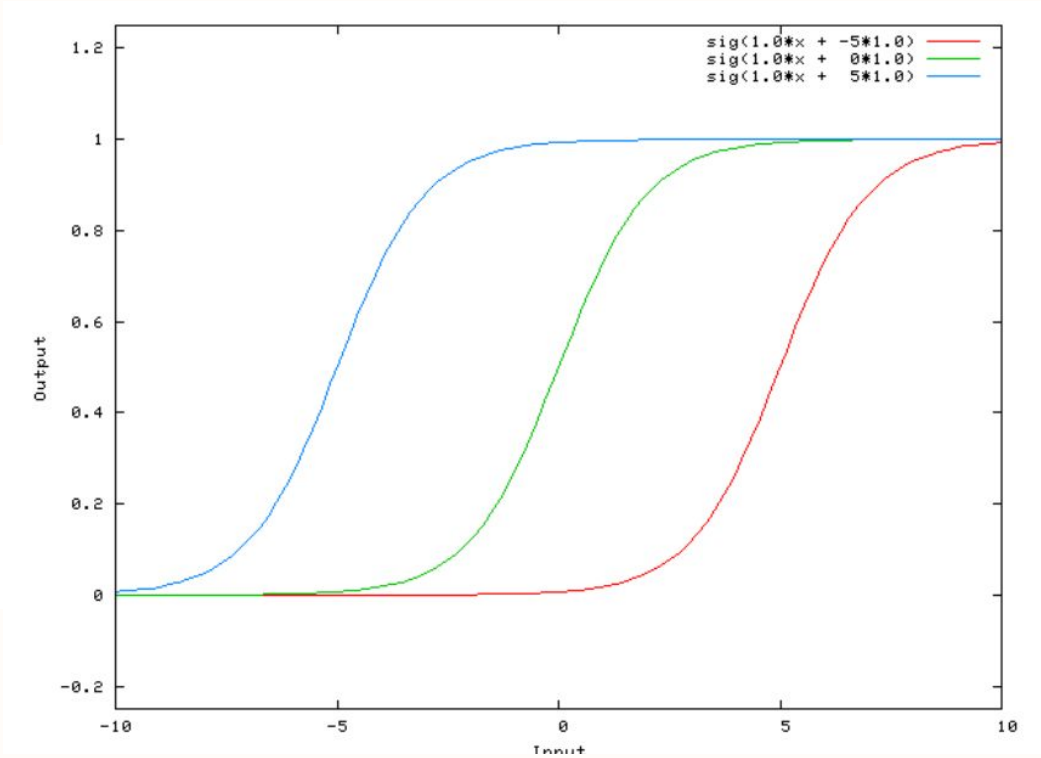
- A não utilização do bias, faz com que apenas o passo da sigmóide mude.



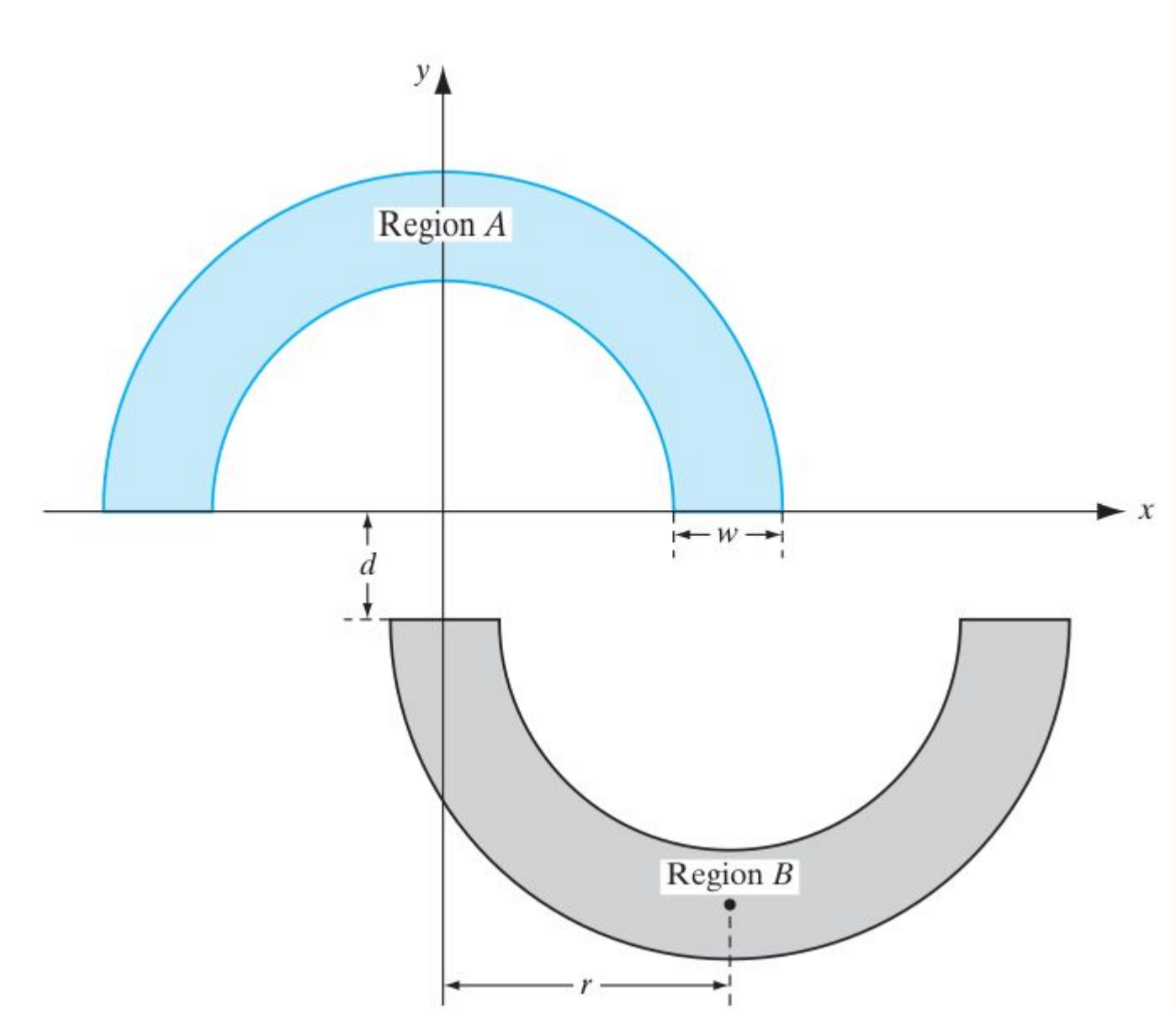
Função do bias

- A não utilização do bias, faz com que apenas o passo da sigmóide mude.



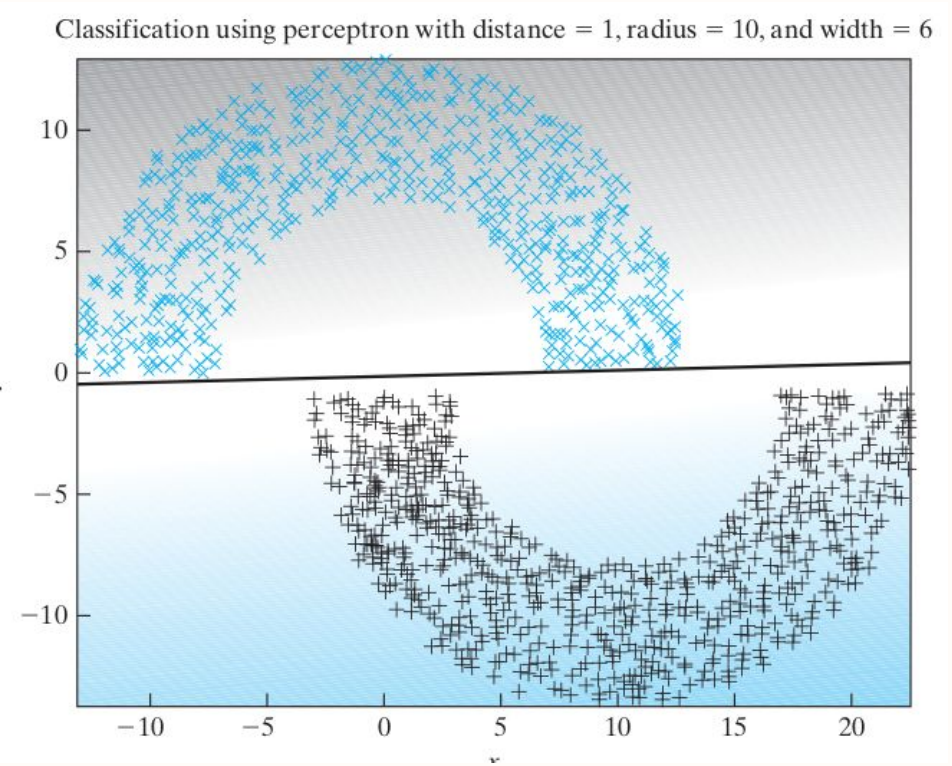


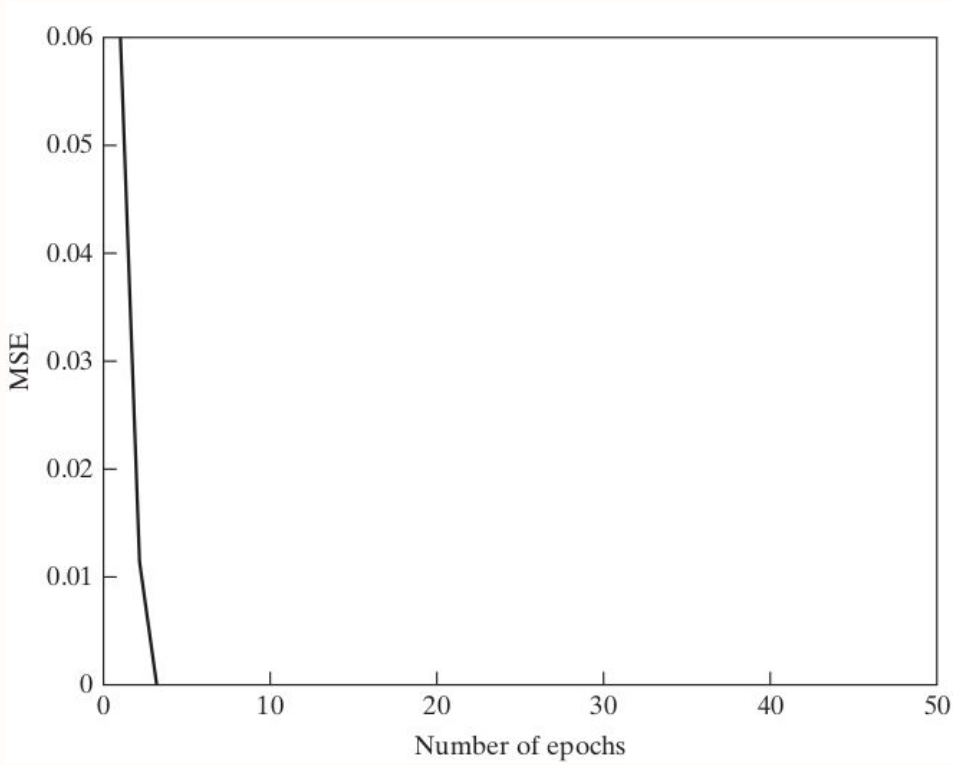
Classificação para Luas Duplas



Simon Haykins (2009)

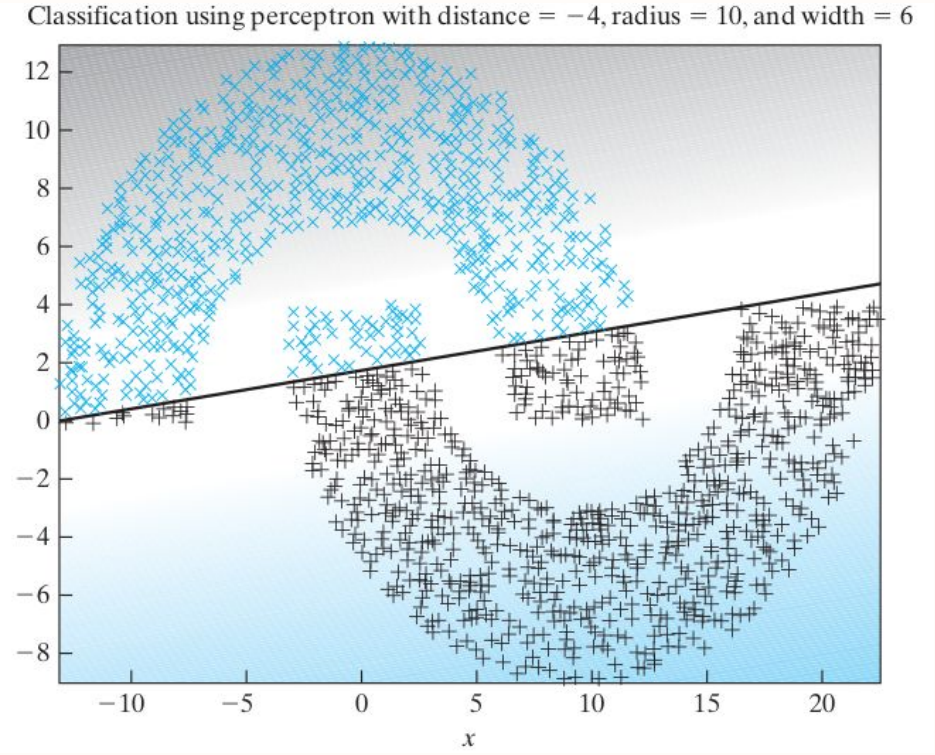
Classificação para Luas Duplas

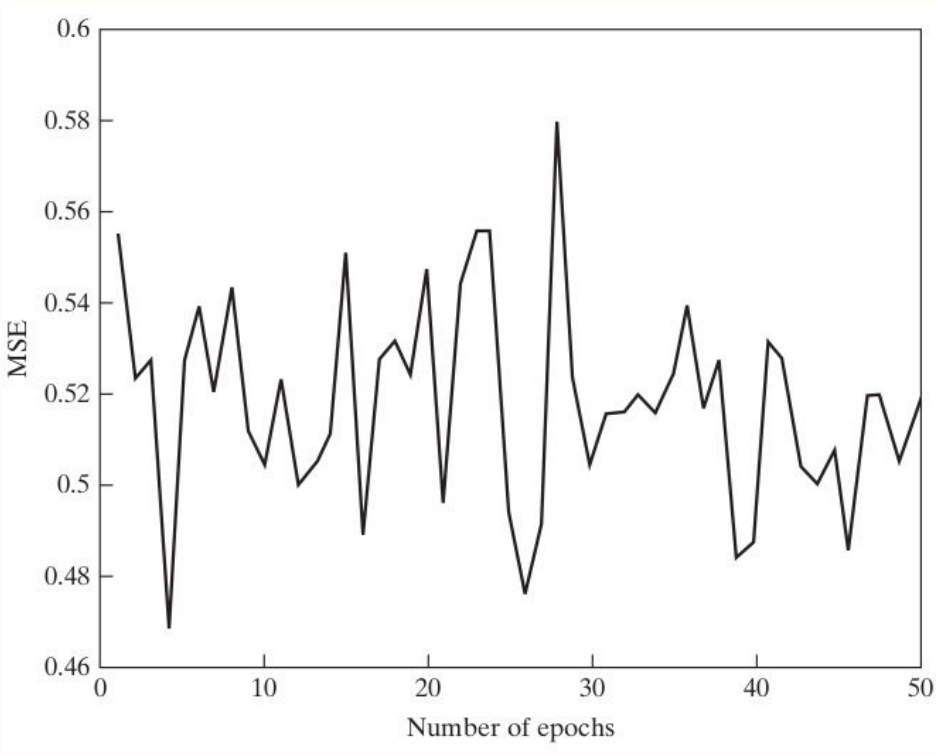




Simon Haykins (2009)

Classificação para Luas Duplas

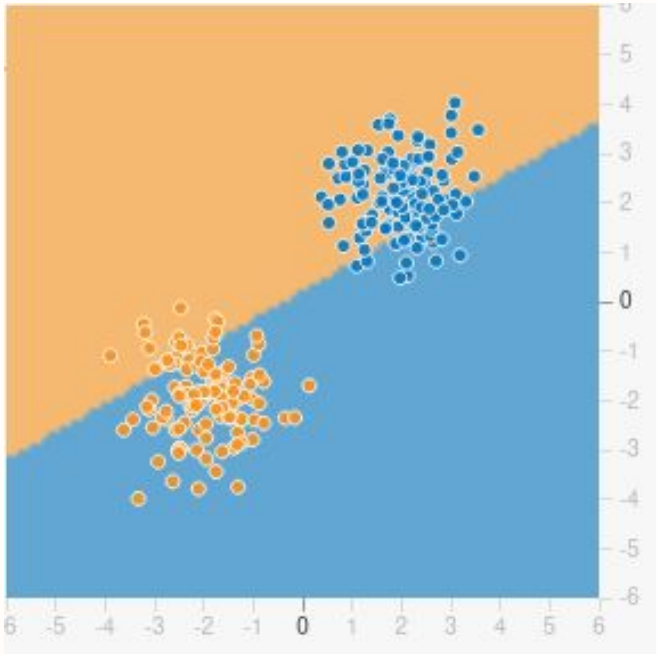




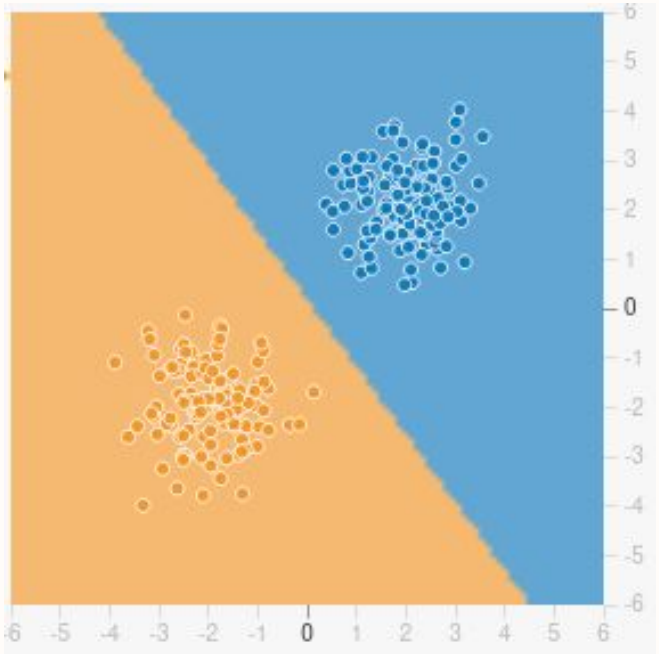
Simon Haykins (2009)

Perceptron - 1 neurônio

Visualmente, pode-se observar uma separação clara entre os dados.

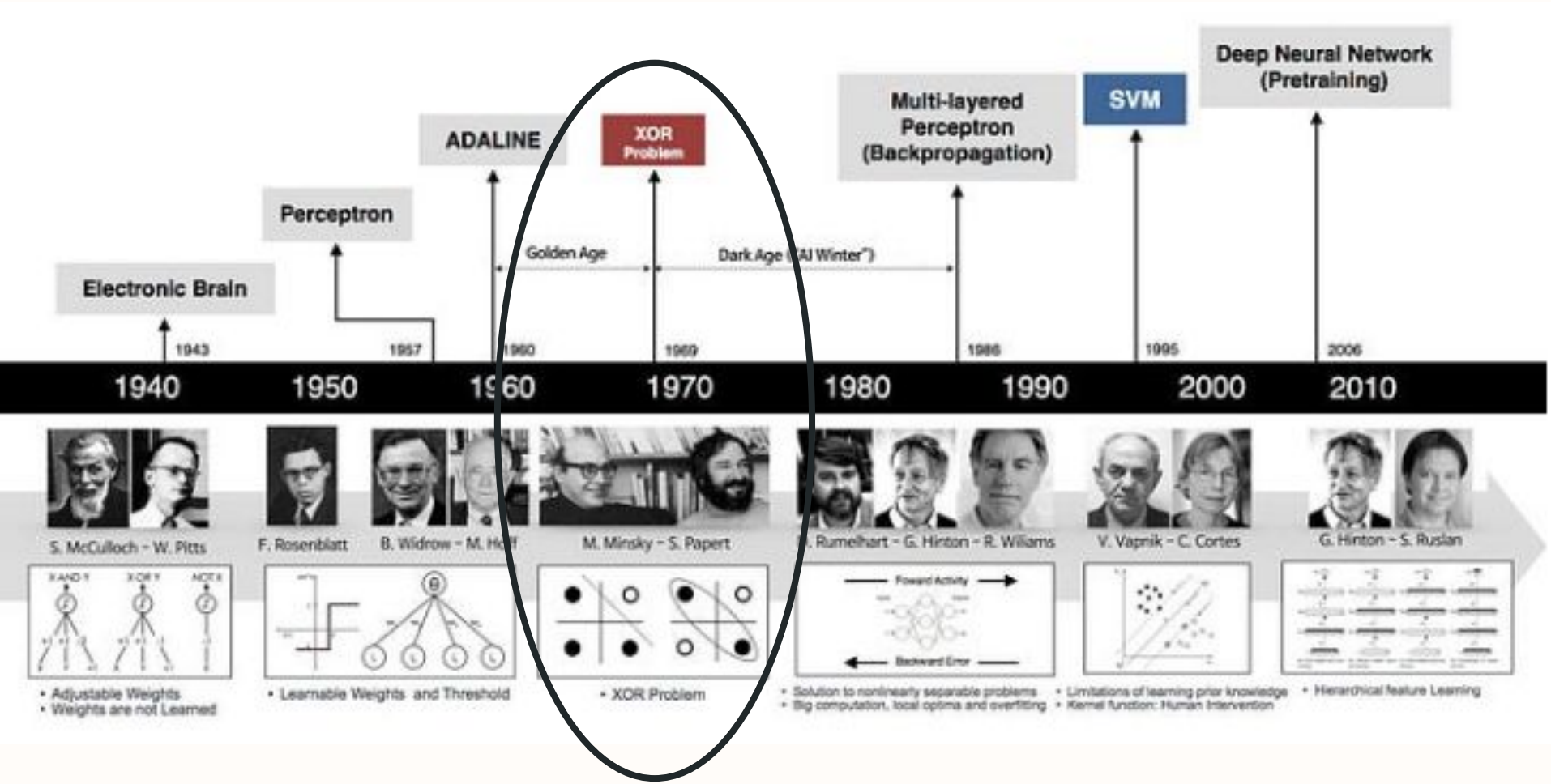


Após algumas iterações, nota-se a separação das regiões de cada classe.



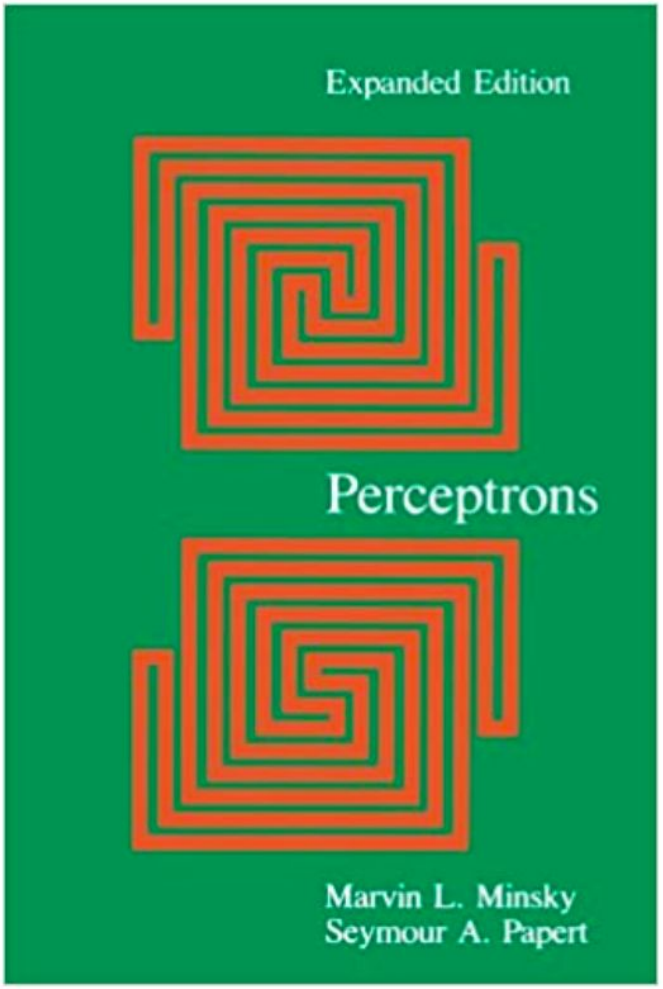
https://playground.tensorflow.org/

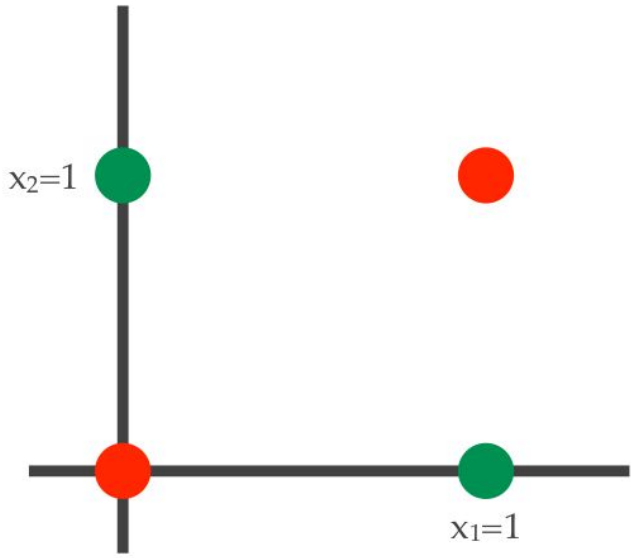
Problema XOR



https://medium.com/@augusto_Pinheiro/redes-neurais-artificiais-133de77c7240

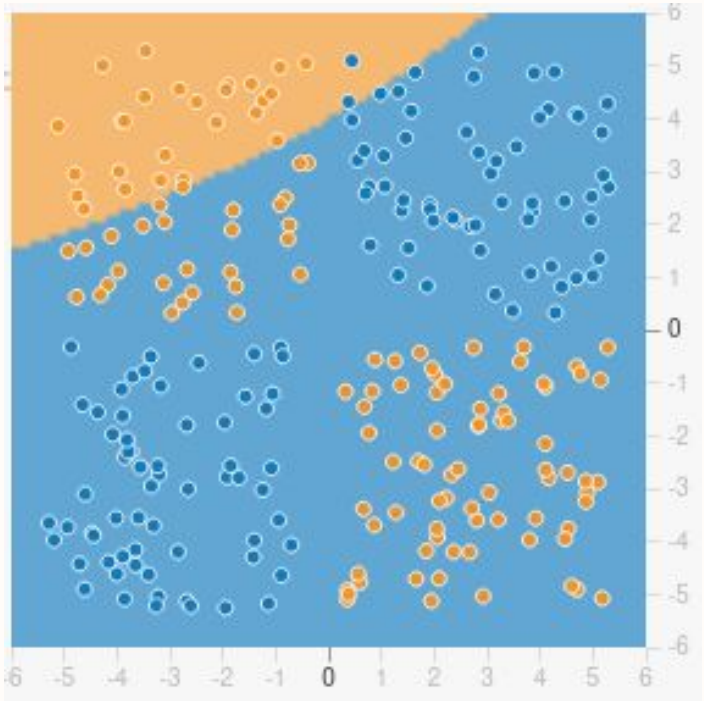
Problema XOR

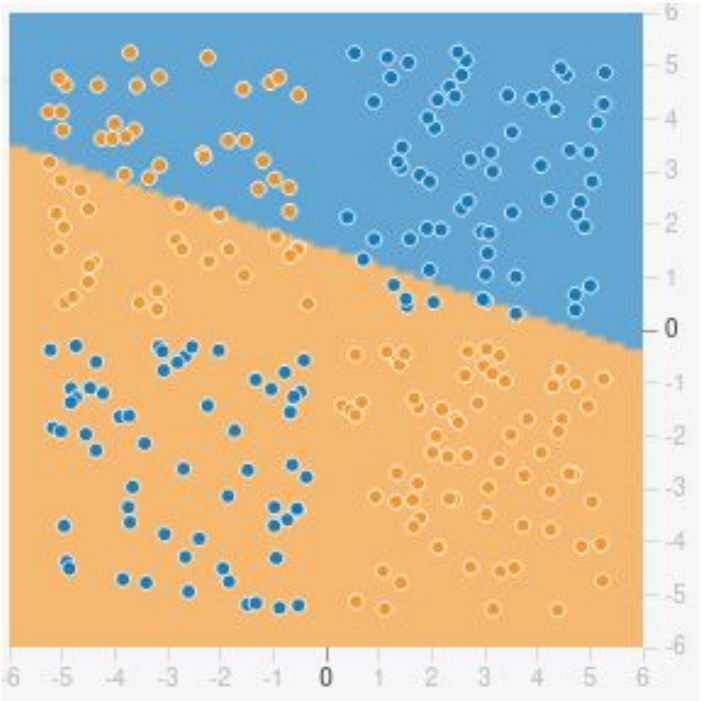




Perceptron - Problema XOR

- Que reta de separação pode ser observada para as duas classes mostradas?





https://playground.tensorflow.org/

Funções de Ativação

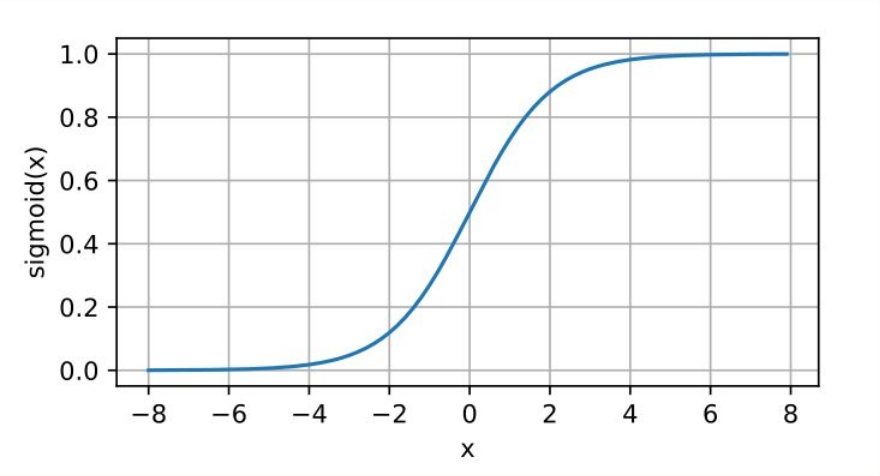
- ➔ Cada perceptron tem como saída um valor que obedece a função linear que o representa
  - ◆ Entretanto, utilizar apenas a combinação linear dos pesos da rede pode limitar a resposta que a arquitetura produz

- ➔ A utilização de funções de ativação têm como objetivo aplicar uma não-linearidade na saída de cada perceptron
  - ◆ suaviza as regiões de fronteiras em cada classe a extração de features que cada neurônio agrega

- ➔ As funções utilizadas devem ser diferenciáveis

Funções de Ativação

- Sigmoid

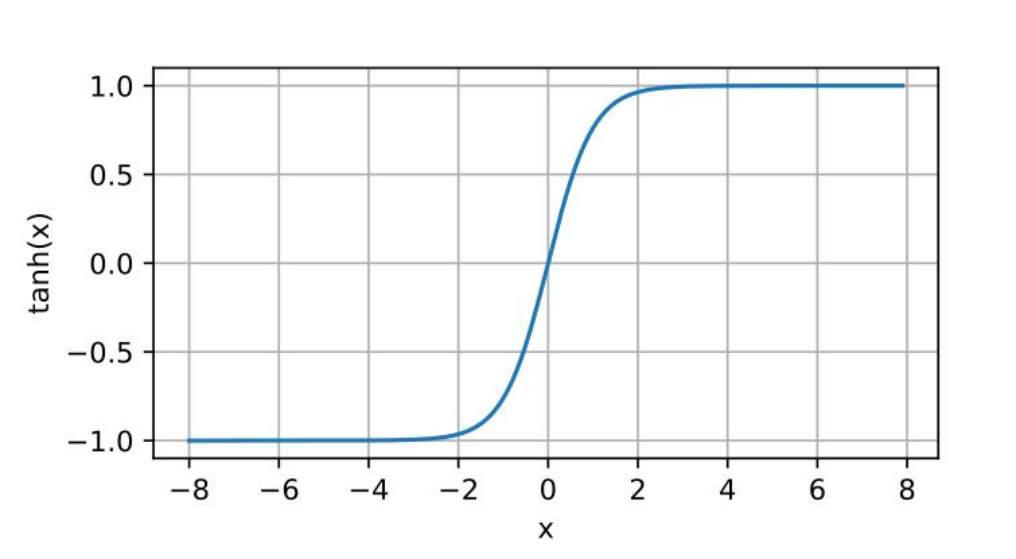


$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{1 + e^x} = 1 - \sigma(-x)$$

$$\frac{d}{dx}\sigma(x) = \sigma(x)(1 - \sigma(x))$$

Funções de Ativação

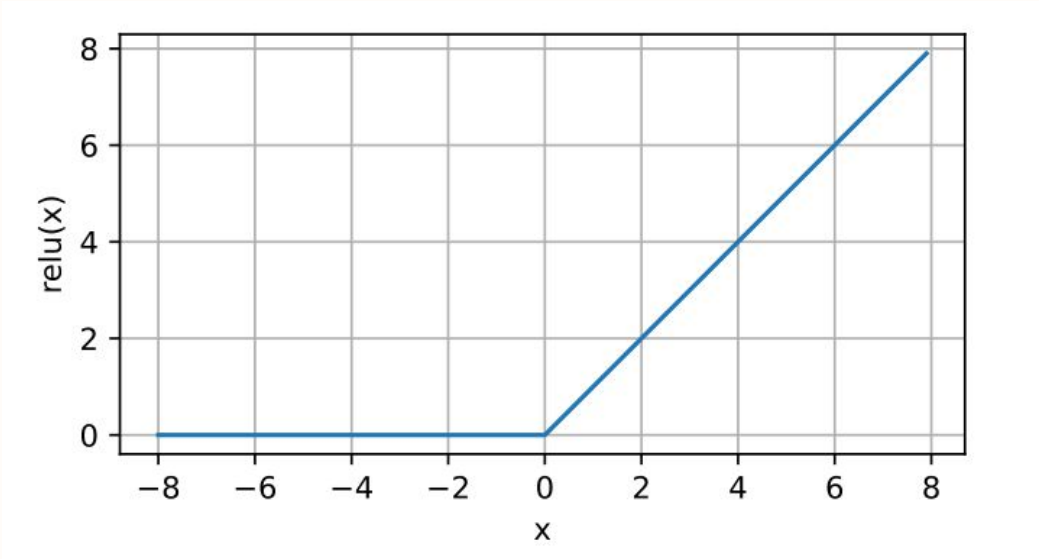
- Tanh



$$\tanh(x)$$

Funções de Ativação

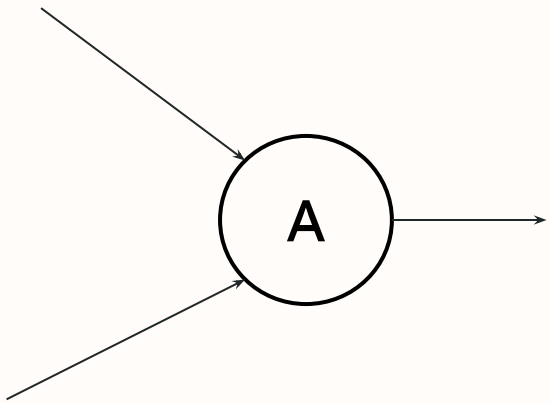
- ReLU

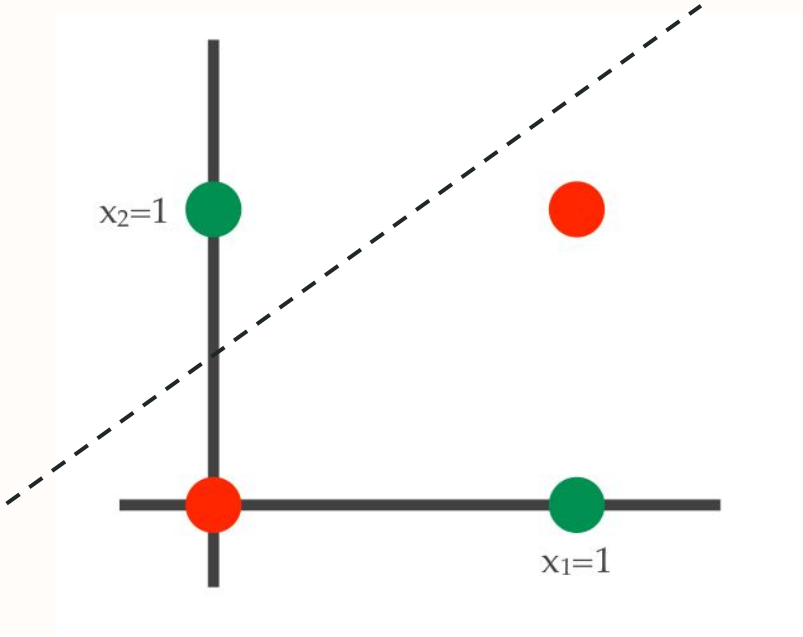


$$\max(0,\ x)$$

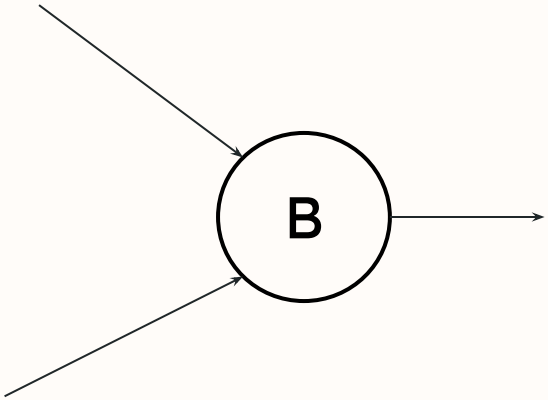
Multilayer Perceptron (MLPs)

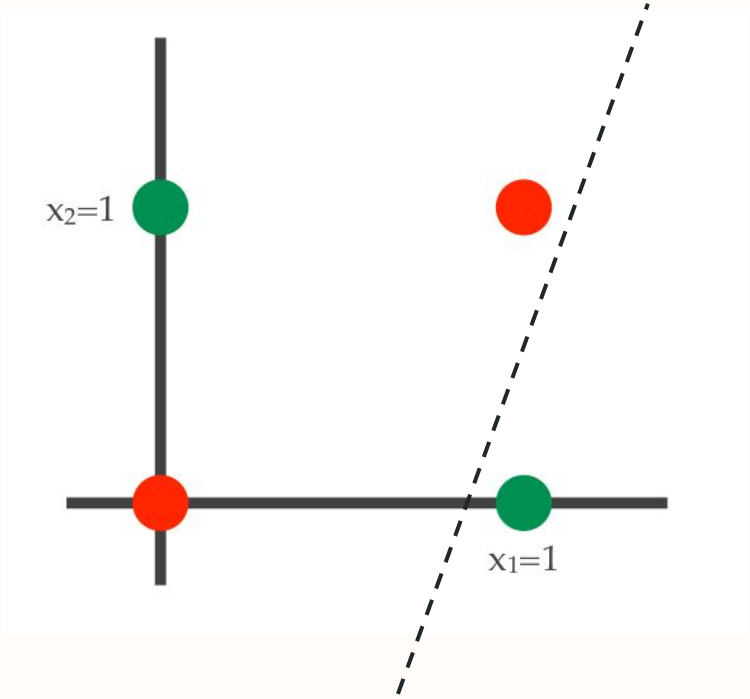
Multilayer Perceptron



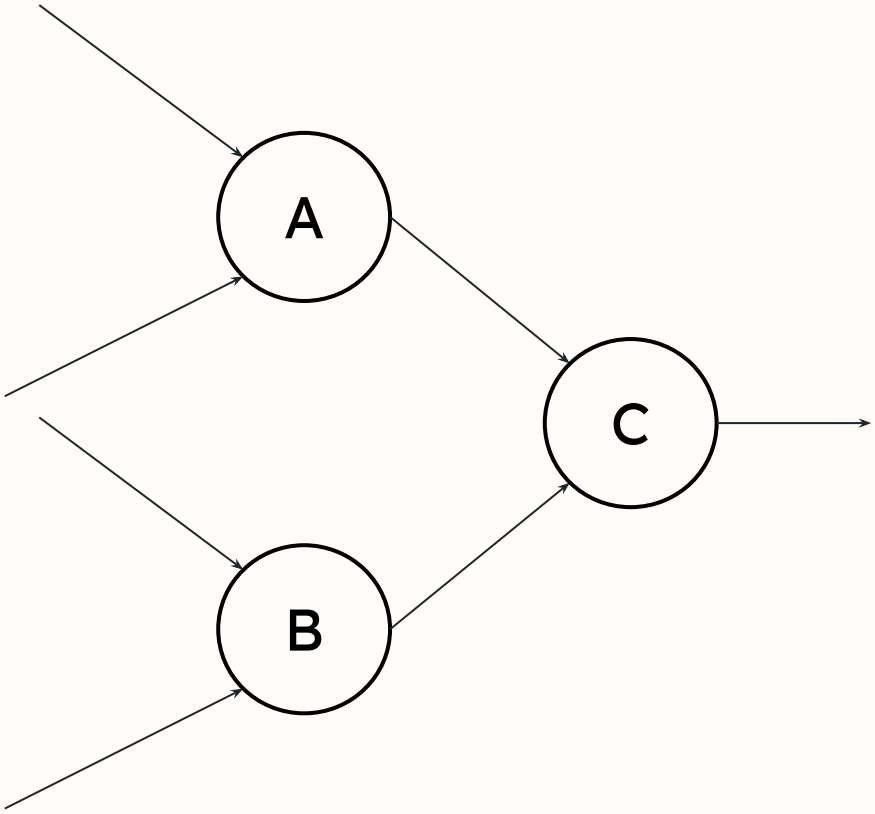


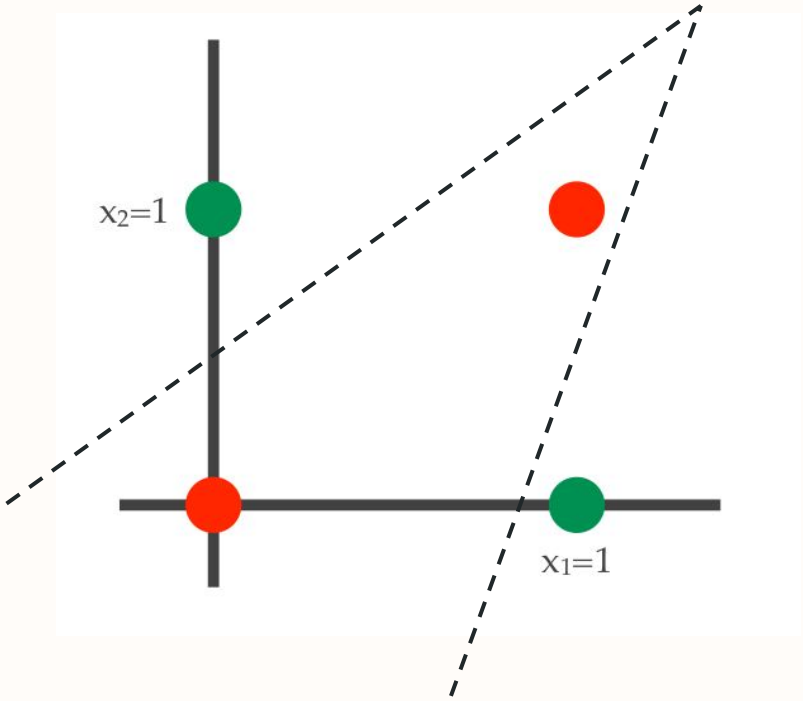
Multilayer Perceptron



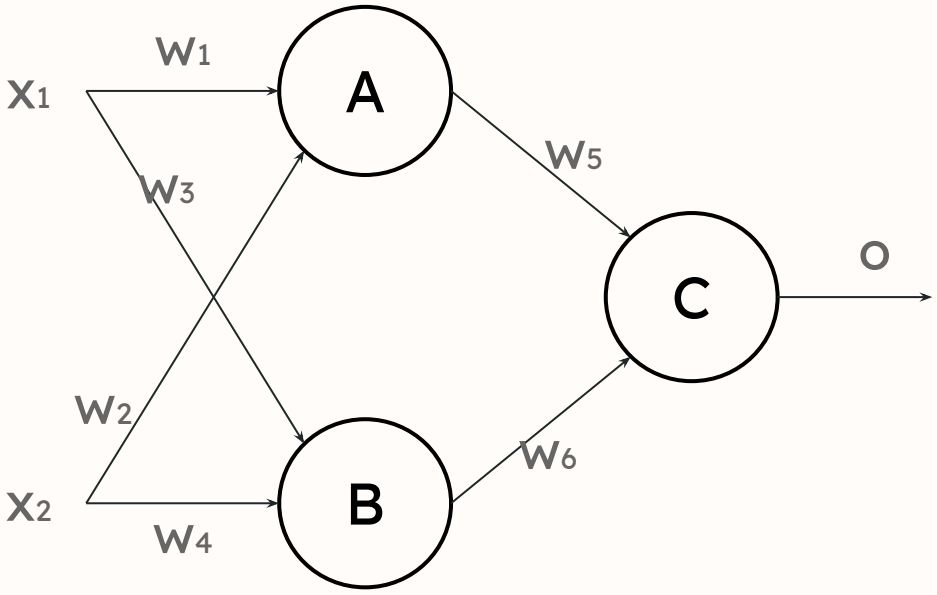


Multilayer Perceptron





Multilayer Perceptron

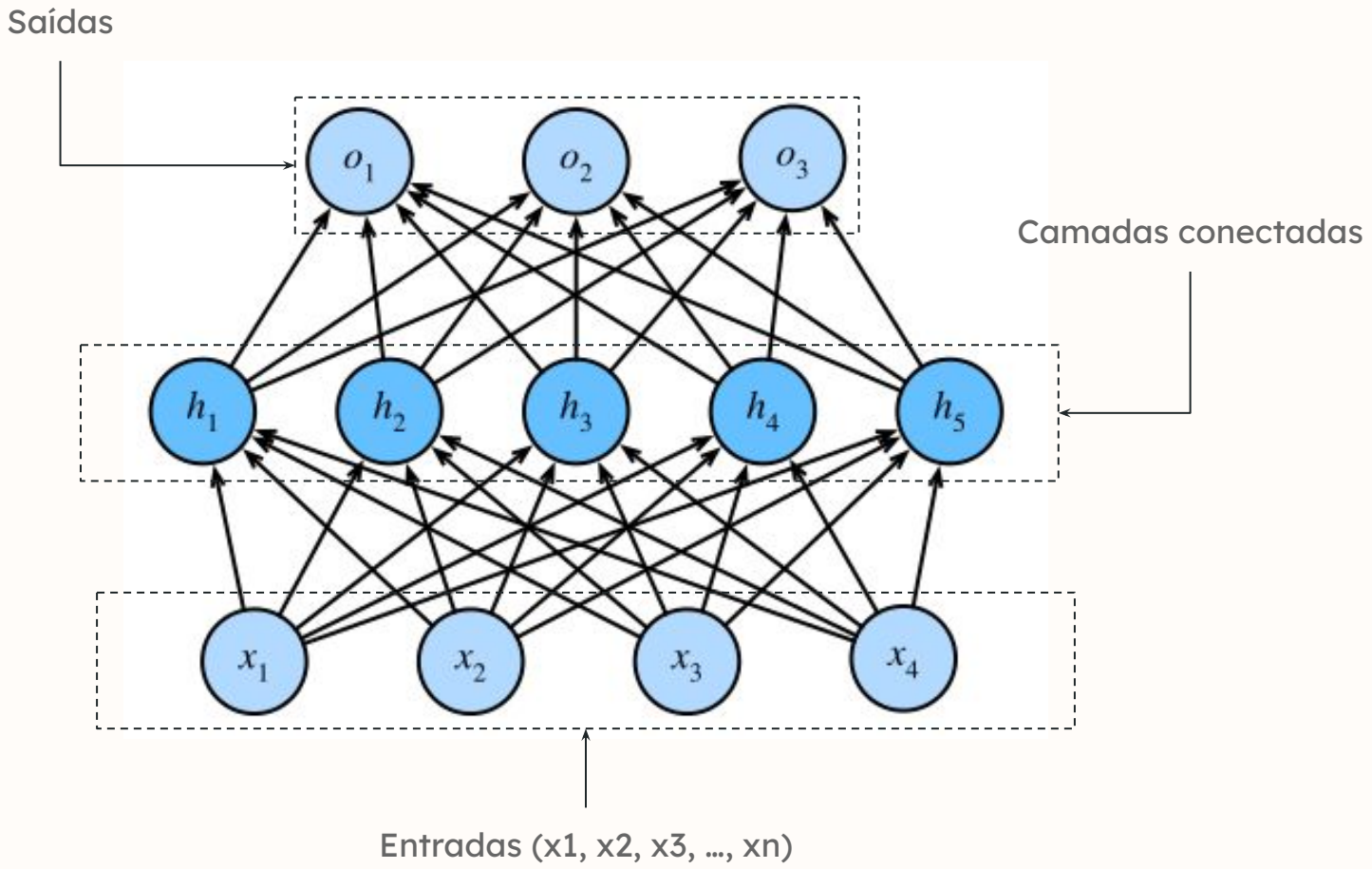


$$A = w_1 x_1 + w_2 x_2 + b_1$$

$$B = w_3 x_1 + w_4 x_2 + b_2$$

$$o = w_5 A + w_6 B + b_3$$

Multilayer Perceptron



Multilayer Perceptron - Parâmetros da arquitetura

1. Vetor de entrada
2. Número de camadas escondidas (Profundidade)
3. Número de neurônios em cada camada escondida (Largura)
4. Vetor de saída (Resposta)

Multilayer Perceptron - Parâmetros da arquitetura

➔ Vetor de entrada (Primeira “camada”)

- Entrada da rede, representado por n valores (número de features / características)

$$X = \{x_1, x_2, x_{3,}, ..., x_n\}, X \in R^n$$

➔ Número de camadas escondidas (Profundidade)

- Representa a profundidade da rede
- Se a rede **R** tem **m** camadas, então o conjunto de camadas L é representado como:

$$L = \{l^1, l^2, l^3, \ldots, l^m\}$$

➔ A entrada X pode ser representada como uma camada quando m=0

Multilayer Perceptron - Parâmetros da arquitetura

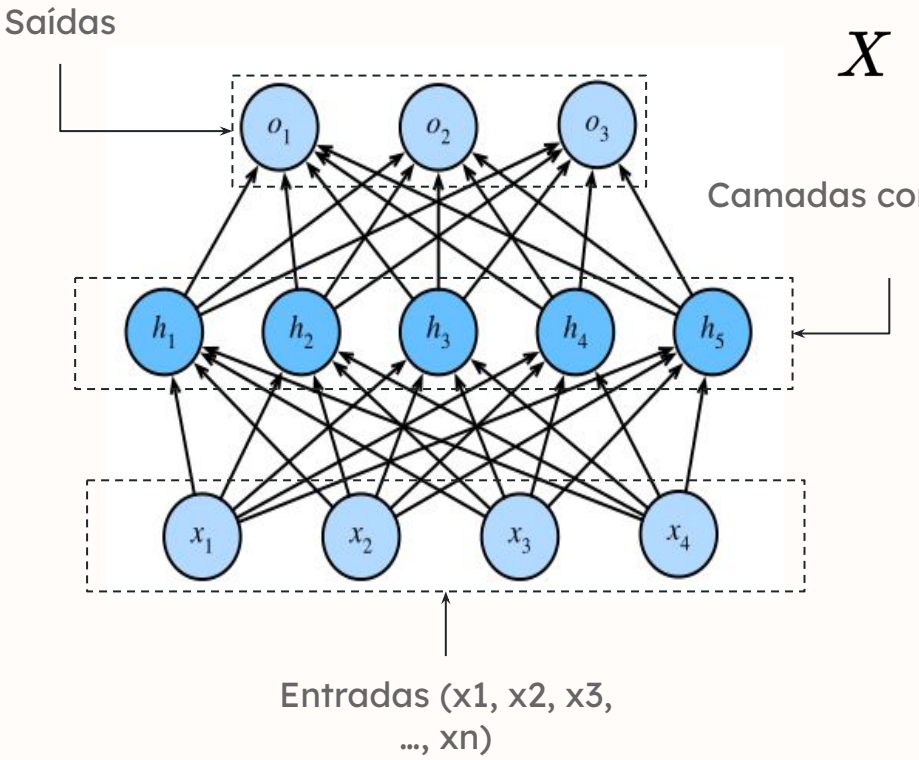
- ➔ Número de neurônios em cada camada escondida (Largura)
  - ◆ Cada camada l, pode ter um conjunto de neurônios próprio, representando a largura da camada

$$l^m = \{l_1^m, l_2^m, l_3^m, \ldots, l_k^m\}, m \in \{1, \ldots, M\}, \forall l \in L$$

- ➔ Vetor de saída (Resposta)
  - ◆ Representa a estimação realizada pela rede

$$O = \{o_1, o_2, \ldots, o_p\}$$

Multilayer Perceptron



$$X = \{x_1, x_2, x_3, x_4\}, X \in R^4$$

$$L = \{l^1\}$$

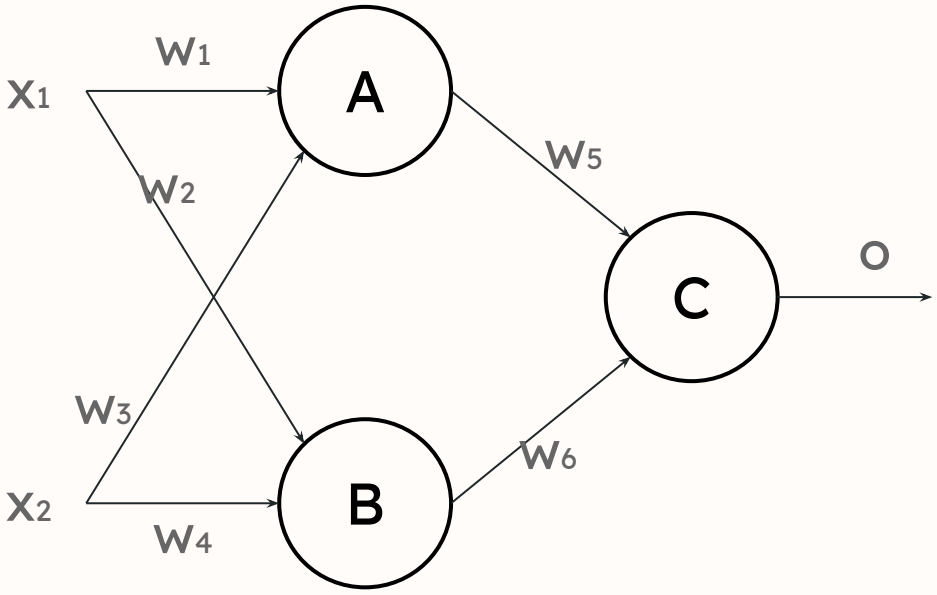
$$l^1 = \{l_1^1, l_2^1, l_3^1, l_4^1, l_5^1\}$$

$$O = \{o_1, o_2, o_3\}$$

Multilayer Perceptron - Exemplo XOR

- ➔ Defina a arquitetura
- ➔ Inicialize os pesos da rede
- ➔ Forneça os vetores de entrada e saída (X e O)
- ➔ Realize o forward e calcula a saída **(o)** para o vetor de entrada **(X)**

Multilayer Perceptron

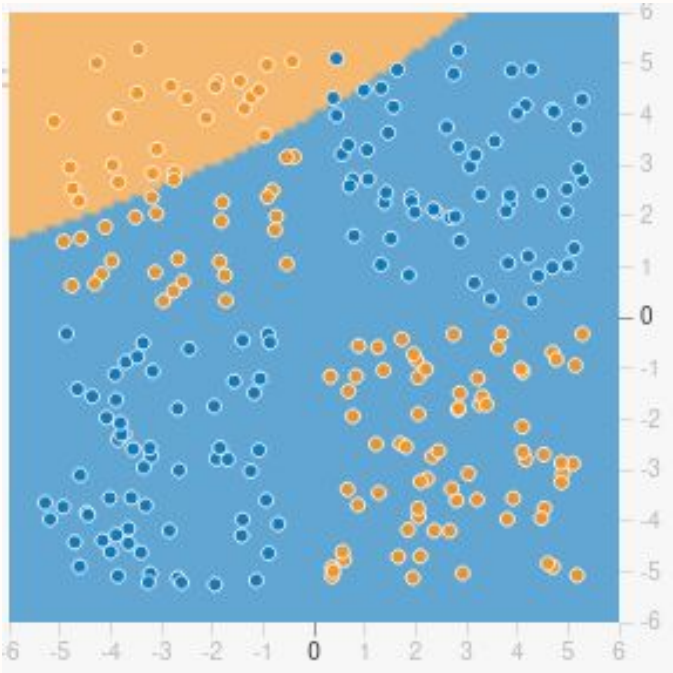


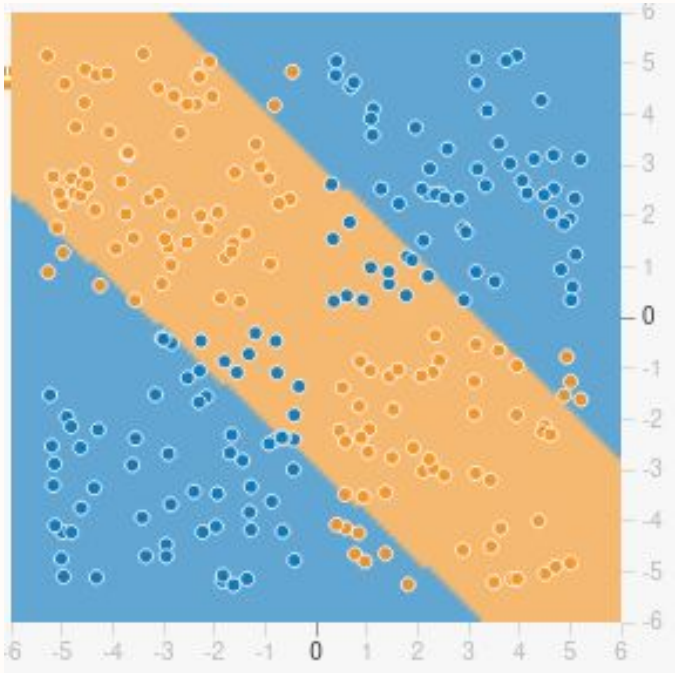
$$A = w_1 x_1 + w_2 x_2 + b_1$$

$$B = w_3 x_1 + w_4 x_2 + b_2$$

$$o = w_5 A + w_6 B + b_3$$

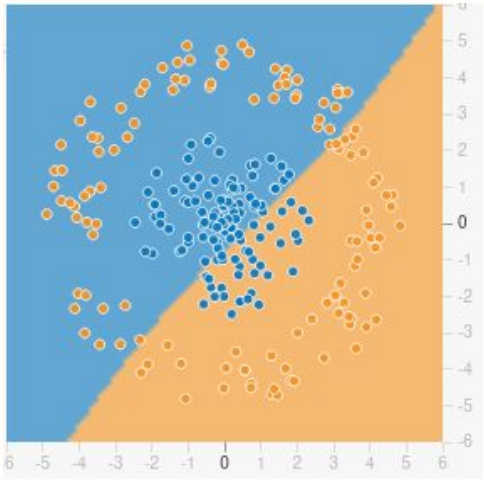
Multilayer Perceptron

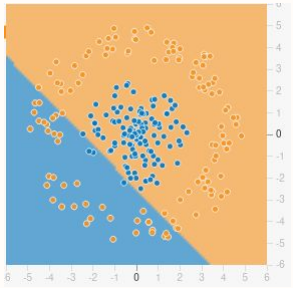


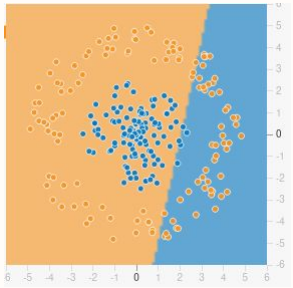


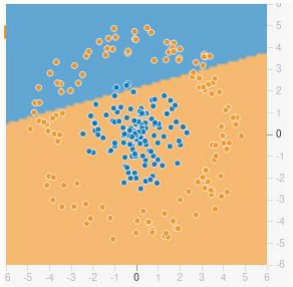
https://playground.tensorflow.org/

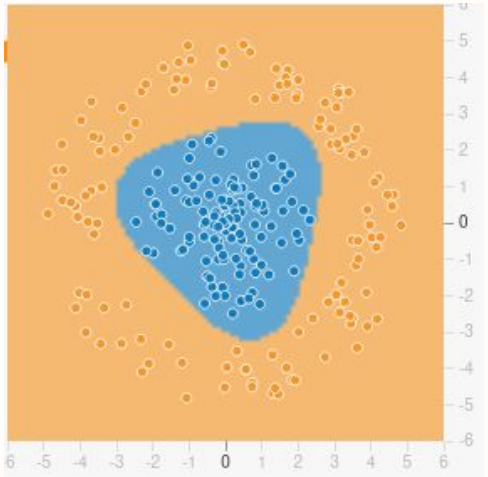
Multilayer Perceptron







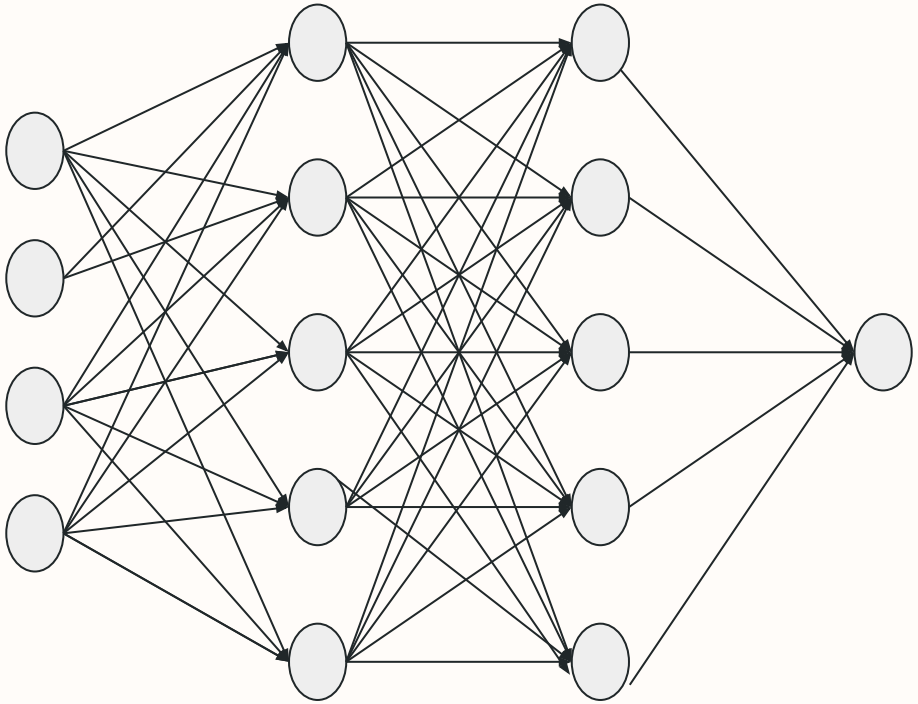




https://playground.tensorflow.org/

Feed Forward Network

- As saídas das primeiras camadas são as entradas das camadas seguintes.
- Os pesos de cada perceptron são ajustados durante o treinamento.

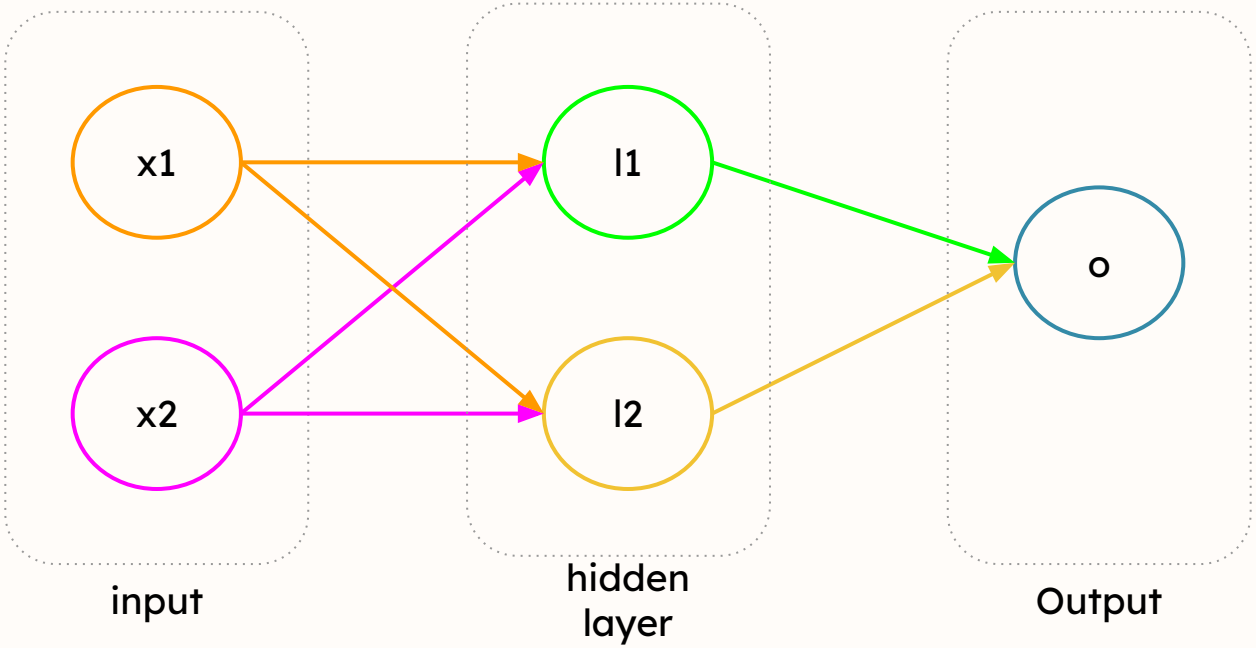


Backpropagation

- ➔ A otimização dos modelos de NN supervisionados é realizada através do **backpropagation**
- ➔ De forma simplificada o backpropagation consiste em realizar 4 passos:

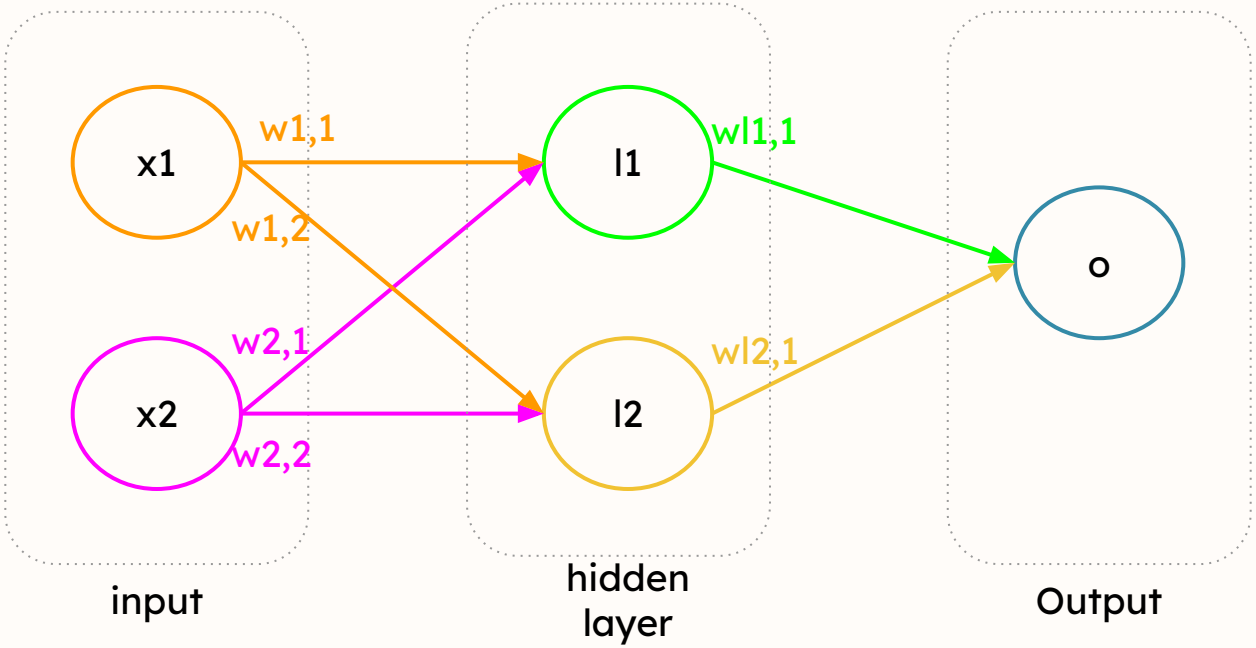
1. **Forward Pass**: Fornecer a entrada ao modelo e computar o resultado através da propagação dos resultados entre as camadas que compõem a rede (e.g., combinações lineares e ativações não lineares)
2. **Cálculo do erro**: Calcular o erro da configuração atual, comparando a distância entre o resultado predito e o esperado (ground truth)
3. **Backward Pass**: Propagar o valor do erro através de toda arquitetura usando o gradiente com respeito aos pesos e biases, percorrendo o caminho inverso ao forward pass
4. **Atualização dos pesos**: Atualizar o valor de cada peso, com base nas informações computadas através do backward pass

Backpropagation



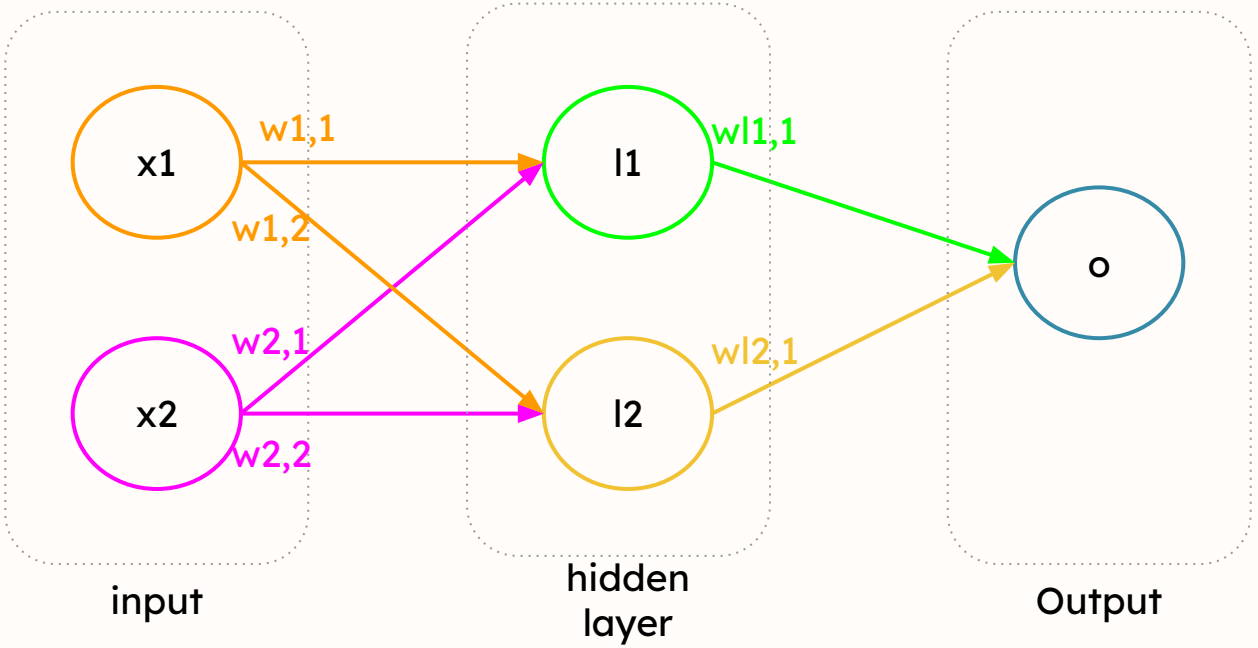
- ➔ Temos uma rede neural multilayer que recebe uma entrada X=(x1,x2) (e.g., 2 features), possui uma camada oculta com 2 neurônios e 1 neurônio de saída
- ➔ Cada feature da entrada é conectada com todos os neurônios da camada posterior; Então, temos um valor w que pondera cada uma destas conexões

Backpropagation



- ➔ Temos uma rede neural multilayer que recebe uma entrada X=(x1,x2) (e.g., 2 features), possui uma camada oculta com 2 neurônios e 1 neurônio de saída
- ➔ Cada feature da entrada é conectada com todos os neurônios da camada posterior; Então, temos um valor w que pondera cada uma destas conexões
- ➔ Da mesma forma, cada unidade da camada oculta tem uma conexão com todos os neurônios da próxima camada

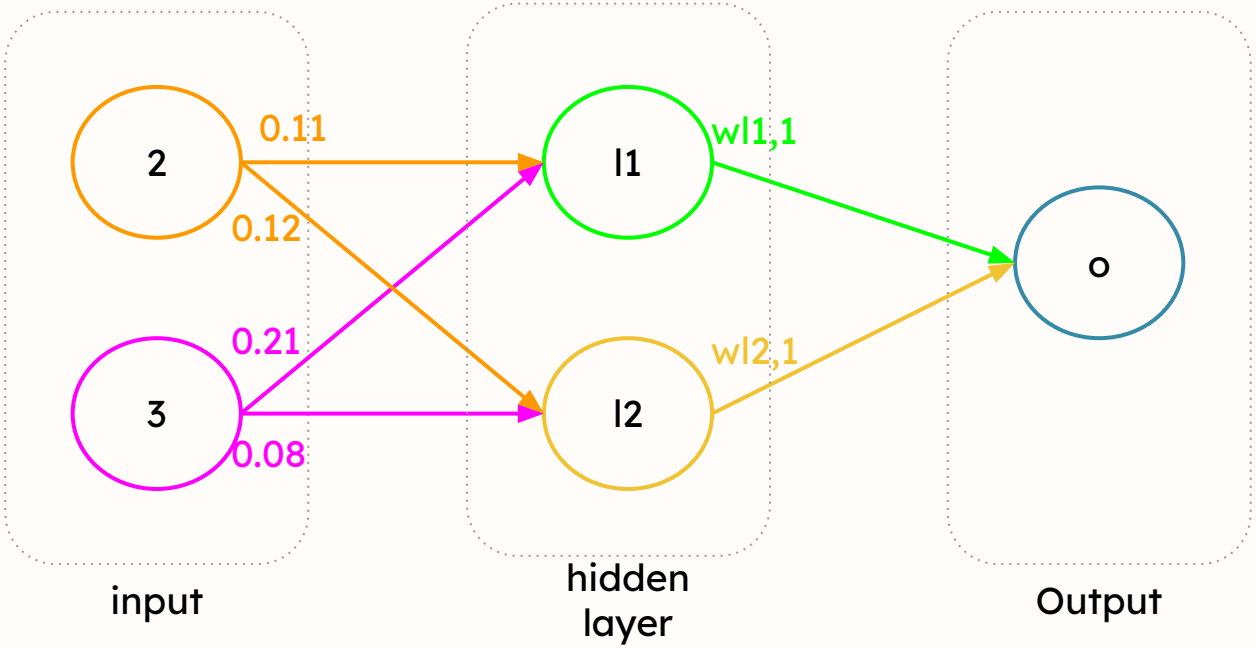
Backpropagation - Forward pass



➔ Como visto anteriormente, realizamos a predição executando o forward pass (vamos desconsiderar o bias como simplificação)

$$l_1^1 = A(x_1 \cdot w_{1,1} + x_2 \cdot w_{2,1}) = z_1^1$$
$$l_2^1 = A(x_1 \cdot w_{1,2} + x_2 \cdot w_{2,2}) = z_2^1$$
$$o = A(z_1^1 \cdot w_{1,1}^1) + A(z_2^1 \cdot w_{2,1}^1) = predi\c{c}ao$$

Backpropagation - Forward pass

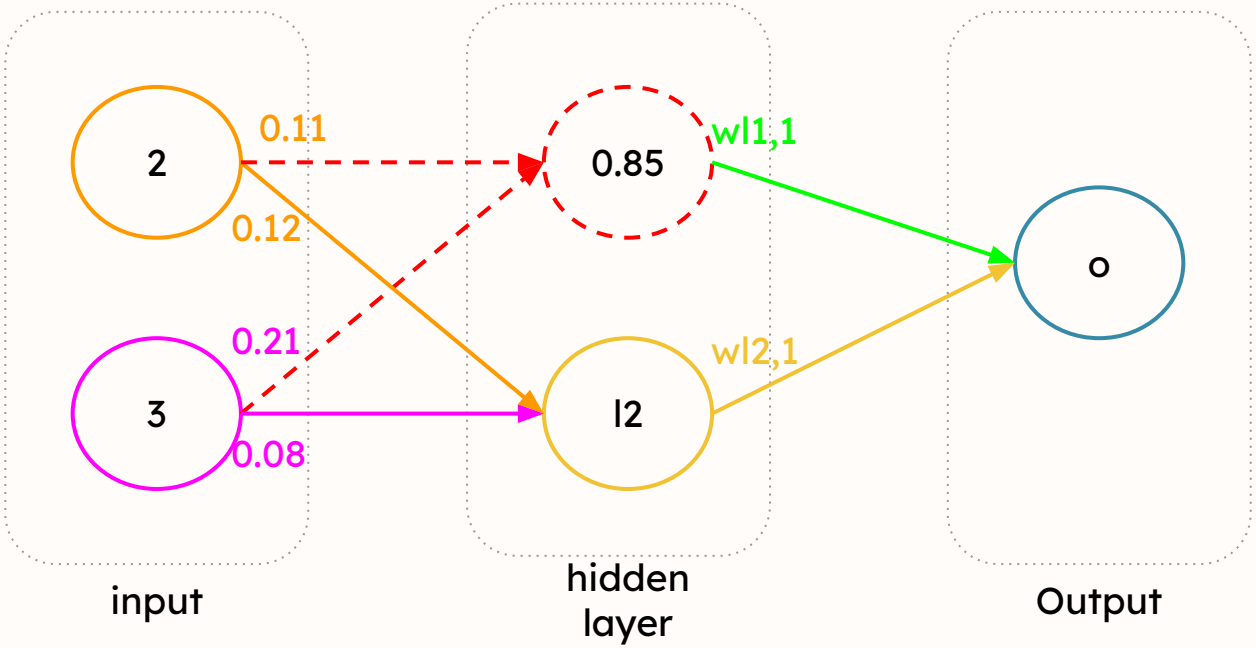


➔ Como visto anteriormente, realizamos a predição executando o forward pass; Definindo a função de ativação A como a função identidade

$$l_1^1 = A(x_1 \cdot w_{1,1} + x_2 \cdot w_{2,1}) = z_1^1$$
$$l_2^1 = A(x_1 \cdot w_{1,2} + x_2 \cdot w_{2,2}) = z_2^1$$
$$o = A(z_1^1 \cdot w_{1,1}^1) + A(z_2^1 \cdot w_{2,1}^1) = prediçao$$

$$l_1^1 = 2 \cdot 0.11 + 3 \cdot 0.21 = 0.85$$
$$l_2^1 = 2 \cdot 0.12 + 3 \cdot 0.08 = 0.48$$

Backpropagation - Forward pass

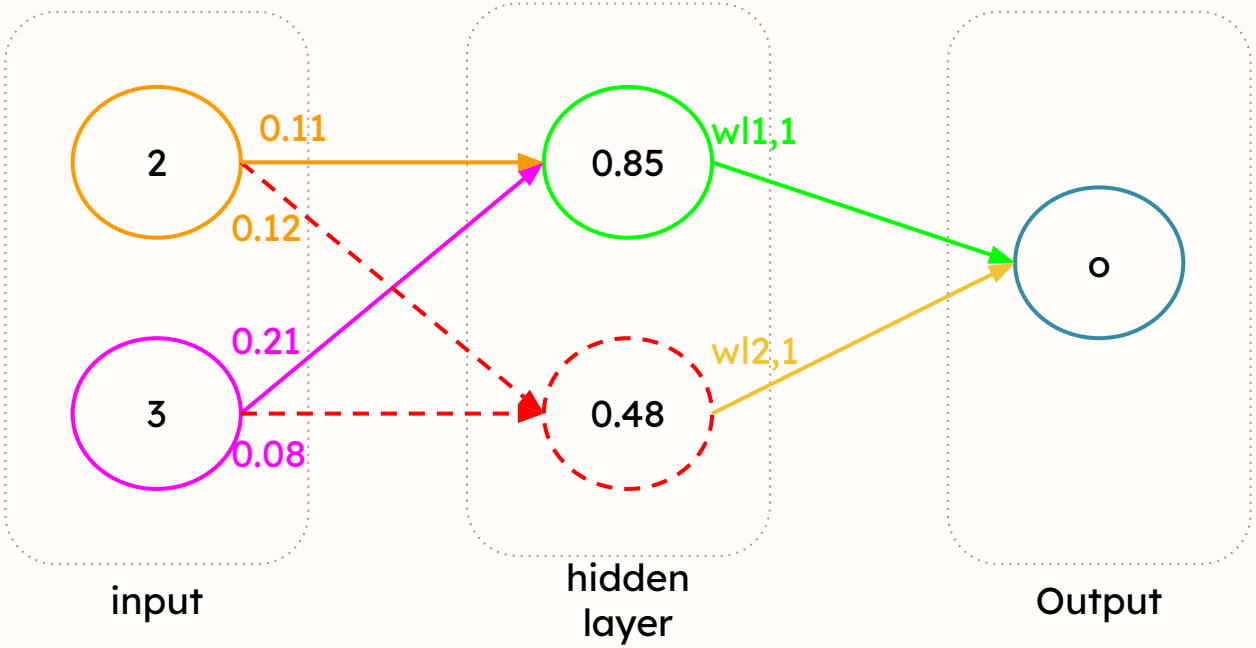


➔ Como visto anteriormente, realizamos a predição executando o forward pass; Definindo a função de ativação A como a função identidade

$$l_1^1 = A(x_1 \cdot w_{1,1} + x_2 \cdot w_{2,1}) = z_1^1$$
$$l_2^1 = A(x_1 \cdot w_{1,2} + x_2 \cdot w_{2,2}) = z_2^1$$
$$o = A(z_1^1 \cdot w_{1,1}^1) + A(z_2^1 \cdot w_{2,1}^1) = prediçao$$

$$l_1^1 = 2 \cdot 0.11 + 3 \cdot 0.21 = 0.85$$
$$l_2^1 = 2 \cdot 0.12 + 3 \cdot 0.08 = 0.48$$

Backpropagation - Forward pass

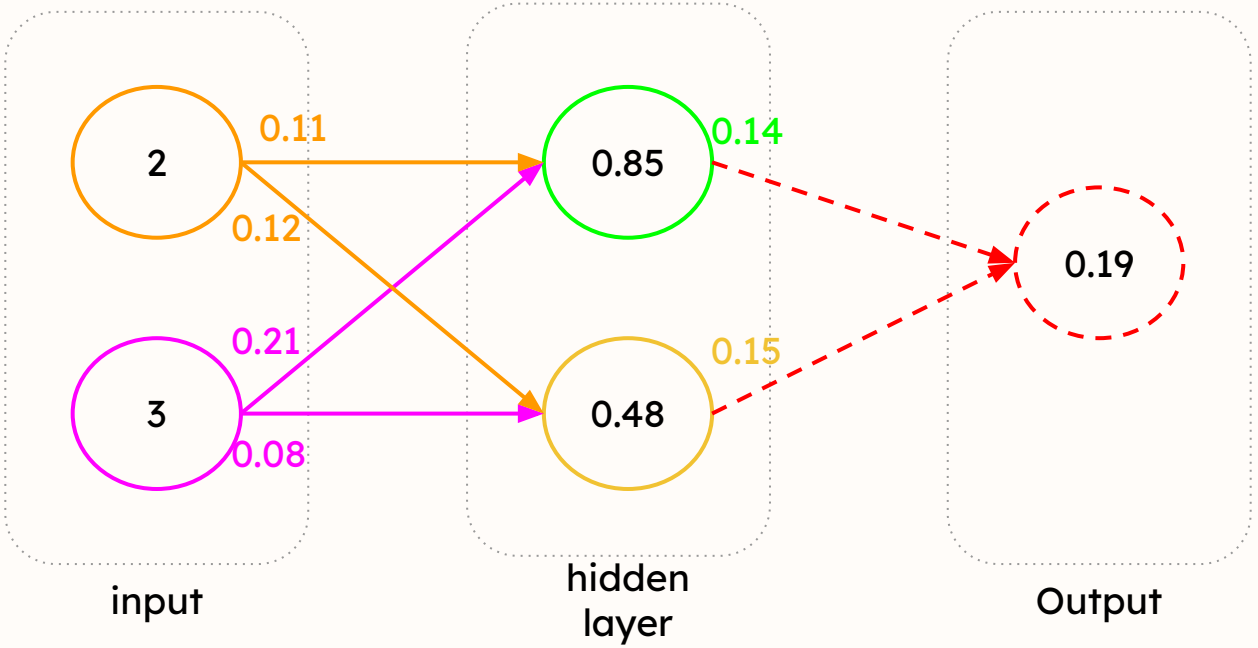


➔ Como visto anteriormente, realizamos a predição executando o forward pass; Definindo a função de ativação A como a função identidade

$$l_1^1 = A(x_1 \cdot w_{1,1} + x_2 \cdot w_{2,1}) = z_1^1$$
$$l_2^1 = A(x_1 \cdot w_{1,2} + x_2 \cdot w_{2,2}) = z_2^1$$
$$o = A(z_1^1 \cdot w_{1,1}^1) + A(z_2^1 \cdot w_{2,1}^1) = prediçao$$

$$l_1^1 = 2 \cdot 0.11 + 3 \cdot 0.21 = 0.85$$
$$l_2^1 = 2 \cdot 0.12 + 3 \cdot 0.08 = 0.48$$

Backpropagation - Forward pass



➔ Como visto anteriormente, realizamos a predição executando o forward pass; Definindo a função de ativação A como a função identidade

$$l_1^1 = A(x_1 \cdot w_{1,1} + x_2 \cdot w_{2,1}) = z_1^1$$
$$l_2^1 = A(x_1 \cdot w_{1,2} + x_2 \cdot w_{2,2}) = z_2^1$$
$$o = A(z_1^1 \cdot w_{1,1}^1) + A(z_2^1 \cdot w_{2,1}^1) = prediçao$$

$$l_1^1 = 2 \cdot 0.11 + 3 \cdot 0.21 = 0.85$$
$$l_2^1 = 2 \cdot 0.12 + 3 \cdot 0.08 = 0.48$$
$$o = 0.85 \cdot 0.14 + 0.48 \cdot 0.15 = 0.191$$

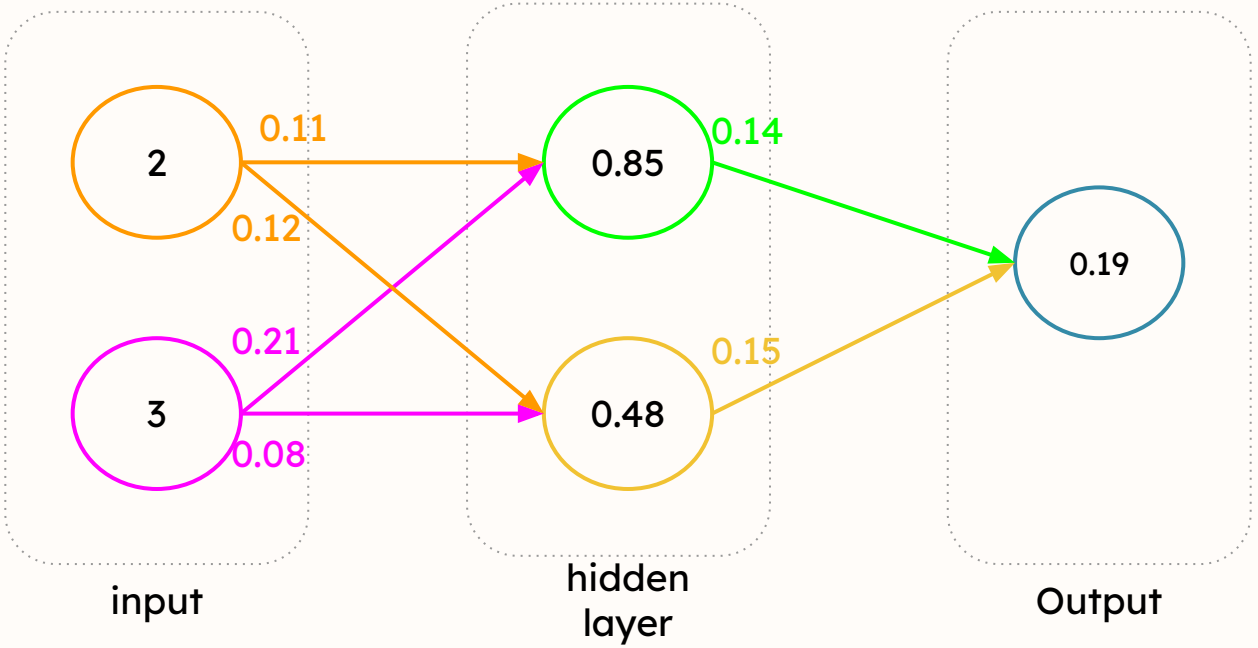
Backpropagation - Forward pass

$$
\begin{aligned}
l_1^1 &= A(x_1 \cdot w_{1,1} + x_2 \cdot w_{2,1}) = z_1^1 \\
l_2^1 &= A(x_1 \cdot w_{1,2} + x_2 \cdot w_{2,2}) = z_2^1 \\
o &= A\left(z_1^1 \cdot w_{1,1}^1\right) + A\left(z_2^1 \cdot w_{2,1}^1\right) = \textit{predi\c{c}ao}
\end{aligned}
\qquad \cdots\cdots\blacktriangleright \qquad
\begin{aligned}
l_1^1 &= 2 \cdot 0.11 + 3 \cdot 0.21 = 0.85 \\
l_2^1 &= 2 \cdot 0.12 + 3 \cdot 0.08 = 0.48 \\
o &= 0.85 \cdot 0.14 + 0.48 \cdot 0.15 = 0.191
\end{aligned}
$$

$$
\begin{bmatrix} x_1 & x_2 \end{bmatrix} \cdot \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} = \begin{bmatrix} l_1^1 & l_2^1 \end{bmatrix} \implies \begin{bmatrix} l_1^1 & l_2^1 \end{bmatrix} \cdot \begin{bmatrix} w_{1,1}^1 \\ w_{2,1}^1 \end{bmatrix} = o
$$

$$
\begin{bmatrix} 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} 0.11 & 0.12 \\ 0.21 & 0.08 \end{bmatrix} = \begin{bmatrix} 0.85 & 0.48 \end{bmatrix} \implies \begin{bmatrix} 0.85 & 0.48 \end{bmatrix} \cdot \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} = 0.191
$$

Backpropagation - Forward pass



- ➔ Nossa predição é dada pela saída da última unidade da rede (o)
- ➔ No aprendizado supervisionado, realizamos o treinamentos a partir de um conjunto de pares entrada-saída (e.g., {X,y})
- ➔ Medimos o erro de predição computando a distância da estimativa de o para a entrada yi, dado uma entrada Xi

$$Erro = \frac{1}{2}(o - y)^2$$

Backpropagation - Ajustando o erro

➔ Como corrigir a saída se o valor de ‘o’ for muito diferente de ‘y’?
- Valores da entrada X e saída y são fixos (constantes)
- Os valores que podem ser alterados são dados pelos parâmetros de cada neurônio (pesos W*)

Backpropagation - Ajustando o erro

- ➔ Como corrigir a saída se o é muito diferente de y?
  - ◆ Valores da entrada X e saída y são fixos (constantes)
  - ◆ Os valores que podem ser alterados são dados pelos parâmetros de cada neurônio (pesos da rede (W*))
- ➔ Os valores de cada ativação da rede dependem do valor definido para cada peso
  - ◆ Para corrigir as predições realizadas precisamos ajustar os valores em W

Backpropagation - Ajustando o erro

➔ Como atualizar o valor de cada w para ter uma boa predição?

- ◆ Sabemos o valor do erro através da predição realizada na última camada da rede neural
- ◆ Pela arquitetura da rede, vimos que todas as camadas possuem um ‘vínculo’, através das conexões ponderadas por cada peso

$$Erro = \frac{1}{2}(o - y)^2$$

➔ Se y = 1, então:

$$Erro = \frac{1}{2}(0.191 - 1)^2 = 0.327$$

Backpropagation - Ajustando o erro

➔ Como atualizar o valor de cada peso w para ter uma boa predição?

- Sabemos o valor do erro através da predição realizada na última camada da rede neural
- Pela arquitetura da rede, vimos que todas as camadas possuem um 'vínculo', através das conexões anteriores
- Como cada camada é indiretamente conectada, podemos propagar o erro percorrendo o caminho inverso ao realizado no forward

$$Erro = \frac{1}{2}(o - y)^2$$

$$l_1^1 = A(x_1 \cdot w_{1,1} + x_2 \cdot w_{2,1}) = z_1^1$$
$$l_2^1 = A(x_1 \cdot w_{1,2} + x_2 \cdot w_{2,2}) = z_2^1$$
$$o = A(z_1^1 \cdot w_{1,1}^1) + A(z_2^1 \cdot w_{2,1}^1) = predi\text{ç}ao$$

Backpropagation - Ajustando o erro

- ➔ Chamamos de backward o passo reverso realizado para atualizar os pesos da rede
- ➔ Aplicamos o Gradient Descent, algoritmo de otimização capaz de encontrar o valor mínimo de uma função
  - ◆ O gradient descent atualiza o peso iterativamente decrementando o valor atual do peso com base no valor da derivada

Learning rate

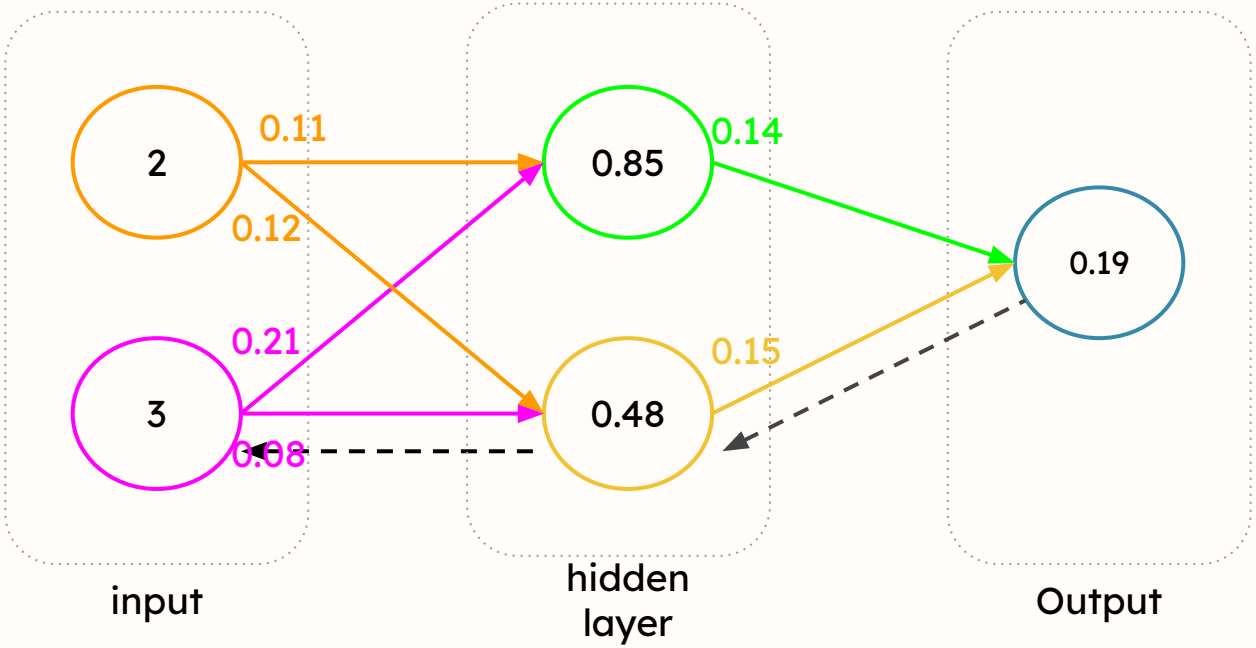
Novo peso

$$W_n' = W_n - a\left(\frac{\partial Erro}{\partial W_n}\right)$$

Peso atual

Derivada do erro com respeito ao peso W

Backpropagation - Ajustando o erro



Backpropagation - Ajustando o erro

$$W_n = W_n - a\left(\frac{\partial Erro}{\partial W_n}\right) \qquad Erro = \frac{1}{2}(o - y)^2$$

$$l_1^1 = A(x_1 \cdot w_{1,1} + x_2 \cdot w_{2,1}) = z_1^1$$
$$l_2^1 = A(x_1 \cdot w_{1,2} + x_2 \cdot w_{2,2}) = z_2^1$$
$$o = A(z_1^1 \cdot w_{1,1}^1) + A(z_2^1 \cdot w_{2,1}^1)$$

$$\frac{\partial Erro}{\partial w_2^1} = \frac{\partial Erro}{\partial o} * \frac{\partial o}{\partial A}^{1} * \frac{\partial A}{\partial w_2^1} \longrightarrow \frac{\partial Erro}{\partial w_2^1} = \frac{\frac{1}{2}(o-y)^2}{\partial o} * \frac{\partial\left(z_1^1 \cdot w_{1,1}^1 + z_2^1 \cdot w_{2,1}^1\right)}{\partial w_2^1}$$

Regra da cadeia

Constante

$$\frac{\partial Erro}{\partial w_2^1} = 2\frac{1}{2}(o-y) * \frac{\partial(o-y)}{\partial o} * (z_2^1) \longrightarrow \frac{\partial Erro}{\partial w_2^1} = 2\frac{1}{2}(o-y) * \frac{\partial(o-y)}{\partial o} * (x_2 \cdot w_{2,2} + x_1 \cdot w_{1,2})$$

$$\frac{\partial Erro}{\partial w_2^1} = (o-y) * (z_2^1) = \Delta h_2^1 \longrightarrow W_2^1 = W_2^1 - a\Delta h_2^1$$

Backpropagation - Ajustando o erro

➔ Da mesma forma, podemos replicar esses passos para todos os pesos da rede

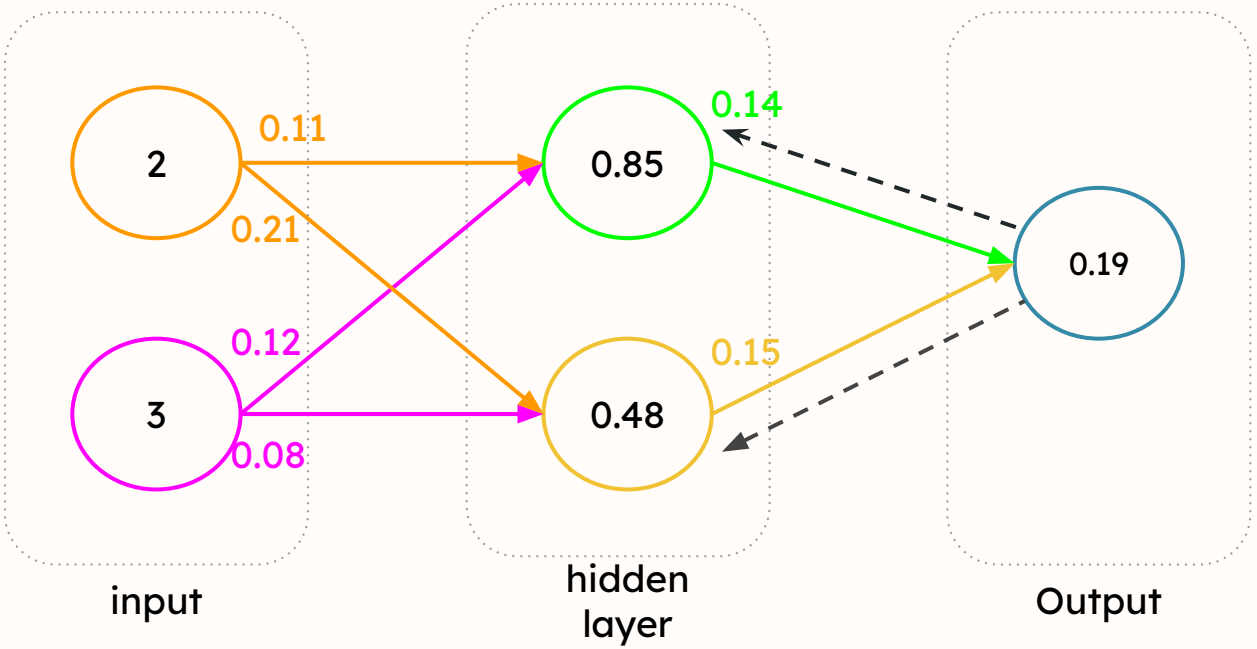
$$w_2^{1\prime} = w_2^1 - a\Delta h_2^1$$
$$w_1^{1\prime} = w_1^1 - a\Delta h_1^1$$

$$\begin{bmatrix} w_{1,1}^1 \\ w_{2,1}^1 \end{bmatrix} = \begin{bmatrix} w_{1,1}^1 \\ w_{2,1}^1 \end{bmatrix} - a\Delta \begin{bmatrix} h_1^1 \\ h_2^1 \end{bmatrix}$$

$$w'_{2,2} = w_{2\,,2} - a\Delta w_2^1 x_2$$
$$w'_{1,2} = w_{2\,,2} - a\Delta w_2^1 x_1$$
$$w'_{2,1} = w_{2\,,1} - a\Delta w_1^1 x_2$$
$$w'_{1,1} = w_{1\,,1} - a\Delta w_1^1 x_1$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} - a\Delta \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} w_{1,1}^1 & w_{2,1}^1 \end{bmatrix}$$

Backpropagation - Ajustando o erro



X = (2,3), y = 1, a = 0.05

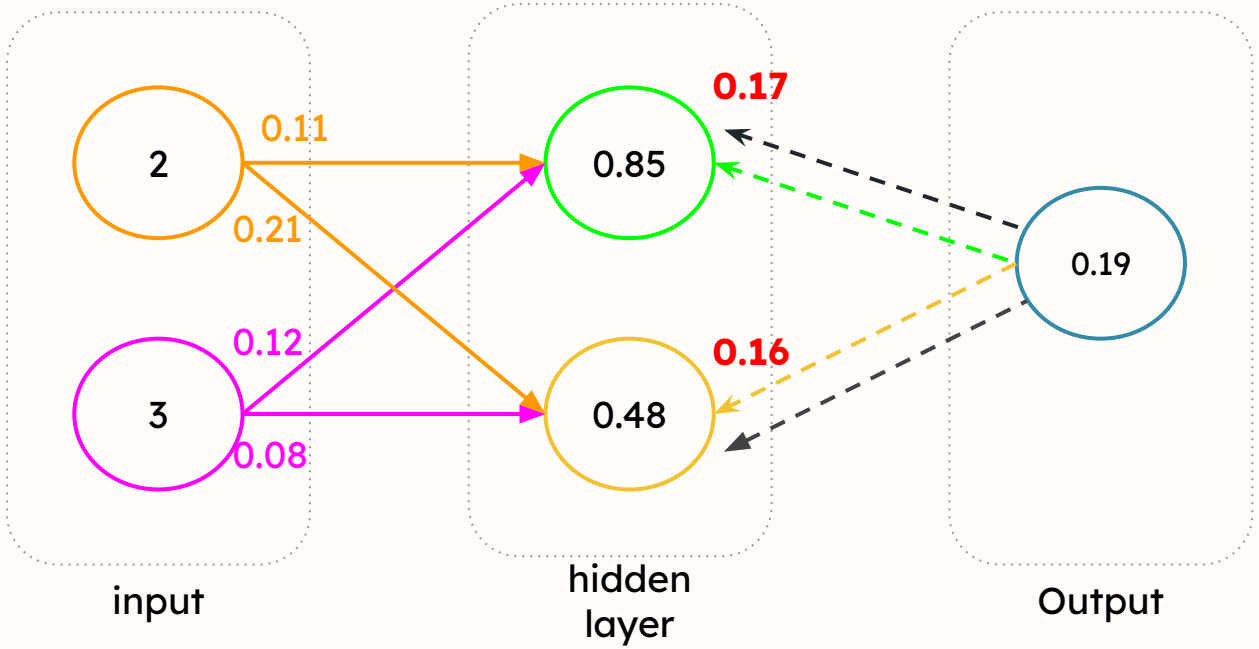
o = 0.191

$$\Delta = (o - y) = (0.191 - 1) = -0.809$$

$$\begin{bmatrix} w^1_{1,1} \\ w^1_{2,1} \end{bmatrix} = \begin{bmatrix} w^1_{1,1} \\ w^1_{2,1} \end{bmatrix} - a\Delta \begin{bmatrix} h^1_1 \\ h^1_2 \end{bmatrix}$$

$$\begin{bmatrix} w^{1'}_{1,1} \\ w^{1'}_{2,1} \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} - 0.05(-0.809) \begin{bmatrix} 0.85 \\ 0.48 \end{bmatrix} = \begin{bmatrix} 0.17 \\ 0.17 \end{bmatrix}$$

Backpropagation - Ajustando o erro



X = (2,3), y = 1, a = 0.05

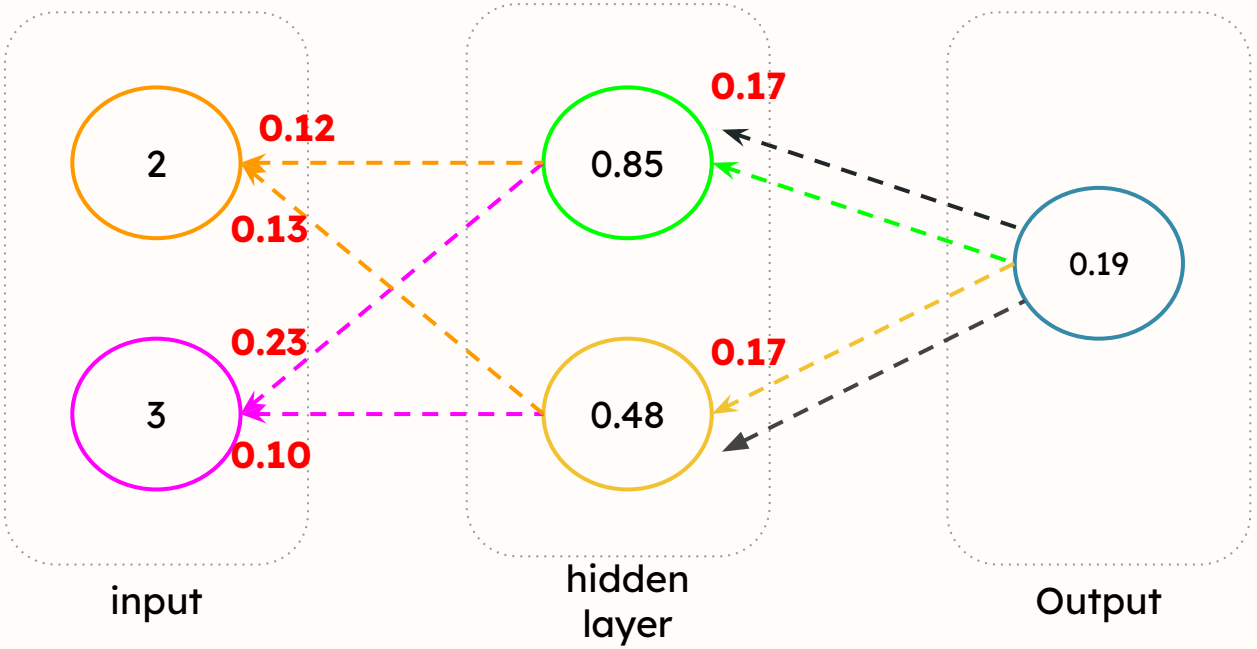
o = 0.191

$$\Delta = (o - y) = (0.191 - 1) = -0.809$$

$$\begin{bmatrix} w^1_{1,1} \\ w^1_{2,1} \end{bmatrix} = \begin{bmatrix} w^1_{1,1} \\ w^1_{2,1} \end{bmatrix} - a\Delta \begin{bmatrix} h^1_1 \\ h^1_2 \end{bmatrix}$$

$$\begin{bmatrix} w^1_{1,1} \\ w^1_{2,1} \end{bmatrix} \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} - 0.05(-0.809) \begin{bmatrix} 0.85 \\ 0.48 \end{bmatrix} = \begin{bmatrix} 0.17 \\ 0.17 \end{bmatrix}$$

Backpropagation - Ajustando o erro



X = (2,3), y = 1, a = 0.05

o = 0.191

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} - a\Delta \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} w^1_{1,1} & w^1_{2,1} \end{bmatrix}$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} = \begin{bmatrix} 0.11 & 0.12 \\ 0.21 & 0.08 \end{bmatrix} - 0.05(-0.809) \begin{bmatrix} 2 \\ 3 \end{bmatrix} \begin{bmatrix} 0.14 & 0.15 \end{bmatrix} = \begin{bmatrix} 0.12 & 0.13 \\ 0.23 & 0.10 \end{bmatrix}$$

Backpropagation - Ajustando o erro

$$Erro = \frac{1}{2}(o - y)^2$$

- ➔ A predição inicial tinha o valor **o=0.191**, com y = 1. Calculando o erro tínhamos **Erro = 0.327**
- ➔ Após o ajuste dos pesos, **o=0.26**. Então, o novo erro se torna **Erro = 0.273**
- ➔ Repetimos o forward e o backward ajustando os pesos até que o Erro se torne igual a 0 (ou o mais próximo possível)
- ➔ O ajuste dos pesos será definido pelo learning rate
- ➔ Na prática, os pesos são ajustados com base em vários exemplos contidos no dataset que possuem as respectivas rotulações definidas

Backpropagation - Parâmetros

➔ Vimos que a rede possui 4 conjuntos de parâmetros: 1. vetor de entrada, 2. número de camadas, 3. largura das camadas, 4. vetor de saída
- ◆ Sabemos a entrada X e a saída O possuem valores fixos
- ◆ Então quantos parâmetros treináveis teremos na rede?

Backpropagation - Parâmetros

- ➔ Vimos que a rede possui 4 conjuntos de parâmetros: 1. vetor de entrada, 2. número de camadas, 3. largura das camadas, 4. vetor de saída
  - ◆ Sabemos a entrada X e a saída O possuem valores fixos
  - ◆ Então quantos parâmetros treináveis teremos na rede?

- ➔ Se a primeira camada escondida tem l1 neurônios então teremos na primeira camada l1 * xn + 1 (n pesos conectados a cada neurônio e 1 valor para o bias)
- ➔ Na segunda camada teremos l1 * l2 + 1, e assim sucessivamente, até lm * o + 1

Backpropagation - Parâmetros

➔ Para uma rede **R** com um conjunto de parâmetros Θ e M camadas escondidas, o número de parâmetros treináveis é dado por:

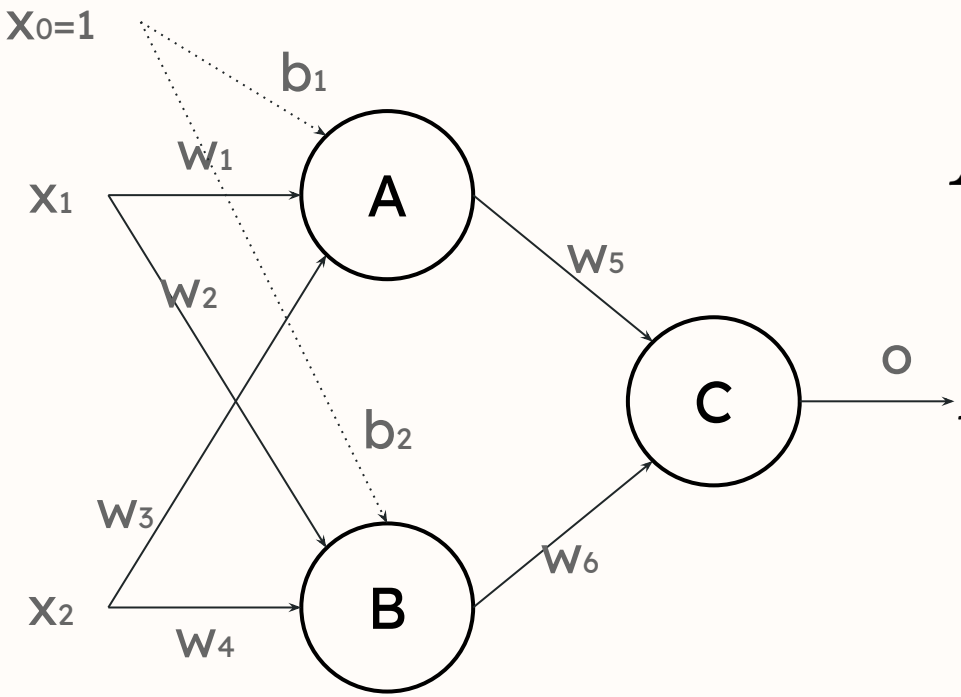
$$|\theta| = \left[\left|l^M\right| \cdot n + 1\right] + \left[\sum_{i=1}^{M-1} \left(\left|l^i\right| \cdot \left|l^{i+1}\right|\right) + 1\right] + \left[\left|l^M\right| \cdot p + 1\right]$$

➔ Para simplificação, se considerarmos a entrada X como a camada 0 e a saída como a camada M+1, então:

$$|\theta| = \sum_{i=0}^{M+1} \left(\left|l^i\right| \cdot \left|l^{i+1}\right|\right) + 1$$

Backpropagation - Parâmetros

- ➔ Vimos que podemos simplificar a notação considerando o vetor de entrada como a camada 0
- ➔ Da mesma forma, podemos considerar para cada camada um termo adicional x0, com x0 = 1, que multiplica o valor de bias



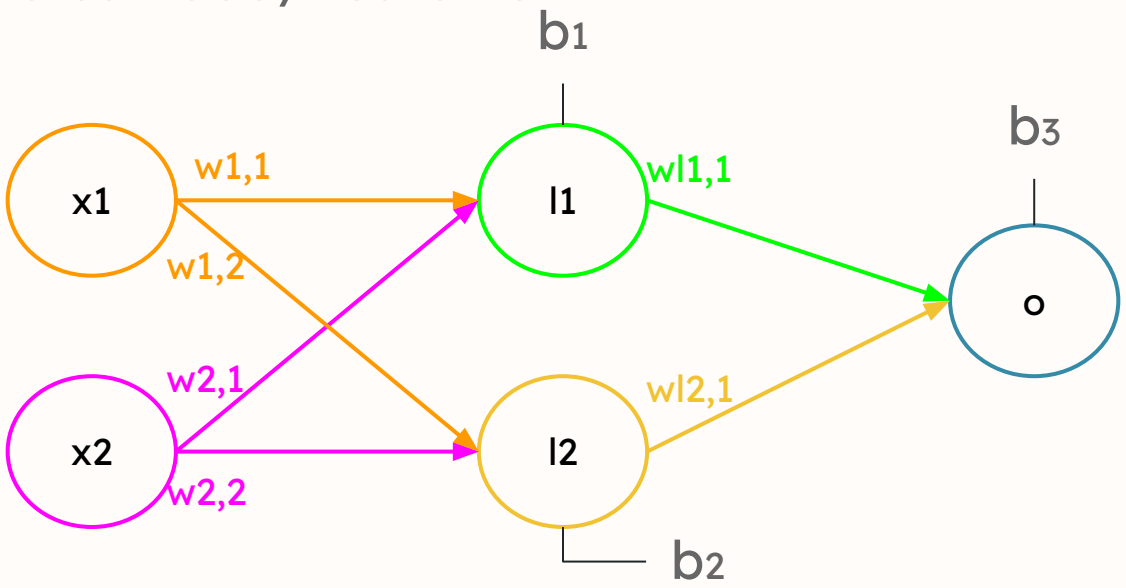
$$A = w_1 x_1 + w_2 x_2 + b_1$$

$$A = w_1 x_1 + w_2 x_2 + 1 \cdot b_1$$

$$A = w_1 x_1 + w_2 x_2 + x_0 b_1$$

Backpropagation - Parâmetros

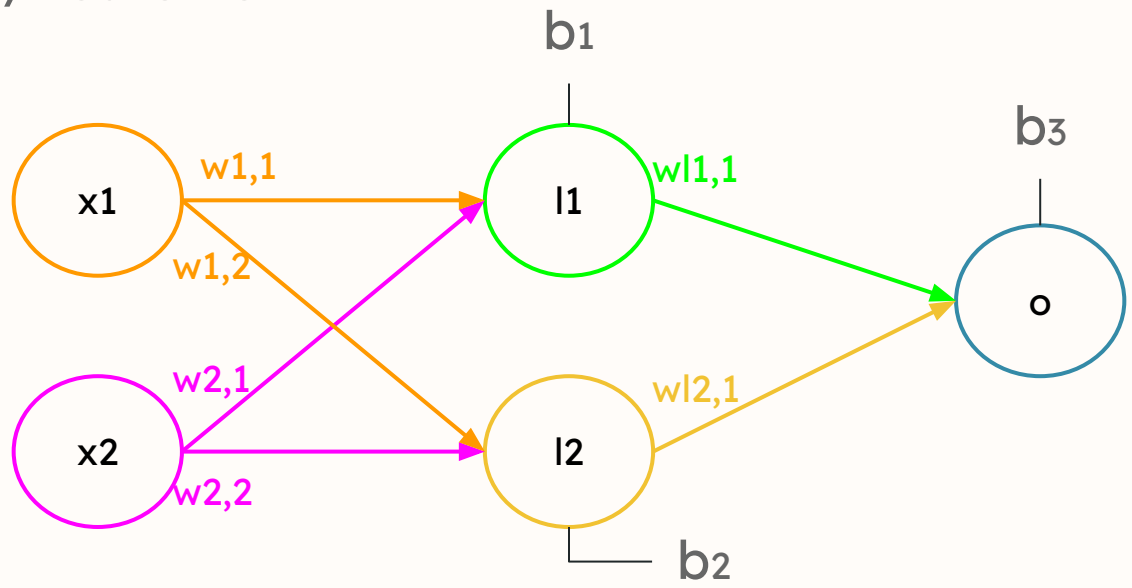
- ➔ Aplicamos o valor adicional no início de cada camada
- ➔ Desta forma podemos simplificar a representação do modelo e utilizar a representação vetorial vista nas equações anteriores para incluir o bias, sem realizar um passo de soma adicional para cada camada/neurônio



$$[x_1 \quad x_2] \cdot \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = [l_1^1 \quad l_2^1] \implies [l_1^1 \quad l_2^1] \cdot \begin{bmatrix} w_{1,1}^1 \\ w_{2,1}^1 \end{bmatrix} + b_3 = o$$

Backpropagation - Parâmetros

- ➔ Aplicamos o valor adicional no início de cada camada
- ➔ Desta forma podemos simplificar a representação do modelo e utilizar a representação vetorial vista nas equações anteriores sem realizar um passo de soma adicional para cada camada/neurônio



$$[1 \quad x_1 \quad x_2] \cdot \begin{bmatrix} b_1 & b_2 \\ w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} = [l_1^1 \quad l_2^1] \implies [1 \quad l_1^1 \quad l_2^1] \cdot \begin{bmatrix} b_3 \\ w_{1,1}^1 \\ w_{2,1}^1 \end{bmatrix} = o$$

Backpropagation - Referências Adicionais

1. **Dive Into Deep Learning**. Cap 5 - Multilayer Perceptrons, 5.3 Forward Propagation, Backward Propagation, and Computational Graphs (https://d2l.ai/chapter_multilayer-perceptrons/backprop.html
2. **Deep Learning Book**. Cap 15 - Algoritmo Backpropagation (https://www.deeplearningbook.com.br/algoritmo-backpropagation-parte-2-treinamento-de-redes-neurais/)

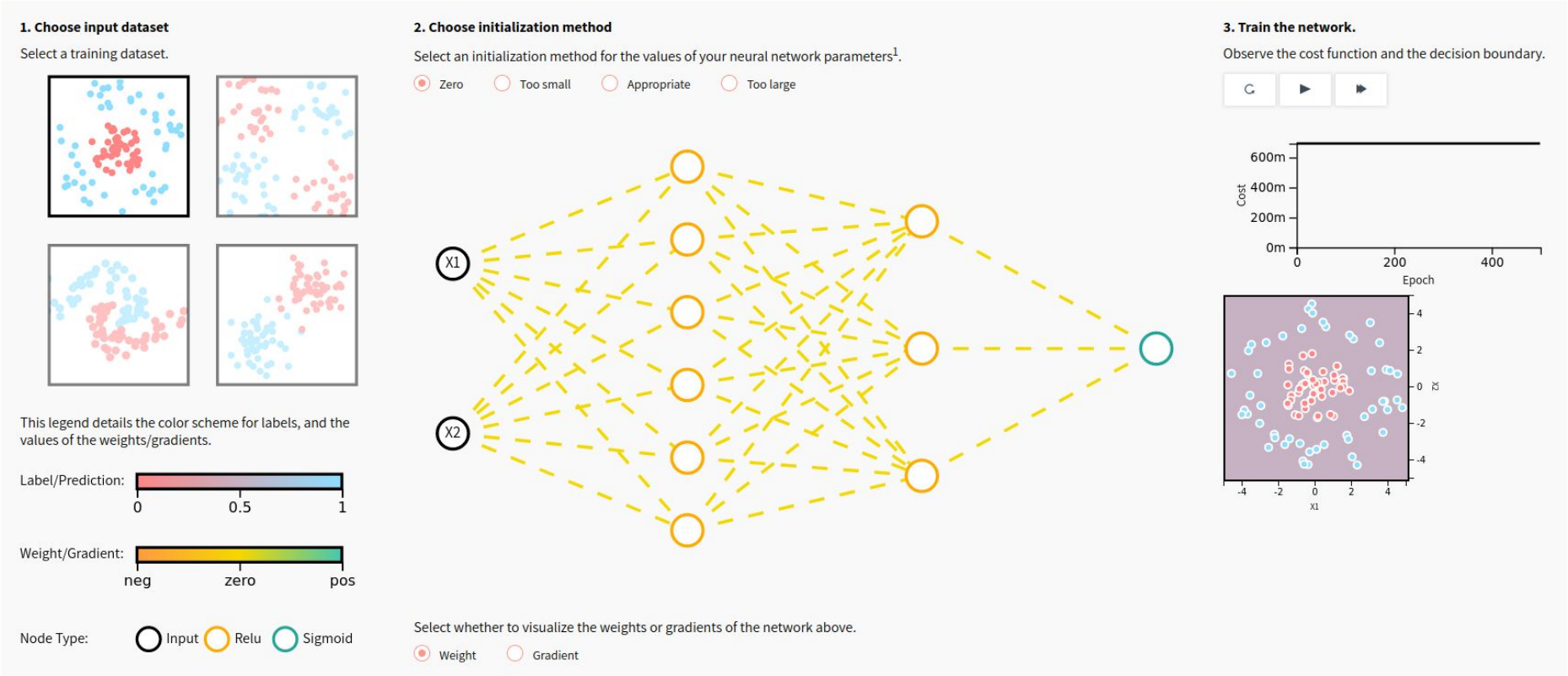
Inicialização dos pesos

"... training deep models is a sufficiently difficult task that most algorithms are strongly **affected by the choice of initialization**. The initial point can determine whether the algorithm converges at all, with some initial points being so unstable that the algorithm encounters numerical difficulties and fails altogether."

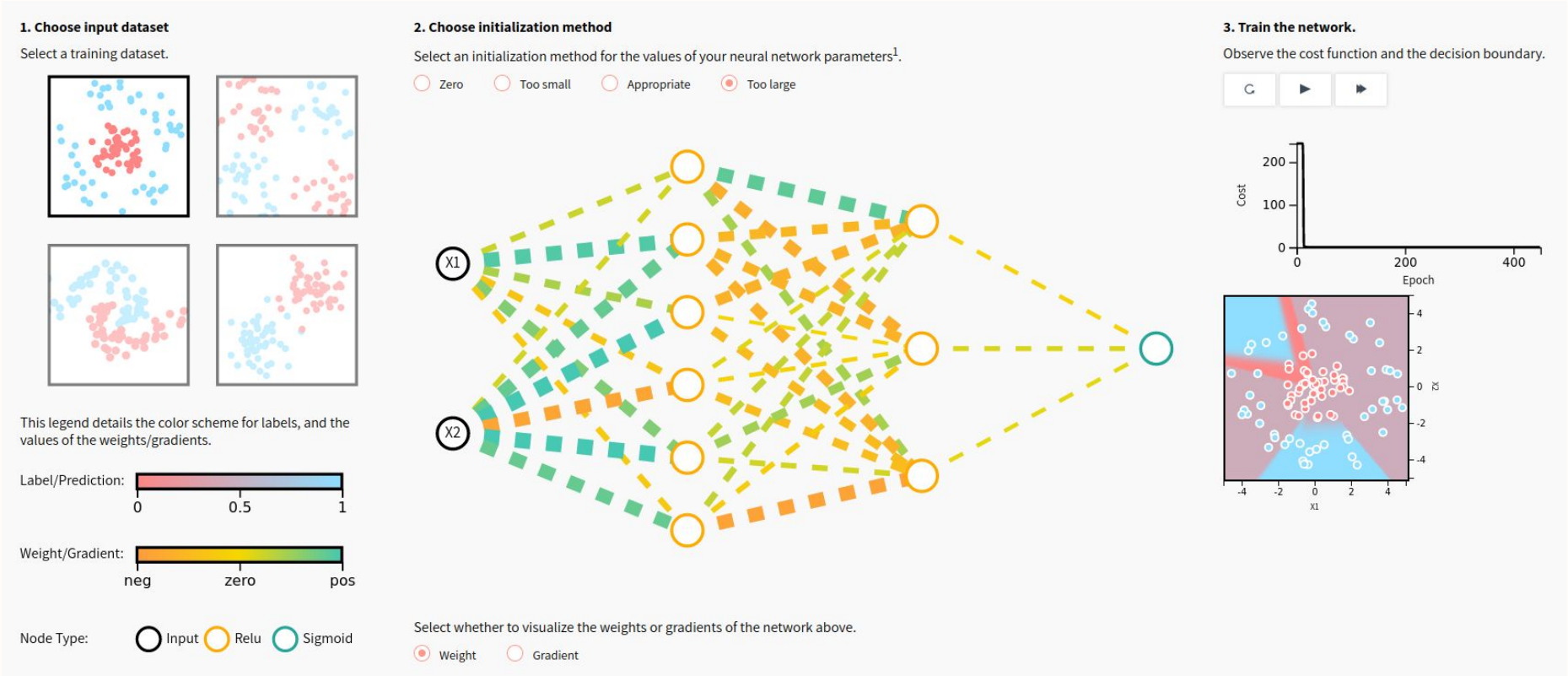
"We almost always initialize all the weights in the model to values drawn **randomly from a Gaussian or uniform distribution**. The choice of Gaussian or uniform distribution does not seem to matter very much, but has not been exhaustively studied. The scale of the initial distribution, however, **does have a large effect on both the outcome of the optimization procedure** and on the ability of the network to generalize."

Deep Learning book (https://www.deeplearningbook.org/), 2016

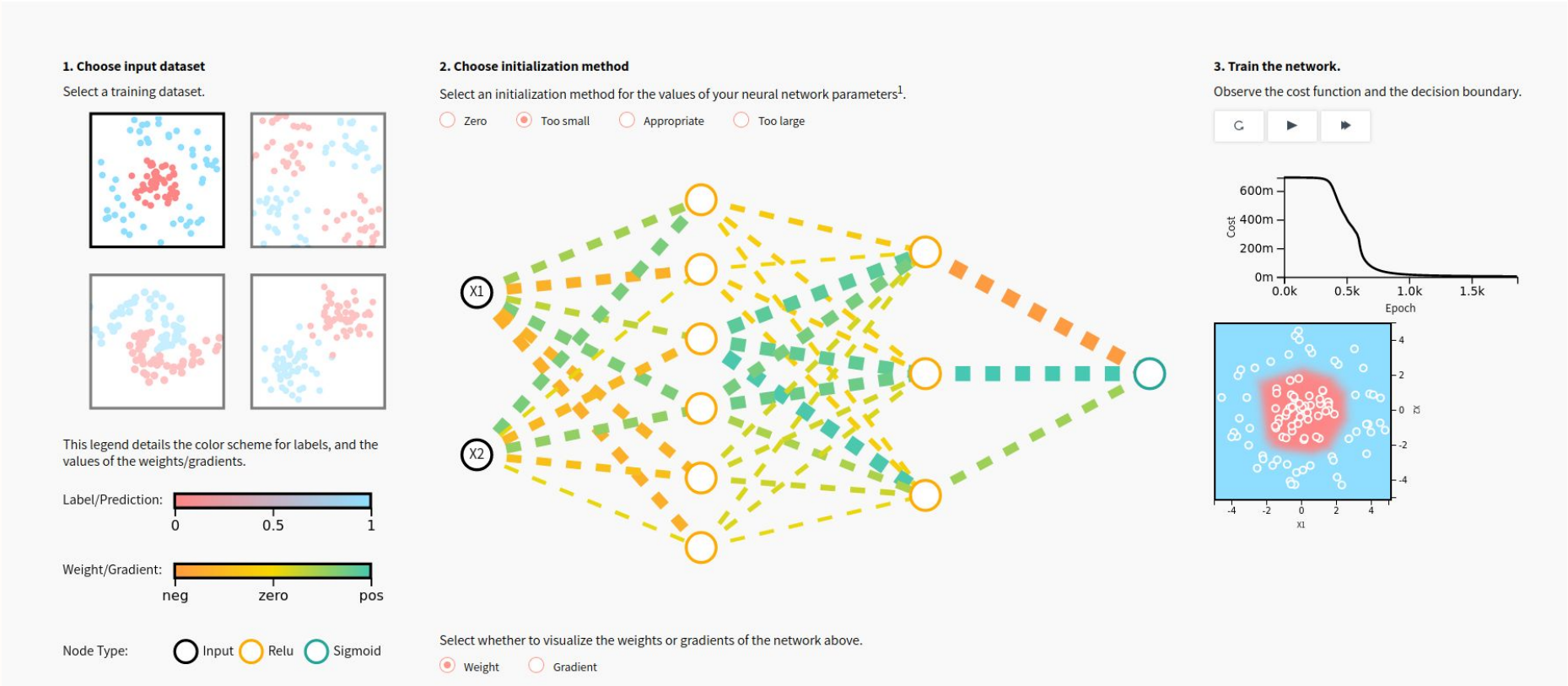
Inicialização dos pesos - inicialização zero



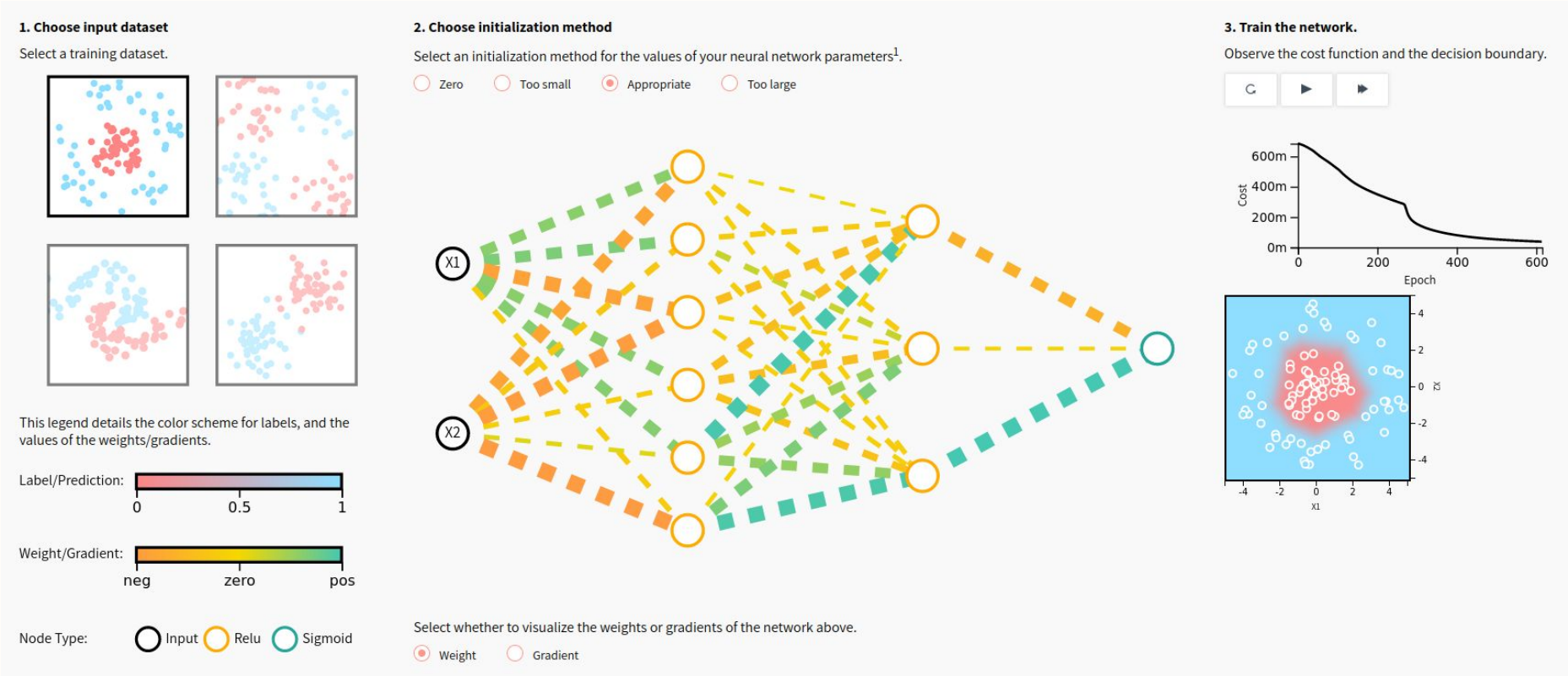
Inicialização dos pesos - inicialização alta



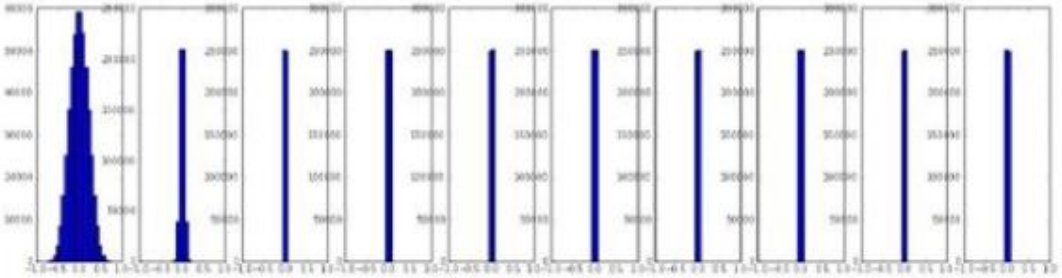
Inicialização dos pesos - inicialização baixa



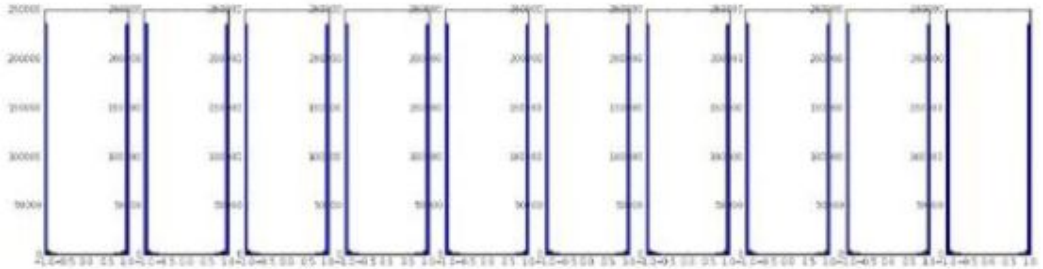
Inicialização dos pesos - inicialização adequada



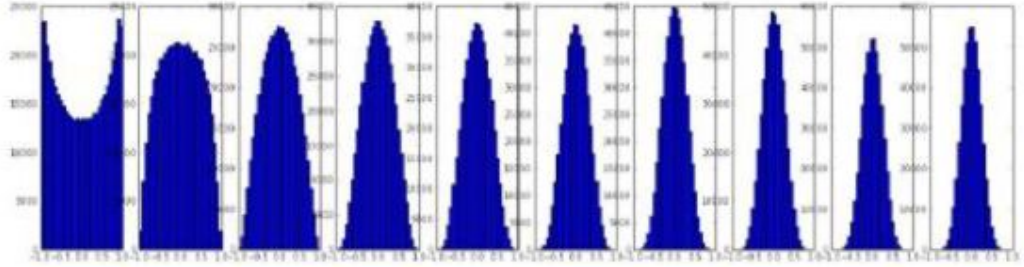
Inicialização dos pesos



➔ inicialização baixa



➔ inicialização alta

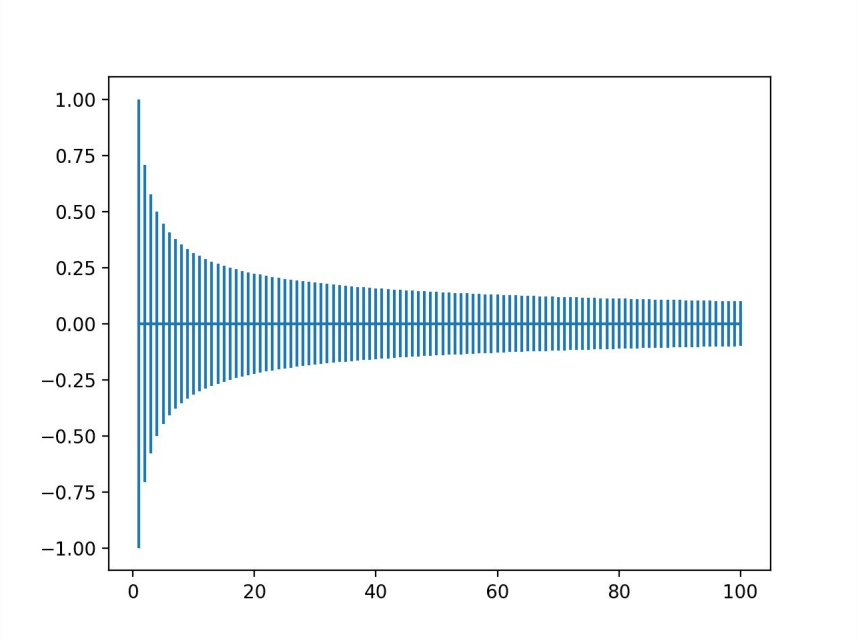


➔ inicialização adequada

Fonte: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Inicialização dos pesos - Inicialização Xavier

- ➔ Inicialização uniforme: Cada peso (w) é atribuído a partir de uma distribuição uniforme em [-x, x] com x = sqrt(6/inputs + outputs)
- ➔ Inicialização normal: Cada peso (w) é atribuído a partir de uma distribuição normal com média 0 e desvio sqrt(2/inputs+outputs)



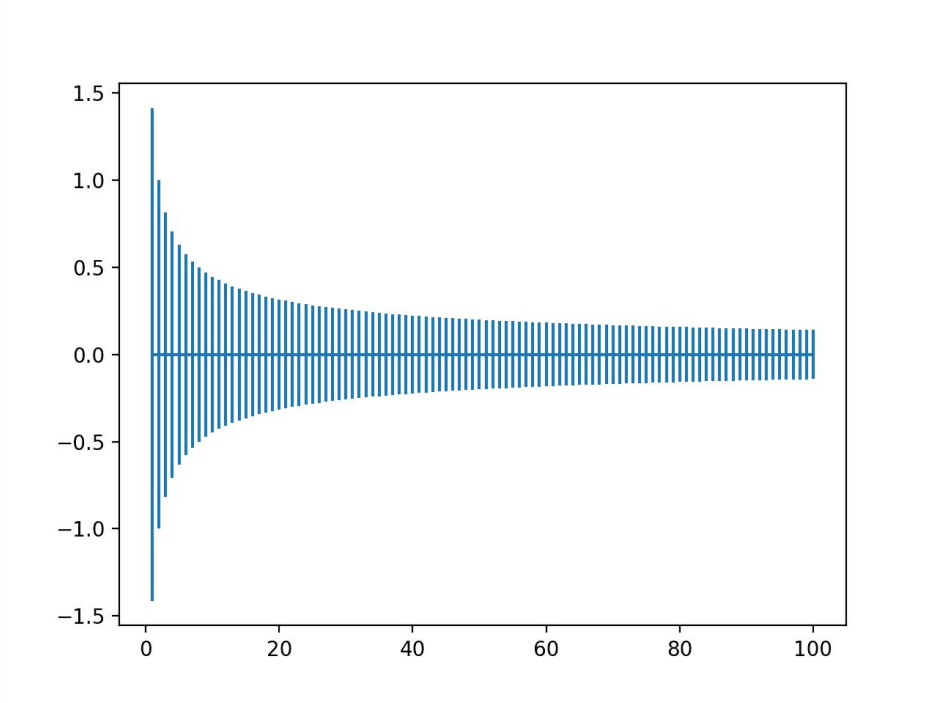
Fonte da imagem: https://machinelearningmastery.com/weight-initialization-for-deep-learning-neural-networks/

Inicialização dos pesos - Inicialização Xavier

- ➔ Tenta garantir que o produto entre x e w não é muito baixo ou muito alto
- ➔ Normalização garante que os dados têm médio 0 e variância 1
  - ◆ Variância dos produtos da rede é 1 (Var(NN) de w.x = 1)

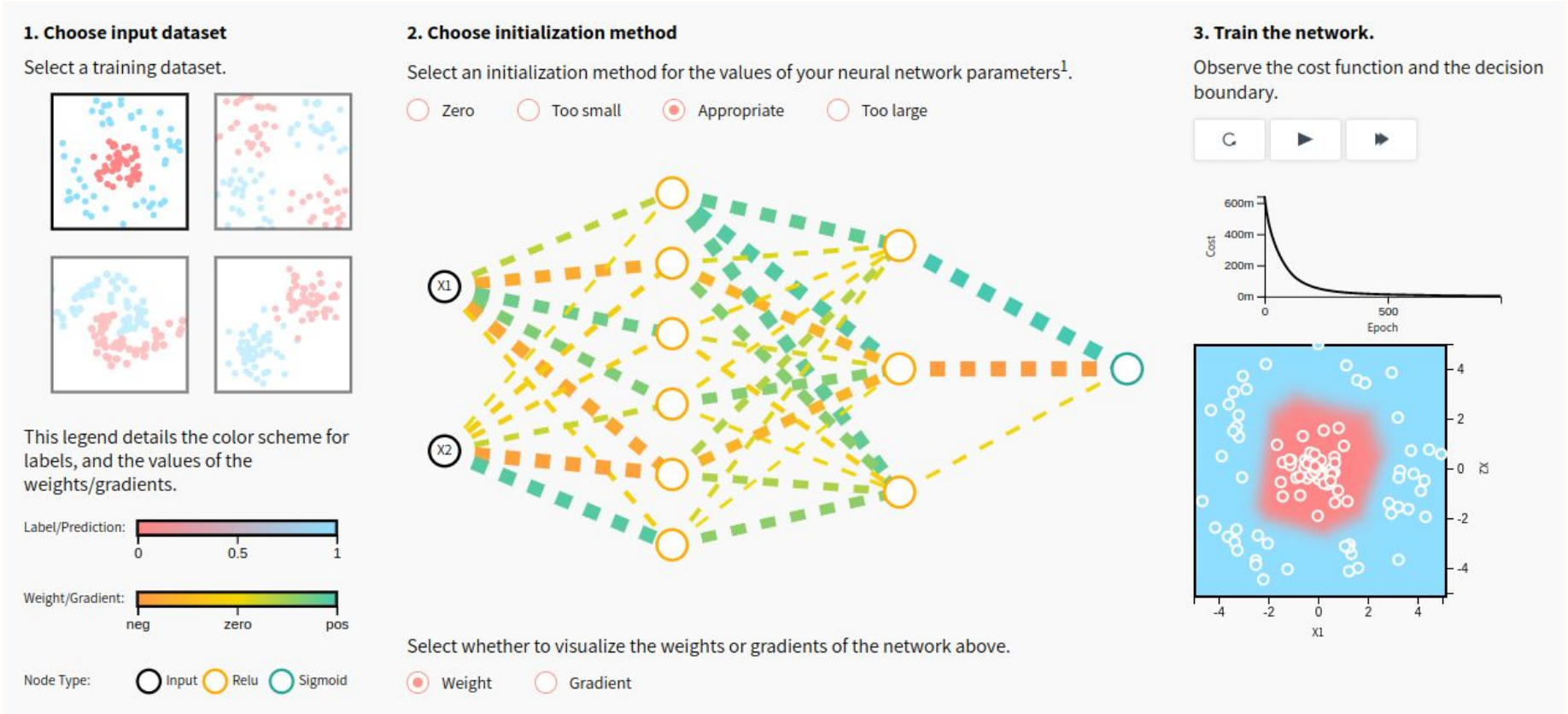
Inicialização dos pesos - Inicialização He

- ➔ Inicialização uniforme: Cada peso (w) é atribuído a partir de uma distribuição gaussiana G(0.0, sqrt(2/inputs))
  - ◆ Utilizada para ativações ReLU



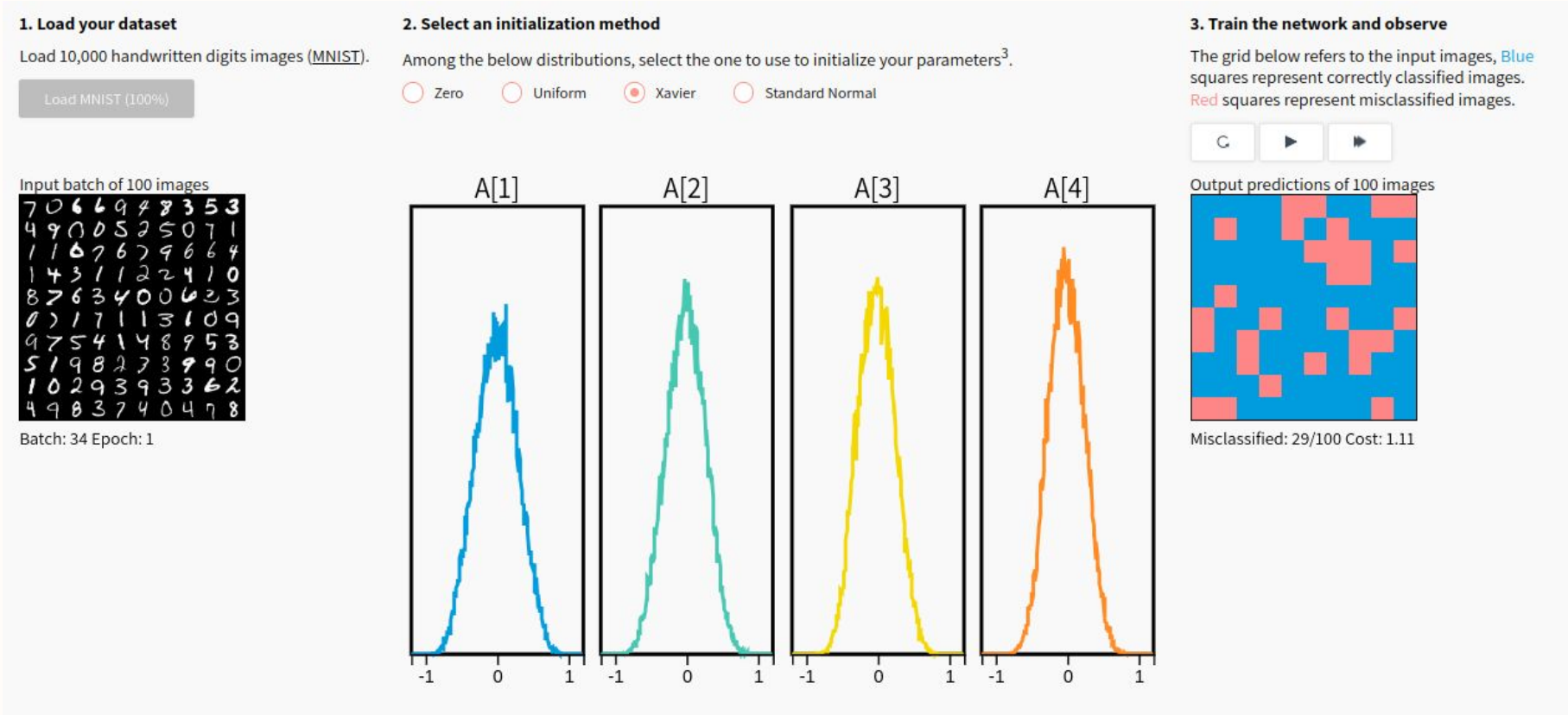
Fonte da imagem: https://machinelearningmastery.com/weight-initialization-for-deep-learning-neural-networks/

Inicialização dos pesos



https://www.deeplearning.ai/ai-notes/initialization/index.html

Inicialização dos pesos

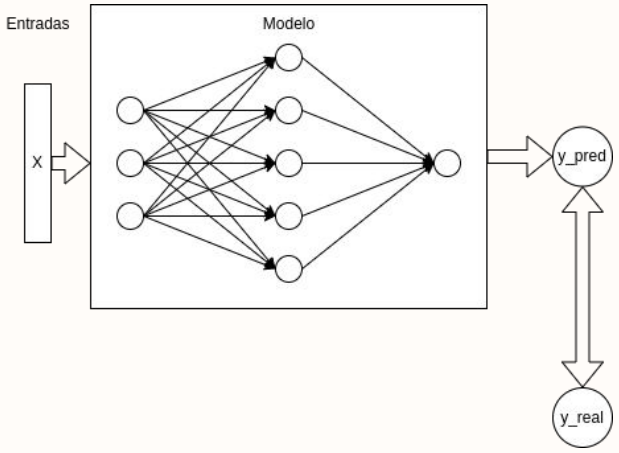


https://www.deeplearning.ai/ai-notes/initialization/index.html

Funções de perda

Funções de Perda

- Também denominada **Funções de Custo**, são funções que medem a discrepância entre os valores preditos dos modelos e os valores reais conhecidos.
- Os modelos têm como objetivo diminuir os erros atrelados às previsões, ou seja, o treinamento dos parâmetros dos modelos devem ser tais que a função seja mínima.



$$L\left(y_{real},\ y_{pred}\right)$$

Funções de Perda

- Para aplicação dos modelos em diferentes tarefas requerem diferentes funções de perda, com objetivo de melhorar os modelos a partir das características de interesse.
- Nesse sentido, algumas funções de perda serão vistas a seguir, tanto para resolução de problemas de regressão como classificação.

Mean Absolute Error (MAE) Loss

- O MAE é obtido a partir da médias dos valores absolutos da diferença entre o valor real e o predito.
- Esse erro é aplicado em problemas de **regressão**.

$$L\left(y_{true},\ y_{pred}\right) = \frac{1}{N}\sum\left|y_{true} - y_{pred}\right|$$

Likelihood Loss

- É uma função simples utilizada para problemas de classificação.
- Consiste no produtório das probabilidades previstas em cada exemplo.

$$p(y \mid x; \theta) = h_\theta(x)^y (1 - h_\theta(x))^{1-y}$$

$$L(\theta) = \prod_{i=1}^{n} p(y_i \mid x_i;\ \theta)$$

Mean Squared Error (MSE) Loss

- O MAE é obtido a partir da médias dos valores quadráticos da diferença entre o valor real e o predito.
- Esse erro é aplicado em problemas de **regressão**. Sendo o mais utilizado para esse tipo de previsão.
- Grandes divergências são penalizadas devido ao valor quadrático das diferenças.

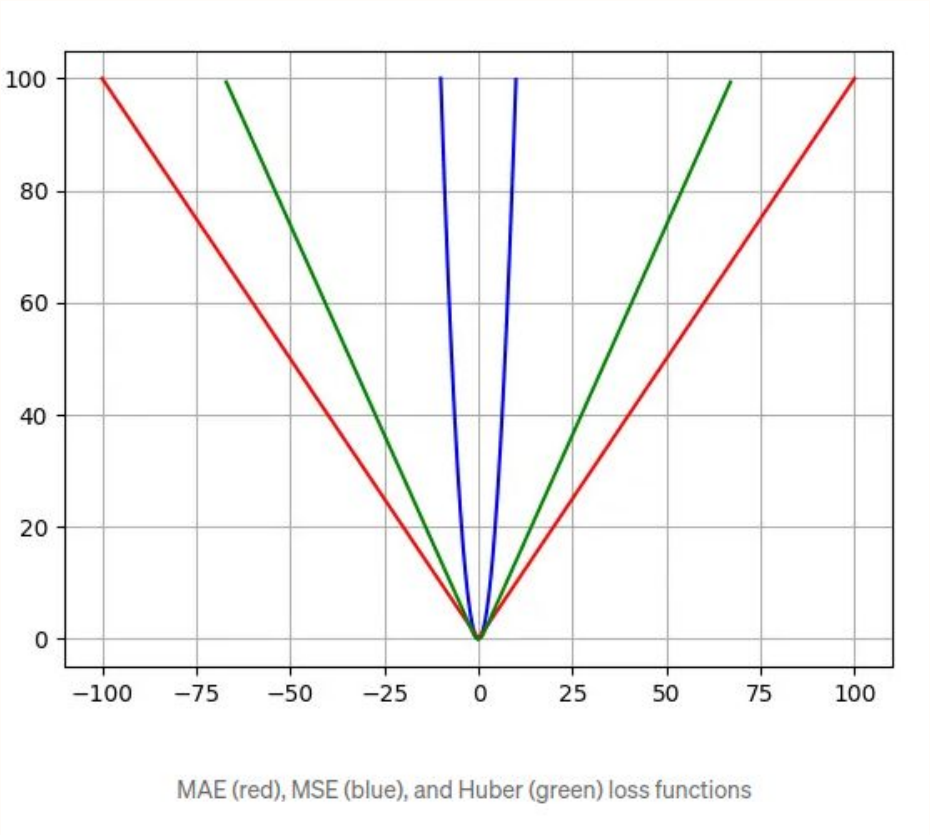
$$L\left(y_{true},\ y_{pred}\right) = \frac{1}{N}\sum\left(y_{true} - y_{pred}\right)^2$$

Huber Loss

- É a combinação entre MAE e MSE, no qual computa o MSE para pequenas divergências e o MAE para grandes divergências.
- Algumas vantagens dessa função é ser menos sensível a outliers e pode prevenir explosão dos gradientes.

$$L(y_{true},\ y_{pred}) = \begin{cases} 0,5\ (y_{true} - y_{pred})^2,\ se\ |y_{true} - y_{pred}| \leq\ \delta \\ \delta(|y_{true} - y_{pred}| - 0,5\delta),\ \text{otherwise} \end{cases}$$

Comparação dos gráficos



MAE (red), MSE (blue), and Huber (green) loss functions

Log-Likelihood Loss

- Função aplicada em problemas de classificação. Trata-se do cálculo do logaritmo da função de verossimilhança e, assim, somar as probabilidades de cada classe.
- A função logarítmica é monotônica e crescente, por esse motivo, o sinal negativo é aplicado para torná-la decrescente.
- Para o caso binário, tem-se que:

$$L\left(y_{true},\ y_{pred}\right) = -\sum\left(y_{true}\log y_{pred} + (1 - y_{true})\log\left(1 - y_{pred}\right)\right)$$

Binary Cross-Entropy Loss

- A aplicação do classificador com saídas probabilísticas entre 0 e 1 podem ser considerada a entropia cruzada para casos binários, no qual tem-se que:
  - Quando a probabilidade da classe se aproxima de 1, os valores são baixos.
  - Quando a probabilidade da classe se aproxima de 0, os valores são altos.
- A entropia cruzada é dada por:

$$H(p,\ q) = -\sum_{x \in X} p(x) \log(q(x))$$

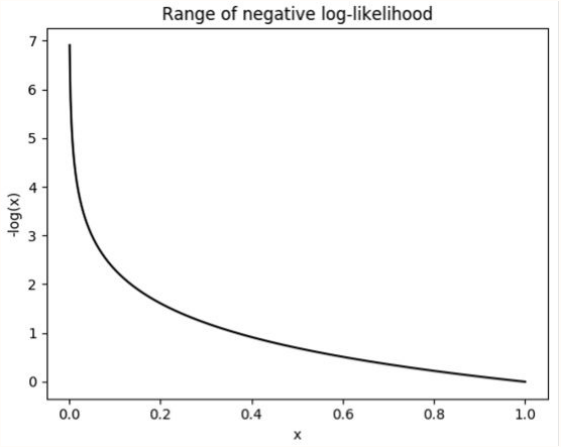


Imagem: https://bit.ly/34lISuB

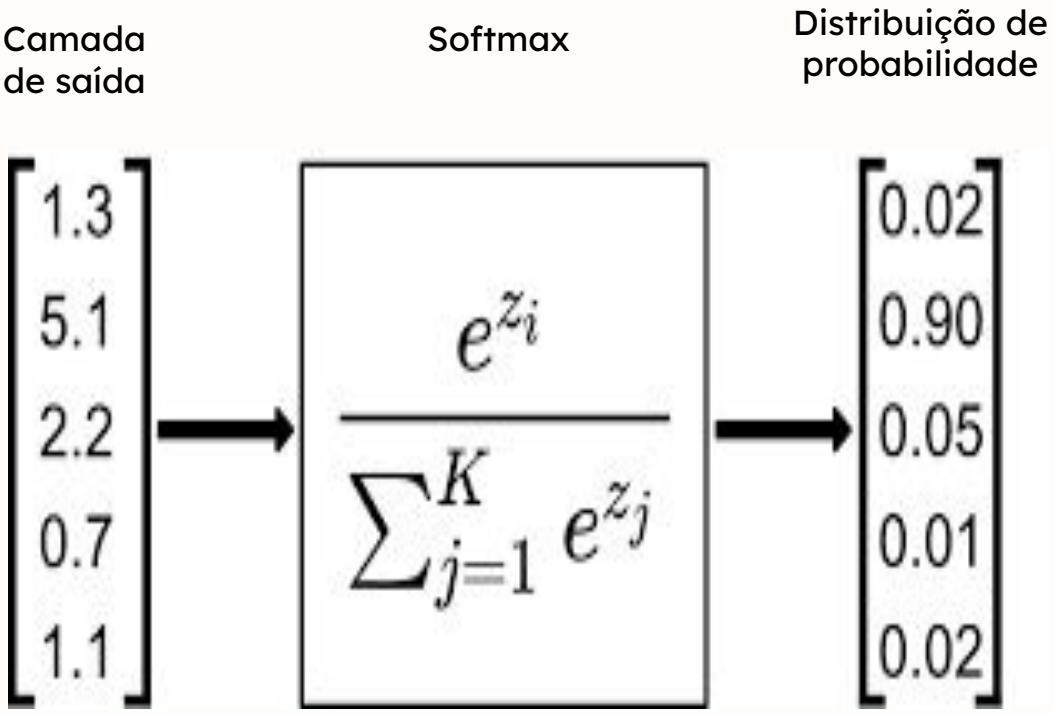
Softmax

- Quando utiliza-se de redes neurais para treinamento de classificadores, dois casos podem acontecer:
  - Os valores dos pesos treinados não estar no intervalo [0, 1].
  - A soma dos pesos não serem igual a 1.
- Nesse sentido, a função softmax tem como objetivo normalizar os pesos de saída dos neurônios de modo a obedecer uma distribuição de probabilidade com as condições ditas acima.
- Sua equação é dada por:

$$\sigma(\mathbf{z}) = \frac{e^{z_i}}{\sum_{c=1}^{K} e^{z_c}}$$

Softmax

- A partir do exemplo a seguir, tem-se:



https://bit.ly/3tgylSR (adaptado)

Cross-Entropy Loss

- Para o caso de múltiplas classes, a entropia cruzada pode ser generalizada tendo como auxílio da função Softmax, devido a necessidade de utilizar a distribuição de probabilidade nas saídas dos modelos.
- Essa função é a mais utilizada para múltiplas classes.
- Veja o exemplo a seguir

Probabilidades

0,7

0,2

0,1

Cross-Entropy

$$-\sum_{c=1}^{K} y_{true} \log\left(y_{pred}\right)$$

One-Hot Encoded Labels

1

0

0

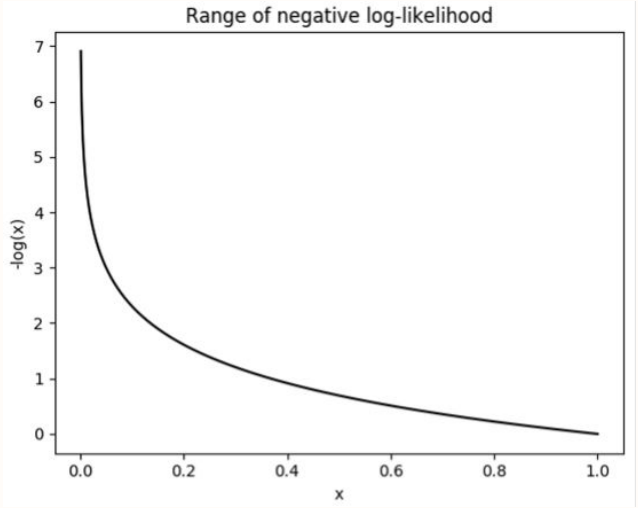


Imagem: https://bit.ly/34lISuB

Multiclass Hinge Loss

- Normalmente utilizada para *Suport Vector Machine* (SVM), essa função soma as divergências entre a predição e o valor real.



	Image #1	Image #2	Image #3
Dog	-0.39	-4.61	1.03
Cat	1.49	3.28	-2.37
Horse	4.21	1.46	-2.27

$$L_i = \sum_{h \neq y_i} \max\left(0,\ s_j - s_{y_i} + \Delta\right)$$

https://pyimagesearch.com/2016/09/05/multi-class-svm-loss/

Kullback-Leibler Divergence Loss

- Tem como objetivo fornecer informações sobre duas distribuições de probabilidade e como elas se diferem uma da outra.
- Exige um esforço computacional maior.

$$L\left(y_{true},\ y_{pred}\right) = y_{true}\left(\log y_{true} - \log y_{pred}\right)$$

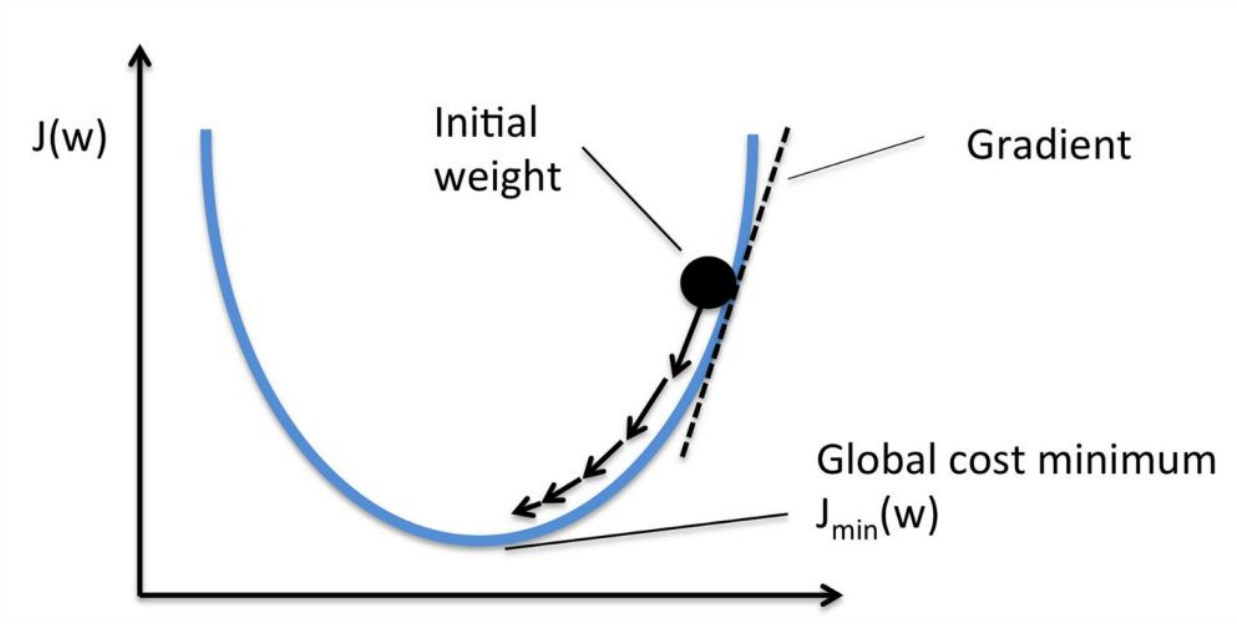
Otimizadores

Otimizadores

- ➔ Funções de perda têm como objetivo mensurar o erro que o modelo possui.
- ➔ Os algoritmos de otimização auxiliam a minimizar o custo dos pesos no treinamento de forma iterativa

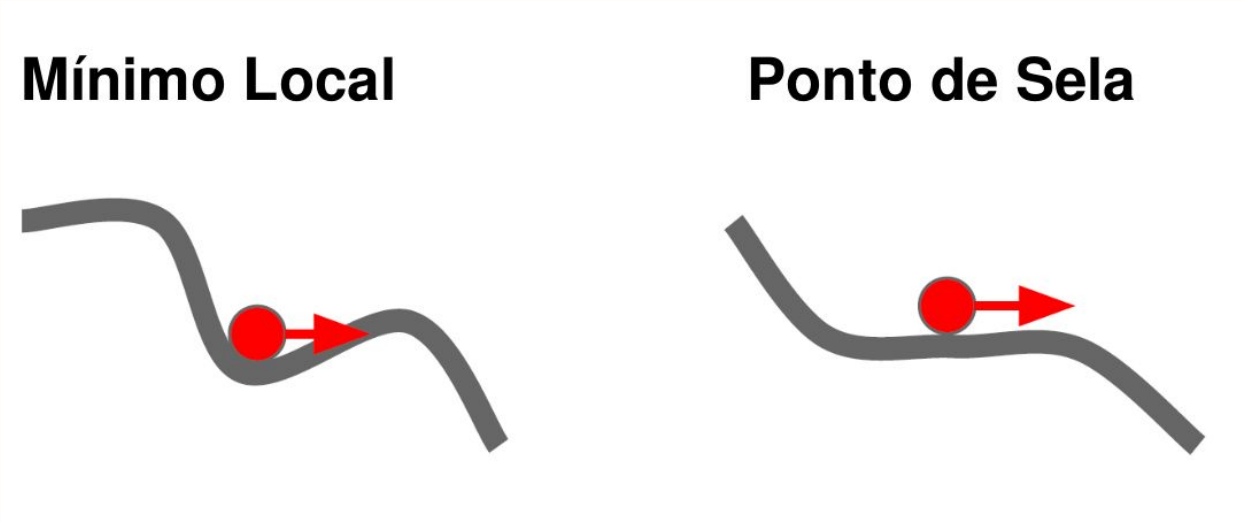
Gradiente Descendente

$$w \leftarrow w - \eta \nabla J(\theta)$$



Problemas de treinamento

➔ Devido ao comportamento das funções de de perda, alguns possíveis problemas podem ocorrer durante o treinamento.



Momentum

- ➔ A aplicação do momento faz com que possíveis problemas com mínimos locais e pontos de sela sejam evitados.
- ➔ O momento consiste em aplicar uma "velocidade" à convergência dos parâmetros, de modo a não aprisioná-lo nesses espaços.

$$v_t = \gamma v_{t-1} + \eta \nabla J(\theta)$$

$$\theta = \theta - v_t$$

Adagrad

➔ Adapta a taxa de aprendizado para pequenas atualizações em características conhecidas e maiores atualizações para características pouco frequente.

$$g_t \leftarrow \nabla f\left(\theta_{t-1}\right)$$

$$s_t = s_{t-1} + g_t^2$$

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{s_t + \epsilon}} g_t$$

RMSProp

- ➔ Divide a taxa de aprendizado por uma média exponencialmente decrescente do quadrado dos gradientes.
- ➔ Desacopla o decaimento do aprendizado de sua taxa.

$$g_t \leftarrow \nabla f\left(\theta_{t-1}\right)$$

$$s_t \leftarrow \gamma s_{t-1} + (1+\gamma)\, g_t^2$$

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{s_t + \epsilon}} g_t$$

Adam

➔ Combina todas as técnicas anteriores, de modo a ser mais eficiente no algoritmo de aprendizado. É o mais utilizado atualmente.

$$g_t \leftarrow \nabla f\left(\theta_{t-1}\right)$$

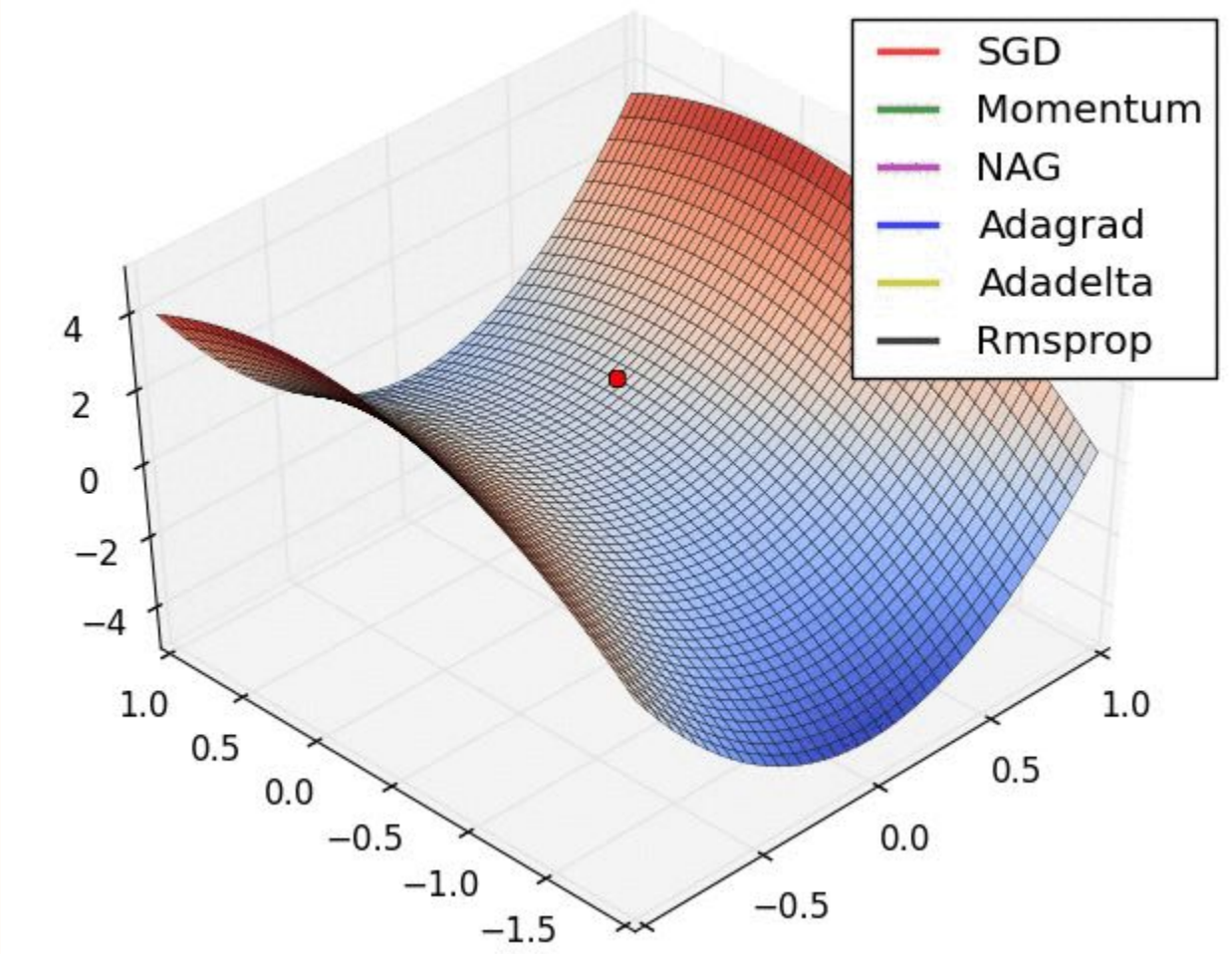
$$v_t \leftarrow \beta_1 v_{t-1} + \left(1+\beta_1\right) g_t$$

$$s_t \leftarrow \beta_2 s_{t-1} + \left(1+\beta_2\right) g_t^2$$

$$\hat{v}_t = \frac{v_t}{1-\beta_1^t} \qquad \hat{s}_t = \frac{s_t}{1-\beta_2^t}$$

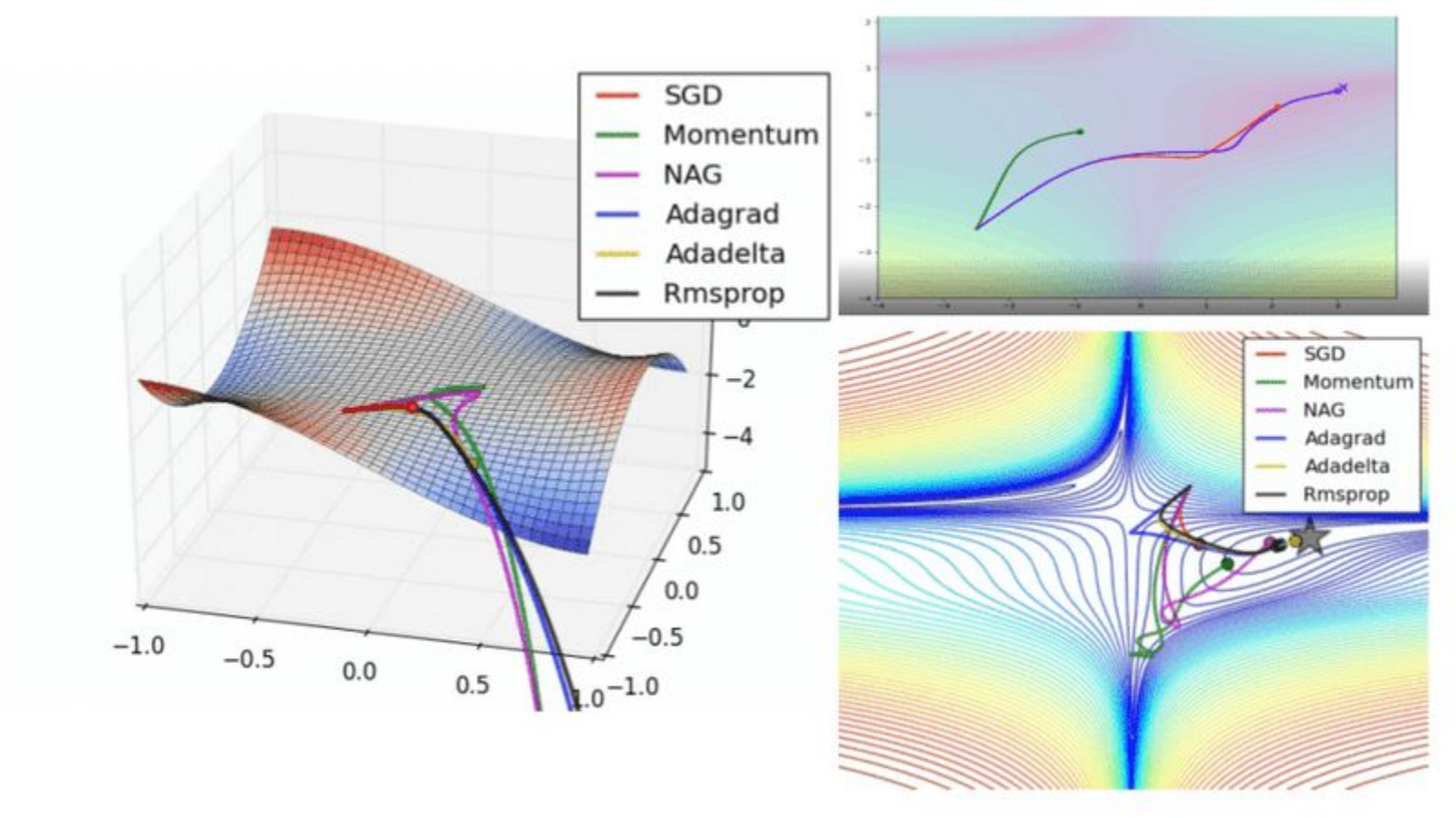
$$x_t \leftarrow x_{t-1} - \frac{\eta \hat{v}_t}{\sqrt{\hat{s}_t + \epsilon}}$$

Otimizadores



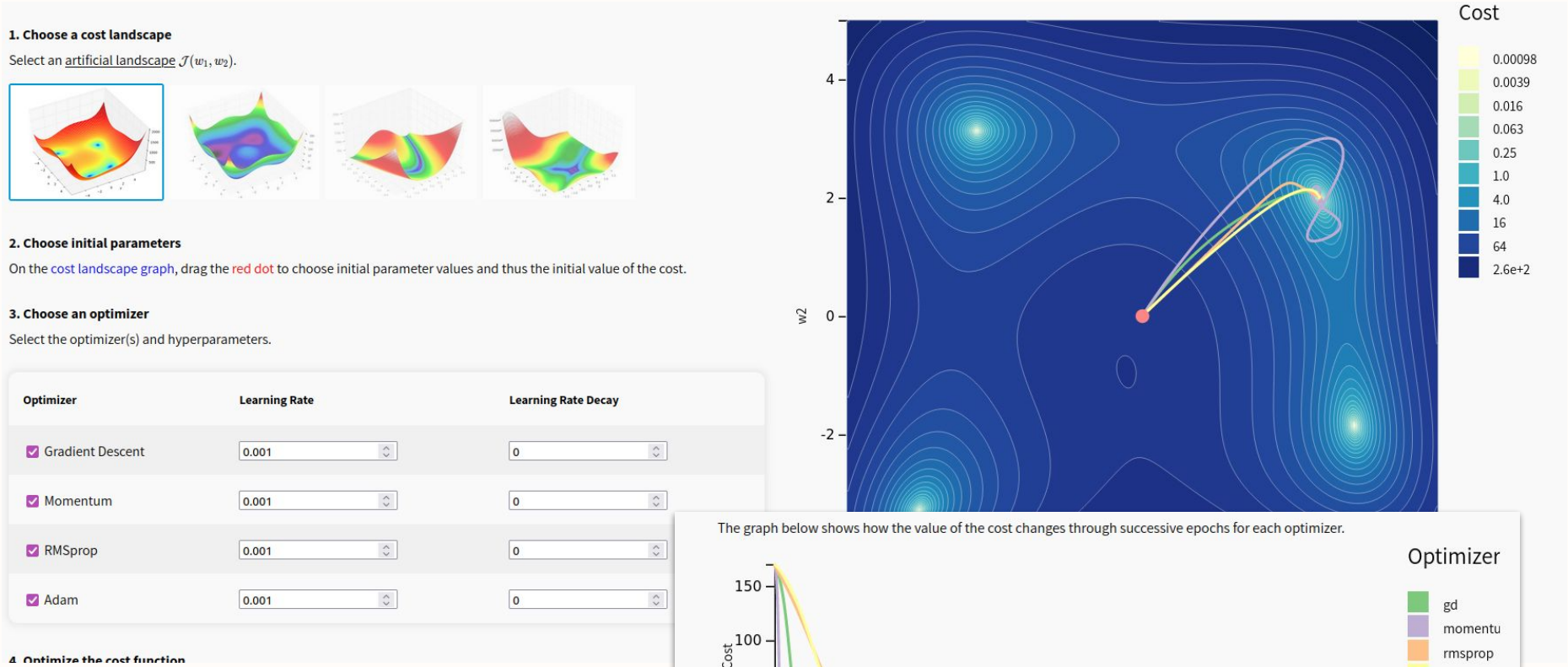
Fonte: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

Otimizadores



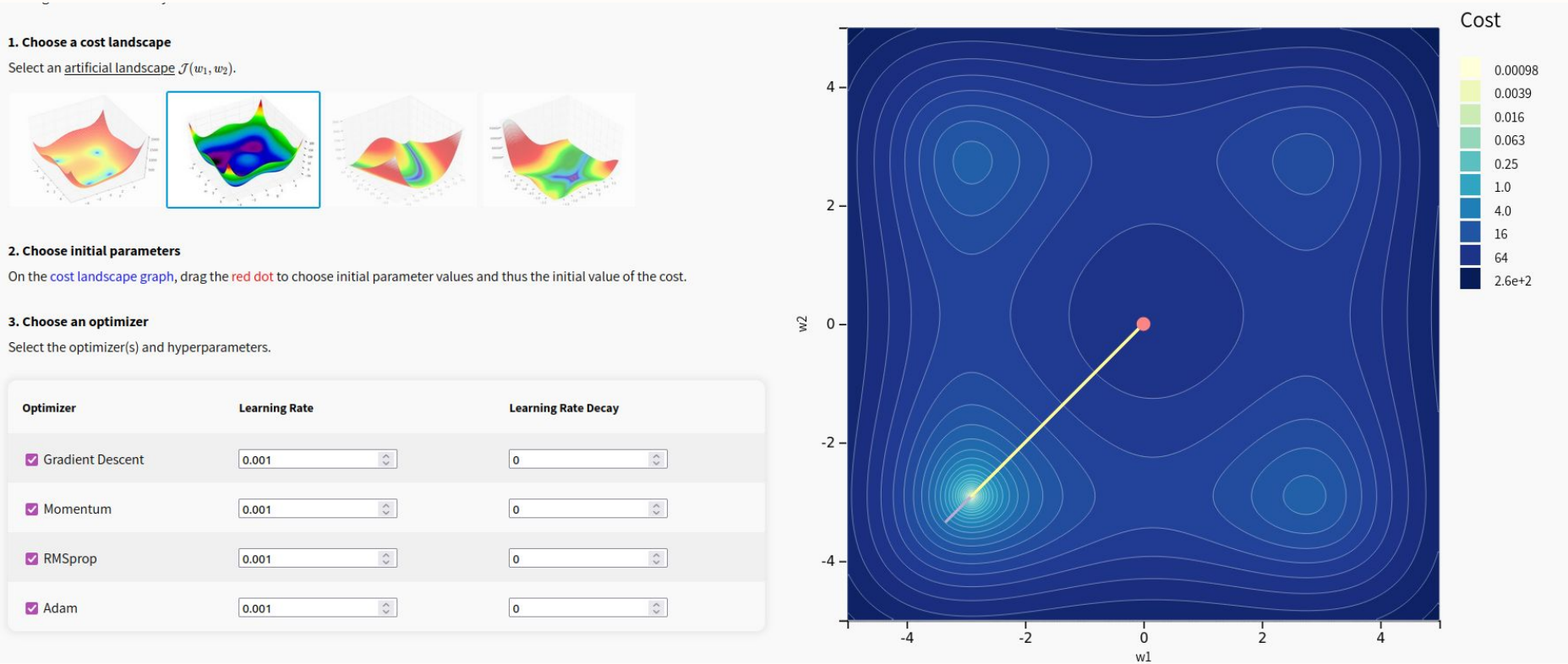
Fonte: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

Otimizadores



https://www.deeplearning.ai/ai-notes/optimization/index.html

Otimizadores



https://www.deeplearning.ai/ai-notes/optimization/index.html

Leituras complementares

- Dive into Deep learning (Cap. 2,3,4,5) (https://d2l.ai/)
- Deep Learning Book (Part I) (https://www.deeplearningbook.org/)
- SKLearn Linear Models (https://scikit-learn.org/stable/supervised_learning.html)
  - https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py
  - https://scikit-learn.org/stable/modules/linear_model.html#stochastic-gradient-descent-sgd
  - https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html
- DeepLearning.AI inicialization (https://www.deeplearning.ai/ai-notes/initialization/index.html)
- DeepLearning.AI optimization (https://www.deeplearning.ai/ai-notes/optimization/index.html)

Obrigado pela atenção!

INSTITUIÇÃO EXECUTORA

Centro de Informática UFPE

FADE UFPE

COORDENADORA

MCTI FUTURO

Softex

APOIO

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO