INSTITUTO FEDERAL
Ceará

Programa de Pós-Graduação
em Ciência da Computação

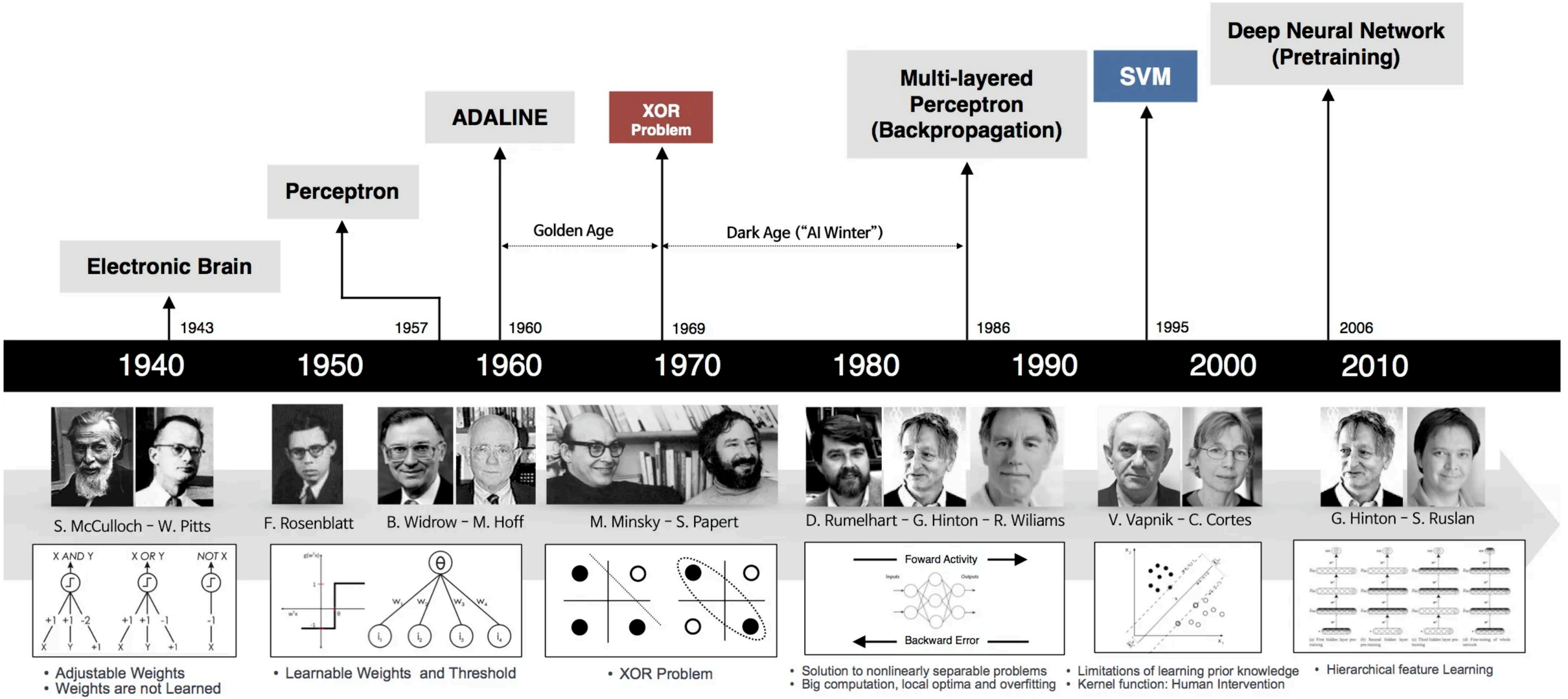
04 Perceptron de Múltiplas Camadas

APRENDIZAGEM PROFUNDA

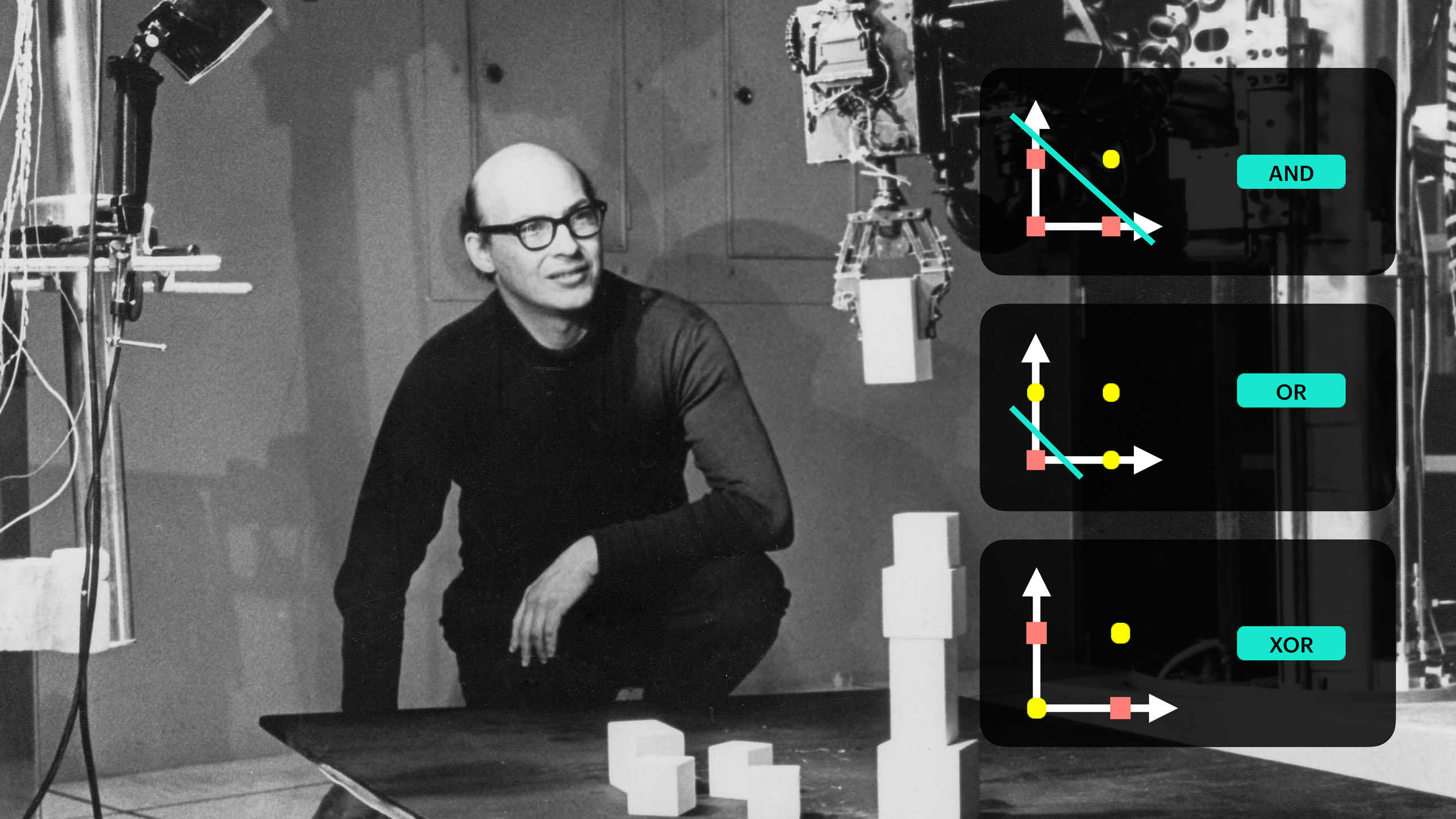
PPGCC – 2023.1

Prof. Saulo Oliveira <saulo.oliveira@ifce.edu.br>

Evolução das Redes Neurais Artificiais

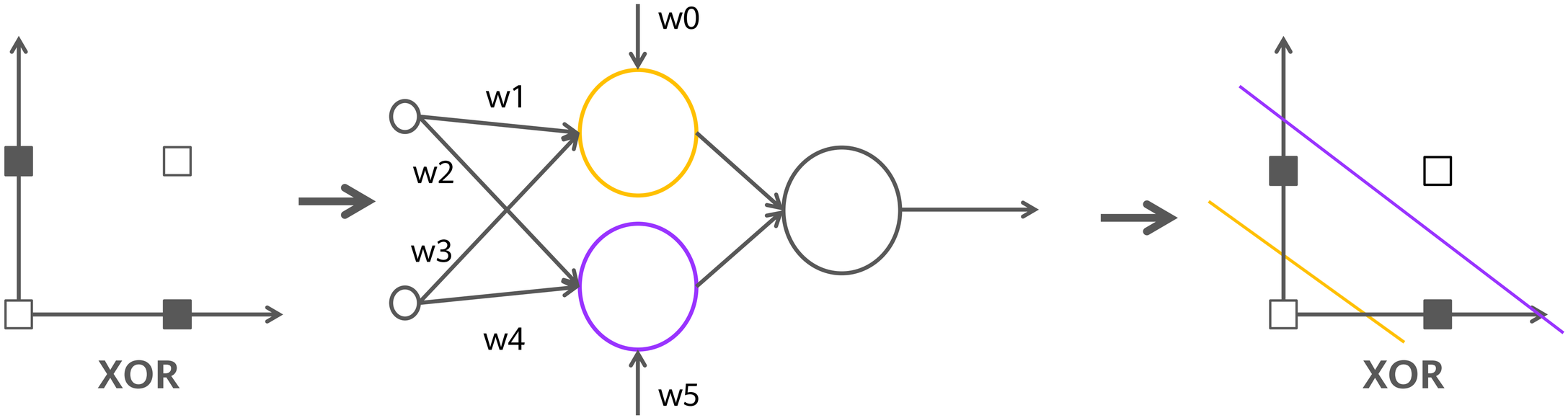


Fonte: https://sefiks.com/2017/10/14/evolution-of-neural-networks/


AND
OR
XOR

E O XOR?
COMO RESOLVE, MDS?

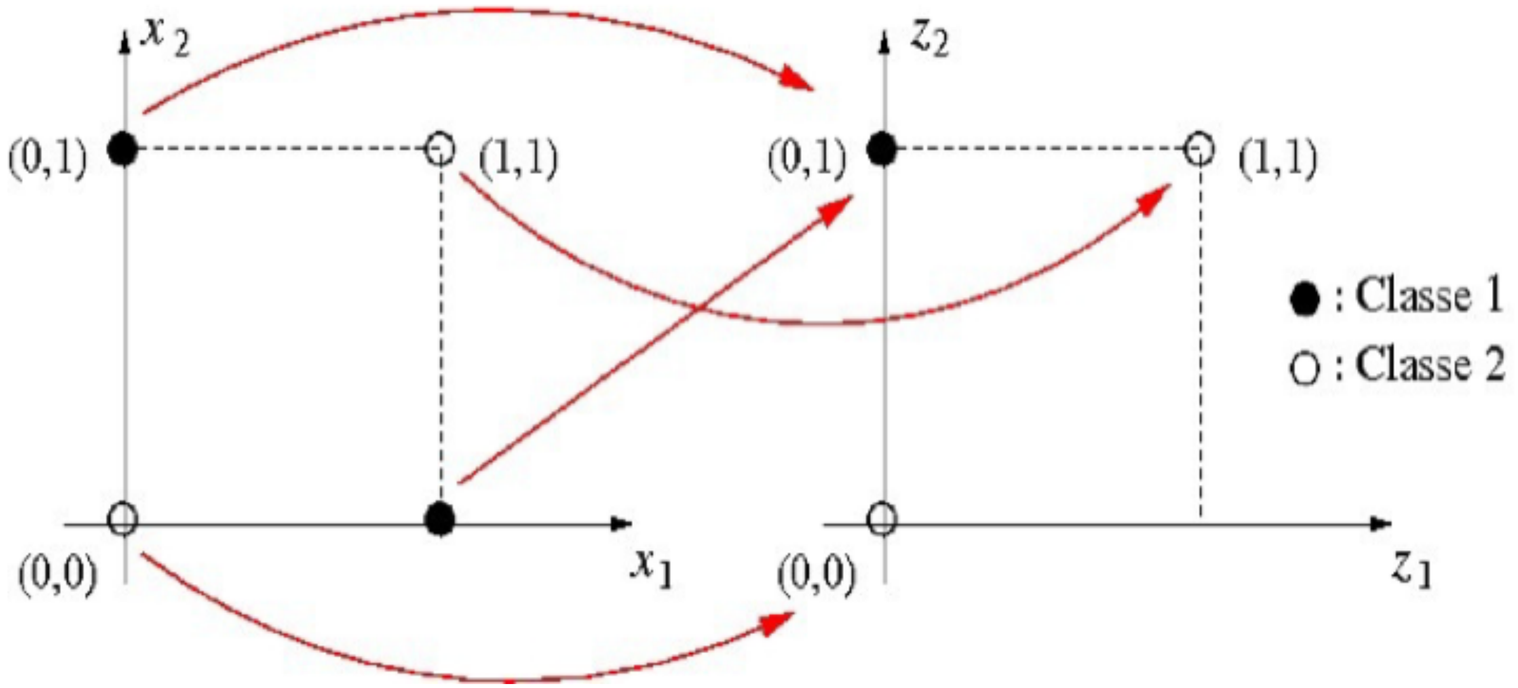
A (BENDITA) CAMADA OCULTA



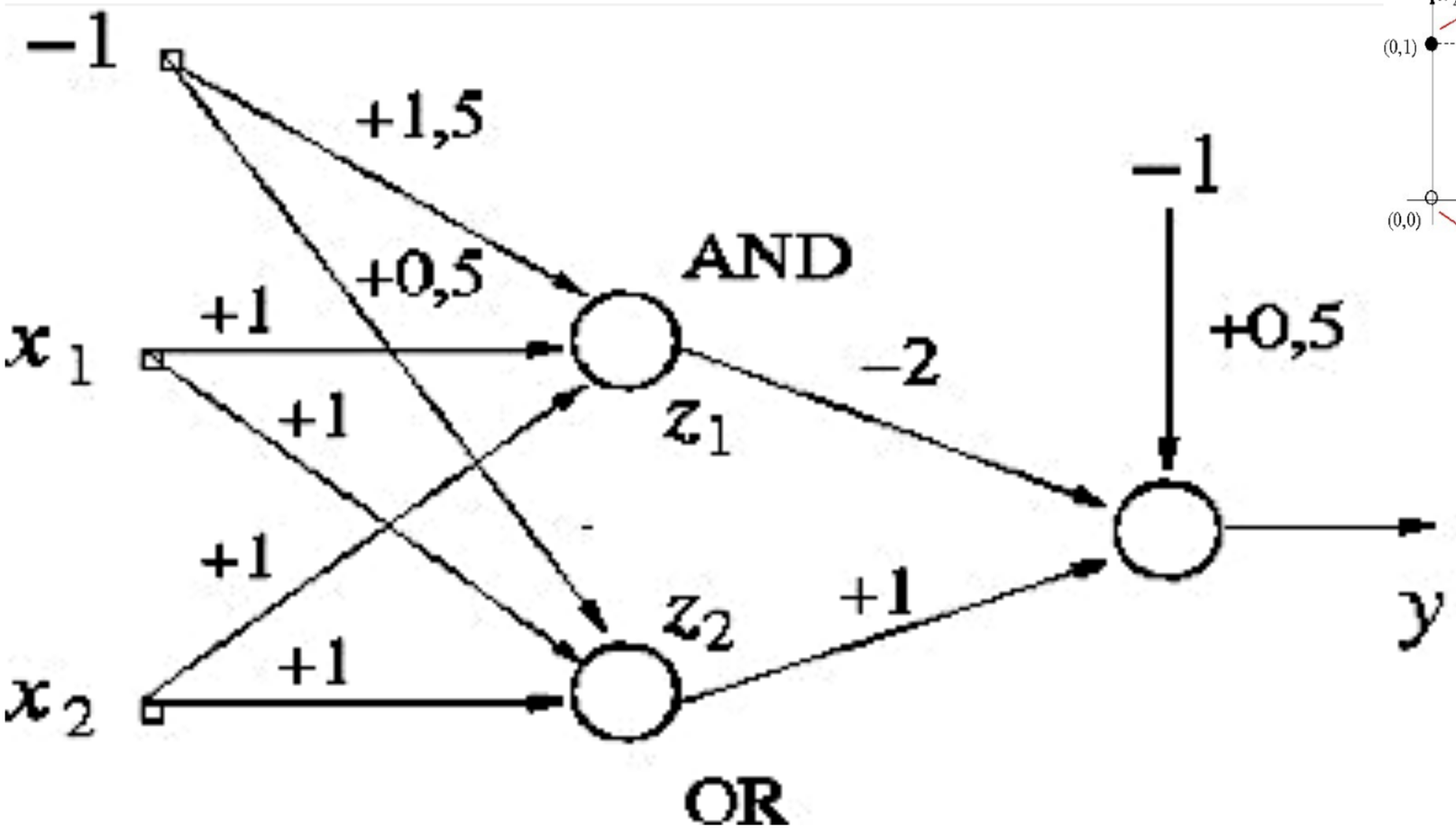
REVISITANDO PORTAS LÓGICAS

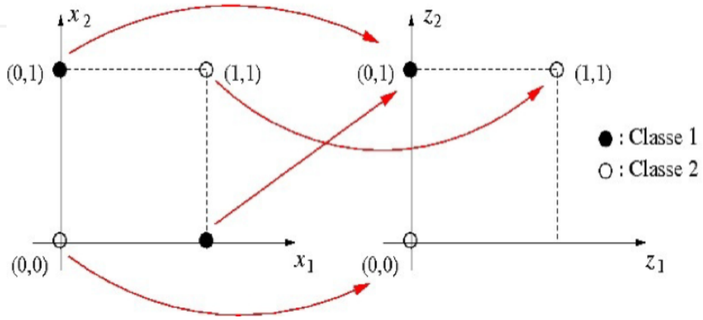
XOR

x_1	x_2	z_1	z_2	y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

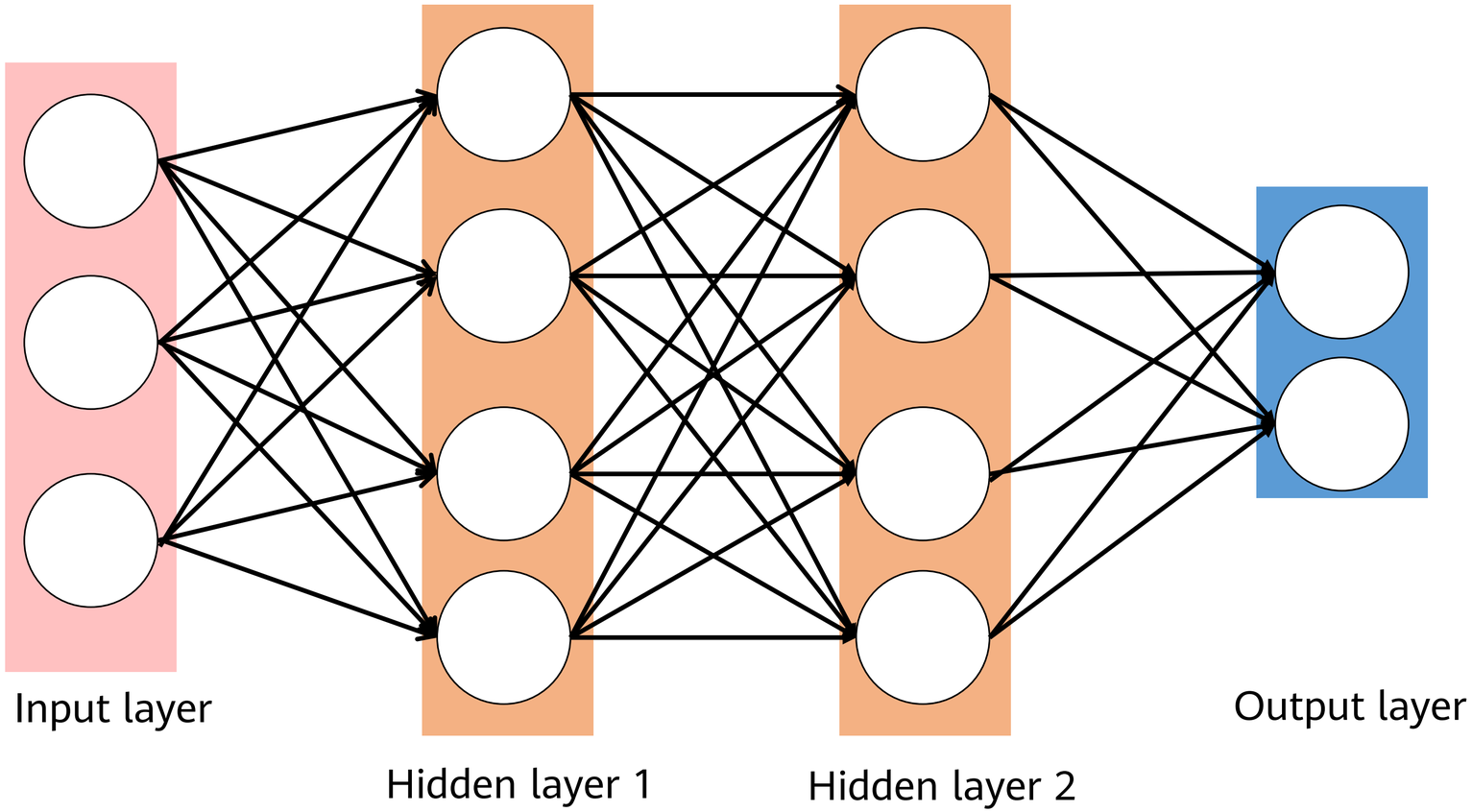


A (BENDITA) CAMADA OCULTA





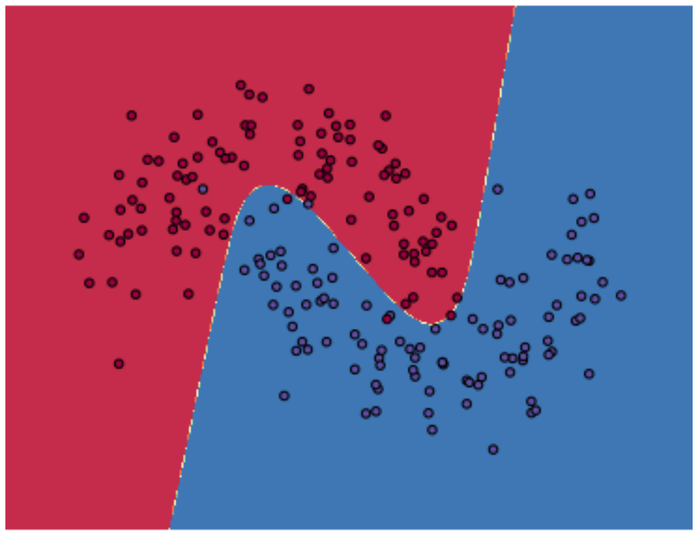
A (BENDITA) CAMADA OCULTA



MAIS CAMADAS... MAIS ESTRUTURAS COMPLEXAS PODEMOS GENERALIZAR

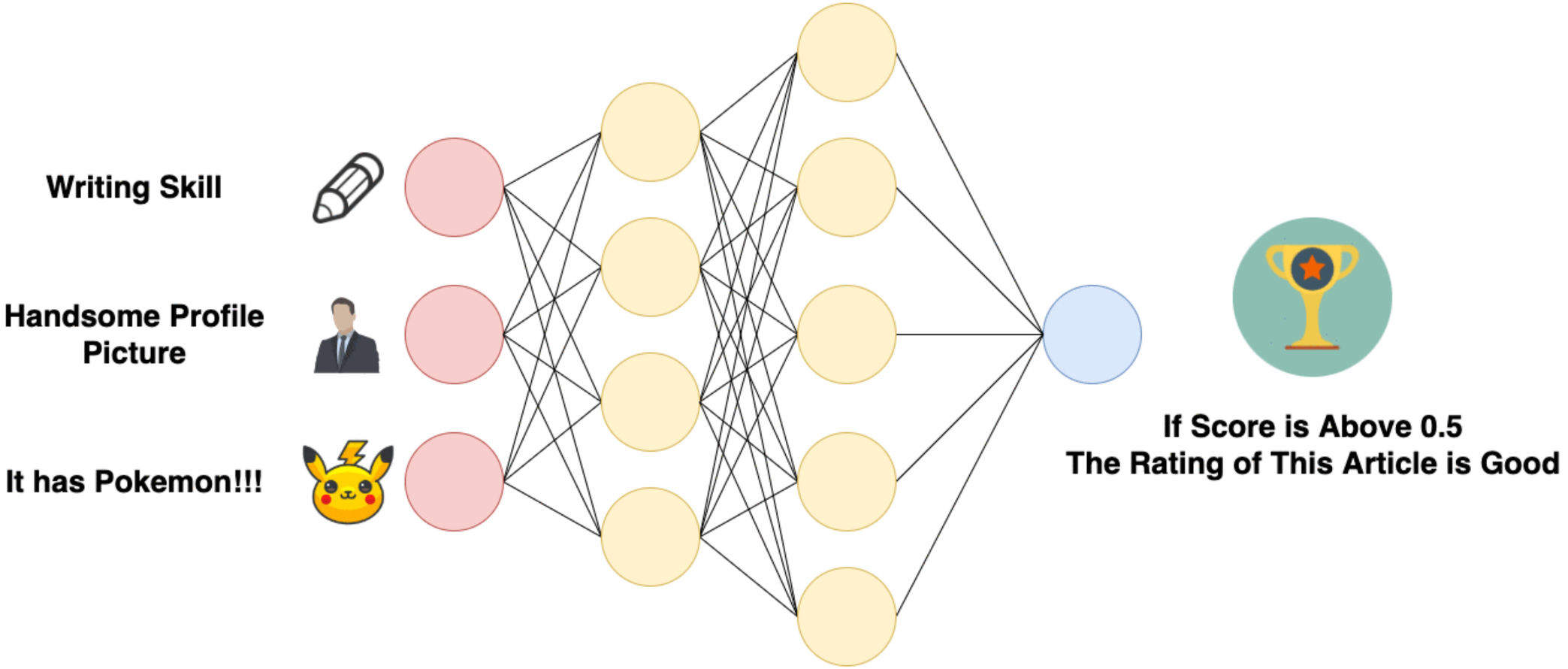
Uma camada

Três camadas



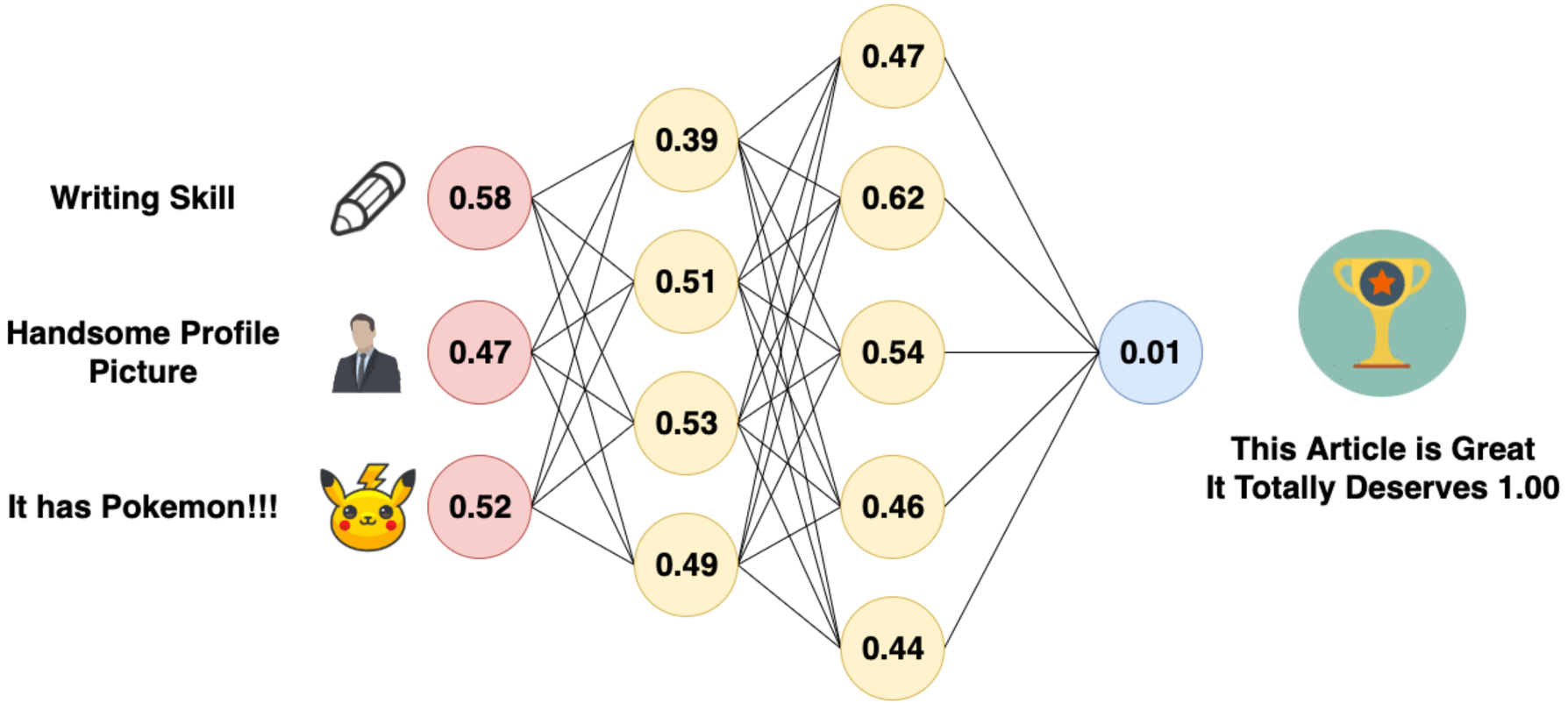
50 camadas !!!

SENTIDO DIRETO



Fonte:https://towardsdatascience.com/understanding-alphago-how-ai-thinks-and-learns-advanced-d70780744dae

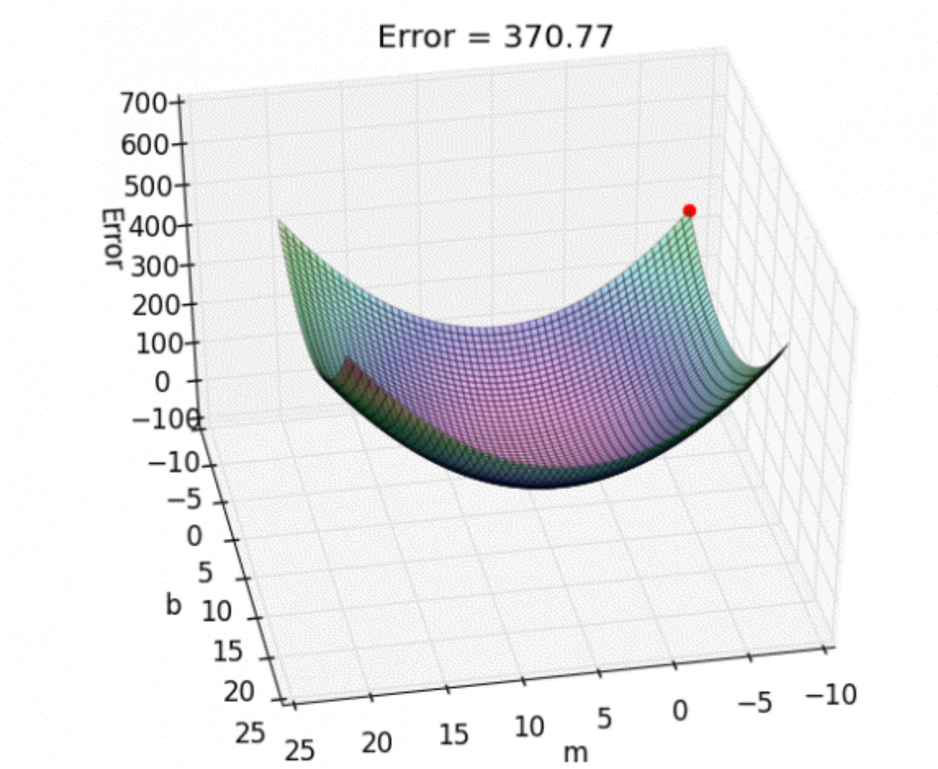
SENTIDO INVERSO

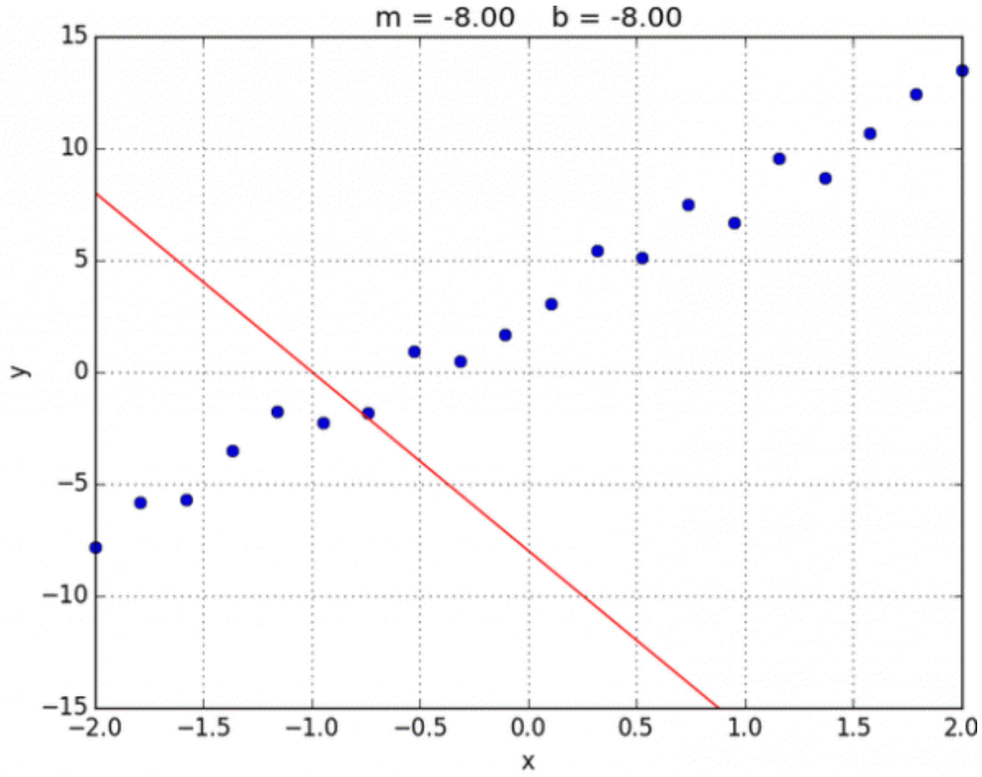


Fonte:https://towardsdatascience.com/understanding-alphago-how-ai-thinks-and-learns-advanced-d70780744dae

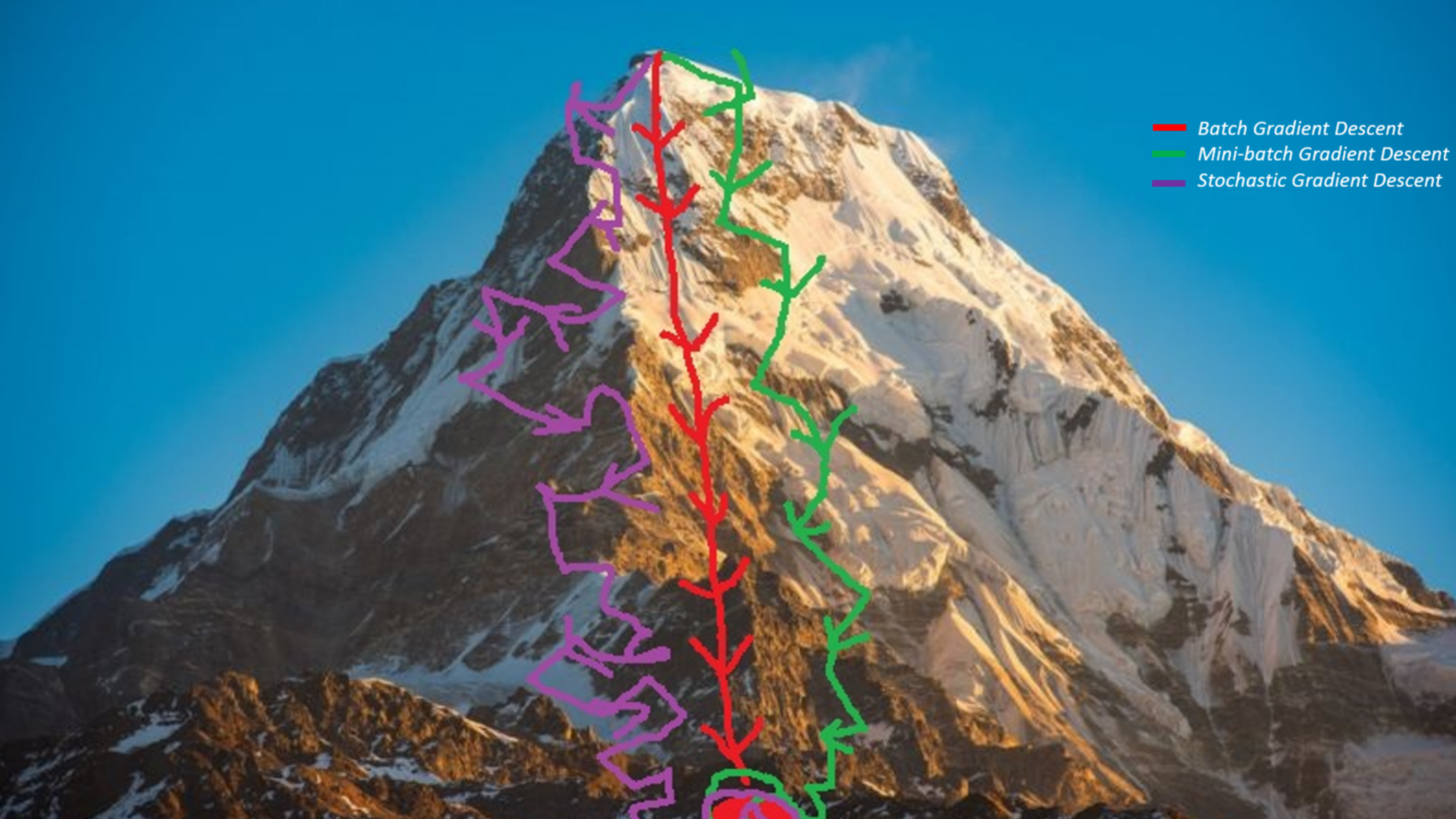
E O TREINAMENTO...?

TREINANDO COM DOIS PARÂMETROS

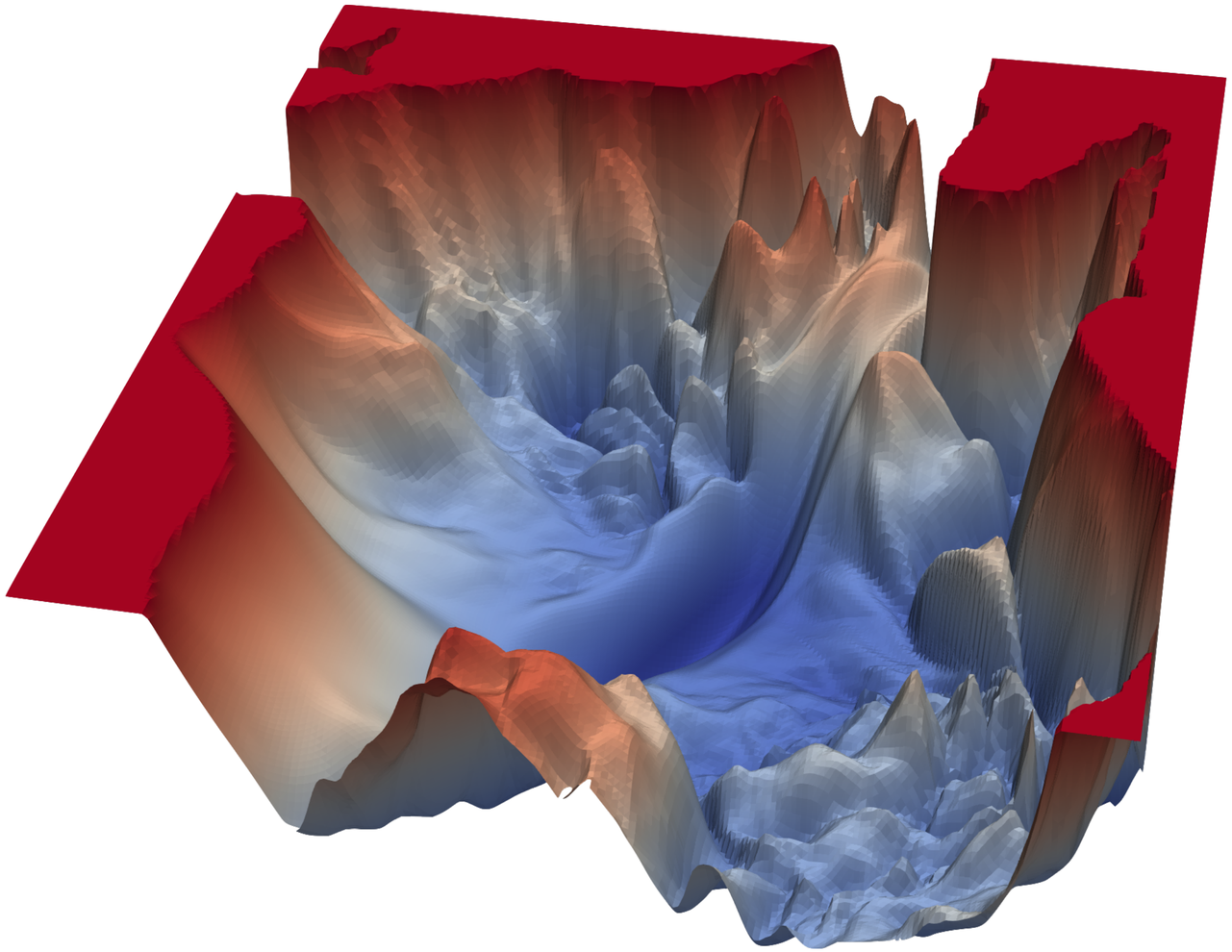




Fonte: https://gifer.com/en/NqNM


Batch Gradient Descent
Mini-batch Gradient Descent
Stochastic Gradient Descent

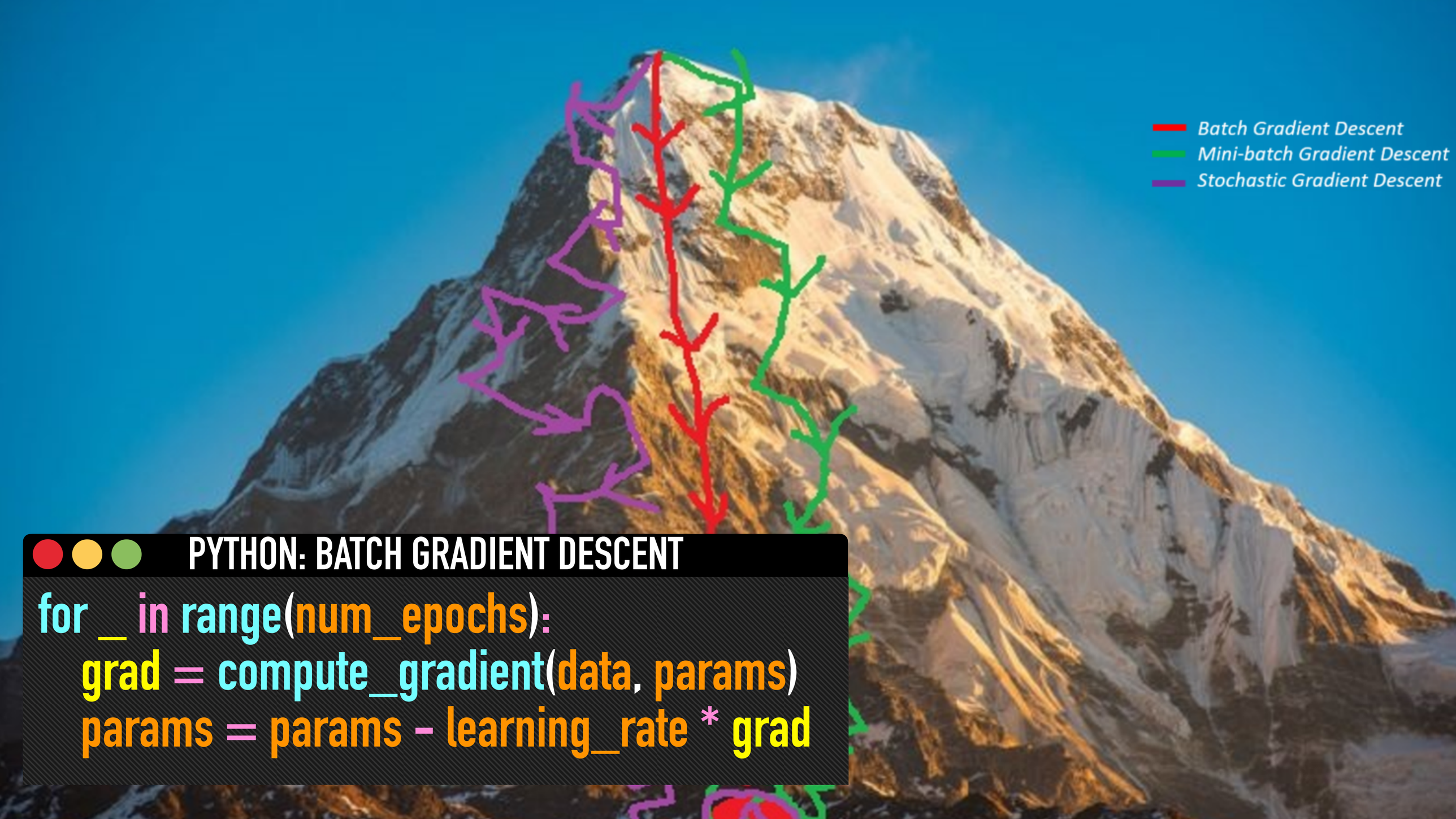
COMO É NA VERDADE... 🤯



- Devido à não linearidade introduzida pela adoção de funções de ativação, a função de perda de uma rede neural não é convexa;
- Muitos mínimos/máximos locais;
- “Alguns“ **mínimos/máximos globais**.

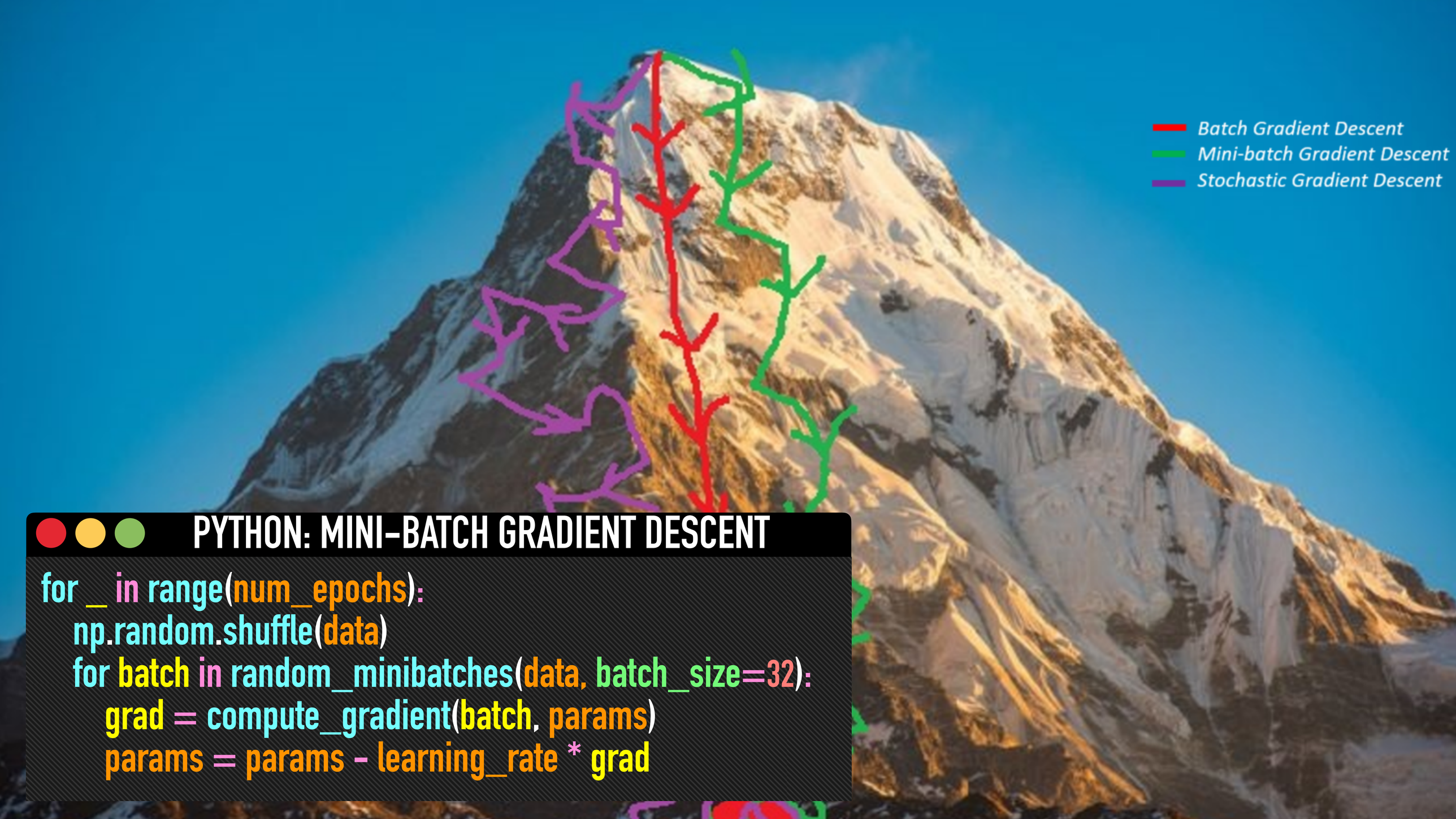
Fonte: LI, Hao et al. Visualizing the loss landscape of neural nets. arXiv preprint arXiv:1712.09913, 2017.

Fonte: https://ml4a.github.io/ml4a/how_neural_networks_are_trained/



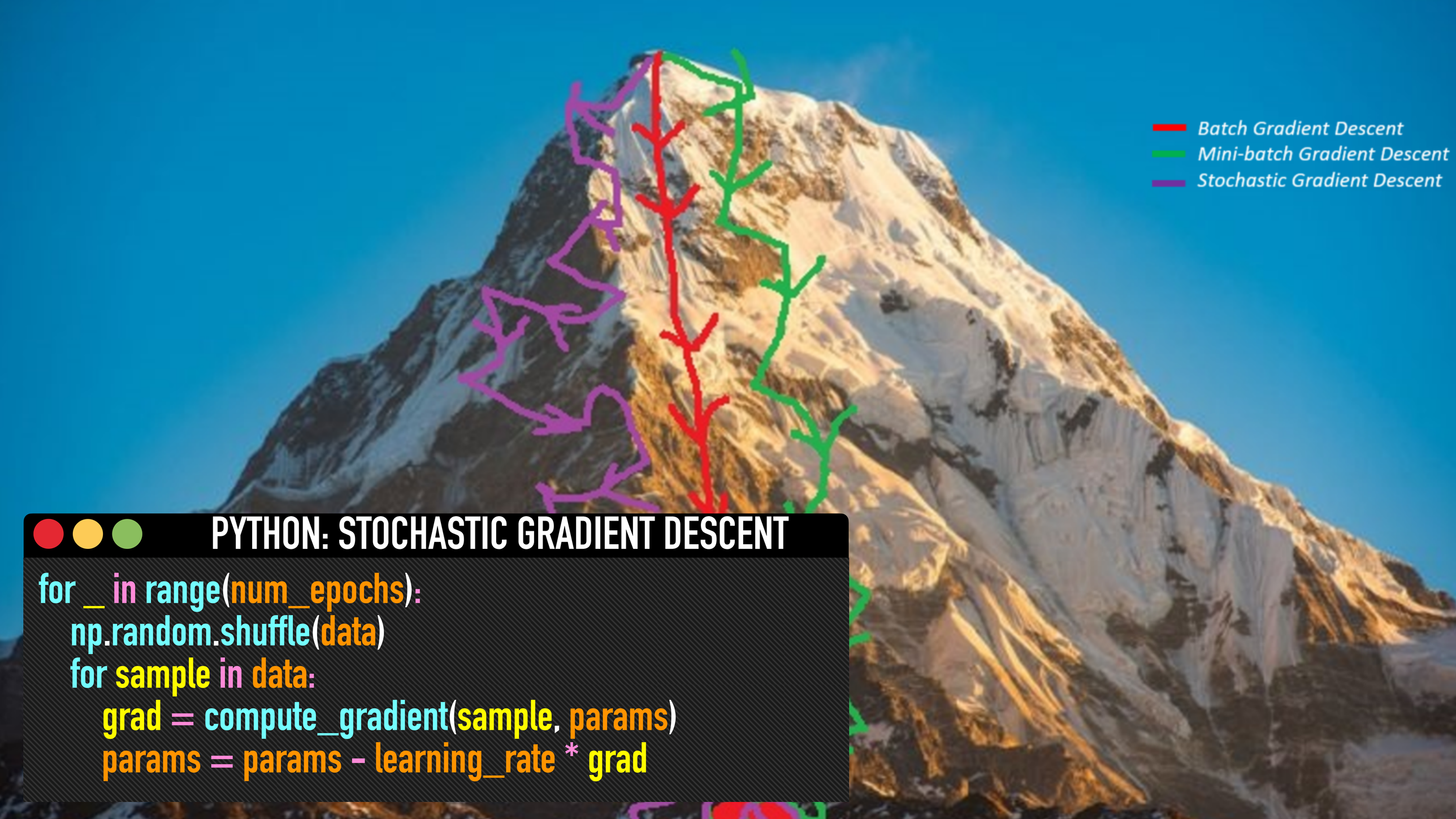
PYTHON: BATCH GRADIENT DESCENT

```
for _ in range(num_epochs):
    grad = compute_gradient(data, params)
    params = params - learning_rate * grad
```



PYTHON: MINI-BATCH GRADIENT DESCENT

```
for _ in range(num_epochs):
    np.random.shuffle(data)
    for batch in random_minibatches(data, batch_size=32):
        grad = compute_gradient(batch, params)
        params = params - learning_rate * grad
```

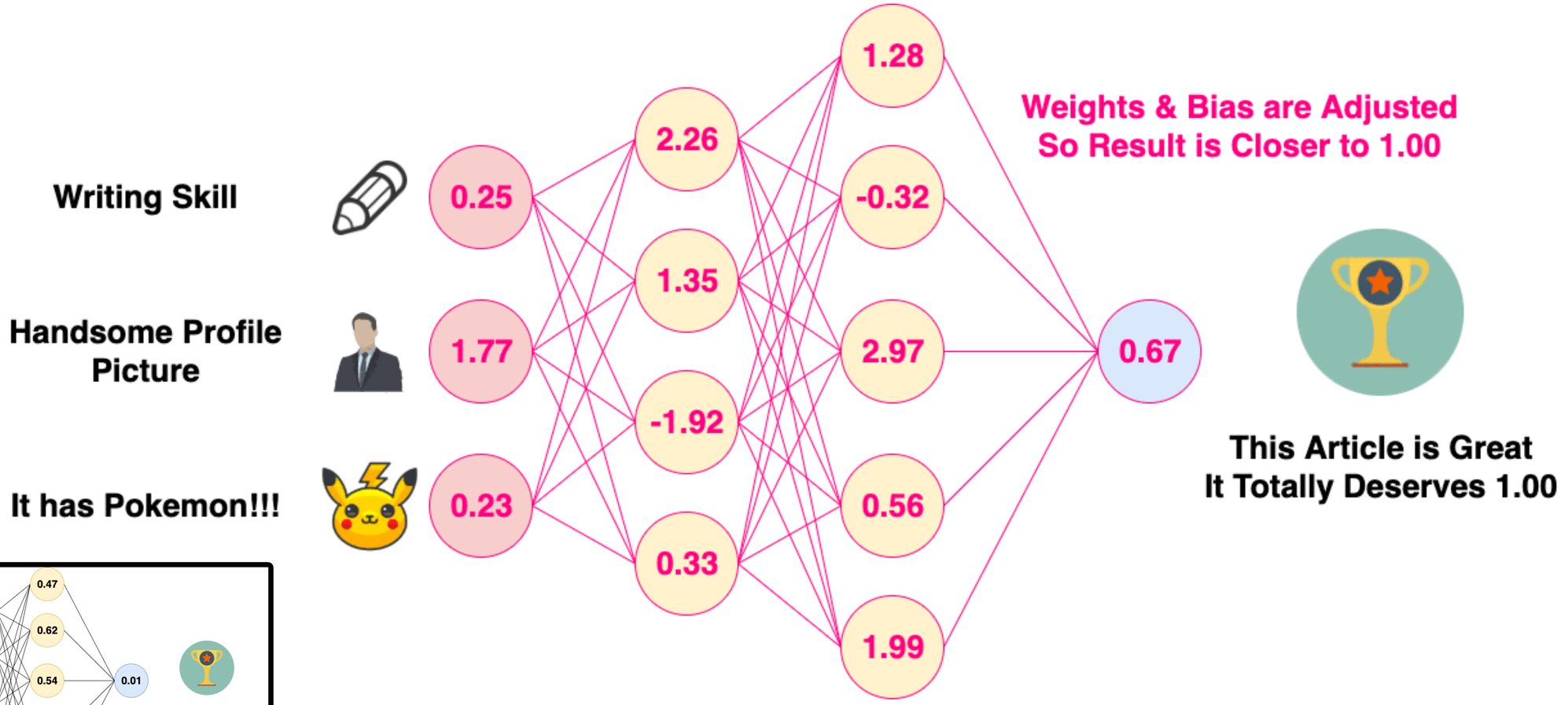


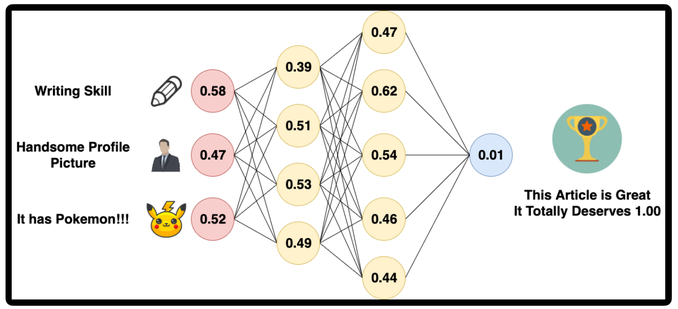
PYTHON: STOCHASTIC GRADIENT DESCENT

```
for _ in range(num_epochs):
    np.random.shuffle(data)
    for sample in data:
        grad = compute_gradient(sample, params)
        params = params - learning_rate * grad
```

RETROPROPAGAÇÃO DO ERRO

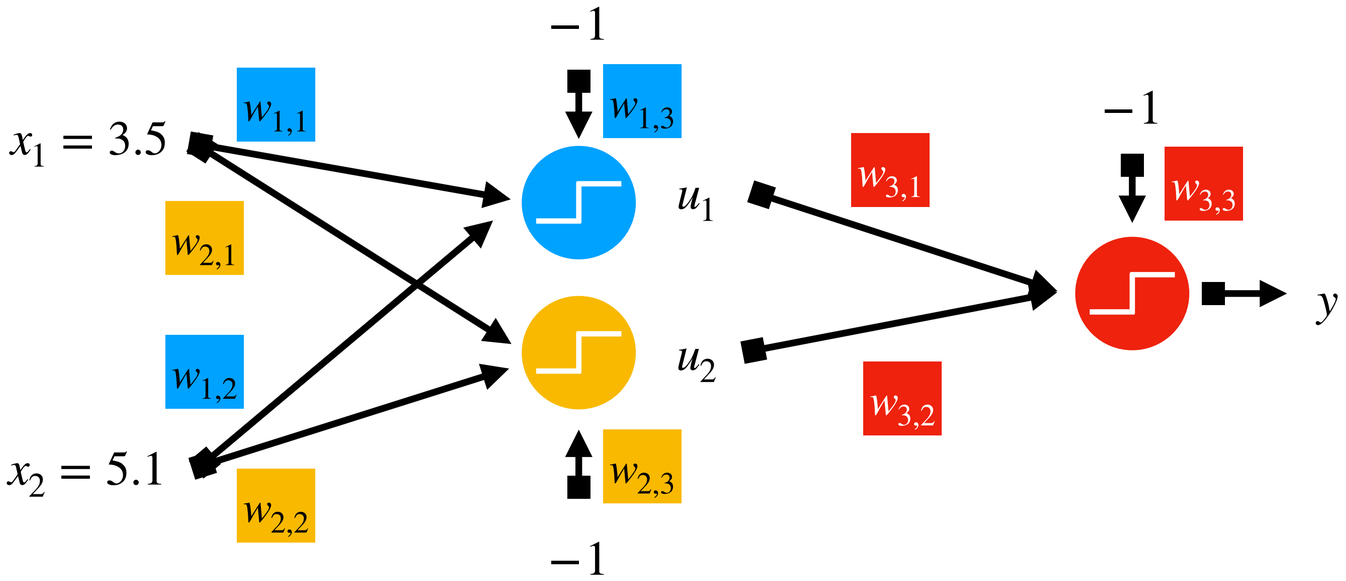
SENTIDO INVERSO





Fonte:https://towardsdatascience.com/understanding-alphago-how-ai-thinks-and-learns-advanced-d70780744dae

CALCULANDO O ERRO DA CAMADA OCULTA



Sabendo que:

a) $y = \varphi\left(w_{3,1}u_1 + w_{3,2}u_2 - w_{3,3}\right);$

b) $u_\star = \varphi\left(x_1 w_{\star,1}, + x_2 w_{\star,2} - w_{\star,3}\right)$

c) $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta e^{(t)} \varphi'\left(\mathbf{x}^\intercal \mathbf{w}^{(t)}\right)\mathbf{x};$

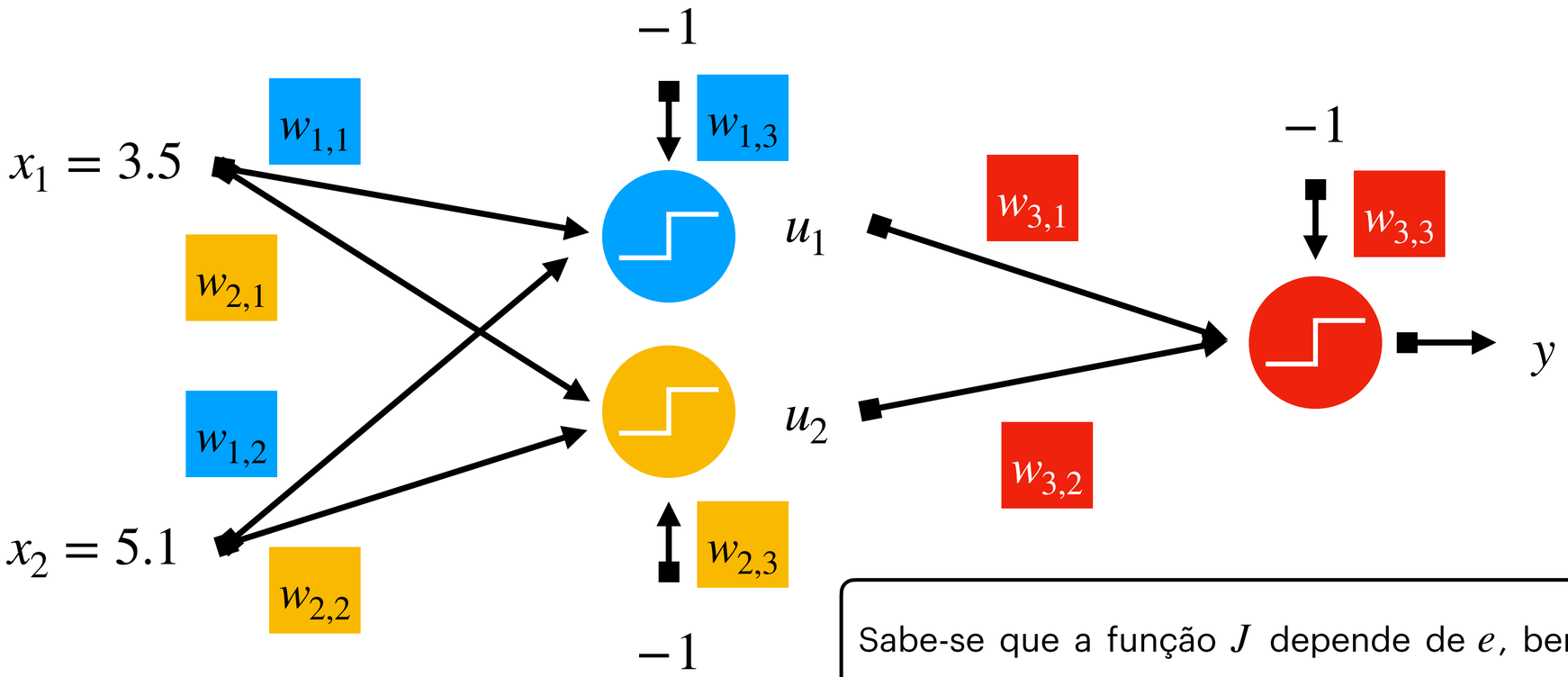
d) $\eta = 0.1;$

e) $\varphi'(u) = \varphi(u)(1 - \varphi(u));$

f) $\varphi(u) = \dfrac{1}{1 + \exp(-u)};$

ITER	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$	u_1	e_1	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$	u_2	e_2	$w_{3,1}$	$w_{3,2}$	$w_{3,3}$	y	e_3	d
0	0,18	0,49	0,81	0.74	**??**	0,43	0,82	0,79	0.87	**??**	0,75	0,13	0,24	**0.71**	**0.29**	1
1											**0.85**	**0.28**	**0.26**	**0.76**	**0.24**	1

CALCULANDO O ERRO DA CAMADA OCULTA



Sabe-se que a função J depende de e, bem como sabe-se que e depende de y e, ainda, sabe-se que y depende de φ que depende de $\mathbf{w}_3$, que, por fim, via $\mathbf{u}$, depende de $\mathbf{w}_2$ e $\mathbf{w}_1$, ambos dependentes de φ. Uffa! 🥵.

TREINANDO UMA MLP (Pulo do 🐈)

- Para fins de notação, tome w como os pesos da camada de saída, m como os pesos associados ao primeiro neurônio da camada oculta e z como os pesos associados ao último neurônio da camada oculta. Assim como $w, m, z \in \theta$, ou seja, θ é o conjunto de todos os pesos;
- Tendo $J = \frac{1}{2}\sum_{i=1}^{N}\left(d_i - \varphi_i\left(y_i\right)\right)^2 = \frac{1}{2}\sum_{i=1}^{N}\left(d_i - \varphi_i\left(\sum_{j=1}^{J}\varphi_j\left(u_j w_j\right)\right)\right)^2$, cada componente do gradiente da função custo ∇J pode ser obtida com base na regra da cadeia, a saber,

$$\frac{\partial J}{\partial \theta} = \frac{1}{2}\frac{\partial}{\partial \theta}\underbrace{\left(e_1^2 + e_2^2 + \ldots + e_N^2\right)}_{N \text{ erros}} = \frac{1}{2}\left(\frac{\partial e_1^2}{\partial \theta} + \frac{\partial e_2^2}{\partial \theta} + \ldots + \frac{\partial e_N^2}{\partial \theta}\right).$$

- Analisando separadamente os termos do somatório acima, tem-se que

$$\frac{\partial e_i^2}{\partial \theta} = \frac{\partial\left(d_i - \varphi(y_i)\right)^2}{\partial \theta} = (2)(-1)\underbrace{\left(d_i - \varphi(y_i)\right)}_{\text{erro, né?}}\left(0 - \varphi'(y_i)\frac{\partial y_i}{\partial \theta}\right) = 2\, e_i\, \varphi'(y_i)\, \frac{\partial y_i}{\partial \theta}.$$

TREINANDO UMA MLP (Pulo do 🐈)

- Mas quanto vale $\frac{\partial y_i}{\partial \theta}$?
- Sabendo que $y = \sum_{j=1}^{J} \varphi_j \left(u_j w_j \right)$, utilizamos o mesmo recurso da regra da cadeia. Assim,

$$\frac{\partial y_i}{\partial \theta} = \frac{\partial \left(\sum_{j=1}^{J} \varphi_j \left(u_j w_j \right) \right)}{\partial \theta} = \frac{\partial \varphi_j \left(u_1 w_1 \right)}{\partial \theta} + \frac{\partial \varphi_j \left(u_2 w_2 \right)}{\partial \theta} + \ldots + \frac{\partial \varphi_j \left(u_J w_J \right)}{\partial \theta}.$$

- Analisando separadamente os termos do somatório acima, tem-se que

$$\frac{\partial y_i}{\partial \theta} = \frac{\partial \varphi_j \left(u_j w_j \right)}{\partial \theta} = w_j \frac{\partial \varphi_j(u_j)}{\partial \theta} = w_j \, \varphi_j'(u_j) \frac{\partial u_j}{\partial \theta}.$$

- Como u_j, é o resultado da soma ponderada das entradas conectadas ao neurônio da camada oculta, podemos reescrever $\frac{\partial y_i}{\partial \theta}$ como $\frac{\partial y_i}{\partial \theta} = w_j \, \varphi_i'(u_i) \, x_k$.

– CANSADOS?

– CANSADOS?
– AINDA NÃO ACABOU!

TREINANDO UMA MLP (Pulo do 🐈)

- Conectando tudo, a regra de aprendizagem para os neurônios da de uma única camada oculta é dada por:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta\varphi'\left(\mathbf{u}^{(t)}\right)\left(\sum_{j=1}^{J} e_j^{(t)}\,\varphi'\left(y_j^{(t)}\right)\,w_j^{(t)}\right)\mathbf{x},$$

em que $\mathbf{u}$ é a saída da camada oculta anterior.

Referências

- RODRIGUES, JARDEL. **Aula 05**: Redes Neurais Multicamadas. IFCE *campus* Jaguaribe, 2021.
- PÁDUA BRAGA, Antônio; DE LEON FERREIRA, André Carlos Ponce; LUDERMIR, Teresa Bernarda. **Redes neurais artificiais**: teoria e aplicações. LTC editora, 2007.
- Richard O. Duda, Peter E. Hart, David G. Stork. **Pattern Classification**. John Wiley & Sons, 2012.