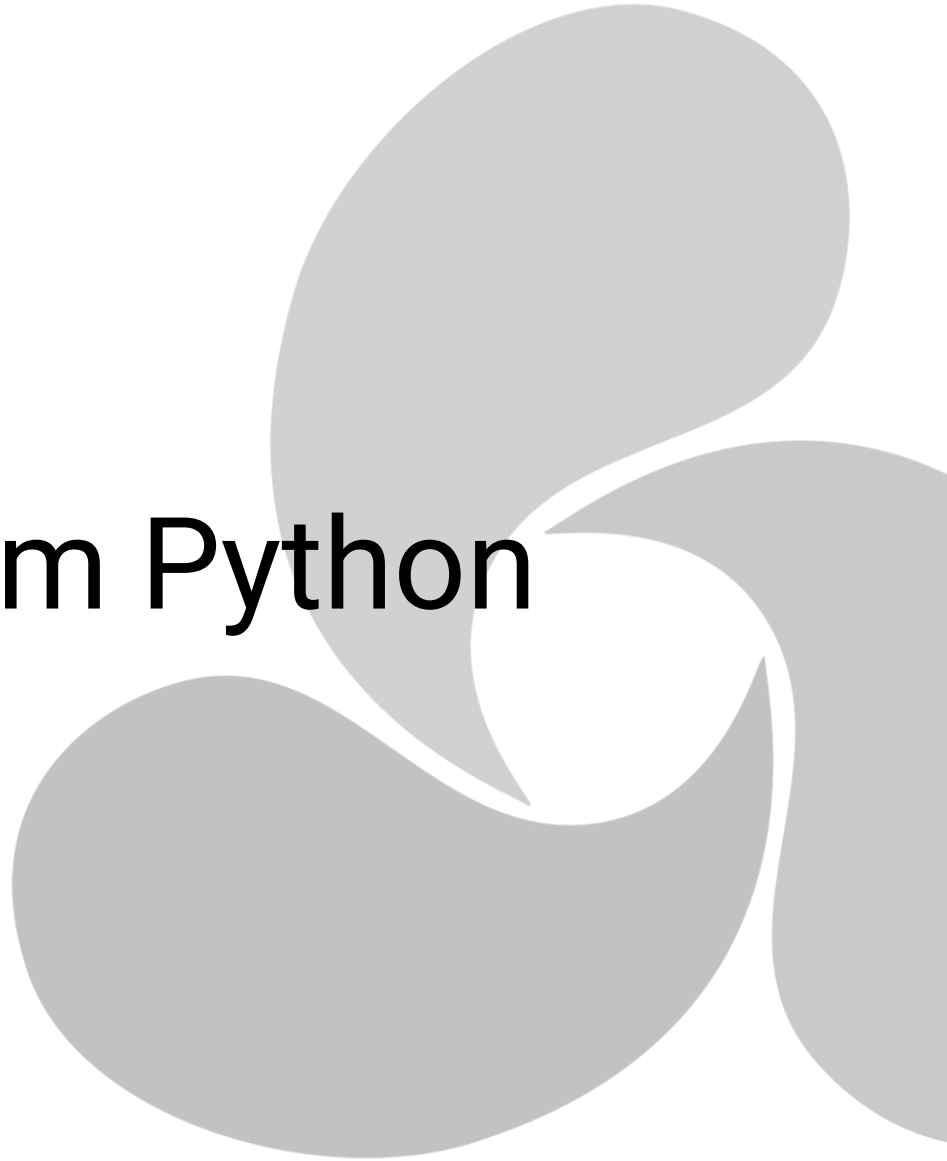
Machine Learning:
uma introdução prática em Python

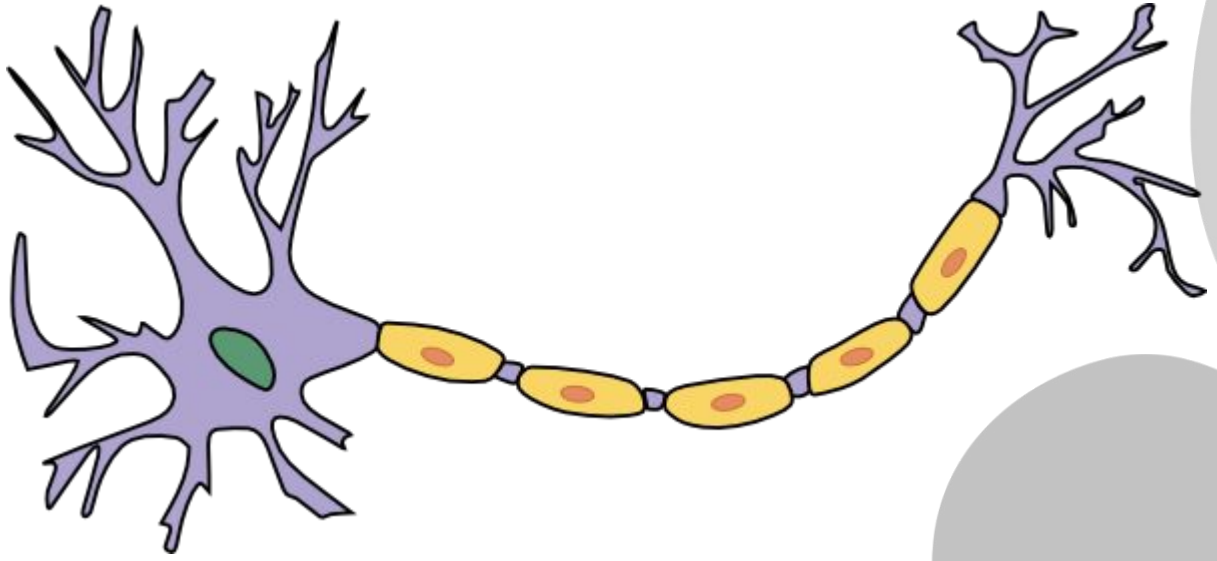
Otacílio “Minho” Neto, minhotmog@gmail.com



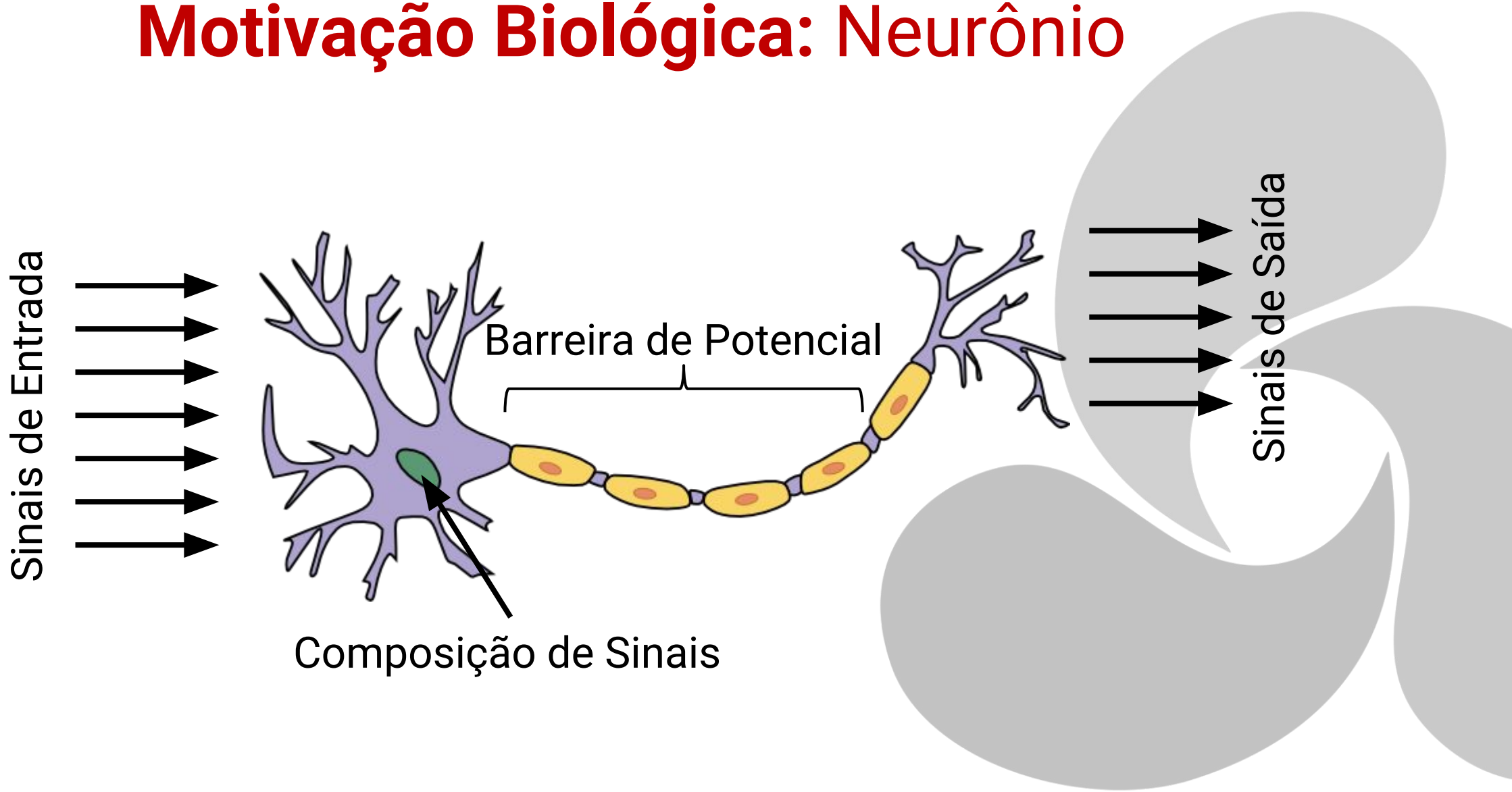
Redes Neurais:
Single-Layer Perceptron

A Rede Neural mais simples que existe!

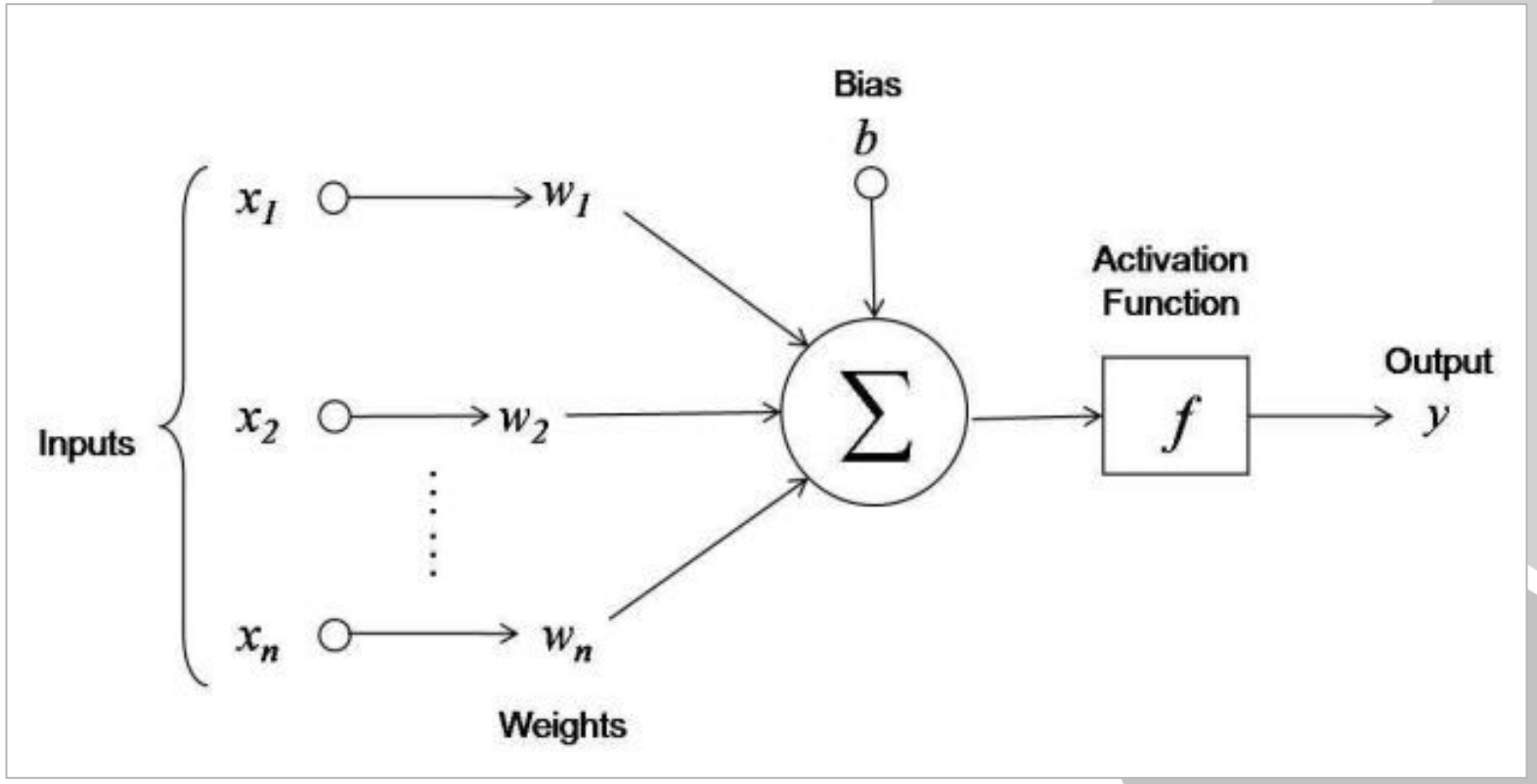
Motivação Biológica: Neurônio



Motivação Biológica: Neurônio



Modelo de McCulloch & Pitts



Fonte: https://blogs.cornell.edu/

Funcionamento de um Neurônio

1. Os sinais de entrada são multiplicados por pesos e somados (*Weighted Sum Function*).

$$s_{net} = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

2. A soma é então passada para uma função de ativação (*Activation Function*).

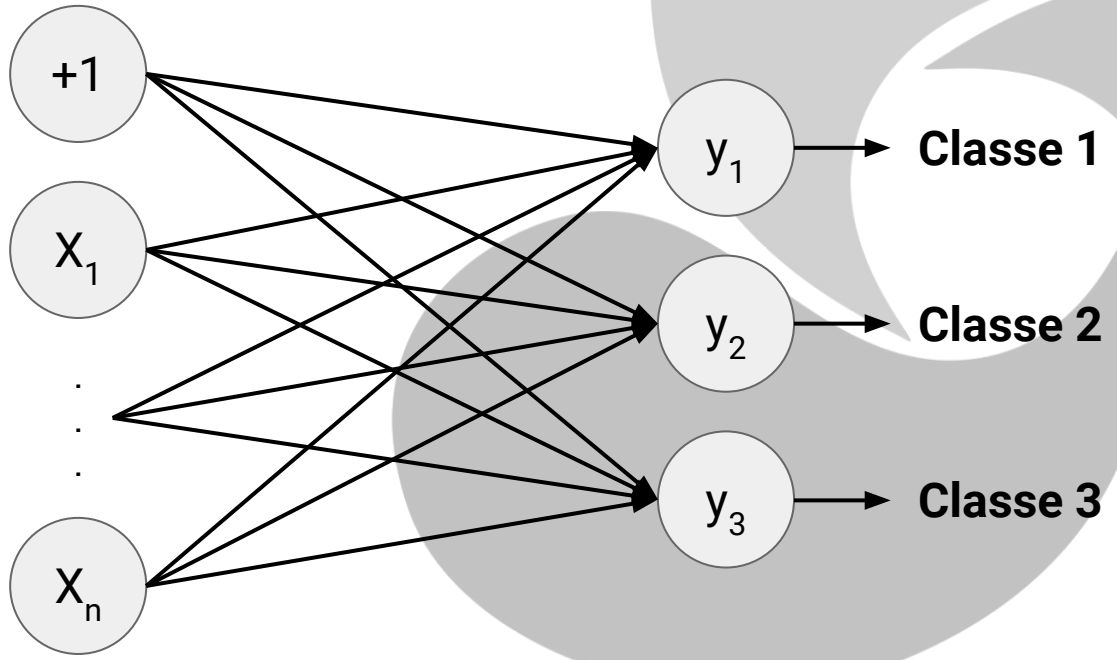
$$\varphi(s_{net}) = \begin{cases} 1 & \text{se } s_{net} \geq \tau \\ 0 & \text{caso contrario} \end{cases}$$

Single-Layer Perceptron

- O **Single-Layer Perceptron** é a arquitetura de Rede Neural Artificial mais simples. Consiste apenas em uma camada densa diretamente ligado às classes de saída. Por esse motivo, seu funcionamento é idêntico à **Regressão Logística**.

SLP para Multiclass Classification

$$\Theta = \begin{bmatrix} \Theta_{00} & \Theta_{10} & \cdots & \Theta_{n0} \\ \Theta_{01} & \Theta_{11} & \cdots & \Theta_{n1} \\ \vdots & \vdots & \ddots & \vdots \\ \Theta_{0n} & \Theta_{1n} & \cdots & \Theta_{nn} \end{bmatrix}$$



Classificação Multiclass

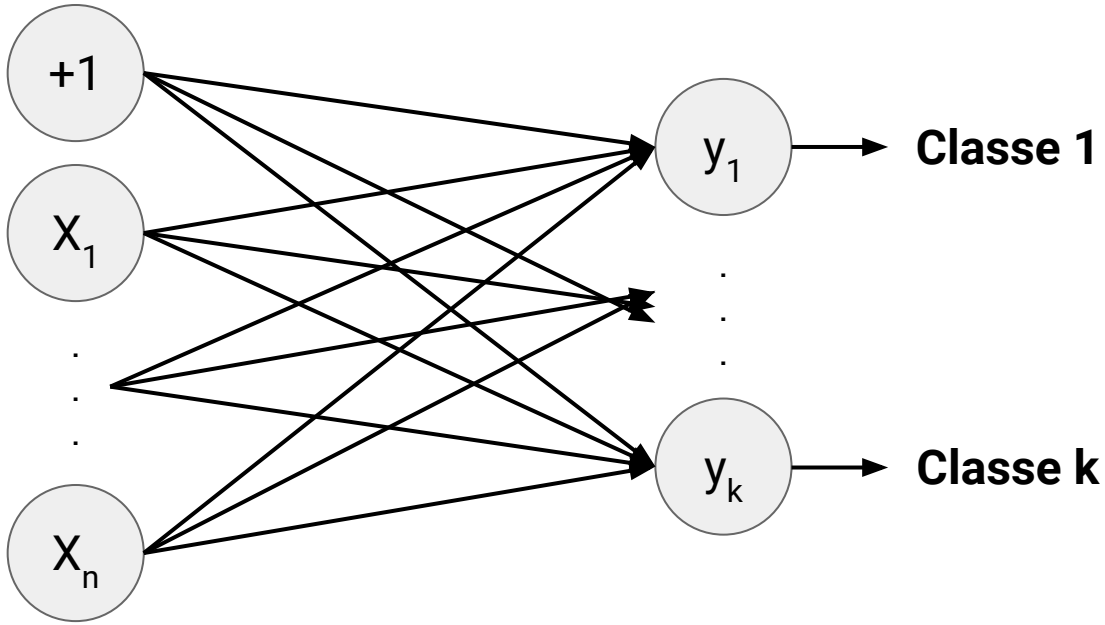
- Nossa Rede Neural irá agregar "*uma Regressão Logística por classe*". Por isso seus parâmetros são matriciais. Essa técnica chama-se "*one-vs-all*";

- Classes são enumeradas no Dataset, mas representadas para a Rede Neural utilizando a notação "*one-hot encoding*":

Exemplar	Classe	Numeração
1	Carro	1
2	Pessoa	2
3	Objeto	3
4	Pessoa	2

Exemplar	Carro	Pessoa	Objeto
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

Resumindo: SLP



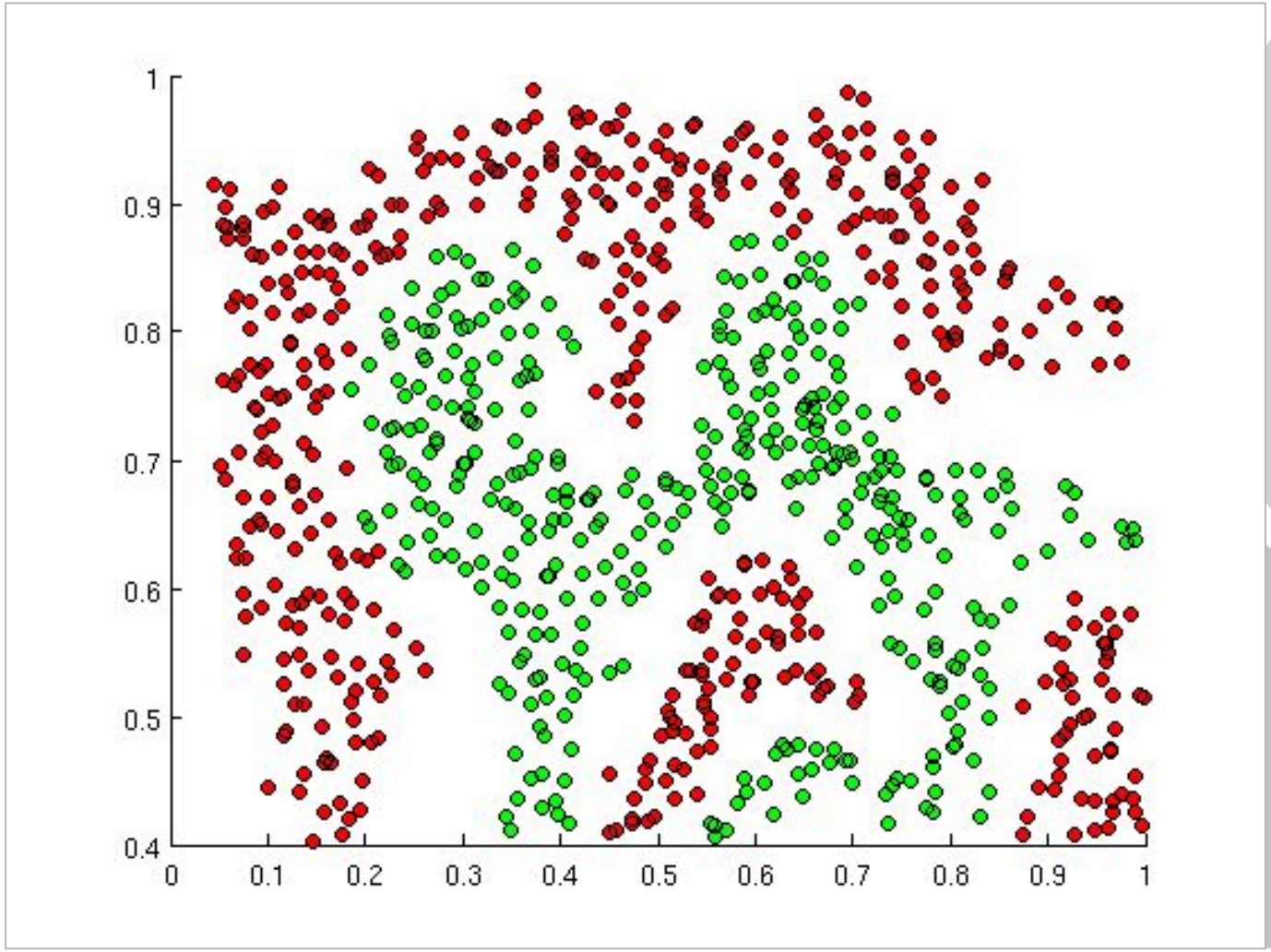
$$X = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ X_1^{(1)} & X_1^{(2)} & \cdots & X_1^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ X_n^{(1)} & X_n^{(2)} & \cdots & X_n^{(m)} \end{bmatrix}; \Theta = \begin{bmatrix} \Theta_{00} & \Theta_{10} & \cdots & \Theta_{k0} \\ \Theta_{01} & \Theta_{11} & \cdots & \Theta_{k1} \\ \vdots & \vdots & \ddots & \vdots \\ \Theta_{0n} & \Theta_{1n} & \cdots & \Theta_{kn} \end{bmatrix} \quad y = \begin{bmatrix} \widehat{y}_1^{(1)} & \widehat{y}_1^{(2)} & \cdots & \widehat{y}_1^{(m)} \\ \widehat{y}_2^{(1)} & \widehat{y}_2^{(2)} & \cdots & \widehat{y}_2^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{y}_k^{(1)} & \widehat{y}_k^{(2)} & \cdots & \widehat{y}_k^{(m)} \end{bmatrix}$$

Hands-On!!!

Redes Neurais:
Multi-Layer Perceptron

Melhorando a qualidade das nossas Linhas de Decisão

Desenhe seu Hiperplano Separador!



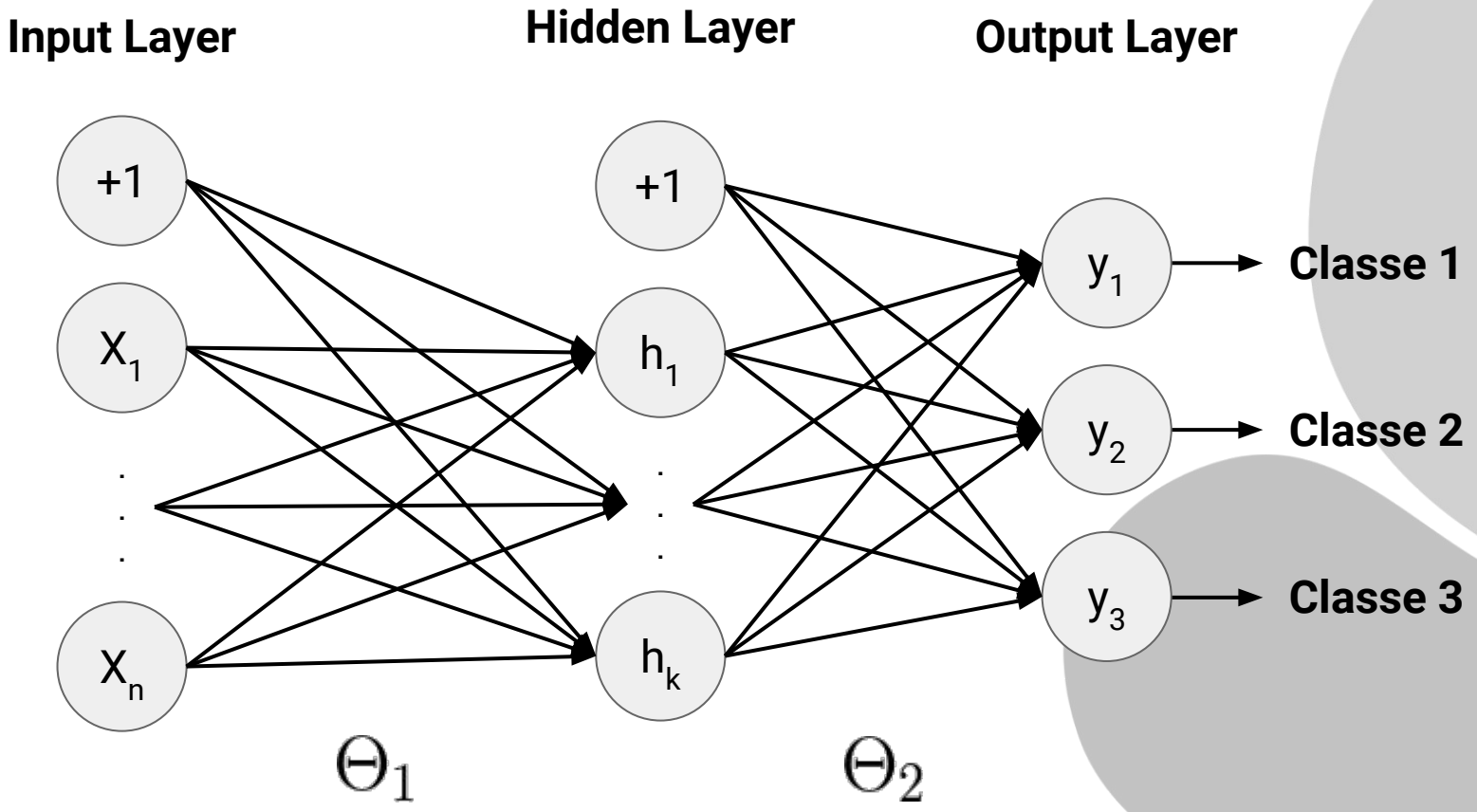
Transformação Não-Linear

- Existem casos interessantes de *Classificação* onde a disposição dos exemplos não permitem uma separação por hiperplanos lineares. Dizemos, então, que os "*dados não são linearmente separáveis*".

- Semelhante à *Regressão Polinomial*, devemos aumentar a complexidade do modelo. No entanto, só criar versões polinomiais não será suficiente.

- As Redes Neurais permitem não só uma transformação não-linear; elas permitem que o **próprio algoritmo defina os parâmetros da transformação!**

Multi-Layer Perceptron



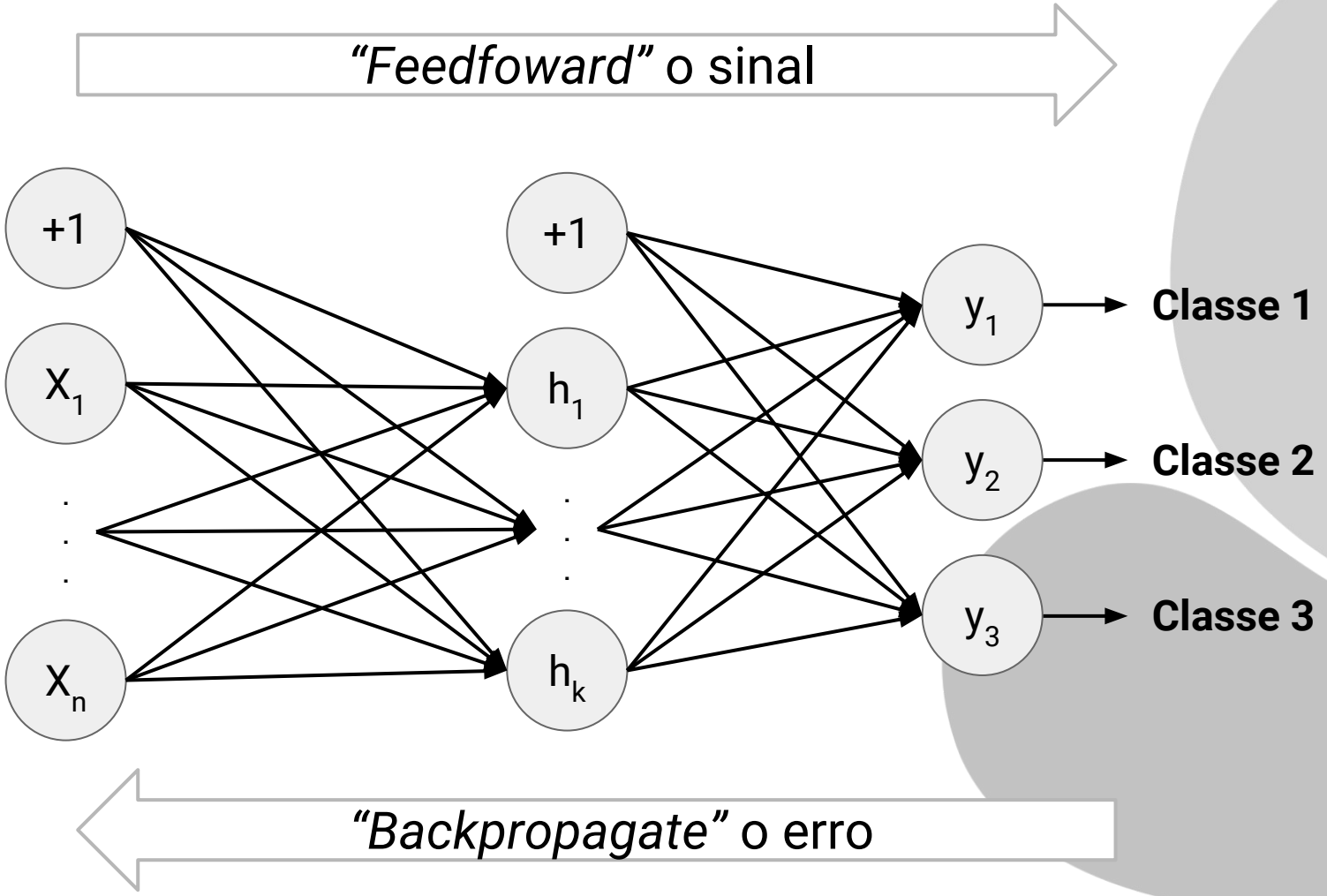
Representação MLP

- A representação do nosso modelo consiste nas duas matrizes de parâmetros:
  - Θ_1 : mapeia os pesos da *Input Layer* para a *Hidden Layer*;
  - Θ_2 : mapeia os pesos da *Hidden Layer* para a *Output Layer*;

- Realizaremos duas ativações para realizar nossa predição. Podemos, então, utilizar o Gradiente Descendente para o treinamento? **Não é tão simples...**

- **Camada de Saída:** erro supervisionado. **Camada Oculta:** erro aproximativo.

Backpropagation



Backpropagation

- **Erro de Predição:**

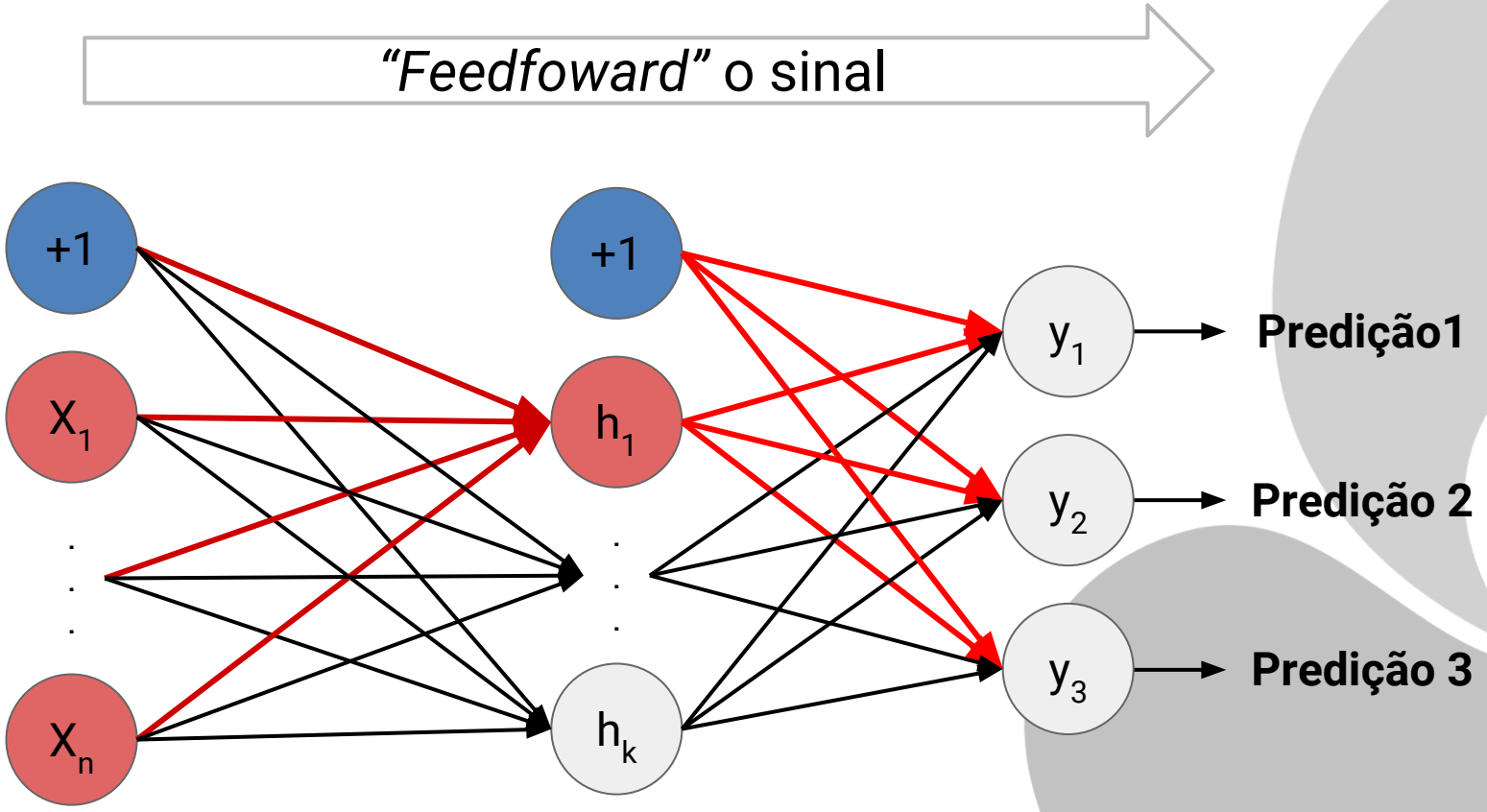
$$\nabla\Theta_2 = \frac{\delta}{\delta\Theta_2}(\frac{1}{2m}\sum(g(\Theta_2^T H) - y)^2)$$
$$= \frac{1}{m}\sum(g(\Theta^T H) - y) \ .* \ H$$

Backpropagation

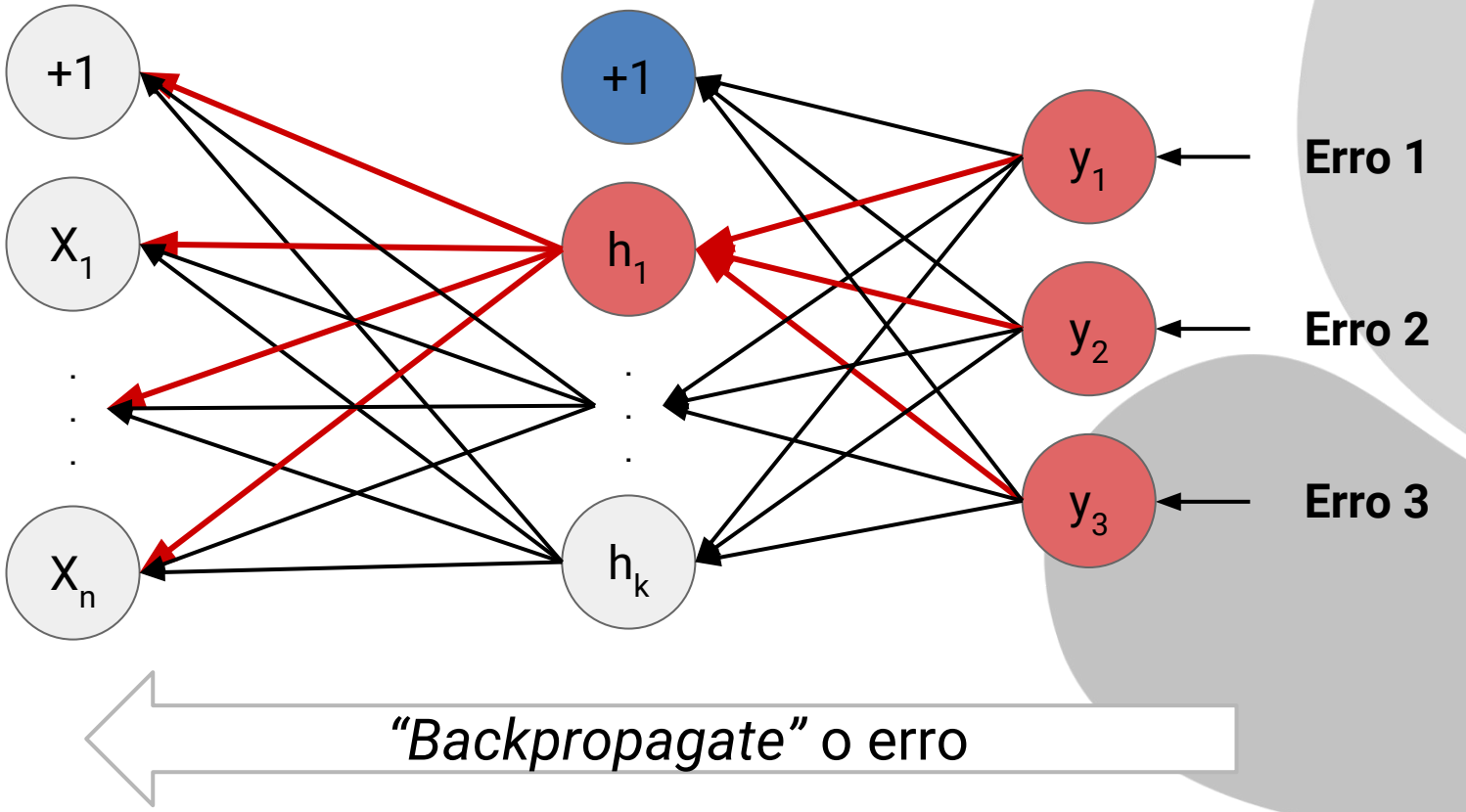
- **Erro de Aproximação (Transformação Não-Linear):**

$$\nabla\Theta_2 = \frac{\delta}{\delta\Theta_2}\left(\frac{1}{2m}\sum(g(\Theta_2^T g(\Theta_1^T X)) - y)^2\right)$$
$$= \frac{1}{m}\sum(\Theta_2^T(\Theta_2^T g(\Theta_1^T X) - y) \;.*\; (\Theta_2^T g(\Theta_1^T X)) \;.*\; (1 - \Theta_2^T g(\Theta_1^T X))$$

Backpropagation (Ex.)



Backpropagation



Backpropagation (Algoritmo)

1. Criar matriz Δ^L que representa o gradiente de Θ^L
2. Realizar Feedfoward Propagation:

   $$A_1 = \Theta_1^T X \quad e \quad A_2 = \Theta_2^T A1$$

3. Realizar o Backpropagation:

   $$\delta^{(2)} = A2 - y \quad e \quad \delta^{(1)} = ((\Theta^2)^T \delta^2) .* (A2 .* (1 - A2))$$

   $$\Delta^1 = \delta^{(1)} A1^T \quad e \quad \Delta^2 = \delta^{(2)} A2^T$$

4. Realizar o Gradient Descendente:

   $$\Theta_j^{(i)} = \Theta_j^{(i)} - \alpha \Delta_j^i$$

Hands-On!!!

Conclusão do Curso

Muito obrigado pela presença! :)

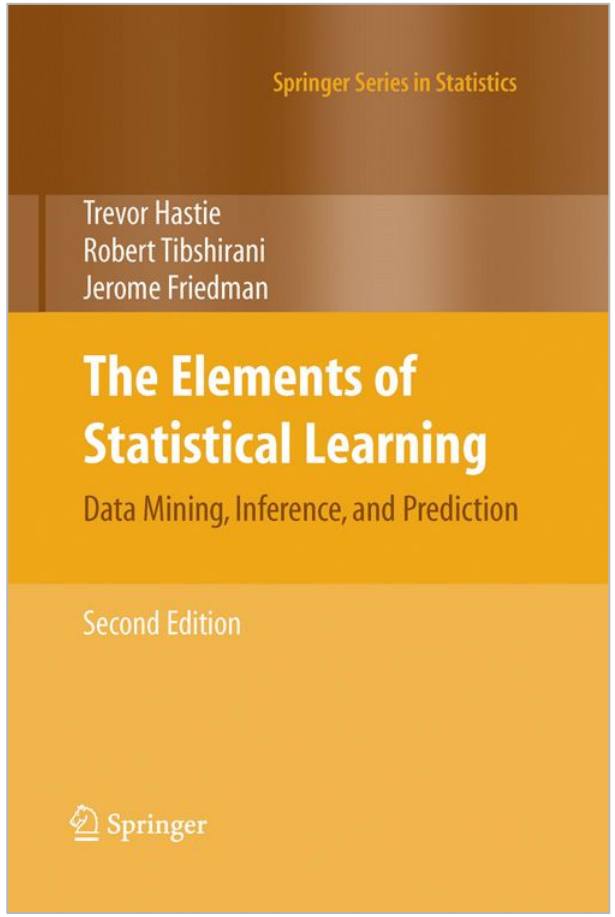
Cursos Online

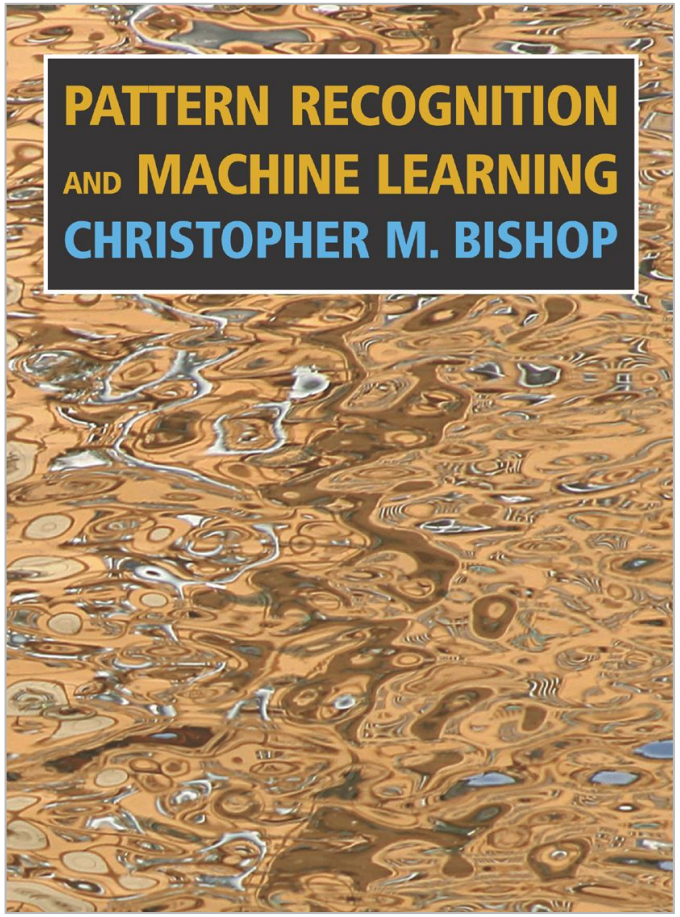
Machine Learning (Andrew Ng)
Link: https://www.coursera.org/learn/machine-learning

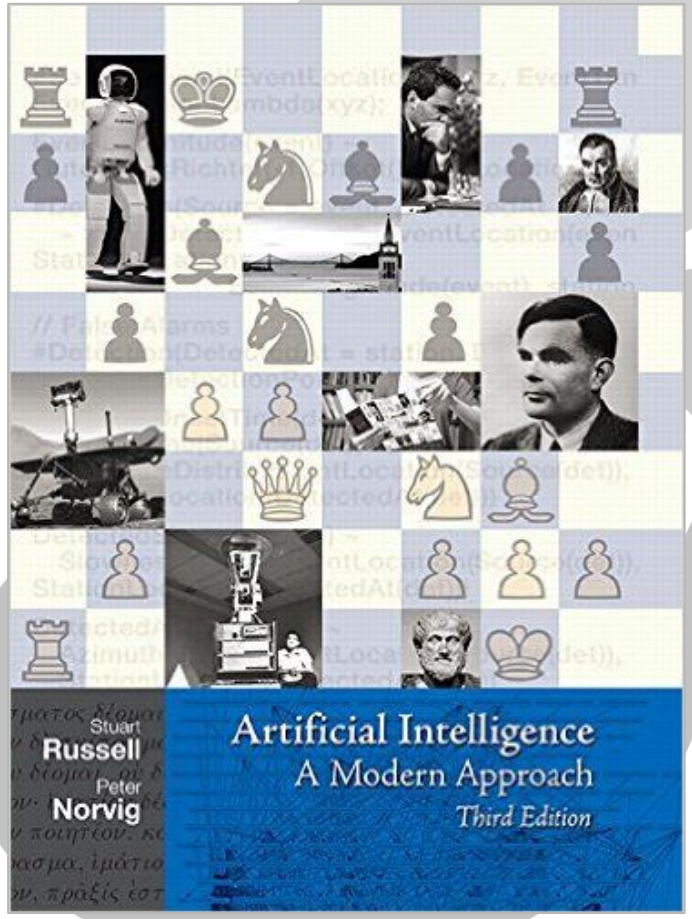
Neural Networks for Machine Learning (Geoffrey Hinton)
Link: https://www.coursera.org/learn/neural-networks

Machine Learning (Columbia)
Link: www.edx.org/course/machine-learning-columbiax-csmm-102x-0

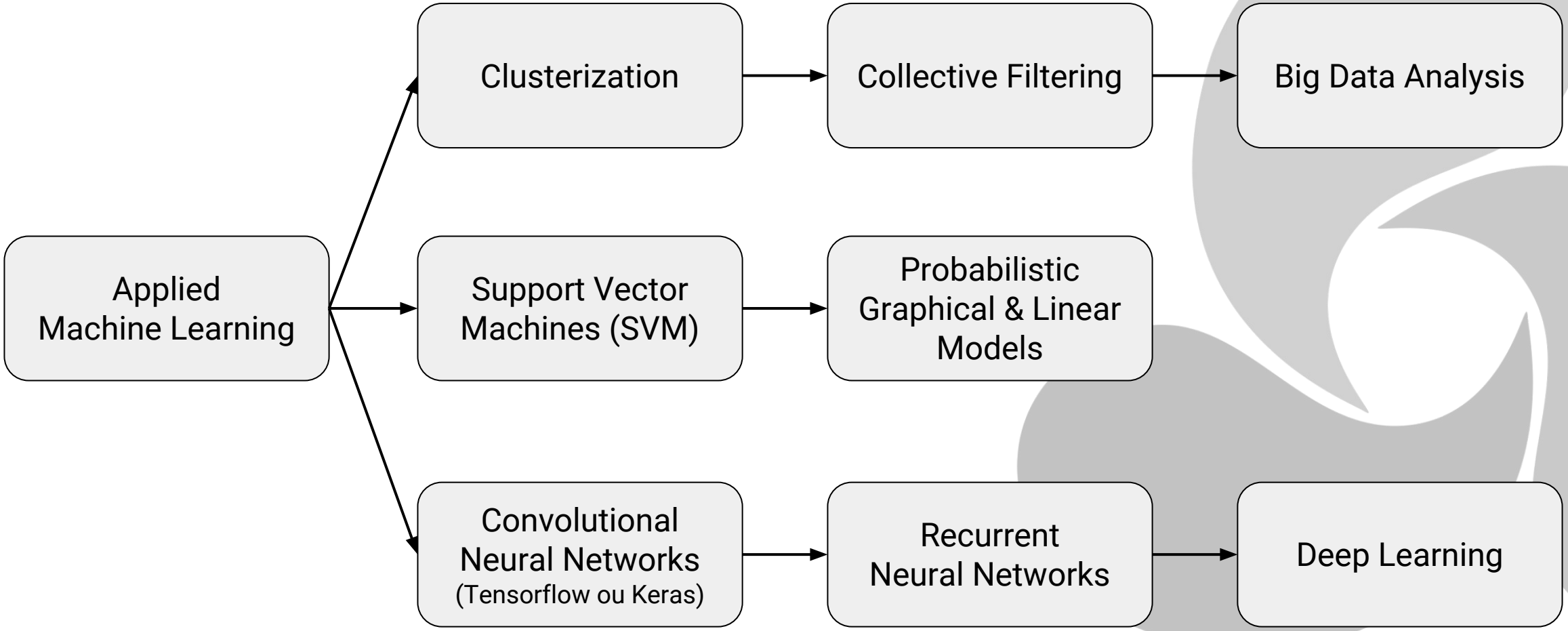
Livros


Springer Series in Statistics
Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of Statistical Learning
Data Mining, Inference, and Prediction
Second Edition
Springer


PATTERN RECOGNITION
AND MACHINE LEARNING
CHRISTOPHER M. BISHOP


Stuart
Russell
Peter
Norvig
Artificial Intelligence
A Modern Approach
Third Edition

Roteiro de Estudo



Obrigado!

minhotmog@gmail.com

www.petcomp.ufc.br