

Machine Learning

*Introdução e aplicação na engenharia
aeronáutica*

Renato Cosin

**Mini curso ministrado na
XVIII SEA – EESC-USP**

<https://sea.eesc.usp.br>



Agenda

☉ Parte 1: Conceitos

- Breve Introdução ao Machine Learning
 - ☉ O que é Machine Learning
 - ☉ Aprendizagem supervisionada e não supervisionada
 - ☉ Classificação e regressão
- Workflow de desenvolvimento de Machine Learning
- Redes Neurais
 - ☉ Uma rede neural muito simples
 - ☉ Como treinar sua rede neural

Agenda

- ◎ Parte 2: Prática
 - Preparação dos dados
 - ◎ Introdução ao Pandas
 - ◎ Tratando dados no Pandas
 - Introdução ao Tensorflow/Keras
 - ◎ A lógica de funcionamento do Tensorflow
 - ◎ “Hello world” no Tensorflow
 - ◎ Introdução ao Keras
 - Exemplo prático de rede neural: Metamodelo de performance de uma aeronave
 - ◎ Montagem do modelo no Keras
 - ◎ Treinamento do modelo
 - ◎ Análise do resultados

Notas

- © Todos os dados apresentados neste curso foram gerados para fins exclusivamente didáticos e não se referem a qualquer aeronave real ou informação proprietária de qualquer espécie.
- © Todas as metodologias, algoritmos e software utilizados são disponíveis abertamente
- © O material deste curso, incluindo os códigos e notebooks, será disponibilizado para que possam estudar os exemplos no futuro
- © A execução dos exemplos requer os seguintes softwares free / open source:
 - Anaconda
 - Python 3
 - Conda – gerenciador de ambientes
 - Numpy
 - Matplotlib
 - Pandas
 - Seaborn
 - Jupyter notebook
 - Tensorflow (com ou sem suporte a GPU)
- © O crédito das imagens será referenciado ao final do curso

Prefácio

Existem milhares de cursos on-line sobre Machine Learning. Fazer algo diferente foi realmente um desafio. O grande balizador para alcançar este objetivo foi a aplicação na engenharia aeronáutica, que é a razão deste curso fazer parte da SEA.

Neste curso abordaremos conceitos básicos de Machine Learning, Redes Neurais e um exemplo pertinente a atuação do engenheiro aeronáutico, mais especificamente no desenvolvimento de aeronaves.

O tema abordado é um universo por si só e cobrir todos os aspectos, se é que isso fosse possível, seria uma tarefa para 6 anos e não 6 horas. Assim, meu objetivo será focar em um exemplo, um algoritmo e um framework, de forma que exista continuidade e que permita uma aplicação efetiva deste conteúdo no futuro.

Machine Learning possui infinitas aplicações e cabe a criatividade de vocês desbravarem este potencial. Meu maior objetivo é trazer inspiração e uma base introdutória para que ingressem no universo do Machine Learning, seja ele aplicado a engenharia ou qualquer outra área

Bom estudo a todos,

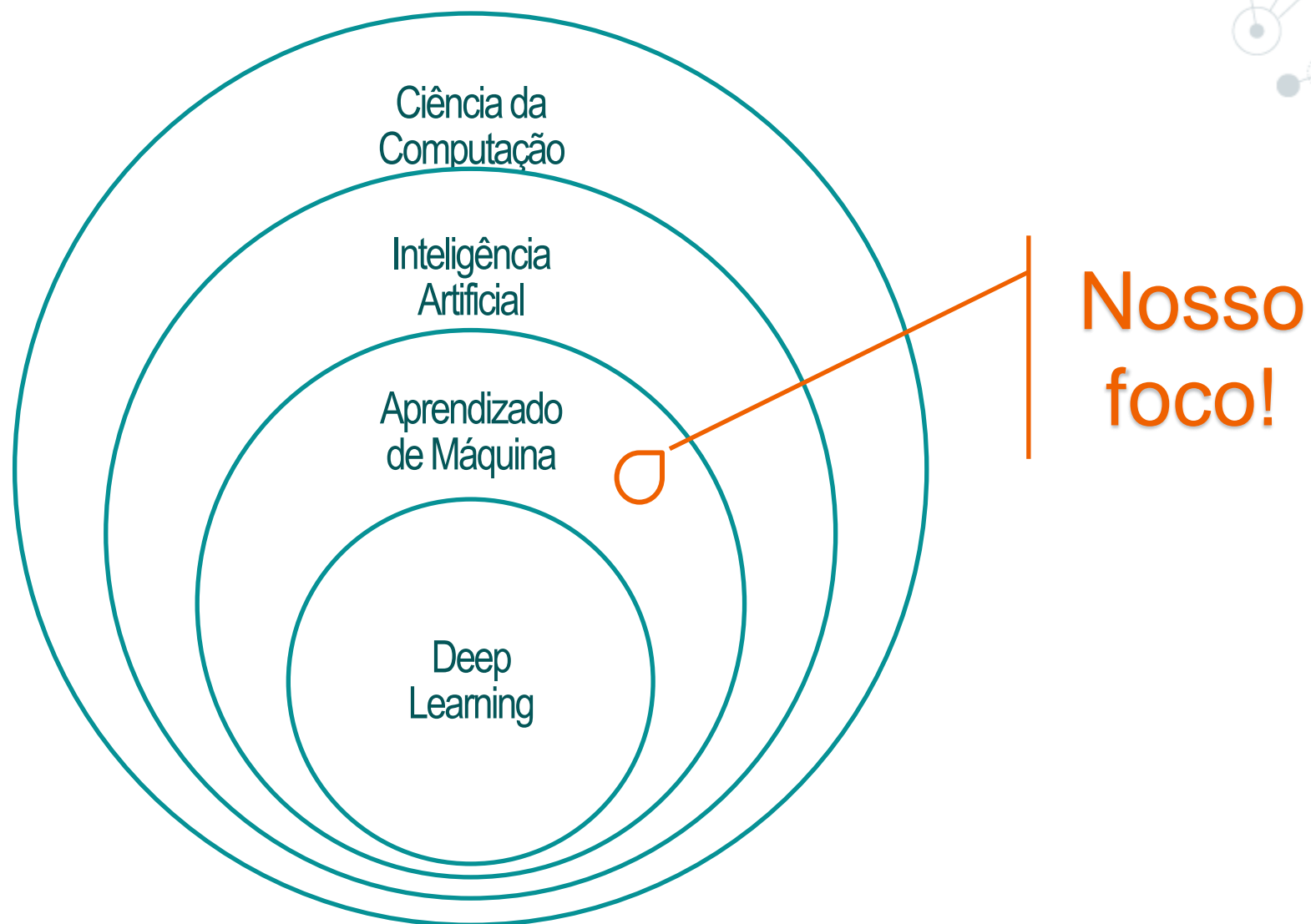
Renato

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are solid dark grey, while others are hollow with a light grey outline. The lines connecting them are thin and light grey, creating a dense, organic structure that tapers off towards the right.

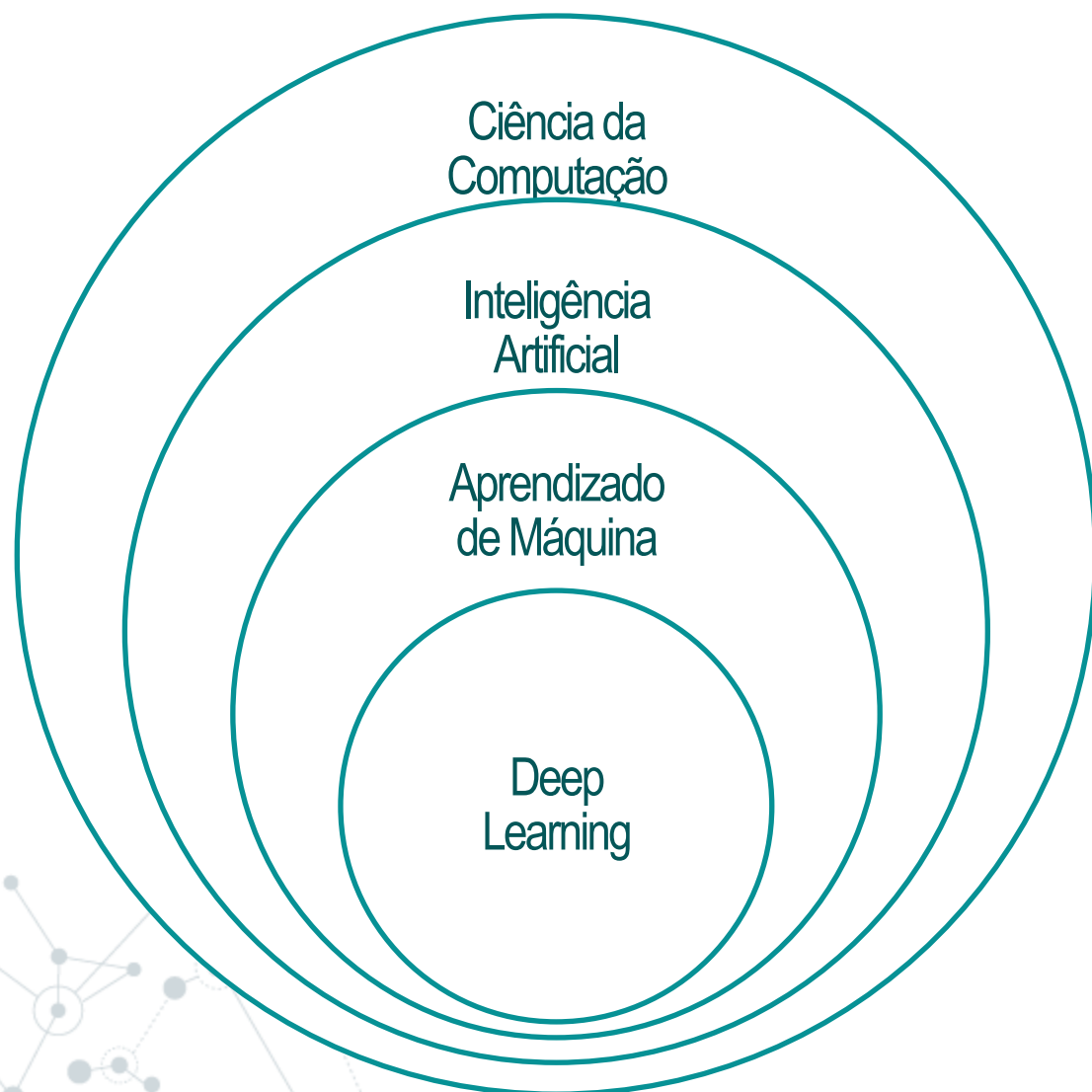
Conceitos

A decorative network diagram in the bottom-right corner, mirroring the style of the top-left one. It consists of a cluster of nodes (solid and hollow circles) connected by thin, light grey lines, forming a complex, interconnected web that tapers off towards the left.

O que é Machine Learning?



O que é Machine Learning?



© Vamos primeiro entender o que é IA

© Machine Learning \neq IA !!

O que é Inteligência Artificial?

Pensando como um humano

- “O novo e interessante esforço para fazer os computadores pensarem (...) máquinas com mentes, no sentido total e literal.” (Haugeland, 1985)
- “[Automatização de] atividades que associamos ao pensamento humano, atividades como a tomada de decisões, a resolução de problemas, o aprendizado...” (Bellman, 1978)

Pensando racionalmente

- “O estudo das faculdades mentais pelo uso de modelos computacionais.” (Charniak e McDermott, 1985)
- “O estudo das computações que tornam possível perceber, raciocinar e agir.” (Winston, 1992)

IA

Agindo como seres humanos

- “A arte de criar máquinas que executam funções que exigem inteligência quando executadas por pessoas.” (Kurzweil, 1990)
- “O estudo de como os computadores podem fazer tarefas que hoje são melhor desempenhadas pelas pessoas.” (Rich and Knight, 1991)

Agindo racionalmente

- “Inteligência Computacional é o estudo do projeto de agentes inteligentes.” (Poole et al., 1998)
- “AI... está relacionada a um desempenho inteligente de artefatos.” (Nilsson, 1998)

O que é Inteligência Artificial?



<https://youtu.be/n5phH2MywI4>

O que é Inteligência Artificial?



A word cloud of various artificial intelligence and automation-related terms. The words are arranged in a cluster, with 'Veículos-autônomos' being the largest and most central. Other prominent words include 'Sistemas-de-sugestão', 'Marketing', 'Chatbots', 'Indústria-4.0', and 'Smart-buildings'. The background features a faint, light blue network diagram with nodes and connecting lines, primarily visible in the top right and bottom left corners.

Smart-cities
Sistemas-de-sugestão
Analytics
Marketing Chatbots
Veículos-autônomos
Aplicações Smart-logistics
RPA
Smart-buildings
Indústria-4.0

O que é Machine Learning?

Algoritmos de Machine Learning:

- Utilizam métodos estatísticos para encontrar padrões em “grandes” quantidades de dados

Aprendizagem:

- Melhorar o desempenho após fazer
observações sobre o mundo

Dados

- Dados numéricos
- Palavras, frases e textos
- Images
- Cliques

Tipos de aprendizagem

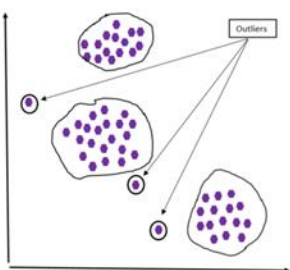
- © Aprendizagem não supervisionada
- © Aprendizagem supervisionada
- © Aprendizagem semi-supervisionada

Aprendizagem não supervisionada

- 🕒 Descoberta de padrões ocultos em conjuntos de dados
- 🕒 Aplicações com dados sem classificação / labels

Detecção de outliers

Objetivo: divisão dos dados em um número definido ou indefinido de categorias



Exemplos de aplicação:

- Detecção de anomalias/falhas em sistemas
- Detecção de fraudes
- Detecção de spams
- Limpeza da dados

Exemplos de algoritmos:

- Gaussian mixture models
- Auto-encoders
- Replicator neural networks
- Restricted Boltzman machines

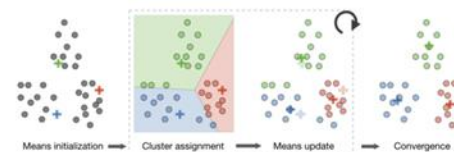
Exemplos de bibliotecas:

- scikit-learn (sklearn)
- Tensorflow
- Pytorch

45

Classificação / Clusterização

Objetivo: divisão dos dados em um número definido ou indefinido de categorias



Exemplos de algoritmos:

- K-means
- Gaussian mixture models
- Nearest neighbors

Exemplos de aplicação:

- Segmentação de grupos de clientes
- Estudos estatísticos de populações, doenças

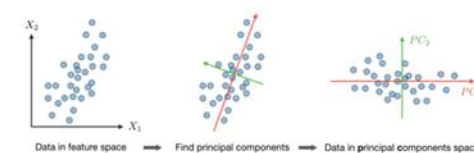
Exemplos de bibliotecas:

- scikit-learn (sklearn)

44

Redução de dimensão

Objetivo: Reduzir quantidade de dimensões



Exemplos de algoritmos:

- Principal component analysis
- Linear discriminant analysis (LDA)

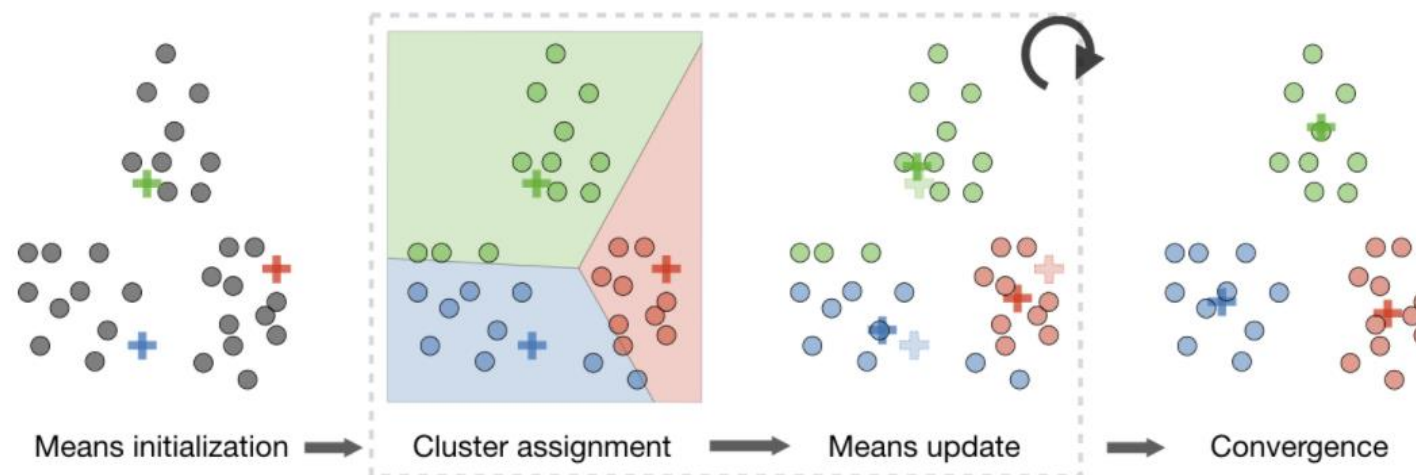
Exemplos de aplicação:

- Metamodelagem
- Análise estatística
- Análise de experimentos
- Melhorar dados de modelos de regressão

46

Classificação / Clusterização

Objetivo: divisão dos dados em um número definido ou indefinido de categorias



Exemplos de algoritmos:

- K-means
- Gaussian mixture models
- Nearest neighbors

Exemplos de aplicação:

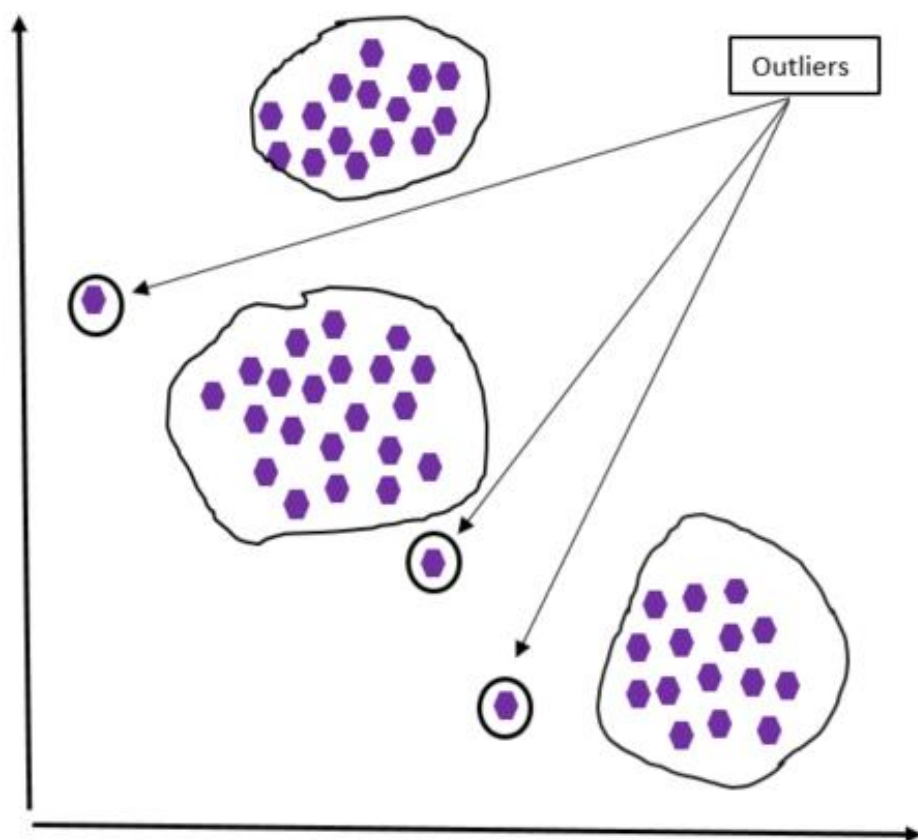
- Segmentação de grupos de clientes
- Estudos estatísticos de populações, doenças

Exemplos de bibliotecas:

- scikit-learn (sklearn)

Detecção de outliers

Objetivo: detecção de pontos de dados errôneos



Exemplos de aplicação:

- Detecção de anomalias/falhas em sistemas
- Detecção de fraudes
- Detecção de spams
- Limpeza da dados

Exemplos de algoritmos:

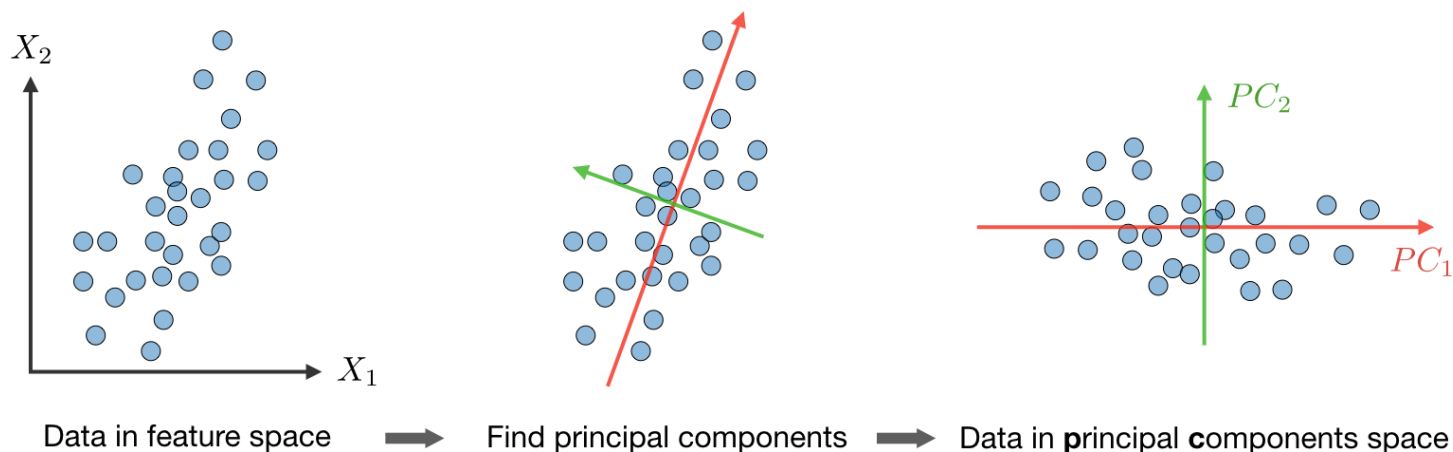
- Gaussian mixture models
- Auto-encoders
- Replicator neural networks
- Restricted Boltzman machines

Exemplos de bibliotecas:

- scikit-learn (sklearn)
- Tensorflow
- Pytorch

Redução de dimensão

Objetivo: Reduzir quantidade de dimensões



Exemplos de algoritmos:

- Principal component analysis
- Linear discriminant analysis

(LDA)

Exemplos de aplicação:

- Metamodelagem
- Análise estatística
- Análise de experimentos
- Melhorar dados de modelos de regressão

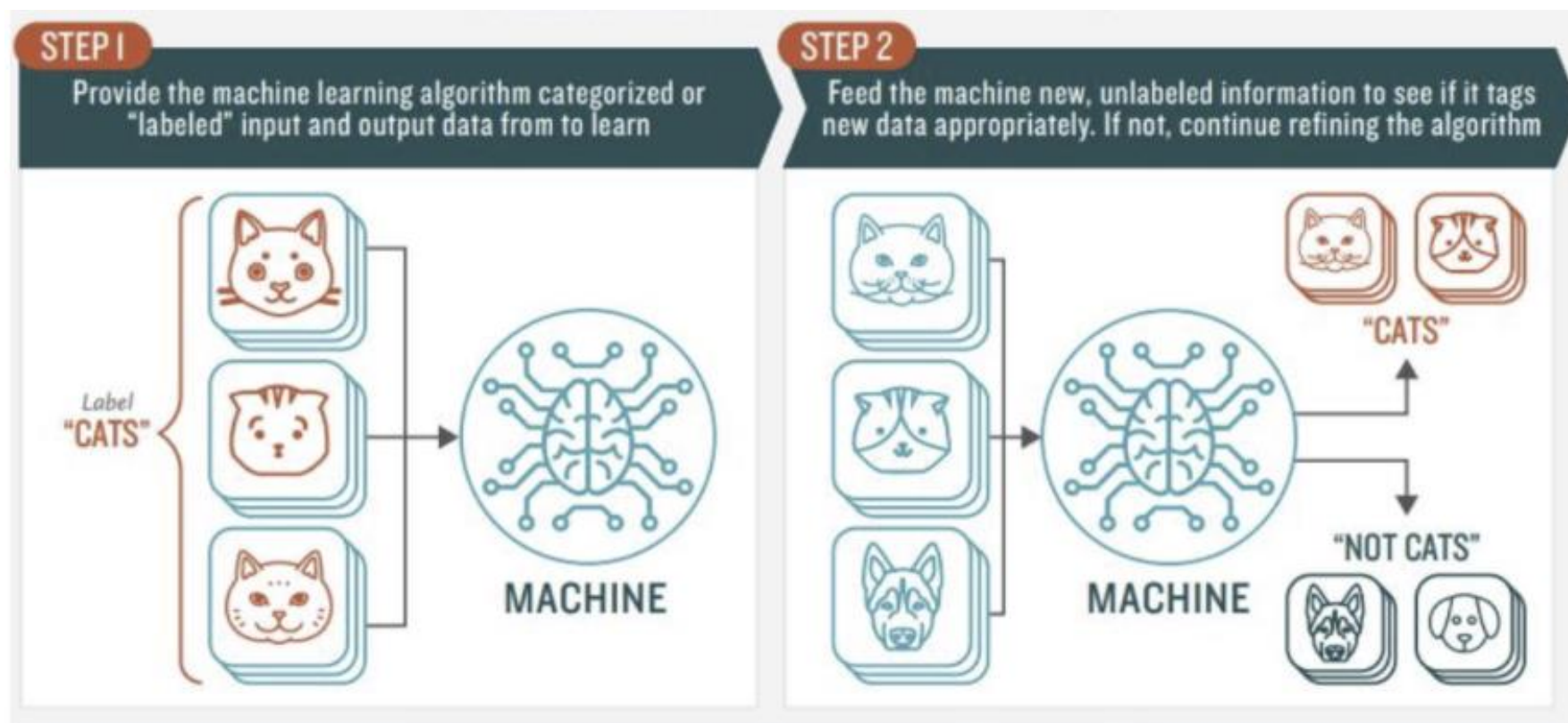
Aprendizagem supervisionada

- © Dados com labels – Y – conhecidos



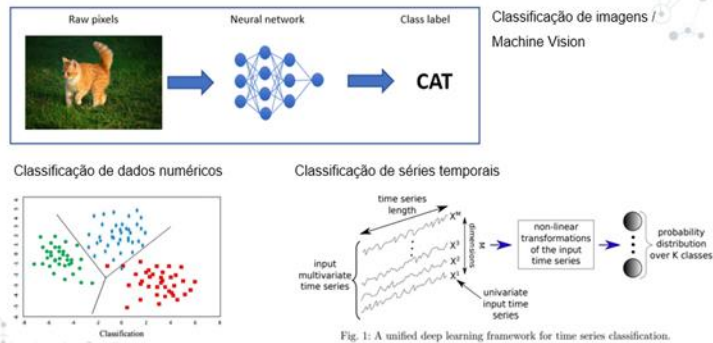
- © Treinamento – aprendizagem – realizada a partir de feedback de erros e acertos em relação aos labels conhecidos
- © Aprendizagem por reforço → Aprendizagem supervisionada

Aprendizagem supervisionada



Aprendizagem supervisionada

Classificação

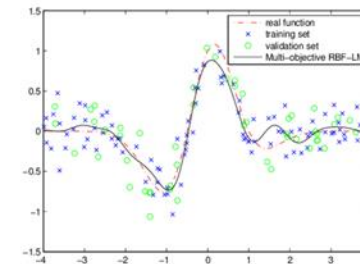


37

Regressão

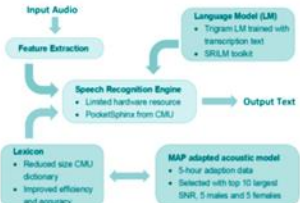
Objetivo:

- Reproduzir resultados de uma função dado um conjunto de entradas numéricas



39

Natural Language Processing - Speech Recognition



Exemplos de algoritmos:

- Redes Neurais de Recorrentes

Exemplos de aplicação:

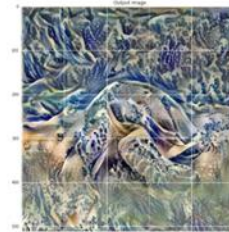
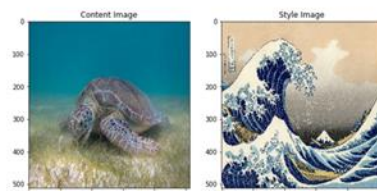
- Machine translation
- Controle por voz
- Interpretação de resenhas / análise de sentimento
- Busca
- Chatbots

Exemplos de frameworks:

- Tensorflow
- Pytorch
- Matlab

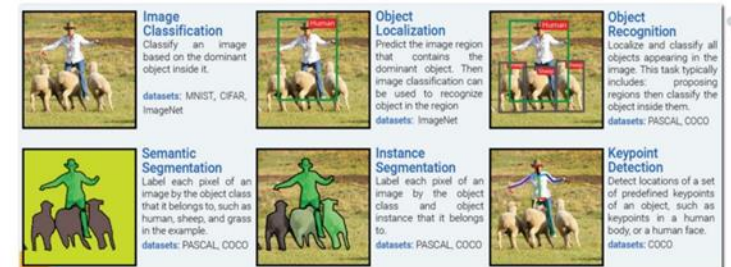
42

Outras classes



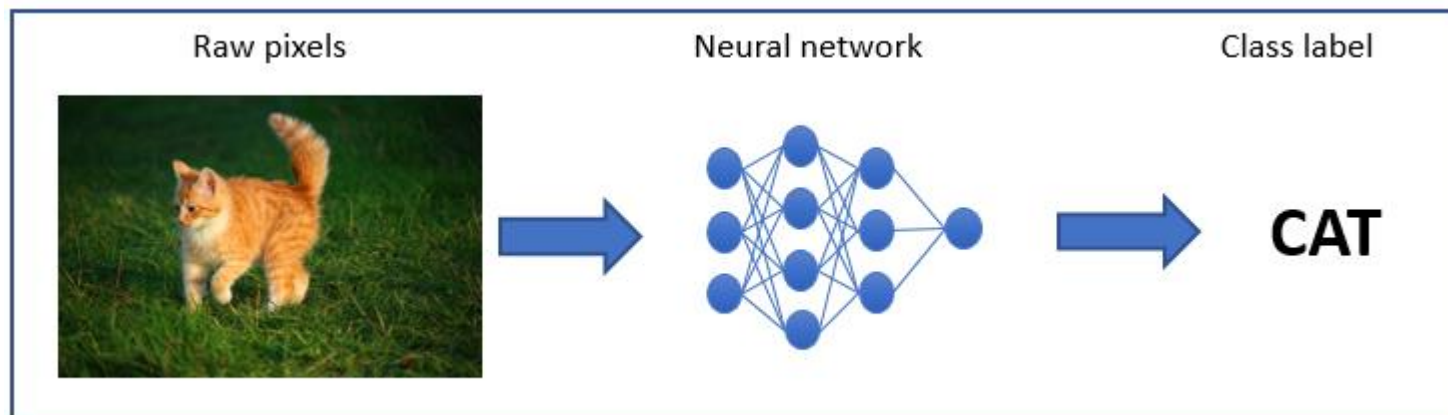
43

Machine Vision



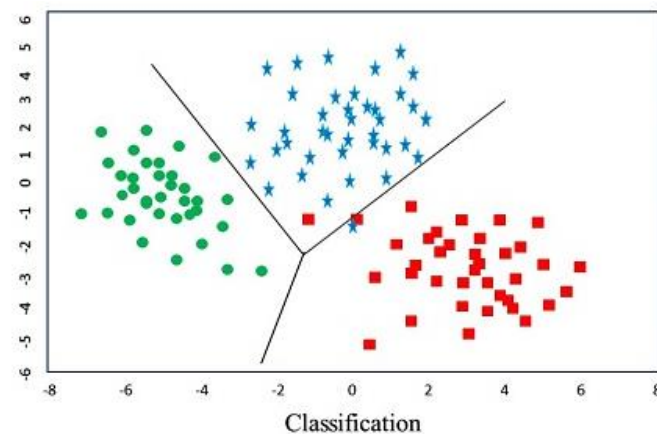
41

Classificação



Classificação de imagens /
Machine Vision

Classificação de dados numéricos



Classificação de séries temporais

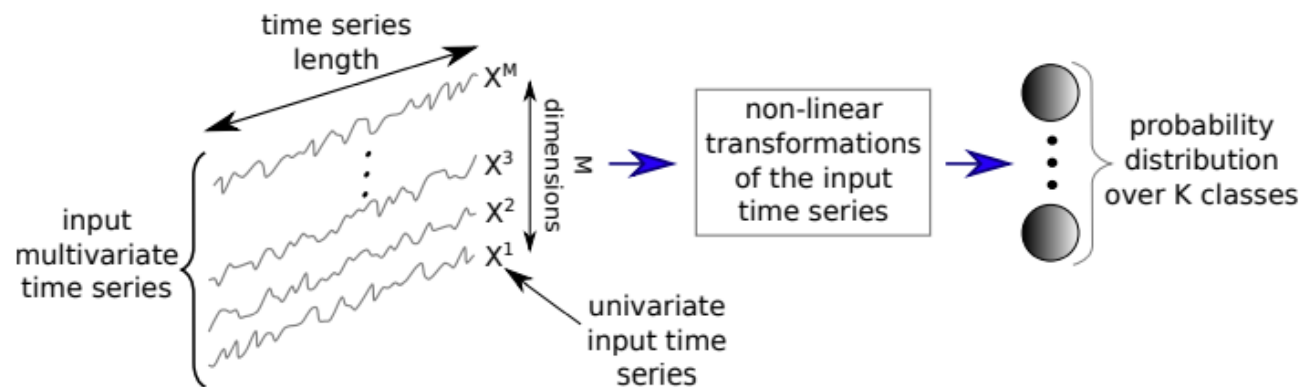


Fig. 1: A unified deep learning framework for time series classification.

Classificação

Exemplos de aplicação:

- Diagnósticos por exames laboratoriais
- Manutenção preditiva

Exemplos de aplicação – Machine

Vision:

- Diagnósticos por imagem
- Filtros de imagens impróprias
- Controle de qualidade industrial
- Busca de imagem
- Prova de vida

Exemplos de algoritmos:

- Redes Neurais
- Redes Neurais Recorrentes
- Support Vector Machines
- Random Forest
- Deep belief networks
- Logistic Regression

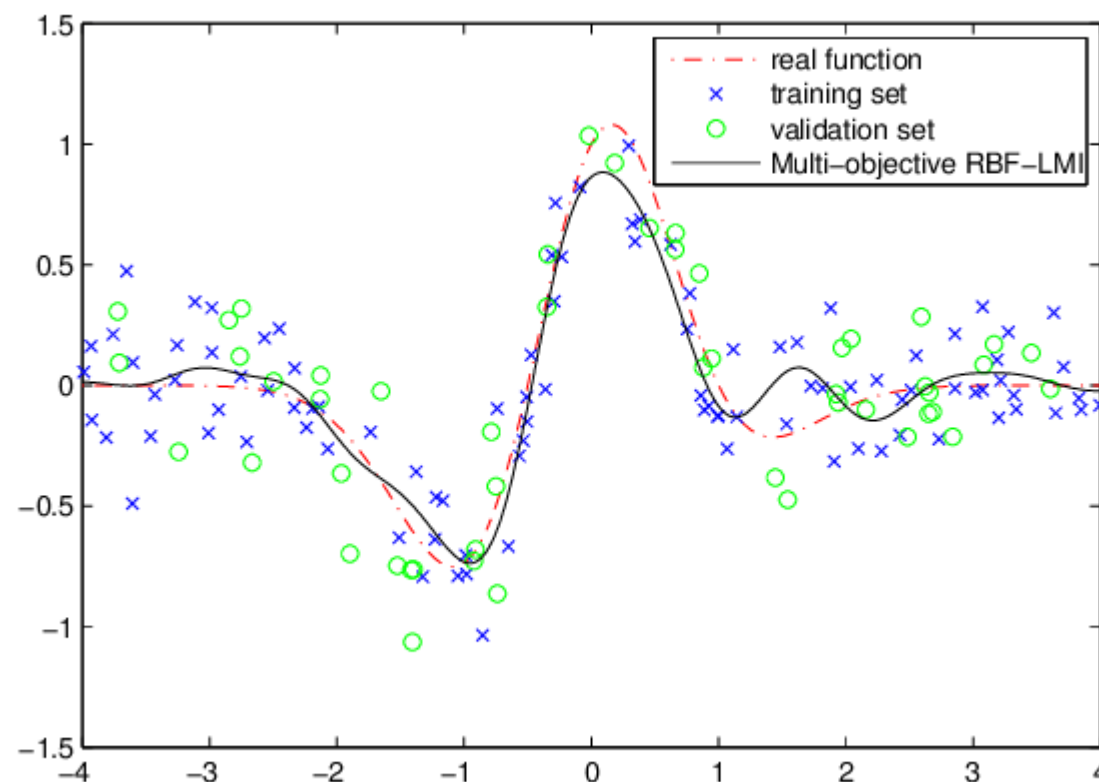
Exemplos de algoritmos – Machine Vision:

- Redes Neurais de Convolução
- Deep Learning

Regressão

Objetivo:

- Reproduzir resultados de uma função dado um conjunto de entradas numéricas



Regressão

Exemplos de aplicação:

- Meta modelos para problemas de otimização
- Redução de custo computacional de funções complexas
- Análise estatística
- Criação de modelos a partir de dados experimentais
- Calibração de modelos
- Calibração de instrumentos
- Interpolação

Exemplos de algoritmos:

- Mínimos quadrados
- Redes Neurais
- Redes Neurais Recorrentes
- Support Vector Machines

Exemplos de frameworks:

- Tensorflow
- Pytorch
- Matlab
- Scikit-learn

Machine Vision



Image Classification

Classify an image based on the dominant object inside it.

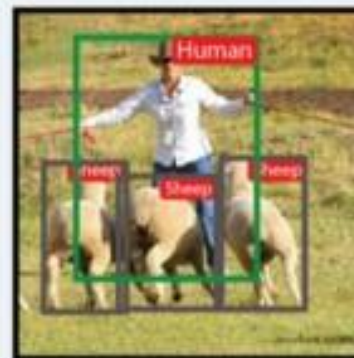
datasets: MNIST, CIFAR, ImageNet



Object Localization

Predict the image region that contains the dominant object. Then image classification can be used to recognize object in the region

datasets: ImageNet



Object Recognition

Localize and classify all objects appearing in the image. This task typically includes: proposing regions then classify the object inside them.

datasets: PASCAL, COCO



Semantic Segmentation

Label each pixel of an image by the object class that it belongs to, such as human, sheep, and grass in the example.

datasets: PASCAL, COCO



Instance Segmentation

Label each pixel of an image by the object class and object instance that it belongs to.

datasets: PASCAL, COCO

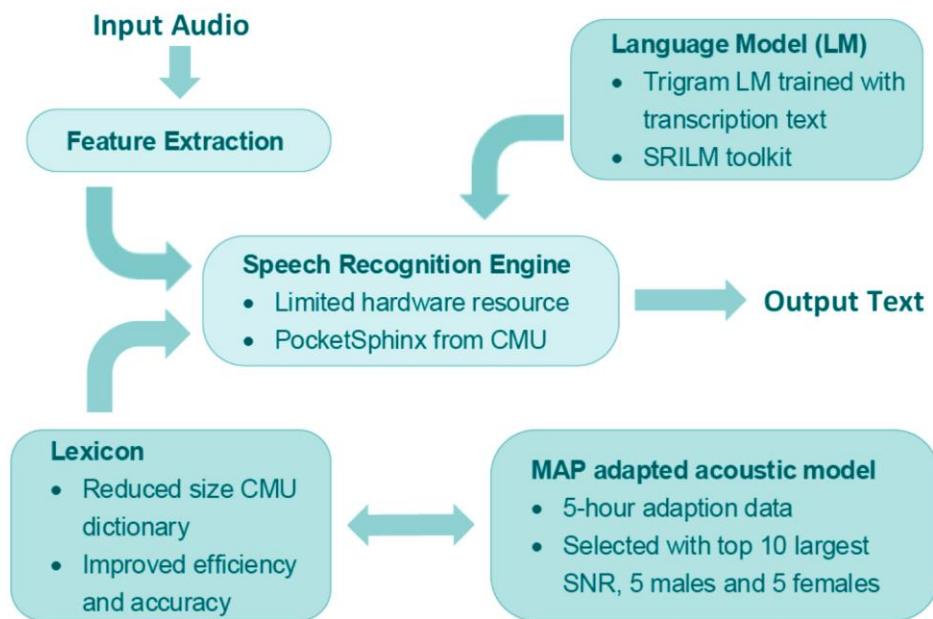


Keypoint Detection

Detect locations of a set of predefined keypoints of an object, such as keypoints in a human body, or a human face.

datasets: COCO

Natural Language Processing - Speech Recognition



Exemplos de aplicação:

- Machine translation
- Controle por voz
- Interpretação de resenhas / análise de sentimento
- Busca
- Chatbots

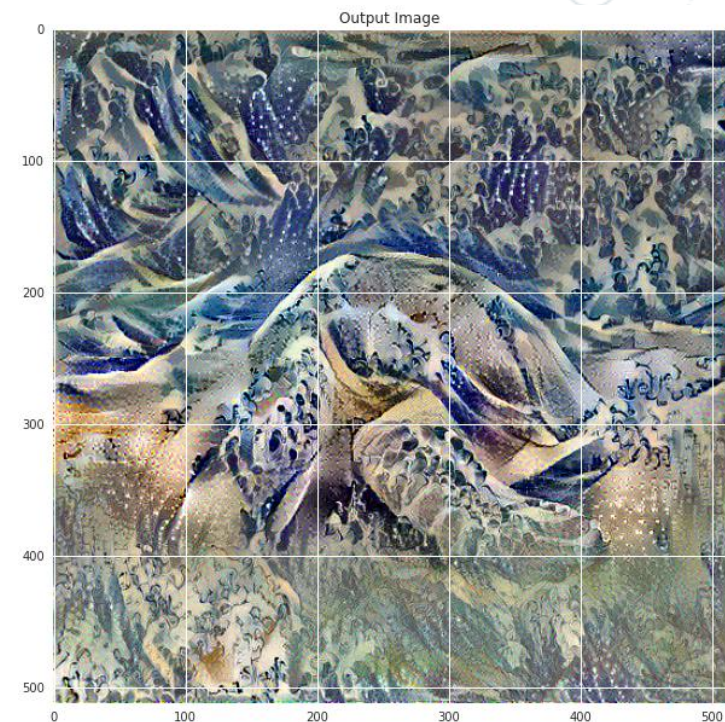
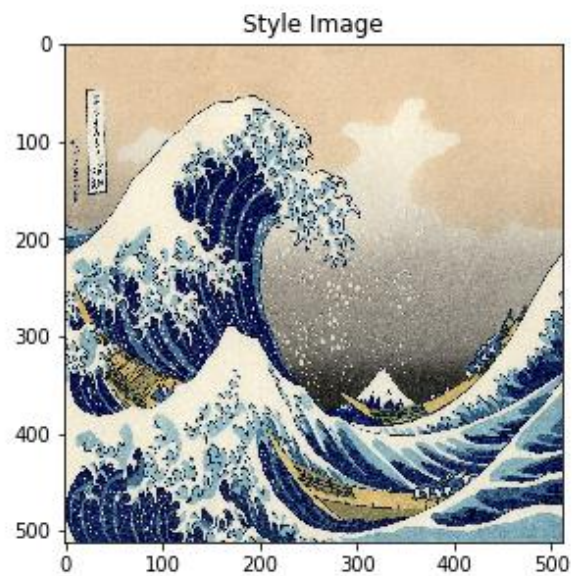
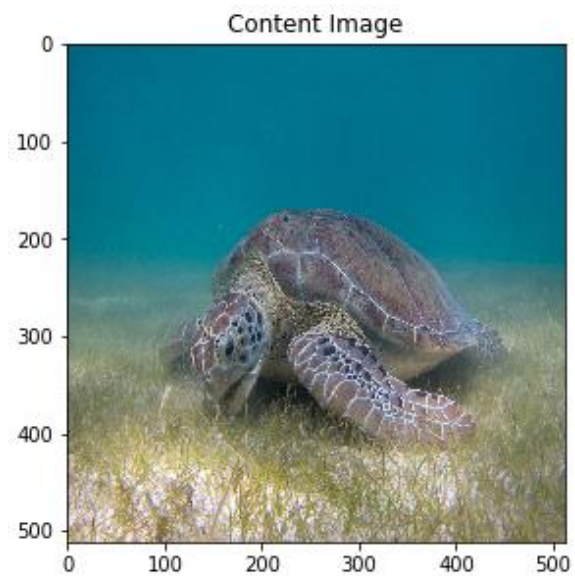
Exemplos de algoritmos:

- Redes Neurais de Recorrentes

Exemplos de frameworks:

- Tensorflow
- Pytorch
- Matlab

Outras classes



Modelos paramétricos versus Modelos não paramétricos

- ◎ Um modelo de aprendizagem que resume os dados com um conjunto de parâmetros de tamanho fixo (independentemente do número de exemplos de treinamento) é chamado de modelo paramétrico.
 - ◎ Exemplos: Redes Neurais, regressão linear
- ◎ Um modelo não paramétrico é aquele que não pode ser caracterizado por um conjunto limitado de parâmetros
 - ◎ Exemplos: Nearest neighbor, SVM

Dados estruturados versus Dados não estruturados

☉ Dados estruturados:

☉ Dados que podem ser organizados em tabelas

☉ Exemplo:

- Tabelas de dados numéricos
- Dados armazenados em arquivos CSV
- Dados armazenados em bancos de dados SQL

☉ Dados não estruturados

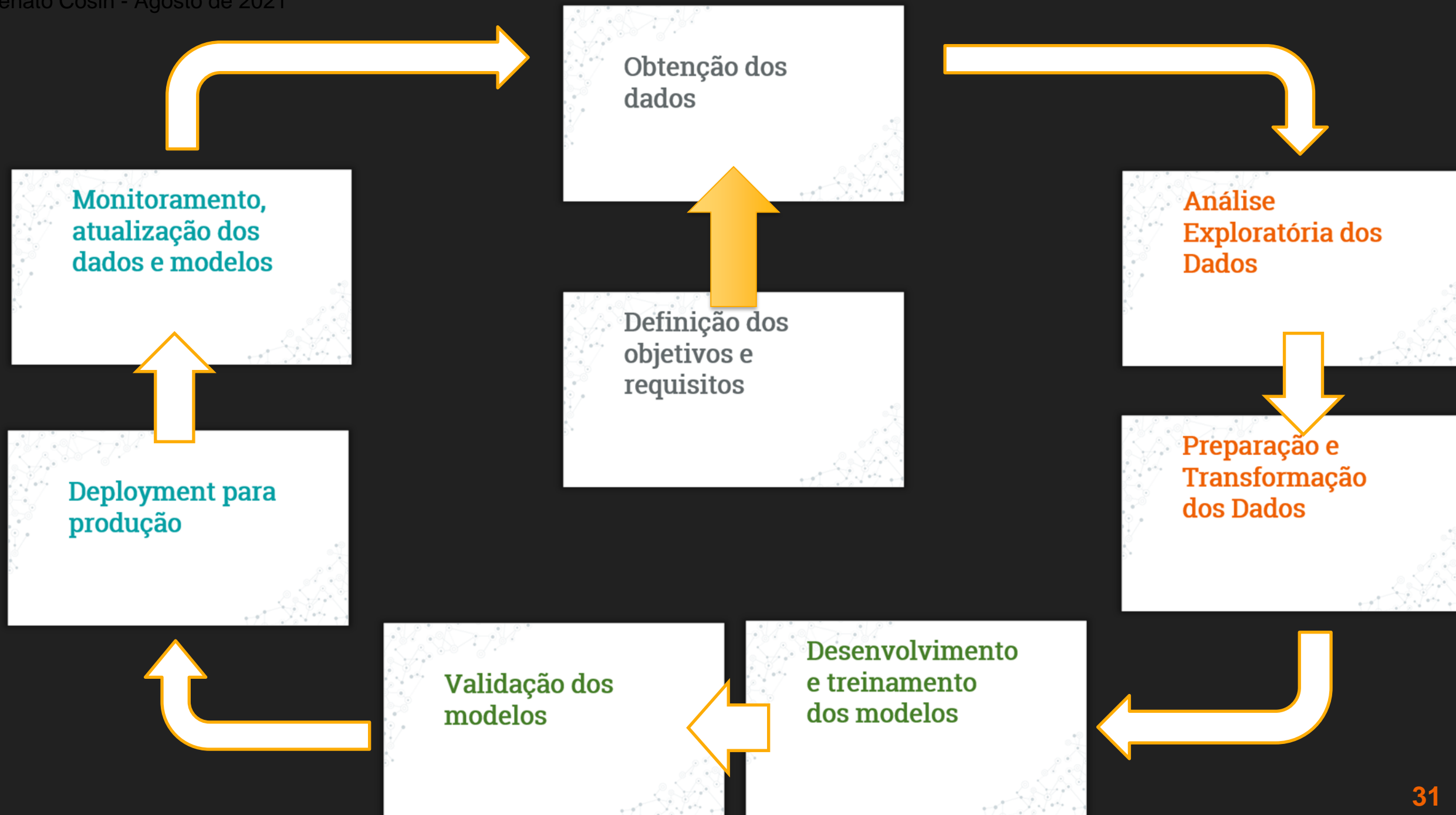
☉ Dados que não podem ser organizados em tabelas

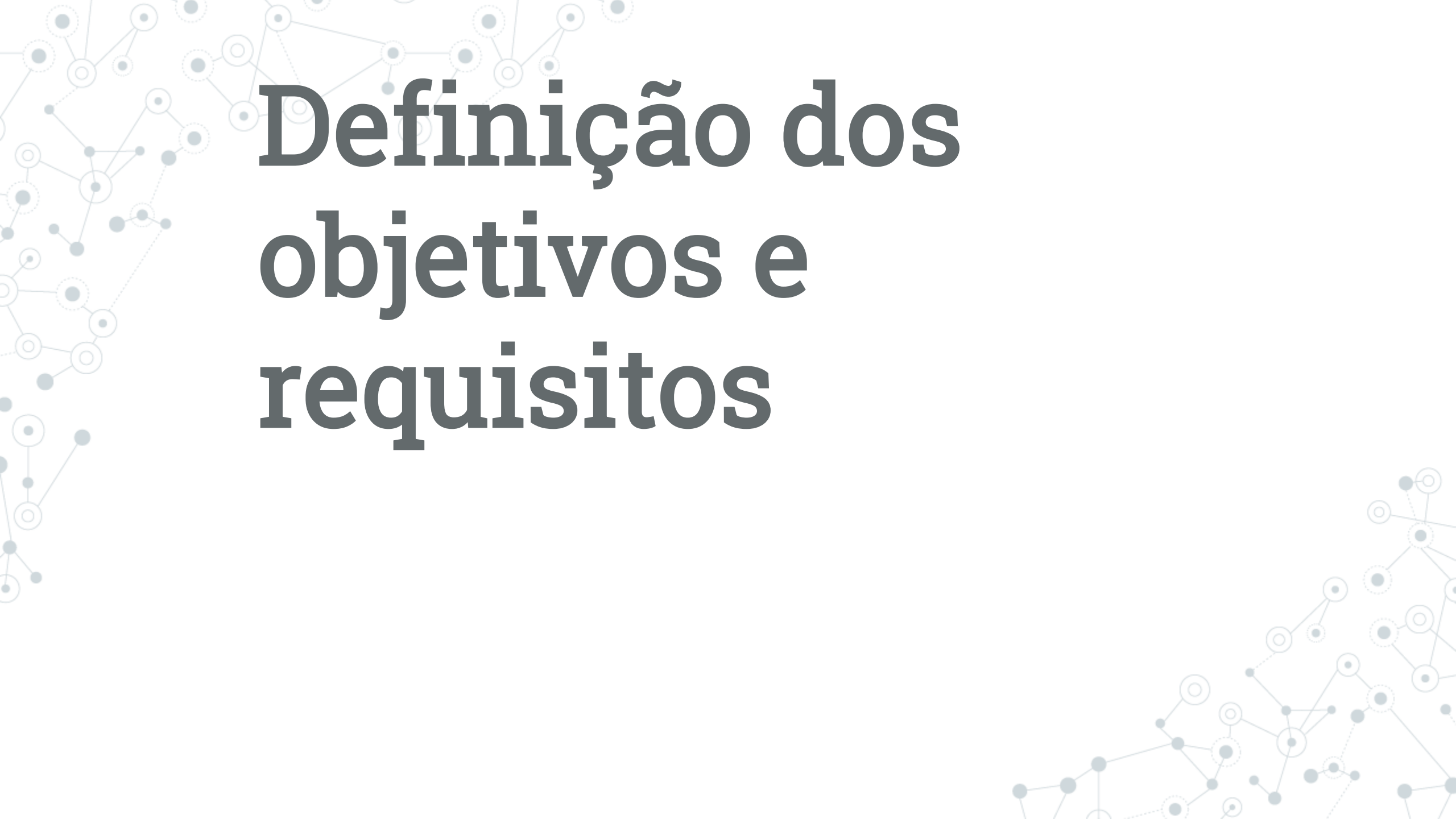
☉ Exemplos:

- Imagens
- Textos
- Áudio



Workflow



The background features a light gray network of nodes and lines, resembling a molecular structure or a data network, positioned in the top-left and bottom-right corners.

Definição dos objetivos e requisitos

Definição dos objetivos e requisitos

- © Qual é a necessidade que levou ao uso do ML?
- © O que meu modelo precisa fazer?
- © Quão acurado precisa ser?
- © Que tipos de dados estarão disponíveis?
- © Em que volume?
- © Quais são os resultados esperados?
- © Qual é a métrica de acurácia?
- © Que poder computacional estará disponível para treinamento?
- © E para inferência?
- © Que tipo de ML poderei usar?

A decorative network diagram in the top-left corner, featuring a cluster of interconnected nodes. Some nodes are solid dark grey circles, while others are hollow circles with a dark grey outline. They are connected by thin, light grey lines, forming a complex web structure.

Obtenção dos dados

A decorative network diagram in the bottom-right corner, similar to the one in the top-left. It consists of a cluster of interconnected nodes, some solid dark grey circles and some hollow circles with dark grey outlines, connected by thin, light grey lines.

Obtenção dos dados

- © Dados públicos
- © Dados simulados ou sintéticos
- © Dados experimentais
- © Dados históricos
- © etc

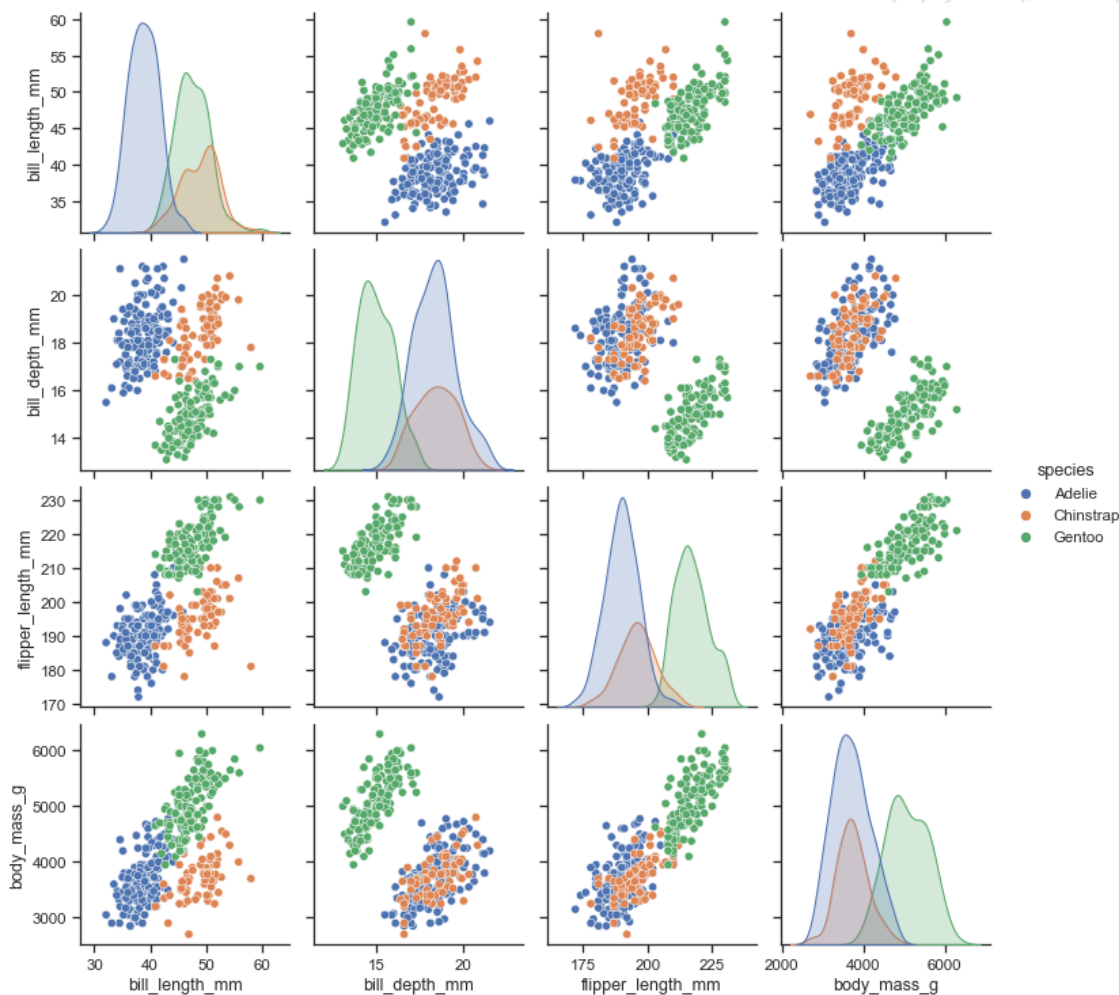


Análise Exploratória dos Dados

Análise Exploratória dos Dados

- ◎ Primeiro contato com os dados
- ◎ Avaliação de qualidade
- ◎ Visualização gráfica dos dados
- ◎ Métricas estatísticas dos dados

Exemplo de visualização de dados:
Scatter Matrix de dataset gerada pela
bibliotec Seaborn (Python)

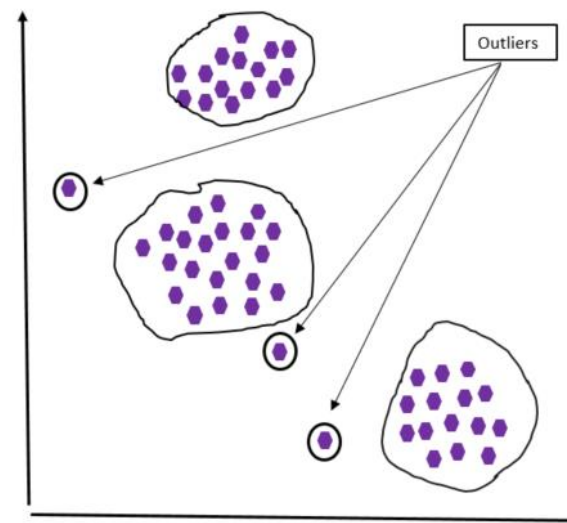




Preparação e Transformação dos Dados

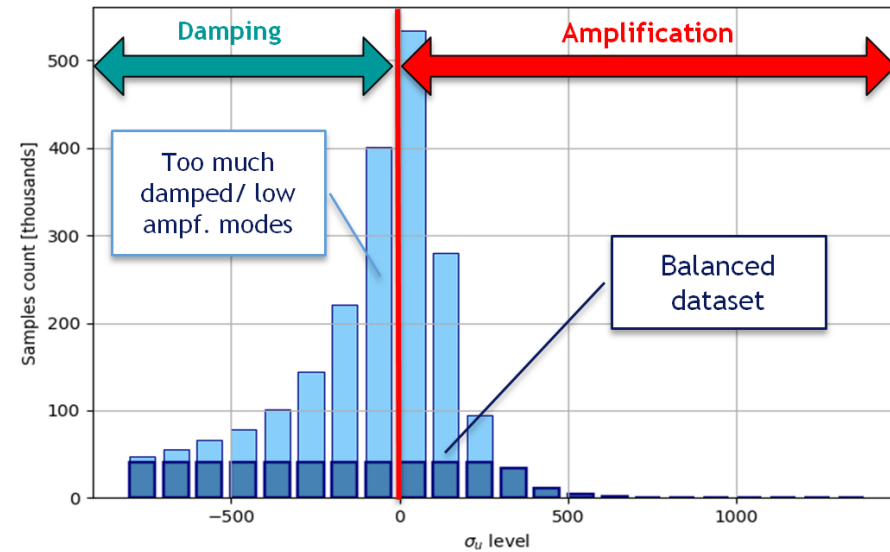
Preparação e Transformação dos Dados

- ◎ Limpeza / filtragem dos dados
- ◎ Remoção de outliers
- ◎ Seleção das features a serem utilizadas
- ◎ Criação de novas features
- ◎ Data augmentation
- ◎ Balanceamento dos dados
- ◎ Segmentação dos dados
 - ◎ Dados de treinamento
 - ◎ Dados de teste
 - ◎ Dados de avaliação



Exemplo conceitual de detecção de outliers, que provavelmente devem ser removidos do dataset

Preparação e Transformação dos Dados



Exemplo de balanceamento de datasets, equilibrando o histograma de label numérica por amplitude da mesma

Exemplo de data augmentation para classificação de imagens

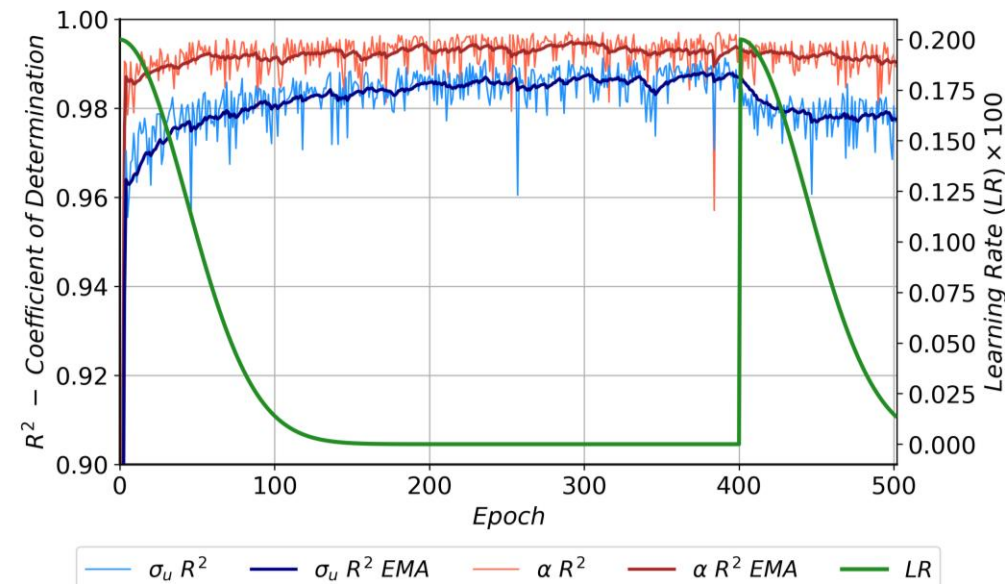


Desenvolvimento e treinamento dos modelos

Desenvolvimento e treinamento dos modelos

- © Definição dos algoritmos
- © Implementação dos modelos
- © Definição / otimização do hiperparâmetros
- © Treinamento

Exemplo de histórico de métrica de erro ao longo de treinamento de modelo de regressão



A decorative network diagram in the top-left corner, consisting of a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are solid grey and others are hollow with a grey outline. The lines connecting them are thin and grey, creating a dense, organic structure.

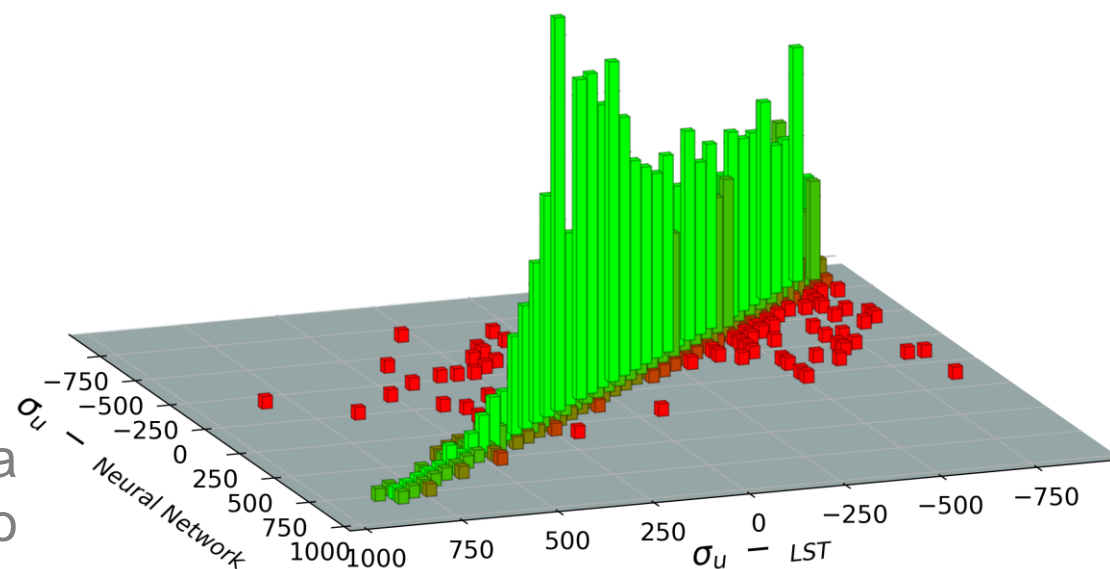
Validação dos modelos

A decorative network diagram in the bottom-right corner, similar to the one in the top-left. It features a cluster of interconnected nodes and lines, with some nodes being solid grey and others hollow with grey outlines, all connected by thin grey lines.

Validação dos modelos

- © Testes de acurácia
- © Validação em condições reais de uso do modelo

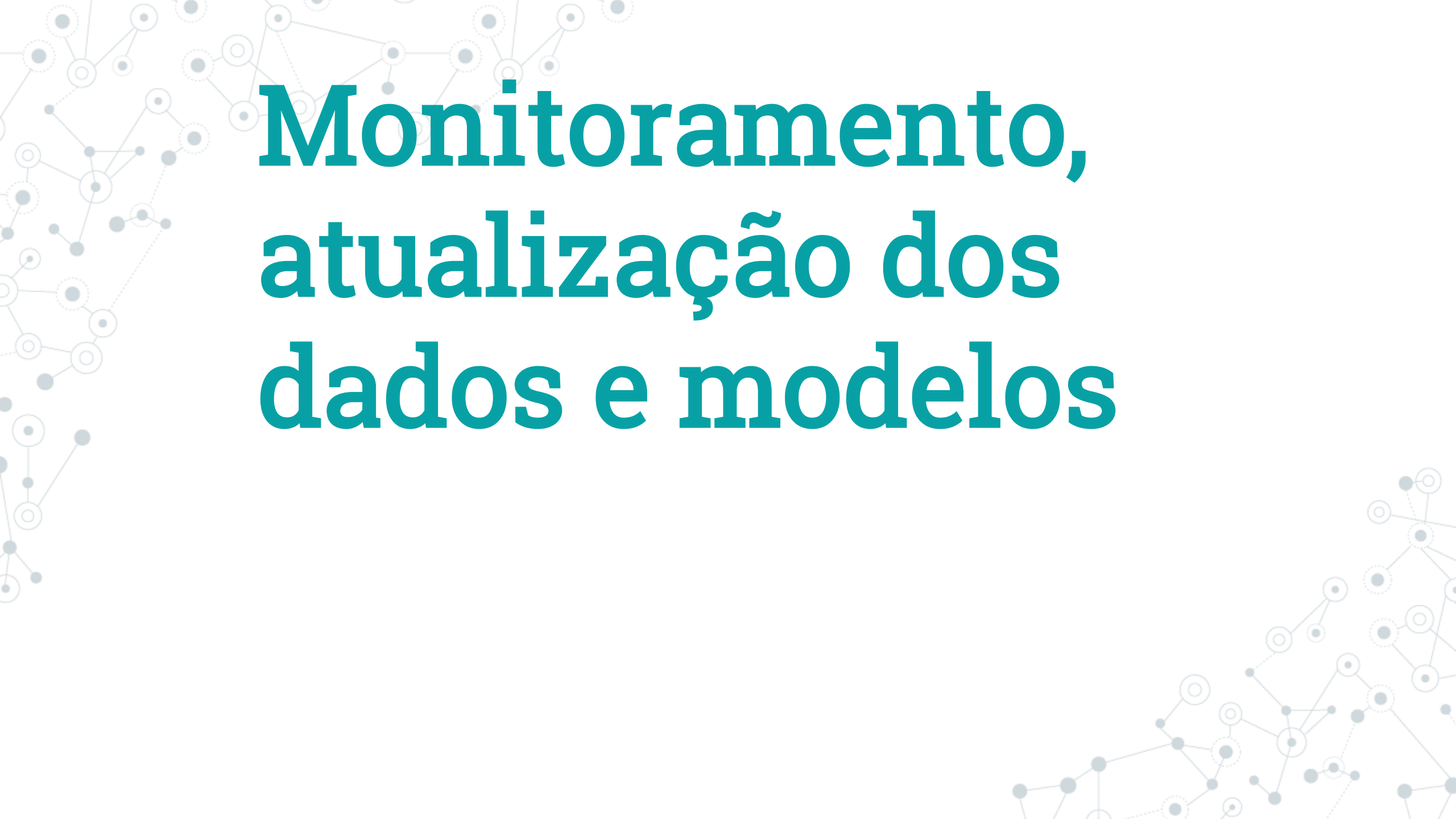
Exemplo de histograma de erros na predição de um modelo de regressão



A decorative network pattern of interconnected nodes and lines, rendered in a light gray color, is positioned in the top-left corner of the slide. The nodes vary in size and some have concentric circles, suggesting a hierarchical or complex network structure.

Deployment para produção

A decorative network pattern of interconnected nodes and lines, rendered in a light gray color, is positioned in the bottom-right corner of the slide. The nodes vary in size and some have concentric circles, suggesting a hierarchical or complex network structure.

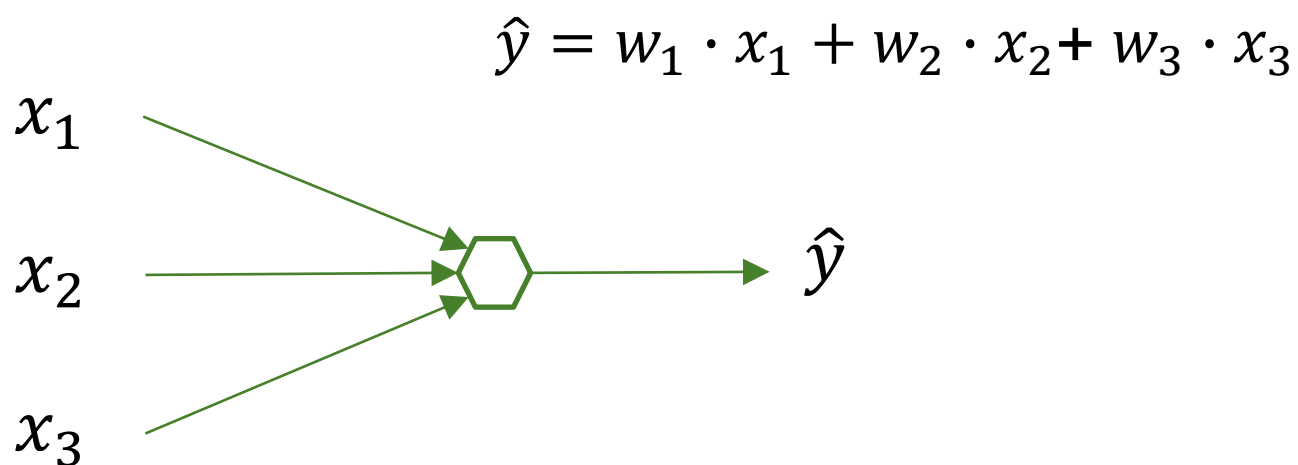


Monitoramento, atualização dos dados e modelos



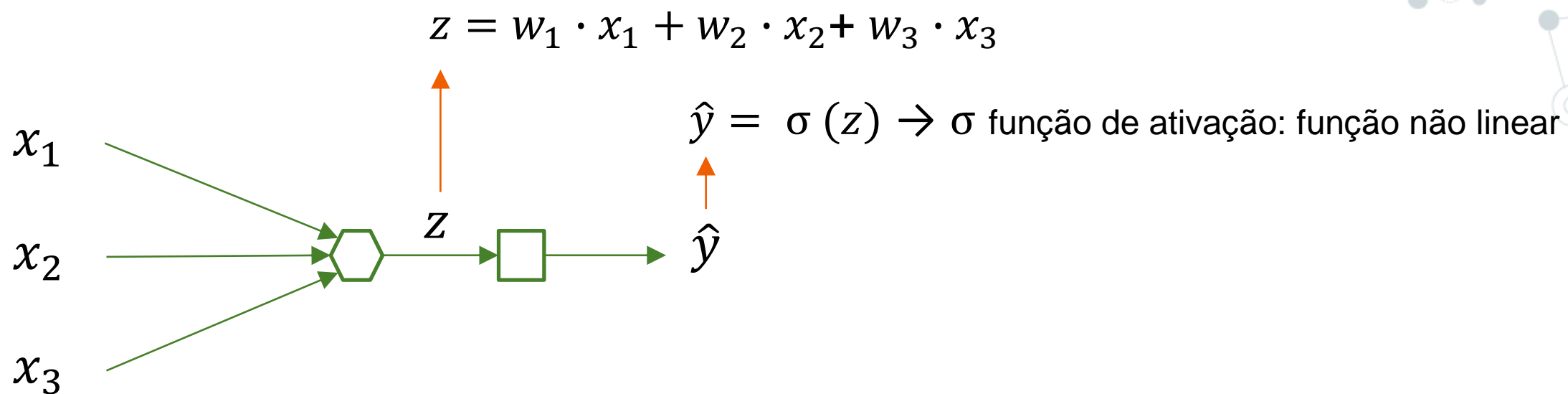
Redes Neurais

Uma rede neural muito simples

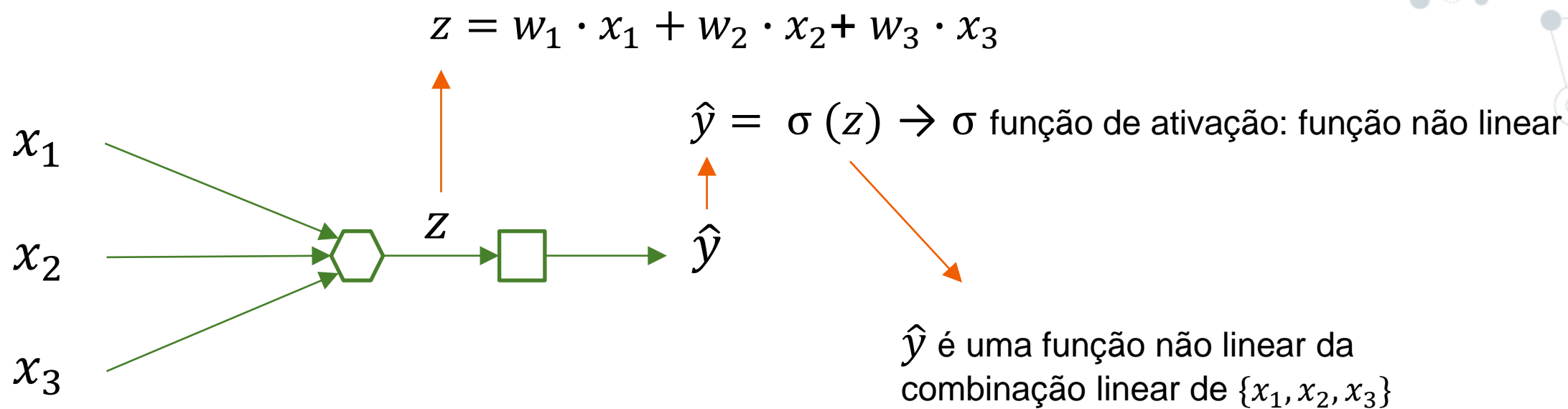


\hat{y} é uma combinação linear de $\{x_1, x_2, x_3\}$

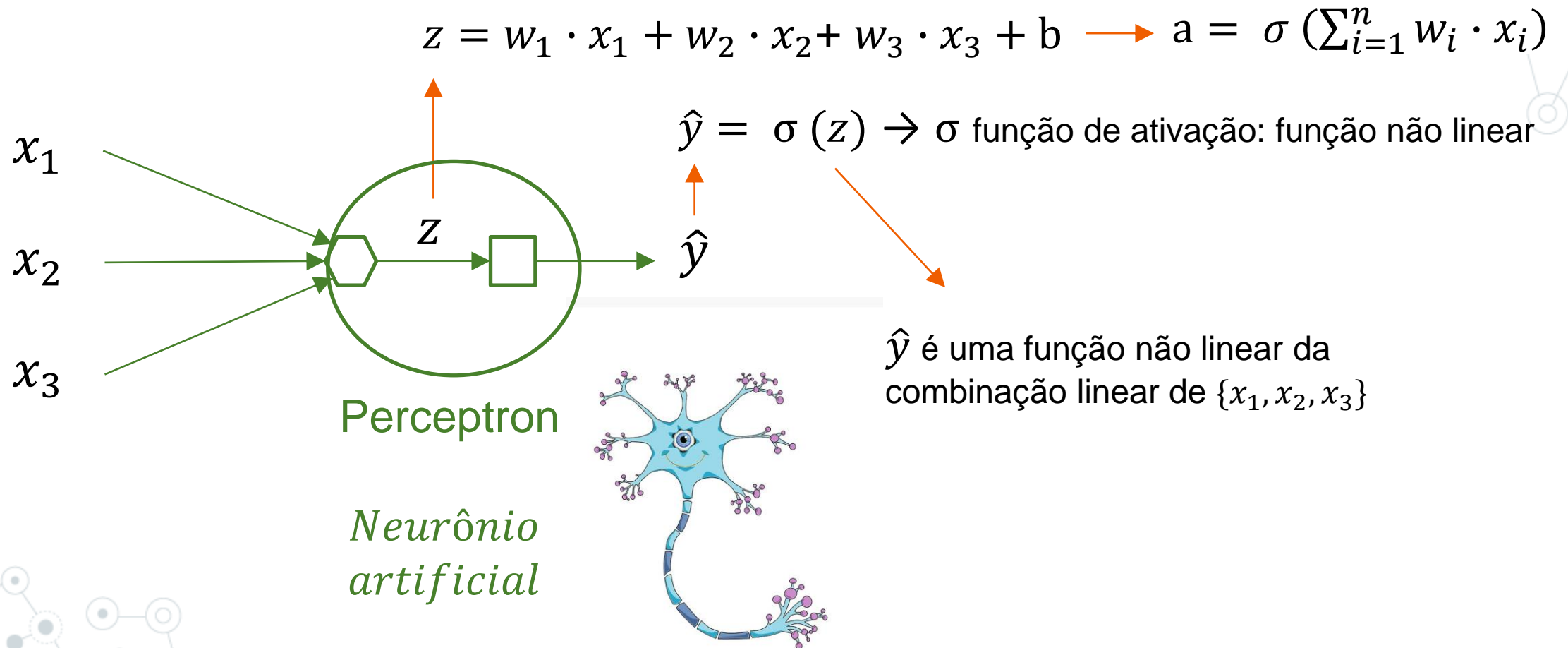
Uma rede neural muito simples



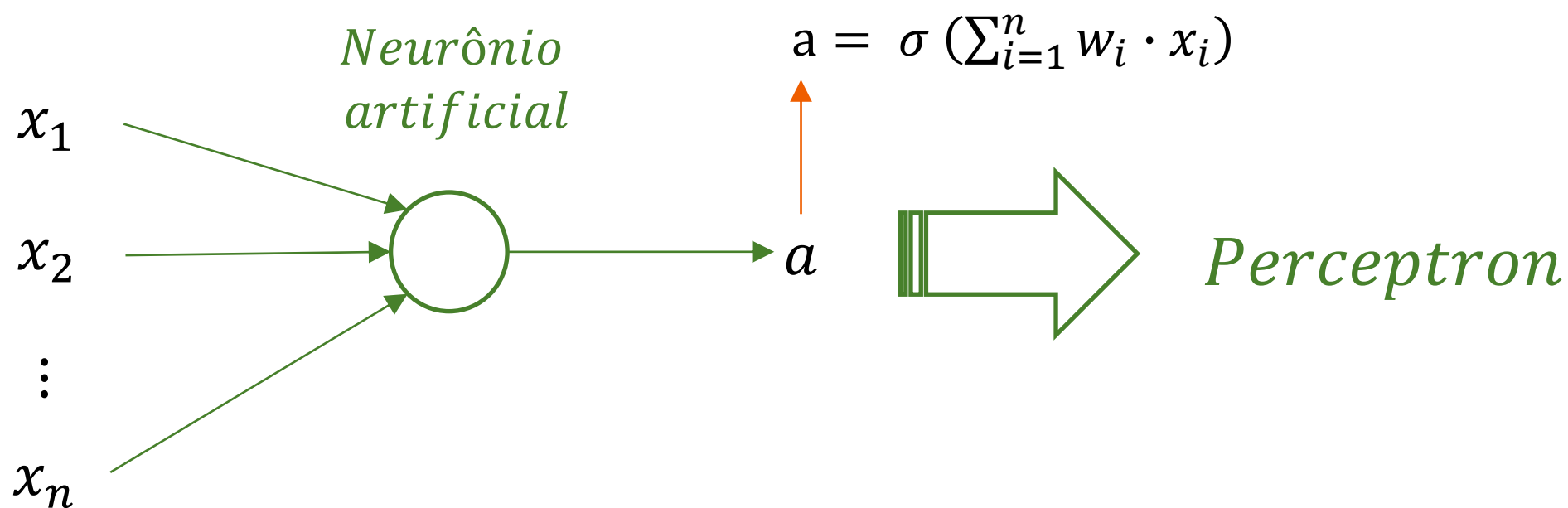
Uma rede neural muito simples



Uma rede neural muito simples

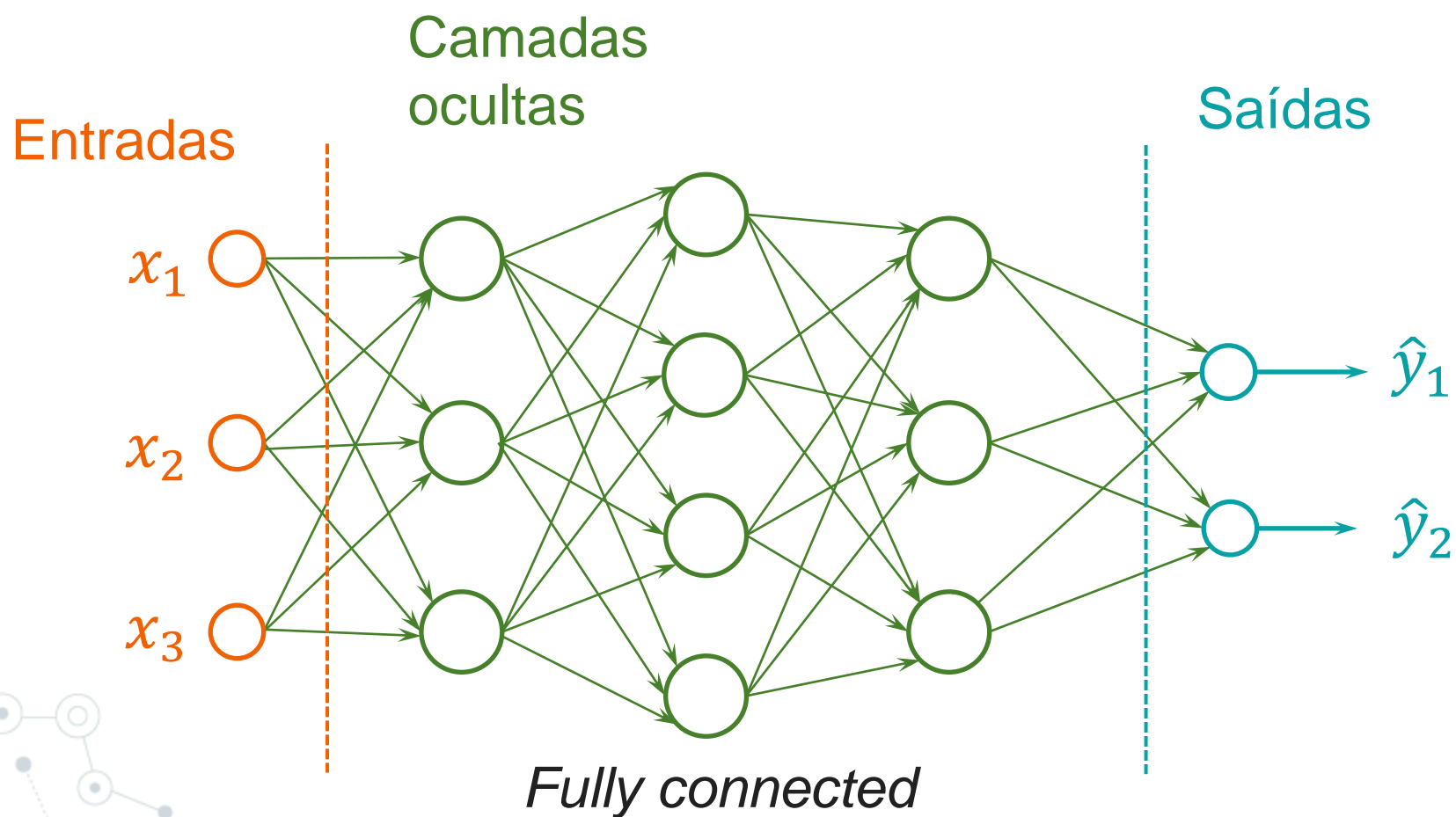


Uma rede neural muito simples

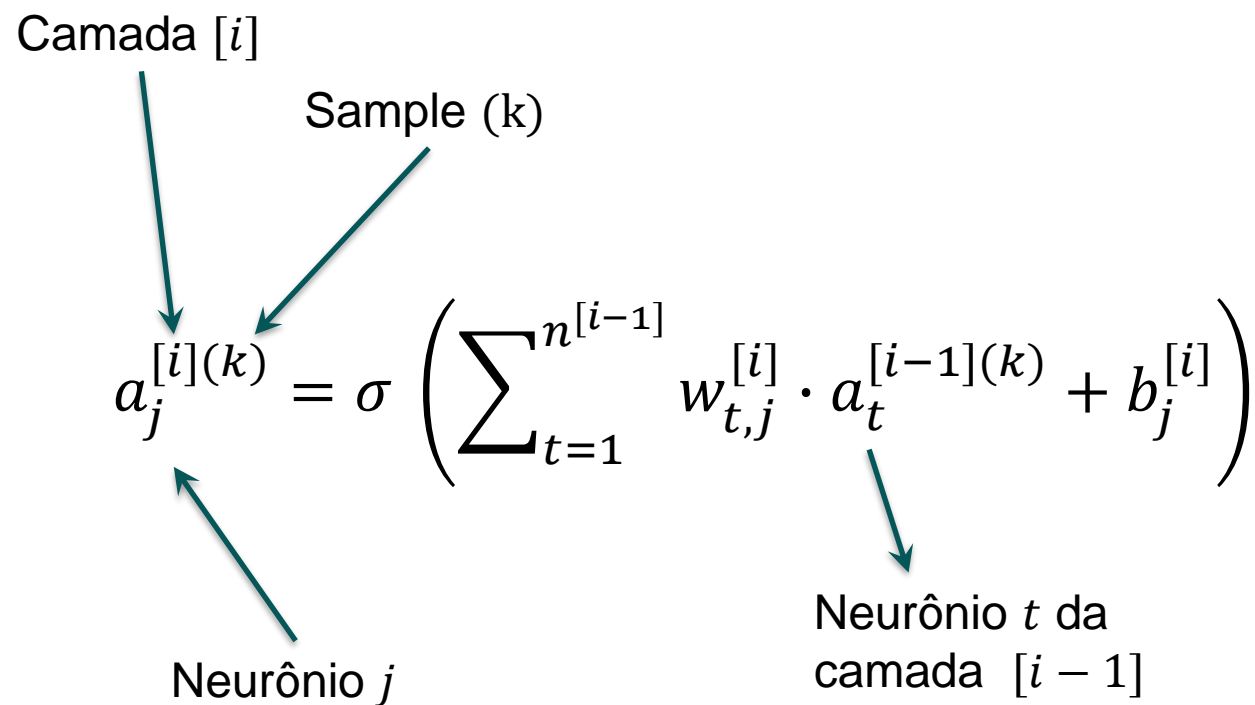


Uma rede neural muito simples

Multi Layer Perceptrons – MLP ∈ Feed forward Neural Networks



Uma rede neural muito simples



The diagram illustrates the calculation of the output for a specific neuron in a neural network layer. It features a central equation with three arrows pointing to its components: one from 'Camada [i]' to the output variable, one from 'Sample (k)' to the sample index, and one from 'Neurônio j' to the neuron index. A fourth arrow points from the term $a_t^{[i-1](k)}$ in the equation to the text 'Neurônio t da camada [i - 1]'. The background includes faint network diagrams in the top right and bottom left corners.

$$a_j^{[i](k)} = \sigma \left(\sum_{t=1}^{n^{[i-1]}} w_{t,j}^{[i]} \cdot a_t^{[i-1](k)} + b_j^{[i]} \right)$$

Camada [i]

Sample (k)

Neurônio j

Neurônio t da camada [i - 1]

Uma rede neural muito simples

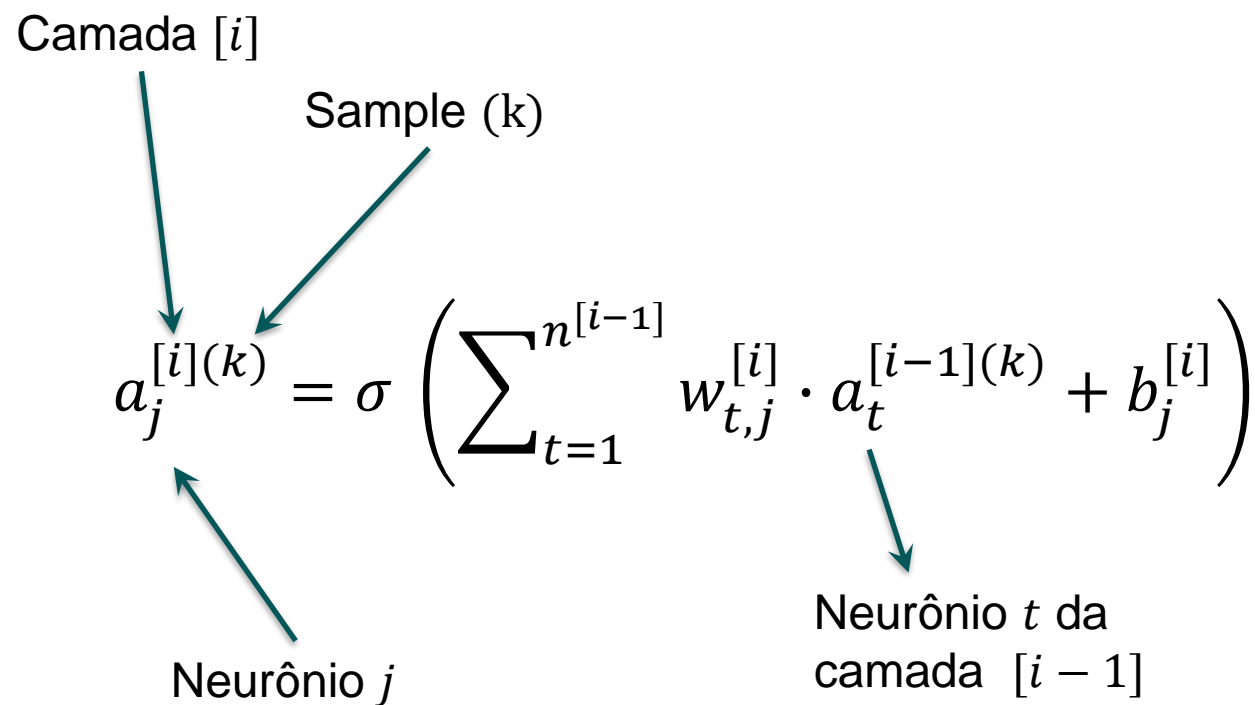
Camada $[i]$

Sample (k)

$$a_j^{[i](k)} = \sigma \left(\sum_{t=1}^{n^{[i-1]}} w_{t,j}^{[i]} \cdot a_t^{[i-1](k)} + b_j^{[i]} \right)$$

Neurônio j

Neurônio t da camada $[i-1]$



```
for k in range(number of samples):  
    for i in range(number of layers)  
        ...
```

Uma rede neural muito simples

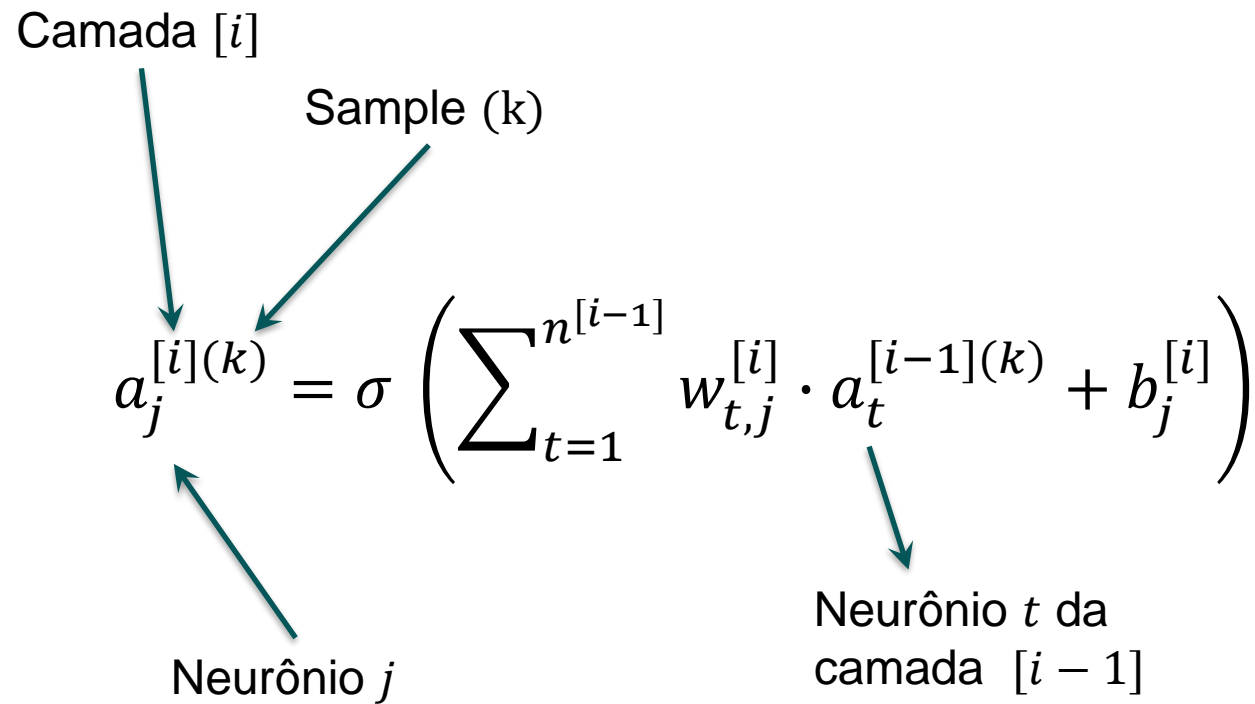
Camada $[i]$

Sample (k)

$$a_j^{[i](k)} = \sigma \left(\sum_{t=1}^{n^{[i-1]}} w_{t,j}^{[i]} \cdot a_t^{[i-1](k)} + b_j^{[i]} \right)$$

Neurônio j

Neurônio t da camada $[i-1]$



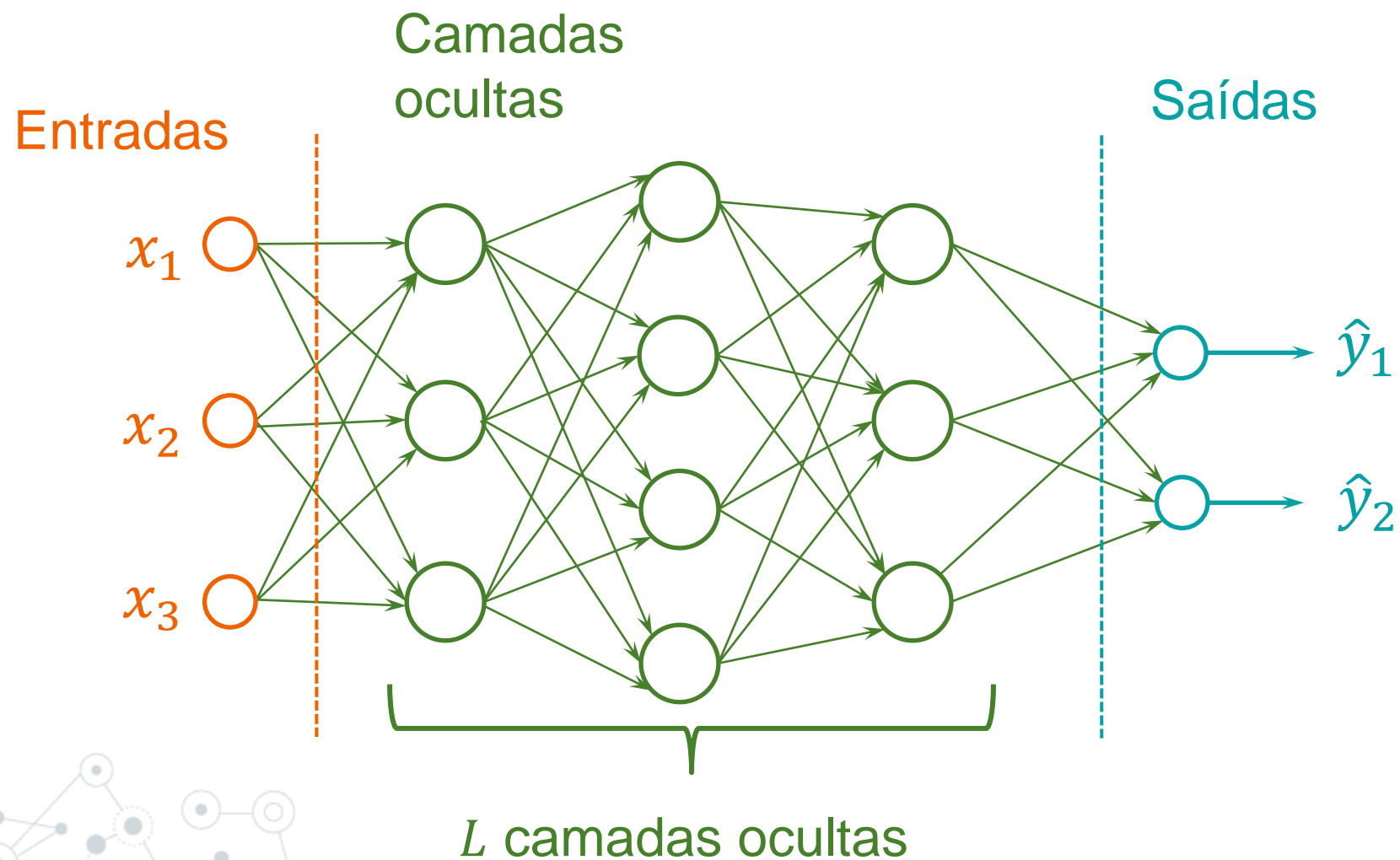
```
for k in range(number of samples):  
    for i in range(number of layers)  
        ...
```



Uma rede neural muito simples

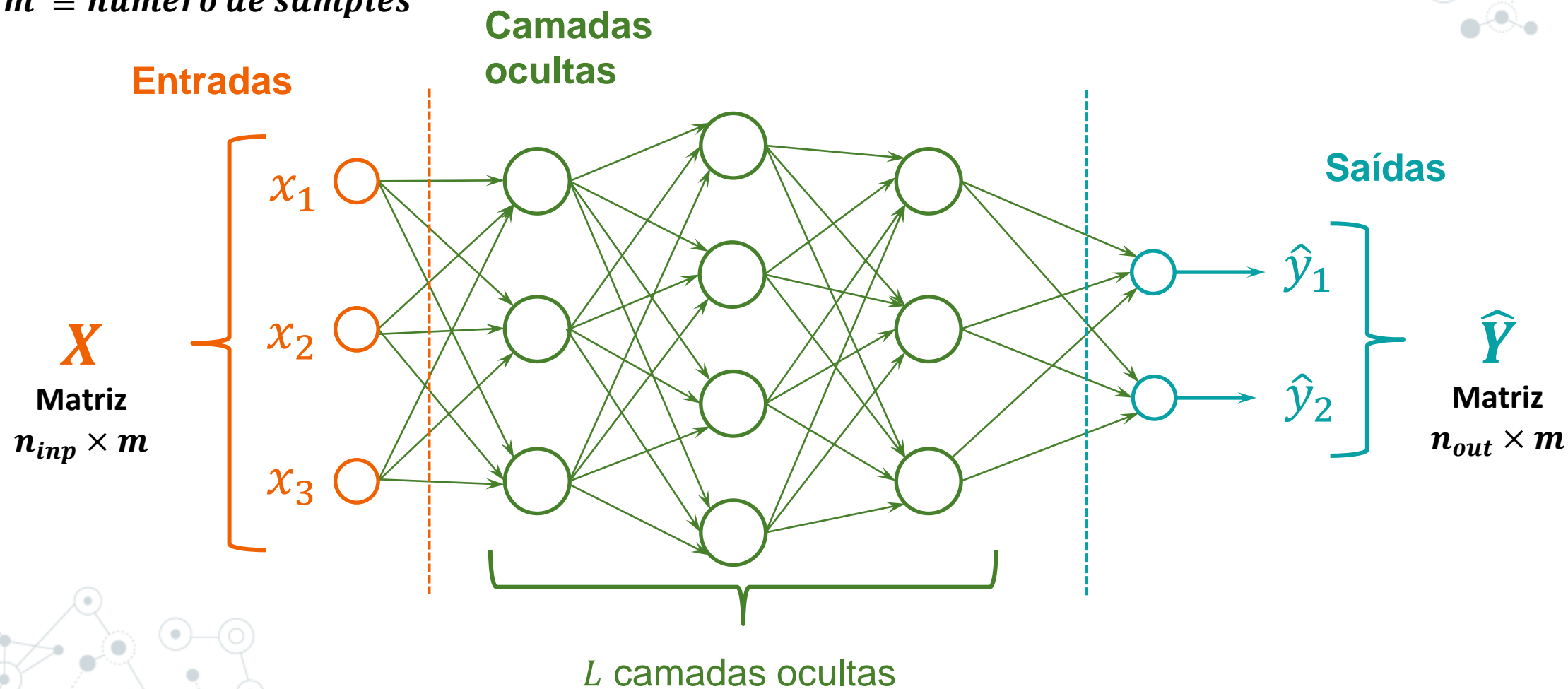
- ◎ A definição termo a termo, neurônio por neurônio, é fundamental para entendermos o conceito... porém:
 - ◎ Muito complexo de se equacionar para redes com mais do que alguns poucos neurônios
 - ◎ Muito complexo de se implementar para redes com mais do que alguns poucos neurônios
 - ◎ [Muito!] Ineficiente computacionalmente, especialmente em Python
- ◎ Solução:
 - ◎ Formulação tensorial (Vetores, Matrizes, “Matrizes 3D”)
 - ◎ Simples de equacionar e implementar
 - ◎ Eficiente computacionalmente

Uma rede neural muito simples



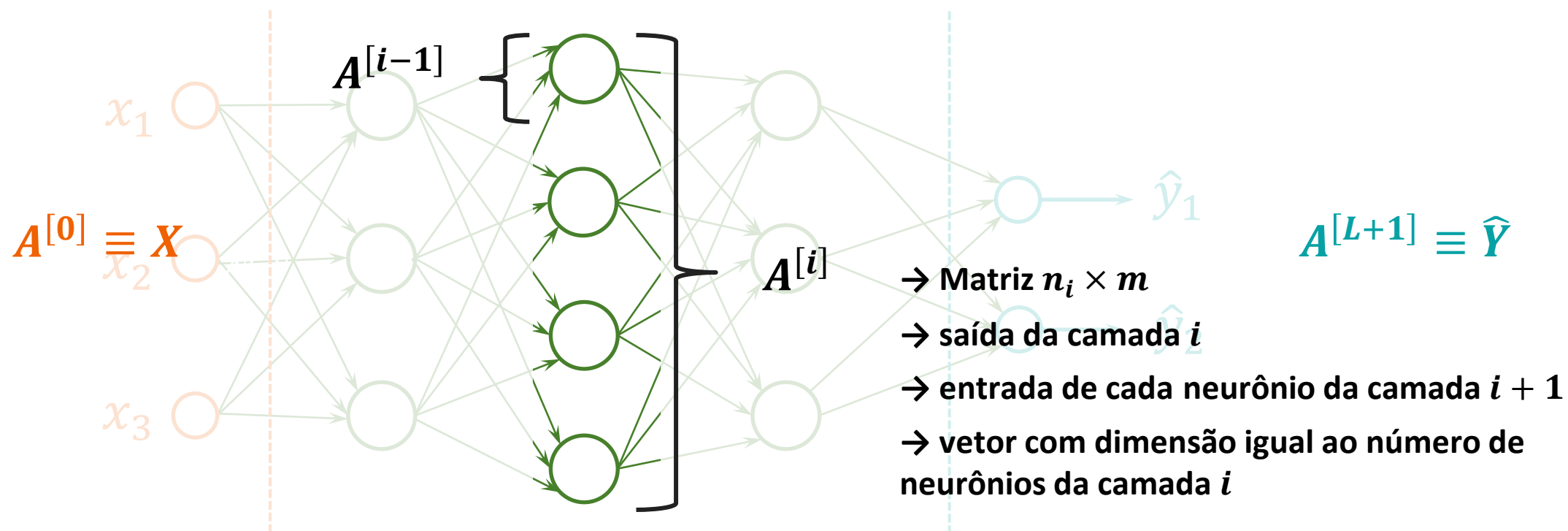
Uma rede neural muito simples

$m \equiv \text{numero de samples}$



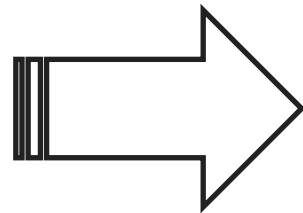
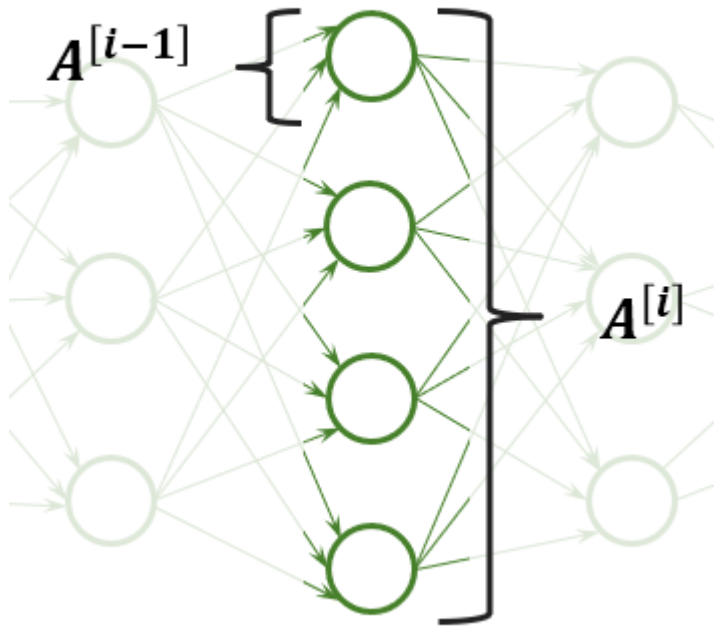
Uma rede neural muito simples

Camada i



Uma rede neural muito simples

Camada i



$$A^{[1]} = \sigma(W^{[1]} \cdot X + b^{[1]})$$

$$A^{[2]} = \sigma(W^{[2]} \cdot A^{[1]} + b^{[2]})$$

$$\vdots$$

$$A^{[L]} = \sigma(W^{[L]} \cdot A^{[L-1]} + b^{[L]})$$

$$Y = \sigma(W^{[L+1]} \cdot A^{[L]} + b^{[L+1]})$$



$$A^{[i]} = \sigma(W^{[i]} \cdot A^{[i-1]} + B^{[i]})$$

Uma rede neural muito simples

$$A^{[i]} = \sigma(W^{[i]} \cdot A^{[i-1]} + b^{[i]})$$



$b^{[i]} \rightarrow$ vetor de biases $(n_i \times 1)$

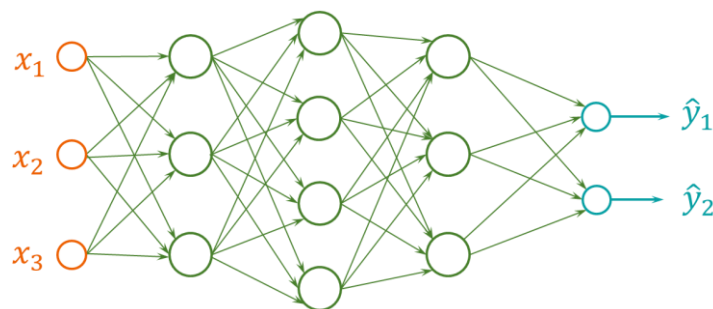
$W^{[i]} \rightarrow$ matriz de pesos $(n_i \times n_{i-1})$

Parâmetros treináveis da camada!

Uma rede neural muito simples

Quantos parâmetros treináveis temos num MLP?

Exemplo:



$$n_{inp} = 3$$

$$n_{out} = 2$$

$$L = 3$$

$$n = \{3, 4, 3\}$$

$$\sum_{i=1}^{L+1} n_i \cdot n_{i-1} + b_i$$

$$\begin{aligned} N_{\theta} &= (3 \cdot 3 + 3) \\ &\quad + (4 \cdot 3 + 4) \\ &\quad + (3 \cdot 4 + 3) \\ &\quad + (2 \cdot 3 + 2) \\ &= 51 \end{aligned}$$

Uma rede neural muito simples

Como, enfim, treinamos a rede?

Ou seja,

Como definimos o valor dos parâmetros treináveis?

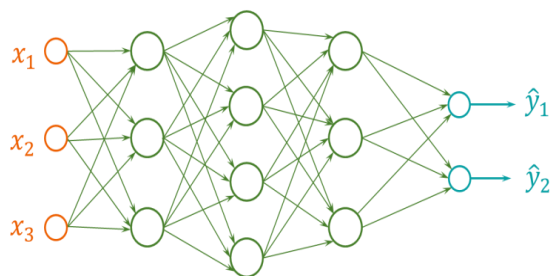


Como treinar sua rede neural

Como treinar sua rede neural

X
Features

Y
Labels reais



\hat{Y}
Labels previstos pelo modelo

Rede Neural definida
conjunto de
parâmetros Θ

← Pesos, biases, etc

Loss function:
métrica de erro
para finalidade de
treinamento

$$\mathcal{L}(Y, \hat{Y})$$

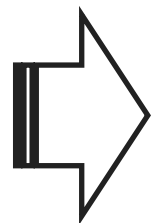


**Objetivo do
treinamento:**
encontrar Θ que
minimize a loss
function!

Como treinar sua rede neural

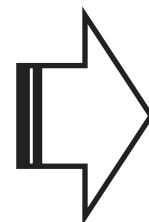
Loss function:
métrica de erro
para finalidade de
treinamento

$$\mathcal{L}(Y, \hat{Y})$$



$$\mathcal{L}(Y, \hat{Y}, \Theta)$$

$\Theta \equiv$ vetor
contendo todos os
parâmetros
treináveis



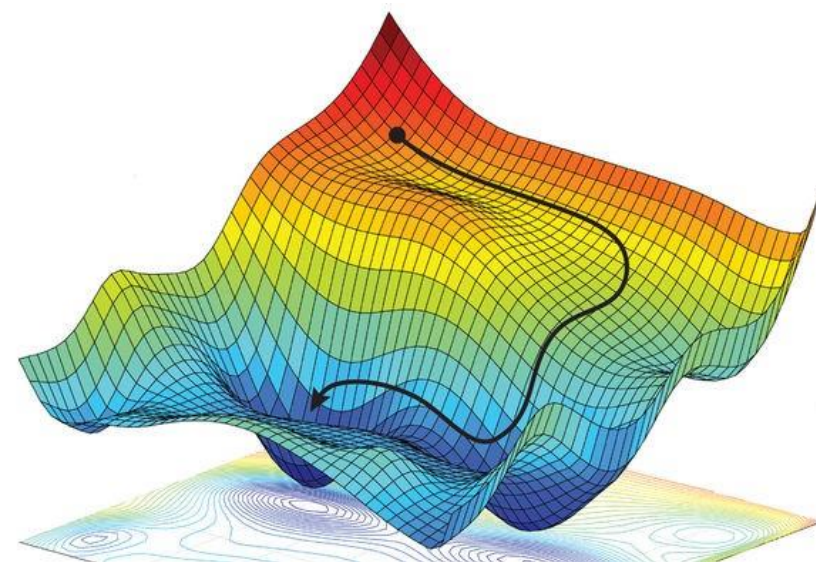
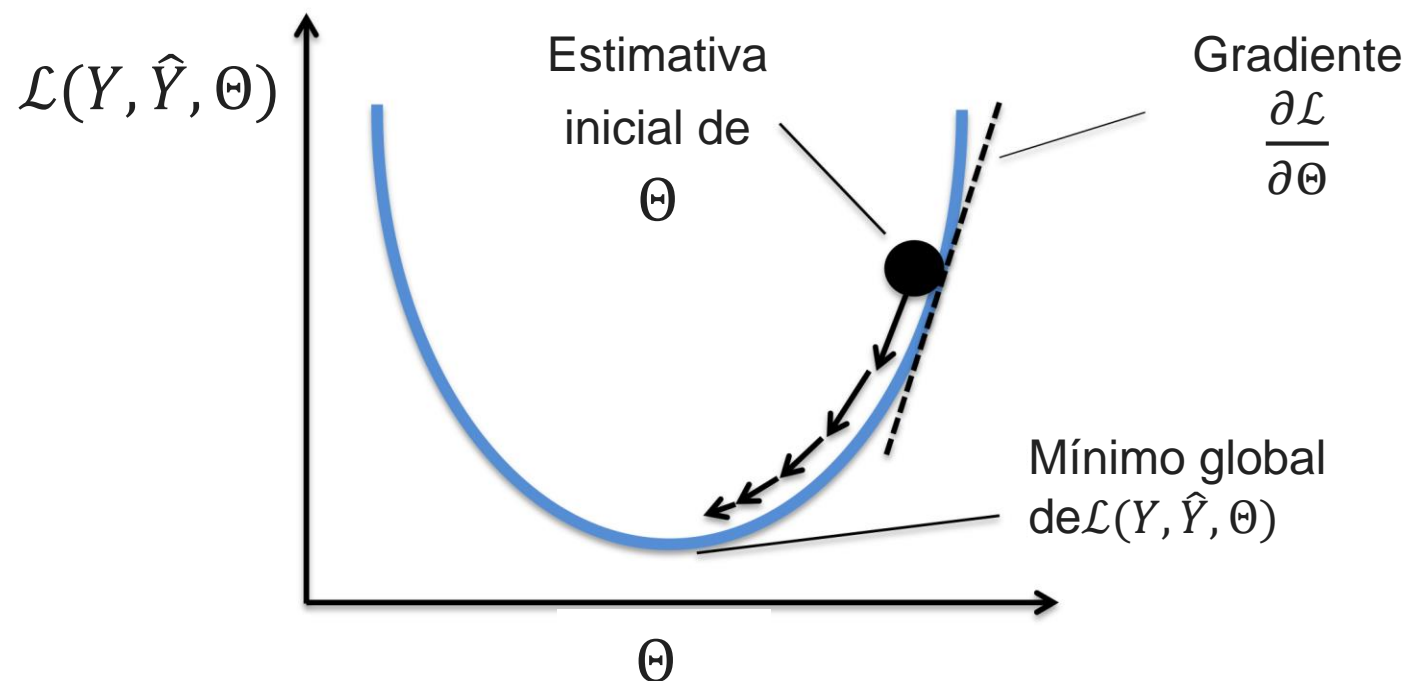
Algoritmo de minimização:

Gradient descent

**Objetivo do
treinamento:**
encontrar Θ que
minimize a *loss
function!*

Como treinar sua rede neural

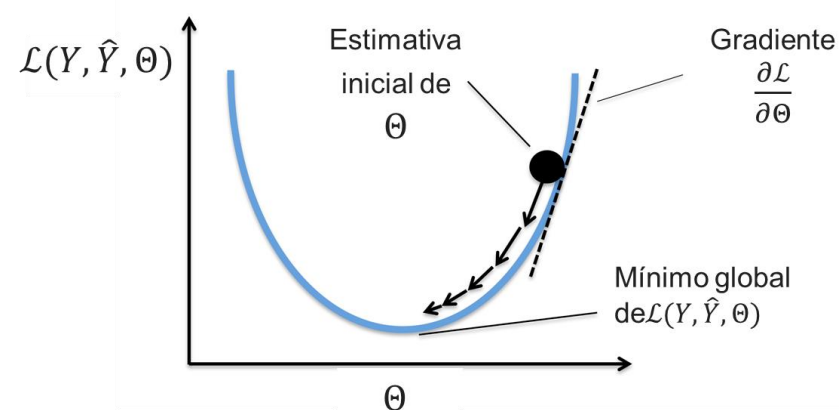
Gradient descent



Como treinar sua rede neural

Gradient descent

Seja θ^i um elemento de Θ na iteração i :



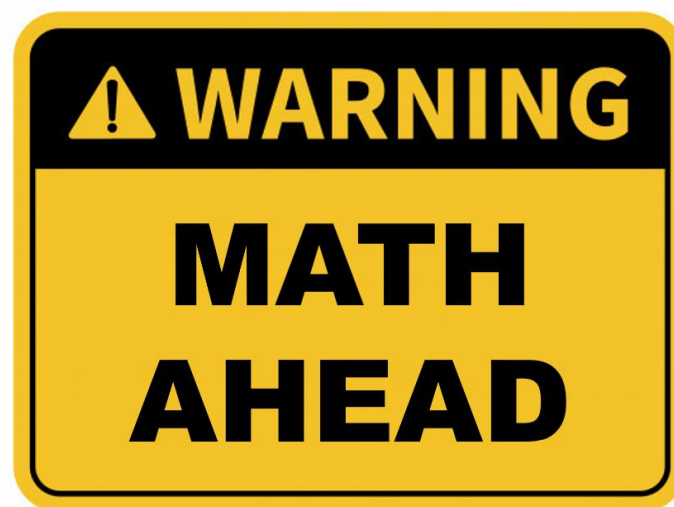
$$\theta^i = \theta^{i-1} - \alpha \cdot \frac{\partial \mathcal{L}}{\partial \theta}$$



$\alpha \rightarrow$ taxa de aprendizagem / learning rate

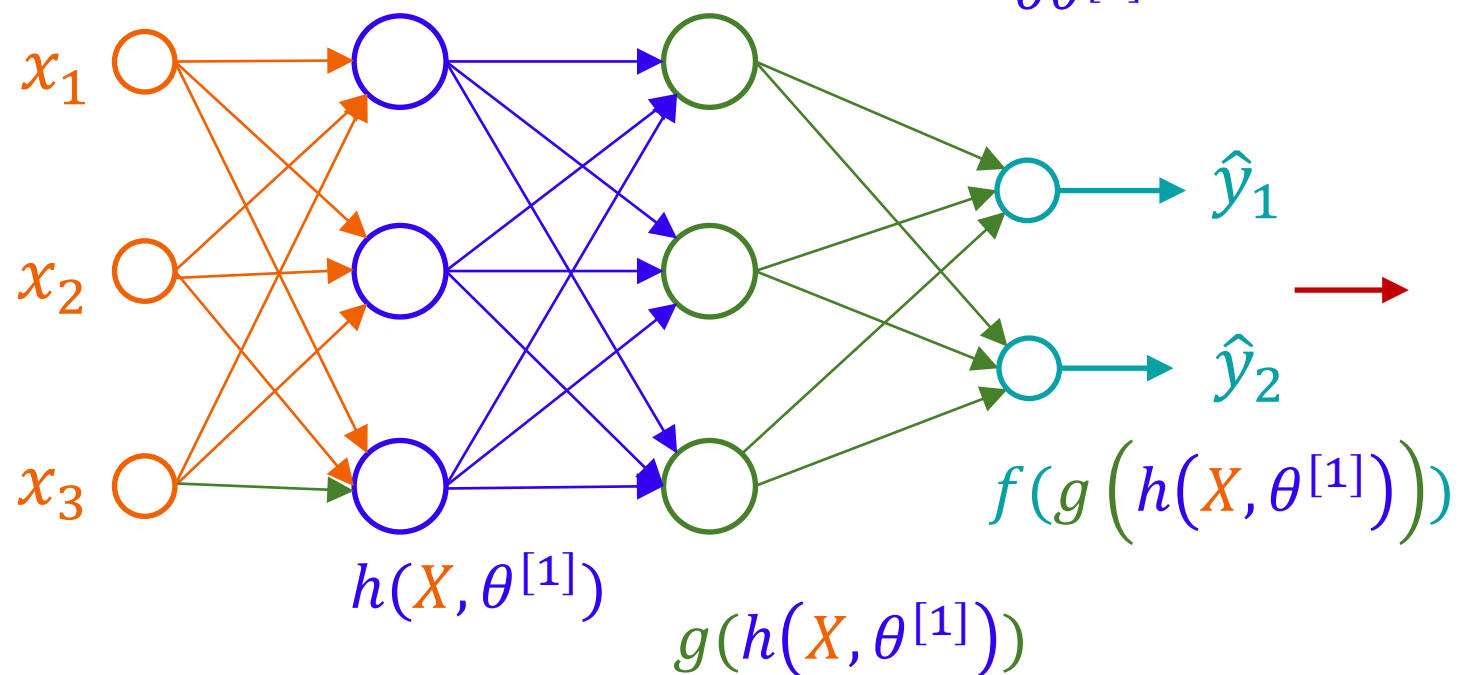
Como treinar sua rede neural

Gradient descent



Como treinar sua rede neural

Analogia com funções: cálculo de $\frac{\partial \mathcal{L}}{\partial \theta^{[1]}}$



$$\mathcal{L}(Y, f(g(h(X, \theta^{[1]}))))$$

Como calcular $\frac{\partial \mathcal{L}}{\partial \theta^{[1]}}$?

R: regra da cadeia

* Os demais $\theta^{[i]}$ estão omitidos por clareza

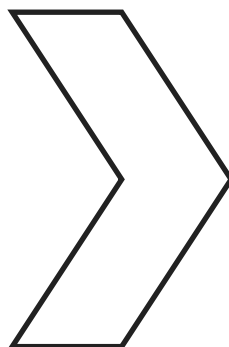
Regra da cadeia

$$y = f(u)$$

$$u = g(x)$$

\therefore

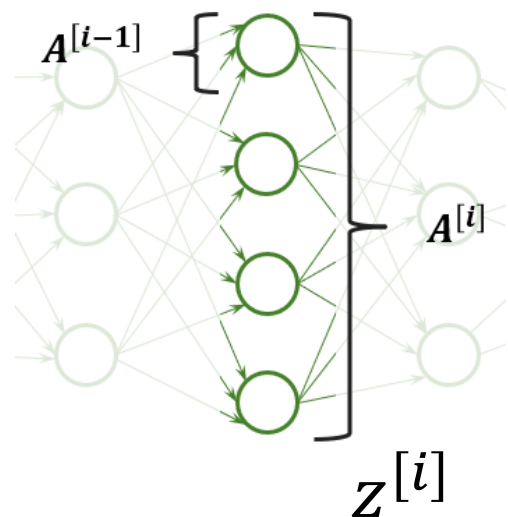
$$y = f(g(x))$$



$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial x}$$

Como treinar sua rede neural

Gradient descent: $\theta^i = \theta^{i-1} - \alpha \cdot \frac{\partial \mathcal{L}}{\partial \theta}$



$$A^{[i]} = \sigma(W^{[i]} \cdot A^{[i-1]} + B^{[i]})$$

$$\frac{\partial \mathcal{L}}{\partial z^{[i]}} = W^{[i+1]T} \frac{\partial \mathcal{L}}{\partial z^{[i+1]}} * \sigma^{[i]'}(z^{[i]})$$

$$\frac{\partial \mathcal{L}}{\partial W^{[i]}} = \frac{\partial \mathcal{L}}{\partial z^{[i]}} a^{[i-1]T} \quad \leftarrow \text{Atualização de } W$$

$$\frac{\partial \mathcal{L}}{\partial b^{[i]}} = \frac{\partial \mathcal{L}}{\partial z^{[i]}} \quad \leftarrow \text{Atualização de } b$$

Como treinar sua rede neural

Gradient descent: $\theta^i = \theta^{i-1} - \alpha \cdot \frac{\partial \mathcal{L}}{\partial \theta}$

Depende do resultado da camada a frente:

Derivadas devem ser calculadas “da saída para a entrada”

Depende do resultado da camada atrás:

Resultados de cada camada devem ser guardados

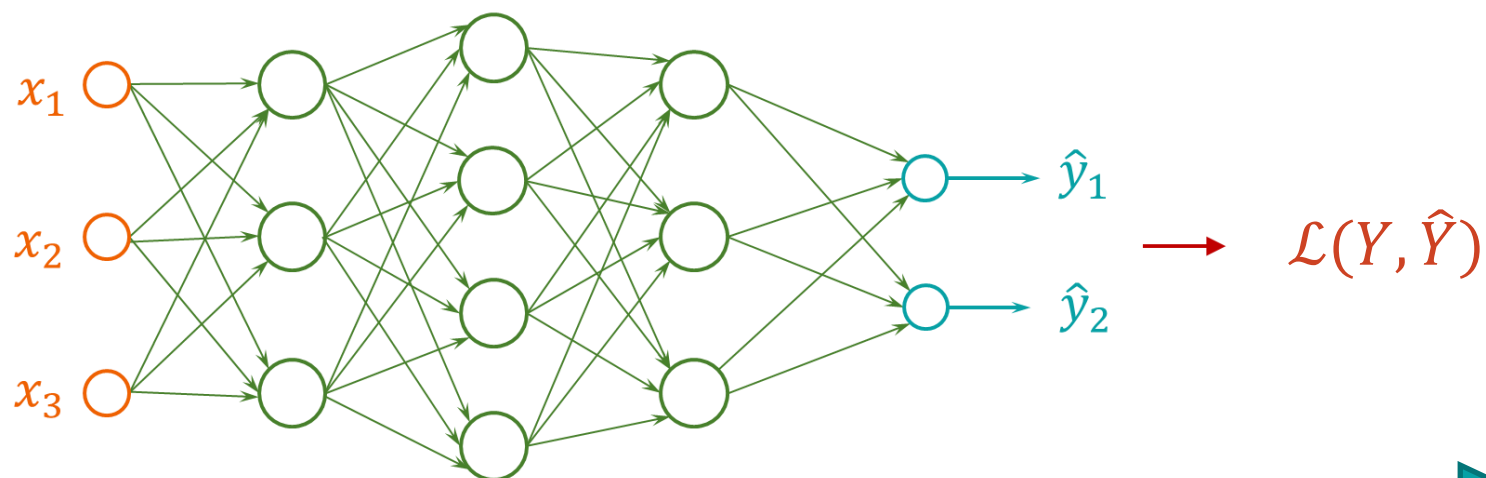
$$\frac{\partial \mathcal{L}}{\partial z^{[i]}} = W^{[i+1]T} \frac{\partial \mathcal{L}}{\partial z^{[i+1]}} * \sigma^{[i]'}(z^{[i]}) \leftarrow \text{Atualização de } W$$

$$\frac{\partial \mathcal{L}}{\partial W^{[i]}} = \frac{\partial \mathcal{L}}{\partial z^{[i]}} a^{[i-1]T}$$

$$\frac{\partial \mathcal{L}}{\partial b^{[i]}} = \frac{\partial \mathcal{L}}{\partial z^{[i]}} \leftarrow \text{Atualização de } b$$

Como treinar sua rede neural

Resumindo:



Forward propagation: cálculo das saídas camada a camada até \hat{Y} e $\mathcal{L}(Y, \hat{Y})$

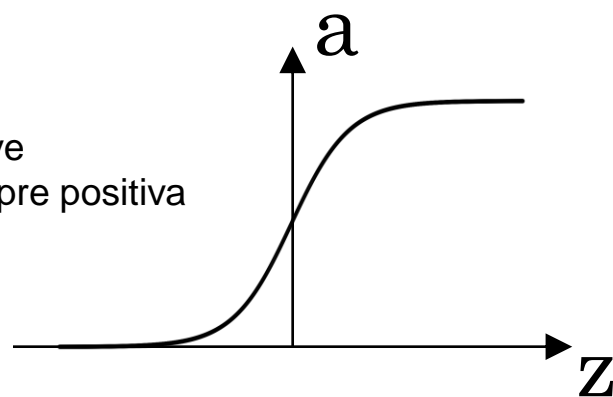
Backward propagation: cálculo das derivadas $\partial \mathcal{L} / \partial \theta$ camada a camada até a primeira, atualizando os parâmetros via gradient descent

Como treinar sua rede neural

- ⦿ Aprendizado supervisionado em redes neurais:
 - ⦿ Back propagation
 - +
 - ⦿ Gradient Descent e suas variações
- ⦿ Válido para outros tipos de redes neurais
 - ⦿ Rede de convolução – CNN
 - ⦿ Redes recorrentes – RNN (LSTM, GRU, etc)

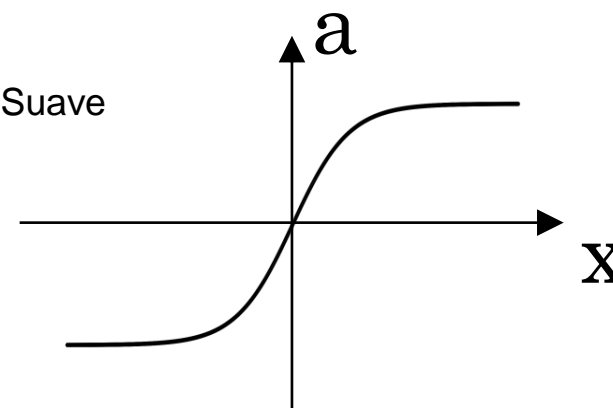
Tipos de função de ativação

- Suave
- Sempre positiva



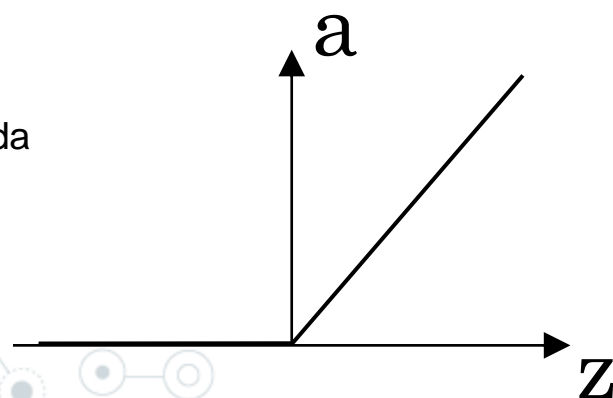
Sigmoid

- Suave



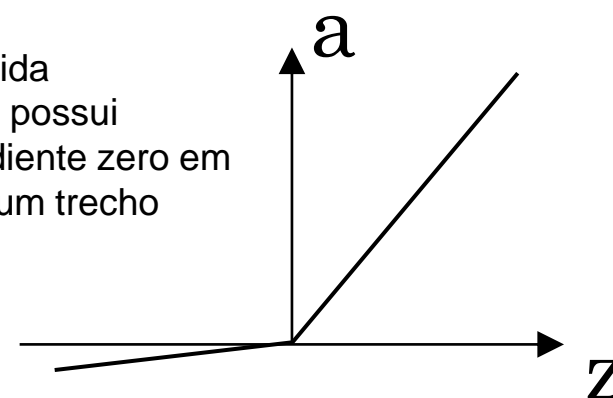
Tanh

- Rápida



ReLU

- Rápida
- Não possui gradiente zero em nenhum trecho



LeakyReLU

Funções de ativação – camada output

- ⦿ Regressão
 - ⦿ Função identidade

- ⦿ Classificação binária – duas classes
 - ⦿ Sigmoid

- ⦿ Classificação múltiplas classes
 - ⦿ Softmax

Problemas na propagação dos gradientes

- ◎ Gradientes se propagam ao longo das camadas
- ◎ Em redes com muitas camadas os gradientes podem apresentar dois problemas
 - ◎ **Vanishing gradients:** gradientes se tornam zeros e “bloqueiam” o treinamento de um conjunto de neurônios
 - ◎ **Exploding gradients:** valor dos gradientes explodem, podendo causar problemas numéricos na computação

Conclusões para guardar na memória

- ◎ Uma rede neural é uma função complexa construída a partir de funções simples
- ◎ O que são os elementos de uma rede neural: camadas, neurônios, pesos, ativação, etc
- ◎ O que é, conceitualmente, back propagation e gradiente descent
- ◎ ~~Equações~~ Equações → quando precisar consulte um livro ou fonte confiável
- ◎ Relação entre número de neurônios e parâmetros treináveis
- ◎ Relação entre número de camadas e problemas na propagação dos gradientes

Machine Learning

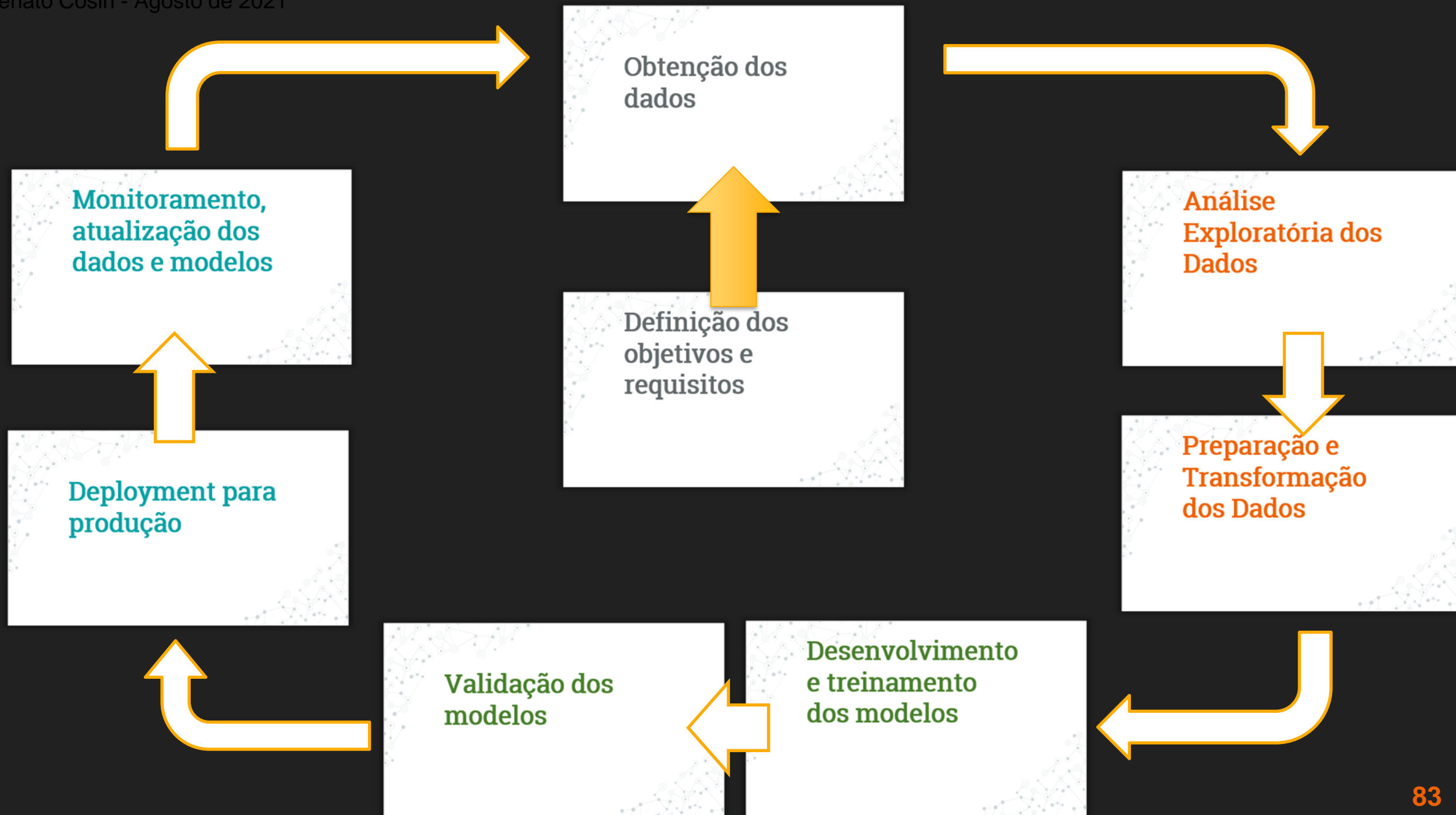
Parte 2

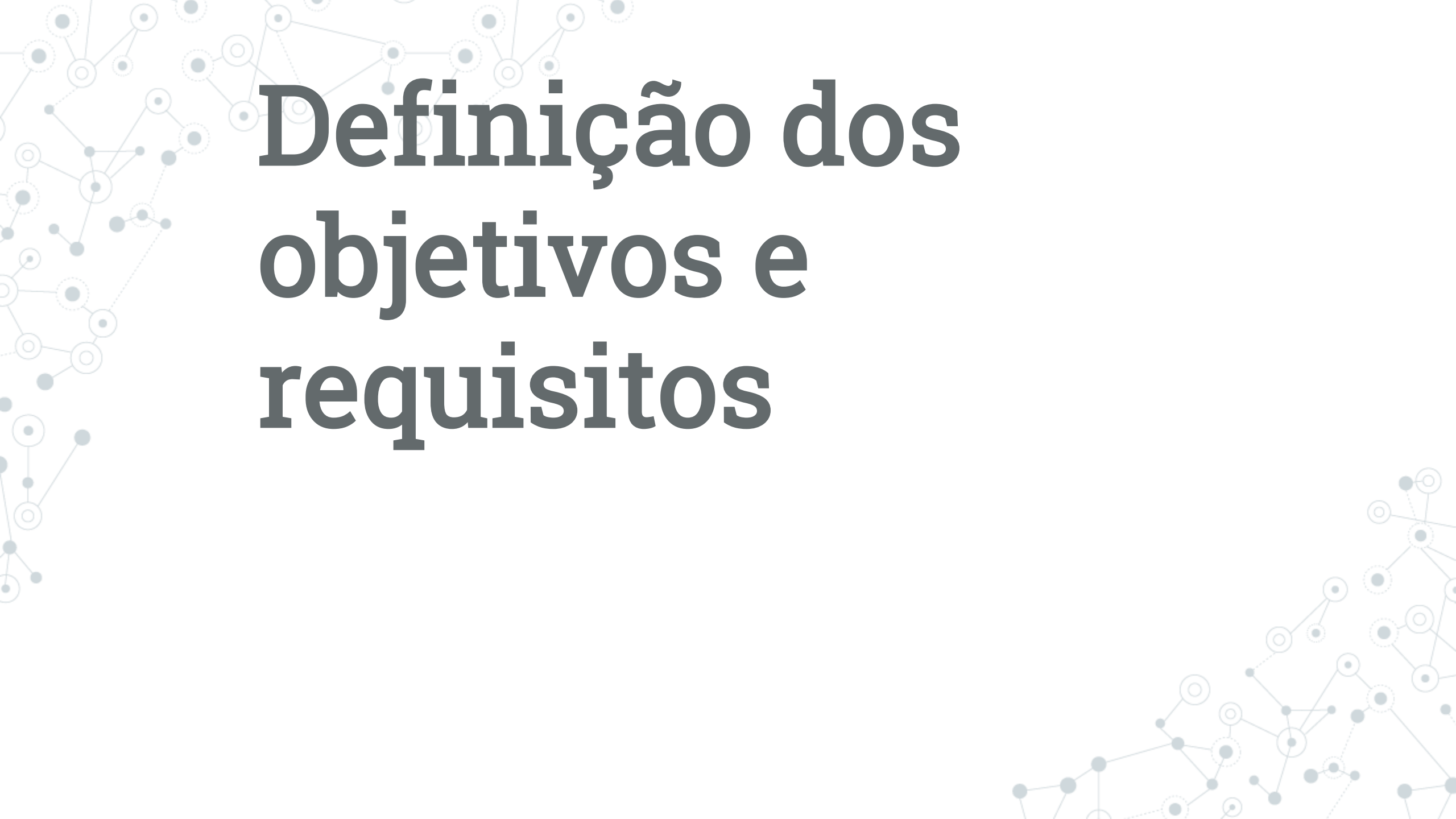
*Introdução e aplicação na engenharia
aeronáutica*

Renato Cosin



Workflow





Definição dos objetivos e requisitos

Definição dos objetivos e requisitos

- ◎ Qual é a necessidade que levou ao uso do ML?
 - ◎ Existe a necessidade de se fazer uma otimização de desempenho de uma aeronave para uma determinada missão
 - ◎ O modelo de desempenho é demasiadamente demorado dado o poder computacional disponível
 - ◎ O uso de meta modelo se faz necessário
- ◎ O que meu modelo precisa fazer?
 - ◎ Reproduzir resultados de modelo de desempenho para uma missão específica
- ◎ Quão acurado precisa ser?
 - ◎ Combustível utilizado: $\pm 5.0\%$
 - ◎ Demais resultados: $\pm 7.5\%$
- ◎ Que tipos de dados estarão disponíveis?
 - ◎ Resultados de estudo paramétrico com modelo de performance
 - ◎ DOE: hiper cubo latino
 - ◎ Número de variáveis: 9; Reais (float)
 - ◎ Número de saídas: 5; Reais (float)
- ◎ Em que volume?
 - ◎ 10k samples

Definição dos objetivos e requisitos

◎ Quais são os resultados esperados?

- ◎ Atender requisito de acurácia em todo espaço de projeto coberto pelos dados de treinamento
- ◎ Tempo de inferência para 1k sample < 10 segundos

◎ Qual é a métrica de acurácia?

- ◎ Erro percentual

◎ Que poder computacional estará disponível para treinamento?

- ◎ CPU: Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz
- ◎ Memória: 16GB RAM
- ◎ GPU: NVIDIA GeForce 940MX; 4GB de memória dedicada

◎ E para inferência?

- ◎ Idem

◎ Que tipo de ML poderei usar?

Definição dos objetivos e requisitos

☉ Que tipo de ML poderei usar?

☉ Aprendizado supervisionado → Regressão

Modelo	Vantagens*	Desvantagens*
Mínimos quadrados	Simple, disponível em muitas ferramentas, poucos hiperparâmetros	Problema excessivamente complexo, muitas dimensões
Redes Neurais MLP	Amplamente disponível; muitas opções de arquitetura; complexidade escalável, lida bem com alta dimensionalidade de não linearidades	Excesso de opções pode dificultar escolha da arquitetura e hiperparâmetros; pode resultar em overfitting
Support Vector Machines Regression - SVR	complexidade escalável, lida bem com alta dimensionalidade de não linearidades	Pouca documentação, menor versatilidade em relação as NNs
CNN	Não se aplica	
RNN	Não se aplica	

* Para este problema específico

A decorative network diagram in the top-left corner, featuring a cluster of interconnected nodes. Some nodes are solid dark grey circles, while others are hollow circles with a dark grey outline. They are connected by thin, light grey lines, forming a complex web-like structure.

Obtenção dos dados

A decorative network diagram in the bottom-right corner, similar to the one in the top-left. It consists of a cluster of interconnected nodes, some solid dark grey circles and some hollow circles with dark grey outlines, connected by thin, light grey lines.

Obtenção dos dados

© Dados simulados → objetivo do metamodelo

Parâmetros fixos – missão da aeronave		Valor
Altitude de decolagem	[ft]	0
Altitude de pouso	[ft]	3000
Payload	[kg]	22000
Delta Temperatura ISA	[°C]	0.0
Razão de descida média	[ft/min]	3000
Velocidade calibrada de descida média	[kt]	250
Mínima razão de subida residual	[ft/min]	300

Obtenção dos dados

Parâmetros - entrada		Valor mínimo	Valor máximo
Area alar	[m ²]	80	200
Alongamento	[--]	7	14
Enflechamento 14/ da corda	[graus]	15	35
Máximo empuxo estático	[lbf]	20000	30000
Velocidade calibrada de subida	[kt]	200	300
Número de Mach máximo de subida	[--]	0.45	0.65
Número de Mach de cruzeiro	[--]	0.7	0.85
Altitude target de cruzeiro	[ft]	30000	45000
Distância da missão	[nm]	1000 constante	

Obtenção dos dados

Parâmetros – Saída

Peso de decolagem	[kg]
-------------------	------

Peso básico operacional	[kg]
-------------------------	------

Peso de combustível	[kg]
----------------------------	-------------

<i>Distância da missão – Verificação de convergência</i>	[nm]
--	------

Tempo da missão	[min]
-----------------	-------

Altitude de cruzeiro	[ft]
----------------------	------

Obtenção dos dados

- ◎ Dataset 1:
 - 200 pontos
 - Apenas peso de combustível como saída
- ◎ Dataset 2:
 - 10 mil pontos
 - Todas as saídas
- ◎ Dataset 3:
 - 100 mil pontos
 - Distância variando
 - Todas as saídas
- ◎ Dataset 4:
 - Dataset 1 + ruído



Análise Exploratória dos Dados



Preparação e Transformação dos Dados

Preparação e Transformação dos Dados

- ◎ Limpeza / filtragem dos dados
- ◎ Remoção de outliers
- ◎ Seleção das features a serem utilizadas
- ◎ Criação de novas features
- ~~◎ Data augmentation~~
- ~~◎ Balanceamento dos dados~~
- ◎ Segmentação dos dados
 - ◎ Dados de treinamento
 - ◎ Dados de teste
 - ~~◎ Dados de avaliação~~



Desenvolvimento e treinamento dos modelos

Desenvolvimento e treinamento dos modelos

- ◎ Definição dos algoritmos: MLP
- ◎ Implementação dos modelos: Keras...
- ◎ Definição / otimização do hiperparâmetros
 - Número de camadas / neurônios
 - Função de ativação
 - Learning rate

A decorative network diagram in the top-left corner, consisting of a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are solid grey and others are hollow with a grey outline. The lines are thin and grey, connecting the nodes in a non-linear fashion. The overall shape of the network is roughly triangular, pointing towards the top-left corner.

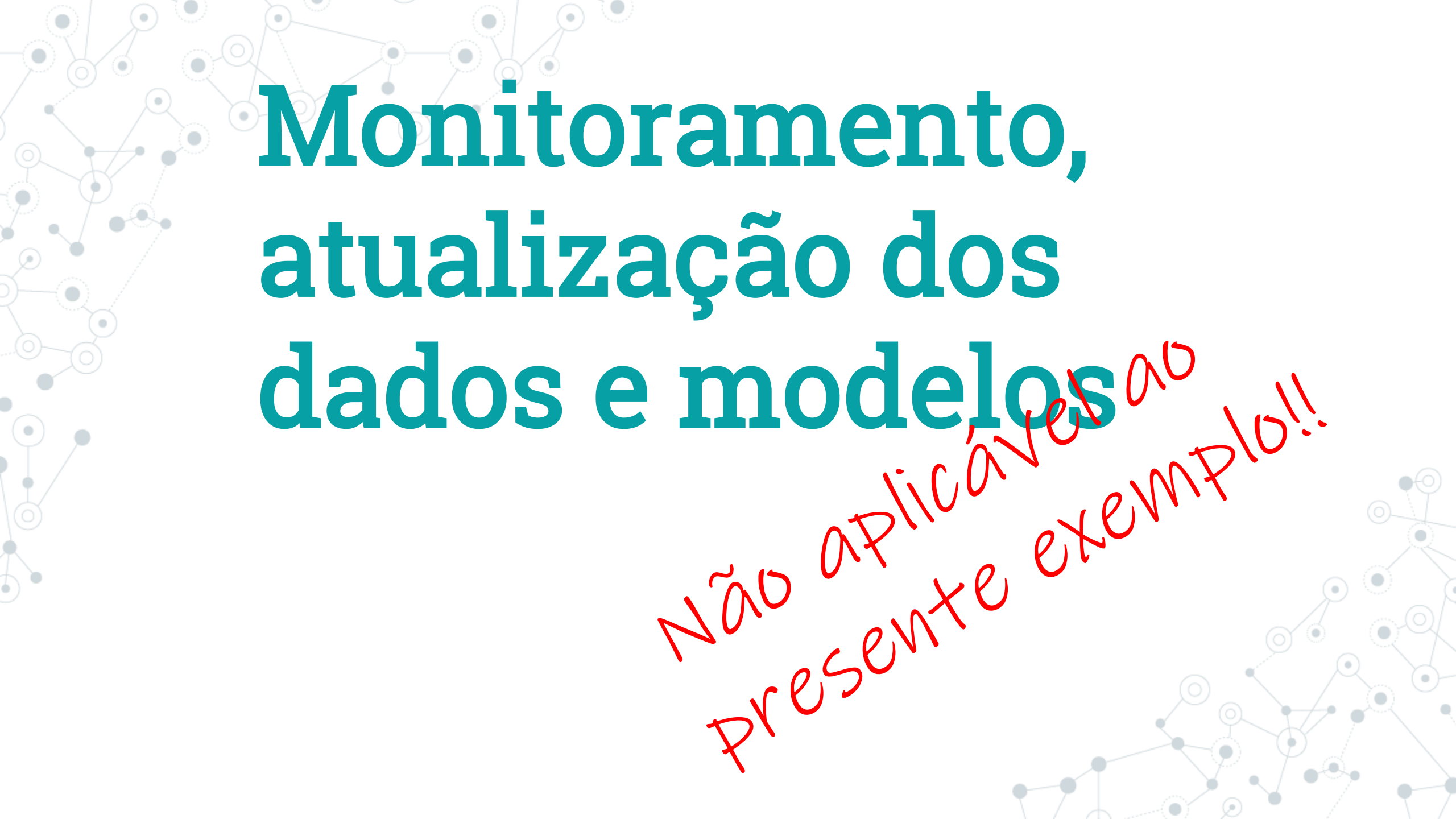
Validação dos modelos

A decorative network diagram in the bottom-right corner, similar to the one in the top-left. It features a complex web of interconnected nodes and lines. The nodes are represented by small circles, some solid grey and some hollow with a grey outline. The lines are thin and grey, connecting the nodes in a non-linear fashion. The overall shape of the network is roughly triangular, pointing towards the bottom-right corner.



Deployment para produção

*Não aplicável ao
presente exemplo!!*

A faint, light gray background pattern consisting of a network of interconnected nodes and lines, resembling a molecular structure or a data network, is visible across the entire slide.

Monitoramento, atualização dos dados e modelos

*Não aplicável ao
presente exemplo!!*



Introdução ao Tensorflow e Keras

O Tensorflow

- © Plataforma de código aberto para Machine Learning
- © Foco em treinamento e inferência de modelos de Deep Learning
- © Funções de baixo nível
- © API's de alto nível. Ex.: Keras
- © Diferenciação automática
- © Gráfos
- © Suporte ao uso de GPU

Gráficos e Tensores no Tensorflow

- ◎ Gráficos: estruturas de dados que contenham um conjunto de objetos que representam as unidades de computação
 - Exemplos:
 - ◎ Neurônio
 - ◎ Multiplicação de matrizes
 - ◎ Convolução de matrizes
 - ◎ Camada de rede neural
- ◎ podem ser salvos, executados e restaurados sem o código Python original

Gráficos e Tensores no Tensorflow

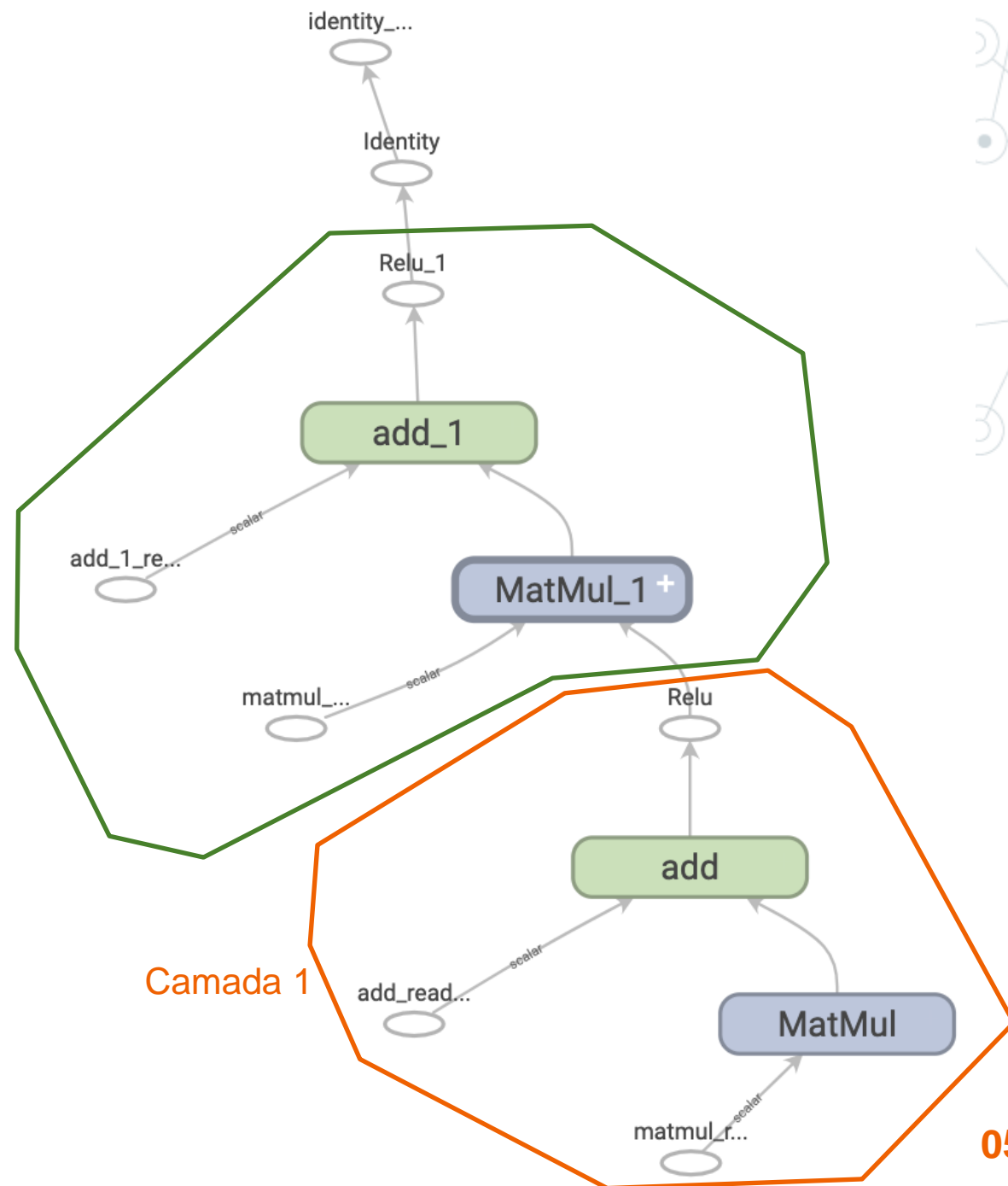
- ◎ Tensores: estrutura de dados que fluem entre os objetos dos gráficos.
 - ◎ Escalares, vetores, matrizes, matrizes multidimensionais
 - ◎ Tipo constante em todos os elementos
 - ◎ Similar ao `numpy.ndarray`

Gráficos e Tensores no Tensorflow

Exemplo de gráfico de rede neural de duas camadas:

Camada 2

Camada 1



Diferenciação automática

- © Cálculo do gradiente de uma computação com respeito a algumas entradas
- © Extremamente útil implementação de algoritmos de aprendizagem de máquina, como backpropagation

Keras

- ◎ API de alto nível para redes neurais do Tensorflow
- ◎ Funções rápida para criação de redes neurais
- ◎ Utiliza back-end do Tensorflow
- ◎ Implementação pronta para tipos de redes neurais mais comuns
- ◎ Suporta CNN's e RNN's

Pandas

- ◎ Biblioteca Python para manipulação e análise de dados
- ◎ Estrutura de dados: DataFrame
- ◎ Dados tabulares
 - ◎ tipo único por coluna
 - ◎ Análogo a Excel ou SQL
- ◎ Principais operações:
 - ◎ Reshape, merge, join
 - ◎ Slicing, fancy indexing, sorting
 - ◎ Filtros
 - Handling de dados faltantes

Nos vemos no Jupyter Notebook !

Arquivos anexos ! ! !

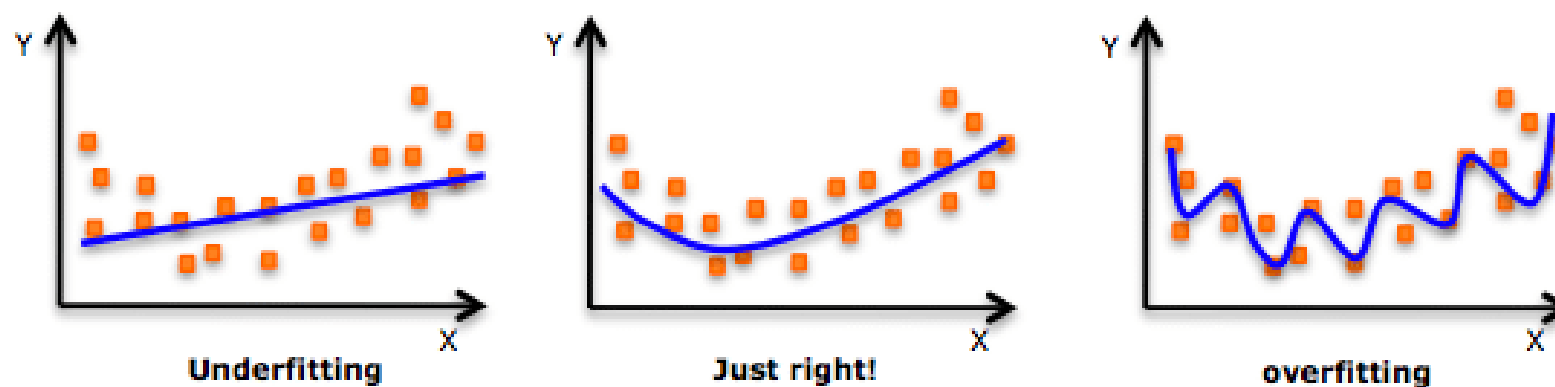


Finalização – Redes Neurais

Conteúdo

- © Overfitting e Underfitting
- © Regularização e dropout
- © Convergência e Early Stopping
- © Batch normalization

Overfitting e Underfitting

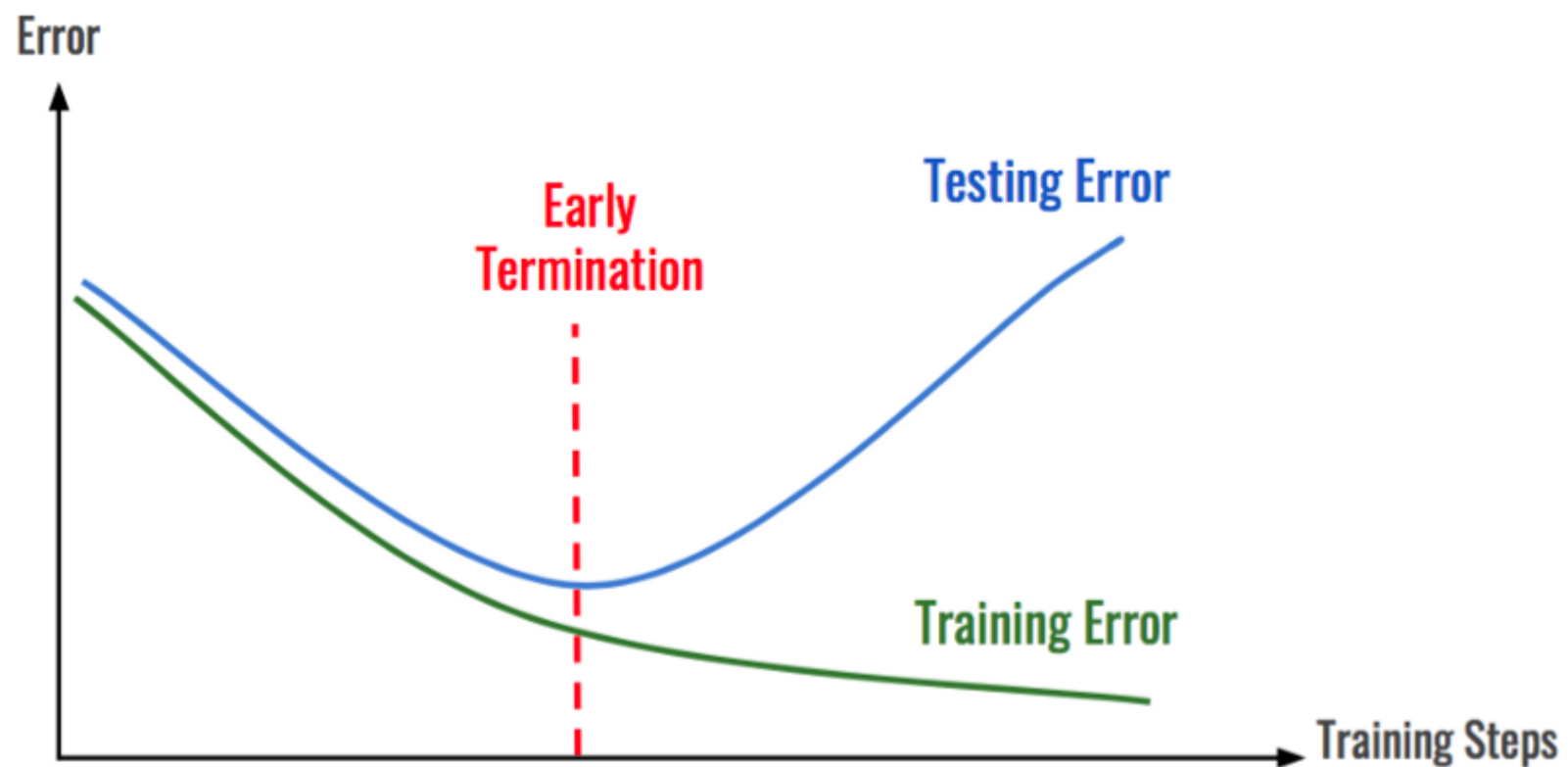


An example of overfitting, underfitting and a model that's "just right!"

Como resolver o overfitting

- © Mais dados
- © Regularização
- © Dropout
- © Early stopping

Early Stopping



Regularização e dropout

L1 Regularization

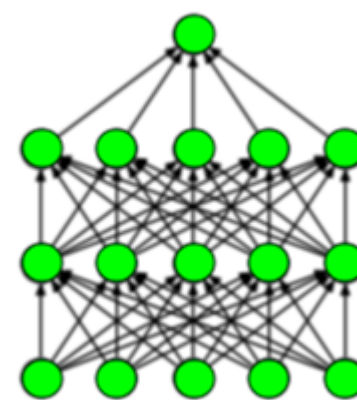
$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M |W_j|$$

L2 Regularization

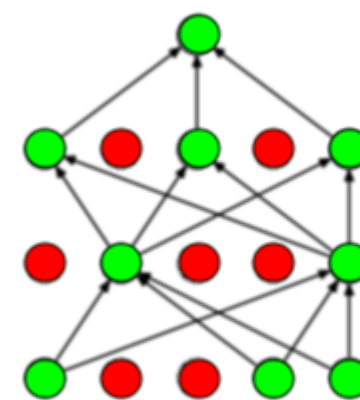
$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M W_j^2$$

Loss function

Regularization
Term



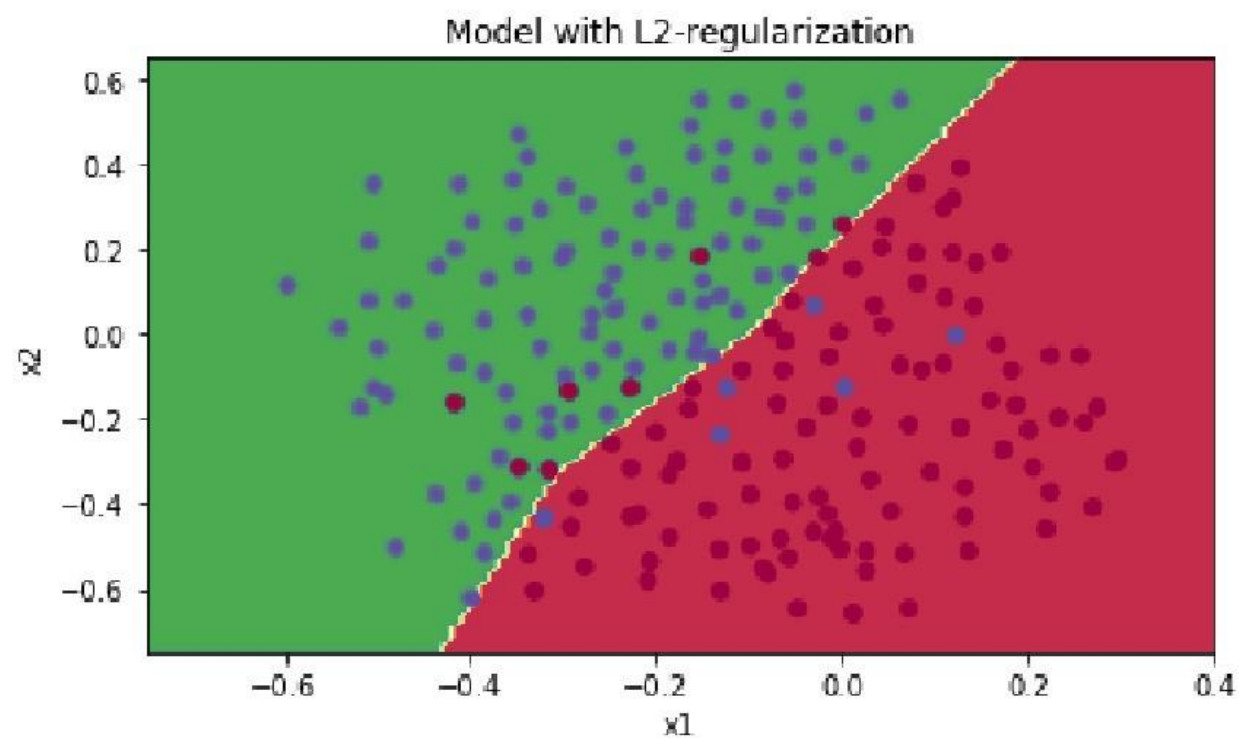
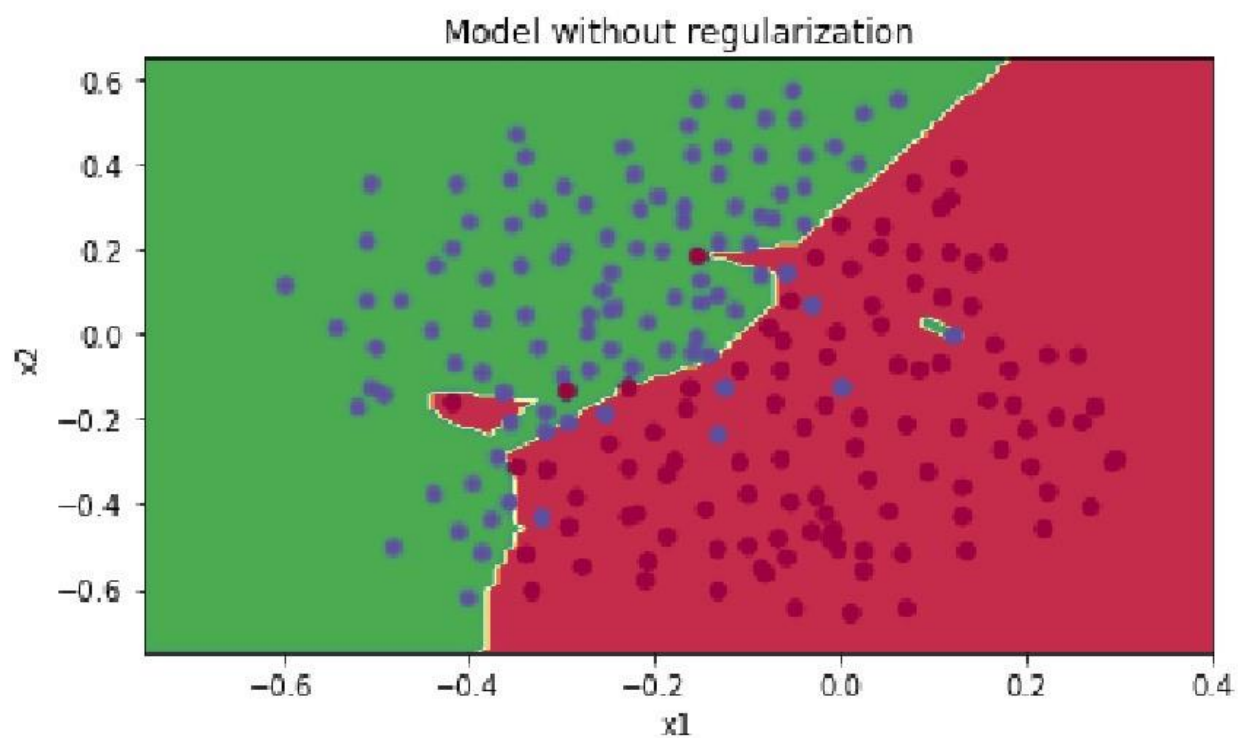
(a) Standard Neural Net



(b) After applying dropout.

Dropout – during training

Regularização e dropout



The diagram illustrates a neural network architecture with four layers: Input, BN Layer, Hidden Layer, and Output. The Input layer consists of four black circles. The BN Layer consists of three blue circles. The Hidden Layer consists of three blue circles. The Output layer consists of two red circles. The Input layer is connected to the BN Layer with weights w_1 . The BN Layer is connected to the Hidden Layer with weights γ, β . The Hidden Layer is connected to the Output layer with weights w_2 . The BN Layer and Hidden Layer are grouped together in a dashed blue box.

Referências

- © <https://www.tensorflow.org/learn>
- © <https://keras.io/api/>
- © https://pandas.pydata.org/docs/user_guide/index.html
- © Russell, Stuart J. (Stuart Jonathan), 1962 - Inteligência artificial / Stuart Russell, Peter Norvig; tradução Regina Célia Simille. – Rio de Janeiro: Elsevier, 2013.
- © Diederik P. Kingma and Jimmy Lei Ba. Adam : A method for stochastic optimization. 2014. arXiv:1412.6980v9

Referências

- ◎ <https://www.deeplearningbook.org/>
- ◎ <https://medium.com/analytics-vidhya/the-perfect-fit-for-a-dnn-596954c9ea39>
- ◎ <https://towardsdatascience.com/work-smarter-not-harder-when-building-neural-networks-6f4aa7c5ee61>
- ◎ <https://towardsdatascience.com/deep-learning-computer-vision-and-automated-optical-inspection-774e8ca529d3>
- ◎ <https://medium.com/@jorgesleonel/supervised-learning-c16823b00c13>
- ◎ <https://www.coursera.org/specializations/deep-learning>
- ◎ <https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-unsupervised-learning>
- ◎ <https://towardsdatascience.com/a-brief-overview-of-outlier-detection-techniques-1e0b2c19e561>

Instalando o Tensorflow

© <https://docs.anaconda.com/anaconda/user-guide/tasks/tensorflow/>

© <https://www.tensorflow.org/install/pip#conda>

Para quem quiser se divertir mais*:

© <https://towardsdatascience.com/installing-tensorflow-with-cuda-cudnn-and-gpu-support-on-windows-10-60693e46e781>

* Desnecessário para modelos simples / Requer placa de vídeo compatível

Obrigado!

Renato Cosin

renato.cosin@gmail.com

www.linkedin.com/in/renato-cosin

