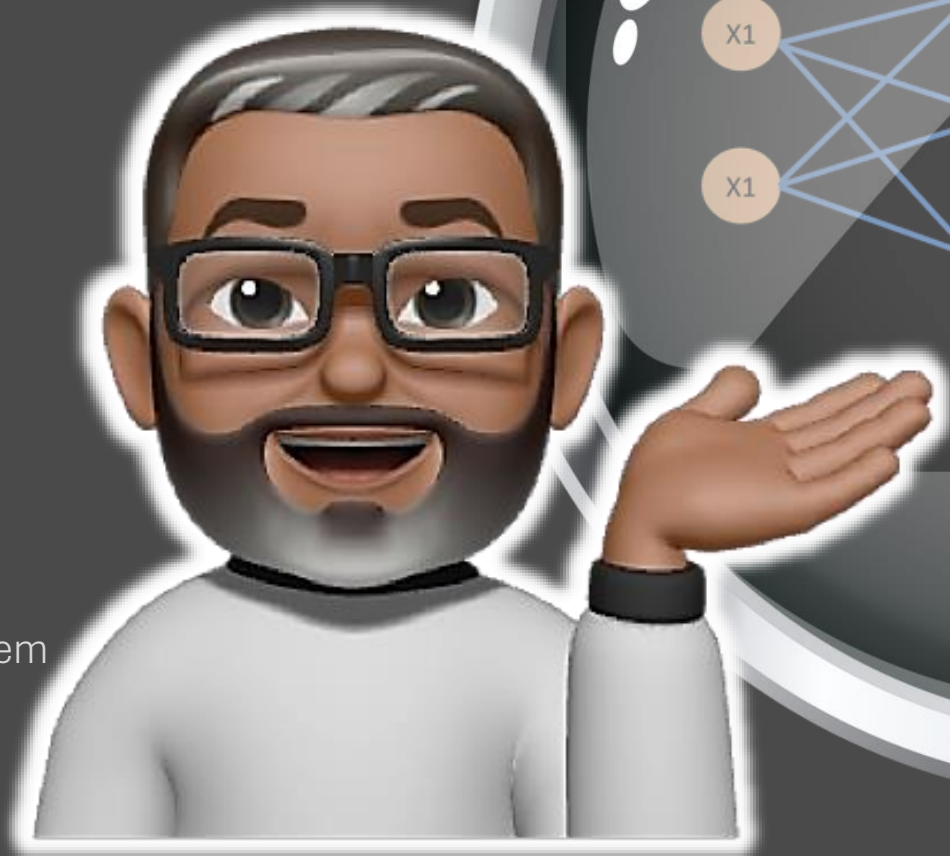
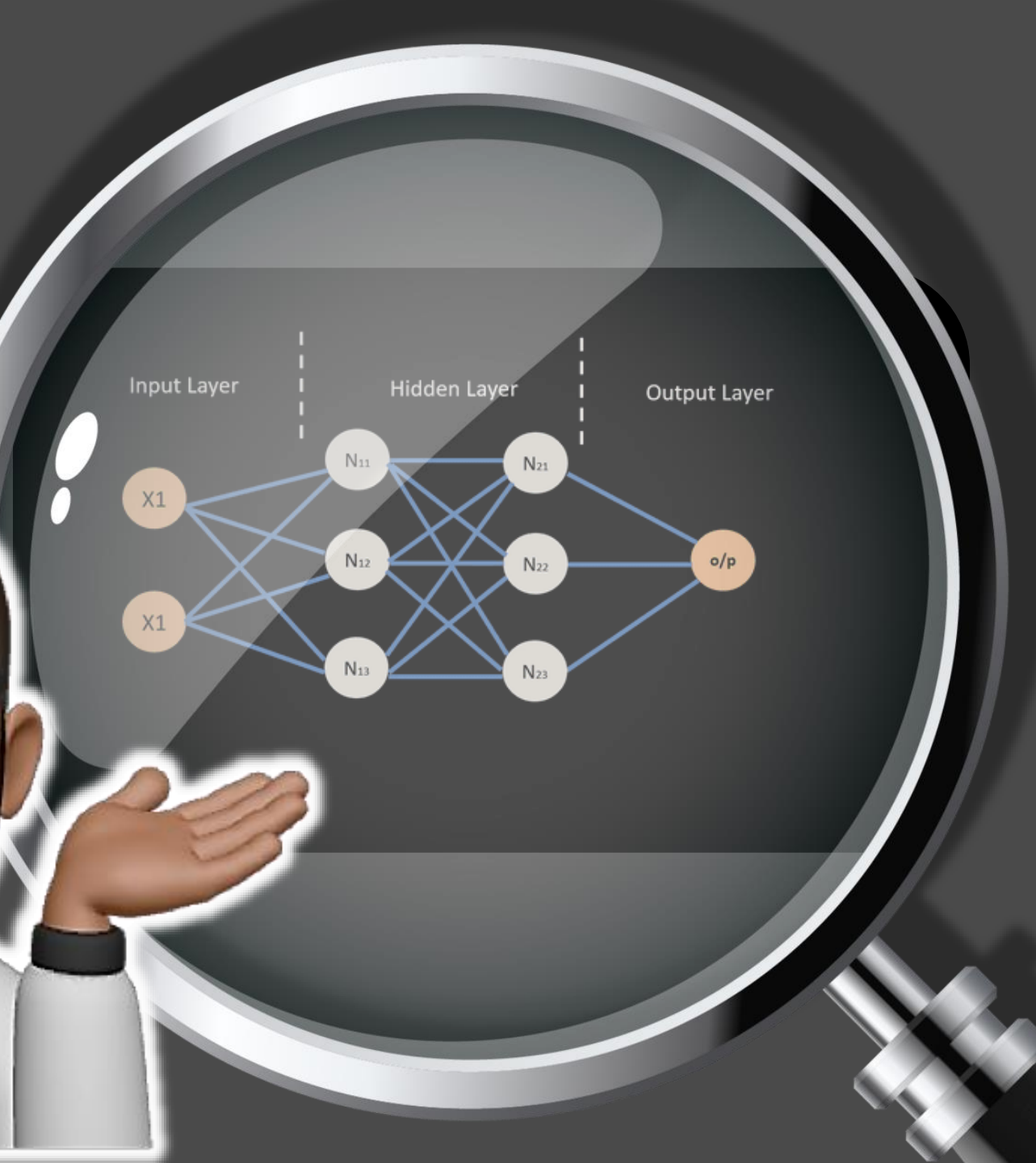
COMO UMA REDE NEURAL MLP APRENDE?

Augusto Uchôa





Tópicos

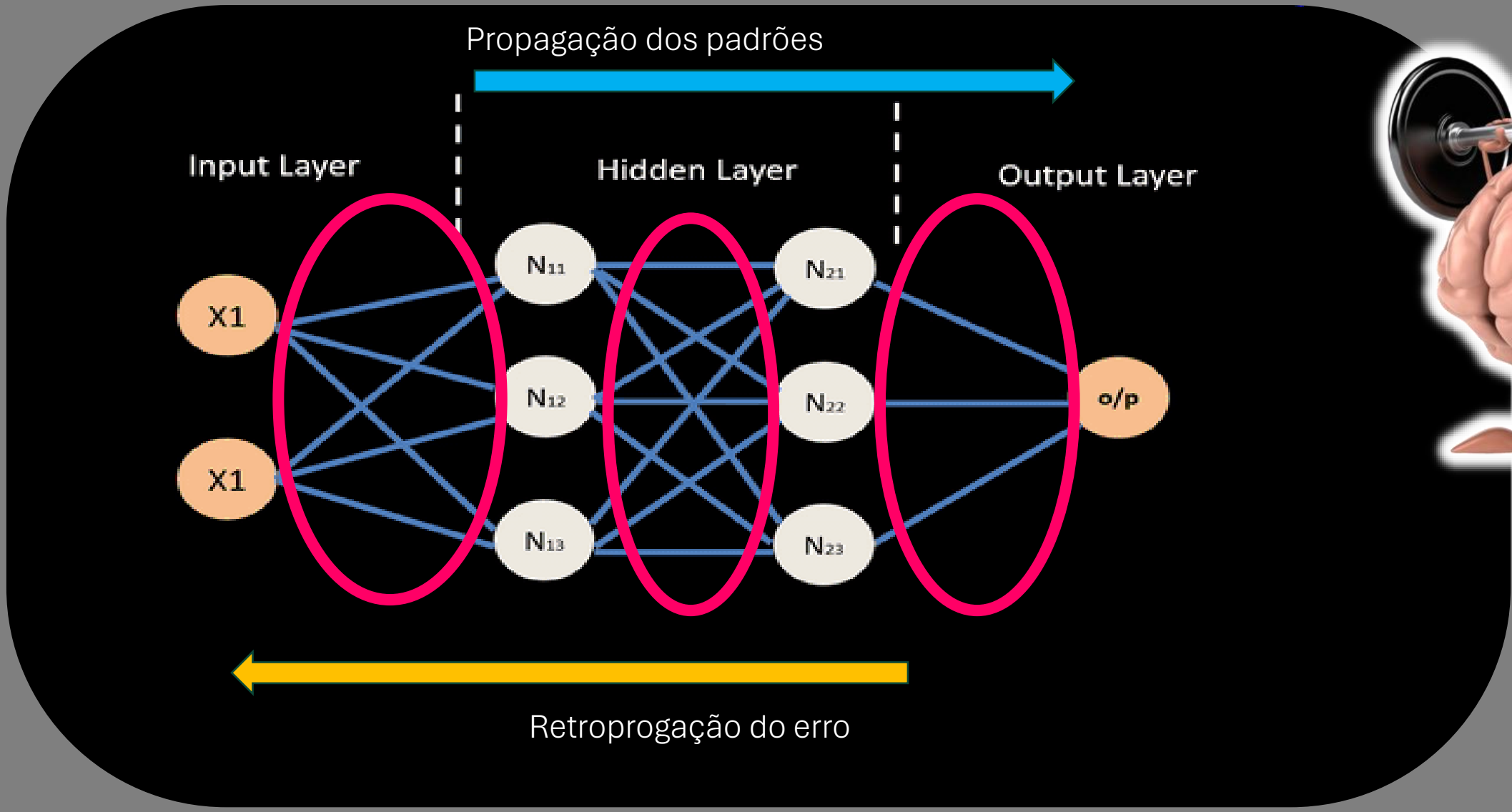
Avançados II

Aula 5

Petran- Programa de Pós-graduação em Engenharia de Transportes

TRILHA DE HOJE:

- Como uma rede MLP aprende?
- O que é o algoritmo backpropagation?
- Fases do backpropagation
- A estratégia do gradiente descendente
- A regra delta generalizada
- O que é a para que servem os Bias?



TODO O APRENDIZADO DA REDE É ARMAZENADO NOS PESOS

ALGORITMO BACKPROPAGATION PADRÃO

É um algoritmo baseado no gradiente descendente, que usa a regra da cadeia, para ajustar os pesos da rede e minimizar os erros por ela cometidos durante seu treinamento,
Segue-se os seguintes passos:

Seleciona-se o conjunto de treinamento

Apresenta-se os vetores de entrada à rede neural

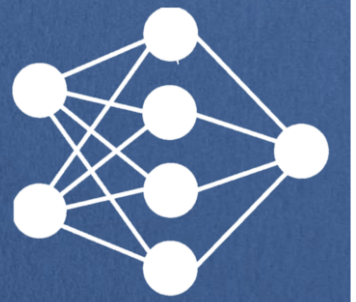
Calcula-se a saída da rede neural

Calcula-se o Erro entre a saída da rede e a saída desejada

Retropropa-se o Erro

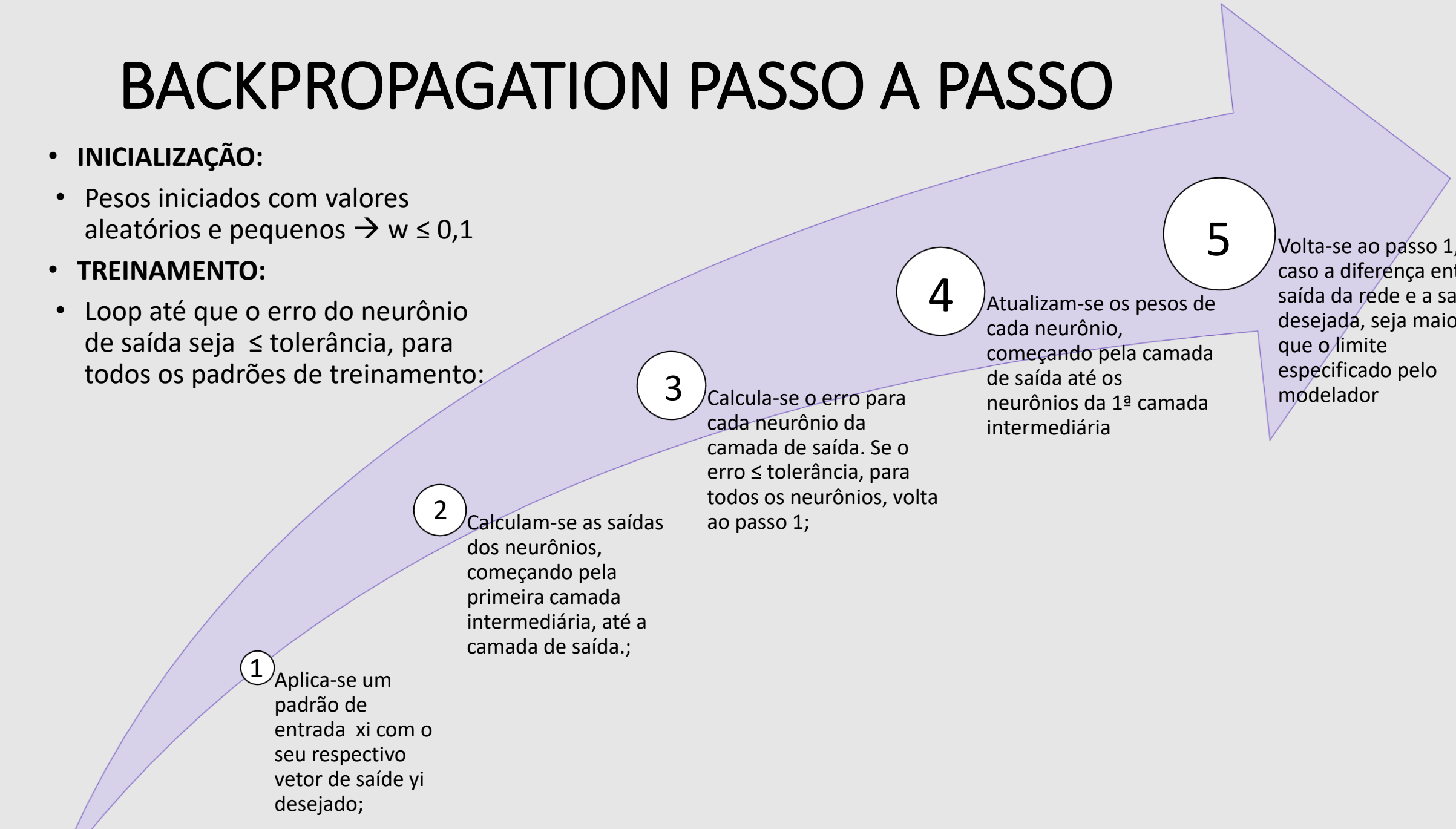
Ajustam-se os pesos da rede afim de minimizar os Erro

Repetem-se todos os passos até encontrar um erro aceitável para o conjunto de treinamento



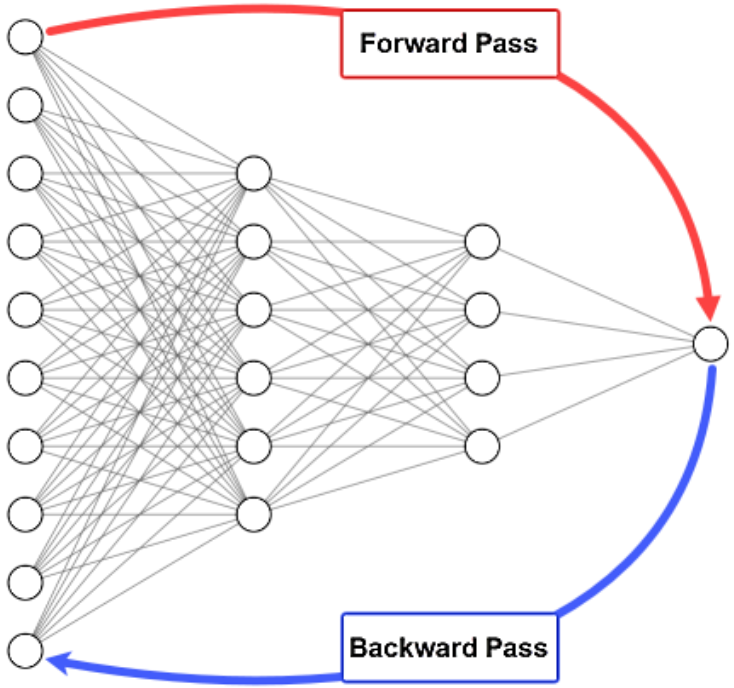
BACKPROPAGATION PASSO A PASSO

- **INICIALIZAÇÃO:**
- Pesos iniciados com valores aleatórios e pequenos → w ≤ 0,1
- **TREINAMENTO:**
- Loop até que o erro do neurônio de saída seja ≤ tolerância, para todos os padrões de treinamento:



1. Aplica-se um padrão de entrada xi com o seu respectivo vetor de saíde yi desejado;
2. Calculam-se as saídas dos neurônios, começando pela primeira camada intermediária, até a camada de saída.;
3. Calcula-se o erro para cada neurônio da camada de saída. Se o erro ≤ tolerância, para todos os neurônios, volta ao passo 1;
4. Atualizam-se os pesos de cada neurônio, começando pela camada de saída até os neurônios da 1ª camada intermediária
5. Volta-se ao passo 1, caso a diferença entre a saída da rede e a saída desejada, seja maior que o limite especificado pelo modelador

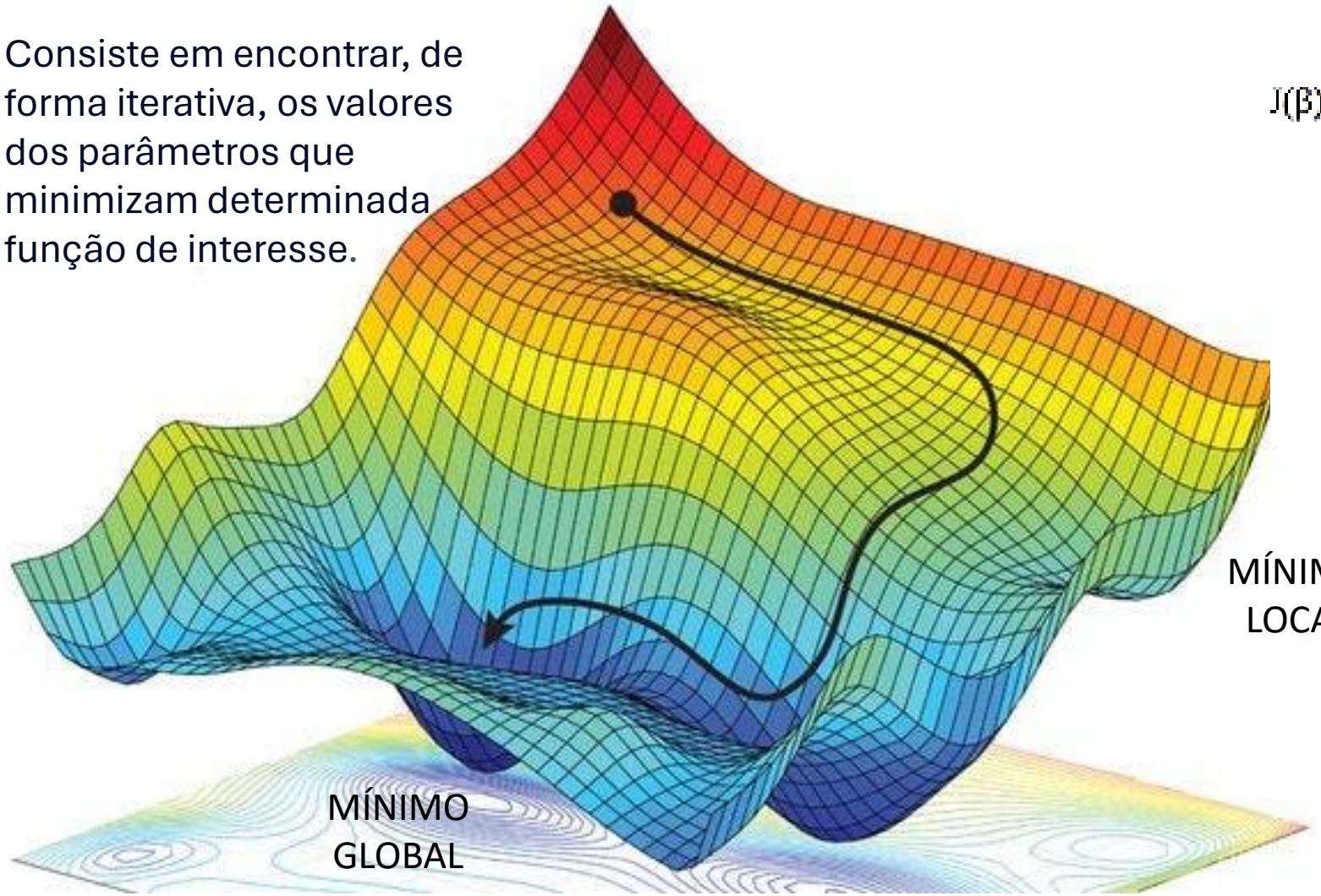
FASES DO BACKPROPAGATION



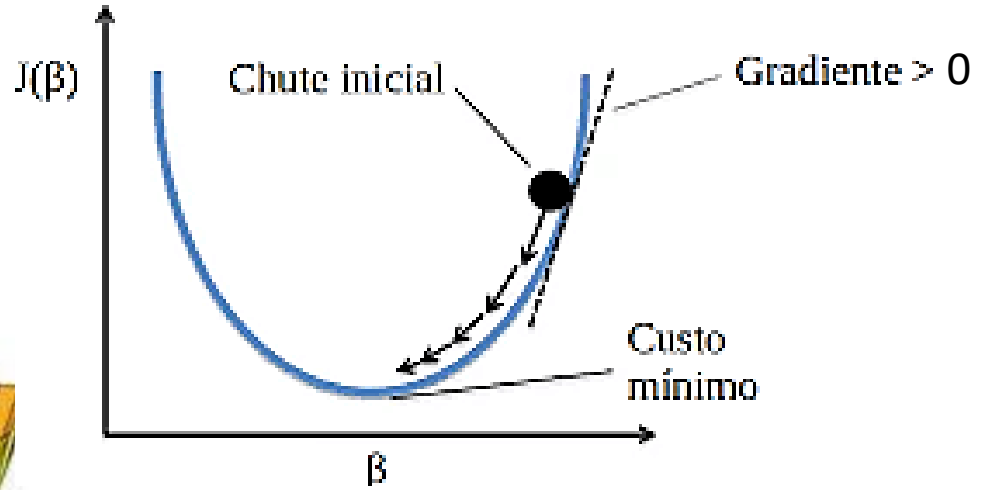
Feed-forward → as entradas se propagam pela rede, da camada de entrada até a camada de saída;

Feed-backward → os erros se propagam na direção contrária ao fluxo de dados, indo da camada de saída até a primeira camada escondida.

Consiste em encontrar, de forma iterativa, os valores dos parâmetros que minimizam determinada função de interesse.



MÍNIMO
LOCAL

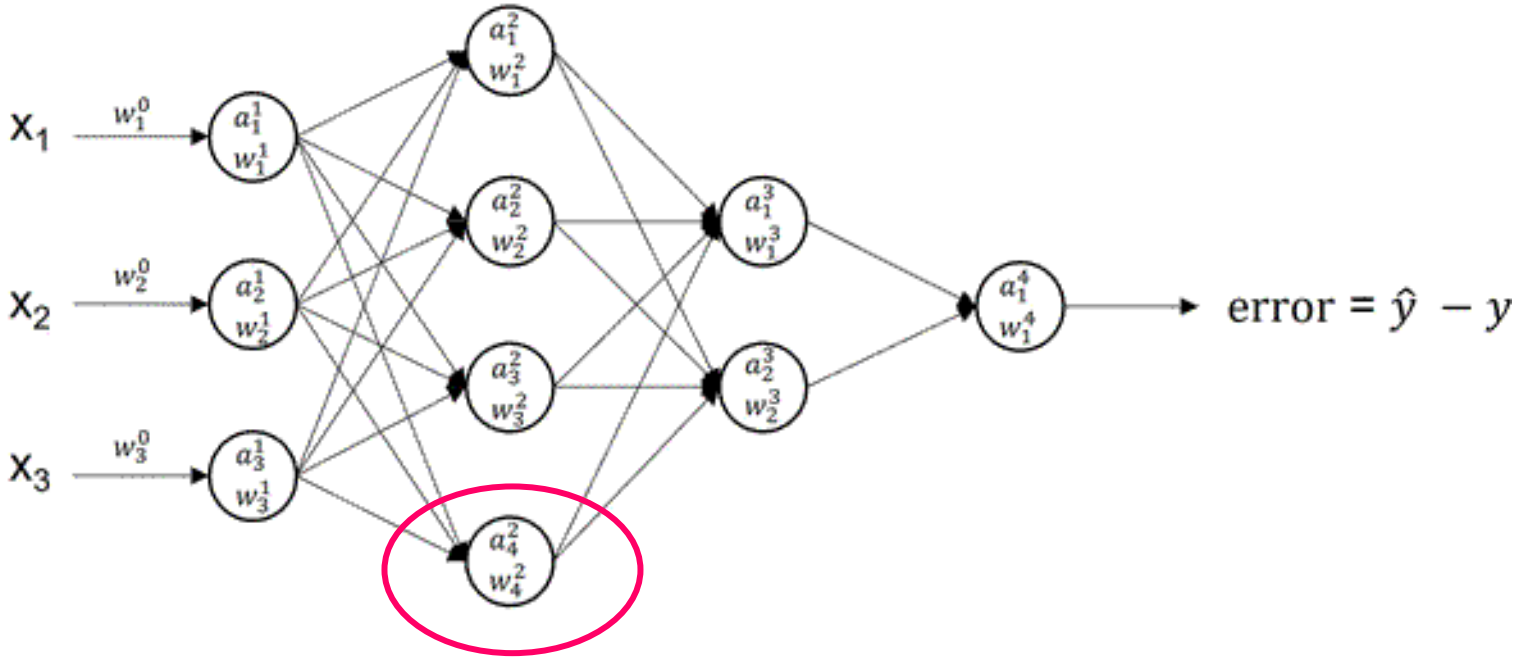


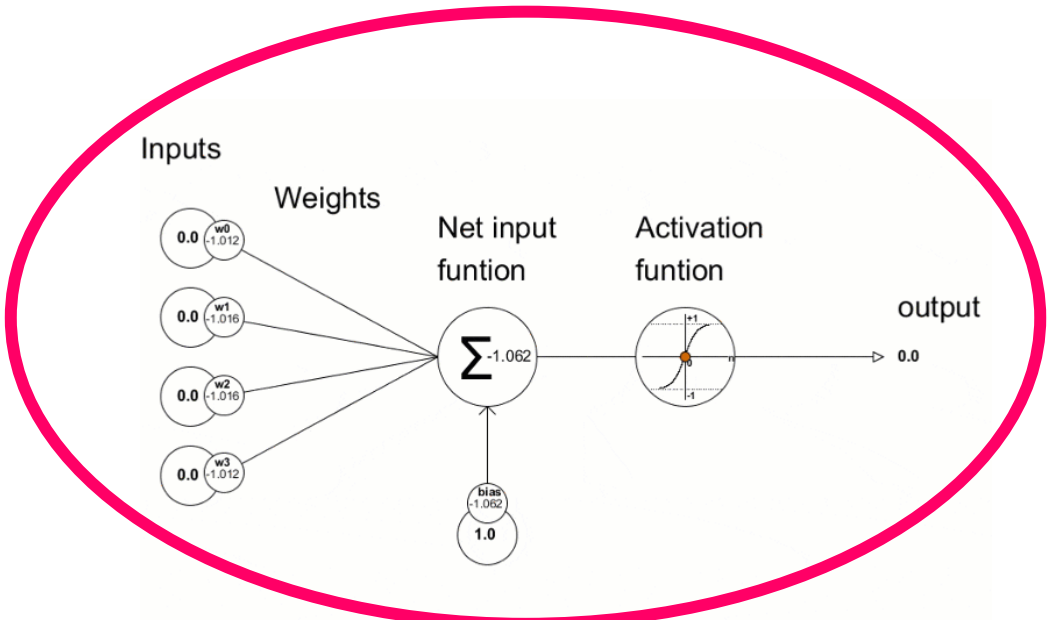
MÉTODO DO GRADIENTE DESCENDENTE

CRITÉRIOS DE PARADA DO TREINAMENTO

Indicam que a rede atingiu um nível satisfatório de desempenho, generalização e estabilidade:

- Erro médio quadrático (RMS)
- Número máximo de épocas (iterações)
- Validação Cruzada (erro do fold de validação)
- Gradiente do erro (mínimo local ou global)



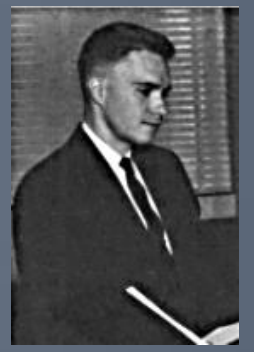


ALGORITMO LM – LEVENBERG-MARQUARDT

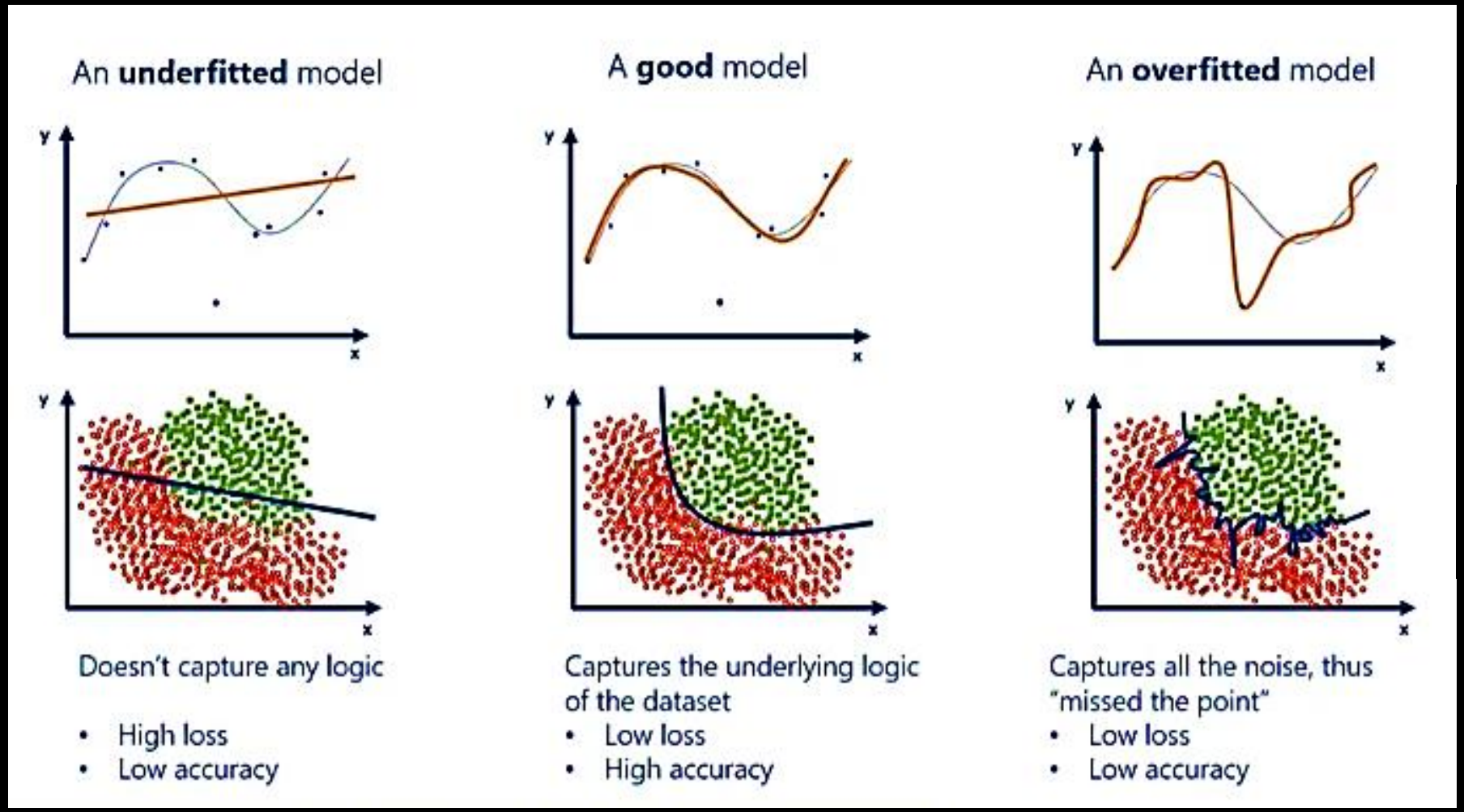
1. Inicialização:
   a) Inicialize os pesos da rede neural de maneira aleatória ou usando outro método apropriado.
   b) Defina o parâmetro de regularização (lambda) inicial.
2. Propagação direta:
   a) Aplique as entradas de treinamento à rede neural e calcule as saídas previstas (predições).
3. Cálculo do erro:
   a) Calcule a diferença entre as saídas previstas e os valores reais (erro).
4. Cálculo do gradiente:
   a) Calcule o gradiente da função de erro em relação aos pesos da rede neural. Isso envolve retropropagar o erro pela rede para calcular as derivadas parciais.
5. Atualização dos pesos:
   a) Atualize os pesos da rede neural usando o método Levenberg-Marquardt:
      i. Calcule a matriz Hessiana (segunda derivada) da função de erro.
      ii. Modifique a matriz Hessiana adicionando a diagonal com o valor do parâmetro de regularização (lambda).
      iii. Resolva o sistema de equações resultante (Hessiana modificada * delta_weights = gradiente) para obter as alterações nos pesos (delta_weights).
      iv. Atualize os pesos da rede neural usando os valores calculados em delta_weights.
6. Avaliação do critério de parada:
   a) Verifique se um critério de parada foi atingido. Isso pode ser um número máximo de iterações, um erro mínimo aceitável ou outro critério relevante.
7. Atualização do parâmetro de regularização:
   a) Dependendo da direção do erro, ajuste o parâmetro de regularização lambda. Se o erro estiver diminuindo, reduza lambda. Se o erro estiver aumentando, aumente lambda.
8. Repita os passos 2 a 7:
   a) Continue iterando até que o critério de parada seja atingido.

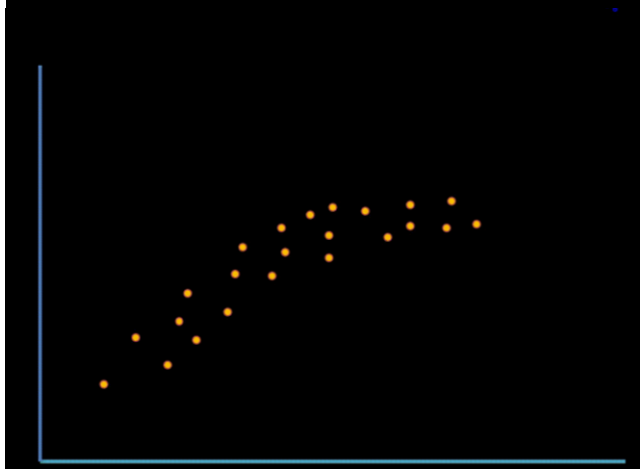
O algoritmo Levenberg-Marquardt é eficaz para treinamento de redes neurais, pois combina técnicas de otimização quase Newton(Gauss-Newton) com o método do gradiente descendente, permitindo uma convergência eficaz para encontrar os mínimos em problemas não lineares.

Kenneth Levenberg e Donald Marquardt, ambos matemáticos americanos que trabalhavam no Instituto Nacional de Padrões e Tecnologia dos EUA, publicaram em 1963 o algoritmo que é amplamente usado em otimização não linear, no treinamento de redes neurais e ajuste de curvas.



An Algorithm for Least-Squares Estimation of Nonlinear Parameters». *SIAM Journal on Applied Mathematics, 1963*

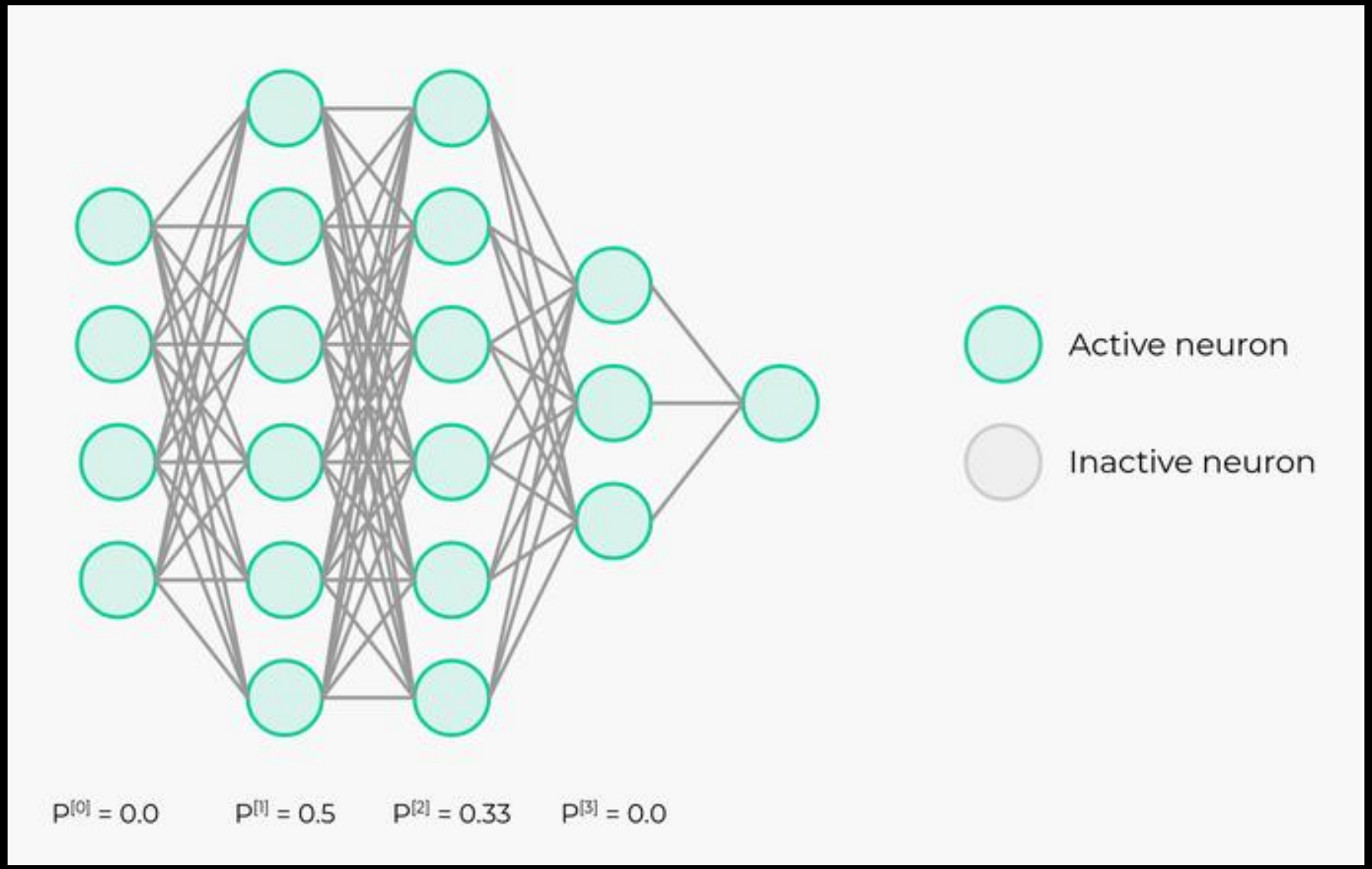




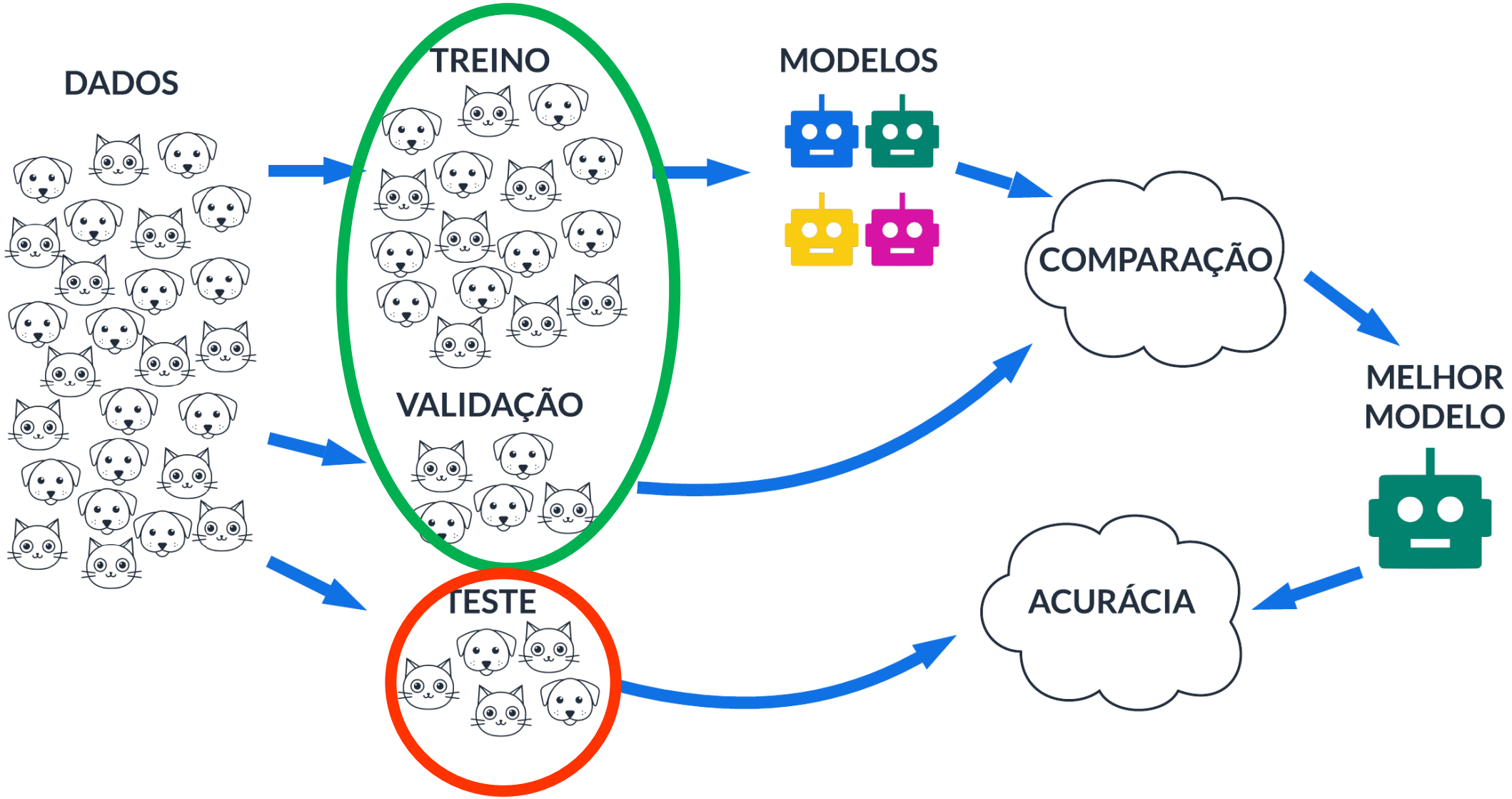
OVERFITTING & UNDERFITTING

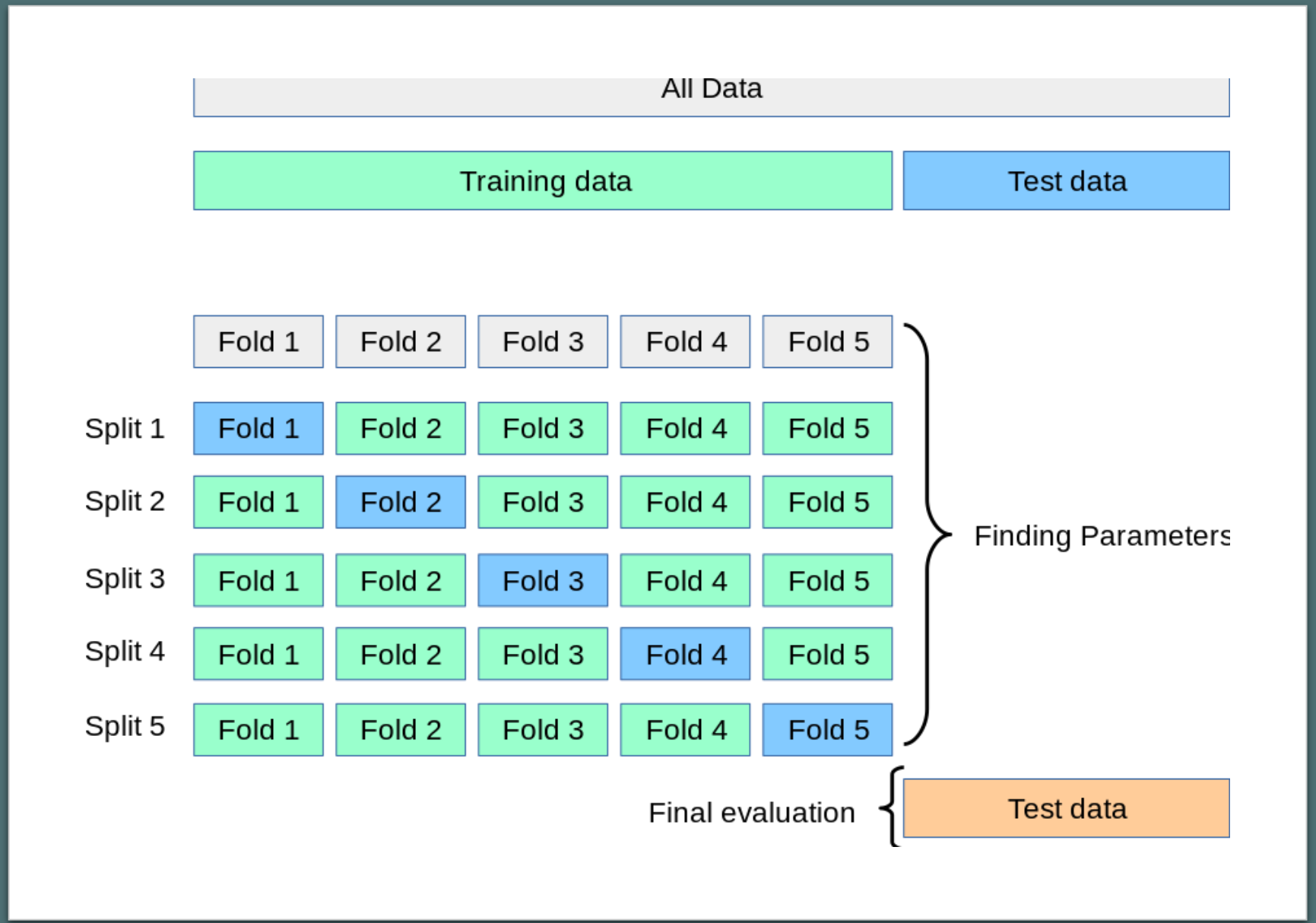
FIQUE ATENTO A ISSO!

TÉCNICA DROPOUT



O QUE PRECISA SER FEITO NO DATASET?





VALIDAÇÃO CRUZADA MÉTODO K-FOLD

DADOS ENVIESADOS & OVERFITTING

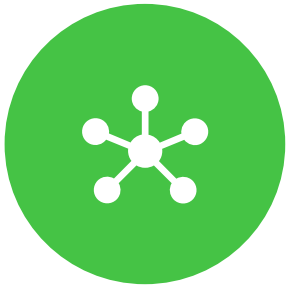
PULGA ATRÁS DA ORELHA?

1. Será que eu preciso saber derivar e integrar para trabalhar com RNAs?
2. Será que consigo evitar o Overfitting no meu modelo? Como?

3. Qual ferramenta eu uso para separar o meu dataset?

PONTOS CHAVE

O processo de aprendizagem de uma RNA



O Algoritmo Backpropagation serve para que e, como ele reduz o erro da rede?

O que é o Dropout e para que serve?

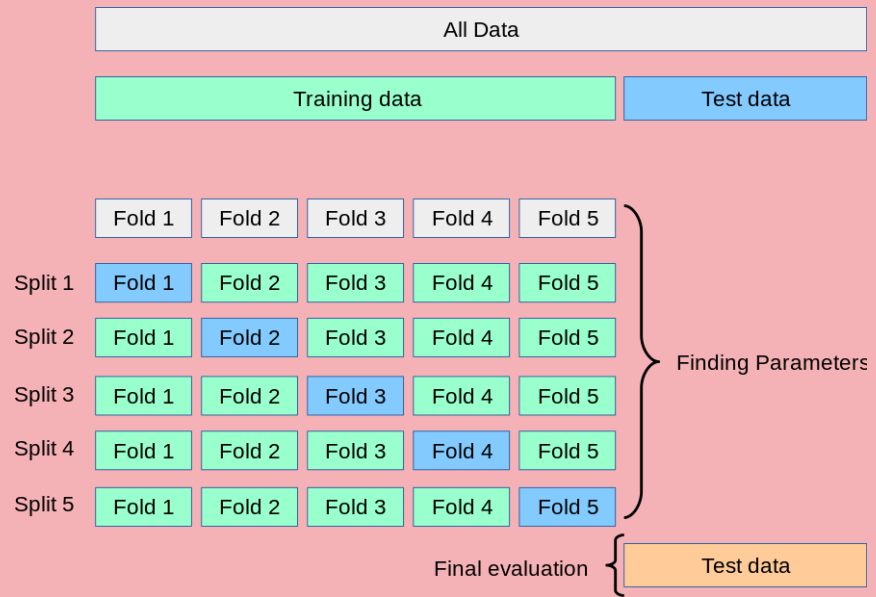
O que é e para que é usada a validação cruzada K-fold?

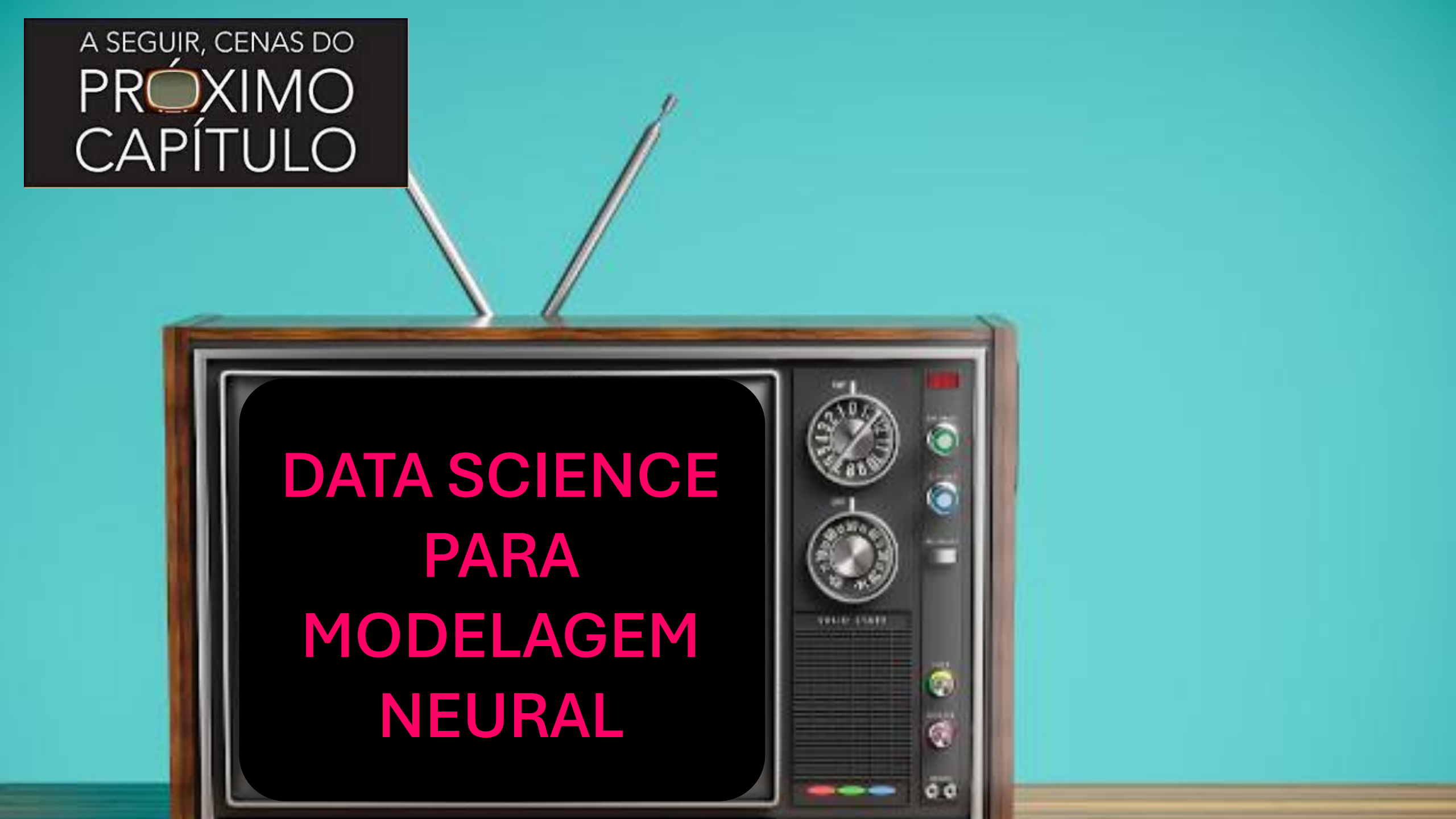
ATIVIDADE 4:

Supondo que tu tens em vista um fenômeno a ser analisado e, que exista um conjunto de dados, coletados por você (primários) ou obtidos de fontes confiáveis (secundários), em formato digital e tabular. Solicita-se que você use qualquer recurso computacional, visando:

1. Criar uma ordem aleatória no conjunto de dados (embaralhar);
2. Após a randomização dos dados, separa-los em 2 subconjuntos, mutuamente exclusivos, da seguinte forma:
   a) 70% corresponderão ao Subconjunto de treinamento e validação da do modelo. Após isso subdivida novamente esse subconjunto em 5 outros subconjunto, mutuamente exclusivos e de mesmo tamanho (ou seja 5 folds;
   b) 30% ao subconjunto de teste, estes dados não serão usados durante a etapa de treinamento da rede, só serão apresentados a ela, após o treinamento, justamente para avaliar a capacidade de acerto (aprendizado) da rede, ou seja, sua acurácia

   Nota: A figura abaixo exemplifica o que deve ser feito com o seu dataset




A SEGUIR, CENAS DO
PRÓXIMO
CAPÍTULO
DATA SCIENCE
PARA
MODELAGEM
NEURAL