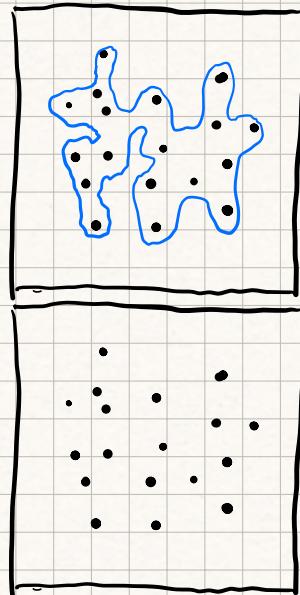


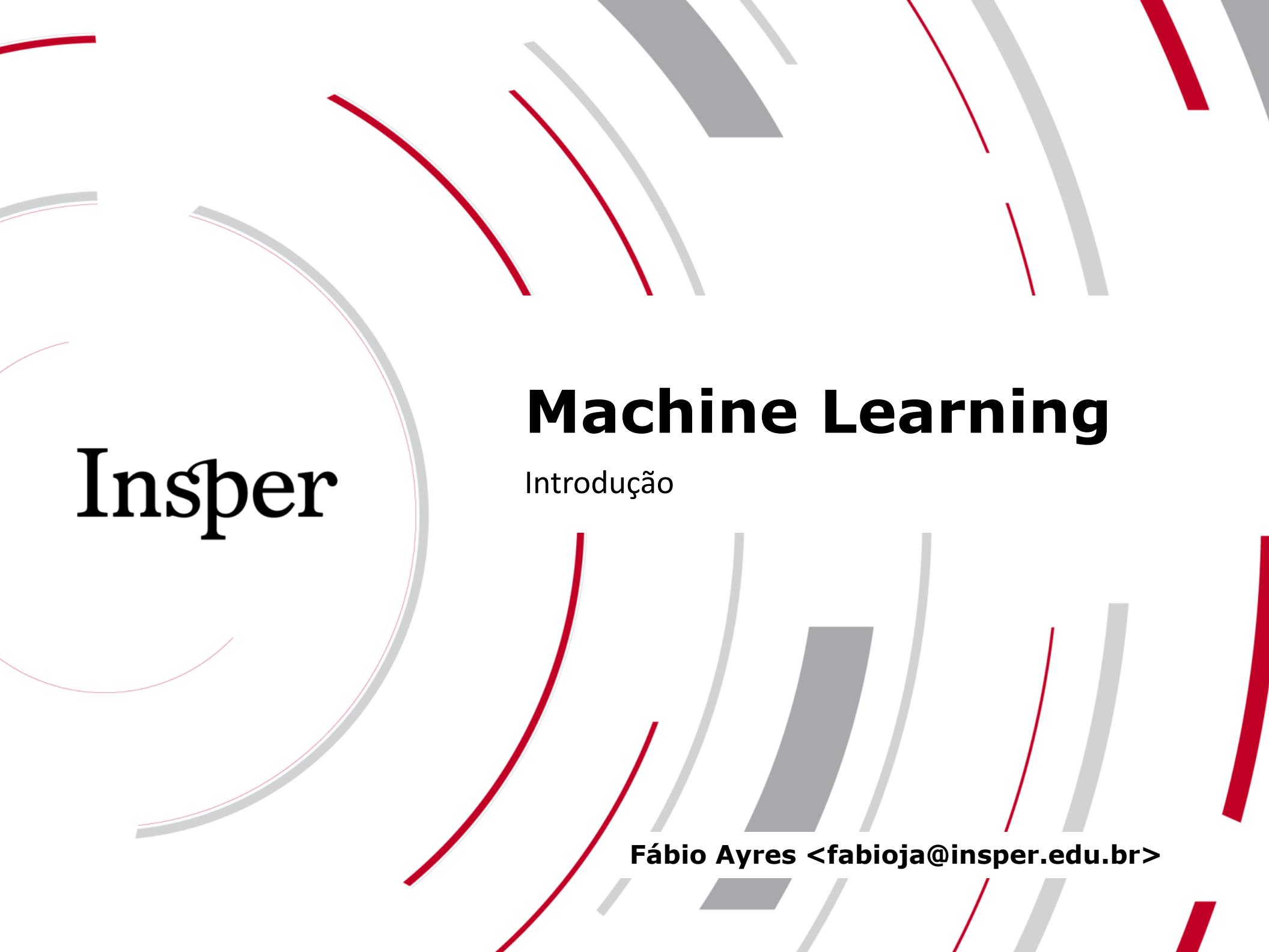
Prof. Fábio Ayres → fabioja@insper.edu.br

## O que é machine learning?

Coleta e análise de dados, a partir disso "aprender" padrões para fazer previsões

- Over Fitting
- Under Fitting



The background features abstract geometric shapes in red and grey, including curved lines and rectangles.

**Insper**

# Machine Learning

Introdução

Fábio Ayres <[fabioja@insper.edu.br](mailto:fabioja@insper.edu.br)>

# Bem vindos!

Prof. Fábio José Ayres

[fabioja@insper.edu.br](mailto:fabioja@insper.edu.br)

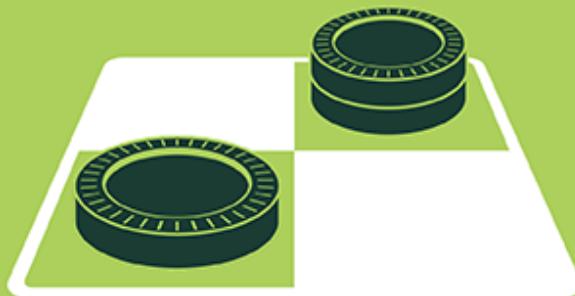
Aulas: 3<sup>a</sup> e 5<sup>a</sup>, 13:30 às 15:30

Atendimento: 5<sup>a</sup> 15:45 às 17:15

# O que é machine learning?

# ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.



## MACHINE LEARNING

Machine learning begins to flourish.



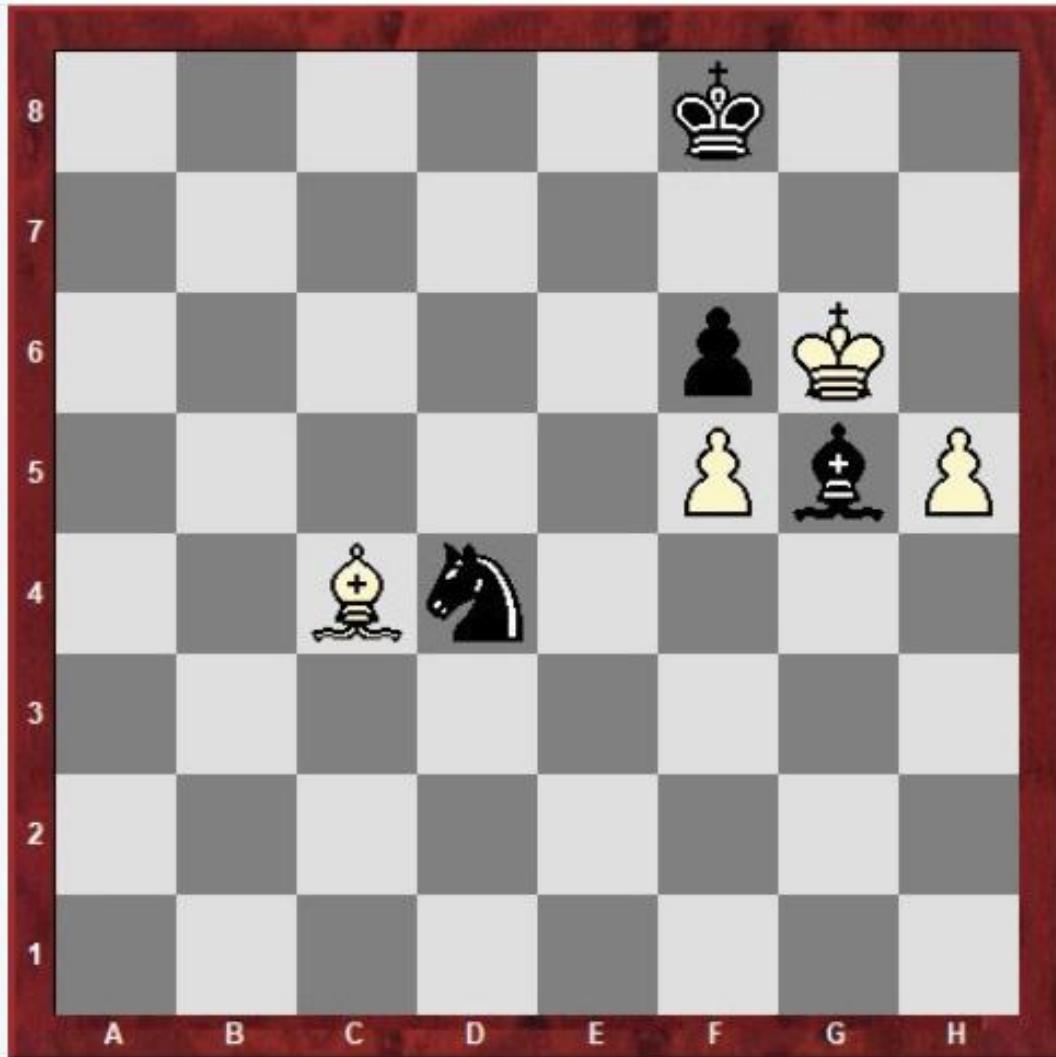
## DEEP LEARNING

Deep learning breakthroughs drive AI boom.



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.

<https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>



Carlsen – Caruana  
World Championship 2018  
Game 6

“Obviamente, mate em 63!”  
- Stockfish

<https://www.chess.com/news/view/world-chess-championship-game-6-caruana-misses-nearly-impossible-win>



Garry Kasparov   
@Kasparov63

The computer shows Black wins with 68..Bh4 here. But had Caruana played the incredible 69.Bd5 Ne2 70.Bf3 Ng1!! they would request metal detectors immediately! No human can willingly trap his own knight like that.

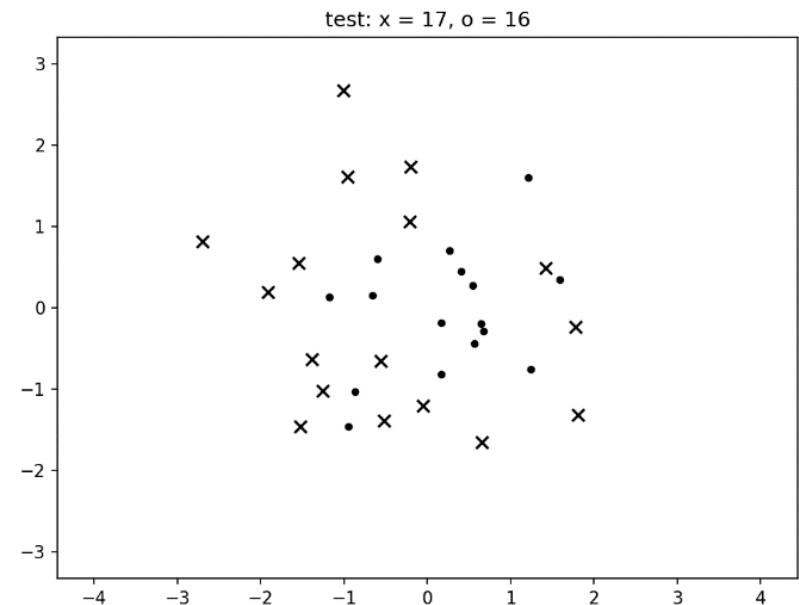
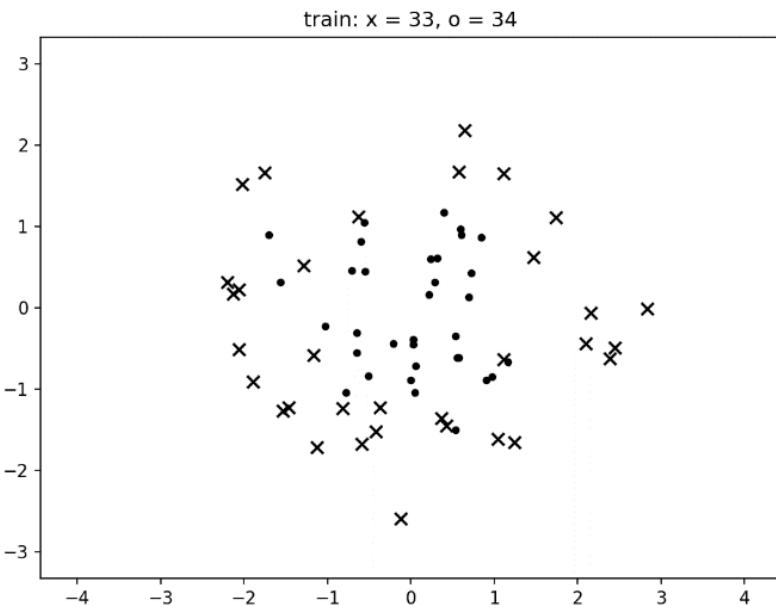
# Mastering the game of Go with deep neural networks and tree search

David Silver<sup>1\*</sup>, Aja Huang<sup>1\*</sup>, Chris J. Maddison<sup>1</sup>, Arthur Guez<sup>1</sup>, Laurent Sifre<sup>1</sup>, George van den Driessche<sup>1</sup>, Julian Schrittwieser<sup>1</sup>, Ioannis Antonoglou<sup>1</sup>, Veda Panneershelvam<sup>1</sup>, Marc Lanctot<sup>1</sup>, Sander Dieleman<sup>1</sup>, Dominik Grewe<sup>1</sup>, John Nham<sup>2</sup>, Nal Kalchbrenner<sup>1</sup>, Ilya Sutskever<sup>2</sup>, Timothy Lillicrap<sup>1</sup>, Madeleine Leach<sup>1</sup>, Koray Kavukcuoglu<sup>1</sup>, Thore Graepel<sup>1</sup> & Demis Hassabis<sup>1</sup>

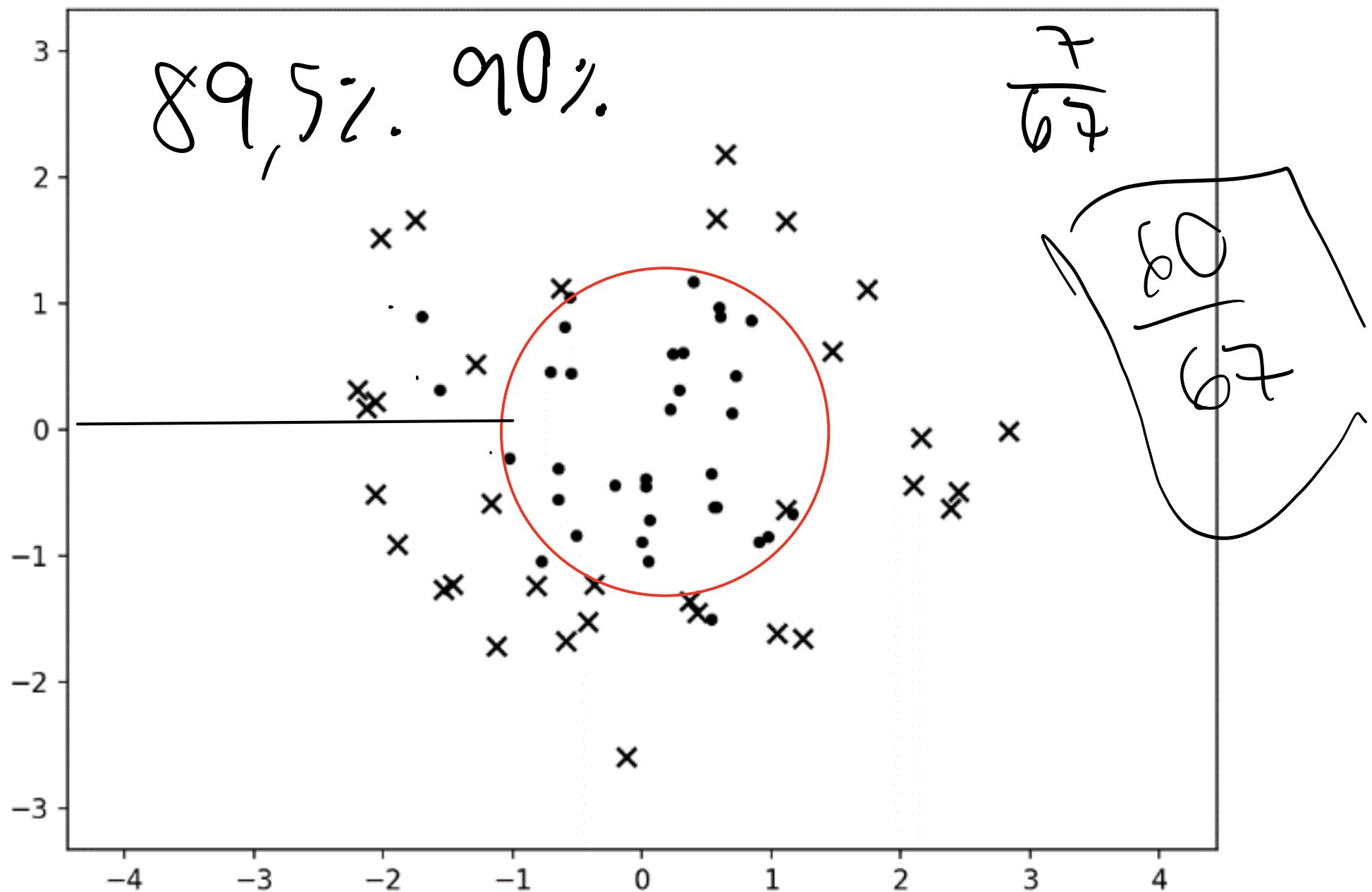
The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses ‘value networks’ to evaluate board positions and ‘policy networks’ to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.

<https://www.nature.com/articles/nature16961.pdf>

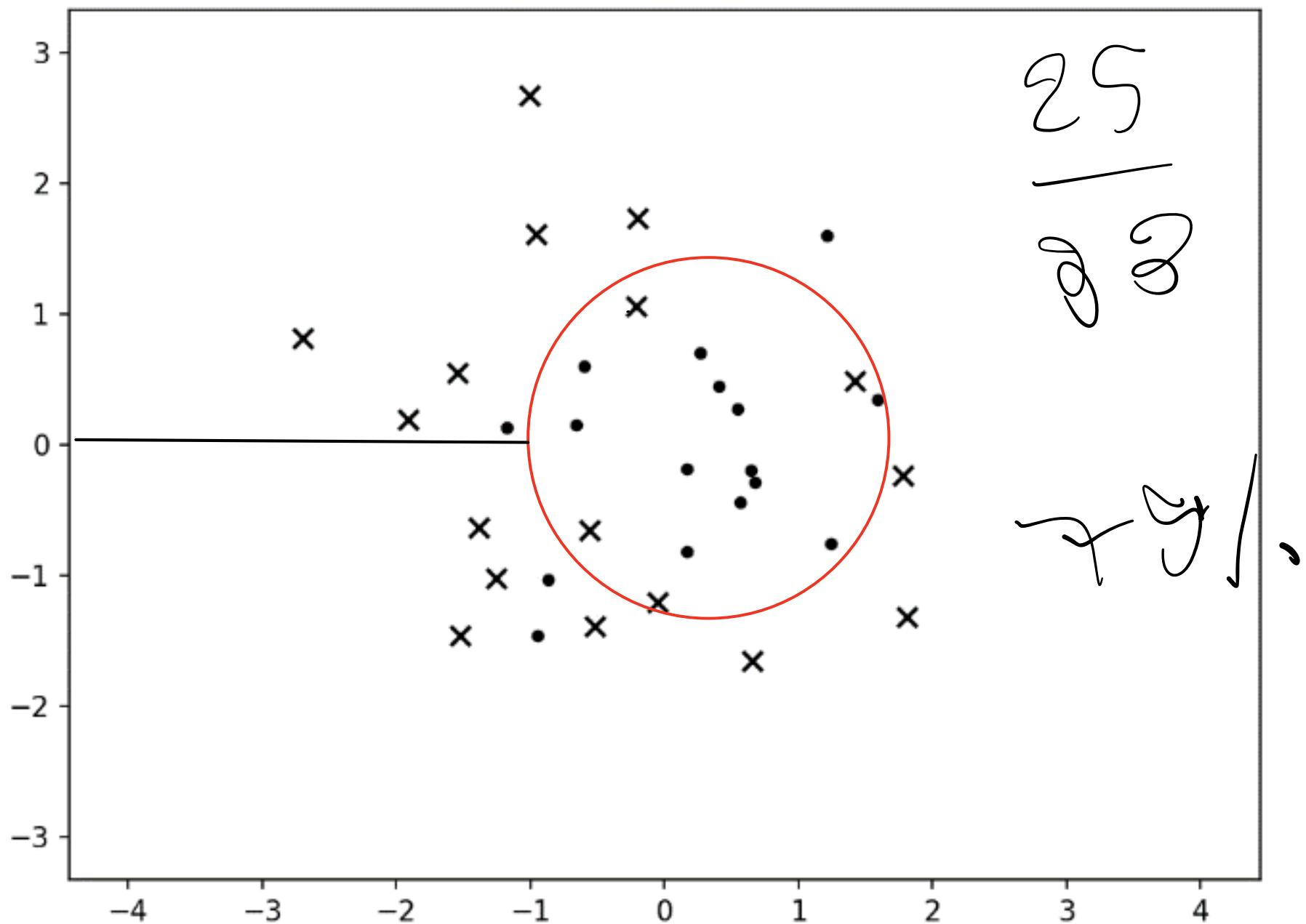
# Atividade



train:  $x = 33$ ,  $o = 34$



test:  $x = 17, o = 16$



# Para pensar

O que aconteceu com ...

- ... o modelo de ajuste perfeito aos dados de treinamento?
- ... o modelo linear?

- Decidir uma **fronteira de decisão** entre as **classes** do problema
- Este é um problema de **classificação**

- Os valores medidos são as **características** ou **atributos (features)** dos objetos
- Em estatística, estas são as **variáveis independentes** ou **variáreis preditoras** dos objetos

# Para pensar

Quais as features de uma pessoa?

# Quais as features de uma pessoa?

- Peso
- Altura
- Cor da pele
- Sexo ao nascer
  - Fuma?
  - Timbre de voz
- Caligrafia
- Impressão digital
- Padrão retinal
  - Pressão sanguínea
  - Colesterol
  - Número de rins
- Data de nascimento
- Nome
- Horas no videogame
- Gatos ou cachorros?

*Etc.!*

- Você deve **escolher** as features interessantes
- Você pode **construir novas features** a partir de outras features!

## Feature engineering

<https://playground.tensorflow.org/>

A Neural Network Playground

Tinker With a Neural Network Right Here in Your Browser.  
Don't Worry, You Can't Break It. We Promise.

Epoch 000,000      Learning rate 0.03      Activation Tanh      Regularization None      Regularization rate 0      Problem type Classification

DATA  
Which dataset do you want to use?  
  
 Ratio of training to test data: 50%  
 Noise: 0  
 Batch size: 10  
 REGENERATE

FEATURES  
Which properties do you want to feed in?  
 $X_1$ ,  $X_2$ ,  $X_1^2$ ,  $X_2^2$ ,  $X_1X_2$ ,  $\sin(X_1)$

+ - 2 HIDDEN LAYERS  
 4 neurons      2 neurons  
 Click anywhere to edit.  
 Weight is -0.48.  
 This is the output from one neuron.  
 Hover to see it larger.  
 The outputs are mixed with varying weights, shown by the thickness of the lines.

OUTPUT  
 Test loss 0.503  
 Training loss 0.509  
  
 Colors shows

# Para pensar

O modelo linear não separa bem as classes na atividade inicial. Mas com novos atributos poderíamos bem usar o modelo linear. Quais seriam estes atributos?

- Na atividade inicial, a **categoria** ou **classe** de cada objeto é conhecida. Esta item de dados é chamado de **rótulo (label)** do objeto.
- Em estatística, esta variável é chamada de **variável dependente** ou **variável resposta**
  - Mais especificamente, trata-se de uma **variável resposta categórica**

## Aprendizado supervisionado

Exemplos incluem a variável dependente

## Aprendizado não-supervisionado

Exemplos não incluem uma variável dependente

Quando a variável dependente é...

- Categórica: problema de **classificação**
- Contínua: problema de **regressão**

# Para pensar

Qual o tipo do problema?

- Baseado nas notas de um aluno em Design de Software, Ciência dos Dados e Megadados, prever sua nota em Machine Learning
- Baseado nas notas e presença de um aluno no primeiro semestre, prever se ele vai seguir no Insper
- Baseado nas notas e participação em entidades, descobrir uma forma de separar os alunos em grupos de interesse comum

## Na atividade inicial:

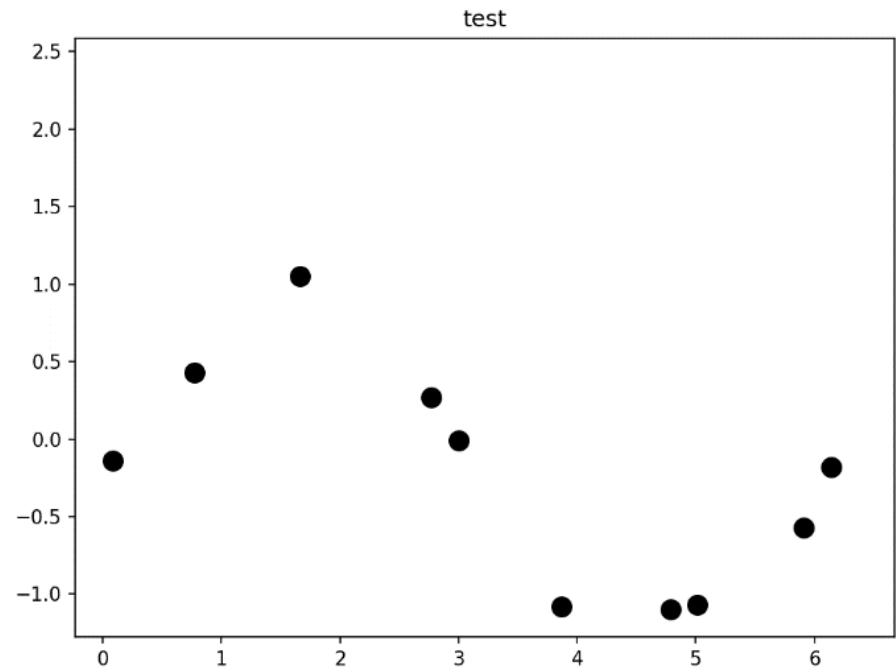
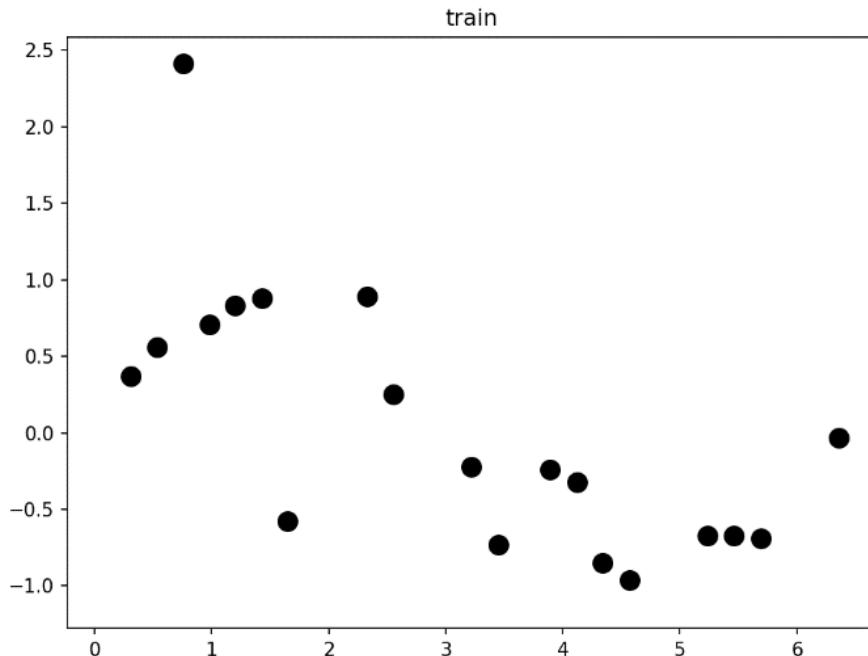
- As regras de construção da fronteira de decisão são o nosso **modelo**
- Achar uma boa fronteira a partir de exemplos é **treinar o modelo**
- Os dados de exemplo são o **conjunto de treinamento**

- Devemos testar o desempenho do modelo treinado em um conjunto independente de dados, chamado de **conjunto de teste**
- Bom desempenho no treinamento não significa bom desempenho no teste!
- Se o desempenho no teste for ruim, volte para o começo (escolha do modelo)!

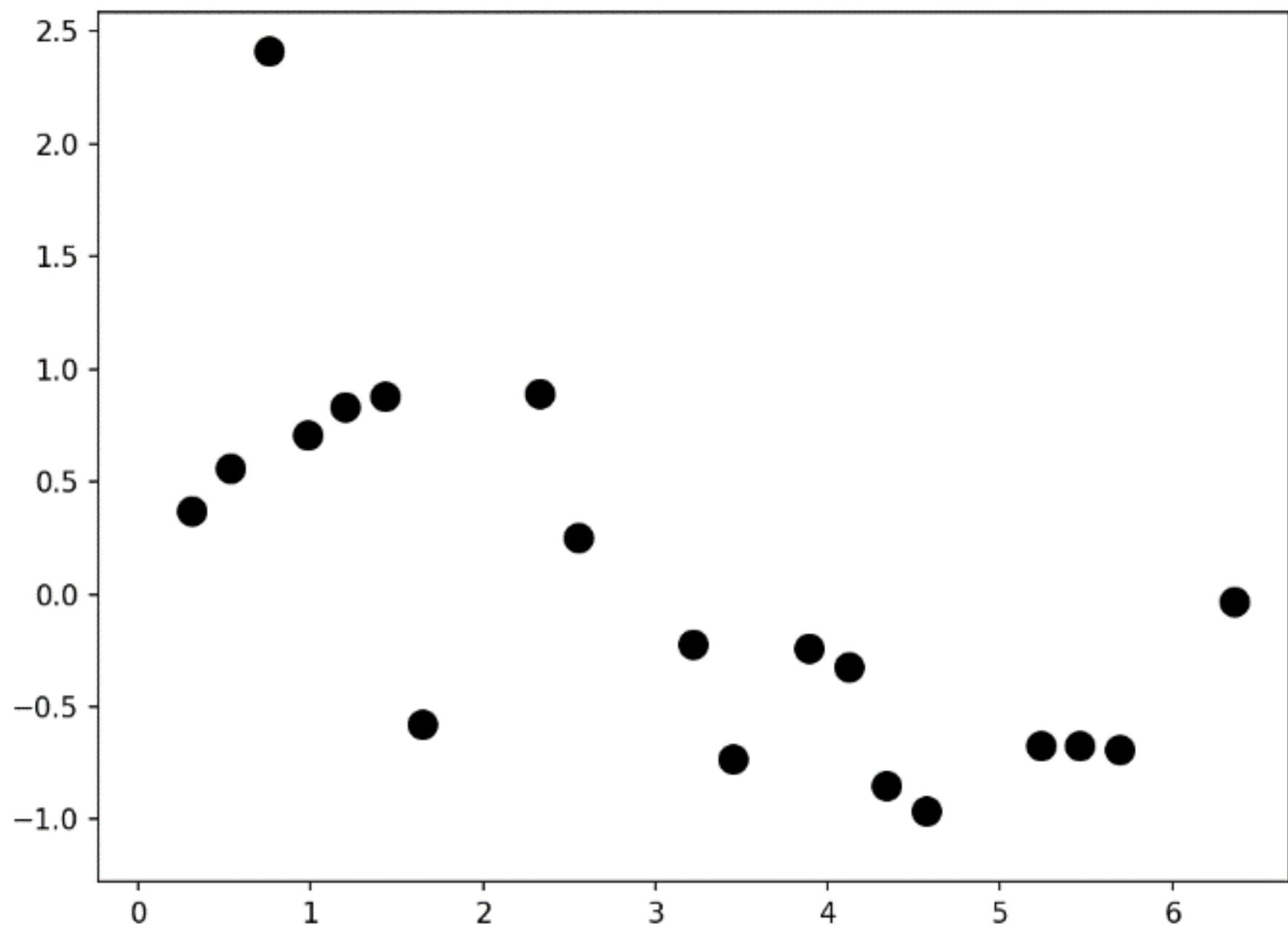
# Atividade

- Construir uma função interpoladora
- Este é um problema de regressão

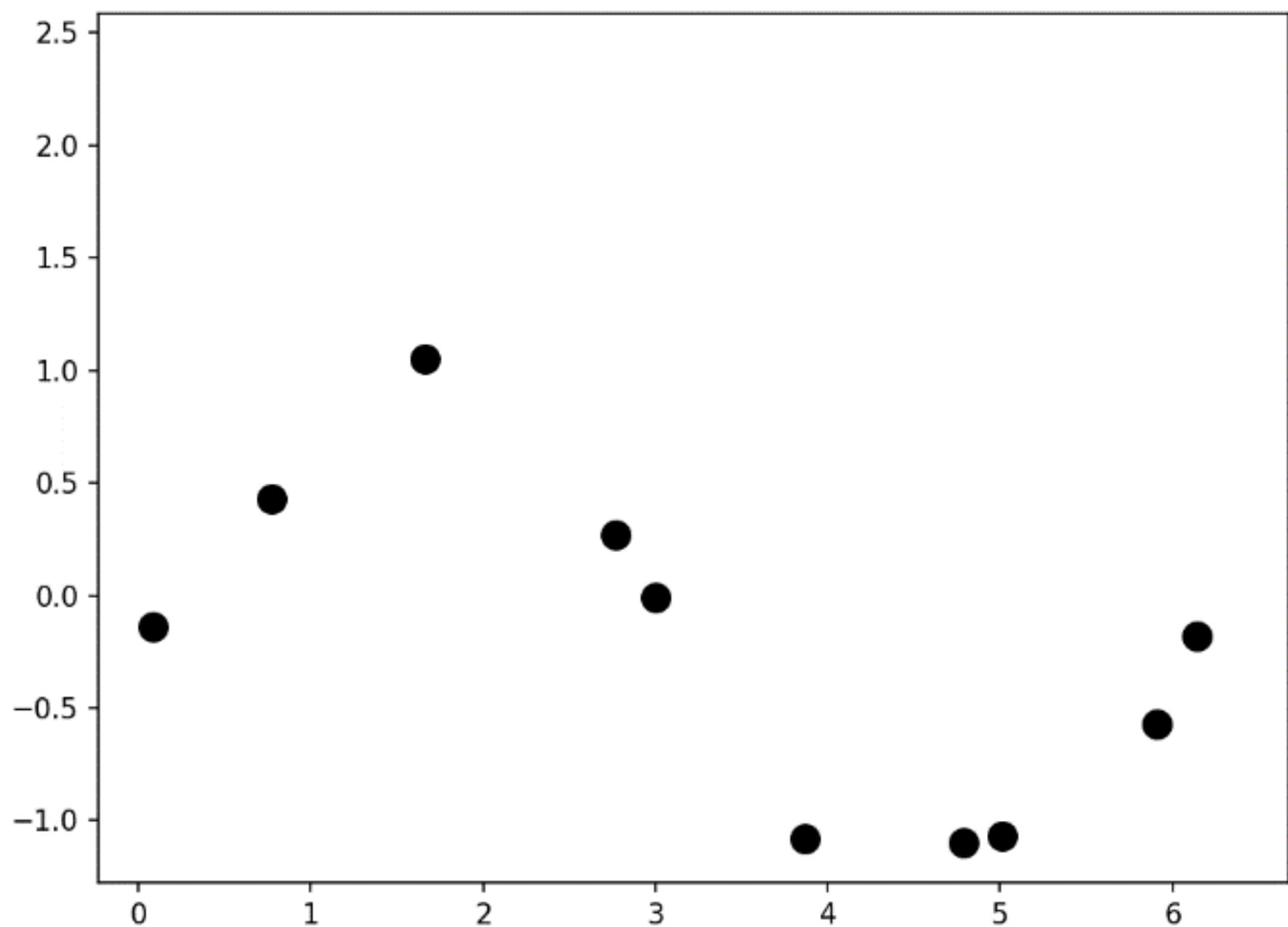
# Atividade



train



test



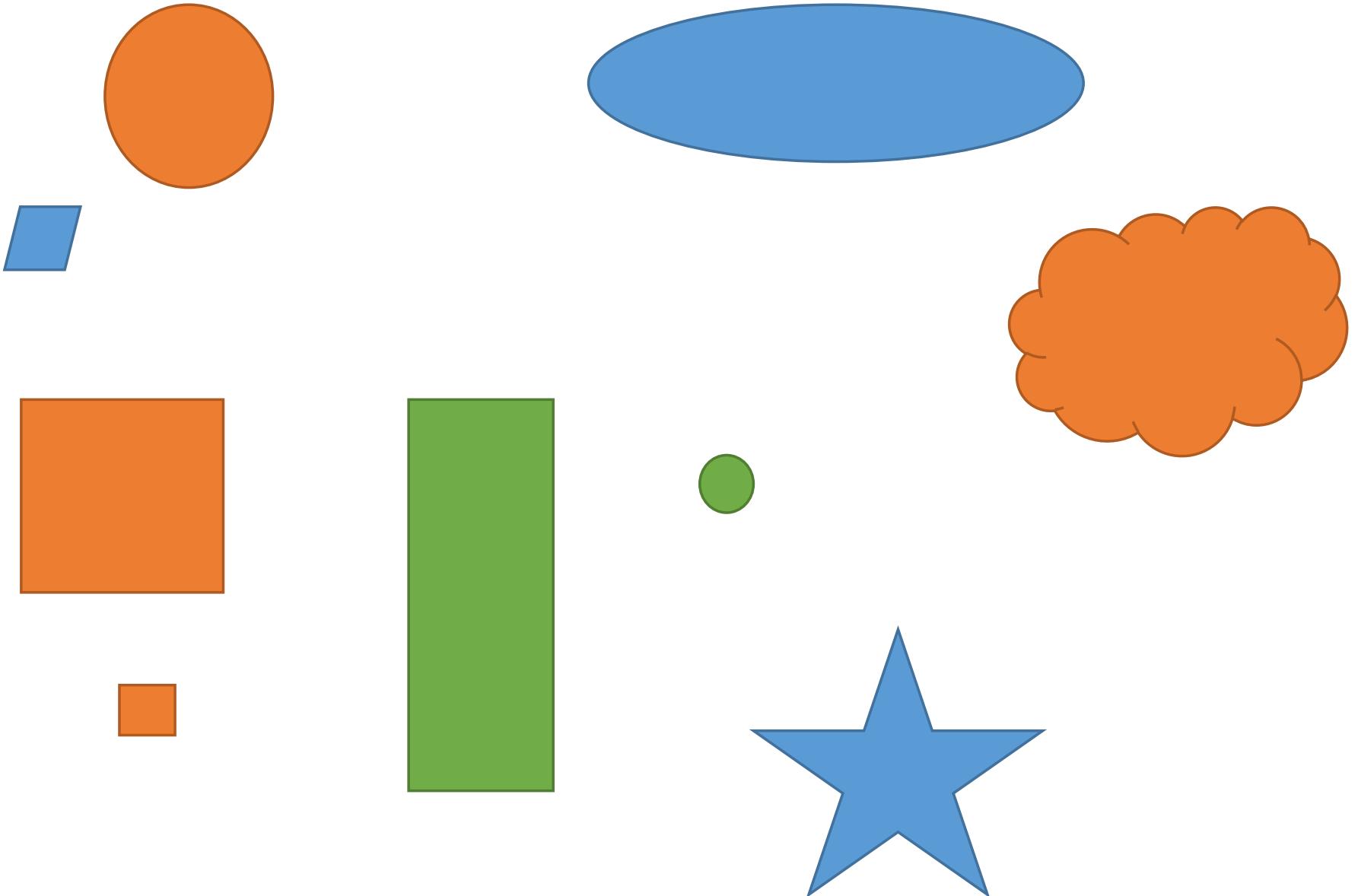
# Para pensar

Novamente: o que aconteceu com ...

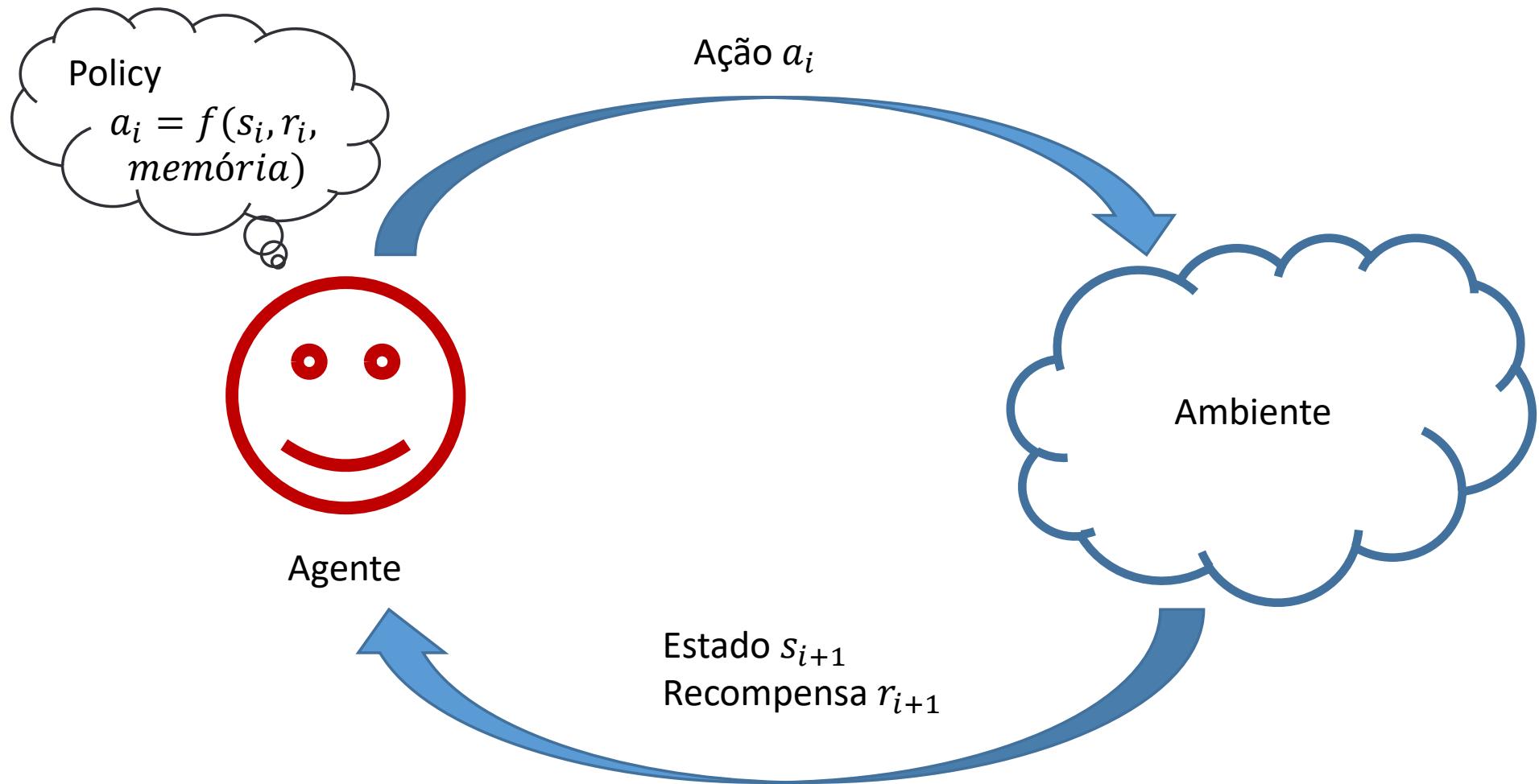
- ... o modelo de ajuste perfeito aos dados de treinamento?
- ... o modelo linear?

# Atividade

- Descubra como separar “naturalmente” os objetos dados em grupos
- Este é um problema de **agrupamento** (**clustering**)



# Reinforcement Learning



# Self-supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation



Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, Sergey Levine



<https://www.youtube.com/watch?v=vgiW0HIQWVE>

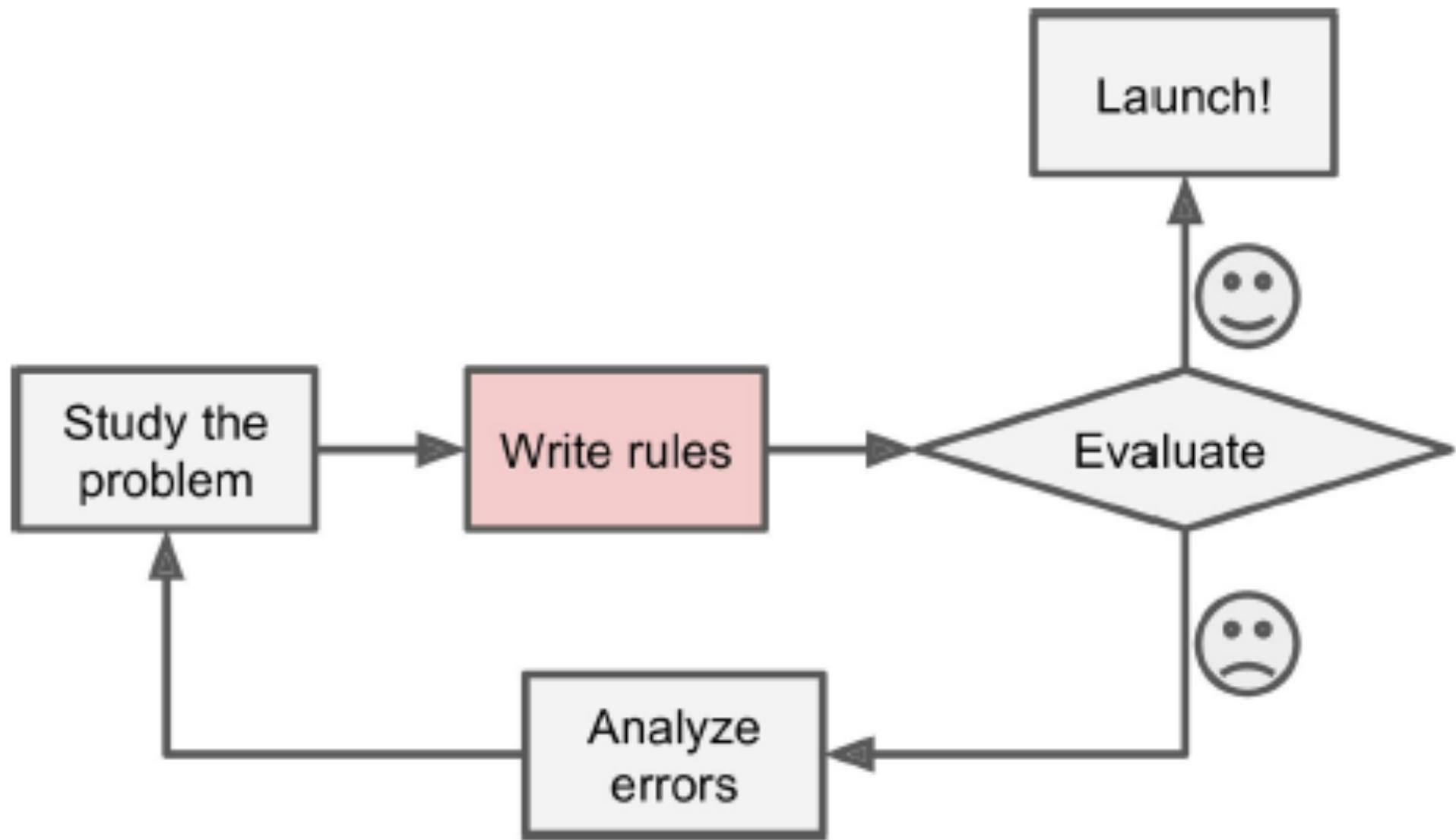


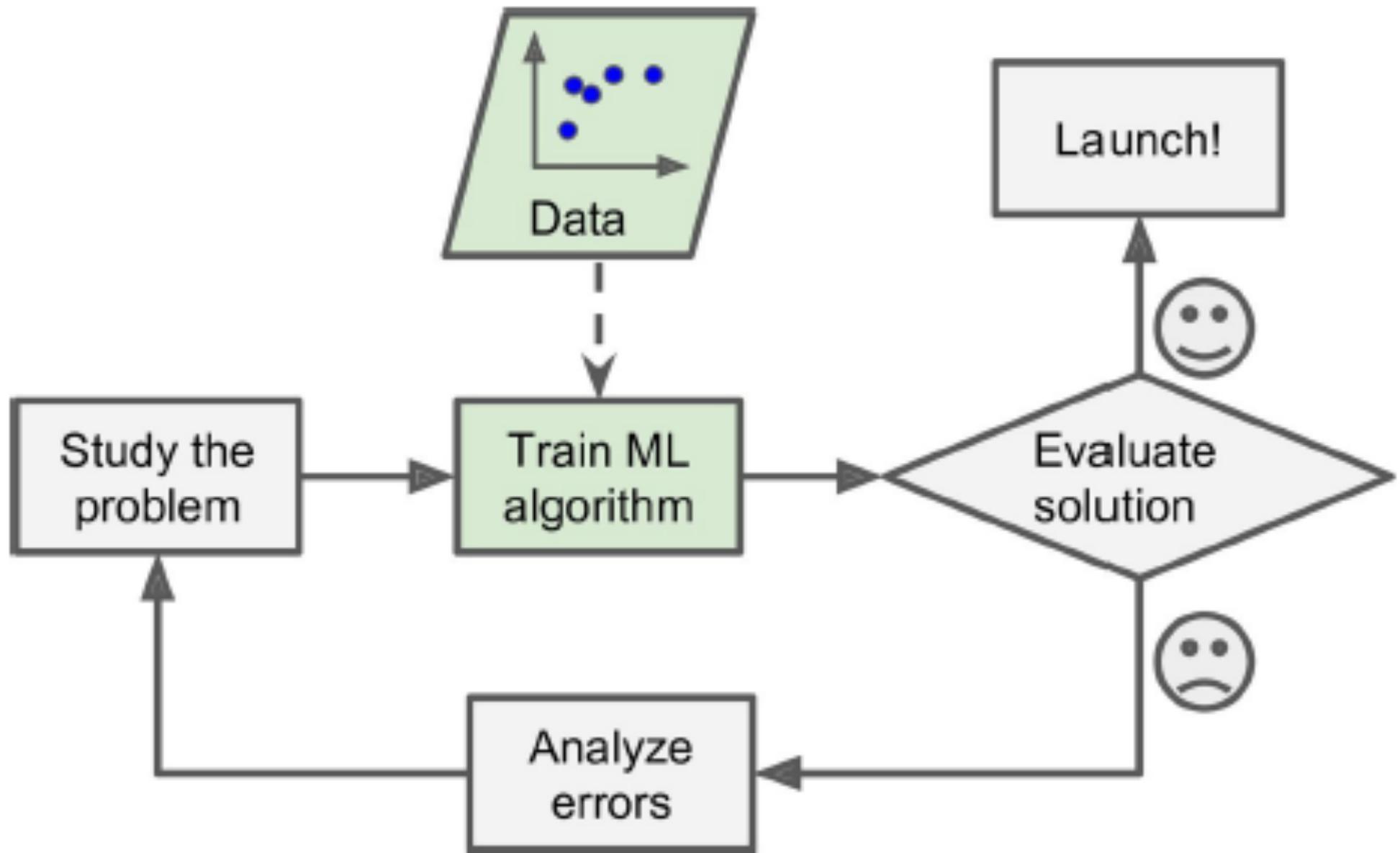
Deep Q network learning to play Seaquest

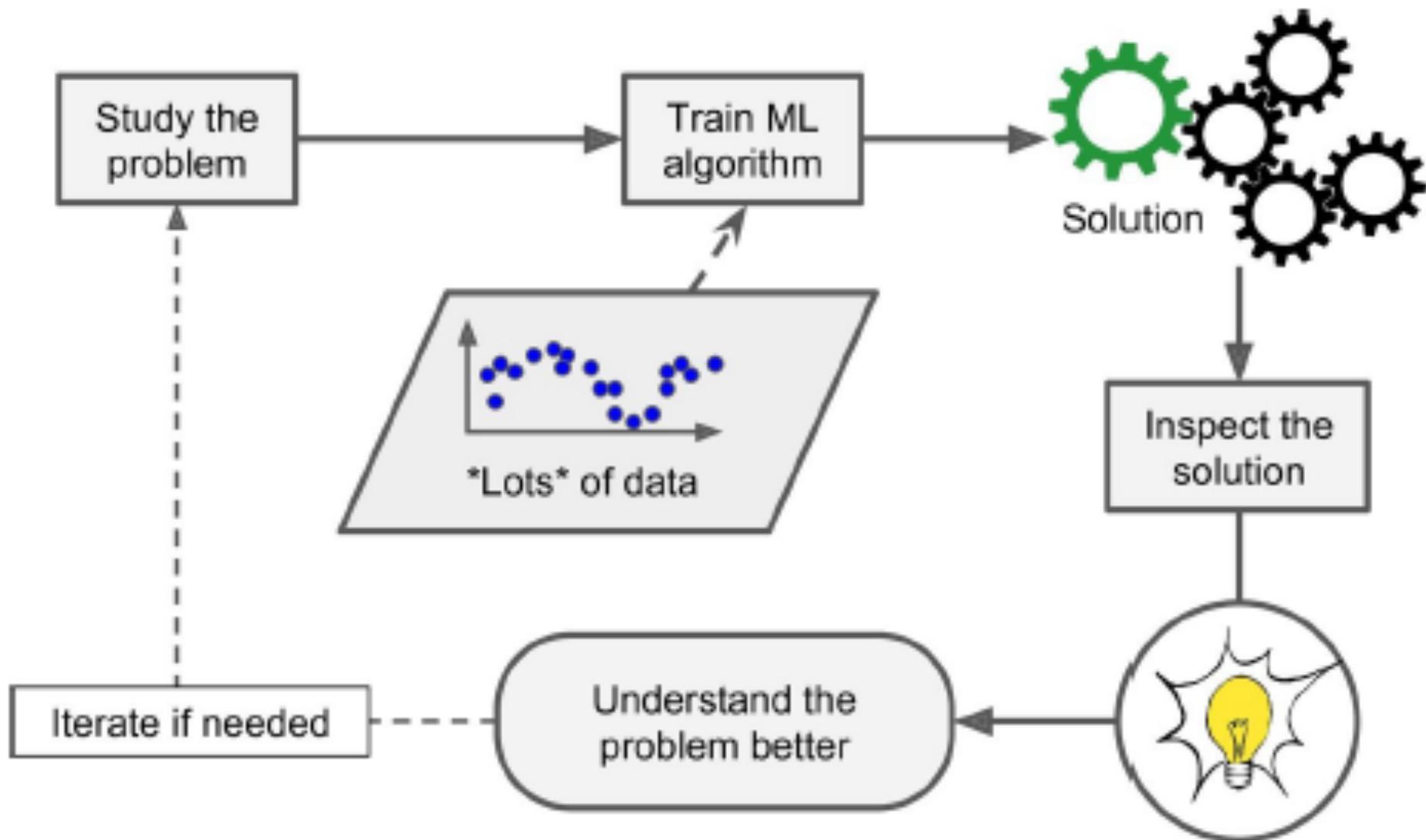
14,622 views

14 21 0 SHARE ...

<https://www.youtube.com/watch?v=5WXVJ1A0k6Q>







# Resumo

O que é machine learning?

- Estudo de técnicas que permitem ao computador realizar tarefas sem a definição explícita de regras, mas sim dependendo de modelos estatísticos e inferência.

# Resumo

## Modelos

- Representações úteis dos fenômenos sendo investigados

Atributos (features) e rótulos (labels) / valores dependentes

- Quantidades de interesse nos modelos. Atributos e rótulos são valores mensuráveis, e os rótulos ou valores dependentes dependem – através do modelo – dos atributos.

# Resumo

## Parâmetros

- Definem o formato exato do modelo.

## Treinamento

- Ajustar os parâmetros para que o modelo reflita a realidade o melhor possível

## Teste

- Estimar a veracidade do modelo e sua capacidade de generalização

# Resumo

## Aprendizado supervisionado

- Construir um modelo usando um conjunto de dados com rótulos (ou valor dependente) explícitos.

## Aprendizado não supervisionado

- Construir um modelo quando não se conhecem os rótulos ou valores dependentes.

# Resumo

## Classificação

- Aprendizado supervisionado com rótulos categóricos

## Regressão

- Aprendizado supervisionado com variável dependente contínua ou ordinal

## Clustering

- Aprendizado não-supervisionado que visa identificar a existência de grupos homogêneos de objetos no conjunto de dados

# Resumo

## Aprendizado por reforço

- Máquina aprende uma política de decisão através da interação com o meio ambiente

# Para a próxima aula

## Instalar softwares básicos

- Instalar Anaconda

<https://www.anaconda.com/products/individual>

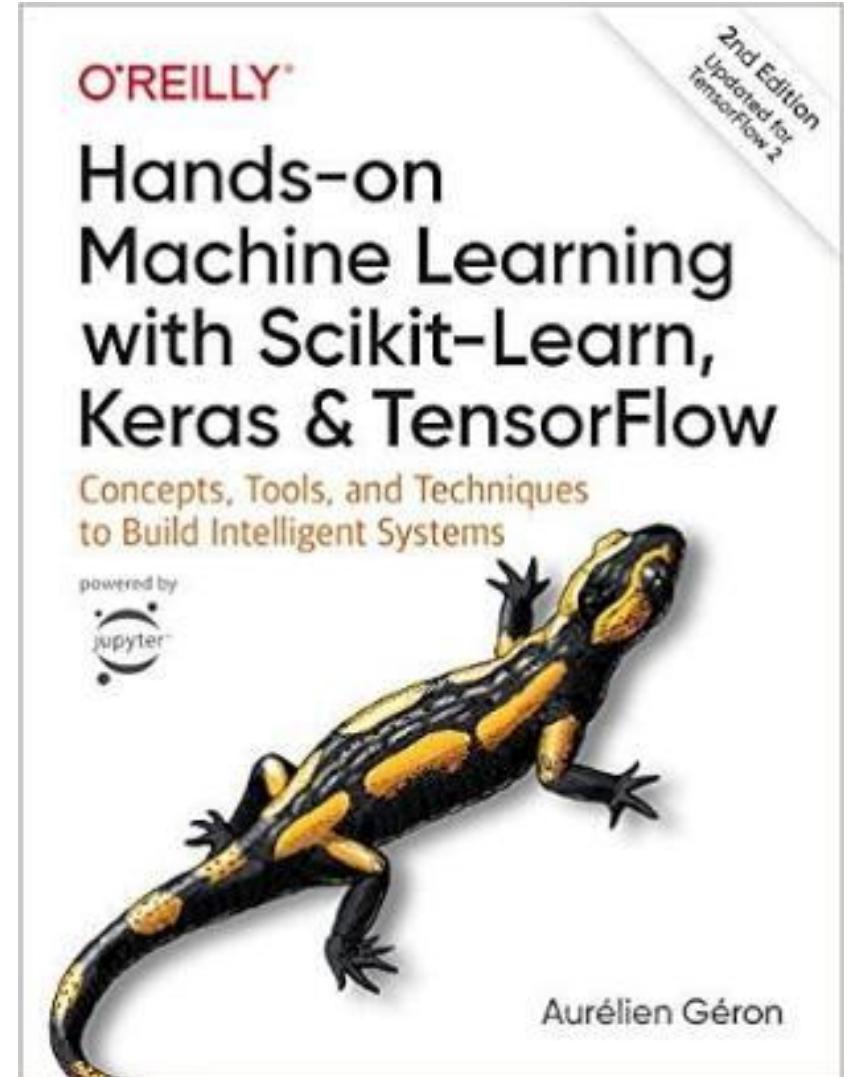
- Instalar scikit-learn

<http://scikit-learn.org/stable/install.html>

## Leituras:

- Capítulos 1 e 2 do livro-texto
- Revisar Python, Pandas, Matplotlib, Numpy, e conceitos de Análise Exploratória de Dados:

<https://www.youtube.com/playlist?list=PLQAOuAPaQsTNfQYcnO1-Jjvo0Y1b8Renj>





Insper

## Tipos de Variáveis Aleatórias

- Contínua: número real
  - Discretas:
    - Ordinal: Existe ordem Ex: Bom, médio, ruim
    - Nominal: Não existe ordem Ex: rótulos, nomes
- Categórica

\* Em MACHINE Learning TODA feature tem que ser numérica:

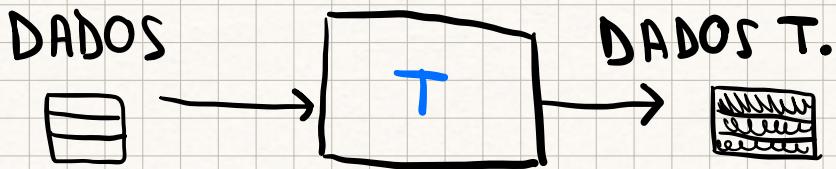
- Contínua: usa como está
- Categórica: cria variáveis dummy

EX:

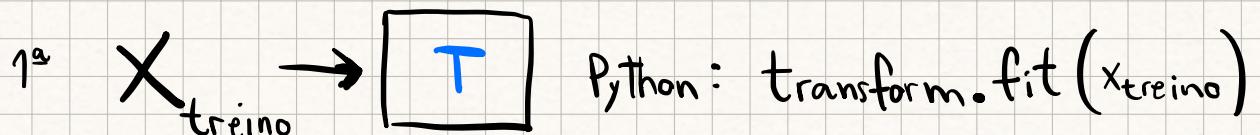
cor:

●	1	0	0
●	0	1	0
●	0	0	1

# TRANSFORMADORES

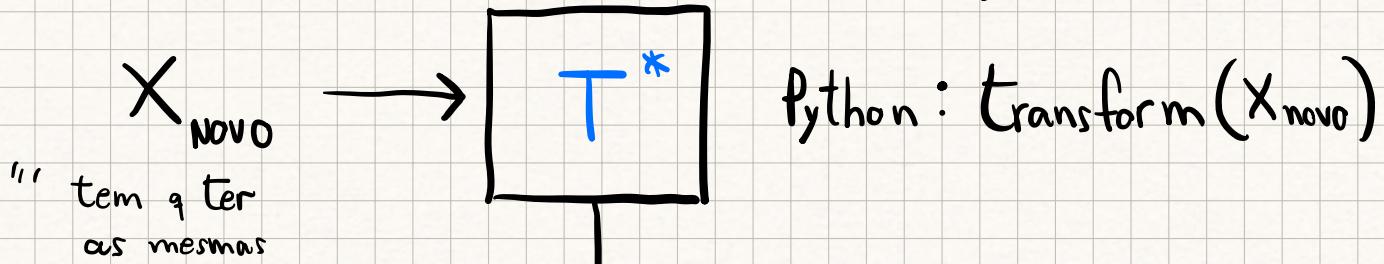


FASE 1. "TREINO" Com DADOS TESTE



FASE 2. APlicar TRANSF. EM OUTROS  
DATASETS

→ É IMPORTANTE QUE AMBOS  
TREINO E TESTE SEJAM  
DO MESMO UNIVERSO



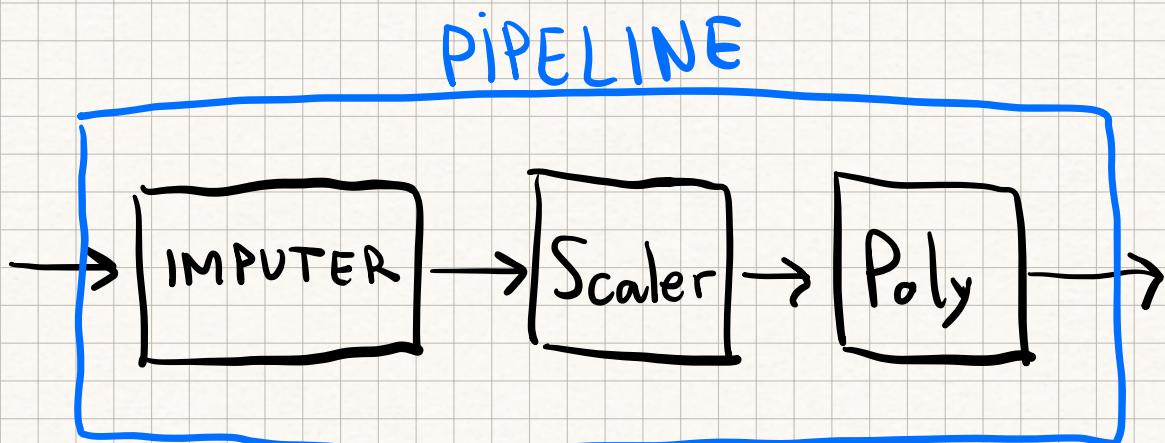
colunas q  
o Xtreino tem

X  
NOVO  
transf.

# PIPELINE

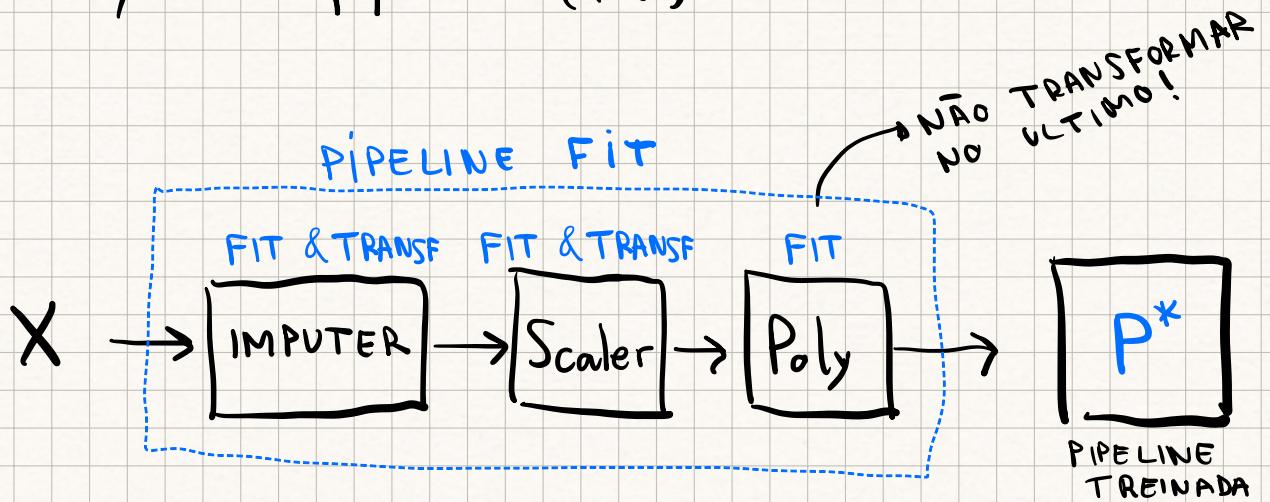
COMBINA VARIAS TRANSFORMAÇÕES  
EM UMA SÓ

PIPE = Pipeline ( [  
    ("imputer", SimpleImputer()),  
    ("scaler", StandardScaler()),  
    ("poly", PolynomialFeatures(degree = 2))  
])



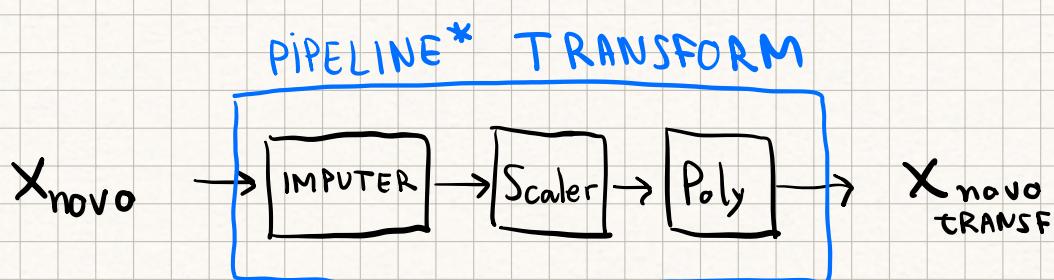
## FASE 1. "TREINAR" o PIPE

Python: pipe.fit(X)



## FASE 2. APLICAR EM OUTRO DATASET

$$X_{\text{novo}} = \underset{\text{TRANSF}}{\text{PIPE}. \text{transform}}(X_{\text{novo}})$$



A VIDA DE UM

# MODELO

## nível 0: DEPLOY

- Modelo já foi **certificado**
- Fazer Treino final

1.

`modelo.fit(X, y)`

→ DADOS COMPLETOS

2. Salvar modelo e colocar em uso

## nível 1: CERTIFICAÇÃO

- modelo já foi escolhido
- Já ajustamos os **hiperparâmetros**

• Treina com  $(X_{\text{treino}}, y_{\text{treino}})$

`modelo.fit(X_treino, y_treino)`

e Avalia.

`y-pred = modelo.predict(X-test)`

rmse = root\_mean\_squared\_error  
(y-pred, y-test)

- Certificar o modelo junto com o business expert
- Se certificado, vai para o nível 0

## nível 2: ESCOLHA DE MODELO

- Temos uma lista de modelos candidatos
- Testar todas as combinações de modelos X hiperparâmetros
- Para cada modelo x hiperparam:
  - Treina em  $(X_{treino\_val}, y_{treino\_val})$
  - $\boxed{\text{modelo. fit}(X_{treino\_val}, y_{treino\_val})}$

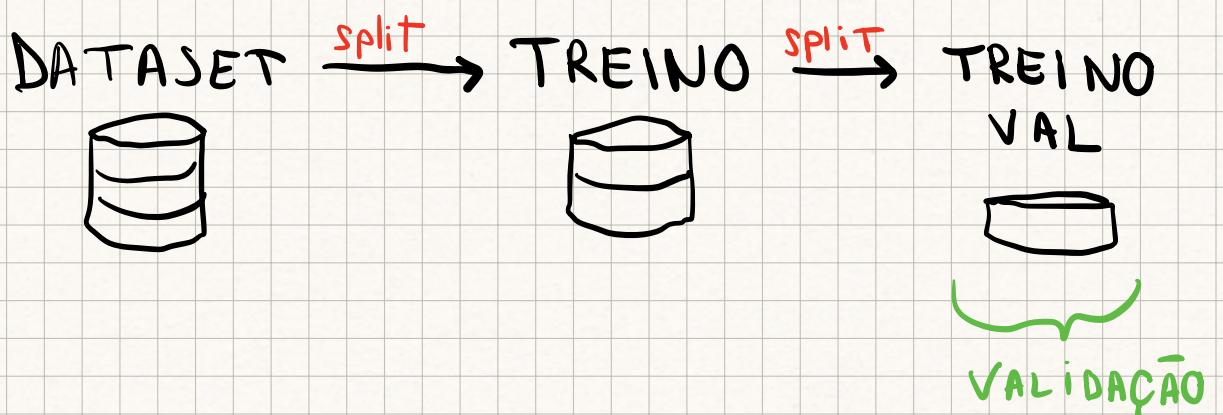
- Avalia em ( $X_{treino}$  val e  $y_{treino}$  val)

$y_{pred} = \text{modelo.predict}(X_{test\_val})$

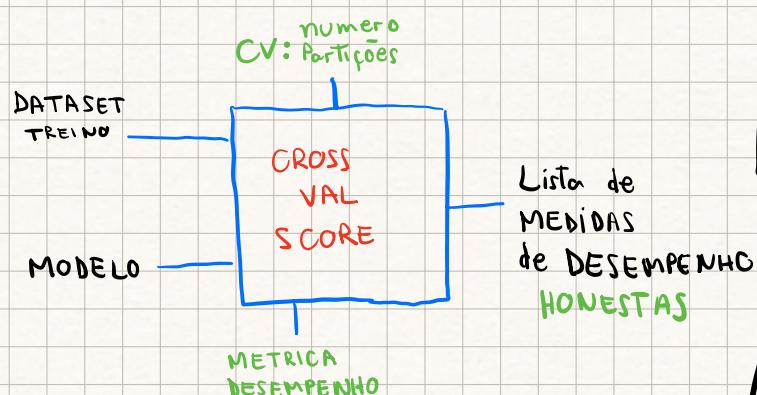
$\text{rmse} = \text{root\_mean\_square\_error}$   
( $y_{pred}$ ,  $y_{test\_val}$ )

- Escolhe o melhor e vai p/ nível 1

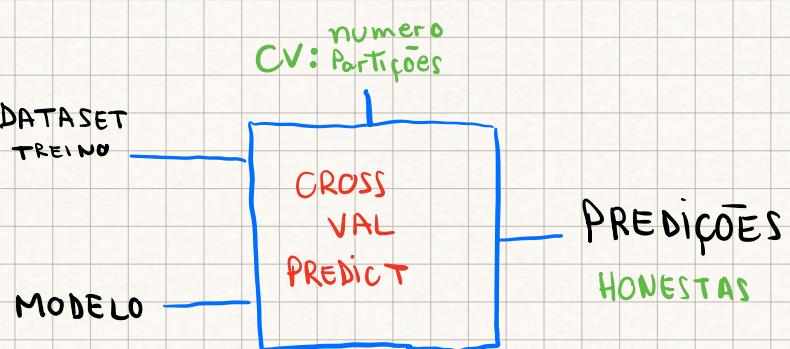
NOTA\*: Splits de DATASET



CROSS-VAL-SCORE



CROSS-VAL-PREDICT



VALIDAÇÃO  
CRUZADA

## Precision e Recall

Para um classificador **BINÁRIO**, as células da matriz de confusão tem nomenclatura especial:

NOME: { TRUE } { FALSE } { Positivo } { Negative }

↑  
SE o  
CLASSIFICADOR  
ACERTOU  
OU  
ERROU  
A PREDIÇÃO

↑  
CLASSE  
PREDITA

### METRICAS DERIVADAS:

$$\bullet \text{ Acurácia} = \frac{\# \text{ACERTOS}}{\# \text{TOTAL}} \therefore \frac{\text{TP} + \text{TN}}{\text{TOTAL}}$$

$$\bullet \text{ Precision} = \frac{\text{# positivos previstos CORRETAMENTE}}{\text{# positivos previstos TOTAL}} \therefore \frac{\text{TP}}{\text{TP} + \text{FP}}$$

*"Dos que classifiquei como POSITIVO, quantos realmente são?"*

$$\bullet \text{ Recall} = \frac{\text{# positivos previstos CORRETAMENTE}}{\text{# positivos reais TOTAL}} \therefore \frac{\text{TP}}{\text{FN} + \text{TP}}$$

*"Dos que são realmente Positivos, quantos eu classifiquei assim"*

## Precision - Recall - Trade OFF

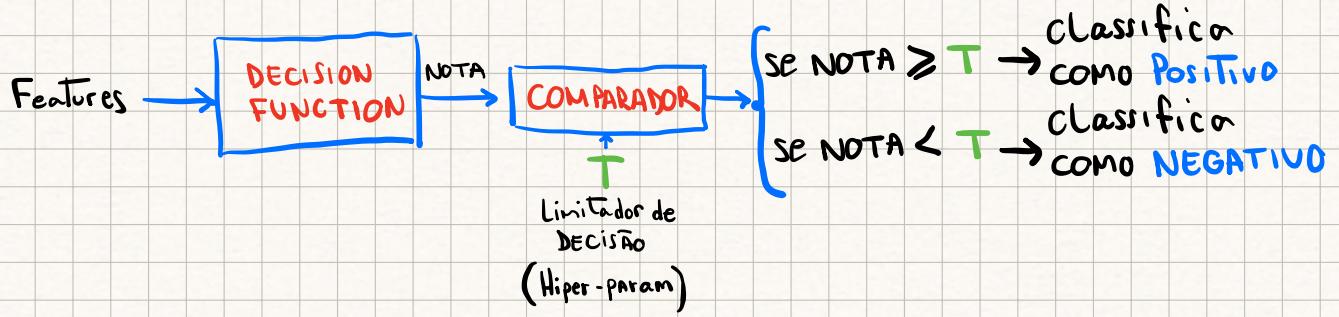
• Precision  $\Rightarrow$  celetividade

• Recall  $\Rightarrow$  abrangência

\* O que irá ditar as mudanças de Precision e Recall é o business! \*

## Decision Function

"Nota" que o classificador dá para a AMOSTRA



## Regularização

A REGULARIZAÇÃO É  
INIMIGA  
DO OVERRFITTING

MODELO: Conjunto de Funções Preditoras

EX: MODELO LINEAR da Forma:

$$\hat{y} = \theta_0 + \theta_1 x_1$$

$$\hat{y} = 7 + 3x_1$$

$$\hat{y} = 0 + 2x_1$$

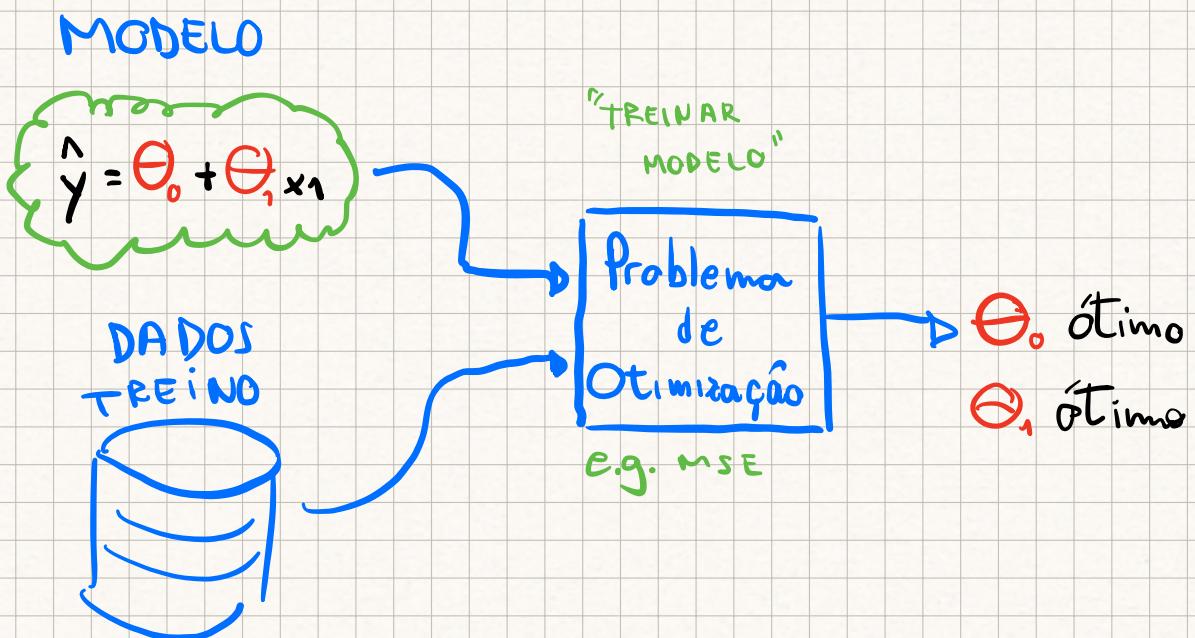
$$\hat{y} = 9 + (-4)x_1$$

$$\hat{y} = 4 + 0x_1$$

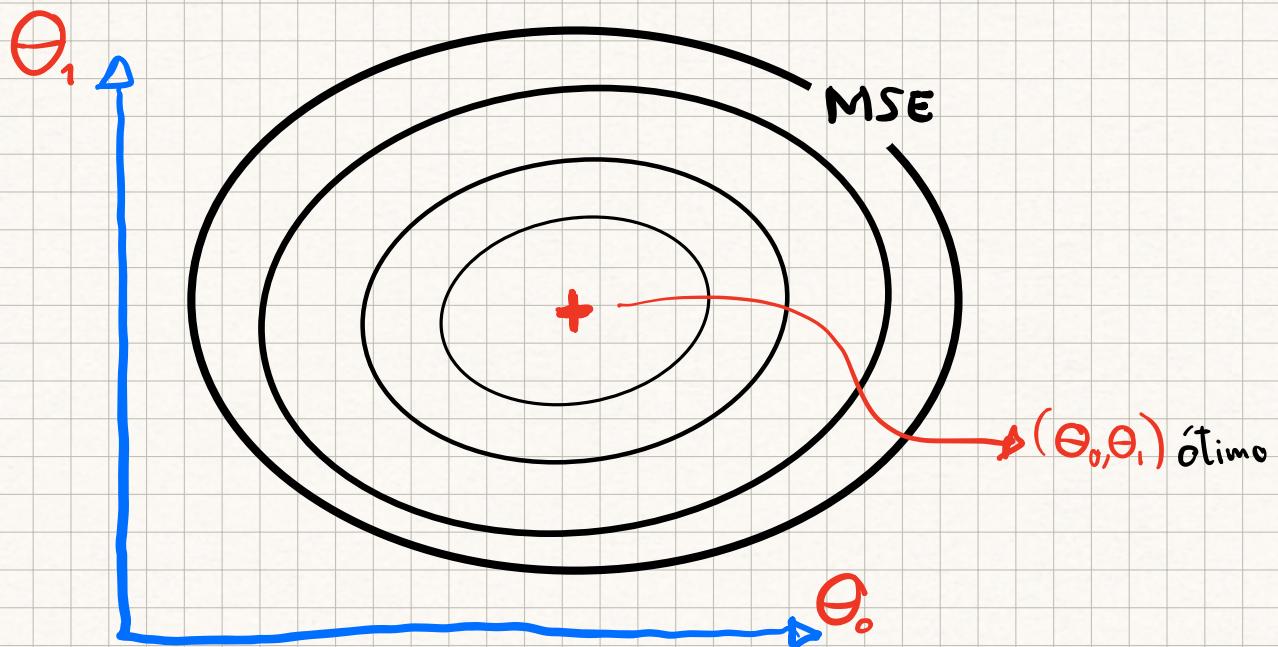
...

Treinar um modelo: escolher a Função Preditora que melhor se ajusta aos dados de Treinamento

EX:



## OVERFITTING



REGULARIZAÇÃO  $\Leftrightarrow$  EMBUTIR CRENÇA  $\text{à priori}$

MOEDA

$P = \text{Prob. CARA}$

<u>RESULTADOS</u>	<u>COM CRENÇA ANTERIOR</u>	<u>SEM CRENÇA ANTERIOR</u>
ANTES	$P = 0.5$	???
1 CARA	$P = 0.5 + 10^{-8}$	$P = 1$
50 CARAS	$P = 0.7$	$P = 1, \text{ com FORÇA}$

ESTIMANDO PROB CARA

1. SEM CRENÇA

$$P = \frac{\#\text{CARAS}}{\#\text{LANÇAMENTOS}}$$

2. COM CRENÇA

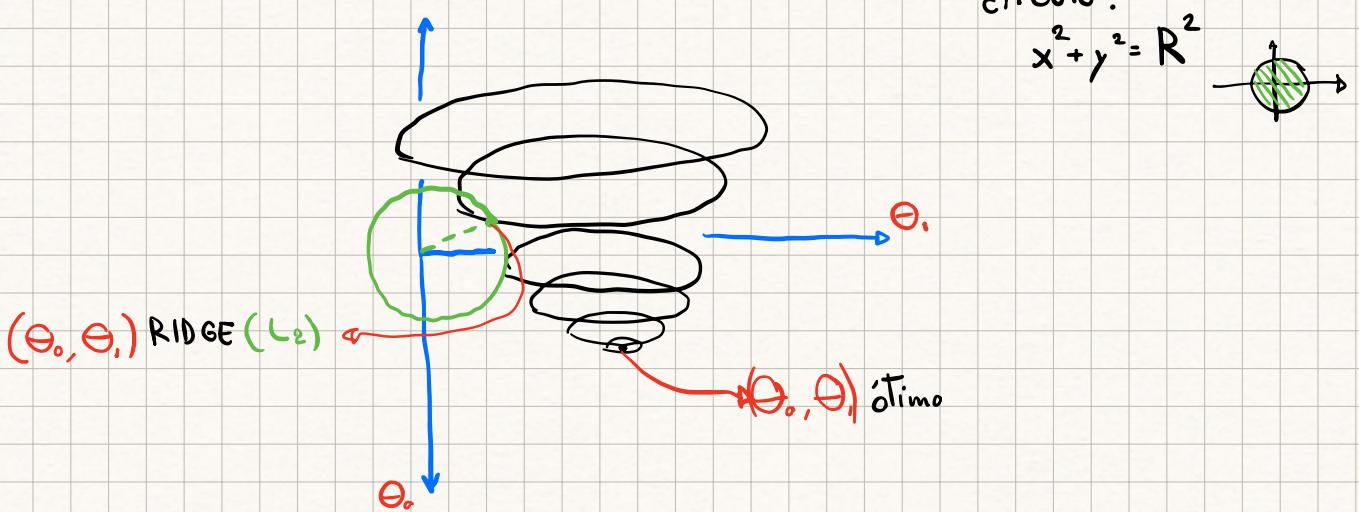
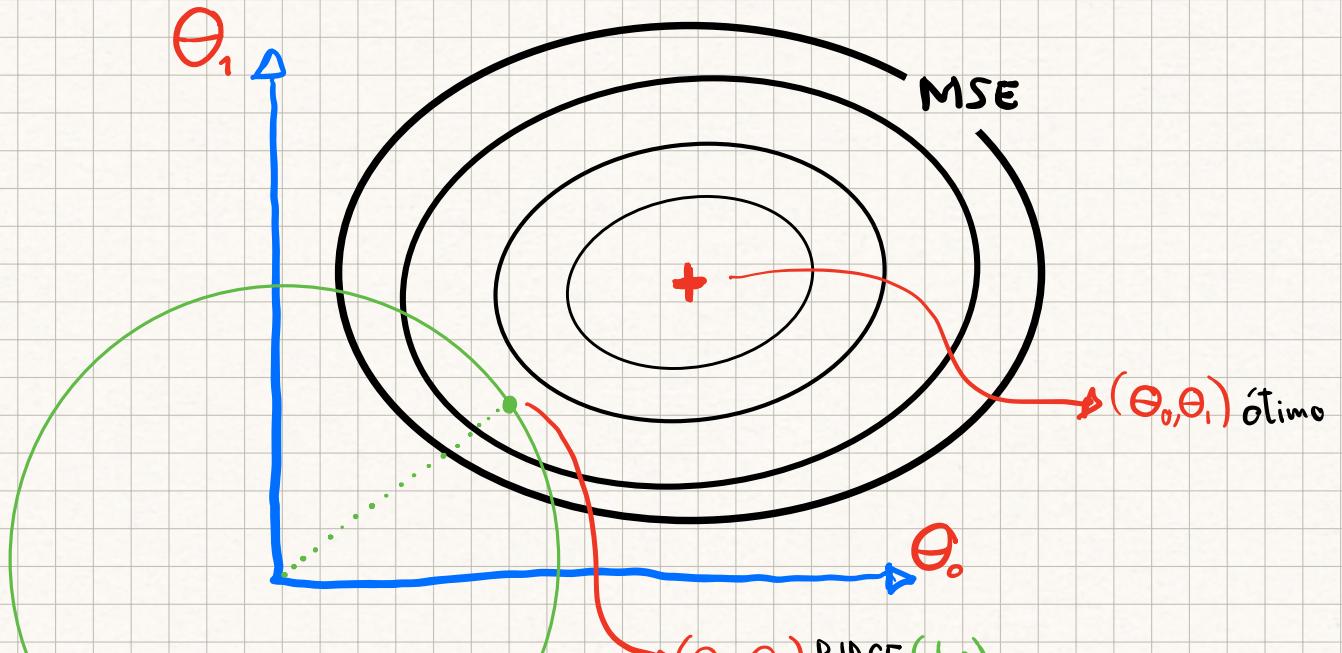
$$P = \frac{\#\text{CARAS} + \alpha}{\#\text{LANÇAMENTOS} + 2\alpha}$$

hiperparam  $\alpha \neq 0$

# REGULARIZAÇÃO DE MODELOS

## M.L.

⇒ RESTRINGIR O ESPAÇO DE PARAMETROS  
 (POR QUE DEPENDENDO, DA OVERFIT)



# REGULARIZAÇÃO RIDGE

1. Sem regularização

$$\vec{\theta}_{\text{ótimo}} = \arg \min \text{MSE}(\vec{\theta}, \text{DT}) \quad \xrightarrow{\text{DATA TREINO}}$$

2. Com regularização

$$\vec{\theta}_{\text{RIDGE}} = \arg \min \text{MSE}(\vec{\theta}, \text{DT}) \quad \xrightarrow{\text{DATA TREINO}}$$

$$\text{SUJEITO A: } \|\vec{\theta}\|^2 \leq \beta$$

Hiperparam:  $(\text{Área de Busca})^2$

# REGULARIZAÇÃO RIDGE 2.

Com regularização

$$\vec{\theta}_{\text{RIDGE}} = \arg \min \text{MSE}(\vec{\theta}, \text{DT})$$

$$\text{SUJEITO A: } \|\vec{\theta}\|^2 \leq \beta$$

↑ multiplicadores  
de Lagrange

$$\vec{\theta}_{\text{RIDGE}} = \arg \min \text{MSE}(\vec{\theta}, \text{DT}) + \alpha^* (\|\vec{\theta}\|^2)$$

Hiperparam  
tem sentido inverso  
em relação ao  $\beta$

## Vantagens do RIDGE

• Combate OVERFIT

• Combate COLINEARIDADE \*GRANDE VANTAGEM

↳ Ex:

$$\hat{y} = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2$$

• o que acontece se, por acidente,  
 $x_1 = x_2$  ?

1. Sem RIDGE: Problemas de Convergência

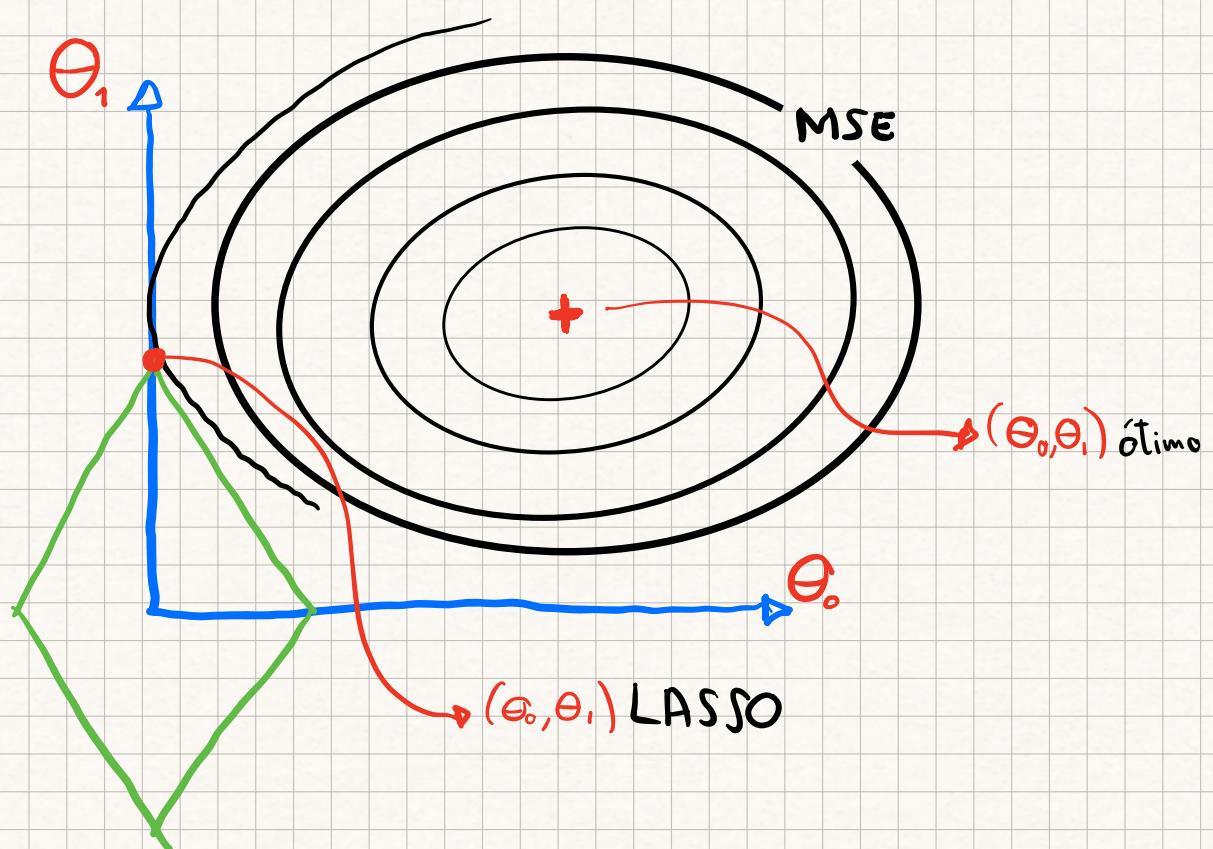
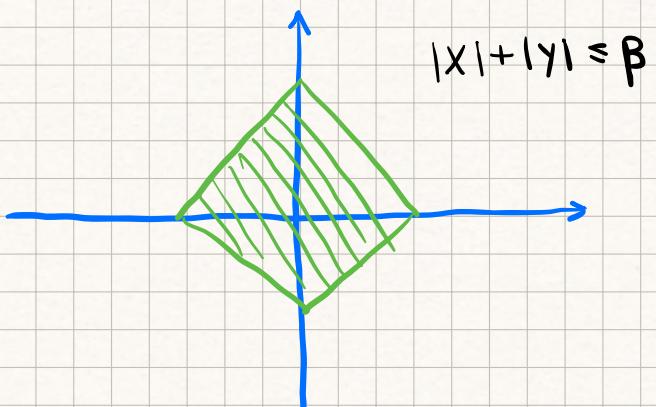
2. Com RIDGE: Converge para  $\Theta_1 = \Theta_2$   
sem problemas de convergência

# REGULARIZAÇÃO LASSO

(Least Absolute Shrinkage Selection Operator)

$$\vec{\theta}_{\text{LASSO}} = \arg \min \text{MSE}(\vec{\theta}, \mathbf{D}\mathbf{T})$$

SUJEITO A:  $\sum_{i=0}^n |\vec{\theta}_i| \leq \beta$



# Ridge

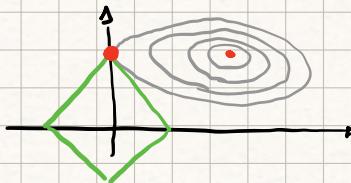
$$L(\vec{\theta}) = \text{MSE}(\vec{\theta}) + \alpha \left( \frac{\text{NORMA}}{\ell_2 \text{ de } \vec{\theta}} \right)$$

- combate **COLINEARIDADE**
- razão intuitiva: força balanceamento dos pesos

# LASSO

$$L(\vec{\theta}) = \text{MSE}(\vec{\theta}) + \alpha \left( \frac{\text{NORMA}}{\ell_1 \text{ de } \vec{\theta}} \right)$$

- seleção automática de features



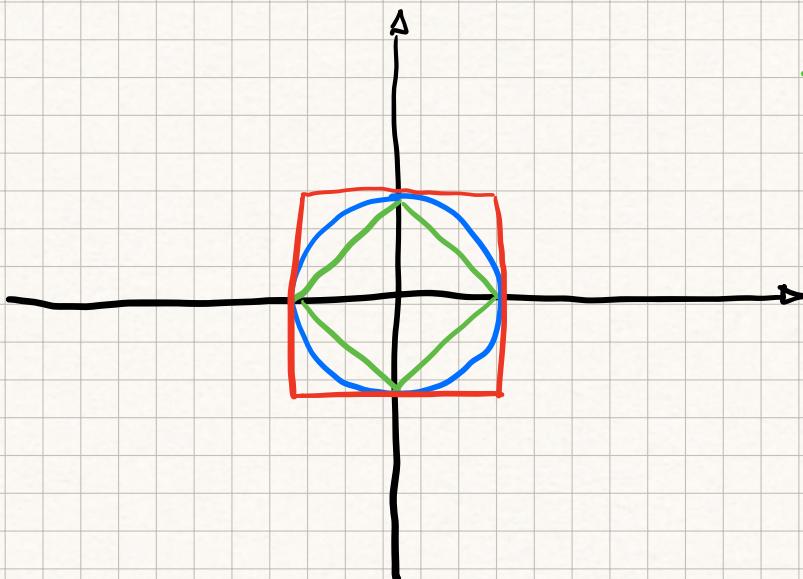
# ELASTIC NET

$$\begin{aligned} L(\vec{\theta}) &= \text{MSE}(\vec{\theta}) + \alpha \cdot r \cdot \left( \frac{\text{NORMA}}{\ell_2 \text{ de } \vec{\theta}} \right) \\ &+ \alpha \cdot (1 - r) \cdot \left( \frac{\text{NORMA}}{\ell_1 \text{ de } \vec{\theta}} \right) \end{aligned}$$

\*ONDE  $0 \leq r \leq 1$

- vantagens de ambos

# GEOMETRICAMENTE



$$l_1(\vec{v}) = \sum_i |v_i|$$
$$l_2(\vec{v}) = \sqrt{\sum_i (v_i)^2}$$
$$l_\infty(\vec{v}) = \max_i |v_i|$$

## REGREÇÃO LOGÍSTICA

• Em primeiro lugar:  
Reg. Logística: NÃO  
é Regressão, c' Classificação

# Unsupervised Learning

- Redução de Dimensionalidade

- PCA
- T-SNE
- UMAP

- Clustering

- K Means
- Mean Shift
- Agglomerative Clustering

Supervised

VS

Unsupervised

Foco: Previsão

Mecanismo: Aprender

a prever baseado nas  
features  $X_T$  e no valor  
desejado do Target  $Y_T$

Mecanismo: Descobrir "padrões"

Foco: • Análise Exploratória

• Feature engineering

• Geração de dados  
(Semi-Supervised)

(TEM RESPOSTA CERTA)

nas features.

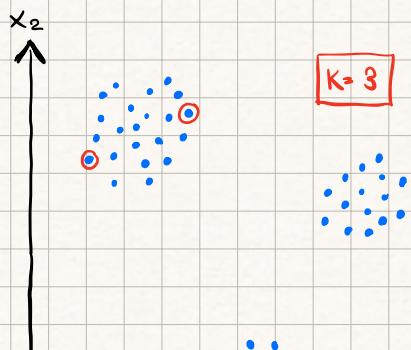
(NÃO TEM RESPOSTA CERTA)

# APRENDIZADO NÃO SUPERVISIONADO

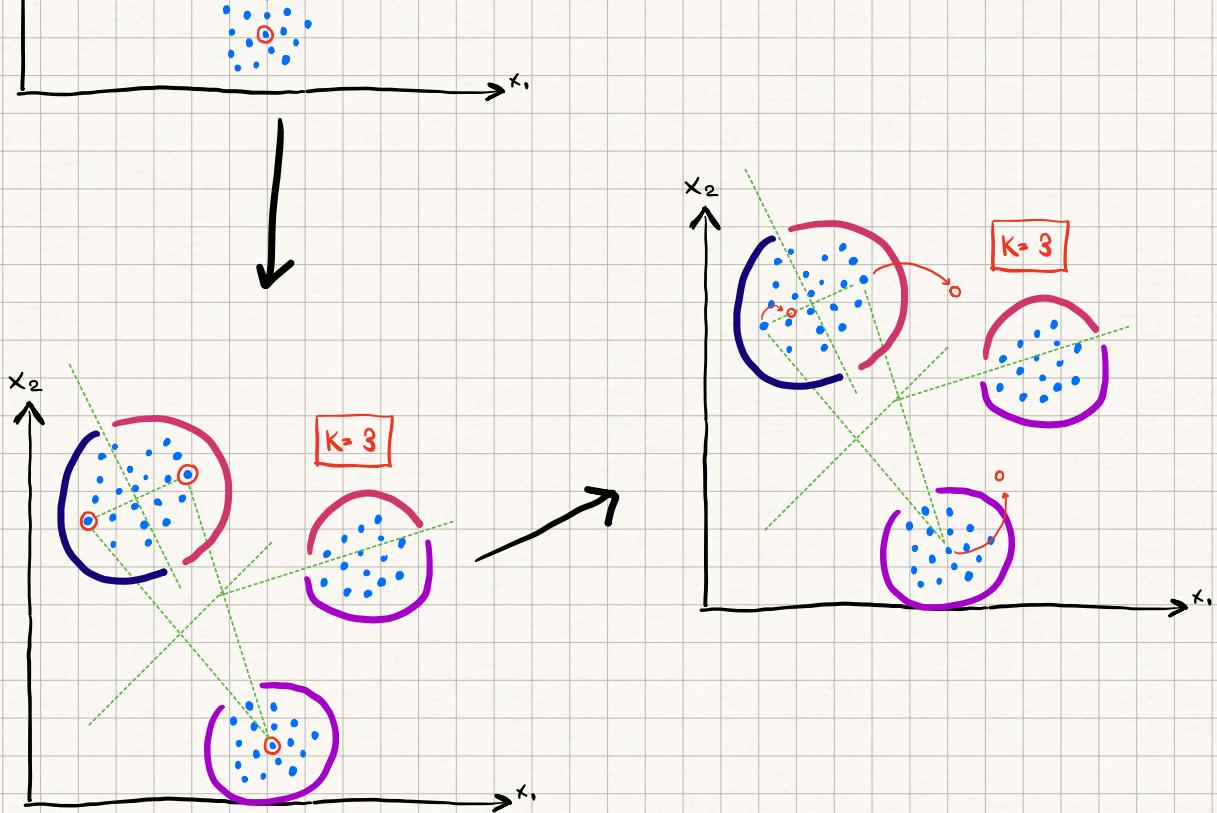
- Clustering: Associar os objetos à agrupamentos "naturais"
- Redução de Dimensionalidade: Encontrar uma representação dos dados com menos dimensões (features)

**CLUSTERING: "AGRUPAMENTO"**  
\* Usar STANDART scaller \*

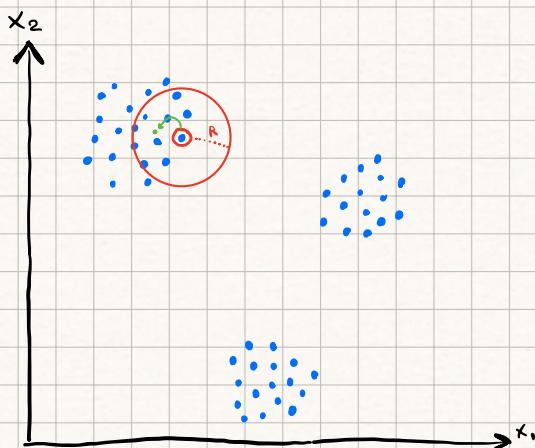
## K-MEANS



- 1- Escolha K = nº de clusters
- 2- Escolha K CENTRÓIDES aleatórios
- 3- Mova os CENTRÓIDES para o local médio de seus clusters (ate parar de mudar)



## MEAN SHIFT



- 1- Definir o  $R$ , raio de influência
- 2- Migrar os pontos (Todos) para o local médio de seus vizinhos (dentro do  $R$ )
- 3- Quando pontos se agrupam 100%, temos a formação de  $J$  Cluster, e assim por diante

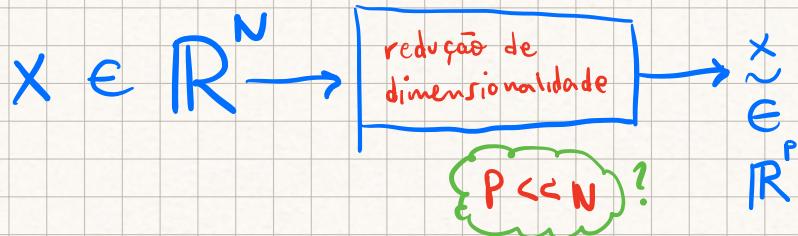
# Enxugando a Informação

Benefícios:- Compactação

- Entendimento

Generalização

# Redução de Dimensionalidade



Motivação : - Compactação :

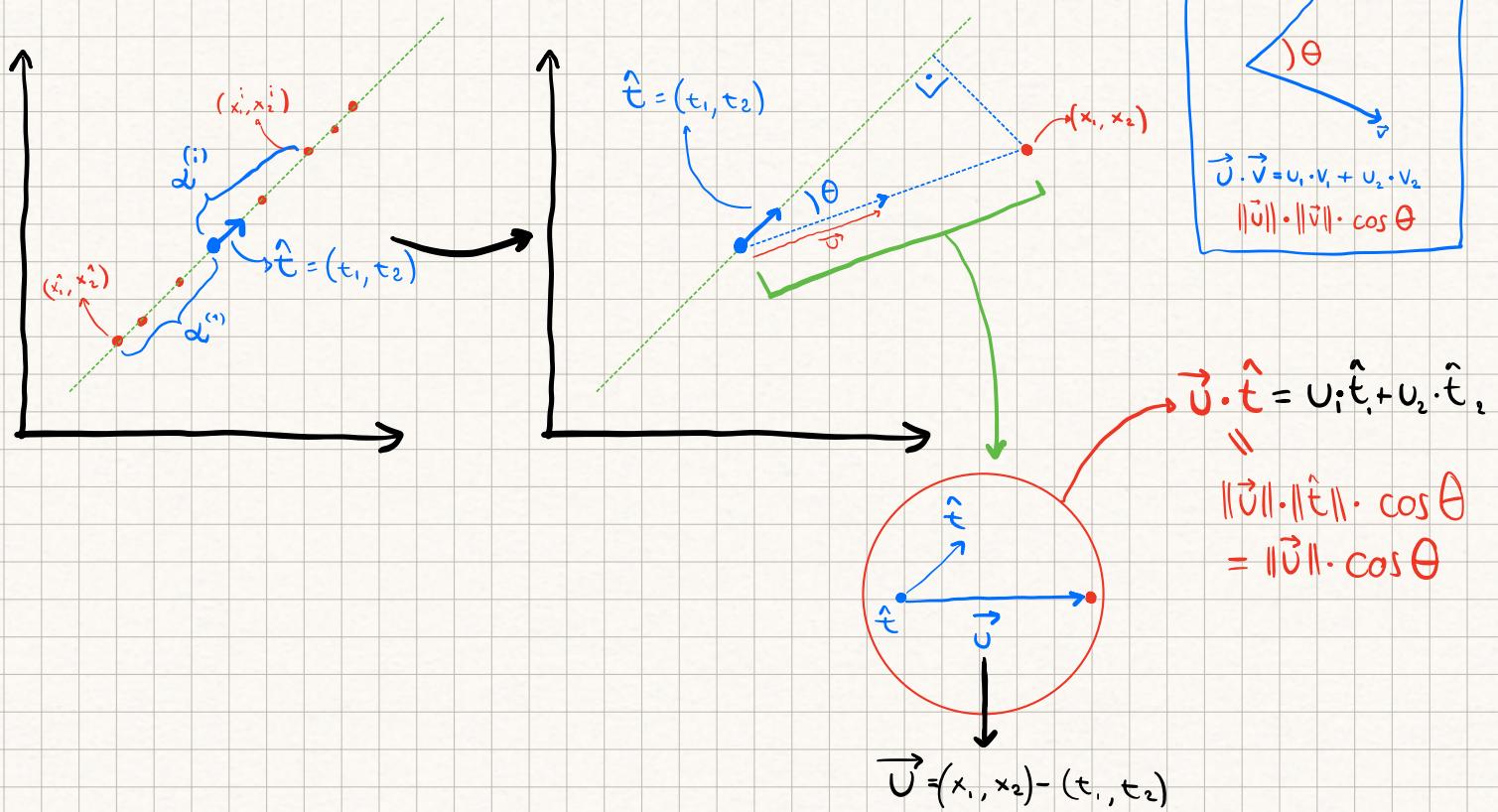
Compressão dos dados com perdas

- Entendimento

Descobrir a "essência" dos dados

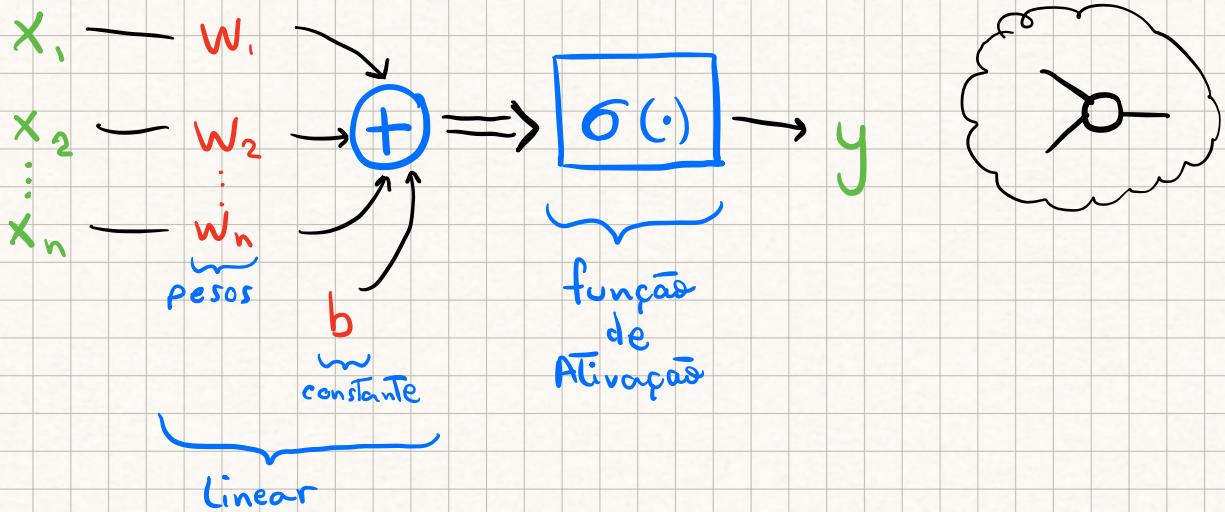
Algumas Técnicas : - PCA  
 - t-SNE  
 - Auto encoders

## PCA

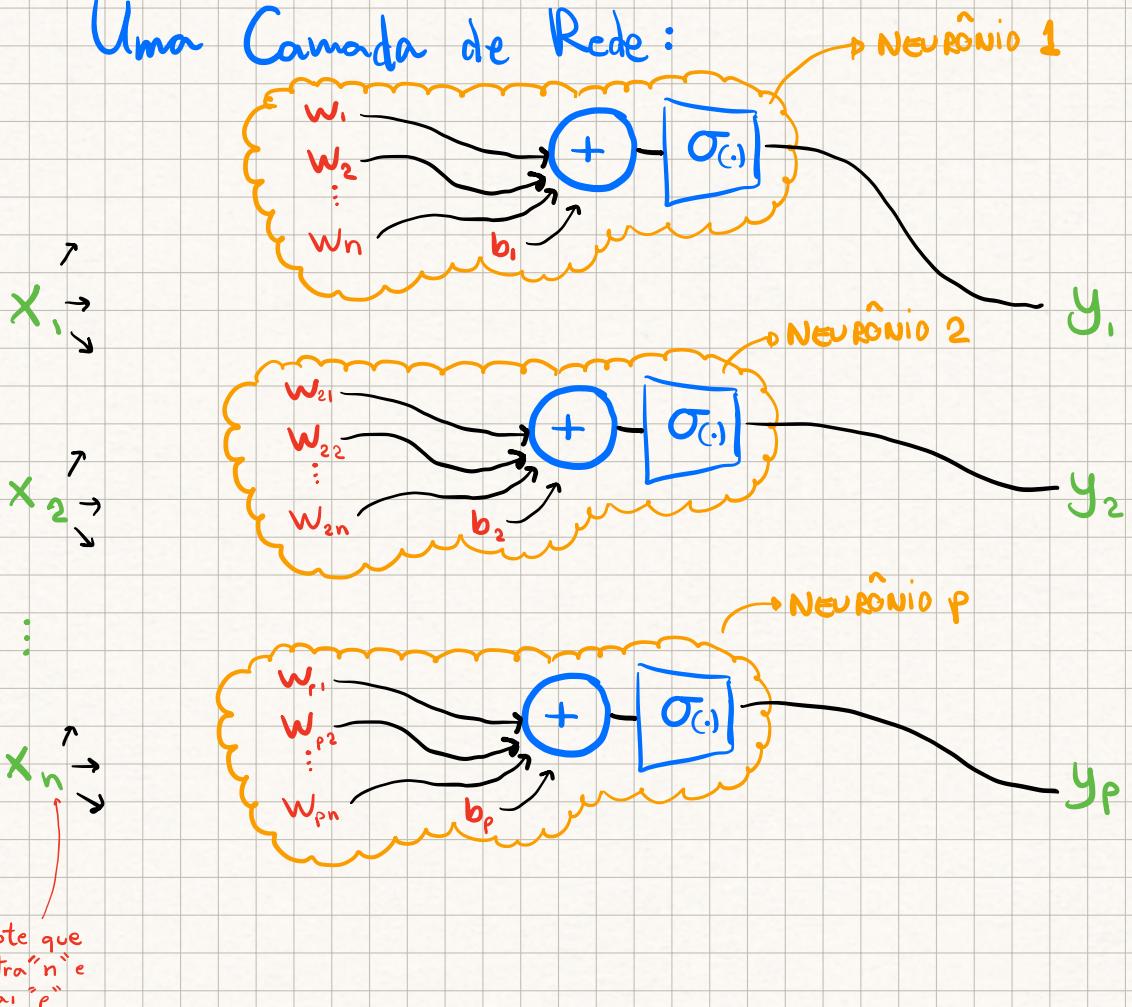


Redes  
 Neurais  
 ...

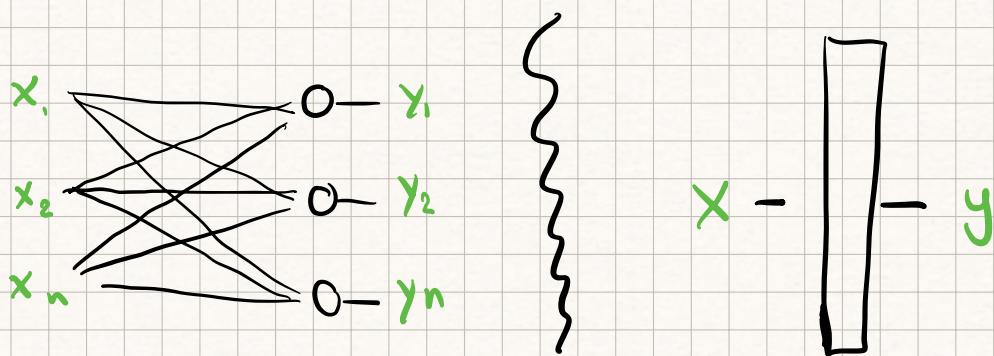
# O Neurônio :



## Uma Camada de Rede:



# Diagrama mais Simples

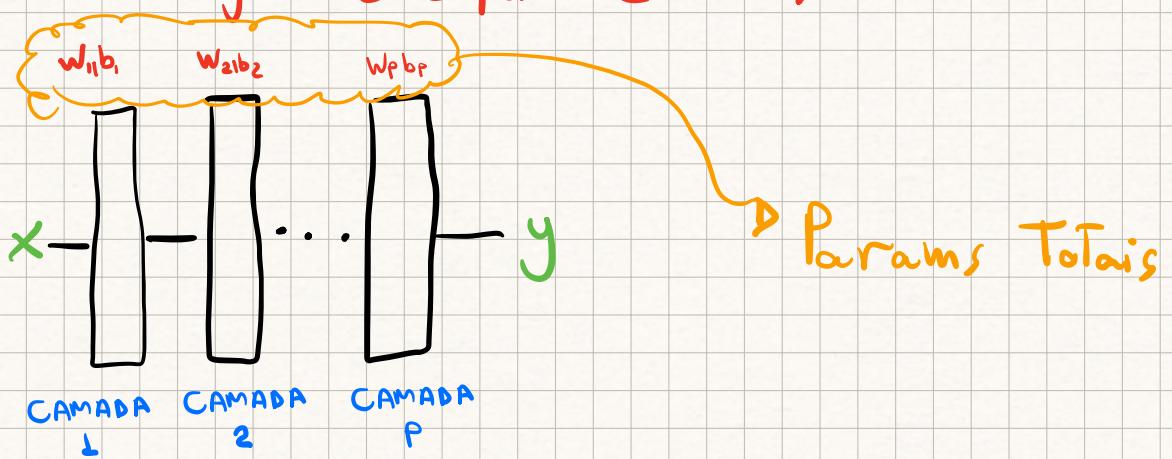


Parâmetros da Camada:

$$W = \begin{bmatrix} w_{11} & w_{12} \dots & w_{1n} \\ w_{21} & w_{22} \dots & w_{2n} \\ w_{31} & w_{32} \dots & w_{3n} \end{bmatrix} \xrightarrow{\quad} b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{bmatrix}$$

Rede Neural

Multi-Layer Perceptron (MLP)



Exemplo: Num MLP de 3 camadas

Temos o seguinte numero de neuronios por camada:

CAM	Nº NEU
1	10
2	5
3	1

A entrada tem 20 features,  
qnts param ao todo tm a rede?

R: 1º camada

$$20 - \boxed{10} \Rightarrow$$

$$20 \times 10 = W_T = 200$$

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ w_{31} & w_{32} & \dots & w_{3n} \end{bmatrix}$$

$$10 = b_T = 10$$

$$\vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{bmatrix}$$

2º camada

$$10 - \boxed{5} \Rightarrow$$

$$10 \times 5 = W_T = 50$$

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ w_{31} & w_{32} & \dots & w_{3n} \end{bmatrix}$$

$$5 = b_T = 5$$

$$\vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{bmatrix}$$

271

3º camada

$$5 - \boxed{1} \Rightarrow$$

$$5 \times 1 = W_T = 5$$

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ w_{31} & w_{32} & \dots & w_{3n} \end{bmatrix}$$

$$1 = b_T = 1$$

$$\vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{bmatrix}$$



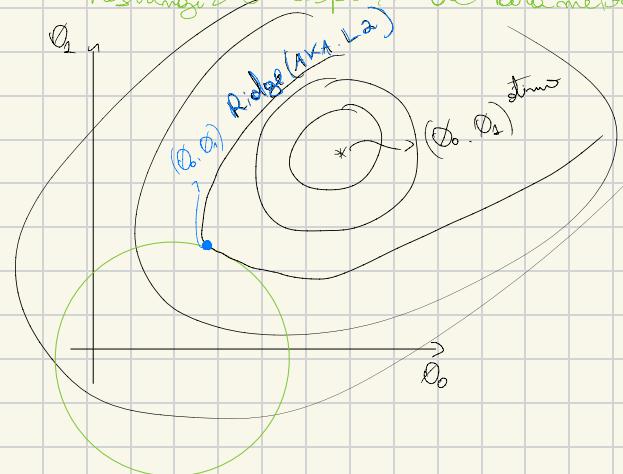
# Regularizações:

\* Uma boa forma de reduzir o overfitting é regularizar o modelo (isso é, restringi-lo). Quanto menor for o grau de liberdade, mais difícil será se sobreajustar aos dados.

## Ridge

\* É uma versão regularizada da regressão linear. Um termo de regularização é adicionado a função de custo. O hiperparâmetro  $\alpha$  controla o quanto você deseja regularizar o modelo. Se  $\alpha = 0$ , a regressão ridge é somente linear. Se  $\alpha$  for muito grande, todos pesos acabarão muito próximos de 0 e o resultado será uma linha que passa pela média dos dados.

→ Restringir o espaço de Parâmetros



### 1) Sem Regularizações

$$\vec{\theta}_{\text{otimo}} = \arg \min_{\vec{\theta}} \text{MSE}(\vec{\theta}, \text{Treino})$$

### 2) Com regularizações

$$\vec{\theta}_{\text{Ridge}} = \arg \min_{\vec{\theta}} \text{MSE}(\vec{\theta}, \text{Treino})$$

$$\|\vec{\theta}\|^2 \leq \beta \rightarrow \text{hiperparâmetro}$$

quadrado do raio  
da área de busca

\* Se você quiser pegar o menor MSE mas que esteja dentro de raio  $\beta$  centrada na origem no espaço dos parâmetros. Se seja o  $\vec{\theta}_{\text{otimo}}$  é o melhor  $\vec{\theta}$  permitido dentro da área possível. Seu isso o modelo pode escolher  $\vec{\theta}$  enormes para minimizar o erro no treino. A limitação ajuda o minimizar o erro sem confor demais em parâmetros muito grandes.

# Vantagens Ridge

- Combate overfitting
- Combate colinearidade

A sua grande vantagem

## Hyperparametros

**Hiper  $\alpha$** : Controla a intensidade da penalização, quanto menor mais liberdade o modelo.

$$\vec{\theta}^* = \arg \min_{\vec{\theta}} (\text{MSE}(\vec{\theta}) + \alpha \|\vec{\theta}\|^2)$$

**Hiper  $\beta$** : Tem sentido de regularização oposto a  $\alpha$ , pois quanto menor o  $\beta$ , maior é o raio onde o  $\vec{\theta}^*$  pode ser selecionado, ou seja menor regularização.

**Poam Standard Scaler**: Com a mesma escala em todas as features a penalização  $\sum \theta_j^2$  age de maneira justa para todas as features, sem favorecer nenhuma dimensão. Ou seja se temos 2 features  $X_1$  e  $X_2$  ( $X_1 = \text{nº de filhos}$  e  $X_2 = \text{renda anual}$ ) o peso associado a  $X_2$ , não precisa ser tão grande para influenciar a predição pq  $X_2$  já é numéricamente grande. Em contraste  $\theta_1$  precisa ser maior para compensar os valores baixos de  $X_1$ . e sem escalar os dados  $L_2$  trata qualquer  $\theta_i$  igualmente.

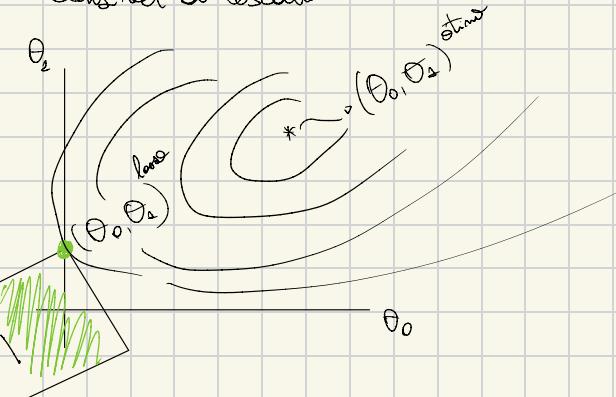
# Lasso

\* É outra versão regularizada da regressão linear. Adiciona um termo de regularização à função de custo, mas usa a norma  $\ell_1$  do vetor de pesos, em vez de metade da quadra de da norma  $\ell_2$  que a ridge usa.

\* Formulação:

$$\min_{\theta} \left( \text{MSE}(\theta) + \lambda \sum_j |\theta_j| \right)$$

- Usa  $\ell_1$  como norma de penalizações
- Tende a zerar coeficientes irrelevantes
- Sensível ao escalar



## Vantagens LASSO

— Seleção automática de features: Se iniciarmos  $\theta_1 = 2$  e  $\theta_2 = 0.5$ , a execução do gradiente descendente reduzirá os dois parâmetros igualmente, assim  $\theta_2$  chegará em 0 primeiro que tende azerar alguns  $\theta_j$ , selecionando automaticamente certas features.

## Hiperparâmetros

\*  $\alpha$ : Igual (quanto maior regularização)

\* StandardScaler: Reis também é sensível ao escalonamento.

# Elastic Net

\* É um meio termo entre Ridge e Lasso. É uma simples combinação dos termos da Ridge e Lasso. É possível controlar a relação dessa combinação em  $\lambda$ . Quando  $\lambda=0$ , a elastic net é a Ridge e quando  $\lambda=1$ , é equivalente a Lasso.

\* Formulação:

$$\min_{\theta} \left( \text{MSE}(\theta) + \lambda_1 \sum_j |\theta_j| + \lambda_2 \sum_j \theta_j^2 \right)$$

## Vantagens Elastic Net

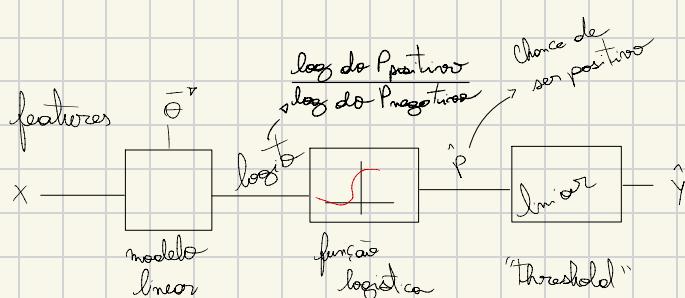
— Vantagens das 2

\* Seleção de variáveis Lasso com robustez contra multicolinearidade do Ridge.

## Hiperparâmetros

\*  $\alpha$ : balanceia relação entre Ridge e lasso.  
 $\alpha=1 \rightarrow \text{Elastic} = \text{Lasso}$  e  $\alpha=0 \rightarrow \text{Elastic} = \text{Ridge}$ .

# Regressão logística



De forma:

• Joga as features no modelo linear, sai o logit (log do Positivo / log do Negativo)

$$\text{logit} = \theta_0 + \theta_1 X_1 + \dots + \theta_n X_m$$

• Joga o logit na função logística, sai o  $\hat{p}$  (Chance de ser da classe positiva).

$$\hat{p} = \sigma(\text{logit})$$

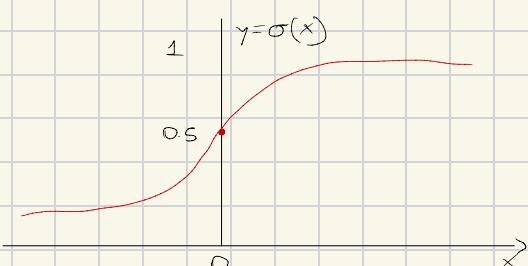
• Se  $\hat{p} > T$  (limiar de decisão) classifica como da classe positiva.

$$\hat{C} = (\hat{p} > T)$$

Mult classe: Um logit para cada classe. Joga os logits na softmax, o  $\hat{p}$  é um vetor de probabilidades, somando 1 a predição final é organizar  $\hat{p}$ .

função logística

$$y = \sigma(x) = \frac{1}{1 + \exp(-x)}$$



Aplicações:

Classificações:

- Binária
- Multiclasses

# Support Vector Machines

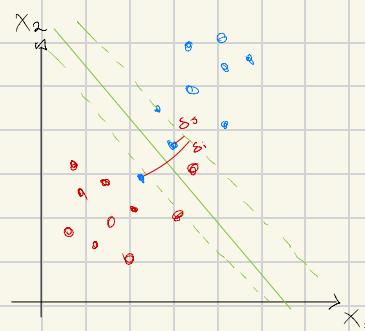
\* A ideia central das máquinas de vetor de suporte é construir uma separação entre classes que seja a maior possível ("a avenida mais larga").

\* Hard SVM: "ninguém pisou na avenida"

- Não se usa (Avenida muito pequena, muitas vezes nem tem solução)

\* Soft SVM: "pode pisar na avenida, mas paga uma penalidade."

- Esse é o que usa
- Hiperparâmetro  $C$ : custo de penalidade
- Baixo  $C$ : pouca penalidade - Baixa regularização - Underfitting
- Alto  $C$ : alta penalidade - Alta regularização - Overfitting



\* Formulação:

$$\text{Score} = w_1 X_1 + w_2 X_2 + \dots + w_m X_m + b$$

$$\text{Classe} = \begin{cases} 1 & \text{se score} > T \rightarrow \text{normalmente} \\ 0 & \text{se score} \leq T \end{cases}$$

Parâmetros:

\* StandardScaler: Sem a normalização o SVM vai priorizar features com maior magnitude pois pensa que essas tem mais importância apenas por conta da escala.

# Clustering

## Supervised learning

\* Foco: Previsão

\* Mecanismo: aprender a prever baseado nas features ( $X_{\text{Treino}}$ ) e ( $Y_{\text{Treino}}$ )

## VS Unsupervised

\* Foco: Análise exploratória

- feature engineering
- Gerção de dados  
(Sem-supervised)

\* Mecanismos: descobrir "padões" nas features (não tem resposta correta)

## Aprendizado não-Supervisionado:

- Clustering: associar os objetos a agrupamentos "naturais"
- Reduzir de dimensionalidade: encontrar uma representação dos dados com menos dimensões (features)

## Funcionamento K-means

\* Escolha de  $K$ : Quantos  $K$  clusters deseja formar (hiperparâmetro definido manualmente)

\* Inicialização dos centroides: Escolhe  $n$  centroides posicionados aleatoriamente no espaço.

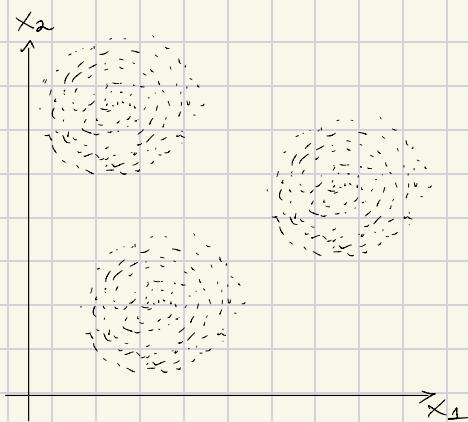
\* Atribuições dos pontos ao centroide: Cada ponto é atribuído ao centroide mais próximo (por distância euclidiana) Formando  $n$  grupos

\* Atualização do Centroide: Cada centroide se move para o centro do seu grupo

\* Volta ao terceiro passo: Repetindo as atribuições dos pontos aos centroides e repositionamento de cada centroide.

\* Para quando: Os pontos atribuídos aos centroides não mudam mais

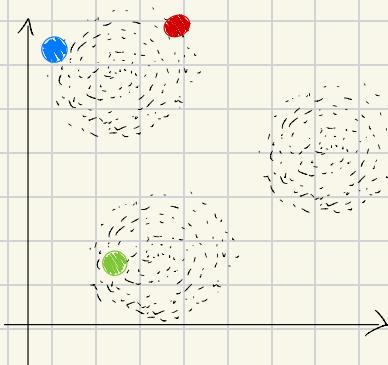
K-means



1- Escolha o  $K = n^2$  de clusters

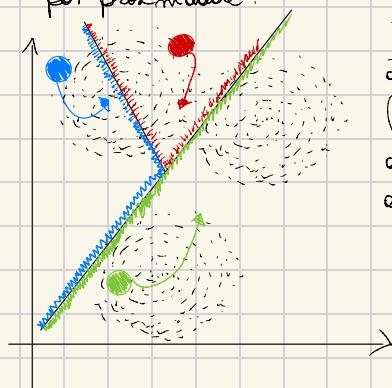
2- Escolha  $K$  centroides

Exemplo:



\* Enquanto não convergir

\* Atribuir os pontos aos centroides por proximidade.



\* Recalcula os centroides até chegar o ponto ótimo (those os pontos que são atribuídos aos seus centroides não mudam mais)

# Mean Shift

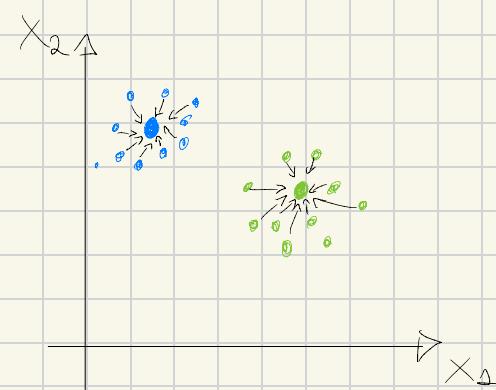
\* Os pontos encontram suas vizinhanças de acordo com um raio e se encontram no meio de caminho até se fundirem em um só.

\* Escolha do hiperparâmetro principal: Define o raio usado para buscar vizinhanças de cada ponto.

\* Para cada ponto: Encontra todos os seus vizinhos dentro do raio. Desloca o ponto atual para o meio do caminho médio entre os pontos.

\* Repete o processo anterior até todos os pontos convergirem para um só.

\* Ilhando o caminho contrário é possível ver quem se agrupou com quem.



# Decision Tree

\* Divide recursivamente os dados com base nas features, escolhendo em cada passo a melhor (divisão) para separar os dados de forma mais "pura".

\* Avalia se o nó deve ser terminal: Se o subconjunto só tem exemplos da mesma classe ou pouquíssimas amostras de outras classes para dividir ou profundidade máxima. Quando isso acontece atribuímos como resposta a classe majoritária.

\* Se é possível dividir: busca qual feature e qual melhor valor de corte para separar melhores os dados (Teste em corte em qual feature e avaliar qual reduz mais a impureza) Gini impurity ou entropia

\* Com a melhor Divisão encontrada: O Conjunto é Separado: um grupo vai para esquerda e outro para a direita (feature  $a \leq t \rightarrow$  esquerda e feature  $a > t \rightarrow$  direita)

\* Cada subconjunto passa pelo mesmo Processo: Continuar Construindo a arvore chamando recursivamente para cada nó

# Hiperparâmetro

\* A profundidade da arvore é muito importante para não ter overfitting

Ensemble

# Machine Learning

- Tópicos de hoje:

- Avaliação cruzada
- Seleção de hiperparâmetros
- escolha de modelo

A vida do modelo...

## Nível 0: Deploy

- modelo já foi certificado
- Agora, fazer o **treino\_final**

1.  $\text{modelo}.f.f(X, y)$  ← dados completos!
2. Salvar modelo e colocar em uso!

## Nível 1: Certificação

- modelo **ESCOLHIDO**
- já escolhemos o modelo

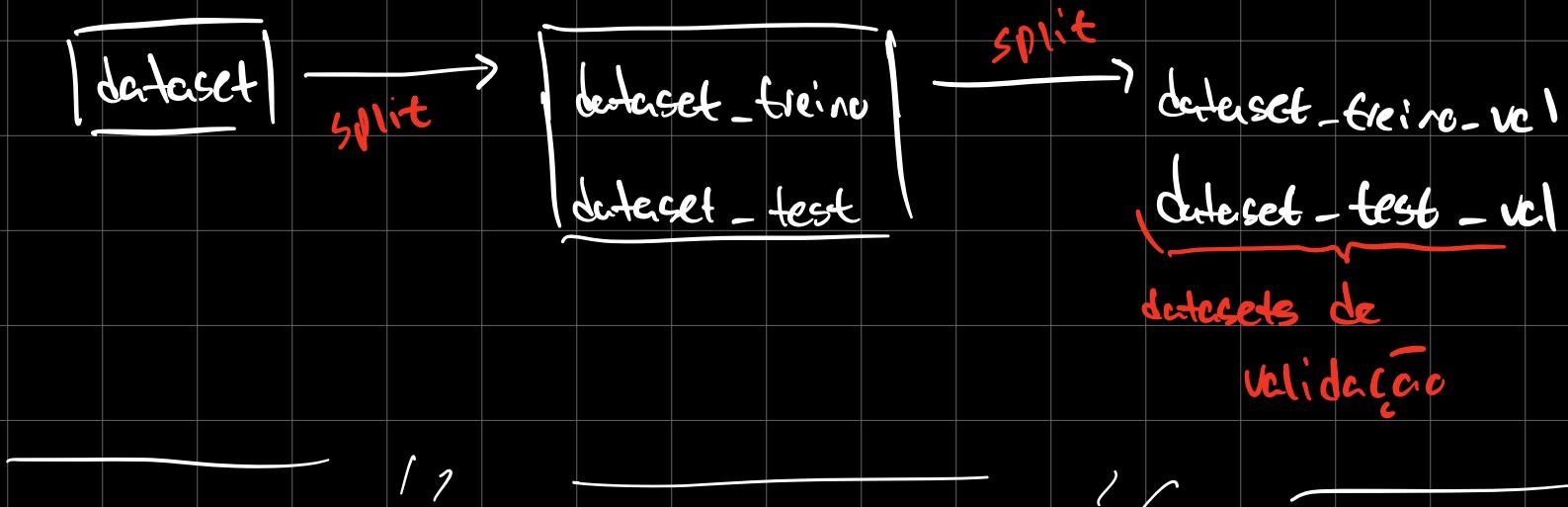
- já ajustamos os  $\hat{w}$  para os hiperparâmetros
- treina com  $(X_{-treino}, y_{-treino})$   
 $\text{modelo}.fit(X_{-treino}, y_{-treino})$
- Avalia com  $(X_{-test}, y_{-test})$   
 $y_{-pred} = \text{modelo}.predict(X_{-test})$   
 $\text{rmse} = \text{root-mean-squared-error}(y_{-pred}, y_{-test})$
- **Certificar** o modelo
  - Decidir, junto com o *business expert*, se o modelo é suficientemente bom.
  - Se **certificado**, vai para nível  $\emptyset$

## Nível 2: ESCOLHA DE MODELO

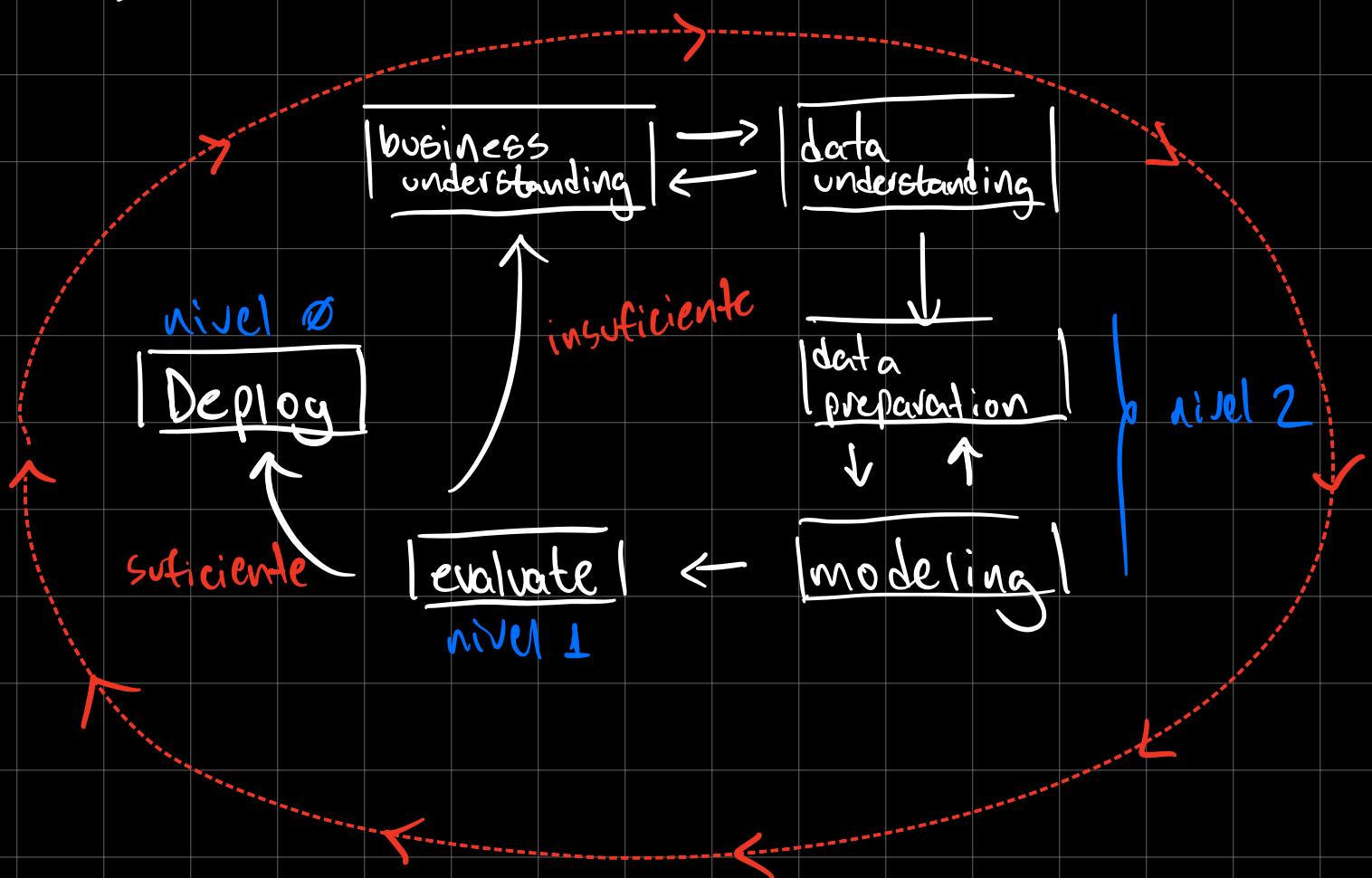
- temos uma lista de modelos candidatos
  - todas as combinações de modelos e respectivos hiperparâmetros
- Para cada modelo:
  - treino em  $(X_{-treino\_val}, y_{-treino\_val})$   
 $\text{modelo}.fit(X_{-treino\_val}, y_{-treino\_val})$
  - Avalia em  $(X_{-test\_val}, y_{-test\_val})$   
 $y_{-pred} = \text{modelo}.predict(X_{-test\_val}, y_{-test\_val})$   
 $\text{rmse} = \text{root-mean-squared-error}(y_{-pred}, y_{-test\_val})$

- Escolhe o melhor modelo e vai p/ nível 1

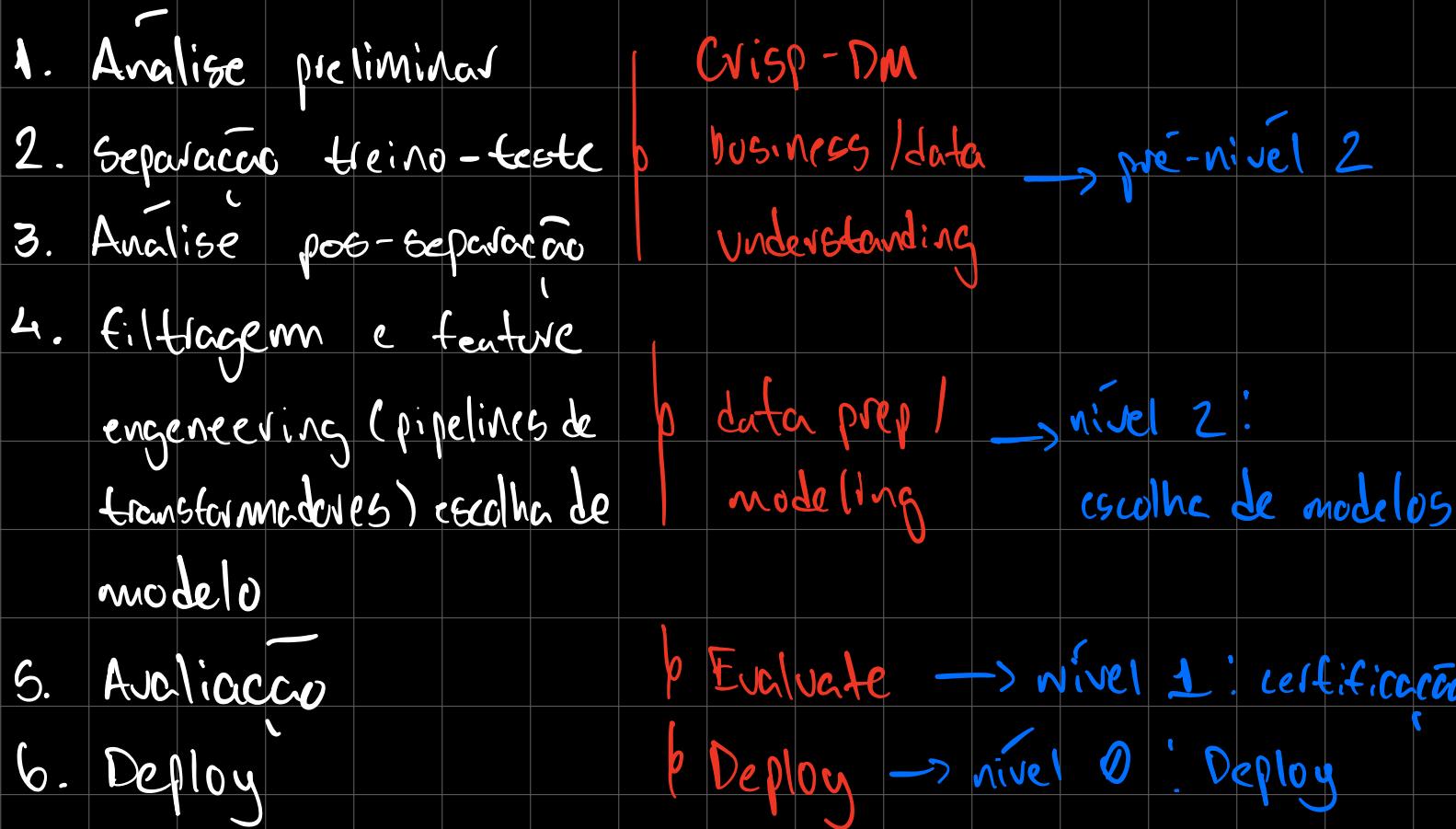
Nota: Splits de dataset:



## CRISP-DM



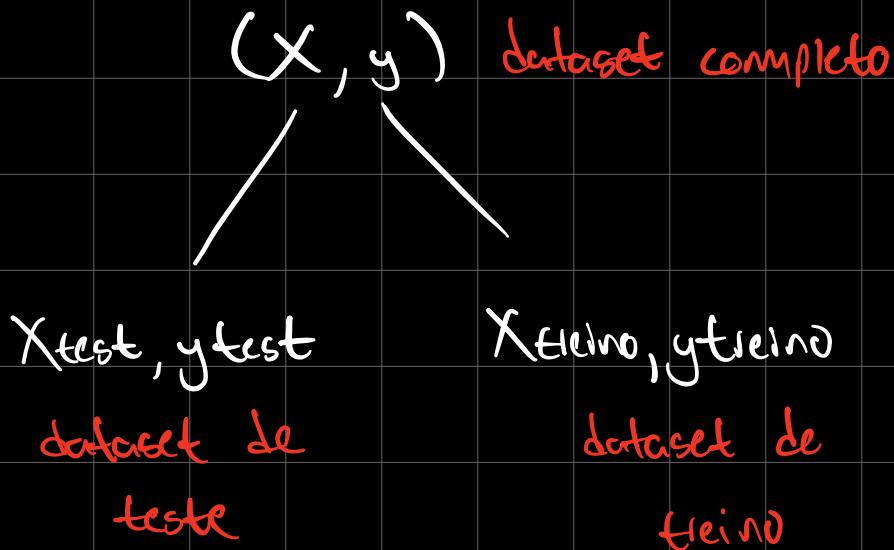
# Etapas do ciclo de processo de machine learning



## Etapa 1: Análise preliminar

- Foco na análise da natureza das features e do target isoladamente
  - prevenir "data snooping"
- Foco também na descoberta de erros grosseiros, outliers, anomalias (eg. saturação), buracos, etc.

## Etapa 2: Separação treino - teste



## Etapa 3: Análise pos-separação

- Usa apenas dataset de treino
- Foco em análise exploratória profunda, incluindo inter-relações

## Etapa 4: Nível 2 - escolha de modelo

- Design de pipeline de pré-process
- Escolha de melhor modelo
  - Inclui escolha de hiperparâmetro

- Estratégias:

- Separação treino-val / teste-val
- Validação cruzada

## Etapa 5: Nível 1 - certificação

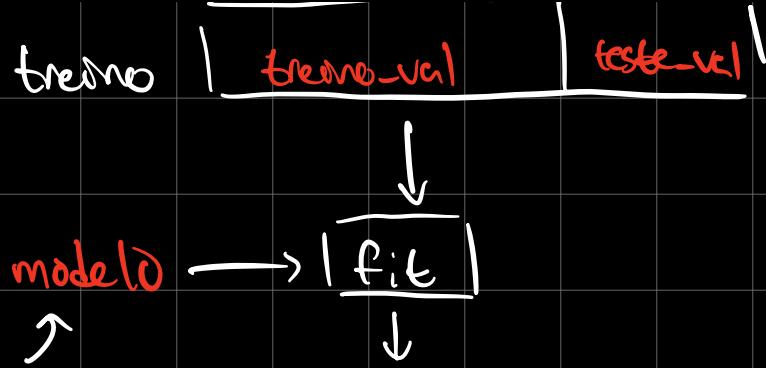
- Recebe o melhor modelo
- Treina no dataset de treino completo
- Avalia no dataset de teste

## Etapa 6: Nível 0 - deploy

- Recebe o melhor modelo, já certificado
- Treina no dataset completo
- Envia o modelo treinado e passa a responsabilidade para o time de MLOPS / DevOps

## Validação cruzada:

- Na estratégia: treino-val | teste-val



pipeline:

incluir: transformadores  
e modelo

modelo  
treinado

- Nova estratégia: validação cruzada

treino: 1 2 3 4 5 CV = 5

round	"treino-val"	"teste-val"
1	2 3 4 5	1
2	1 3 4 5	2
3	1 2 4 5	3
4	1 2 3 5	4
5	1 2 3 4	5

16:07 - 27/02/2025

PARA DE  
SER MALA!

Ayer!

Para cada modelo:

Para cada round:

treino

aulia



ideia

		predita	
		N	P
Real	N	+TN	FP
	P	FN	+P

acuracia:  $\frac{+P + TN}{+P + TN + FP + FN}$

precision:  $\frac{+P}{+P + FP}$

recall:  $\frac{+P}{+P + FN}$

$F_1$ :  $\frac{1}{F_1} = \frac{1}{2} \left( \frac{1}{\text{precision}} + \frac{1}{\text{recall}} \right)$

$$\Rightarrow F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- Sensibilidad (sensitivity, True-positive fraction - tPF)

- Especificidad (specificity, 1 - FPF)
 

False positive fraction

$$\text{Sensibilidad} = \text{recall} = \frac{TP}{TP + FN}$$

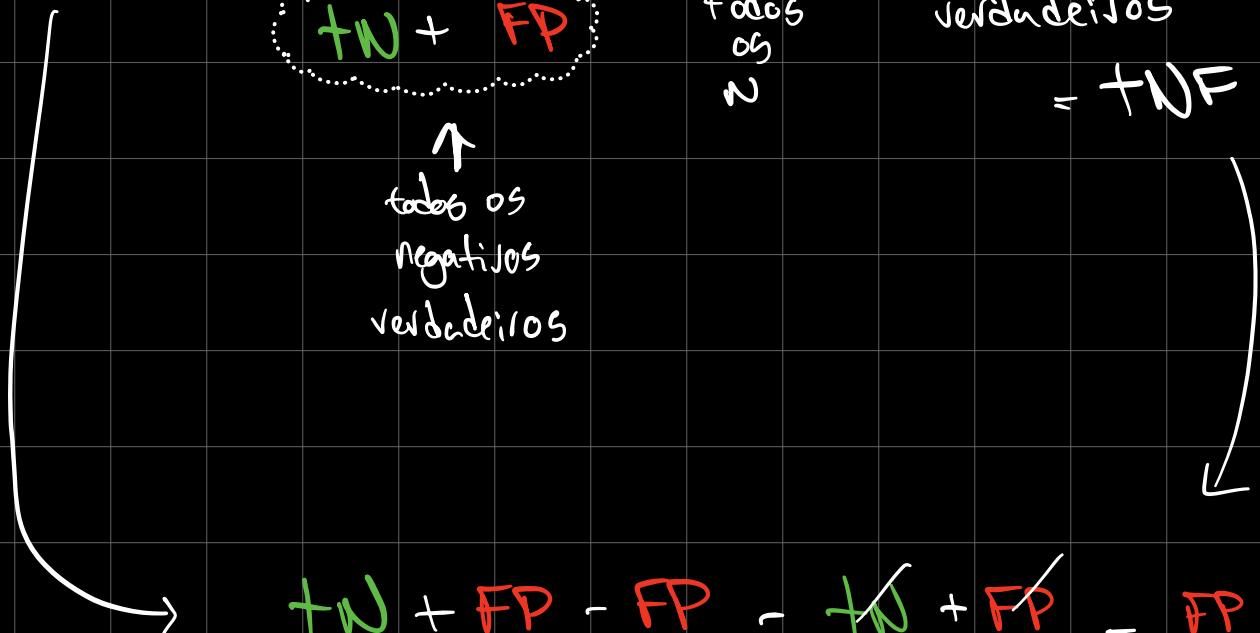
$$\text{Especificidad} = \text{recall} = \frac{TF}{TF + FP}$$

dos negativos

Área medice especificidad sensibilidad	busca info / datos precision recall
--	---

$$\text{Sensibilidade} = \frac{+P}{(+P + FN)} = \frac{+P}{\underset{\substack{\uparrow \\ \text{todos os} \\ \text{positives} \\ \text{seais}}}{(+P + FN)}} = \frac{+P}{\substack{\text{todos os} \\ P}} = \frac{+P}{\substack{\text{positivos verdadeiros}}} = +PF$$

$$\text{especificidade} = \frac{+N}{(+N + FP)} = \frac{+N}{\underset{\substack{\uparrow \\ \text{todos os} \\ \text{negativos} \\ \text{verdadeiros}}}{(+N + FP)}} = \frac{+N}{\substack{+ todos os \\ N}} = \frac{+N}{\substack{\text{negativos} \\ \text{verdadeiros}}} = +NF$$



$$\frac{+N + FP - FP}{+N + FP} = \frac{\cancel{+N} + \cancel{FP}}{\cancel{+N} + \cancel{FP}} - \frac{FP}{+N + FP}$$

$$= 1 - FPF$$

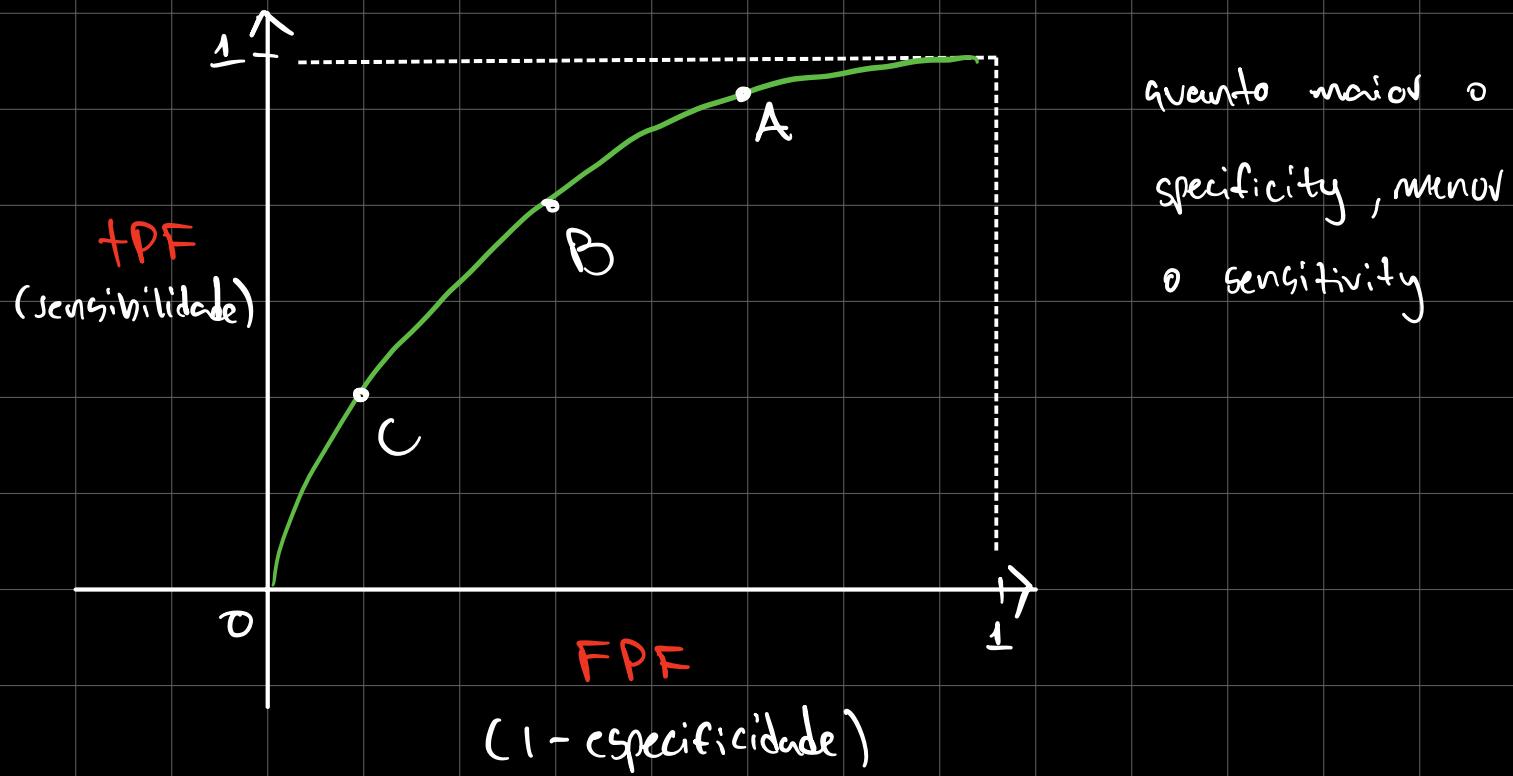
Sensibilidade  $\times$  ( $1 -$  especificidade)

TPF

$\times$

FPF

Receiver - Operating - Characteristic - Curve (curva ROC)



A: alta sensibilidade, baixa especificidade

exemplo: teste de covid (valorizando detecção dos positivos)

B: sensibilidade e especificidade balanceados

exemplo: wiog tem mesmo custo

$\Rightarrow$  valorizada acuracia

C: Alta especificidade, baixa sensibilidade

exemplo: julgamento criminal

# Modelo Linear

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

Podemos considerar que, no modelo linear, existe uma "feature virtual"  $x_0$  valendo 1, e associado ao peso  $\theta_0$

$$\hat{y} = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 \dots \theta_n x_n$$

$\uparrow$   
1

## O problema da colinearidade

Dizemos que existe colinearidade entre as features se podemos construir uma feature como combinacão linear das outras.

Exemplos:

a) features:  $x_0$ : feature virtual valendo 1

$\text{peso-kg}$ : peso em kg

$\text{peso-lb}$ : peso em lb

tem combinação linear pois

$$\text{peso\_kg} = 0.454 \text{ peso\_lb}$$

b) features:  $x_0$ : feature virtual valendo 1

C: temp em  $^{\circ}\text{C}$

F: temp em  $^{\circ}\text{F}$

tem combinação linear pois

$$F = \frac{9}{5} C + 32 x_0$$

e porque isso é um problema?

Observe que os seguintes modelos lineares para o exemplo (a) são equivalentes (não inventados)

i)  $\hat{y} = 14.3 + 5 \text{ peso\_kg} + 0 \text{ peso\_lb}$

ii)  $\hat{y} = 14.3 + 0 \text{ peso\_kg} + 5 \cdot 0.454 \text{ peso\_lb}$

iii)  $\hat{y} = 14.3 + 5 \alpha \text{ peso\_kg} + 5 \cdot (1-\alpha) \cdot 0.454 \text{ peso\_lb}$

$\Rightarrow$  temos infinitos modelos idênticos

$\Rightarrow$  Algoritmo de treino não converge!

## Regularização

A regularização é a inimiga do overfitting

modelo: conjunto de funções preditoras

exemplo: modelo linear da forma:

$$\hat{y} = \theta_0 + \theta_1 x_1$$

$$\hat{y} = 7 + 3x_1$$

$$\hat{y} = 0 + 2x_1$$

$$\hat{y} = 4 + 0x_1$$

$$\hat{y} = 9 + (-4)x_1$$

etc. . .

treinar o modelo: escolher a função preditora que melhor se ajusta aos dados de treinamento.

Exemplo:

model 0

$$\hat{y} = \theta_0 + \theta_1 x_1$$

"reiniciar o modelo"

problema de optimização

$\theta_0$  ótimo

$\theta_1$  ótimo

dados de treinamento

$x_{train}, y_{train} \Rightarrow$

$x \quad y$

-	-
-	-
-	-

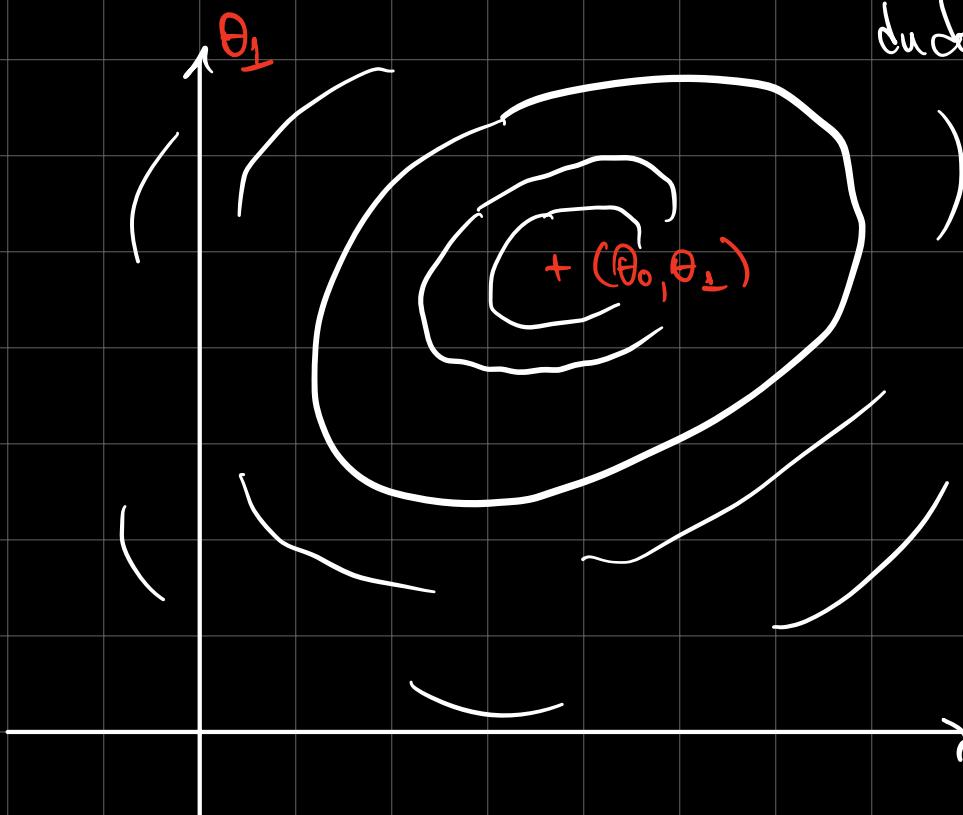
e.g. MSE

Overfitting: "sobreajuste"  $\rightarrow$  ajusta "em excesso" aos

dados de treinamento



"memorizou" os  
dados de treinamento  
com ruido e tudo



regularização  $\Leftrightarrow$  embutir crenças  
à priori

exemplo: uma moeda  $p$  = probabilidade de cara

Resultados	com crença anterior	sem crença anterior
Antes de tudo	$p = 0,5$	???
1 cara	$p = 0,5 + 10^{-8}$	$p = 1$
50 caras	$p = 0,7$	$p = 1$ (com força)

estimando probabilidade de cara

1. Sem crença anterior

$$P = \frac{\# \text{caras}}{\# \text{lancamentos}}$$

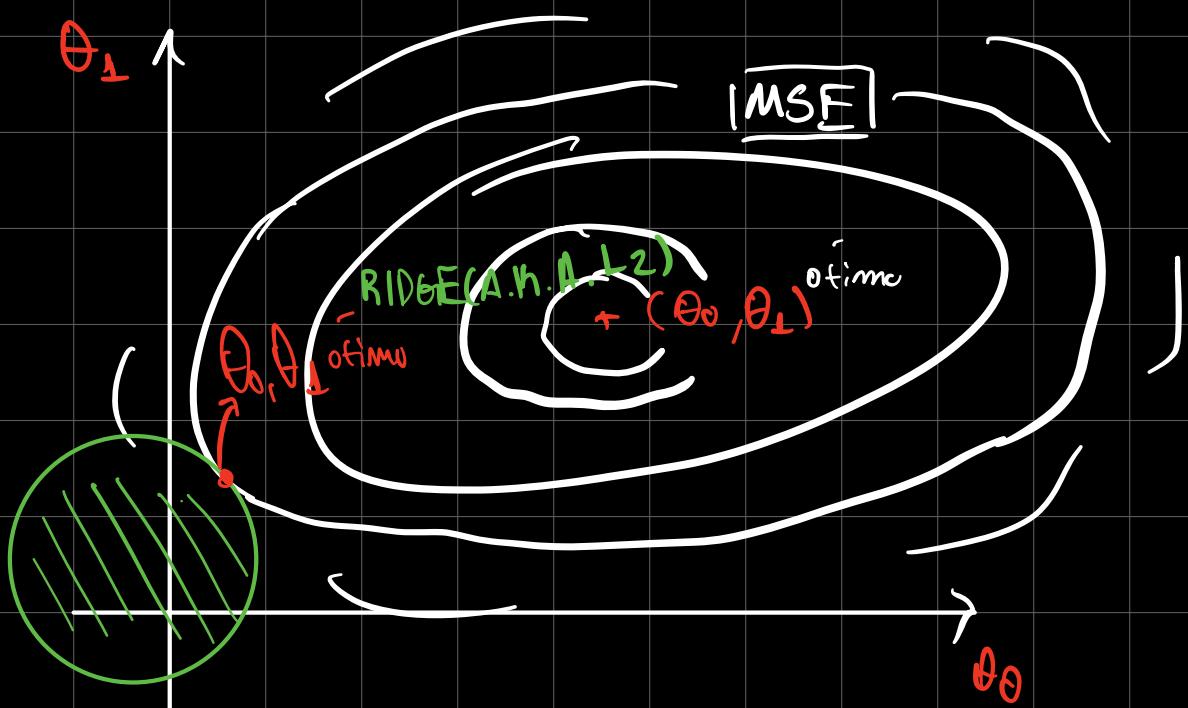
2. Com crença anterior

$$P = \frac{\# \text{caras} + \alpha}{\# \text{lancamentos} + 2\alpha}$$

hiperparâmetro  $\alpha \neq 0$

# Regularização em modelos de machine learning

⇒ Restringir o espaço de parâmetros



## Regularização RIDGE

1. Sem regularização

$$\vec{\theta}_0^{\text{optimo}} = \underset{\vec{\theta}}{\operatorname{arg\,min}} \text{MOE}(\vec{\theta}_0, \text{conjunto de treino})$$

2. com regularização

$$\vec{\theta}_0^{\text{RIDGE}} = \underset{\vec{\theta}}{\operatorname{argmin}} \text{MSE}(\vec{\theta}, \text{conjunto de treino})$$

Sujeito A:

$$\|\vec{\theta}\|_2^2 \leq B \quad \begin{matrix} \curvearrowleft \\ \downarrow \\ \text{nórm. L}_2 \end{matrix} \quad \text{hiperparâmetro: quadrado do raio da área de busca.}$$

Otima formulacão do RIDGE:

$$\vec{\theta}_{\text{RIDGE}} = \underset{\vec{\theta}}{\operatorname{argmin}} \text{MSE}(\vec{\theta}, \text{conjunto de treino})$$

$$\text{sujeito a: } \|\vec{\theta}\|_2^2 \leq B$$

$\uparrow \downarrow$  multiplicador de Lagrange

$$\vec{\theta}_{\text{RIDGE}} = \underset{\vec{\theta}}{\operatorname{argmin}} \text{MSE}(\vec{\theta}, \text{conjunto de treino}) + \alpha \|\vec{\theta}\|_2^2$$

hiperparâmetro, tem sentido inverso  
em relação a  $B$

Vantagens do RIDGE:

- combate overfitting

grande vantagem do RIDGE

- combate colinearidade

COMMUNICATE  
WINTER TERM

Exemplo:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

O que acontece se, por acidente,  $x_1 = x_2$ ?

1. Sem RIDGE: problemas de convergência

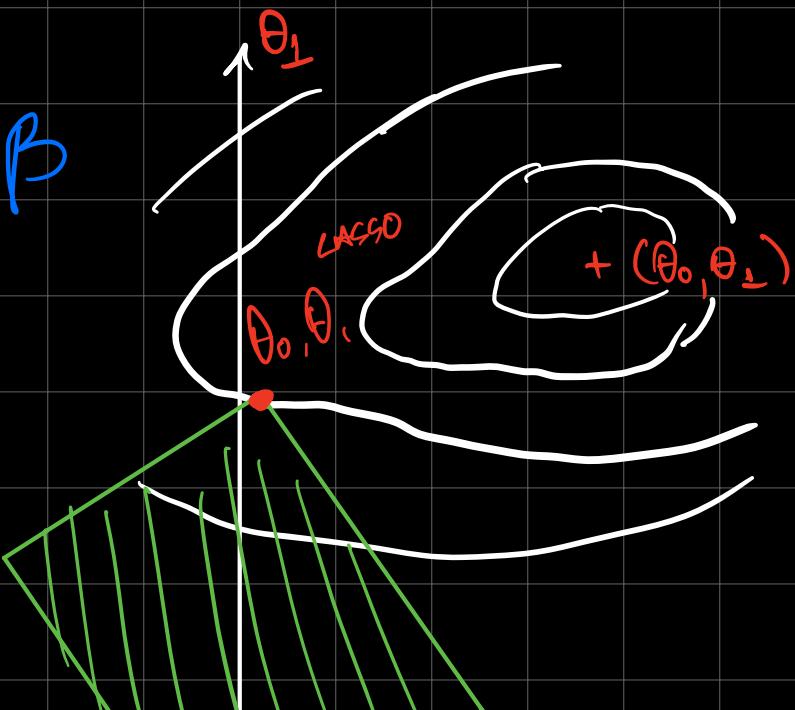
2. Com RIDGE: converge para  $\theta_1 = \theta_2$

$\Rightarrow$  sem problemas de convergência.

Regularização LASSO (least absolute shrinkage selection operator)

$$\vec{\theta}_{\text{LASSO}} = \underset{\vec{\theta}}{\operatorname{arg\,min}} \text{MSE}(\vec{\theta}, \text{conjunto de treino})$$

$$\text{sujeito a } \sum_{i=0}^n |\theta_i| \leq \beta$$



$\theta_0$

'raizão mística':

$$\vec{\theta} = ((\vec{X} \cdot \vec{X} + \alpha \vec{I})^{-1} \vec{X}^T \vec{y})$$

O L2 se soma os quadrados dos valores dos pesos

Não se tem opções

igualmente válidas, tem uma que é menor

raizão intuitiva: força  
balanceamento dos pesos

resolve colinearidade

= colinearidade  
incerteza dos pesos

com L2  
ele não  
é tão livre para  
jogar os pesos  
onde quiser

## RIDGE

$$\vec{L}(\vec{\theta}) = \text{MSE}(\vec{\theta}) + \alpha (\text{Norma}_{L2} \text{ de } \vec{\theta})$$

seleção  
mística:

???

seleção automática de  
features

'raizão intuitiva':  
é o fato de que a região

permisível  
do lasso

é uma  
bola  
pontuda

## LAGSO

$$\vec{L}(\vec{\theta}) = \text{MSE}(\vec{\theta}) + \alpha (\text{Norma}_{L1} \text{ de } \vec{\theta})$$

\* sendo assim,

uma foi gerada

e por conta disso

as curvas de nível

evoluem de forma que interseccionam

a região valida em uma curva, \*

tem vantagens dos dois modelos

é dc colinearidade e tem varias features

## Elastic Net

$$L(\vec{\theta}) = \text{MSE}(\vec{\theta}) + \alpha \cdot \Gamma (\text{Norma})$$

Norma  $\ell_p$  de vetor:

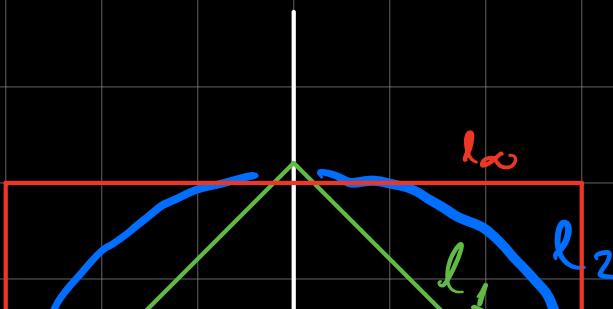
$$\ell_p(\vec{\theta}) = \left( \sum_i |\theta_i|^p \right)^{\frac{1}{p}}$$

Geometricamente

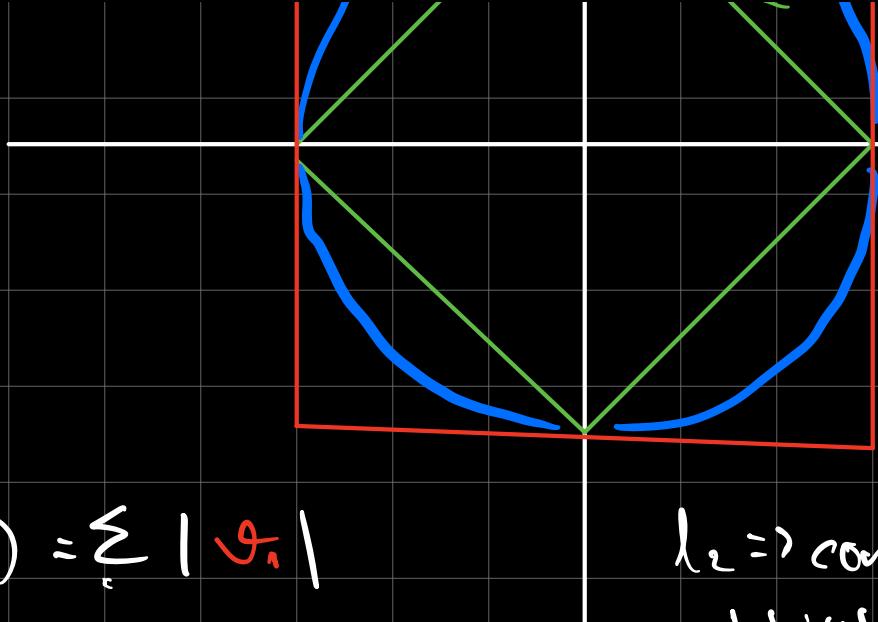
$\ell_1$  e  $\ell_2$  são os para devolver

$\ell_\infty$  é ideia de medir as coisas

em norma  $\infty$  utilizada pelo



pessoal de  
controle



$l_1 \Rightarrow$  eliminar os  
coeficientes

$$l_1(\vec{\vartheta}) = \sum_i |\vartheta_i|$$

$l_2 \Rightarrow$  combate colinearidade  
estabilidade numérica

$$l_2(\vec{\vartheta}) = \sqrt{\sum_i \vartheta_i^2}$$

$l_\infty \Rightarrow$  garantia de extensão  
máxima

$$l_\infty(\vec{\vartheta}) = \max_i |\vartheta_i|$$

//

## Régressão Logística

em primeiro lugar, regressão logística não é regressão é  
classificação

problema: quanto estudar?

faseon = 1      não passou = 0



(x) = horas de estudo



- interpola não pode ser ligar os pontos nem regressão, não faz sentido

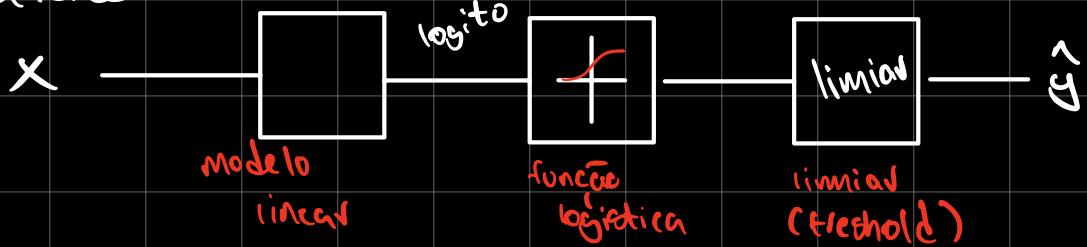
interpolacão binária (0, 1) e suave de transição.

- quero recall ou precision?

precision: baseado em horas de estudo quero saber se vai passar ou não.

Regressão logística:

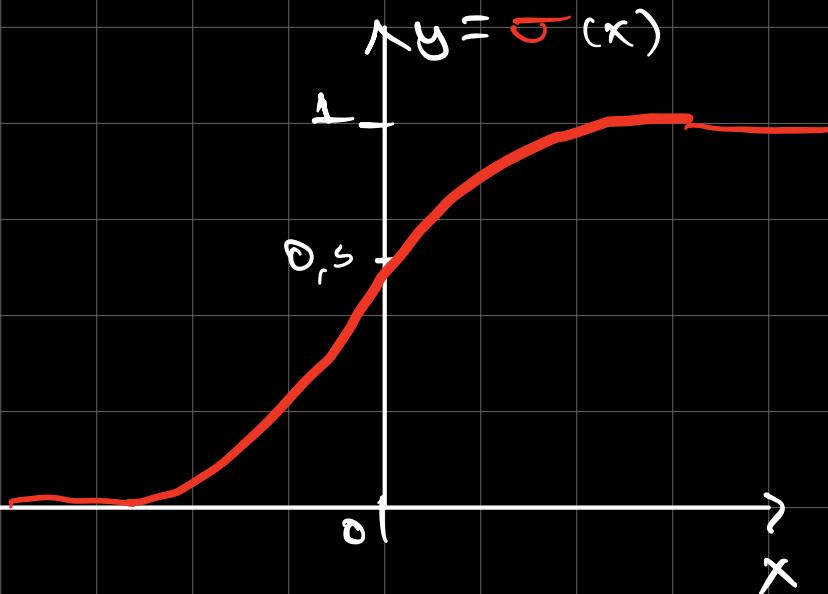
features



$$\text{"logito"} = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

Função logística:

$$y = \sigma(x) = \frac{1}{1 + \exp(-x)}$$



Na regressão logística

$$1. \hat{p} = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n)$$

$$2. \hat{y} = \begin{cases} 0 & \text{se } \hat{p} < T \\ 1 & \text{se } \hat{p} > T \end{cases}$$

# O problema da otimização

Máximo verossimilhança

"qual a chance de observar meus dados para um certo conjunto de parâmetros do modelo?"

$$\text{prob}(\text{Dados} \mid \text{parâmetros})$$

$$L(\vec{\theta}) = \text{prob}(D \mid \vec{\theta})$$

Na regressão logística:

Dados:  $m$  exemplos  $(x_i, g_i)$  iid

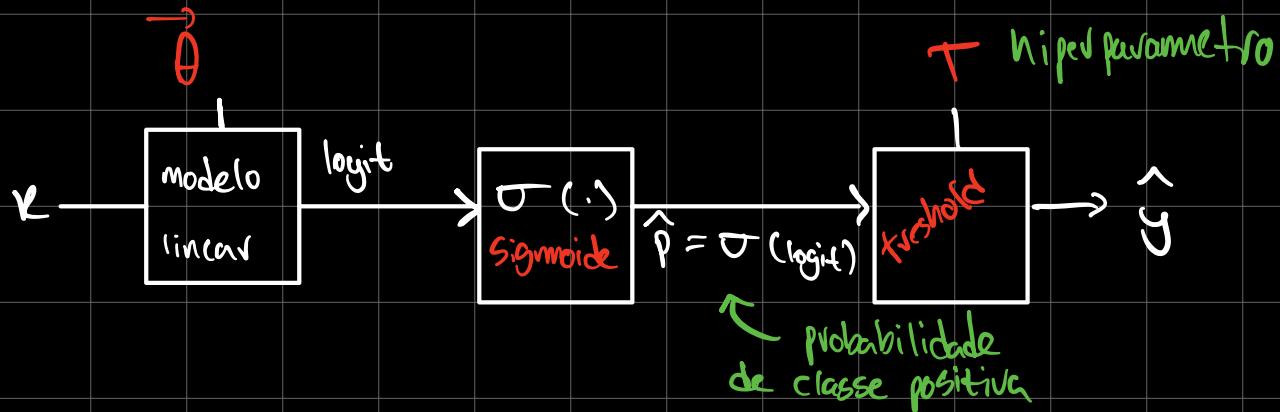
↑  
features      ↗ targets

Modelos:  $\hat{p}_i = \sigma(\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n)$

likelihood

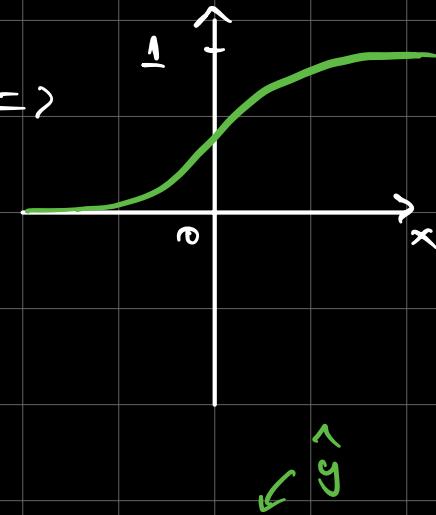
$$L(\vec{\theta}) = (\prod_{y_i=0} (1 - \hat{p}_i)) (\prod_{y_i=1} \hat{p}_i)$$

# Regressão Logística



$$\text{logit} = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

$$\text{Sigmoid: } g = \sigma(x) = \frac{1}{1 + e^{-x}} \Rightarrow$$



Exemplo: usar reg. log p/ prever ocorrência de chuva de tarde em função de:

- Temp 11h da manhã  $x_1 [^{\circ}\text{C}]$
- Umidade 11h da manhã  $x_2 [\%]$
- Cobertura de nuvens 11h  $x_3 [\cdot]$

$$\hat{p} = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3)$$

## dataset de treino:

$x_1$	$x_2$	$x_3$	$y$
30	87	72	1 $\leftarrow (x^{(1)}, y^{(1)}) \leftarrow \hat{p}^{(1)} = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3)$
12	90	100	1 $\leftarrow (x^{(2)}, y^{(2)}) \leftarrow \hat{p}^{(2)} = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3)$
24	43	5	0 $\leftarrow (x^{(3)}, y^{(3)})$
:	:	:	:
22	70	68	0 $\leftarrow (x^{(m)}, y^{(m)})$

Otimização do modelo por máxima verossimilhança

$$\vec{\theta}^{\text{ótimo}} = \arg \max_{\vec{\theta}} \log [\underbrace{\text{Prob} (\text{dataset treino} | \vec{\theta})}_{\text{depende de } \vec{\theta}}]$$

$$\begin{aligned} \log [\text{Prob} (\text{dataset treino} | \vec{\theta})] \\ = \sum_{i=1}^m [(1 - y_i) \log (1 - \hat{p}_i) + y_i \log (\hat{p}_i)] \end{aligned}$$

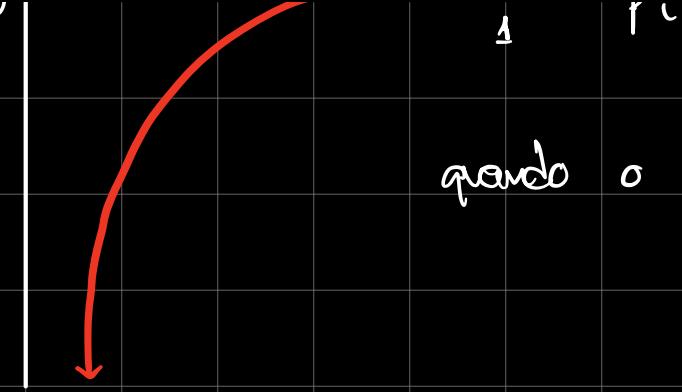
caso 1:  $y_i = 1$  (dia de chuva)

O termo da soma fica:

$$(1 - 1) \log (1 - \hat{p}_i) + 1 \log (\hat{p}_i) = \log (\hat{p}_i)$$

$$\log (\hat{p}_i)$$

$$0 \rightarrow \hat{p}_i$$

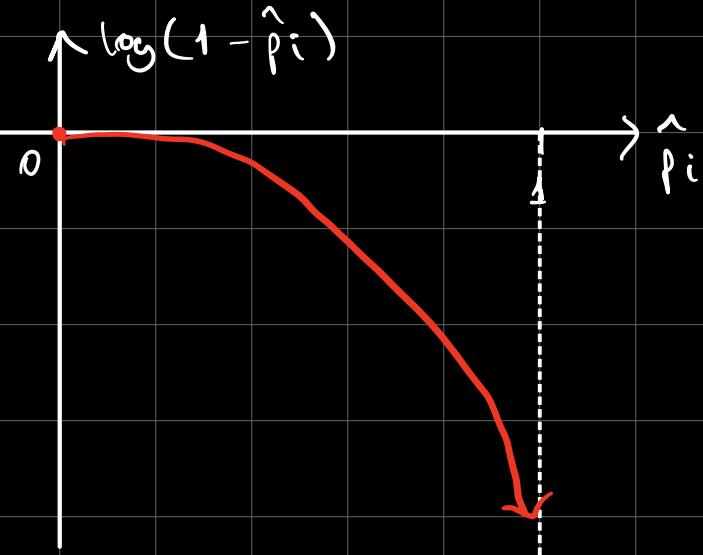


quando o  $\hat{y}_i$  é 1

Caso 2:  $y_i = 0$

O termo da soma fica:

$$(1-0) \log(1-\hat{p}_i) + 0 \log(\hat{p}_i) = \log(1-\hat{p}_i)$$



Otimização por minimização da entropia cruzada

Entropia de uma V.A. discreta:

$X = \{ \text{Valor 1, Valor 2, ..., Valor } n \}$

$p_i = \text{Prob}[X = \text{Valor } i]$

Exemplo: Saquinho com bolinhas coloridas

índice	bolinha	quantidade	$p_i$
--------	---------	------------	-------

1	•	160
2	•	80
3	•	10

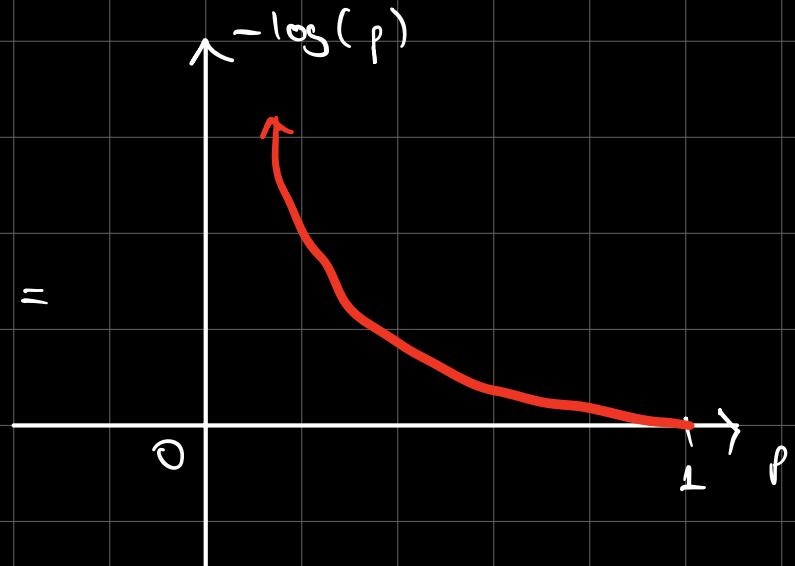
0.8

0.15

0.05

informação = surpresa

$$-\log(\text{prob}) =$$

 $-\log(p)$ 

Valor médio da informação

para  $N$  observações, temos:

# valor 1 =  $n_1$  de vezes em que deu valor 1

⋮

# valor  $N$  =  $n_N$  de vezes em que deu valor  $N$ .

$$\begin{aligned} \text{Valor} \\ \text{médio} = \frac{1}{N} & \left[ (\# \text{valor } 1) \cdot (-\log p_1) \right. \\ & + (\# \text{valor } 2) \cdot (-\log p_2) \\ & + \dots \\ & \left. + (\# \text{valor } N) \cdot (-\log p_N) \right] \end{aligned}$$

No limite:  $\text{Valor médio} = p_1 \log p_1 + p_2 \log p_2 + \dots + p_n \log p_n$

Entropia = Valor médio informação de uma fonte

$$H_x = - \sum_{i=1}^n p_i \log p_i$$

Exemplo:

$$H_{\text{soquinho}} = - (0.8 \log 0.8 + 0.15 \log 0.15 + 0.05 \log 0.05)$$

• • •

Entropia Cruzada:

$$H_x = - \sum_{i=1}^n p_i^{(x)} \log p_i^{(y)}$$

mas e o machine learning?

chance da  
instância amostra  
ser da classe k

$$L = \frac{1}{m} \sum_{i=1}^m \left[ - \sum_{k=1}^c \underbrace{\left[ y_i == k \right]}_{\substack{\text{colchetes (brackets)} \\ \text{de inversão}}} \cdot \log(p_i^{(k)}) \right]$$

$\overrightarrow{\text{amostras}}$

$c \leftarrow$  número de classes

por exemplo: classificação binária

$$L = \frac{1}{m} \sum_{i=1}^m \left[ -[y_i = 0] \log(\hat{p}_i^{(0)}) + [y_i = 1] \log(\hat{p}_i^{(1)}) \right]$$

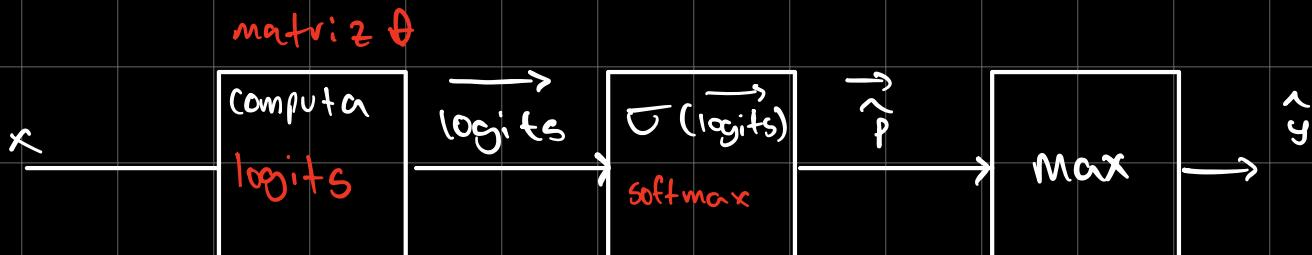
$$L = -\frac{1}{m} \sum_{i=1}^m \left[ (1-y_i) \log(1-\hat{p}_i) + y_i \log(\hat{p}_i) \right]$$

$$L = -\frac{1}{m} \text{Prob} \left[ \begin{smallmatrix} \text{dataset} \\ \text{treino} \end{smallmatrix} \mid \vec{\theta} \right]$$

Conclusão:

maximizar  $\Leftrightarrow$  minimizar perda  
verosimilhança de entropia cruzada.

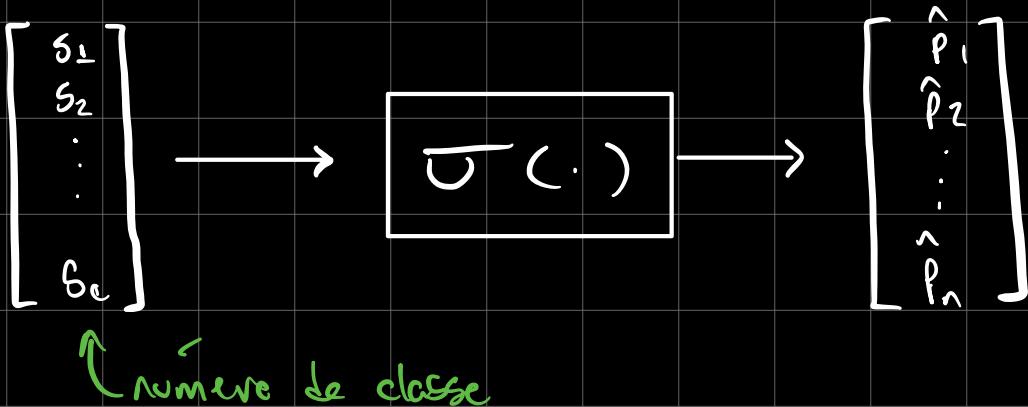
## Regressão logística multiclasse



para cada classe:

$$\text{logits} = \theta_{0,n} + \theta_{1,n} l_1 + \dots + \theta_{n,n}$$

# Softmax

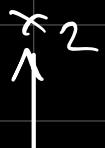


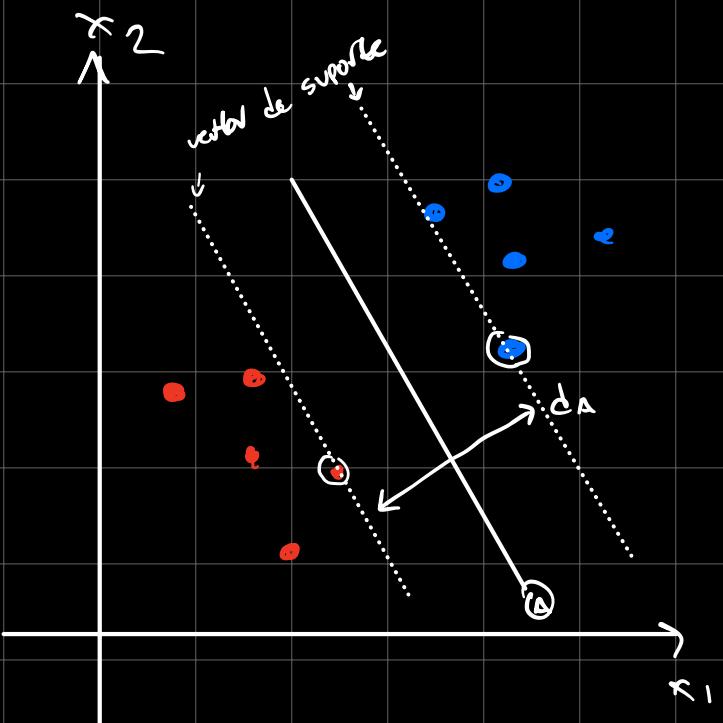
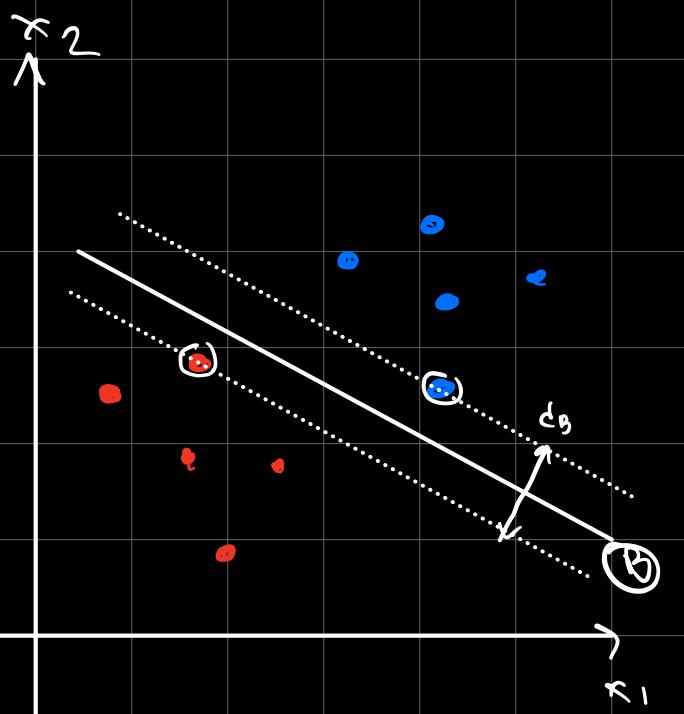
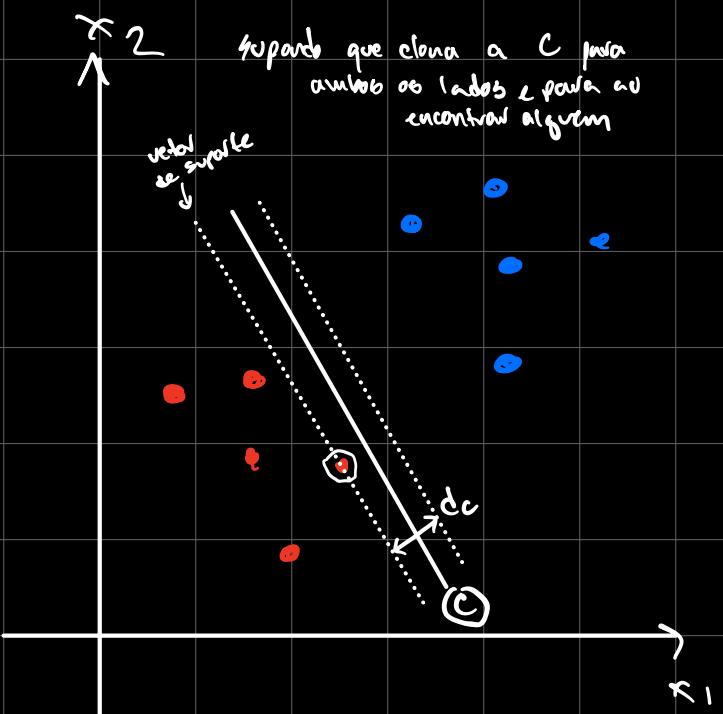
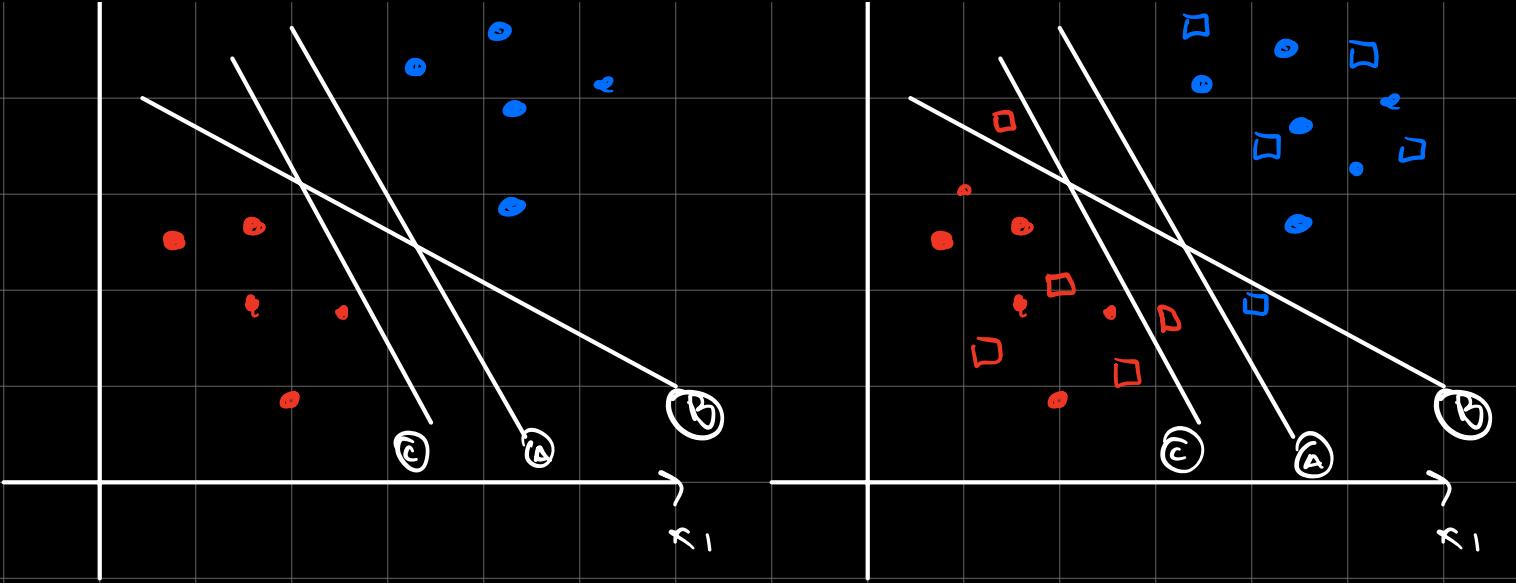
$$\hat{p}_i^* = \frac{e^{s_i}}{\sum_{k=1}^c e^{s_k}}$$

$$\sum_{i=1}^c \hat{p}_i^* = \sum_{i=0}^c \left[ \frac{e^{s_i}}{\left( \sum_{k=1}^c e^{s_k} \right)} \right] = \frac{1}{\left( \sum_{k=1}^c e^{s_k} \right)} \sum_{i=1}^c e^{s_i} = 1$$

# Support vector machine

- Máquina de vetores de suporte
- Ideia: escolher uma fronteira de separação que maximizem a separação entre classes.





SVM, definição 1: (intuitiva)

Achar linha separadora que maximizem a "largura da avenida".

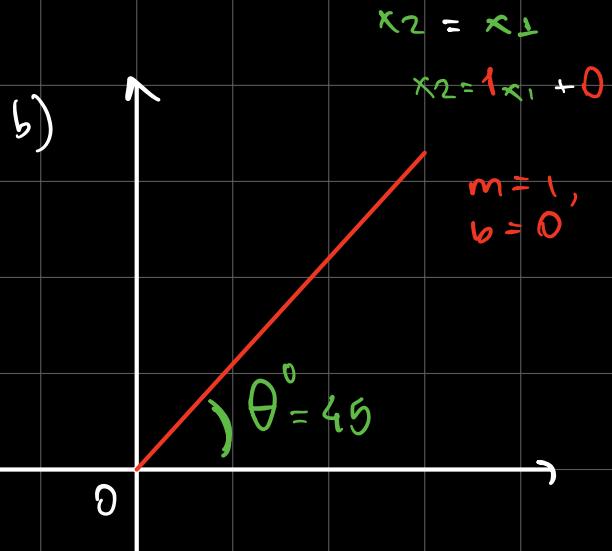


Definindo a linha separadora:

Tentativa 1:

$$x_2 = mx_1 + b$$

a)



c)



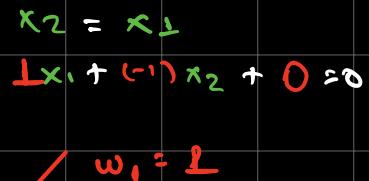
Tentativa 2:  $x_1$  e  $x_2$  tem o mesmo status:

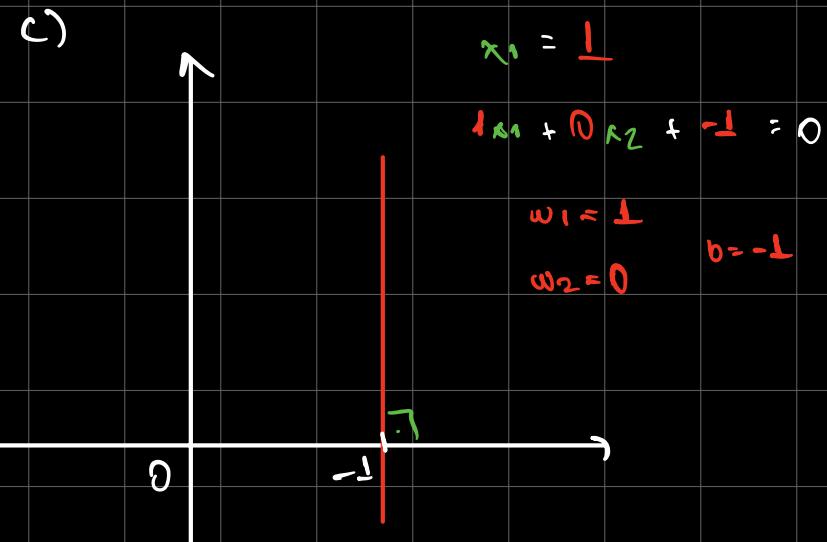
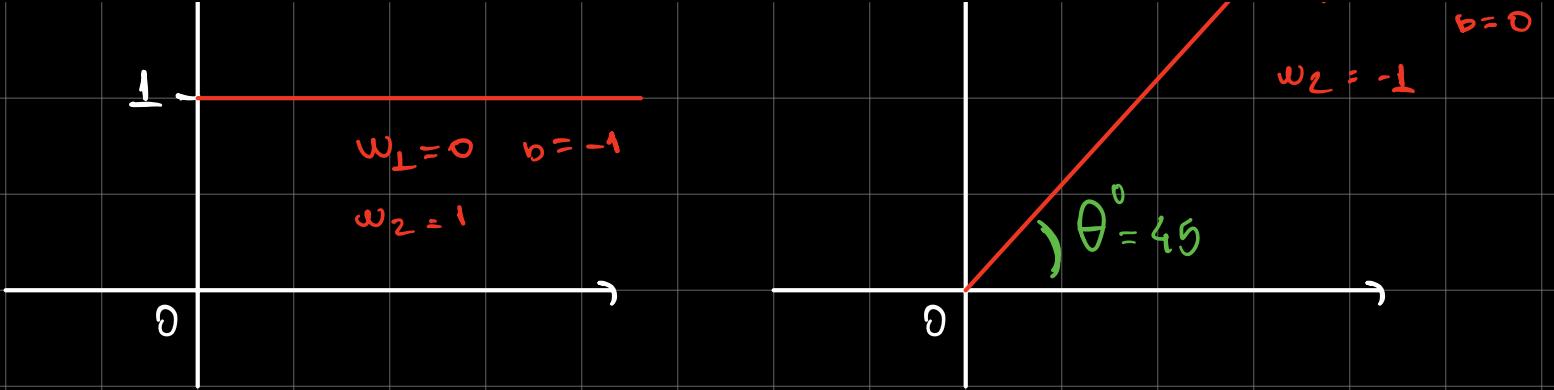
$$\omega_1 x_1 + \omega_2 x_2 + b = 0$$

a)

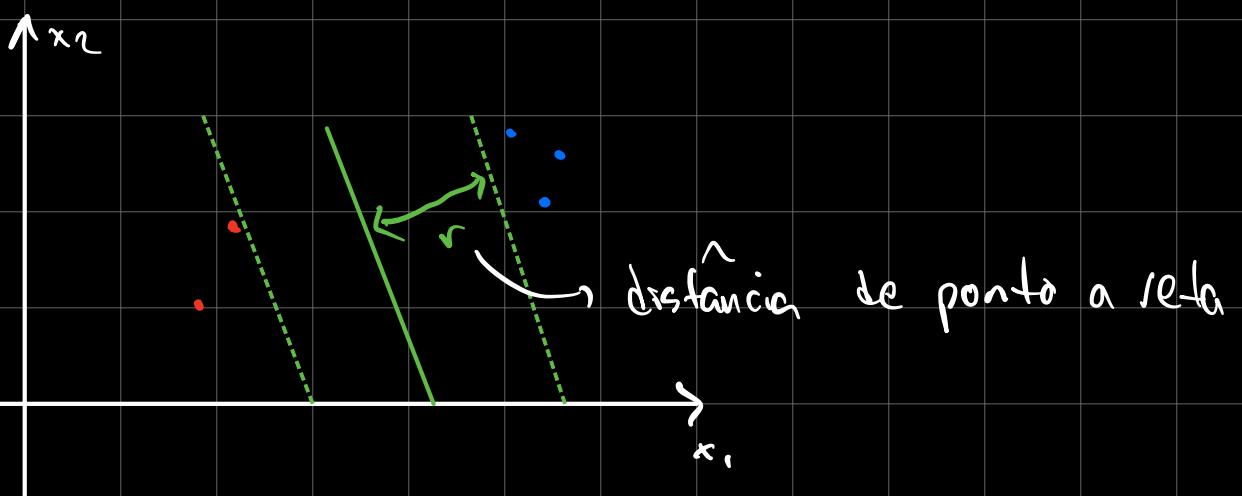


b)





Definindo "largura da avenida"



- Definindo

$$w_1x_1 + w_2x_2 + b = \begin{cases} \text{zero na linha média} \end{cases}$$

$\pm 1$  nas "calendas"

(motivo: ambiguidade)

$$\Rightarrow r^2 = \frac{1}{w_1^2 + w_2^2}$$

No caso geral ( $n$  features):  $r^2 = \frac{1}{w_1^2 + w_2^2 + \dots + w_n^2}$

Nota:  $w^T w = w_1^2 + \dots + w_n^2$  onde

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

SVM, definição 2:

Achar  $w_{opt}, b$  que maximiza  $\frac{1}{w^T w}$  sujeito a  
"todas foto da omissão"

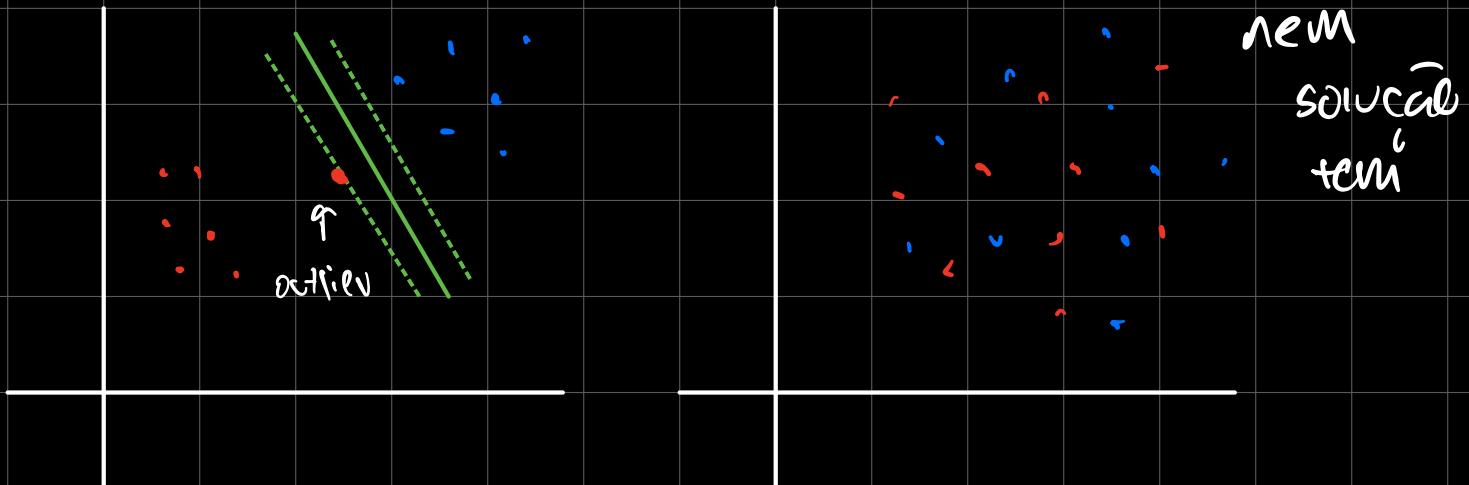
SVM, definição final: HARD SVM

$w^{óptimo}$ ,  $b^{óptimo}$

$$w^{óptimo} = \arg \min_w w^T w$$

$\text{sujeito a } y_i (\vec{w}^T \vec{x}_i + b) \stackrel{\text{óptimo}}{\geq} 1$   
 ↗  
 $\begin{cases} +1 \text{ si positivos} \\ -1 \text{ si negativos} \end{cases}$

## Problemas no Dantzig



SVM, definição realmente final: SOFT SVM

$$\vec{w}^{\text{optimo}} = \underset{\vec{w}}{\operatorname{arg\,min}} \left[ \vec{w}^T \vec{w} + C \left( \sum_{i=1}^m \xi_i \right) \right]$$

sujeito a:

$$y_i (\vec{w}^T \vec{x}_i + b) \geq 1 - \xi_i$$

Algoritmo: optimização quadrática.

# Complexidade computacional das SVM

1) Linear SVM:  $O(m, n)$

$m$ : de  
exemplos  
de treinamento

$n$ : de features

exemplo:

Se uma SVM linear leva 1 min para treinar com um dataset de  $m$  exemplos e  $n$  features, quanto tempo levará pt treinar com 1 dataset de  $10m$  exemplos e  $2n$  features?

Resposta: 20 minutos

2) SVM com kernels  $\rightarrow$  SVM não-linear

$O(m^2 n)$ : polinomial

$O(m^3 n)$ : radial-basis functions (RBF)

Exemplo: Nas mesmas condições do exemplo anterior, para uma SVM de kernel RBF, qual o tempo de execução?

Resposta:

$$m' = 10m$$

$$n' = 2n \Rightarrow O((m')^3 n') =$$

$$= O((10m)^3 (2n)) =$$

$$= O(1000 \cdot 2 \cdot m^3 n) =$$

$$\approx 2000 \cdot O(m^3 n)$$

Antes: 1 min  $\Rightarrow 33,3$  h

## ÁRVORE DE DECISÃO

dataset



- Escolhe uma feature
  - Escolhe um ponto de corte

dataset ↙

esq

↖ dataset

d.r

# algoritmo Árvore de decisão:

Árvore de decisão (dataset):

Se não dé para dividir (dataset):

nó = Nó()

nó.tipo = FOLHA

nó.resultado = <algum scs.aqui>

return nó

feature, threshold =

achar\_melhor\_feature\_threshold (dataset)

dataset 1, dataset 2 = /

divide\_dataset (dataset, feature, threshold)

nó\_esquerdo = Árvore de decisão (dataset 1)

nó\_direito = Árvore de decisão (dataset 2)

nó = Nó()

nó.esq = nó\_esquerdo

nó.dir = nó\_direito

return nó

# Unsupervised learning

- Redução de dimensionalidade
  - PCA
  - t-SNE
  - UMAP

- Clustering
  - K-means
  - mean Shift
  - Agglomerative clustering

Supervised

Foco: previsão

Mecanismo:

Aprender a prever

baseado nas features

( $X$  treino) e no valor

desejado (target)

Não tem resposta certa

vs

Unsupervised

Foco: análise exploratória

Mecanismo: feature engineering

Mecanismo: gerar de dados

(semi-supervisionado)

Mecanismo: descobrir "padrões" nas features

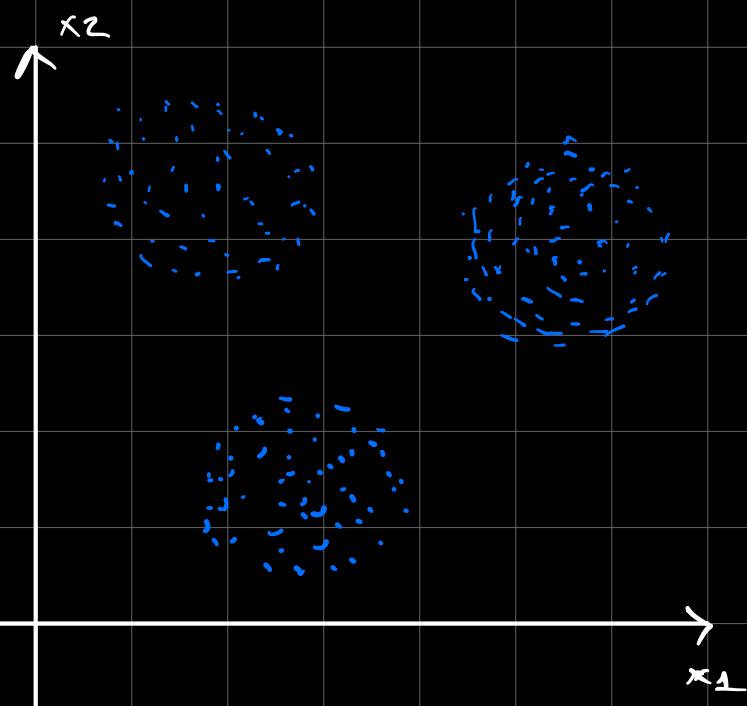
Não tem resposta certa

Aprendizado não-supervisionado:

- Clustering: associar os objetos à agrupamentos "naturais."

- Reduzir a dimensionalidade: encontrar uma representação dos dados com menos dimensões (features)

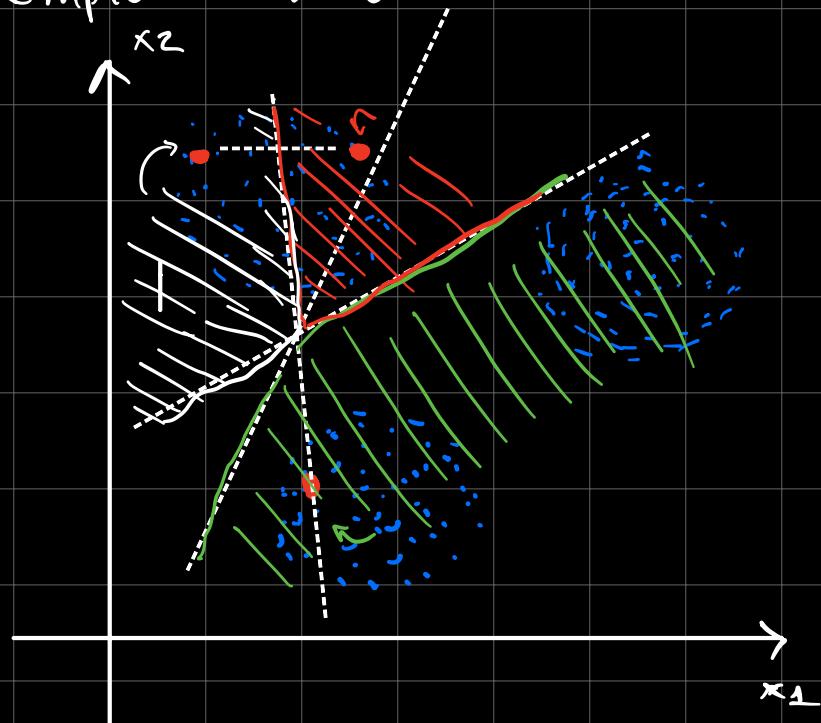
## k-means



1. escolha  $k = n$ - de clusters

2. escolha  $k$  centroide

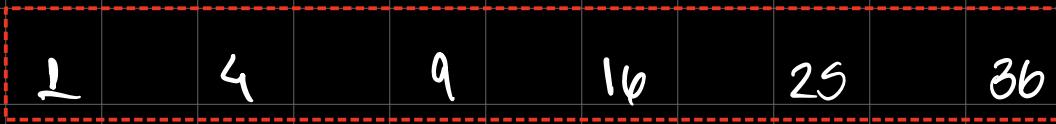
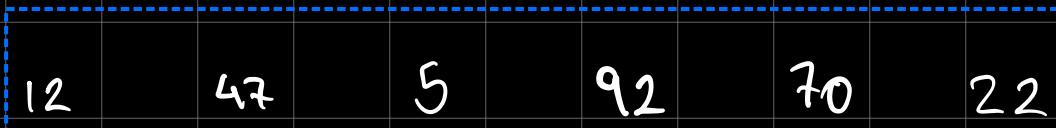
exemplo :  $k = 3$



enquanto não convergencia

- atribuir os pontos  
aos centroides, por  
aproximidade

# Enxugando a informação



Situação	Volume de dados teórico	Volume de informação prático
MNIST	784 bytes	1 dígito decimal ( $\approx$ meio byte)
a sequência: 1, 2, 3, 4, 5, 6, 7, 8, 9 10, 11, 12, 13, 14, 15 16, 17, 18, 19, 20... 4321	4320 números 2 bytes $= 8640 \text{ bytes}$	[ $k$ for $k$ in range(1, 4322) if $k \neq 16$ ] $\Rightarrow 38 \text{ bytes}$

## Benefícios

- Compactação
- Entendimento  
 $\hookrightarrow$  Orientação

# Redução de dimensionalidade

↓

$$x \in \mathbb{R}^n \rightarrow \boxed{\text{redução de dimensionalidade}} \rightarrow x \in \mathbb{R}^p \quad p \ll n$$

Motivação:

- Compactação:

↳ compressão de dados com perdas (lossy compression)

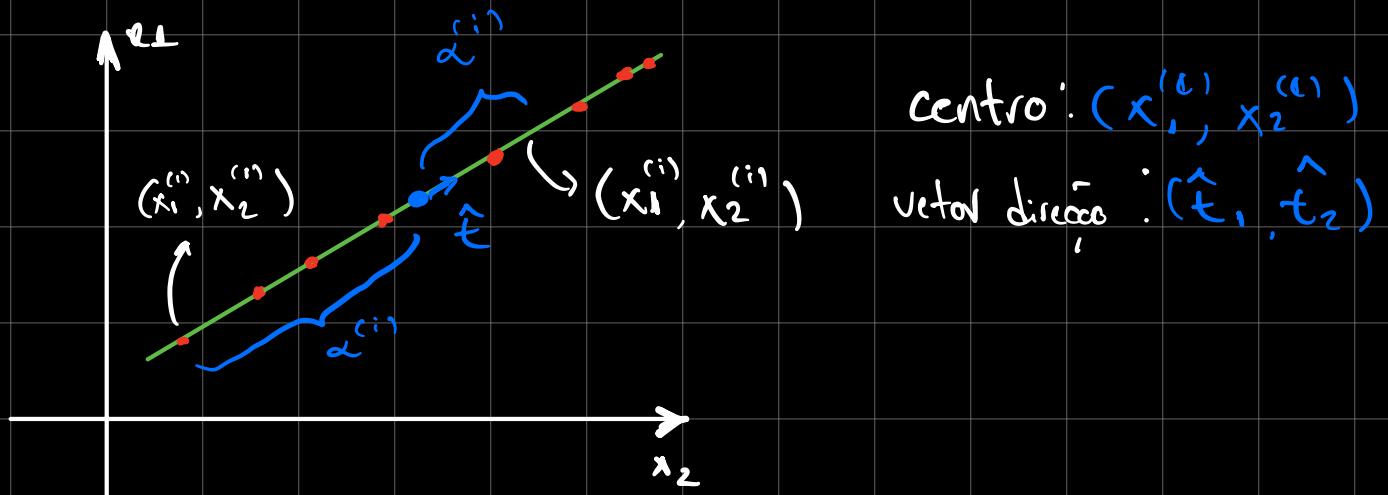
- Entendimento:

↳ descobrir a "essência" dos dados

## Algumas técnicas

- PCA
- t-SNE
- Autoencoders

## Principal Component Analysis (PCA)



Ponto

coords  
orig.

coords  
novo sistema

transf.  
antigo  $\rightarrow$  novo

trans  
novo  $\rightarrow$  antigo

$\perp$

$$(x_1^{(1)}, x_2^{(1)})$$

$$d^{(1)}$$

$\downarrow$   
projeção de  
 $(x_1^{(2)} - x_1^c)$  na  
direção  $\hat{e}$

$$d^{(1)} = (x_1^c - x_1^e) \cdot \hat{e}$$

$$x^2 = x + d^{(1)} \cdot \hat{e}$$

MEMORIZE!

$\perp$

$$(x_1^c, x_2^c)$$

$$(x_1^e, x_2^e)$$

$$\theta$$

$$(x_1^{(2)}, x_2^{(2)})$$

$$\vec{u}$$

$$\vec{e}$$

$$\vec{u} \cdot \vec{e} = u_1 e_1 + u_2 e_2$$

$$\|\vec{u}\| \|\vec{e}\| \cos \theta$$

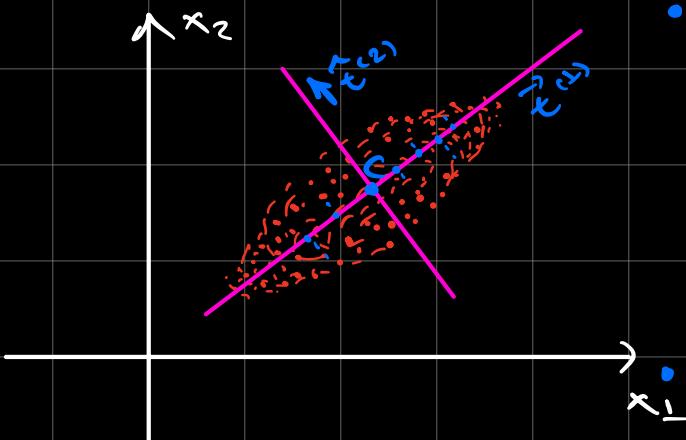
$$\vec{u} \cdot \hat{e} = u_1 \hat{e}_1 + u_2 \hat{e}_2$$

$$\|\vec{u}\| \cdot \|\hat{e}\| \cdot \cos \theta$$

$$\vec{u}(x_1^e, x_2^e) - (x_1^c, x_2^c)$$

$$\approx \|\vec{u}\| \cdot \cos \theta$$

Mais um exemplo:



$$x \rightarrow [Rowve x^c] \rightarrow \underline{x}$$

Porque?

! MÁGICA !

$$\begin{pmatrix} \underline{x} \\ + \\ \underline{x} \end{pmatrix}$$

Autovetores  
e  
Autovalores

$\hat{c}^{(i)}$ : autovetores normalizados

por ordem de autovalor decrescente

## REDES NEURAIS

O neurônio:



O neurônio

$$x_1 - w_1$$

$$x_2 - w_2$$

:

$$x_n - w_n$$

pesos

$b$   
constante

$$\boxed{\sigma(\cdot)} \rightarrow y$$

função  
de ativação

linear

Uma camada de rede:

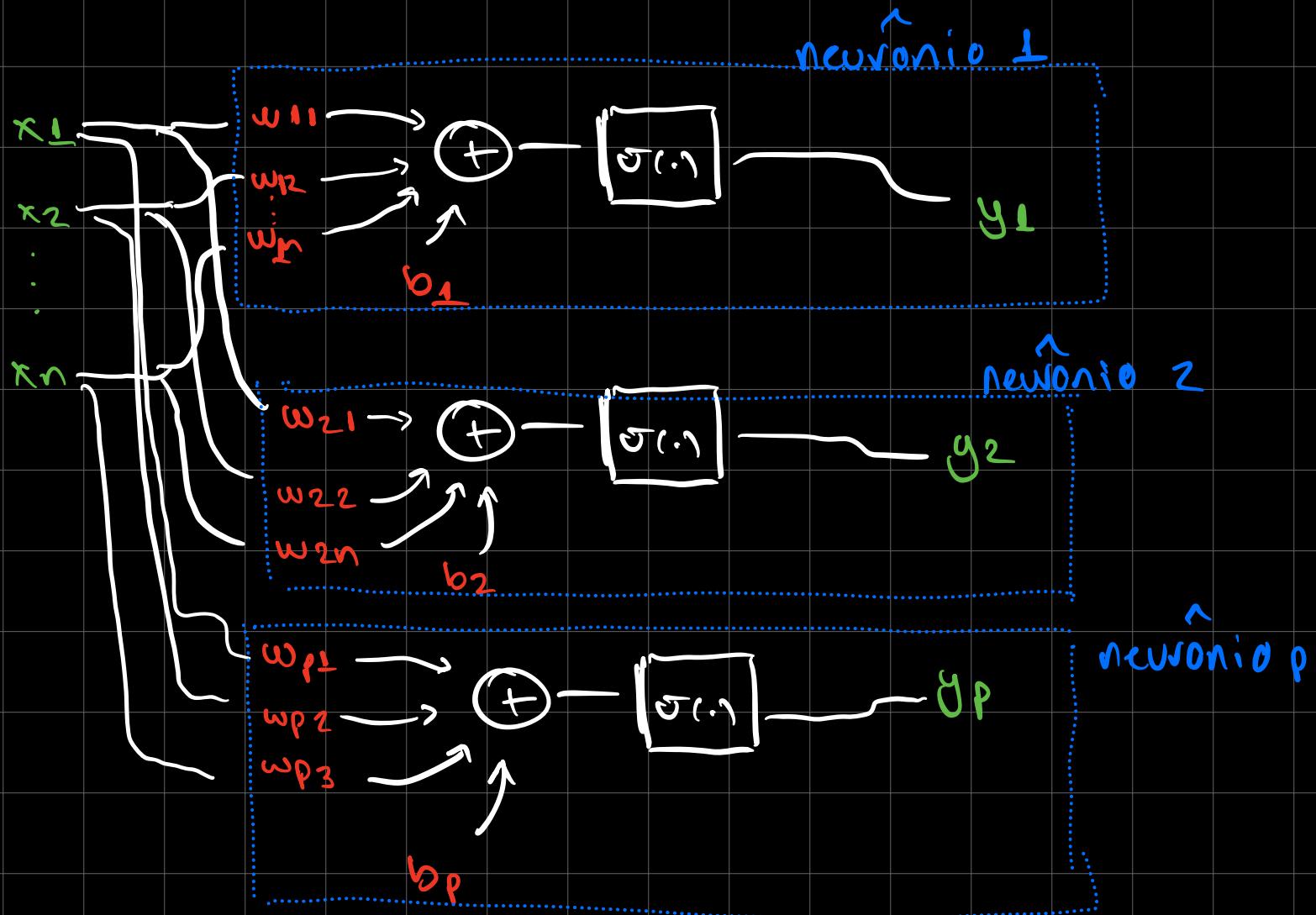


Diagramma mais simples



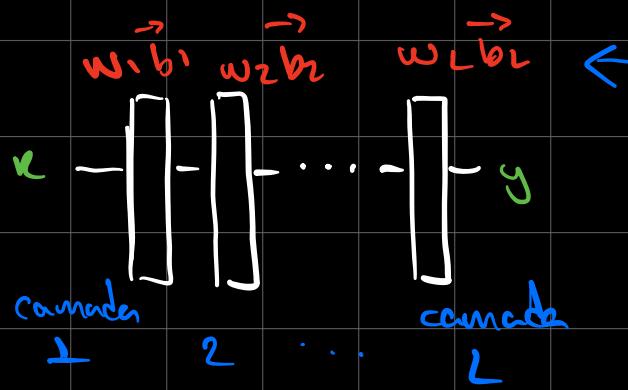
Parâmetros de chamada:

$$w = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{p1} & w_{p2} & \dots & w_{pn} \end{bmatrix}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{bmatrix}$$

Uma rede neural

multi-Layer Perception



O conjunto completo de parâmetros:

todas as matrizes de pesos e vetores das constantes

Exemplo: Numa MLP de 3 camadas temos o seguinte  
número de neurônios por camada:

camada

$L$

$n$ : neurônios

10

2                    5  
3                    1

A entrada tem 20 features, quantos parâmetros tem essa rede

1ª camada:

$$20 - \boxed{\quad} - 10 \Rightarrow w_1 \in \mathbb{R}^{10 \times 20} \Rightarrow 200 \text{ pesos}$$

$$b_1 \in \mathbb{R}^{10} \Rightarrow 10 \text{ ctes}$$

2ª camada

$$10 - \boxed{\quad} - 5 \Rightarrow w_2 \in \mathbb{R}^{5 \times 10} \Rightarrow 50 \text{ pesos}$$

$$b_2 \in \mathbb{R}^5 \Rightarrow 5 \text{ ctes}$$

3ª camada

$$5 - \boxed{\quad} - 1 \Rightarrow w_3 \in \mathbb{R}^{1 \times 5} \Rightarrow 5 \text{ pesos}$$

$$b_3 \in \mathbb{R}^1 \Rightarrow 1 \text{ cte}$$

271 parâmetros

Como as redes neurais rodam rápido?

LMNP:  $s = xw^T + b^T \rightarrow y = f(s)$

↳ por exemplo:  $f(\cdot)$



$B$  exemplos:  $X \in \mathbb{R}^{B \times n}$   
↑ batch

$Y \in \mathbb{R}^{B \times p}$

$$s = xw^T +$$

modelos lineares

- regressão:

- Simples

- Ridge

- Lasso

- Elastic Net
- Classificação: Regressão Logística

- Support vector machine
  - Hard SVM
  - Soft SVM

- Decision trees
- Modelos de ensemble
- Clustering
- Redução de dimensionalidade
- Introdução à redes neurais