Introdução à Redes Neurais

Dia 2: Tratamento e Pré-processamento dos Dados

Enzo L. Fernandes

Universidade Estadual Paulista – *Campus* Botucatu

14 de Dezembro de 2021

Conteúdos

Parte 1: Pré-Processamento

Por que fazer Pré-processamento?

Os dados correspondem à experiência **E** a partir da qual o modelo "aprende" a executar a tarefa **T**, por isso, de forma geral:

$$\text{Dados bons} \rightarrow \text{Resultados bons}$$

As vezes os dados não possuem boa qualidade mas eles podem ser molhorados (até certo ponto) durante o pré-processamento:

- Análise Exploratória
- Limpeza dos dados
- *Feature Selection*
- *Feature Engineering*
- Separação entre Treinamento/Teste
- Normalização

Quase todas as operações são feitas atributo a atributo.

Pré-processamento: Análise Exploratória

Primeiro contato com os dados. Podem ser exploradas as distribuições dos atributos, relações entre atributos (principalmente entre o atributo meta) e outras informações sobre o domínio que possam auxiliar o processo de treinamento.

De maneira geral, os atributos podem ser analisados de duas maneiras:

1. Análise Gráfica
   - boxplots
   - matriz de correlação
   - distribuições dos valores dos atributos (histogramas)
   - relações atributo a atributo (dispersão)
2. Análise Numérica
   - percentis dos atributos
   - valores máximo, minimo, média, mediana, moda, *etc*
   - correlação entre o atributo meta
   - verificação do tipo de cada atributo (útil para as próximas etapas do pré-processamento)

Pré-processamento: Limpeza dos Dados

Problemas comuns enfrentados:

- falha na coleta ou registro das informações (humana ou não)
- valores faltantes (`NaN`, `-9999`, `NULL`, *etc*)
- falta de padronização nos formatos de amostras
- conjuntos de dados desbalanceados
- poucos dados ou dados muito ruidosos

Pré-processamento: Limpeza dos Dados

Possíveis abordagens para limpeza dos dados:

- *Outliers*
  - Detecção (boxplots, relação entre média e mediana do atributo, *etc*)
  - Tratamento: preencher os valores ou remover as amostras.
- Valores faltantes
  - verificar a quantidade de valores faltantes
  - preencher valores utilizando média, mediana, *etc*
  - Remover as linhas em que eles aparecem

Outras operações recorrentes:

- padronização do nome dos atributos (utilizado quando o conjunto de dados está em arquivos separados)
- tratamento de caracteres especiais (ç, ã, ô, *etc*)
- padronização nas unidades de medidas e formatos de registro

Pré-processamento: Limpeza dos Dados

Exemplo de amostras ambíguas:

Metros Quadr.	Andares	n° Banheiros	Esquina	DDD	Preço (K R$)
210	2	4	"Não"	15	350
120	1	2	"Não"	14	250
80	1	1	"Não"	11	180
900	3	7	"Sim"	11	850
245	2	3	"Sim"	14	450
215	1	3	"Sim"	11	310
210	2	4	"Não"	15	500
190	1	2	"Não"	15	220

Pode-se manter o valor mais próximo à media do atributo alvo, mas normalmente essas amostras são descartadas.

Pré-processamento: *Feature Selection*

Feature Selection

Escolha (ou seleção) de quais atributos devem ser utilizados para o treinamento dos modelos.

Considere o seguinte conjunto de dados:

Nome do Dono	**m^2**	**Andares**	**n° Banheiros**	**Esquina**	**DDD**	**Preço (K R$)**
"João"	210	2	4	"Não"	15	350
"Clara"	120	1	2	"Não"	14	250
"José"	80	1	1	"Não"	11	180
"Luzia"	900	3	7	"Sim"	11	850
"Teresa"	245	2	3	"Sim"	14	450
"Carlos"	215	1	3	"Sim"	11	310
"Pedro"	290	2	3	"Sim"	11	500
"Ana"	190	1	2	"Não"	15	220

O atributo "Nome do Dono" é realmente necessário?

Pré-processamento: *Feature Selection*

Em casos "óbvios" o atributo pode ser removido, mas sempre é pertinente validar com o especialista do domínio.

E o atributo "DDD"? Ele deve ser utilizado?

E quando não há especialista?

Existem algoritmos voltados para seleção de atributos. A ideia é testar combinações e comparar a performance selecionando o melhor subconjunto de atributos:

- *Sequential Forward Feature Selection*
- *K-Best*
- *Recursive Feature Elimination*

Por vezes os algoritmos são capazes de encontrar combinações contra-intutivas que não seriam facilmente encontradas por especialistas humanos.

Pré-processamento: *Feature Extraction*

Feature Extraction

Redução do número de atributos (dimensionalidade) mantendo características em um conjunto menor de atributos.

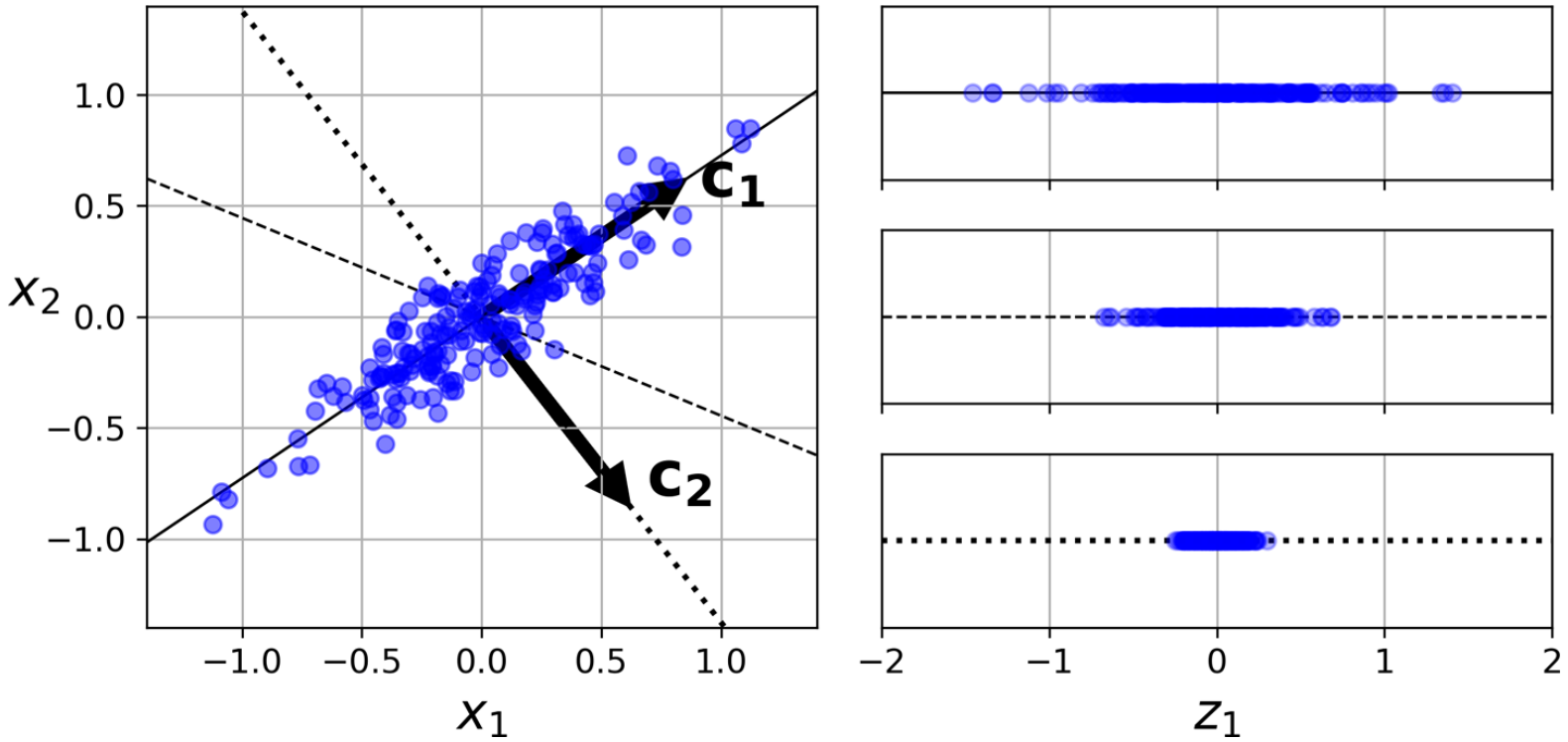
Alguns métodos utilizados para reduzir a dimensionalidade dos conjuntos de dados:

- *Principal Component Analysis* e suas variantes
- *Locally Linear Embedding*

Os objetos transformadores devem ser armazenados para o processamento de novas amostras no futuro.

Pré-processamento: *Feature Extraction*

Exemplo de aplicação de *Feature Extraction* utilizando PCA



Fonte: *Hands-on Machine Learning with Scikit-Learn and Keras/Tensorflow* – A. Géron

Pré-processamento: Separação entre Treinamento Teste

O conjunto de dados deve ser divido em 2 partes sendo elas:

1. **Conjunto de Treinamento:** conjunto utilizado para o treinamento e ajuste dos hiper-parâmetros do modelo
2. **Conjunto de Teste:** conjunto utilizado para a avaliação final do modelo

Atenção!

Os conjuntos de treinamento e teste **jamais** devem ser misturados, essa mistura pode gerar uma percepção otimista sobre a performance do modelo!

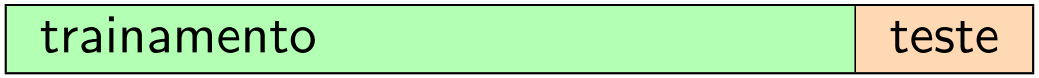
Transformadores

Nesta etapa, os transformadores são treinados apenas com os dados de treinamento

Pré-processamento: Separação entre Treinamento Teste

Existem diferentes formas de realizar a separação dos dados:

- *Holdout*: o conjunto de dados é dividido em 2 partes sendo uma % para treinamento e outra para teste:



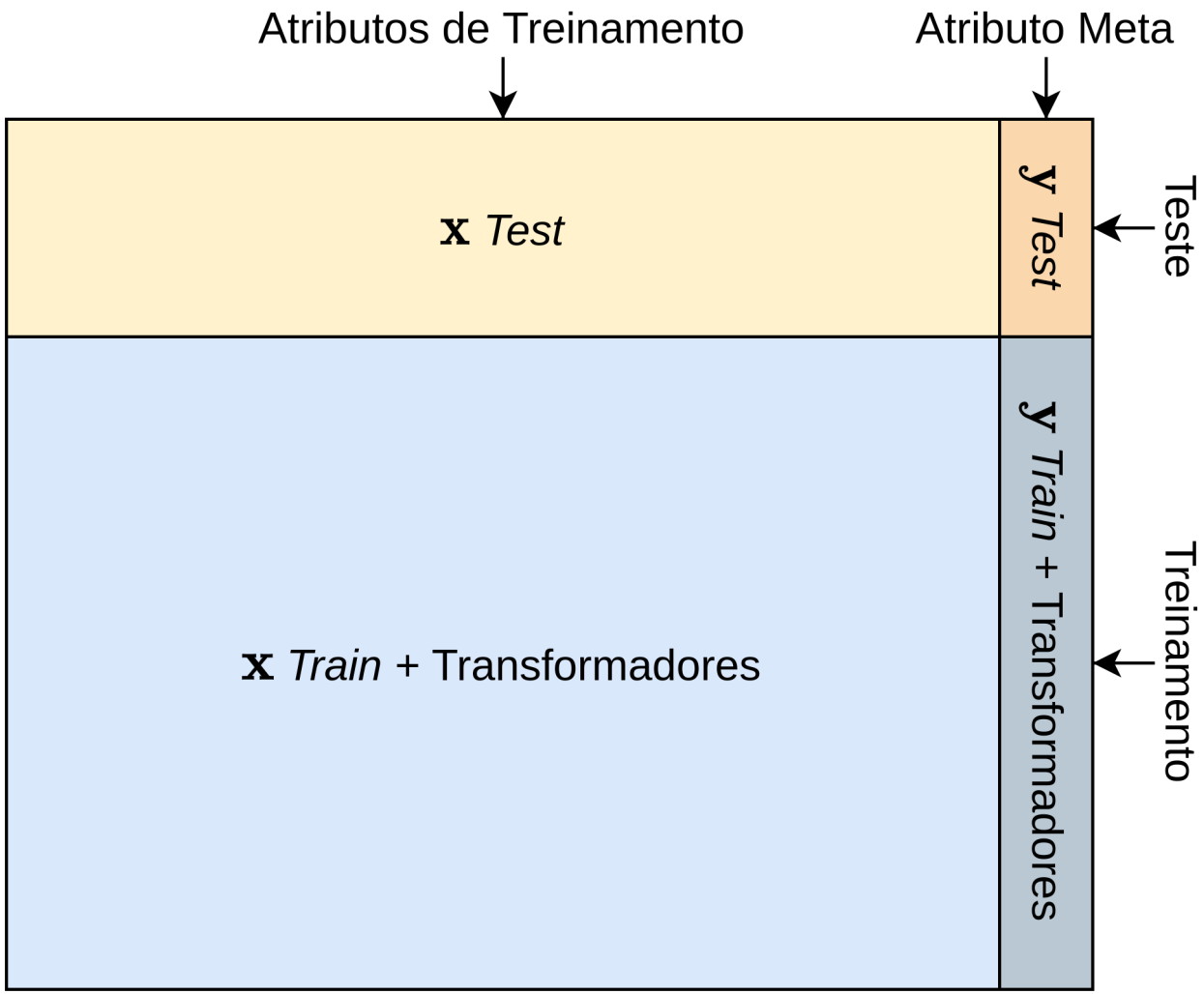
- *Cross-Validation*: conjunto é divido em k "partes" com o mesmo tamanho (chamados de *folds*) da seguinte forma:

fold 1	teste				
fold 2		teste			
fold 3			teste		
fold 4				teste	
fold 5					teste

Exemplo com $k = 5$

O treinamento é realizado k vezes utilizando os conjuntos separados; permite que todo o conjunto de dados seja utilizado para treinamento, mas sem que haja mistura

Pré-processamento: Separação entre Treinamento Teste



Pré-processamento: Normalização

Os dados utilizados como entrada nas Redes Neurais (e em muitos outros algoritmos de Aprendizado de Máquina) precisam **ter a escala dos atributos padronizada**.

✓ Processo chamado de *Normalização* – ou padronização – é feito atributo a atributo depois de todas as transformações.

feature 2	*feature 3*	*feature 4*	*feature 5*
True	10	1.092	1
True	7	450	0
False	9	2.938	0
True	8	4.536	1
False	7	993	1

Normalmente os valores ficam restritos a $[0, 1]$ ou valores pequenos, mas próximos a 0.

Pré-processamento: Normalização

O processo de *Normalização* pode ser feito de diferentes maneiras. Duas mais utilizadas são:

Min Max Scaling

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}$$

Standard Scaling

$$x_i' = \frac{x_i - \text{média}(x_i)}{\text{desvio padrão}(x_i)}$$

Lembrando que x_i é o i-ésimo atributo de x

- ✓ Acelera a convergencia dos modelos
- ✓ Evita que atributos específicos tenham maior influência no ajuste dos pesos entre os neurônios
- ✓ Obrigatório em alguns algoritmos de Machine Learning

Para outros métodos de Normalização confira: *Different Scalers on Data – Scikit-Learn*

Pré-processamento: Normalização

Exemplo de Normalização utilizando *Min Max Scaling*:

feature 2	*feature 3*	*feature 4*	*feature 5*
1	10	1.092	1
1	7	450	0
0	9	2.938	0
1	8	4.536	1
0	7	993	1

$\downarrow$

feature 2	*feature 3*	*feature 4*	*feature 5*
1	1.000	0.157	1
1	0.000	0.000	0
0	0.667	0.608	0
1	0.333	1.000	1
0	0.000	0.133	1

Pré-processamento: Normalização

Exemplo de Normalização utilizando *Standard Scaling*:

feature 2	*feature 3*	*feature 4*	*feature 5*
1	10	1.092	1
1	7	450	0
0	9	2.938	0
1	8	4.536	1
0	7	993	1

↓

feature 2	*feature 3*	*feature 4*	*feature 5*
1	1.380	-0.535	1
1	-0.920	-0.913	0
0	0.613	0.550	0
1	-0.153	1.491	1
0	-0.920	-0.593	1

Pré-processamento: Normalização

Observações sobre a normalização:

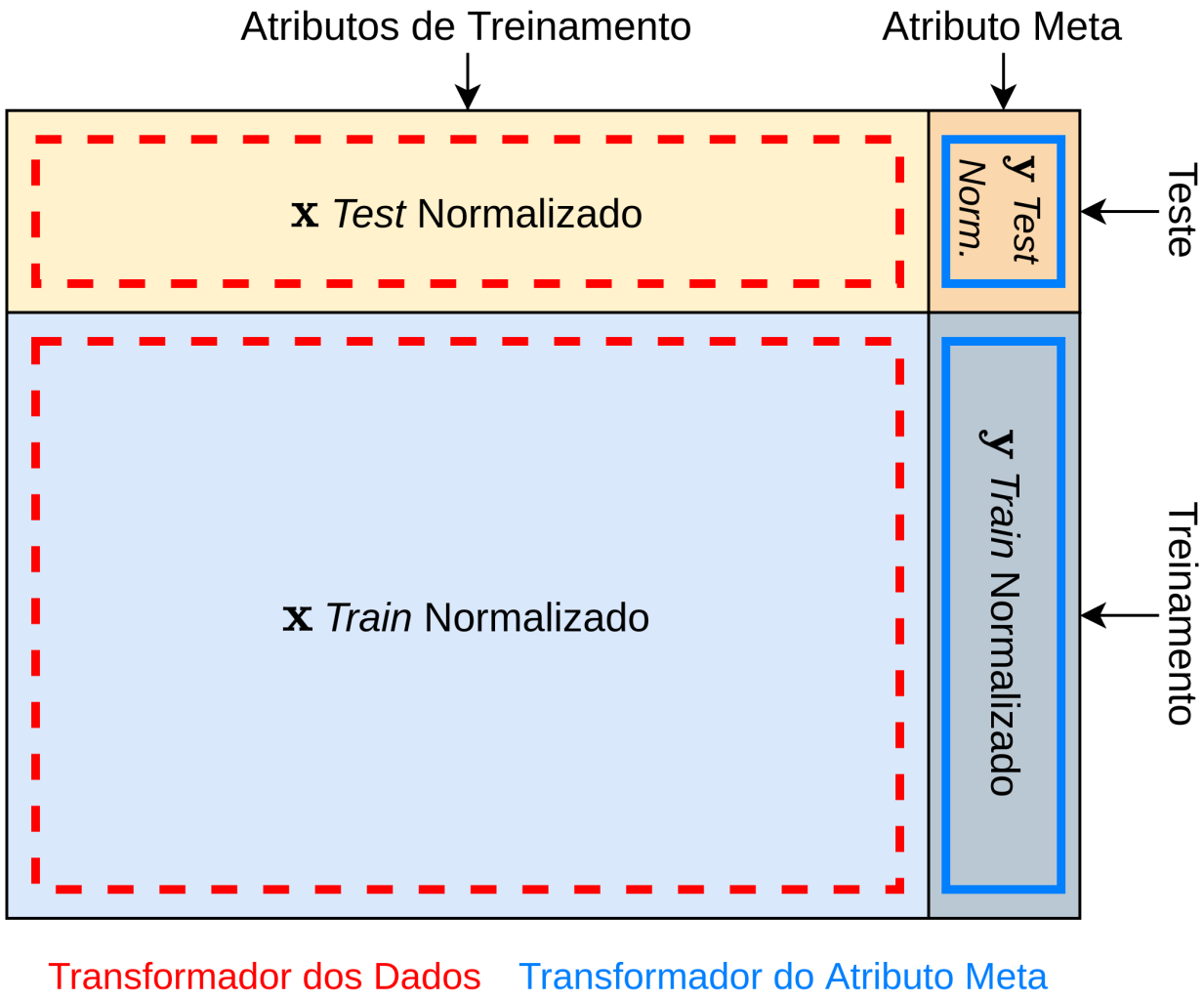
- a normalização é feita em todos os atributos (até no meta)[1]
- os objetos normalizadores são "treinados" utilizando os mesmos dados de treinamento dos modelos
- esses objetos são salvos para realizar a **Transformação Inversa** das previsões feitas pelos modelos
- os modelos devem ser avaliados em termos dos dados na mesma escala dos dados originais (as previsões devem passar pelo processo de transformação inversa)

Atenção!

As trasformações feitas nos dados de teste (normalização, redução de dimensionalidade, *etc*) devem ser realizadas com os mesmos transformadores usados no treinamento.

[1] Por detalhes de implementação, a normalização dos dados e do atributo meta é feita por objetos normalizadores distintos

Pré-processamento: Normalização



Pré-processamento: Normalização

- ✓ **Transformador dos Dados:** É mantido para fazer a normalização de novos dados de entrada no futuro (incluindo no teste).
- ✓ **Transformador do Atributo Alvo:** É mantido para a transformação inversa das previsões feitas pelos modelo.

As informações usadas para realizar a normalização (média, mediana, *etc*) devem ser apenas obtidas dos dados de treinamento. Nenhuma informação sobre o conjunto de testes pode ser passada para os modelos em nenhuma etapa.

Parte 2: Processo de Treinamento e Avaliação

Processo de Treinamento e Avaliação

Uma vez que os dados foram pré-processados o *pipeline* continua...

- geração das previsões no conjunto de testes
- transformação inversa utilizando os objetos normalizadores
- avaliação da performance
- **verificar a existência de *overfitting* ou *underfitting***...

Problemas Comuns: Underfit e Overfit

Durante o processo de treinamento o 2 grandes problemas podem ocorrer:

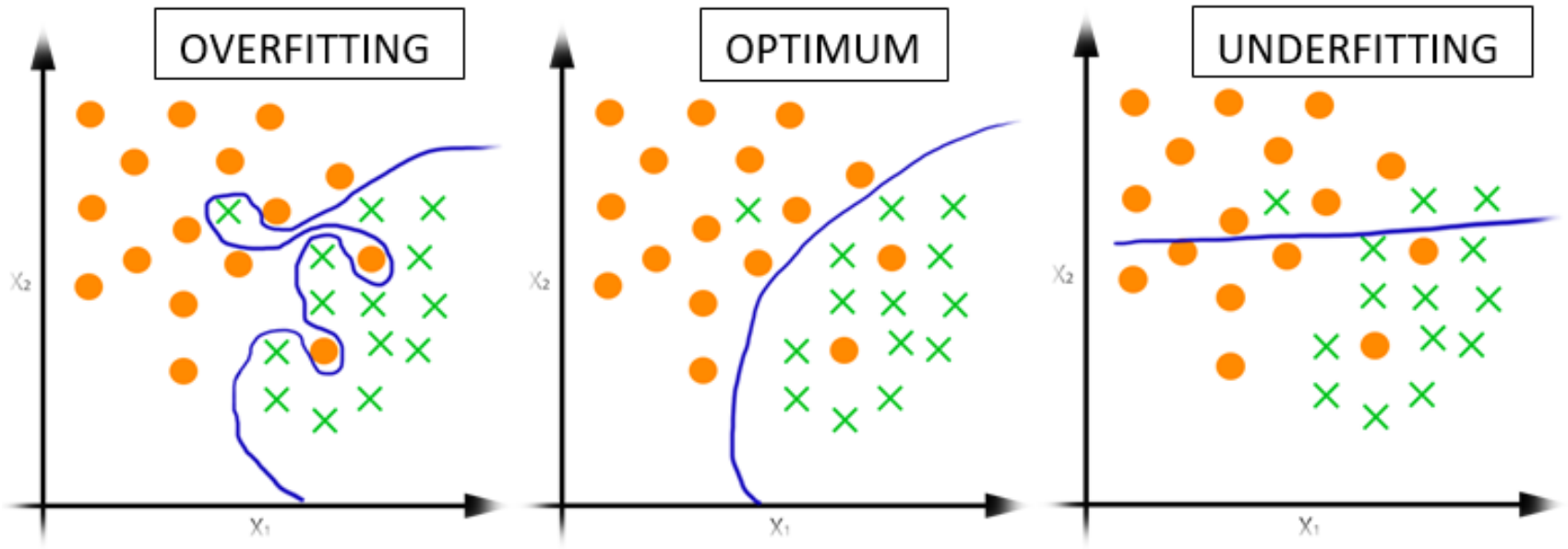
Underfit

Também chamado de subajuste, ocorre quando a função-hipótese h_θ não tem complexida suficiente para se ajustar aos dados de treinamento de forma adequada. É dito que o modelo tem **alto viés.**

Overfit

Também chamado de sobreajuste, ocorre quando a função-hipótese h_θ tem uma complexidade muito alta – maior do que o apropriado – para se ajustar aos dados. É dito que o modelo tem **alta variância.**

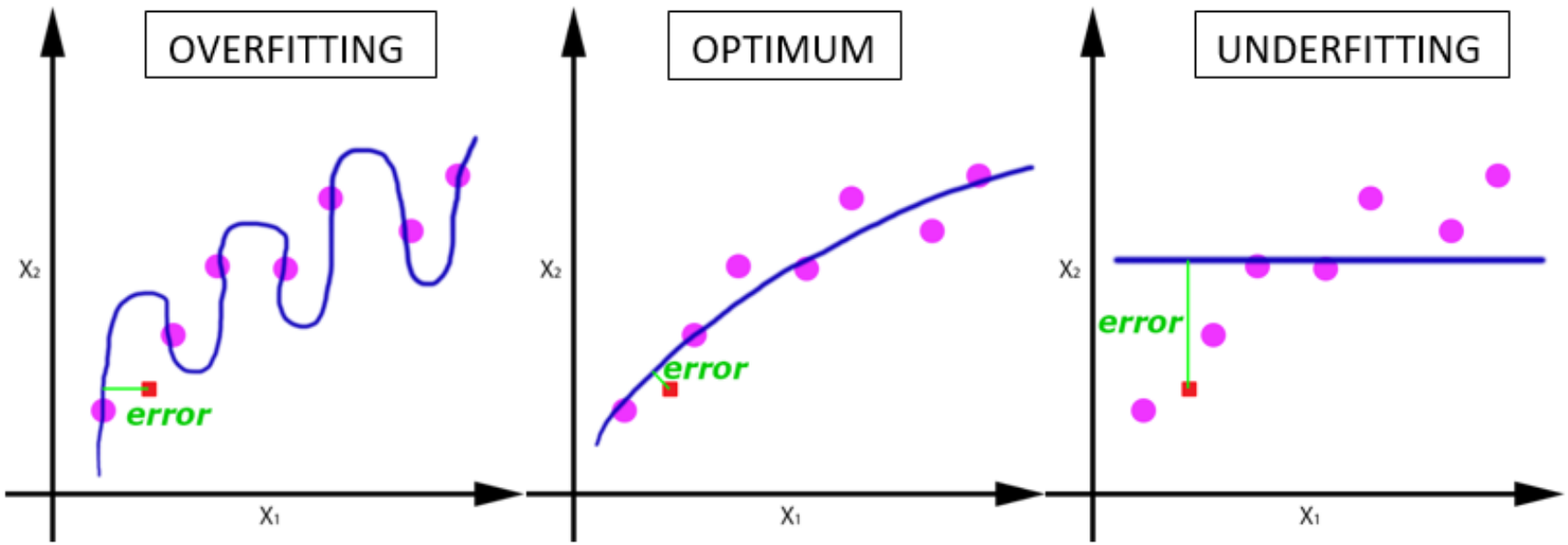
Problemas Comuns: Underfit e Overfit



Overfit e Underfit em uma tarefa de classificação.

Fonte: `https://machinelearningmedium.com/2017/09/08/overfitting-and-regularization/`

Problemas Comuns: Underfit e Overfit



Overfit e Underfit em uma tarefa de regressão.

Fonte: `https://machinelearningmedium.com/2017/09/08/overfitting-and-regularization/`

Problemas Comuns: Como detectá-los?

Ambos os problemas podem ser detectados usando as curvas de aprendizagem durante o treinamento ou pelas métricas de desempenho após a avaliação:

- ✓ **Ideal:** os erros no treinamento e no teste são pequenos e estão próximos
- ✗ **Overfit/Sobreajuste:** existe uma grande diferença entre o erro no treinamento e o do teste
  - treinamento $\rightarrow$ erro baixo e sempre diminuindo
  - teste $\rightarrow$ grande diferença para o de treinamento[2]
- ✗ **Underfit/Subajuste:** ambos os erros estão próximos, mas são muito altos

[2] O erro no teste pode começar a aumentar enquanto o de treinamento continua a diminuir.

Problemas Comuns: Como detectá-los?

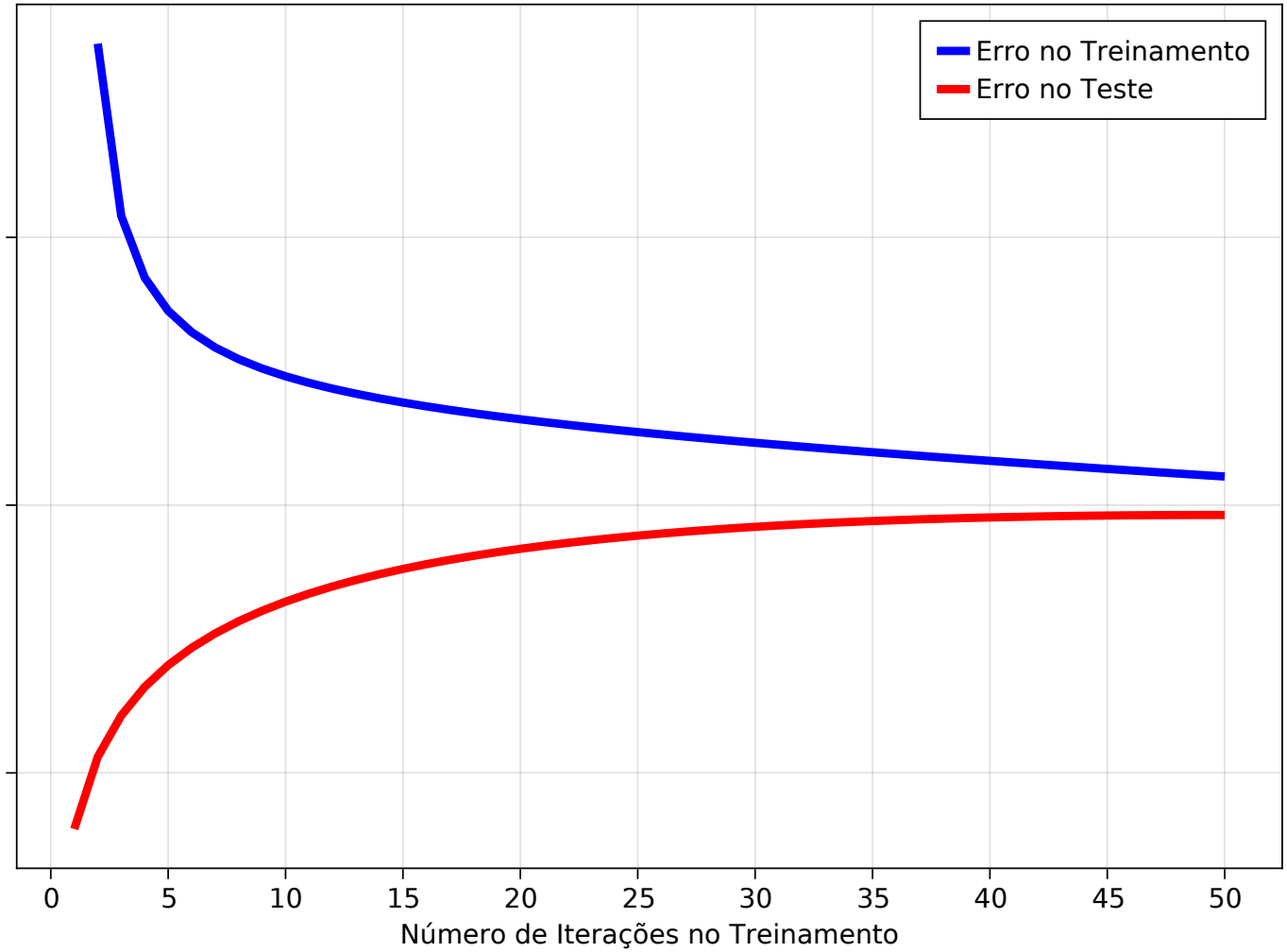


Figura: Cenário ideal (erros baixos e próximos)

Problemas Comuns: Como detectá-los?

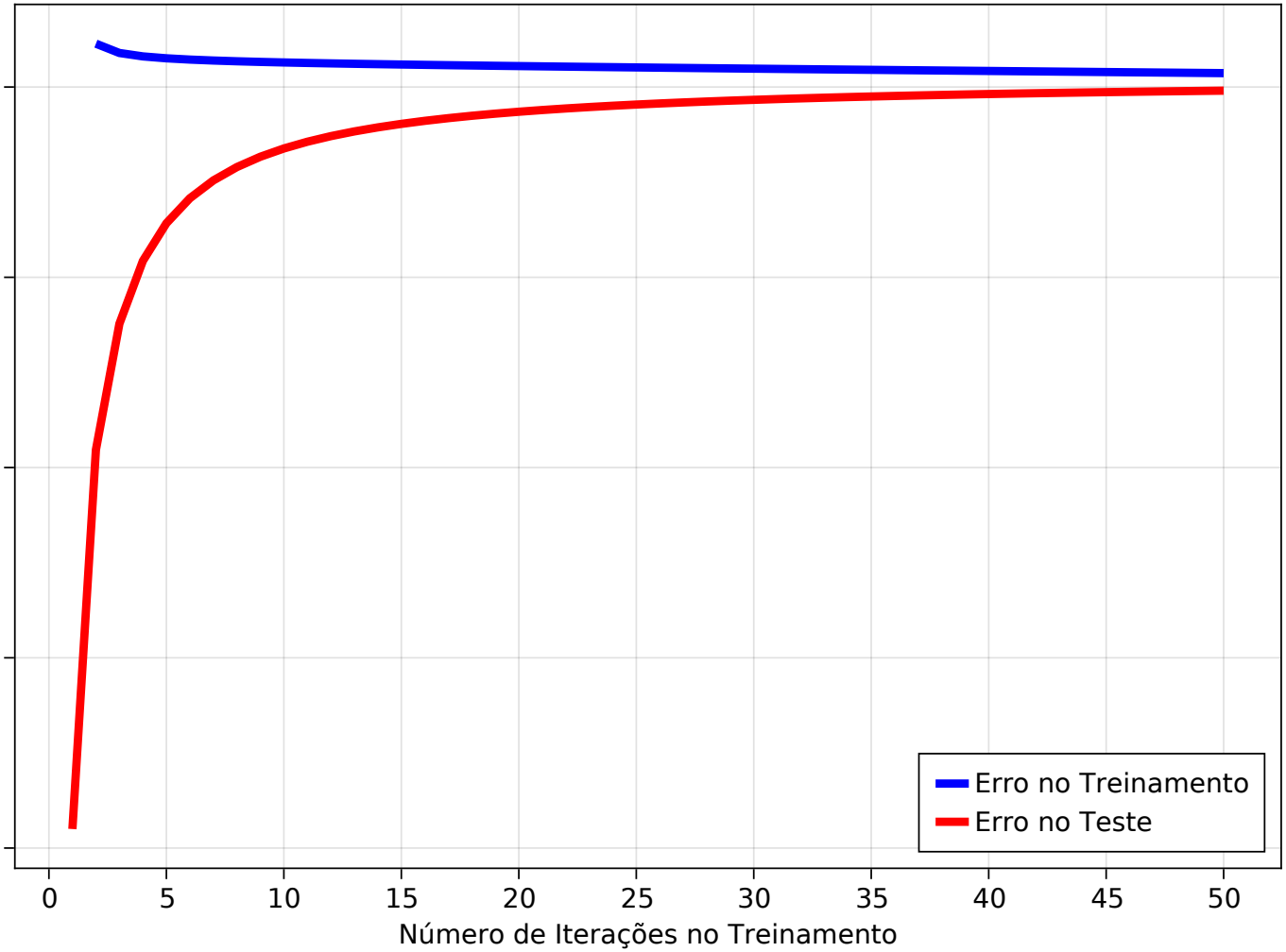


Figura: Underfit (erros próximos, mas muito altos)

Problemas Comuns: Como detectá-los?

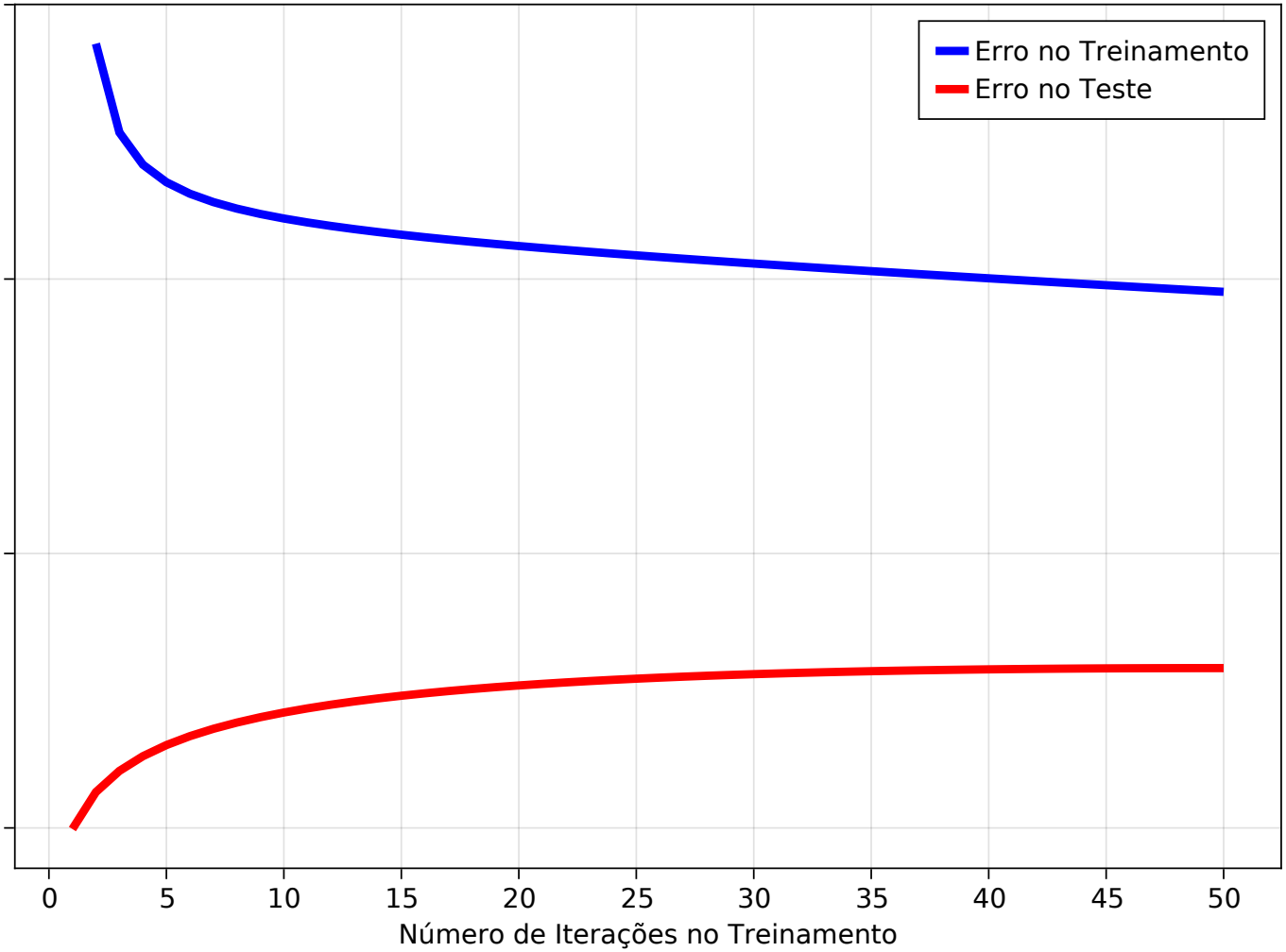


Figura: Overfit (grande diferença entre os dois erros)

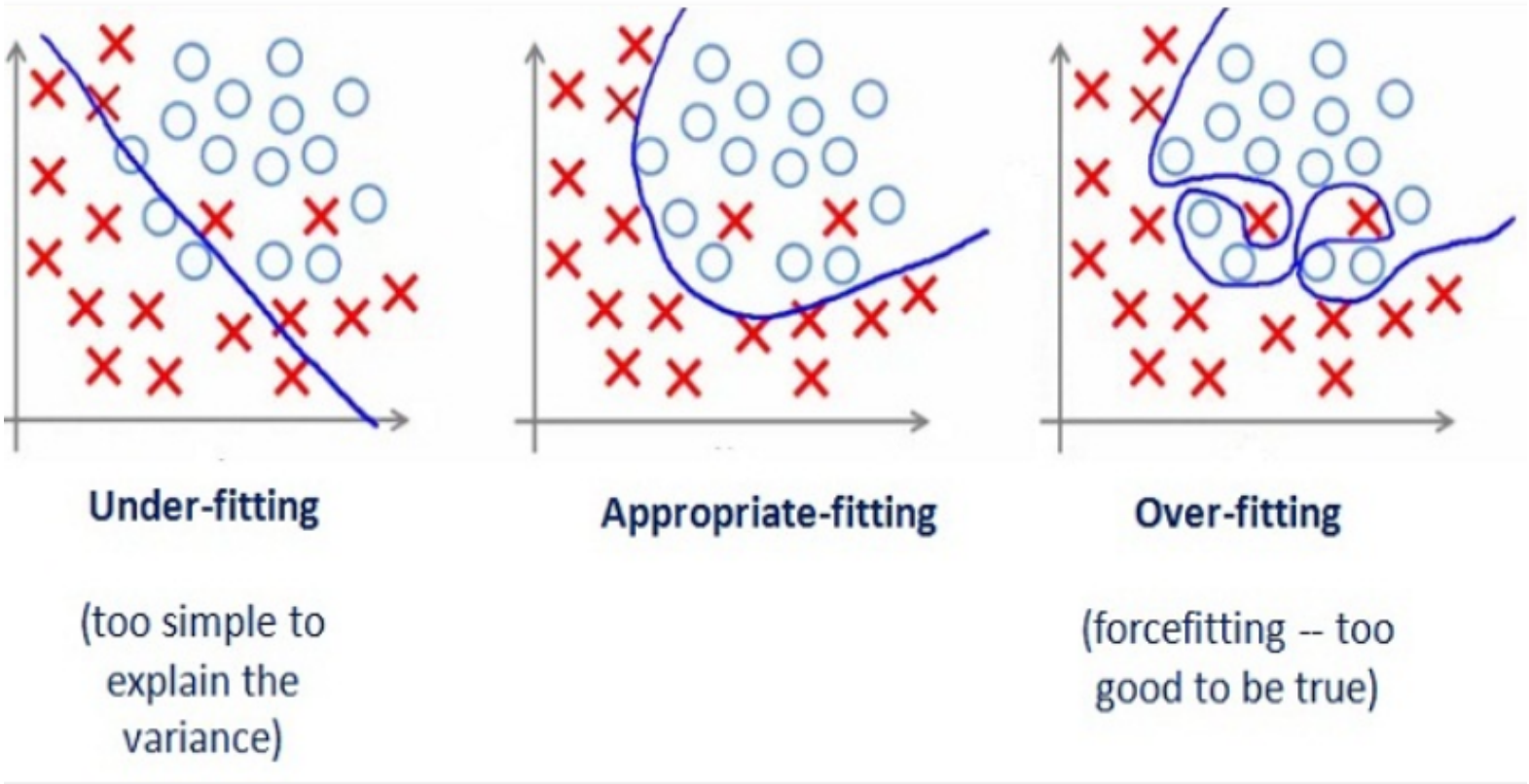
Problemas Comuns: Como evitá-los?

- **Underfit:**
  - ✓ Aumentar a complexidade da hipótese
    (nas Redes Neurais → aumentar o número de neurônios por camada ou o número de camadas ocultas)
  - ✓ Aumentar a quantidade de *features* nos conjuntos de treinamento
  - ✓ Diminuir a *Regularização* durante o treinamento do modelo
- **Overfit:**
  - ✓ Reduzir a complexidade da hipótese
    (nas Redes Neurais → reduzir o número de neurônios por camada ou o número de camadas ocultas)
  - ✓ Aumantar a quantidade de amostras no treinamento (se possível).
  - ✓ Aumentar a *Regularização* durante o treinamento do modelo

Regularização

É uma restrição do grau de liberdade do modelo, força o processo de treinamento não apenas a se ajustar aos dados, mas também a encontrar a funçãoo-hipótese h_θ mais simples possível.

Problemas Comuns: Como evitá-los?



Avaliação de Performance

A avaliação de performance depende da tarefa que é realizada pelo modelo:

Regressão

- Erro Quadrático Médio
- Erro Absoluto Médio
- R_2 Score

Classificação

- Precisão
- Acurácia
- F1-Score

Convenção

Para os cálculos a convenção é utilizar:

$$y^{(i)} \rightarrow i\text{-ésima previsão}$$

$$\hat{y}^{(i)} \rightarrow i\text{-ésima valor esperado}$$

Para outras métricas confira: *Metrics and Scoring – Scikit-Learn*

Avaliação de Performance: Métricas de Regressão

Erro Quadrático Médio *(Mean Squared Error)*

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

Erro Absoluto Médio *(Mean Absolute Error)*

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y^{(i)} - \hat{y}^{(i)}\right|$$

R^2 Score

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad \text{onde} \quad \bar{y} = \frac{1}{n}\sum_{i=0}^{n} y_i$$

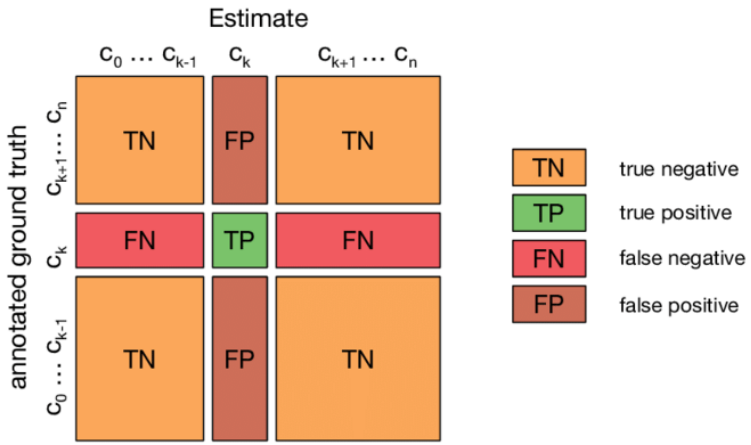
Avaliação de Performance: Métricas de Classificação

As métricas de classificação são construidas sobre a Matriz de Confusão.

Duas classes

		Previsão	
		Classe +	Classe −
Real	Classe +	VP	FN
	Classe −	FP	VN

Múltiplas classes



Múltiplas Classes

Normalmente as métricas de desempenho são agregadas (normalmente pela média) para as classes positivas e negativas em cada classificação.

Avaliação de Performance: Métricas de Classificação

Precisão

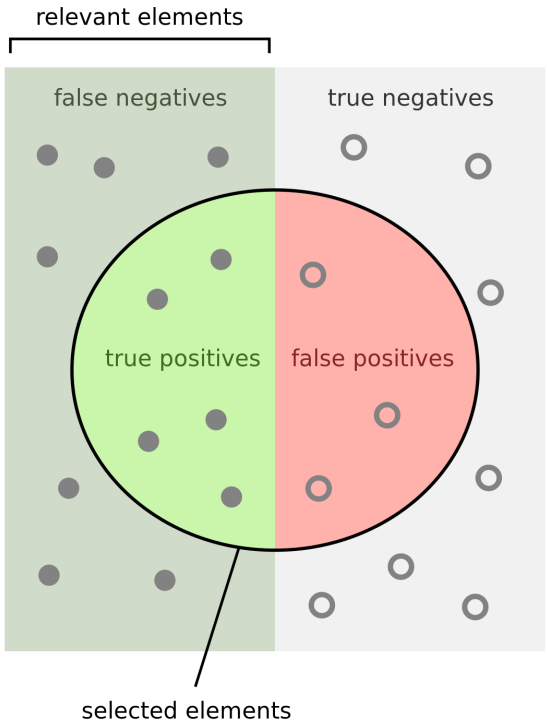
$$P_+ = \frac{VP}{VP + FP} \qquad P_- = \frac{VN}{VN + FN}$$

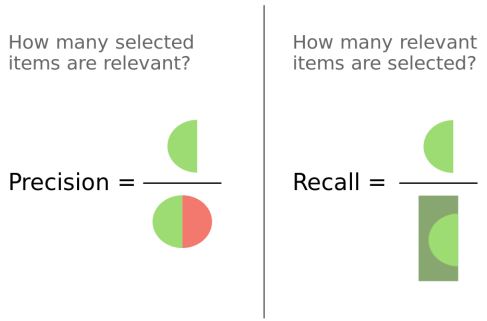
Revocação – *Recall*

$$R_+ = \frac{VP}{VP + FN} \qquad R_- = \frac{VN}{VN + FP}$$

F1-Score (média hormônica da Precisão e Revocação)

$$F = 2\frac{P \times R}{P + R}$$





Ferramentas

Linguagem de Programação: Python 3 (ver 3.9)

Bibliotacas:

- **Pandas:** manipulação dos conjuntos de dados (pré-processamento)
- **Scikit-Learn:** implementações de métodos utilizados em Machine Learning (modelos, pré-processamento, avaliação, *etc*)
- **Keras/Tensorflow:** implementações das Redes Neurais (além de outras funcionalidades)

Ambiente: Google Collab