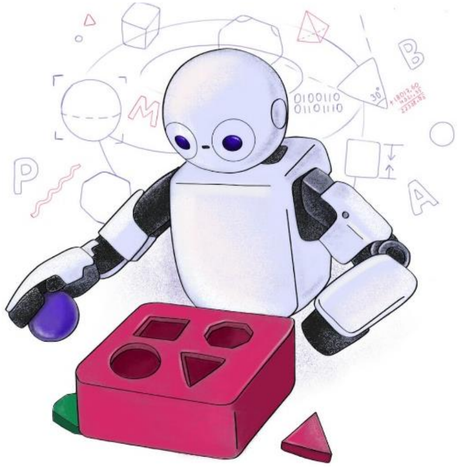
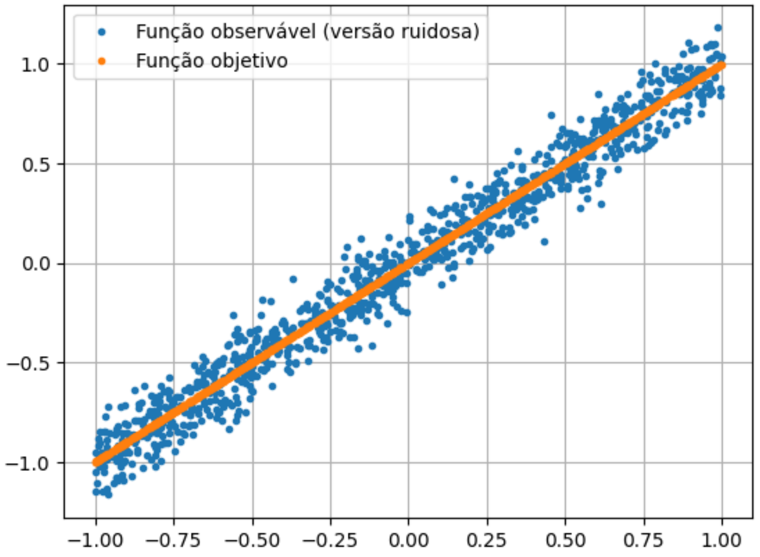
TP555 - Inteligência Artificial e Machine Learning: *Regressão com Modelos Não-Lineares com Relação aos Atributos*

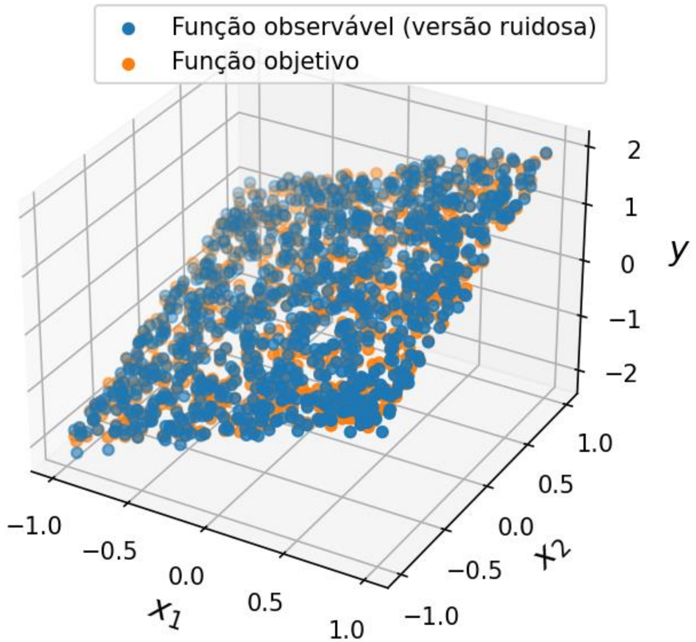


Inatel

Felipe Augusto Pereira de Figueiredo
felipe.figueiredo@inatel.br

Até agora, usamos ***funções hipótese com formato de hiperplanos***, e.g., retas e planos, para aproximar ***mapeamentos lineares*** entre os atributos e o valor esperado, mas ***e se os mapeamentos forem não lineares***?

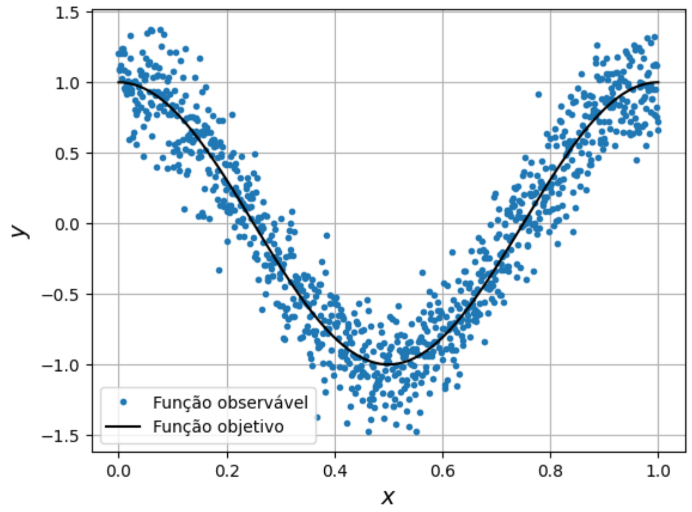


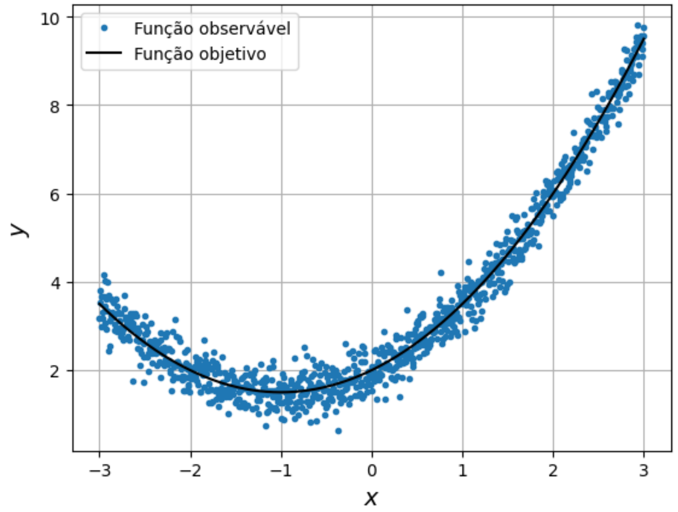


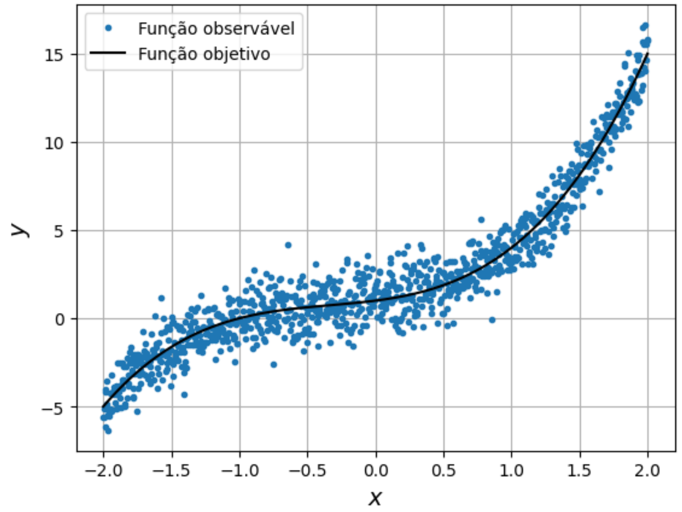
O que podemos fazer quando ***hiperplanos*** ***não se ajustam bem aos dados***?

Mapeamentos não lineares

- Observem as figuras abaixo, uma ***reta*** claramente ***não seria uma boa escolha para aproximar esses mapeamentos não lineares***.
  - ***Retas não capturariam o comportamento das funções abaixo***, pois elas não têm ***complexidade ou flexibilidade*** (i.e., graus de liberdade) o suficiente para isso.
- Portanto, qual tipo de função hipótese seria mais apropriada para aproximar esses comportamentos não lineares?







Extensão para modelos não-lineares com relação aos atributos

- ***Modelos não-lineares*** constroem uma aproximação por meio da ***combinação linear de funções-base não-lineares***, da forma

$$h(\boldsymbol{x}) = \hat{y}(\boldsymbol{x}) = a_0 + a_1 b_1(\boldsymbol{x}) + \cdots + a_M b_M(\boldsymbol{x}),$$

onde $b_m(\boldsymbol{x}): \mathbb{R}^K \rightarrow \mathbb{R}$, denota a i-ésima ***função-base***.

- Portanto, por ser linear com relação aos pesos, os r*esultados encontrados anteriormente são facilmente estendidos para o caso não-linear* bastando redefinir o ***vetor de atributos***, $\boldsymbol{x}(i)$, como o ***vetor de funções-base***

$$\boldsymbol{x}(i) = \left[1, b_1\big(\boldsymbol{x}(i)\big), \ldots, b_M\big(\boldsymbol{x}(i)\big)\right]^T.$$

- Exemplos de funções base:
  - $b_m(\boldsymbol{x}) = x_1 * x_2$
  - $b_m(\boldsymbol{x}) = \log_{10} x_1$
  - $b_m(\boldsymbol{x}) = x_1^2$

Linearização

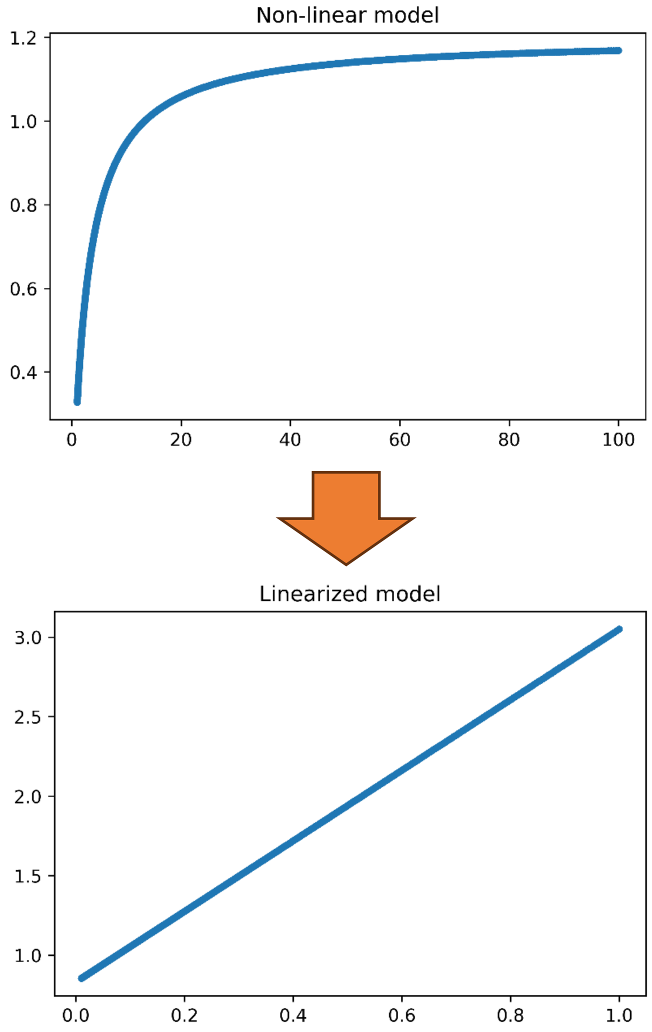
- Em geral, processos e modelos físicos ***não são lineares*** (e desta vez em relação aos pesos também), o que dificulta sua interpretação.
- Porém, existem ***modelos não-lineares*** que são ***intrinsecamente lineares***, pois podem ser tornados lineares em relação aos pesos ***através de uma modificação***.
- Por exemplo, as equações abaixo podem ser transformadas em uma combinação linear de funções base não lineares.

$$y = a_0 {x_1}^{a_1} e^{a_2 x_2},$$

$$y = \frac{a_0 x_1}{a_1 + x_1}.$$

- A vantagem é que modelos lineares são mais fáceis de interpretar.

Exemplo: Linearização



A equação,

$$y = \frac{a_0 x_1}{a_1 + x_1},$$

pode ser reescrita como

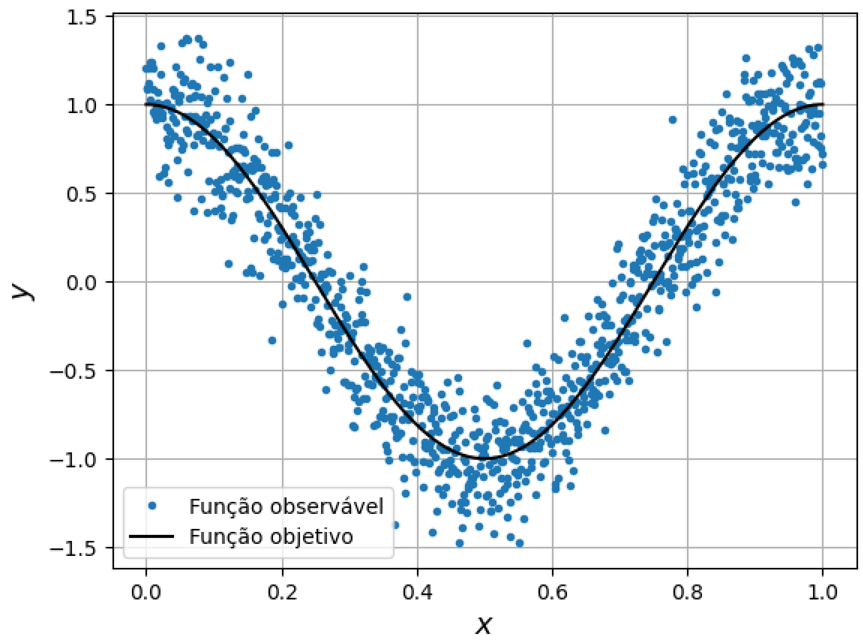
$$\frac{1}{y} = \frac{a_1 + x_1}{a_0 x_1} = \frac{1}{a_0} + \frac{a_1}{a_0}\frac{1}{x_1} = a_0' + a_1' x_1',$$

que é linear em relação aos pesos transformados.

- Depois do modelo ser linearizado, os pesos podem ser encontrados com a equação normal ou com o gradiente descendente.

Exemplo: model_linearization.ipynb

Regressão polinomial



- O teorema da aproximação de ***Stone-Weierstrass*** diz que *"qualquer função contínua no intervalo fechado [a, b] pode ser uniformemente aproximada por um* ***polinômio****"*.
- Portanto, podemos aproximar ***qualquer tipo de mapeamento (linear ou não linear)*** com ***polinômios***, bastando apenas ***encontrar o grau (ou ordem) ideal***.

Regressão polinomial

- Exemplo de um polinômio de grau 4* com três atributos, x_1, x_2 e x_3

$$y(x_1, x_2, x_3) = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_3 + a_5 {x_1}^3 + a_4 \boxed{x_1 x_2 {x_3}^2}$$

- O grau do polinômio é o maior valor resultante da soma dos expoentes dos atributos de um monômio
- Percebam que em alguns monômios existe a ***combinação dos atributos originais***, formando novos atributos (ou ***funções base***).
- Um problema com polinômios é que a presença de ***outliers*** nos dados de treinamento impacta o desempenho do modelo.

Exemplo: polynomial_regression_example_with_outliers.ipynb

Regressão polinomial

- Por simplicidade didática, inicialmente, nós consideraremos ***funções hipóteses polinomiais em uma variável*** (i.e., com ***um*** atributo)

$$h(x_1(n)) = a_0 + a_1 x_1(n) + a_2 {x_1}^2(n) + \cdots + a_M {x_1}^M(n) = \boldsymbol{a}^T \boldsymbol{x}(n),$$

onde $n = 1 \ldots N$ é o número da amostra, M é o grau do polinômio, $\boldsymbol{a} = [a_0 \;\; a_1 \;\; a_2 \;\; \ldots \;\; a_M]^T \in \mathbb{R}^{M+1 \times 1}$, $\boldsymbol{x}(n) = [x_0 \;\; x_1(n) \;\; {x_1}^2(n) \ldots \;\; {x_1}^M(n)]^T \in \mathbb{R}^{M+1 \times 1}$ e $x_0 = 1$ é o atributo de *bias,* associado ao peso de *bias*, a_0.

- ***Todos resultados encontrados anteriormente*** (equação normal, vetor gradiente para implementação do algoritmo do gradiente descendente, escalonamento) ***são diretamente estendidos para funções hipótese polinomiais***.

Regressão polinomial

- Só precisamos nos lembrar que o ***vetor de atributos***, $\boldsymbol{x}$, e consequentemente, a ***matriz de atributos***, $\boldsymbol{X}$, são ***compostos pelas funções base***.
- Por exemplo, para a seguinte ***função hipótese polinomial***

$$h(x_1(n)) = a_0 + a_1 x_1(n) + a_2 {x_1}^2(n) + \cdots + a_M {x_1}^M(n),$$

a ***matriz de atributos polinomial***, $\boldsymbol{X}$, fica da seguinte forma

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1(0) & {x_1}^2(0) & \cdots & {x_1}^M(0) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1(N-1) & {x_1}^2(N-1) & \cdots & {x_1}^M(N-1) \end{bmatrix} \in \mathbb{R}^{N \times M+1},$$

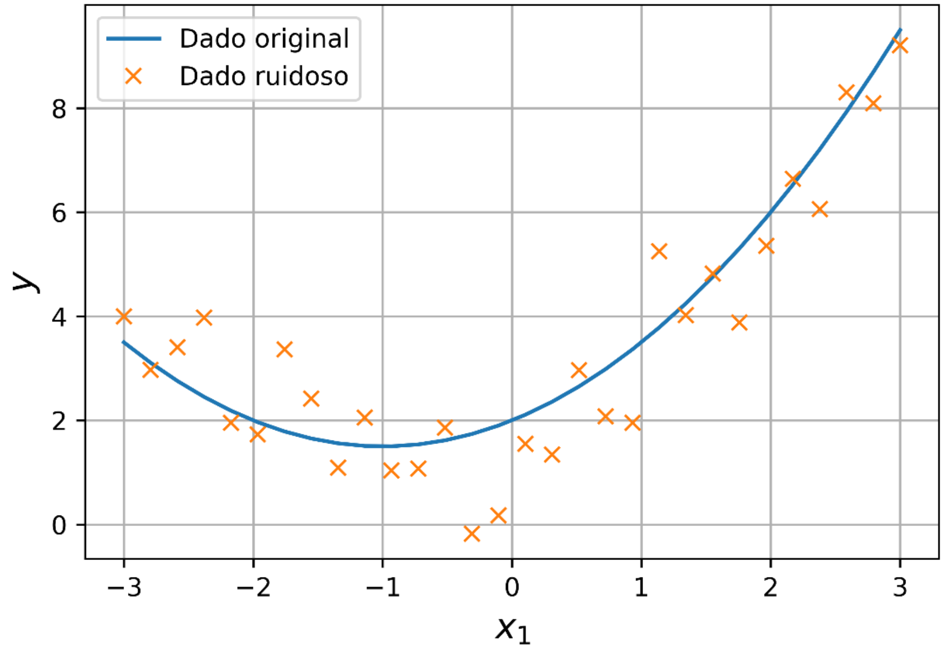
onde cada coluna contém uma ***função base***.

* A biblioteca SciKit-Learn possui uma função que cria essas matrizes automaticamente a partir dos atributos originais.

Exemplo: polynomial_features_from_scikit.ipynb

Porém, o desafio agora é ***encontrar o grau do polinômio*** que melhor aproxime os dados.

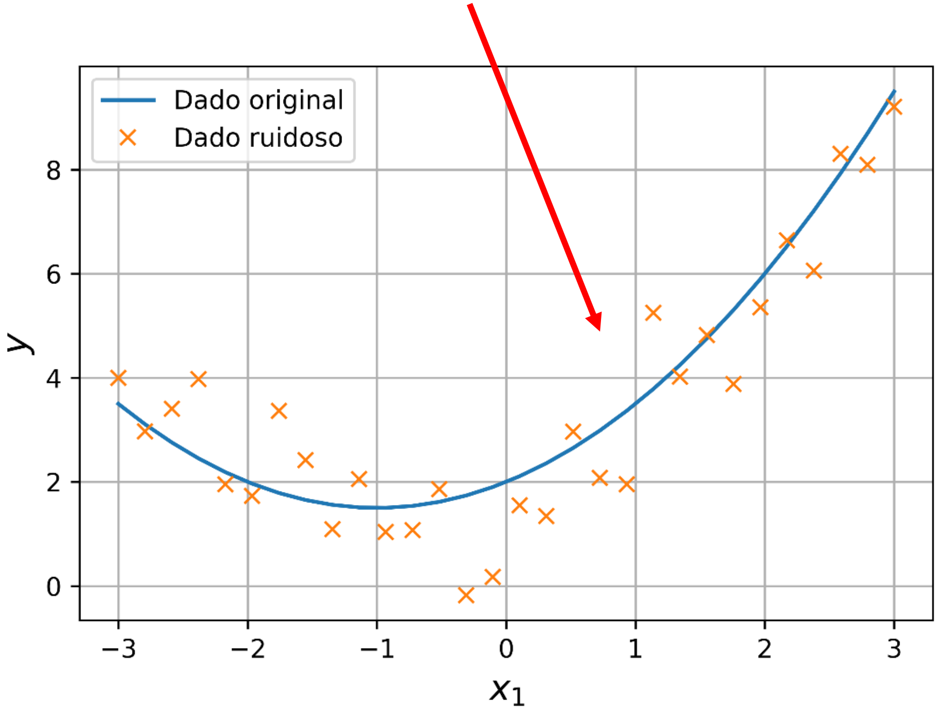
Exemplo de regressão usando polinômio



Usando apenas os ***dados ruidosos***, ***qual a ordem e pesos de um polinômio*** para que ele se aproxime da ***função objetivo*** da melhor forma possível?

Exemplo de regressão usando polinômio

Função objetivo: polinômio de ordem 2.



A partir do dados ruidosos, queremos encontrar um polinômio (pesos e ordem) que melhor se aproxime da função objetivo.

- Para exemplificar essa questão da busca pela ordem do polinômio aproximador geramos ***30 exemplos*** da seguinte função objetivo (polinômio de ordem 2)

$$y(x_1(n)) = 2 + x_1(n) + 0.5{x_1}^2(n),$$

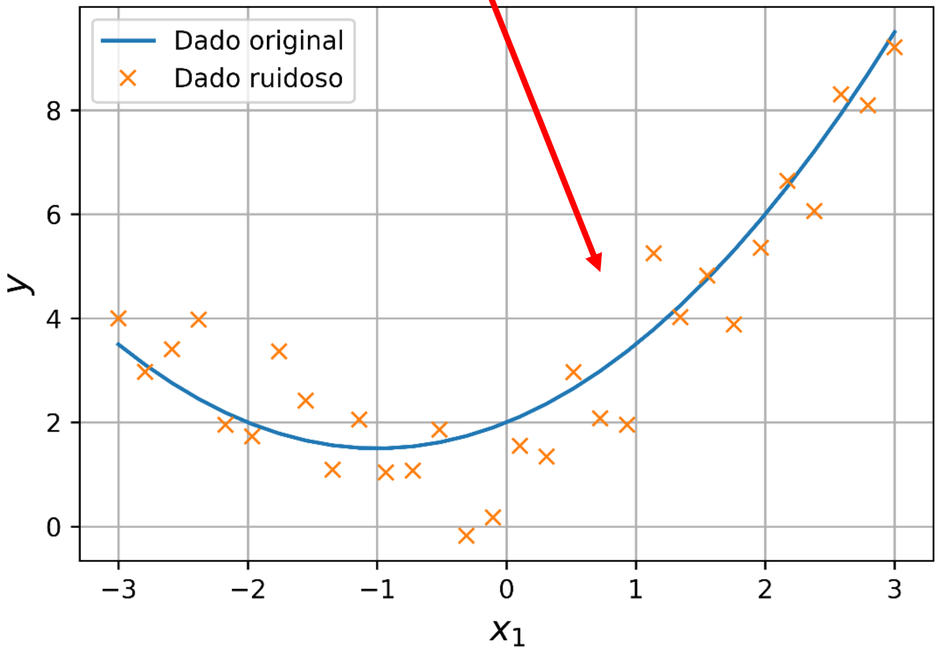
e adicionamos ruído Gaussiano branco, $w(n)$

$$y_{\text{noisy}}(x_1(n)) = y(x_1(n)) + w(n),$$

onde $x_1(n)$ são valores linearmente espaçados entre -3 e 3 e $w(n) \sim N(0,1)$.

Exemplo de regressão usando polinômio

Função objetivo: polinômio de ordem 2.



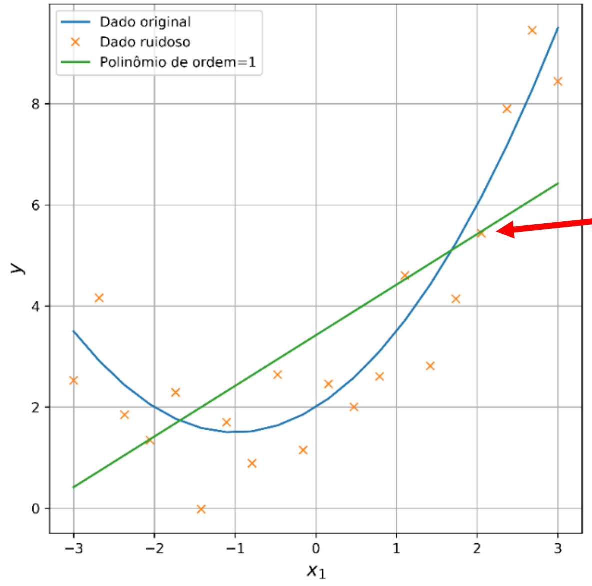
A partir do dados ruidosos, queremos encontrar um polinômio (pesos e ordem) que melhor se aproxime da função objetivo.

- Vamos usar uma ***função hipótese polinomial*** para ***aproximar a função objetivo*** ***a partir dos dados ruidosos***.
- Porém, surge uma dúvida, ***e se não soubéssemos a ordem por trás do modelo gerador, qual grau deveríamos utilizar***?

Exemplo: polynomial_regression.ipynb

Regressão polinomial: Qual ordem usar?

Polinômio de ordem 1.



Reta não é flexível o suficiente para se contorcer e aproximar o comportamento da função objetivo.

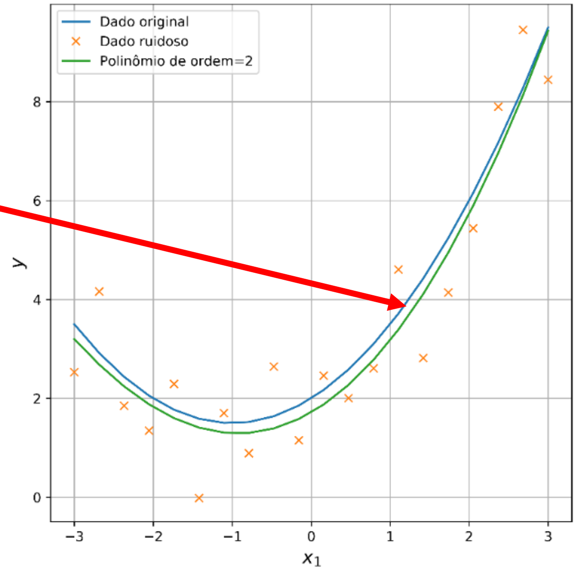
Flexibilidade e **grau de generalização** muito baixos.

- Polinômio de ordem 1 (i.e., reta) não tem flexibilidade o suficiente para aproximar o comportamento por trás das amostras ruidosas, ou seja, a função objetivo.
- O erro (MSE) é alto para exemplos dos conjuntos de treinamento e de validação (i.e., exemplos não vistos durante o treinamento).
- Efeito conhecido como ***subajuste*** ou ***underfitting***: ***flexibilidade*** e ***grau de generalização*** muito baixos.

Regressão polinomial: Qual ordem usar?

Ordem ótima pois é a mesma do modelo gerador.

Polinômio de ordem 2.

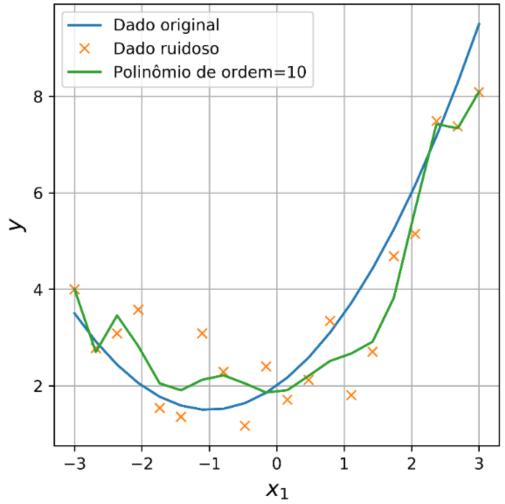


Encontra relação de compromisso entre **flexibilidade** e **generalização**: **flexibilidade** e **grau de generalização** em ***equilíbrio***.

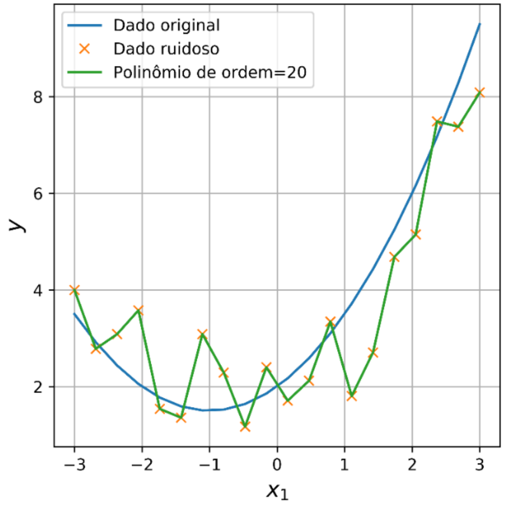
- Porém, como esperado, o polinômio de ordem 2 produz a melhor aproximação da função objetivo, errando pouco para exemplos dos conjuntos de treinamento e validação.
  - Esse modelo encontra uma relação de compromisso entre ***flexibilidade*** e ***grau de generalização***.
  - Essa aproximação será melhor quanto maior for o conjunto de treinamento e/ou menor o ruído.

Regressão polinomial: Qual ordem usar?

Polinômio de ordem 10.



Polinômio de ordem 20.



Polinômio de ordem 30.

- Polinômios com ordem maior do que 2 tendem a produzir ***aproximações perfeitas*** dos exemplos disponíveis, i.e., o modelo acaba ***memorizando*** os exemplos de treinamento.
- O erro para as amostras do conjunto de treinamento é muito baixo.
- Porém, essa aproximação se distancia bastante do modelo gerador.
- Portanto, esses modelos apresentarão ***erros significativamente maiores*** quando forem apresentados a exemplos de validação.
- Efeito conhecido como ***sobreajuste*** ou ***overfitting***: ***flexibilidade*** muito alta e ***grau de generalização*** muito baixo.

Resumo sobre subajuste e sobreajuste

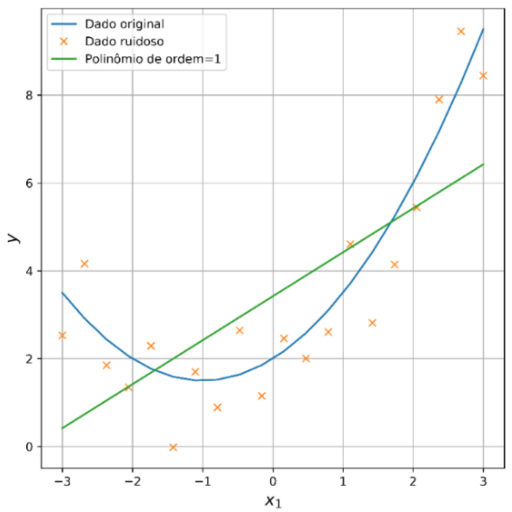
- **Subajuste**: situação em que o modelo falha em aproximar o ***mapeamento verdadeiro devido à falta de flexibilidade (ou capacidade)***.
  - Ocorre devido ao modelo não ter graus de liberdade suficientes para a aproximação.
  - O modelo produz ***erros significativos tanto quando apresentado ao próprio conjunto de treinamento quanto a dados inéditos***.
  - Se o modelo está subajustando, mesmo que o número de exemplos aumente indefinidamente, esta situação não vai desaparecer, é necessário ***aumentar a flexibilidade do modelo***, ou seja, no caso da regressão polinomial, sua ordem.

Resumo sobre subajuste e sobreajuste

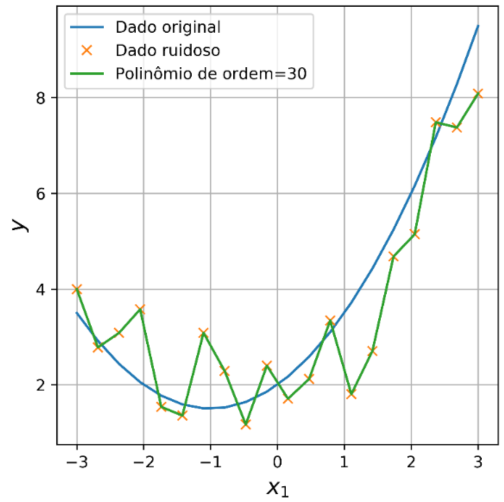
- **Sobreajuste**: situação em que o modelo se ajusta tão bem aos exemplos de treinamento que ele aprende até o ruído presente nos mesmos (baixo ***erro de treinamento***).
- Porém, o modelo produz erros significativos quando apresentado a dados inéditos (alto erro de ***erro de validação***).
  - Ocorre devido ao alto grau de flexibilidade do modelo.
  - Se o modelo está sobreajustando, então é necessário ***diminuir sua flexibilidade*** ou ***aumentar o conjunto de treinamento*** até que o erro de validação fique próximo do erro de treinamento.
- Nosso objetivo será encontrar um modelo que apresente uma relação de compromisso entre ***flexibilidade*** e ***capacidade de generalização***.
  - Flexibilidade suficiente para capturar o comportamento geral e generalizar bem.

Observações quanto à regressão polinomial

Polinômio de ordem 1



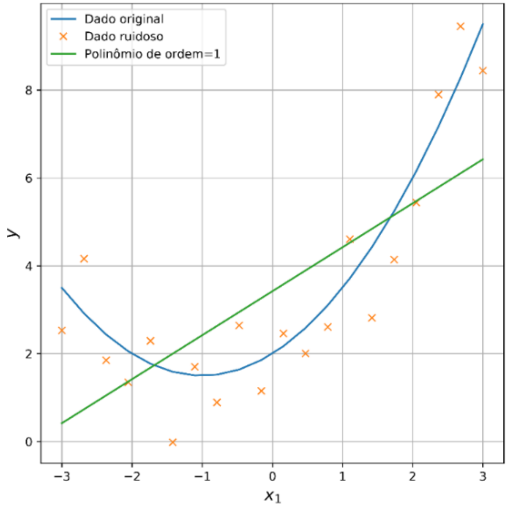
Polinômio de ordem 30



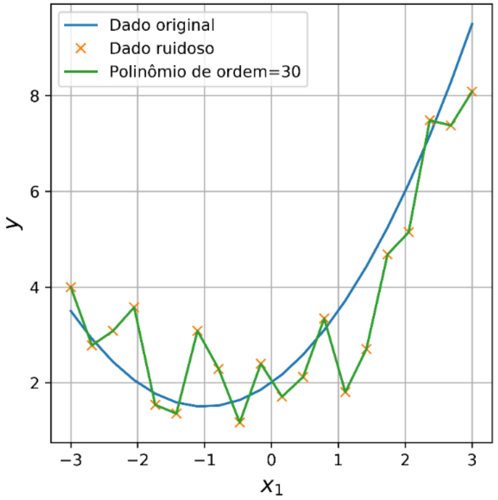
- O objetivo da regressão é encontrar um modelo com ***alta capacidade de generalização*** a ***partir do conjunto de treinamento dado***.
- Porém, vimos que ***polinômios de ordem inferior ao da função objetivo*** ***não conseguem aproximar*** o ***mapeamento verdadeiro***, pois eles ***não têm complexidade o suficiente*** para capturar a curvatura da função.

Observações quanto à regressão polinomial

Polinômio de ordem 1



Polinômio de ordem 30

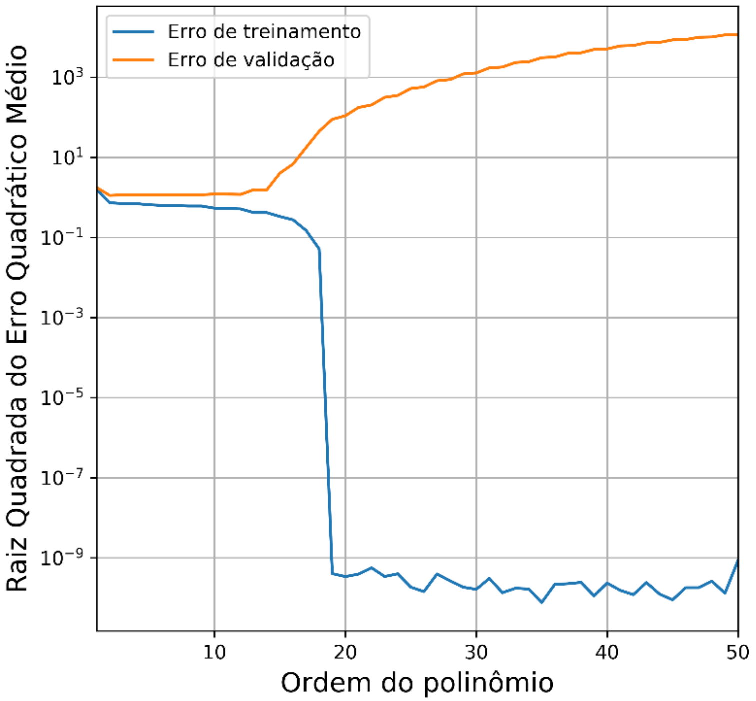


- Notamos também que é ***possível encontrar uma aproximação ótima com relação aos exemplos de treinamento*** que produz um mapeamento, $\hat{y} = h(\boldsymbol{x})$, que ***difere significativamente do mapeamento verdadeiro***.
- Portanto, precisamos encontrar ***estratégias*** que nos ***auxiliem a identificar*** um modelo que apresenta um bom ***balanço entre flexibilidade e capacidade de generalização***, se distanciando de modelos que ***subajustam*** ou ***sobreajustam***.

Observações quanto à regressão polinomial

- Uma estratégia simples e bastante usada para selecionar o modelo é ***comparar os erros de treinamento e validação***.
- Para isso, geralmente, dividimos o ***conjunto total de amostras*** em conjuntos de
  - ***Treinamento***: usado para treinar o modelo.
  - ***Validação***: usado para ajustar os hiperparâmetros do modelo (e.g., passo de aprendizagem, ordem).
  - ***Teste***: usado para avaliar a capacidade de generalização do modelo.
- OBS.: Esses conjuntos devem ter amostras diferentes, mas ***devem ser representativos do fenômeno sendo modelado***.

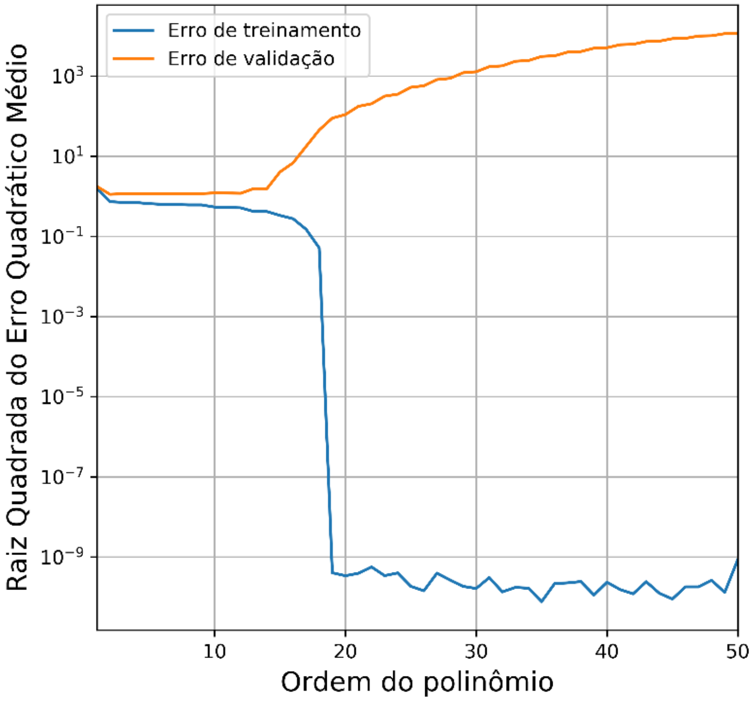
Erro de treinamento versus erro de validação



- Como vimos, o ***erro de treinamento*** diminui com o aumento da ordem do polinômio, porém, isso não significa que estamos obtendo um modelo melhor.
- Modelos mais complexos, com erro de treinamento pequeno, não necessariamente são modelos que ***generalizam*** melhor.
- Esses modelos tendem a apresentar alto ***erro de validação***.

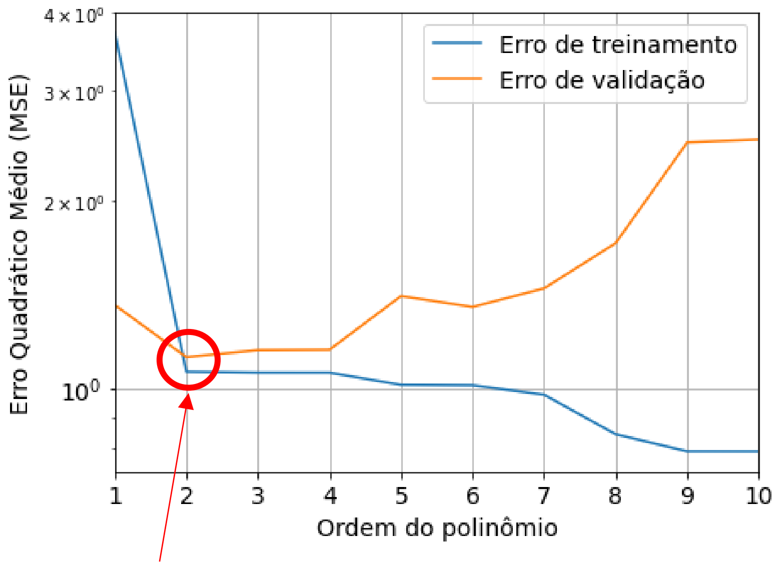
Exemplo: training_error_versus_validation_error.ipynb

Erro de treinamento versus erro de validação



- Já modelos com baixa complexidade apresentam alto erro de treinamento e de validação.
- Por outro lado, os ***modelos ideais*** mapeiam valores não vistos durante o treinamento em valores muito próximos aos esperados, i.e., generalizam.
- Portanto, modelos que ***generalizam bem*** apresentam ***erros de validação e de treinamento*** ***pequenos e próximos.***

Flexibilidade e generalização de um modelo



Modelo ótimo: relação de compromisso entre flexibilidade e capacidade de generalização.

- A ***flexibilidade (ou complexidade)*** de um modelo diz respeito à sua ***capacidade de aprender as regularidades ou características dos dados do conjunto de treinamento***.
  - Medida através do **erro de treinamento**.
- O ***grau de generalização*** de um modelo diz respeito a ***qualidade da aproximação gerada por ele quando exposto a dados não vistos durante o seu treinamento***.
  - Medido através do **erro de validação**.
- O ***modelo ideal*** é aquele que encontra uma ***relação de compromisso*** entre a ***flexibilidade*** e a ***capacidade de generalização***.

Para casa

- Façam o exercício 5 da lista #3.

Erros envolvidos no processo de aproximação

- O processo de aproximação (i.e., predição) envolve dois tipos de erros:
  - ***Redutível***;
  - ***Irredutível***.
- O erro ***redutível*** é dividido em erros de
  - Variância;
  - Viés;
  - Computação;
  - Amostragem.

Erros envolvidos no processo de aproximação

- **Erro de *variância* (ou estimação)**: é o erro devido à ***sensibilidade excessiva do modelo a variações nos dados de treinamento***, ou seja, esse erro ***mede o quanto o modelo varia com os dados de treinamento***.
  - Depende do nível de complexidade do modelo.
  - Um modelo com alto grau de complexidade (e.g., polinômio de alto grau) com relação à quantidade de amostras de treinamento apresenta ***alta variância*** e, portanto, se ***sobreajusta***.
  - Em outras palavras, a ***alta variância*** faz com que um modelo ***aprenda o ruído*** presente no conjunto de treinamento, resultando em ***baixo erro de treinamento e alto erro de validação***.

Erros envolvidos no processo de aproximação

- **Erro de viés (em inglês, bias)**: é o ***erro devido a suposições erradas feitas sobre os dados***, ou seja, sobre o formato do modelo gerador.
  - Também conhecido como ***erro de representação*** ou ***aproximação***.
  - Depende do nível de ***complexidade do modelo*** usado para a aproximação.
    - Um modelo com baixa complexidade provavelmente não irá capturar o comportamento do modelo gerador.
  - Modelos com ***alto viés*** tendem a ***subajustar*** aos dados de treinamento, perdendo relações importantes entre os atributos e os valores esperados.
  - Ou seja, um ***alto valor de viés*** leva a ***altos erros tanto de treinamento quanto de validação***.
  - Em outras palavras, o ***modelo é muito simples*** para representar a relação entre as variáveis de entrada e a saída.

Exemplo: variancia_e_vies.ipynb

Erros envolvidos no processo de aproximação

- **Erro de computação**: é o erro decorrente do fato de que nem sempre é possível explorar devidamente o espaço de hipóteses.
  - Também conhecido como **erro de otimização**.
  - **Motivos possíveis**: mínimos locais, limitação dos recursos computacionais para a busca/treinamento do modelo e uso de representação numérica de precisão finita (i.e., int8 e arredondamento).
    - Por exemplo, erros de arredondamento podem se acumular ao longo do tempo.
- **Erro de amostragem**: ocorre quando o ***conjunto de treinamento não é representativo da população de interesse***.
  - Leva a um modelo que ***não generaliza bem***.

Erros envolvidos no processo de aproximação

- **Erro irredutível**: é o erro devido ao ruído contido nos dados.
  - Também conhecido como ***erro de Bayes***.
  - Como o próprio nome diz, é o erro que ***não pode ser reduzido*** mesmo com modelos ideais (i.e., boa relação de compromisso entre erros de variância e de viés).
  - Por melhor que seja o modelo, os dados geralmente terão uma certa quantidade de ruído que não pode ser removida.

polynomial_regression_mse_big_data_set.ipynb
bias_variance_tradeoff.ipynb

Como avaliar e escolher a melhor hipótese?

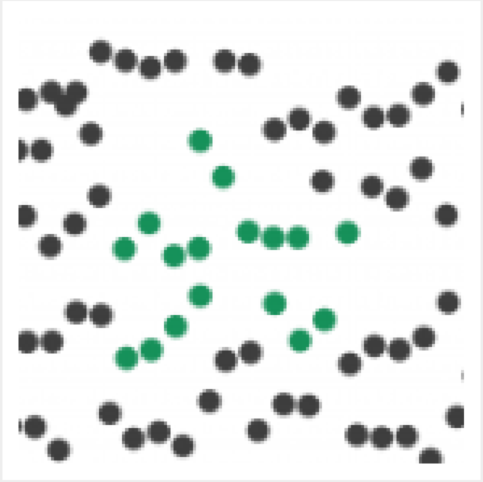
- Após todas essas informações, algumas perguntas surgem a respeito do modelo e seu treinamento.
- Como escolhemos a ***complexidade da função hipótese (i.e., modelo)*** quando não conhecemos o ***mapeamento verdadeiro***?
  - Ou seja, o ***quão complexo (i.e., flexível)*** deve ser o modelo?
- Como podemos dizer que o modelo está ***ajustando demais ou insuficientemente*** ao conjunto de treinamento?
- O quão bem o modelo prediz valores de saídas para entradas não vistas durante o treinamento (ou seja, ***o quão bem ele generaliza***)?
- ***Resposta***: podemos fazer isso através de ***técnicas de validação cruzada***.

Validação cruzada

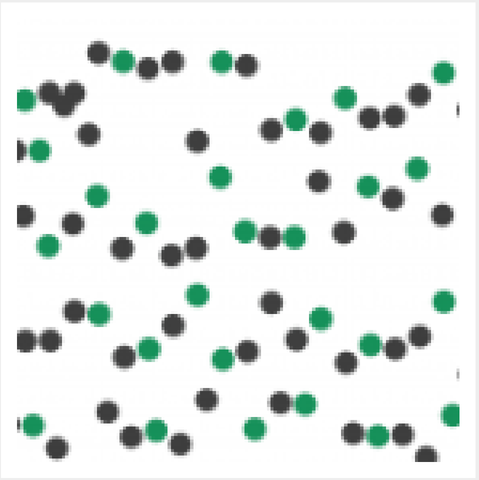
- Técnicas de ***validação cruzada*** podem ser usadas para se avaliar ***quantitativamente*** o ***sobreajuste*** e ***subajuste*** de um modelo, de forma a encontrar o ***modelo ótimo*** (i.e., ponto de compromisso entre os erros de viés e de variância).
  - ***OBS***.: *validação cruzada é usada para encontrar os valores ótimos de hiperparâmetros de modelos de aprendizado de máquina* (e.g., passo de aprendizagem, ordem do modelo, coeficiente de momentum, número de camadas e nós de uma rede neural, etc.).
- Na ***validação cruzada***, nós, geralmente, dividimos o conjunto total de exemplos em dois subconjuntos, o de ***treinamento*** e o de ***validação*** (algumas vezes chamado de conjunto de teste).
- O ***objetivo*** desta ***divisão*** é treinar o modelo com o conjunto de treinamento e avaliar sua ***capacidade em predizer*** valores de saída para dados que não foram utilizados durante o treinamento (i.e., com o conjunto de validação).
  - Ou seja, avaliamos sua capacidade de ***generalização*** com o conjunto de validação.

Estratégias para validação cruzada

- Existem diversas estratégias para validação cruzada, sendo as mais utilizadas: ***holdout***, ***k-fold*** e ***leave-p-out***.
  - Existem outras estratégias, mas elas podem ser vistas como variações de uma dessas três.
- Para aplicarmos estas estratégias, nós devemos nos assegurar que os ***dois conjuntos sejam suficientemente representativos do mapeamento que se pretende aproximar*** de modo que não ocorra o fenômeno conhecido como ***viés de seleção*.**
  - O ***viés de seleção*** é o erro introduzido pela seleção de amostras para análise de um modelo de tal forma que os conjuntos de amostras não sejam representativos da população que se pretende analisar.



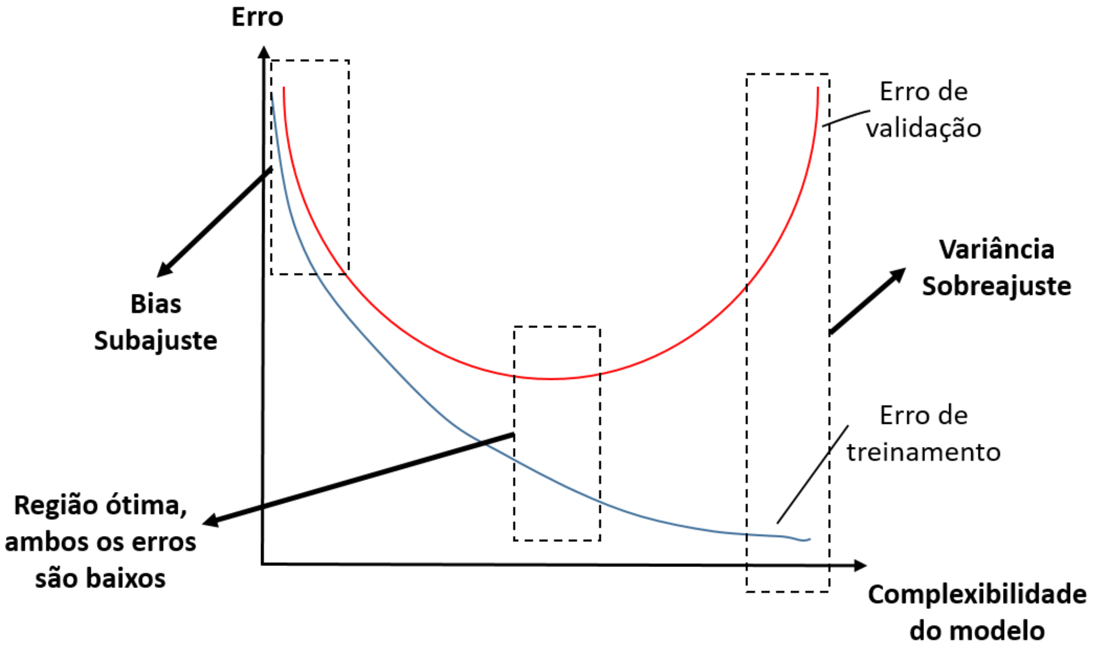
Viés de seleção: conjuntos não-representativos.



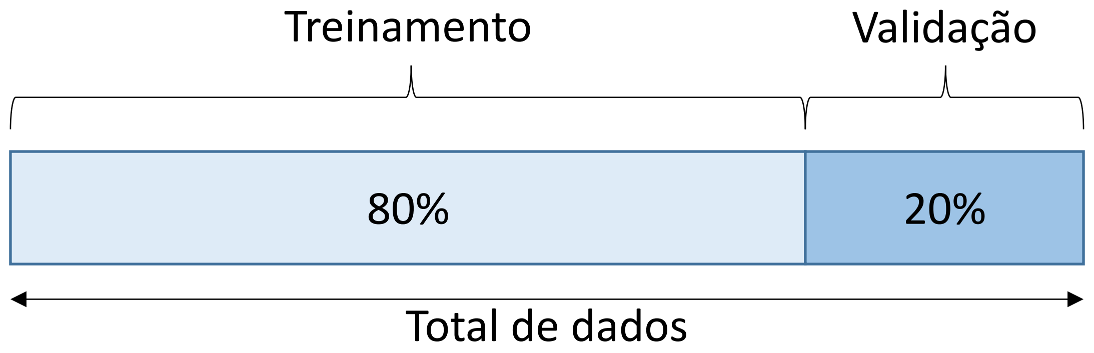
Amostragem aleatória correta: conjuntos representativos.

Comportamento "*típico*" dos erros

- A figura mostra o comportamento "***típico***" dos ***erros de treinamento e validação*** durante o processo de validação cruzada:
  - Aqui o objetivo é determinar a ***flexibilidade*** ótima do modelo.
- **Subajuste**: os erros de treinamento e validação são elevados (***viés alto***).
- **Sobreajuste**: o erro de treinamento é baixo, mas o erro de validação é alto (***variância alta***).
- **Região ótima**: ambos os erros são baixos (***relação de compromisso entre viés e variância***).



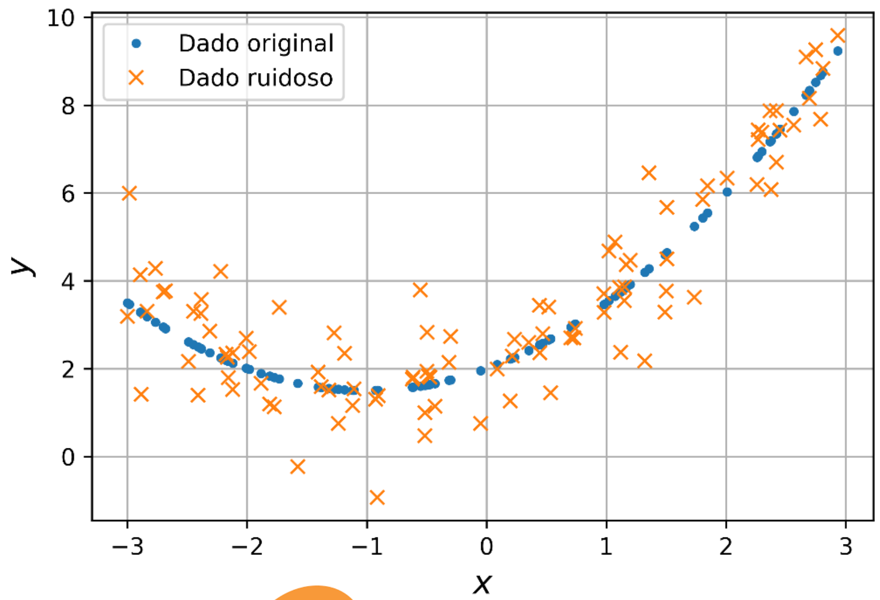
Holdout



- É a estratégia de validação cruzada ***mais simples*** entre as três e não acarreta em aumento da complexidade computacional, pois tem-se apenas ***um único par de conjuntos de treinamento e validação***.
- Nessa abordagem, divide-se ***aleatóriamente*** o conjunto de dados em ***p*** (%) para treinamento e ***(100 - p)*** (%) para validação.
- Normalmente divide-se o conjunto de dados em 70/80% para treinamento e 20/30% para validação.
- **Desvantagem**
  - Pode sofrer com o problema do ***viés de seleção***, o que, consequentemente, acarreta em ***alta variância***, pois a validação cruzada pode depender muito de quais amostras vão para o conjunto de treinamento e quais vão para o conjunto de validação.
  - Os resultados vão depender da divisão aleatória feita para os conjuntos de treinamento e validação.

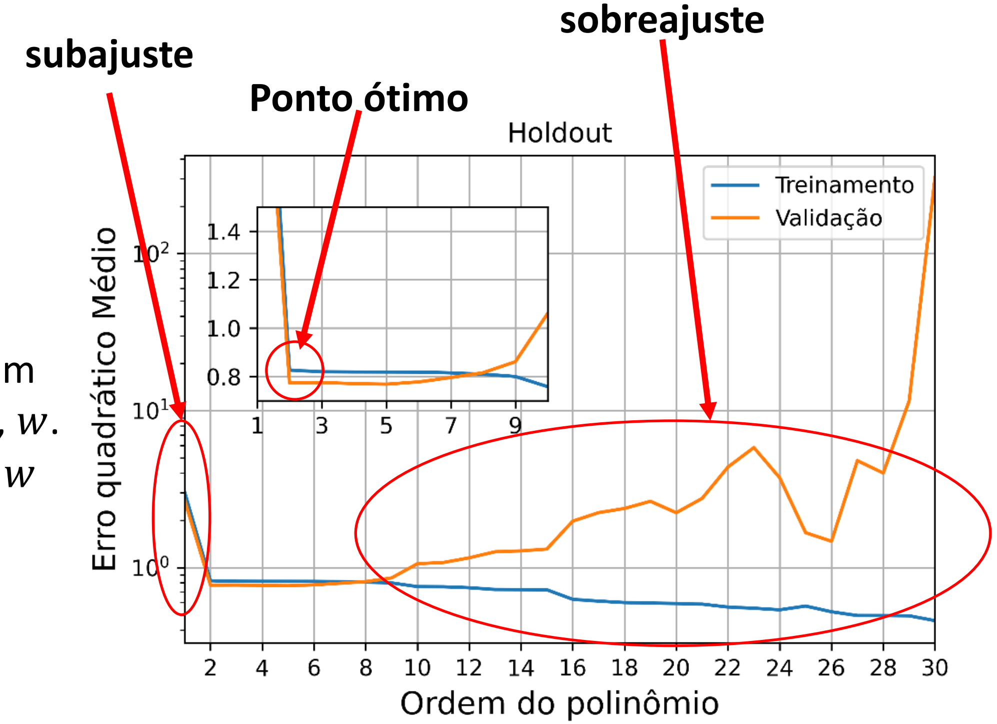
Exemplo: holdout_and_kfold_comparison_of_shuffling.ipynb

Holdout: Exemplo



Função observável é um polinômio de segunda ordem com ruído Gaussiano branco, w.

$$y_{noisy} = 2 + x + 0.5x^2 + w$$



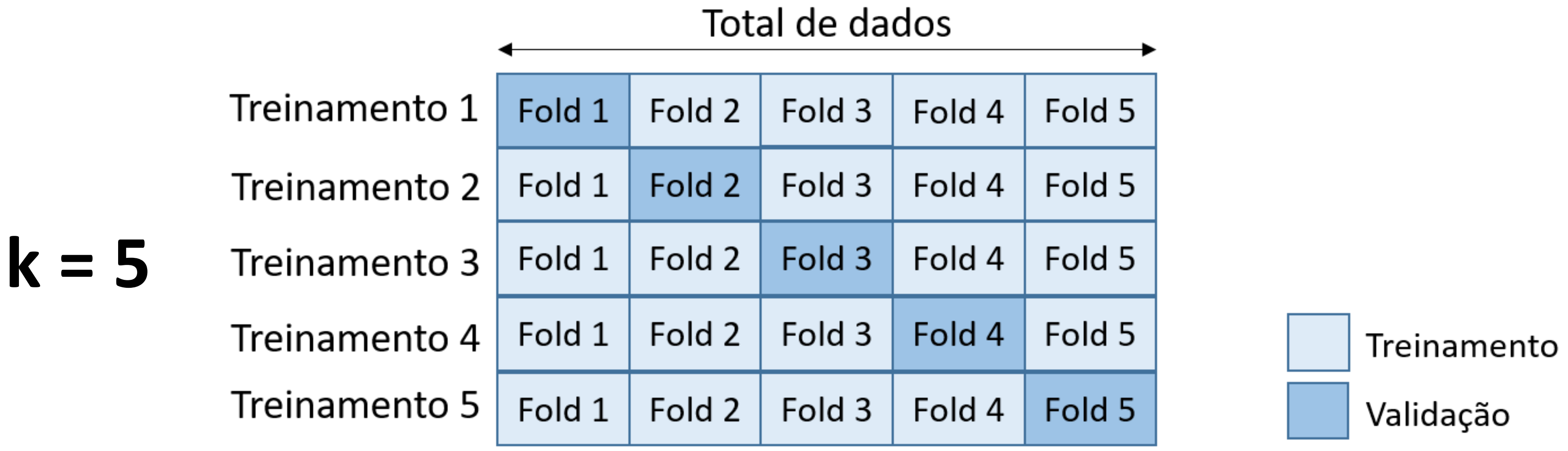

```
# Split the whole set into random training and validation set.
x_train, x_val, y_train, y_val = train_test_split(x, y_noisy, test_size=0.3, random_state=10)
```

- 70% para conjunto de treinamento e 30% para conjunto de validação.
- Tempo médio para execução com N = 100 é de aproximadamente 160 ms.
- Erro de treinamento ***diminui*** conforme a ordem do polinômio aumenta.
- Erro de validação ***aumenta*** conforme a ordem do polinômio aumenta.
- Qual ordem escolher?
  - O ponto onde ***ambos*** os erros sejam mínimos (balanço entre flexibilidade e grau de generalização).

[Exemplo: validacao_cruzada.ipynb]

k-Fold

- Estratégia mais elaborada que a do Holdout.
- Consiste em dividir o conjunto total de dados em **k** *folds* (partições) de tamanhos iguais (se possível) e realizar **k** treinamentos distintos, onde cada um dos **k** treinamentos considera **k-1** *folds* para treinamento e **1** *fold* para validação.



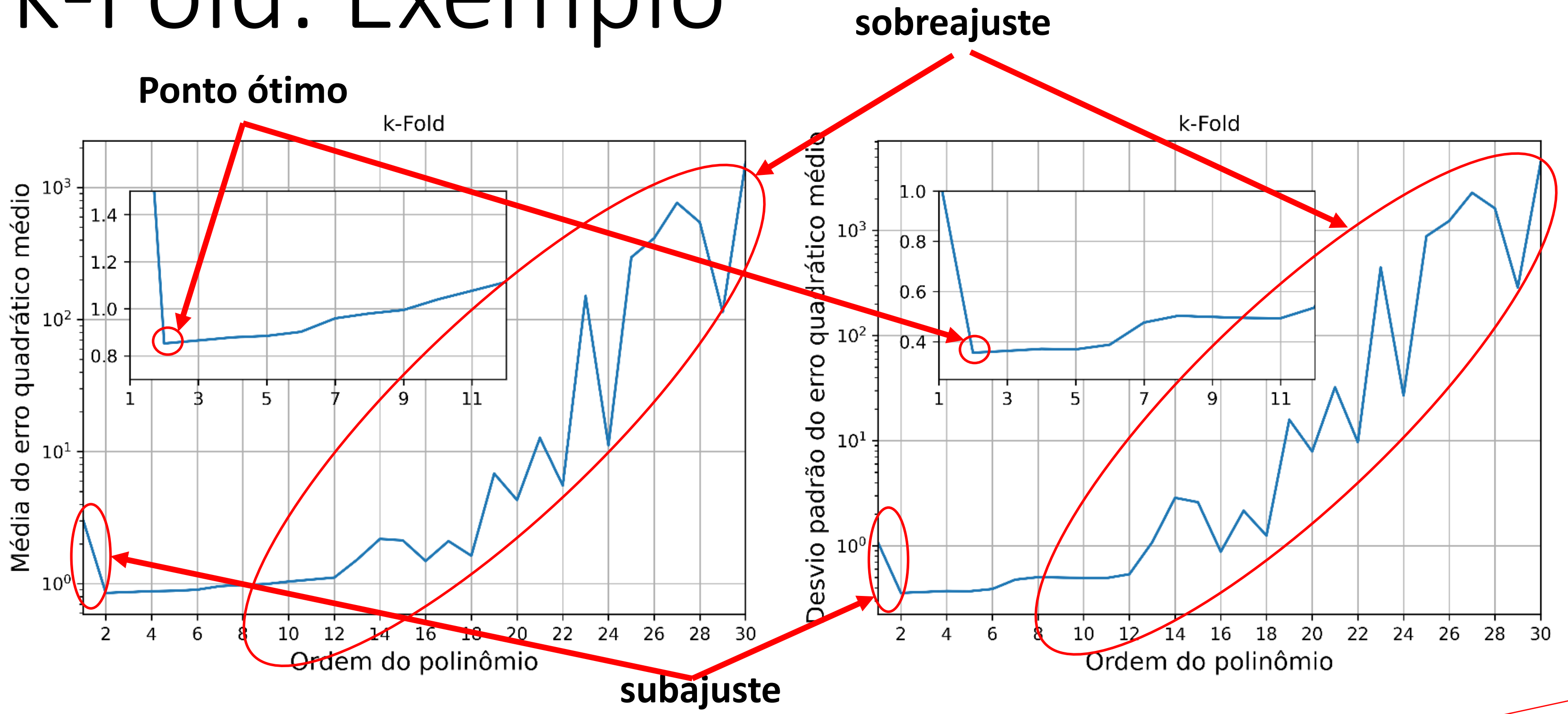
- Cada exemplo entra em um conjunto de validação exatamente **1** vez e em um conjunto de treinamento **k-1** vezes.
- O desempenho do modelo é dado pela ***média dos erros de validação*** calculados para cada um dos **k** *folds* de validação.

Exemplo: holdout_and_kfold_comparison_of_shuffling.ipynb

k-Fold

- Ajuda a reduzir o problema do ***viés de seleção*** em relação ao ***holdout***, pois não usamos apenas ***um par de conjuntos de treinamento e validação***.
- Ou seja, o k-Fold garante que ***todos os exemplos do conjunto de dados original apareçam nos conjuntos de treinamento e validação***, o que, consequentemente, reduz a ***variância*** do modelo.
    - A ***variância*** do modelo resultante é reduzida à medida que **k** aumenta.
    - Normalmente, utiliza-se **k** = 5 ou **k** = 10.
- Porém, tenham em mente que o valor de ***k deve ser escolhido de forma que cada conjunto de treinamento e validação seja grande o suficiente para ser estatisticamente representativo*** do conjunto de dados original.
- O k-Fold é bastante ***útil quando se tem conjuntos de dados pequenos a moderados***.
- **Desvantagem**
    - O treinamento deve ser executado novamente do zero **k** vezes, o que significa que leva-se **k** vezes mais tempo para se fazer a avaliação do modelo (treinamento + validação).

k-Fold: Exemplo



Conforme o modelo se ***sobreajusta*** aos dados de treinamento, sua variância aumenta, devido a redução de seu grau de generalização.

Em teoria, a variância deve ser muito baixa para modelos com alto grau de generalização.

- **k** = 10 folds: 10 iterações com 9 grupos para treinamento e 1 para teste.
- shuffle=**True**: os exemplos são "embaralhados" antes de dividí-los em **k** folds.
- ***cross_val_score*** recebe as instâncias de um modelo de regressão e de um de validação cruzada.
- ***Scoring***: quanto mais positivo, melhor.

```
kfold = KFold(n_splits=10, shuffle=True, random_state=100)
scores = cross_val_score(regressor, x, y_noisy, scoring='neg_mean_squared_error', cv=kfold)
```

- Usa-se a mesma função observável do exemplo anterior.
- Tempo médio para execução com N = 100 exemplos é de aproximadamente 1.9 s.
- Gráficos mostram a média e desvio padrão do MSE para as 10 etapas de treinamento/validação.
- Média e desvio padrão do MSE aumentam com a ordem do polinômio.
- Qual ordem escolher?
  - O ponto onde ***ambos***, média e desvio padrão do MSE, sejam mínimos.

Exemplo: validacao_cruzada.ipynb

Leave-p-out

- ***Valida*** um modelo usando ***todas as combinações possíveis*** de ***p*** exemplos como ***conjunto de validação*** e os ***N-p*** exemplos restantes como conjunto de treinamento.
- Para um conjunto de dados com N amostras, essa estratégia produz

$$\binom{N}{p} = \frac{N!}{p!(N-p)!},$$

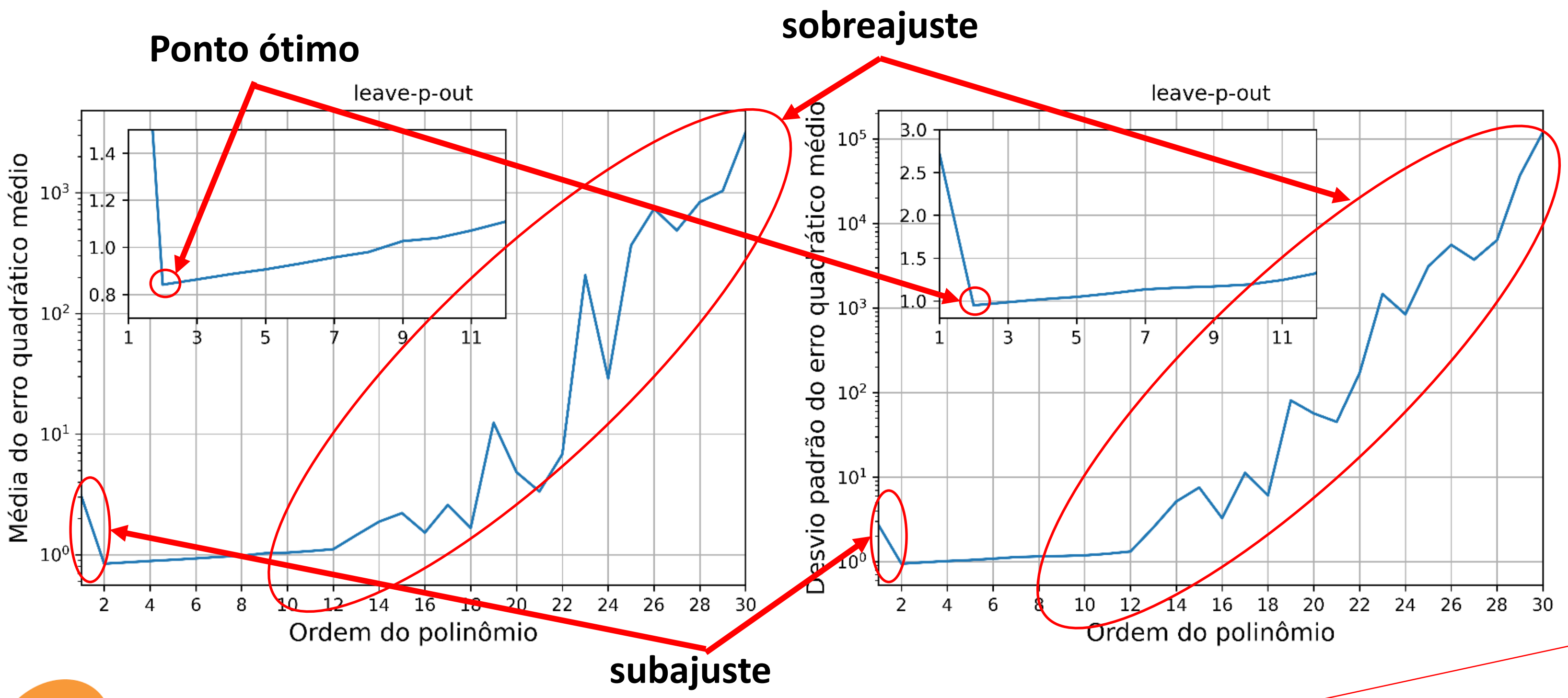
A equação mostra quantos subconjuntos distintos de p exemplos podemos criar a partir de N exemplos.

pares de conjuntos treinamento/teste, portanto, a complexidade computacional desta estratégia aumenta drasticamente com o aumento de ***p***. Exemplos para $N = 100$:

  - p = 1 -> 100 combinações
  - p = 2 -> 4.950 combinações
  - p = 5 -> 75.287.520 combinações

- Fornece ***estimativas de erro e desvio padrão mais precisas*** do que as abordagens anteriores, pois tem-se ***mais etapas de treinamento/validação***.
- **Desvantagem**
  - É uma ***estratégia exaustiva***, pois treina e valida o modelo para ***todas as combinações possíveis de p amostras*** e, para uma base de dados grande e um valor de ***p*** moderadamente grande, pode se tornar inviável computacionalmente.
- No caso do k-Fold, quando **k=N** (i.e., número ***folds*** igual ao número total de exemplos), então o k-Fold se torna equivalente à estratégia **leave-one-out**, ou seja, ***p*** = 1.

Leave-p-out: Exemplo



- Para ordem igual a 1, a média e desvio padrão são elevados: **subajuste**.
- Conforme a ordem aumenta, ambos diminuem, atingindo o **ponto ótimo** quando igual a 2.
- Porém, conforme a ordem continua a aumentar, ambos aumentam, indicando **sobreajuste**.

```
lpocv = LeavePOut(p=2)
scores = cross_val_score(regressor, x, y_noisy, scoring='neg_mean_squared_error', cv=lpocv)
```

- **p** = 2: 4950 combinações possíveis com 98 exemplos para treinamento e 2 para validação.
- ***cross_val_score*** recebe as instâncias de um modelo de regressão e de validação cruzada.
- ***Scoring***: quanto mais positivo, melhor.

- Usa-se a mesma função observável do exemplo anterior.
- Tempo médio para execução com N = 100 é de aproximadamente 810 [s] (+ de 13 [m]).
- Gráficos mostram a média e desvio padrão do MSE para as 4950 etapas de treinamento/validação.
- Média e desvio padrão do MSE aumentam com a ordem do polinômio.
- Qual ordem escolher?
  - O ponto onde ***ambos***, média e desvio padrão do MSE, sejam mínimos.

[Exemplo: validacao_cruzada.ipynb]

Qual estratégia utilizar?

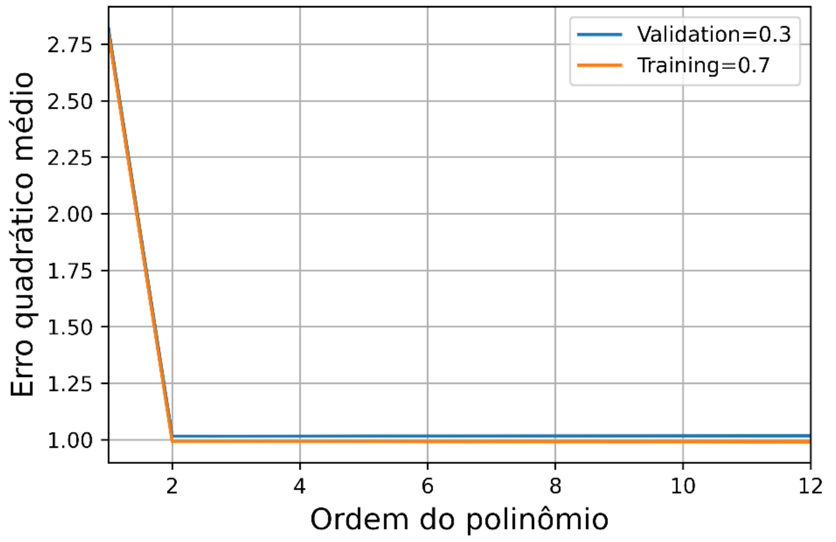
- A abordagem ***leave-p-out*** é a que ***dá indicações mais claras*** de qual ordem usar, ***pois usa um maior número de pares treinamento/validação***, aumentando a confiabilidade da média e do desvio padrão do MSE.
  - Porém, ela é bastante custosa em relação ao tempo necessário para se executá-lo, mesmo com uma base de 100 amostras leva-se mais de 13 minutos!
  - Portanto, deve-se utilizá-la com ***bases relativamente pequenas e pequenos valores para o parâmetro p***.
- Para bases maiores, o ***k-fold*** é uma opção melhor e mais eficiente do que o ***holdout*** e menos custosa do que o ***leave-p-out***.
- Para bases muito grandes, o ***holdout*** já daria boas indicações sobre qual ordem utilizar.
  - A probabilidade dos conjuntos de treinamento e validação obtidos com o ***holdout*** não serem ***representativos*** é inversamente proporcional ao tamanho do conjunto original.

[Exemplo: validacao_cruzada_base_maior.ipynb]

[Exemplo: holdout_com_bases_de_tamanhos_diferentes.ipynb]

Qual ordem escolher para o modelo?

- Qual ordem escolher ***se os erros de treinamento e validação são pequenos e praticamente constantes para várias ordens de polinômio?***
- Uma resposta é usar o ***princípio da navalha de Occam***.
- A ***navalha de Occam*** é um ***princípio lógico*** que postula que entre múltiplas explicações adequadas e possíveis para o mesmo conjunto de fatos, deve-se optar pela mais simples delas.
  - Ou seja, deve-se preferir explicações mais simples às mais complicadas/complexas.
- Portanto, usando a ***navalha de Occam*** escolhemos a função hipótese ***menos complexa (i.e., mais simples)***, ***mas que se ajusta bem aos dados***.



- Mesma função observável dos exemplos anteriores.
- Base de dados com **10000 exemplos**.
- Holdout com 30% para validação.
- Vejam que teoricamente, qualquer ordem maior ou igual a 2 já seria uma boa escolha.
- **Qual ordem escolher?**

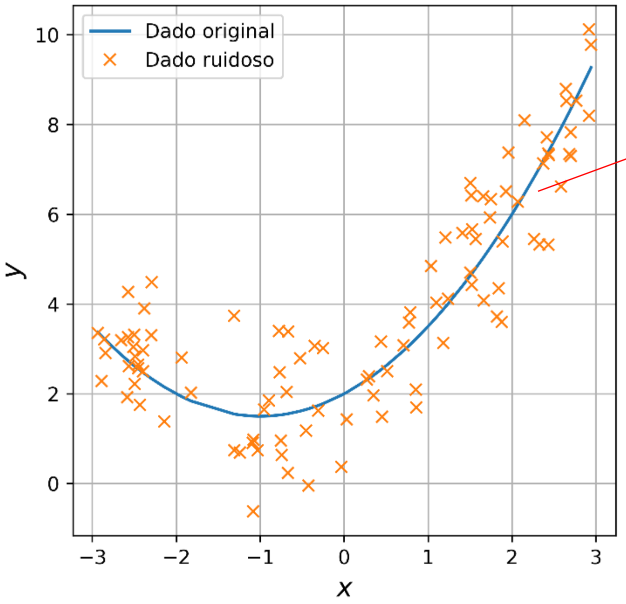
Curvas de Aprendizado

- ***Curvas de aprendizado***: são gráficos que ***comparam o desempenho do modelo nos conjuntos de treinamento e de validação em função do tamanho do conjunto de treinamento***.
  - **OBS.**: O tamanho do conjunto de validação permanece constante.
- São ferramentas úteis para ***avaliar o desempenho do modelo de aprendizado de máquina à medida que mais dados são fornecidos para o treinamento***.
- Essa comparação é normalmente usada para avaliar:
  - Qual é o ***melhor nível de complexidade de um modelo*** (no caso de polinômios, sua ordem). Em outras palavras, são usadas para determinar o ponto de equilíbrio entre os erros de variância (sobreajuste) e de viés (subajuste).
  - O quanto o ***modelo se beneficia de mais dados*** (por exemplo, se temos "dados suficientes" ou se o desempenho do modelo ficará melhor se aumentarmos a base de dados).
  - Podemos também usar as curvas para ***otimização dos hiperparâmetros*** do modelo.

Para casa

- Façam o exercício xxxx da lista #3.

Curvas de Aprendizado: Exemplo

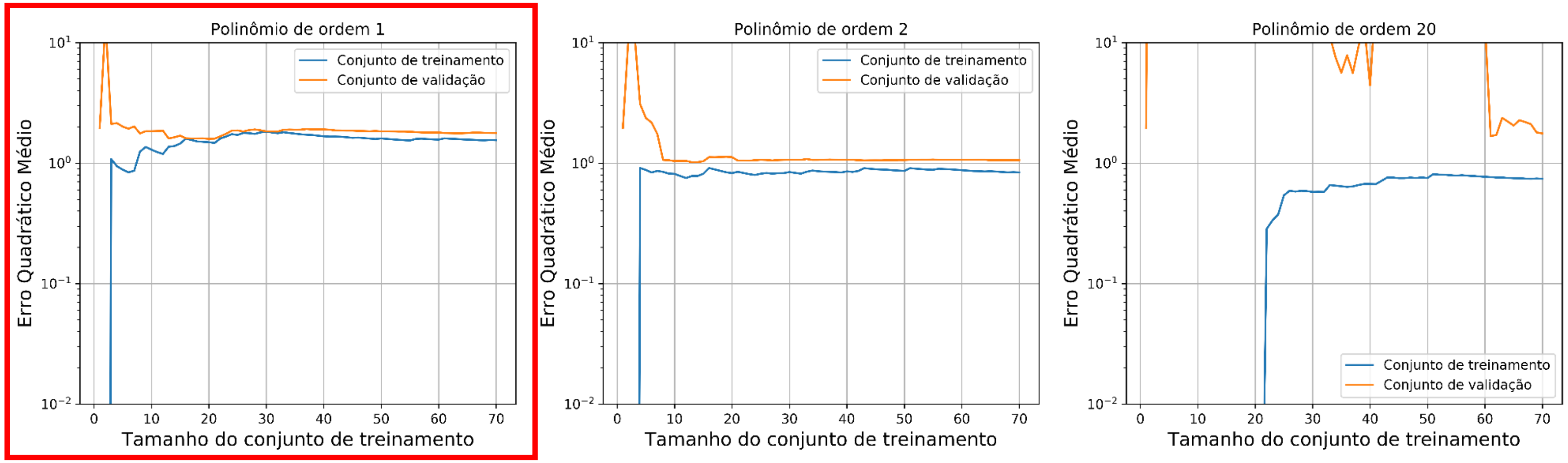


Polinômio de ordem 2.

- Caso não conhecêssemos o ***mapeamento verdadeiro*** dos dados ruidosos da figura acima, qual ordem de polinômio melhor aproximaria o ***mapeamento verdadeiro***?
  - Além da validação cruzada, podemos usar as chamadas curvas de aprendizado para encontrar a ordem do polinômio aproximador.

Exemplo: curvas_de_aprendizado.ipynb

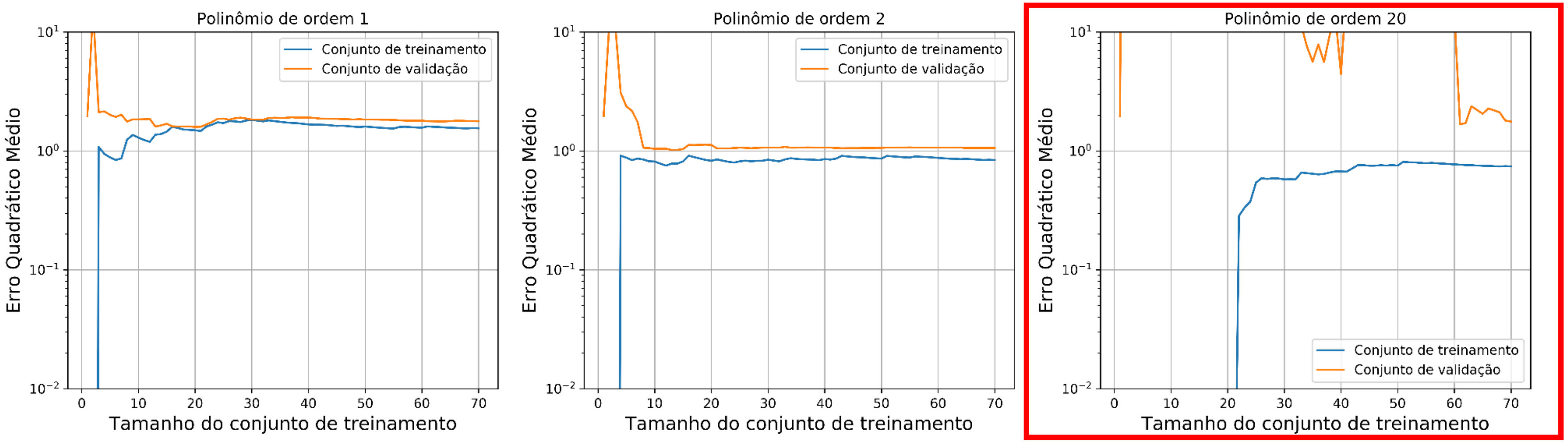
Curvas de Aprendizado: *Polinômio de ordem 1*



- Com 1 ou 2 exemplos no conjunto de treinamento, o modelo se ajusta perfeitamente (reta), porém, conforme o número de exemplos aumenta, é impossível para ele se ajustar (dados não-lineares e ruído).
- ***Erro nos exemplos de treinamento aumenta até atingir um platô e não diminui mesmo com o aumento do conjunto de treinamento***, pois o modelo não tem flexibilidade.
- ***Erro de validação é alto quando o modelo é treinado com poucos exemplos***, porém, ***diminui conforme o conjunto de treinamento aumenta***, terminando em um platô próximo do erro de treinamento.
- Essas curvas são típicas de um modelo que está ***subajustando*** (erros altos e próximos).
- **O que fazer?** Aumentar a ordem do modelo.

Exemplo: curvas_de_aprendizado.ipynb

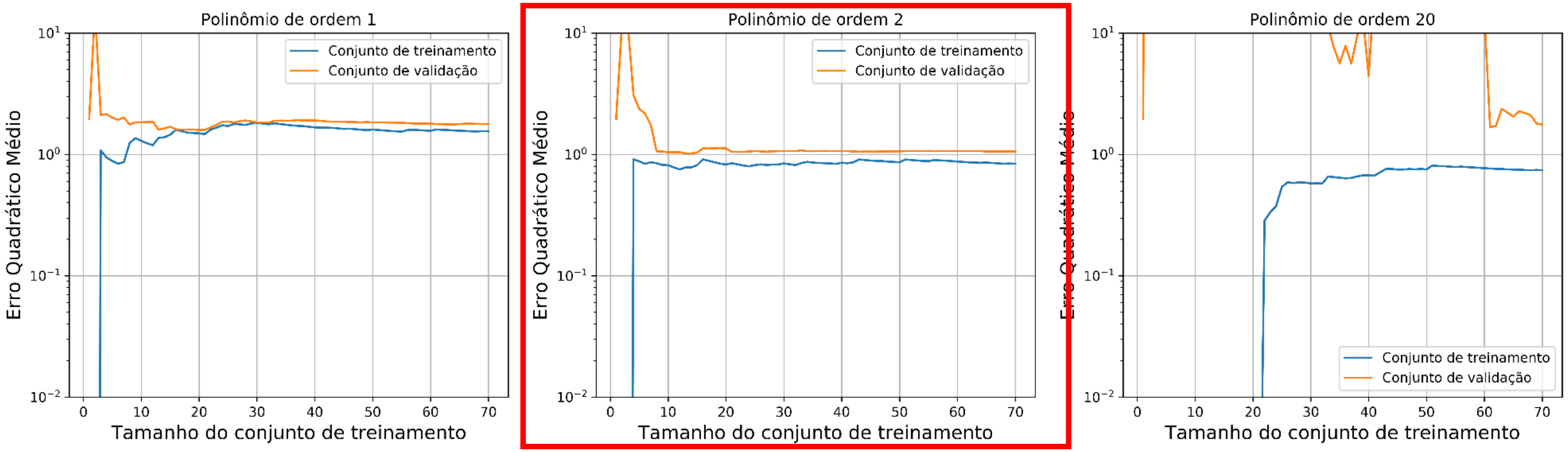
Curvas de Aprendizado: *Polinômio de ordem 20*



- O erro de treinamento é menor do que com o modelo com polinômio de ordem 1.
- Porém, há uma diferença considerável entre as curvas do erro de treinamento e validação.
- Isso significa que o modelo tem um desempenho melhor no conjunto de treinamento do que no conjunto de validação, indicando que ele está ***sobreajustando***.
- A performance do modelo melhora caso o conjunto de treinamento aumente.
  - Com um conjunto maior, a tendência é que ambos os erros convirjam para o MSE mínimo.
- **O que fazer?** Diminuir a ordem do modelo ou aumentar o conjunto de treinamento.

Exemplo: curvas_de_aprendizado.ipynb

Curvas de Aprendizado: *Polinômio de ordem 2*



- A diferença entre os erros diminui com o aumento do conjunto de treinamento se tornando pequena.
- Tanto o erro de treinamento quanto o de validação são menores do que com o modelo com polinômio de ordem 1 (i.e., reta).
- Isso é a indicação de um modelo que está se ajustando bem aos dados de treinamento (i.e., flexibilidade) e é capaz de generalizar bem para os dados de validação.
- Aumentar o conjunto de treinamento faz com que a diferença entre as duas curvas se torne ainda menor.
- **O que fazer?** Escolher esta ordem de polinômio.

Exemplo: curvas_de_aprendizado.ipynb

Regularização: penalizando a complexidade dos modelos

- Anteriormente, nós vimos como escolher o melhor modelo de regressão utilizando ***validação cruzada*** ou ***curvas de aprendizado***.
  - Em ambos os casos, escolhemos o modelo ***menos complexo***, ***mas que generaliza bem***.
- Uma abordagem alternativa é ***minimizar conjuntamente o erro e a complexidade da função hipótese***.
- Essa abordagem combina ***medidas de erro e de complexidade em uma única função de erro***, possibilitando que encontremos uma ***função hipótese*** com
  - a complexidade/flexibilidade ideal,
  - os pesos que minimizam o erro.
- Existem duas técnicas que seguem essa abordagem:
  - **Regularização**: ***penaliza*** funções hipótese muito complexas, ou seja, muito flexíveis.
  - **Early-stopping**: encerra o treinamento de ***algoritmos iterativos*** (e.g., gradiente descendente) quando o erro de validação for o menor possível (chamado de ***regularização temporal***).
- O objetivo das duas técnicas é deixar o modelo ***mais regular*** (i.e., menos complexo) de tal forma que ele ***não se sobreajuste*** ao conjunto de treinamento.

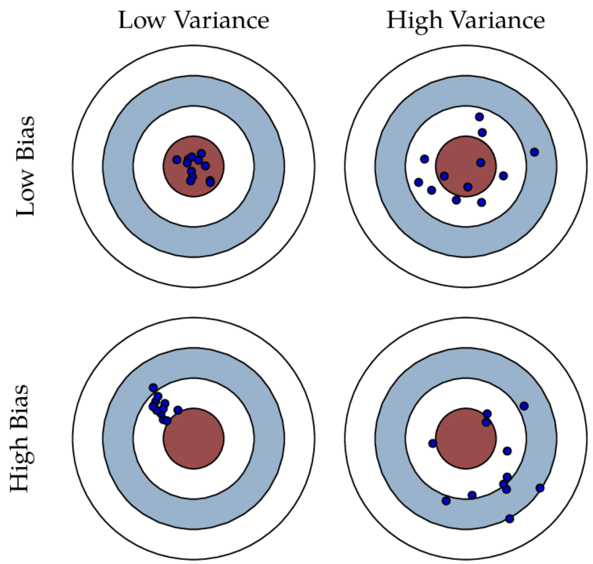
Regularização: penalizando a complexidade dos modelos

- Um claro sinal de um modelo que se ***sobreajustou aos dados de treinamento***, ou seja, um modelo ***flexível demais***, são ***pesos com magnitudes extremamente altas***.
- Portanto, a ideia por trás da ***regularização*** é ***restringir o aumento da magnitude dos pesos*** de forma a ***reduzir o risco de sobreajuste.***
  - Para tanto, incorpora-se ao ***processo de treinamento restrições*** proporcionais a alguma ***norma*** do ***vetor de pesos***.
- ***Técnicas de regularização reduzem o risco de sobreajuste*** do modelo ao conjunto de treinamento, ***aumentando sua capacidade de generalização***.
  - Quanto menos graus de liberdade o modelo tiver, mais difícil será para ele se ***sobreajustar*** aos dados de treinamento.
- A ***regularização*** força o algoritmo de aprendizado não apenas a capturar o ***comportamento do modelo gerador***, mas também a ***manter os pesos do modelo os menores possíveis***.

[Exemplo: absolute_value_of_weights_when_model_overfits.ipynb]

Regularização: penalizando a complexidade dos modelos

- Por ***restringir a magnitude dos pesos***, essas técnicas também são conhecidas como técnicas de ***shrinkage*** (i.e., redução ou encolhimento).
- Essas técnicas permitem diminuir a ***variância*** do modelo ao custo de introduzir algum ***viés***.
  - Encontrar uma ***relação de compromisso*** entre ***viés e variância*** permite que ***minimizemos o erro total do modelo***.
- **OBS.**: Deve-se aplicar o ***escalonamento de atributos*** quando se utiliza regularização.
  - A regularização pode ser influenciada pela escala dos atributos e, se eles não estiverem escalonados, alguns atributos podem acabar tendo mais peso do que outros, o que pode prejudicar o desempenho do modelo.
  - Escalonando, os pesos são penalizados de forma justa, pois garantimos que o modelo aprenda a importância relativa de cada atributo.
- As principais técnicas de regularização são: *Rigde*, LASSO e *elastic-net*.



Ridge regression

- Ao invés de minimizarmos apenas o ***erro quadrático médio***, como fizemos antes, introduzimos um ***termo de penalização (ou regularização)*** proporcional à ***norma Euclidiana*** (ou seja, a norma L2) do vetor de pesos:

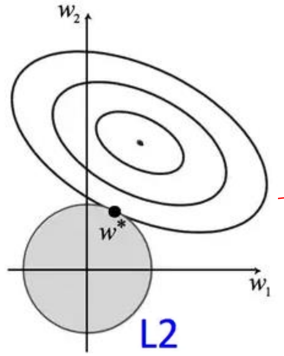
$$\min_{\boldsymbol{a} \in \mathbb{R}^{K+1\times 1}} \left(\|\boldsymbol{y} - \boldsymbol{\Phi a}\|^2 + \boxed{\lambda\|\boldsymbol{a}\|_2^2}\right),$$

Se $\|\boldsymbol{a}\|_2^2$ aumenta, λ deve diminuir para que o erro seja minimizado.

onde $\lambda \geq 0$ é o ***fator de regularização***, $\boldsymbol{\Phi}$ é a matriz de atributos, $\boldsymbol{a}$ é o vetor de pesos e $\|\boldsymbol{a}\|_2^2 = \sum_{i=1}^{K} {a_i}^2$ (***OBS***.: somatório inicia em 1 e não em 0).

- O aumento na ***flexibilidade*** de um modelo é representado pelo aumento da magnitude de seus pesos e, se quisermos minimizar a função de erro, essas magnitudes precisam ser restringidas.
- **OBS**.: o peso a_0 não é considerado no cálculo da ***norma L2***, pois a ***complexidade*** se deve apenas à ordem do modelo.
  - a_0 apenas dita o ***deslocamento*** em relação ao eixo das ordenadas e não influência na complexidade da ***função hipótese***, pois não é multiplicado por nenhum atributo.

Ridge regression



Região de factibilidade: possíveis valores que os pesos podem assumir. O **raio do círculo é inversamente proporcional ao fator de regularização, λ**.

- Podemos re-escrever o problema de regularização anterior como um ***problema de otimização com restrição*** da seguinte forma

$$\min_{\boldsymbol{a} \in \mathbb{R}^{K+1\times 1}} \|\boldsymbol{y} - \boldsymbol{\Phi a}\|^2$$
$$\text{s. } a. \ \|\boldsymbol{a}\|_2^2 \leq c,$$

onde c restringe a magnitude dos pesos (***raio da região de factibilidade***) e é inversamente proporcional à λ.

- Observem que
  - Conforme o valor de c diminui, menor poderá ser a magnitude dos pesos, até que no limite, quando $c \rightarrow 0$, então $a_i \rightarrow 0, \forall i$.
  - Por outro lado, conforme c aumenta, maior poderá ser a magnitude dos pesos, até que no limite, quando $c \rightarrow \infty$, então a_i pode assumir qualquer valor.
  - Portanto, o parâmetro c (e, consequentemente, λ) ***controla o compromisso entre a redução do erro de aproximação e a limitação da magnitude dos pesos***.

Ridge Regression

- A equação de ***erro regularizado***, $\|\boldsymbol{y} - \boldsymbol{\Phi a}\|^2 + \lambda\|\boldsymbol{a}\|_2^2$, ***continua sendo quadrática com relação aos pesos***, e, portanto, a ***superfície de erro continua sendo convexa***.
- Desta forma, podemos encontrar uma solução de ***forma fechada*** seguindo o mesmo procedimento que usamos para encontrar a ***equação normal***

$$\boldsymbol{a} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boxed{\lambda \boldsymbol{I}'})^{-1}\,\boldsymbol{\Phi}^T\boldsymbol{y}, \text{ onde } \boldsymbol{I}' = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

termo de regularização
=
derivada do termo de regularização

matriz identidade com o primeiro elemento igual a 0, pois não queremos regularizar o peso de bias.

- **Observações**
  - mesmo que a matriz $\boldsymbol{\Phi}$ não possua ***posto completo*** (i.e., matriz singular), a inversa na equação acima sempre existirá por conta da adição da ***derivada do termo de regularização*** à diagonal principal da matriz quadrada $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$.
  - como a ***norma L2*** é ***diferenciável***, a regressão *Ridge* também pode ser resolvida iterativamente através do ***algoritmo do gradiente descendente***.
  - o ***termo de regularização*** deve ser ***adicionado apenas à função de erro durante o treinamento***. Depois que o modelo é treinado, a avaliação do desempenho do modelo não utiliza a regularização.

Ridge regression com gradiente descendente

$$J_e(\boldsymbol{a}) = \frac{1}{N}\sum_{i=0}^{N-1}(y(i) - \hat{y}(i))^2 = \frac{1}{N}\sum_{i=0}^{N-1}\big(y(i) - h(\boldsymbol{x}(i), \boldsymbol{a})\big)^2$$

$$J_e(\boldsymbol{a}) = \frac{1}{N}\|\boldsymbol{y} - \boldsymbol{\Phi a}\|^2 + \lambda\|\boldsymbol{a}\|_2^2$$

$$\frac{\partial J_e(\boldsymbol{a})}{\partial a_k} = -\frac{2}{N}\sum_{i=0}^{N-1}[y(i) - \hat{y}(i)]\, x_k(i) + 2\lambda a_k\ ,\ \ k = 1, \ldots, K$$

$$\frac{\partial J_e(\boldsymbol{a})}{\partial a_0} = -\frac{2}{N}\sum_{i=0}^{N-1}[y(i) - \hat{y}(i)]\, x_0(i)\ ,$$

Gradiente com relação ao peso de bias

$$\frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}} = -\frac{2}{N}\boldsymbol{\Phi}^T(\boldsymbol{y} - \boldsymbol{\Phi a}) + 2\lambda \boldsymbol{I}'\boldsymbol{a},$$

Equação geral do vetor gradiente em formato matricial.

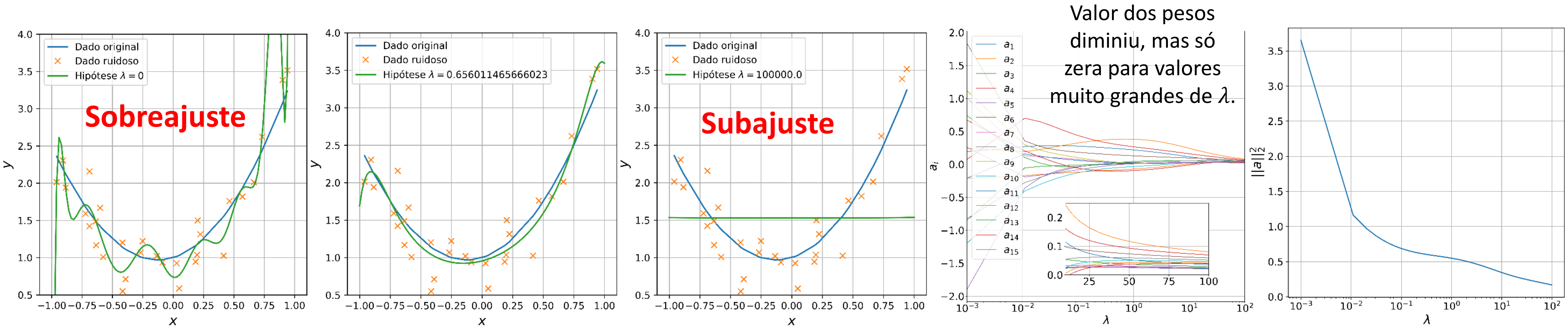
onde $\boldsymbol{I}'$ é uma ***matriz identidade*** de tamanho $K + 1$, onde o primeiro elemento é feito igual a 0, pois não queremos regularizar o peso de bias.

- A ***equação de atualização dos pesos*** é dada por

derivada do termo de regularização

$$\boldsymbol{a} = \boldsymbol{a} + 2\alpha\left[\frac{1}{N}\boldsymbol{\Phi}^T(\boldsymbol{y} - \boldsymbol{\Phi a}) - \boxed{\lambda \boldsymbol{I}'\boldsymbol{a}}\right].$$

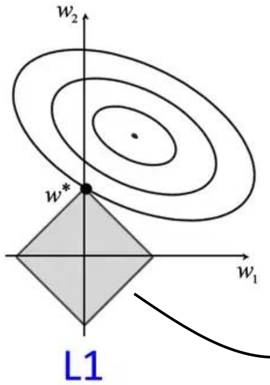
Ridge Regression: Exemplo



- Função hipótese polinomial de grau 15.
- Modelo treinado com 30 amostras geradas a partir de $y_{\text{noisy}} = 1 + 0.5x + 2x^2 + w$, onde $x \sim U(-1,1)$ e $w \sim N(0,1)$.
- Com $\lambda = 0$, regressão de Ridge se torna uma regressão polinomial sem regularização.
- Conforme λ aumenta, o modelo não se "*contorce*" tanto e passa a se ajustar aos dados de treinamento.
- Se λ continuar aumentando, todos os pesos acabarão muito próximos de zero e o resultado será uma linha reta que passa pela ***média dos dados de treinamento (i.e., o valor do peso de bias)***.
- **O aumento de λ leva a hipóteses menos complexas.** Isso reduz a variância do modelo, mas aumenta seu bias. Ou seja, ele tende a ***subajustar***.
- Conforme λ aumenta, os pesos e a norma L2 do vetor de pesos diminuem.
- Utiliza-se técnicas de ***validação cruzada*** para encontrar o valor ideal de λ.

Exemplo: ridge_regression.ipynb

LASSO regression



Região factibilidade da norma L1.

- A ***regressão LASSO*** (do inglês *Least Absolute Shrinkage and Selection Operator*) adiciona à função de erro um ***termo de penalização*** proporcional à **norma L1** do vetor de pesos.

$$\min_{\boldsymbol{a} \in \mathbb{R}^{K+1\times 1}} \left(\|\boldsymbol{y} - \boldsymbol{\Phi a}\|^2 + \boxed{\lambda\|\boldsymbol{a}\|_1}\right),$$

onde $\|\boldsymbol{a}\|_1 = \sum_{i=1}^{K}|a_i|$ e $\lambda \geq 0$ é o ***fator de regularização***.

- Podemos re-escrever o ***problema de regularização*** acima como um ***problema de otimização com restrição*** da seguinte forma

$$\min_{\boldsymbol{a} \in \mathbb{R}^{K+1\times 1}} \|\boldsymbol{y} - \boldsymbol{\Phi a}\|^2$$
$$\text{s. } a. \|\boldsymbol{a}\|_1 \leq c,$$

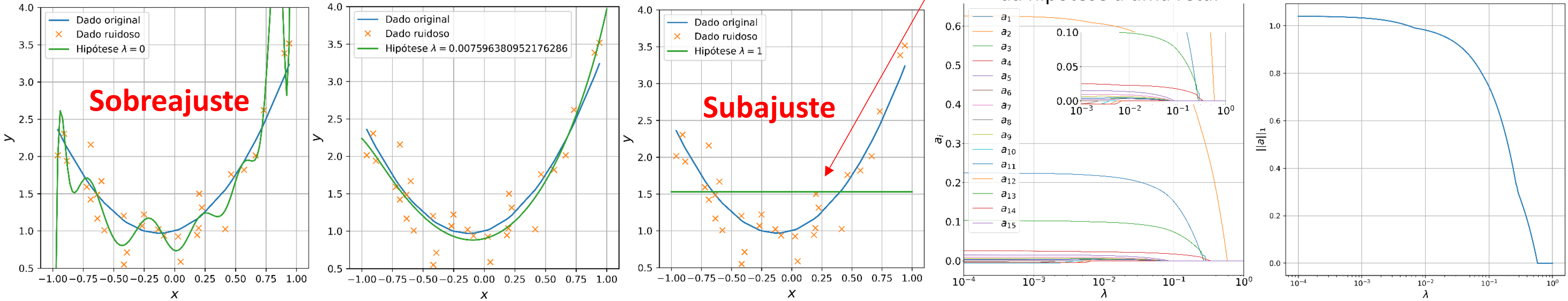
c restringe a ***área do quadrado*** e é igual a distância da origem até qualquer um dos vértices.

onde c restringe a magnitude dos pesos e é ***inversamente proporcional à λ.***

OBS.: Assim como na regressão de Ridge, a_0 também não faz parte do cálculo da norma.

LASSO Regression

Valor dos pesos se torna igual a zero, restringindo a flexibilidade da hipótese a uma reta.



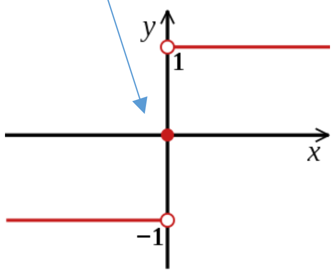
- Mesmas funções geradora e hipótese do exemplo anterior.
- Valores pequenos de λ fazem LASSO se comportar como regressão tradicional e valores muito grandes fazem os pesos serem anulados.
- A regularização com ***norma L1*** tem como ***vantagem*** a produção de ***modelos esparsos***.
  - Ou seja, vários elementos do vetor de pesos acabam sendo ***anulados***, indicando que os atributos correspondentes são irrelevantes para o processo de regressão.
- Isso sugere a ocorrência implícita de um processo de ***seleção automática de atributos***, que leva a ***modelos*** mais ***regulares***, ou seja, ***menos complexos***.
- **Desvantagem**: como a ***norma L1*** não possui derivada no ponto $a_i = 0, \forall i$, o problema da minimização ***não possui solução em forma fechada***, ***mas pode ser implementado com o GD***.
- Utiliza-se técnicas de validação cruzada para encontrar o valor ideal de λ.

Exemplo: lasso_regression.ipynb

LASSO regression com gradiente descendente

$$J_e(\boldsymbol{a}) = \frac{1}{N}\|\boldsymbol{y} - \boldsymbol{\Phi a}\|^2 + \lambda\|\boldsymbol{a}\|_1 = \frac{1}{N}\sum_{i=0}^{N-1}(y(i) - \hat{y}(i))^2 + \lambda\sum_{k=1}^{K}|a_k|$$

indeterminação



$$\frac{\partial |x|}{\partial x} = \mathrm{sign}(x), \text{para } x \neq 0.$$

$$\frac{\partial J_e(\boldsymbol{a})}{\partial a_k} = -\frac{2}{N}\sum_{i=0}^{N-1}[y(i) - \hat{y}(i)]\, x_k(i) + \lambda\, \mathrm{sign}(a_k)\,,\ \ k = 1, \dots, K$$

$$\frac{\partial J_e(\boldsymbol{a})}{\partial a_0} = -\frac{2}{N}\sum_{i=0}^{N-1}[y(i) - \hat{y}(i)]\, x_0(i)\,,$$

Gradiente com relação ao peso de bias

$$\frac{\partial J_e(\boldsymbol{a})}{\partial \boldsymbol{a}} = -\frac{2}{N}\boldsymbol{\Phi}^T(\boldsymbol{y} - \boldsymbol{\Phi a}) + \lambda \boldsymbol{I}'\mathrm{sign}(\boldsymbol{a}),$$

Equação geral do vetor gradiente em formato matricial.

onde $\boldsymbol{I}'$ é uma ***matriz identidade*** de tamanho $K + 1$, onde o primeiro elemento é feito igual a 0 e *a **função sign*** ou ***signum*** é uma função matemática ímpar que extrai o sinal de um número real.

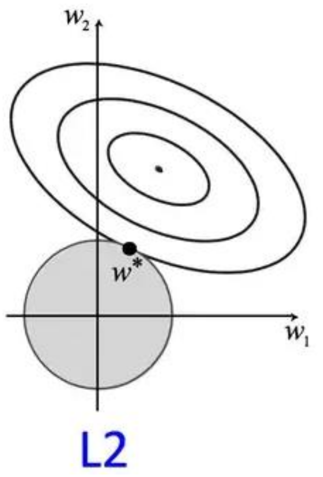
- A ***equação de atualização dos pesos*** é dada por

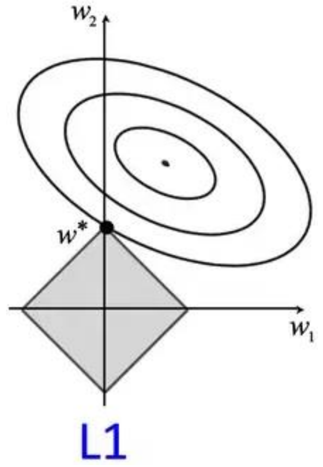
$$\boldsymbol{a} = \boldsymbol{a} + \alpha\left[\frac{2}{N}\boldsymbol{\Phi}^T(\boldsymbol{y} - \boldsymbol{\Phi a}) - \frac{\lambda}{2}\boldsymbol{I}'\mathrm{sign}(\boldsymbol{a})\right].$$

Implementações práticas consideram que sign(0) = 0.

Vantagem da regressão LASSO sobre Ridge

- A vantagem da regressão LASSO sobre a Ridge está na ***forma de quadrado da região de factibilidade*** criada pela penalização com a norma L1.
- Como veremos a seguir, ***região de factibilidade em forma de diamante*** leva à ***eliminação de alguns dos pesos*** (i.e., os pesos são zerados).
  - A regressão LASSO tende a produzir modelos esparsos.
- Os pesos que são zerados são aqueles ***correspondentes aos atributos que são menos relevantes para a predição do modelo*** (ou seja, que não contribuem para a predição).
- Isso pode ser útil para ***reduzir a complexidade do modelo*** e ***melhorar sua capacidade de generalização*** e ***deixá-lo mais interpretável***.
- Além disso, a regressão LASSO também pode ser usada para ***seleção de recursos***, onde os ***atributos mais relevantes são selecionados automaticamente e os outros descartados***.



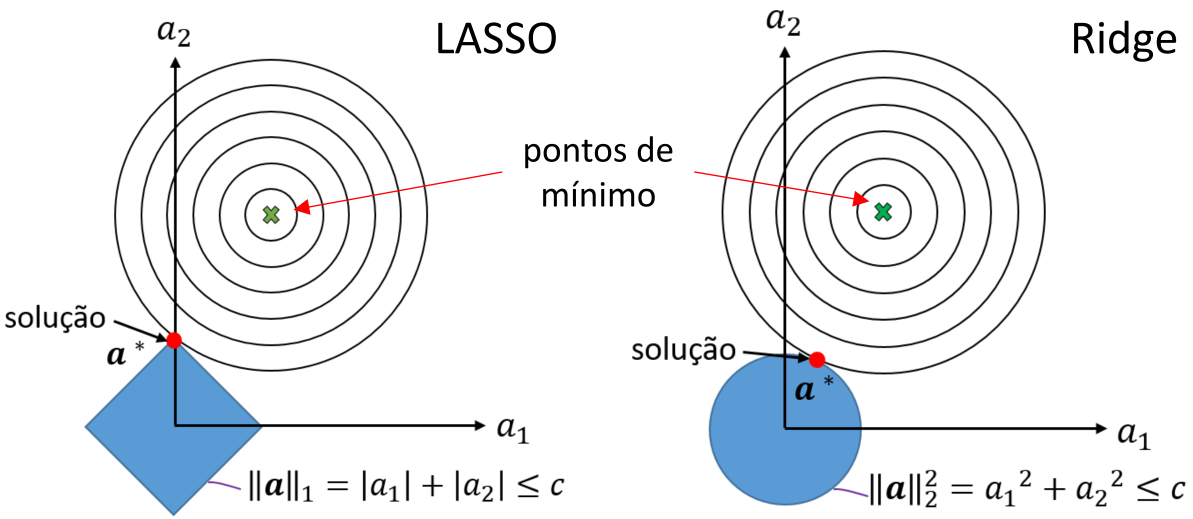


Vantagem da regressão LASSO sobre Ridge

- A ***regressão Ridge*** não apresenta esta característica, pois ela ***tende a manter os pesos não nulos*** (i.e., os pesos nunca são totalmente anulados), produzindo ***modelos com pesos pequenos e distribuídos ao longo dos atributos***.
- Podemos também entender a diferença entre as regressões Ridge e LASSO ao compreender que:
  - A norma L2 ***penaliza (i.e., encolhe) mais fortemente pesos com magnitudes grandes e penaliza menos fortemente pesos com magnitudes pequenas*** (devido ao quadrado na norma L2).
  - Já a norma L1 ***penaliza mais agressivamente todos os pesos***, o que significa que a ***norma L1 tende a reduzir os pesos para zero de forma mais uniforme*** do que a norma L2 (devido a usar apenas o valor absoluto dos pesos).

Por que a regressão LASSO produz modelos esparsos?

- O quadrado azul representa o conjunto de pontos $\boldsymbol{a}$ no espaço de pesos bidimensional que tenham norma L1 menor do que c.
- A solução deve estar dentro do quadrado, o mais próximo do mínimo.



- O círculo azul representa o conjunto de pontos $\boldsymbol{a}$ no espaço de pesos bidimensional que tenham norma L2 menor do que c.
- A solução deve estar dentro do círculo, o mais próximo do mínimo.

▪ A figura mostra as ***curvas de nível*** da ***função de erro*** de um problema de regressão linear com dois pesos (a_1 e a_2) e as regiões do ***espaço de hipóteses*** onde as restrições L1 e L2 são válidas.

▪ A solução para ambos os métodos corresponde ao ponto, dentro da ***região de factibilidade mais próximo do ponto de mínimo*** da função de erro.

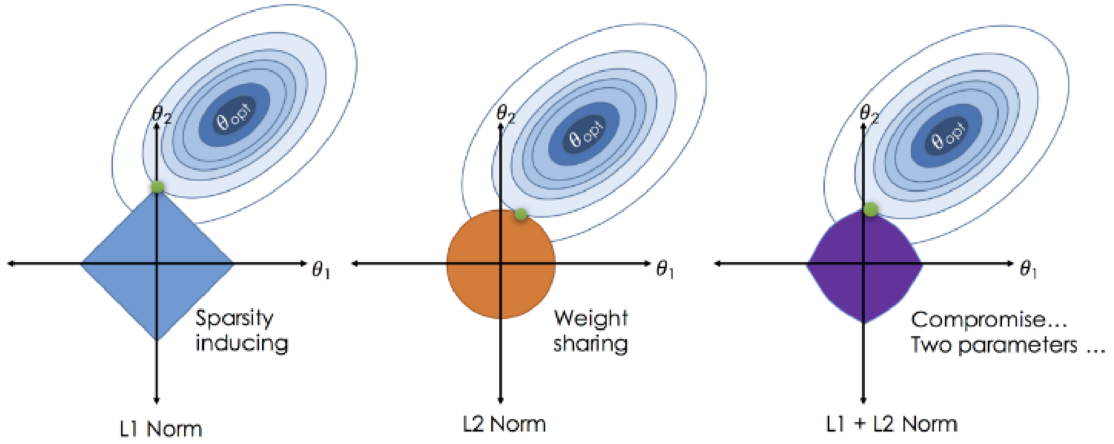
▪ É fácil ver que para uma posição arbitrária do mínimo, será comum que um ***vértice*** (ou cantos) do quadrado seja o ponto mais próximo do ponto de mínimo da função de erro.

▪ Os ***vértices*** na ***região de factibilidade*** da restrição L1 aumentam as chances de alguns pesos assumirem o valor zero, pois são eles que possuem valor igual a zero em alguma das dimensões (i.e., pesos).

Limitações

- Por não fazer ***seleção de atributos***, a regressão Ridge resulta em um modelo:
  - ***Com baixa interpretabilidade***: não se consegue determinar quais atributos são e não são importantes para a predição.
  - ***Complexo***: por ***manter todos os pesos no modelo, necessita de uma maior quantidade de cálculos*** para realizar predições, se tornando computacionalmente intensiva quando se lida com um grande número de atributos.
- A regressão LASSO:
  - ***Pode não selecionar o melhor atributo quando há forte correlação positiva*** entre dois ou mais atributos.
  - Se o ***fator de regularização não for ideal***, pode levar a um ***modelo muito simples*** que não inclui todas os atributos importantes (devido à seleção de atributos).
  - ***Numericamente instável*** para valores pequenos do ***fator de regularização, principalmente para casos subdeterminados, i.e., K > N.***
- Nesses casos, a regressão Elastic-Net é mais indicada, pois minimiza as limitações das duas regressões.

Elastic-Net



O hiperparâmetro κ dita a relação de compromisso entre as duas regularizações.

- ***Elastic-net*** é uma regularização que combina as regressões Ridge e LASSO de forma a ***resolver as limitações de ambas***.
- Realiza ***seleção automática de atributos*** e ***regularização*** simultaneamente.
- Nada mais é do que uma ***combinação linear*** entre as penalizações baseadas nas normas L1 e L2 do vetor de pesos.

$$\min_{\boldsymbol{a} \in \mathbb{R}^{K+1 \times 1}} \left(\|\boldsymbol{y} - \boldsymbol{\Phi a}\|^2 + \lambda \left[\kappa \|\boldsymbol{a}\|_1 + (1-\kappa)\|\boldsymbol{a}\|_2^2 \right] \right),$$

onde $\kappa \,\epsilon\, [0, 1]$ é o ***fator de elasticidade*** entre as duas normas, ou seja, estabelece uma ***relação de compromisso entre as duas normas***.

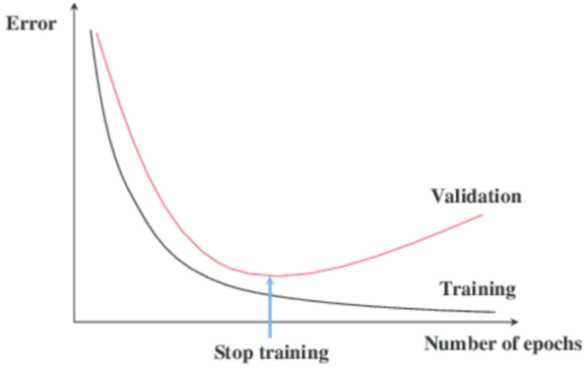
- Quando κ = 0, é equivalente à regressão Ridge, e quando κ = 1, ela é equivalente a regressão LASSO.
- A seleção dos hiperparâmetros κ e λ pode ser feita por meio de ***validação cruzada***.

Exemplo: elastic_net_regression.ipynb

Quando utilizar cada tipo de regressão?

- **Regressão Ridge**: é um bom começo. No entanto, se você suspeitar que apenas alguns atributos são realmente úteis, você deve preferir LASSO ou Elastic-Net.
- **Regressão LASSO**: boa para ***seleção automática de atributos***.
  - No entanto, pode se comportar erraticamente se o ***número de atributos, K, for maior que o número de exemplos de treinamento, N***. Nesse caso, ele ***encontra no máximo N pesos diferentes de zero***, mesmo que os *K* atributos sejam relevantes.
  - Se houverem ***atributos fortemente correlacionados entre si***, ela ***seleciona um deles aleatoriamente e ignora os demais***, o que não é bom para a interpretação do modelo.
  - Nestes casos, deve-se usar a regressão Elastic-Net.
- **Regressão Elastic-Net**: é mais versátil que as anteriores, pois o fator de elasticidade, κ, pode ser ajustado de forma a encontrar uma relação de compromisso entre as normas L1 e L2.
  - Quando há vários atributos correlacionados entre si, a regressão LASSO provavelmente escolherá um deles aleatoriamente, enquanto o Elastic-Net provavelmente ***escolherá todos***, facilitando a ***interpretação do modelo***.
  - Uma proporção de 50% (i.e., $\kappa = 0.5$) entre as penalizações L1 e L2 é uma boa escolha inicial para esse parâmetro.

Early-stopping: Parada antecipada



- O algoritmo do ***gradiente descendente*** tende a aprender modelos cada vez mais ***complexos*** à medida que o número de épocas aumenta.
  - Ou seja, ele se ***sobreajusta*** ao conjunto de treinamento ao longo do tempo.
- Uma forma de se ***regularizar*** algoritmos de ***aprendizado iterativo***, como o ***gradiente descendente***, é interromper seu treinamento assim que o ***erro de validação*** comece a crescer sistematicamente.
- Essa abordagem é chamada de ***early-stopping*** e pode ser vista como uma ***regularização temporal***.
- Assim como as outras abordagens, ela tem o objetivo de evitar o ***sobreajuste*** de um modelo.
- Ao se ***regularizar no tempo***, a complexidade do modelo pode ser controlada, melhorando sua ***generalização***.
- Mas como saber quando interromper o treinamento?
  - Ou seja, qual é o critério de parada?

Early-stopping: critério de parada

- Existem duas estratégias para se definir o critério de parada:
  - Interromper o treinamento quando o valor do ***erro de validação*** aumentar por ***P*** épocas sucessivas, sendo o parâmetro ***P*** chamado de ***paciência***.
    - **Problema**: como o erro de validação pode oscilar bastante (e.g., SGD) e apresentar um comportamento pouco previsível, nem sempre é fácil se desenvolver detectores automáticos de mínimos e encerrar o treinamento.
  - Outra estratégia é permitir que o treinamento prossiga por um determinado número de épocas, mas sempre armazenando os pesos associados ao ***menor erro de validação***. Ao final do treinamento, os ***pesos associados ao menor erro de validação são considerados*** para realizar predições com o modelo.

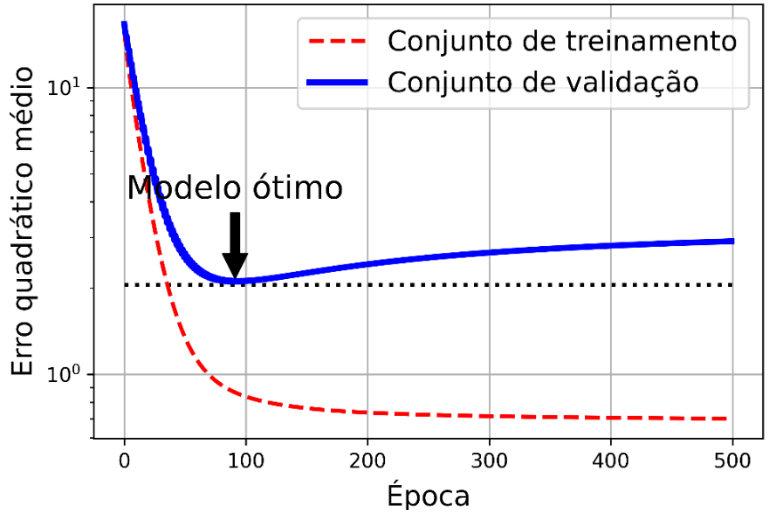
Early-stopping: exemplo

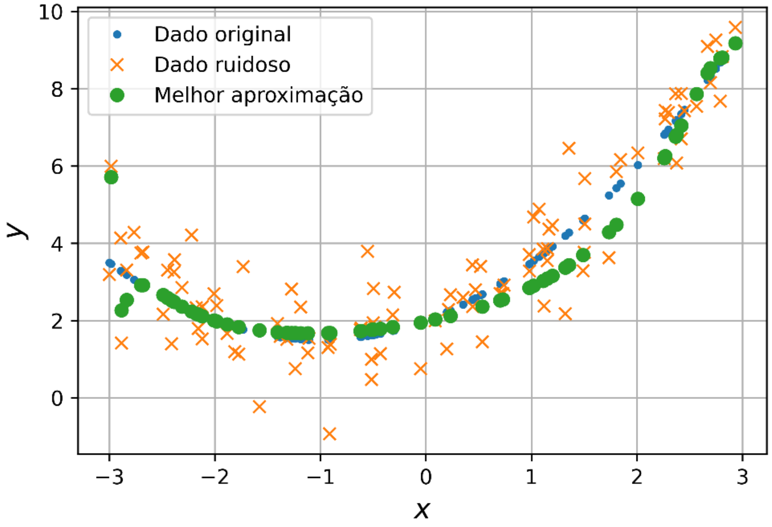
- A figura mostra os erros de treinamento e validação de um modelo de regressão polinomial com grau igual a 90 treinado usando-se o GDE com apenas 70 amostras.
- A função observável é dada por

$$y_{\text{noisy}} = \boxed{2 + x + 0.5x^2} + w,$$

onde $x \sim U(-3,3)$ e $w \sim N(0,1)$. Função verdadeira

- A ***ordem do modelo é muito maior do que a ordem da função verdadeira*** (alta flexibilidade), além de ser maior do que o número de amostras de treinamento (sobreajuste).
- À medida que as épocas passam, o algoritmo aprende e seu erro de treinamento diminui, juntamente com o erro de validação.
- No entanto, após aproximadamente 100 épocas, o erro de validação para de diminuir e começa a crescer.
- Isso indica que o modelo começou a ***sobreajustar*** aos dados de treinamento.
- Com a parada antecipada, usa-se os pesos que resultaram no menor erro de validação ao longo de todo o treinamento, garantindo que o modelo apresente uma boa ***generalização***.

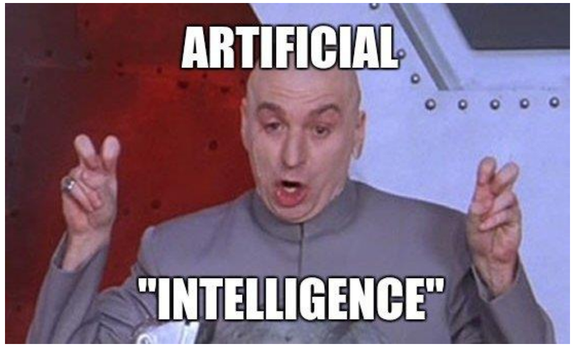


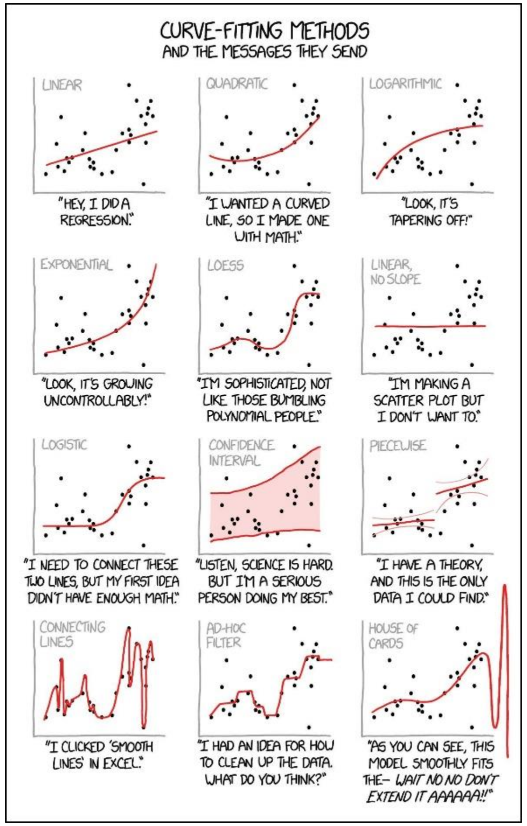


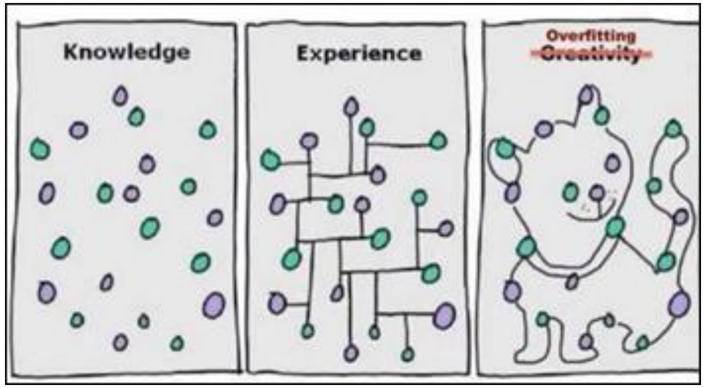
Exemplo: early_stopping.ipynb

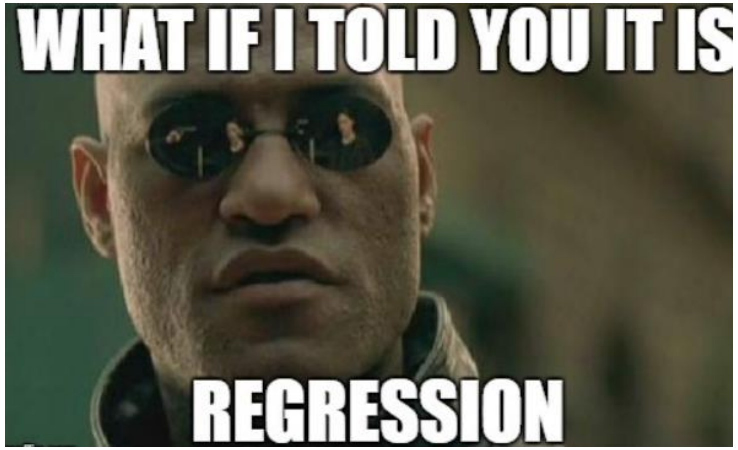
Obrigado!


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS


ARTIFICIAL
"INTELLIGENCE"


CURVE-FITTING METHODS
AND THE MESSAGES THEY SEND
LINEAR
"HEY, I DID A REGRESSION."
QUADRATIC
"I WANTED A CURVED LINE, SO I MADE ONE WITH MATH."
LOGARITHMIC
"LOOK, IT'S TAPERING OFF!"
EXPONENTIAL
"LOOK, IT'S GROWING UNCONTROLLABLY!"
LOESS
"I'M SOPHISTICATED, NOT LIKE THOSE BUMBLING POLYNOMIAL PEOPLE."
LINEAR, NO SLOPE
"I'M MAKING A SCATTER PLOT BUT I DON'T WANT TO."
LOGISTIC
"I NEED TO CONNECT THESE TWO LINES, BUT MY FIRST IDEA DIDN'T HAVE ENOUGH MATH."
CONFIDENCE INTERVAL
"LISTEN, SCIENCE IS HARD. BUT I'M A SERIOUS PERSON DOING MY BEST."
PIECEWISE
"I HAVE A THEORY, AND THIS IS THE ONLY DATA I COULD FIND."
CONNECTING LINES
"I CLICKED 'SMOOTH LINES' IN EXCEL."
AD-HOC FILTER
"I HAD AN IDEA FOR HOW TO CLEAN UP THE DATA. WHAT DO YOU THINK?"
HOUSE OF CARDS
"AS YOU CAN SEE, THIS MODEL SMOOTHLY FITS THE- WAIT NO NO DON'T EXTEND IT AAAAAA!!"

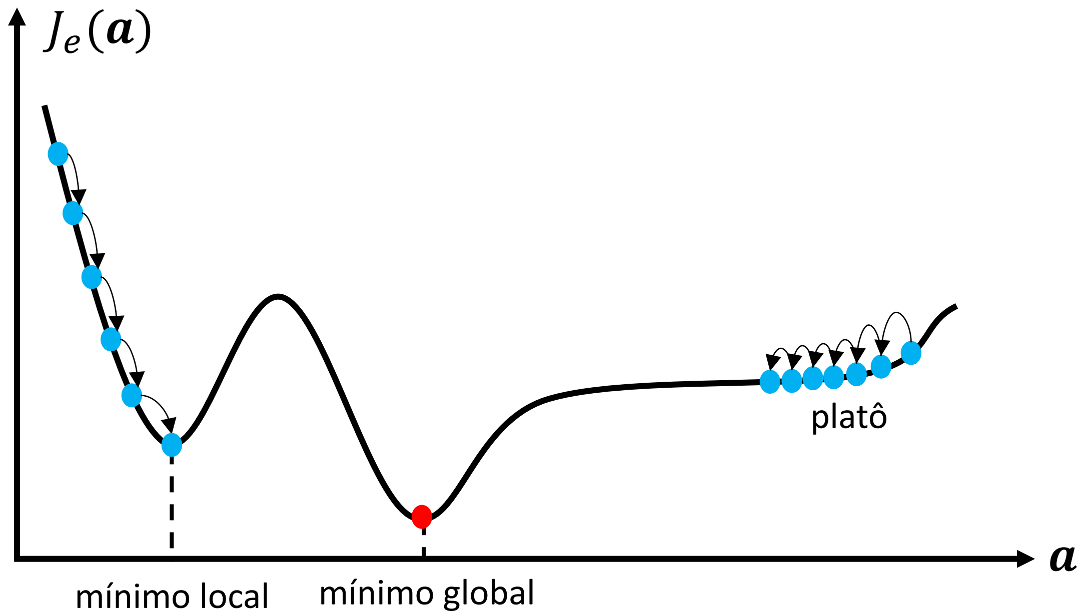

Knowledge
Experience
Overfitting
~~Creativity~~

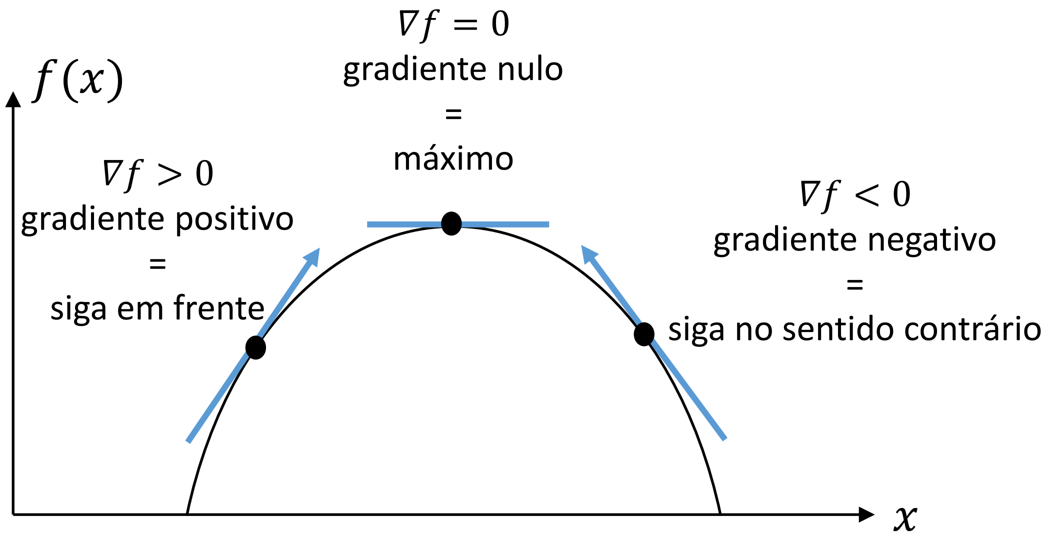

WHAT IF I TOLD YOU IT IS
REGRESSION

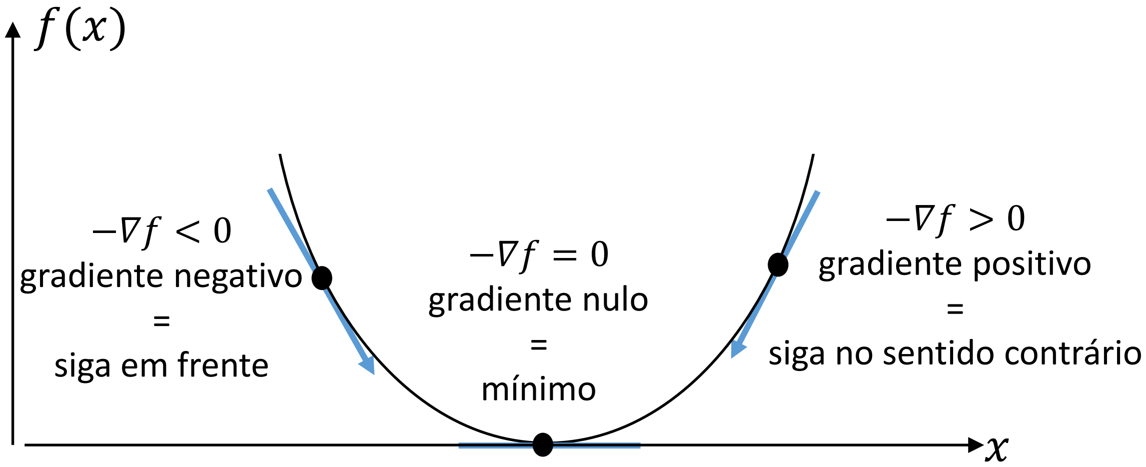

WHOA
I KNOW LINEAR REGRESSION

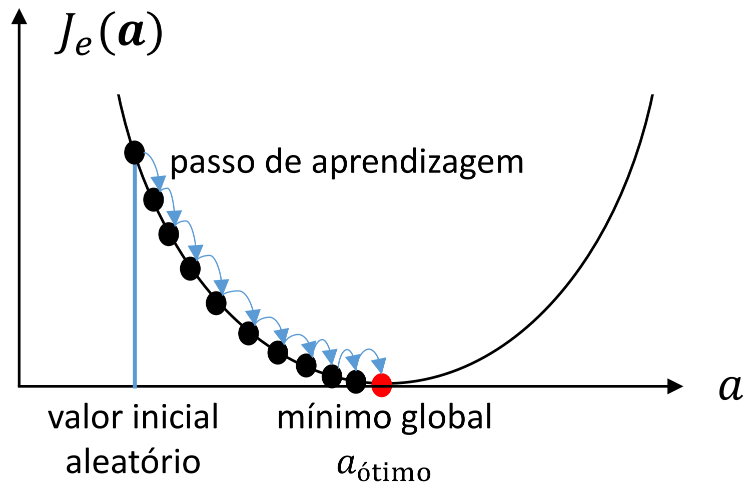

LETS LEARN
REGRESSION
memegenerator.net

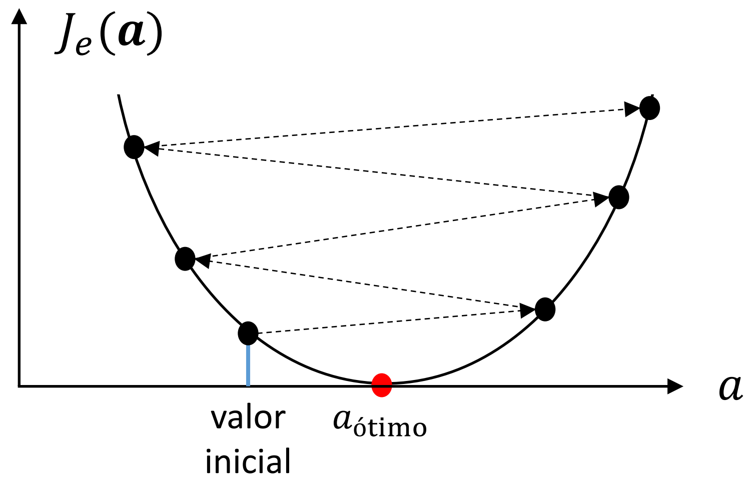
FIGURAS


$J_e(\boldsymbol{a})$
$\boldsymbol{a}$
mínimo local
mínimo global
platô

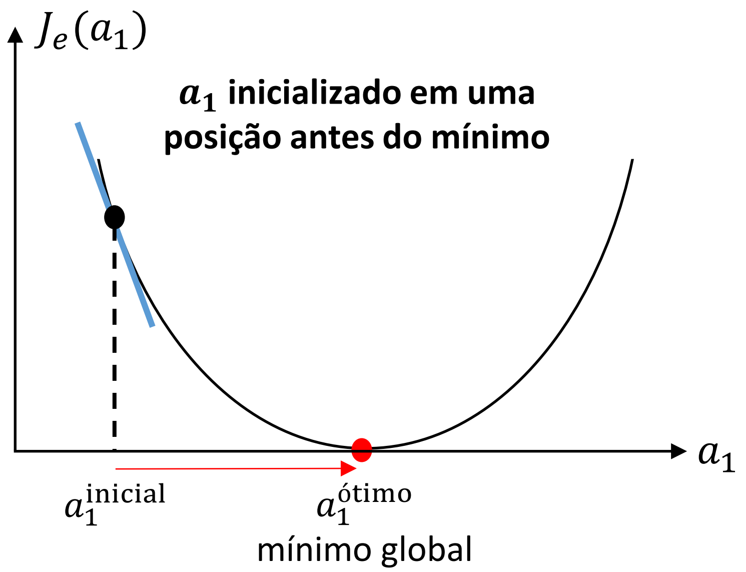

$f(x)$
$\nabla f = 0$
gradiente nulo
=
máximo
$\nabla f > 0$
gradiente positivo
=
siga em frente
$\nabla f < 0$
gradiente negativo
=
siga no sentido contrário
x


$f(x)$
$-\nabla f < 0$
gradiente negativo
=
siga em frente
$-\nabla f = 0$
gradiente nulo
=
mínimo
$-\nabla f > 0$
gradiente positivo
=
siga no sentido contrário
x


$J_e(\boldsymbol{a})$
passo de aprendizagem
a
valor inicial
aleatório
mínimo global
$a_{\text{ótimo}}$

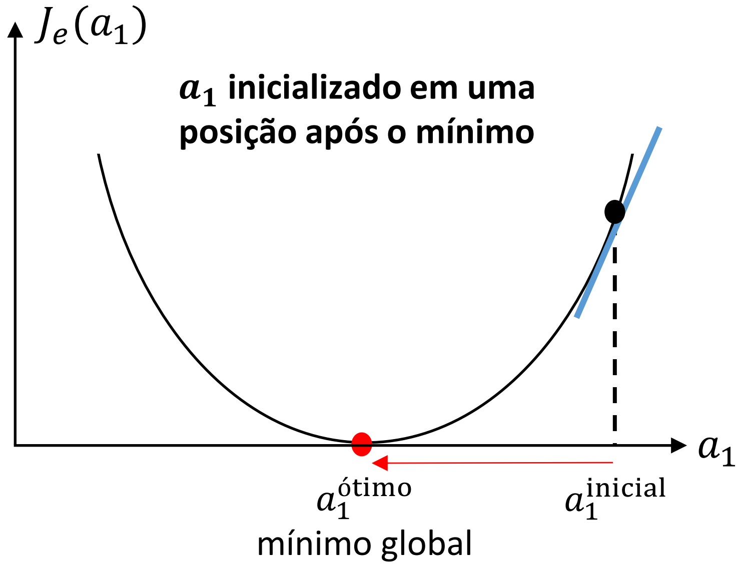


$J_e(\boldsymbol{a})$
a
valor inicial
$a_{\text{ótimo}}$



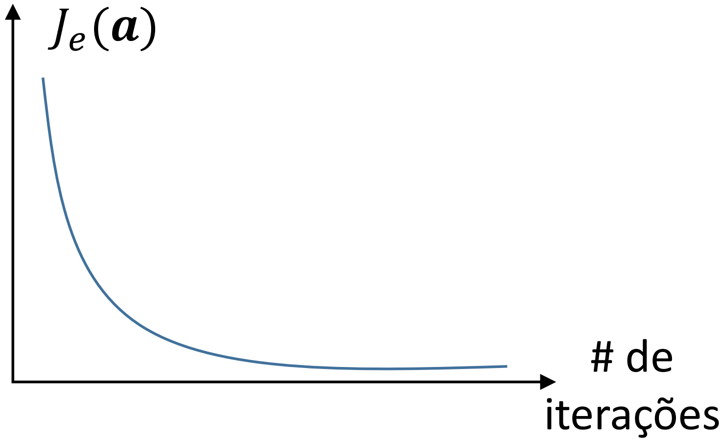
gradiente negativo: $a_1 = a_1^{\text{inicial}} + \alpha \nabla J_e(a_1)$

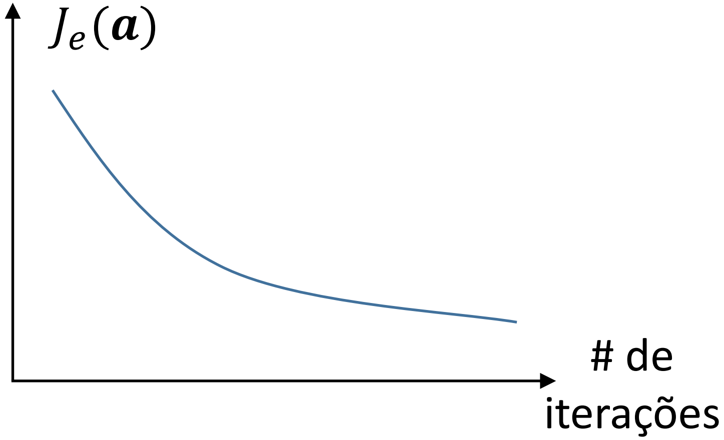
a_1 aumenta e se aproxima do mínimo

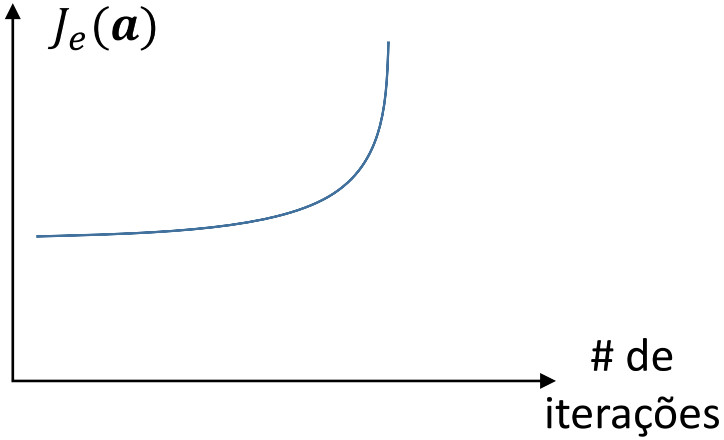


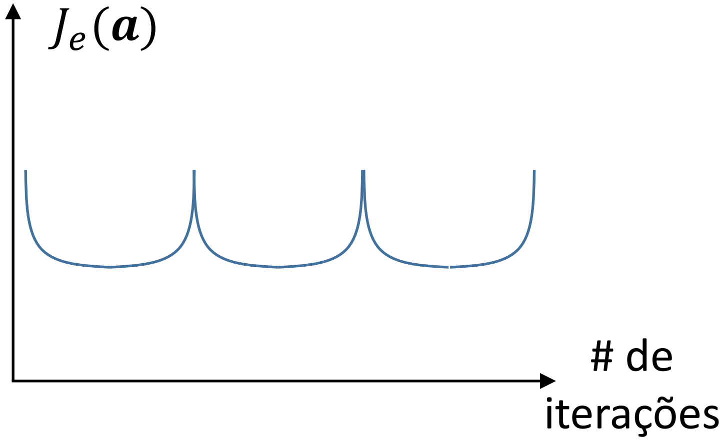
gradiente positivo: $a_1 = a_1^{\text{inicial}} - \alpha \nabla J_e(a_1)$

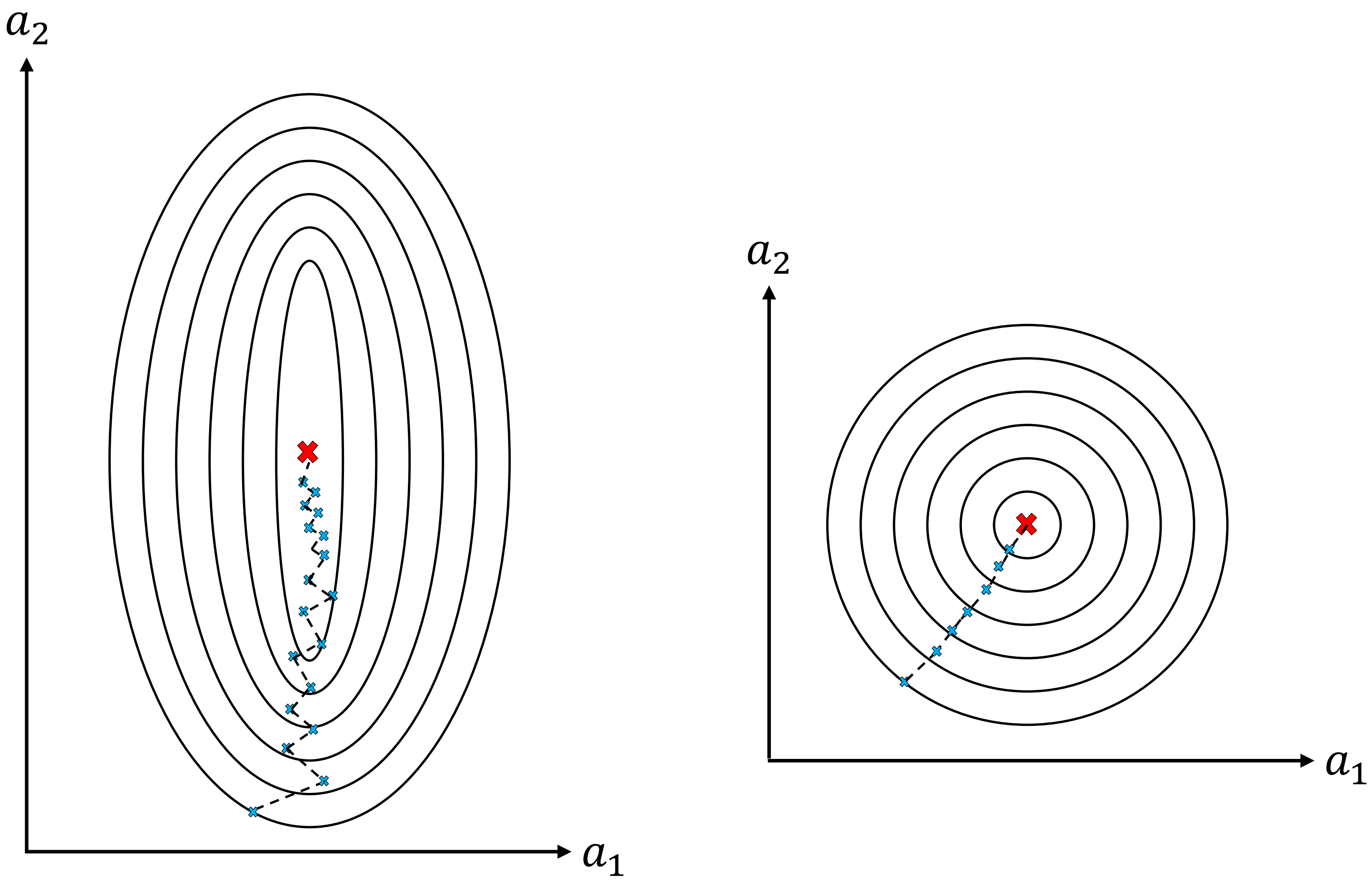
a_1 diminiu e se aproxima do mínimo


$J_e(\boldsymbol{a})$
de iterações

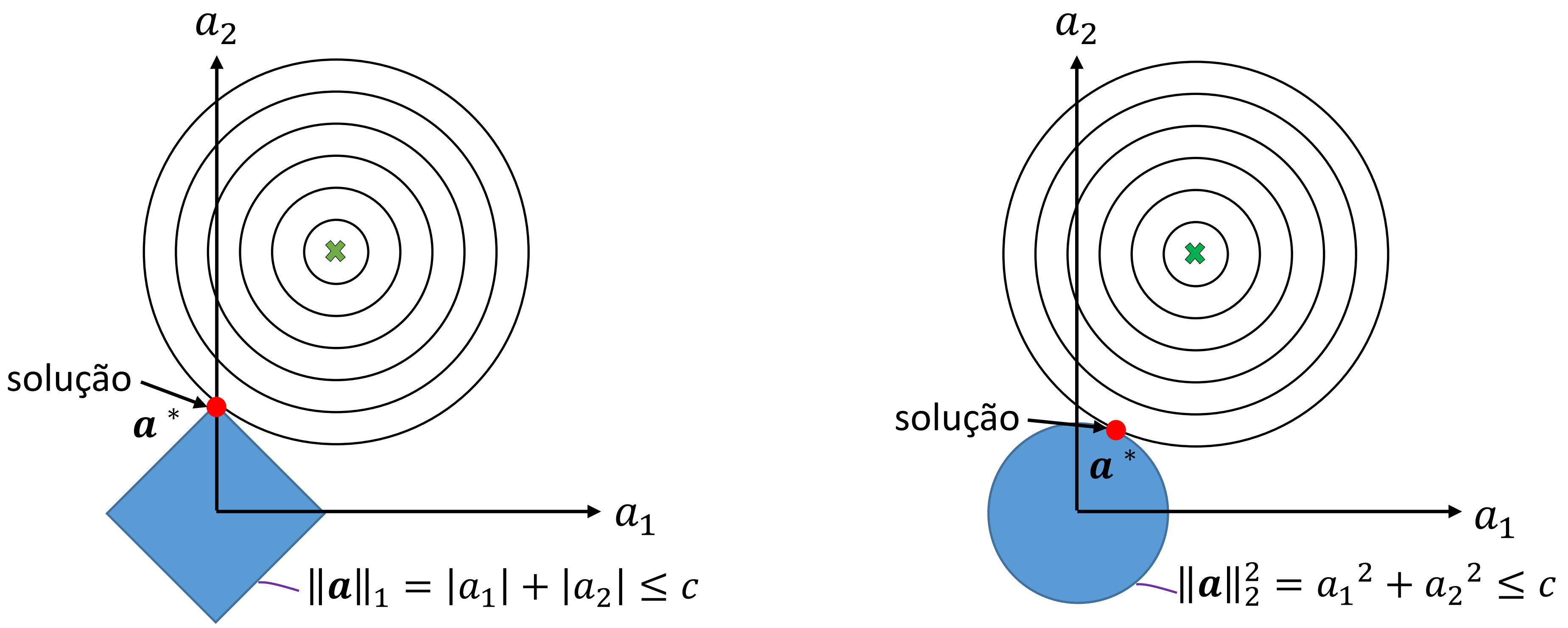

$J_e(\boldsymbol{a})$
de iterações


$J_e(\boldsymbol{a})$
de iterações

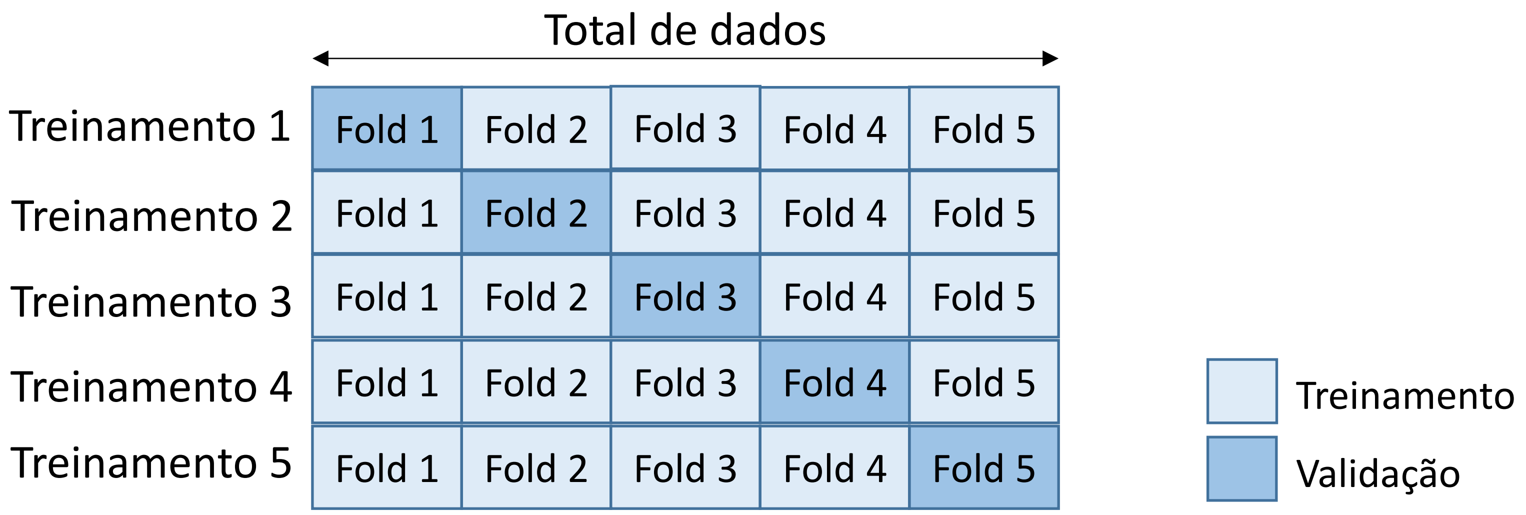

$J_e(\boldsymbol{a})$
de iterações



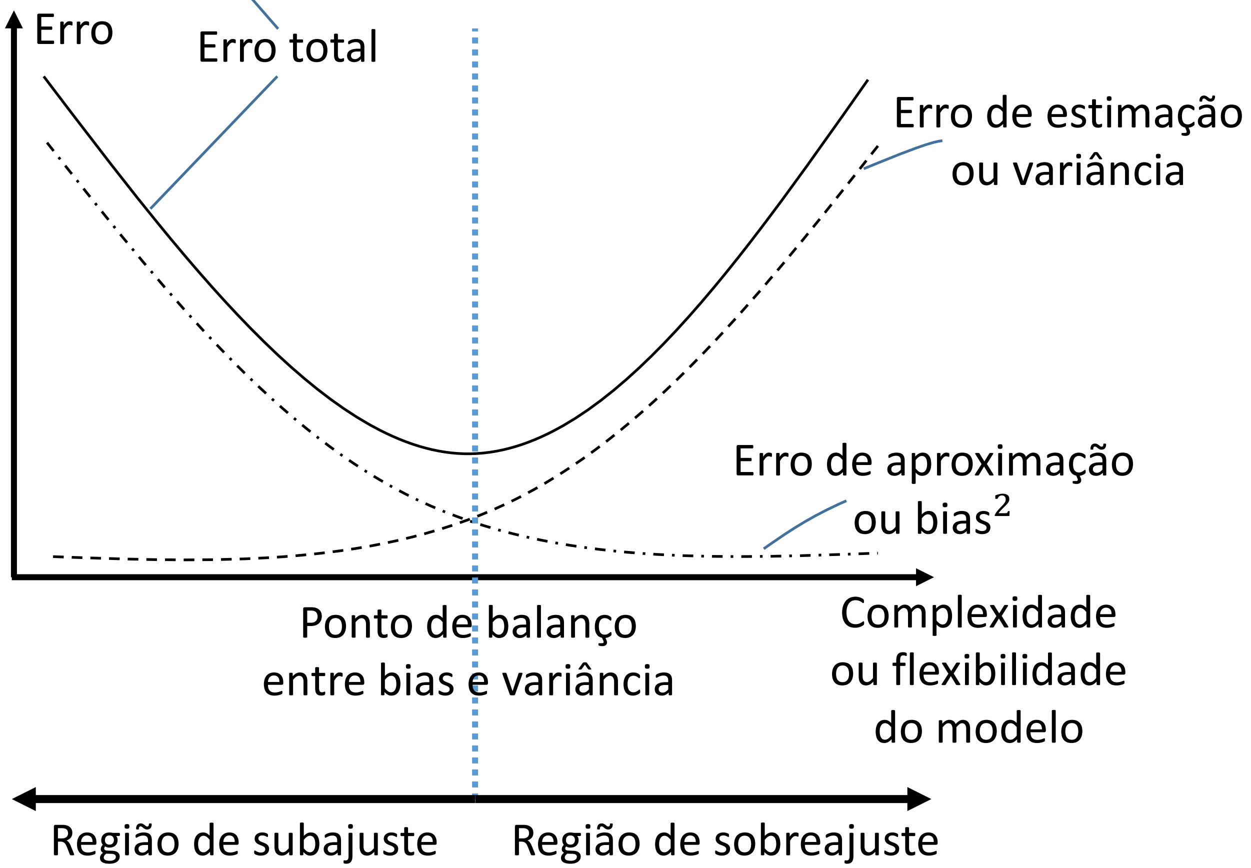
a_2
a_1
a_2
a_1



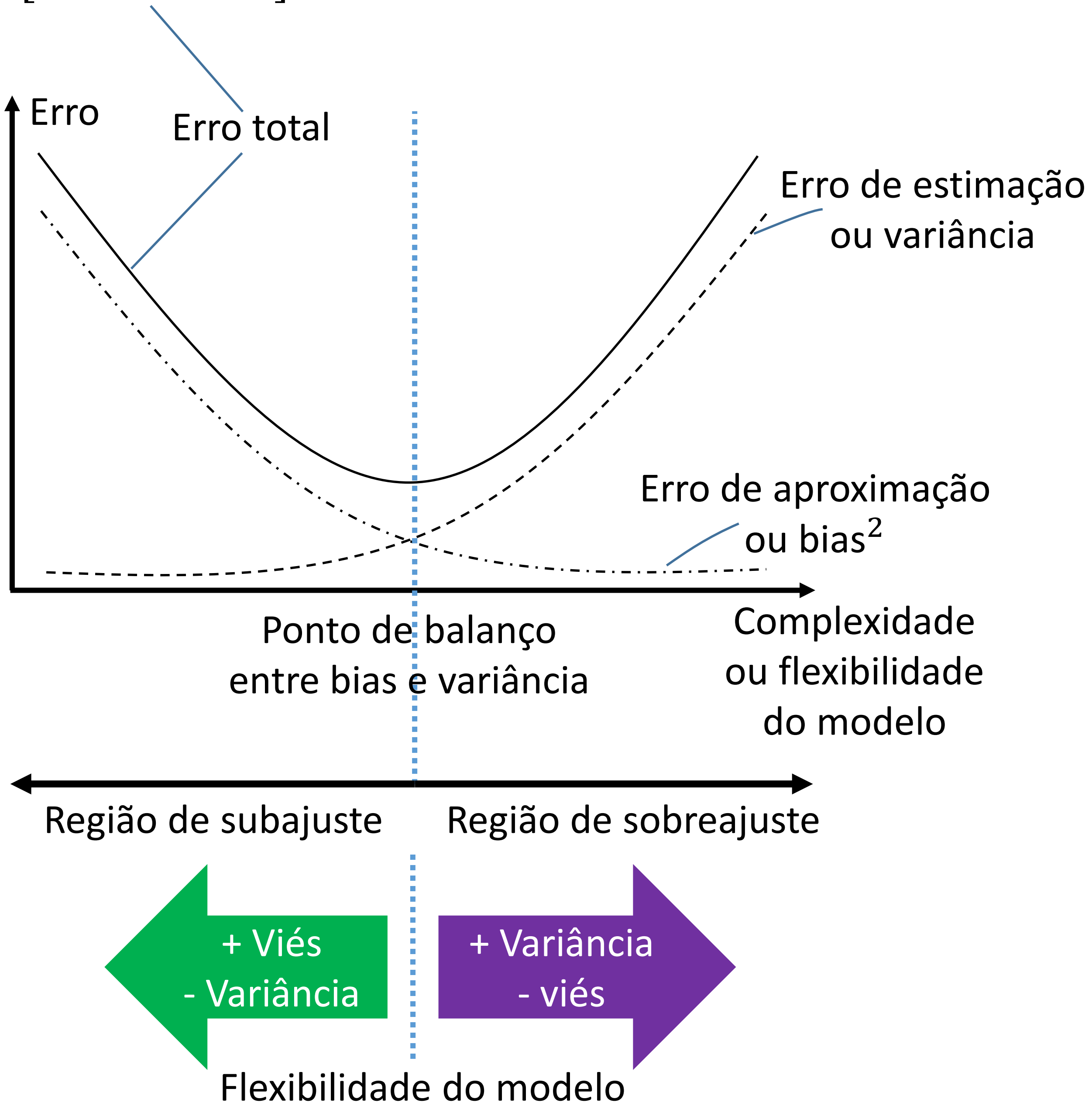
a_2
solução
$\boldsymbol{a}^*$
a_1
$\|\boldsymbol{a}\|_1 = |a_1| + |a_2| \leq c$
a_2
solução
$\boldsymbol{a}^*$
a_1
$\|\boldsymbol{a}\|_2^2 = {a_1}^2 + {a_2}^2 \leq c$

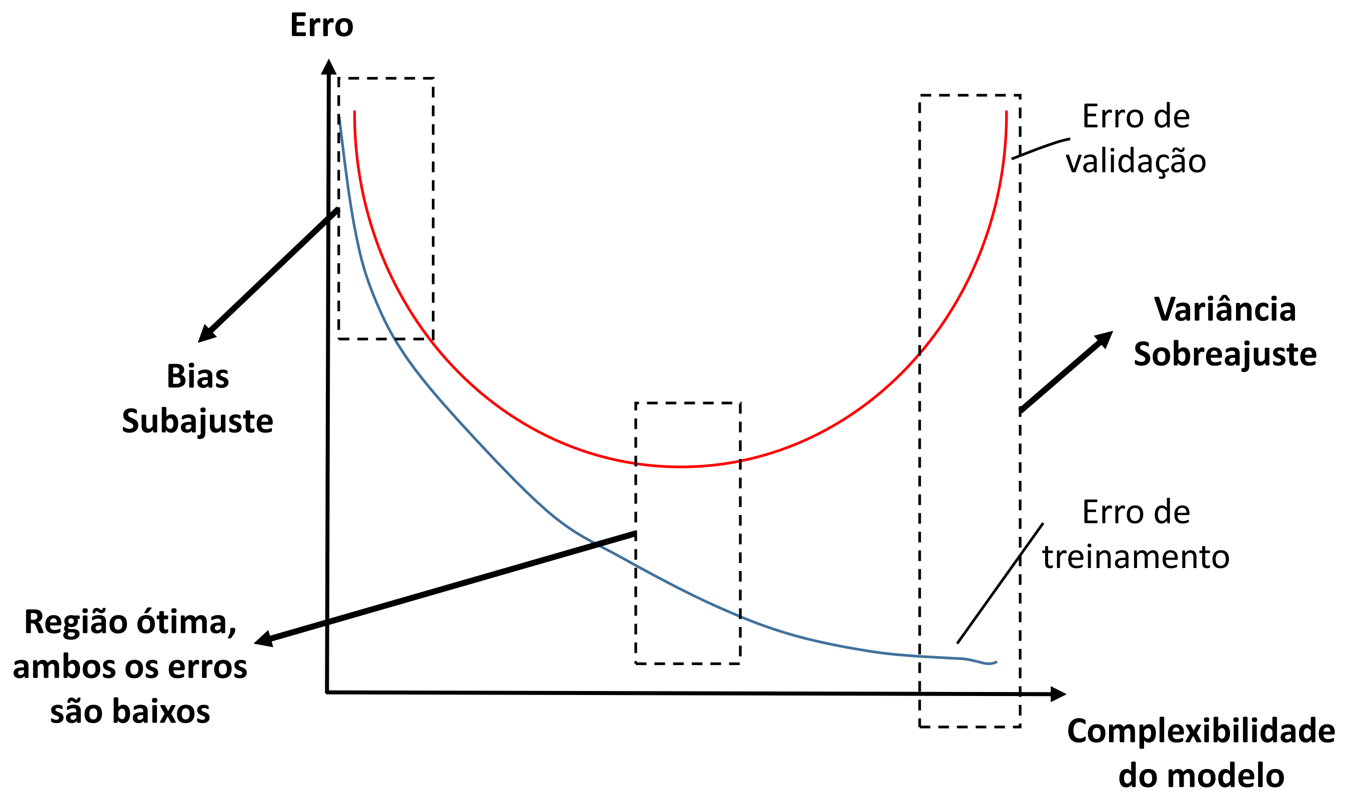

Total de dados
Treinamento 1
Treinamento 2
Treinamento 3
Treinamento 4
Treinamento 5
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Treinamento
Validação

$$E\left[\left(y_{noisy}-\hat{y}\right)^2\right] = \text{bias}^2 + \text{variância} + \text{erro irredutível}$$



$$E\left[\left(y_{noisy}-\hat{y}\right)^2\right] = \text{bias}^2 + \text{variância} + \text{erro irredutível}$$




Erro
Erro de
validação
Variância
Sobreajuste
Bias
Subajuste
Erro de
treinamento
Região ótima,
ambos os erros
são baixos
Complexibilidade
do modelo