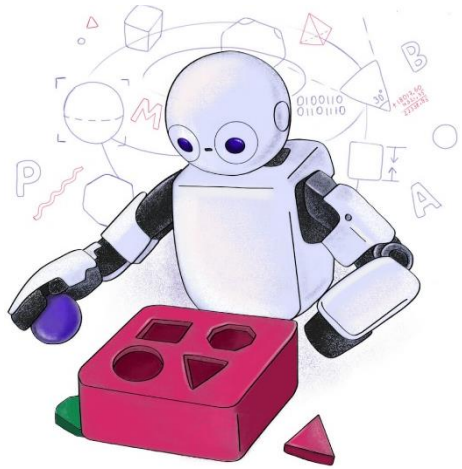
TP557 - Tópicos avançados em IoT e Machine Learning:

Datasets



Inatel

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

O que vamos ver?

- Anteriormente, aprendemos como criar classificadores, em particular para classificação de imagens, utilizando redes neurais.
- Após treinar o modelo, medimos sua acurácia e, após alguns testes básicos, verificamos que o modelo treinado reconhece as imagens muito bem.
- Porém, essa análise simplista pode nos levar a uma falsa sensação de segurança.
- Assim, neste tópico vamos explorar alguns problemas em torno desta análise superficial e aprender alguns métodos que podemos utilizar para evitar erros ao treinarmos uma rede neural de forma ingênua.

Reconhecendo calçados

- Imaginem uma situação onde queremos treinar uma rede neural para reconhecer diferentes tipos de calçados.
- É uma tarefa similar a ensinar alguém que nunca viu um calçado antes sobre o que eles realmente são para que no futuro quando essa pessoa ver um objeto ela poder decidir se ele é um calçado ou não.

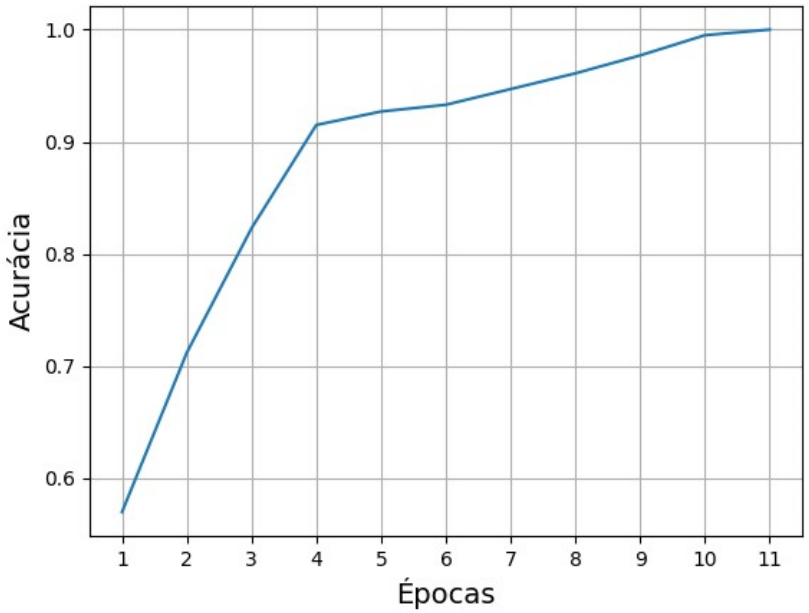
Passos a serem seguidos

Quanto mais exemplos de calçados em nossa base de dados, melhor!

- Sabemos que há uma enorme variedade de calçados e não há uma regra rígida sobre o que faz de um calçado um calçado.
- Normalmente, seguindo o *workflow* de trabalho com ML, nós:
  - coletaríamos o maior número possível de imagens de calçados,
  - treinaríamos uma rede neural usando esse conjunto,
  - e usaríamos o modelo treinado (i.e., inferências).

Resultados do treinamento

Acurácia de treinamento: **0.570**
Acurácia de treinamento: **0.712**
Acurácia de treinamento: **0.823**
Acurácia de treinamento: **0.915**
Acurácia de treinamento: **0.927**
Acurácia de treinamento: **0.933**
Acurácia de treinamento: **0.947**
Acurácia de treinamento: **0.961**
Acurácia de treinamento: **0.977**
Acurácia de treinamento: **0.995**
Acurácia de treinamento: **1.000**



- Durante o treinamento, poderíamos observar resultados como os mostrados ao lado.
- O modelo atinge uma acurácia de 100% em apenas 11 épocas!
- Isso pode significar que criamos um modelo incrível que pode reconhecer calçados.
- Então vamos usá-lo para realizar inferências com imagens inéditas de calçados!

Hora de usar o modelo treinado

- Mas então mostramos um sapato como este ao lado e ele falha em reconhecê-lo como um calçado.
- Pensamos que o modelo era 100% preciso em reconhecer calçados.
- Mas a realidade é que temos 100% de acurácia no reconhecimento dos tipos de calçados nos quais treinamos a rede neural e essa acurácia de 100% nos levou a uma falsa sensação de segurança de que o modelo funcionaria muito bem com qualquer outra imagem.

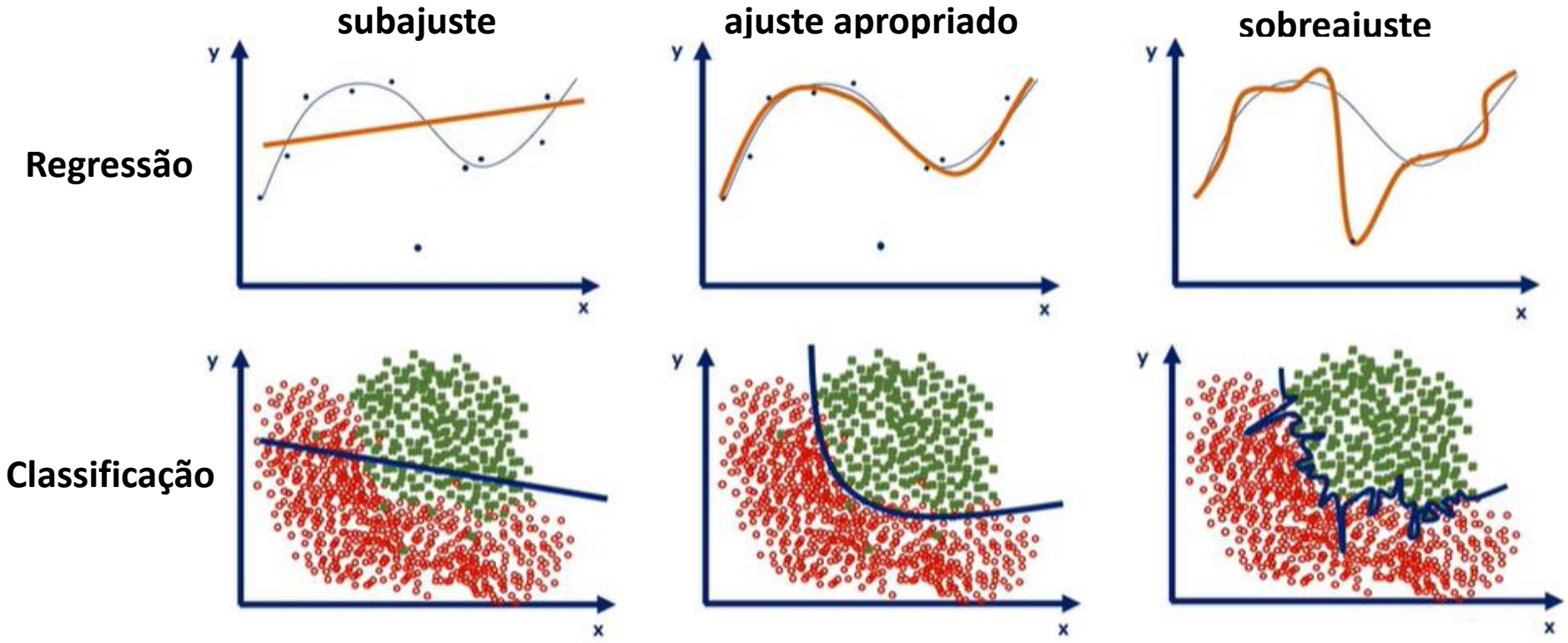
Acabamos de verificar que nosso modelo, inicialmente, perfeito não é tão perfeito assim.

O que fazer?

Sobreajuste

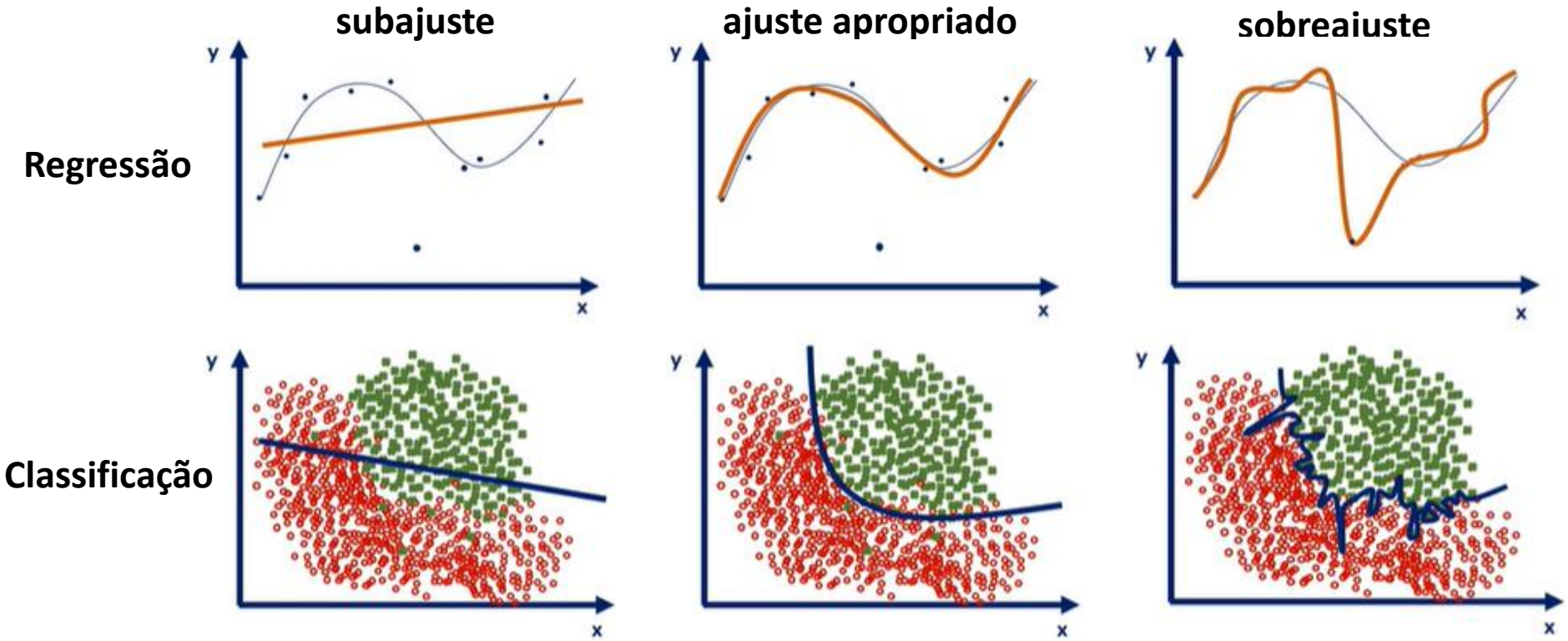
- Nosso modelo falhou em atingir o objetivo final, que era ***generalizar***.
- Muito provavelmente ele ficou demasiadamente bom para reconhecer calçados apenas no conjunto em que foi treinado.
  - Problema conhecido somo ***sobreajuste***.
- Precisamos de uma forma para analisar e evitar que o modelo se sobreajuste aos dados do conjunto de treinamento.
- Para isso, dividimos o conjunto total de exemplos em subconjuntos.

Subajuste e sobreajuste



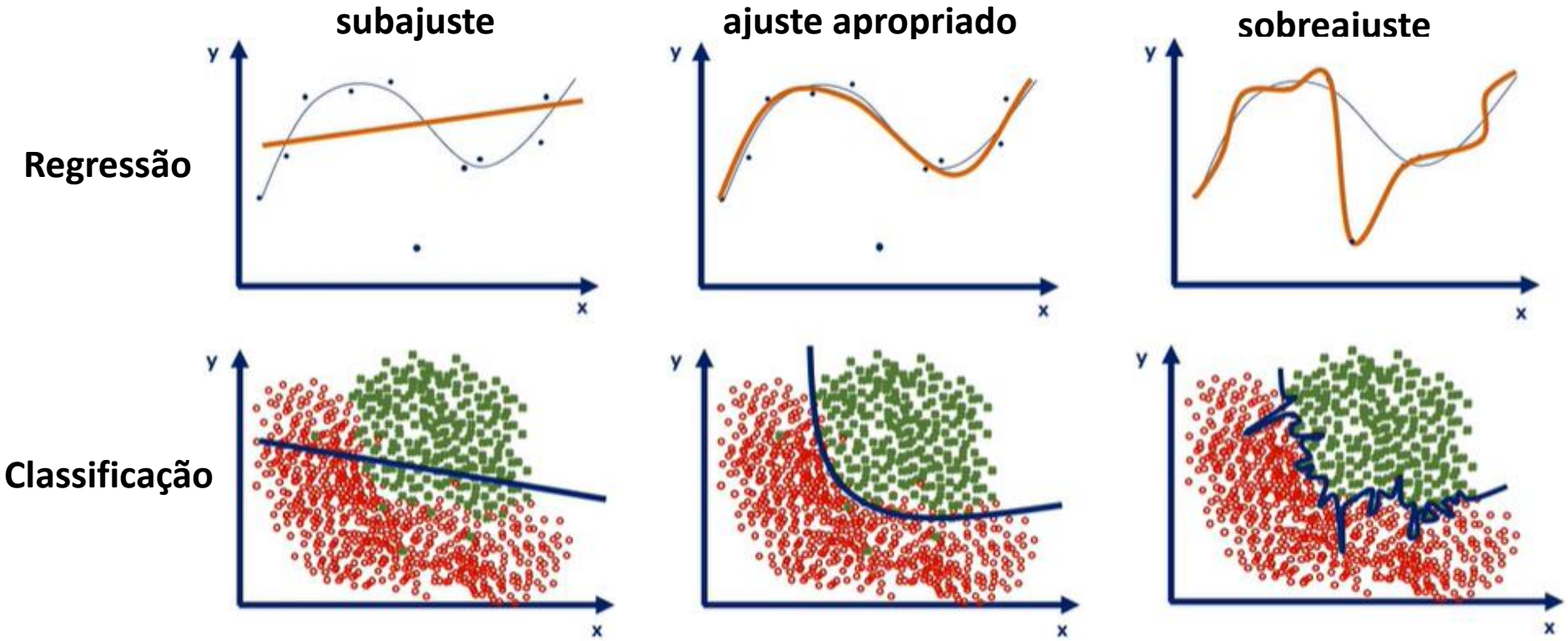
- Antes de falarmos sobre a divisão do conjunto total de dados, vamos falar rapidamente sobre dois problemas comuns que modelos de ML podem apresentar, o ***subajuste*** e o ***sobreajuste***.

Subajuste e sobreajuste



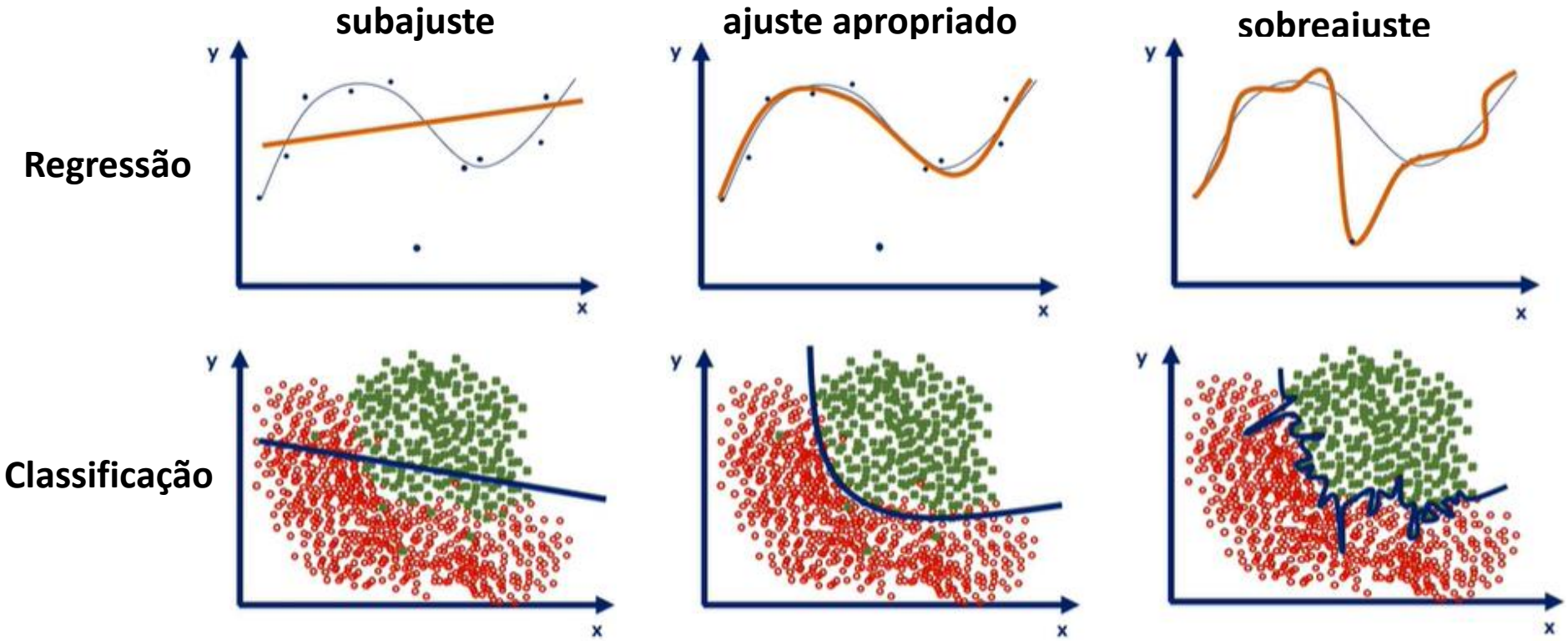
- O ***subajuste*** ocorre quando um modelo não é capaz de capturar adequadamente os relações e padrões presentes nos dados de treinamento.
  - Em outras palavras, o ***modelo é muito simples*** para representar a complexidade dos dados.
- O modelo apresenta erro no conjunto de treinamento muito alto.

Subajuste e sobreajuste



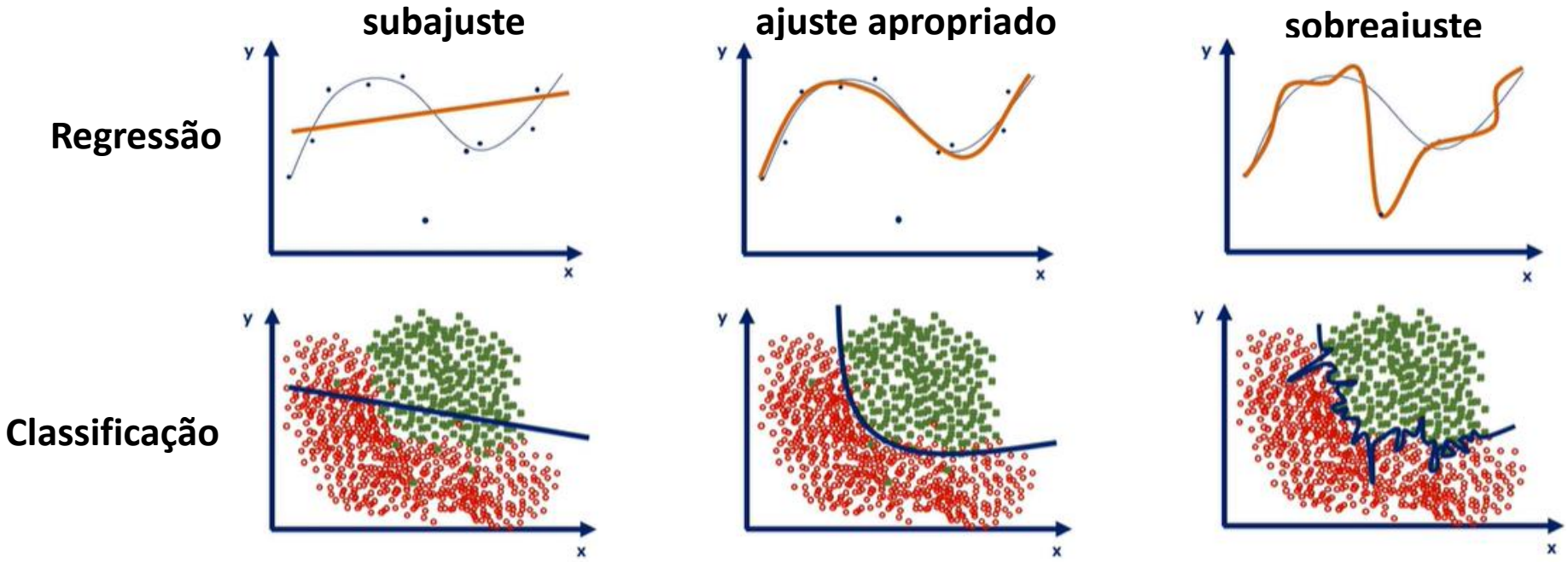
- Algumas causas do ***subajuste*** são: ***modelo muito simples*** (i.e., sem complexidade), ***poucas épocas de treinamento*** (i.e., insuficiente) e ***falta de dados*** (modelo falha em aprender as características relevantes).

Subajuste e sobreajuste



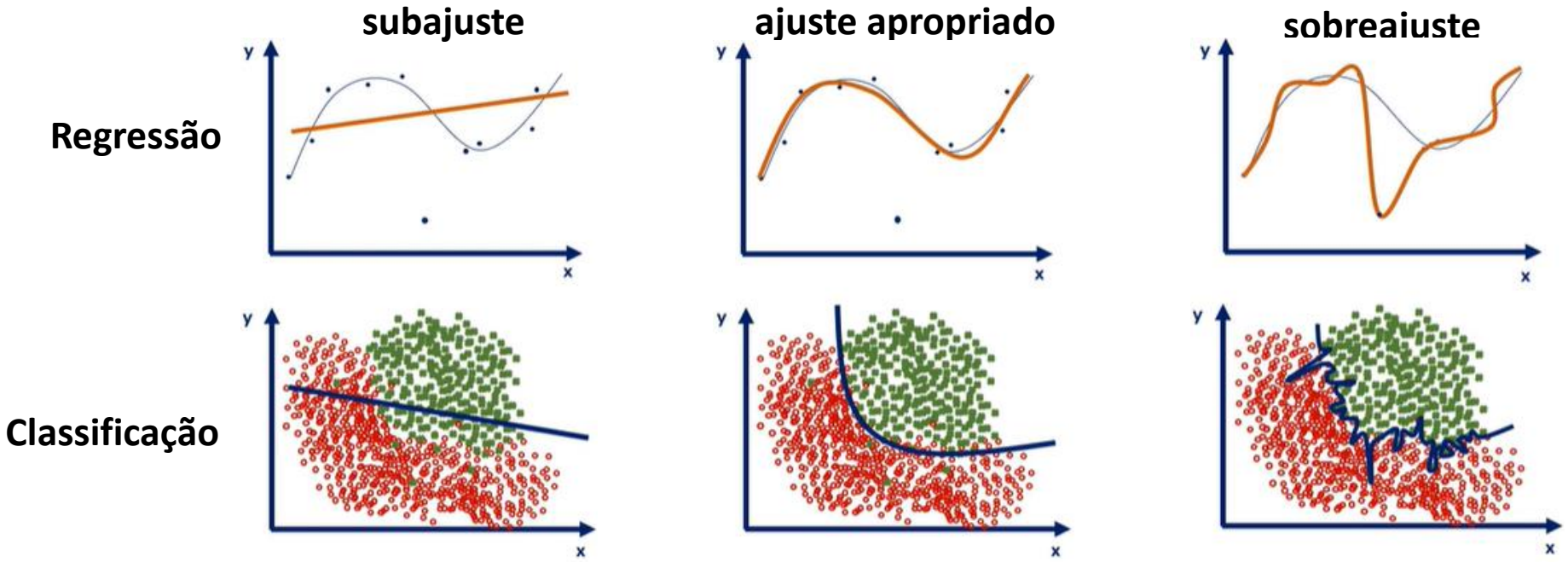
- Para mitigar o problema, podemos ***aumentar a complexidade*** do modelo (e.g., aumentar camadas e neurônios), ***ajustar os hiperparâmetros*** (e.g., passo de aprendizagem), ***treinar por mais épocas*** e ***aumentar o conjunto de treinamento*** se possível.

Subajuste e sobreajuste



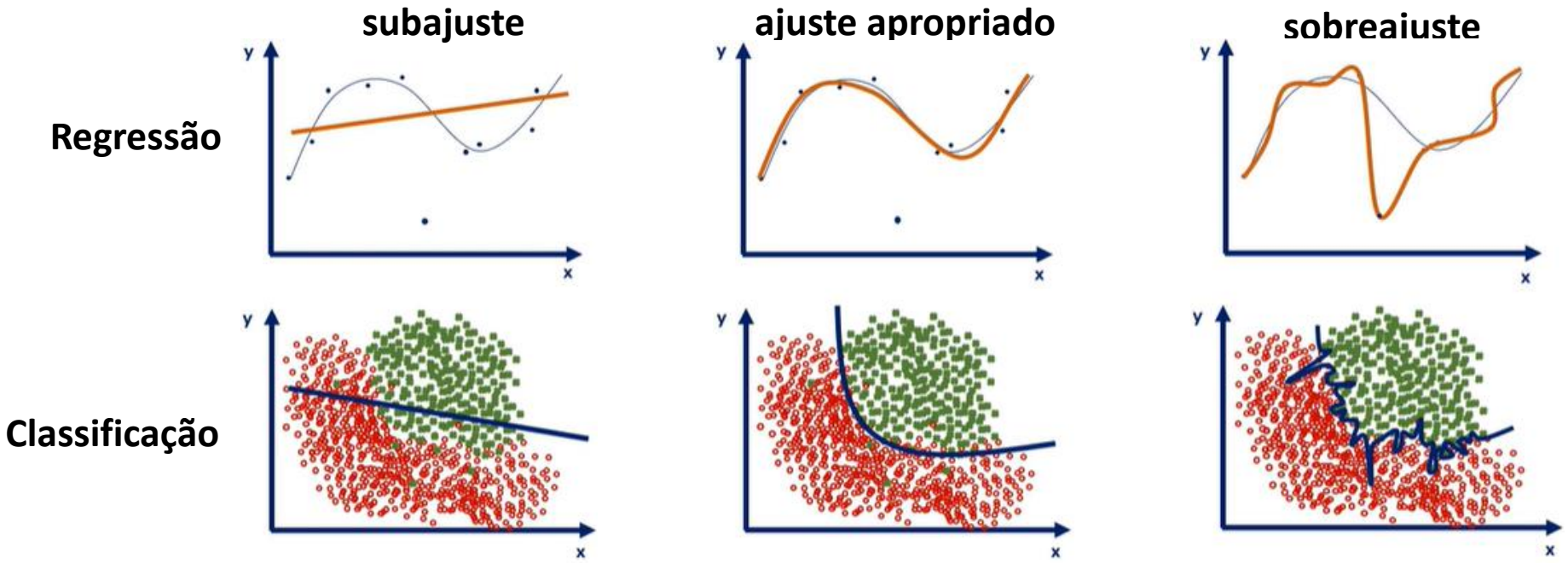
- O ***sobreajuste*** ocorre quando o modelo se ***ajusta excessivamente aos dados de treinamento*** e acaba perdendo a capacidade de generalizar para dados que não foram vistos durante o treinamento.
  - Em outras palavras, o modelo ***memoriza os dados de treinamento*** em vez de aprender padrões gerais.
- O modelo apresenta erro no conjunto de treinamento muito baixo, próximo de zero.

Subajuste e sobreajuste



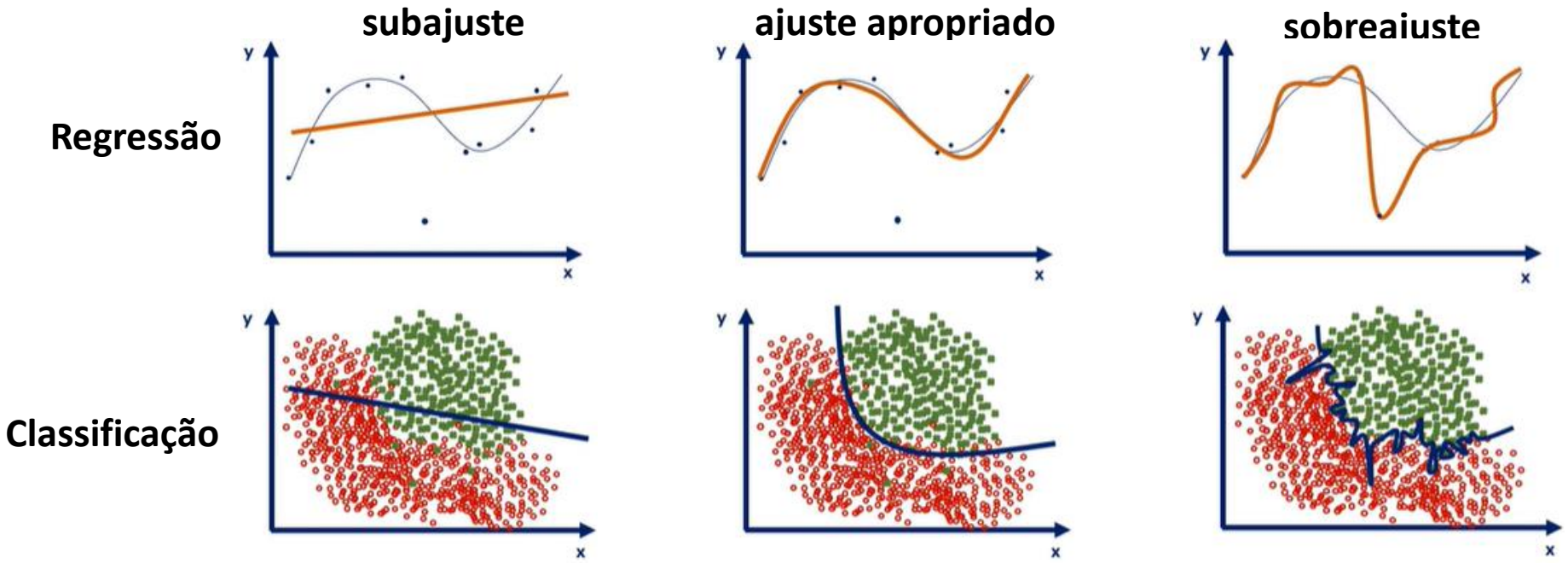
- Algumas causas do ***sobreajuste*** são: ***modelo muito complexo*** em relação à quantidade de dados, ***treinamento excessivo*** (leva à memorização dos dados de treinamento), ***falta de dados*** (modelo tem poucos exemplos para aprender padrões gerais).

Subajuste e sobreajuste



- Para mitigar o problema, podemos ***aumentar os dados de treinamento***, se possível, ***reduzir a complexidade do modelo*** (e.g., removendo camadas ou neurônios), ***aplicar técnicas de regularização*** (e.g., *dropout*, regularizações L1 ou L2, ***early-stop*** (acompanhar os erros de treinamento e de teste ajuda a identificar quando o sobreajuste está ocorrendo).

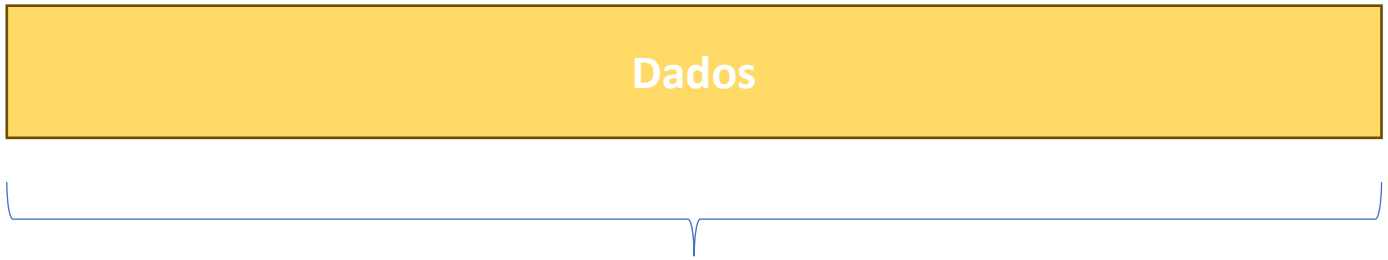
Subajuste e sobreajuste



- Encontrar o equilíbrio certo entre a complexidade (ou flexibilidade) do modelo e sua capacidade de generalização é essencial para obter um modelo com bom desempenho.
- Ou seja, devemos encontrar um equilíbrio entre um modelo muito simples (subajuste) e um modelo muito complexo (sobreajuste).

Como encontramos esse equilíbrio?

Dividir para conquistar!



Conjunto de dados com todos os exemplos (e.g., imagens) que foram coletados.

- Nossa ideia inicial foi treinar o modelo com todas os exemplos, pois quanto mais dados tivermos, melhor será seu treinamento.
- Porém, quando a rede é treinada com todos os exemplos que possuímos, nós não temos um contexto para mensurar o quão bem ela se sai com dados nunca vistos anteriormente.
  - Generalizar vs. Sobreajustar.

Dividir para conquistar!

- E se dividirmos o conjunto total de exemplos em conjuntos menores?
  - ***Conjunto de treinamento***
  - ***Conjunto de validação***
  - ***Conjunto de teste***

Conjunto de treinamento

- ***Conjunto de treinamento***: usado para ajustar os parâmetros (i.e., pesos) do modelo.
- É o maior dos três subconjuntos.
- ***Tamanho do subconjunto***: 70% a 80% do total.

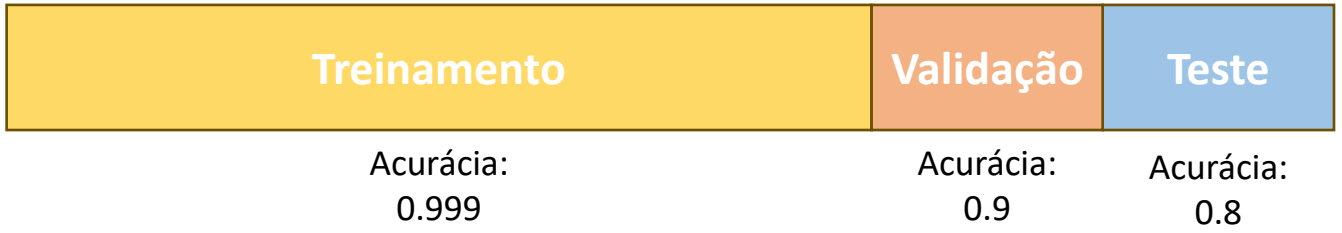
Conjunto de validação

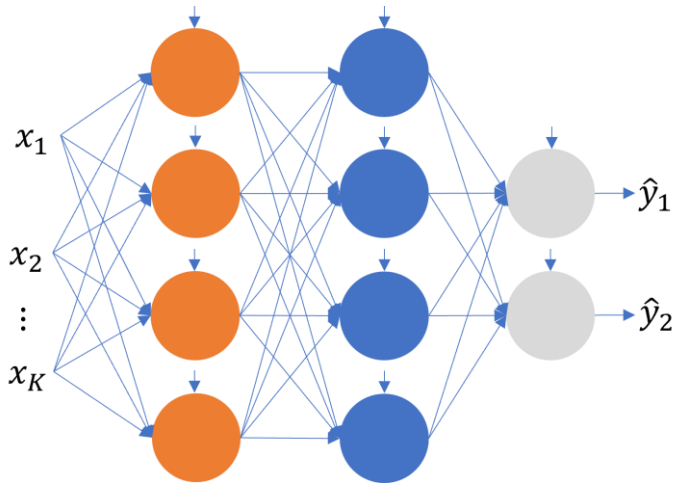
- ***Conjunto de validação***: usado para avaliar o desempenho do modelo em dados inéditos e ajustar hiperparâmetros.
  - **Hiperparâmetros**: parâmetros que não são aprendidos durante o treinamento do modelo, mas que influenciam seu aprendizado
- ***Tamanho do subconjunto***: 10% a 15% do total.
- A validação é importante para evitar o sobreajuste.

Conjunto de teste

- ***Conjunto de teste*:** conjunto mantido completamente separado durante todo o processo de desenvolvimento do modelo.
- É usado apenas no final para avaliar o desempenho do modelo em dados inéditos.
- ***Tamanho do subconjunto***: 10% a 15% do total.

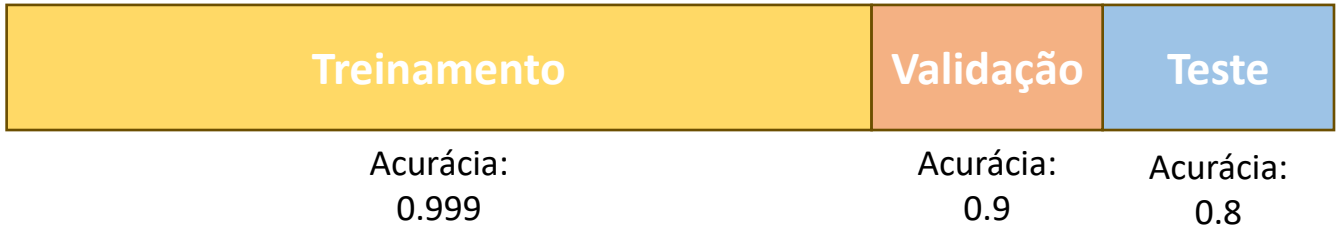
Aplicando a metodologia

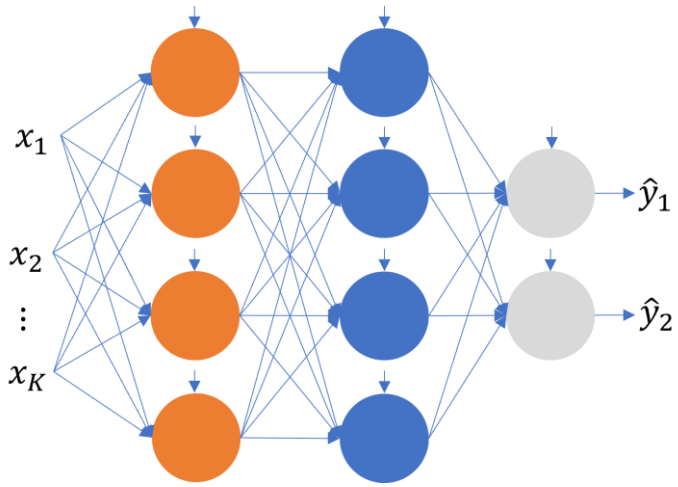




- Seguindo essa metodologia, poderíamos, por exemplo, escolher uma arquitetura de rede neural e treiná-la para resolver um problema de classificação.
- Nos dados de treinamento, a acurácia é de 99.9%.
- Mas ela é pior nos outros dois conjuntos.
- Ela é de 90% e 80% nos conjunto de validação e teste, respectivamente.
- O que isso pode indicar?

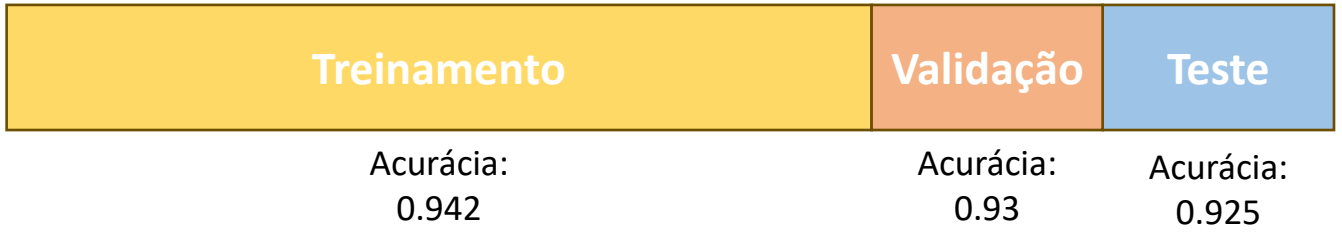
Aplicando a metodologia

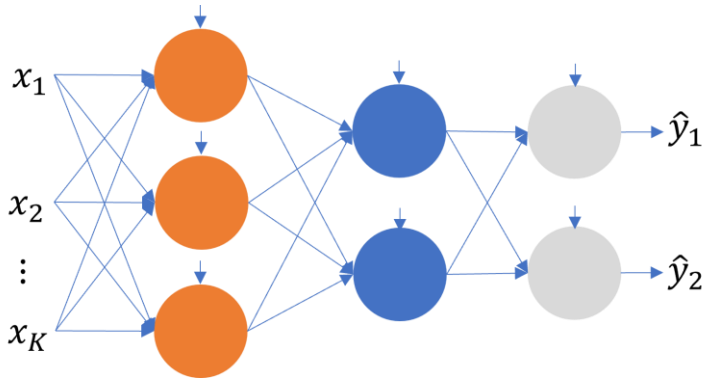




- Podemos estar diante de uma situação igual a da detecção de calçados.
- Projetamos uma rede neural que é ótima nos dados de treinamento, mas não tão boa nos outros dados.
  - Uma indicação de sobreajuste.
- Os 99.9% nos fazem pensar que temos uma rede muito melhor do que realmente temos.
- E se reprojetarmos a rede (e.g., reduzir sua complexidade) e tentamos novamente?

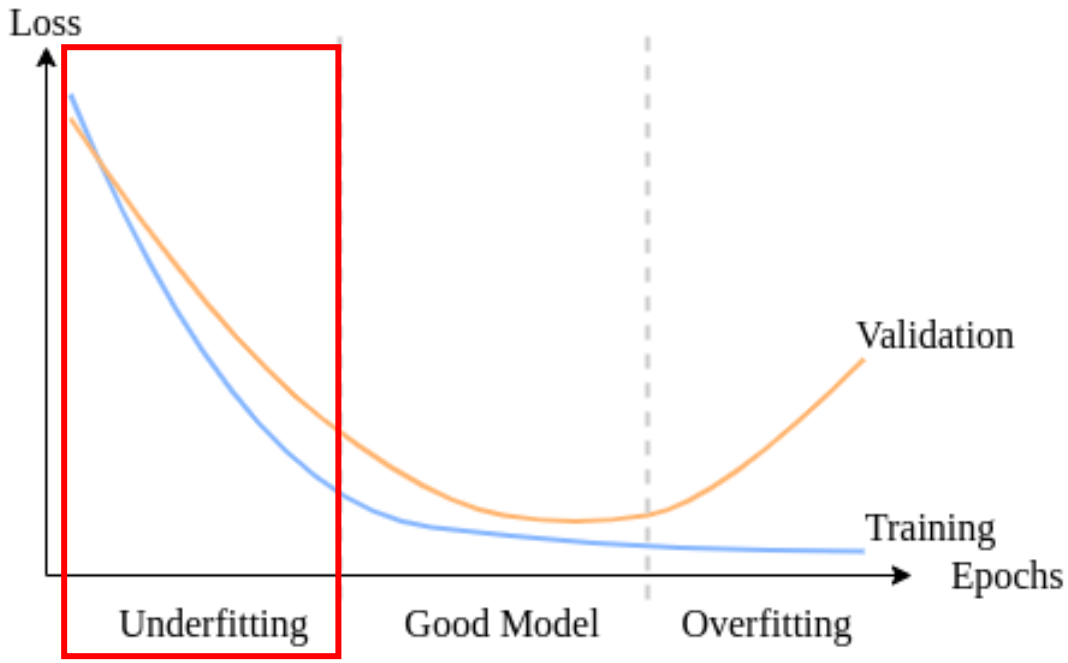
Aplicando a metodologia





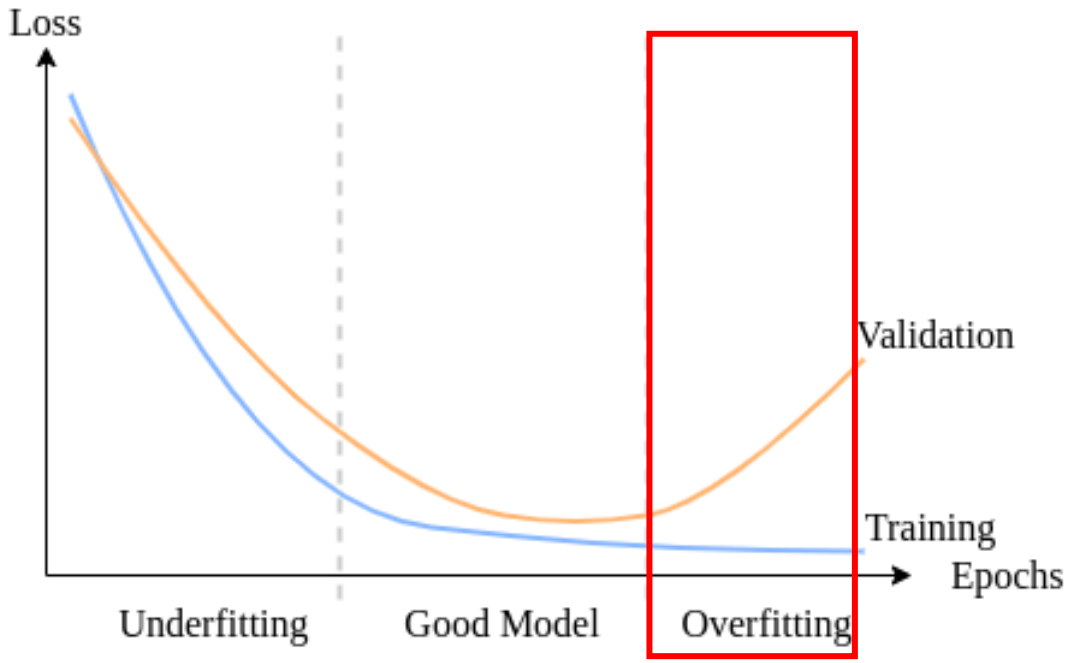
- A acurácia no conjunto de treinamento pode diminuir, mas o mais importante é manter a acurácia da rede nos conjuntos de validação e teste o mais próximos do treinamento.
- Essa proximidade dos valores nos dará uma forte indicação da verdadeira acurácia da rede.
- À luz dessas informações, vamos revisitar nosso exemplo dos dígitos escritos à mão.

Analisando o erro



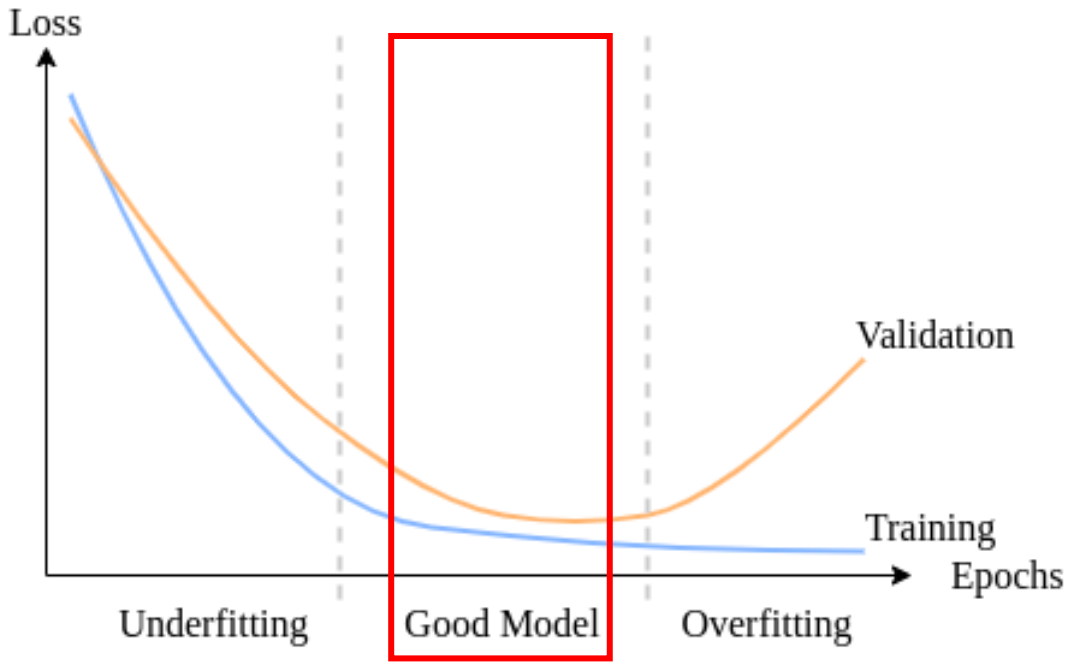
- Podemos extrair informações muito importantes a partir dos resultados de erro ao longo das épocas de treinamento de um modelo.
- ***Subajuste***: ambos os erros são altos.
- O modelo não tem complexidade o suficiente para encontrar um padrão geral e/ou ainda não treinou o suficiente.
- Aumentar o passo de aprendizagem pode ajudar a mitigar o problema.

Analisando o erro



- ***Sobreajuste***: erro de treinamento pequeno e erro de validação alto.
- O modelo tem complexidade maior do que a necessária e/ou treinou por um número grande de épocas.
  - Quando um modelo vê o mesmo conjunto muitas vezes, a tendência é que ele memorize os dados, da mesmo forma que ocorre conosco.

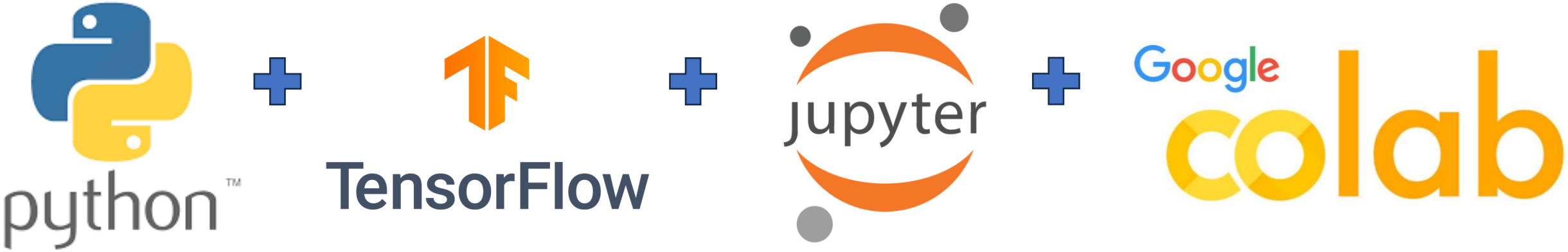
Analisando o erro



- ***Generalização***: ambos os erros são pequenos e próximos.
- Balanço entre complexidade e capacidade de generalização do modelo.
- O modelo tem a complexidade ideal para capturar o padrão geral por trás dos dados e com isso generalizar bem.
- Poderíamos encerrar o treinamento assim que o erro de validação começar a aumentar consistentemente.

Exemplo

- Exemplo: Detecção de dígitos escritos à mão com dados de validação e teste

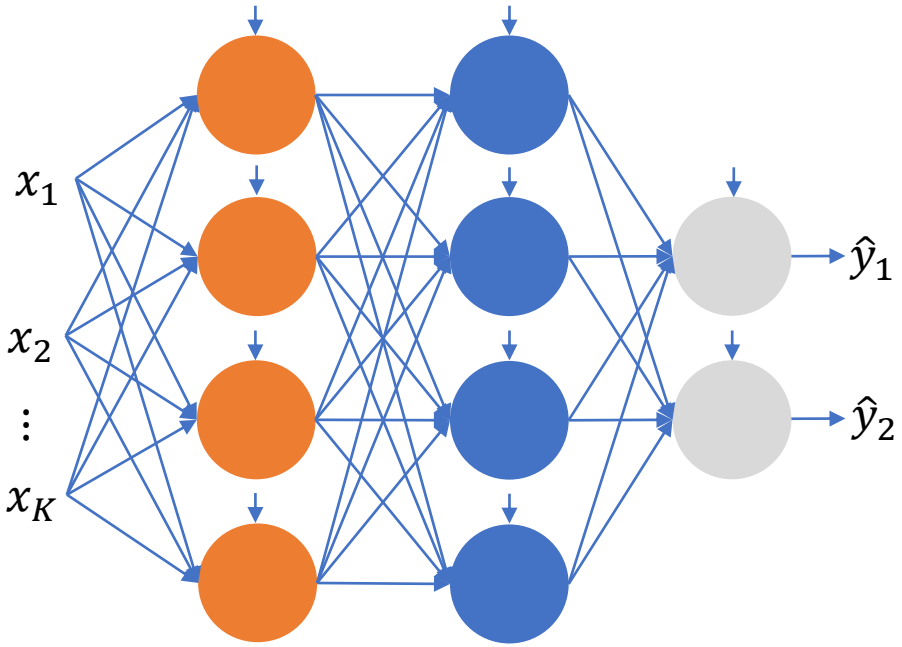


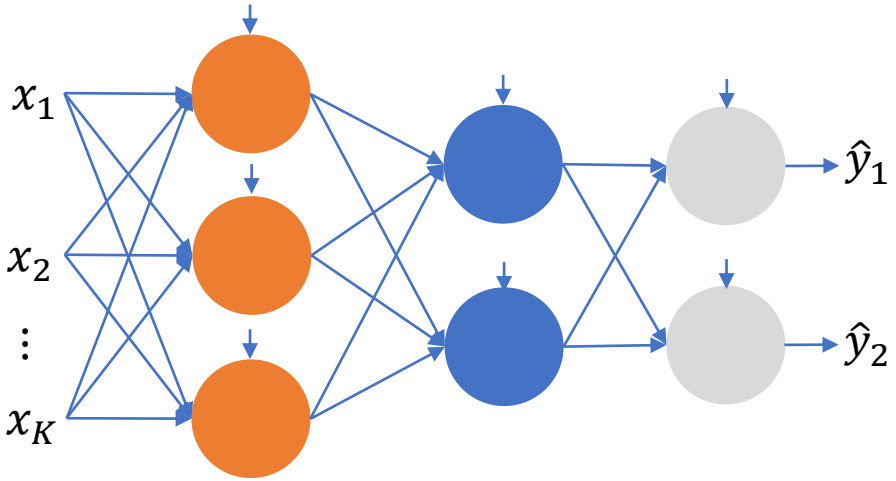
Atividades

- Quiz: “***TP557 - Datasets***”.
- Exercício: Analisando os resultados do treinamento de um modelo de ML.

Perguntas?

Obrigado!


x_1
x_2
x_K
$\hat{y}_1$
$\hat{y}_2$


x_1
x_2
x_K
$\hat{y}_1$
$\hat{y}_2$