

Redes Neurais e Aprendizado Profundo



Nas aulas anteriores, implementamos os seguintes algoritmos de Aprendizado de Máquina:

- Regressão Linear (para problemas de regressão)
- Regressão Logística (para problemas de classificação)

OBS: Com isso, finalizamos a **Parte 1** da disciplina.

Nas aulas anteriores, implementamos os seguintes algoritmos de Aprendizado de Máquina:

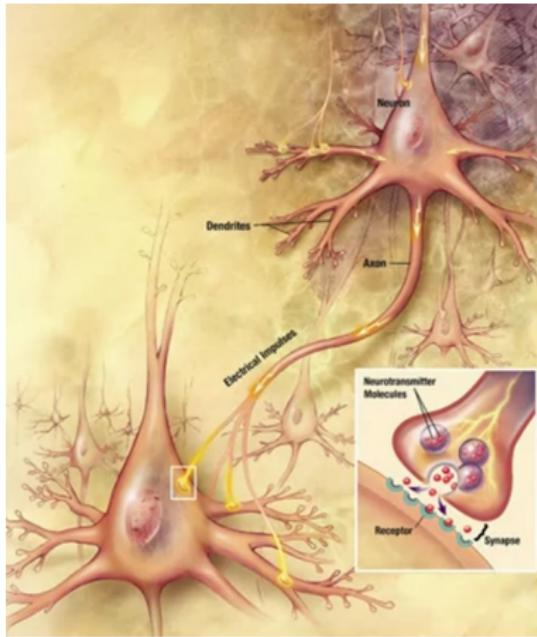
- Regressão Linear (para problemas de regressão)
- Regressão Logística (para problemas de classificação)

OBS: Com isso, finalizamos a **Parte 1** da disciplina.

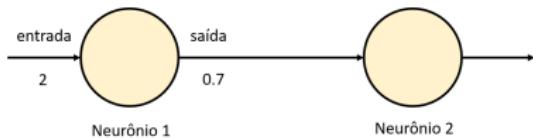
Vamos agora entrar na **Parte 2** da disciplina, onde estudaremos algoritmos **mais avançados**, em especial, as **Redes Neurais Artificiais** (RNAs)

- **Origem:** Algoritmos que imitam o funcionamento do cérebro humano (redes neurais biológicas).
- **Aplicações modernas:** Reconhecimento de voz, Processamento de imagens, Processamento de linguagem natural (texto), Modelos Generativos, etc

Mas como funcionam os neurônios biológicos?

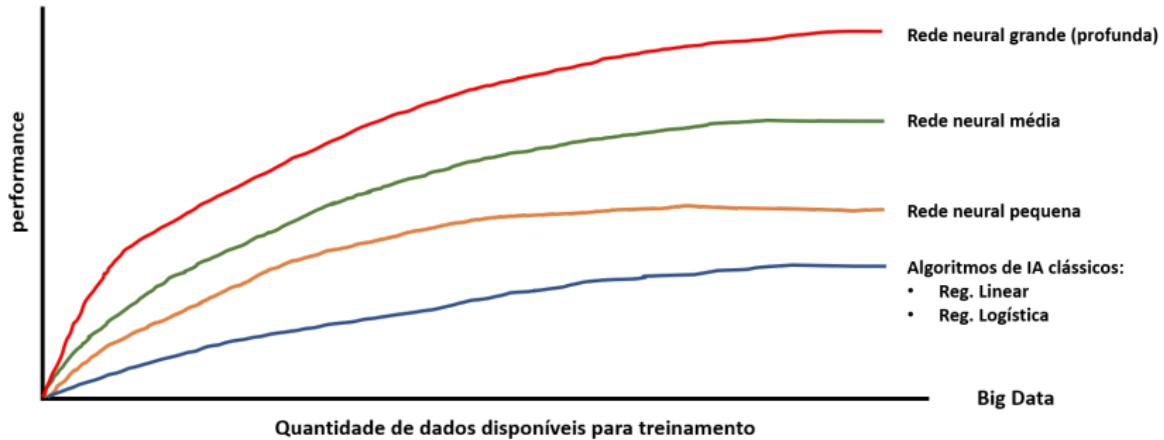


- **Possuem entradas**, que recebem sinais elétricos de outros neurônios
- Eses sinais são recebidos, processados e enviados por meio das suas **saídas** a outros neurônios.
- Os neurônios seguintes recebem esses sinais de saída como sendo suas entradas, e o ciclo continua.
- Esse raciocínio serviu de inspiração para a criação das redes neurais artificias.



- Para nós, um neurônio é um modelo (função), que recebe uma ou mais entradas, faz alguma conta com esses valores numéricos, e os disponibiliza em sua(s) saída(s)

Necessidade de algoritmos mais complexos

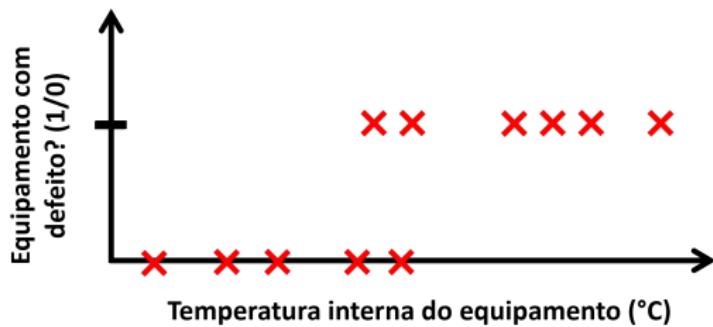


- Devido ao aumento de dados disponíveis digitalmente
- Devido à elevada capacidade de processamento a nível de hardware (GPUs, por exemplo)
- Assim, aplicações bastante complexas se tornaram possíveis (ChatGPT, por exemplo)

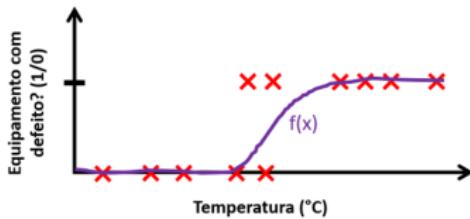
Exemplo: Equipamento com defeito

Suponha:

- Que você possui um equipamento de alto custo
- Que você deseja realizar manutenções preventivas periodicamente
- Que o fabricante do equipamento forneceu o seguinte conjunto de dados para auxiliar no diagnóstico de falhas



Exemplo: Equipamento com defeito



Tratando como um modelo de **regressão logística**, temos:

Entrada:

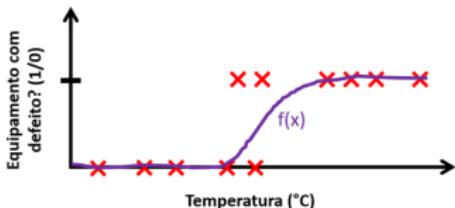
x : temperatura do equipamento

Saída:

$f(x)$: probabilidade do equip. com temp. x estar com defeito

$$f(x) = \frac{1}{1 + e^{-(wx+b)}} \rightarrow \text{Sigmoid}$$

Exemplo: Equipamento com defeito



Tratando como um modelo de **regressão logística**, temos:

Entrada:

x : temperatura do equipamento

Saída:

$f(x)$: probabilidade do equip. com temp. x estar com defeito

$$f(x) = \frac{1}{1 + e^{-(wx+b)}}$$

Tratando como uma **rede neural com um único neurônio**, temos:

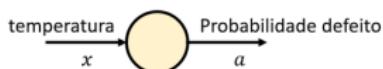
Entrada:

x : temperatura do equipamento

Saída:

$f(x)$: probabilidade do equip. com temp. x estar com defeito

$$a = f(x) = \frac{1}{1 + e^{-(wx+b)}}$$



- a refere-se à quantidade de ativação feita pelo neurônio
- Nessa representação, note que o neurônio é uma unidade de regressão logística, ou seja, cada neurônio está sendo ativado pela função sigmoide.
- Também seria possível usar uma função de ativação linear, tal que $a = f(x) = wx + b$. Algumas pessoas chamam este caso de neurônio "sem ativação". Note que isso resulta num modelo de regressão linear.

Exemplo: Equipamento com defeito

Buscando tornar o problema **mais realista**, vamos considerar agora que o equipamento estar com defeito ou não depende de 4 características principais (e não apenas da sua temperatura).

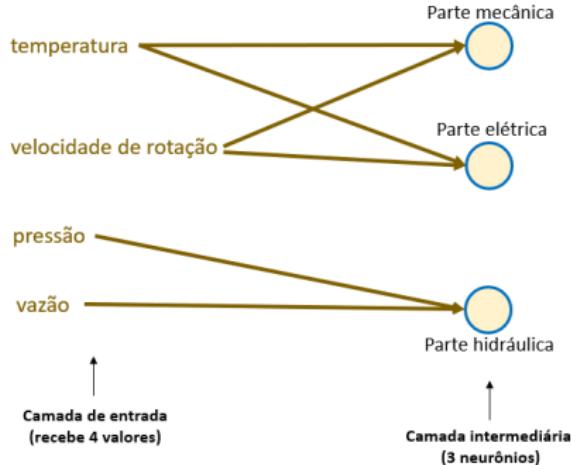
As características agora são:

- temperatura [$^{\circ}\text{C}$]
- velocidade de rotação [rpm]
- pressão [Pa]
- vazão [m^3/s]

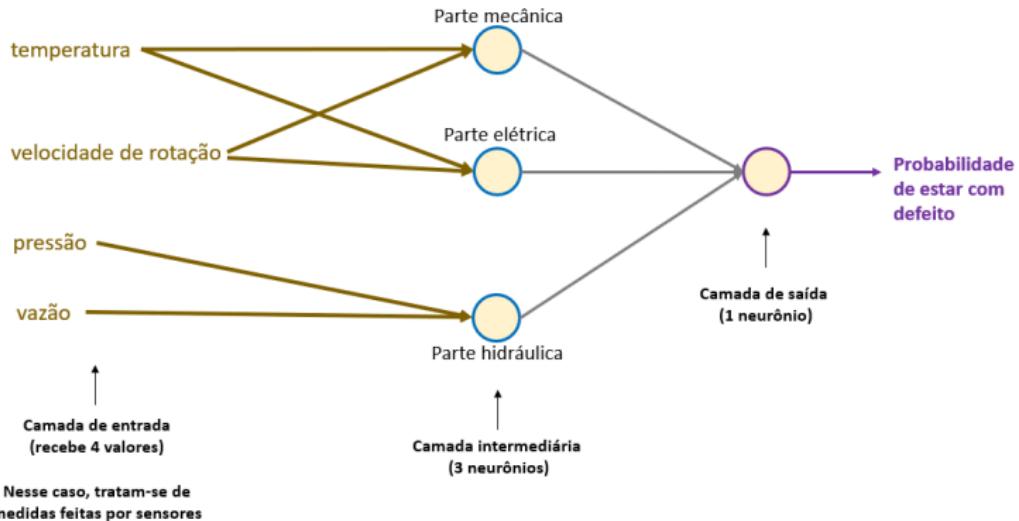
Pergunta:

Como poderíamos criar uma rede neural que descreve esse comportamento?

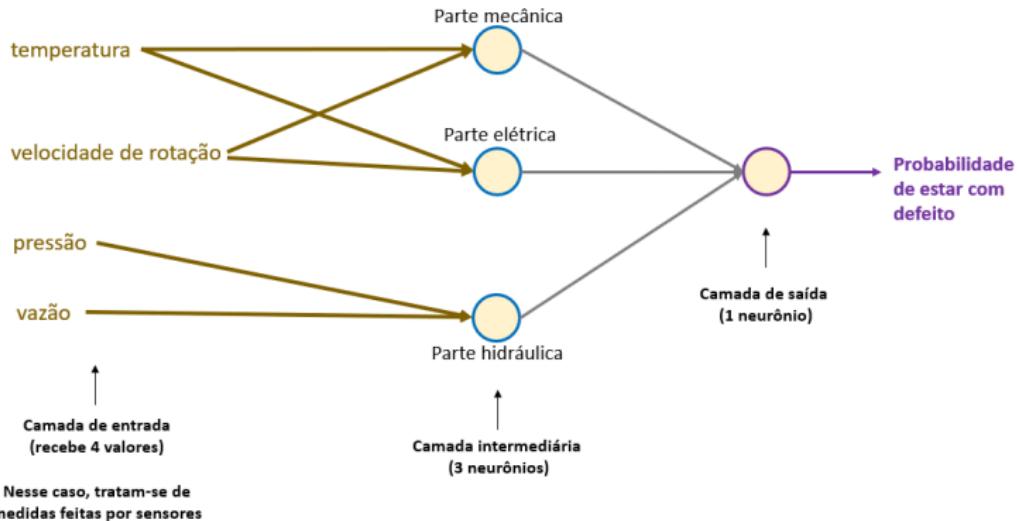
Exemplo: Equipamento com defeito



Exemplo: Equipamento com defeito

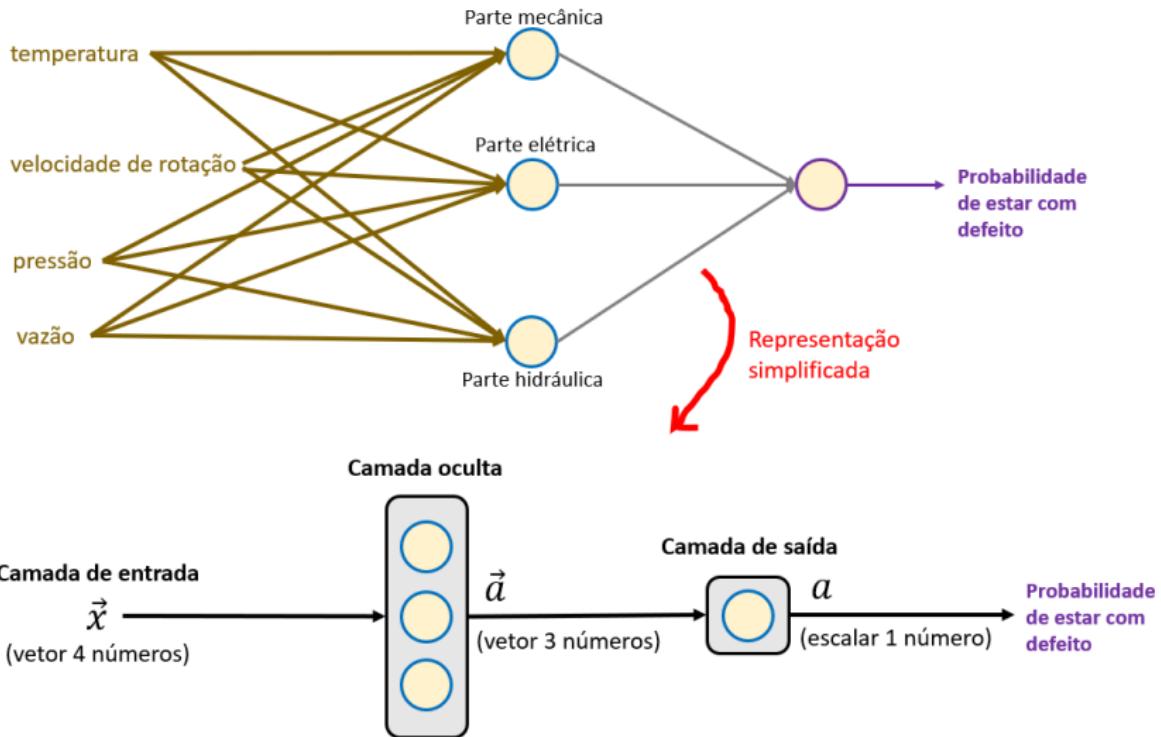


Exemplo: Equipamento com defeito

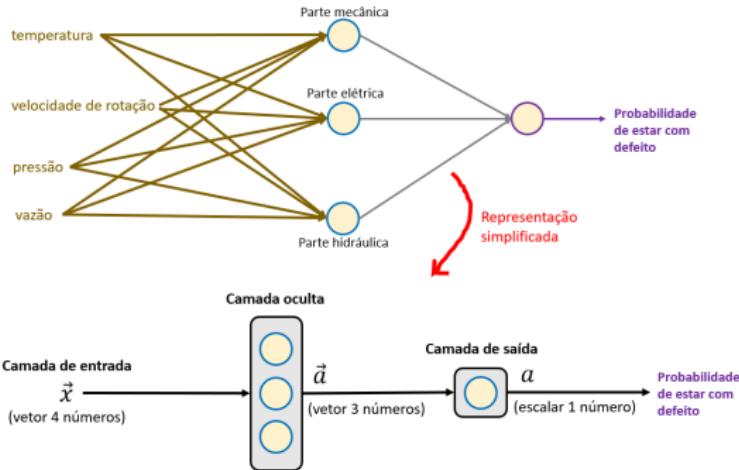


- A rede recebe 4 números por meio das suas 4 entradas
- Os três neurônios da Camada intermediária (também chamada de 'camada oculta') calculam suas 3 ativações correspondentes com base nos valores presentes em sua(s) entrada(s)
- Por fim, o neurônio da Camada de saída recebe esses 3 valores e calcula seu próprio valor único de ativação
- Cada neurônio pode ser entendido como sendo uma unidade elementar da regressão logística

Exemplo: Equipamento com defeito

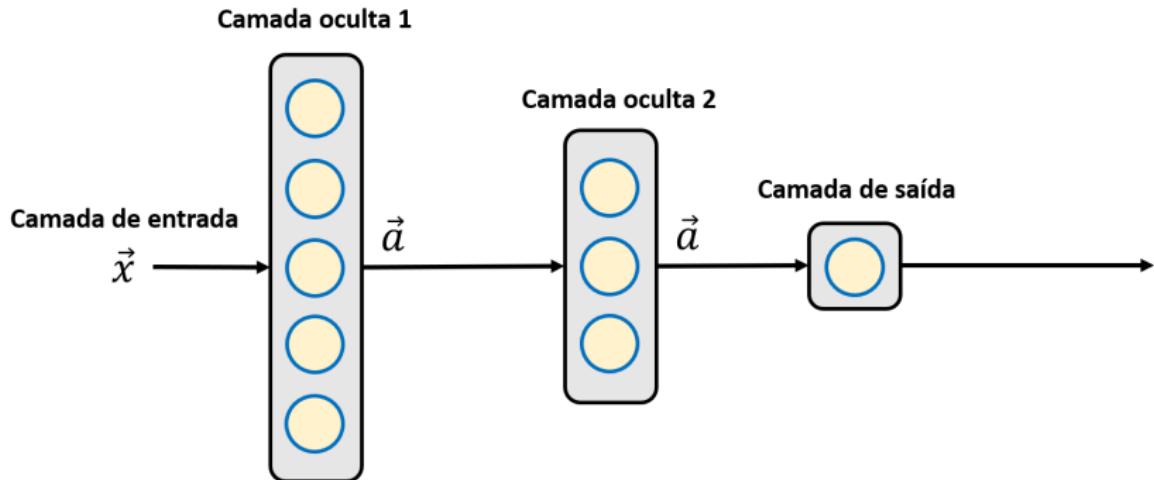


Exemplo: Equipamento com defeito



Simplificando a ideia:

- Na prática, é difícil identificar quais são as entradas mais relevantes para cada neurônio, igual nós fizemos para esse exemplo
- Por isso, nas redes neurais de fato usadas nós consideramos que cada neurônio tem acesso a todos os valores que vêm da camada anterior
- A ideia é que rede neural consiga aprender de forma autônoma quais características presentes nas suas entradas são relevantes e quais não são
- Ou seja, não é tão necessário fazer **Engenharia de Características** (como fizemos no caso da Regressão linear), já que a tendência é que a própria rede crie características intermediárias que ela acha que são mais apropriadas
- Isso faz com que as Redes Neurais sejam tão poderosas.



Perguntas:

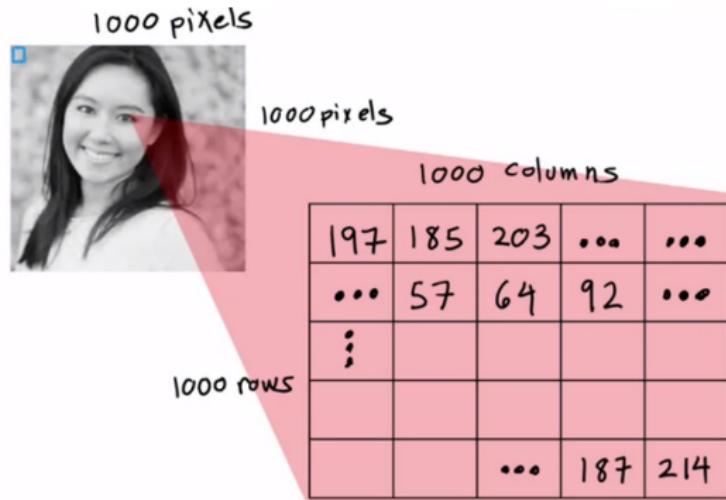
Quantas camadas escolher para a rede? Quantos neurônios colocar em cada camada?

Resposta:

Trata-se de um problema de **definição da arquitetura da rede**.

Um outro Exemplo: Reconhecimento facial

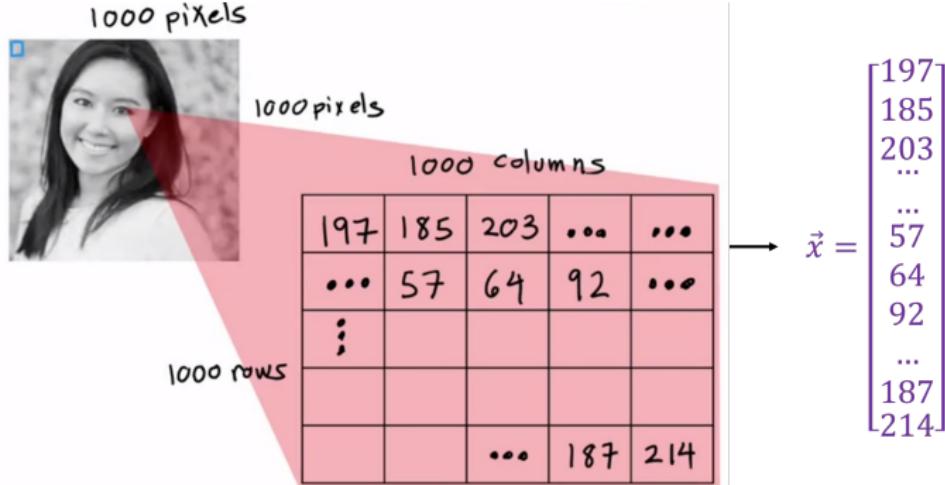
Suponha que você deseja identificar a identidade de uma pessoa a partir de uma imagem.



Fonte: **Machine Learning Specialization**, deeplearning.ai, Stanford Online, Coursera.org.

- Para o computador, a imagem é uma matriz de pixels (1000×1000 , nesse caso), onde cada pixel possui um próprio valor de intensidade de brilho (de 0 a 255, nesse caso)
- Seria possível montar um vetor de entrada para a nossa rede neural a partir da matriz?

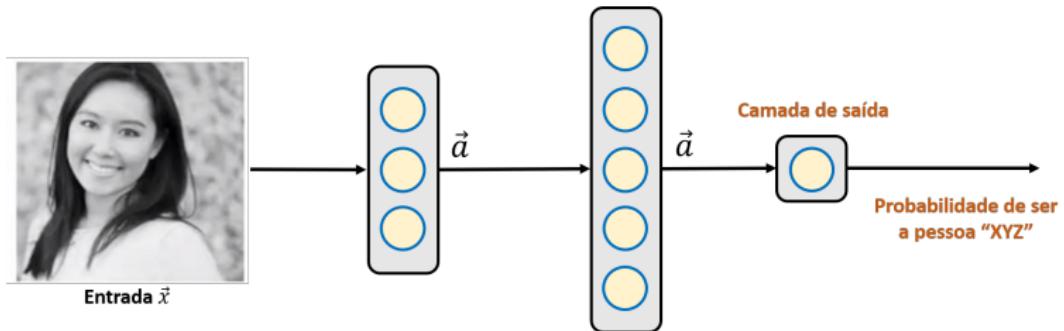
Um outro Exemplo: Reconhecimento facial



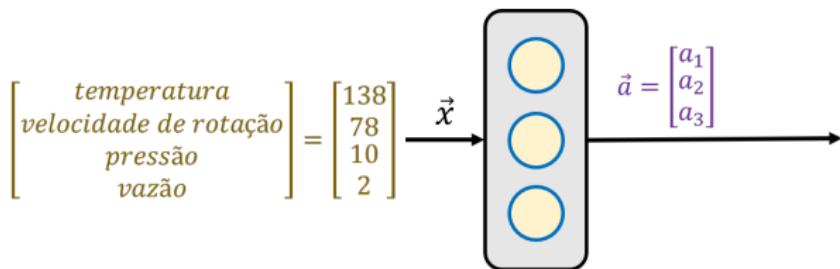
Fonte: **Machine Learning Specialization**, deeplearning.ai, Stanford Online, Coursera.org.

Sim, basta 'desenrolar' a matriz. Em inglês, esse procedimento é chamado de *flatten*

Um outro Exemplo: Reconhecimento facial

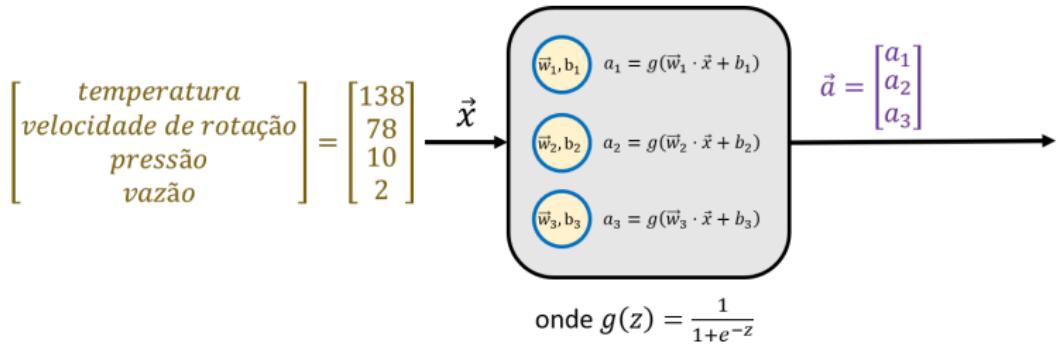


Voltando para o exemplo de defeito em equipamento (pensando apenas na camada oculta)



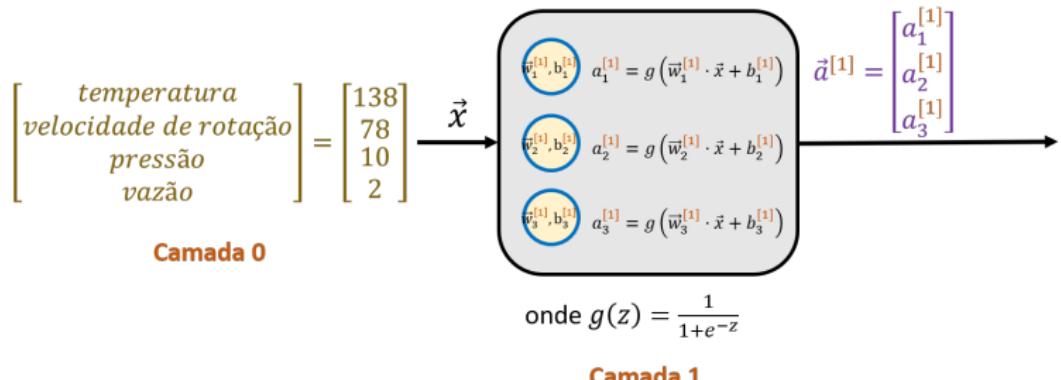
Perguntas:

- Como calcular a ativação de cada neurônio?
- Quantos valores o vetor \vec{a} terá?



Lembre-se:

- Cada neurônio consiste em uma unidade de regressão logística



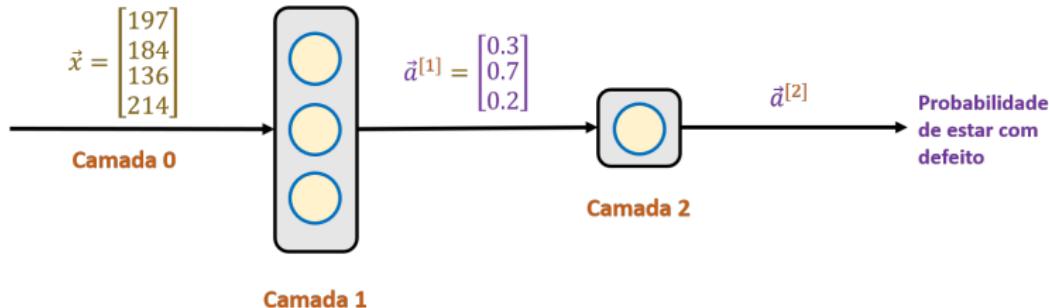
Importante!

- Usaremos o sobrescrito $[i]$ para nos referirmos a valores que correspondem à camada i .
- Pense também o seguinte: Os valores agora calculados para $\vec{a}^{[1]}$, por exemplo,

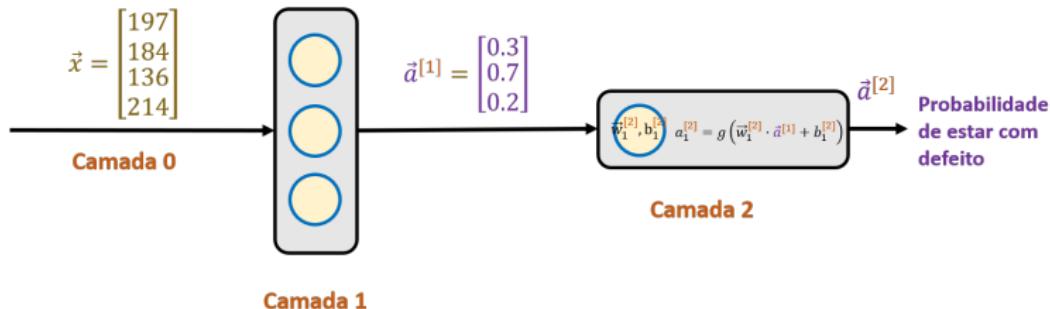
$$\vec{a}^{[1]} = \begin{bmatrix} 0.3 \\ 0.7 \\ 0.2 \end{bmatrix}$$

servem como valores de entrada para a camada que viria na sequência (camada 2).

Ou seja, por enquanto nós temos:



- Como calcular a ativação $\vec{a}^{[2]}$ do neurônio presente na Camada 2?

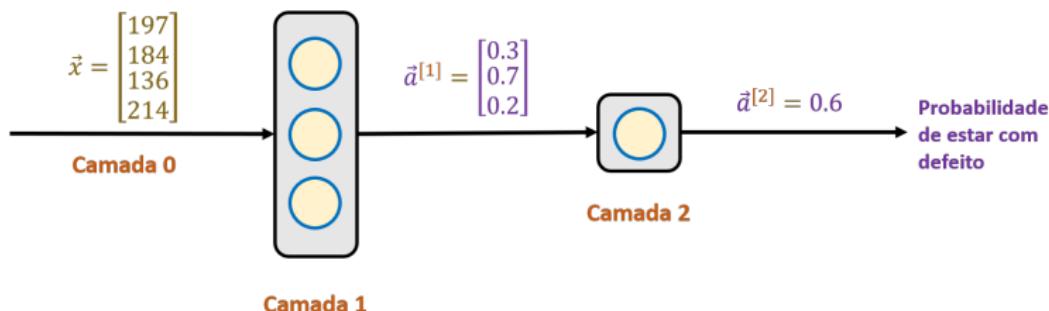


- Fazemos de forma análoga ao que havíamos feito para a Camada 1
- Note que a entrada para o neurônio da Camada 2 não é \vec{x} , mas sim $\vec{a}^{[1]}$.
- $\vec{a}^{[2]}$ vai resultar num valor entre 0 e 1, por exemplo,

$$\vec{a}^{[2]} = 0.6$$

Para o nosso exemplo, esse valor consiste na probabilidade do equipamento apresentar defeito.

Portanto, temos o seguinte resultado final:

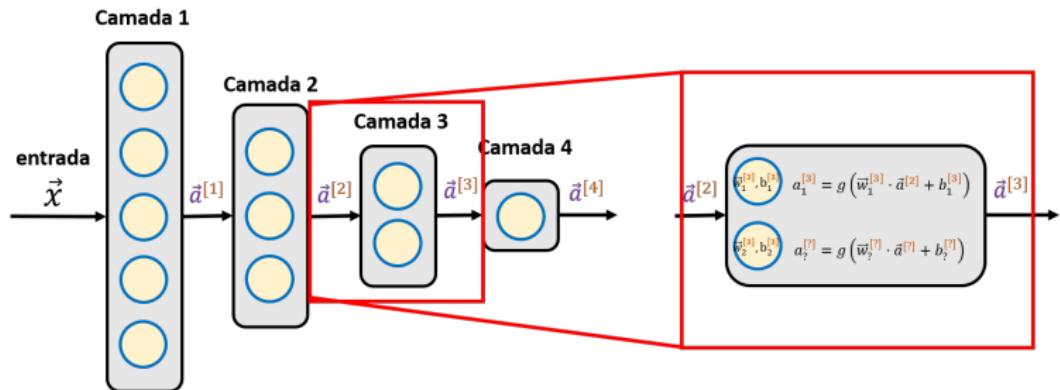


Pergunta:

E aí, esse equipamento está com defeito ou não?

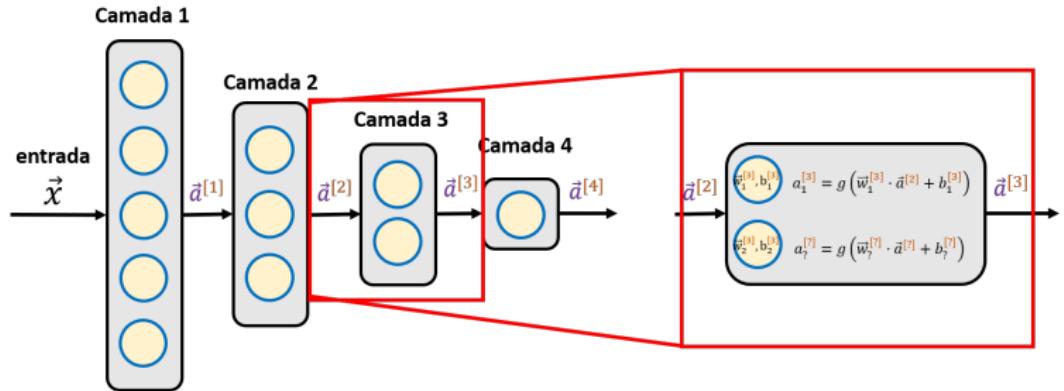
Resposta:

Depende. Se considerarmos um valor de limiar de 0.5, então podemos dizer que está sim com defeito ou na iminência de apresentar um defeito (manutenção preventiva).



Perguntas:

- A) Qual é a camada de entrada dessa rede? E sua camada de saída?
- B) Quantas camadas ocultas essa rede possui?
- C) Com relação à Camada 3, quais seriam os subscritos e sobrescritos para o seu segundo neurônio?



Podemos definir que a ativação gerada pelo j -ésimo neurônio da l -ésima camada é dada por

$$a_j^{[l]} = g(\vec{w}_j^{[l]} \cdot \vec{a}^{[l-1]} + b_j^{[l]})$$

Para que essa notação possa valer também para a Camada 1, definimos também

$$\vec{x} = \vec{a}^{[0]}$$

Vamos agora sistematizar o processo de realizar previsões a partir de uma rede neural já treinada.

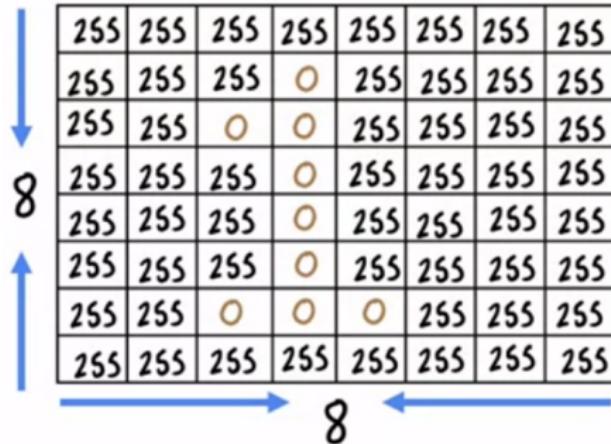
Quando usamos uma rede já treinada para fazermos previsões, trata-se de um problema de **inferência**.

Exemplo: Reconhecendo dígitos 0 e 1 escritos à mão

1 1	0 0	0 0	1 1	0 0	1 1	1 1	0 0	0 0
0 0	1 1	1 1	1 1	1 1	1 1	0 0	1 1	
1 1	0 0	1 1	1 1	1 1	1 1	0 0	0 0	
0 0	1 1	1 1	1 1	1 1	1 1	0 0	0 0	0 0

Exemplo: Reconhecendo dígitos 0 e 1 escritos à mão

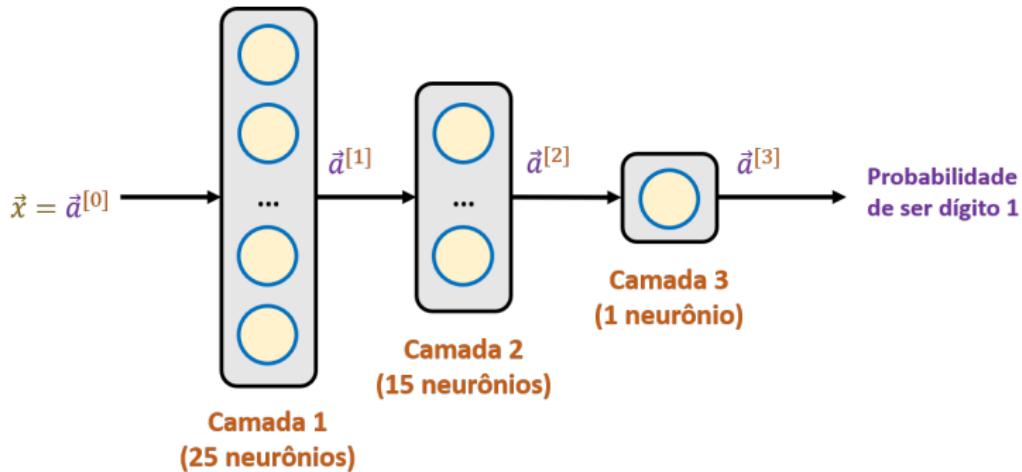
Considere que você tem uma imagem ($8 \times 8 = 64$ pixels) que possui o dígito 0 ou 1 escrito à mão:



- Valor 0 significa cor preta
- Valor 255 significa cor branca
- Valores entre 0 e 255 implicam em diferentes tons de cinza
- \vec{x} é o vetor (matriz desenrolada) contendo o valor de brilho para os 64 pixels

Exemplo: Reconhecendo dígitos 0 e 1 escritos à mão

Usaremos uma rede neural com a seguinte arquitetura:



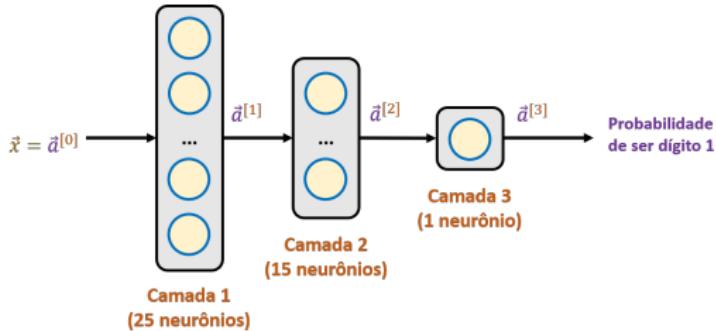
Pergunta:

Como calcular $\vec{d}^{[3]}$ a partir de $\vec{d}^{[0]} = \vec{x}$?

Resposta:

Usaremos a estratégia que acabamos de aprender (*forward propagation*) → iremos da esquerda para a direita, propagando os cálculos!

Exemplo: Reconhecendo dígitos 0 e 1 escritos à mão



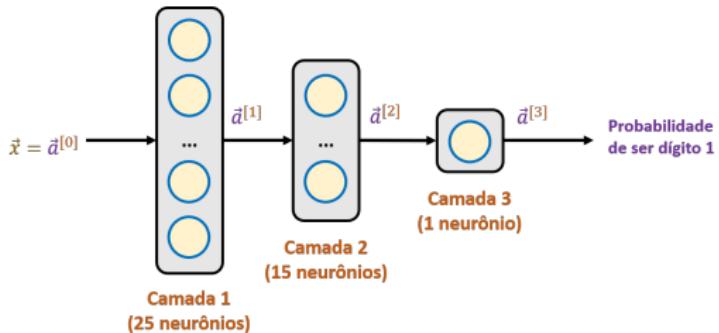
Para a Camada 1, temos:

$$\vec{d}^{[1]} = \begin{bmatrix} g(\vec{w}_1^{[1]} \cdot \vec{x} + b_1^{[1]}) \\ \dots \\ g(\vec{w}_{25}^{[1]} \cdot \vec{x} + b_{25}^{[1]}) \end{bmatrix}$$

Para a Camada 2, temos:

$$\vec{d}^{[2]} = \begin{bmatrix} g(\vec{w}_1^{[2]} \cdot \vec{d}^{[1]} + b_1^{[2]}) \\ \dots \\ g(\vec{w}_{15}^{[2]} \cdot \vec{d}^{[1]} + b_{15}^{[2]}) \end{bmatrix}$$

Exemplo: Reconhecendo dígitos 0 e 1 escritos à mão



Finalmente, para a Camada 3, temos:

$$\vec{a}^{[3]} = g(\vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]})$$

se $a^{[3]} \geq 0.5$, $\hat{y} = 1$ (imagem contém dígito 1).

se $a^{[3]} < 0.5$, $\hat{y} = 0$ (imagem contém dígito 0).

De olho no código!

Vamos agora explorar o conhecimento básico adquirido sobre neurônios e camadas usando código.

Clique no link abaixo para acessar o código:

https://colab.research.google.com/github/xaximppv2/master/blob/main/codigo_aula_15_neuronios_e_camadas.ipynb

OBS: Nesse código, vamos aproveitar para iniciar nossos estudos em cima do pacote **TensorFlow**, que consiste numa ferramenta de machine learning bastante utilizada.

Parte 1

Rode todo o código. Responda às questões nele contidas e complete-o, se necessário.

OBS: Caso esse seja seu primeiro contato com **TensorFlow**, é normal que demore um tempo até que você concorde com o que está sendo feito e explicado no código.

Parte 2

- 1 Adicione amostras tanto ao problema de regressão como ao problema de classificação (sem comprometer de forma significativa a capacidade do modelo de explicar esses dados)

Inferência usando Redes Neurais (*forward propagation*)

Tensorflow



Na aula anterior, fizemos uma introdução sobre redes neurais e estudamos o nosso primeiro código usando Tensorflow.

Nesta aula, iremos ver com mais detalhes como calcular o valor de saída de uma rede neural, supondo que todos os seus parâmetros já sejam conhecidos antecipadamente (problema de **inferência**)

Na aula anterior, fizemos uma introdução sobre redes neurais e estudamos o nosso primeiro código usando Tensorflow.

Nesta aula, iremos ver com mais detalhes como calcular o valor de saída de uma rede neural, supondo que todos os seus parâmetros já sejam conhecidos antecipadamente (problema de **inferência**)

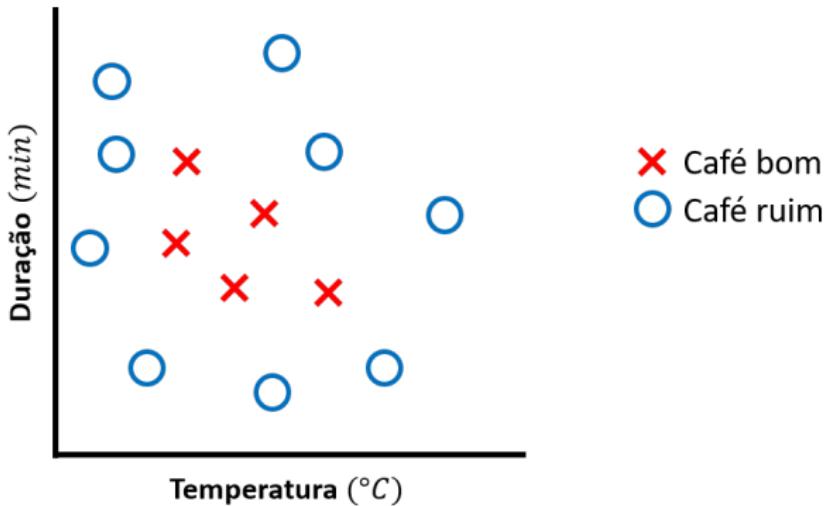
OBS: Nesta aula, usaremos o **Tensorflow** como um facilitador para resolvemos o problema de inferência. Na próxima aula, faremos isso usando apenas o NumPy.

Exemplo: torrefação de café



- Um algoritmo de aprendizado de máquina seria capaz de otimizar a qualidade do café que resulta do processo de torrefação?
- Dois parâmetros influenciam bastante na qualidade do café resultante: **Temperatura e tempo de torrefação.**

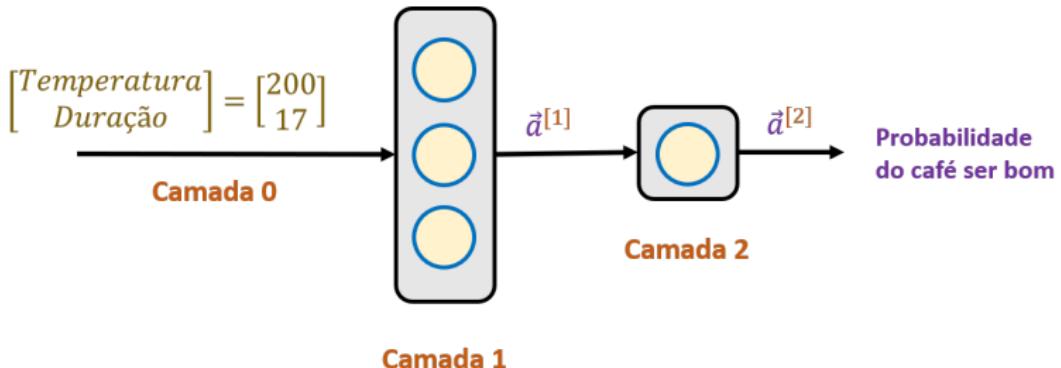
Exemplo: torrefação de café



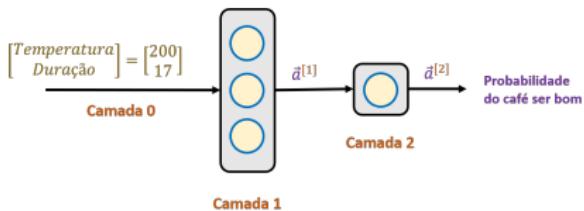
- Se fica pouco tempo torrando, o café fica sub-torrado
- Se é torrado com uma temperatura muito baixa, o café também fica sub-torrado.
- Se a duração ou a temperatura forem muito altas, o café fica sobre-torrado.
- Com isso, temos um pequeno triângulo que corresponde à região onde o resultado é satisfatório.

Exemplo: torrefação de café

Seria possível ter uma rede neural que consegue determinar se o café é bom com base num conjunto (temperatura, duração)?



Exemplo: torrefação de café



Usando Tensorflow para calcular as saídas da Camada 1

```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation='sigmoid')
a1 = layer_1(x)
```

Usando Tensorflow para calcular a saída da Camada 2

```
layer_2 = Dense(units=1, activation='sigmoid')
a2 = layer_2(a1)
```

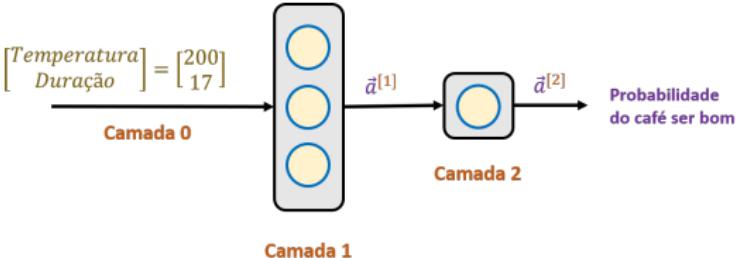
Fazendo a previsão final

```
if a2 >= 0.5:
    yhat = 1
else:
    yhat = 0
```

- NumPy é uma biblioteca para cálculos matriciais, etc
- Tensorflow é um pacote específico para Aprendizado de Máquina.

Observações

- É importante sabermos como os dados são representados tanto em Numpy como também no Tensorflow



Cálculos para a primeira camada:

```
x = np.array([[200.0, 17.0]])  
layer_1 = Dense(units=3, activation='sigmoid')  
a1 = layer_1(x)
```

Numpy representando uma matriz (1,2)

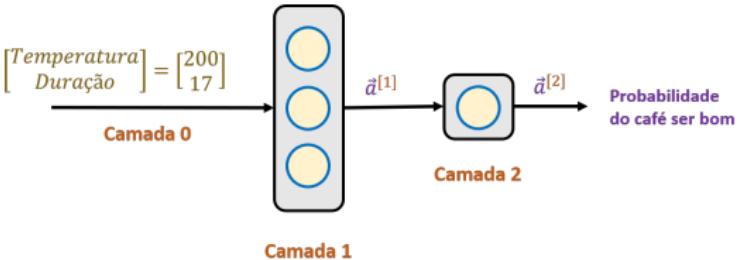
```
tf.Tensor([[0.2 0.7 0.3]], shape=(1, 3), dtype=float32)
```

Tensorflow representando uma matriz (1,3)

```
a1.numpy()
```

```
array([[0.2, 0.7, 0.3]], dtype=float32)
```

Convertendo da representação Tensorflow para NumPy



Idem para a Camada 2:

```
layer_2 = Dense(units=1, activation='sigmoid')  
a2 = layer_2(a1)
```

`tf.Tensor([[0.8]], shape=(1, 1), dtype=float32)` → Tensorflow representando uma matriz (1,1)

```
a2.numpy()
```

`array([[0.8]], dtype=float32)` → Convertendo da representação Tensorflow para NumPy

Observações finais:

- Tecnicamente, quando passamos uma **array** criada pelo NumPy para o Tensorflow, ele gosta de criar sua própria forma de representar esses dados (ele chama arrays pelo termo **tensores**)
- Podemos converter Tensores em arrays (e vice-versa) facilmente.

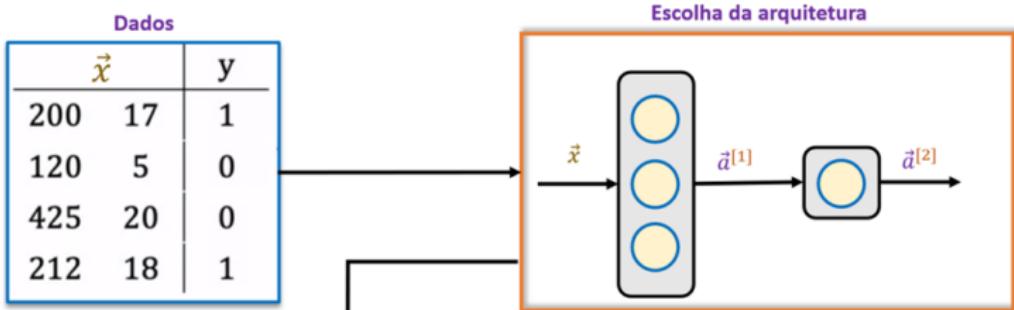
- Ao invés de fazermos os cálculos camada por camada, podemos criar o nosso modelo completo (nossa rede neural completa) ainda mais facilmente.

```
layer_1 = Dense(units=3, activation="sigmoid")
layer_2 = Dense(units=1, activation="sigmoid")
model = Sequential([layer_1, layer_2])
```

OU

```
model = Sequential([
    Dense(units=3, activation="sigmoid"),
    Dense(units=1, activation="sigmoid")])
```

Exemplo completo: Criando e treinando o modelo



Criando modelo e treinando:

```
layer_1 = Dense(units=3, activation="sigmoid")
layer_2 = Dense(units=1, activation="sigmoid")
model = Sequential([layer_1, layer_2])
x = np.array([[200.0, 17.0],
              [120.0, 5.0],
              [425.0, 20.0],
              [212.0, 18.0]])
y = np.array([1,0,0,1])
model.compile(...)
```

→ Veremos depois!

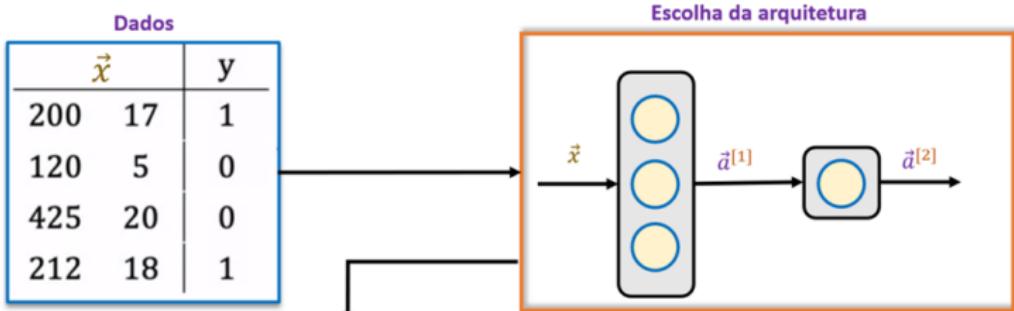
```
model.fit(x,y)
```

→ Treina todos os parâmetros!

Fazendo previsão para novo par (temperatura, duração)

```
model.predict(x_new)
```

Exemplo completo: Criando e treinando o modelo



Criando modelo e treinando:

```
layer_1 = Dense(units=3, activation="sigmoid")
layer_2 = Dense(units=1, activation="sigmoid")
model = Sequential([layer_1, layer_2])
x = np.array([[200.0, 17.0],
              [120.0, 5.0],
              [425.0, 20.0],
              [212.0, 18.0]])
y = np.array([1,0,0,1])
model.compile(...)      → Veremos depois!
model.fit(x,y)         → Treina todos os parâmetros!
```

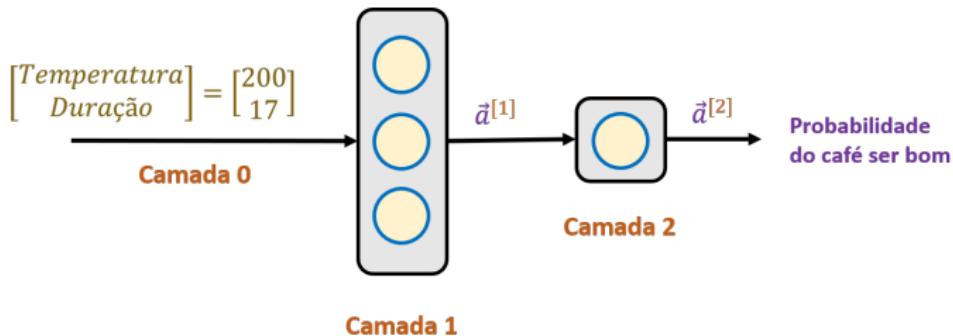
Fazendo previsão para novo par (temperatura, duração)

```
model.predict(x_new)
```

Observação:

Note que estamos criando e treinando uma rede neural complexa (com uma ferramenta "estado da arte") usando apenas algumas **poucas linhas de código**. → É necessário saber o que está acontecendo por trás?

- Mas afinal de contas, quais são os parâmetros da rede neural abaixo?
- Qual é o número parâmetros por camada?
- E o número total de parâmetros?



De olho no código!

Vamos agora construir e treinar uma rede neural aplicada ao problema de torrefação de café (usando Tensorflow).

Acesse o Python Notebook usando o QR code ou o link abaixo:

https://colab.research.google.com/github/xaximpvp2/master/blob/main/codigo_aula15_Torrando_cafe_com_tensorflow.ipynb



Parte 1

Rode todo o código. Responda às questões nele contidas e complete-o, se necessário.

Parte 2

- 1 Testando sistematicamente diversos valores para os dados de entrada (Temperatura, Duração), busque estabelecer **graficamente** a fronteira de decisão definida pela rede neural treinada. O que essa fronteira de decisão representa?

Dica: Lembre-se que, na fronteira de decisão, ocorrem transições do tipo $\hat{y} = 1 \rightarrow \hat{y} = 0$ ou o contrário

Inferência usando Redes Neurais (*forward propagation*)

Tensorflow versus NumPy



Na aula anterior, usamos o Tensorflow como um facilitador para resolvemos o problema de inferência.

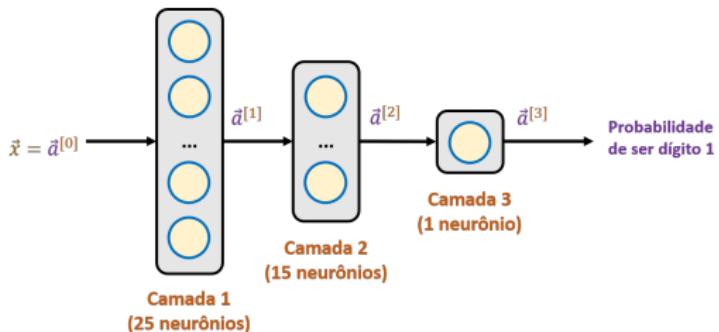
Pergunta:

Se você tivesse que implementar a **propagação para frente** do zero em Python, sem usar Tensorflow, como você faria?

OBS: Alguém criou o ambiente Tensorflow, assim como o ambiente PyTorch. E se você quisesse criar um ambiente equivalente?

Exemplo: Reconhecendo dígitos 0 e 1 escritos à mão

Em uma aula anterior nossa, vimos o seguinte exemplo:



Para a Camada 1, temos:

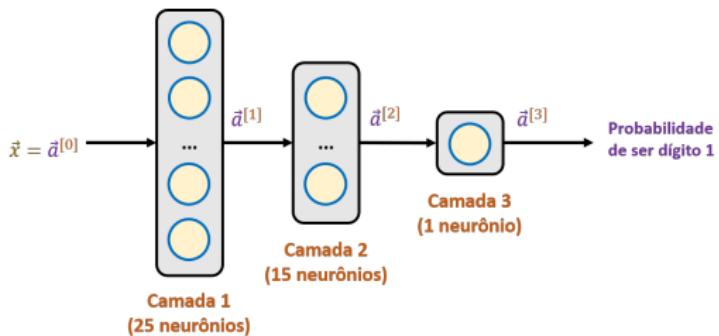
$$\vec{a}^{[1]} = \begin{bmatrix} g(\vec{w}_1^{[1]} \cdot \vec{x} + b_1^{[1]}) \\ \vdots \\ g(\vec{w}_{25}^{[1]} \cdot \vec{x} + b_{25}^{[1]}) \end{bmatrix}$$

Para a Camada 2, temos:

$$\vec{a}^{[2]} = \begin{bmatrix} g(\vec{w}_1^{[2]} \cdot \vec{a}^{[1]} + b_1^{[2]}) \\ \vdots \\ g(\vec{w}_{15}^{[2]} \cdot \vec{a}^{[1]} + b_{15}^{[2]}) \end{bmatrix}$$

Exemplo: Reconhecendo dígitos 0 e 1 escritos à mão

Em uma aula anterior nossa, vimos o seguinte exemplo:



Finalmente, para a Camada 3, temos:

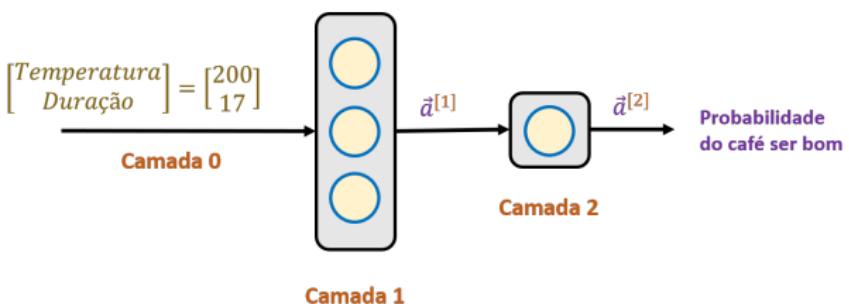
$$\vec{a}^{[3]} = g(\vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]})$$

se $a^{[3]} \geq 0.5$, $\hat{y} = 1$ (imagem contém dígito 1).

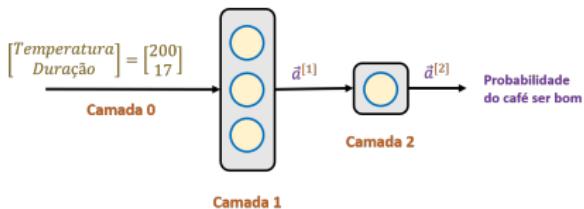
se $a^{[3]} < 0.5$, $\hat{y} = 0$ (imagem contém dígito 0).

Exemplo: torrefação de café

Como ficaria para o exemplo de Torrefação de café abaixo?



Exemplo: torrefação de café



Para a Camada 1, temos:

$$\vec{a}^{[1]} = \begin{bmatrix} g(\vec{w}_1^{[1]} \cdot \vec{x} + b_1^{[1]}) \\ g(\vec{w}_2^{[1]} \cdot \vec{x} + b_2^{[1]}) \\ g(\vec{w}_3^{[1]} \cdot \vec{x} + b_3^{[1]}) \end{bmatrix}$$

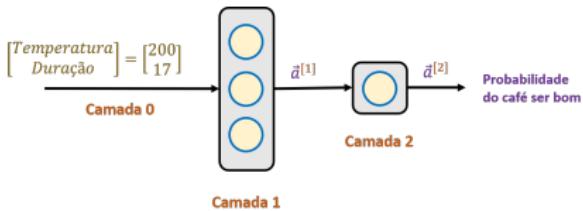
Para a Camada 2, temos:

$$\vec{a}^{[2]} = g(\vec{w}_1^{[2]} \cdot \vec{a}^{[1]} + b_1^{[2]})$$

se $a^{[2]} \geq 0.5$, $\hat{y} = 1$ (café bom).

se $a^{[2]} < 0.5$, $\hat{y} = 0$ (café ruim).

Exemplo: torrefação de café



Pergunta: O que cada linha de código abaixo faz?

```
x = np.array([200, 17])
```

```
w1_1 = np.array([1, 2])
b1_1 = np.array([-1])
z1_1 = np.dot(w1_1,x)+b1_1
a1_1 = sigmoid(z1_1)
```

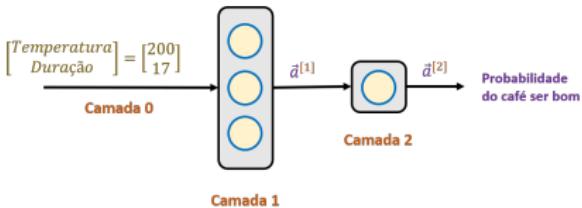
```
w1_2 = np.array([-3, 4])
b1_2 = np.array([1])
z1_2 = np.dot(w1_2,x)+b1_2
a1_2 = sigmoid(z1_2)
```

```
w1_3 = np.array([5, -6])
b1_3 = np.array([-1])
z1_3 = np.dot(w1_3,x)+b1_3
a1_3 = sigmoid(z1_3)
```

```
a1 = np.array([a1_1, a1_2, a1_3])
```

```
w2_1 = np.array([1, 2, 3])
b2_1 = np.array([1])
z2_1 = np.dot(w2_1,a1)+b2_1
a2_1 = sigmoid(z2_1)
```

Exemplo: torrefação de café



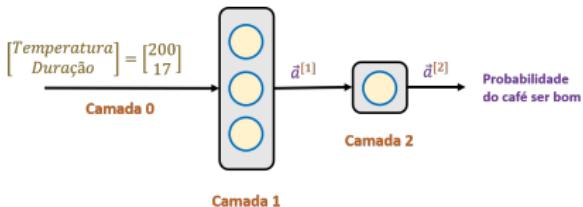
Generalizando o conceito de propagação para frente:

```
x = np.array([200, 17])  
  
W1 = np.array([[ 1, -3,  5],  
              [ 2,  4, -6]])  
b1 = np.array([-1,  1, -1])  
W2 = np.array([[1],[2],[3]])  
b2 = np.array([1])  
  
def dense(a_in,W,b):  
    unidades = W.shape[1]  
    a_out = np.zeros(unidades)  
    for j in range(unidades):  
        w = W[:,j]  
        z = np.dot(w,a_in) + b[j]  
        a_out[j] = sigmoid(z)  
    return a_out  
  
def sequential(x):  
    a1 = dense(x,W1,b1)  
    a2 = dense(a1,W2,b2)  
    f_x = a2  
    return f_x
```

Annotations explaining the code:

- Annotations point to the weight matrices W_1 and W_2 , and bias vectors b_1 and b_2 :
 - "Estoca todos os parâmetros da camada 1"
 - "Estoca todos os parâmetros da camada 2"
- An annotation points to the `dense` function definition: "Definindo função que calcula saída de uma dense = camada"
- An annotation points to the `return a_out` statement: "Retornando saída calculada da camada"
- An annotation points to the `sequential` function definition: "Calculando sequencialmente as saídas das camadas"

Exemplo: torrefação de café



Implementação equivalente usando Vetorização (operações matriciais):

```
X = np.array([[200, 17]]) → Aqui, todas as variáveis são matrizes, ou seja, arrays 2D  
W1 = np.array([[1, -3, 5],  
              [2, 4, -6]])  
B1 = np.array([-1, 1, -1])  
W2 = np.array([[1],[2],[3]])  
B2 = np.array([1])  
  
def minha_camada_densa_vetorizada(A_in,W,B):  
    Z = np.matmul(A_in,W)+B → Como funciona essa multiplicação de matrizes?  
    A_out = sigmoid(Z)  
    return A_out  
  
def sequencial(X):  
    A1 = minha_camada_densa_vetorizada(X,W1,B1)  
    A2 = minha_camada_densa_vetorizada(A1,W2,B2)  
    F_X = A2  
    return F_X
```

Sejam os dois vetores abaixo

$$\vec{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad \text{e} \quad \vec{w} = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

Calcule:

- O produto escalar $\vec{a} \cdot \vec{w}$
- O transposto de \vec{a} , ou seja, \vec{a}^T
- O transposto de \vec{w} , ou seja, \vec{w}^T
- A multiplicação vetor-vetor $\vec{a}^T \vec{w}$

Sejam as duas matrizes abaixo:

$$A = \begin{bmatrix} 200 & 17 \end{bmatrix} \quad \text{e} \quad W = \begin{bmatrix} 1 & -3 & 5 \\ 2 & 4 & -6 \end{bmatrix}$$

Calcule:

- A é um vetor ou uma matriz?
- Qual é o número de linhas e de colunas da matriz A ? E da matriz W ?
- É possível realizar a multiplicação matriz-matriz AW ?
- Seja a multiplicação $C = AW$. Qual será a dimensão de C ?
- Quanto vale $C = AW$?

Calcule $f(\vec{x})$ para os dois códigos abaixo

```
x = np.array([200, 17])

W1 = np.array([[ 1, -3,  5],
              [ 2,  4, -6]])
b1 = np.array([-1, 1, -1])
W2 = np.array([[1],[2],[3]])
b2 = np.array([1])

def dense(a_in,W,b):
    unidades = W.shape[1]
    a_out = np.zeros(unidades)
    for j in range(unidades):
        w = W[:,j]
        z = np.dot(w,a_in) + b[j]
        a_out[j] = sigmoid(z)
    return a_out

def sequential(x):
    a1 = dense(x,W1,b1)
    a2 = dense(a1,W2,b2)
    f_x = a2
    return f_x
```

versus

```
X = np.array([200, 17])

W1 = np.array([[1, -3, 5],
              [2, 4, -6]])
B1 = np.array([-1, 1, -1])
W2 = np.array([[1],[2],[3]])
B2 = np.array([1])

def minha_camada_dense_vetorizada(A_in,W,B):
    Z = np.matmul(A_in,W)+B
    A_out = sigmoid(Z)
    return A_out

def sequencial(X):
    A1 = minha_camada_dense_vetorizada(X,W1,B1)
    A2 = minha_camada_dense_vetorizada(A1,W2,B2)
    F_X = A2
    return F_X
```

Resposta:

Sejam as três matrizes abaixo:

$$X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} -1 & -1 \end{bmatrix}$$

Pergunta:

- É possível somar $X + b$?

Observação:

- O NumPy possui uma capacidade de **broadcasting** incrível. Entretanto, devemos ficar atentos para verificar se estamos entendendo corretamente a operação que está sendo de fato realizada.

De olho no código!

Usando Numpy, no código a seguir iremos implementar a propagação para frente para o problema de reconhecimento de dígitos escritos à mão.

Entretanto, o código se inicia usando o Tensorflow para encontrar parâmetros adequados para o modelo.

Acesse o Python Notebook usando o QR code ou o link abaixo:

https://colab.research.google.com/github/xaximppv2/master/blob/main/codigo_aula_16_reconhecendo_digito_0_e_1_escritos_a_mao.ipynb



Clique no link abaixo para baixar os dados necessários para rodar o código:

https://ufprbr0-my.sharepoint.com/:f/g/personal/ricardo_schumacher_ufpr_br/EvZlARkJilxEpaj_S4kjkskBQDohnTWgZxqKRd3qxrE9Pw?e=dEZ6az

OBS: Para adicionar os dados ao ambiente do Colab Notebook, no menu do canto esquerdo da tela do Colab clique em "Arquivos" e depois "Fazer upload para o armazenamento da sessão". Então carregue os arquivos baixados.

Parte 1

Rode todo o código. Responda às questões nele contidas e complete-o, se necessário.

Parte 2

- 1 Inclua no código um comando que calcula a taxa de acerto da rede neural. Qual foi a taxa de acerto obtida? Quantas amostras o modelo classificou erroneamente?

OBS: Lembre-se do comando `(np.mean(Y == ychapeu) * 100)` utilizado na atividade de programação anterior.

Mais sobre Tensorflow e funções de ativação



Nas aulas anteriores, aprendemos:

- como fazer previsões usando uma rede neural já treinada no NumPy ou no Tensorflow. → **(problema de propagação para frente - inferência)**
- Também aprendemos a treinar um modelo usando Tensorflow

Nas aulas anteriores, aprendemos:

- como fazer previsões usando uma rede neural já treinada no NumPy ou no Tensorflow. → **(problema de propagação para frente - inferência)**
- Também aprendemos a treinar um modelo usando Tensorflow

Pergunta:

Como o Tensorflow faz para encontrar valores adequados para os parâmetros $w_j^{[l]}, b^{[l]}$?

Relembrando da Regressão Logística

Passo 1: Definição do modelo

$$f_{\vec{w}, b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + b)}}$$

Pergunta: Quais são as características desse modelo?

Relembrando da Regressão Logística

Passo 1: Definição do modelo

$$f_{\vec{w}, b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + b)}}$$

Pergunta: Quais são as características desse modelo?

Passo 2: Definição da função perda e da função custo

$$L(f_{\vec{w}, b}(\vec{x}^{(i)}), y^{(i)}) = -y^{(i)} \log(f_{\vec{w}, b}(\vec{x}^{(i)})) - (1 - y^{(i)}) \log(1 - f_{\vec{w}, b}(\vec{x}^{(i)}))$$

$L(f_{\vec{w}, b}(\vec{x}^{(i)}), y^{(i)})$ é a função de **perda logística**, e mede quanto o modelo $f_{\vec{w}, b}(\vec{x}^{(i)})$ está errado em relação ao rótulo verdadeiro $y^{(i)}$ para uma dada amostra i . É também chamada de **função de entropia cruzada binária**

$$J(\vec{w}, b) = \frac{1}{m} \sum_{i=1}^m L(f_{\vec{w}, b}(\vec{x}^{(i)}), y^{(i)}) \quad \rightarrow \quad \text{Função custo: média das perdas}$$

Relembrando da Regressão Logística

Passo 1: Definição do modelo

$$f_{\vec{w}, b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + b)}}$$

Pergunta: Quais são as características desse modelo?

Passo 2: Definição da função perda e da função custo

$$L(f_{\vec{w}, b}(\vec{x}^{(i)}), y^{(i)}) = -y^{(i)} \log(f_{\vec{w}, b}(\vec{x}^{(i)})) - (1 - y^{(i)}) \log(1 - f_{\vec{w}, b}(\vec{x}^{(i)}))$$

$L(f_{\vec{w}, b}(\vec{x}^{(i)}), y^{(i)})$ é a função de **perda logística**, e mede quanto o modelo $f_{\vec{w}, b}(\vec{x}^{(i)})$ está errado em relação ao rótulo verdadeiro $y^{(i)}$ para uma dada amostra i . É também chamada de **função de entropia cruzada binária**

$$J(\vec{w}, b) = \frac{1}{m} \sum_{i=1}^m L(f_{\vec{w}, b}(\vec{x}^{(i)}), y^{(i)}) \quad \rightarrow \quad \text{Função custo: média das perdas}$$

Passo 3: Minimizar a função custo

Encontrar os valores de \vec{w}, b que minimizam $J(\vec{w}, b)$ pelo Método do Gradiente:

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b) \quad j = 1, \dots, n$$

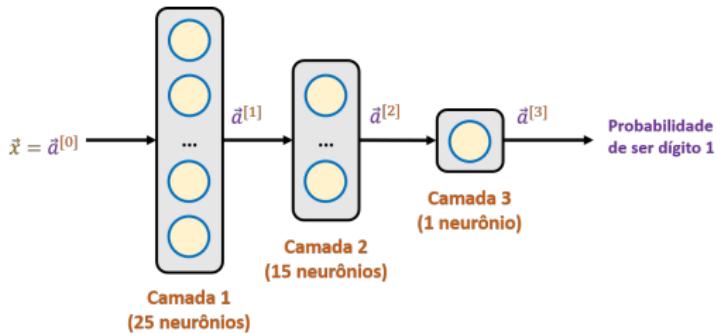
$$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b)$$

E para as redes neurais, como fica?

E para as redes neurais, como ficam esses 3 passos?

Passo 1: Definição do modelo

Qual seria o modelo $f_{W,B}(\vec{x})$ para a rede neural abaixo?



$$\vec{a}^{[1]} = \begin{bmatrix} g(\vec{w}_1^{[1]} \cdot \vec{x} + b_1^{[1]}) \\ \dots \\ g(\vec{w}_{25}^{[1]} \cdot \vec{x} + b_{25}^{[1]}) \end{bmatrix} \rightarrow \vec{a}^{[2]} = \begin{bmatrix} g(\vec{w}_1^{[2]} \cdot \vec{a}^{[1]} + b_1^{[2]}) \\ \dots \\ g(\vec{w}_{15}^{[2]} \cdot \vec{a}^{[1]} + b_{15}^{[2]}) \end{bmatrix}$$

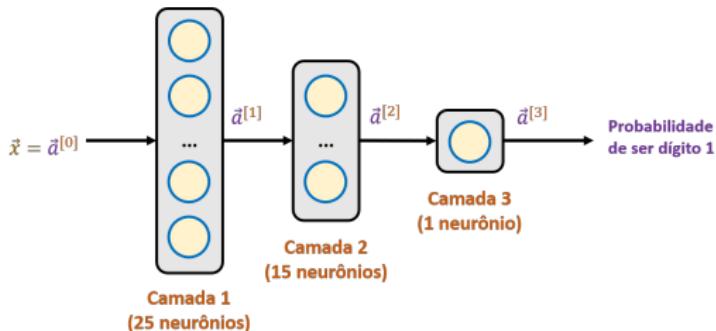
Finalmente:

$$\vec{a}^{[3]} = f_{W,B}(\vec{x}) = g(\vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]})$$

OBS: Na representação acima, W, B representam todos os parâmetros da rede neural.

Passo 1: Definição do modelo

Qual seria o modelo $f_{W,B}(\vec{x})$ para a rede neural abaixo?



No Tensorflow, fazemos a definição desse modelo a partir do seguinte código:

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

model = Sequential([
    Dense(units=25, activation='sigmoid'),
    Dense(units=15, activation='sigmoid'),
    Dense(units=1, activation='sigmoid'),
```

Passo 2: Definição da função perda e da função custo

$$L(f_{W,B}(\vec{x}^{(i)}), y^{(i)}) = -y^{(i)} \log(f_{W,B}(\vec{x}^{(i)})) - (1 - y^{(i)}) \log(1 - f_{W,B}(\vec{x}^{(i)}))$$

Ou seja, é a mesma função perda usada na Regressão Logística (**função de entropia cruzada binária**)

$$J(W, B) = \frac{1}{m} \sum_{i=1}^m L(f_{W,B}(\vec{x}^{(i)}), y^{(i)}) \quad \rightarrow \quad \text{Função custo: média das perdas}$$

No Tensorflow, fazemos essa definição compilando o modelo da seguinte forma:

```
model.compile(loss= BinaryCrossentropy()) from tensorflow.keras.losses import  
BinaryCrossentropy
```

Keras

Passo 2: Definição da função perda e da função custo

$$L(f_{W,B}(\vec{x}^{(i)}), y^{(i)}) = -y^{(i)} \log(f_{W,B}(\vec{x}^{(i)})) - (1 - y^{(i)}) \log(1 - f_{W,B}(\vec{x}^{(i)}))$$

Ou seja, é a mesma função perda usada na Regressão Logística (**função de entropia cruzada binária**)

$$J(W, B) = \frac{1}{m} \sum_{i=1}^m L(f_{W,B}(\vec{x}^{(i)}), y^{(i)}) \quad \rightarrow \quad \text{Função custo: média das perdas}$$

No Tensorflow, fazemos essa definição compilando o modelo da seguinte forma:

```
model.compile(loss= BinaryCrossentropy()) from tensorflow.keras.losses import  
BinaryCrossentropy
```

Keras

Alternativamente, para problemas de regressão, podemos usar:

$$J(W, B) = \frac{1}{m} \sum_{i=1}^m (f_{W,B}(\vec{x}^{(i)}) - y^{(i)})^2 \quad \rightarrow \quad \text{erro quadrático médio}$$

```
model.compile(loss= MeanSquaredError()) from tensorflow.keras.losses import  
MeanSquaredError
```

Passo 3: Minimizar a função custo

Encontrar os valores de $w_j^{[l]}, b^{[l]}$ que minimizam $J(W, B)$ pelo Método do Gradiente:

$$w_j^{[l]} = w_j^{[l]} - \alpha \frac{\partial}{\partial w_j^{[l]}} J(W, B)$$

$$b^{[l]} = b^{[l]} - \alpha \frac{\partial}{\partial b^{[l]}} J(W, B)$$

Observação:

As derivadas acima podem ser calculadas usando uma metodologia conhecida como **propagação para trás** (*back propagation*)

Passo 3: Minimizar a função custo

Encontrar os valores de $w_j^{[l]}, b^{[l]}$ que minimizam $J(W, B)$ pelo Método do Gradiente:

$$w_j^{[l]} = w_j^{[l]} - \alpha \frac{\partial}{\partial w_j^{[l]}} J(W, B)$$

$$b^{[l]} = b^{[l]} - \alpha \frac{\partial}{\partial b^{[l]}} J(W, B)$$

Observação:

As derivadas acima podem ser calculadas usando uma metodologia conhecida como **propagação para trás** (*back propagation*)

No Tensorflow, aplicamos um método alternativo ao método do gradiente padrão usando o comando:

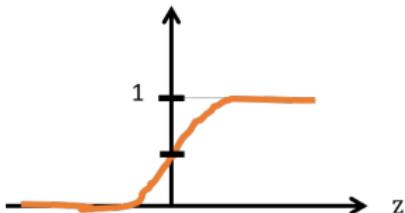
```
model.fit(X, y, epochs=100)
```

Até agora, usamos a função **sigmoide** como função de ativação para todos os neurônios da nossa rede.

Pergunta:

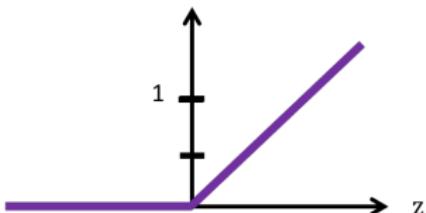
Seria possível utilizar outras funções de ativação? Se sim, como eu faço para escolher a função de ativação mais adequada para cada camada da minha rede?

Conhecendo a função **Unidade Linear Retificada** (ReLU)



Função sigmoide:

$$g(z) = \frac{1}{1+e^{-z}}$$

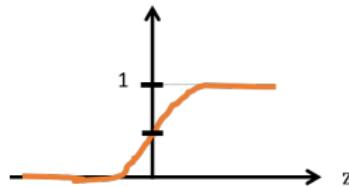


Função ReLU:

$$g(z) = \max(0, z)$$

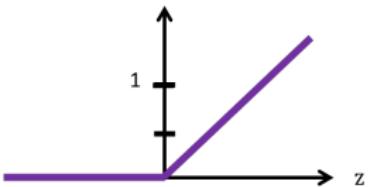
- Assim como a função sigmoide, a função **ReLU** é vastamente utilizada em redes neurais.
- Note que a saída não fica limitada ao intervalo (0, 1). Ao invés disso, ela pode assumir qualquer valor real ≥ 0 .

Conhecendo a função de Ativação Linear



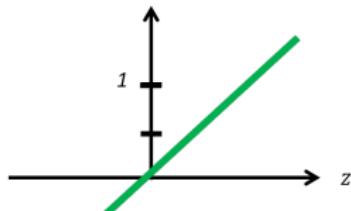
Função sigmoide:

$$g(z) = \frac{1}{1+e^{-z}}$$



Função ReLU:

$$g(z) = \max(0, z)$$



Função Linear:

$$g(z) = z$$

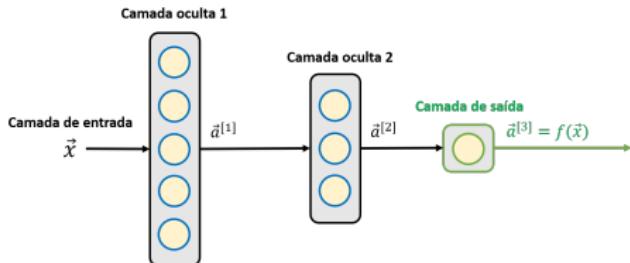
- Como a função de ativação linear é dada por

$$g(z) = z = \vec{w} \cdot \vec{x} + b$$

alguns usuários falarão que usar a essa função específica equivale a não utilizar nenhuma função de ativação.

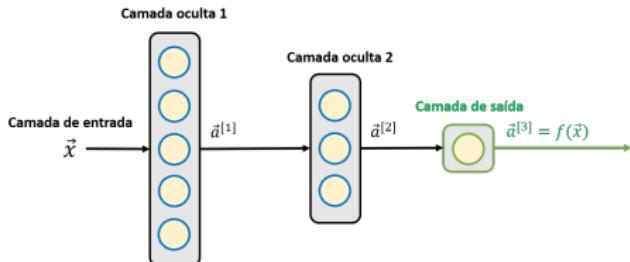
- Essas 3 funções de ativação, e também a função de ativação conhecida como tangente hiperbólica, são as mais utilizadas no contexto de redes neurais

Qual função de ativação devo escolher?



Pensando primeiro na **camada de saída**...

Qual função de ativação devo escolher?

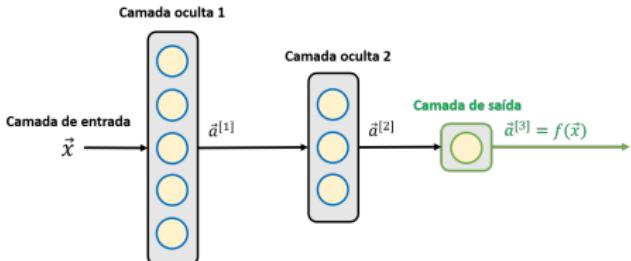


Pensando primeiro na **camada de saída**...

Problemas de classificação

- Se for um problema onde a variável alvo y é 0 ou 1 (problema de classificação binária), então quase sempre a função **sigmoide** será a escolha mais adequada.
- Assim, a saída da nossa rede $f(\vec{x})$ estará no intervalo $(0,1)$ e pode ser interpretada como sendo a probabilidade de y ser 1.

Qual função de ativação devo escolher?



Pensando primeiro na **camada de saída**...

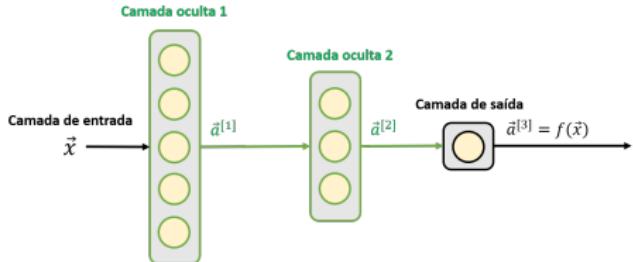
Problemas de classificação

- Se for um problema onde a variável alvo y é 0 ou 1 (problema de classificação binária), então quase sempre a função **sigmoide** será a escolha mais adequada.
- Assim, a saída da nossa rede $f(\vec{x})$ estará no intervalo $(0,1)$ e pode ser interpretada como sendo a probabilidade de y ser 1.

Problemas de regressão

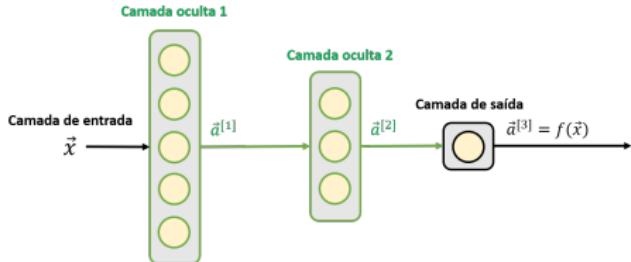
- Se for um problema de regressão onde y pode assumir qualquer valor real positivo ou negativo, então podemos usar função **linear**.
- Se for um problema de regressão onde y pode assumir apenas valores reais positivos (≥ 0), então podemos usar a função **ReLU**.
- Em ambos os casos, a saída da nossa rede $f(\vec{x})$ pode ser interpretada como sendo a nossa estimativa para o y .

Qual função de ativação devo escolher?



Pensando agora nas **camadas ocultas**...

Qual função de ativação devo escolher?



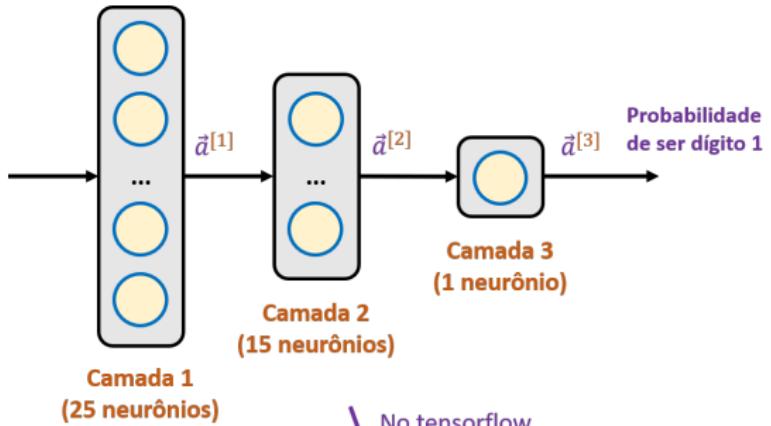
Pensando agora nas **camadas ocultas**...

- Hoje em dia, a função de ativação **ReLU** é a mais utilizada nas camadas ocultas.
- Isso porque ela pode ser calculada mais rapidamente (não requer o cálculo de uma exponencial).
- Com a sua definição $\max(0, z)$, a função ReLU possui a capacidade de desativar (zerar) características numa rede, acionando-as apenas quando se faz necessário!

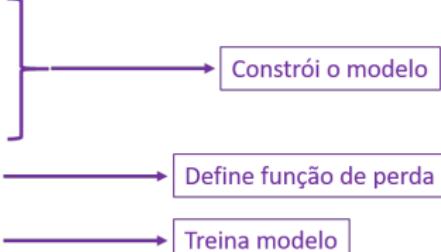
Observação final sobre funções de ativação

Há também outros tipos de funções de ativação. Fique à vontade para pesquisar outras possíveis funções na Internet...

Exemplo no Tensorflow: reconhecimento de dígitos 0 e 1



```
modelo = Sequential(  
    [  
        Dense(units=25, activation="relu"),  
        Dense(units=15, activation="relu"),  
        Dense(1, activation="sigmoid")  
    ]  
)  
modelo.compile(  
    loss=BinaryCrossentropy()  
)  
modelo.fit(  
    X,y,epochs=50  
)
```



Seja o sistema elétrico de potência abaixo, divido em 2 áreas (Área Interna e Área Externa)

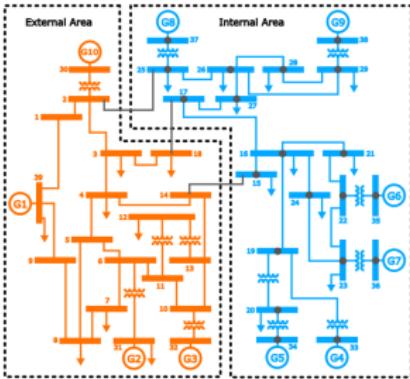


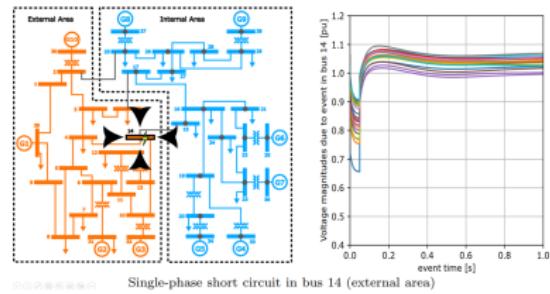
Fig. 1 39-bus New England IEEE benchmark system divided into internal area and external area.
Gray circles shown above the buses of the internal area indicate the installation of PMUs.

Supondo que tem uma falta (curto-circuito monofásico) ocorrendo nesse sistema, como localizar se esse evento está ocorrendo na ÁREA EXTERNA ($y = 0$) ou na ÁREA INTERNA ($y = 1$)?

De olho no código!

No nosso modelo, podemos usar como características \vec{x} os sinais elétricos adquiridos por PMUs instaladas em diferentes pontos do sistema.

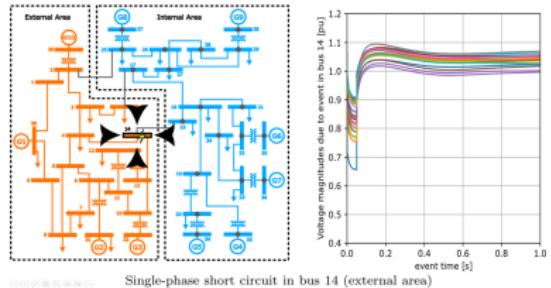
Veja abaixo um exemplo de medição feita por PMUs para um curto-circuito na Área Externa.



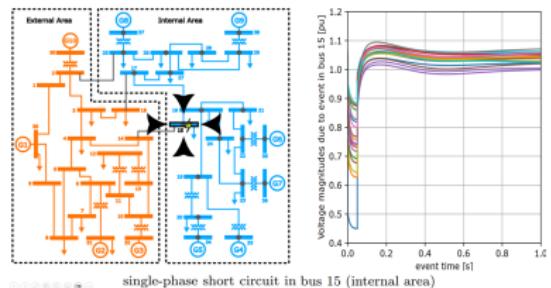
De olho no código!

No nosso modelo, podemos usar como características \vec{x} os sinais elétricos adquiridos por PMUs instaladas em diferentes pontos do sistema.

Veja abaixo um exemplo de medição feita por PMUs para um curto-circuito na Área Externa.



Por outro lado, veja abaixo um exemplo de medição feita por PMUs para um curto-circuito na Área Interna.



De olho no código!

Acesse o Python Notebook usando o QR code ou o link abaixo:

https://colab.research.google.com/github/xaximpv2/master/blob/main/codigo_aula17_localizacao_falta_SEP.ipynb



Clique nos links abaixo para baixar os dados necessários para rodar o código:

- https://ufprbr0-my.sharepoint.com/:x/g/personal/ricardo_schumacher_ufpr_br/EaouvKXaSBtDg5ezTEyMApEBHFQ01bGav75PyfdpLZ-BHw?e=eXD1KA
- https://ufprbr0-my.sharepoint.com/:x/g/personal/ricardo_schumacher_ufpr_br/ESGKr0rZIHJF1rftuM0l5EB0oLt8TTOUJk2DBaEWxuKWA?e=vkjrr7
- https://ufprbr0-my.sharepoint.com/:x/g/personal/ricardo_schumacher_ufpr_br/EZN1REORoRVhhAS3RCemp1YBpHR61m0255fsvaR5Wmt_rg?e=70tNtg
- https://ufprbr0-my.sharepoint.com/:x/g/personal/ricardo_schumacher_ufpr_br/Ed4U8TZVq9hGo2D7FHYYB-8B_7r-TCCFCjc3qAWhH-ykgQ?e=gcsu9l

OBS: Para adicionar os dados ao ambiente do Colab Notebook, no menu do canto esquerdo da tela do Colab clique em "Arquivos" e depois "Fazer upload para o armazenamento da sessão". Então carregue os arquivos baixados.

Parte 1

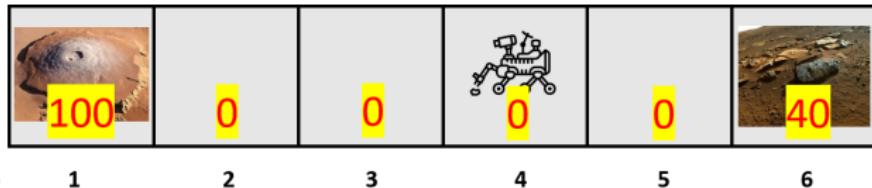
Rode todo o código. Responda às questões nele contidas e complete-o, se necessário.

- Por que resultados ligeiramente diferentes são obtidos cada vez que o código é rodado?

Parte 2

- 1 Testando diferentes valores para o número de camadas da rede e para o número de neurônios em cada camada, busque uma topologia de rede neural que resulta numa taxa de acerto para os dados de validação > 80%

Suponha que um veículo espacial em Marte está em busca de amostras específicas de solo, e que o mesmo possui 6 posições (estados) possíveis, conforme ilustrado abaixo.



A figura acima também mostra em vermelho a recompensa associada a cada estado. Os estados 1 e 6 representam amostras de solo de interesse científico.

OBS: Considere que os estados 1 e 6 são **terminais**. Ou seja, considere que, ao atingir qualquer um desses dois estados, independentemente da ação praticada, é decretado "fim de jogo". Nesse contexto, por exemplo, isso pode significar que o robô terá que repousar para recarregar sua bateria.

Questão 1

Partindo do estado inicial 4 e considerando um fator de desconto $\gamma = 0.9$, note que o **Retorno** obtido pelo robô caso o mesmo decida andar sempre para a esquerda (\leftarrow) será de 72.9 pontos. Escolha a alternativa que representa o cálculo correto deste valor.

- A) $\text{Retorno} = 1 \times 0 + 0.9 \times 0 + 0.9^2 \times 0 + 0.9^3 \times 100$
- B) $\text{Retorno} = 1 \times 0 + 0.9 \times 0 + 0.9^2 \times 40$

Atividade avaliativa

Questão 2

Partindo do estado inicial 4 e considerando um fator de desconto $\gamma = 0.9$, note que o **Retorno** obtido pelo robô caso o mesmo decida andar sempre para a esquerda (\leftarrow) será de 32.4 pontos. Escolha a alternativa que representa o cálculo correto deste valor.

- A) $\text{Retorno} = 1 \times 0 + 0.9 \times 0 + 0.9^2 \times 0 + 0.9^3 \times 100$
- B) $\text{Retorno} = 1 \times 0 + 0.9 \times 0 + 0.9^2 \times 40$

Questão 3

Comparando os valores de Retorno obtidos nas questões 1 e 2 (que considera $\gamma = 0.9$), qual é a melhor sequência de ações a ser praticada pelo robô?

- A) Ir apenas para \leftarrow
- B) Ir apenas para \rightarrow

Questão 4

Partindo do estado inicial 4 e considerando um fator de desconto $\gamma = 0.3$, note que o **Retorno** obtido pelo robô caso o mesmo decida andar sempre para a esquerda (\leftarrow) será de 2.7 pontos. Escolha a alternativa que representa o cálculo correto deste valor.

- A) $\text{Retorno} = 1 \times 0 + 0.3 \times 0 + 0.3^2 \times 0 + 0.3^3 \times 100$
- B) $\text{Retorno} = 1 \times 0 + 0.3 \times 0 + 0.3^2 \times 40$

Questão 5

Partindo do estado inicial 4 e considerando um fator de desconto $\gamma = 0.9$, note que o **Retorno** obtido pelo robô caso o mesmo decida andar sempre para a esquerda (\leftarrow) será de 3.6 pontos. Escolha a alternativa que representa o cálculo correto deste valor.

- A) Retorno = $1 \times 0 + 0.3 \times 0 + 0.3^2 \times 0 + 0.3^3 \times 100$
- B) Retorno = $1 \times 0 + 0.3 \times 0 + 0.3^2 \times 40$

Questão 6

Comparando os valores de Retorno obtidos nas questões 4 e 5 (que considera $\gamma = 0.3$), qual é a melhor sequência de ações a ser praticada pelo robô?

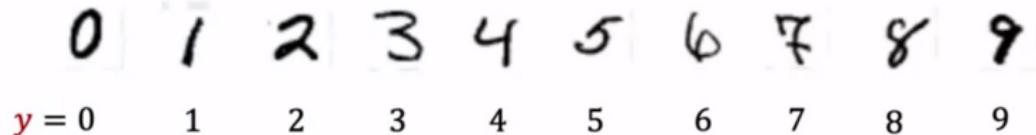
- A) Ir apenas para \leftarrow
- B) Ir apenas para \rightarrow

Problemas de Classificação com múltiplas classes



Nas aulas anteriores, aprendemos bastante sobre classificação binária. Agora falaremos sobre como resolver problemas de classificação onde temos múltiplas classes.

Classificação com múltiplas classes (multi-classe)



- Em problemas de classificação binária, y possui apenas dois valores possíveis: 0 ou 1.
- Em problemas de classificação com múltiplas classes, mais classes podem existir...

Localização de faltas em sistemas elétricos de potência

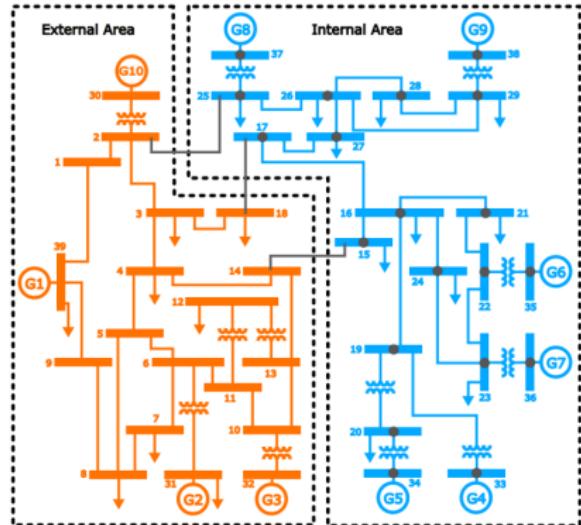
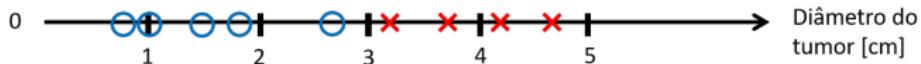


Fig. 1 39-bus New England IEEE benchmark system divided into internal area and external area.
Gray circles shown above the buses of the internal area indicate the installation of PMUs.

OBS: E se quisermos dividir o sistema em mais áreas? $y = 1, 2, 3, 4, 5, 6?$

- A Regressão **Softmax** é uma generalização do método de Regressão Logística, que se restringia a problemas de classificação binária.



Regressão Logística
(apenas duas saídas possíveis $y = 0, 1$)

Calculamos $z = \vec{w} \cdot \vec{x} + b$, e depois fazemos:

$$a_1 = g(z) = \frac{1}{1 + e^{-z}}$$

Onde a_1 pode ser interpretado como sendo
 $a_1 = P(y = 1|\vec{x})$ **X**

Como calculamos $P(y = 0|\vec{x})$? **O**

Simples,

$$a_2 = P(y = 0|\vec{x}) = 1 - a_1$$

Exemplo: Se $a_1 = 0,63$, quanto vale a_2 ?

Regressão Logística versus Softmax

Regressão Logística

(apenas duas saídas possíveis $y = 0, 1$)

Calculamos $z = \vec{w} \cdot \vec{x} + b$, e depois fazemos:

$$a_1 = g(z) = \frac{1}{1 + e^{-z}}$$

Onde a_1 pode ser interpretado como sendo

$$a_1 = P(y = 1 | \vec{x}) \text{ } \textcolor{red}{X}$$

Como calculamos $P(y = 0 | \vec{x})$? O

Simples,

$$a_2 = P(y = 0 | \vec{x}) = 1 - a_1$$

Exemplo: Se $a_1 = 0,63$, quanto vale a_2 ?

Regressão Softmax

(exemplo com $y = 1, 2, 3, 4$)

Calculamos

$$z_1 = \vec{w}_1 \cdot \vec{x} + b_1$$

$$z_2 = \vec{w}_2 \cdot \vec{x} + b_2$$

$$z_3 = \vec{w}_3 \cdot \vec{x} + b_3$$

$$z_4 = \vec{w}_4 \cdot \vec{x} + b_4$$

Então fazemos:

$$a_1 = P(y = 1 | \vec{x}) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} \text{ } \textcolor{red}{X}$$

$$a_2 = P(y = 2 | \vec{x}) = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} \text{ } \textcolor{blue}{O}$$

$$a_3 = P(y = 3 | \vec{x}) = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} \text{ } \textcolor{green}{\Delta}$$

$$a_4 = P(y = 4 | \vec{x}) = \frac{e^{z_4}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} \text{ } \textcolor{purple}{\square}$$

Exemplo: Se $a_1 = 0.2$, $a_2 = 0.35$, $a_3 = 0.15$, quanto vale a_4 ?

Generalização da Regressão Softmax

(exemplo com N classes, tal que $y = 1, 2, \dots, N$)

Calculamos

$$z_j = \vec{w}_j \cdot \vec{x} + b_j \quad \text{para } j = 1, \dots, N$$

Então fazemos:

$$a_j = P(y = j | \vec{x}) = \frac{e^{z_j}}{\sum_{k=1}^N e^{z_k}}$$

onde $a_1 + a_2 + \dots + a_N = 1$

Como fica a função custo para a Regressão Softmax?

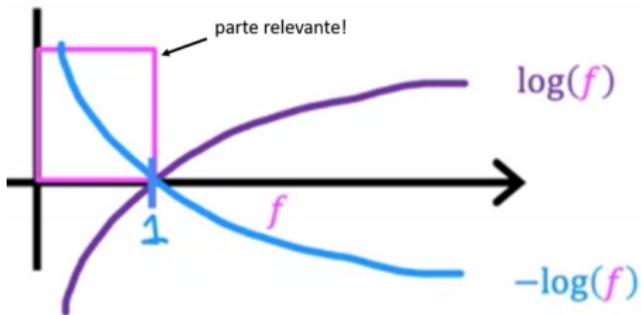
Lembrando que, na Regressão Logística, tínhamos

Duas classes ($N = 2$):

$$a_1 = P(y = 1 | \vec{x}) \quad \text{e} \quad a_2 = P(y = 0 | \vec{x}) = 1 - a_1$$

perda = função de entropia cruzada binária = $\begin{cases} -\log a_1 & , \text{ se } y^{(i)} = 1 \\ -\log a_2 & , \text{ se } y^{(i)} = 0 \end{cases}$

função custo = $J(\vec{w}, b) = \text{média das perdas}$



Como fica a função custo para a Regressão Softmax?

Transferindo essa ideia para a Regressão Softmax, temos

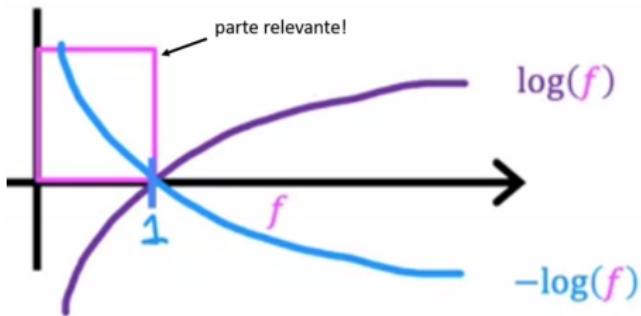
N classes:

$$a_1 = P(y = 1 | \vec{x}) \quad a_2 = P(y = 2 | \vec{x}) \quad \dots \quad a_N = P(y = N | \vec{x})$$

perda = função de entropia cruzada para N classes =

$$\begin{cases} -\log a_1 & , \text{ se } y^{(i)} = 1 \\ -\log a_2 & , \text{ se } y^{(i)} = 2 \\ \dots & \dots \\ -\log a_N & , \text{ se } y^{(i)} = N \end{cases}$$

função custo = $J(\vec{w}_1, b_1, \dots, \vec{w}_N, b_N) =$ média das perdas

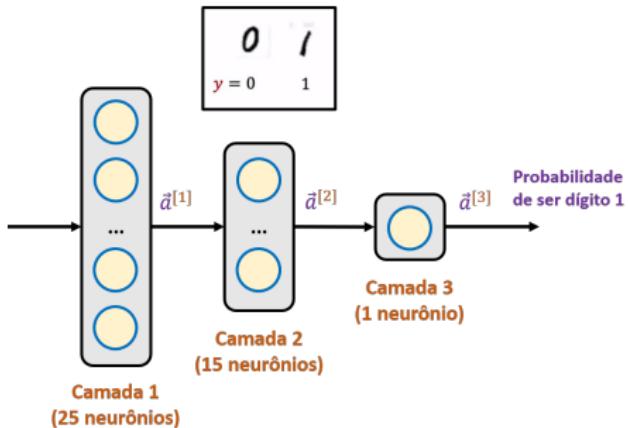


Redes Neurais com saída Softmax

Redes Neurais com saída Softmax

Para que seja possível usar redes neurais no contexto de classificação multi-classe, podemos inserir o modelo de regressão **Softmax na camada de saída** da rede.

Relembrando primeiro da rede com função de saída sigmoide



Pergunta:

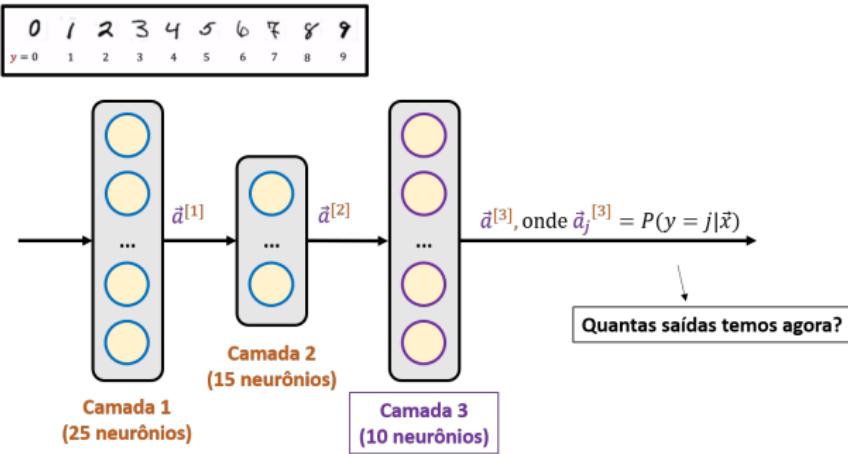
Como fazímos para calcular $\vec{a}^{[3]}$ quando a camada de saída tinha a função sigmoide?

Resposta:

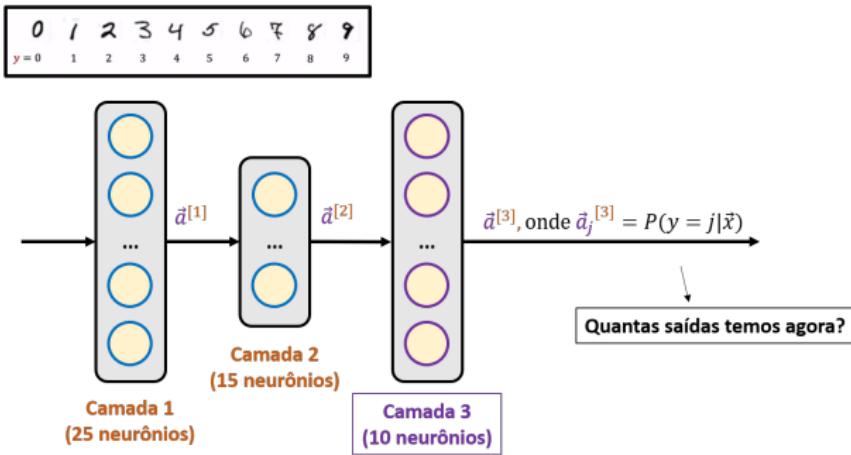
Fazíamos $z_1^{[3]} = \vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]}$ e depois

$$\vec{a}_1^{[3]} = g(z_1^{[3]}) = P(y = 1 | \vec{x})$$

Agora, no problema multi-classe, podemos usar a camada de saída Softmax



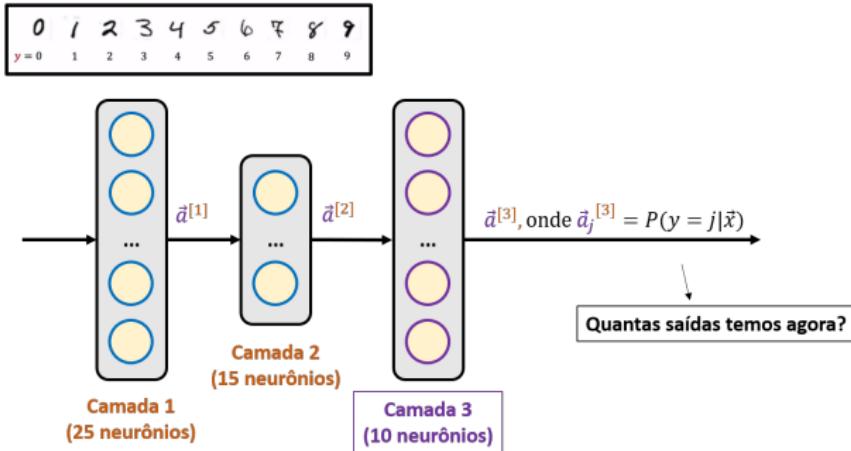
Agora, no problema multi-classe, podemos usar a camada de saída Softmax



Agora, com a função de ativação Softmax na camada de saída, calculamos $z_j^{[3]} = \vec{w}_j^{[3]} \cdot \vec{a}^{[2]} + b_j^{[3]}$, para $j = 1, \dots, 10$ e depois calculamos cada $\vec{d}_j^{[3]} = P(y = j | \vec{x})$ como sendo

$$\vec{d}_j^{[3]} = \frac{e^{z_j}}{e^{z_1} + e^{z_2} + \dots + e^{z_{10}}}$$

Agora, no problema multi-classe, podemos usar a camada de saída Softmax



Observação final (*nível hard*):

Note que $\vec{a}_j^{[3]}$ é função de z_1, z_2, \dots, z_N .

- Isso é uma característica interessante que diferencia a ativação Softmax das demais ativações vistas anteriormente (sigmoide, relu e linear), onde \vec{a}_j é função tão somente de z_j . Por exemplo, para a função sigmoide, teríamos:

$$\vec{a}_j^{[3]} = \frac{1}{1 + e^{-z_j}}$$

Implementação intuitiva:

```
modelo = Sequential(  
    [  
        Dense(units=25, activation="relu"),  
        Dense(units=15, activation="relu"),  
        Dense(10, activation="softmax")  
    ]  
)  
modelo.compile(  
    loss=SparseCategoricalCrossEntropy()  
)  
modelo.fit(  
    X,y,epochs=50  
)
```

10 unidades softmax na camada de saída

Função custo para múltiplas classes $y=0,1,2,3\dots$

De olho no código!

De olho no código!

Iremos agora verificar como implementamos a função Softmax na prática, assim como sua integração junto ao Tensorflow.

Acesse o Python Notebook usando o QR code ou o link abaixo:



https://colab.research.google.com/github/xaximppv2/master/blob/main/codigo_aula19_classificacao_multiclasse.ipynb

Parte 1

Rode todo o código. Certifique-se de que você o compreendeu.

Parte 2

- 1** Teste diferentes combinações de valores z_1, z_2, z_3, z_4 e verifique as probabilidades resultantes a_1, a_2, a_3, a_4 obtidas a partir da função Softmax.
- 2** Modifique o código para que existam 5 classes ao invés de 4. Obtenha a acurácia correspondente.

O método Adam

Esta aula constitui um tópico adicional da disciplina. Trata-se de um conteúdo opcional. Sua atividade não valerá nota e não precisa ser enviada.



- Até agora, estudamos Redes Neurais imaginando que o treinamento dos seus parâmetros é feito usando uma aplicação tradicional do Método do Gradiente, tal como, por exemplo,

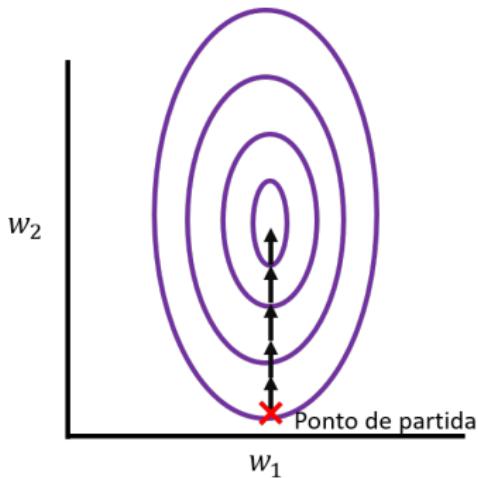
$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$$

onde α é um valor fixo > 0 (taxa de aprendizado).

- Entretanto, hoje em dia existem métodos de otimização avançada aplicada à redes neurais que vão ajustando α automaticamente durante o processo de treinamento. Nesse contexto insere-se o **Método Adam**

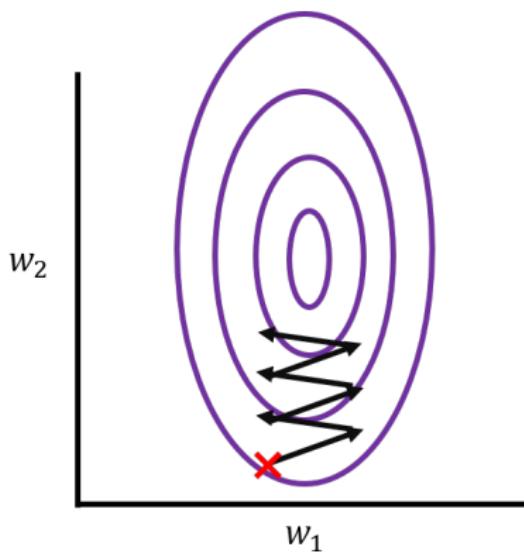
Método do Gradiente versus Método Adam

Primeiro caso: Utilizando um valor fixo e pequeno para α , o Método do Gradiente tenderá a dar vários pequenos passos em direção ao mínimo de $J(\vec{w}, b)$, conforme ilustrado abaixo:



- Para um exemplo como esse, o Método Adam perceberá que é possível **aumentar** o valor de α para agilizar o processo de chegada até o mínimo de $J(\vec{w}, b)$.

Segundo caso: utilizando um valor fixo e muito grande para α , o Método do Gradiente tenderá a dar passos maiores, porém poderá ocorrer o que está ilustrado abaixo:



- Nesse caso, o Método Adam perceberá que é melhor **reduzir** o valor de α para suavizar o processo de chegada até o mínimo de $J(\vec{w}, b)$.

Características do Método Adam

- A sigla 'Adam' abrevia a expressão **Adaptive Moment estimation**
- Ele utiliza uma taxa de aprendizado α diferente para cada parâmetro que está sendo treinado, conforme exemplo abaixo:

$$w_1 = w_1 - \alpha_1 \frac{\partial}{\partial w_1} J(\vec{w}, b)$$

...

$$w_{12} = w_{12} - \alpha_{12} \frac{\partial}{\partial w_{12}} J(\vec{w}, b)$$

Características do Método Adam

- A sigla 'Adam' abrevia a expressão Adaptive Moment estimation
- Ele utiliza uma taxa de aprendizado α diferente para cada parâmetro que está sendo treinado, conforme exemplo abaixo:

$$w_1 = w_1 - \alpha_1 \frac{\partial}{\partial w_1} J(\vec{w}, b)$$

...

$$w_{12} = w_{12} - \alpha_{12} \frac{\partial}{\partial w_{12}} J(\vec{w}, b)$$

Ideia principal:

- Se w_j (ou b_j) está se movendo de forma constante, com pouca variação em termos de direção, aumentar α_j (Primeiro caso que vimos)
- Se w_j (ou b_j) está oscilando muito em termos de direção, diminuir α_j (Segundo caso que vimos)

Características do Método Adam

- A sigla 'Adam' abrevia a expressão **Adaptive Moment estimation**
- Ele utiliza uma taxa de aprendizado α diferente para cada parâmetro que está sendo treinado, conforme exemplo abaixo:

$$w_1 = w_1 - \alpha_1 \frac{\partial}{\partial w_1} J(\vec{w}, b)$$

...

$$w_{12} = w_{12} - \alpha_{12} \frac{\partial}{\partial w_{12}} J(\vec{w}, b)$$

Ideia principal:

- Se w_j (ou b_j) está se movendo de forma constante, com pouca variação em termos de direção, aumentar α_j (Primeiro caso que vimos)
- Se w_j (ou b_j) está oscilando muito em termos de direção, diminuir α_j (Segundo caso que vimos)

Observação:

- Nós já estamos utilizando o Método Adam nos nossos códigos, entregando a ele um palpite inicial acerca da taxa de aprendizado α .
- Ainda vale a pena **mantermos a estratégia** de testar diferentes valores para α para verificar se a convergência torna-se mais rápida ou não, apesar do método já ter uma estratégia automatizada para seleção desse parâmetro ao longo das iterações.

De olho no código!

De olho no código!

Iremos agora verificar como usar o Tensorflow para treinar um modelo de reconhecimento de dígitos de 0 a 9 escritos à mão.

Acesse o Python Notebook usando o QR code ou o link abaixo:

https://colab.research.google.com/github/xaximpv2/master/blob/main/codigo_aula19_topico_adicional.ipynb



Acesse os dados necessários para rodar o código usando os links abaixo:

https://ufprbr0-my.sharepoint.com/:f/g/personal/ricardo_schumacher_ufpr_br/E15c4hFwzLFNunFo17IRxuIB8_7_D29BISiKHsc8JY2ANQ?e=QCzPS4

OBS: Para adicionar os dados ao ambiente do Colab Notebook, no menu do canto esquerdo da tela do Colab clique em "Arquivos" e depois "Fazer upload para o armazenamento da sessão". Então carregue os arquivos baixados.

Parte 1

Rode todo o código. Certifique-se que você o compreendeu.

Parte 2

- 1 Verifique se o aumento de número de épocas aumenta a taxa de acerto do modelo.

Como implementar/testar métodos e modelos



Na nossa disciplina, já aprendemos sobre os seguintes métodos:

- Regressão Linear
- Regressão Logística
- Redes Neurais (aprendizado profundo)

Levando em conta uma perspectiva mais geral, falaremos nessa aula sobre **dicas de como implementar/testar de forma eficiente algoritmos de aprendizado de máquina.**

Exemplo inicial

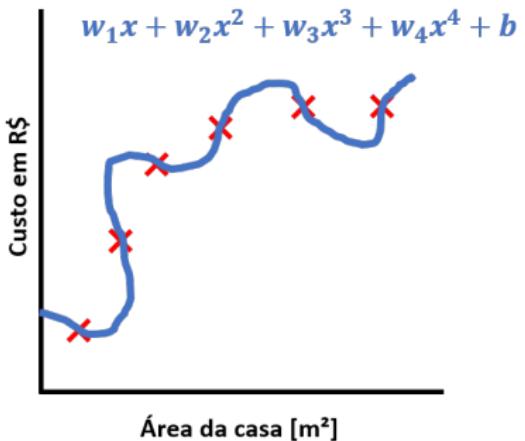
Suponha que você está implementando uma Regressão Linear com regularização para o problema de estimação do módulo da tensão num determinado ponto de um sistema elétrico:

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m \left(f_{\vec{w}, b} (\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

Você percebeu que as previsões feitas pelo modelo atual são inaceitáveis. **O que fazer?**
Podemos tentar o seguinte:

- Conseguir mais dados
- Excluir características (testar modelo mais simples)
- Incluir características (testar modelo mais complexo)
- Adicionar características polinomiais ao modelo ($x_1^2, x_2^2, x_1 x_2, \dots$)
- Aumentar λ
- Reduzir λ

Pergunta: Qual dessas alternativas devemos testar primeiro?



Perguntas

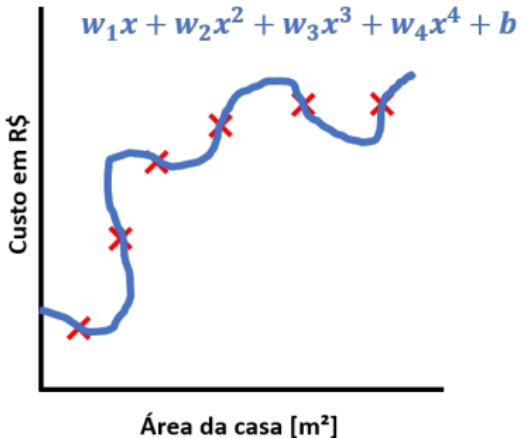
- Qual é o valor da função custo associado a esse modelo?
- Trata-se de um bom modelo para esses dados?

Observação: Quando o modelo leva em conta mais do que uma característica, é mais difícil fazermos essa análise graficamente. Como proceder então?

Podemos **testar quantitativamente** a qualidade do nosso modelo fazendo o seguinte:

- 1 Alimente seu modelo com um conjunto de dados que ele não "viu" durante seu treinamento
- 2 Utilize alguma métrica quantitativa para verificar sua performance.

Como você acha que o modelo abaixo se sairia nesse teste?



Avaliando a qualidade de um modelo: Regra 70/30

Área da casa [m ²]	Custo em R\$
32	51.000
149	265.000
78	110.000
36	49.000
132	280.000
60	120.000
59	99.000
129	279.000
62	124.000
220	315.000

Dados de treinamento
Use cerca de 70% dos dados para treinar os parâmetros do seu modelo

Dados de teste
Use cerca de 30% dos dados para validar o modelo treinado

"Um bom modelo é capaz de performar bem mesmo quando aplicado sobre o conjunto de dados de teste (dados que ele não teve acesso durante seu treinamento)."

Notação para dados de treinamento:

$$\begin{aligned} & \left(x^{(1)}, y^{(1)} \right) \\ & \dots \\ & \left(x^{(m_{trein})}, y^{(m_{trein})} \right) \end{aligned}$$

→ conjunto de dados de treinamento contendo m_{trein} amostras.

Notação para dados de teste:

$$\begin{aligned} & \left(x_{teste}^{(1)}, y_{teste}^{(1)} \right) \\ & \dots \\ & \left(x_{teste}^{(m_{teste})}, y_{teste}^{(m_{teste})} \right) \end{aligned}$$

→ conjunto de dados de treinamento contendo m_{teste} amostras.

Pergunta: O que acontecerá se você decidir usar uma regra do tipo 80/20?

Pergunta:

Como avaliar a qualidade de um modelo de regressão que foi treinado a partir da minimização da função custo

$$J(\vec{w}, b) = \frac{1}{2m_{trein}} \sum_{i=1}^{m_{trein}} \left(f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m_{trein}} \sum_{j=1}^n w_j^2 \quad ?$$

Exemplo

Pergunta:

Como avaliar a qualidade de um modelo de regressão que foi treinado a partir da minimização da função custo

$$J(\vec{w}, b) = \frac{1}{2m_{trein}} \sum_{i=1}^{m_{trein}} \left(f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m_{trein}} \sum_{j=1}^n w_j^2 \quad ?$$

Resposta:

Verifique como o modelo $f_{\vec{w}, b}(\vec{x})$ se sai ao ser aplicado ao conjunto de dados de teste:

$$J_{teste}(\vec{w}, b) = \frac{1}{2m_{teste}} \sum_{i=1}^{m_{teste}} \left(f_{\vec{w}, b}(\vec{x}_{teste}^{(i)}) - y_{teste}^{(i)} \right)^2$$

Exemplo

Pergunta:

Como avaliar a qualidade de um modelo de regressão que foi treinado a partir da minimização da função custo

$$J(\vec{w}, b) = \frac{1}{2m_{trein}} \sum_{i=1}^{m_{trein}} \left(f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m_{trein}} \sum_{j=1}^n w_j^2 \quad ?$$

Resposta:

Verifique como o modelo $f_{\vec{w}, b}(\vec{x})$ se sai ao ser aplicado ao conjunto de dados de teste:

$$J_{teste}(\vec{w}, b) = \frac{1}{2m_{teste}} \sum_{i=1}^{m_{teste}} \left(f_{\vec{w}, b}(\vec{x}_{teste}^{(i)}) - y_{teste}^{(i)} \right)^2$$

Pergunta:

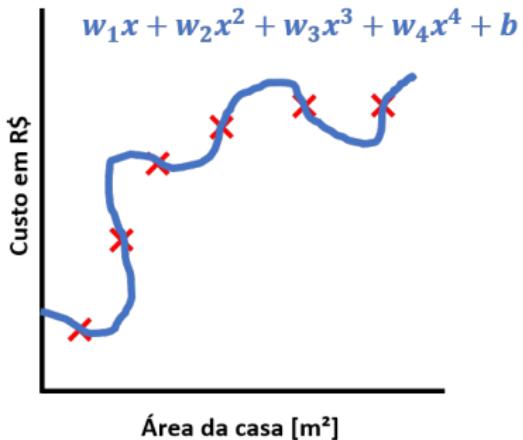
Seja

$$J_{trein}(\vec{w}, b) = \frac{1}{2m_{trein}} \sum_{i=1}^{m_{trein}} \left(f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)} \right)^2$$

Quem naturalmente tenderá a ser maior, $J_{teste}(\vec{w}, b)$ ou $J_{trein}(\vec{w}, b)$?

Exemplo

Suponha que no gráfico abaixo encontram-se representados apenas $(x^{(i)}, y^{(i)}) \rightarrow$ dados de treinamento.



Pergunta:

Teremos

- A) $J_{teste}(\vec{w}, b) >> J_{trein}(\vec{w}, b)$?
- B) $J_{teste}(\vec{w}, b) << J_{trein}(\vec{w}, b)$?

OBS: Trata-se de um indicativo de **overfitting**.

Exemplo (problema de classificação)

Pergunta:

Como avaliar a qualidade de um modelo de classificação que foi treinado a partir da minimização da função

$$J(\vec{w}, b) = -\frac{1}{m_{trein}} \sum_{i=1}^{m_{trein}} \left[y^{(i)} \log \left(f_{\vec{w}, b} (\vec{x}^{(i)}) \right) + (1 - y^{(i)}) \log \left(1 - f_{\vec{w}, b} (\vec{x}^{(i)}) \right) \right] \\ + \frac{\lambda}{2m_{trein}} \sum_{j=1}^n w_j^2$$

Resposta:

Verifique como o modelo $f_{\vec{w}, b} (\vec{x})$ se sai ao ser aplicado ao conjunto de dados de teste:

$$J_{teste}(\vec{w}, b) = -\frac{1}{m_{teste}} \sum_{i=1}^{m_{teste}} \left[y_{teste}^{(i)} \log \left(f_{\vec{w}, b} (\vec{x}_{teste}^{(i)}) \right) + (1 - y_{teste}^{(i)}) \log \left(1 - f_{\vec{w}, b} (\vec{x}_{teste}^{(i)}) \right) \right]$$

Seja

$$J_{trein}(\vec{w}, b) = -\frac{1}{m_{trein}} \sum_{i=1}^{m_{trein}} \left[y^{(i)} \log \left(f_{\vec{w}, b} (\vec{x}^{(i)}) \right) + (1 - y^{(i)}) \log \left(1 - f_{\vec{w}, b} (\vec{x}^{(i)}) \right) \right]$$

Quem naturalmente tenderá a ser maior, $J_{teste}(\vec{w}, b)$ ou $J_{trein}(\vec{w}, b)$?

Exemplo (problema de classificação)

Alternativamente, em problemas de classificação, também é muito comum usarmos a **taxa de acerto** (ou a **taxa de erro**) para avaliarmos a qualidade de um modelo.

Considerando a utilização da taxa de erro, podemos definir:

$$J_{teste}(\vec{w}, b) = \frac{\text{quantidade de amostras em que } \hat{y} \neq y_{teste}}{m_{teste}} \times 100\%$$

$$J_{trein}(\vec{w}, b) = \frac{\text{quantidade de amostras em que } \hat{y} \neq y_{trein}}{m_{trein}} \times 100\%$$

Utilizando a taxa de erro, quanto maior $J_{teste}(\vec{w}, b)$, pior o modelo. Caso tivéssemos utilizado a taxa de acerto como métrica, seria o contrário.

Suponha que você calculou $J_{teste}(\vec{w}, b)$ para 10 modelos diferentes:

$$1) \quad f_{\vec{w}, b}(\vec{x}) = w_1 x + b \quad \rightarrow \quad J_{teste}^{<1>}(\vec{w}, b)$$

$$2) \quad f_{\vec{w}, b}(\vec{x}) = w_1 x + w_2 x^2 + b \quad \rightarrow \quad J_{teste}^{<2>}(\vec{w}, b)$$

$$3) \quad f_{\vec{w}, b}(\vec{x}) = w_1 x + w_2 x^2 + w_3 x^3 + b \quad \rightarrow \quad J_{teste}^{<3>}(\vec{w}, b)$$

⋮

$$10) \quad f_{\vec{w}, b}(\vec{x}) = w_1 x + w_2 x^2 + w_3 x^3 + \cdots + w_{10} x^{10} + b \quad \rightarrow \quad J_{teste}^{<10>}(\vec{w}, b)$$

Suponha que $J_{teste}^{<5>}(\vec{w}, b)$ seja o menor erro dentre todos $J_{teste}^{<d>}(\vec{w}, b)$, $d = 1, \dots, 10$.

Pergunta:

É razoável eleger o quinto modelo como o melhor modelo?

Resposta:

Sim, porém... (continua no próximo slide)

- A partir do momento que você utilizou o conjunto de dados de teste para refinar algum hiperparâmetro do seu projeto (ordem do polinômio, número de camadas de uma rede neural, taxa de aprendizado, etc), esse conjunto de dados não pode mais ser chamado de conjunto de dados teste. É melhor chamar esse conjunto de dados de:

Conjunto de dados de validação (val) = dados de desenvolvimento = dados de validação cruzada (cv)

- A partir do momento que você utilizou o conjunto de dados de teste para refinar algum hiperparâmetro do seu projeto (ordem do polinômio, número de camadas de uma rede neural, taxa de aprendizado, etc), esse conjunto de dados não pode mais ser chamado de conjunto de dados teste. É melhor chamar esse conjunto de dados de:

Conjunto de dados de validação (val) = dados de desenvolvimento = dados de validação cruzada (cv)

- Ou seja, $J_{teste}^{<5>}(\vec{w}, b)$ deixa de ser uma estimativa justa para o erro de generalização verdadeiro, já que pelo menos 1 hiperparâmetro está sendo selecionado com base nesse valor. Por isso, $J_{teste}^{<5>}(\vec{w}, b)$ nesse caso deve ser chamado de $J_{val}^{<5>}(\vec{w}, b)$, sendo que $J_{val}^{<5>}(\vec{w}, b)$ tenderá a ser menor que o erro de generalização verdadeiro.

- A partir do momento que você utilizou o conjunto de dados de teste para refinar algum hiperparâmetro do seu projeto (ordem do polinômio, número de camadas de uma rede neural, taxa de aprendizado, etc), esse conjunto de dados não pode mais ser chamado de conjunto de dados teste. É melhor chamar esse conjunto de dados de:

Conjunto de dados de validação (val) = dados de desenvolvimento = dados de validação cruzada (cv)

- Ou seja, $J_{teste}^{<5>}(\vec{w}, b)$ deixa de ser uma estimativa justa para o erro de generalização verdadeiro, já que pelo menos 1 hiperparâmetro está sendo selecionado com base nesse valor. Por isso, $J_{teste}^{<5>}(\vec{w}, b)$ nesse caso deve ser chamado de $J_{val}^{<5>}(\vec{w}, b)$, sendo que $J_{val}^{<5>}(\vec{w}, b)$ tenderá a ser menor que o erro de generalização verdadeiro.
- Isso nos leva à ideia de dividirmos o conjunto inicial de dados em 3 partes, ao invés de apenas duas (continua no próximo slide...)

Área da casa [m ²]	Custo em R\$	
32	51.000	
149	265.000	
78	110.000	
36	49.000	
132	280.000	
60	120.000	
59	99.000	
129	279.000	
62	124.000	
220	315.000	

Dados de treinamento
Use cerca de 60% dos dados para treinar os parâmetros do seu modelo (w e b)

Dados de validação
Use cerca de 20% dos dados para selecionar hiperparâmetros (número de neurônios em cada camada, número de camadas, parâm. de regularização λ , α etc)

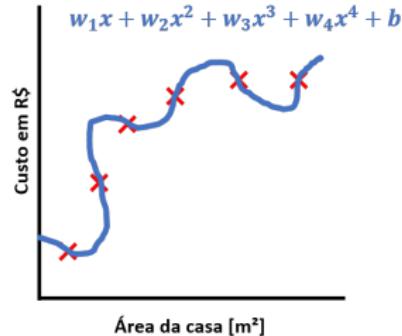
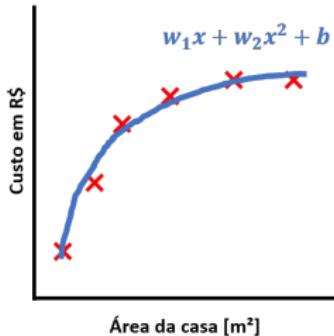
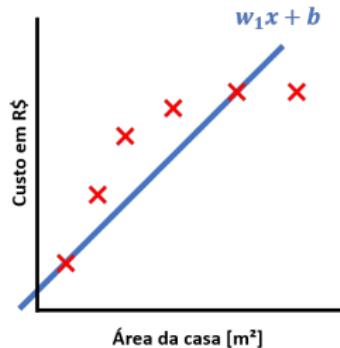
Dados de teste
Use cerca de 20% dos dados para testar seu único modelo final. Assim, você obterá uma estimativa justa para o erro de generalização

Observação final:

O conjunto de dados teste serve tão-somente para fornecer uma estatística justa para o erro de generalização verdadeiro (ou taxa de acerto verdadeira, caso seja o caso). Ele **nunca** deve ser tomado como base para selecionar qualquer tipo de parâmetro ou hiperparâmetro. Caso ele venha ser usado para este fim, ele deixa de ser o conjunto de dados de teste.

Exemplo

Suponha que nos gráficos abaixo encontram-se representados apenas $(x^{(i)}, y^{(i)}) \rightarrow$ dados de treinamento.



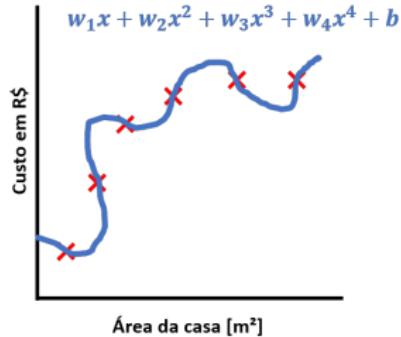
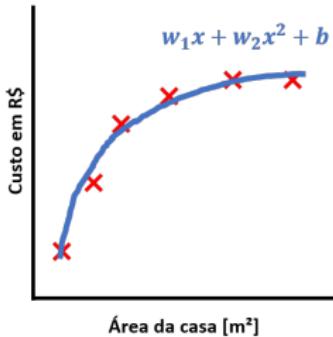
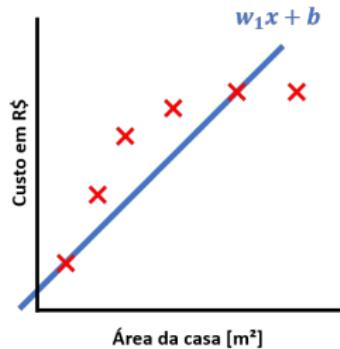
Pergunta

Olhando apenas para o gráfico da esquerda, escolha uma das opções abaixo.

- 1 J_{trein} é pequeno e J_{cv} será pequeno.
- 2 J_{trein} é pequeno e J_{cv} será grande.
- 3 J_{trein} é grande e J_{cv} será pequeno.
- 4 J_{trein} é grande e J_{cv} será grande.

Exemplo

Suponha que nos gráficos abaixo encontram-se representados apenas $(x^{(i)}, y^{(i)}) \rightarrow$ dados de treinamento.



Pergunta

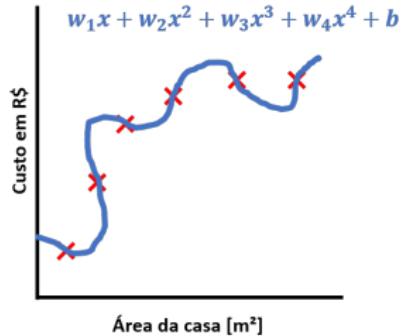
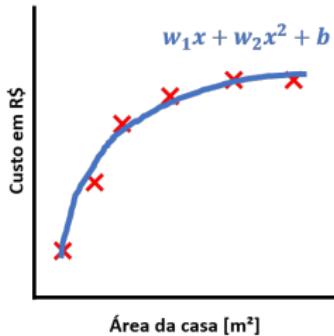
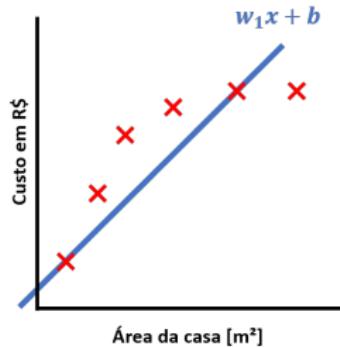
Olhando apenas para o gráfico da esquerda, escolha uma das opções abaixo.

- 1 J_{trein} é pequeno e J_{cv} será pequeno.
- 2 J_{trein} é pequeno e J_{cv} será grande.
- 3 J_{trein} é grande e J_{cv} será pequeno.
- 4 J_{trein} é grande e J_{cv} será grande.

Conclusão importante: Um modelo com alto viés (high bias), não apresenta uma boa performance nem mesmo para os dados de estimação (o modelo subestima até mesmo esses dados)...

Exemplo

Suponha que nos gráficos abaixo encontram-se representados apenas $(x^{(i)}, y^{(i)}) \rightarrow$ dados de treinamento.



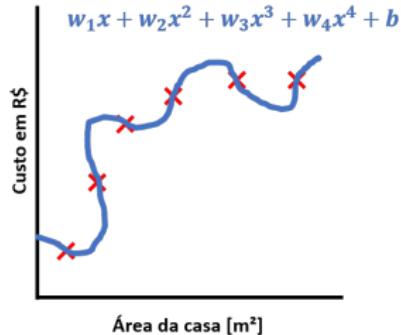
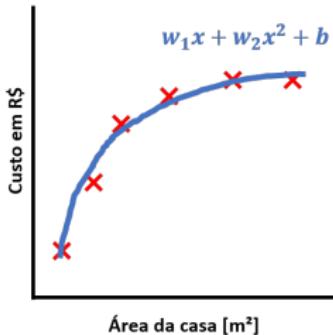
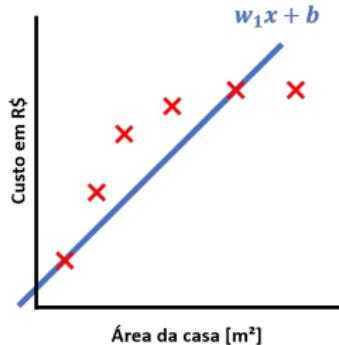
Pergunta

Olhando apenas para o gráfico da direita, escolha uma das opções abaixo.

- 1 J_{trein} é pequeno e J_{cv} será pequeno.
- 2 J_{trein} é pequeno e J_{cv} será grande.
- 3 J_{trein} é grande e J_{cv} será pequeno.
- 4 J_{trein} é grande e J_{cv} será grande.

Exemplo

Suponha que nos gráficos abaixo encontram-se representados apenas $(x^{(i)}, y^{(i)}) \rightarrow$ dados de treinamento.



Pergunta

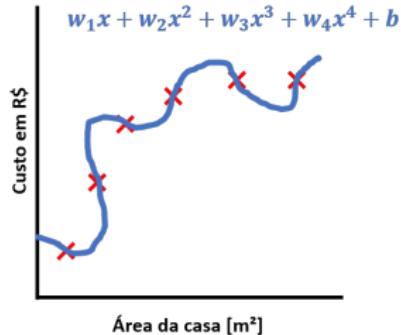
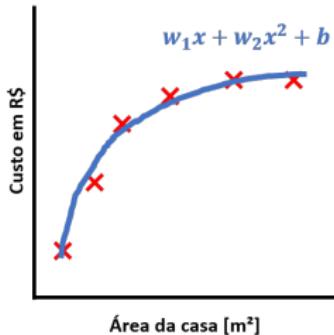
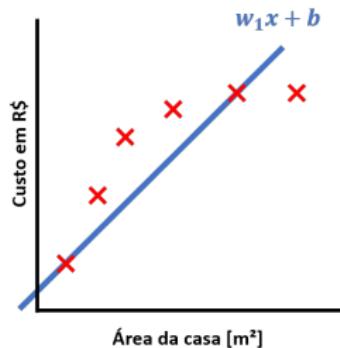
Olhando apenas para o gráfico da direita, escolha uma das opções abaixo.

- 1 J_{trein} é pequeno e J_{cv} será pequeno.
- 2 J_{trein} é pequeno e J_{cv} será grande.
- 3 J_{trein} é grande e J_{cv} será pequeno.
- 4 J_{trein} é grande e J_{cv} será grande.

Conclusão importante: Um modelo com elevada variância (high variance), apresenta uma boa performance para dados que ele já viu, mas sua performance deteriora consideravelmente para dados não vistos anteriormente...

Exemplo

Suponha que nos gráficos abaixo encontram-se representados apenas $(x^{(i)}, y^{(i)}) \rightarrow$ dados de treinamento.



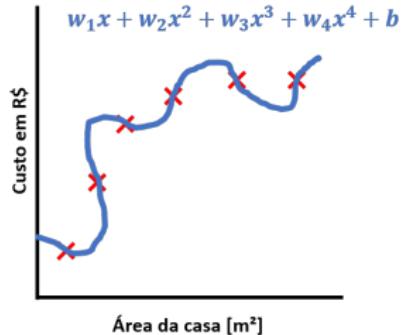
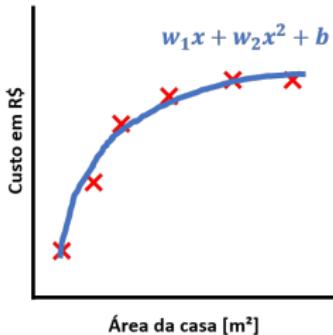
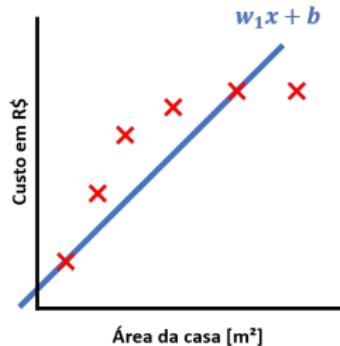
Pergunta

Olhando apenas para o gráfico do meio, escolha uma das opções abaixo.

- 1 J_{trein} é pequeno e J_{cv} será pequeno.
- 2 J_{trein} é pequeno e J_{cv} será grande.
- 3 J_{trein} é grande e J_{cv} será pequeno.
- 4 J_{trein} é grande e J_{cv} será grande.

Exemplo

Suponha que nos gráficos abaixo encontram-se representados apenas $(x^{(i)}, y^{(i)}) \rightarrow$ dados de treinamento.



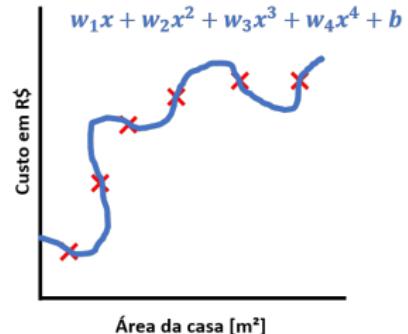
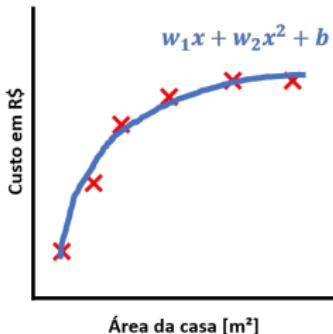
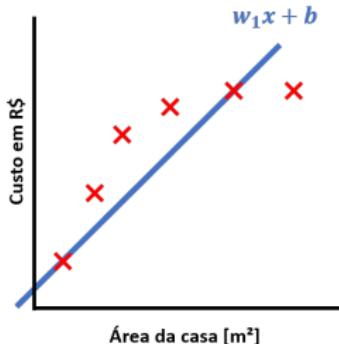
Pergunta

Olhando apenas para o gráfico do meio, escolha uma das opções abaixo.

- 1 J_{trein} é pequeno e J_{cv} será pequeno.
- 2 J_{trein} é pequeno e J_{cv} será grande.
- 3 J_{trein} é grande e J_{cv} será pequeno.
- 4 J_{trein} é grande e J_{cv} será grande.

Conclusão importante: Um modelo adequado apresenta uma boa performance tanto para os dados de estimação como também para dados não vistos anteriormente...

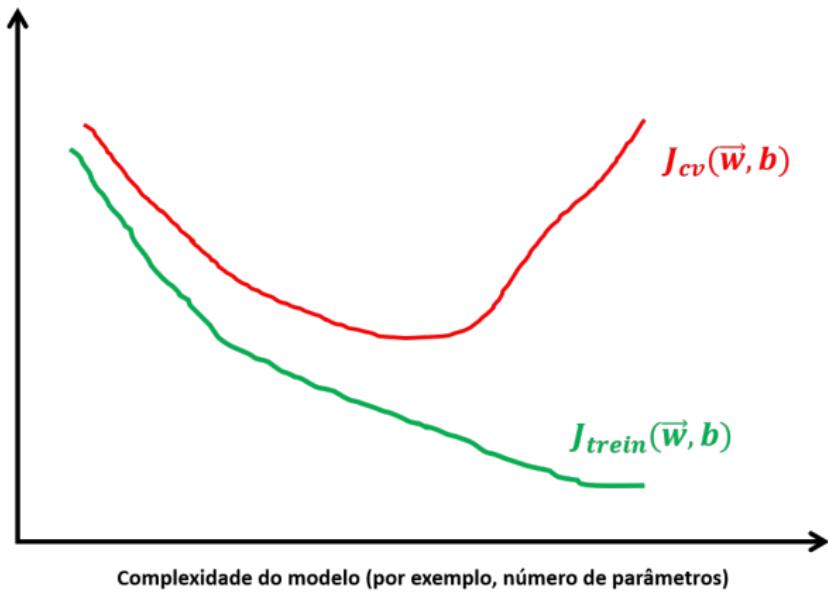
Exemplo



Pergunta

Analisando os três modelos acima, qual modelo possui maior ordem? Ou seja, é mais complexo?

Com base na resposta para essa pergunta e levando em conta o que já estudamos, podemos perceber o seguinte:
(continua no próximo slide)



Pergunta

Onde queremos estar nessa figura?

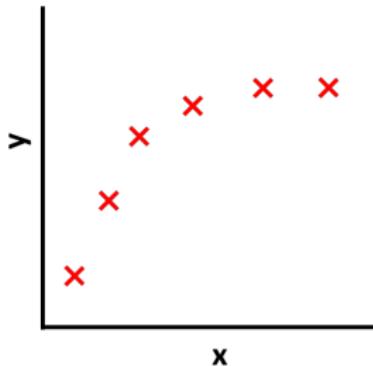
Regra prática:

Teste desde modelos mais simples até modelos altamente complexos. Selecione aquele que resulta no menor J_{cv} .

Como a regularização afeta o desempenho do seu modelo?

Suponha que você deseja ajustar um modelo do tipo $f_{\vec{w}, b}(\vec{x}^{(i)}) = w_1x + w_2x^2 + \dots + w_nx^n + b$ para os dados abaixo usando Regressão Linear com regularização:

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m \left(f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$



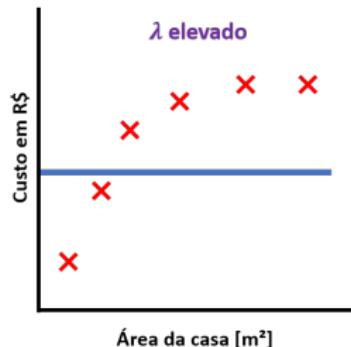
Pergunta:

Supondo, por exemplo, $n = 8$. Qual será o modelo resultante caso você escolha $\lambda = 1000000$?

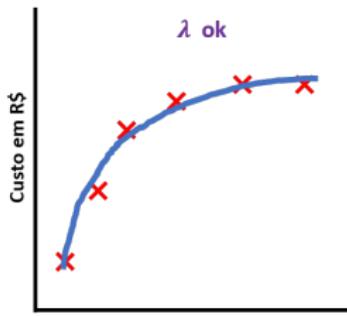
Como a regularização afeta o desempenho do seu modelo?

Suponha que nos gráficos abaixo encontram-se representados apenas $(x^{(i)}, y^{(i)}) \rightarrow$ dados de treinamento.

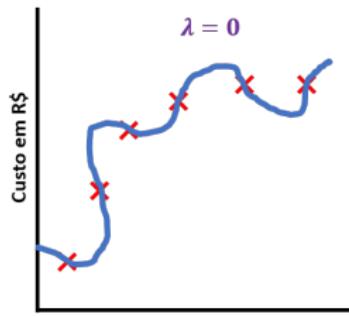
$$f = w_1x + w_2x^2 + \dots + w_8x^8 + b$$



$J_{trein}(\vec{w}, b)$ e $J_{cv}(\vec{w}, b)$ elevados
(alto viés)



$J_{trein}(\vec{w}, b)$ e $J_{cv}(\vec{w}, b)$
pequenos

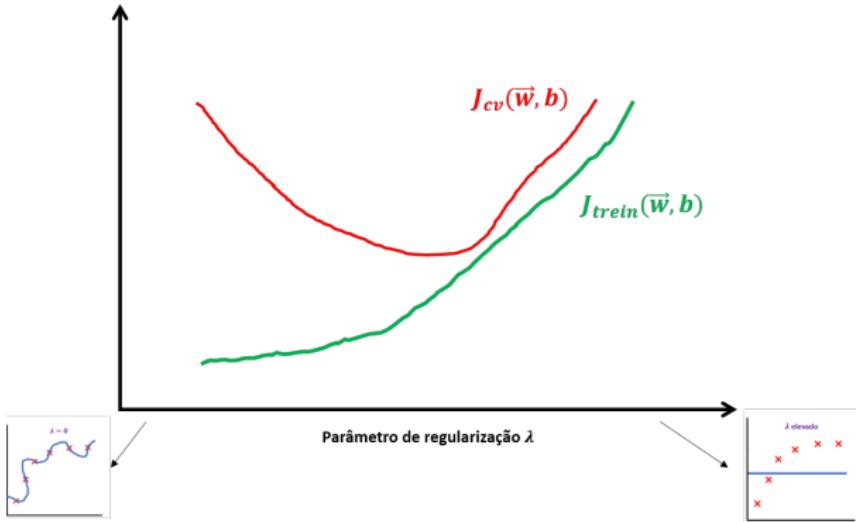


$J_{trein}(\vec{w}, b) \ll J_{cv}(\vec{w}, b)$ (alta
variância)

Pergunta

Analizando os três modelos acima, quando $\lambda = 0$ teremos um modelo mais simples ou mais complexo?

Como a regularização afeta o desempenho do seu modelo?



Pergunta

Onde queremos estar nessa figura?

Regra prática:

Teste diferentes valores para λ (0; 0.01; 0.02; 0.04; 0.08; ...; 10). Selecione aquele que resulta no menor J_{cv} .

Suponha as seguintes taxas de erro:

- $J_{train} = 10.8\%$
- $J_{cv} = 14.8\%$
- Performance humana média = 10.6 %

Suponha as seguintes taxas de erro:

- $J_{train} = 10.8\%$
- $J_{cv} = 14.8\%$
- Performance humana média = 10.6 %

Pergunta:

- O modelo encontra-se subestimando os dados de treinamento? Em outras palavras, temos um problema de alto viés?

Resposta:

- Não, afinal, para os dados de treinamento o modelo alcança praticamente um nível humano de performance.

Suponha as seguintes taxas de erro:

- $J_{train} = 10.8\%$
- $J_{cv} = 14.8\%$
- Performance humana média = 10.6 %

Pergunta:

- O modelo encontra-se subestimando os dados de treinamento? Em outras palavras, temos um problema de alto viés?

Resposta:

- Não, afinal, para os dados de treinamento o modelo alcança praticamente um nível humano de performance.

Pergunta:

- O modelo encontra-se sobreestimando os dados de treinamento? Em outras palavras, temos um problema de alta variância?

Resposta:

- Podemos dizer que sim, afinal J_{cv} é significativamente maior que J_{train} .

No exemplo do slide anterior, nós usamos a **performance humana** como **performance de referência**. Entretanto, dependendo da aplicação, existem outras maneiras para se estabelecer tal referência, quais sejam:

- Comparar a performance do seu algoritmo com outros existentes na literatura
- Usar da sua experiência no assunto para definir um valor razoável de referência

Suponha as seguintes taxas de erro:

- $J_{train} = 16.8 \%$
- $J_{cv} = 17.2 \%$
- Performance de referência = 10.1 %

Suponha as seguintes taxas de erro:

- $J_{train} = 16.8\%$
- $J_{cv} = 17.2\%$
- Performance de referência = 10.1 %

Pergunta:

- O modelo encontra-se sobreestimando os dados de treinamento? Em outras palavras, temos um problema de alta variância?

Resposta:

- Não, afinal J_{cv} é próximo de J_{train} .

Um outro exemplo numérico

Suponha as seguintes taxas de erro:

- $J_{train} = 16.8\%$
- $J_{cv} = 17.2\%$
- Performance de referência = 10.1 %

Pergunta:

- O modelo encontra-se sobreestimando os dados de treinamento? Em outras palavras, temos um problema de alta variância?

Resposta:

- Não, afinal J_{cv} é próximo de J_{train} .

Pergunta:

- O modelo encontra-se subestimando os dados de treinamento? Em outras palavras, temos um problema de alto viés?

Resposta:

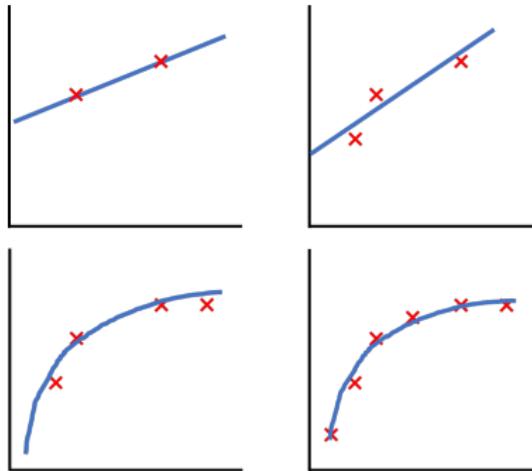
- Podemos dizer que sim, afinal, mesmo para os dados de treinamento o modelo não alcança um valor próximo ao de referência.

Já vimos como a complexidade do modelo pode afetar sua performance. Também fizemos tal análise com base na regularização. Agora buscaremos responder à seguinte pergunta:

Como o número de amostras afeta o desempenho do seu modelo?

Como o número de amostras afeta o desempenho do seu modelo?

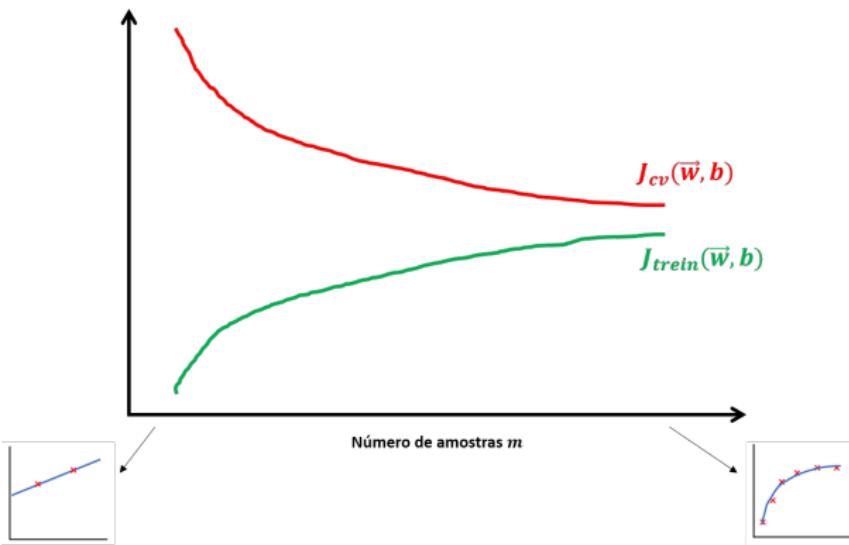
Suponha que você deseja ajustar um modelo do tipo $f_{\vec{w}, b}(\vec{x}^{(i)}) = w_1 x + w_2 x^2 + b$ para os dados de treinamento abaixo, onde m varia.



Observação:

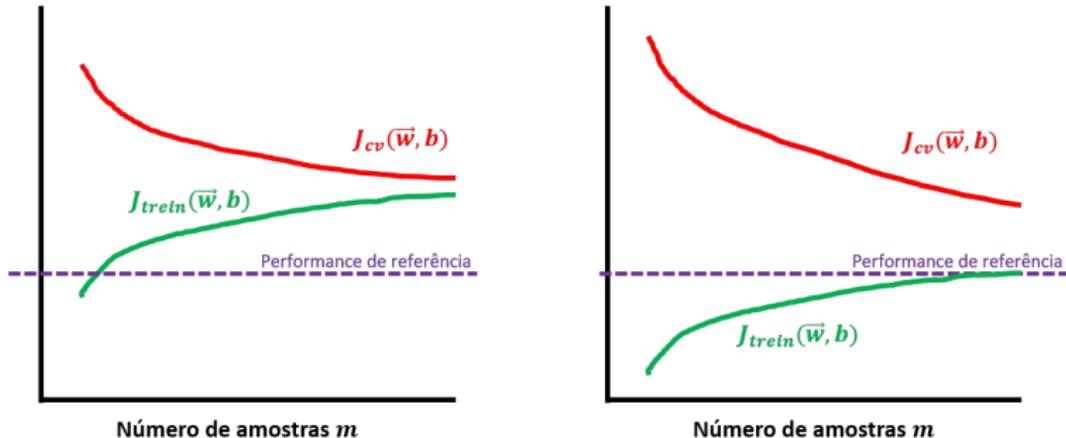
À medida com que mais amostras são utilizadas, o modelo vai conhecendo e se adequando melhor ao comportamento mais fundamental presente nos dados. J_{trein} tende a aumentar um pouco, porém J_{cv} tende a cair.

Como o número de amostras afeta o desempenho do seu modelo?



Pergunta

Onde queremos estar nessa figura?



Perguntas

- Qual é o tipo de problema que está provavelmente acontecendo com o modelo da esquerda? Sobrestimação ou Subestimação?
- Qual é o tipo de problema que está provavelmente acontecendo com o modelo da direita? Sobrestimação ou Subestimação?
- Qual dos dois tipos de problema mais se beneficia do aumento do número de amostras?

De olho no código!

De olho no código!

Iremos agora verificar como implementar na prática os conceitos vistos até aqui.

Acesse o Python Notebook usando o QR code ou o link abaixo:



https://colab.research.google.com/github/xaximpv2/master/blob/main/codigo_aula_21_Dicas_para_implementar_e_testar_metodos_e_modelos.ipynb

Parte 1

Rode todo o código. Certifique-se de que você o compreendeu.

Parte 2

- 1** Compare as curvas obtidas por meio do código com as curvas teóricas apresentadas nos slides. Faça uma análise das diferenças e semelhanças.
- 2** Mostre que aumentar o número de amostras m tende a não resolver problemas de alto viés (subestimação). **Dica:** teste um modelo bem simples para os dados (por exemplo, uma reta).

Como implementar/testar métodos e modelos (redes neurais)



Suponha que você está implementando uma Regressão Linear com regularização para o problema de estimação do módulo da tensão num determinado ponto de um sistema elétrico complexo:

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m \left(f_{\vec{w}, b} (\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

Você percebeu que as previsões feitas pelo modelo atual são inaceitáveis. **O que fazer?**
Podemos tentar o seguinte:

- Conseguir mais dados → (em caso de sobrestimação)

Relembrando da última aula...

Suponha que você está implementando uma Regressão Linear com regularização para o problema de estimação do módulo da tensão num determinado ponto de um sistema elétrico complexo:

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m \left(f_{\vec{w}, b} (\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

Você percebeu que as previsões feitas pelo modelo atual são inaceitáveis. **O que fazer?**
Podemos tentar o seguinte:

- Conseguir mais dados → (em caso de sobrestimação)
- Excluir características (testar modelo mais simples) → (em caso de sobrestimação)

Suponha que você está implementando uma Regressão Linear com regularização para o problema de estimação do módulo da tensão num determinado ponto de um sistema elétrico complexo:

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m \left(f_{\vec{w}, b} (\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

Você percebeu que as previsões feitas pelo modelo atual são inaceitáveis. **O que fazer?**

Podemos tentar o seguinte:

- Conseguir mais dados → (em caso de sobrestimação)
- Excluir características (testar modelo mais simples) → (em caso de sobrestimação)
- Incluir características (testar modelo mais complexo) → (em caso de subestimação)

Relembrando da última aula...

Suponha que você está implementando uma Regressão Linear com regularização para o problema de estimação do módulo da tensão num determinado ponto de um sistema elétrico complexo:

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m \left(f_{\vec{w}, b} (\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

Você percebeu que as previsões feitas pelo modelo atual são inaceitáveis. **O que fazer?**

Podemos tentar o seguinte:

- Conseguir mais dados → (em caso de sobrestimação)
- Excluir características (testar modelo mais simples) → (em caso de sobrestimação)
- Incluir características (testar modelo mais complexo) → (em caso de subestimação)
- Adicionar características polinomiais ao modelo ($x_1^2, x_2^2, x_1x_2, \dots$) → (em caso de subestimação)

Relembrando da última aula...

Suponha que você está implementando uma Regressão Linear com regularização para o problema de estimação do módulo da tensão num determinado ponto de um sistema elétrico complexo:

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m \left(f_{\vec{w}, b} (\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

Você percebeu que as previsões feitas pelo modelo atual são inaceitáveis. **O que fazer?**

Podemos tentar o seguinte:

- ➊ Conseguir mais dados → (em caso de sobrestimação)
- ➋ Excluir características (testar modelo mais simples) → (em caso de sobrestimação)
- ➌ Incluir características (testar modelo mais complexo) → (em caso de subestimação)
- ➍ Adicionar características polinomiais ao modelo ($x_1^2, x_2^2, x_1x_2, \dots$) → (em caso de subestimação)
- ➎ Aumentar λ → (em caso de sobrestimação)

Relembrando da última aula...

Suponha que você está implementando uma Regressão Linear com regularização para o problema de estimação do módulo da tensão num determinado ponto de um sistema elétrico complexo:

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m \left(f_{\vec{w}, b} (\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

Você percebeu que as previsões feitas pelo modelo atual são inaceitáveis. **O que fazer?**

Podemos tentar o seguinte:

- ➊ Conseguir mais dados → (em caso de sobrestimação)
- ➋ Excluir características (testar modelo mais simples) → (em caso de sobrestimação)
- ➌ Incluir características (testar modelo mais complexo) → (em caso de subestimação)
- ➍ Adicionar características polinomiais ao modelo ($x_1^2, x_2^2, x_1x_2, \dots$) → (em caso de subestimação)
- ➎ Aumentar λ → (em caso de sobrestimação)
- ➏ Reduzir λ → (em caso de subestimação)

Relembrando da última aula...

Suponha que você está implementando uma Regressão Linear com regularização para o problema de estimação do módulo da tensão num determinado ponto de um sistema elétrico complexo:

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m \left(f_{\vec{w}, b} (\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

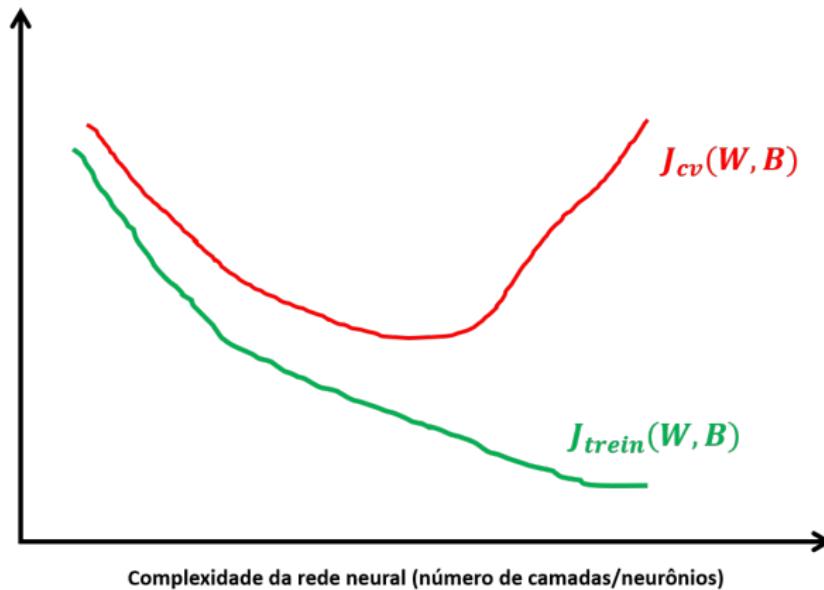
Você percebeu que as previsões feitas pelo modelo atual são inaceitáveis. **O que fazer?**

Podemos tentar o seguinte:

- Conseguir mais dados → (em caso de sobrestimação)
- Excluir características (testar modelo mais simples) → (em caso de sobrestimação)
- Incluir características (testar modelo mais complexo) → (em caso de subestimação)
- Adicionar características polinomiais ao modelo ($x_1^2, x_2^2, x_1x_2, \dots$) → (em caso de subestimação)
- Aumentar λ → (em caso de sobrestimação)
- Reduzir λ → (em caso de subestimação)

Pergunta: Essas mesmas considerações se aplicam às **redes neurais**?

Underfitting e overfitting usando J_{trein} e J_{cv} (redes neurais)



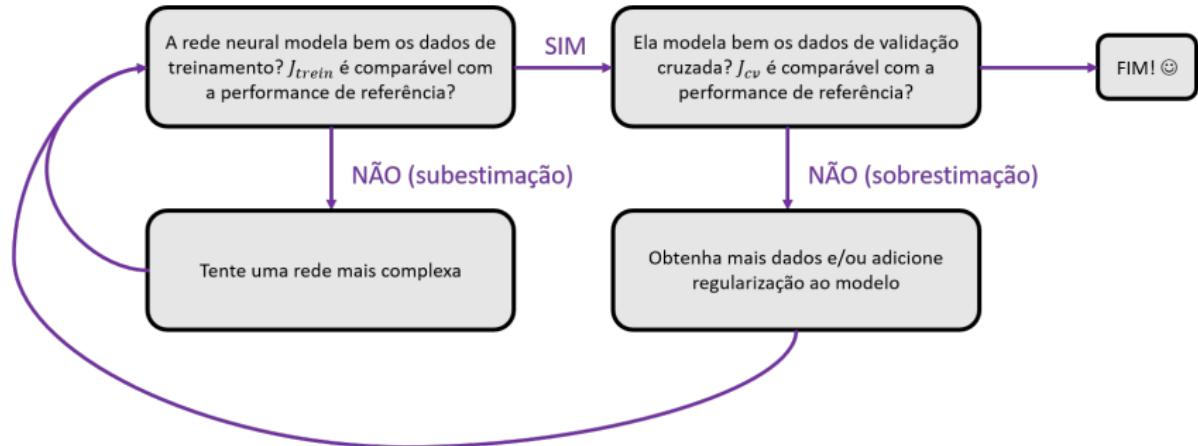
Pergunta

Onde queremos estar nessa figura?

Regra prática:

Teste desde modelos mais simples até modelos altamente complexos. Selecione aquele que resulta no menor J_{cv} .

Um passo-a-passo muito útil



Dica 1:

Desde que você escolha um valor adequado para o parâmetro de regularização λ , é possível usar uma rede neural bastante complexa (muitas camadas e neurônios) sem ter o problema de sobreestimação. **OBS:** Veremos no código dessa aula como implementamos regularização ao treinar uma rede neural (para facilitar, escolheremos um λ igual para todas as camadas).

Dica 1:

Desde que você escolha um valor adequado para o parâmetro de regularização λ , é possível usar uma rede neural bastante complexa (muitas camadas e neurônios) sem ter o problema de sobreestimação. **OBS:** Veremos no código dessa aula como implementaremos regularização ao treinar uma rede neural (para facilitar, escolheremos um λ igual para todas as camadas).

Dica 2:

Na dúvida, teste uma rede neural mais complexa e faça uma busca pelo valor de λ ótimo para aquela estrutura. Isso simplifica muito o processo de escolha da arquitetura de uma rede. Os resultados podem ser muito melhores do que aqueles obtidos por meio de uma rede simples. Porém, um dos efeitos colaterais disso será o aumento no tempo computacional gasto para treinar o modelo.

De olho no código!

De olho no código!

Iremos agora verificar como avaliar a qualidade de uma rede neural na prática.

Acesse o Python Notebook usando o QR code ou o link abaixo:



https://colab.research.google.com/github/xaximpv2/master/blob/main/codigo_aula21_Avaliando_a_performance_de_uma_rede_neural.ipynb

Parte 1

Rode todo o código. Certifique-se de que você o compreendeu.

Parte 2

- 1 Refaça todas as análises presentes no código usando a **taxa de acerto** ao invés da **taxa de erro**.

Dados representativos e transferência de conhecimento

Esta aula constitui um tópico adicional da disciplina. Trata-se de um conteúdo opcional. Sua atividade não valerá nota e não precisa ser enviada.



Importância de se ter uma base de dados representativa

Importância de se ter uma base de dados representativa

Busque sempre garantir que a quantidade de dados que você tem explica de forma suficiente o problema que está sendo modelado (porém, não basta apenas quantidade, é preciso ter diversidade nas amostras).

Importância de se ter uma base de dados representativa

Busque sempre garantir que a quantidade de dados que você tem explica de forma suficiente o problema que está sendo modelado (porém, não basta apenas quantidade, é preciso ter diversidade nas amostras).

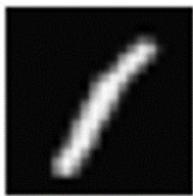
Exemplo:

Você deseja construir um modelo que seja capaz de reconhecer dígitos 0/1 escritos à mão e o seu conjunto de dados possui apenas as 4 amostras abaixo. Será que esse conjunto de amostras contempla uma variedade suficiente de dígitos 0/1 escritos à mão? Basta apenas duplicar essas 4 amostras 1000 vezes, por exemplo?

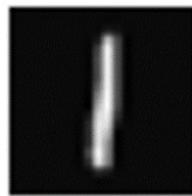
0



1



1



0

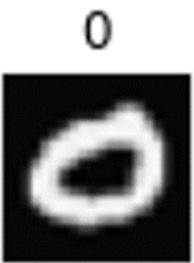
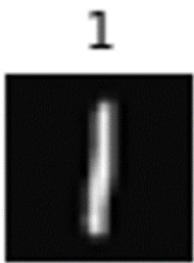
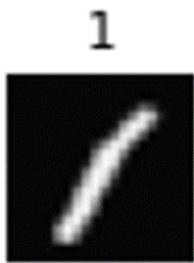


Importância de se ter uma base de dados representativa

Busque sempre garantir que a quantidade de dados que você tem explica de forma suficiente o problema que está sendo modelado (porém, não basta apenas quantidade, é preciso ter diversidade nas amostras).

Exemplo:

Você deseja construir um modelo que seja capaz de reconhecer dígitos 0/1 escritos à mão e o seu conjunto de dados possui apenas as 4 amostras abaixo. Será que esse conjunto de amostras contempla uma variedade suficiente de dígitos 0/1 escritos à mão? Basta apenas duplicar essas 4 amostras 1000 vezes, por exemplo?



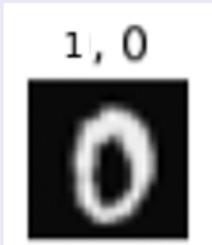
Quando a base de dados não é representativa, é possível que ocorra $J_{cv} \gg J_{trein}$. Ou seja, o modelo se ajusta aos dados de estimativa, mas isso não é suficiente para explicar o problema como um todo. Por isso, o modelo performa mal para novas amostras.

Após treinar um modelo, busque sempre **analisar os erros cometidos pelo modelo**. A análise dos erros pode prover diversas informações relevantes, por exemplo:

- Percebe-se que o modelo erra mais para uma determinada categoria. Existe diversidade suficiente nas amostras de exemplo para essa categoria? → **Solução possível:** Obter mais amostras para essa categoria!

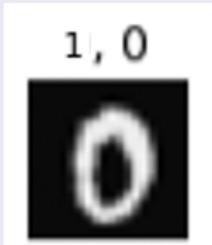
Após treinar um modelo, busque sempre **analisar os erros cometidos pelo modelo**. A análise dos erros pode prover diversas informações relevantes, por exemplo:

- Percebe-se que o modelo erra mais para uma determinada categoria. Existe diversidade suficiente nas amostras de exemplo para essa categoria? → **Solução possível:** Obter mais amostras para essa categoria!
- Existem amostras rotuladas de forma equivocada e o modelo classificou corretamente?

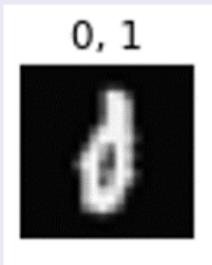


Após treinar um modelo, busque sempre **analisar os erros cometidos pelo modelo**. A análise dos erros pode prover diversas informações relevantes, por exemplo:

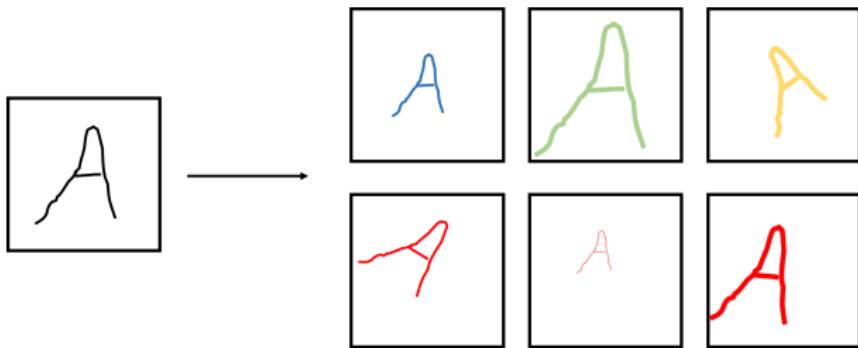
- Percebe-se que o modelo erra mais para uma determinada categoria. Existe diversidade suficiente nas amostras de exemplo para essa categoria? → **Solução possível:** Obter mais amostras para essa categoria!
- Existem amostras rotuladas de forma equivocada e o modelo classificou corretamente?



- Tratam-se de erros "aceitáveis"?

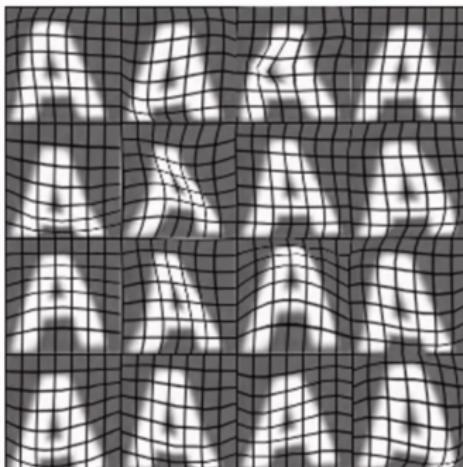
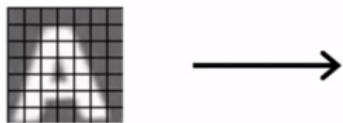


Consiste em criar novas amostras a partir de amostras já existentes (com o objetivo de aumentar a diversidade):



Consiste em criar novas amostras a partir de amostras já existentes (com o objetivo de aumentar a diversidade):

Exemplo de distorções possíveis:



Consiste em criar novas amostras a partir de amostras já existentes (com o objetivo de aumentar a diversidade):

Em um sistema de reconhecimento de voz, poderíamos gravar uma pessoa falando, por exemplo: "Alexa, timer de 10 minutos". Em seguida, poderíamos sobrepor a esse áudio diversos tipos de ruído de fundo:

- Pessoas conversando
- Trânsito intenso
- Música de fundo

Observação importante: Adicionar ruído aleatório branco aos dados, em geral, não trás benefício. Para que sejam úteis, as amostras criadas precisam ter relação com o que o modelo possivelmente irá encontrar nas amostras de validação cruzada e de teste.

Geração sintética de dados

Consiste em criar novas amostras do zero, sem usar amostras já existentes:



Real data

[Adam Coates and Tao Wang]



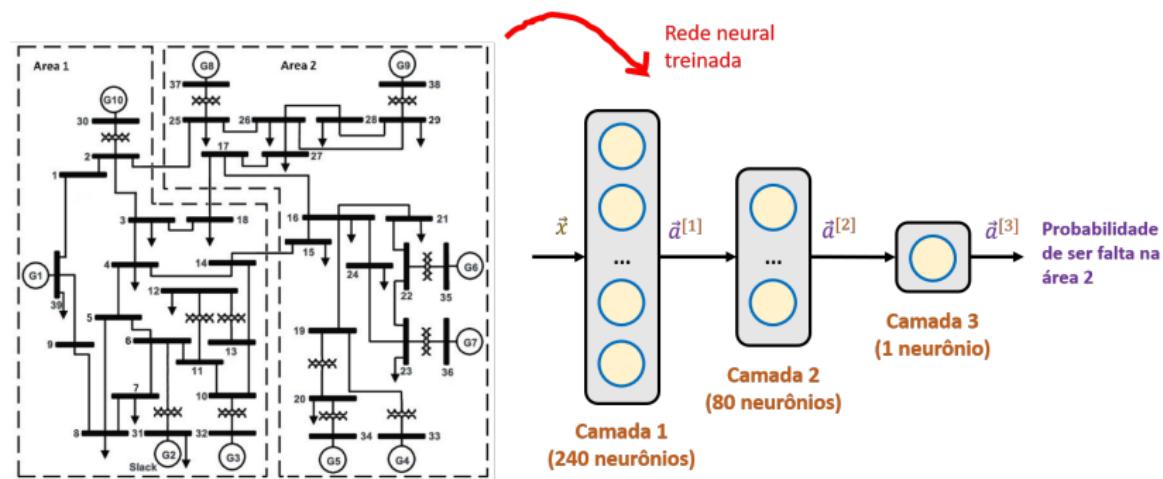
Synthetic data

Os dados sintéticos acima apresentados foram criados usando um editor de texto comum. Percebe-se que eles representam bem letras encontradas no "mundo real".

Transferindo conhecimento de uma aplicação para outra

Transferindo conhecimento de uma aplicação para outra

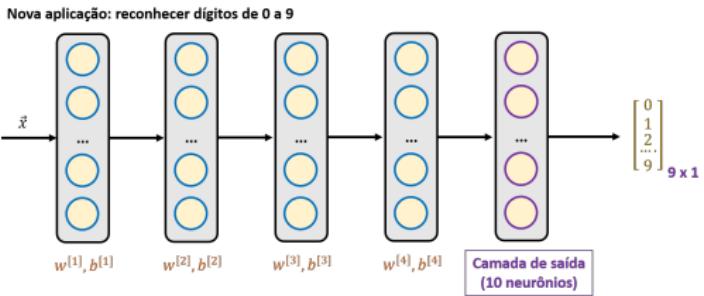
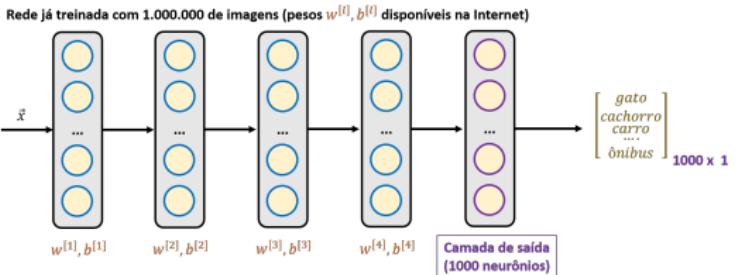
Suponha que você **simulou** o sistema abaixo e treinou um localizador de área sob falta.



Pergunta:

Seria possível considerar os pesos dessa rede neural já treinada como pesos iniciais para modelagem de um outro sistema onde serão usados dados medidos reais?

Um outro exemplo:



- Opção 1: treinar apenas os parâmetros da camada de saída (que precisa ser integralmente substituída)
- Opção 2: treinar todos os parâmetros, mas usando parâmetros vindos da rede original como valores iniciais

De olho no código!

Em aulas anteriores, resolvemos o problema de classificação de microchips usando Regressão Logística + Características Polinomiais. Nessa aula, você irá resolver o mesmo problema via Redes Neurais.

Acesse o Python Notebook usando o QR code ou o link abaixo:

https://colab.research.google.com/github/xaximpv2/master/blob/main/codigo_aula21_topico_adicional_resolvendo_microchip_usando_RNA.ipynb



Acesse os dados necessários para rodar o código usando o QR code ou o link abaixo:

https://ufprbr0-my.sharepoint.com/:t/g/personal/ricardo_schumacher_ufpr_br/Ee6CfYb1cDFEkmfx8FCVXS4B80-1f5UV3dZunU3R_hY-JQ?e=D1WRIf



OBS: Para adicionar os dados ao ambiente do Colab Notebook, no menu do canto esquerdo da tela do Colab clique em "Arquivos" e depois "Fazer upload para o armazenamento da sessão". Então carregue os arquivos baixados.

Parte 1

Rode todo o código. Certifique-se que você o compreendeu.

Parte 2

- 1 Treine uma rede neural para o problema e mostre que ela representa um classificador adequado para essa aplicação.

Conjuntos de dados com distribuição não uniforme (desbalanceados)



Conjuntos de dados com distribuição não uniforme (desbalanceados)

Pergunta:

Taxa de acerto/erro é sempre uma boa métrica?

Exemplo:

- Você está construindo um algoritmo que reconhece uma determinada doença rara ($y = 1$ caso a doença esteja presente, e $y = 0$ caso não esteja)
- Seu conjunto de dados é formado por dados de 1000 pacientes, onde apenas 5 deles possui a doença
- Você verificou que a taxa de acerto do seu modelo para os **dados de teste** é de 99.5%.
- Tudo certo então?

Conjuntos de dados com distribuição não uniforme

Pergunta:

Taxa de acerto/erro é sempre uma boa métrica?

Exemplo:

- Você está construindo um algoritmo que reconhece uma determinada doença rara ($y = 1$ caso a doença esteja presente, e $y = 0$ caso não esteja)
- Seu conjunto de dados é formado por dados de 1000 pacientes, onde apenas 5 deles possui a doença
- Você verificou que a taxa de acerto do seu modelo para os **dados de teste** é de 99.5%.
- Tudo certo então?

Pergunta:

- Qual seria a taxa de acerto do modelo que considera $\hat{y} = 0$ para qualquer paciente?
- O que esse modelo aprendeu sobre a doença?

Conjuntos de dados com distribuição não uniforme

Pergunta:

Taxa de acerto/erro é sempre uma boa métrica?

Exemplo:

- Você está construindo um algoritmo que reconhece uma determinada doença rara ($y = 1$ caso a doença esteja presente, e $y = 0$ caso não esteja)
- Seu conjunto de dados é formado por dados de 1000 pacientes, onde apenas 5 deles possui a doença
- Você verificou que a taxa de acerto do seu modelo para os **dados de teste** é de 99.5%.
- Tudo certo então?

Pergunta:

- Qual seria a taxa de acerto do modelo que considera $\hat{y} = 0$ para qualquer paciente?
- O que esse modelo aprendeu sobre a doença?

Resposta:

- Temos muitos $y = 0$ e poucos $y = 1$. Ou seja, os dados possuem uma distribuição não uniforme. Nesses casos, a taxa de acerto pode não ser uma boa métrica para avaliar a qualidade do seu modelo.

Matriz de confusão

The diagram illustrates the transformation of a raw confusion matrix into a labeled one. On the left, a raw confusion matrix is shown with columns labeled by the true class (1 or 0) and rows labeled by the predicted class (1 or 0). The values are: True Class 1, Predicted Class 1: 15; True Class 1, Predicted Class 0: 5; True Class 0, Predicted Class 1: 10; True Class 0, Predicted Class 0: 70. An arrow points from this matrix to a labeled confusion matrix on the right. The labeled matrix has the same structure but includes labels for each cell: True Class 1, Predicted Class 1 is labeled "VERDADEIROS POSITIVOS" (True Positives); True Class 1, Predicted Class 0 is labeled "FALSOS POSITIVOS" (False Positives); True Class 0, Predicted Class 1 is labeled "FALSOS NEGATIVOS" (False Negatives); and True Class 0, Predicted Class 0 is labeled "VERDADEIROS NEGATIVOS" (True Negatives).

		CLASSE VERDADEIRA	
		1	0
CLASSE PREVISTA	1	15	5
	0	10	70

→

		CLASSE VERDADEIRA	
		1	0
CLASSE PREVISTA	1	VERDADEIROS POSITIVOS	FALSOS POSITIVOS
	0	FALSOS NEGATIVOS	VERDADEIROS NEGATIVOS

Matriz de confusão

The diagram illustrates the transformation of a raw confusion matrix into a labeled one. On the left, a raw confusion matrix is shown with columns labeled by the true class (1 or 0) and rows labeled by the predicted class (1 or 0). The values are: True Class 1, Predicted Class 1: 15; True Class 1, Predicted Class 0: 5; True Class 0, Predicted Class 1: 10; True Class 0, Predicted Class 0: 70. An arrow points to the right, where the same matrix is labeled to indicate true and false positives/negatives.

CLASSE VERDADEIRA		CLASSE VERDADEIRA	
		1	0
CLASSE PREVISTA	1	15	5
	0	10	70

→

CLASSE VERDADEIRA		CLASSE VERDADEIRA	
		1	0
CLASSE PREVISTA	1	VERDADEIROS POSITIVOS 15	FALSOS POSITIVOS 5
	0	FALSOS NEGATIVOS 10	VERDADEIROS NEGATIVOS 70

Considere a matriz de confusão acima. Pergunta-se:

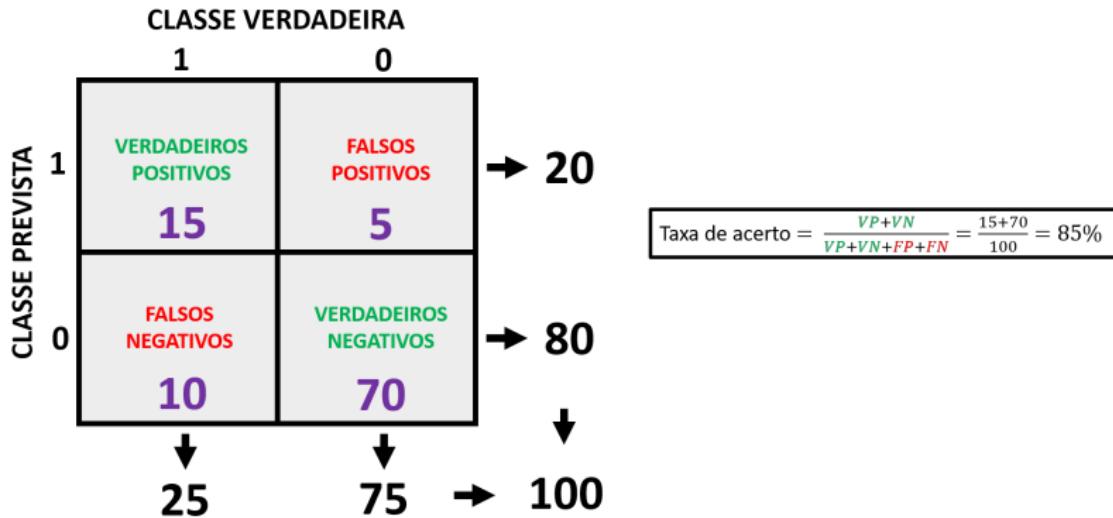
- Quantos pacientes que possuem a doença foram corretamente considerados doentes pelo modelo? (verdadeiros positivos) → $(y = 1, \hat{y} = 1)$
- Quantos pacientes que não possuem a doença foram corretamente considerados não doentes pelo modelo? (verdadeiros negativos) → $(y = 0, \hat{y} = 0)$
- Quantos pacientes que possuem a doença foram incorretamente considerados não doentes pelo modelo? (falsos negativos) → $(y = 1, \hat{y} = 0)$
- Quantos pacientes que não possuem a doença foram incorretamente considerados doentes pelo modelo? (falsos positivos) → $(y = 0, \hat{y} = 1)$

		CLASSE VERDADEIRA		
		1	0	
CLASSE PREVISTA	1	VERDADEIROS POSITIVOS 15	FALSOS POSITIVOS 5	→ 20
	0	FALSOS NEGATIVOS 10	VERDADEIROS NEGATIVOS 70	→ 80
		↓ 25	↓ 75	→ 100

Pergunta-se ainda:

- Quantos pacientes são de fato doentes? (amostras com $y = 1$)
- Quantos pacientes são não doentes? (amostras com $y = 0$)
- Qual é a proporção entre classes 1 e classes 0?
- Quantos pacientes foram classificados como doentes? (amostras com $\hat{y} = 1$)
- Quantos pacientes foram classificados como não doentes? (amostras com $\hat{y} = 0$)
- O número total de pacientes é dado por $VP+VN+FP+FN$?

Matriz de confusão

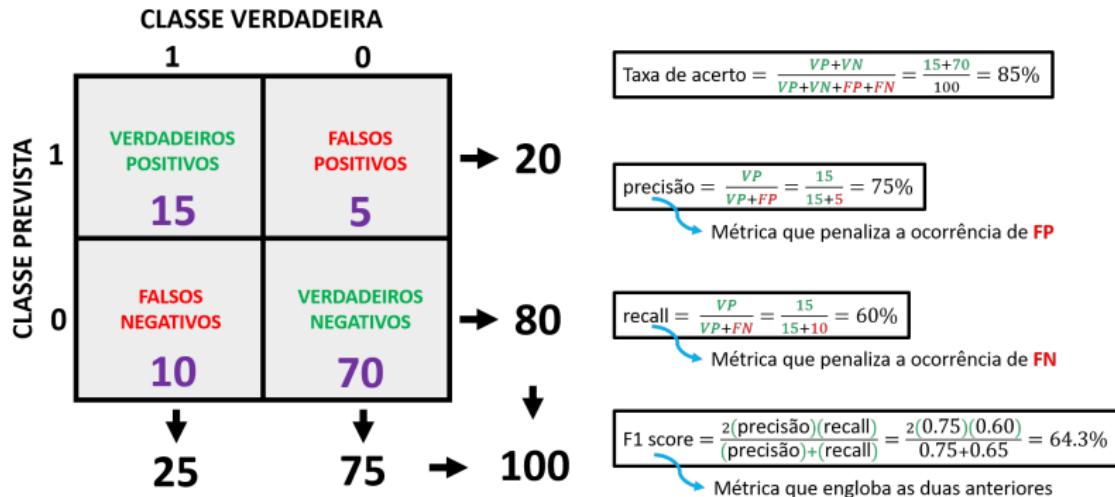


Pergunta-se ainda:

- Parece um bom modelo para você?
- Você concorda que a taxa de acerto desse modelo é dada conforme abaixo?

$$\text{taxa de acerto} = \frac{\text{verdadeiros positivos} + \text{verdadeiros negativos}}{\text{número total de amostras}}$$

Mais duas métricas importantes: Precisão e Recall (Revocação)



$$\text{precisão} = \frac{\text{pacientes considerados doentes e que eram de fato doentes}}{\text{pacientes considerados doentes}}$$

$$\text{recall} = \frac{\text{pacientes considerados doentes e que eram de fato doentes}}{\text{pacientes de fato doentes}}$$

Voltando para o exemplo anterior onde o modelo considera $\hat{y} = 0$ para qualquer paciente

CLASSE VERDADEIRA		
		0
1	VERDADEIROS POSITIVOS	FALSOS POSITIVOS
1	0	0
0	FALSOS NEGATIVOS	VERDADEIROS NEGATIVOS
0	5	995

CLASSE PREDITA → 0

↓ 5 ↓ 995 → 1000

$$\text{Taxa de acerto} = \frac{VP+VN}{VP+VN+FP+FN} = \frac{0+995}{1000} = 99.5\%$$

$$\text{precisão} = \frac{VP}{VP+FP} = \frac{0}{0+0} = div/0$$

Métrica que penaliza a ocorrência de FP

$$\text{recall} = \frac{VP}{VP+FN} = \frac{0}{0+5} = 0\%$$

Métrica que penaliza a ocorrência de FN

$$\text{F1 score} = \frac{2(\text{precisão})(\text{recall})}{(\text{precisão})+(\text{recall})} = \frac{2(div/0)(0)}{div/0+0} = ?$$

Métrica que engloba as duas anteriores

Não parece um bom modelo...

Algumas conclusões até aqui...

- 1 Quando temos um conjunto de dados desbalanceado (mais $y = 1$ do que $y = 0$), vimos que a taxa de acerto (acurácia) pode não ser uma boa métrica.

- 1 Quando temos um conjunto de dados desbalanceado (mais $y = 1$ do que $y = 0$), vimos que a taxa de acerto (acurácia) pode não ser uma boa métrica.
- 2 Uma solução para este problema consiste em utilizar outras métricas, tais como **precisão**, **recall** (revocação) e **F1 score**.

- 1 Quando temos um conjunto de dados desbalanceado (mais $y = 1$ do que $y = 0$), vimos que a taxa de acerto (acurácia) pode não ser uma boa métrica.
- 2 Uma solução para este problema consiste em utilizar outras métricas, tais como **precisão**, **recall** (revocação) e **F1 score**.
- 3 É desejável que um modelo apresente precisão e recall próximos de 100 %. Isso também resultará num F1 score próximo de 100 %.

- 1 Quando temos um conjunto de dados desbalanceado (mais $y = 1$ do que $y = 0$), vimos que a taxa de acerto (acurácia) pode não ser uma boa métrica.
- 2 Uma solução para este problema consiste em utilizar outras métricas, tais como **precisão**, **recall** (revocação) e **F1 score**.
- 3 É desejável que um modelo apresente precisão e recall próximos de 100 %. Isso também resultará num F1 score próximo de 100 %.
- 4 Entretanto, é muito importante sempre avaliar o caso concreto que está sob estudo.

- 1 Quando temos um conjunto de dados desbalanceado (mais $y = 1$ do que $y = 0$), vimos que a taxa de acerto (acurácia) pode não ser uma boa métrica.
- 2 Uma solução para este problema consiste em utilizar outras métricas, tais como **precisão**, **recall** (revocação) e **F1 score**.
- 3 É desejável que um modelo apresente precisão e recall próximos de 100 %. Isso também resultará num F1 score próximo de 100 %.
- 4 Entretanto, é muito importante sempre avaliar o caso concreto que está sob estudo.
- 5 Veremos um exemplo disso agora sobre transações financeiras fraudulentas.

Exemplo: transações financeiras fraudulentas

Suponha que você treinou uma rede neural que objetiva identificar transações financeiras fraudulentas envolvendo cartão de crédito.

Suponha que você treinou uma rede neural que objetiva identificar transações financeiras fraudulentas envolvendo cartão de crédito.

- 1 O conjunto de dados de validação possui registro de um total de 56961 transações financeiras

Suponha que você treinou uma rede neural que objetiva identificar transações financeiras fraudulentas envolvendo cartão de crédito.

- 1 O conjunto de dados de validação possui registro de um total de 56961 transações financeiras
- 2 Desse total, 56886 são transações legítimas ($y = 0$) e 75 são fraudulentas ($y = 1$)

Suponha que você treinou uma rede neural que objetiva identificar transações financeiras fraudulentas envolvendo cartão de crédito.

- 1 O conjunto de dados de validação possui registro de um total de 56961 transações financeiras
- 2 Desse total, 56886 são transações legítimas ($y = 0$) e 75 são fraudulentas ($y = 1$)
- 3 Note o evidente desbalanceamento presente nos dados.

Suponha que você treinou uma rede neural que objetiva identificar transações financeiras fraudulentas envolvendo cartão de crédito.

- 1 O conjunto de dados de validação possui registro de um total de 56961 transações financeiras
- 2 Desse total, 56886 são transações legítimas ($y = 0$) e 75 são fraudulentas ($y = 1$)
- 3 Note o evidente desbalanceamento presente nos dados.
- 4 Note que um modelo hipotético do tipo $\hat{y} = 0$ teria uma taxa de acerto muito próxima de 100 %.
Entretanto, nenhuma operação fraudulenta seria detectada por este modelo.

Suponha que você treinou uma rede neural que objetiva identificar transações financeiras fraudulentas envolvendo cartão de crédito.

- 1 O conjunto de dados de validação possui registro de um total de 56961 transações financeiras
- 2 Desse total, 56886 são transações legítimas ($y = 0$) e 75 são fraudulentas ($y = 1$)
- 3 Note o evidente desbalanceamento presente nos dados.
- 4 Note que um modelo hipotético do tipo $\hat{y} = 0$ teria uma taxa de acerto muito próxima de 100 %. Entretanto, nenhuma operação fraudulenta seria detectada por este modelo.

A rede neural foi submetida aos dados acima descritos, resultando na **matriz de confusão** do próximo slide.

		CLASSE VERDADEIRA	
		1	0
CLASSE PREDICTA	1	VERDADEIROS POSITIVOS 67	FALSOS POSITIVOS 1767
	0	FALSOS NEGATIVOS 8	VERDADEIROS NEGATIVOS 55119
		75	56886 → 56961

Pergunta-se:

Transações financeiras fraudulentas

		CLASSE VERDADEIRA	
		1	0
CLASSE PREVISTA	1	VERDADEIROS POSITIVOS 67	FALSOS POSITIVOS 1767
	0	FALSOS NEGATIVOS 8	VERDADEIROS NEGATIVOS 55119
		75	56886 → 56961

→ 1834
→ 55127
↓
↓

Pergunta-se:

- Quantas transações fraudulentas foram corretamente consideradas fraudulentas pelo modelo? (verdadeiros positivos) → ($y = 1, \hat{y} = 1$)

		CLASSE VERDADEIRA	
		1	0
CLASSE PREVISTA	1	VERDADEIROS POSITIVOS 67	FALSOS POSITIVOS 1767
	0	FALSOS NEGATIVOS 8	VERDADEIROS NEGATIVOS 55119
		1834	55127
		75	56886 → 56961

Pergunta-se:

- Quantas transações fraudulentas foram corretamente consideradas fraudulentas pelo modelo? (verdadeiros positivos) → ($y = 1, \hat{y} = 1$)
- Quantas transações não fraudulentas foram corretamente consideradas não fraudulentas pelo modelo? (verdadeiros negativos) → ($y = 0, \hat{y} = 0$)

Transações financeiras fraudulentas

		CLASSE VERDADEIRA	
		1	0
CLASSE PREVISTA	1	VERDADEIROS POSITIVOS 67	FALSOS POSITIVOS 1767
	0	FALSOS NEGATIVOS 8	VERDADEIROS NEGATIVOS 55119
		1834	55127
		75	56886 → 56961

Pergunta-se:

- Quantas transações fraudulentas foram corretamente consideradas fraudulentas pelo modelo? (verdadeiros positivos) → ($y = 1, \hat{y} = 1$)
- Quantas transações não fraudulentas foram corretamente consideradas não fraudulentas pelo modelo? (verdadeiros negativos) → ($y = 0, \hat{y} = 0$)
- Quantas transações fraudulentas foram incorretamente consideradas não fraudulentas pelo modelo? (falsos negativos) → ($y = 1, \hat{y} = 0$)

		CLASSE VERDADEIRA	
		1	0
CLASSE PREVISTA	1	VERDADEIROS POSITIVOS 67	FALSOS POSITIVOS 1767
	0	FALSOS NEGATIVOS 8	VERDADEIROS NEGATIVOS 55119
		1834	55127
		75	56886 → 56961

Pergunta-se:

- Quantas transações fraudulentas foram corretamente consideradas fraudulentas pelo modelo? (verdadeiros positivos) → ($y = 1, \hat{y} = 1$)
- Quantas transações não fraudulentas foram corretamente consideradas não fraudulentas pelo modelo? (verdadeiros negativos) → ($y = 0, \hat{y} = 0$)
- Quantas transações fraudulentas foram incorretamente consideradas não fraudulentas pelo modelo? (falsos negativos) → ($y = 1, \hat{y} = 0$)
- Quantas transações não fraudulentas foram incorretamente consideradas fraudulentas pelo modelo? (falsos positivos) → ($y = 0, \hat{y} = 1$)

Transações financeiras fraudulentas

		CLASSE VERDADEIRA	
		1	0
CLASSE PREVISTA	1	VERDADEIROS POSITIVOS 67	FALSOS POSITIVOS 1767
	0	FALSOS NEGATIVOS 8	VERDADEIROS NEGATIVOS 55119
		1834	55127
		75	56886 → 56961

Pergunta-se:

- Quantas transações fraudulentas foram corretamente consideradas fraudulentas pelo modelo? (**verdadeiros positivos**) → $(y = 1, \hat{y} = 1)$
- Quantas transações **não** fraudulentas foram corretamente consideradas não fraudulentas pelo modelo? (**verdadeiros negativos**) → $(y = 0, \hat{y} = 0)$
- Quantas transações fraudulentas foram incorretamente consideradas não fraudulentas pelo modelo? (**falsos negativos**) → $(y = 1, \hat{y} = 0)$
- Quantas transações **não** fraudulentas foram incorretamente consideradas fraudulentas pelo modelo? (**falsos positivos**) → $(y = 0, \hat{y} = 1)$
- E aí, trata-se de um bom modelo?

Transações financeiras fraudulentas

		CLASSE VERDADEIRA	
		1	0
CLASSE PREVISTA	1	VERDADEIROS POSITIVOS 67	FALSOS POSITIVOS 1767
	0	FALSOS NEGATIVOS 8	VERDADEIROS NEGATIVOS 55119
		75	56886 → 56961

→ 1834
→ 55127
↓
↓

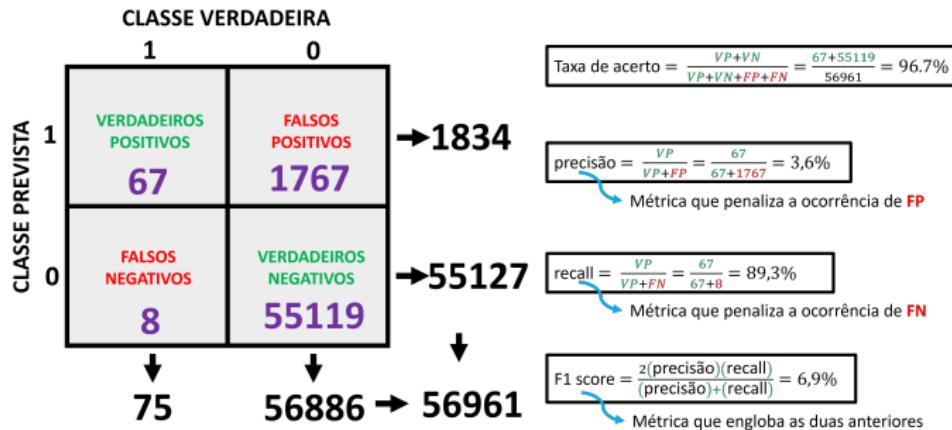
Pergunta-se:

- Quantas transações fraudulentas foram corretamente consideradas fraudulentas pelo modelo? (**verdadeiros positivos**) → ($y = 1, \hat{y} = 1$)
- Quantas transações **não** fraudulentas foram corretamente consideradas não fraudulentas pelo modelo? (**verdadeiros negativos**) → ($y = 0, \hat{y} = 0$)
- Quantas transações fraudulentas foram incorretamente consideradas não fraudulentas pelo modelo? (**falsos negativos**) → ($y = 1, \hat{y} = 0$)
- Quantas transações **não** fraudulentas foram incorretamente consideradas fraudulentas pelo modelo? (**falsos positivos**) → ($y = 0, \hat{y} = 1$)
- E aí, trata-se de um bom modelo?
- Note que o modelo é bastante razoável, pois ele foi capaz de identificar 67 das 75 transações fraudulentas presentes. Entretanto, 1767 transações não fraudulentas (de um total de 56886) foram classificadas como fraudulentas. "Na próxima vez que sua compra usando cartão de crédito não for autorizada, você já tem uma ideia do que pode ter acontecido".

Ou seja, acabamos de perceber que o modelo treinado é adequado à aplicação em tela (identificação de transações possivelmente fraudulentas). Entretanto, ao calcularmos as métricas precisão, recall e F1 score, chegamos nos seguintes resultados numéricos:

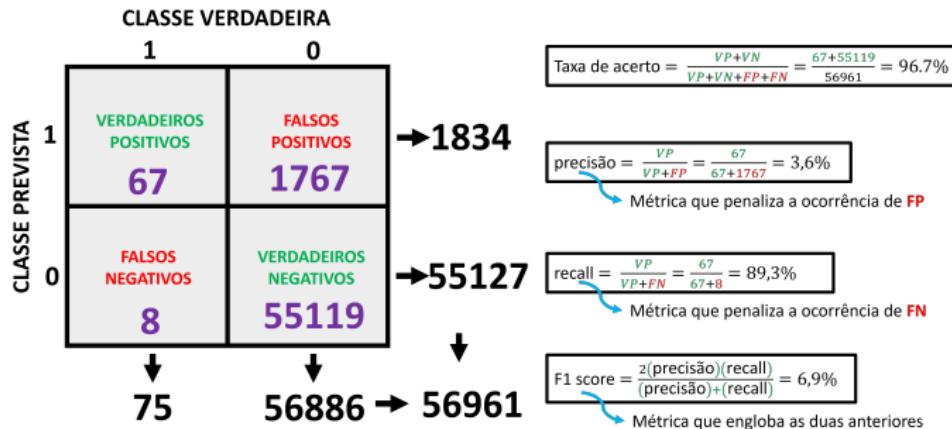
Transações financeiras fraudulentas

Ou seja, acabamos de perceber que o modelo treinado é adequado à aplicação em tela (identificação de transações possivelmente fraudulentas). Entretanto, ao calcularmos as métricas precisão, recall e F1 score, chegamos nos seguintes resultados numéricos:



Conclusões:

Ou seja, acabamos de perceber que o modelo treinado é adequado à aplicação em tela (identificação de transações possivelmente fraudulentas). Entretanto, ao calcularmos as métricas precisão, recall e F1 score, chegamos nos seguintes resultados numéricos:



Conclusões:

- Busque por boas métricas, mas use-as com cautela.
- Nunca se desconecte do problema concreto (real) que está sendo investigado.

- Quando temos mais rótulos do tipo $y = 1$, podemos, no momento do treinamento do modelo, dar um peso maior aos rótulos do tipo $y = 1$ em comparação com os rótulos do tipo $y = 0$.
- Isso fará com que a função custo penalize mais intensamente um erro de previsão relacionado ao rótulo $y = 1$ em comparação com um erro de previsão relacionado ao rótulo $y = 0$.
- Veremos como fazer isso nos próximos slides.

Dados desbalanceados: atribuindo pesos diferentes para cada classe

Opção 1:

$$\text{peso classe } 0 = \frac{1}{\text{quantidade de amostras com } y = 0}$$

$$\text{peso classe } 1 = \frac{1}{\text{quantidade de amostras com } y = 1}$$

Por exemplo, se temos 100 amostras com rótulo $y = 0$ e 50 com rótulo $y = 1$, então

$$\text{peso classe } 0 = \frac{1}{100} = 0.01$$

$$\text{peso classe } 1 = \frac{1}{50} = 0.02$$

Dados desbalanceados: atribuindo pesos diferentes aos rótulos

Opção 1:

$$\text{peso classe } 0 = \frac{1}{\text{quantidade de amostras com } y = 0}$$

$$\text{peso classe } 1 = \frac{1}{\text{quantidade de amostras com } y = 1}$$

Por exemplo, se temos 100 amostras com rótulo $y = 0$ e 50 com rótulo $y = 1$, então

$$\text{peso classe } 0 = \frac{1}{100} = 0.01$$

$$\text{peso classe } 1 = \frac{1}{50} = 0.02$$

Opção 2:

$$\text{peso classe } 0 = \frac{1}{\text{quantidade de amostras com } y = 0} \times \frac{\text{total de amostras}}{2}$$

$$\text{peso classe } 1 = \frac{1}{\text{quantidade de amostras com } y = 1} \times \frac{\text{total de amostras}}{2}$$

Para o mesmo exemplo, teríamos

$$\text{peso classe } 0 = \frac{1}{100} \times \frac{150}{2} = 0.75$$

$$\text{peso classe } 1 = \frac{1}{50} \times \frac{150}{2} = 1.5$$

De olho no código!

De olho no código!

Vamos agora ver um caso com dados reais, de transações fraudulentas envolvendo cartões de crédito. Os dados encontram-se fortemente desbalanceados e, por este motivo, faremos uma atribuição de pesos diferentes para cada classe.

Acesse o Python Notebook usando o QR code ou o link abaixo:



https://colab.research.google.com/github/xaximpv2/master/blob/main/codigo_aula22_classificacao_desbalanceada.ipynb

- Link para acesso à base de dados: <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>
- Acessando essa base de dados, note que não é possível saber o significado real de cada característica do problema.
- Isso porquê uma técnica do tipo PCA (Principal Component Analysis) foi utilizada para reduzir a dimensionalidade dos dados e abstrair seu sentido original (questões de sigilo).
- PCA é uma técnica de aprendizado não supervisionado usada para reduzir a dimensionalidade de um conjunto de dados, buscando manter as características (informações) principais presentes nele. A ideia central do PCA é transformar um grande número de variáveis correlacionadas em um número menor de variáveis não correlacionadas chamadas de componentes principais.

Parte 1

Rode todo o código. Responda às questões nele contidas e complete-o, se necessário.

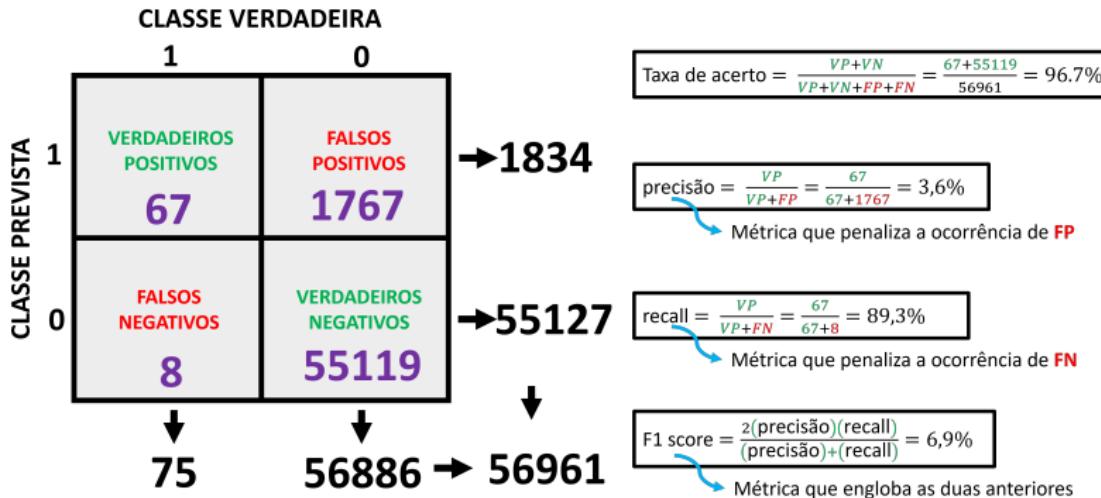
Parte 2

- 1) Interprete adequadamente os resultados obtidos pela matriz de confusão.
- 2) Descreva qual procedimento você poderia adotar visando reduzir a quantidade de falsos positivos gerados pelo modelo.

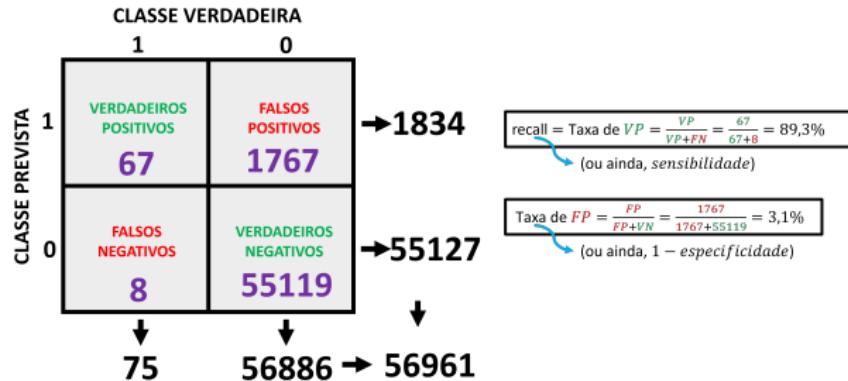
Novas métricas para classificação: ROC e AUC



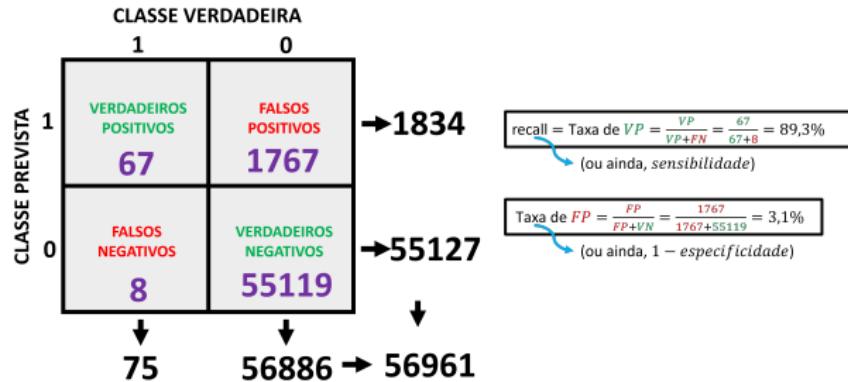
No caso das operações financeiras, chegamos em



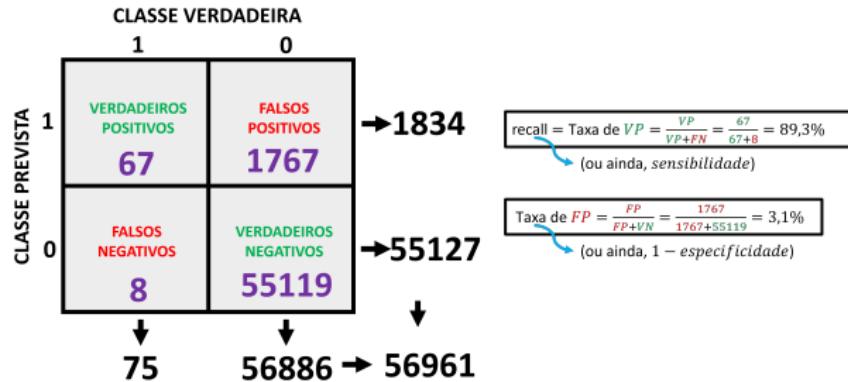
Algumas novas considerações sobre a matriz de confusão



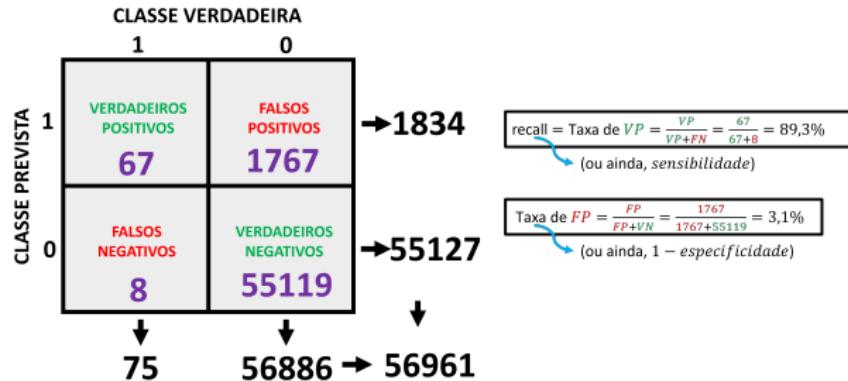
- 1 O *recall* (revocação) é também chamado de **taxa de verdadeiros positivos**, ou ainda, de **sensibilidade**.



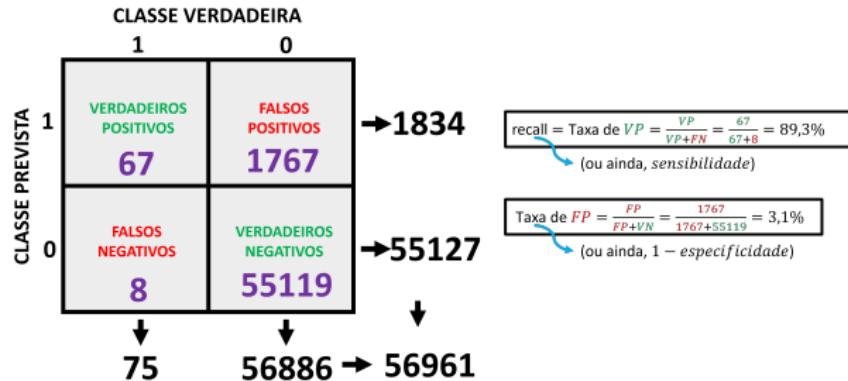
- O *recall* (revocação) é também chamado de **taxa de verdadeiros positivos**, ou ainda, de **sensibilidade**.
- Para este exemplo específico, ela representa a quantidade de transações corretamente classificadas como fraudulentas em relação à quantidade total de operações fraudulentas.



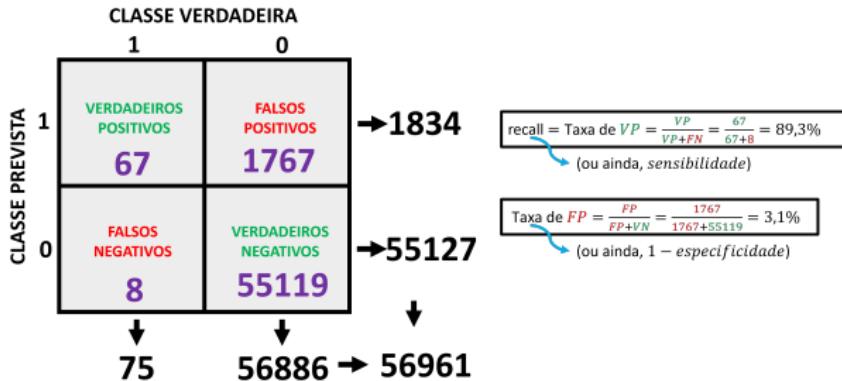
- 1 O *recall* (revocação) é também chamado de **taxa de verdadeiros positivos**, ou ainda, de **sensibilidade**.
- 2 Para este exemplo específico, ela representa a quantidade de transações corretamente classificadas como fraudulentas em relação à quantidade total de operações fraudulentas.
- 3 Quanto maior a **taxa de verdadeiros positivos**, melhor.



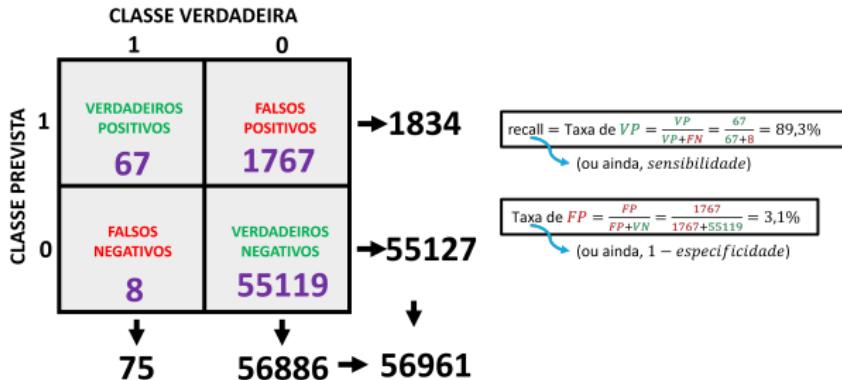
- 1 O **recall** (revocação) é também chamado de **taxa de verdadeiros positivos**, ou ainda, de **sensibilidade**.
- 2 Para este exemplo específico, ela representa a quantidade de transações corretamente classificadas como fraudulentas em relação à quantidade total de operações fraudulentas.
- 3 Quanto maior a **taxa de verdadeiros positivos**, melhor.
- 4 Analogamente, também podemos obter a chamada **taxa de falsos positivos**, sendo ela calculada pela expressão mostrada acima.



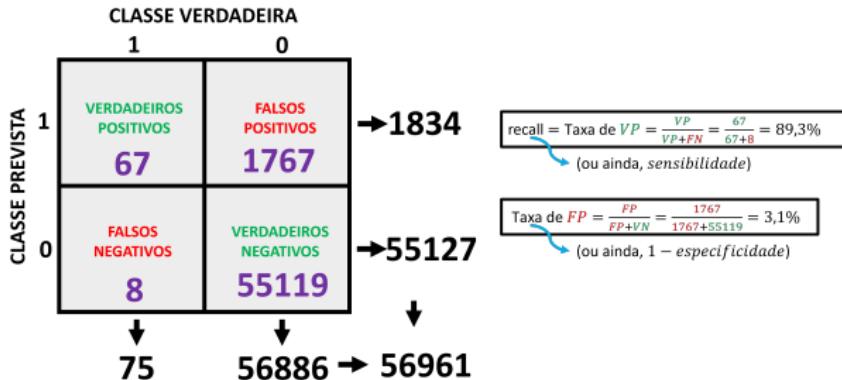
- 1 O **recall** (revocação) é também chamado de **taxa de verdadeiros positivos**, ou ainda, de **sensibilidade**.
- 2 Para este exemplo específico, ela representa a quantidade de transações corretamente classificadas como fraudulentas em relação à quantidade total de operações fraudulentas.
- 3 Quanto maior a **taxa de verdadeiros positivos**, melhor.
- 4 Analogamente, também podemos obter a chamada **taxa de falsos positivos**, sendo ela calculada pela expressão mostrada acima.
- 5 Para este exemplo específico, ela representa a quantidade de operações incorretamente classificadas como fraudulentas em relação ao total de operações não fraudulentas.



- O *recall* (revocação) é também chamado de **taxa de verdadeiros positivos**, ou ainda, de **sensibilidade**.
- Para este exemplo específico, ela representa a quantidade de transações corretamente classificadas como fraudulentas em relação à quantidade total de operações fraudulentas.
- Quanto maior a **taxa de verdadeiros positivos**, melhor.
- Analogamente, também podemos obter a chamada **taxa de falsos positivos**, sendo ela calculada pela expressão mostrada acima.
- Para este exemplo específico, ela representa a quantidade de operações incorretamente classificadas como fraudulentas em relação ao total de operações não fraudulentas.
- Quanto maior a **taxa de falsos positivos**, pior.



- O **recall** (revocação) é também chamado de **taxa de verdadeiros positivos**, ou ainda, de **sensibilidade**.
- Para este exemplo específico, ela representa a quantidade de transações corretamente classificadas como fraudulentas em relação à quantidade total de operações fraudulentas.
- Quanto maior a **taxa de verdadeiros positivos**, melhor.
- Analogamente, também podemos obter a chamada **taxa de falsos positivos**, sendo ela calculada pela expressão mostrada acima.
- Para este exemplo específico, ela representa a quantidade de operações incorretamente classificadas como fraudulentas em relação ao total de operações não fraudulentas.
- Quanto maior a **taxa de falsos positivos**, pior.
- A **taxa de falsos positivos** é também chamada de "complemento da especificidade" ($1 - \text{especificidade}$)



- 1 O **recall** (revocação) é também chamado de **taxa de verdadeiros positivos**, ou ainda, de **sensibilidade**.
- 2 Para este exemplo específico, ela representa a quantidade de transações corretamente classificadas como fraudulentas em relação à quantidade total de operações fraudulentas.
- 3 Quanto maior a **taxa de verdadeiros positivos**, melhor.
- 4 Analogamente, também podemos obter a chamada **taxa de falsos positivos**, sendo ela calculada pela expressão mostrada acima.
- 5 Para este exemplo específico, ela representa a quantidade de operações incorretamente classificadas como fraudulentas em relação ao total de operações não fraudulentas.
- 6 Quanto maior a **taxa de falsos positivos**, pior.
- 7 A **taxa de falsos positivos** é também chamada de "complemento da especificidade" ($1 - \text{especificidade}$)
- 8 Para esse caso acima, seria possível reduzir a **taxa de FP**, mantendo a **taxa de VP**? Nós veremos isso agora, estudando dois conceitos correlacionados entre si: **ROC e AUC**.

Métricas ROC e AUC

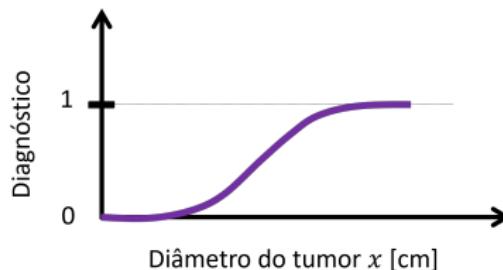
Métricas ROC e AUC

Começaremos com um exemplo mais simples, levando em conta um modelo de Regressão Logística.

- Considera que você tinha o **conjunto de dados de treinamento** ilustrado abaixo.
- Com esses dados de treinamento, um modelo de Regressão Logística foi treinado.



Como pode ser visto, o modelo resultante é conforme abaixo, e pode ser usado para classificar **novas amostras**



continua no próximo slide...

Suponha que você submeteu o modelo resultante a um novo conjunto de dados (dados de validação), utilizando um valor de limiar de 0.5 para realização da previsão.



Suponha que você submeteu o modelo resultante a um novo conjunto de dados (dados de validação), utilizando um valor de limiar de 0.5 para realização da previsão.



Analisando a figura, note que é possível construir a seguinte matriz de confusão resultante:

		CLASSE VERDADEIRA	
		1	0
CLASSE PREVISTA	1	VERDADEIROS POSITIVOS 3	FALSOS POSITIVOS 1
	0	FALSOS NEGATIVOS 1	VERDADEIROS NEGATIVOS 3

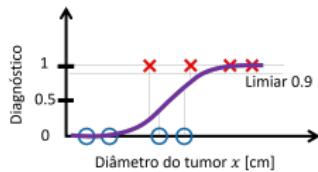
$$\text{Taxa de } VP = \frac{VP}{VP+FN} = \frac{3}{3+1} = 75\%$$

(= recall = sensibilidade)

$$\text{Taxa de } FP = \frac{FP}{FP+VN} = \frac{1}{1+3} = 25\%$$

(ou ainda, 1 - especificidade)

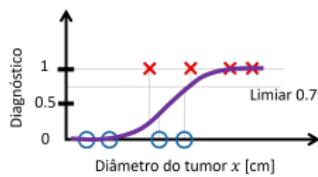
Para o mesmo conjunto de dados, note que, ao variar o valor de limiar, diferentes matrizes de confusão são obtidas.



		CLASSE VERDADEIRA	
		1	0
CLASSE PREVISTA	1	VERDADEIROS POSITIVOS 2	FALSOS POSITIVOS 0
	0	FALSOS NEGATIVOS 2	VERDADEIROS NEGATIVOS 4

Taxa de $VP = \frac{VP}{VP+FN} = \frac{2}{2+2} = 50\%$

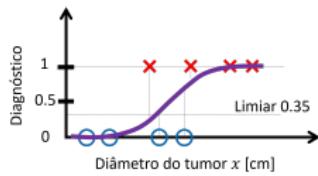
Taxa de $FP = \frac{FP}{FP+VN} = \frac{0}{0+4} = 0\%$



		CLASSE VERDADEIRA	
		1	0
CLASSE PREVISTA	1	VERDADEIROS POSITIVOS 3	FALSOS POSITIVOS 0
	0	FALSOS NEGATIVOS 1	VERDADEIROS NEGATIVOS 4

Taxa de $VP = \frac{VP}{VP+FN} = \frac{3}{3+1} = 75\%$

Taxa de $FP = \frac{FP}{FP+VN} = \frac{0}{0+4} = 0\%$



		CLASSE VERDADEIRA	
		1	0
CLASSE PREVISTA	1	VERDADEIROS POSITIVOS 3	FALSOS POSITIVOS 2
	0	FALSOS NEGATIVOS 1	VERDADEIROS NEGATIVOS 2

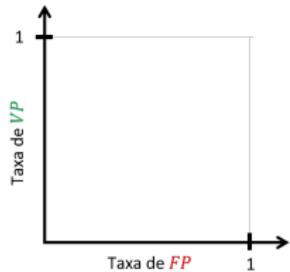
Taxa de $VP = \frac{VP}{VP+FN} = \frac{3}{3+1} = 75\%$

Taxa de $FP = \frac{FP}{FP+VN} = \frac{2}{2+2} = 50\%$

- Observe que, à medida que o valor de limiar diminui, a **taxa de verdadeiros positivos** e a **taxa de falsos positivos** tendem a aumentar.

- Usando vários valores de limiar, podemos construir uma tabela com 3 colunas: valor de limiar, taxa de VP, taxa de FP
- A partir desta tabela, podemos construir um gráfico que relaciona a **taxa de verdadeiros positivos** com a **taxa de falsos positivos** para diferentes **valores de limiar**

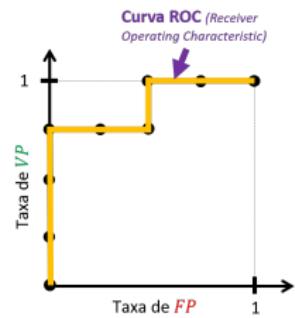
Valor de limiar	Taxa de <i>VP</i>	Taxa de <i>FP</i>
0	100%	100%
0,05	100%	75%
0,2	100%	50%
0,35	75%	50%
0,5	75%	25%
0,75	75%	0%
0,9	50%	0%
0,95	25%	0%
0,95	0%	0%



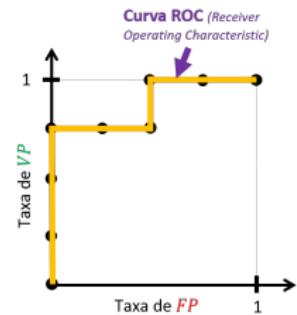
Valor de limiar	Taxa de <i>VP</i>	Taxa de <i>FP</i>
0	100%	100%
0,05	100%	75%
0,2	100%	50%
0,35	75%	50%
0,5	75%	25%
0,75	75%	0%
0,9	50%	0%
0,95	25%	0%
0,95	0%	0%



Valor de limiar	Taxa de <i>VP</i>	Taxa de <i>FP</i>
0	100%	100%
0,05	100%	75%
0,2	100%	50%
0,35	75%	50%
0,5	75%	25%
0,75	75%	0%
0,9	50%	0%
0,95	25%	0%
0,95	0%	0%

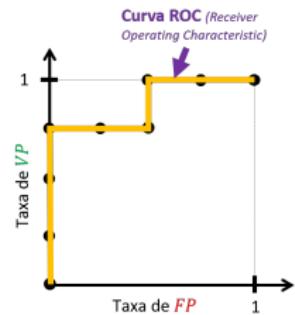


Valor de limiar	Taxa de <i>VP</i>	Taxa de <i>FP</i>
0	100%	100%
0,05	100%	75%
0,2	100%	50%
0,35	75%	50%
0,5	75%	25%
0,75	75%	0%
0,9	50%	0%
0,95	25%	0%
0,95	0%	0%



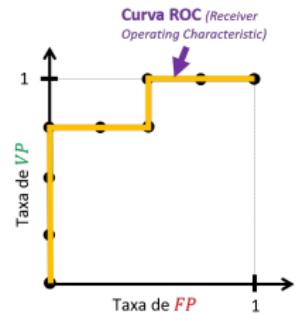
- A curva produzida é chamada de Curva ROC

Valor de limiar	Taxa de <i>VP</i>	Taxa de <i>FP</i>
0	100%	100%
0,05	100%	75%
0,2	100%	50%
0,35	75%	50%
0,5	75%	25%
0,75	75%	0%
0,9	50%	0%
0,95	25%	0%
0,95	0%	0%



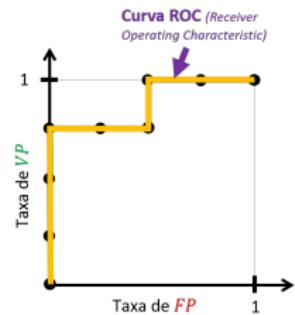
- A curva produzida é chamada de Curva ROC
- Pois bem, olhando para essa figura, qual é o melhor valor de limiar possível?

Valor de limiar	Taxa de <i>VP</i>	Taxa de <i>FP</i>
0	100%	100%
0,05	100%	75%
0,2	100%	50%
0,35	75%	50%
0,5	75%	25%
0,75	75%	0%
0,9	50%	0%
0,95	25%	0%
0,95	0%	0%



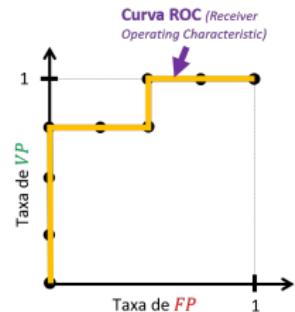
- A curva produzida é chamada de Curva ROC
- Pois bem, olhando para essa figura, qual é o melhor valor de limiar possível?
- Note que usar 0.2 é melhor do que usar 0 ou 0.05

Valor de limiar	Taxa de <i>VP</i>	Taxa de <i>FP</i>
0	100%	100%
0,05	100%	75%
0,2	100%	50%
0,35	75%	50%
0,5	75%	25%
0,75	75%	0%
0,9	50%	0%
0,95	25%	0%
0,95	0%	0%



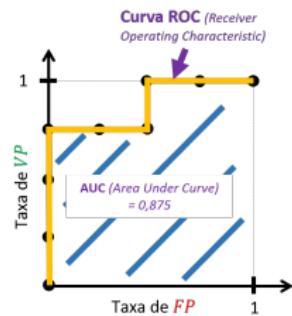
- A curva produzida é chamada de Curva ROC
- Pois bem, olhando para essa figura, qual é o melhor valor de limiar possível?
- Note que usar 0.2 é melhor do que usar 0 ou 0.05
- Note que usar 0.75 é melhor do que usar 0.5 ou 0.35

Valor de limiar	Taxa de <i>VP</i>	Taxa de <i>FP</i>
0	100%	100%
0,05	100%	75%
0,2	100%	50%
0,35	75%	50%
0,5	75%	25%
0,75	75%	0%
0,9	50%	0%
0,95	25%	0%
0,95	0%	0%



- A curva produzida é chamada de Curva ROC
- Pois bem, olhando para essa figura, qual é o melhor valor de limiar possível?
- Note que usar 0.2 é melhor do que usar 0 ou 0.05
- Note que usar 0.75 é melhor do que usar 0.5 ou 0.35
- Agora, entre 0.2 e 0.75, depende do que você quer. Você aceita aumentar sua taxa de VP mesmo que isso signifique também aumentar sua taxa de FP?

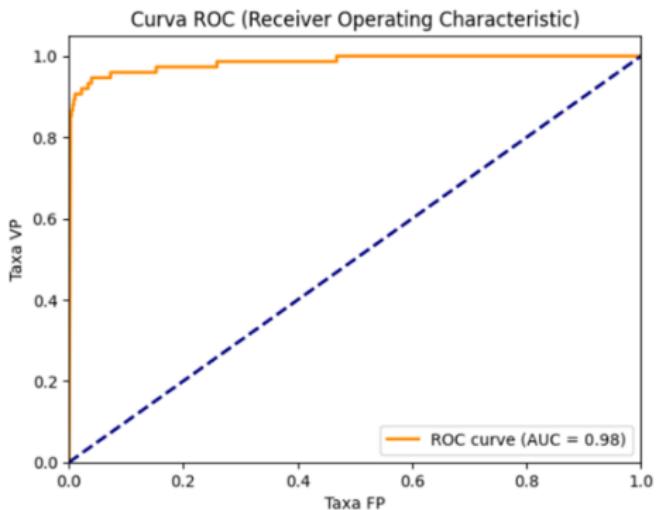
Valor de limiar	Taxa de <i>VP</i>	Taxa de <i>FP</i>
0	100%	100%
0,05	100%	75%
0,2	100%	50%
0,35	75%	50%
0,5	75%	25%
0,75	75%	0%
0,9	50%	0%
0,95	25%	0%
0,95	0%	0%



- A métrica dada pela área sob a curva ROC é chamada de AUC.
- Nesse caso, os valores foram calculados a partir de um modelo de regressão logística. Vamos supor que nós apliquemos também redes neurais ao problema, e que ela nos retorne uma AUC de 0.9. A ideia é dizer que a rede neural performou melhor que a regressão logística.

Voltando ao exemplo de transações financeiras fraudulentas...

	Valor limiar	Taxa de VP	tакса de FP
0	0.00	1.000000	1.000000
1	0.02	0.946667	0.040748
2	0.04	0.933333	0.034525
3	0.06	0.920000	0.031080
4	0.08	0.920000	0.028197
5	0.10	0.920000	0.026070
6	0.12	0.920000	0.024523
7	0.14	0.920000	0.023398
8	0.16	0.920000	0.022361
9	0.18	0.906667	0.021183
10	0.20	0.906667	0.020339
11	0.22	0.906667	0.019759
12	0.24	0.906667	0.019126
13	0.26	0.906667	0.018458
14	0.28	0.906667	0.018054
15	0.30	0.906667	0.017526
16	0.32	0.906667	0.017175
17	0.34	0.906667	0.016753
18	0.36	0.906667	0.016366
19	0.38	0.906667	0.016243
20	0.40	0.906667	0.016014
21	0.42	0.906667	0.015698
22	0.44	0.906667	0.015188
23	0.46	0.906667	0.014854
24	0.48	0.906667	0.014468
25	0.50	0.906667	0.014257
26	0.52	0.906667	0.013975
27	0.54	0.906667	0.013712
28	0.56	0.906667	0.013466
29	0.58	0.906667	0.013290
30	0.60	0.906667	0.013096
31	0.62	0.906667	0.012780
32	0.64	0.906667	0.012516
33	0.66	0.906667	0.012323
34	0.68	0.906667	0.012182
35	0.70	0.906667	0.011989
36	0.72	0.906667	0.011813
37	0.74	0.906667	0.011655
38	0.76	0.906667	0.011532
39	0.78	0.906667	0.011426
40	0.80	0.906667	0.011198
41	0.82	0.906667	0.010969
42	0.84	0.893333	0.010776
43	0.86	0.893333	0.010424
44	0.88	0.893333	0.010126
45	0.90	0.893333	0.009879
46	0.92	0.893333	0.009633
47	0.94	0.893333	0.009317
48	0.96	0.893333	0.008772
49	0.98	0.880000	0.008016
50	1.00	0.853333	0.002953



Observações finais

Observações finais

- As métricas ROC e AUC são muito utilizadas por diversos usuários de ML. Entretanto, pessoalmente falando, eu prefiro usar diretamente a tabela para escolher o valor de limiar.

Observações finais

- As métricas ROC e AUC são muito utilizadas por diversos usuários de ML. Entretanto, pessoalmente falando, eu prefiro usar diretamente a tabela para escolher o valor de limiar.
- Note que as taxas de VP e FP (em função de diferentes valores de limiar) tendo-se como base um único modelo.

Observações finais

- As métricas ROC e AUC são muito utilizadas por diversos usuários de ML. Entretanto, pessoalmente falando, eu prefiro usar diretamente a tabela para escolher o valor de limiar.
- Note que as taxas de VP e FP (em função de diferentes valores de limiar) tendo-se como base um único modelo.
- O ideal é calcular tais métricas com base nos dados de validação, para que as conclusões são mais representativas.

Observações finais

- As métricas ROC e AUC são muito utilizadas por diversos usuários de ML. Entretanto, pessoalmente falando, eu prefiro usar diretamente a tabela para escolher o valor de limiar.
- Note que as taxas de VP e FP (em função de diferentes valores de limiar) tendo-se como base um único modelo.
- O ideal é calcular tais métricas com base nos dados de validação, para que as conclusões são mais representativas.
- Ao invés de usar as taxas de VP e FP para compor os eixos y e x, respectivamente, muitas pessoas plotam, por exemplo, precisão x recall. Nesse caso, muda o ponto do gráfico que deve ser considerado o “melhor dos mundos”

De olho no código!

Vamos agora voltar ao exemplo de transações fraudulentas estudado na aula anterior, e implementaremos o cálculo das métricas ROC e AUC.

Acesse o Python Notebook usando o QR code ou o link abaixo:



https://colab.research.google.com/github/xaximpv2/master/blob/main/codigo_aula23_classificacao_desbalanceada_ROC_AUC.ipynb

- Link para acesso à base de dados: <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>
- Apenas uma observação adicional: Também existe um OUTRO exemplo no site do Keras que envolve essa questão de transações financeiras fraudulentas. Ele pode ser encontrado nesse site qui: https://keras.io/examples/timeseries/event_classification_for_payment_card_fraud_detection/

Parte 1

Rode todo o código. Responda às questões nele contidas e complete-o, se necessário.

Parte 2

- 1 Fazendo uma análise cuidadosa de como diferentes valores de limiar afetam as taxas de VP e FP resultantes para os dados de validação deste problema, escolha um valor de limiar adequado. Justifique e detalhe sua escolha final.

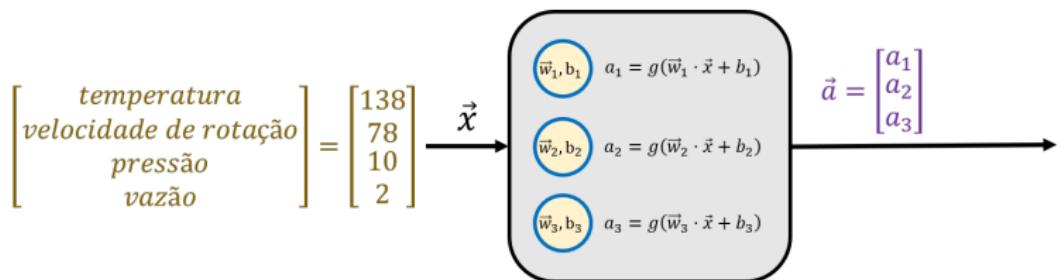
Camadas convolucionais e *Dropout*

Esta aula constitui um tópico adicional da disciplina. Trata-se de um conteúdo opcional. Sua atividade não valerá nota e não precisa ser enviada.



Camadas convolucionais

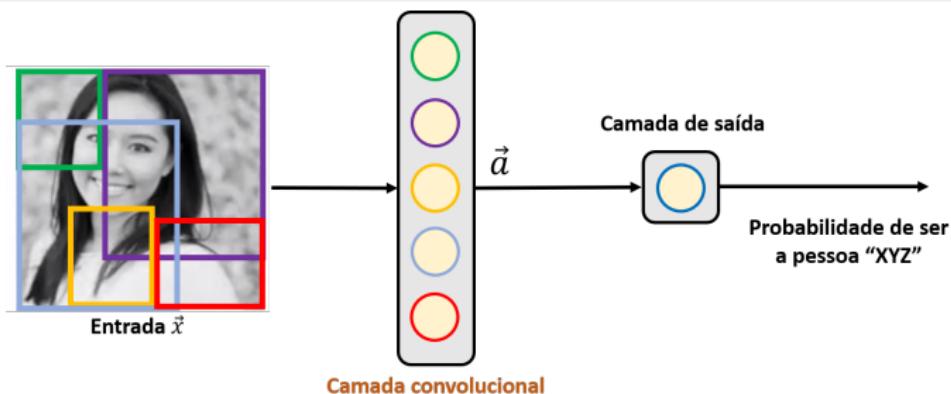
- Conforme ilustrado abaixo, em camadas do tipo **Dense**, a saída de cada neurônio é uma função de **todas as ativações** de saída vindas da camada anterior.
- No exemplo, por se tratar da Camada 1, note que as ativações de saída vindas da camada anterior são o próprio $\vec{x} = \vec{a}^{[0]}$ (características de entrada).



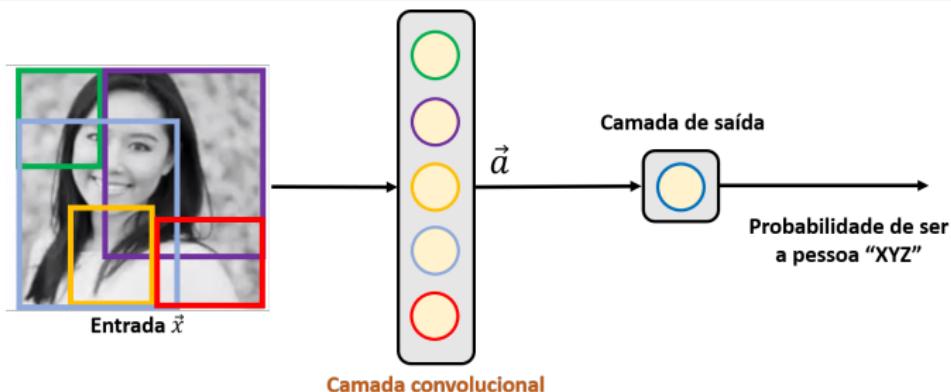
A seguir, veremos um exemplo de um outro tipo de camada que é possível de ser utilizada numa rede neural

Camadas convolucionais

- Conforme ilustrado abaixo, em camadas do tipo **Convolucionais**, a saída de cada neurônio é uma função apenas de **parte das ativações** de saída vindas da camada anterior.



- Conforme ilustrado abaixo, em camadas do tipo **Convolucionais**, a saída de cada neurônio é uma função apenas de **parte das ativações** de saída vindas da camada anterior.



Qual é a vantagem de fazermos isso?

- O processamento torna-se mais rápido.
- Menos dados de treinamento são necessários, já que menos parâmetros estarão presentes no modelo (tendência menor de ocorrer *overfitting*)

Observação

- Se a sua rede neural possui muitas camadas convolucionais, então ela pode também ser chamada de **Rede neural convolucional**

Desvantagens de usar redes neurais clássicas para classificação de imagens

- Requer uma quantidade muito grande de cálculos
- Pixels locais são tratados da mesma maneira que pixels distantes
- sensível à posição do objeto na imagem

Pergunta:

Como podemos detectar pequenos elementos numa imagem?

Resposta:

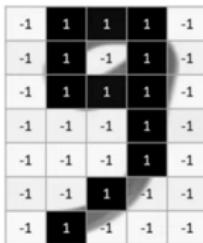
Simples! Usando filtros.

Exemplo:

Suponha que você quer encontrar, numa imagem, onde ela possui

- uma circunferência
- uma linha vertical
- uma linha diagonal

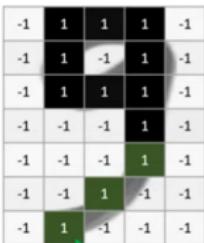
Nesse caso, os três filtros abaixo podem ser usados.



Filtro que detecta uma circunferência



Filtro que detecta uma linha vertical



Filtro que detecta uma linha diagonal

Aplicando o filtro que detecta circunferência:

-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1

Filtro que detecta uma circunferência

*
(conv.)

1	1	1
1	-1	1
1	1	1

=

-0.11	1	-0.11
-0.55	0.11	-0.33
-0.33	0.33	-0.33
-0.22	-0.11	-0.22
-0.33	-0.33	-0.33

Mapa de características: Nova imagem que destaca a posição onde está presente a circunferência

Observações

- Note que o filtro acima foi aplicado à imagem original usando um passo de 1 unidade.
- É possível usar um passo maior. Isso reduz ainda mais a dimensionalidade da imagem resultante.

Outros filtros interessantes para identificar um 9 numa imagem:

Filtro que detecta uma circunferência

$$\begin{matrix} 9 & * & \begin{matrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 1 \end{matrix} & = & \begin{matrix} & & & & \\ & & 1 & & \\ & & & & \\ & & & & \\ & & & & \end{matrix} \end{matrix}$$

Filtro que detecta uma linha vertical

$$\begin{matrix} 9 & * & \begin{matrix} -1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & -1 \end{matrix} & = & \begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & 1 & & \end{matrix} \end{matrix}$$

Filtro que detecta uma linha diagonal

$$\begin{matrix} 9 & * & \begin{matrix} -1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \end{matrix} & = & \begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & 1 & & \end{matrix} \end{matrix}$$

Filtro detector de olhos:



$$* \begin{array}{|c|c|c|} \hline \text{eye} \\ \hline \text{detector} \\ \hline \end{array} = \boxed{\begin{array}{|c|c|} \hline \textcolor{green}{\blacksquare} & \textcolor{green}{\blacksquare} \\ \hline \end{array}}$$

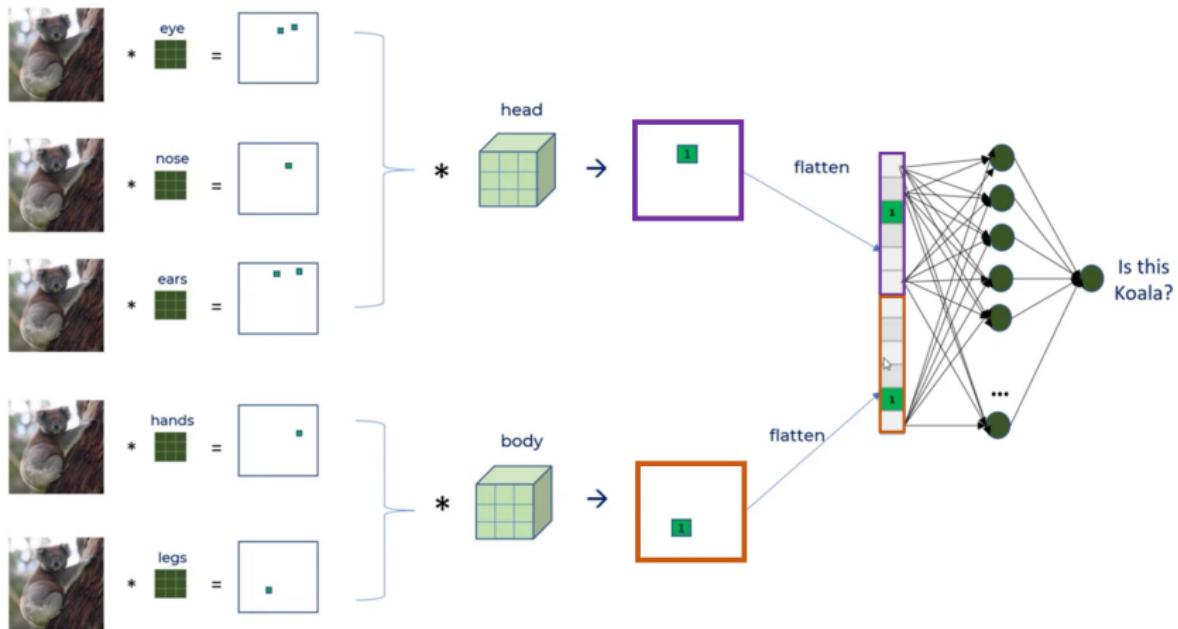


$$\begin{array}{|c|c|c|} \hline \text{eye} \\ \hline \text{detector} \\ \hline \end{array} = \boxed{\begin{array}{|c|c|} \hline \textcolor{green}{\blacksquare} & \textcolor{green}{\blacksquare} \\ \hline \end{array}}$$



$$* \begin{array}{|c|c|c|} \hline \text{eye} \\ \hline \text{detector} \\ \hline \end{array} = \boxed{\begin{array}{|c|c|c|c|} \hline \textcolor{green}{\blacksquare} & & \textcolor{green}{\blacksquare} & \textcolor{green}{\blacksquare} \\ \hline & & \textcolor{green}{\blacksquare} & \textcolor{green}{\blacksquare} \\ \hline \end{array}}$$

Note que o filtro funciona independentemente da posição ou quantidade de olhos na imagem. Essa é uma das grandes vantagens das camadas convolucionais.



Observações:

- Note que a rede acima aplica dois filtros 3D para localizar a cabeça e o corpo na imagem.
- Esses dois resultados são então “desenrolados” e colocados como entrada de uma rede neural clássica (com camadas do tipo Dense)

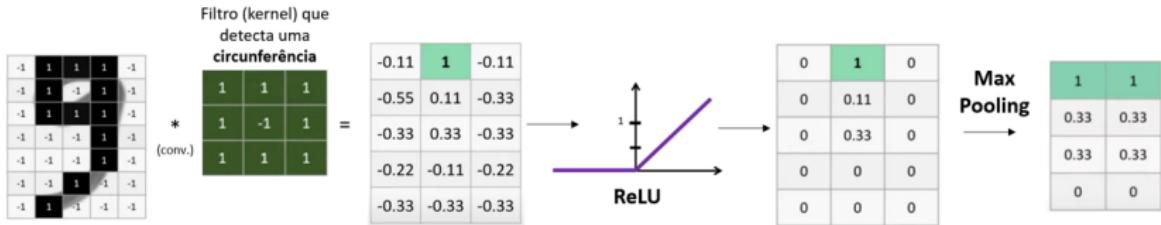
- Nas redes neurais clássicas, usamos funções de ativação para adicionar não linearidades matemáticas.
- Nas camadas convolucionais, também podemos fazer isso. Por exemplo, usando a função ReLU ao mapa de características obtido no slides anteriores, chegamos em:



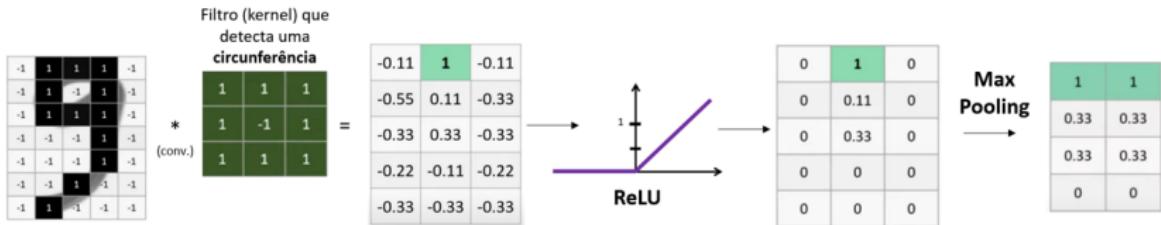
Mais observações:

- Caso você queira que o mapa de características tenha a mesma dimensão que a figura original, você pode fazer "preenchimento forçado" (*padding*). Para o exemplo acima, poderia, por exemplo, adicionar zeros em volta.
- Entretanto, muitas vezes queremos justamente reduzir ainda mais as dimensões do nosso problema, para que haja também redução de custo computacional. Nesse contexto, existe o chamado *pooling*.

Exemplo de filtro de *Max Pooling* 2 por 2 com passo 1:



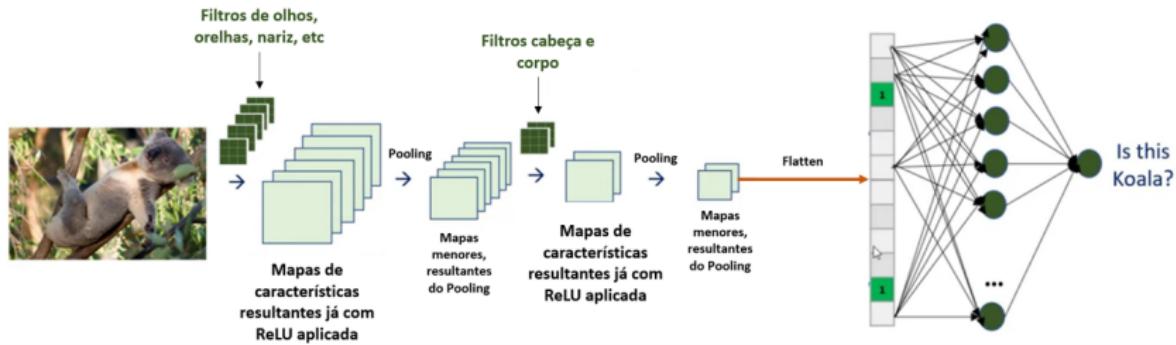
Exemplo de filtro de *Max Pooling* 2 por 2 com passo 1:



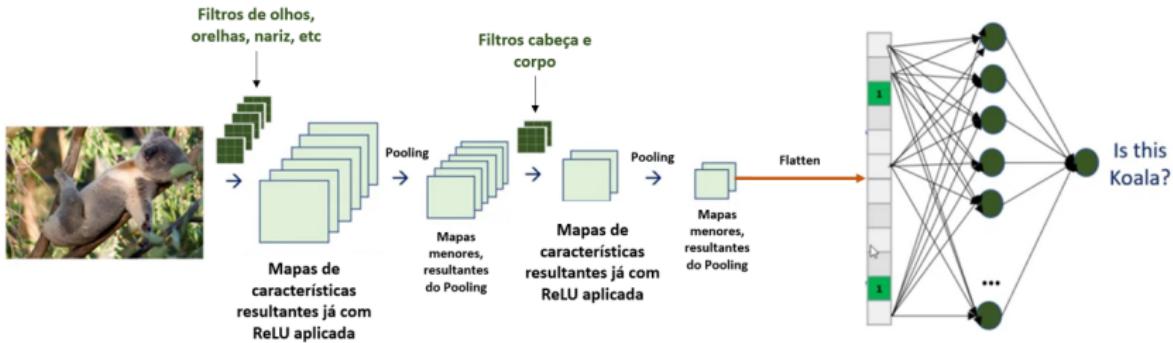
Observações

- Ao invés de *Max Pooling*, poderíamos também usar *average Pooling*.
- Redução de dimensionalidade vem acompanhada de redução de parâmetros a serem estimados.

Rede convolucional completa:



Rede convolucional completa:



Observações:

- Os valores dos filtros acima são parâmetros que devem ser treinados.
- São hiperparâmetros: O tamanho dos filtros, o passo usado para movê-los quando do cálculo da convolução, o tipo de Pooling (Max ou average), e as funções de ativação.
- Para mais detalhes sobre redes convolucionais, assistir <https://www.youtube.com/watch?v=zfiSAzpy9NM&t=1289s>

Dropout

O **Dropout** é uma técnica usada no treinamento de redes neurais para reduzir o risco de overfitting, que é quando o modelo aprende demais com os dados de treinamento e não generaliza bem para novos dados.

Passo-a-passo para implementar o Dropout:

O **Dropout** é uma técnica usada no treinamento de redes neurais para reduzir o risco de overfitting, que é quando o modelo aprende demais com os dados de treinamento e não generaliza bem para novos dados.

Passo-a-passo para implementar o Dropout:

- 1 Em cada iteração, alguns neurônios da rede são desativados aleatoriamente (ou seja, seus valores de saída são zerados).

O **Dropout** é uma técnica usada no treinamento de redes neurais para reduzir o risco de overfitting, que é quando o modelo aprende demais com os dados de treinamento e não generaliza bem para novos dados.

Passo-a-passo para implementar o Dropout:

- 1 Em cada iteração, alguns neurônios da rede são desativados aleatoriamente (ou seja, seus valores de saída são zerados).
- 2 Isso força a rede a não depender demais de neurônios específicos, incentivando os neurônios restantes a trabalhar de forma mais independente e colaborativa.

O **Dropout** é uma técnica usada no treinamento de redes neurais para reduzir o risco de overfitting, que é quando o modelo aprende demais com os dados de treinamento e não generaliza bem para novos dados.

Passo-a-passo para implementar o Dropout:

- 1 Em cada iteração, alguns neurônios da rede são desativados aleatoriamente (ou seja, seus valores de saída são zerados).
- 2 Isso força a rede a não depender demais de neurônios específicos, incentivando os neurônios restantes a trabalhar de forma mais independente e colaborativa.
- 3 No final, todos os neurônios são usados durante a inferência (quando o modelo faz previsões), mas suas conexões são ajustadas para refletir a média de quando estavam "ligados" ou "desligados" no treinamento.

O **Dropout** é uma técnica usada no treinamento de redes neurais para reduzir o risco de overfitting, que é quando o modelo aprende demais com os dados de treinamento e não generaliza bem para novos dados.

Passo-a-passo para implementar o Dropout:

- 1 Em cada iteração, alguns neurônios da rede são desativados aleatoriamente (ou seja, seus valores de saída são zerados).
- 2 Isso força a rede a não depender demais de neurônios específicos, incentivando os neurônios restantes a trabalhar de forma mais independente e colaborativa.
- 3 No final, todos os neurônios são usados durante a inferência (quando o modelo faz previsões), mas suas conexões são ajustadas para refletir a média de quando estavam "ligados" ou "desligados" no treinamento.

Observações finais

- A rede aprende representações mais gerais e confiáveis.

O **Dropout** é uma técnica usada no treinamento de redes neurais para reduzir o risco de overfitting, que é quando o modelo aprende demais com os dados de treinamento e não generaliza bem para novos dados.

Passo-a-passo para implementar o Dropout:

- 1 Em cada iteração, alguns neurônios da rede são desativados aleatoriamente (ou seja, seus valores de saída são zerados).
- 2 Isso força a rede a não depender demais de neurônios específicos, incentivando os neurônios restantes a trabalhar de forma mais independente e colaborativa.
- 3 No final, todos os neurônios são usados durante a inferência (quando o modelo faz previsões), mas suas conexões são ajustadas para refletir a média de quando estavam "ligados" ou "desligados" no treinamento.

Observações finais

- A rede aprende representações mais gerais e confiáveis.
- É controlado por um parâmetro chamado taxa de dropout, que indica a proporção de neurônios a serem desativados (exemplo: 0.2 significa que 20% dos neurônios serão desligados a cada iteração).

O **Dropout** é uma técnica usada no treinamento de redes neurais para reduzir o risco de overfitting, que é quando o modelo aprende demais com os dados de treinamento e não generaliza bem para novos dados.

Passo-a-passo para implementar o Dropout:

- 1 Em cada iteração, alguns neurônios da rede são desativados aleatoriamente (ou seja, seus valores de saída são zerados).
- 2 Isso força a rede a não depender demais de neurônios específicos, incentivando os neurônios restantes a trabalhar de forma mais independente e colaborativa.
- 3 No final, todos os neurônios são usados durante a inferência (quando o modelo faz previsões), mas suas conexões são ajustadas para refletir a média de quando estavam "ligados" ou "desligados" no treinamento.

Observações finais

- A rede aprende representações mais gerais e confiáveis.
- É controlado por um parâmetro chamado taxa de dropout, que indica a proporção de neurônios a serem desativados (exemplo: 0.2 significa que 20% dos neurônios serão desligados a cada iteração).

Resumo:

- É como se você fosse o treinador de um time de futebol, e você vai tirando jogadores do seu time aleatoriamente para que eles aprendam a se adaptar quando da falta de alguns jogadores específicos.

O **Dropout** é uma técnica usada no treinamento de redes neurais para reduzir o risco de overfitting, que é quando o modelo aprende demais com os dados de treinamento e não generaliza bem para novos dados.

Passo-a-passo para implementar o Dropout:

- 1 Em cada iteração, alguns neurônios da rede são desativados aleatoriamente (ou seja, seus valores de saída são zerados).
- 2 Isso força a rede a não depender demais de neurônios específicos, incentivando os neurônios restantes a trabalhar de forma mais independente e colaborativa.
- 3 No final, todos os neurônios são usados durante a inferência (quando o modelo faz previsões), mas suas conexões são ajustadas para refletir a média de quando estavam "ligados" ou "desligados" no treinamento.

Observações finais

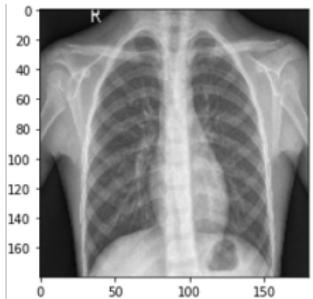
- A rede aprende representações mais gerais e confiáveis.
- É controlado por um parâmetro chamado taxa de dropout, que indica a proporção de neurônios a serem desativados (exemplo: 0.2 significa que 20% dos neurônios serão desligados a cada iteração).

Resumo:

- É como se você fosse o treinador de um time de futebol, e você vai tirando jogadores do seu time aleatoriamente para que eles aprendam a se adaptar quando da falta de alguns jogadores específicos.
- Assim, fazemos com que "a equipe trabalhe com membros diferentes em cada treino", garantindo que o aprendizado não dependa de poucos indivíduos específicos, mas sim do esforço coletivo.

Um exemplo complexo

Um exemplo complexo



→ Probabilidade de ser pneumonia

```
def conv_block(filters, inputs):
    x = layers.SeparableConv2D(filters, 3, activation="relu", padding="same")(inputs)
    x = layers.SeparableConv2D(filters, 3, activation="relu", padding="same")(x)
    x = layers.BatchNormalization()(x)
    outputs = layers.MaxPool2D()(x)

    return outputs

def dense_block(units, dropout_rate, inputs):
    x = layers.Dense(units, activation="relu")(inputs)
    x = layers.BatchNormalization()(x)
    outputs = layers.Dropout(dropout_rate)(x)

    return outputs
```

```
def build_model():
    inputs = keras.Input(shape=(IMAGE_SIZE[0], IMAGE_SIZE[1], 3))
    x = layers.Rescaling(1.0 / 255)(inputs)
    x = layers.Conv2D(3, activation="relu", padding="same")(x)
    x = layers.Conv2D(16, 3, activation="relu", padding="same")(x)
    x = layers.Conv2D(16, 3, activation="relu", padding="same")(x)
    x = layers.MaxPool2D()(x)

    x = conv_block(32, x)
    x = conv_block(64, x)

    x = conv_block(128, x)
    x = layers.Dropout(0.2)(x)

    x = conv_block(256, x)
    x = layers.Dropout(0.2)(x)

    x = layers.Flatten()(x)
    x = dense_block(512, 0.7, x)
    x = dense_block(128, 0.5, x)
    x = dense_block(64, 0.3, x)

    outputs = layers.Dense(1, activation="sigmoid")(x)

    model = keras.Model(inputs=inputs, outputs=outputs)
    return model
```

De olho no código!

De olho no código!

Vamos agora rodar esse exemplo complexo que trata da detecção de pneumonia com base no Raio X do paciente.

Acesse o código diretamente pelo site do Keras, usando o QR code ou o link abaixo:



https://keras.io/examples/vision/xray_classification_with_tpus/

Observação importante:

Note que o site do Keras possui diversos outros exemplos interessantes de aplicações envolvendo ML. São aplicações envolvendo Aprendizado Supervisionado, Não Supervisionado, Aprendizado por Reforço, etc.

Parte 1

Rode todo o código. Responda às questões nele contidas e complete-o, se necessário.

Parte 2

- Explique em detalhes a arquitetura de rede neural convolucional que está sendo utilizada. Também identifique a taxa de dropout utilizada no código.