Inferência usando Redes Neurais (*forward propagation*)

Tensorflow

Onde estamos e para onde vamos

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**)

Onde estamos e para onde vamos

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

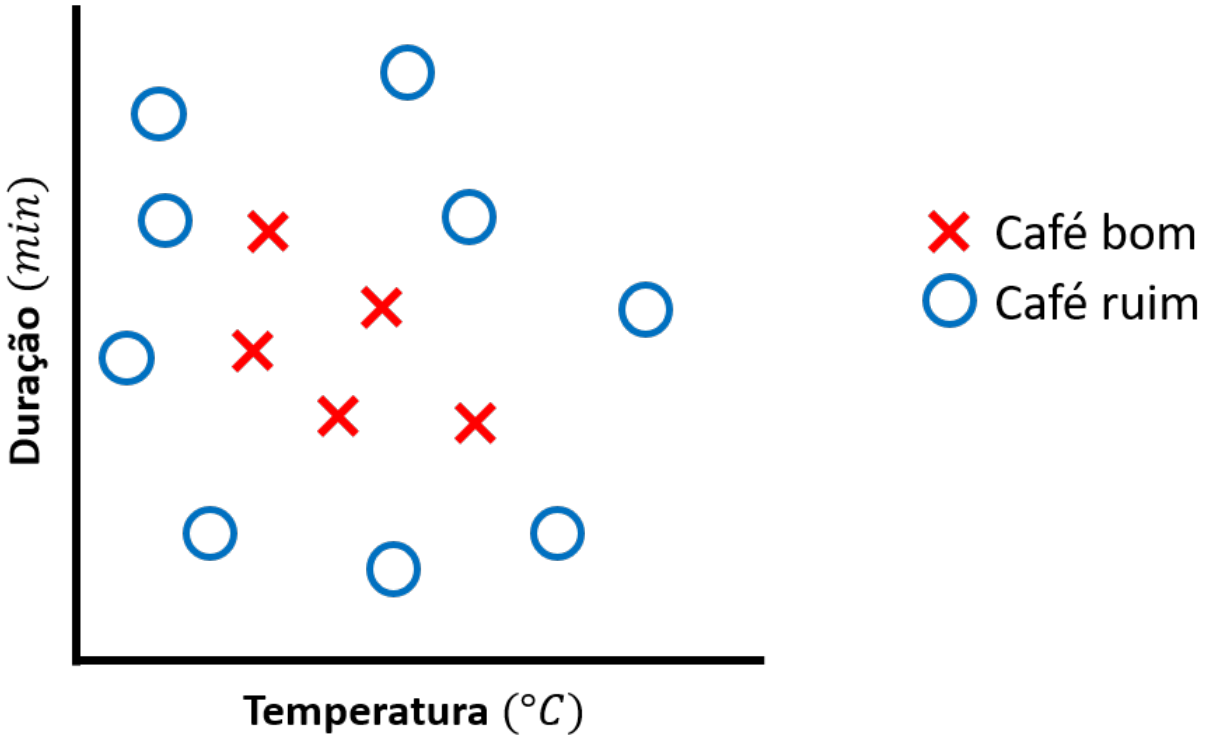
Nesta aula, iremos ver com mais detalhes como calcular o valor de saída de uma rede neural, supondo que todos os seus parâmetros já sejam conhecidos antecipadamente (problema de **inferência**)

OBS: Nesta aula, usaremos o **Tensorflow** como um facilitador para resolvermos 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**.

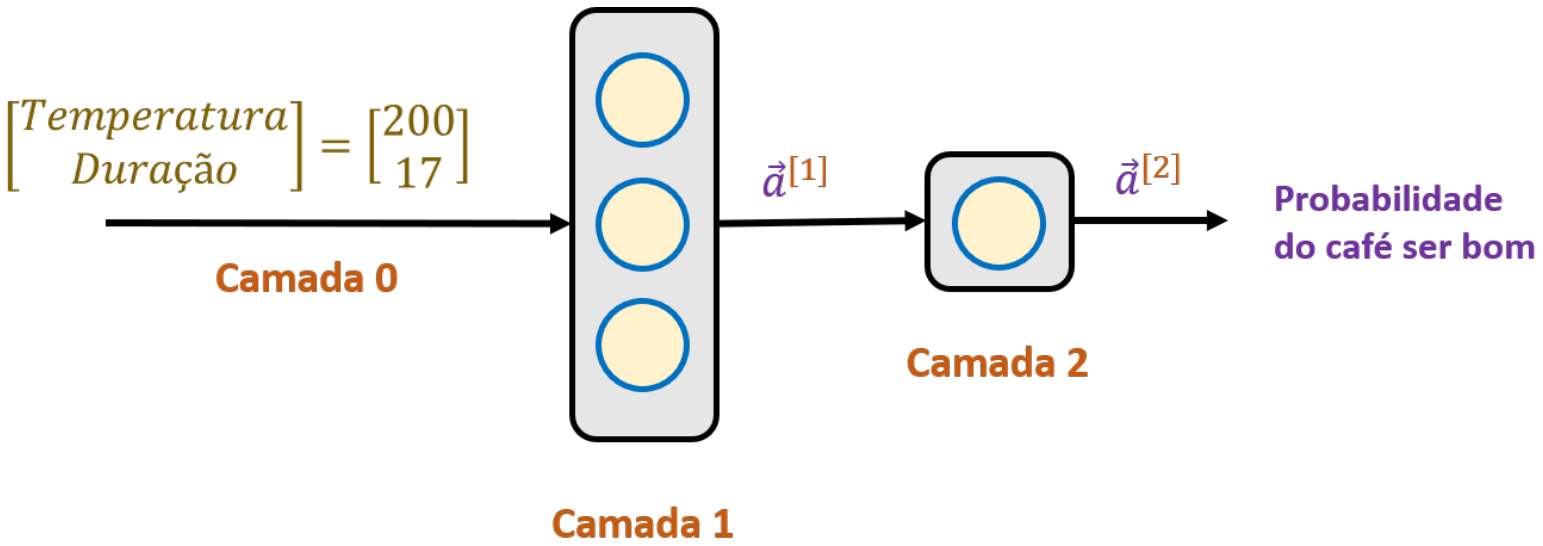
Exemplo: torrefação de café



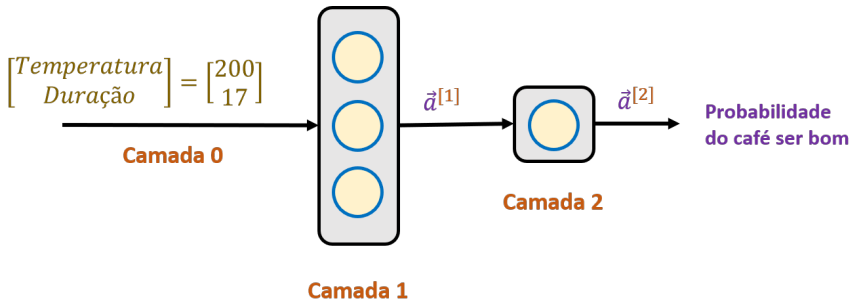
- Se fica pouco tempo torrando, o café fica sub-torrado
- Se é torrado com uma temperatura muito baixa, o café também fica sub-torrado
- Se a duração ou a temperatura forem muito altas, o café fica sobre-torrado.
- Com isso, temos um pequeno triângulo que corresponde à região onde o resultado é satisfatório.

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

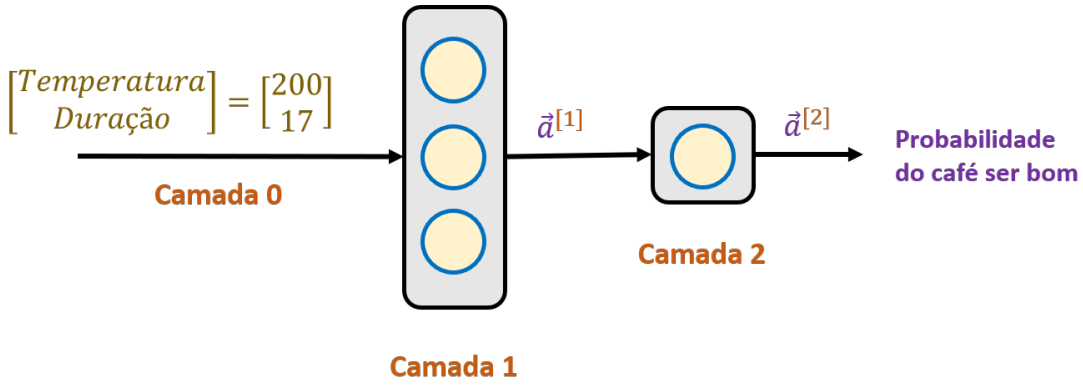
```
if a2 >= 0.5:
  yhat = 1
else:
  yhat = 0
```

NumPy versus Tensorflow

- 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)
```

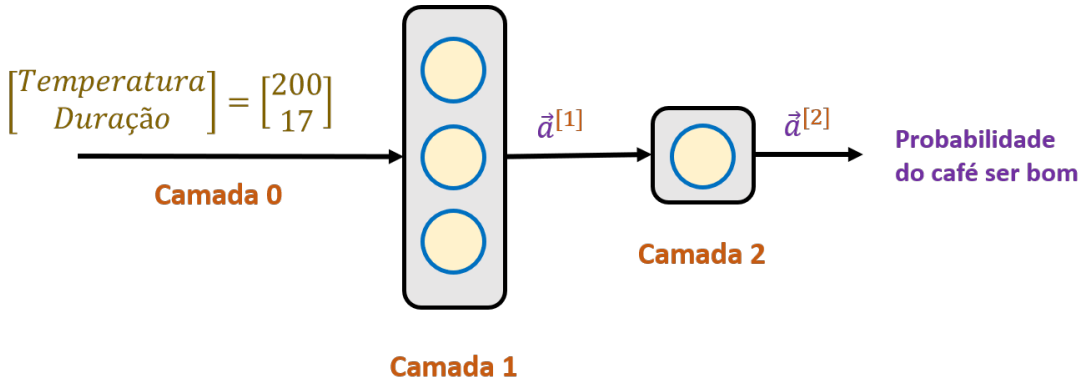
Numpy representando uma matriz (1,2)

```
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

NumPy versus Tensorflow

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 chamada arrays pelo termo **tensores**)
- Podemos converter Tensores em arrays (e vice-versa) facilmente.

Uma forma ainda mais simples de criar um modelo via Tensorflow

- 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

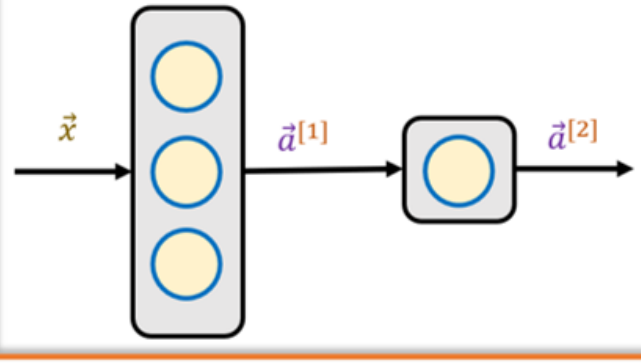
```
model = Sequential([
  Dense(units=3, activation="sigmoid"),
  Dense(units=1, activation="sigmoid")])
```

Exemplo completo: Criando e treinando o modelo

Dados

$\vec{x}$		y
200	17	1
120	5	0
425	20	0
212	18	1

Escolha da arquitetura



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

→ Veremos depois!

→ Treina todos os parâmetros!

Fazendo previsão para novo par (temperatura, duração)

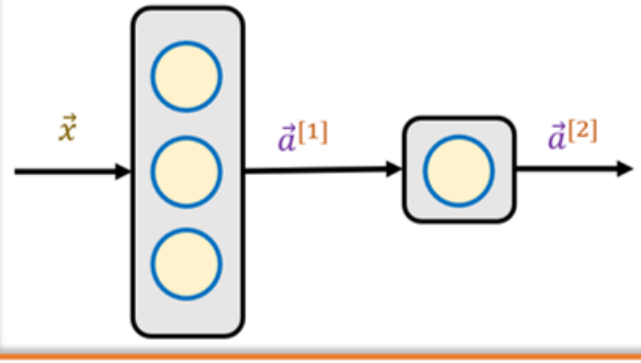
```
model.predict(x_new)
```

Exemplo completo: Criando e treinando o modelo

Dados

$\vec{x}$		y
200	17	1
120	5	0
425	20	0
212	18	1

Escolha da arquitetura



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

→ Veremos depois!

→ Treina todos os parâmetros!

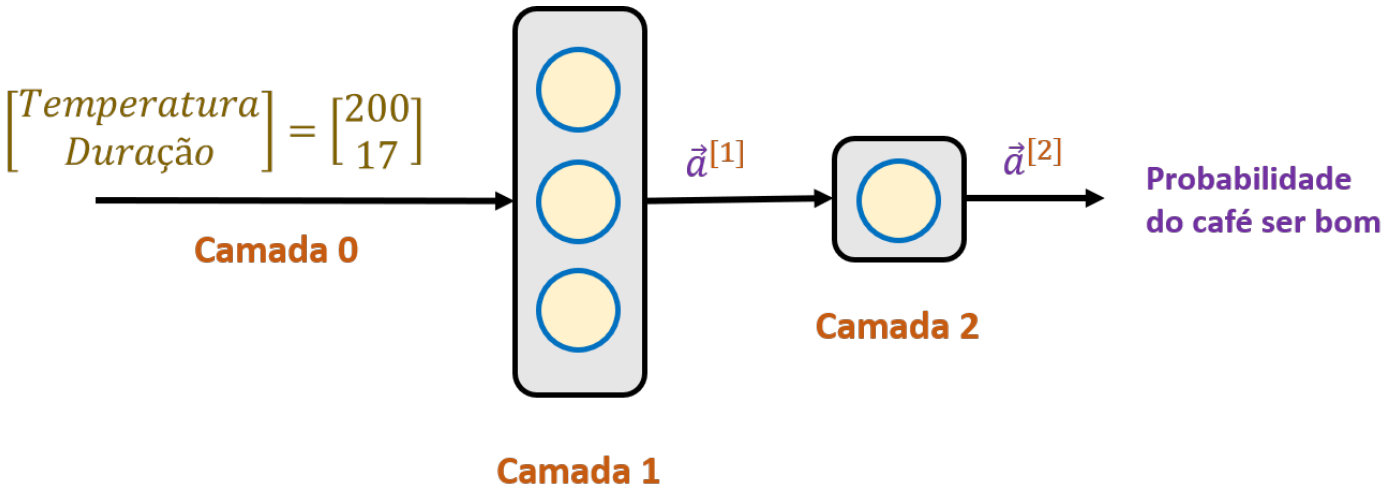
Fazendo previsão para novo par (temperatura, duração)

```
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!

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`

Atividade de aula

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