testing Tensorflow layers, computing inferences

In [1]:
```
import numpy as np
import tensorflow as tf
import pandas as pd
import matplotlib.pyplot as plt
```

In [31]:
```
x = np.array([[200, 17]])
x
```

Out[31]:
```
array([[200,  17]])
```

building a layer

In [40]:
```
layer_1 = tf.keras.layers.Dense(3)
layer_1.get_weights()
```

Out[40]:
```
[]
```

getting the output from layer (using `layer_1(x)`)

In [41]:
```
a1 = layer_1(x)
a1
```

Out[41]:
```
<tf.Tensor: shape=(1, 3), dtype=float32, numpy=array([[-158.43204 , -166.576
97 ,  -71.295975]], dtype=float32)>
```

getting weights and biases from `layer_1`

In [42]:
```
layer_1.get_weights()
```

Out[42]:
```
[array([[-0.8783076, -0.8768722, -0.4467891],
        [ 1.0134983,  0.5174984,  1.0624616]], dtype=float32),
 array([0., 0., 0.], dtype=float32)]
```

In [43]:
```
layer_1.get_weights()[0]
```

Out[43]:
```
array([[-0.8783076, -0.8768722, -0.4467891],
       [ 1.0134983,  0.5174984,  1.0624616]], dtype=float32)
```

computing the weighted sum and adding bias manually

In [44]:
```
z = np.dot(x, layer_1.get_weights()[0]) + layer_1.get_weights()[1]
```

the solution computed manually and one via `layer_1(x)` are the same

In [45]:
```
z
```

Out[45]:
```
array([[-158.43204498, -166.57696402,  -71.29596972]])
```

In [46]:
```
a1
```

Out[46]:
```
<tf.Tensor: shape=(1, 3), dtype=float32, numpy=array([[-158.43204 , -166.576
97 ,  -71.295975]], dtype=float32)>
```

testing another example

In [60]:
```
X = np.array([0., 1, 2, 3, 4, 5]).reshape(-1,1)
print(X)
print(X.shape)
```

```
[[0.]
 [1.]
 [2.]
 [3.]
 [4.]
 [5.]]
(6, 1)
```

buiding a Sequential model

In [57]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])

model.build(input_shape=X.shape)
```

In [58]:
```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_9 (Dense)             (6, 1)                    2

=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
```

getting inference from `model(X[0])`

In [91]:
```
model(X[0])
```

Out[91]:
```
<tf.Tensor: shape=(1, 1), dtype=float32, numpy=array([[0.5]], dtype=float32)
>
```

manually computing the inference

In [68]:
```
layer = model.get_layer('dense_9')
layer.get_weights()
```

Out[68]:
```
[array([[-0.02784336]], dtype=float32), array([0.], dtype=float32)]
```

In [74]:
```
z = np.dot(X[0], layer.get_weights()[0]) + layer.get_weights()[1]
z
```

Out[74]:
```
array([0.])
```

In [75]:
```
1/(1+np.exp(-z))
```

Out[75]:
```
array([0.5])
```

testing multiple layers without `Sequential()`

each row of `a1` is the output of the 5 neurons of `L1` for each row of input

(currently it is for `X[1]`)

In [84]:
```
L1 = tf.keras.layers.Dense(5, input_shape=X.shape)
a1 = L1(X[1].reshape(-1,1))
a1
```

Out[84]:
```
<tf.Tensor: shape=(1, 5), dtype=float32, numpy=
array([[-0.47178674, -0.6800461 ,  0.2084868 ,  0.6195059 , -0.8502991 ]],
      dtype=float32)>
```

`a2` denotes the output of 4 neurons of `L2`

In [85]:
```
L2 = tf.keras.layers.Dense(4)
a2 = L2(a1)
a2
```

Out[85]:
```
<tf.Tensor: shape=(1, 4), dtype=float32, numpy=
array([[ 1.0415831 , -0.15106367, -0.7438401 ,  0.01679859]],
      dtype=float32)>
```

`a3` denotes the output of 1 neuron of `L3`

In [86]:
```
L3 = tf.keras.layers.Dense(1)
a3 = L3(a2)
a3
```

Out[86]:
```
<tf.Tensor: shape=(1, 1), dtype=float32, numpy=array([[1.193719]], dtype=flo
at32)>
```

compililing the separate layers into a sequential model

In [87]:
```
model2 = tf.keras.Sequential([
    L1,L2,L3
])
```

In [88]:
```
model2.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_15 (Dense)            (None, 6, 5)              10

 dense_16 (Dense)            (None, 6, 4)              24

 dense_17 (Dense)            (None, 6, 1)              5

=================================================================
Total params: 39
Trainable params: 39
Non-trainable params: 0
_________________________________________________________________
```

In [90]:
```
model2(X[1].reshape(-1,1))
```

Out[90]:
```
<tf.Tensor: shape=(1, 1, 1), dtype=float32, numpy=array([[[1.193719]]], dtype=float32)>
```

In [ ]: