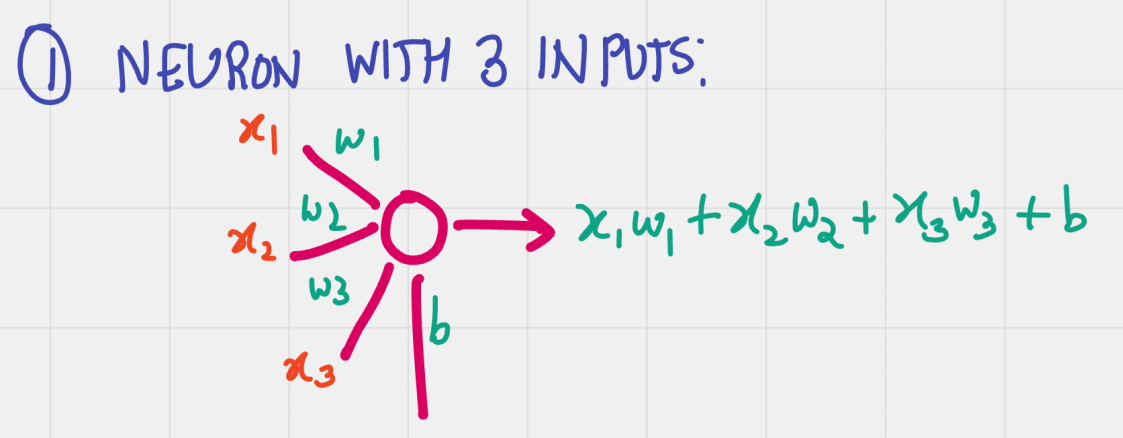
handout

October 14, 2024

0.1 BUILDING NEURAL NETWORKS FROM SCRATCH PART 1: CODING NEURONS AND LAYERS

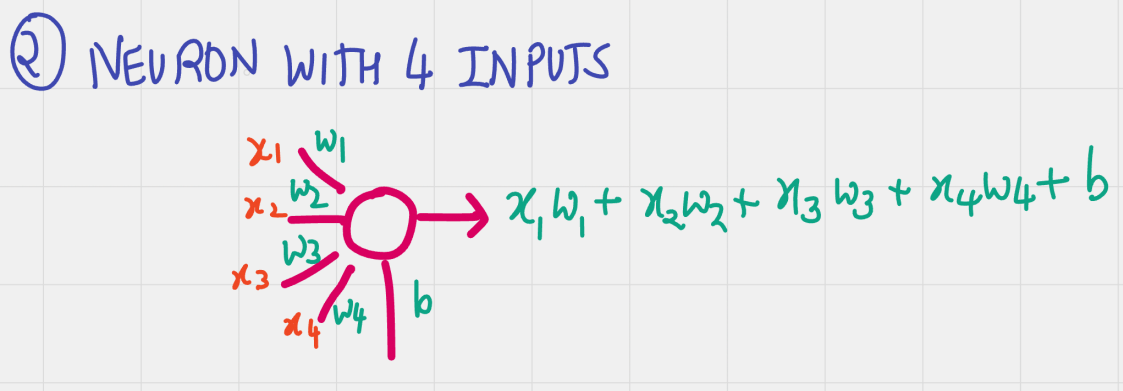
CODING OUR FIRST NEURON: 3 INPUTS



[46]:
```
inputs = [1, 2, 3]
weights = [0.2, 0.8, -0.5]
bias = 2

outputs = (inputs[0]*weights[0] + inputs[1]*weights[1] + inputs[2]*weights[2] +␣
 ↪bias)

print(outputs)
```

```
2.3
```

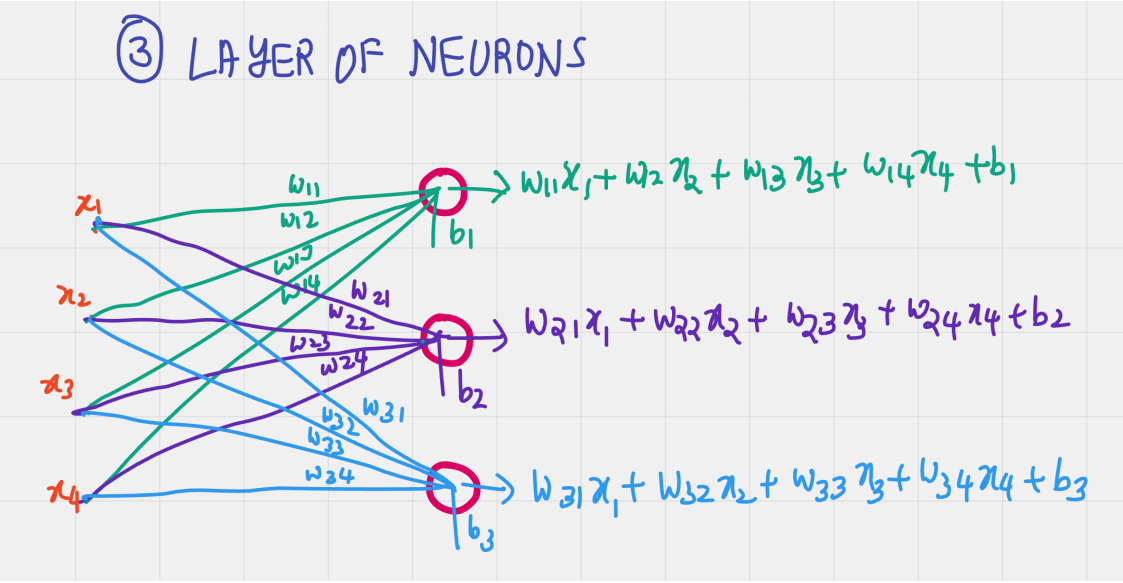
CODING OUR SECOND NEURON: 4 INPUTS



[47]:
```
inputs = [1.0, 2.0, 3.0, 2.5]
weights = [0.2, 0.8, -0.5, 1.0]
bias = 2.0
output = (inputs[0]*weights[0] +
 inputs[1]*weights[1] +
 inputs[2]*weights[2] +
 inputs[3]*weights[3] + bias)

print(output)
```

4.8

CODING OUR FIRST LAYER



[1]:
```
inputs = [1, 2, 3, 2.5]

weights = [[0.2, 0.8, -0.5, 1],
 [0.5, -0.91, 0.26, -0.5],
```

```
 [-0.26, -0.27, 0.17, 0.87]]

weights1 = weights[0] #LIST OF WEIGHTS ASSOCIATED WITH 1ST NEURON : W11, W12,␣
 ↪W13, W14
weights2 = weights[1] #LIST OF WEIGHTS ASSOCIATED WITH 2ND NEURON : W21, W22,␣
 ↪W23, W24
weights3 = weights[2] #LIST OF WEIGHTS ASSOCIATED WITH 3RD NEURON : W31, W32,␣
 ↪W33, W34

biases = [2, 3, 0.5]

bias1 = 2
bias2 = 3
bias3 = 0.5

outputs = [
 # Neuron 1:
 inputs[0]*weights1[0] +
 inputs[1]*weights1[1] +
 inputs[2]*weights1[2] +
 inputs[3]*weights1[3] + bias1,
 # Neuron 2:
 inputs[0]*weights2[0] +
 inputs[1]*weights2[1] +
 inputs[2]*weights2[2] +
 inputs[3]*weights2[3] + bias2,
 # Neuron 3:
 inputs[0]*weights3[0] +
 inputs[1]*weights3[1] +
 inputs[2]*weights3[2] +
 inputs[3]*weights3[3] + bias3]

print(outputs)
```

```
[4.8, 1.21, 2.385]
```

USING LOOPS FOR BETTER AND EASIER CODING

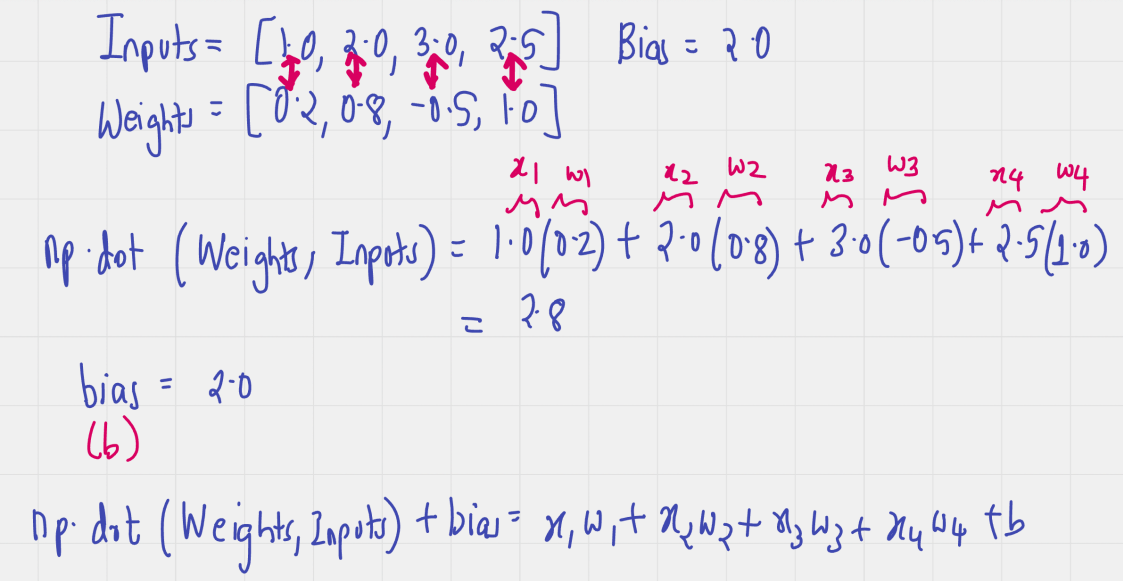
[50]:
```
inputs = [1, 2, 3, 2.5]

##LIST OF WEIGHTS
weights = [[0.2, 0.8, -0.5, 1],
 [0.5, -0.91, 0.26, -0.5],
 [-0.26, -0.27, 0.17, 0.87]]

##LIST OF BIASES
biases = [2, 3, 0.5]
```

```
# Output of current layer
layer_outputs = []

# For each neuron
for neuron_weights, neuron_bias in zip(weights, biases):
 # Zeroed output of given neuron
 neuron_output = 0
 # For each input and weight to the neuron
 for n_input, weight in zip(inputs, neuron_weights):
 # Multiply this input by associated weight
 # and add to the neuron's output variable
   neuron_output += n_input*weight ## W31*X1 + W32*X2 + W33*X3 + W34*X4
   # Add bias
 neuron_output += neuron_bias ## ## W31*X1 + W32*X2 + W33*X3 + W34*X4 + B3
 # Put neuron's result to the layer's output list
 layer_outputs.append(neuron_output)
print(layer_outputs)
```

```
[4.8, 1.21, 2.385]
```

USING NUMPY

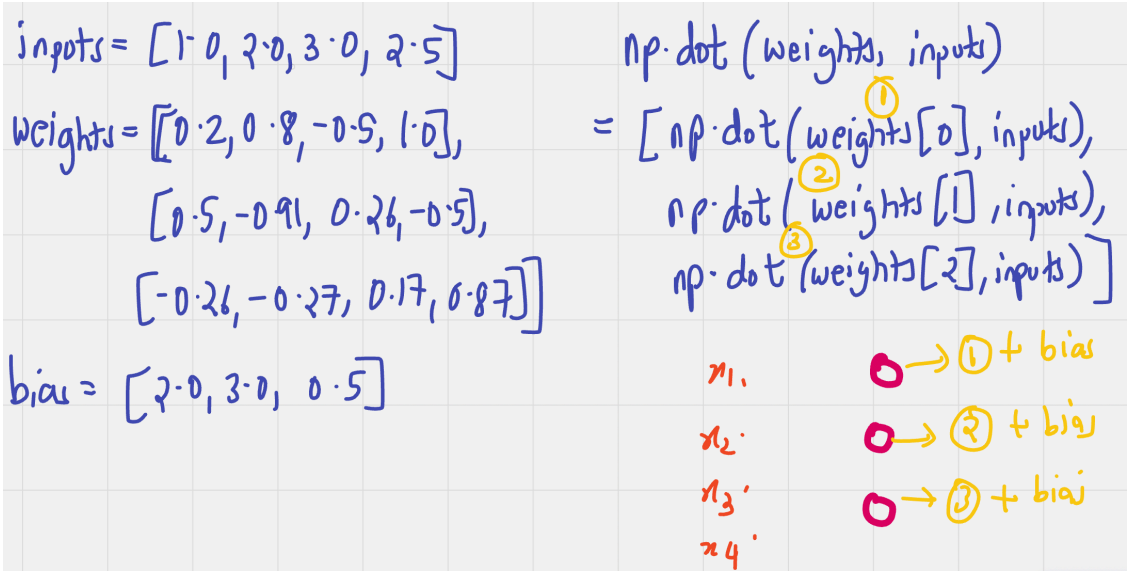
SINGLE NEURON USING NUMPY



[42]:
```
import numpy as np
inputs = [1.0, 2.0, 3.0, 2.5]
weights = [0.2, 0.8, -0.5, 1.0]
bias = 2.0
outputs = np.dot(weights, inputs) + bias
print(outputs)
```

4.8

LAYER OF NEURONS USING NUMPY

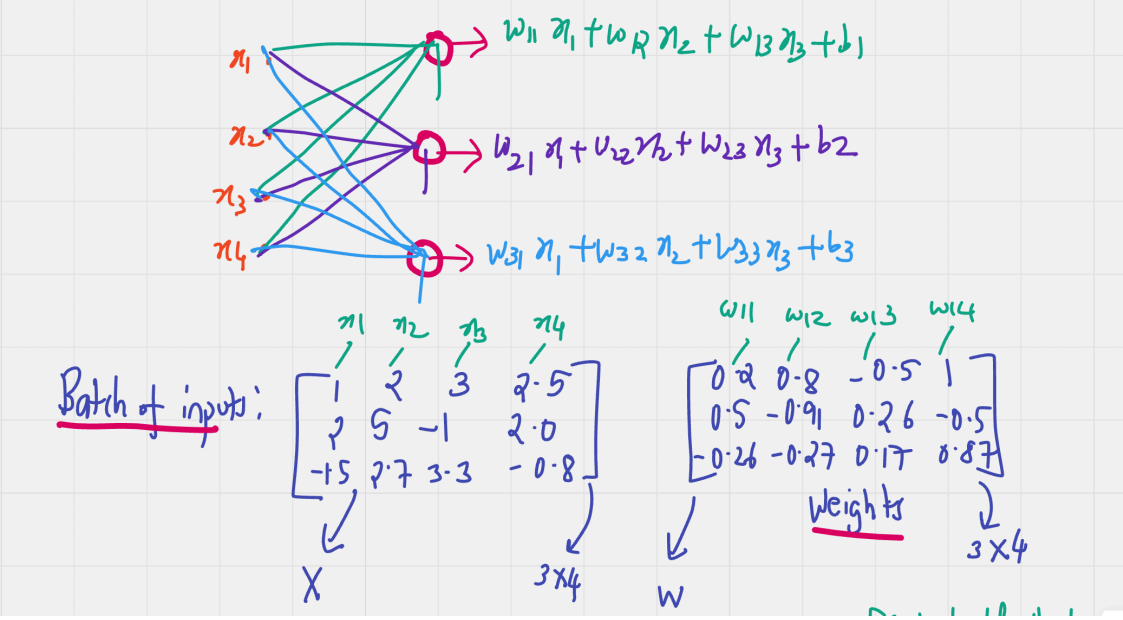


[43]:
```
##  In plain Python, we wrote this as a list of lists. With NumPy, this will be␣
↪a 2-dimensional array, which we'll call a matrix.
```

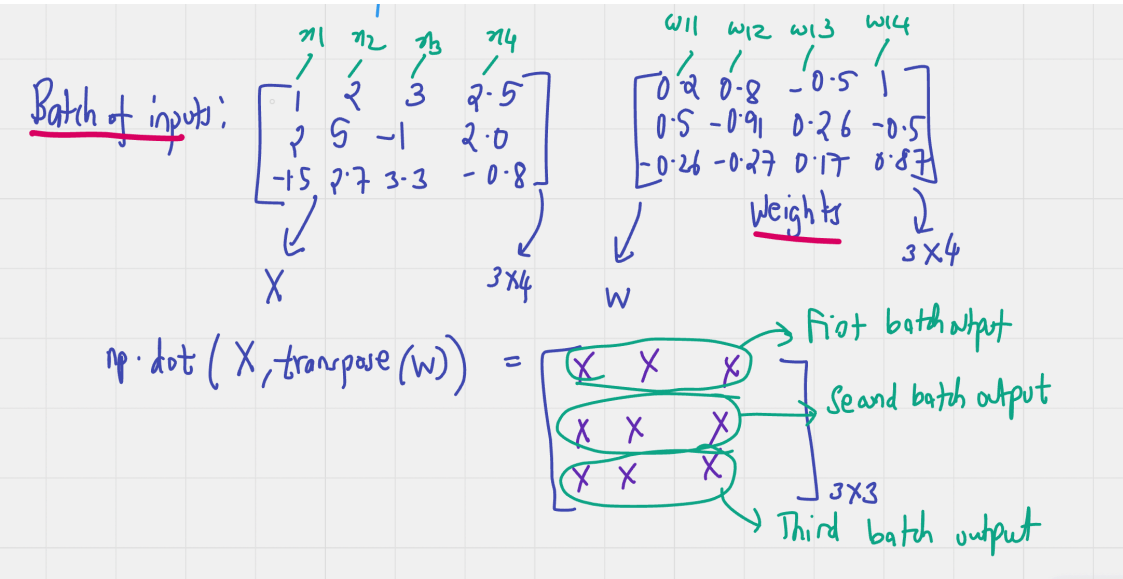
[44]:
```
inputs = [1.0, 2.0, 3.0, 2.5]
weights = [[0.2, 0.8, -0.5, 1],
 [0.5, -0.91, 0.26, -0.5],
 [-0.26, -0.27, 0.17, 0.87]]
biases = [2.0, 3.0, 0.5]

# A dot product of a matrix and a vector results in a list of dot products.
#The np.dot() method treats the matrix as a list of vectors and performs a dot␣
↪product of each of those vectors with the other vector

layer_outputs = np.dot(weights, inputs) + biases
print(layer_outputs)
```

```
[4.8   1.21  2.385]
```

LAYER OF NEURONS AND BATCH OF DATA USING NUMPY



NEED TO TAKE TRANSPOSE OF WEIGHT MATRIX



[45]:
```
inputs = [[1.0, 2.0, 3.0, 2.5], [2.0, 5.0, -1.0, 2.0], [-1.5, 2.7, 3.3, -0.8]]
weights = [[0.2, 0.8, -0.5, 1],
 [0.5, -0.91, 0.26, -0.5],
 [-0.26, -0.27, 0.17, 0.87]]
biases = [2.0, 3.0, 0.5]

# NOTE: WE CAN'T TRANSPOSE LISTS IN PYTHON, SO WE HAVE THE CONVERT THE WEIGHTS␣
 ↪MATRIX INTO AN ARRAY FIRST
outputs = np.dot(inputs, np.array(weights).T) + biases
print(outputs)
```

```
[[ 4.8    1.21   2.385]
 [ 8.9   -1.81   0.2  ]
 [ 1.41   1.051  0.026]]
```

[ ]: