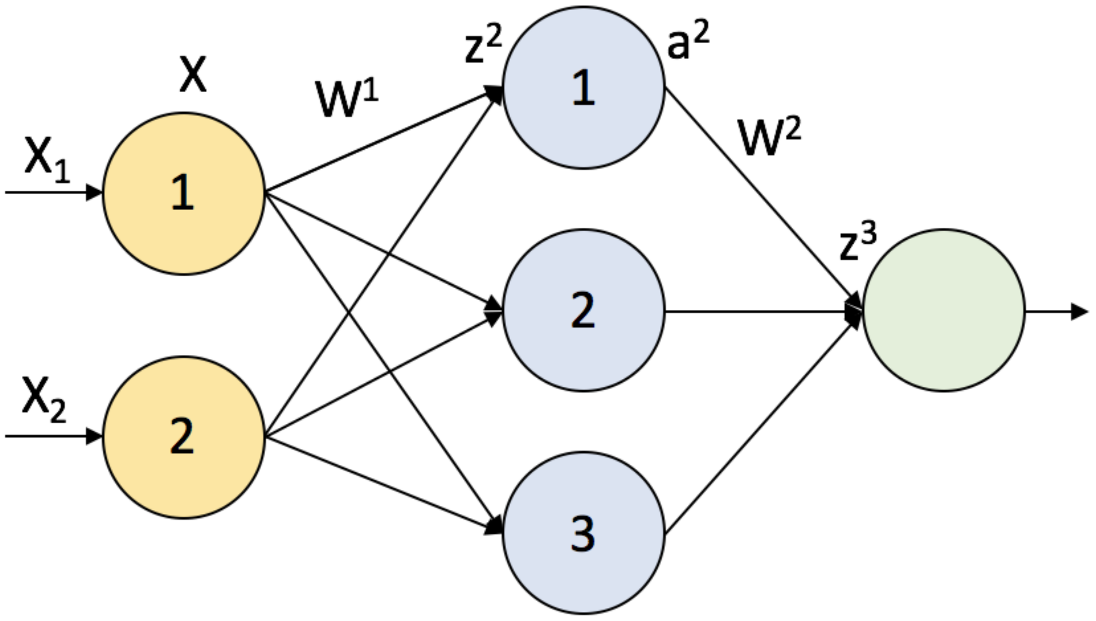
Neural Networks – The Big Picture

1. Introduction

In the first chapter, we have seen what is a neural network, and what can we expect from them. This approach was looking at the neural networks from the outside. They are black boxes that we have seen are actually performing mathematical operations.

In order to fully understand neural networks, it's time to look at them from the inside now. In this chapter, we will look at how these neural networks are decomposed into the smallest bricks, and how to code one from scratch based on the behaviour we will be demystifying. Indeed, this decomposition we are going to do is the reason why neural networks and computationally efficient, and we can use them to solve complex problems were large datasets are required.

To start with, let's take a look at our old friend again, and let's try to figure out what are the smallest pieces we can extract from it.



The only difference is that the dimension of the input and the neurons in the hidden layer have been fixed, to get a better understanding while we go over the process. We are going to discover on the way were all the magic comes from.

2. Make the network a function

As we so in the first chapter, neural networks are just a representation of mathematical operations that are taking place. Therefore, we can represent all the variables as vectors in the different dimensions.

First, we have an input of 2 dimensions. This means, that we have 2 different concepts, for example, temperature and pressure. Out input neurons are moving these 2 dimensions to 3 dimensions, which is the number of neurons in the hidden layer. Thus, the dimensions of that matrix must be (2x3). Also, it is not represented for clearer visualization, but we have a term ‘b’, to add the bias after the matrix multiplication. This lead as to this situation:

$$X = \begin{array}{|c|c|}\hline x_1 & x_2 \\ \hline\end{array}$$

$$W^1 = \begin{array}{|c|c|c|}\hline W^1_{11} & W^1_{12} & W^1_{13} \\ \hline W^1_{21} & W^1_{22} & W^1_{23} \\ \hline\end{array}$$

$$b^1 = \begin{array}{|c|c|c|}\hline b^1_1 & b^1_2 & b^1_3 \\ \hline\end{array}$$

Now we have all the tools we need to reach the hidden layer. We just need to apply the matrix multiplication:

$$z^2 = XW^1 + b^1 \qquad \text{(Eq. 1)}$$

The result of this will be:

$$z^2 = \begin{array}{|c|c|c|}\hline z^2_1 & z^2_2 & z^2_3 \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline x_1 w^1_{11} + x_2 w^1_{21} + b^1_1 & x_1 w^1_{12} + x_2 w^1_{22} + b^1_2 & x_1 w^1_{13} + x_2 w^1_{23} + b^1_3 \\ \hline\end{array}$$

The 3 dimensions corresponding to each of the neurons in the hidden layer, makes sense!

Now, let’s take a close look to what is inside these blocks. It looks like each input weight is connecting every input i with every neuron j. That is why the notation w_{ij}, connecting i with j.

Once we are in the layer, it's time to apply the activation function. The result of this will be:

$$a^2 = \begin{bmatrix} a_1^2 & a_2^2 & a_3^2 \end{bmatrix} = \begin{bmatrix} f(z_1^2) & f(z_2^2) & f(z_3^2) \end{bmatrix}$$

We are applying the function, element-wise to every element of the net input of each neuron, so we keep our dimension of 3 as the number of hidden neurons.

The transformation required now according to the neural network scheme is from 3 to 1, as the next layer only has 1 neuron. The last neuron usually does not perform any function as we don't want to apply a non-linearity at this point. Thus, the matrix that connects these two layers must be (3x1).

$$W^2 = \begin{bmatrix} W_1^2 \\ W_2^2 \\ W_3^2 \end{bmatrix}$$

$$b^2 = \begin{bmatrix} b^2 \end{bmatrix}$$

Now we have all the tools to apply the weights of our hidden layer. The equation that needs to be perform then is:

$$z^3 = a^2 \cdot W^2 + b^2 \qquad \text{(Eq. 2)}$$

$$z^3 = \begin{bmatrix} a_1^2 & a_2^2 & a_3^2 \end{bmatrix} \cdot \begin{bmatrix} W_1^2 \\ W_2^2 \\ W_3^2 \end{bmatrix} + \begin{bmatrix} b^2 \end{bmatrix} = \begin{bmatrix} a_1 w_1^2 + a_2 w_2^2 + a_3 w_3^2 + b^2 \end{bmatrix}$$

As our final input does not perform any operation (yet), we can say that z^3 is the final output after the inputs have gone through all the network. This is what we called feed forward operation.

3. Let's increase the dimensions a bit more – Observations

We have accomplished them the whole forward for an observation of 2 dimensions. So, for example, we had $\text{Temperature} = 20\ ºC, P = 1$ bar. But who can learn anything with only one observation? We need to know more in order to improve our knowledge about what is surrounding us. If we want to learn how to play chess, we definitely have to try it several times. Does this fact complicate the process? Of course not! Let's go through it.

Let's say that, for example, we have 4 observations (of course you need much more, but I just picked the smallest number different from the already used 1,2,3, so at the end you can make the relationships easier; we will reach that point don't worry). Thus, our input looks like this:

$$X = \begin{bmatrix} x_1(1) & x_2(1) \\ x_1(2) & x_2(2) \\ x_1(3) & x_2(3) \\ x_1(4) & x_2(4) \end{bmatrix}$$

Now, the variables corresponding to the network itself remain constant. We have not changed anything in the network, we are just using a batch of observation as input, instead of a single observation. Therefore, if we perform the forward, our results are:

$$z^2 = XW^1 + b^1$$

$$X \cdot W + b^1 = \begin{bmatrix} x_1(1) & x_2(1) \\ x_1(2) & x_2(2) \\ x_1(3) & x_2(3) \\ x_1(4) & x_2(4) \end{bmatrix} \cdot \begin{bmatrix} W^1_{11} & W^1_{12} & W^1_{13} \\ W^1_{21} & W^1_{22} & W^1_{23} \end{bmatrix} + \begin{bmatrix} b^1_1 & b^1_2 & b^1_3 \end{bmatrix}$$

$$z^2 = \begin{bmatrix} z^2_1(1) & z^2_2(1) & z^2_3(1) \\ z^2_1(2) & z^2_2(2) & z^2_3(2) \\ z^2_1(3) & z^2_2(3) & z^2_3(3) \\ z^2_1(4) & z^2_2(4) & z^2_3(4) \end{bmatrix} = \begin{bmatrix} x_1(1)w^1_{11} + x_2(1)w^1_{21} + b^1_1 & x_1(1)w^1_{12} + x_2(1)w^1_{22} + b^1_2 & x_1(1)w^1_{13} + x_2(1)w^1_{23} + b^1_3 \\ x_1(2)w^1_{11} + x_2(2)w^1_{21} + b^1_1 & x_1(2)w^1_{12} + x_2(2)w^1_{22} + b^1_2 & x_1(2)w^1_{13} + x_2(2)w^1_{23} + b^1_3 \\ x_1(3)w^1_{11} + x_2(3)w^1_{21} + b^1_1 & x_1(3)w^1_{12} + x_2(3)w^1_{22} + b^1_2 & x_1(3)w^1_{13} + x_2(3)w^1_{23} + b^1_3 \\ x_1(4)w^1_{11} + x_2(4)w^1_{21} + b^1_1 & x_1(4)w^1_{12} + x_2(4)w^1_{22} + b^1_2 & x_1(4)w^1_{13} + x_2(4)w^1_{23} + b^1_3 \end{bmatrix}$$

Again, the only difference is that we have a new row per each different observation. The next steps are to apply element-wisely the activation function with z^2 as the net input of each hidden neuron. Therefore, the shape of a^3 is exactly the same size as z^2.

$$a^2 = \begin{bmatrix} a^2_1(1) & a^2_2(1) & a^2_3(1) \\ a^2_1(2) & a^2_2(2) & a^2_3(2) \\ a^2_1(3) & a^2_2(3) & a^2_3(3) \\ a^2_1(4) & a^2_2(4) & a^2_3(4) \end{bmatrix} = \begin{bmatrix} f\left(z^2_1(1)\right) & f\left(z^2_2(1)\right) & f\left(z^2_3(1)\right) \\ f\left(z^2_1(2)\right) & f\left(z^2_2(2)\right) & f\left(z^2_3(2)\right) \\ f\left(z^2_1(3)\right) & f\left(z^2_2(3)\right) & f\left(z^2_3(3)\right) \\ f\left(z^2_1(4)\right) & f\left(z^2_2(4)\right) & f\left(z^2_3(4)\right) \end{bmatrix}$$

And finally, applying (Eq. 2 to calculate the output:

$$z^3 = \begin{bmatrix} a_1^2(1) & a_2^2(1) & a_3^2(1) \\ a_1^2(2) & a_2^2(2) & a_3^2(2) \\ a_1^2(3) & a_2^2(3) & a_3^2(3) \\ a_1^2(4) & a_2^2(4) & a_3^2(4) \end{bmatrix} \cdot \begin{bmatrix} W_1^2 \\ W_2^2 \\ W_3^2 \end{bmatrix} + \begin{bmatrix} b^2 \end{bmatrix} = \begin{bmatrix} a_1^2(1)w_1^2 + a_2^2(1)w_2^2 + a_3^2(1)w_3^2 + b^2 \\ a_1^2(2)w_1^2 + a_2^2(2)w_2^2 + a_3^2(2)w_3^2 + b^2 \\ a_1^2(3)w_1^2 + a_2^2(3)w_2^2 + a_3^2(3)w_3^2 + b^2 \\ a_1^2(4)w_1^2 + a_2^2(4)w_2^2 + a_3^2(4)w_3^2 + b^2 \end{bmatrix}$$

4. Summary

If we give now names to the different constants to generalize the mechanism we have the following:

Symbol	**Variable**
n	Input dimension
m	Layer dimension
k	Batch dimension (observations)
o	Output dimension

With these, we can define the shape of the variables:

Symbol	**Variable**
X	$(k \cdot n)$
W^1	$(n \cdot m)$
b^1	(m)
z^2	$(k \cdot m)$
W^2	$(m \cdot o)$
b^2	(o)
a^2	$(k \cdot m)$
z^3	$(k \cdot o)$