SuperDataScience

HOW MANY HIDDEN LAYERS AND HOW MANY NEURONS TO INCLUDE IN AN ARTIFICIAL NEURAL NETWORK?

1. WHAT ARE HIDDEN LAYERS?

A common question for developers of artificial neural networks is about the number of hidden layers and the number of neurons in each layer. To answer this question, we first need to understand what the role of hidden layers in neural networks is.

A hidden layer is a set of neurons that represents a mathematical transformation applied to the previous layer. This transformation is a linear operation of the type

$$wX + b$$

w — is the weight

X — is the value of the neuron in the previous layer

b — is the bias unit

Thus, if the problem we are trying to solve is linear in nature (if the neural network is trying to determine the equation of the line in a regression problem; or if it is trying to classify linearly separable data in a classification problem), no hidden layers are needed. The direct connection between the input layer and the output layer is sufficient to generate an efficient model.

Hidden layers are included in cases of a non-linear nature. They have the necessary property to solve these cases because, at their output, we apply an activation function of a non-linear nature.

2. HOW MANY HIDDEN LAYERS?

It is well accepted in the literature that, in the vast majority of cases, the use of a single hidden layer is sufficient, since this structure is capable of approximating any non-linear equation (such as, for example, quadratic or exponential equations). Two hidden layers are already capable of representing any relationship between the data, even those that cannot be represented by equations. More than two hidden layers are only needed in even more complex problems such as time series and computer vision, where there is a certain interrelation between the dimensions that the data contain (time in the first case, and geometric shapes in the second).

That said, before developing the structure of the neural network it's important to know the characteristics of the data we are working with. Depending on the nature of the problem, using a few hidden layers can lead to underfitting (imagine trying to solve a quadratic problem without using hidden layers), while overfitting is possible with too many layers (the same problem with three hidden layers). The decision must be made by the smaller number of hidden layers that solves the problem, which can be measured by the performance metrics applied to a validation dataset.

3. HOW MANY NEURONS IN A HIDDEN LAYER?

The number of neurons in each layer, on the other hand, is a more empirical issue, with no explicit rules for an ideal calculation. Jeff Heaton, the author of Introduction to Neural Networks for Java, suggests three initial approaches, which we will apply as an example

for a network containing 30 neurons in the input layer and 2 neurons in the output layer:

- The number of hidden neurons must be between the size of the input layer and the size of the output layer. Using the average number between the two layers is a good option; that is, in our example, the value of (30 + 2) / 2 = 16 neurons
- The number of hidden neurons must be 2/3 the size of the input layer, plus the size of the output layer. Thus, the hidden layer in our example must contain 30 * 2/3 + 2 = 22 neurons
- The number of hidden neurons must be less than twice the size of the input layer. That is, in our example, the hidden layer must contain less than 60 neurons

Based on these three estimates, we could build three initial models and make small adjustments to the interval between the two best results. For example, if the problem reaches the smallest errors in networks with 16 and 22 hidden neurons, we can test new networks with values in that range.

If we are working with more layers, we can include a first layer based on the criteria above, and a second one considering that the first layer is your input layer. That is, for the neural network defined above in the format 30> 16> 2, we can include the second layer hidden between 16> 2 which, by the same criteria, would have (16 + 2) / 2 = 9 neurons.

The above recommendations are a good starting point, but the final decision can benefit from the intuition of the developer well acquainted with the data he is working with.