Hidden-Layer neural network - Part A

A feedforward neural network is the simplest type of artificial neural network in which connections between Â perceptrons do not form a cycle. The time it takes to train a model depends on the number of parameters trained, so this knowledge can really help us sometimes.Â   
Â a feedforward neural network that defines the mapping y = f (x; Î¸) and learns the values â€‹â€‹of the parameters Î¸ to help find the best approximation of the function.

**To find the total number of parameters, the following sum must be made:**

* product of the number of neurons of the input layer and of the first hidden layer
* the sum of the product of the number of neurons between Â two consecutive hidden layers
* products. of the number of neurons in the final hidden layer and the output layerÂ
* the total number of neurons in all Â hidden Â and output layers

*arrow\_forward*

Part A and Part B

According to the question, the expression upon solving gives N+1 layers in which N is the number of hidden layers. Q being the number of trainable parameters is

(i Ã— u1 + u1 Ã— u2 + u2 Ã— u3 + u3 Ã— o) + u1 + u2 + u3+ o.

The input dimension and the number of data points are really important as it is said the the trainable parameters can be found when we know the input dimension N. If the input and output points are not given it is difficult to test the hidden layers. Also, in this question we are not aware of the number of hidden layers. So, in this case the data points helped as we could test the number of data points \* number of inputs (input dimension) which gave us the hidden layer. So by calculating we get the number of trainable parameters.

*arrow\_forward*

Difference with Kernel Methods

**The difference between Feed Forward Neural Networks and the Kernel Methods is that -**

The kernel takes the input function, transforms it into another space, then basically works with those transformed functions (when we apply the kernel trick, we do all that implicitly, but it doesn't change that Â fact), Â then perform a linear classification in this new space.Â   
Â A neural network takes the input entities, transforms them into another space, then works with those transformed entities (hidden layer activation) and then does Â linear classification in this new space.Â   
Â Now the main difference is that a neural network learns to represent this Â "on its own".

For example, we can Â influence the network slightly. either by Â applying specific

modifiers or by changing the activation functions or possibly by learning specific algorithms. But in the grand scheme of things, the network will learn an obscure feature space that we won't understand and Â will have trouble interpreting.

One of the major advantages of neural networks is that they are compact and parametric; when you start training your model stays the same size and test time prediction is a matter of matrix multiplication/convolution and basic maths. Adapting multipliers to dual forces you to respect support vectors, and the number you need to maintain grows rapidly with the complexity of the decision boundary. It also means that things get very computationally intensive at the time of testing.