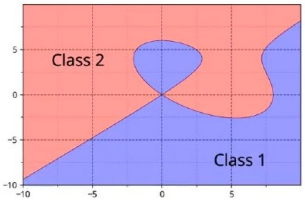
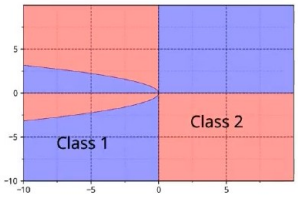
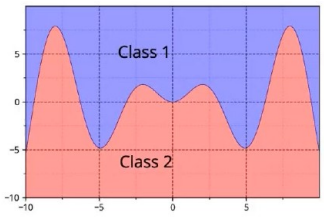
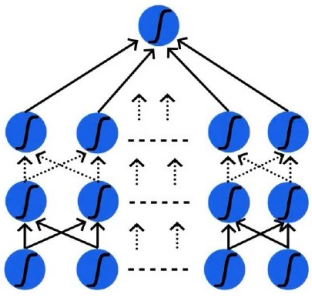
Illustrative Proof of Universal Approximation Theorem

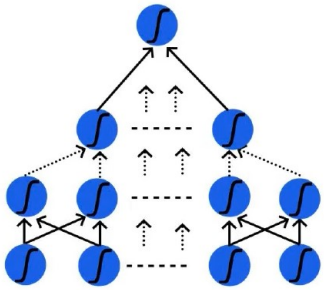
Niranjan Kumar Follow
Mar 20 · 6 min read

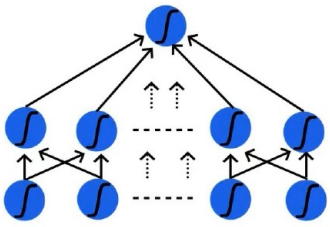












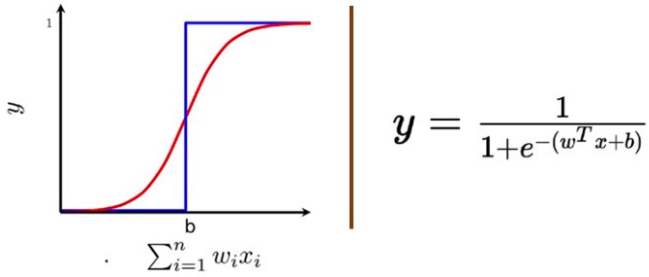
In this post, we will talk about the Universal approximation theorem and we will also prove the theorem graphically. This is a follow-up post of my previous post on Sigmoid Neuron.

Citation Note: The content and the structure of this article is based on the deep learning lectures from One-Fourth Labs — Padhai.

Sigmoid Neuron

Before we go into the actual discussion of Universal approximation theorem, I will give a brief recap on working of Sigmoid Neuron and how it will handle non-linearly separable data.

A sigmoid neuron is similar to the perceptron neuron, for every input `xi` it has weight `wi` associated with the input. The weights indicate the importance of the input in the decision-making process. The output from the sigmoid is not 0 or 1 like the perceptron model instead it is a real value between 0–1 which can be interpreted as a probability. The most commonly used sigmoid function is the logistic function, which has a characteristic of an "**S**" shaped curve.



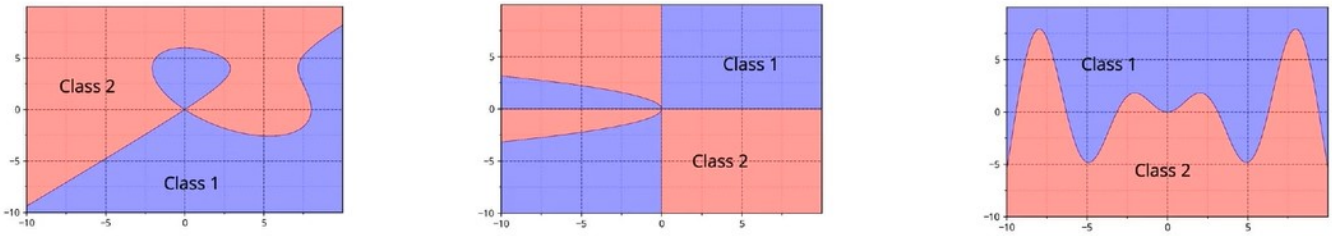
Sigmoid Neuron Representation

Handling Non-Linear Data

From my previous post on Sigmoid Neuron, we have seen that even though we have introduced non-linear Sigmoid it is still not able to effectively separate ones and zeroes. The important point is that from a rigid decision boundary in perceptron, we have taken our first step in the direction of creating a decision boundary that works well for non-linearly separable data. Hence the sigmoid neuron is the building block of a deep neural network.

Modeling Complex Relations

In real life, we deal with complex functions where the relationship between input and output might be complex, unlike the simple sigmoid neuron or perceptron.



Complex Functions

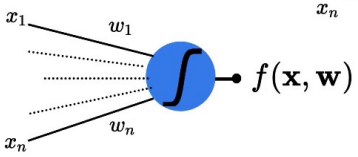
> *How do we even come up with such complex functions?*

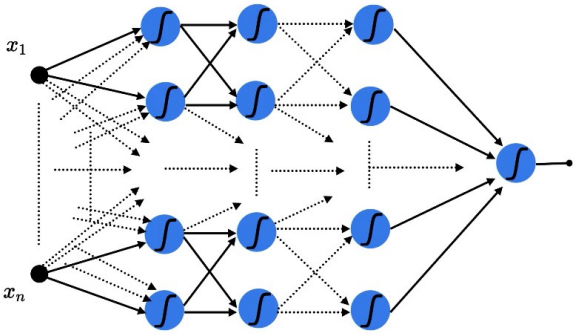
It's hard to come up with such functions there could be infinite possibilities for such a function, we need a simple approach. To solve this problem, let's take an analogy of building a house.

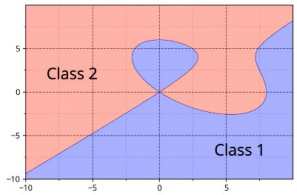
You can think of the house as a complex function, the way we will build a house is we start with a building a block i.e... brick in this case. First, we will lay the foundation and on top of that we will lay another layer and we will just keep continuing in this manner till we get very complex output i.e… house. In this process, we are combining the very simple building blocks in an effective way so that we can get this house.

$$f(x_1, .., x_n) = \frac{1}{1+e^{-(w_1 * x_1 + ... + w_n * x_n + b)}}$$

$$f(\mathbf{x}, \mathbf{w}) = \frac{1}{1+e^{-(w*x+b)}}$$







Modeling Complex Functions

In our case instead of bricks and houses, we are interested in complex functions. The building block in building complex functions is sigmoid neuron. The way we are going to create complex functions is that we will combine the sigmoid neurons in an effective way so that there will be an interaction between the neurons present at different layers. So that the predicted output is a complex function of inputs and there will be weights associated across multiple layers. Now the question arises, how do we decide how many layers to use, how many neurons in each layer etc… these are valid questions but the discussion on these questions is out of the scope of this article.

Now I need to prove to you that the output from the complex function created using the network of neurons is very close to the true relationship between input and the output.

> *Deep neural network with a certain number of hidden layers should be able to approximate any function that exits between input and output.*

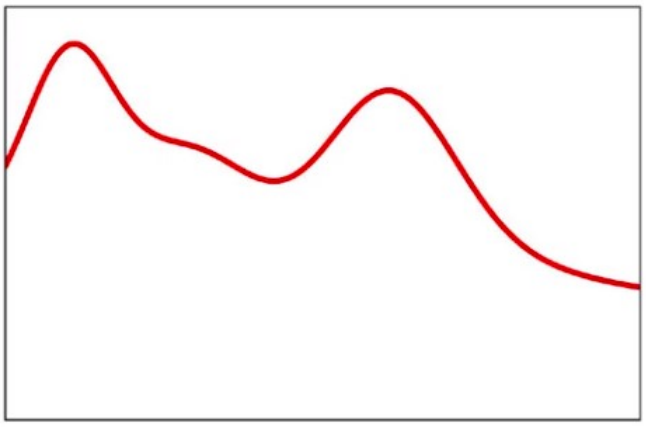
Universal approximation theorem says:
that a feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of **R**, under mild assumptions on the activation function—Wikipedia

In simple terms, UAT says that—you can always come up with a deep neural network that will approximate any complex relation between input and output.

Proof of Universal Approximation Theorem

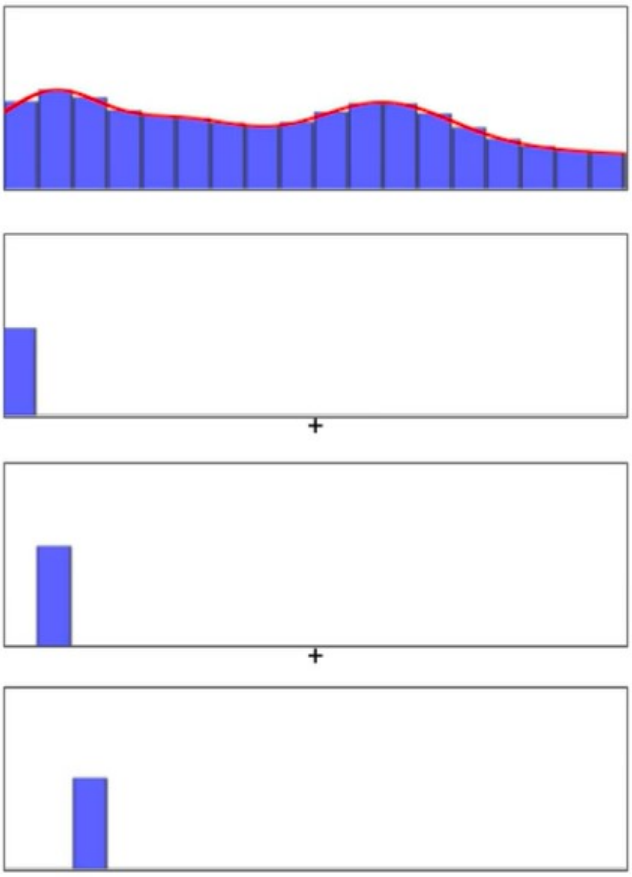
In this section, we will see the illustrative proof of Universal Approximation theorem.

For this illustrative proof, we will consider a simple example where we have one-dimensional input **x** and output **y**, the graph shown below represents the true relationship between input and the output. Let's assume that I don't know what that function equation between **x** and **y** and I can't come up with a representation for the equation.



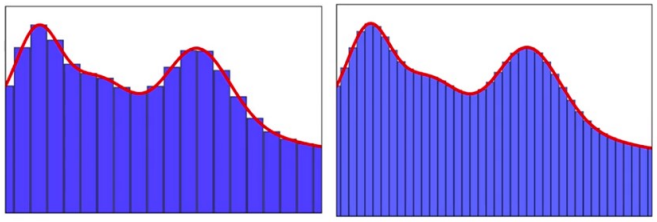
True relationship between x and y

To solve this problem, I will break this function into multiple smaller parts so that each part is represented by a simpler function. By combining the series of smaller functions (rectangular bars/towers) I can approximate the relation between **x** and **y** as close to the true relationship possible.



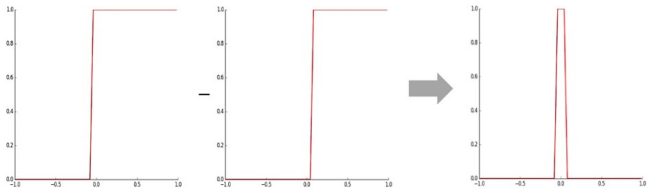
Combination of simpler functions

The key point to note in this case is that I don't have to worry about coming up with complex equations to represent the relationship between input and output. I can just come up with a simple function and use the combination of these functions to approximate the relationship to my true relationship. The more functions that I choose in this method the better will be my approximation.



How do we come up with these rectangles/towers and how it will tie back to the sigmoid neuron?

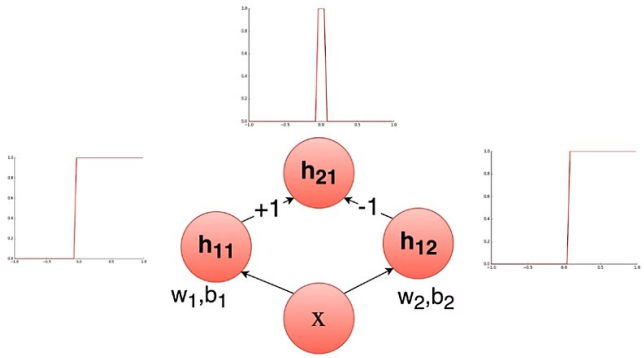
Let's take two sigmoid functions having a very steep slope and notice that they have a different place at which they peak. The left sigmoid peaks just before zero and right sigmoid peaks just after zero. If I just subtract these two functions the net effect is going to be a tower (rectangular output).



If we can get the series of these towers, then we can approximate any true function between input and output.

Neural Network

Can we come up with a neural network to represent this operation of subtracting one sigmoid function from another?.



Neural network for subtraction

if we have an input **x** and it is passed through the two sigmoid neurons and the output from these two neurons are combined in another neuron with weights +1 and -1 i.e… same is as subtracting these two outputs, then we will get our tower. Now you can see that we have our building block ready which is a connection of three sigmoid neurons. If we can construct many such building blocks and add all of them up, we can approximate any complex true relationship between input and output. This is called the *representational power of deep neural networks*, to approximate any kind of relationship between input and output.

Photo by Sergey Pesterev on Unsplash

Conclusion

In this post, we briefly looked at the working of sigmoid neuron and its ability to handle non-linear data. We then looked at how can we model complex relationships and saw the formal definition of Universal Approximation theorem and then went on to see the illustrative proof of Universal Approximation theorem.

Recommended Reading

Sigmoid Neuron — Deep Neural Networks

The building block of the deep neural networks is called the sigmoid neuron.

towardsdatascience.com

Niranjan Kumar - Hacker Noon

Read writing from Niranjan Kumar in Hacker Noon. Intern at HSBC Analytics. ML and DL Enthusiast. Writer at...

hackernoon.com

> *In my next post, we will discuss the feed forward neural works in detail for both regression and classification tasks. Stay tuned for that.*

Niranjan Kumar is working as an intern at HSBC Analytics division. He is passionate about deep learning and AI. He is one of the top writers at Medium in Artificial Intelligence. Connect with me on LinkedIn or follow me on twitter for updates about upcoming articles on deep learning and Artificial Intelligence.