


# DATA STUFF

Your Source Of Knowledge, Tips And Tricks For Anything Data Related  
(<http://www.datastuff.tech/>)



 Let's get some math going

## WHY DO NEURAL NETWORKS NEED AN ACTIVATION FUNCTION?

Keep teams flowing



(<https://youradchoices.com>)

 July 1, 2019 ([Http://Www.datastuff.tech/Machine-Learning/Why-Do-Neural-Networks-Need-An-Activation-Function/](http://www.datastuff.tech/machine-learning/why-do-neural-networks-need-an-activation-function/))  Strikingloo  
([Http://Www.datastuff.tech/Author/Strikingloo/](http://www.datastuff.tech/author/strikingloo/))

Why do Neural Networks Need an Activation Function? Whenever you see a Neural Network's architecture for the first time, one of the first things you'll notice is they have a lot of interconnected layers.

**Each layer in a Neural Network has an activation function, but why are they necessary? And why are they so important? Learn the answer here.**

### What are activation functions?

To answer the question of what Activation Functions are, let's first take a step back and answer a bigger one: What is a Neural Network?

## What are Neural Networks?

A Neural Network is a Machine Learning model that, given certain input and output vectors, will try to “fit” the outputs to the inputs.

What this means is, given a set of observed instances with certain values we wish to predict, and some data we have on each instance, it will try to generalize those data so that it can predict the values correctly for new instances of the problem.

As an example, we may be designing an image classifier (typically with a Convolutional Neural Network (<http://www.datastuff.tech/machine-learning/convolutional-neural-networks-an-introduction-tensorflow-eager/>)). Here, the inputs are a vector of pixels. The output could be a numerical class label (for instance, 1 for dogs, 0 for cats).


This would train a Neural Network to predict whether an image contains a cat or a dog.

But what is a mathematical function that, given a set of pixels, returns 1 if they correspond to the image of a dog, and 0 to the image of a cat?

Coming up with a mathematical function that did that by hand would be impossible. **For a human.**

So what we did is invent a Machine that finds that function for us.

It looks something like this:

 Image result for neural network mlp

Single hidden layer Neural Network. Source  
([https://docs.opencv.org/2.4/modules/ml/doc/neural\\_net\\_works.html](https://docs.opencv.org/2.4/modules/ml/doc/neural_net_works.html)).

But you may have seen this picture many times, recognize it for a Neural Network, and still not know exactly what it represents.

Here, each circle represents a neuron in our Neural Network, and the vertically aligned neurons represent each layer.

How do Neural Networks work?

A neuron is just a **mathematical function**, that takes inputs (the outputs of the neurons pointing to it) and **returns outputs**.

These outputs serve as inputs for the next layer, and so on until we get to the final, output layer, which is the actual value we return.

There is an input layer, where each neuron will simply return the corresponding value in the inputs vector.

For each set of inputs, the Neural Network's goal is to make each of its outputs **as close as possible to the actual expected values**.

Again, think back at the example of the image classifier.

If we take 100x100px pictures of animals as inputs, then our input layer will have 30000 neurons. That's 10000 for all the pixels, times three since a pixel is already a triple vector (RGB values).

We will then run the inputs through each layer. We get a **new vector as each layer's output, feed it to the next layer as inputs**, and so on.

Each neuron in a layer will return a single value, so a layer's output vector will have as many dimensions as the layer has neurons.

So, which value will a neuron return, given some inputs?

What does a Neuron do?

A neuron will take an input vector, and do three things to it:

- Multiply it by a weights vector.
- Add a bias value to that product.
- Apply an **activation function** to that value.

And we finally got to the core of our business: that's what activation functions do.

We'll typically use **non-linear functions as activation functions**. This is because the linear part is already handled by the previously applied product and addition.

## What are the most commonly used activation functions?

I'm saying non-linear functions and it sounds logic enough, but what are the typical, commonly used activation functions?

Let's see some examples.


### ReLU

ReLU stands for "Rectified Linear Unit".


Of all the activation functions, this is the one that's **most similar to a linear one**:

- For non-negative values, it just applies the identity.
- For negative values, it returns 0.

In mathematical words,

 
$$f(x) = x^+ = \max(0, x)$$

This means all negative values will become 0, while the rest of the values just stay as they are.

 Image result for ReLU

This is a biologically inspired function, since neurons in a brain will either

“fire” (return a positive value) or not (return 0).

Notice how combined with a bias, this actually **filters out any value beneath a certain threshold**.

Suppose our bias had a value of -b. Any input value lower than b, after adding the bias will become negative. This turns to a 0 after applying ReLU to it.

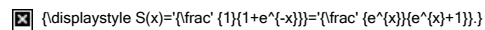
## Sigmoid

The sigmoid function takes any real number as input, and **returns a value between 0 and 1**. Since it is continuous, it effectively “smushes” values:

If you apply the sigmoid to 3, you get 0.95. Apply it to 10, you get 0.999... And it will keep approaching 1 without ever reaching it.


The same happens in the negative direction, except there it converges to 0.

Here’s the mathematical formula for the sigmoid function.


$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

As you see, it approaches 1 as x approaches infinity, and approaches 0 if x approaches minus infinity.

It is also symmetrical, and has a value of 1/2 when its input is 0.

 Image result for sigmoid

Since it takes values between 0 and 1, this function is extremely useful as an output if you want to model a probability.

It’s also helpful if you wish to apply a “filter” to partially keep a certain value (like in an LSTM’s forget gate (<http://www.datastuff.tech/machine-learning/lstm-how-to-train-neural-networks-to-write-like-lovecraft/>)).

## Why do Neural Networks Need an Activation

## Function?

We've already talked about the applications some different activation functions have, in different cases.

Some let a signal through or obstruct it, others filter its intensity. There's even the `tanh` ([https://en.wikipedia.org/wiki/Hyperbolic\\_function](https://en.wikipedia.org/wiki/Hyperbolic_function)) activation function: instead of filtering, it turns its input into either a negative or positive value.

But what why do our Neural Networks need Activation Functions? **What would happen if we didn't use them?**

I found the explanation for this question in Yoshua Bengio's awesome Deep Learning book (<https://amzn.to/305g2MF>), and I think it's perfectly explained there.

We could, instead of composing our linear transformations with non-linear functions, make each neuron simply return their result (effectively composing them with the identity instead).


But then all of our layers would simply stack one affine (product plus addition) transformation after another. Each layer would simply add a vector product, and vector addition, to the previous one.

It can be shown (and you can even convince yourself if you try the math with a small vector on a whiteboard) that this composition of affine transformations, is equivalent to a single affine transformation.

Effectively, this whole "Neural Network" where all activation functions have been replaced by the identity would be **nothing more than a vector product and a bias addition**.

There are many problems a linear transformation can't solve, so we would effectively be **shrinking the quantity of functions** our model could estimate.


As a very simple but earthshaking example, consider the XOR operator.


 XOR values table

Try to find a two-element vector, plus a bias that can take  $x_1$  and  $x_2$ , and turn them into  $x_1 \text{ XOR } x_2$ . Go ahead, I'll wait.

...

Exactly, you can't. nobody can. However, consider

 formula for a neural network that solves the XOR problem.

 defining vectors with latex

If you work the math, you'll see this has the desired output for each possible combination of 1 and 0.

Congratulations! You've just trained your first Neural Network!

And it's learned a problem a linear model could never have learned.

## Conclusions

I hope after this explanation, you now have a better understanding of why Neural Networks need an Activation Function.

In future articles, I may cover other Activation Functions and their uses, like

SoftMax and the controversial Cos.

So what do you think? Did you learn anything from this article? Did you find it interesting? Was the math off?

Feel free to contact me on Twitter (<http://www.twitter.com/strikingloo>), Medium (<http://www.medium.com/@strikingloo>) or dev.to (<http://www.dev.to/strikingloo>) for anything you want to say or ask to me!

If you want to level up as a Data scientist, check out my best Machine Learning books (<http://www.datastuff.tech/data-science/3-machine-learning-books-that-helped-me-level-up-as-a-data-scientist/>) list and my Bash tutorial (<http://www.datastuff.tech/programming/terminal-tutorial-more-productive/>).

#### Related Posts:

Convolutional Neural  
Networks: Python  
Tutorial...

LSTM: How to Train  
Neural Networks to  
Write like Lovecraft

---

📁 Machine Learning ([Http://Www.datastuff.tech/Category/Machine-Learning/](http://www.datastuff.tech/Category/Machine-Learning/))

🔍 Deep Learning ([Http://Www.datastuff.tech/Tag/Deep-Learning/](http://www.datastuff.tech/Tag/Deep-Learning/)), Math ([Http://Www.datastuff.tech/Tag/Math/](http://www.datastuff.tech/Tag/Math/)), Neural Networks ([Http://Www.datastuff.tech/Tag/Neural-Networks/](http://www.datastuff.tech/Tag/Neural-Networks/))

Comments are closed.