Understanding of Activation Functions in

Sergey Sereda

Click here

Sergey Sereda

Senior Consultant (Mobile & Device Dev) at Avanade, Xamarin & Azure Certified

+ Follow

Published May 26, 2020

This is a second part of our series of articles. Other parts you can find here:

1. An introduction to Artificial Neural Networks

1. What is the Activation Function and why it's so important.

Activation function – one of the important parts of the neural network, it's used by each neuron in the hidden layer and output layer to decide whether a neuron should be activated or not. It's like a simple mathematical "gate" between different layers.

For an activation function it is important that it should be computationally efficient to be able to calculate activation functions (mathematical equations) of thousands and thousands of neurons of the neural network.

Desirable properties of an activation functions:

- Non linearity - if the activation function is linear, it doesn't matter how many hidden layers we have, all of them will be collapsed to just one and we'll have a simple linear regression model.
- Continuously differentiable - is important to be able to enable gradient-based optimization methods.

- Range of possible results - when the range of the activation function is finite, gradient-based training methods tend to be more stable. When the range is infinite, training is generally more efficient.
- Monotonic or not - when the activation function is monotonic, the error surface associated with a single-layer model is guaranteed to be convex.
- Approximates identity near the origin - when activation functions have this property, the neural network will learn efficiently when its weights are initialized with small random values. When the activation function does not approximate identity near the origin, special care must be used when initializing the weights.

2. How the activation function works

Let’s explain how the activation function works as simple as we can:

- Each neuron obtains data from the previous layer (input values and weights through the synapses).
- We calculate a weighted sum of inputs and add a bias

- Weighted sum will be put to the activation function.
- Based on the result from the activation function, the neuron will be activated or not.

3. Types of activation functions

There are about **30** different types of activation functions. Anyway researchers from all over the world are still trying to find the most effective by performance and computational efficiency activation function.

Presently the most popular functions are:

- Binary step (threshold-base)
- Linear
- Sigmoid
- TanH / Hyperbolic Tangent
- ReLU (Rectified Linear Unit)
- Leaky ReLU
- Parametric ReLU
- Swish
- Softmax

4. Description of various activation functions

Binary step (threshold-base) activation function

The simplest one. It just shows us on a basis of some threshold if the neuron should be activated or not. Can’t be used for multi value outputs.

Linear activation function

Linear function looks a little bit better than a Binary Step because it can give us a multiple results but we also have several problems:

- With that function there is not possible to use backpropagation to train our model.
- Because of the nature of the linear function all layers of the whole neural network can be collapsed to one. Because a combination of linear functions it’s the same linear function.

Sigmoid activation function

Give a smooth gradient and values predicted between 0.0 and 1.0, which normalize the output for each neuron. There is also a **Bipolar Sigmoid** function which has a range from -1.0 to 1.0.

But unfortunately we have several problems:

- Vanishing gradient - for the very high and very low values of X there is almost no change in Y (prediction) value.
- It’s computationally expensive.
- Outputs not zero centered, which can cause difficulties during the optimization step.

TanH / Hyperbolic Tangent

It’s very close to the Sigmoid function but with several differences. We still have a smooth gradient, but values predicted between -1.0 and 1.0. Values are zero centered. That activation function is still computationally expensive.

ReLU (Rectified Linear Unit)

Very popular right now activation function, computationally efficient and allows a backpropagation. The main problem is a **Dying ReLU** – for 0 or less than zero values of X the network cannot perform backpropagation and cannot learn.

Leaky ReLU

That activation function appeared as an attempt to resolve a problem with the **Dying ReLU** of **ReLU** activation function. It has all the strength of a simple **ReLU** activation function, but prevents **Dying ReLU** problems because of a small slope in the negative area. Which gives us a possibility to perform backpropagation and train a model in a negative area too. But we got a new problem: results are not consistent in the negative area.

Parametric ReLU (PReLU)

It has all the strength of a simple **ReLU** activation function, but we have a parameter which allows us to change a slope in a negative area, perform backpropagation and train a model in a negative area. Also we can learn the most appropriate value of parameter .

Swish

Activation function which was presented by the researchers from Google. It performs a little bit better than **ReLU and Leaky ReLU** but still has a high computational efficiency. Ranges are from minus infinity to infinity.

Softmax

Typically used by the output layer that needs to classify results into multiple categories, because it normalizes the outputs for each predicted class between 0 and 1, and divides by their sum, which gives a probability of predicted classes. Ranges from 0 to 1.

5. Examples

Let’s check how different activation functions will activate neurons with different inputs.

Binary step (threshold-base) activation function

1st, 4th and 5th neurons will be activated.

ReLU (Rectified Linear Unit)

1st, 4th and 5th neurons will be activated.

Softmax

Only the 1st neuron will be activated.

Let’s test several activation functions: **ReLU**, **TanH** and **Sigmoid**. We have an ANN with 6 layers. 5 hidden layers with the 500 neurons and one output layer with the **Sigmoid** activation function. That ANN was trained 5 times with the different activation functions. Let’s check the results:

This table shows that in our case the best activation function is **ReLU**. But in any case we always should test which activation function will suit our needs best.

6. Summary

We described just a small quantity of existing activation functions and it’s really hard to provide some rule of thumb in which situation we should use which activation function. We should always be experimenting and trying to find a better activation function for our needs, as it’s a critical part of the whole neural network and its performance. The baddest part is that there is no activation function that will work in all cases. But anyway there is some considerations which probably can help:

- **Sigmoid** functions and their combinations will work better in the case of classification problems.
- We should avoid **Sigmoid** and **TanH** functions, because they cannot be used in networks with many layers due to the vanishing gradient problem.
- **TanH** should be avoided most of the time due to dead neuron problems.
- We should use the **ReLU** activation function in hidden layers, but be careful with the learning rate and monitor the fraction of dead units.

- If we encounter a problem with the **ReLU** (dead neurons problem for example) we should try variations of **ReLU**: **Leaky ReLU** and **Parametric ReLU**.
- An output layer can be a **Linear** activation function in case of regression problems.
- **Softmax** is typically used in an output layer for multiple classification problems.

7. Bibliography

- https://en.wikipedia.org/wiki/Activation_function
- https://dashee87.github.io/deep%20learning/visualising-activation-functions-in-neural-networks/
- https://towardsdatascience.com

5 · 2 Comments

Like Comment Share

Aru

Arup Kumar Sahoo 2w

Excellent presentation

Like Reply

Ser

Sergey Sereda 2w

Thank you!

Like Reply

To view or add a comment, sign in

More articles by this author

An introduction to Artificial Neural Networks

An introduction to Artificial Neural…

Apr 28, 2020

© 2021

About

Accessibility

User Agreement

Privacy Policy

Cookie Policy

Copyright Policy

Brand Policy

Guest Controls

Community Guidelines

Language