Machine Learning Guide for Petroleum Professionals: Part 2

Image by Shutterstock

Hey! I am glad to have you back. In part 1, we explored the linear regression model having a linear activation function. In this part, we will dive deep into non-linear activation functions and on top of that, understand why we need activation functions. By the time you complete this and part 1, you will have a ground to understand deep learning from scratch (part 3 of this series). Let's begin.

Previously we covered the linear activation function, $\hat{y} = wx + b$, which can only predict linear correlations while performing poorly when applied to non-linear functions. That's why we need non-linear activation functions which help machine learning models to fit complex and real-world data.

Before delving deeper into the different types of activation functions, it's important to note that in the previous part, we discussed linear regression models with one layer, as illustrated in Figure 1.

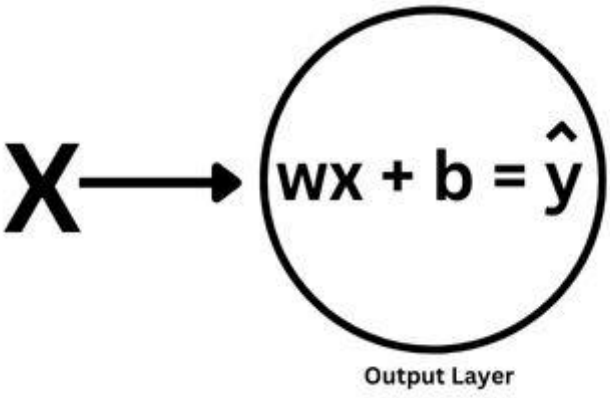

Figure 1: Linear Regression Model. Image by Author.

This model has input X and its corresponding output layer. A circle denotes a neuron, and a single circle means a single neuron. Here, $\hat{y}$ is the output of the last layer (final output).

However, when we have hidden layers (layers between input and output), it's important to understand the notation used to represent multiple hidden layers. In a model with multiple hidden layers, the output from the first hidden layer is denoted by a1, which then becomes the input to the second hidden layer. Similarly, the output from the second hidden layer is denoted by a2, which then becomes the input to the third hidden layer, and so on. The output from the last layer is the final output and is denoted by ŷ. See Figure 2.

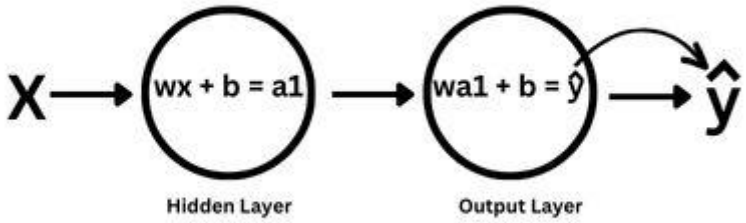

Figure 2: One Hidden Layer model. Image by Author.

Additionally, we previously denoted a linear activation function with ŷ (ŷ = wx + b), but now we are replacing it with z (z = wx + b). We combine this linear activation function with a non-linear version, denoted by g(z), read as g of z. Here g represents the non-linear activation function and z represents the linear activation function. This combination results in the output, denoted by 'a' (specifically, a1, a2, a3, etc. according to its corresponding hidden layer). That's a lot of new things to take in. To summarize, the new notations introduced in this section are:

- z is the output of the linear activation function.
- g is any non-linear activation function (discussed below).
- g(z) is the output after the non-linear activation function is applied to (or combined with) the linear activation function.
- a(n) is the output of a particular nth hidden layer. Mathematically, it is a = g(z).
- ŷ is the final output (output of the last layer).

We will discuss this in more detail in part 3. For now, take a moment to visualize Figure 3 and familiarize yourself with the concept of multiple hidden layers.

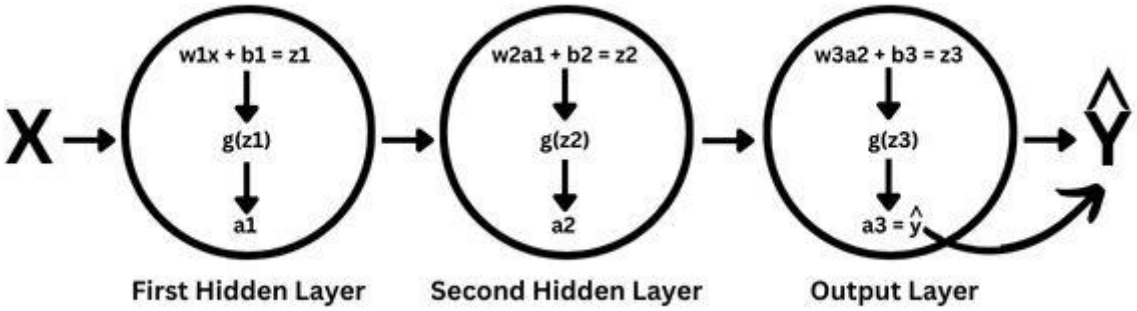

Figure 3: Two Hidden Layer Model. Image by Author.

With a solid understanding of the notation used in machine learning, it is now time to explore the various types of activation functions and understand when to use each one. Activation functions play a crucial role in machine learning by introducing non-linearity, allowing the model to fit complex, real-world data. In this section, we will take a closer look at several popular activation functions and examine their unique properties and suitability for different types of data and model architectures.

1) ReLU Activation Function:

ReLU, short for Rectified Linear Unit, is a widely used non-linear function in machine learning. It is represented in Figure 4.

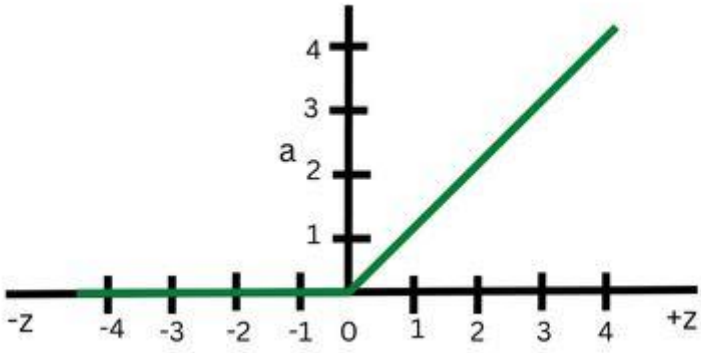

Figure 4: ReLU Activation Function. Image by Author.

From the above graph, we can see that for zero and all the negative values of z, the output g(z) or 'a' is equal to zero (non-linear) and for all positive values of z, g(z) or 'a' is equal to the value of z (linear function). This combination makes it suitable for fitting non-linear functions. Therefore, if we are certain that our final output, ŷ, cannot be negative, ReLU is a great choice. Mathematically, it can be represented as:

$$a = g(z) = \max(0, z)$$

2) Leaky ReLU Activation Function:

Leaky ReLU (LReLU) activation function is an improved version of the ReLU activation function. It differs from ReLU in that instead of setting all negative values to zero, a small

slope value is used to maintain some information from negative inputs. This is represented graphically in Figure 5 as follows:

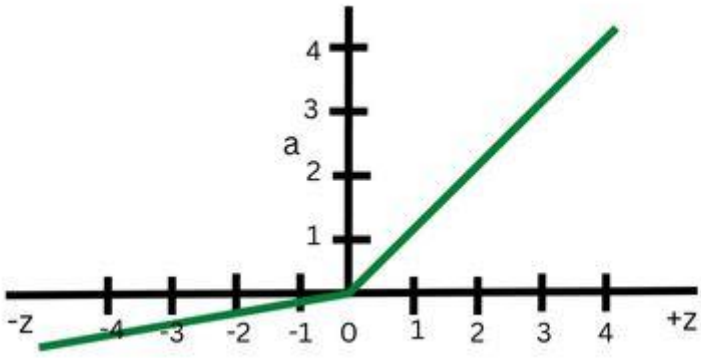

Figure 5: Leaky Relu Activation Function. Image by Author.

The mathematical representation of Leaky ReLU (LReLU) is defined as follows:

$$a = g(z) = \max(\alpha z, z)$$

where α is a small slope parameter, typically set to 0.01. This makes LReLU a suitable choice when the output of a function can be positive, zero, or have small negative values.

3) Tanh Activation Function:

Tanh activation function (also known as a hyperbolic tangent function) is commonly used when the output range is -1 and +1. It produces an "S" shaped curve, as can be seen in the visual representation in Figure 6.

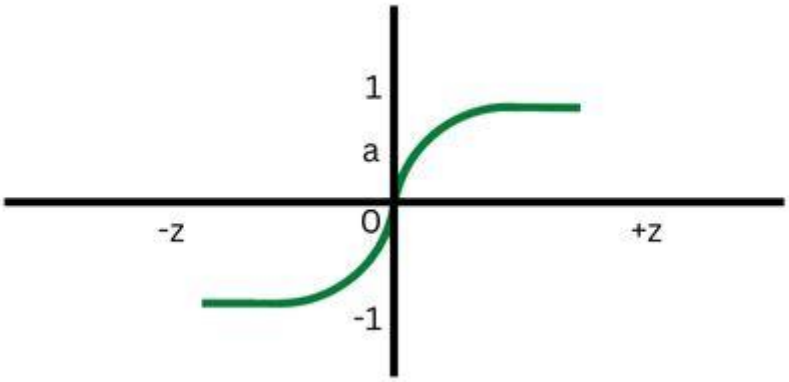

Figure 6: Tanh Activation Function. Image by Author.

Mathematically, it is defined as:

$$a = g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Here, z is a linear function equal to wx + b, and e^z can be written as $e^{(wx+b)}$. For large values of z, such as 10, e^z (e^{10}) will be a very large number and e^{-z} (e^{-10}) will be a very small number. This means the numerator and denominator will be approximately equal to

each other, resulting in a final value of 1. Similarly, for small values of z, such as -10, the final result will be -1.

4) Sigmoid Activation Function:

Sigmoid activation function, also known as logistic function, is used for binary classification (logistic regression problems). Its output is the probability, range from 0 to 1, and, like tanh, it is a "S" shaped curve as shown below in Figure 7:

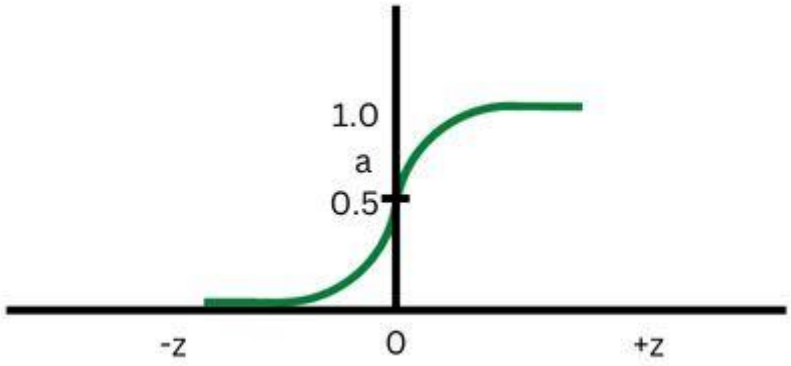

Figure 7: Sigmoid Activation Function. Image by Author.

Mathematically, it is:

$$a = g(z) = \frac{1}{1+e^{-z}}$$

When the input value, z, is a large number, such as 10, the value of e^{-z} becomes very small (0.000045), making the denominator approximately equal to 1, resulting in a final output of 1. On the other hand, when the input value is a small number, such as -10, the value of e^{-z} becomes very large (22026.46), making the denominator a very large number, resulting in a final output close to 0. At z = 0, the output is 0.5. To use the sigmoid function for binary classification, we can set a threshold such that values equal to or greater than 0.5 are classified as one class (for example, oil), and values less than 0.5 are classified as the other class (for example, gas).

5) Softmax Activation Function:

Softmax function, also referred to as multi-class logistic regression, is used for multi-class classification problems where we need to predict more than 2 classes. It is designed to provide the probability of each category, where the sum of the probabilities for all categories equals 1. Similar to sigmoid, the output range of the softmax function is from 0 to 1, and it is visually represented as a curve that maps the input values to the probabilities of each category, shown in below Figure 8:

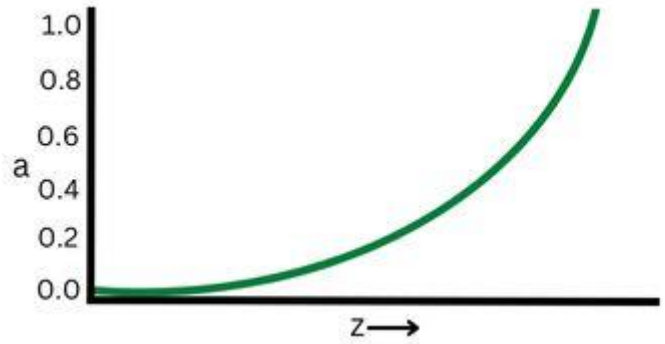

Figure 8: Softmax Activation Function. Image by Author.

Mathematically, it is:

$$a = g(z) = \frac{e^{zi}}{\Sigma e^{zj}}$$

where zi is the i-th element of the vector z and the denominator is the sum of the exponential of all the elements of the vector z. This equation ensures that the sum of the output is equal to 1 and each output is between 0 and 1, which can be interpreted as a probability of each class. The class with the highest probability can be selected as the prediction.

For example, for three classes, we have:

$$a1 = g(z1) = \frac{e^{z1}}{e^{z1} + e^{z2} + e^{z3}}$$

$$a2 = g(z2) = \frac{e^{z2}}{e^{z1} + e^{z2} + e^{z3}}$$

$$a3 = g(z3) = \frac{e^{z3}}{e^{z1} + e^{z2} + e^{z3}}$$

Let say a1 is oil, a2 is gas, and a3 is water and we calculated the values as a1 = 0.2, a2 = 0.25, and a3 = 0.55 for a specific region. In this case, a3 has the largest value, so, that region is a water.

Conclusion and Key Takeaways

This concludes our discussion of the five commonly used activation functions. While there are other activation functions, these five are the most widely used. When choosing an activation function for your model, it is important to consider the specific requirements of your problem and the properties of the different functions. Consider the following:

For the output layer:

- If you have a binary classification problem, use a sigmoid activation function.
- If your problem involves multi-class classification, use the softmax activation function.

- For outputs that are zero or positive, use the ReLU or linear activation function.
- For outputs that can be negative, zero, or positive, use the Leaky ReLU or linear activation function.
- If the output ranges from -1 to +1, use the Tanh activation function.

For hidden layers:

ReLU and variants of ReLU (such as Leaky ReLU) are the most commonly used activation functions. These activation functions have the advantage of being computationally efficient and easy to train. Tanh and sigmoid functions are also sometimes used in hidden layers, although they are less common as they can make the training process slow and difficult.

Thank you for reading to the end. In the next part of our machine learning journey, we will delve into the mathematical concepts behind deep learning, aka Artificial Neural Networks (ANN).

Read part 3 here.