Introduction to Softmax Activation Function for Neural Network

S shipra_saxena
07 Aug, 2024

8 min read

29

Introduction

The **activation function** is an integral part of a neural network. A neural network is a simple **linear regression** model without an activation function. This means the activation function gives non-linearity to the neural network gradient parameter. In this article, we will discuss the SoftMax activation function, which is popularly used for multiclass classification problems. Let's first understand the neural network architecture for a multiclass classification problem and why other activation functions can not be used in this case.

Overview:

- The activation function is one of the building blocks of the Neural Network
- Understand how the Softmax activation function works in a multiclass classification problem.

Table of contents

What is SoftMax Activation Function?

The SoftMax activation function is commonly used in **machine learning**, particularly in neural networks for classification tasks. An activation function converts a vector of raw prediction scores (logits) into probabilities.

Key Characteristics of the SoftMax Function:

1. **Normalization**: The SoftMax activation function normalizes the input values into a probability distribution, ensuring that the sum of all output values is 1. This makes it suitable for classification problems where the output needs to represent probabilities over multiple classes.
2. **Exponentiation**: By exponentiating the inputs, the SoftMax function in machine learning amplifies the differences between the input values, making the largest value more pronounced in the output probabilities.
3. **Differentiability:** The SoftMax function is differentiable and essential for backpropagation in neural networks.

Application of Softmax Activation Function

1. **Neural Networks**: SoftMax activation function is commonly used in the final layer of neural networks to handle multi-class classification problems. It converts the logits (raw output scores) into probabilities, allowing the network to distribute probability across different classes.
2. **Probability Distribution**: SoftMax function transforms a vector of logits into a **probability distribution**. Each output vector element represents the probability that the input belongs to the corresponding class.
3. **Loss Function**: In machine learning, the SoftMax function is often combined with the cross-entropy **loss function** during training. The cross-entropy loss measures the difference between the predicted probability distribution (from SoftMax) and the actual distribution (one-hot encoded labels), guiding the model's learning process.
4. **Soft Attention Mechanisms**: SoftMax activation function is used in **attention mechanisms** within models like transformers to weigh the importance of different elements in a sequence. It helps assign attention weights, normalizing them to sum to 1.
5. **Action Selection**: In **reinforcement learning**, the SoftMax function can convert action value estimates

into probabilities, allowing stochastic action selection based on these probabilities.

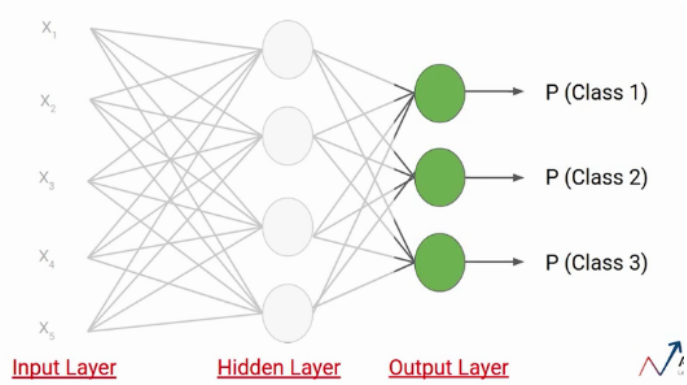
6. **Model Averaging**: In ensemble learning, the SoftMax function can combine predictions from multiple models by averaging their probability distributions, resulting in a more robust final prediction.

Example

Suppose we have the following dataset: For every observation, we have five features from FeatureX1 to FeatureX5, and the target variable has three classes.

Feature X1	Feature X2	Feature X3	Feature X4	Feature X5	Target
1	22	569	35	0	Class 1
1	7	351	75	1	Class 2
1	45	451	542	1	Class 2
1	5	572	8	0	Class 1
0	22	565	44	1	Class 3
0	24	243	546	1	Class 3
1	78	953	42	0	Class 2

Now, let's create a simple neural network to solve this problem. Here, we have an Input layer with five neurons, as we have five features in the dataset. Next, we have one hidden layer with four neurons. Each of these neurons uses inputs, weights, and biases to calculate a value, which is represented as Zij here.



For example, the first neuron of the first layer is represented as Z11 Similarly the second neuron of the first layer is represented as Z12, and so on.

We apply the activation function, let's say a tanh activation function, to these values and send the values or result to the output layer.

The number of neurons in the output layer depends on the number of classes in the dataset. Since we have three classes in the dataset, we will have three neurons in the output layer. Each of these neurons will give the probability of individual classes. This means the first neuron will give you the probability that the data point belongs to class 1. Similarly, the second neuron will give you the probability that the data point belongs to class 2.

Also Read: **Unlocking The Power of Activation Functions in Neural Networks**

Why is Softmax Used in the Last Layer?

Here's how the softmax **function** in machine learning works in the last layer of a neural network :

Input: The softmax activation function takes a vector of real numbers (z) as input. These values typically represent the outputs from the final hidden layer of the **neural network**, often accessed via an API.

Exponentiation: Each element in the input vector z is exponentiated using the mathematical constant e (approximately 2.718). This step ensures all the values become positive. The derivative of this step is crucial for backpropagation.

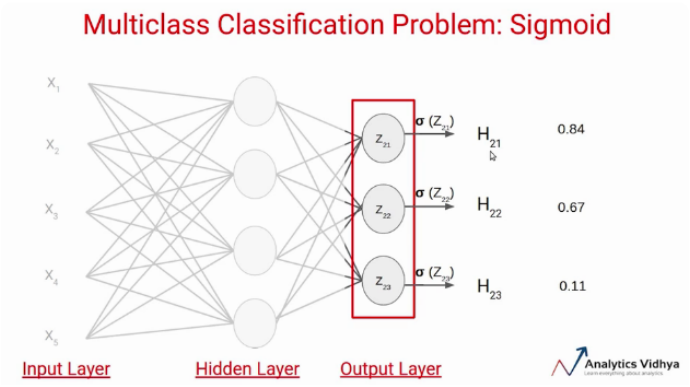
Normalization: After exponentiation, all the elements are summed up. This is a key step for ensuring that the probabilities add up to 1.

Probability Calculation: Each exponentiated value from step 2 is then divided by the sum obtained in step 3. This process normalizes the values, forcing them between 0 and 1. The cross-entropy loss function often uses these probabilities to measure a classifier's performance.

Output: The result is a new vector the same size as the input vector z. However, each element in the output vector now represents a probability between 0 and 1. The argmax function is typically used to select the index of the highest probability, determining the predicted class generalization.

Why Not Sigmoid?

Suppose we calculate the Z value using weights and biases of this layer and apply the sigmoid activation function over these values. We know that the **sigmoid activation function** gives the value between 0 and 1 suppose these are the values we get as output.



There are two problems in this case-

First, if we apply a threshold of 0.5, this network says the input data point belongs to two classes. Secondly, these probability values are independent of each other. That means the probability that the data point belongs to class 1 does not consider the probability of the other two classes.

The sigmoid activation function is not preferred in multi-class classification problems.

Using Softmax Activation Function in Output

In the above example, we will use the Softmax activation function in the output layer instead of sigmoid. The Softmax activation function calculates relative probabilities. That means it uses the values of Z21, Z22, and Z23 to determine the final probability value.

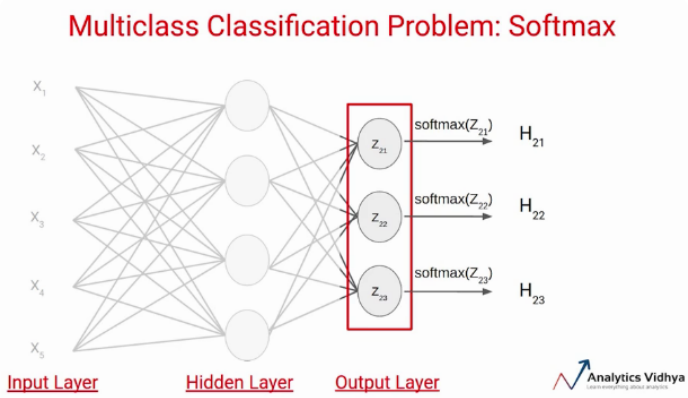
Let's see how the softmax activation function works. Like the sigmoid activation function, the SoftMax function in machine learning returns the probability of each class. Here is the equation for the SoftMax activation function.

$$softmax(z_i) = \frac{exp(z_i)}{\sum_j exp(z_j)}$$

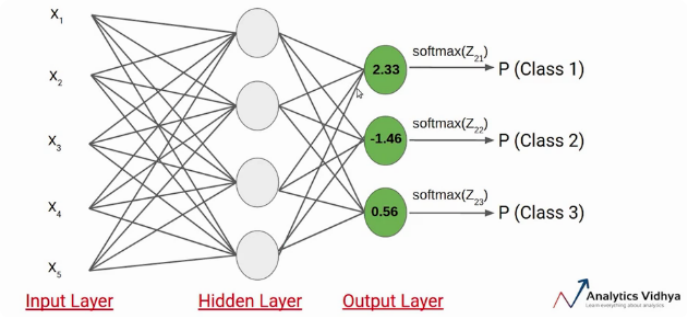
Here, Z represents the values from the neurons of the output layer. The exponential acts as the non-linear function. Later, these values are divided by the sum of exponential values to normalize them and then convert them into probabilities.

Note that, when the number of classes is two, it becomes the same as the sigmoid activation function. In other words, sigmoid is simply a variant of the Softmax function. To learn more about this concept, refer to **this link**.

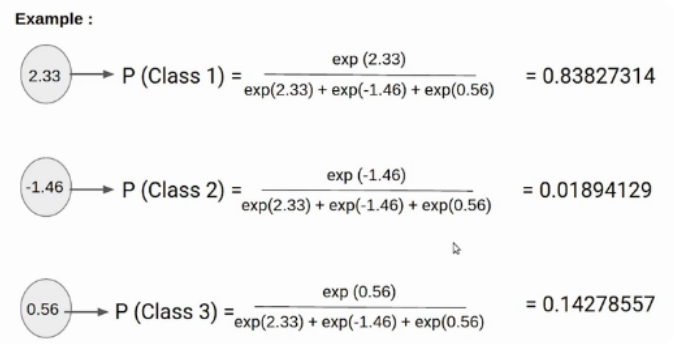
Let's understand with a simple example how the softmax function works. We have the following neural network.



Suppose the values of Z21, Z22, and Z23 are 2.33, -1.46, and 0.56, respectively. Now, the SoftMax activation function is applied to each of these neurons, and the following values are generated.



These are the probability values that a data point belonging to the respective classes. Note that, in this case, the sum of the probabilities is equal to 1.



In this case, the input belongs to class 1. So, if the probability of any of these classes is changed, the probability value of the first class would also change.

Why is Softmax Function Useful in CNN?

- The Softmax function allows **CNNs** to output a probability distribution over the possible classes. This is important because it will enable CNN to make more accurate predictions.
- The softmax activation function in machine learning first normalizes the input vector so that all numbers in the vector are equal to 1. Then, it exponentiates each number in the vector and divides by the sum of all the exponentiated numbers. This results in a vector of probabilities, where each probability is between 0 and 1 and represents the probability that the input belongs to a particular class.
- The probability distribution output by the softmax function can then be used to make a more accurate prediction about the class of an input image. For example, if the CNN predicts whether an image contains a cat or a dog, the probability distribution can indicate how likely the image contains a cat and how likely it is that the image contains a dog.

When to Use Softmax Activation Function vs ReLU?

Softmax Function is typically used in the last layer of a neural network to predict the class of an input image. It is also used in other applications, such as natural language processing and machine translation.

ReLU is typically used in the hidden layers of a neural network to add non-linearity. It is efficient and can help neural networks learn more complex relationships between the input and output data.

Why is Softmax used in CNN?

Here is how Softmax function in machine learning is used in CNN :

- **CNN Processes Image:** The CNN takes an image as input and performs various convolutional and pooling operations to extract features.
- **Final Layer Generates Logits:** After processing, the final layer of the CNN outputs a set of numbers called logits. These logits represent the raw scores or activation levels for each class the CNN can classify. There will be one logit for each class.
- **Softmax Takes Over:** The Softmax function takes these logits as input.
- **Exponentiation:** Softmax activation function applies an exponent function (often $exex$) to each logit value. This emphasizes the differences between the logits, making the higher-scoring classes stand out more.
- **Normalization:** Softmax function in machine learning then divides each exponentiated value by the sum of all the exponentiated values, ensuring the final outputs add up to 1.
- **Probability Distribution:** The result is a vector of numbers between 0 and 1, representing probabilities. Each value corresponds to the probability of the image belonging to a specific class.
- **Decision and Interpretation:** The class with the highest probability value is CNN's predicted class. This probability value also reflects CNN's confidence level in its prediction.

In machine learning, functions like softmax output are implemented in frameworks such as numpy and Python to facilitate the process. The softmax function, through exponentiation, transforms the logits into a probability

distribution. This method is crucial in determining the loss function during model training and optimization. CNN's ability to make precise predictions hinges on these fundamental principles.

Conclusion

This article is all about the SoftMax activation function. In it, we saw why we should not use activation functions like sigmoid or threshold in multiclass classification problems and how the SoftMax function works through an example. In this article, these algorithms will show different SoftMax output values and different output vectors.

If you want to kick-start your Data Science Journey and want every topic under one roof, your search stops here. Check out Analytics Vidhya's Certified AI & ML BlackBelt Plus Program

Python Softmax | Soft Max Function | Softmax | Softmax Layer | Softmax Loss Function | What Is Softmax

S shipra_saxena
07 Aug, 2024

Advanced | Deep Learning | Videos

Frequently Asked Questions

What is the softmax function?

The softmax activation function is a mathematical function that converts a vector of real numbers into a probability distribution. It exponentiates each element, making them positive, and then normalizes them by dividing by the sum of all exponentiated values. This ensures that the output probabilities add up to one, making it suitable for multiclass classification tasks.

Q2. What is the difference between sigmoid and softmax functions?

Responses From Readers