1. What is the function of a summation junction of a neuron? What is threshold activation function?

Activation function is one of the building blocks on Neural Network

Learn about the different activation functions in deep learning

Code activation functions in python and visualize results in live coding window

This article was originally published in October 2017 and updated in January 2020 with three new activation functions and python codes.

Introduction

The Internet provides access to plethora of information today. Whatever we need is just a Google (search) away. However, when we have so much information, the challenge is to segregate between relevant and irrelevant information.

When our brain is fed with a lot of information simultaneously, it tries hard to understand and classify the information into “useful” and “not-so-useful” information. We need a similar mechanism for classifying incoming information as “useful” or “less-useful” in case of Neural Networks.

2.What is a step function? What is the difference of step function with threshold function?

A step function is a function like that used by the original Perceptron. The output is a certain value, A1, if the input sum is above a certain threshold and A0 if the input sum is below a certain threshold. The values used by the Perceptron were A1 = 1 and A0 = 0.

Binary step function is one of the most common activation function in neural networks out there. But before we get into it let's take a look at what activation functions and neural networks are.

Table of contents:

Brief overview of neural networks and activation functions

What is binary step function?

Conclusion

Brief overview of neural networks and activation functions

Neural networks are a powerful machine learning mechanism that mimic how the human brain learns. Perceptrons are the basic building blocks of a neural network. A perceptron can be defined as anything that takes multiple inputs and produces one output.

Activation functions are mathematical equations that determine the output of a neural network. They basically decide to deactivate neurons or activate them to get the desired output thus the name, activation functions.  
In a neural network, input data points are fed into the neuron. Each neuron has a respective weight which are multiplied by the inputs and added to a staic bias value(unique to each neuron).

x = (weight \* input) + bias

This is then passed to an apropriate activation function.

Y = Activation function(∑(weight\*input) + bias)

The output achieved will be again fed into the neurons in the next layer and the same process is repeated.

Activation functions can be categorized into three main categories:

Binary Step Function

Linear Activation Function

Non-Linear Activation functions

Sigmoid function

tanh function

ReLU function

to name a few.

3.Explain the McCulloch–Pitts model of neuron.

The McCulloch-Pitts neural model, which was the earliest ANN model, has only two types of inputs — Excitatory and Inhibitory. The excitatory inputs have weights of positive magnitude and the inhibitory weights have weights of negative magnitude. The inputs of the McCulloch-Pitts neuron could be either 0 or 1.

McCulloch-Pitts Neuron — Mankind’s First Mathematical Model Of A Biological Neuron

It is very well known that the most fundamental unit of deep neural networks is called an artificial neuron/perceptron. But the very first step towards the perceptron we use today was taken in 1943 by McCulloch and Pitts, by mimicking the functionality of a biological neuron.

Note: The concept, the content, and the structure of this article were directly taken from the awesome lectures and the material offered by Prof. [Mitesh M. Khapra](https://www.cse.iitm.ac.in/~miteshk/) on [NPTEL](http://nptel.ac.in/)’s [Deep Learning](https://onlinecourses.nptel.ac.in/noc18_cs41/preview) course

4.Explain the ADALINE network model.

MADALINE (Many ADALINE) is a three-layer (input, hidden, output), fully connected, feed-forward artificial neural network architecture for classification that uses ADALINE units in its hidden and output layers, i.e. its activation function is the sign function. The three-layer network uses memistors.

ADALINE (Adaptive Linear Neuron or later Adaptive Linear Element) is an early single-layer artificial neural network and the name of the physical device that implemented this network.[[1]](https://en.wikipedia.org/wiki/ADALINE#cite_note-1)[[2]](https://en.wikipedia.org/wiki/ADALINE#cite_note-2)[[3]](https://en.wikipedia.org/wiki/ADALINE#cite_note-3)[[4]](https://en.wikipedia.org/wiki/ADALINE#cite_note-4)[[5]](https://en.wikipedia.org/wiki/ADALINE#cite_note-5) The network uses memistors. It was developed by Professor Bernard Widrow and his doctorate student Ted Hoff at Stanford University in 1960. It is based on the McCulloch–Pitts neuron. It consists of a weight, a bias and a summation function.

The difference between Adaline and the standard (McCulloch–Pitts) perceptron is that in the learning phase, the weights are adjusted according to the weighted sum of the inputs (the net). In the standard perceptron, the net is passed to the activation (transfer) function and the function's output is used for adjusting the weights.

A multilayer network of ADALINE units is known as a MADALINE

contents

[1Definition](https://en.wikipedia.org/wiki/ADALINE#Definition)

[2Learning algorithm](https://en.wikipedia.org/wiki/ADALINE#Learning_algorithm)

[3MADALINE](https://en.wikipedia.org/wiki/ADALINE#MADALINE)

[4See also](https://en.wikipedia.org/wiki/ADALINE#See_also)

[5References](https://en.wikipedia.org/wiki/ADALINE#References)

[6External links](https://en.wikipedia.org/wiki/ADALINE#External_links)

Definition[[edit](https://en.wikipedia.org/w/index.php?title=ADALINE&action=edit&section=1)]

Adaline is a single layer neural network with multiple nodes where each node accepts multiple inputs and generates one output. Given the following variables as:

{\displaystyle x} is the input vector

{\displaystyle w} is the weight vector

{\displaystyle n} is the number of inputs

{\displaystyle \theta } some constant

{\displaystyle y} is the output of the model

then we find that the output is {\displaystyle y=\sum \_{j=1}^{n}x\_{j}w\_{j}+\theta }. If we further assume that

{\displaystyle x\_{0}=1}

{\displaystyle w\_{0}=\theta }

then the output further reduces to: {\displaystyle y=\sum \_{j=0}^{n}x\_{j}w\_{j}}

Learning algorithm[[edit](https://en.wikipedia.org/w/index.php?title=ADALINE&action=edit&section=2)]

Let us assume:

{\displaystyle \eta } is the [learning rate](https://en.wikipedia.org/wiki/Learning_rate) (some positive constant)

{\displaystyle y} is the output of the model

{\displaystyle o} is the target (desired) output

then the weights are updated as follows {\displaystyle w\leftarrow w+\eta (o-y)x}. The ADALINE converges to the least squares error which is {\displaystyle E=(o-y)^{2}}.[[6]](https://en.wikipedia.org/wiki/ADALINE#cite_note-6) This update rule is in fact the [stochastic gradient descent](https://en.wikipedia.org/wiki/Stochastic_gradient_descent) update for [linear regression](https://en.wikipedia.org/wiki/Linear_regression).

5.What is the constraint of a simple perceptron? Why it may fail with a real-world data set?

Perceptron networks should be trained with adapt, which presents the input vectors to the network one at a time and makes corrections to the network based on the results of each presentation. Use of adapt in this way guarantees that any linearly separable problem is solved in a finite number of training presentations. Perceptrons can also be trained with the function train, which is presented in the next chapter. When train is used for perceptrons, it presents the inputs to the network in batches, and makes corrections to the network based on the sum of all the individual corrections. Unfortunately, there is no proof that such a training algorithm converges for perceptrons. On that account the use of train for perceptrons is not recommended.

Perceptron networks have several limitations. First, the output values of a perceptron can take on only one of two values (0 or 1) due to the hard-limit transfer function. Second, perceptrons can only classify linearly separable sets of vectors. If a straight line or a plane can be drawn to separate the input vectors into their correct categories, the input vectors are linearly separable. If the vectors are not linearly separable, learning will never reach a point where all vectors are classified properly. Note, however, that it has been proven that if the vectors are linearly separable, perceptrons trained adaptively will always find a solution in finite time. You might want to try demop6. It shows the difficulty of trying to classify input vectors that are not linearly separable.

It is only fair, however, to point out that networks with more than one perceptron can be used to solve more difficult problems. For instance, suppose that you have a set of four vectors that you would like to classify into distinct groups, and that two lines can be drawn to separate them. A two neuron network can be found such that its two decision boundaries classify the inputs into four categories.

6.What is linearly inseparable problem? What is the role of the hidden layer?

Over the past few years, neural network architectures have revolutionized many aspects of our life with applications ranging from self-driving cars to predicting deadly diseases. Generally, every neural network consists of vertically stacked components that are called layers. There are three types of layers:

An Input Layer that takes as input the raw data and passes them to the rest of the network.

One or more Hidden Layers that are intermediate layers between the input and output layer and process the data by applying complex non-linear functions to them. These layers are the key component that enables a neural network to learn complex tasks and achieve excellent performance.

An Output Layer that takes as input the processed data and produces the final results.

Below we can see a simple [feedforward neural network](https://en.wikipedia.org/wiki/Feedforward_neural_network) with two hidden layers:

In the above neural network, each neuron of the first hidden layer takes as input the three input values and computes its output as follows:  
  
where  are the input values,  the weights,  the bias and  an activation function. Then, the neurons of the second hidden layer will take as input the outputs of the neurons of the first hidden layer and so on.

Now let’s discuss the importance of hidden layers in neural networks. As mentioned earlier, hidden layers are the reason why neural networks are able to capture very complex relationships and achieve exciting performance in many tasks.

Based on the previous equation, the output value  is equal to a linear combination along with a non-linearity. Therefore, the model is similar to a [linear regression model](https://en.wikipedia.org/wiki/Linear_regression). As we already know, a linear regression attempts to fit a linear equation to the observed data.

In most machine learning tasks, a linear relationship is not enough to capture the complexity of the task and the linear regression model fails. Here comes the importance of the hidden layers that enables the neural network to learn very complex non-linear functions.

7.Explain XOR problem in case of a simple perceptron.

The XOr problem is that we need to build a Neural Network (a perceptron in our case) to produce the truth table related to the XOr logical operator. This is a binary classification problem. Hence, supervised learning is a better way to solve it. In this case, we will be using perceptrons.

The XOR Problem

The XOR, or “exclusive or”, problem is a classic problem in ANN research. It is the problem of using a neural network to predict the outputs of XOR logic gates given two binary inputs. An XOR function should return a true value if the two inputs are not equal and a false value if they are equal

XOR is a [classification problem](https://en.wikipedia.org/wiki/Statistical_classification) and one for which the expected outputs are known in advance. It is therefore appropriate to use a supervised learning approach.

On the surface, XOR appears to be a very simple problem, however, Minksy and Papert (1969) showed that this was a big problem for neural network architectures of the 1960s, known as perceptrons.

Perceptrons

Like all ANNs, the perceptron is composed of a network of [\*units](https://en.wikipedia.org/wiki/Artificial_neuron)\*, which are analagous to biological neurons. A unit can receive an input from other units. On doing so, it takes the sum of all values received and decides whether it is going to forward a signal on to other units to which it is connected. This is called activation. The [activation function](https://en.wikipedia.org/wiki/Activation_function) uses some means or other to reduce the sum of input values to a 1 or a 0 (or a value very close to a 1 or 0) in order to represent activation or lack thereof. Another form of unit, known as a bias unit, always activates, typically sending a hard coded 1 to all units to which it is connected.

Perceptrons include a single layer of input units — including one bias unit — and a single output unit (see figure 2). Here a bias unit is depicted by a dashed circle, while other units are shown as blue circles. There are two non-bias input units representing the two binary input values for XOR. Any number of input units can be included.

8.Design a multi-layer perceptron to implement A XOR B.

The perceptron is a classification algorithm. Specifically, it works as a linear binary classifier. It was invented in the late 1950s by Frank Rosenblatt.

The perceptron basically works as a threshold function — non-negative outputs are put into one class while negative ones are put into the other class.

Though there’s a lot to talk about when it comes to neural networks and their variants, we’ll be discussing a specific problem that highlights the major differences between a single layer perceptron and one that has a few more layers

Among various logical gates, the XOR or also known as the “exclusive or” problem is one of the logical operations when performed on [binary](https://analyticsindiamag.com/a-hands-on-guide-to-linear-discriminant-analysis-for-binary-classification/) inputs that yield output for different combinations of input, and for the same combination of input no output is produced. The outputs generated by the XOR logic are not [linearly](https://analyticsindiamag.com/a-guide-to-quadratic-approximation-with-logistic-regression/) separable in the hyperplane. So  In this article let us see what is the XOR logic and how to integrate the XOR logic using [neural](https://analyticsindiamag.com/kernel-regularizers-with-neural-networks/) networks.

9.Explain the single-layer feed forward architecture of ANN.

The output layer is formed when different weights are applied to input nodes and the cumulative effect per node is taken. After this, the neurons collectively give the output layer to compute the output signals

Processing of ANN depends upon the following three building blocks −

Network Topology

Adjustments of Weights or Learning

Activation Functions

In this chapter, we will discuss in detail about these three building blocks of ANN

Network Topology

A network topology is the arrangement of a network along with its nodes and connecting lines. According to the topology, ANN can be classified as the following kinds −

Feedforward Network

It is a non-recurrent network having processing units/nodes in layers and all the nodes in a layer are connected with the nodes of the previous layers. The connection has different weights upon them. There is no feedback loop means the signal can only flow in one direction, from input to output. It may be divided into the following two types −

Single layer feedforward network − The concept is of feedforward ANN having only one weighted layer. In other words, we can say the input layer is fully connected to the output layer.

Multilayer feedforward network − The concept is of feedforward ANN having more than one weighted layer. As this network has one or more layers between the input and the output layer, it is called hidden layers.

Feedback Network

As the name suggests, a feedback network has feedback paths, which means the signal can flow in both directions using loops. This makes it a non-linear dynamic system, which changes continuously until it reaches a state of equilibrium. It may be divided into the following types −

Recurrent networks − They are feedback networks with closed loops. Following are the two types of recurrent networks.

Fully recurrent network − It is the simplest neural network architecture because all nodes are connected to all other nodes and each node works as both input and output.

Jordan network − It is a closed loop network in which the output will go to the input again as feedback as shown in the following diagram.

Adjustments of Weights or Learning

Learning, in artificial neural network, is the method of modifying the weights of connections between the neurons of a specified network. Learning in ANN can be classified into three categories namely supervised learning, unsupervised learning, and reinforcement learning.

Supervised Learning

As the name suggests, this type of learning is done under the supervision of a teacher. This learning process is dependent.

During the training of ANN under supervised learning, the input vector is presented to the network, which will give an output vector. This output vector is compared with the desired output vector. An error signal is generated, if there is a difference between the actual output and the desired output vector. On the basis of this error signal, the weights are adjusted until the actual output is matched with the desired output.

Unsupervised Learning

As the name suggests, this type of learning is done without the supervision of a teacher. This learning process is independent.

During the training of ANN under unsupervised learning, the input vectors of similar type are combined to form clusters. When a new input pattern is applied, then the neural network gives an output response indicating the class to which the input pattern belongs.

There is no feedback from the environment as to what should be the desired output and if it is correct or incorrect. Hence, in this type of learning, the network itself must discover the patterns and features from the input data, and the relation for the input data over the output.

Reinforcement Learning

As the name suggests, this type of learning is used to reinforce or strengthen the network over some critic information. This learning process is similar to supervised learning, however we might have very less information.

During the training of network under reinforcement learning, the network receives some feedback from the environment. This makes it somewhat similar to supervised learning. However, the feedback obtained here is evaluative not instructive, which means there is no teacher as in supervised learning. After receiving the feedback, the network performs adjustments of the weights to get better critic information in future.

Activation Functions

It may be defined as the extra force or effort applied over the input to obtain an exact output. In ANN, we can also apply activation functions over the input to get the exact output. Followings are some activation functions of interest −

Linear Activation Function

It is also called the identity function as it performs no input editing. It can be defined as −

F(x)=xF(x)=x

Sigmoid Activation Function

It is of two type as follows −

Binary sigmoidal function − This activation function performs input editing between 0 and 1. It is positive in nature. It is always bounded, which means its output cannot be less than 0 and more than 1. It is also strictly increasing in nature, which means more the input higher would be the output. It can be defined as

F(x)=sigm(x)=11+exp(−x)F(x)=sigm(x)=11+exp(−x)

Bipolar sigmoidal function − This activation function performs input editing between -1 and 1. It can be positive or negative in nature. It is always bounded, which means its output cannot be less than -1 and more than 1. It is also strictly increasing in nature like sigmoid function. It can be defined as

F(x)=sigm(x)=21+exp(−x)−1=1−exp(x)1+exp(x)

10.Explain the competitive network architecture of ANN.

Neural Network is a series of algorithms that are trying to mimic the human brain and find the relationship between the sets of data. It is being use in various use-cases like in regression, classification, Image Recognition and many more.

As we have talked above that [neural](https://blog.knoldus.com/convolutional-neural-network-in-tensorflow/) networks tries to mimic the human brain then there might be the difference as well as the similarity between them.

Some major differences between them are biological neural network does parallel processing whereas the Artificial neural network does series processing also in the former one processing is slower (in millisecond) while in the latter one processing is faster (in a nanosecond).

Architecture of ANN

A neural network consists of three layers. The first layer is the input layer. It contains the input neurons that send information to the hidden layer. The hidden layer performs the computations on input data and transfers the output to the output layer. It includes weight, activation function, cost function.

The connection between neurons is known as weight, which is the numerical values. The weight between neurons determines the learning ability of the neural network. During the learning of artificial neural networks, weight between the neuron changes.

Working of ANN

Firstly, the information is feed into the input layer. Which then transfers it to the hidden layers, and interconnection between these two layers assign weights to each input randomly at the initial point. Then bias is add to each input neuron and after this, the weight sum which is a combination of weights and bias is pass through the activation function. Activation Function has the responsibility of which node to fire for feature extraction and finally output is calculate. Therefore this whole process is known as Forward Propagation. After getting the output model to compare it with the original output and the error is known and finally, weights are updates in backward propagation to reduce the error and this process continues for a certain number of epochs (iteration). Finally, model weights get updates and prediction is done.

Some Merits of ANN

It has a parallel processing ability. It has the numerical strength that performs more than one task at the same time.

After training, ANN can infer unseen relationships from unseen data, and hence it is generalise.

Unlike many machine learning models, ANN does not have restrictions on datasets like data should be Gaussian distribute or nay other distribution.

Applications of ANN

There are many applications of ANN. Some of them are :

Medical

We can use it in detecting cancer cells and analysing the MRI images to give detailed results.

Forecast

We can use it in every field of business decisions like in finance and the stock market, in economic and monetary policy.

Image Processing

We can use satellite imagery processing for agricultural and defense use

11.Consider a multi-layer feed forward neural network. Enumerate and explain steps in the backpropagation algorithm used to train the network.

Backpropagation:

Backpropagation is a supervised learning algorithm, for training Multi-layer Perceptrons (Artificial Neural Networks).

I would recommend you to check out the following [Deep Learning Certification](https://www.edureka.co/ai-deep-learning-with-tensorflow) blogs too:

[What is Deep Learning?](https://www.edureka.co/blog/what-is-deep-learning)

[Deep Learning Tutorial](https://www.edureka.co/blog/deep-learning-tutorial)

[TensorFlow Tutorial](https://www.edureka.co/blog/tensorflow-tutorial/)

[Neural Network Tutorial](https://edureka.co/blog/neural-network-tutorial)

But, some of you might be wondering why we need to train a Neural Network or what exactly is the meaning of training.

Why We Need Backpropagation?

While designing a Neural Network, in the beginning, we initialize weights with some random values or any variable for that fact.

Now obviously, we are not superhuman. So, it’s not necessary that whatever weight values we have selected will be correct, or it fits our model the best.

Okay, fine, we have selected some weight values in the beginning, but our model output is way different than our actual output i.e. the error value is huge.

Now, how will you reduce the error?

Basically, what we need to do, we need to somehow explain the model to change the parameters (weights), such that error becomes minimum.

Let’s put it in an another way, we need to train our model.

12.What are the advantages and disadvantages of neural networks?

What are the Advantages of Neural Networks?

The approximation theorem and various other mathematical tools are the basis of complex computer algorithms. However, the programmer needs a computer understandable set up of instructions to command it for tasks. It can be designed to perform the task quite conveniently. The key advantages of neural networks are as follows.

1. Efficiency

Unlike humans, a machine doesn't get tired if it runs within well-specified limits. Also, it can work continuously, saving a lot of time producing more remarkable results. If programmed correctly, a machine can complete a task quickly, which can take humans longer hours.

2. Continuous Learning

A neural network is designed to learn and improve its results continuously. Once the system is trained, it can produce output without the need for complete inputs. With the use, the program or applications become more user-friendly.

3. Data retrieval

The most crucial benefit of using cloud services or online data management is its retrieval. If the hardware suffers any damages or failure, the whole program backup is available online. So there is no need to provide training to the system again and again.

4. Multitasking is one of the common advantages of Neural Networks

New advanced programs are designed in a way that is capable of producing multiple results and multitasking. The user is free to scroll through different tasks at the same time. It is not possible in simpler networks or programs.

5. Wide Applications

Neural Networks are designed to make machines work like humans, and hence the replacement comes with many advantages of Neural Networks along with numerous applications. Medical, engineering, mining, agriculture, etc., can find numerous benefits from security to day-to-day tasks using this technology.

What are the Disadvantages of Neural Networks?

These algorithms are designed to recognize preferences and leave the unimportant ones to determine the output. These preferences can differ at different times resulting in a different decision. A computer-dependent decision is based on a fraction of essential qualities/values/requirements at a given time. These approximate results may lead to wrong decisions. Due to its complex nature, there are several disadvantages of Neural Networks that need to rectified.

1. Hardware dependent

Although the data is stored online, artificial networks still require hardware to create them in the first place. The hardware cost increases with the complexity of the problem, and its setup requires additional efforts to maintain them.

2. Complex Algorithms are foreseen disadvantages of Neural Networks

All the programming needed to be done initially requires lengthy and complex programs to be written. For example, it may require months to create an algorithm capable of working a specified task.

3. Black Box Nature

Even when the results are accurate human analysts can't track and check the derivations. Most neural networks are black-box systems generating results based on experience and not on specified programs, making it difficult for modifications.

4. Approximate Results

Various theorems are used to give only a probable value. All the theories used are not entirely suitable to give results possible for all situations, and the desired output may not be obtained. This uncertainty is among the eye-opening problems with Neural Networks.

5. Data-dependency

Whatever data is fed to the machine, it acts accordingly. The more amount of data is used during training, the more accurate the results are. Dependency on data is one of the leading disadvantages of Neural Networks, as some have to be on the maintenance side to watch it. Since there are errors in the data, the result will be faulty, which poses serious threats.

13.Write short notes on any two of the following:

* + 1. Biological neuron
    2. ReLU function
    3. Single-layer feed forward ANN
    4. Gradient descent
    5. Recurrent networks

Biological Neurons

Typical biological neurons are individual cells, each composed of the main body of the cell along with many tendrils that extend from that body. The body, or soma, houses the machinery for maintaining basic cell functions and energy processing (e.g., the DNA-containing nucleus, and organelles for building proteins and processing sugar and oxygen). There are two types of tendrils: dendrites, which receive information from other neurons and bring it to the cell body, and axons, which send information from the cell body to other neurons.

ReLU function

The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

In this tutorial, you will discover the rectified linear activation function for deep learning neural networks.

After completing this tutorial, you will know:

The sigmoid and hyperbolic tangent activation functions cannot be used in networks with many layers due to the vanishing gradient problem.

The rectified linear activation function overcomes the vanishing gradient problem, allowing models to learn faster and perform better.

The rectified linear activation is the default activation when developing multilayer Perceptron and convolutional neural networks.

Single layer feed forward ANN

A single-layer neural network represents the most simple form of neural network, in which there is only one layer of input nodes that send weighted inputs to a subsequent layer of receiving nodes, or in some cases, one receiving node. This single-layer design was part of the foundation for systems which have now become much more complex.

Gradient descent

Gradient descent is an optimization algorithm which is commonly-used to train machine learning models and neural networks. Training data helps these models learn over time, and the cost function within gradient descent specifically acts as a barometer, gauging its accuracy with each iteration of parameter updates.

Recurrent networks

Apple’s Siri and Google’s voice search both use Recurrent Neural Networks (RNNs), which are the state-of-the-art method for sequential data. It’s the first algorithm with an internal memory that remembers its input, making it perfect for problems involving sequential data in machine learning. It’s one of the algorithms responsible for the incredible advances in deep learning over the last few years. In this article, we’ll go over the fundamentals of recurrent neural networks, as well as the most pressing difficulties and how to address them.

Introduction on Recurrent Neural Networks

A Deep Learning approach for modelling sequential data is Recurrent Neural Networks (RNN). RNNs were the standard suggestion for working with sequential data before the advent of attention models. Specific parameters for each element of the sequence may be required by a deep feedforward model. It may also be unable to generalize to variable-length sequences.