Artificial Neural Networks

Introduction to Neural Networks

In computer science, artificial neural networks, also referred to as ANNs, are a computational information processing approach inspired by the mammalian nervous system. ANNs consist of a large number of interconnected elementary processing units called artificial neurons, whose structure and functionality are modeled after the biological neurons.(1)

In comparison to conventional problem solving methods, artificial neural networks offer an adaptive approach to solve various problems through their ability to learn. Conventional algorithms usually focus on solving specific problems using a set of instructions. Therefore in order to solve a problem using these systems, a certain solution needs to be known and implemented.

In contrast, artificial neural networks do not rely on a complex processing system but a number of interconnected neurons, which operate in parallel according to the input given. While they are not constructed to solve a specific problem, they can be trained by using a set of examples. Throughout this learning process the network adjusts and adapts itself similar to the human learning process.

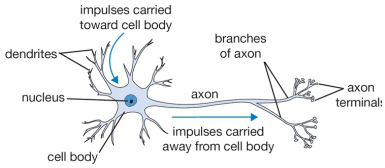


Figure 1: *the structure of a basic biological neuron*(6)

Inspiration

In biology, a nervous system consists of a number of specialized cells called neurons that communicate with each other through a number of complex connections. These cells have the ability to transmit information utilizing electrical and chemical signals and this communication process is called neurotransmission.

In a nervous system, this connectivity between neurons creates a structure called a neural network in which a neuron is linked to other neurons by its dendrites. These dendrites serve as a receptor and listen to incoming signals from other neurons. Depending on this input signal, the cell body may send an impulse through its axon to transmit a new signal to other neurons using the axon terminals, which are received by dendrites of other neurons.

The structure of an artificial neuron, illustrated in figure 2, was based on this concept(2). Similar to a biological neuron, a simple processing unit receives a signal and generates an output depending on the input.

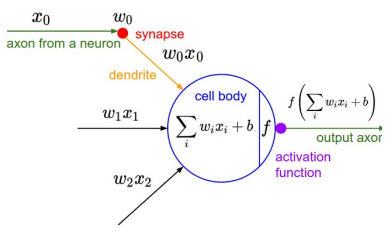


Figure 2: *the structure of a single artificial neuron*(6)

In more detail, this process is carried out in 4 steps:

- Receiving information: the processing unit obtains the information as input $x_1, x_2, \ldots, x_n$.
- Weighting: each input is weighted by its corresponding weights denoted as $w_0, w_1, w_2, \ldots, w_n$.
- Activation: an activation function f is applied to the sum of all the weighted inputs z.
- Output: an output is y generated depending on z.

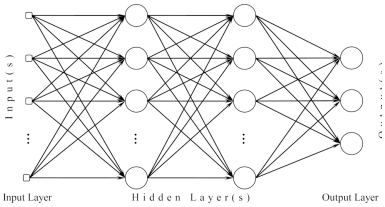


Figure 3: *the structure of a fully connected 3-Layer neural network (image source)*

$$y = f(z) \quad \text{where} \quad z = \sum_{i=1}^{n} w_i x_i + w_0$$

The Structure of Neural Networks

Neural networks consist of a number interconnected neurons. Depending on their inputs and outputs, these neurons are generally arranged into three different layers as illustrated in figure 3.

- Input Layer: The input layer is the first layer in an artificial neural network and it is dimensioned according to the input. It serves as an interface between the data and the network.
- Hidden Layer(s): The hidden layer(s) consist of a number of connected neurons, which allow the information to be processed and transmitted from the input layer, between hidden layers and ultimately to the output layer.
- Output Layer: The output layer is the last layer in the network and similar to the input layer it is dimensioned to fit the proper output. It represents the result of the information process.

Depending on the connectivity between the neurons in the hidden layer, the type of the neural network can be divided into two categories, namely Feed-Forward and Feedback Networks.

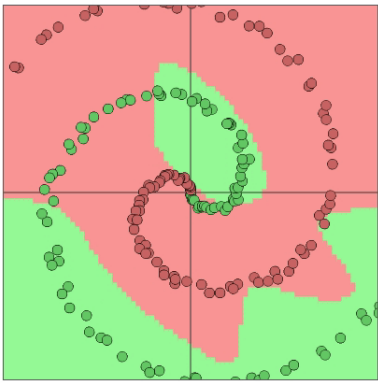


Figure 4: *the illustration of a neural network solving a two dimensional classification problem with two distinguishable classes* [7]

Feed-Forward Networks

In a feed-forward neural network, the information only flows in one direction, from input to output. The output of each neuron in the hidden layer is passed to another neuron as an input in the next layer. Since there are no possible loops within the structure, the information process is finished once it reaches the output layer.

Feedback Networks

In a feedback network, the information can travel in both directions, which means a neuron can be linked to any neuron within the hidden layer regardless of the layer hierarchy. This structure allows the network to have loops, where the information can flow continuously and update the network until it reaches a state of equilibrium.

Applications

The learning capabilities of artificial neural networks along with their flexibility make them an attractive approach for many tasks. Nowadays, they are used in a wide spectrum of areas including but not limited to Pattern Recognition, Classification, Function Approximation and Regression, Signal Processing, Time Series Estimation.

Multilayer Perceptrons

A multilayer perceptron, also referred to as MLP, is a feed-forward artificial neural network and it consist of multiple layers, which are fully connected to each other. In other words, every single neuron in a multilayer perceptron is linked to the neurons in the next layer.

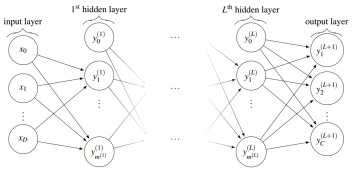


Figure 5 demonstrates the general structure of a $(L+1)$-Layer perceptron which consists of one input layer, L hidden layers and one output layer. The number of neurons in the input layers is denoted by D and the number of neurons in the output layer is denoted by C. Using this notation the output of the ith neuron in lth layer can be computed as:

$$y_i^l = f(z_i^l) \quad \text{where} \quad z_i^l = \sum_{k=1}^{m^{(l-1)}} w_{i,k}^l y_k^{l-1} + w_{i,0}^l$$

Figure 5: *the illustration of a fully connected* $(L+1)$*-layer perceptron with* D *inputs and* C *output neurons* [(3)]

where the m_l denotes the number of neurons in the lth hidden layer. The activation function is denoted by f and weights are represented as w where $w_{i,k}^{l}$ is the weight from the kth neuron in $(l-1)$th layer to the ith neuron in the lth layer. $w_{i,0}^{l}$ serves as the external input called the bias for the ith unit in lth layer.[(4)]

In a more compact form, the functionality of a multilayer perceptron, which maps D inputs into C outputs, can be formulated as[(3)]:

$$y(.,w) : \mathbb{R}^D \to \mathbb{R}^C, x \mapsto y(x,w)$$

The training of these networks is considered to be a challenging task, especially in deep neural networks. They require proper fine-tuning and initilization and must be designed carefully.

Activation Functions

The activation function F of a neuron represents the function which maps its weighted inputs into an output and as a result determines its behavior.

$$y = f(z) \quad \text{where} \quad z = \sum_{i=1}^{n} w_i x_i + w_0$$

There are many different forms of possible activation functions for neural networks and this section covers some of the most preferred ones briefly introducing their advantages and disadvantages.

Threshold Function

A threshold function is a function which generates the output 1 if the input exceeds a certain value n. If this criteria is not fulfilled, this function takes the value 0.

A common threshold function is the Heaviside step function ($n = 0$) which can be defined as:

$$H(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$$

This function does not offer a good solution since it does not fulfill the requirements of the backpropagation process of a neuron.

Sigmoid Function

The sigmoid function is one of the most widely used activation functions and brings a solution to the problem mentioned above. It generates an S-shaped curve, which maps its inputs into the interval of values of $]0,1[$. The sigmoid function has a non-zero gradient throughout its definition region, which makes it a good candidate when utilizing the backpropagation algorithm.

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

On the other hand, using a sigmoid function has its own drawbacks. One of the main problems encountered while using a sigmoid activation function is called the vanishing gradient problem. At each step of backpropagation, the weights are updated according to the gradient of the error function with respect to the weights. If the neuron gets saturated, the gradient can become significantly small, preventing a successful training process.

It is also important to note, that the output values of a sigmoid function are not zero centered. As a result, the gradient values at each step of the backpropagation process can only be all negative or all positive depending on the gradient of the error. This problem can decelerate the training process of a neural network.

Hyperbolic Tangent Function

Similar to the sigmoid function, a hyperbolic tangent function (a $tanh$ function) generates a S-shaped curve but it maps its inputs to values $]-1, 1[$. Even though, a $tanh$ neuron still suffers from the vanishing gradient problem, it generates zero-centered outputs and eliminates the problem mentioned above.

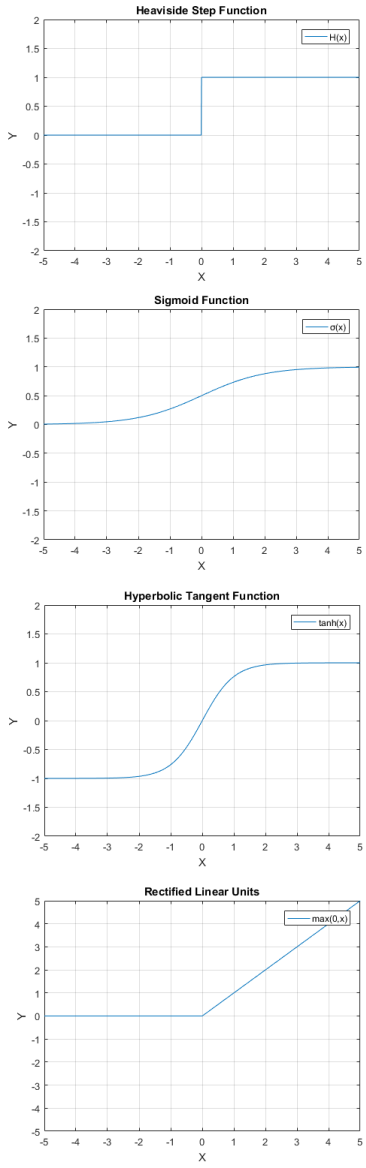
$$f(x) = tanh(x)$$

Rectified Linear Units

The rectified linear units, also referred to as ReLUs, utilize a specialized ramp function for the activation, which is formulated as:

$$f(x) = max(0, x)$$

Since they use a computationally less complex ramp function for the activation, these units are considered as an useful alternative to the sigmoid and $tanh$ function, particularly in more complex structures featuring millions of neurons (such as convolutional neural networks[(5)]). More detail about the advantages of rectified linear units in these structures can be found in section "Layers of a convolutional neural network".



Literature

[1] Research paper on basic of Artificial Neural Network (2014, Sonali B. Maind and Priyanka Wankar)

[2] The perceptron, a perceiving and recognizing automaton (1957, Frank Rosenblatt)

[3] Understanding convolutional neural networks (2014, David Stutz)

[4] Neural networks for pattern recognition (1995, Christopher M. Bishop)

[5] Imagenet classification with deep convolutional neural networks (2012, Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton)

Weblinks

[6] http://cs231n.github.io/neural-networks-1/ / (Last visited: 27.01.2017)

[7] http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html / (Last visited: 27.01.2017)