Neural Network

Shuvrajeet Das

May 12, 2021

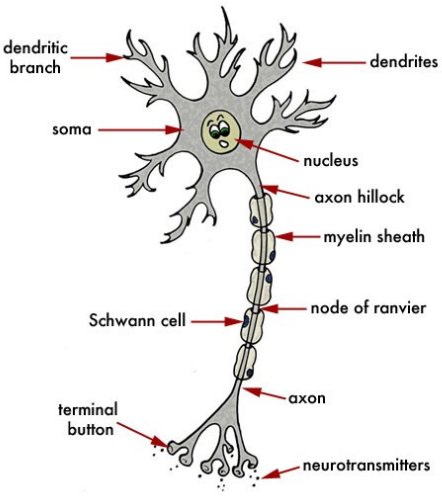
Abstract

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature.

1 Introduction

The human brain consists of neurons or nerve cells which transmit and process the information received from our senses. Many such nerve cells are arranged together in our brain to form a network of nerves. These nerves pass electrical impulses i.e the excitation from one neuron to the other.
The dendrites receive the impulse from the terminal button or synapse of an adjoining neuron. Dendrites carry the impulse to the nucleus of the nerve cell which is also called as soma. Here , the electrical impulse is processed and then passed on to the axon. The axon is longer branch among the dendrites which carries

Figure 1: Single Neuron



the impulse from the soma to the synapse. The synapse then , passes the impulse to dendrites of the second neuron. Thus, a complex network of neurons is created in the human brain.

2 Application

Neural networks can be used in different fields. The tasks to which artificial neural networks are applied tend to fall within the following broad categories:

1. Function approximation, or regression analysis, including time series prediction and modeling.

2. Classification, including pattern and sequence recognition, novelty detection and sequential decision making.

3. Data processing, including filtering, clustering, blind signal separation and compression.

Application areas of ANNs include nonlinear system identification and control (vehicle control, process control), game-playing and decision making (backgammon, chess, racing), pattern recognition (radar systems, face identification, object recognition), sequence recognition (gesture, speech, handwritten text recognition), medical diagnosis, financial applications, data mining (or knowledge discovery in databases, "KDD"), visualization and e-mail spam filtering.

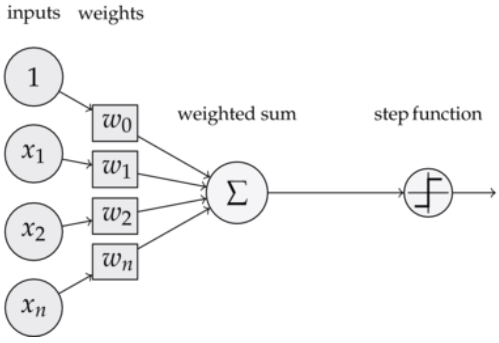
3 Maths behind Neural Network:

Recalling back the ideas from the logistic regression:

$$p(x) = \sigma(wx + b) \quad \text{where, } \sigma(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

Perceptrons:

Figure 2: Perceptron



Perceptrons — invented by Frank Rosenblatt in 1957, are the simplest neural network that consist of n number of inputs, only one neuron and one output, where n is the number of features of our dataset.

In the modern sense, the perception is an algorithm for learning a binary classifier called a threshold function:

A function that maps its input x (a real-valued vector) to an output value f(x) (a single binary value):

$$f(x) = \begin{cases} 1 & \text{if } w.x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

But in this case we find the linear output then making it as non-linear using a sigmoid function. ∵ we're using a non linear activation for not having the problem of non-convex optimization we're going to use a softmax activation in our final layer for multiclass classification.

What is a Neural Network?

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems

Neural networks, in the world of finance, assist in the development of such process as time-series forecasting, algorithmic trading, securities classification, credit risk modeling and constructing proprietary indicators and price derivatives.

A neural network works similarly to the human brain's neural network. A

"neuron" in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis.

A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

In a multi-layered perceptron (MLP), perceptrons are arranged in interconnected layers. The input layer collects input patterns. The output layer has classifications or output signals to which input patterns may map.

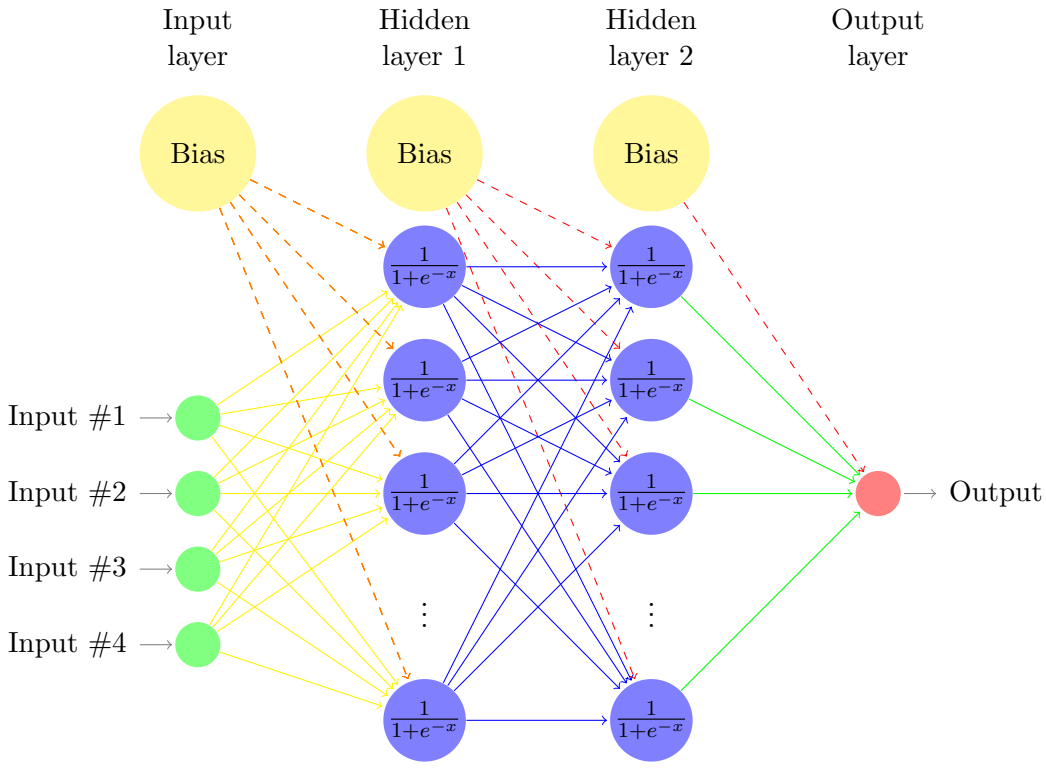


Figure 3: ANN diagram representation.

3.1 Initialize Network

Each neuron has a set of weights that need to be maintained. One weight for each input connection and an additional weight for the bias. We will need to store additional properties for a neuron during training, therefore we will use a dictionary to represent each neuron and store properties by names such as 'weights' for the weights.

A network is organized into layers. The input layer is really just a row from our training dataset. The first real layer is the hidden layer. This is followed by the output layer that has one neuron for each class value.

The dimension of the weights of the i^{th} layer can be defined as the product of number of neurons in this layer and the number of elemnts in the previous layer.

Let the number of neurons in:
1. Input Layer = 4
2. Hidden Layer 1 = n
3. Hidden Layer 2 = m
4. Output Layer = 3

Let the dimensions be:
1. Input Layer = (4,k)
2. Hidden Layer 1 weights = (n,4)
3. Hidden Layer 1 output dimension = (n,k)
4. Hidden Layer 2 weights = (m,n)
5. Hidden Layer 2 output dimension = (m,k)
6. Output Layer weights = (3,m)
7. Output Layer weights output dimension = (3,k)

3.2 Forward Propagation

Hidden Layer 1 : $H^1_{(n,k)} = \sigma(w_{(n,4)}X_{(4,k)} + b_1)$ where X is the input
Hidden Layer 2 : $H^2_{(m,k)} = \sigma(w_{(m,n)}H^1_{(n,k)} + b_2)$
Output Layer : $O_{(3,k)} = softmax(w_{(3,m)}H^2_{(m,k)} + b_3)$
Here $\sigma(x) = \frac{1}{1+e^{-x}}$
and Softmax$(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$

3.3 Loss Calculation

We were going to use the categorical cross entropy for the calculation of loss in our model

i.e.

$$CCE = L = -\sum_{i}^{C} y^i log(O^i) \tag{2}$$

3.4 Backpropagation

Now, to update a weight $W_(i, j)$ that connects a neuron j in the output layer with a neuron i in the previous layer, we need to calculate the partial derivative of the error function using the chain rule:

for w

$$\begin{aligned}
&\frac{\partial L}{\partial w_{(3,m)}} = \frac{\partial L}{\partial O_{(3,k)}} \frac{\partial O_{(3,k)}}{\partial Z_{(3,k)}} \frac{\partial Z_{(3,k)}}{\partial w_{(3,m)}} \\
&\frac{\partial Z_{(3,k)}}{\partial w_{(3,m)}} = H^2_{(m,k)} \quad \frac{\partial L}{\partial O_{(3,k)}} = \frac{y^i}{O^i_{(3,k)}} \\
&\frac{\partial O_{(3,k)}}{\partial Z_{(3,k)}} = O^i_{(3,k)}(\delta_{ij} - O^j_{(3,k)}) \\
&\frac{\partial L}{\partial w_{(3,m)}} = \frac{y^i}{O^i_{(3,k)}} O^i_{(3,k)}(\delta_{ij} - O^j_{(3,k)}) H^2_{(m,k)} \\
&\frac{\partial L}{\partial w_{(3,m)}} = y^i(\delta_{ij} - O^j_{(3,k)}) H^2_{(m,k)}
\end{aligned} \tag{3}$$

for bias 3,

$$\begin{aligned}
&\frac{\partial L}{\partial b_3} = \frac{\partial L}{\partial O_{(3,k)}} \frac{\partial O_{(3,k)}}{\partial Z_{(3,k)}} \frac{\partial Z_{(3,k)}}{\partial b_3} \\
&\frac{\partial Z_{(3,k)}}{\partial b_3} = 1 \qquad \frac{\partial L}{\partial O_{(3,k)}} = \frac{y^i}{O^i_{(3,k)}} \\
&\frac{\partial O_{(3,k)}}{\partial Z_{(3,k)}} = O^i_{(3,k)}(\delta_{ij} - O^j_{(3,k)}) \\
&\frac{\partial L}{\partial b_3} = \frac{y^i}{O^i_{(3,k)}} O^i_{(3,k)}(\delta_{ij} - O^j_{(3,k)}) H^2_{(m,k)} \\
&\frac{\partial L}{\partial b_3} = y^i(\delta_{ij} - O^j_{(3,k)})
\end{aligned} \tag{4}$$

Here $\delta_{ik} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j \end{cases}$

similarly for weights 1 and 2,

$$\begin{aligned} \frac{\partial L}{\partial w_{(m,n)}} &= \frac{\partial L}{\partial w_{(3,m)}}(1 - H^2_{(m,k)})H^1_{(n,k)} \\ \frac{\partial L}{\partial w_{(n,4)}} &= \frac{\partial L}{\partial w_{(m,n)}}(1 - H^1_{(n,k)})X_{(4,k)} \end{aligned} \tag{5}$$

similarly for bias 1 and 2,

$$\begin{aligned} \frac{\partial L}{\partial b_2} &= \frac{\partial L}{\partial b_3}(1 - H^2_{(m,k)}) \\ \frac{\partial L}{\partial b_1} &= \frac{\partial L}{\partial b_2}(1 - H^1_{(n,k)}) \end{aligned} \tag{6}$$

CALCULATIONS:

$$\begin{aligned} \frac{\partial L}{\partial O_{(3,k)}} &= \frac{\partial}{\partial softmax(w_{(3,m)}H^2_{(m,k)}+b_3)}(-\sum_i^C y^i log(softmax(w_{(3,m)}H^2_{(m,k)} + b_3)^i)) \\ &= \frac{y^i}{O^i_{(3,k)}} \end{aligned}$$

Let's assume, $O_j = \frac{e^{Z_j}}{\omega}$ and $\omega = \sum_i e^{Z_i}$

$\therefore log(O_j) = Z_j - log(\omega)$

$O_j = e^{Z_j - log(\omega)}$

$\frac{\partial O_j}{\partial Z_k} = e^{Z_j - log(\omega)}(\frac{\partial Z_j}{\partial Z_k} - \frac{\partial}{\partial Z_k} log(\omega))$

$\frac{\partial O_j}{\partial Z_k} = O_j(\delta_j k - \frac{1}{\omega}\frac{\partial \omega}{\partial Z_k})$

$\frac{\partial \omega}{\partial Z_k} = \Sigma_i e^{Z_i}\delta_{ik} = e^{Z_k}$

$\frac{\partial O_j}{\partial Z_k} = O_j(\delta_{jk} - O_k)$

Finally , $\frac{\partial O^j_{(3,k)}}{\partial Z^k_{(3,k)}} = O^j_{(3,k)}(\delta_{jk} - O^k_{(3,k)})$

$\frac{\partial Z_{(3,k)}}{\partial w_{(3,m)}} = \frac{\partial w_{(3,m)} H^2_{(m,k)} + b_3}{\partial w_{(3,m)}} = H^2_{(m,k)}$

3.5 Conclusion

Hence our model can be updated and trained.