Math behind Artificial Neural Networks

Sai · Follow
Published in Analytics Vidhya
6 min read · Jul 19, 2020

Artificial neural networks seen to be useful in many applications in recent times like prediction, classification, recognition,translation and many more. The current example is an application of simple ANN in predicting the output given the input numbers.

We will be considering an example of an machine which takes in input A, B, C and produces an Output. The example includes training the artificial neural network with set of data(training data) and testing with a different set of data which network wasn't fed before(test data). Data in this case, is collected from experiments with different experimental settings. Example, given input settings A=1,B=1,C=1 produces an output =1, number of runs has to be performed with different experimental settings to get the data. Acquired data has to be divided into two sets- training set(used for training the neural network, test set- used for testing the performance of trained neural network) with ratio of train to test data generally being 80:20.

ANN is similar to human neural network consisting of connected neurons processing information. ANN architecture is shown below with three layers, input layer- layer through which input information is fed, hidden layer- layer connecting input and output layers processing the information, output layer-layer delivering the output.

Below is the sample data set with inputs A,B,C and output labelled "Target". I have

shown only 10 rows of data, in general there has to be more sets of data for training the network.

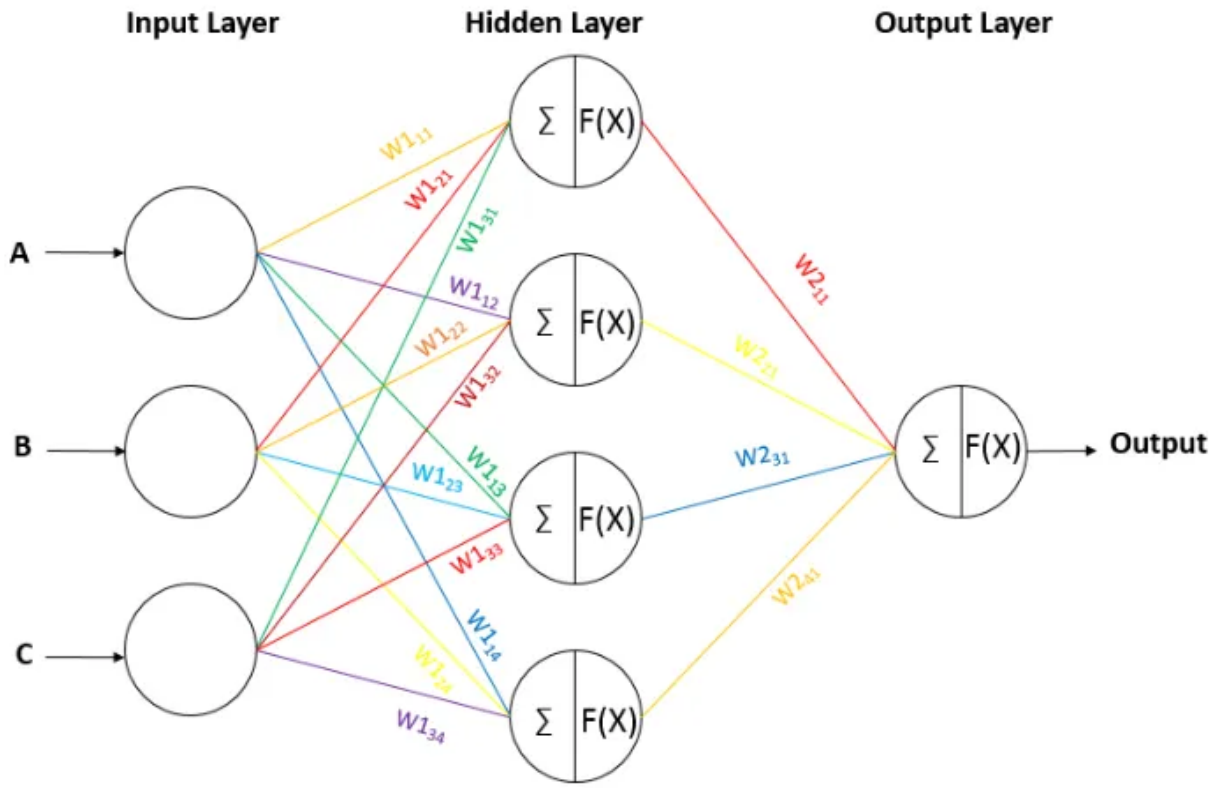
#	A	B	C	Target
1	100	100	100	100
2	90	90	90	90
3	80	80	80	80
4	70	70	70	70
5	60	60	60	60
6	50	50	50	50
7	40	40	40	40
8	30	30	30	30
9	20	20	20	20
10	10	10	10	10
Max	100	100	100	100
Min	10	10	10	10

Normalizing the inputs is not needed for the current example, however input normalization is must when inputs are of different scales.

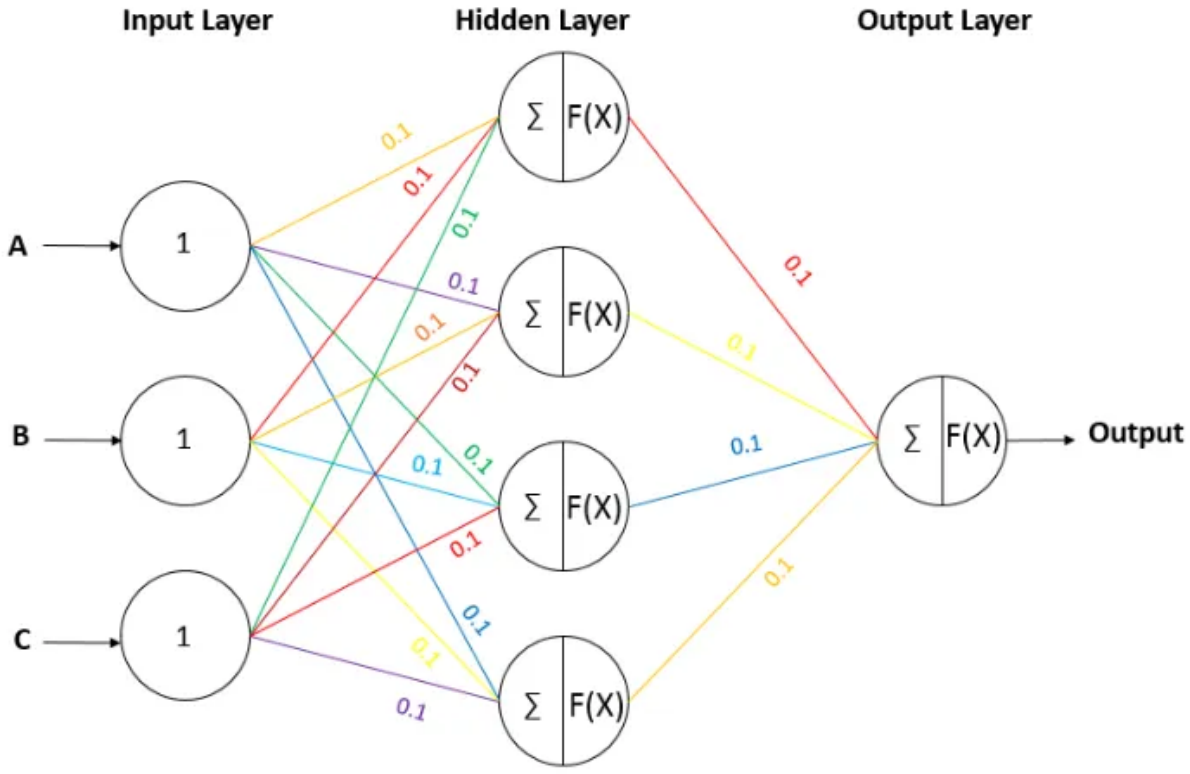
$$\text{Normalized data value} = \frac{\text{Original data value} - \text{Minimum value}}{\text{Maximum Value} - \text{Minimum Value}}$$

#	A	B	C	Target
1	1	1	1	1
2	0.888889	0.888889	0.888889	0.888889
3	0.777778	0.777778	0.777778	0.777778
4	0.666667	0.666667	0.666667	0.666667
5	0.555556	0.555556	0.555556	0.555556
6	0.444444	0.444444	0.444444	0.444444
7	0.333333	0.333333	0.333333	0.333333
8	0.222222	0.222222	0.222222	0.222222
9	0.111111	0.111111	0.111111	0.111111
10	0	0	0	0

A simple feed forward back propagation artificial neural network is shown below with one input, hidden and output layer.



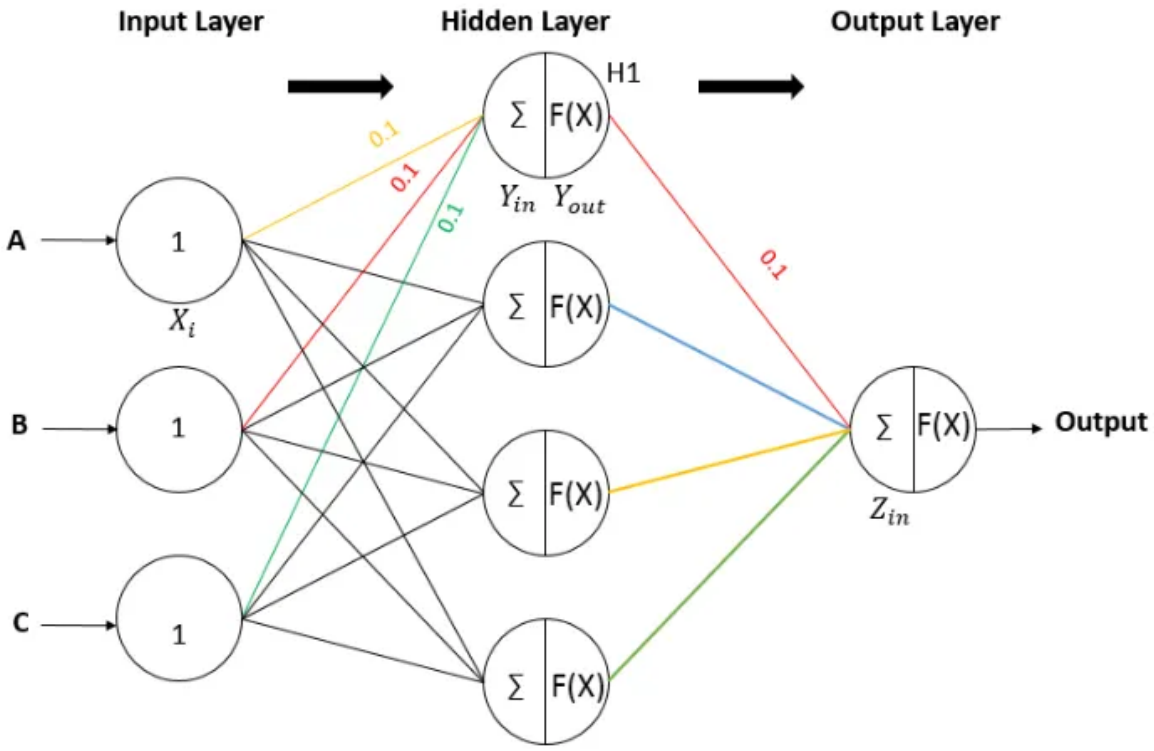
For better understanding, let us consider feeding only one input set from first row in the above table, A=1,B=1,C=1 with an output target =1. Once the architecture is chosen, one has to initialize the weights for synapses connecting different layers. We can initialize weights randomly or choose small values to start with. Here, I have initialized weight of 0.1 for all the connections, making the math simple in following discussion.



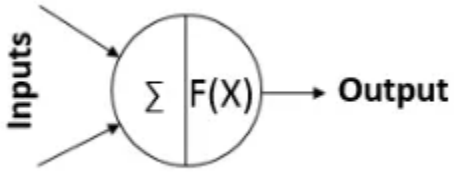
Forward Pass and Back-Propagation

The training consists of two steps -Forward pass : The inputs pass through the network into output layer producing the output. Back-propagation: The error is propagated backwards into the network adjusting the weights.

Forward Pass:



A simple neuron is shown below, it does two functions



1. Summation of input values with respective weights at each node.

2. Activating the input signal using activation function F(X). There are few activation functions like sigmoid, ReLu, tanh etc. in practice, current example uses sigmoid activation function at nodes in both layers.

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

Let us only consider calculations at hidden node 1, H1 in the above ANN figure,

Value at hidden layer 1 = (1 * 0.1) + (1 * 0.1) + (1 * 0.1) = 0.3

$$Y_{in}=\sum X_i * W1_{ij}$$

Applying sigmoid activation function = sigmoid(0.3) = 1/(exp(0.3)+1) = 0.57

$$Y_{out}=F\ (Y_{in})$$

Similar calculations are done at each hidden node and the values are passed onto output layer. Value at node in output layer,

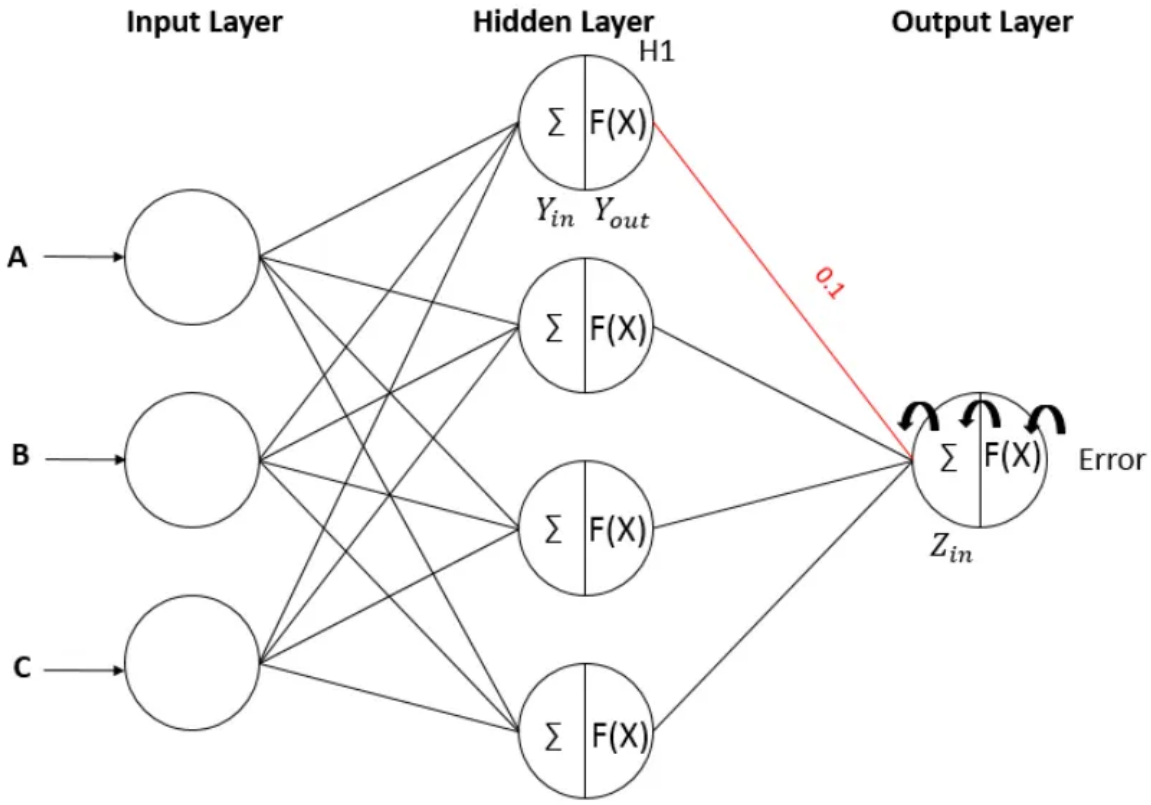
Value at Output node = (0.57 * 0.1) + (0.57 * 0.1) + (0.57 * 0.1) + (0.57 * 0.1) = 0.228

$$Z_{in}=\sum Y_{out} * W2_{jk}$$

Applying sigmoid activation function = sigmoid(2.28) = 1/(exp(2.28)+1) = 0.56

$$Output=F\ (Z_{in})$$

Back Propagation:



Error is the difference between the value predicted by network(output) and the original value(Target), value of error is Error = 0.5 * $(1–0.56)^2$ = 0.0968. Error value here is calculated only for one data point. Error is normally calculated once over all the data points or even in batches.

Error propagation at Output node:

Change of error with respect to only one weight is shown below, which is the weight connecting node H1 in hidden layer to output layer.

Below the term on left hand side of equation is the change by which weights connecting hidden-output layer has to be updated to minimize the error so that output value matches the target. By applying chain rule we get:

$$\frac{\partial Error}{\partial W2_{11}} = \frac{\partial Error}{\partial output} * \frac{\partial output}{\partial Z_{in}} * \frac{\partial Z_{in}}{\partial W2_{11}}$$

Splitting and calculating each term in the above equation:

1. $$\frac{\partial Error}{\partial Output} = 0.5 * \frac{\partial (Target - Output)^2}{\partial Output} = -(Target\text{-}Output)$$
2. $$\frac{\partial Output}{\partial Z_{in}} = \frac{\partial f(Z_{in})}{\partial Z_{in}}$$
3. $$\frac{\partial Z_{in}}{\partial W2_{11}} = \frac{\partial (\sum Y_{out} * W2_{jk})}{\partial W2_{11}}$$

Grouping all the terms 1, 2, 3 to get the error information by which the weight

$$\frac{\partial Error}{\partial W2_{11}} = -(Target\text{-}Output) * f'(Z_{in}) * Y_{out}$$

Error information by which the weight ... to ...

Artificial Intelligence | Artificial Neural Network | Data Science | Predictive Analytics | Predictive Modeling

... 0.57 = -0.0617

Similar calculations for Error information at hidden node:

$$\frac{\partial Error}{\partial W2_{11}} = -(Target-Output)*f'(Z_{in}) * Y_{ou}$$

Written by Sai

3 Followers · Writer for Analytics Vidhya

Splitting and calculating each term in the above equation:

1. $$\frac{\partial Error}{\partial Y_{out}} = \frac{\partial Error}{\partial output} * \frac{\partial output}{\partial Z_{in}} * \frac{\partial Z_{in}}{\partial Y_{out}}$$

2. $$\frac{\partial Y_{out}}{\partial Y_{in}} = \frac{\partial f(Y_{in})}{\partial Y_{in}}$$

3. $$\frac{\partial Y_{in}}{\partial W1_{11}} = \frac{\partial(\sum X_i * W1_{ij})}{\partial W1_{11}}$$

Sai in Analytics Vidhya

SQL for Data Scientists

In this article, I will walk through few important SQL functions by solving problems- hoping this article will help people who are...

4 min read · Sep 1, 2020

15

Error term = -(1–0.56) * 0.246 * 0.1 * 0.245 * 1 = -0.00265

The weights get updated by error information at each node in hidden and output layers.

Kia Elsinga · Analytics Vidhya

updated weights connecting hidden-output layers,

=0.1 * (0.68614)

With the new updated weights the feed forward pass and back propagation iterates until the weights settle at particular value to minimize the error making the output value match the target value. The algorithm iterates for all the rows in data table and finally the ANN gets settled with particular weights making it ready for prediction.

The final step after training is to feed the input data into trained neural network for prediction results.

This article provides the basic understanding of artificial neural networks, however there are many concepts to explore like bias, learning rate, activation functions, momentum factor, architectures which makes the neural network robust and efficient..

Please report any mistakes. Thank you for reading.

How to create a Python library

Ever wanted to create a Python library, albeit for your team at work or for some open source project online? In this blog you will learn

7 min read · Jan 26, 2020

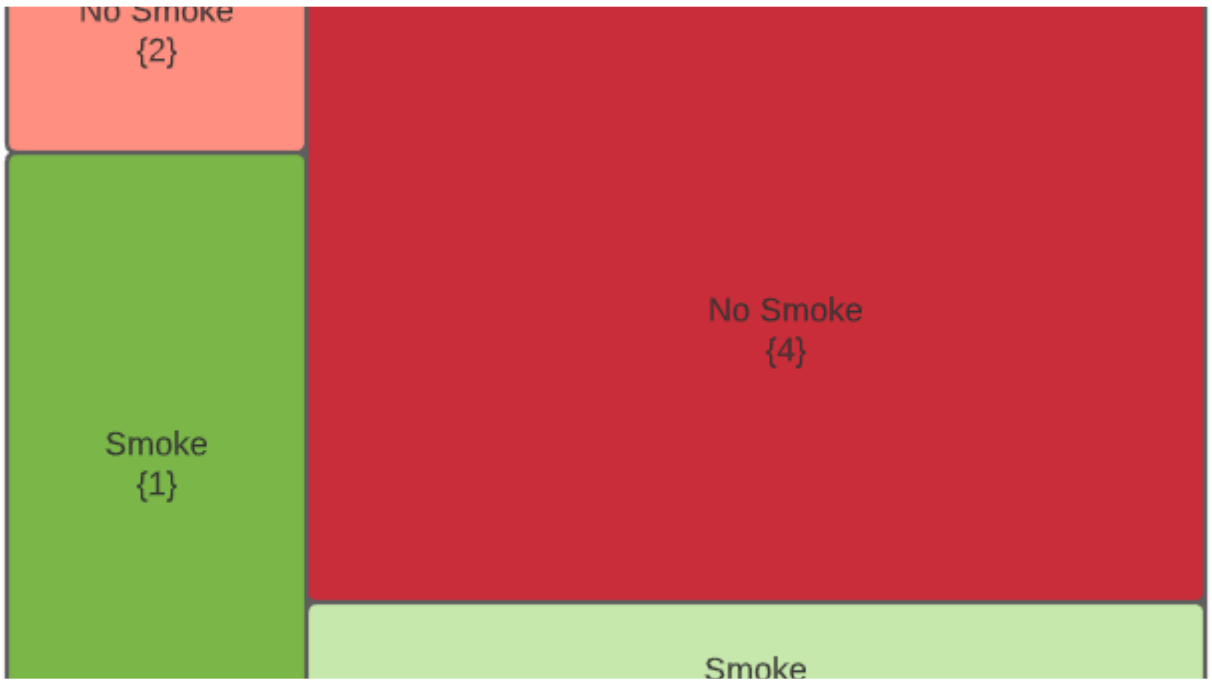
197K 23

Harikrishnan N B in Analytics Vidhya

Confusion Matrix, Accuracy, Precision, Recall, F1 Score

Binary Classification Metric

6 min read · Dec 10, 2019



Sai in Analytics Vidhya

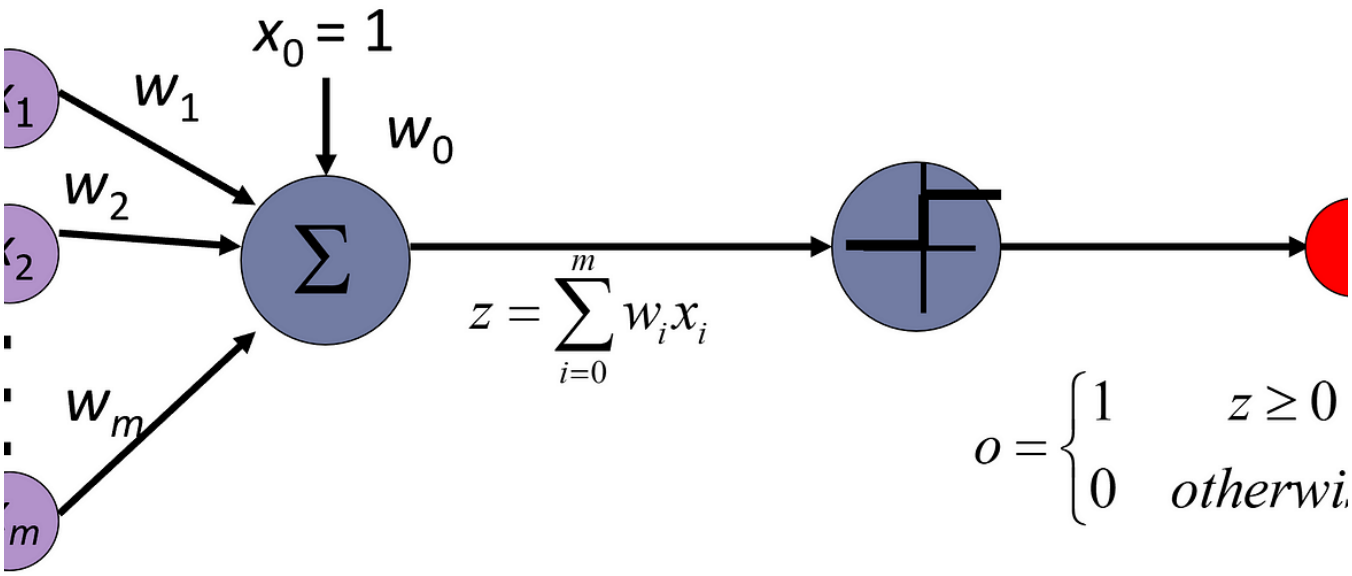
Bayes Theorem and Text Classification using Naive Bayes Classifier

This article discusses Bayes theorem and how Naive Bayes classifier is used in text classification.

5 min read · Aug 31, 2020

7

Recommended from Medium



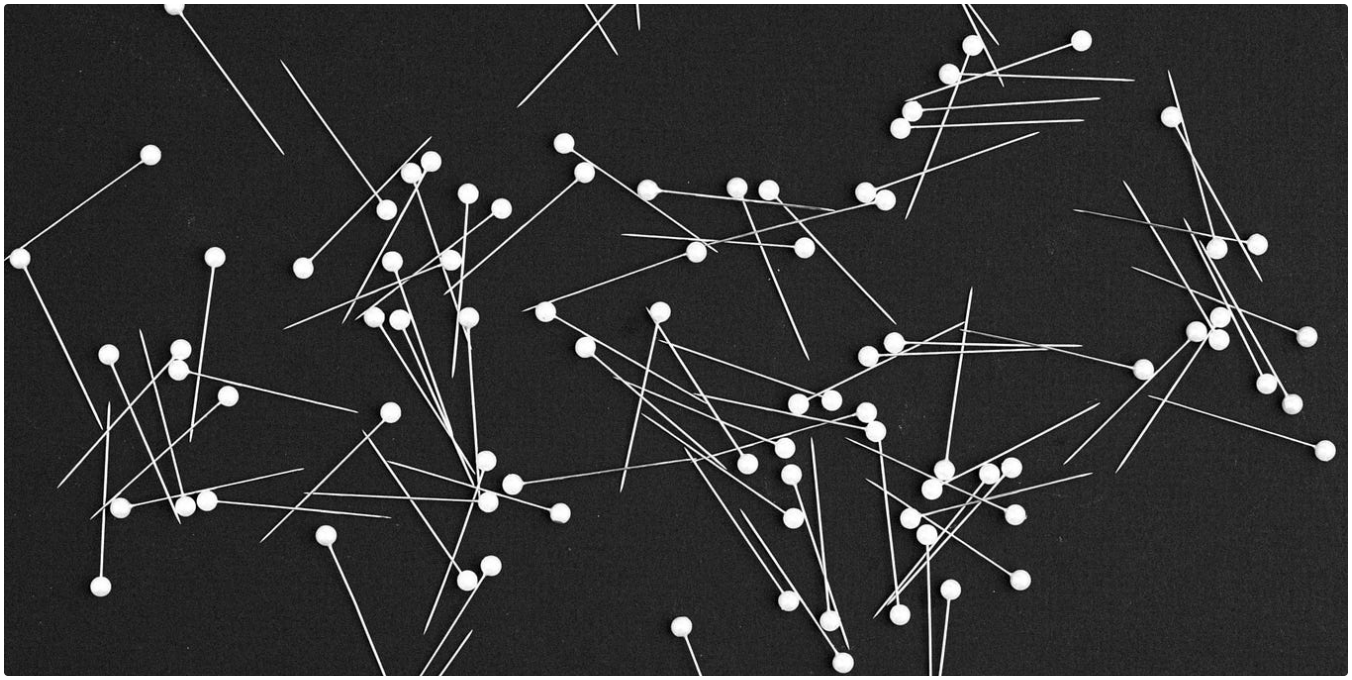
Dr. Roi Yehoshua in Towards Data Science

Perceptrons: The First Neural Network Model

Overview and implementation in Python Perceptrons are one of the earliest computational models of neural networks (NNs), and they form the

· 14 min read · Mar 28

239 7



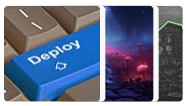
Vikas Kumar Ojha in Geek Culture

Binary Neural Networks: A Game Changer in Machine Learning

This blog explains about binary neural networks which have potential of revolutionizing deep learning if proper efforts are made.

· 9 min read · Feb 19

Lists



Predictive Modeling w/ Python

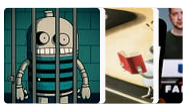
20 stories · 268 saves

ChatGPT prompts

24 stories · 254 saves

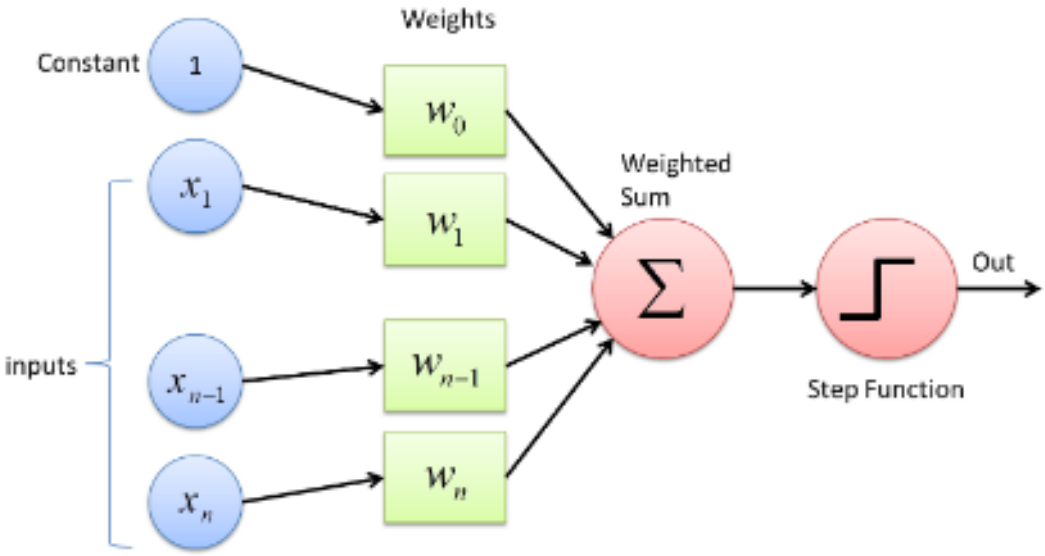
ChatGPT

21 stories · 109 saves



AI Regulation

6 stories · 78 saves



Joseph Robinson, Ph.D. in Towards Data Science

From Basic Gates to Deep Neural Networks: The Definitive Perceptron Tutorial

Demystifying Mathematics, Binary Classification, and Logic Gates

21 min read · Apr 28

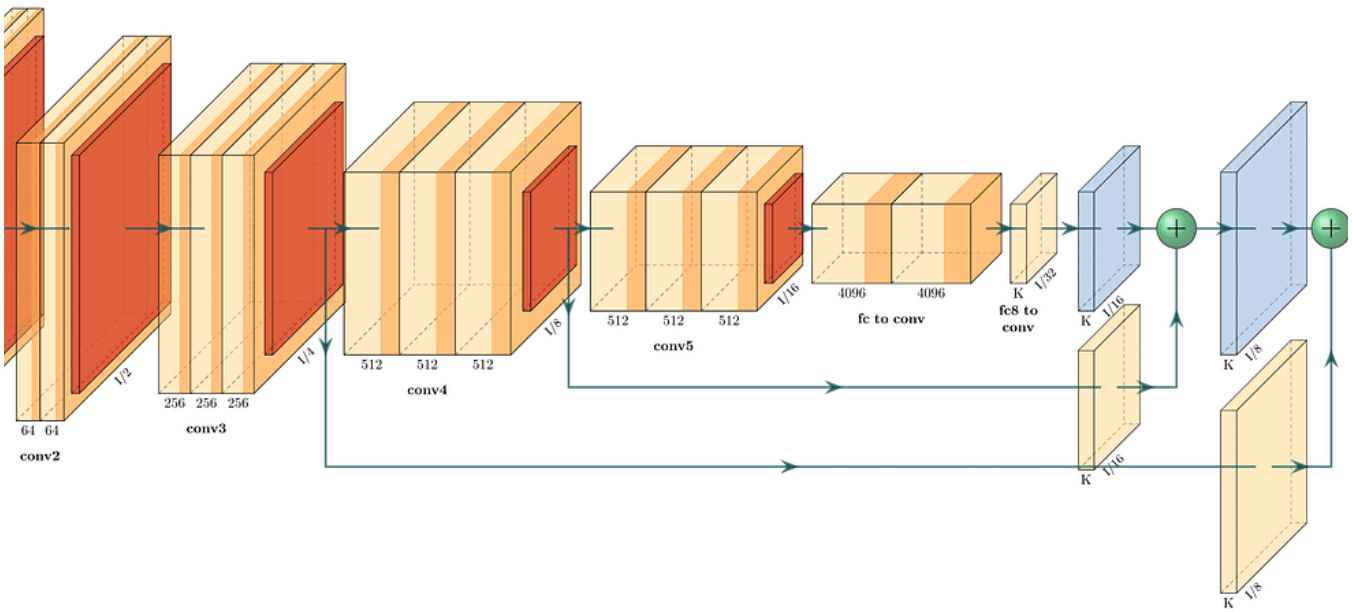
400 3

Matt Chapman in Towards Data Science

The Portfolio that Got Me a Data Scientist Job

Spoiler alert: It was surprisingly easy (and free) to make

· 10 min read · Mar 24



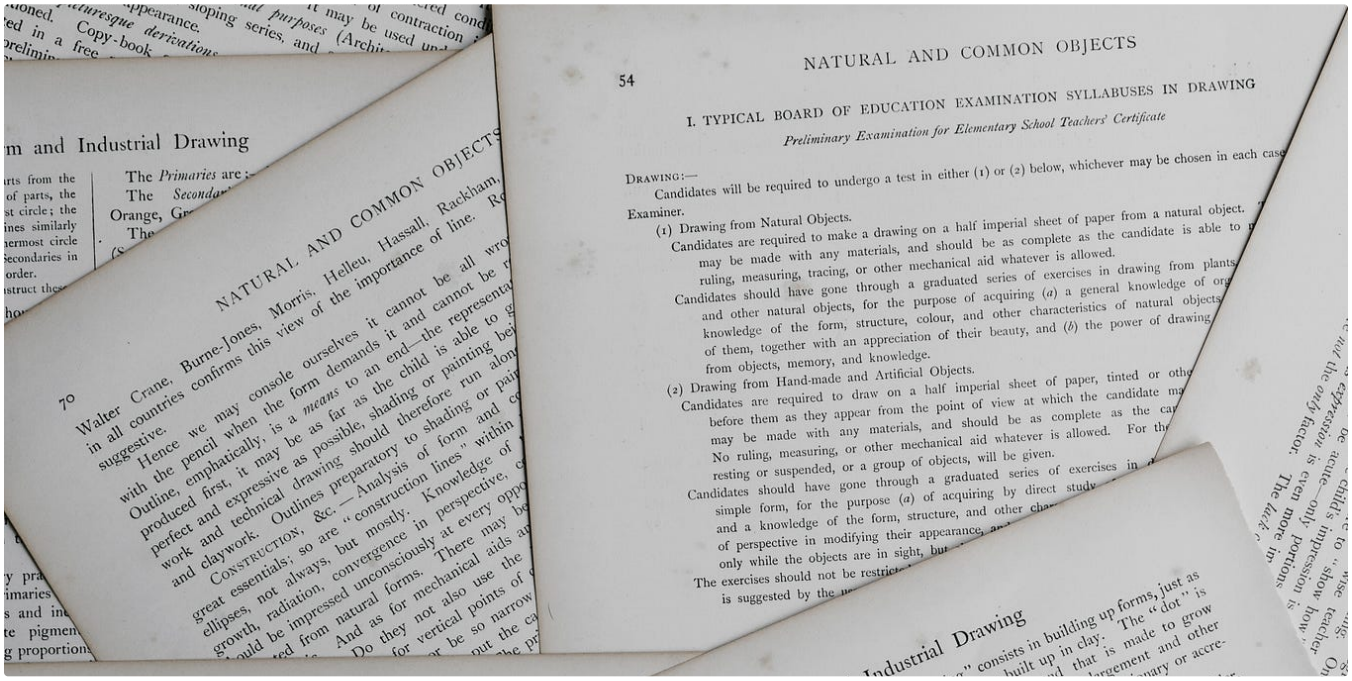
Clément Delteil in Towards AI

Creating Stunning Neural Network Visualizations with ChatGPT and PlotNeuralNet

Presenting PlotNeuralNet, a LaTeX / Python package to visualize Neural Networks

✨ · 8 min read · Mar 8

159



Youssef Hosni in Towards AI

How to Read Machine Learning Papers Effectively

The field of machine and deep learning is evolving very fast, and there are new research outputs every day. Therefore you will need to read...

✨ · 10 min read · Oct 9, 2022

1.2K 9

See more recommendations