International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 04 | Apr 2021 www.irjet.net p-ISSN: 2395-0072

Basic Overview of Neural Networks, Stochastic Gradient Descent and Activation Function

Mr. Abhishek Jani[1] Mr. Shubh Jani[2] Mr. Abhishek Hatui[3] Mr. Shreyansh Dave[4], Prof. Kalpesh Kubal[5]

[1,2,3,4]Student, Thakur Polytechnic College, Mumbai, Maharashtra, India
[5]Lecturer, Thakur Polytechnic College, Mumbai, Maharashtra, India

---

Abstract: *Inspired by the sophisticated functionality of human brains where hundreds of billions of interconnected neurons process information in parallel, researchers have successfully tried demonstrating certain levels of intelligence on silicon. Examples include language translation and pattern recognition software. While simulation of human consciousness and emotion is still in the realm of science fiction, we, in this chapter, consider artificial neural networks as universal function approximators. Especially, we introduce neural networks which are suited for time series forecasts.*

Key-words: Neural Networks, Stochastic Gradient Descent, Neuron, Activation function.

Introduction*:* Artificial neural network (ANN) is inspired by biological neural network. ANN is nothing but large number of processes with many interconnections in a huge parallel computing system. ANN model's try to mimic the organizational thinking capabilities like humans. A type of network assumes node as artificial neurons, these are called as artificial neural networks (ANN). Inspired by natural neurons an artificial neuron is computational model of it. ANN is mostly used in fields related with information processing. Replicating real neural network ANN is used in various ways like studding behavior of animals also for engineering purpose, such as pattern recognition, forecasting, data compression

Neuron: Neuron is a basic building block of artificial neural network. Fig (1) is an actual image of neuron seen from microscope and to create artificial neural network we need to recreate this inner machine as the whole purpose of deep learning is to mimic how the human brain works in hope that we can create an infrastructure for machine to be able to learn as human brain has one of the most powerful learning mechanisms on the planet. The very first step to create artificial neural network is to recreate a neuron. Fig (2) is diagram of a neuron where the center there is the body called Neuron there are branches called Dendrites and long tail called Axon. Dendrites are the receiver of the Neuron and the Axon is the transmitter of the Neuron so that the Neuron can work in larger pairs. Fig (3) we can see Dendrites of one Neuron is connected to Axon of other Neuron this connection is called synapse. Fig (4) shows the recreation of Neuron where a neuron is getting input values from other neurons (input layer neurons) passed through synapse and creating an output signal. so, the input layer neurons are also connected to other neurons not shown in the diagram. In analogy of human brain, the input layer is senses. The input signals are independent variables; these variables are standardized so it is easier for neural network to process them. The output value can be continuous, binary or categorical. Every synapse has some weight and these weights are important because weights are how neural network learn. By the weights neural network decides what signals are important and what signals are not important. And inside the neuron all the values that it is getting is added up and applies an activation function to the sum and from that the neuron understand what signal to pass or weather to pass the signal or not.

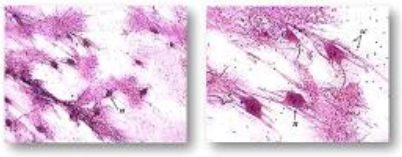


Fig (1)

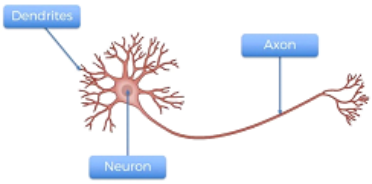


Fig (2)

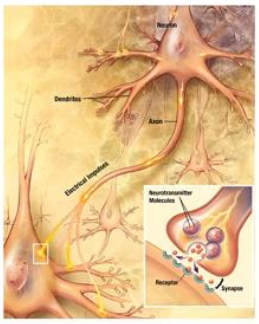


Fig (3)

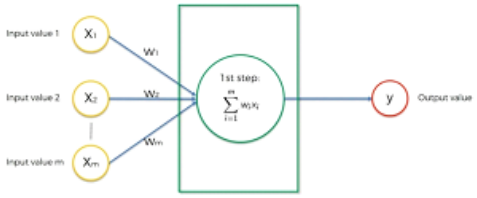
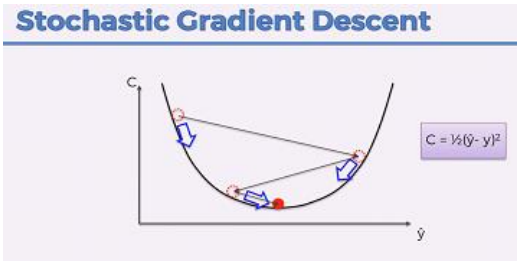
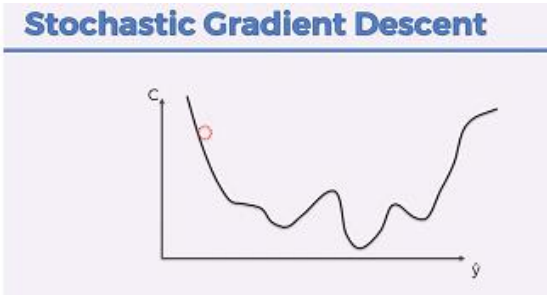


Fig (4)

Stochastic Gradient Descent:



The Gradient descent uses convex cost function. It convex into one global minimum which we are going to find using gradient descent. But what if our cost function is not convex? And looks something like this



First of all, how could that happen?

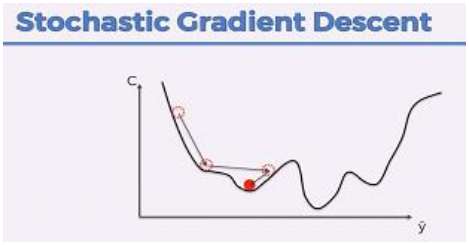
This could happen if:

1.We choose a cost function which is not the squared difference between ŷ and y.

2.We do choose a cost function which is squared difference between ŷ and y but then, in the multi-dimensional space it can turn into something which is not convex.

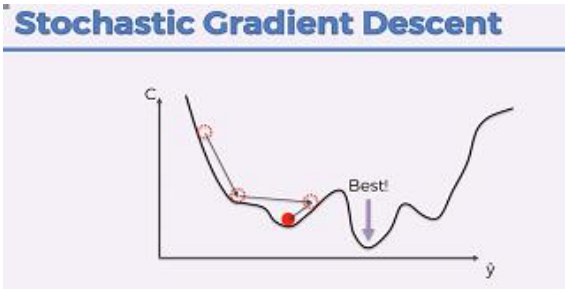
So, what would happen in this case when we try to apply normal gradient descent method?

Something like this will happen we will find the local minimum of the cost function.



Local minimum

Rather than the global minimum, and therefore we do not have the correct weights and hence we don't have an optimized neural network. We will have a subpar neural network.

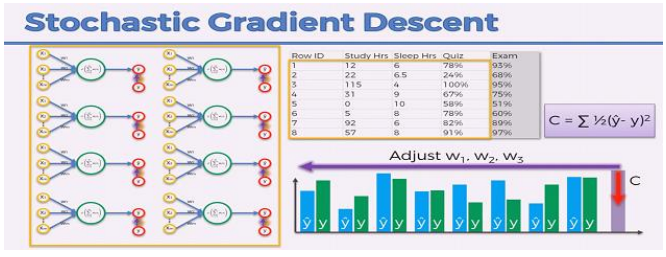


Global minimum

So, what do we do in this case? Well, the answer is Stochastic gradient descent as the it doesn't require its cost function to be convex.

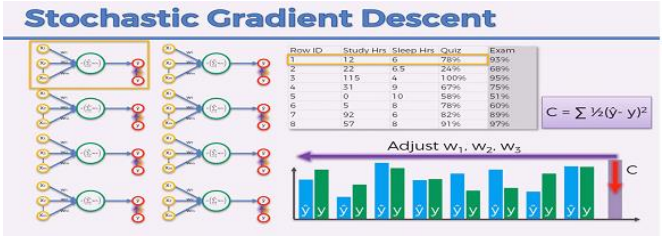
Let's have look in the differences between gradient descent and stochastic gradient descent with an example:

Gradient descent is when we take all of our rows and we plug them into our neural network and once we plug them in, we calculate our cost function using $C = \Sigma \frac{1}{2}(\hat{y}-y)^2$ and then we adjust the weights. The proper term for this method is batch gradient descent method.

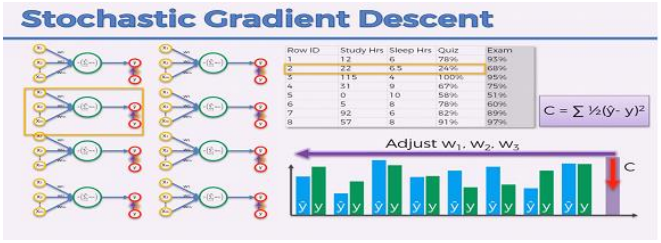


Batch Gradient Descent Method

In Stochastic gradient method we take the rows one by one. We take first row, then run our neural network, we look at the cost function and then we adjust the weights.

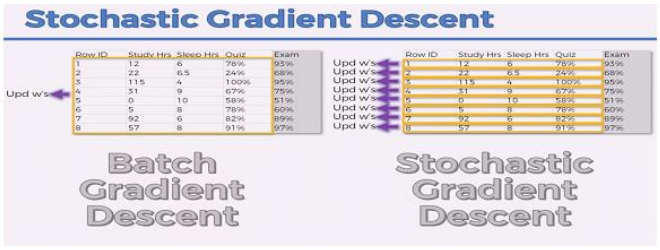


Stochastic Gradient Method



Then we move on to the second row and run it through the neural network, we look at the cost function and then we adjust the weights. We continue the same for all the remaining rows.

Batch Gradient Descent and Stochastic Gradient Descent comparison:



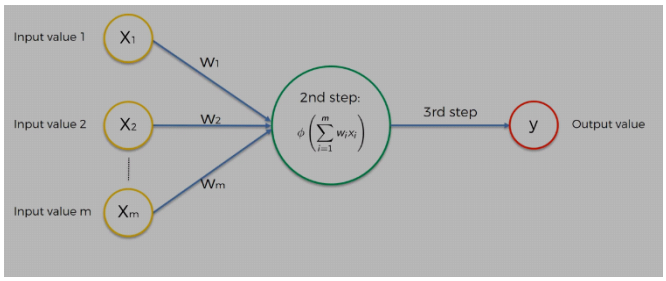
The main difference is that:

1.Stochastic gradient descent helps you to avoid the problem where you find the local minimum rather than the global minimum. The reason for that is because Stochastic gradient descent has a much higher fluctuation because it could afford them, it's doing one iteration or one row at a time and therefore the fluctuation is much higher and hence, it's much more likely to find the global minimum.

2.The other difference is that it is faster. The first impression that you might have is that it is slower because it is doing every single row one at a time but, actually it is faster. Because it doesn't have to load up the data in the memory and wait to run until all the rows are loaded. It can just run one row one by one as a result it is much light weight algorithm.

3. The gradient descent algorithm is a deterministic algorithm and stochastic gradient descent is stochastic algorithm meaning it is random. The weights in gradient descent method is always same as long as the starting weight is similar whereas in stochastic gradient descent the weights are always different as you are picking your rows possibly at random and you are updating your neural network in a stochastic manner therefore every single time you run the stochastic gradient method even though you have the same starting weights you are going to have different process or iteration.

Activation Function: An activation function is a very important feature of an artificial neural network; they basically decide whether the neuron should be activated or not



In the above figure, (***x1,x2,…xm***)is the input signal vector that gets multiplied with the weights(***w1,w2,…wm***). This is followed by accumulation (i.e., summation + addition of bias ***b***). Finally, an activation function ***f*** is applied to this sum.

Sigmoid Activation Function:

The Sigmoid Function curve looks like a S-shape.

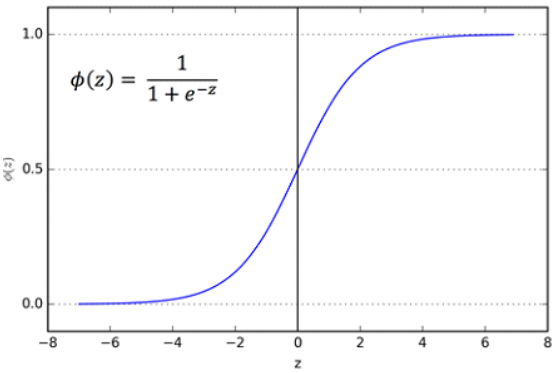


Fig: Sigmoid Function

Equation: f(x) = 1 / 1 + exp(-x)

Range: (0 to 1)

Pros:

1.The function is **differentiable**. That means, we can find the slope of the sigmoid curve at any two points

2. The function is **monotonic** but function's derivative is not

Cons:

1.It gives rise to a problem of "**vanishing gradients**", since the Y values tend to respond very less to changes in X

2.Secondly, its output isn't zero centered. It makes the gradient updates go too far in different directions. **0 < output < 1, and it makes optimization harder.**

3. Sigmoid saturate and kill gradients.

4. Sigmoid have slow convergence.

ReLU (Rectified Linear Unit) Activation Function:

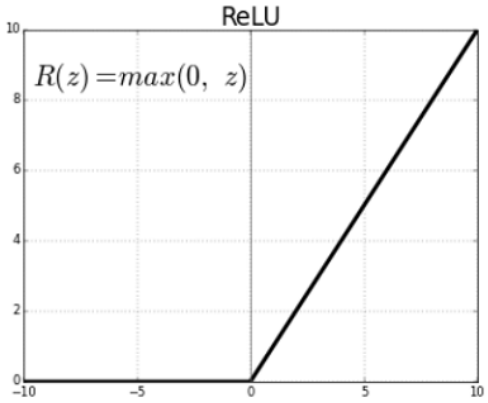


Fig RELU funnction

Equation: f(x) = max (0, x)

Range: (0 to infinity)

Pros:

- The function and its **derivative** both are **monotonic**.
- Due to its functionality, it does not activate all the neuron at the same time
- It is efficient and easy for computation.

Cons:

- The outputs are not zero centered similar to the sigmoid activation function
- When the gradient hits zero for the negative values, it does not converge towards the minima which will result in a dead neuron while back propagation.

Limitations:

Accuracy: the accuracy of the Ann may vary from time to time. to get the highest possible accuracy it takes a lot of trial and error to set the most optimal dimensionality of the output space.

the accuracy may also vary for different datasets. The batch size and the number of epochs also changes the accuracy.

perquisite: The Ann requires a lot of data before it can give proper results. It requires much more data than traditional machine learning techniques.

hardware: Ann requires powerful processor as it processes huge samples of data.

Conclusion:

The Ann is based on the actual neurons in our brain. The Ann can be used in many situations but one of the most vital use of Ann is that it can be used for predicting results accurately.

The stochastic gradient descent is superior than the gradient descent algorithm in many cases. Use stochastic gradient descent algorithm for a fast and accurate ann.

The activation function is very important part of the hidden layer in an ann. The most commonly used activation function is RelU (rectifier function). The type of the Ann can be decided by the activation function used in the final output node.

Acknowledgments:

We would like to express our sincere thanks to our HOD Mrs. Vaishali Rane, our Gide Mr. Kalpesh Kubal and all the staff in the faculty of computer engineering department for their valuable assistance.

Reference:

1. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
2. https://en.wikipedia.org/wiki/Stochastic_gradient_descent
3. https://www.ibm.com/cloud/learn/neural-networks