Activation Functions and Training Algorithms for Deep Neural Network

Gayatri Khanvilkar
Department of Information Technology
Vidyalankar Institute of Technology
Mumbai, India
Email: khanvilkarg7@gmail.com

Prof. Deepali Vora
Department of Information Technology
Vidyalankar Institute of Technology
Mumbai, India
Email: deepali.vora@vit.edu.in

Abstract:

Deep learning is the fastest growing field in machine learning which achieves great results in many applications. Deep neural network is one the architecture of Deep learning. Training of Deep neural network is difficult and mainly faces two challenges i.e. overfitting and computation time. Training algorithms and activation functions plays a vital role in overall training and prediction using deep Neural Network. This paper gives an overview of Activation Functions (Sigmoid, Tanh and ReLu) and Training Algorithms (Greedy layer wise Training and Dropout) used for Deep Neural Network. The paper underlines a brief comparative study of activation functions and training algorithms of Deep Neural Network.

1 Introduction

Machine learning gives computer ability to learn complex things by its own, without explicitly programmed. [1] Deep learning is a part of machine learning methods based on learning data representations. Learning can be supervised, partially supervised or unsupervised. [2] Learning or credit assignment is about finding weights that make the neural network exhibit desired behaviour. Depending on the problem and connection of neurons, such behaviour may require long chains of computational stages. Deep Learning is about accurately assigning credit across many such stages. [3] Traditional machine learning depends upon standard neural network but if there are more number of layers then that neural network called as deep neural network which has more prediction accuracy than NN. [4]

2 Deep Neural Network

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by biological nervous systems, such as the brain, information processing of brain. It is composed of large number of highly interconnected devices known as neurons. Theses neurons works together to solve any problem. Each neuron operates on two modes: 1) Training Mode where neuron train for fair to particular input. 2) Using Mode where, if particular input is detected as input to neuron then its associated output will be current output. Neural networks are typically organized into three types of layers 1) Input Layer, 2) Hidden Layer and 3) Output Layer etc. Input layer get activated through sensors data collected from environment. And other layers get activated through input which is nothing but output of previous layers. [5]

Deep-learning networks are different from the standard single-hidden-layer neural networks by their depth. This depth is nothing but number of layers through which data passes. If there are more than one hidden layer in neural network then that NN consider as Deep neural network. [4]

We can essentially stack layers of neurons on top of each other. The lowest layer takes input data, then each neuron stores some information about that data. Each neuron in the layer sends information to the next layer neurons. This new neuron will learn a more abstract version of the data. So, as we go from lower layer to

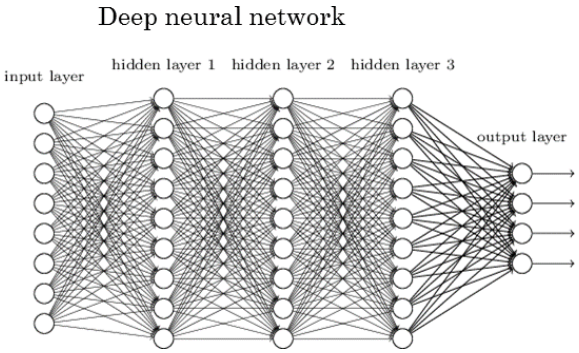

Figure 2.1 Deep Neural Network [6]

higher layer the more abstract features we can learn. [6]

3 Activation function

Activation function in neural network is inspired by biological neuron in human brain. [7] Activation Function in Artificial neural network calculate weighted sum of input, adds bias and then decides, neuron should fire or not.

Mathematical form of Activation Function:

$$\mathbf{X = \sum (WEIGHT * INPUT) + BIAS}$$

Where X is between -∞ to +∞. [8]

If final sum i.e. X value is above threshold then neuron get activate. This threshold is determined by Activation function to turn on neuron. Activation function has two main features.

- Activation function limits the amplitude range of the output signal to a finite value.
- It provides non-linearity property to estimate any function. [8]

3.1 Role and Importance of Activation Function

If we do not apply activation function then output of neural network will be linear equation. These linear equations are easy to solve but doesn't has sufficient power to learn complex data like images, audio, videos etc. If we do not apply activation function then neural network will be simple linear regression model which limited in power and many times doesn't perform well. If we do not apply activation function then neural network won't be able to complex data, can't handle big datasets etc. [9]

3.2 Most popular types of Activation functions:

- Sigmoid or Logistic
- Tanh—Hyperbolic tangent
- ReLu -Rectified linear units

3.2.1 Sigmoid or Logistic

It is an activation function of form

$$\mathbf{A = \frac{1}{1+e^{-x}}}$$

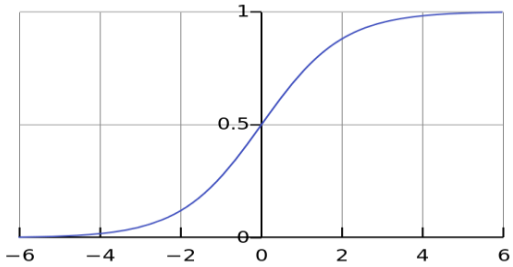

Figure 3.1 Sigmoid Activation function [8]

It looks like smooth step function, S in shaped. It is non-linear in nature. If we observe fig 3.3, between values of X -2 to 2, Y values are very steep. That means small change in X value reflects huge change in Y value. It has tendency to bring y values on either side of curve. Because of this it makes clear decision on prediction and good for classifiers. Unlike linear activation function, sigmoid activation function output is bounded in range 0 to 1. It is easy to use and understand but if we observed fig 3.3, towards either end of curve value of Y is not responding to the change of value of X. this means value of gradient in that region is very small that causes problem of Vanishing gradient. Its output isn't zero centred. It makes the gradient updates go too far in different directions. 0 < output < 1, and it makes optimization harder. [8] [9]

3.2.2 Tanh—Hyperbolic tangent

Its mathematical formula is

$$\mathbf{f(x) = Tanh(x) = \frac{2}{1 + e^{-2x}} - 1}$$

This looks very similar to sigmoid.

$$\mathbf{Tanh(x) = 2sigmoid(2x) - 1}$$

All characteristics of Tanh activation function are similar to Sigmoid

activation function. Output of Tanh is bounded in range of -1 to +1. It is zero-centred and hence optimization is easier.

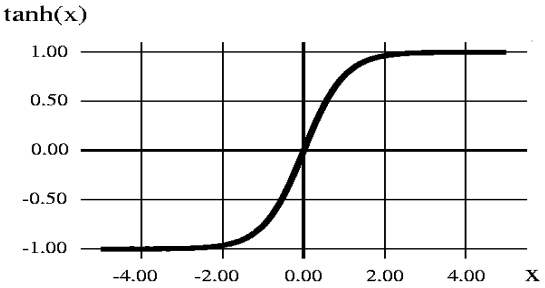

Figure 3.2 Tanh Activation function [9]

While selecting between Tanh and Sigmoid, Tanh preferred over Sigmoid, as it gives optimized solution and has strong gradient. [8] [9] Tanh activation function required more computational time than LReLU. It is due to the exponential unsupervised pre-training. [10]

3.2.3 ReLu -Rectified linear units

It has become very popular in the past few years.

R(x) = max (0, x) i.e. if x < 0 R(x) = 0 and if x >= 0 R(x) = x.

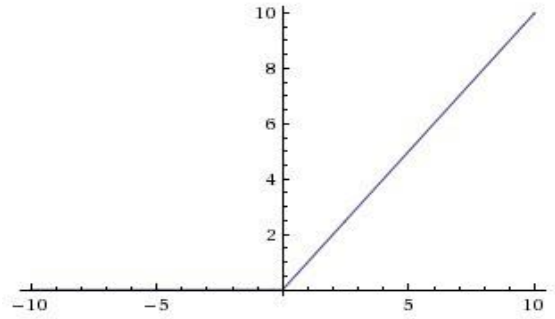

Figure 3.3 ReLu Activation function [8]

This would look linear function, but ReLu is nonlinear in nature. It is not bound activation function. The output range of ReLu is [0, inf). This means it can blow up the activation.

Rectified Unit Family:

- ReLu

  ReLu is not purely linear but consist of linear segments. Such functions are called as piecewise linear. This ReLu function's positive part is an identity function and negative part is set to zero.
- Leaky ReLu (LReLu)

  It assigns a relative smaller and predefined slop to the negative part.
- Randomized ReLu(RReLu)

  It samples the slope from a uniform distribution during training and set it to the mean of the distribution during test, which helps to decrease the overfitting via this randomized strategy.
- Parametric ReLu(PReLu)

  It allows the slope to be learned from data and improves the representation ability.
- Exponential Linear Unit(ELU)

  It exponentially reduces the slope from the predefined and fixed threshold to zero and is beneficial to speed up model learning.

According to different negative parts of activation functions, there are three categories: 1) Zero-type 2) linear-type and 3) exponential-type. Among them, ReLu belongs to the first type, LReLu and PReLu belongs to second and ELU belongs to third type.

ReLu give us benefit of sparsity of the activation. Imagine a network with random initialized weights and almost 50% of the network yields 0 activation because of the characteristic of ReLu (output 0 for negative values of x). This means a fewer neurons are firing (sparse activation) and the network is lighter. Because of horizontal line in ReLu (for negative values of X) gradients can be fragile during training and can die. That means weights will not get adjusted and neurons will stop responding to variations in error or input. This is called Dead Neuron or dying ReLu problem. To fix this problem one modification was introduced called Leaky ReLu. It introduces a small slope to keep the updates alive. One more limitation is that it should only be used within Hidden layers of a Neural Network Model. Hence for output layers we should use a SoftMax function for a Classification problem, and for a regression problem it should simply use a linear function. [8] [9] [11] [12]

3.3 Summary of Activation Functions

Summary of Activation functions is given as follows:

Table 3.1 Summary of Activation Functions

Activation Functions/ Parameters	Sigmoid	Tanh	ReLu
Nature	Non-Linear	Non-Linear	Non-Linear
Mathematical Representation	$A=\frac{1}{1+e^{-x}}$	$\text{Tanh}(x)=\frac{2}{1+e^{-2x}}-1$	R(x) = max (0, x)
Complexity	Easy to understand and apply	Easy to understand and apply	Simple and efficient
Output Range	0 to 2	-1 to 1	O to ∞
Activation Blow-up	Activation bound in range so no activation blowing	Activation bound in range so no activation blowing	Activation not bound in range so activation blowing
Optimization	This function isn't zero centred so optimizations is harder	This function is zero centred so optimizations is easier.	Give optimized solution.
Gradient	This function has smooth and small gradient, which gets kill.	This function has strong gradient than sigmoid.	In this gradient get fragile and die during training.
Computation Time	Requires more computation time	Requires more computation time than ReLu	Requires less computation time than sigmoid and Tanh
Sparsity Property	No	No	Yes
Problem of Dead Neuron	No	No	Yes Can be solved by LeakyReLu
Vanishing Gradient Problem	Yes	Yes	No
Layers	Can apply on any layer of NN	Can apply on any layer of NN	Can apply on only Hidden layer of NN
Convergence	Slow	Slower than ReLU	improved than Tanh
Classification and regression	Good for classification	Good in classification	For classification need to use softmax and for regression need to use linear function.

4 Training Algorithms

Training Algorithms are typically classified into three broad categories, depending on the nature of the learning "signal" or "feedback" available to a learning system. These are as follows:

- Supervised learning:

  In this computer knows the desired output of inputs and

accordingly adjust weights i.e. general rule.

- Unsupervised learning:

  In this computer doesn't knows the desired output of inputs, leaving on its own to find appropriate weights

- Reinforcement learning:

  A computer program interacts with a dynamic environment in which it must perform a certain goal.

Between supervised and unsupervised learning is semi-supervised learning, where the teacher gives an incomplete training signal: a training set with some (often many) of the target outputs missing. The following algorithms can be used in supervised and unsupervised learning both. [1]

4.1 Greedy Algorithm

Greedy algorithm is a step by step training algorithm. In this each step should be able to give the local optimal solution. The original problem gets divided into a similar, but small size sub problems and in each step the choice made is the best choice for the current. Each step can be guaranteed give a local optimal solution, but the overall solution is not necessarily the optimal.

Greedy algorithms can produce optimal solutions for some mathematical problems, but not for others. Most problems for which greedy algorithm work well, will have two properties:

- **Greedy choice property**

  The choice made by a greedy algorithm may depends on choices made so far, but not on future choices or all the solutions to the subproblem.

- **Optimal substructure**

  When the optimal solution of a problem contains the optimal solution of the sub problem, it is called the optimal sub structure. [13]

It is faster than other optimization methods like dynamic programming. Examples of such greedy algorithms are Kruskal's algorithm and Prim's algorithm. [14]

The greedy layer-wise training method pre-trains by training each layer one by one, with the way of autoencoder trained. In the process of pre-training, at first an autoencoder neural network is consisting of input layer and the first hidden layer, after this autoencoder is trained the first hidden layer and the second hidden layer will form another autoencoder. This process continue until the training of the last hidden layer is completed. The encoding step for the greedy layer-wise training is given by running the autoencoder training. Decoder of each autoencoder is imaginary, which will be abandoned after the autoencoder of each layer is trained. [15]

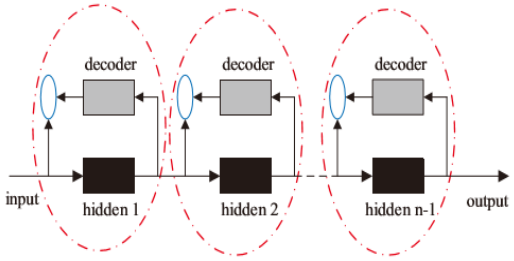

Figure 4.2 Layer-wise pre-training of deep networks [18]

Advantages

- Simple and easy to understand
- It can train neural network in supervised and unsupervised manner.
- It is an effective method to solve the problems fast

Disadvantages

- No guarantee of optimized solution.
- Can give optimised solution to some problems but can't for other problems.

4.2 Dropout Algorithm

Overfitting is a serious problem in the neural network model. A "Dropout" technique solves this problem and improves the performance. It can apply to both supervised learning and unsupervised learning.

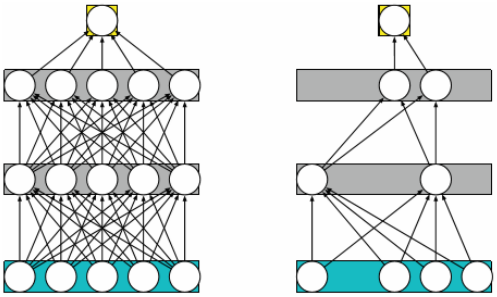

Figure 4.4 Diagram for a neural network with dropout. Left: a neural network with two hidden layers. Right: the same neural network after applying dropout. [16]

When applying dropout to the deep neural network, the hidden units with a probability of 0.5 and input units with a probability of 0.2 randomly omit. In this hidden units as well as their incoming and outgoing connections are temporarily excluded in the network during the training. Another way of looking at dropout training is that dropout generates many different virtual subsets of the original neural network and then these subsets are averaged to give a final network that generalizes well. [16] [17] [18]

Advantages

- Simple and easy to understand
- It can use in both supervised learning and unsupervised learning.
- Solves overfitting problem of neural network
- Improves performance
- Dropout algorithm is more effective than classical data-driven algorithms.
- Make neural network more robust.

Disadvantages

- Sparse and noisy input to dropout simply reduce the amount of data available for learning. [19]

4.3 Summary of Algorithms:

Summary of training algorithm is given as follows.

Table 4.1 Summery of Algorithms

Algorithm / parameters	**Greedy Algorithm**	**Dropout Algorithm**
Key concept	Divide large problem to subproblem.	Generate different virtual subnets of original neural network.
Solution	No guarantee of optimized solution.	Can give optimized solution.
Optimization	Optimization depends on nature of problem.	No such problem.
Drop nodes	Does not dropout nodes	It dropout nodes.
Advantages	Faster than other algorithms in optimization	Solves problem of overfitting and improve performance
Applications	Task Scheduling, Network routing, Graph colouring, etc.	Image Classification, Automatic prediction of heart rejection, Wind turbine Gearbox failure identification

5 Conclusion

Deep neural network has more number of hidden layers than standard neural network, training such neural network is tricky. Artificial Neural Network train with the help of Activation Function and Training Algorithms.

Nowadays ReLu is most widely used because unlike Sigmoid and Tanh it does not face problem of vanishing gradient. But it may face problem of dead neuron that can be overcome by using Leaky ReLu. But use of activation function is depends on type of problem. If there is classification problem then sigmoid can solve this classification problem more efficiently and in faster way than ReLU.

In case of selection of algorithms also need to check type of problem. If problem can be divided into same small problem then that problem can be solved by Greedy algorithm but it doesn't give guaranteed optimised solution for each problem. Greedy and Dropout both algorithms improve performance of neural network but problem of overfitting can be solved by Dropout algorithm only.

6 References

1. Wikipedia contributors. "Machine learning." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 24 Oct. 2017. Web. 29 Oct. 2017
2. Wikipedia contributors. "Deep learning." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 23 Oct. 2017. Web.
3. Schmidhuber, Jürgen. "Deep learning in neural networks: An overview." Neural networks 61 (2015): 85-117.
4. Deeplearning4j Development Team. Deeplearning4j: Open-source distributed deep learning for the JVM, Apache Software Foundation License 2.0. http://deeplearning4j.org
5. "NEURAL NETWORKS by Christos Stergiou and Dimitrios Siganos **",** https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html, September 2017.
6. "opening up deep learning for everyone", http://www.jtoy.net/2016/02/14/opening-up-deep-learning-for-everyone.html, October 2017.
7. Lau, Mian Mian, and King Hann Lim. "Investigation of activation functions in deep belief network." Control and Robotics Engineering (ICCRE), 2017 2nd International Conference on. IEEE, 2017.
8. "Understanding Activation Functions in Neural Networks", https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0, September 2017.
9. "Activation functions and it's types-Which is better?", https://medium.com/towards-data-science/activation-functions-and-its-types-which-is-better-a9a5310cc8f, September 2017.
10. "The Vanishing Gradient Problem", https://medium.com/@anishsingh20/the-vanishing-gradient-problem-48ae7f501257, September 2017.
11. Qian, Sheng, et al. "Adaptive activation functions in convolutional neural networks." Neurocomputing (2017) .
12. "What is a piecewise linear function?", https://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.2/ilog.odms.cplex.help/CPLEX/UsrMan/topics/discr_optim/pwl/02_pwl_defn.html, October 2017.
13. Liu, Jun, Chuan-Cheng Zhao, and Zhi-Guo Ren. "The Application of Greedy Algorithm in Real Life." DEStech Transactions on Engineering and Technology Research mcee (2016).
14. Wikipedia contributors. "Greedy algorithm." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 19 Apr. 2017. Web. 28 Oct. 2017.
15. Wang, Jian-Guo, et al. "A mothed of improving identification accuracy via deep learning algorithm under condition of deficient labeled data." Control

Conference (CCC), 2017 36th Chinese. IEEE, 2017.

16. Tong, Li, et al. "Predicting heart rejection using histopathological whole-slide imaging and deep neural network with dropout." Biomedical & Health Informatics (BHI), 2017 IEEE EMBS International Conference on. IEEE, 2017.
17. Wang, Long, et al. "Wind turbine gearbox failure identification with deep neural networks." IEEE Transactions on Industrial Informatics 13.3 (2017): 1360-1368.
18. Ko, ByungSoo, et al. "Controlled dropout: A different approach to using dropout on deep neural network." Big Data and Smart Computing (BigComp), 2017 IEEE International Conference on. IEEE, 2017.
19. "Regularizing neural networks with dropout and with DropConnect", http://fastml.com/regularizing-neural-networks-with-dropout-and-with-dropconnect/ , October 2017.