Investigate Different Activation Functions used in Deep Neural Network

XJTLU
Suzhou Jiangsu Province China
1406090

ABSTRACT

Machine learning achieves great success in image processing, big data analysis and nature language processing. It is proved that deep structure is better than shadow structure in real use. Activation function is a vital parameter in DNN. It provides non-linearity, enables the architecture to represent non-linear functions. Many different activation functions are put up to enhance the use of DNN. Different activation has different properties. In this case, a very important thing to use DNN is to select a suitable activation function. It highly affects the effectiveness, validation, and efficiency of the network. And training DNN is also not easy. Investigate properties of activation function can help people tuning parameters and get a good learning outcome. This project includes some experiments to investigate inner properties of activation function and provide some suggestions base on the experiment result to help people use DNN to solve real world problem.

INTRODUCTION

Background

Machine learning was put up in late 20th and made great success in many computer areas like computer vision, object identification, predication and nature langue processing [1-3]. In real use, the scale of machine learning architecture should fit the abstract-level of the problem. Too large scale architecture applied to solve low complex level problem may cause over-fitting problem and too small scale architecture cause under-fitting problem [4]. CNN enables people to build large deep neural network. Even though, as universal approximation theorem indicates, the single layer is capable of represent most of interesting functions it is proved experientially and theoretically that deep architecture has much more superiorities than shadow neural network [5, 6]. And deep architecture meets tuning problems because of vanishing, overfitting or under-fitting [6].

Problem Statement

Activation function is a vital parameter in deep artificial neural network. It provides non-linearity which enables machine learning module to represent non-linear function. Then, the module has capability to solve real world problem. Activation function strongly affects effectiveness and efficiency of the neural network. In past decades, many activation functions are purposed to enhance the performance of machine learning. However, there is still not a perfect activation function. Different activation function has different properties and the selection depends on the complexity and the aim. Even the most classical function, sigmoid, is still perform better than other activation functions sometime. So, how to select activation function is a problem in real world use. Another problem is about training. The training strategies are different for different activation application because of different properties [7]. For example, initiate by random distribution with mean 1 is good for sigmoid but it not performs good for ReLu (It hardly work actually). Study the properties of different activation application can help to solve these two problems and enhance the usability and training efficiency. In this project, some investigation study is done to explore inner properties of different activation function application in DNN and some training strategy is purposed based on the experiment results.

Literature Review

Activation Function: Artificial Neuro: A node structure in artificial neural network is as figure1 shown. The activation function is a nonlinear function to simulate the mechanism of real neural cell. It maps weighted inputs into -1 to 1 or 0 to 1 and the output sends to next nodes (or itself). This procedure is called forward propagation. There are one or more neural in each layer and the nodes are sparsely or fully connected between layers. The information is forward layer by layer and the results are exported at the final output layer. The activation function is a vital parameter in neural. It plays an important role in effectiveness, efficiency and training strategy of the architecture which will be detailed next.

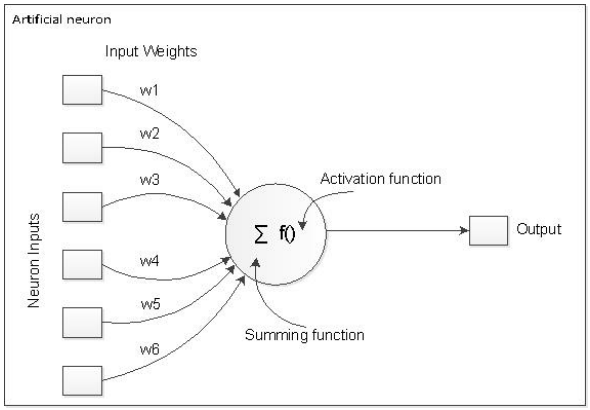

Figure 1. Artificatial Neural Node

Activation Function: As mentioned before, activation function is used to provide non-linearity activate nodes to stimulate the mechanism of real neuro. For example, the taylor expansion of logistic sigmoid function is shown in equation1.

$$sigmoid(x) \approx \frac{1}{2} + \frac{z}{4} - \frac{x^2}{48} + \frac{x^5}{480} + O(x^6) \qquad (1)$$

And the input equation is as shown. The X is matrix of features, W is weight matrix b is a constant for bias.

$$Z = XW + b \tag{2}$$

As these two equations (equation1 and equation2) we can see, the activation provides non-linearities between input features (power 3 element and power 5 elements in this case). And power 3 combines three features together, power combine 5 features together, and training selects the combination which affects the learning outcome.

It bridged the gap between computational neuroscience and machine learning modules. As study, the easier converged and optimized activation function is preferred [8]. And there are many kinds of activation functions.

Sigmoid function and hyperbolic tangent function are very traditional and still used in recurrent neural networks (RNN). The equitation and plots are as shown.

$$sigmoid(x) == \frac{1}{1+e^{-x}} \tag{3}$$

$$tanh(x) = 2sigmoid(2x) - 1 \tag{4}$$

The plot is as in figure2. From the figure, sigmoid defined

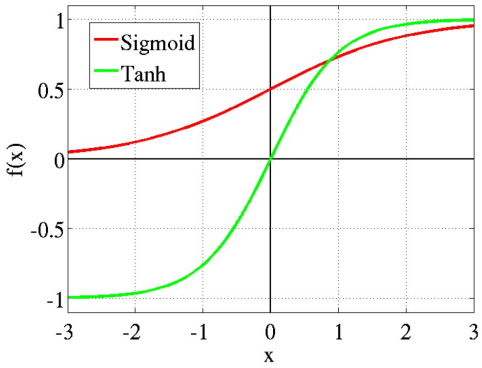

Figure 2. Sigmoid and Tanh plot

from negative infinite to positive infinite. It is bounded to 0 also 1 and it is differentiable everywhere. This function can help to find loss of global minimum. However, there are two shortcomings of sigmoid activation function. The first one is nonzero-centering properties which increase the number of iterations during training [9]. This issue will detailed in the experiment discussion next. Another is that, the saturated property cause both vanishing problem and converge efficiency problem. As the figure2 shown, the differential coefficient is small when the plot close to 0 or 1. So, the gradient is vanished, and learning rate is small which makes the module hare to converge. The update method is backpropagation with help of chain rule. So, the gradient keeps vanishing from the back layers to the front layers which harms the effectiveness and efficiency of training [10]. Hyperbolic tangent is variant of logistic sigmoid function which put up to replace the sigmoid function. It, as the figure shown, is zero-centering which speed up the converge but the saturated property is still exit.

ReLu: To speed up training, rectified linear unit (ReLu) was first purposed for restricted Boltzmann machines in 2010 [11]. And win great success in convolution neural network (CNN) in 2011. It makes great progress to find global minimum and get better learning outcomes in computer vision and sentiment analysis [12]. The function and plot are as shown in equation5 and figure3.

$$g(x) = max(0, x) = \begin{cases} x & if\ x \geq 0 \\ 0 & if\ x < 0 \end{cases} \tag{5}$$

One of big advantage of ReLu is sparse property. Sparse characteristic has advantage in machine learning and data processing [13]. The ReLu enables architecture represents sparsely. When weighted input is negative, the relevant neuro is died and keep output zero. There are also some other advantages like computation and avoid vanishing.

However, one of potential problem is sparsity may hurt the ability for optimization. Some initialization strategy should be applied because of this property. And another potential problem this property cause is update problem. As the plot shown, the output of ReLu is positive. So, for every batch of learning, all the weights increase or decrease in union. It can not allow some weights increase and some decrease in a batch. This make z plot update slow done the speed to converge and may reduce the finial performance.

Derivate functions of ReLu are purposed to improve the performance. Parameter ReLu is one of derivate purposed in 2015. The positive domain is same with ReLu and negative part is a linear function with a trainable slop alpha [14]. And leaky ReLu is PReLu with a small alpha which purposed in 2013 (alpha is commanly equals to 0.01) [8]. The equation is as shown.

$$g(x) = max(0, x) = \begin{cases} x & if\ x \geq 0 \\ \alpha x & if\ x < 0 \end{cases} \tag{6}$$

As figure3 shown below, PReLu is not saturate at negative infinite which is very different form ReLu, Sigmoid and hyperbolic tangent function. This property harms the robust of system and may reduce finial performance in real world application.

Then exponential linear unit (ELU) is put up [15]. As equation7 shown, it also has same plot with ReLu when x¿=0 and it is an exponential plot when x¡0. As the figure3 shown, contrast to LRelu, ELU saturate to negative value to avoid shortcomings of LReLu mentioned before.

$$f_{ELU}(h_k) = \begin{cases} h_k \\ \alpha(exp(h_k) - 1) \end{cases} \tag{7}$$

EXPERIMENT AND DISCUSSION

There are two parts of experiment. For the first part, general experiment about all activation functions are done. I use same architecture, same batch and best performance of

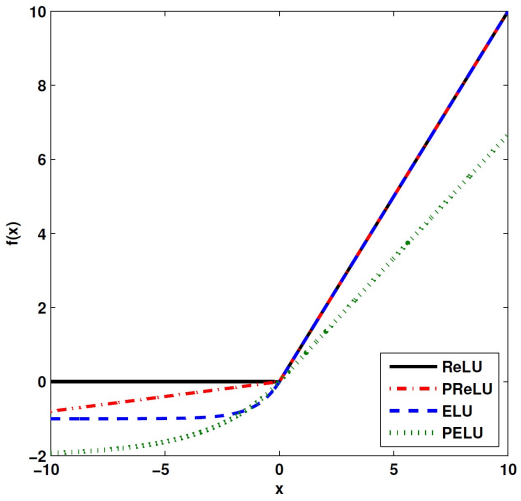

Figure 3. ReLU and its derivative function plots

the learning outcomes, time consuming, loss and error are recorded. Then, some results can be concluded, and some hypothesis can be put up. For the second part, some experiments developed to study the individual properties or prove the hypothesis put up on the first experiment.

Experiment Setup

I considered two architecture and two image data set detailed below. The data sets are both divided into training set, testing set and validation set. There are two architectures applied in this experiment. The one is with 7 layers and another is 22 layers to validate the experiment result on both large scale DNN and small scale DNN.

- MNIST digital recognition data set is a classical digit data set. It contains 70000 samples and divided into 50000/ 10000/10000 for training, validating and testing. The samples are labeled besides there are 10 classes about 0 to 9 digits. Images in data set is 28×28grey code picture.

- CIFAR10 image set is provided by Kirzhevsky and Hinton in 2009. There are 60K labeled images into 10 classes. They are divided into 50k /5k /5k for training, validating and testing. Images in data set are $32 \times 32 \times 32$ RGB image.

- CIFAR100 is a more complex image data set compared with CIFAR-10. There are 100 classes and each class contains 500 image for training and 100 for testing.

- Variant Lenet-5 architecture is applied in this project to test study the situation in small scale architecture. The layer information is as shown below.

The structure of Variant LeNet	
Input Layer	28×28
Convolution layer1	5/5/1/32. The kernel size is 5×5. The depth is 1. There are 32 kernels. The sampling step is 1 for each dimensionality (batch, height, width, output channels).
Pooling layer1	Max-pooling for size 2×2 and it is padding to the input size of 28×28
Convolution layer2	5/5/32/64. The kernel size is $5 \times 5 \times 32$ and there are 64 kernels in this layer. The sampling step is also 1 for each dimensionality (batch, height, width, output channels).
Pooling layer	Max-pooling for size 2×2 with padding.
Fully connected layers$\times 2$	There are two fully connected layer for mapping.
Classifier	Softmax Classifier

- GoogleLeNet inception V4 is a 22-layer architecture. In this project, it is used to study activation function properties in large scale neural networks. The structure is as shown in figure4.

Experiment1:The general performance and training properties of activation functions

In this experiment, I study the general properties of different activation function application. Sigmoid, tanh, ReLu, LReLu and ELU are applied on variant LeNet-5 architecture to solve digital classification problem. The accuracy and loss are recorded as shown.

Experiment results: The accuracy and loss of MNIST digital set by variant lenet-5 architecture with different activation function is as shown below. And the plots of accuracy by each interaction is shown in figures. And all activation functions are applied on variant LenNet-5.

Name	Smoothed	Value	Step	Time	Relative
ELU	0.9893	0.9893	19.90k	Sun Dec 16, 23:22:02	2m 28s
LReLu	0.9914	0.9914	19.90k	Sun Dec 16, 19:35:49	2m 55s
relu	0.9926	0.9926	19.90k	Wed Dec 12, 13:27:11	2m 27s
sigmoid	0.9864	0.9864	19.90k	Mon Dec 17, 08:28:21	2m 31s
tanh	0.9903	0.9903	19.90k	Sun Dec 16, 16:06:03	2m 28s

Figure 4. Final Accuracy

Several things are got form figure4. Firstly, ReLu achieves best performance compared with all the other activation functions. Variants of ReLu (LReLu, ELU) performs better than sigmoid or variant of sigmoid function. However, hyperbolic tangent function performs better than ELU. And batch 100 makes it hard to find the global minimum.

	Name	Smoothed	Value	Step	Time	Relative
	ELU	1.000	1.000	19.90k	Sun Dec 16, 23:22:02	2m 28s
	LReLu	1.000	1.000	19.90k	Sun Dec 16, 19:35:49	2m 55s
	relu	1.000	1.000	19.90k	Wed Dec 12, 13:27:11	2m 27s
	sigmoid	0.9800	0.9800	19.90k	Mon Dec 17, 08:28:21	2m 31s
	tanh	1.000	1.000	19.90k	Sun Dec 16, 16:06:03	2m 28s

Figure 5. Final Loss

Compared with figure4 (loss is shown below in figure5), we can get ELU, LReLu, relu and tanh all achieve zero loss but not the accuracy.

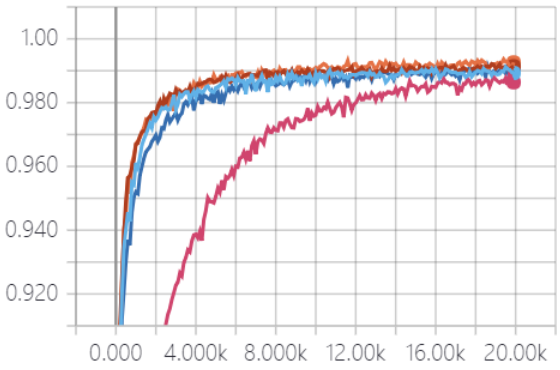

Figure 6. Error Plots

Thirdly, as plots in figure6 shown, ELU,ReLu,tanh and LReLu converged with similar iteration. The sigmoid functions converged much slower than all the other activation functions.

Discussion and Hypothesis: Combine the first result and the second result we can get there are some overfitting issues. The loss is zero for relu, Lrelu, ELU and tanh but the errors are not zero. Small loss and big error is what so called overfitting. Besides, it shows the importance of spares property of a neural network and the superiorities of ReLu sparse representation. What is more, superiorities of ELU and LReLu is not performed. The hypothesis is the complexities of problem is not enough.

For, the third result, it is got that sigmoid is coverage slow. The hypothesis is because of the nonzero-centering property. In CNN, the backpropagation is by chain rule. The equation is as shown in equation8. It propagates the gradient form front layer to initial layer.

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial w_i} = x_i \eta \frac{\partial L}{\partial f} \tag{8}$$

And the update by gradient is as shown in equation9.

$$W_i = w_i - \eta x_i \frac{\partial L}{\partial f} \tag{9}$$

η is learning rate and it is a positive constant. And all the x_i is positive because of sigmoid function so, the updates sign(positive or negnative) for all dimensions are same which makes z updatas. It slows the training speed. 'Z' updates is slow. It is very easy to observe on 2 dimentation but things are same in high dimentations. It is as the figure shown.

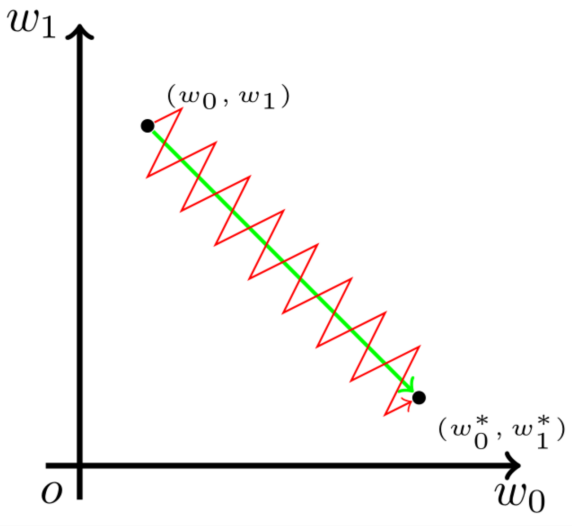

Figure 7. 'Z' Plots

Experiment2: overfitting for ELU and LReLu

It is hypothesized that ELU and LReLu has better performance in complex model to solve high-abstract problem. So, ELU, LReLu and ReLu are applied on GoogLeNet inception V4 to solve CIFAR10 classification problem. The test error is as shown. LReLu and ELU performs better than ReLu.The result is as table shown below.

	ReLu	ELU	LReLu
Accuracy	7.93 ± 0.4	9.38 ± 0.2	8.79 ± 0.3

There is another experiment based on CIFAR100 published in 2015 [15]. The result is as shown, and it matches the result of my experiment. The performs of ELU is 5% better than ReLu. And performance of LReLu is 3% better than ReLu.

Discussion: Combine the result of first experiment and second experiment we can get ReLu sparse representation have positive effect on avoiding overfitting. However, on complex and high-level abstract problem, the things may different. Dying nodes can reduce the optimization ability of the architecture. Some output about negative weighted inputs is still needed.

Experiment3:Initialization of Rectifier Neural Networks

Because ReLu is the most popular activation function [16]. Study the property of ReLu is very helpful to solve real word problem. This experiment is designed to see the initialization strategy of ReLu activation function. It is recommended that the best way for ReLu is normal distribution with standard deviation 2/n (n is the number of nodes in such layer) [7]. So, there are two part in this experiment. The first one is to prove the superiorities of normal distribution initialization

and the second one is to find the best standard deviation for normal distribution initialization.
Three strategies are applied on this experiment. The first is random from 0 to 1 the second is start at some constants. And the third one is normal distribution with standard deviation is 0.1.
The first strategies accuracy is about 0.098, it finds nothing.
The second one and the third one result is as shown in table below.

Constant			
	0	0.1	0.2
Accuracy	0.96 ± 0.24	0.93 ± 0.12	0.87 ± 0.36

The normal distribution with mean is 0 is as shown.

Normal Distribution				
	0	0.05	0.125	0.2
Accuracy	0.994 ± 0.027	0.996 ± 0.002	0.924 ± 0.16	0.86 ± 0.13

Discussion: The second result is less then third one. So, normal distribution is the best strategy for ReLu.
And, as different standard deviations the result, it is shows that, 0 to $\frac{1}{n}$ has less impacts compared with more than $\frac{1}{n}$. The performance significantly reduces when standard deviation is more than $\frac{1}{n}$. However,$\frac{1}{n}$ is not state of art choice. The best choice is close to $\frac{1}{n}$. Some more effort to find it out.

CONCLUSION

This project developed some experiments to investigate the properties of different activation function in Deep neural network. Several things are found in this project. The first one is for saturation activation function (sigmoid and tanh) sigmoid is low performance and it lacks the ability to find the global minimum loss. What is more, it is hard to converge because of the nonzero-centering property. The second one is ReLu is a good activation function to solve problems. Sparse representation is helpful to overcome overfitting especially in low abstract problem. Besides ELU and LReLu have better performance in high abstract problem by overcoming the dying problem of ReLu. And the last thing is, for ReLu the best training strategy is normal distribution and the best standard deviation is around $\frac{1}{n}$.
Some experience can be concluded form the investigation. The first one is, in CNN, ReLu is the best choice to start. And, if the performance is not good and it is not because of overfitting problem (the loss is bigger than accuracy). ELU and LReLu can be used do optimization. The next is, when using ReLu, normal distribution with standard deviation 1/n is a good choice to start. It is high possibility not the best choice, but it is close. This method makes it easier to find the optimized standard deviation around $\frac{1}{n}$.

Evaluation

This project investigates properties of activation function. In this project, I applied two deep CNN architectures and two data sets to do the experiment. Some suggestions are put up based on the experiment result. However, there are still some shortcomings. First, because of equipment constrain, I use googleLeNet and CIFAR-10 to study the situation in high-level abstraction problem, the complexity is actually not enough. Secondly, I find, compared with ReLu, ELU and LReLu may have advantages to solve high-abstraction problem by overcoming the dying problem. However, I do not test the robust of ELU and LReLu. The output of ReLu is 0 to 1 and ElU ouput is also bounded but not LReLu. But LReLu unbounded output properties may not good for some large negative weighted input.

Future Study

For future, the experiment about larger scale CNN should be made to see the ability of activation functions on solving complex problem to make the result more convinced. Besides, robust is a very important property of activation function but it is not included in this project. And only two data sets are applied in this experiment. More data sets about image and other areas is needed to prove the validation of the experiment result.

REFERENCE

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097-1105.

[2] S. Zhou, Q. Chen, and X. Wang, "Active deep networks for semi-supervised sentiment classification," in Proceedings of the 23rd International Conference on Computational Linguistics: Posters, 2010, pp. 1515-1523: Association for Computational Linguistics.

[3] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, D. I. J. C. Fotiadis, and s. b. journal, "Machine learning applications in cancer prognosis and prediction," vol. 13, pp. 8-17, 2015.

[4] A. Ng, "Machine Learning Yearning," ed, pp. 49-10, 2017.

[5] T. Chen and H. J. I. T. o. N. N. Chen, "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems," vol. 6, no. 4, pp. 911-917, 1995.

[6] Y. J. F. Bengio and t. i. M. Learning, "Learning deep architectures for AI," vol. 2, no. 1, pp. 1-127, 2009.

[7] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010, pp. 249-256.

[8] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in Proc. icml, 2013, vol. 30, no. 1, p. 3.

[9] H. Chung, S. J. Lee, and J. G. Park, "Deep neural network using trainable activation functions," in Neural Networks (IJCNN), 2016 International Joint Conference on, 2016, pp. 348-352: IEEE.

[10] B. Xu, R. Huang, and M. J. a. p. a. Li, "Revise Saturated Activation Functions," 2016.

[11] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807-814.

[12] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pp. 315-323.

[13] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in Advances in neural information processing systems, 2007, pp. 801-808

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026-1034.

[15] D.A. Clevert, T. Unterthiner, and S. J. a. p. a. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," 2015.

[16] H. Ide and T. Kurita, Improvement of learning for CNN with ReLU activation by sparse regularization, in Neural Networks (IJCNN), 2017 International Joint Conference on, 2017, pp. 2684-2691: IEEE.