Research on overfitting of deep learning

Haidong Li, Jiongcheng Li, Xiaoming Guan, Binghao Liang, Yuting Lai, Xinglong Luo
Guangdong Planning and Designing Institute of Telecommunications CO, LTD
Guangdong Guangzhou 510630

***Abstract*—Deep learning has been widely used in search engines, data mining, machine learning, natural language processing, multimedia learning, voice recognition, recommendation system, and other related fields. In this paper, a deep neural network based on multilayer perceptron and its optimization algorithm are studied. MNIST handwritten digital datasets were used to verify the reliability of the model and the optimization algorithm. The important correlation between the overfitting and the activation functions, network structures, training epochs, learning rates is verified. The study of overfitting is of great significance to reduce generalization error. This paper proposes an innovative activation function called: modified-sigmoid which is based on the well-known sigmoid function. This activation function can effectively improve the accuracy of the model and inhibit the overfitting problem.**

***Index Terms*—Deep learning; Neural network; Overfitting; Modified-sigmoid;**

I. Introduction

The theory and application of deep learning have been developed rapidly in recent years [1]. Usually, We hope that the training error and generalization error of the neural network are low enough. The overfitting and underfitting problems during the training of neural network have a dominant influence on the generalization ability of a deep neural network model [2]. One of the most important factors, which can cause the overfitting problem of machine learning, is the complexity of the model which depends on the number of hyperparameters.

A. Model

1) Multilayer perceptron: The multilayer perceptron is a feedforward neural network [3]. A multilayer perceptron is a mathematical function mapping some set of input values to output values. Each layer of neurons can receive input from the previous layer and produce output to the next layer [3]. The first layer is called the input layer, the last layer is called the output layer, and the layers in between is called the hidden layer. There are no feedback connections in this network. The signal propagates one-way from the input layer to the output layer, which can be represented by a directed acyclic graph.

2) Activation function: Each layer of the neural network needs a nonlinear function, which is called activation function, before generating the output signal. when there is no nonlinear activation function, this layer of neural network is transformed linearly, so that no matter how many layers the network contains, the final output can be expressed by the linear transformation of the input. Therefore, it has no difference from the single layer neural network. Nonlinear transformation can greatly increase the information which is stored in the network, and the mode of nonlinear transformation is not fixed [4]. The effect of activation function in neural networks is to generate nonlinear decision boundaries by nonlinear combinations of weighted input. There are three commonly used activation functions: Sigmoid activation function, Tanh activation function, and ReLu activation function.

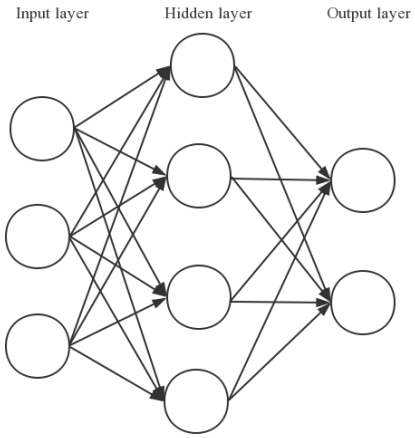

Fig. 1. Structure diagram of a multilayer perceptron

3) Sigmoid: Sigmoid activation function takes the range of 0 to 1, which is an S-shaped curve and is easy to understand. But the existing shortcomings, including 1) the problem of disappearing gradient; 2) the none zero centered output, which will make the gradient update inconsistency in different directions; 3) easy saturation makes gradient vanish; 4) slow convergence speed.

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

4) Tanh: The output of the activation function is centered around 0 and has a range of (-1, 1). The Tanh function is very similar to the Sigmoid function, with similar advantages and disadvantages. The main difference is that the range of the former is (-1.0,1.0).

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

5) ReLu: Relu function is a piecewise function. Its convergence speed is faster than Tanh. It prevents the problem of gradient disappearance [5]. The problem of ReLu is that some gradients are weak and die during training. This may cause the neurons not being reactivated at any data point.

978-1-7281-6092-4/19/$31.00 ©2019 IEEE
DOI 10.1109/CIS.2019.00025

$$ReLu(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

6) Modified-Sigmoid: The sigmoid activation function is a near S-shaped curve. Modified-sigmoid activation function is an improvement of Sigmoid activation function. By defining a hyperparameter , Modified-sigmoid activation function is shown as follows:

$$Modified - Sigmoid(x) = \begin{cases} x & -\omega \leq x \leq \omega \\ Sigmoid & x < -\omega \ or \ x > \omega \end{cases}$$

The function images of Sigmoid activation function, Tanh activation function, ReLu activation function and Modified-sigmoid activation function ($\omega = 2$) are shown in figure 2.

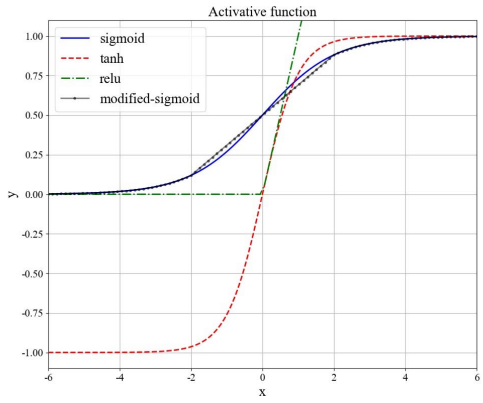

Fig. 2. Diagram of four activation functions

B. Cost function

Cost function is used to measure the deviation between the predicted result and the actual result. Handwritten digit recognition is a multi-classification problem. Cross-entropy [6] algorithm is used to construct the cost function. N is the number of categories, y^{label}, y^{pred} are probability distributions.

$$crossEntropy = -\sum_{i=0}^{n} y_i^{label} logy_i^{pred}$$

C. Stochastic gradient descent

A common problem in deep learning is that a large training set can improve the accuracy of the model, but increase the difficulty of calculation [7]. The core idea of the stochastic gradient descent is that the gradient can be approximated by a mini batch of samples. In each iteration of the training process, samples in each mini batch are uniformly extracted from the training set:

$$\mathbb{B} = \{x^{(1)}, \ldots, x^{(m')}\}$$

The number of mini batch samples m' is usually a relatively small number 2^n. When the number of training samples grows, m' is gradually stable [8]. Even when fitting billions of training samples, only a few hundred samples are used for each gradient update calculation [9]. The calculation of the gradient can be expressed as

$$g = \frac{1}{m} \nabla_\theta \sum_{i=1}^{m'} L(x^{(i)}, y^{(i)}, \theta)$$

With mini batch samples, the estimations are updated by stochastic gradient descent as the following equation, where ε is the learning rate.

$$\theta \leftarrow \theta - \varepsilon g$$

D. Back-propagation

The back-propagation method is based on the gradient descent strategy. Back-propagation adjusts the parameters of the deep learning model towards the negative gradient direction [10]. When Sigmoid is adopted as the activation function, the loss function of cross-entropy is shown as follows:

$$J = -\sum_{i=1}^{K} y_i ln(z_i)$$
$$z_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

In the training set, assume that $y_s = 1$, the rest $y_{y \neq s} = 0$. Two conditions when $k = s$ and $k \neq s$ are discussed in the following section.

When $k = s$:

$$\frac{\partial J}{\partial x_k} = \frac{\partial(-y_s ln(\frac{e^{x_s}}{\sum_{j=1}^{K} e^{x_j}}))}{\partial x_s}$$
$$= -y_s \times \frac{\sum_{j=1}^{K} e^{x_j}}{e^{x_s}} \times \frac{e^{x_s} \times \sum_{j=1}^{K} e^{x_j} - e^{2x_s}}{(\sum_{j=1}^{K} e^{x_j})^2}$$
$$= -y_s \times \frac{\sum_{j=1}^{K} e^{x_j} - e^{x_s}}{\sum_{j=1}^{K} e^{x_j}} = -y_s(1 - z_s)$$

When $k \neq s$:

$$\frac{\partial J}{\partial x_k} = \frac{\partial(-y_s ln(\frac{e^{x_s}}{\sum_{j=1}^{K} e^{x_j}}))}{\partial x_s}$$
$$= -y_s \times \frac{\sum_{j=1}^{K} e^{x_j}}{e^{x_s}} \times \frac{e^{x_s} \times \sum_{j=1}^{K} e^{x_j} - e^{2x_s}}{(\sum_{j=1}^{K} e^{x_j})^2}$$
$$= -y_s \times \frac{\sum_{j=1}^{K} e^{x_j} - e^{x_s}}{\sum_{j=1}^{K} e^{x_j}} = -y_s(1 - z_s)$$

In conclusion, after optimization, the gradient of the model is:

$$\frac{\partial J}{\partial x_k} = z_k - y_k$$

II. Experiment

In this paper, the deep neural network was trained on the MNIST datasets which contain 60,000 training samples and 10,000 test samples. Firstly, we train the deep neural network to get the baseline. After that, we study the influence of network structure, training epochs, learning rate, and activation function on the model accuracy and overfitting.

A. Algorithm

Model parameters were trained with stochastic gradient descent in this paper. Assume that the total number of samples

is S_n , which were divided into groups and the number of samples in each group was $\frac{S_n}{N}$. The model parameters were adjusted after each batch. The detailed steps are as follow:

1) Initialize the weight matrix of neurons network;
2) Divide all training samples into N groups;
3) Calculate the prediction and training error through the forward propagation of the neural network;
4) According to the back-propagation method 1.5, calculate the weight optimization gradient $\frac{\partial J}{\partial x_k} = z_k - y_k$ and optimize the weight matrix;
5) Each training epochs contains $\frac{S_n}{N}$ steps. After a training epoch, we evaluate the performance measures using a test set of data.

B. Result

MNIST datasets were used for training and testing the model in this paper. The number of samples in the training set was 55,000, and the number of samples in the test set was 10,000. The neural network model architecture contains three layers of neurons. The first layer is called input layer. The resolution of each handwritten digital image in the MNIST datasets is 28*28, with a total of 784 pixels. The second layer is called hidden layer, which contains 60 neurons. The third layer is called output layer. Handwritten digits are all Arabic numerals, so the number of neurons in the output layer is ten. In this paper, we adopt the Python as program language and the neural network model is based on the TensorFlow deep learning framework. The detailed experimental results are discussed in the following sections.

1) Influence of training epochs on model accuracy: In this section, Sigmoid function was adopted as the activation function. The minimum sample size of stochastic gradient descent is 64 and the learning rate is 1.0. As shown in figure 3, the test accuracy of our model was no longer significantly improved after 15 training epochs.

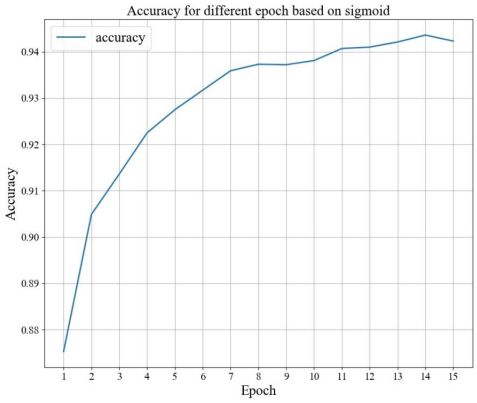

Fig. 3. Influence of training epochs on model accuracy

2) Influence of model architecture on model accuracy: In this part, the number of training epochs was fixed to 15. The minimum sample size of the stochastic gradient descent was 64 and the learning rate was 1.0. Deep neural network with 1-4 hidden layers was used to build the model. The number of neurons in each hidden layer was 60. used to test the accuracy of the model with 1 4 hidden layers respectively. As shown in figure 4, when the number of training epochs is 15, the model with 4 hidden layers has the highest training accuracy. When the number of training rounds is 40, the model will overfit with the increase of hidden layers.

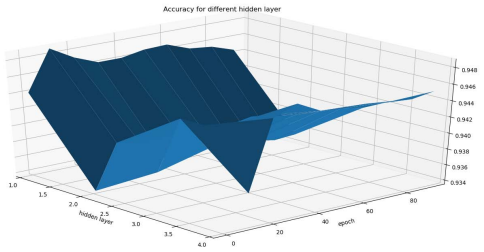

Fig. 4. Influence of training epochs and model architecture on model accuracy

3) Influence of learning rate on model accuracy: In this part, the number of training epochs was set to 15. The minimum sample size of the stochastic gradient descent was 64 and the learning rate was set to 0.001, 0.01, 0.1 and 1.0, respectively. As shown in figure 5, when Tanh was used as the activation function, the model has the highest accuracy with a learning rate of 0.1. When the learning rate increased to 1, the accuracy of the model will decrease because of overfitting.

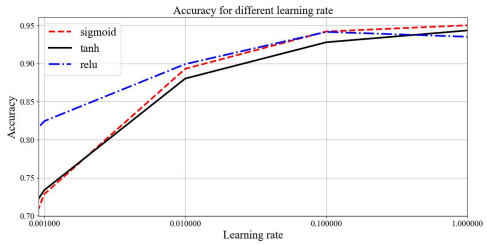

Fig. 5. Influence of learning rate on model accuracy

4) Effect of activation function on model accuracy: In this part, we mainly study the influence of activation functions on the accuracy of our model, as shown in figure 6. Four different activation functions were compared in this part. As shown in figure 6, when the model architecture, training epochs, and other hyperparameters are fixed, activation function plays an important role in preventing the model from overfitting. Herein, we propose an innovative activation function called Modified-sigmoid. After optimization, we found that our neural network possesses the best test accuracy with hyperparameter $\omega = 2$. The results show that our Modified-sigmoid function can effectively improve the accuracy of the neural

network and prevent the overfitting of the neural network to some extent.

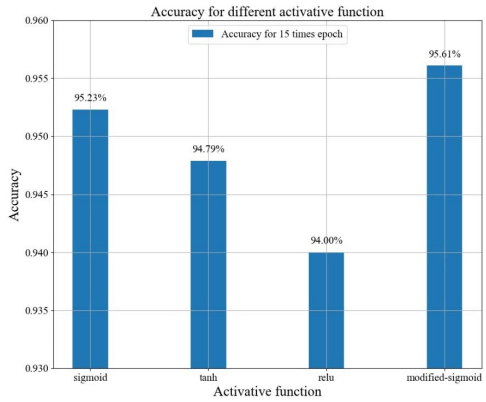

Fig. 6. Effect of activation function on model accuracy

III. Conclusion

In this paper, we propose a new activation function to prevent the model from overfitting. At the same time, the structure of neural network, the number of training epochs and the learning rate also have a great influence on the overfitting which has been intensively studied in this paper. MNIST datasets were used to test the accuracy of our algorithm. We found that the accuracy of the model no longer improved significantly when the number of training epoch exceeded 15. Increasing the number of hidden layers can improve the accuracy of training set, but cause overfitting in test set. A single hidden layer with 60 neurons has been proved to have best performs in test set. In addition, through a lot of studies, we found that the model has high accuracy and fast convergence when we set the learning rate equal to 1. On the basis of the above, we innovatively propose a new activation function of neural network called: modified-sigmoid. The results show that this activation function can effectively improve the accuracy of the model and inhibit the overfitting issue. Therefore, this modified-sigmoid function has broad prospects in deep neural network. However, this paper only studies the overfitting problem of deep neural network on image datasets. In the next step, we will use non-image datasets to verify the performance of our activation function. Besides, we will strictly prove in mathematics that modified-sigmoid, as an activation function, can prevent overfitting and improve the accuracy of deep neural network.

References

[1] Deep Learning [M]. People's Posts and Telecommunications Publishing House, Ian Goodfellow, 2017.

[2] Machine Learning [m]. Tsinghua University Press, Zhou Zhihua, 2016.

[3] Research on model building and overfitting of deep learning [J]. Tao Li, Yang Shuo, Yang Wei. Computer Times. 2018(02).

[4] A neural network model for overfitting problem in deep learning [J]. Liu Danfeng, Liu Jianxia. Journal of Natural Science of Xiangtan University. 2018(02).

[5] Analysis and research on the overfitting problem of image recognition based on the convolution neural network [J]. Xie Luyang, Xia Zhaojun, Zhu Shaohua, Zhang Daiqing, Zhao Fengkui. Software engineering.

[6] Overfitting of convolutional neural networks [J]. Ren Yili, Luo Lu. Information Systems Engineering, 2019(05).

[7] An introduction to neural networks and deep learning. SUK H. Deep Learning for Medical Image Analysis. 2017.

[8] Research Note: Prognostic model research: overfitting, validation, and application. [J]. Retel Helmrich Isabel Ra,van Klaveren David, Steyerberg Ewout W. Journal of physiotherapy. 2019(4).

[9] Research on the overfitting of convolutional neural networks [J]. Ren Yili, Luo Lu. Information Systems Engineering.2019 (05).

[10] Handwritten numeral recognition based on convolution neural network [J]. Li Sifan, Gao Faqin. Journal of Zhejiang University of Science and Technology (Natural Science Edition). 2017(03).