IMPROVING NEURAL NETWORK BY USING DIFFERENT METHOD

Peini Zhu
College of computer science and engineering, Australia National University
u6125835@anu.edu.au

Abstract. Use neural network to solve a two classes logistic regression problem, predict the mushroom whether it is poisonous. Improving the network model by comparing different preprocessing data method, comparing different activation function, comparing different opitmizer gradient algorithm in traditional feedforward neural network by using error back propagation algorithm. Improving the network by using sharing weight bidirectional network model. Analysed the result of training mushroom dataset, and evaluate the convergence ability and generalization ablity of this network model.

Keywords: Adam gradient, bidirectional network, share weight, Relu activation

1 Introduction

Artificial neural network is very popular nowadays, especially the feedforward neural network is very traditional. Usually using error back propagation algorithm to train weight in neural network. In this paper, starting building a simple feedforward neural network with error back propagation to train a logistic regression mushroom dataset. Then, improving the neural network step by step. There are 2 things important in feedforward neural network, one is activation function, the other is optimize gradient algorithm. Try to compare the difference of activation function (Relu, sigmoid, tanh, log sigmoid and SoftMax), and select the best one for this training set to improve the accuracy. Try to compare the convergence of using different optimize gradient algorithm (SGD, Momentum, AdaGrad, RMSprop, Adam), and use the best optimizer to improve the network model. Try to use sharing weight bidirectional network model to improve the testing accuracy, improve the normalization ability of this network model. Last, compare the difference performance of each step, and analysis the reason, also compare to training result of using traditional machine learning method, to analysis the benefit of using neural network training model.

2 Dataset

Choose Mushroom Data Set [1] from the UCI website as training and testing data. This dataset is a traditional classification problem with discrete data feature, which will be good for learning logistic regression problems in neural network.

This dataset has 8124 instances, with 22 attributes. The aim of this classification problem is trying to find out whether this mushroom is poisonous or edible (also tagged as ‘labels’). The shape of this dataset is (8124*23), the first column is 'labels', the column 1-23 are 22 attributes

No null data in this dataset, and 2480 missing values, which are all for attribute ‘stalk-root’. There are 2 classes in this problem. The description of this dataset pointed out that the edible labels (represent as ‘e’ at the first column) has 4208 instances (51.8%), and poisonous labels (represent as ‘p’ at the first column) has 3916 (48.2%).

2.1 Pre-processing Data

This dataset is using 22 features collected from a mushroom, then using these features to estimate edible or poisonous. Each of input data is a vector with 22 features, in raw data file, all features represent as a letter. Usually, we need to transfer the value of each feature into integer. So, use feature ‘cap-shape’ as an example, the easiest way is to use 0 represent ‘b’, 1 represent ‘c’, 2 represent ‘x’, 3 represent ‘f’, 4 represent ‘k’, 5 represent ‘s’.

2.2 Normalization methods

However, most of the data features need to normalization. The pre-processing data will affect the result of training model. Kuźniar and Zając [2] pointed out some methods of data pre-processing for neural networks.

Next, comparing impact of these data pre-processing method in same neural networks.

i. keep origin: no normalization.

ii. scaling to the interval $(0.1 - 0.9)$:

$$x = \frac{0.9x \cdot (x - x_{min}) - 0.1 \cdot (x - x_{max})}{x_{max} - x_{min}} \quad \textbf{(1)}$$

iii. dividing by the range of data

$$x = \frac{x}{data\_size} \quad \textbf{(2)}$$

iv. using polynomial functions

$$x = x^{\alpha} \ (here\ choose\ \alpha = 3) \quad \textbf{(3)}$$

v. using exponential functions

$$x = e^{\alpha x} \ (here\ choose\ \alpha = 3) \quad \textbf{(4)}$$

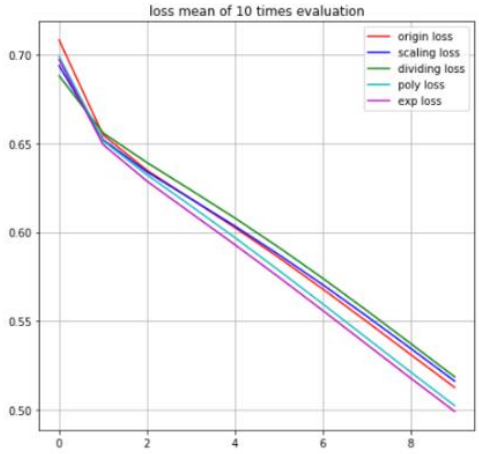

Fig. 1. After 10 times of running test for each normalization functions. This graph shows the mean of 10 times evaluation loss in each normalization functions. The red line represents the origin function, blue line represent the scaling function (math formula 1) and so on. The loss becomes to small when training more data.

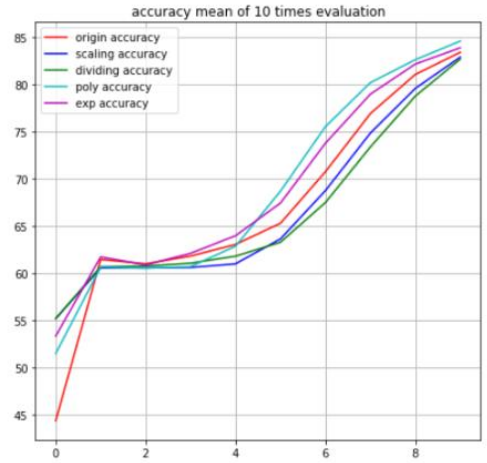

Fig. 2. After 10 times of running test for each normalization functions. This graph shows the mean of 10 times evaluation accuracy rate in each normalization functions. The accuracy rate becomes to greater when training more data.

In this classify problem, from the result figures above, it looks like there are little difference between each normalization methods. So, in this problem, decide this later.

3 Implement a neural network

Building a simple neural network first with the Sigmoid activation function, and the optimizer using SGD. However, the result is not quite good when using a very simple neural network.

Now improving the neural network to get a better training result. There are 2 things important in basic neural network, activation function and the optimizer. There are many choices for activation function, not only Sigmoid function. Moreover, we can use tanh function or rectified linear units (ReLUs) for the hidden layers, especially Relu is popular in deep learning networks nowadays. From now on, there are many improved gradient algorithm used in network optimizer, like Momentum, AdaGrad, RMSProp and Adam, etc.

3.1 Activation function

Using sigmoid as an activation function has the following problems:

i. Gradient saturation. When the function activation value is close to 0 or 1, the gradient of the function is close to zero. In the back-propagation calculation gradient, each residual is close to 0, and the calculated gradient is also close to zero.

ii. The convergence speed of the parameters is very slow, which will affect the training speed. Even if it is a good function for logistic regression, it was rarely used in neural networks.

The tanh function, which is similar with sigmoid function, also has gradient saturation problems. But in some cases, it performance better than sigmoid function, it also will lead a result in low training speed sometimes.

The advantages of the Relu activation function compare to sigmoid and tanh functions are:

i. The gradient is not saturated. So, in the process of back propagation, it could reduce the situation of gradient disappear. Moreover, the parameters of the first few layers of the neural network can be quickly updated.

ii. The calculation speed is fast. In the forward propagation process, the sigmoid and tanh functions exist exponent which will take more time when calculate, while the Relu function only needs to check the threshold which is simple and fast.

SoftMax function is a good function for logistic regression, however it is usually used for the last output layer, it is usually give the probability. So, it is not a traditional activation function.

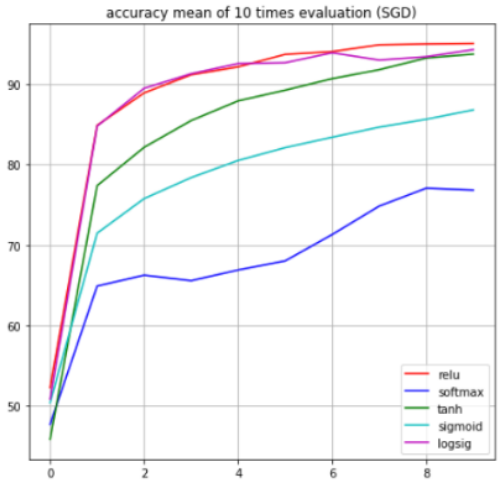

Fig. 3. After 10 times of running test for each normalization functions. This graph shows the mean of 10 times evaluation accuracy rate in each activation functions. The accuracy rate becomes to greater when training more data. Only Relu and log sigmoid activation function performance better, could finally reach 90+ percent in training accuracy rate.

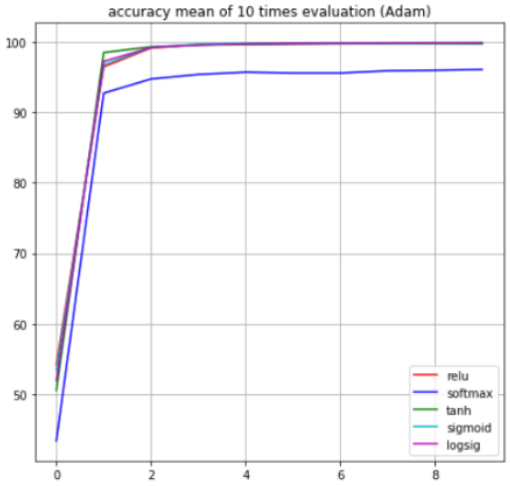

Fig. 4. This graph similar with figure.3. shows the mean of 10 times evaluation accuracy rate in each activation functions, but changed the gradient algorithm. The accuracy rate grows much faster when training more data. Most of activation function could finally reach a better result.

Choosing a proper activation function is very important, even if using a high level gradient algorithm, lots of activation function could finally reach a better result, but when selecting a proper activation function could save training time for big dataset problems. The convergence of network is one of evaluation standard, it reflects the

performance of this network model, when choosing an activation function, need to consider the convergence of this function for specific problems.

3.2 Optimize – gradient algorithm

The optimizer in a neural network also is very important, it related to the speed of gradient, also influence the convergence of network. As Dublin[3] stated that gradient descent is an important part of algorithms to optimize neural networks. There are many optimizer algorithms.

i. Stochastic gradient descent:

The original gradient algorithm is Stochastic Gradient Descent (SGD), it is very traditional, however the gradient speed is quite slow, which will impact the result of training in limit time. The process of SGD is train each parameter one by one, and update the parameters w (dx here is the gradient):

$$w = w - learning\_rate * dx \quad (5)$$

ii. Momentum:

Momentum is an algorithm improved from SGD, add a momentum in it to helps accelerate SGD gradient speed:

$$w = w - learning\_rate * dx + b_1 * m \quad (6)$$

iii. AdaGrad:

Another advanced algorithm is AdaGrad, add force of friction in it to make gradient speed faster.

$$v = v + dx^2 \quad (7)$$

$$w = w - learning\_rate * dx/\sqrt{v} \quad (8)$$

iv. RMSProp:

Combine Momentum and AdaGrad, comes with RMSProp, but in this algorithm some part of Momentum missed.

$$v = b_1 * v + (1 - b_1)dx^2 \quad (9)$$

$$w = w - learning\_rate * dx/\sqrt{v} \quad (10)$$

v. Adam

Adam is an improve algorithm from RMSProp, also combined both Momentum and AdaGrad, this method is trying to add the missed part of Momentum in RMSProp:

$$w = w - learning_{rate} * m/\sqrt{v} \quad (11)$$

where Momentum part:

$$m = b_1 * m + (1 - b_1) * dx \quad (12)$$

and AdaGrad part:

$$v = b_2 * v + (1 - b_2)dx^2 \quad (13)$$

Using same simple network model, the activation function chooses Relu function. Next, only changing the optimizer algorithms. The figure 5 and figure 6 show an intuitive comparison of several optimize algorithms.

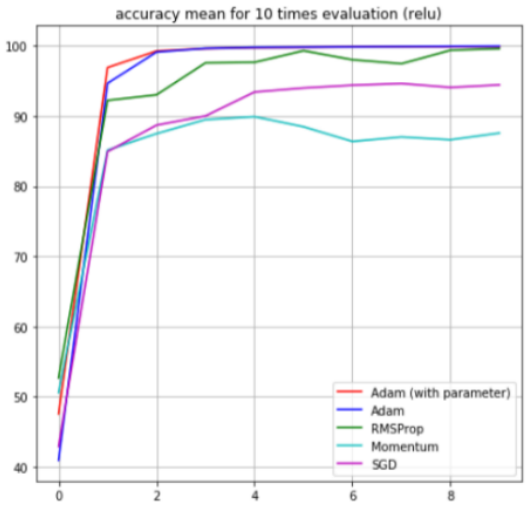

Fig. 5. After 10 times of running test for each gradient algorithm. This graph shows the mean of 10 times evaluation accuracy rate in each gradient algorithm. The accuracy rate becomes to greater when training more data. Only Adam gradient algorithm have a better training result, which could reach nearly 100 percent in training dataset accuracy rate very fast.

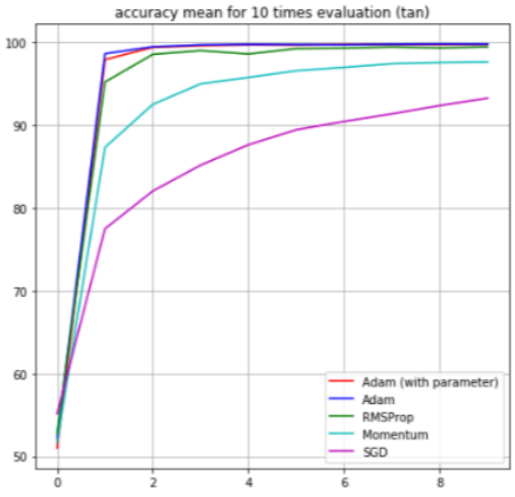

Fig. 6. This graph similar with figure.5. shows the mean of 10 times evaluation accuracy rate in each gradient algorithm, but changed the activation function to tanh. The accuracy rate grows much slower when using SGD and Momentum.

Choosing a better gradient algorithm is equally important, by using a proper activation function and an improved gradient algorithm, the convergence of network will be better. This will have a great reflects on the performance of this network model when training large dataset or pictures.

Above all, in using basic feedforward neural network by using error back propagation algorithm, improving the neural network by using Relu activation function and Adam gradient algorithm. And preprocessing dataset by using polynomial function. All these methods improved the training accuracy of this logistic regression problem.

3.3 Evaluate

All analyses above are related to the Convergence evaluation of network model. These convergence evaluations based on checking then training dataset accuracy rate and loss.

Next, evaluating the normalization ability of this network model. Generalization ability is used to evaluate how this trained model works for fresh samples. A good trained model should with higher normalization ability, which usually used to predict new samples. In this training dataset, separating the dataset in two parts, one part with 3/4 dataset is used for training model, and the other 1/4 dataset is used for testing the normalization ability of model. The method is very simple: predicating the result of test data features and compare with true label values, and calculate the accuracy rate of this predict:

$$accuarcy\_rate = \frac{correct\_predicts}{test\_dataset\_size} \tag{14}$$

The network model is defined now, which is quite good with a perfect convergence ability in training accuracy rate, using this to evaluate the testing accuracy rate which could directly reflect the normalization ability. Here continues the first part, comparing different normalization data method.

Table 1. Shows the data of testing accuracy rate of using different normalization data method

Normalization method	Best of accuracy rate	Mean of accuracy rate
Keep origin	82.67%	81.52%
Scaling to interval	93.31%	92.62%
Dividing by range	92.71%	92.47%
using polynomial	82.13%	79.39%
using exponential	77.71%	73.58%

4 Continue Improving Neural Network Model

4.1 Algorithm

In the process of human cognition, human have the ability to collect effects and get result of consequences, and also, they could use the consequences to find out the effects. Using this characteristic to build the neural network model similarly. So Gedeon[4] proposed an algorithm named "Shared Weights and Bidirectional Networks", which is using this idea.

The neural network model only has one hidden layer, the weight between layers has a symmetry. The first part is Positive Neural Network, a classical feedforward neural network with a hidden layer, use to learn like from effects to find out consequences. Each input vector $x = [x_1, \dots, x_i]^T \in R^{i\times 1}$, where i is the number of input neurons, and the output is also a vector $\hat{y} = [\hat{y}_1, \dots, \hat{y}_o]^T \in R^{o\times 1}$, where o is the number of output neurons. So, the weight matrix in connection between input layer and hidden layer is $W_{p1} \in R^{p\times r}$ where h is the number of hidden neurons. And the weight matrix in connection between hidden layer and output layer is $W_{p2} \in R^{o\times p}$.
The second part is Negative Neural Network, use to learn like from consequences to find out effects. Input vector is $y = [y_1, \dots, y_o]^T \in R^{o\times 1}$, where o is the number of input neurons, output vector $\hat{x} = [\hat{x}_1, \dots, \hat{x}_r]^T \in R^{r\times 1}$ and r is the number of output neurons, and the weight matrix $W_{n1} \in R^{p\times o}$ and $W_{n2} \in R^{r\times p}$, similar with first part above, but symmetry. Next, shared the weight in Positive Neural Network and Negative Neural Network, so we have:

$$W_{p1} = {W_{n2}}^T \in R^{p\times r}, W_{p2} = {W_{n1}}^T \in R^{o\times p} \quad \textbf{(15)}$$

4.2 Analysis the advantage of this network model

i. Convergence:
The Positive Neural Network part is same with feedforward neural network by using error back propagation. So, in the Positive Neural Network part, the convergence is quite good by using improve gradient and use proper activation function. The convergence of Negative Neural Network is a bit faster but very similar with Positive Neural Network.
ii. Generalization:
From the result of using this method, it indeed improves the generalization ability of this network model. This method is effective in preventing over-fitting and enhancing the network generalization ability.

4.3 Compare to traditional machine learning method

Incremental Hill-Climbing Search Applied to Bayesian Network Structure Learning[7]
The 'Mushroom dataset is generally the most difficult to learn incrementally in the sense that incremental algorithms obtain the lowest time gain.' The neural network

could handle some hard problems, while using machine learning mathematic method is hard to train them.

5 Conclusion and future research

4.1 Conclusion

Through this research, I have a deep understanding in building a network model to train dataset and find some methods to improve the training result.
i. proper preprocessing raw data;
ii. choose a proper activation function;
iii. using an improved optimize gradient algorithm to make the network convergence;
iv. using other improved network design algorithm, like Shared Weights and Bidirectional Networks[5] to avoid overfitting problem.

Neural network model is quite useful new technique for training dataset, it could solve wild range of problems, the core task is find the proper neural parameter matrix.

4.2 Future research

As in the data description, it mentioned some research from others, as some data missed in feature 'stalk-root', this feature need to add some more constrain on it. Also, the network model could be improved by adding more layer with some logic relationship between different layer mentioned in data description. Moreover, the learning rate also impact the convergence of network, maybe it will be convergence faster with an auto adjust learning rate method.

Acknowledgments

Thanks to professor T. D. Gedeon and tutor Jo Plested.

References

[1] Mushroom Data Set, https://archive.ics.uci.edu/ml/datasets/mushroom
[2] Krystyna, K., Maciej, Z., Some methods of pre-processing input data for neural networks. Computer Assisted Methods in Engineering and Science, 22: 141–151, 2015.
[3] Aylien, L.D., An overview of gradient descent optimization algorithms, 2017.
[4] Gedeon, T. D., Catalan, J.A., Jin, J., Image Compression using Shared Weights and Bidirectional Networks.
[5] Josep, R.A.,Incremental Hill-Climbing Search Applied to Bayesian Network Structure Learning