International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Rectification of Corrupted Neural Networks

Prakhar Dogra

[1]Delhi Technological University, Department of Computer Science and Engineering, Shahbad, Bawana Road, Delhi – 110042, India

Abstract: ***Imagine if the data set provided for training an artificial neural network turns out to be corrupted. This paper presents a method that can be used to rectify the said neural network after it has been trained but on some corrupted data. In order to rectify the neural network we are provided with a replacement data set for the corrupted data. The proposed method uses the old weights of the corrupted neural network to determine the new weights of the rectified neural network model. Moreover, the proposed method is compared with the present typical method for solving the above stated problem.***

Keywords: Neural network, error, corrupted, Cost function, weights

1. Introduction

Consider a simple example of statistics. Assume that there are 10 numbers in a set that have a mean value of x. But now we are informed that a number in the set was incorrect and should have been something else. Now we need to find a new mean. There is a typical way to find way the new mean. We replace the incorrect number with the correct number. Then we find the mean. But there is a much smarter way to do it. We could use the formula below:

New mean = (Old mean*number of elements in the set – incorrect number + correct replacement number)/ number of elements in the set

Now imagine this problem in an artificial neural network. Consider that we have already trained a neural network. After that, we realize (or come to know of) that some of the outputs in the data were incorrect. Later, we are given the replacement output values and asked to find the new correct weights for the neural network model. For example, consider the MNIST database of handwritten digits that has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST [1]. The digits have been size-normalized and centred in a fixed-size image. Now assume that about 1000 images in the training data turned out to be incorrectly classified (the output values were incorrect or the images were incorrectly labelled). A typical way to solve this problem is to simply replace the incorrectly classified output data with correctly classified output data and then train the neural network from the beginning (re-initialize the parameters) [2].

In this paper, a method is explained where we don't need to apply the above typical method in order to get correct weights for the neural network. A new method is proposed that takes lesser computation to achieve the rectified neural network. The paper is organised as follows. Section 2 explains the typical method that is usually used to solve the above stated problem. Section 3 explains the proposed method. Section 4 gives a brief about the implementation and analysis of the proposed method explianed in Section 3. Section 5 briefs about the applications of proposed method on different types of problems. Finally, section 6 compares the proposed method to the typical method used.

2. Typical Solution

Since we have already trained the artificial neural network before realizing that some of the data is corrupted, the typical method suggests that we need to train the neural network from the beginning (re-initialize the parameters) [2]. Consider an example of the MNIST data set [1] (say an image that displays the digit 7 but its output data shows that the image is a hand-written digit 1). Now this example is corrupted. Assuming that the neural network has already been trained on the data set that has some corrupted data examples (or examples with incorrect output value), we should know that our neural network has been corrupted. It means that the weights of each node of each layer have been corrupted. There is a typical way to solve this problem. After we are provided with the correct data set, we need to re-initialize the parameters (or weights) of the neural network. Then we have to apply forward propagation [3] and back-propagation [4] [5] method in order to train the neural network. In this method, the corrupted data set and the weights are discarded and replaced. This method requires the same amount of computation as required when the data set was corrupted.

The accuracy and time required for this method and the proposed method (discussed in section 3) are compared in Section 6.

3. The Proposed Method

The proposed method can be explained by taking examples from MNIST data set [1]. We have to assume that some of the examples in the data set have incorrect output values like an image of a digit 7 has output value of 1. In the previous section, we have explained that the method requires the cost function values of the corrupted neural network in order to reverse the effects of the incorrect examples with the new correct examples.

The proposed method can be explained in the following steps:

1. Storing the corrupted data

The corrupted data is the data set containing some examples that have incorrect output values (some images are

incorrectly labelled). The weights created while training the neural network with the corrupted data set are also stored so they can be used during the rectification process. Corrupted data set is compared with the new data set in order to check which examples have incorrect output values. Then these corrupt data set examples are stored along with their respective correct output values and their respective indices from the original data set (corrupt) in a separate file. This creates an incorrect-correct data set (i.e., the output values). This data set is then used further in the proposed method to reverse the effects on the corrupted neural network.

2. Modified Back Propagation

For the proposed method we have modified the back propagation algorithm. In the modified back propagation algorithm the error term for the first iteration is calculated as the difference in the incorrect and correct output values of the new (not corrupt) and courrpted dataset respectively. Then we follow the usual back propagation algorithm to update the weights. We use forward propagation to find the hypothesis term and subtract it from the output value (given) to find the error term δ. Then using the traditional back propagation algorithm we calculate the error terms of the previous layers. Then we calculate the numerical gradients of the respective layers using the respective error terms (denoted by δ). Following that we update the weights of the neural network using the respective numerical gradients.

In short we can say that we backpropagate only once with the error term as the difference in the incorrect and correct output values.[6] Then we follow the usual back propagation algorithm to update the values. Usually the data set (incorrect-correct output data set) is much smaller than the original dat set it takes much lesser time to rectify the data set.[7] This can be seen from the results presented in the next section.

In the proposed method, we use the saved incorrect-correct data set to create an array of respective error terms. For example, the hypothesis predicts the image as digit 7 and the incorrect ouput value is 9 whereas the correct output value is 1. The error term of this example will be a vector $\delta = [-1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ -1\ 0]^T$. In the vector above, the values are assumed to be either 0 or 1 or -1. But in actual implementation, these values are usually floating-point values varying between -1 and 1 and are usually very near to either -1, 0 or 1. After that, we can run a series of back-propagation and forward propagations (as explained above) to update the old weights.[8] We would just need the last updated weights of the neural network. Here the number of iterations required need not be equal to when originally training the neural network. Since the incorrect-correct example pair data set is smaller than the original data set, we can reduce the number of iterations because the smaller data set will require less amount of tuning and hence less number of iterations.[9]

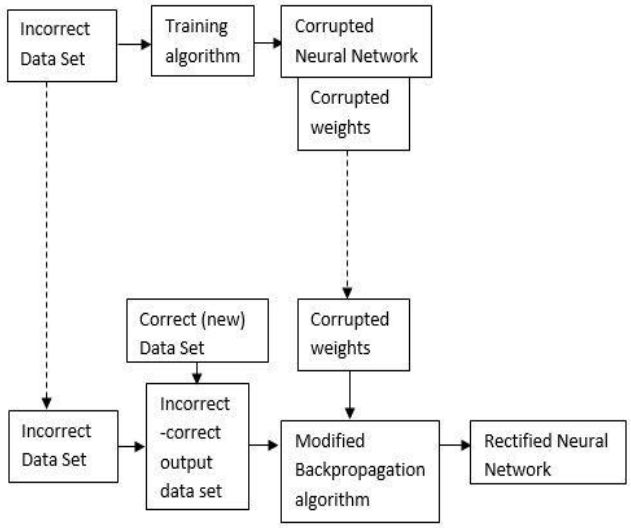

Figure 1: Proposed method

4. Implementation and Analysis

The proposed method was tested on several data sets. All the data sets were corrupted by replacing original values with random (output values within the range of the original set) values at random positions in the data sets. One of the results of our implementation have been shown below. These are the results are of the popular MNIST data set for Hand-written digit recognition problem. There were 5000 examples in the original data set and out those examples, 200 random examples were corrupted (incorrect output values). As you can see from the table below, the percentage accuracy dropped by nearly 5% when 4% examples were corrupted. It means that the weights in the neural network got corrupted. After rectification process the accuracy wasn't exactly restored when the neural network was trained on the original (no incorrect outout values) data set but the accuracy increased by nearly 4% when the corrupted neural network (corrupted weights) was tested.

Table 1: Percentage accuracies in different cases

Rectification / *Data set*	*Before Rectification*	*After Rectification*
Corrupted Dataset	90.56%	94.31%
New Data set (original)	95.7%	____

As the number of corrupted examples were increased, the accuracy of the corrupted neural network also decreased. Moreover after rectification the accuracy also increased from the corrupted neural network case but not in a linear fashion as expected. The reason might be due to random examples being chosen everytime the original output values were replaced with random values in order to make corrupted data set.

Moreover, the relation between the time elapsed for the neural network to rectify and the number of examples being corrupted wasn't linear either. This might also be due to the random examples being chosen everytime the original output values were replaced with random values in order to make corrupted data set.

5. Applications

The proposed method is a solution to the problem when an artificial neural network is trained over a data set that consists of some examples that have incorrect output values. This improved backpropagation algorithm [10][11] can be applied to similar types of problems. For example, we have a data set that consists of a million images of very small number of people (say 20). And after training the network we realise that some of those images were incorrectly classified due to incorrect output values. Later we are provided with a correct data set. Rather than re-training the neural network from the beginning (re-initialising the parameters), we can simply apply the above proposed method to train the neural network. This method can be applied to almost any neural network that faces the above stated problem. Neural network can be for object images, hand-written images, data sets that require complex neural networks to learn, etc.

The proposed method can be modified and used in Software Re-Engineering and Reverse Enginering problems. Where, if we need to change the testing data, we might not actually be required to compute all the values from the scratch typically and use the new method similar to the above discussed improved back propagation [10][11] to get the new values. For example, lets take the concept of black box testing. We don't know what code lies inside the software but what we want is to test out a lot of values to see if any error comes out or not. Now assume that an error does come. What can we do in such a situation. We can use the basic approach of the above proposed method and modify the software without going too deep into the code of the software.

6. Comparison between Typical and Proposed Method

As explained in Section 2, the typical method involves the replacement of incorrect examples with correct examples and then trains the neural network from the beginning. And as explained in previous sections, the proposed method uses stored values of old neural network weights to rectify it using the new correct examples. As we can see from the explanations provided in above sections, the typical method uses slightly less memory usage because it doesn't prefer to store the old weights of the neural network (corrupted). But the proposed method requires less computation. Firstly, it doesn't calculate the cost function values of all the elements. It simply calculates cost function of the incorrect examples and the replacement correct examples. Moreover, while training the neural network, only the incorrect-correct example pairs are trained rather the complete data set. Suppose that we have a MNIST data set that consists of 60000 images for training. Out of these we have 1000 examples with incorrect output values. We can simply observe that the size of the array used for training the neural network (especially when using a vectorized implementation approach [5]). Moreover, there wasn't any relation found between the number of examples being corrupted and the percentage increase in the accuracy after rectification, probably due to the randomness of the selection process.

References

[1] LeCun, Yann; Corinna Cortes; Christopher J.C. Burges. "MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges". Retrieved 17 August 2013.
[2] Handwritten Digit Recognition by Neural Networks with Single-Layer Training. S. KNERR, L. PERSONNAZ, G. DREYFUS, Senior Member, IEEE. IEEE TRANSACTIONS ON NEURAL NETWORKS, vol. 3, 962 (1992).
[3] Introduction to multi-layer feed-forward neural networks. Daniel Svozil, Vladimir KvasniEka, JiE Pospichal. Chemometrics and Intelligent Laboratory Systems 39 (1997) 43-62.
[4] Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (8 October 1986). "Learning representations by back-propagating errors". Nature 323 (6088): 533–536. doi:10.1038/323533a0
[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Leaning Internal Representaions by Ewvr Propagation in Rumelhart, D. E. and McClelland, J. L., Pamllel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press, Cambridge Massachusette, 1986.
[6] A. Olaru, S. Olaru, L. Ciupitu, "Research of the Neural Network by Back Propagation Algorithm", Advanced Materials Research, Vols. 463-464, pp. 1151-1154, Feb. 2012
[7] Kuldip Vora and Shruti Yagnik and Mtech Scholar, "A Survey on Backpropagation Algorithms for Feedforward Neural Networks", International Journal of Engineering Development and Research, ISSN: 2321-9939
[8] Md Nasir Sulaiman and Maslina Darus. "An improved error signal for the backpropagation model for classification problems."International Journal of Computer Mathematics. 01/2001; 76(3):297-305. DOI: 10.1080/00207160108805026
[9] Tasos Falas and A-G Stafylopatis, "The impact of the error function selection in neural network-based classifiers", IJCNN'99. International Joint Conference on Neural Networks, 1999. Volume 3 Pges 1799-1804.
[10] M C Bossan, "A modified backpropagation algorithm for neural classifiers." Proceedings of the 38th Midwest Symposium on Circuits and Systems (1995).
[11] J Lv, Z Yi, "An Improved Backpropagation Algorithm Using Absolute Error Function (2005)", ISNN 2005, LNCS 3496.

Author Profile

Prakhar Dogra is an undergraduate student (B.Tech) in Computer Engineering at Delhi Technological University (formerly as Delhi College of Engineering). He has till date published 2 other research papers both in the field of Compter Science and Engineering.