MUTING HIGHER ERROR PRONE HIDDEN UNIT USING BADNESS TECHNIQUE

Sarveshwaran Rajarajan
Masters of Computing,
Australian National University

Abstract. Neural networks(NNs) has become a trending field of study with rapid technological growth and development in the field of Machine Learning and Artificial Intelligence. Neural network has gained world over popularity from its beginning with the introduction of Feed Forward NNs. Many researches were conducted with the help of Feed Forward NNs to solve challenging real world problems having only few neurons in the middle layer, so called the "hidden layer". When the results were analysed, it was found that having more hidden units yielded better results in terms of accuracy but increased overall execution time with its dense structure. To overcome this problem, researchers came up with the idea of multiple hidden layers network called the Deep Neural Network (DNNs) which distributed the neurons among various hidden layers. But the problem of increased execution time persisted and also hindered the performance of learning in Neural Network. It was later observed by experiments that not all neurons in the hidden layer between the input and output layers were actually contributing to the final outcome of the learning. In such a case, those particular hidden units in the hidden layer can be found and muted that would help neural network to have a performance boost during the network learning. Finding and muting neurons can be employed by using various strategies but this paper is concentrated on technique called "Badness" i.e., muting the neuron which has the maximum back propagated error. Incorporating badness technique into the Deep Neural Network implementation yielded more performance and speed retaining the same solution compared to those without it. Accuracy of the network before and after incorporating badness technique remained approximately same around 97% when testing across unknown test samples with loss climbing down to 0.01 from being 2 during the start of network learning. The main aim of this paper is to boost the performance of the neural network without compromising solution accurateness.

Keywords: Deep Neural Networks, Backpropagation, Hidden unit, Cross Entropy Loss, Stochastic Gradient Descent, Cross validation

1 Introduction

According to (Gedeon and Harris), Deep Neural Network is considered to implement badness technique employing five layer network structure. It consists of one input layer, three middle hidden layers and one output layer. Succeeding from input layer, each layer is connected in sequence to other layers until output layer with each having weighted units connected to all other weighted units in hidden layer. Important reason for this type of connection is to extract the features from the input data, assign them with weights and propagate it equally among the connected units in the network. The final output layer classifies the input data into classes specified in the dataset. Here the layers are defined as the linear tensors with its shape depending on the number of features for the input layer, number of hidden neurons for the hidden layer and number of classes to be classified for the output layer. Added, sigmoid function is used as the activation function for weights propagating from hidden layer to output layer. Training the network using back propagation technique makes the NNs to train slower because as the network proceeds with execution, output error gradient increases and weight calculation becomes complex for certain units in the hidden layer ultimately resulting in lower performance. Output error gradient plays an important role in optimization for deciding the performance of the network by manipulating the weights getting transferred from one layer to other. Having the neural network configured, the next important thing required is the dataset for which this classification task is to be performed. The chosen classification system involves in the dataset which is used to identify the type of erythemato-squamous diseases(Archive.ics.uci.edu, n.d.). The main motivation to choose the dataset is to explore the medicinal field usage of NNs where the output layer classifies the dataset as either psoriasis, lichen planus, cronic dermatitis, pityriasis rubra pilaris, pityriasis rosea and seboreic dermatitis(Archive.ics.uci.edu, n.d.). Diagnosing this disease was initially meant to occur using biopsy but it was found that it needs histopathological features too which was then determined through the usage of microscope. Twelve clinical features and twenty two histopathological features are considered for constructing this dataset. This feature set contains family history column of data with values ranging from 1 (if history of disease is observed) and 0 otherwise, age representing the age of the patients and other columns of data with values from 0 to 3 where 0 represents the presence of no trace of disease and 3 represents the highest trace.

2 Dataset Configuration

According to Archive.ics.uci.edu, the dataset contains total instances of three hundred and sixty six which is randomly split into train set and test set with each contributing 85% and 15%.The technique of cross validation is employed to split them. This dataset also contains 34 features overall that are used to classify them among one of the six classes

3 Methods in Experiment

To implement the "Badness" technique, Deep Neural Network is employed with five layer network structure. Input data is extracted from csv file and are then wrapped up in the tensor to feed into the input layer of the Neural Network. Here, I have considered,

Layer Name	**Data Size**
Input	Number of features x Number of instances in dataset
Hidden first	Number of instances in the input dataset times x Number of hidden unit connections in hidden layer.
Hidden Second	(Number of hidden unit connections in hidden layer) x (Number of hidden unit connections in hidden layer / 2)
Hidden Third	(Number of hidden unit connections in hidden layer / 2) x (Number of hidden unit connections in hidden layer / 4)
Output	(Number of hidden unit connections in hidden layer / 4) x (Number of output classes + 1)

The main reason for choosing three hidden neurons in the network structure is because to gradually reduce the network connections from the first hidden layer network to the output layer. Various hyper parameters like learning rate as 0.1 and number of epochs as 2000 is all chosen because including all these parameters, loss is minimized at the end of training considerably, say for example in the beginning it was around 2 and reduces down to 0.01 at the end and accuracy is also greatly improved (more than 95%) compared to its beginning step of 20%. Various experiments were done before concluding with this learning rate and number of epochs like initially learning rate was fixed with 0.001 and number of epochs were around 200 only. When running with these parameters, it was found that network learning was slower. Then it was decreased to 0.1, network learning was made faster but accuracy remained lower around 86%. Applying brute force approach of increasing the number of epochs gradually, increased the accuracy with loss incurring very minimal value. Carrying out brute force experiments, it was found that 2000 epochs yielded results around 98%. Hence the learning rate and number of epochs was fixed. Output layer consists of seven classes with zeroth class being assigned with no classification results and class from one to six with classified results. Here, the network is initialized with five layers with each propagating weights to the next layer till it reaches the maximum accuracy by minimizing the loss incurred. Once the network is declared with tensors and initialized with weights obtained from the dataset, the forward function(net) is called. This forward function computes the weight from input data in the input layer and sends it across all the connection units in the first hidden layer. Since the first hidden layer is dense compared to the other hidden layers in the network, there is more chance of high mathematical computations happening. It was found that tweaking the hidden neuron in the first hidden layer can create a drastic change in the network learning as this provides the starting point for network learning. Many researches have been conducted in deciding the number of hidden neurons to be present in the hidden layers which corresponds directly to either input layer or its previous hidden layer or output layer. According to Sheela K. and S.N, 2013, network stability depends mostly on the error gradient propagated from the hidden to output layer. If the error gradient is more for a particular hidden unit, then there is more chance that it will deviate the network from obtaining optimal result. Also, if the network has more neurons, the network size increases leading to much higher arithmetic operations leading to increase in computational cost often leading to overfitting of training data(Augasta M. and T, 2018).

3.1 Muting Neurons

To overcome the overfitting of input data, most error prone gradient unit has to identified and muted in such a way that the highest error values are not transmitted in the network ensuring the reliability in the final classification results. Finding the most error prone hidden unit is implemented in the forward function according to, "*Sum of cumulative back propagated error = summing of weights in the given row of elements in the hidden layer*". Muting the hidden neuron is implemented in the first hidden layer such that highest error prone unit is cut off at the beginning of weight propagation itself. Once the hidden neuron with maximum error is found, then its corresponding weighted values are assigned with value zero(muting). Once the corresponding row elements of the found hidden neuron is assigned with zero values, all the weighted connection from that neuron to the succeeding ones carry only zero value. Since the weighted connection is the multiplication of weights of various neuron values, its contribution becomes null helping the network to push towards the optimal solution than leading to local minima or maxima. This method creates tolerance in the network such that the contribution of other hidden units in the hidden layer is maximized. To calculate the error gradient, CrossEntropyLoss() is used. When the predicted probability of input labels are diverged from obtained label from training, cross entropy loss increases and vice versa(Ml-cheatsheet.readthedocs.io, n.d.). Once the loss is calculated, the network is instantiated with zero gradients and the error is propagated back to the hidden layer and the gradient weights are accumulated. Finally, Stochastic Gradient Descent optimiser(SGD) is called to minimize the objective function leading to faster convergence. This helps to reduce the high cost incurred by the back propagation in the neural network. Finally it uses the Softmax function to classify the dataset based on the classes desired by assigning probability values to every classes (Ufldl.stanford.edu, n.d.). For example, if a test image is to be classified as Dog or Cat, the neural network outputs the probability of test image for both Dog and Cat. This probability value is compared with the threshold and test image is classified accordingly.

4 Results and Discussions

Implementing the Deep Neural Networks without muting the higher error gradient hidden unit and those with muted gradient unit produced similar results in accuracy with negligible difference of +/- 1 to 2. Similar accuracy is the first evidence that the Badness technique doesn't alter the meaning of doing this experiment. If the accuracy was changed, then it could have given a hint that not only that more error prone hidden unit was getting muted, but also some contributing hidden neurons. The main motivation for this paper is to retain the accuracy at the highest value possible and at the same time reducing the computational costs in the network. Implementing the Badness technique in Deep Neural Network resulted in performance boost of 1.2X (means Normal execution was 40 seconds but with badness implementation, it resulted around 37 seconds) as high cost arithmetic operations caused by most error prone gradient units are muted. This performance boost became more evident as the number of hidden neuron increased and as the number of epochs increased leading to number of hidden neuron count in first hidden layer. Since this layer was dense, it was reduced to half in second hidden layer and by quarter in the third hidden layer. Muting the selected hidden units greatly helped the neural network to converge faster towards the local minima that guaranteed to reach global minima at the end of training. Convergence of weights may get stuck in local minima or becomes harder to even reach local minima if there are more cumulative error gradient unit in the network. Added, implementing badness also ensures that the data doesn't overfit into the training set as more the hidden neurons present in the network, there is more chance of network getting overfit with training samples. Finally, Cross validation ensures that the data considered for training and testing is taken in random order to maintain the generality of the network training. Comparing the implementation of Badness on Dermatology dataset with research paper by (Pappa, Freitas and Kaestner, n.d.), we can find out that the overall loss value is marginally lower. Loss obtained without muting and with muting the higher error gradient graph is displayed below in Figure 1 and Figure 2.

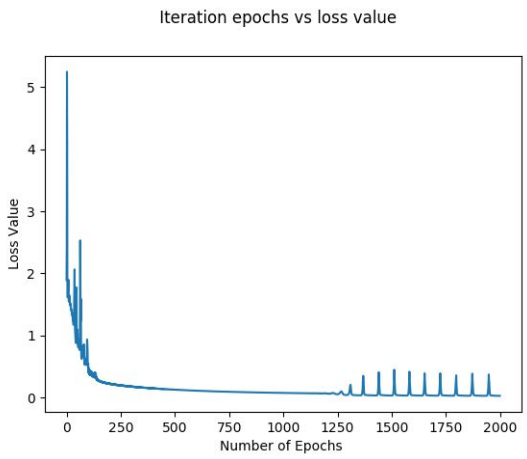

Figure 1 : Without muting the hidden neuron

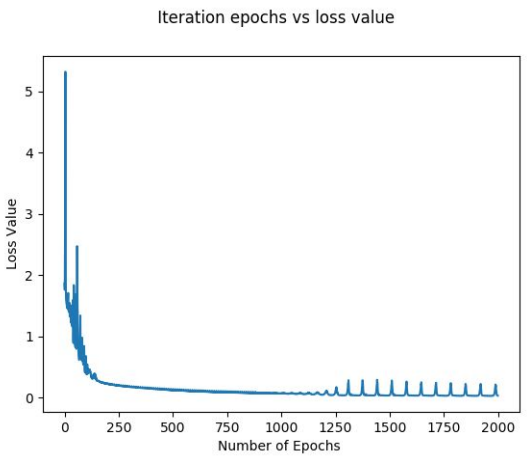

Figure 2: Muting the hidden neurons

4.1 Graph

Here the graph is constructed using the Figure() in Python Imaging library. Here the X-axis corresponds to the Number of epochs the network took and Y-axis corresponds to the Loss incurred the network during the feature learning. The spikes in the graph represents that earlier the number of iterations, higher the loss value. When the experiment started at zeroth epoch, the loss value is more and it gradually decreases as the network starts to learn the feature. The spikes during the end of the network learning also symbolizes the fluctuation of loss value calculation due to the removal of hidden neurons having the highest error prone gradient.

5 Conclusion and Future Work

This work proposed the Badness algorithm for optimizing the Deep Neural Network having bigger network size. The error gradient calculation and loss graphs (Figure 1 and Figure 2) shows almost no difference between them indicating that solutions is not compromised after training. The main aim of this paper is to achieve the performance boost during training which is summarized to be 1.2X than the basic Deep Neural Network. without Badness implemented. This paper illustrates that muting the higher error gradient in the neural network once in every fifty epochs actually resulted in good performance boost. The reason behind doing the muting once every fifty epochs is because that the network would have learned considerable amount of features contributing not only to the final result but as well as to the error. Muting neurons once every fifty epochs ensures that the learning curve diverges to minimal loss. Currently, I am evaluating this technique by conducting series of experiments with different dataset to illustrate the performance boost of network training and also in retaining the muted neuron to ignore the gradient passed on from its back propagation. Overall, the Badness technique doesn't affect the end accuracy as this technique is not about changing the final classification results but it helps in diverging to the optimal solution faster. In this paper, highest error gradient hidden unit is muted such that it stops from further contributing to the actual classification results whereas in future it can be expanded to removing the hidden neuron from the network structure. This also ensures that the gradient of that neuron will have no effect on the classification results.

6 References

Archive.ics.uci.edu. (n.d.). UCI Machine Learning Repository: Dermatology DataSet. [online] Available at: http://archive.ics.uci.edu/ml/datasets/Dermatology [Accessed 24 Apr. 2018].

Gedeon, T. and Harris, D. (n.d.). Network Reduction Techniques. [ebook] pp.1-4. Available at: https://wattlecourses.anu.edu.au/pluginfile.php/1678707/mod_folder/content/0/NetRednTech%20paper.pdf [Accessed 24 Apr. 2018].

Sheela K., G. and S.N, D. (2013). Review on Methods to Fix Number of Hidden Neurons in Neural Networks. [ebook] Hindawi, pp.1-6. Available at: http://downloads.hindawi.com/journals/mpe/2013/425740.pdf [Accessed 24 Apr. 2018].

Augasta M., G. and T, K. (2013). Pruning algorithms of neural networks - a comparative study. [ebook] Versita, pp.1-3. Available at: https://www.degruyter.com/downloadpdf/j/comp.2013.3.issue-3/s13537-013-0109-x/s13537-013-0109-x.pdf [Accessed 24 Apr. 2018].

Ml-cheatsheet.readthedocs.io. (n.d.). Loss Functions — ML Cheatsheet documentation. [online] Available at: http://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html [Accessed 24 Apr. 2018].

Ufldl.stanford.edu. (n.d.). Unsupervised Feature Learning and Deep Learning Tutorial. [online] Available at:http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/ [Accessed 24 Apr. 2018].

Pappa, G., Freitas, A. and Kaestner, C. (n.d.). A Multiobjective Genetic Algorithm for Attribute Selection. [ebook] pp.4-6. Available at:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80.2880&rep=rep1&type=pdf [Accessed 25 Apr. 2018].