Assessment of Accuracy Enhancement of
Back Propagation Algorithm by Training
the Model using Deep Learning

BABY KAHKESHAN and SYED IMTIYAZ HASSAN

Department of Computer Science & Engineering, Jamia Hamdard, New Delhi, India.
Corresponding author E-mail: s.imtiyaz@gmail.com

http://dx.doi.org/10.13005/ojcst/10.02.07

(Received: April 05, 2017; Accepted: May 11, 2017)

ABSTRACT

Deep learning is a branch of machine learning which is recently gaining a lot of attention
due to its efficiency in solving a number of AI problems. The aim of this research is to assess the
accuracy enhancement by using deep learning in back propagation algorithm. For this purpose, two
techniques has been used. In the first technique, simple back propagation algorithm is used and
the designed model is tested for accuracy. In the second technique, the model is first trained using
deep learning via deep belief nets to make it learn and improve its parameters values and then back
propagation is used over it. The advantage of softmax function is used in both the methods. Both
the methods have been tested over images of handwritten digits and accuracy is then calculated. It
has been observed that there is a significant increase in the accuracy of the model if we apply deep
learning for training purpose.

Keywords: Machine Learning, Deep Learning, Deep Belief Nets, Back Propagation,
Restricted Boltzmann Machines, Artificial Neural Networks, Softmax function.

INTRODUCTION

Deep learning is a branch of the rapidly
growing field of machine learning. Machine learning
is a field of computer science which makes use of
AI concepts, to enable computers to learn from
problem solutions and make use of the knowledge
for new problems. Machine learning have proven
useful in a mining structured, semi-structured and
unstructured data. The values generated by this
process may be applied in various fields as reported
in1,2,3,4. Handwritten digit recognition is the capability
of an automated system to recognize digits written
by various human hands and has its applications
in bank checks details recognition, zip codes
recognition and others. This research proposes
a solution for the same using deep learning and
back propagation algorithms. An artificial neural
network has been built for this purpose and then tested with the required algorithms. For the purpose of validation, MNIST data set of handwritten digits has been used.

This paper explores the applicability of deep learning in handwritten digit recognition system. The proposed solution in this research makes use of deep belief nets.

**Background**

A brief overview of necessary concepts used is presented in this section.

**Artificial Neural Network**

Artificial Neural Network (ANN) is a network based on the concept of biological neurons. It is composed of layers, with each layer consisting of nodes called neurons. First layer is the input layer, layers next are called hidden layer and outermost is the output layer. Each node receives an input signal, does some processing and provides the output using weights assigned to the edges.

**Back Propagation**

Back Propagation algorithm is an algorithm which is used to train artificial neural network and aids in classification. This is a supervised form of learning, in which training examples with known class labels are first provided to the neural network. Output predicted by network and the known output of the training example are compared and differences are back propagated through the network. Then the learned network can be used to predict output for any unseen input.

**Deep Learning**

The main concept behind deep learning is that data can be represented in many ways. Each layer in a deep learning neural network represent items at a greater level of abstraction. Higher abstraction layers learn from the features from the lower abstraction layers. For example, if a network has to learn digit ‘8’, firstly the network can learn that there are two similar figures stacked upon one another, then it can learn that those figures resemble a circle. Various terms evolve in the study of deep learning such as deep belief nets, deep convolutional nets etc. Deep belief nets are networks consisting of Restricted Boltzmann Machines (RBMs) discovered by Geoff Hinton [6]. RBMs represent the abstractions involved in deep belief nets.

**Restricted Boltzmann Machines**

In simplest terms, RBM has two layers of neurons. First layer is called the visible layer and next layer as the hidden layer. Nodes in the visible and hidden layer are connected with each other. So, nodes in the visible layer can send signals to the nodes in the hidden layer but cannot send signals amongst the nodes of the same layer. That’s why they are called restricted. There is also a reconstruction phase involved in RBMs where nodes in the hidden layer send signals to the visible layer. Visible layer units correspond to the input observations received such that for an image composed of pixels, it may represent each pixel unit.

Hidden units represent the dependencies between the observations. Figure 1 shows a very simplified version of RBM with 3 visible nodes and 4 hidden units and its processing. Each unit in the
visible layer receives input x. It multiplies the same with the corresponding weights between visible and hidden layer. At the hidden layer, the weighted sum is added to the bias and these are then provided as inputs to an activation function and finally provide an output ‘a’. This is the feed forward phase.

There is also reconstruction phase in the RBM in which outputs from the hidden layers are passed to the visible layer with the same set of weights as done in feed forward phase. The weighted sums through the hidden layer are added to the visible layer biases and the outputs produced are compared with the original inputs and the difference is recorded as error and then this is used for bias and weight adjustment similar to that done in back propagation.

**Literature Review**

This section deals with the field of latest work done in this field. Deep learning has recently gained much attention and can be used for a number of problems as shown in where it has been used for a simple problem such as movie recommendation. It has been shown that deep belief nets can be made to learn one layer at a time. Handwritten digit recognition finds its application in postal system, optical character recognition and can be developed using a neural network. In, RBMs which are basically Restricted BMs (Boltzmann Machines) have been used as a building block of deep belief nets. RBMs are generally trained using contrastive divergence learning procedure which needs some practical experience for deciding parameters like learning rate, momentum, and the values of weights. RBMs’ learning involves two passes. Forward pass shows the probability of output given the input. But, backward pass shows the probability of same inputs with the resultant outputs. Classification, a supervised learning technique is a two-step process in which the first step involves construction of a model after some training set with known class labels are shown to it and the second step involves predicting the class label of a new sample. Artificial neural network can be trained using back propagation by steps as shown in. Softmax function has been used at the end of linear regression to determine the probability of an output to fall in each of the applicable classes. Deep belief nets with stacked RBMs have been used for many applications such as speech and phone recognition. Later distributed techniques like Hadoop Map Reduce have been used for taking the advantages of deep learning over very large data sets similar to the one done where also map reduce algorithm has been used for sentiment analysis.

**proposed methodology and pseudocode**

In the proposed method, an ANN has been implemented. First it is trained using back propagation. In the second case, deep belief nets have been used in which the network is first trained using deep learning and then refined using back propagation. MNIST database of handwritten digits has been used for testing the network. MNIST database is a dataset available online which consists of 60,000 training examples to help train the model and 10,000 test examples from different areas which can be tested for checking accuracy. Each handwritten digit image is composed of 28*28 = 784 pixels. Fig 2 shows a very simple representation of pixel values for a sample digit ‘5’. (All pixels are not represented due to space constraints).

**Pseudo code for Back Propagation**

1. Create an ANN with n number of layers.
2. Initialize weights for all edges and biases for all nodes with small random numbers.
3. While terminating condition is not satisfied
DO
i. For each item in training data set do
A. call ‘feedforward’ function passing the training
data item to forward the inputs through all the
layers
B. compare the outputs of the outermost
layer computed post feed forward with the
expected outputs and calculate error. Back
propagate the errors up to the input layer by
calling ‘backpropagate’ function
C. Update the weights of each edge by calling
the ‘updateWeights’ function
D. Update the bias of each node by calling the
‘updateBias’ function
ii. End For
4. End While

feedforward function
Inputs: item to be feed forwarded
Outputs: forwards the input calculating the inputs
and outputs at each layer

Pseudo code
1. Assign the inputs of the first layer from the
item input. For MNIST handwritten digit
recognition there are 784 pixels with values
ranging from 0 to 255. Normalize each pixel
value by dividing the same by 255. Assign
the normalized pixel values to each of the
784 input nodes.
2. Make the outputs of the input layer same as
inputs. That is for each input layer node j set
Oj = Ij, where Oj , Ij are output and input of
jth no respectively.
3. For each hidden layer unit j with a previous
layer i, where wij is the weight of edges from
a node in layer i to a node in layer j and biasj
is the bias value for node j.
Assign lj = SiwijOi+ biasj [11]
Set Oj = 1 / (1 + exp(-lj)) [11]
4. For output layer unit j with a previous layer i call
‘softmax’ function.
5. End

Softmax function [15, 16]
Output: Calculate ate outputs for the
output layer in terms of probability distribution. For
a particular input digit to the network, outputs show
how much is the probability of that digit is digit 0,
how much is 1 and so on

Algorithm
1. Initialize a variable sum = 0
2. for each node j of the output layer i.
   sum = sum + exp(-lj) , where exp is the
   exponential function
3. End for
4. for each node j of the output layer
   i. Set Oj = exp(lj)/ sum
5. End

backpropagate function
Input: actual output of the training sample T j
Algorithm
1. For each unit j in the output layer calculate the
   error Errj as
   i. Set Errj= Tj – Oj  [11]
2. End for
3. for each unit j in the hidden layer from the last
   to the first hidden layer
   i. Set Errj = (1- Oj) * (1+Oj) * S k     Errjwjk  [11]  
   - here k is the next highest layer
4. End for

updateweights function
Algorithm
1. For each weight wij in the neural network
   i. set ?wij = (learningRate) Errj * Oj
   ii. updatewij as :
      wij= wij+ ?wij [7]
2. End for

updatebias function
Algorithm
1. for each bias biasj in the neural network
   i. set biasj = (learningRate) Errj
   ii. Update biasj as:
      biasj = biasj + ? biasj 2. End for

Pseudo Code for Deep Learning with RBMs

The basic structure for this also remains
same as back propagation above except that in
the Pre-training step each two consecutive layer
starting from the first layer are processed by RBM
processing

Algorithm
1. For each pair of consecutive layers in the Network
do
   i. call ‘doRBMProcessing’ function

Assign lj = SiwijOi+ biasj [11]
Set Oj = 1 / (1 + exp(-lj)) [11]
2. End for.
doRBMProcessing function

Inputs: Layer X and Layer Y of the neural network to be performed by RBM Processing. It is assumed that initializations of weights of weights, biases and input layer have already been done as above.

Algorithm
1. While termination criteria is not satisfied do
2. For each unit j in the Layer Y
   i. Assign $I_j = S_{i, w_{j,i}}I_i + \text{bias}_j$ /* where i is a node of LayerX */
   ii. Set $O_j = 1 / (1 + \exp(-I_j))$
3. End For
4. For each unit j in the Layer X
   i. Assign $I_j = S_{i, w_{j,i}}O_i + \text{bias}_j$ /* where i is a node of LayerY */
   ii. Set $O_j = 1 / (1 + \exp(-I_j))$
5. End For
6. Compare Layer X inputs with the actual inputs and calculate error with respect to the original inputs, then back propagate the error and update weights and biases using the same logic as described in the case of normal back propagation.
7. End While

RESULTS AND DISCUSSIONS

Each training digit from the training data sample is passed through the layer in the learning phase. Then each digit from the test data sample is tested. Network is made to learn for different number of epochs. One epoch implies that the network is tested with the entire dataset once. Then each digit from the test data set is tested and accuracy is calculated with following formula:

$$\text{Accuracy} = \frac{\text{Number of correctly classified instances} \times 100}{\text{Total Number of Instances}}$$

Table 1 and Table 2 show the observations for both the cases. Case I is where the network is first made to learn by simple back propagation algorithm and then tested. Deep learning has not been used in Case I. Case II is where deep learning has been used in the learning phase and then the network has been tested.

Fig 3 shows the graphical representation depicting the relation between the above two cases and depicts that accuracy increases with the number of epochs and is greater for case II then case I. On the negative side case II takes more time in learning than case I. So that can be made better by using parallel technologies such as Map Reduce.

CONCLUSION AND FUTURE SCOPE

An attempt has been made in this research to design and implement a basic deep belief network with a focus on improved accuracy. For this purpose the network has been successfully tested with MNIST images of handwritten digits. It has been concluded from the results that deep learning fine-tuned with back propagation shows a greater accuracy than when only back propagation is used alone. When the number of epochs are 1 and 2, an accuracy of about 2.785% is achieved with deep learning. For number of epochs as 3 and 4, this has been reduced to 0.34% which can be attributed
to the fact to greater number of epochs and long training time involved in deep learning for training RBMs'. This can be extended in future to implement deep learning with distributed technologies such as Map Reduce to reduce the time involved. The solution implemented in this research is generic and can be tested for other such tasks also.

REFERENCES