Asian Journal of Information Technology 16 (2-5): 268-273, 2017
ISSN: 1682-3915
© Medwell Journals, 2017

MNIST Classification using Deep Learning

Majid Hameed Khalaf, Belal Al-Khateeb and Rabah Nory Farhan
College of Computer, University of Anbar, Ramadi, Iraq

Abstract: Lately, deep learning has seen enormous using in computer vision and classification applications. In this study, an implementation of deep architecture is done in order to compare two classification architectures, those are usual neural networks that contain unique hidden layer with the notion of deep learning as a "Deep Belief Networks" (DBN) that represented by many layers. Both architectures are implemented on the images of MNIST digit dataset for the classification purpose. The usual network of digit recognition was trained as a supervised learning using backpropagation algorithms while DBN was trained using two stages, one as unsupervised learning and the other as a supervised learning. In the unsupervised learning, we used the contrastive divergence algorithm and with the supervised used back propagation as a fine tuning networks. The features are extracted as pixels from the image represented that digit to train the networks that depended on the intensity of pixel in image that white color represented as a 0's and black color represented as a 1's. DBN is performs as many layers each layer represent as a Restricted Boltzmann Machines (RBM) as a stack that will represent in sequence. The learning of DBNs consisting of two steps, a pre-training step and a fine-tune step. DBNs gave a higher performance as compared with the usual neural networks with an accuracy of approximately 98.58% for classification of handwrite digit of MNIST dataset.

Key words: Restricted Boltzmann Machine, deep belief networks, contrastive divergence, backpropagation neural networks, MNIST

INTRODUCTION

Machine learning is widely used to solve functional problems by learning from a given input. The center quality of machine learning is to keep a good representation of data and create a generalized algorithm to predict inconspicuous information appropriately (Ethem, 2014). The fields of artificial neural network grew from pursuing of understanding the brain and its learning process. This started with approximations of a single neuron and progressed to deep neural networks that are winning modern machine learning competitions. Machine learning is a comprehensive field of researches, especially pertaining to research conducted by Hinton (2010). Deep Learning (DL) is a branch of machine learning in light of an arrangement of calculations that endeavor to model abnormal state reflections in information by utilizing model designs with complex structures or something else, made out of multiple non-linear transformations (Lesmeister, 2015; Deng and Yu, 2014). Profoundness of architecture indicates to the number of phases of arrangement of non-linear operations in the learned service (Cho, 2014). Deep learning permits computational models that are made out of numerous preparing layers to learn representations of information with various levels of deliberation. Deep learning is a hot area of the field of machine learning that introduced the objective of moving machine learning nearer to one of its original objectives. Deep learning techniques became essential recently because of their very high recognition rates in several applications. These techniques have significantly improved for object detection, visual recognition, speech recognition and others scope like drug discovery and genomics (Cun *et al*., 2015). Facebook is also planning to empower its big data with deep learning methods to make predictions about its users (Cun *et al*., 2015). Deep learning finds complex structure in big data sets by using the backpropagation algorithm to describe how a machine should change its internal parameters that are used to calculate the representation in each layer from the representation in the former layer (Cun *et al*., 2015).

There are a lot of algorithms to machine learning like supervised learning, semi-supervised learning and unsupervised learning. Supervised learning generates a classifier or relapse capacity from labeled data, semi-supervised learning make use of both labeled data and unlabeled data and unsupervised learning use data without labeling (Deng and Yu, 2014; Cun *et al*., 2015). In this study, we are focusing on implementation of DBN and comparing it with the ordinary neural network to solve a classification problem of MNIST dataset classification.

Corresponding Author: Majid Hameed Khalaf, College of Computer, University of Anbar, Ramadi, Iraq

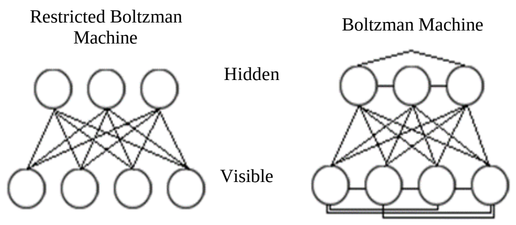

Fig. 1: Difference between RBM and BM

Restricted Boltzmann Machine: Restricted Boltzmann Machines (RBMs) are the most vastly used techniques, since they obtained interesting results in different research areas. The RBM represent as an undirected graphical model for binary vector observations that have been successfully applied to a large variety of problems and data such as images (Dahl, 2010), movie user preferences (Salakhutdinov *et al*., 2007), motion capture data (Taylor *et al*., 2011), speech recognition (Dahl *et al*., 2012) and many others.

RBMs is described as binary unit that are connected with the stochastic binary unit using symmetrically weighted connections (Hinton, 2010). The pixels relate to visible units of RBM on the grounds that their states are observed the feature detectors compare to hidden units.

RBM is an energy-based generative model that consists of two layers: first as a visible layer represented as v and the second is a hidden layer represented as h, the two layers are fully connected with each other while connections within a layer are not allowed in the same hidden layer if the connection is occurred it is called Boltzmann Machine (Hinton, 2010). This implies each visible unit is associated with every hidden unit by undirected weighted associations as shown in Fig. 1.

Contrastive divergence: Usually, the RBMs are training using the contrastive divergence training procedure (Hinton, 2010). RBMs follow the encoder-decoder paradigm where both the encoded representation and the decoded reconstruction are stochastic by natural. The encoder-decoder architecture is very useful because after training phase the feature vector can be calculated in a fast way and by reconstructing the input we can assess how well the model was able to capture the relevant information from the data (Ethem, 2014; Deng and Yu, 2014; Hinton, 2010). The energy value defined by the following function:

$$E\left(v,h\right) = -\sum ai\ vi - \sum bj\ hj - \sum vi\ hj\ wij \qquad (1)$$

Where:

v = A binary vector of visible units v = [0, 1]

h = A binary vector of hidden units h = [0, 1]

a, b = The biases of them respectively

w = The weights between visible and hidden layers

The network computes the probability for each possible pair of a visible and hidden layers by this energy function:

$$P(v, h) = 1/Ze\char`^\left(-E\left(v, h\right)\right) \qquad (2)$$

where, Z is a normalization constant called partition function by analogy with physical systems, given by the sum of all energy configurations (Lesmeister, 2015):

$$Z = \sum e\char`^\left(-E\left(v, h\right)\right) \qquad (3)$$

The network need to compute the probability that assigns to a visible vector, v is given by sum over each possible hidden vectors h as follows:

$$P(v) = 1/\ Ze\char`^\left(-E\left(v, h\right)\right) \qquad (4)$$

Hence, given a specific training vector v its probability can be raised by optimizing the weights and the biases of the network in order to lower the energy of that particular vector while raising the energy of all the others (Bengio, 2009). It can perform a stochastic gradient ascent on the log likelihood forked acquired from the training vectors by computing the derivative of the log probability with respect to the network parameters (Hinton, 2010):

$$\left(\vartheta\log P\left(v\right)\right)/\vartheta wij = \left[\left(vi\ hj\right)\ data - \left(vi\ hj\right)\ model\right] \qquad (5)$$

This equation leads to adjusting the weights by a simple learning rule to perform a stochastic gradient ascent to the training vector as follows:

$$\Delta wij\ = \alpha\left[\left(vi\ hj\right)\ data\text{-}\left(vi\ hj\right)\ model\right] \qquad (6)$$

where, α is a learning rate of the networks. Since, in RBM there are no connections between any two units in the same layer that given a particular random input configuration v, all the hidden units are autonomous of each other and the probability of h given v as follows:

$$P\ \left(hi = 1|v\right) = f\left(ai + \sum vj\ wij\right) \qquad (7)$$

where, the f(x) is the sigmoid function 1/(1+e^(-z)), for implementation purposes, hi is set to 1 when P (hi = 1|v) is greater than a given threshold number (distributed between 0 and 1) and 0 otherwise. Similarly, given a specific hidden state, h, the probability of v given h is given as follows:

$$P(vi = 1|h) = f\ (bi + \sum hj\ wij) \tag{8}$$

A much faster learning procedure was proposed by Hinton (2010). This starts by setting the states of the visible units to a training vector. The change in a weight is then given by:

$$\Delta wij = (hvi\ hj\ data - hvi\ hj\ recon) \tag{9}$$

Deep belief networks: Deep neural networks are feed forward neural networks with many layers (Ethem, 2014). A DBN is composed of several RBM layers. Hence, training a DBN consists of independent training of each RBMs, starts with the low level of RBM and gradually moving up in the hierarchy (Deng and Yu, 2014; Erhan *et al*., 2010). The learning strategy consists of two phases as shown in Fig. 2. Each hidden layer is represented as a Restricted Boltzmann Machine (RBM) and the network is trained layer by layer in a rapacious unsupervised learning, this phase is called pre-training phase. In the second phase, the refinement phase, the network is unfolded as a feed forward neural network in which the weights of hidden units are initialized using the weights forward of the conforming RBMs. A competitive layer is added as final output layer of the feed forward network which is trained using a supervised algorithm, to classify the inputs into the appropriate class type (Nuovo *et al*., 2015).

Experimental setup: In this study, the experimental setup used for MNIST digit dataset recognition is described.

MNIST: The MNIST dataset is composed of 70,000 gray-scale images of size 28×28 pixels representing handwritten digits (0-9), Fig. 3 and 4 show a sample MINST image. MNIST dataset is comprised of two sets, one for training (60,000 images) and one for testing (10,000 images). Images have been stochastically binaries according to their pixel intensity as by Salakhutdinov and Murray (2008). Pixel intensities of the standard MNIST data set range from 0 (black) to 255 (white). The 28× 28 = 784 pixels per MNIST image, converted to [0;1] according to the value of threshold (127) and finally, fed into the artificial neural network.

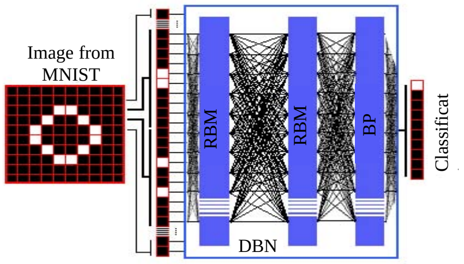

Fig. 2: Deep Belief Network (DBN)

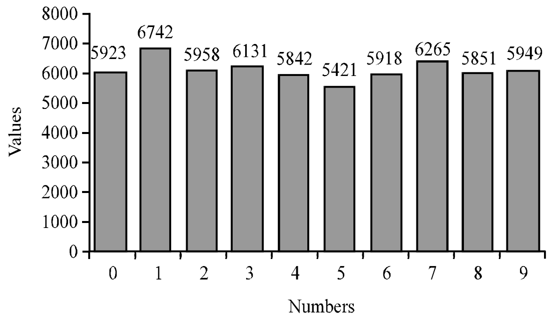

Fig. 3: MINTS training graph

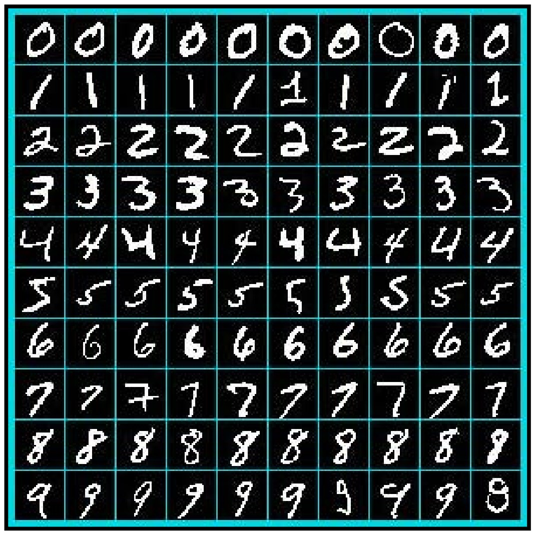

Fig. 4: Sample image in the MINST database

MATERIALS AND METHODS

In this study, the preprocessing that are used to extract the image features is described. The first step is to get the image from dataset after that binaries the image into white and black depending of the intensity of pixel, the selected threshold is set to 127, if the value of the pixel is greater than the threshold then set it to zero otherwise

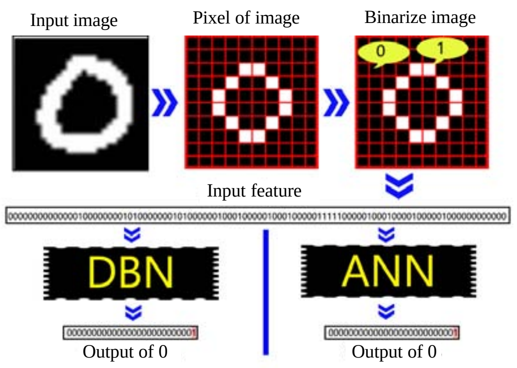

Fig. 5: Pre-processing of digit 0

set it to one. Finally, a vector of binary representation of the digit's features is generated in a row by row manner in order to be fed to the network. Figure 5 shows the preprocessing of the digit 0.

The architecture of the used ANN consists of 784 neurons as an inputs layer that represented as a vector of zeros and ones and variable number of hidden layers (changed in each training network to produce the best neural network to avoiding the over and under fitting). On the side of hidden layer, the network has weights that connected the input to hidden and hidden to output layers. These weights are randomly initialized with range of [-0.3, 0.3] then adjusted in training phase to produce the best networks. The adjusting weight is done by supervised learning using the backpropagation algorithm. The network has 10 neurons output layer. Each neuron in out put layer represent as a class of character depend on the index of these character. The following steps represents the proposed backpropagation ANN:

- Initialize the network by selecting the number of Input and output neurons, hidden neurons and layers and the output encoding
- Initialize all weights and biases to small random values, typically [-0.3, 0.3]
- Repeat until termination criterion satisfied
- Present a training example and propagate it through the network (forward pass)
- Calculate the actual output
- Adapt weights starting from the output layer and working backwards

Also in the architecture of the proposed DBN, the input, three hidden layers and output layers are the same as ANN, this is done for the comparing purpose between them. The training is in two phases the first phase is to adjust the weight between the input and hidden neurons by unsupervised learning using contrastive divergence algorithm and the second phase use supervised learning using the backpropagation algorithm to the last layer. The training in DBN is done for learn each layer at one time. The learning will work in the first phase as unsupervised and adjust the weight for all epoch then move to the next phase as supervised learning using back propagation algorithms. Algorithm 1-3 show Basic procedure for training an unsupervised DBN and algorithm 2 represents a RBM epoch training.

Algorithm 1 (basic procedure for training an unsupervised DBN):

```
D←training data.
    S←stacking of RBMs comprising to represent the DBN
Foreach rbm in S do
        if rbm index not last RBM then
        rbm ← Train_RBM(rbm, D)
```

Algorithm 2 (Restricted Boltzmann Machine epoch training):

```
D←training data.
rbm← RBM to train.
η←learning Rate.
α← momentum value.
batch_size←number of data vectors to use for each training batch.
epoch←maximum number of epoch to training.
For i in epochs range do
      Foreach batch do
            rbm←RBM_update(batch, rbm, η, α).
Increment i.
```

RBM step updates as seen in Algorithm 3 consists of one or more steps of contrastive divergence, followed by updating the weights, biases and momentums of the model.

Algorithm 3 (Single batch parameter update for RBM):

```
D←training data.
rbm ←RBM for training.
η←learning rate.
α←momentum value.
batch_size←number of data vectors to use for each training batch
epoch←maximum number of epoch to training.
// starting contrastive divergence CD_1.
Calculate p(h0) from batch.
Sample h0.
Calculate p(v) from h0.
Sample v.
Calculate p(h1) using v.
// computing gradients for this batch.
∆w←h0 batch' – p(h1) v'
∆bias←batch – v
∆hbias ←h - p(h1)
// adjusting weights:
w_vel← (wv * α) + (η * ∆w)
vbias_vel← (vbias_vel * α) + (η * ∆vbias)
vbias_vel← (hbias_vel * α) + (η* ∆hbias)
w ←w + w_vel
vbias←vbias + vbias_vel
hbias←hbias+ hbias_vel
```

Table 1: ANN and DBN results

Architecture of networks (number of neurons for each layers)	No. epochs (last layer)	Results of network		
		No. of correct identifications	No. incorrect identifications	Error rate (%)
784-500-10/NN	100	8677	1323	13.23
784-300-10/NN	100	8586	1414	14.14
784-500-300-10/DBN	100	8832	1168	11.68
784-500-300-200-10/DBN	50	9183	817	8.17
784-500-500-2000-10/DBN	0	9676	324	3.24
784-500-500-2000-10/DBN	10	9841	159	1.59
784-500-500-2000-10/DBN	40	9858	142	1.42

RESULTS AND DISCUSSION

The trained RBM with the lowest validation error was selected and then used to evaluate the performance on the MNIST test set. Results are summarized in Table 1. The best deep neural network has an error rate of only 1.42% (142 out of 10,000 are incorrectly classified). Investigation has proved that the majority of the 142 misclassified digits have either few features or no main attributes, meaning that even human perception will face difficulties to correctly identify them.

Networks that contain up to 1662000 weights can be trained using the standard gradient descent algorithm to achieve test errors below the 98.58% after 40 epochs in <80 h of training all training set that contain 60000 image.

CONCLUSION

In this study, a deep neural network architecture for MNIST dataset classification is presented where a stack of restricted Boltzmann machine to implement deep belief network for MNIST dataset is used. A comparison between MLP neural networks and deep belief networks for digit classification is done in order to identify 10 digits. Multilayer perceptron's are initialized with random weights whereas DBNs are initialized with the weights that trained using unsupervised contrastive divergence procedure.

The obtained results showed that DBNs gave a higher performance as compared with the MLP with an accuracy of approximately 98.58% for digit classification because of the DBN make a pre-training process in the unsupervised phase that make the networks working as a clustering process before the classification.

One of possible enhancements for this work is how to speed-up the training time for each RBM in DBN by excluding some neurons that are not important in feature detection and likewise extend benchmark results to different models like ConvNets and datasets MNIST.

REFERENCES

Bengio, Y., 2009. Learning deep architectures for AI. Found. Trends Mach. Learn., 2: 1-127.

Cho, K., 2014. Foundations and Advances in Deep Learning. Aalto University, Espoo, Finland, ISBN:978-952-60-5575-6, Pages: 277.

Cun, L.Y., Y. Bengio and G. Hinton, 2015. Deep learning. Nat., 521: 436-444.

Dahl, G., A.R. Mohamed and G.E. Hinton, 2010. Phone Recognition with the Mean-Covariance Restricted Boltzmann Machine. University of Toronto Press, Toronto, Canada, pp: 469-477.

Dahl, G.E., D. Yu, L. Deng and A. Acero, 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE. Trans. Audio Speech Lang. Process., 20: 30-42.

Deng, L. and D. Yu, 2014. Deep learning: Methods and applications. Found. Trends Signal Process., 7: 197-387.

Erhan, D., Y. Bengio, A. Courville, P.A. Manzagol and P. Vincent *et al*., 2010. Why does unsupervised pre-training help deep learning?. J. Mach. Learn. Res., 11: 625-660.

Ethem, A., 2014. Introduction to Machine Learning. 3rd Edn., MIT Press, Cambridge, Massachusetts.

Hinton, G., 2010. A Practical Guide to Training Restricted Boltzmann Machines. University of Toronto Press, Toronto, Canada,.

Lesmeister, C., 2015. Mastering Machine Learning with R. Packt Publishing Ltd, Birmingham, England, ISBN:978-1-78398-452-7, Pages: 374.

Nuovo, D.A., M. Vivian and A. Cangelosi, 2015. A deep learning neural network for number cognition: A bi-cultural study with the iCub. Proceedings of the 2015 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics, August 13-16, 2015, IEEE, Catania, Italy, ISBN:978-1-4673-9320-1, pp: 320-325.

Salakhutdinov, R. and I. Murray, 2008. On the quantitative analysis of deep belief networks. Proceedings of the 25th International Conference on Machine Learning, July 5-9, 2008, ACM, New York, USA., ISBN:978-1-60558-205-4, pp: 872-879.

Salakhutdinov, R., A. Mnih and G. Hinton, 2007. Restricted boltzmann machines for collaborative filtering. Proceedings of the 24th International Conference on Machine Learning, June 20-24, 2007, ACM, New York, USA., ISBN:978-1-59593-793-3, pp: 791-798.

Taylor, G.W., G.E. Hinton and S.T. Roweis, 2007. Modeling human motion using binary latent variables. Adv. Neural Inf. Process. Syst., 19: 1345-1345.