Deep Learning Training Complexity Dropping by Neural Network Pre-Training Method

Shivnath Ghosh, Santanu Koley

Abstract

Our recent research history says deep multilayer architectures might be efficient and successful to represent and trained the several algorithms. Theoretically it seems, training of neural network architecture is an easy task but the experiments' result analysis says it is difficult to train the deep learning architecture because the behavior learning layers depends on training data. Here, our objective is to answer the question why training of deep architecture is difficult and find out the standard gradient descent formula. Through the experiment it would be confirmed that unsupervised pre-training increase the learning rate and demonstrate the strength of learning procedure with respect to arbitrary initialization. This paper shows the role of pre training in terms regularization and optimization for deep learning architecture.

Keywords: Deep learning; Multilayer feed forward; Supervised pre-training; Learning rate.

Introduction

Deep learning methods intend to learn attributes hierarchies from different lower levels to higher levels architectures. With the help of composition of lower level features deep architecture learns. They include learning methods, multilevel abstractions for function mapping the input to output directly from data, automatically. For the higher level of abstraction such automatic learning has an important role. Because deep learning method is based on behavior of sensory to associatory and internal representation of associatory units, they put forwards capacity to naturally influence, unsupervised data and data from similar task to improve challenging problems [1]. Conceptual mathematical driving force for deep learning architecture comes from complexity theory [2].Theoretical result analysis suggest that complicated function with respect to deep architecture can be represented by higher level of abstraction [3].

Taking into account most recent experimental results a Restricted Boltzmann Machine (RBM) based on unsupervised learning provides better result to training deep architecture [4]. Unsupervised generative architecture supports for each layer of deep learning, classification task, machine learning dimensional reduction task, aggregation and regression [5], [6], [7]. Here our main concern

finding out the difficulties with the training of deep learning architecture and come up with a solution through pre training unsupervised method.

It can be summarized the training algorithm by including some useful research results such as deep neural and multitasking [8], unsupervised learning of distribution [9], advanced neural information [10], advanced neural processing system [11]. In our experiment output vector is h(x) = sigmoid (b + Wx) for general neural network, where x is in the form of stochastic noise, b stand for bias, W weight matrix and hidden layer sigmoid =1/ (1+ exp (-a)). If stochastic corruption Ci (x) = xi for fixed size random subset and for noisy input $x^{\llcorner}$ = sigmoid (c + WT h (C (x))) [12].

Experimental Setting And Datasets

Experimental setting and datasets may be discussed into two different types as Finite Dataset Reference and Experimental Setting requirement for training.

A. Finite Dataset Reference:

The Mixed National Institute of Standards & Technology (MNIST) database has a training set of 60,000 examples, and a test set of 10,000 examples. This is a subset of National Institute of Standards and Technology (NIST). The digits have properties that are centered in a fixed-size image. A very good dataset for researchers, learners who wants real world data for experiment, with putting fewer efforts on pre-processing.

There are four files available: http://www. http://yann.lecun.com/exdb/mnist/

I. Training set image (9912422 bytes)

II. Training set levels (28881 bytes)

III. Test set image (1648877)

IV. Test set labels (4542 bytes).

B. Experimental Setting requirement for training:

This experiment includes the feed forward multilayer neural network (deep architecture) training with pre- training data and without pre-training (may have variable number of layers).Without pre training requirements- Number of layers, learning rate(0.002, 0.005, 0.007), cost updation ($0,10^{-4}, 10^{-6}, 10^{-8}$), number of hidden units (300,750,1150). For the pre training process that includes all above plus pre training and pre training pre training rate (0.001, 0.01, 0.35, and 0.4).

Deep Learning Expected Effort and Experiment

A deep architecture, which is composition of several layers with random ordinary initial charge, will provide certain result and with pre training process it will behave in some better consequence. It is already clear that without pre training more than 5-layers training process leads the error due to error propagation. So,

increasing the number of layers without pre-training directly proportional to mounting the chance of decision toward local maxima. Fig 1 and 2 with 400 initial (experiment on MNIST data set) show the error propagation. Effect of depth performance suggest that pre-training is vital for random ordinary inputs. It is been noted for deep learning hidden layers are expected to be more than to 5 layers for better results. The results prove that pre-trained weights perform like supervised learning and easy to optimization, minimize error propagation. For better understanding it can be relate to supervised learning with optimized objective or similar to gradient based optimization. It may possible that training error in initial states is negligible but accumulative error is not ignorable in without pre-training deep learning.

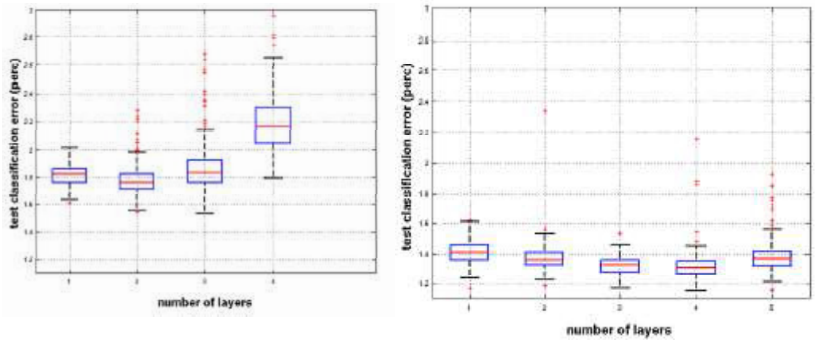

Figure 1: 400 random ordinary inputs with 5-layer without pre-training and with training consequences.

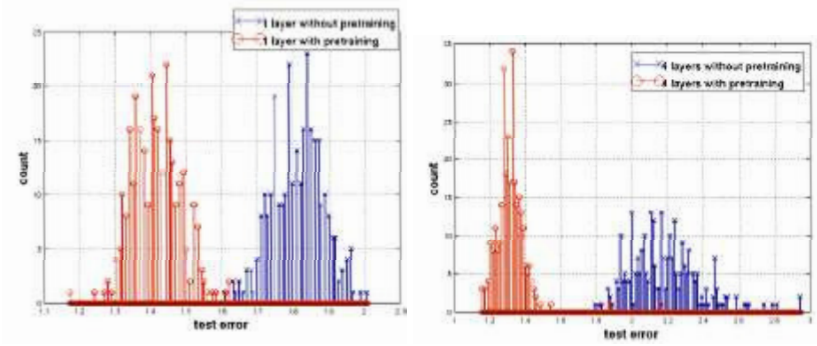

Figure 2. Trained and without pre training histogram representation, error obtained MNIST data set.

Pre-training process has a similar effect such as good "prior" knowledge, parameters or regularize. For pre-trained models:

$$P_{\text{pre-training}}(\theta) = \sum_{k} 1_{\theta \in R_k} \pi_k / v_k,$$

For without pre-training:

$$P_{\text{no-pre-training}}(\theta) = \sum_k 1_{\theta \in R_k} r_k / v_k$$

The effect of pre-training and without pertaining can be analysing by fig3. Where behaviour of pre-training is found uniform regularized.

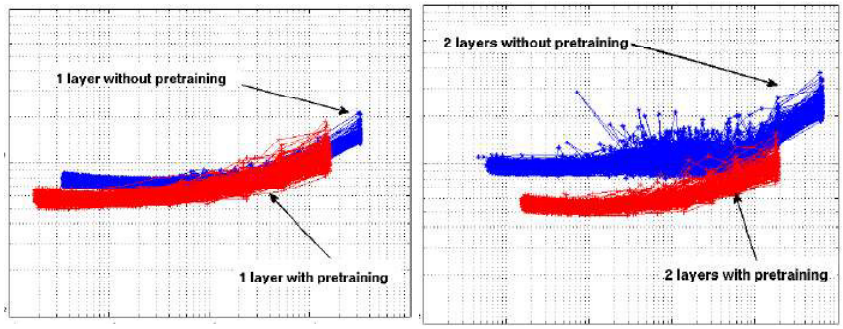

Figure 3. Behaviour of different layers in deep learning with pre-training and without pre-training, data set from MNIST.

Pre-tranning Effect Analysis Number Analysis and Unit per Layers

Here we want to explore the performance change effect cause of number of layers and units. More number of layers provides better regularities and less number of layers need additional bias with pre-training set. Deep learning architecture keeps consistent performance and act like as a regularizes. So it performs well in classification. Fig-4 suggests more hidden layer provides better consistency. It is observed that pre-training helps for laser layers and deeper network and small network has limited capacity to generalize and classification [13].

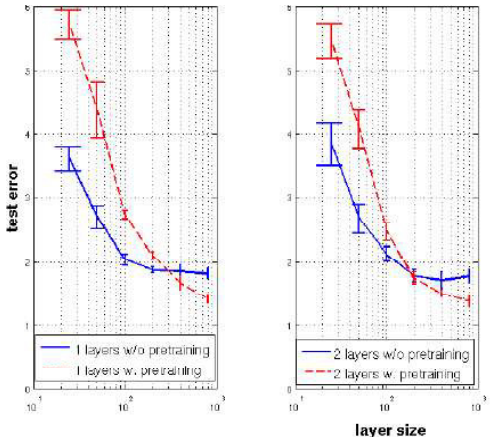

Figure 4. Experiments on MNIST, Effect of layer size, Pre-training and without pre-training error effect with 1, 2 hidden layers.

Effect of Initialization Stratrgies

After the decision the number of hidden layers and processing nodes next question is in which layer the pre-training data should be applied and/ or combination of layers. Fig-5 an analysis and result about the training data applied.

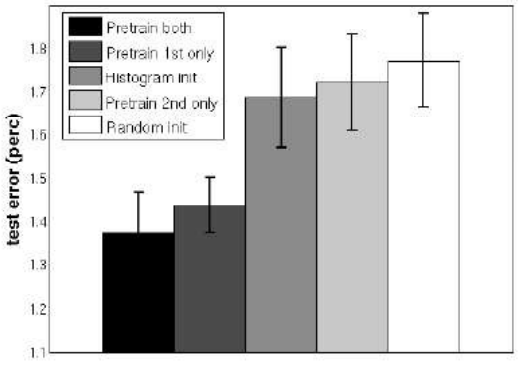

Figure 5. Role & importance initialization techniques, test error obtained, effect with pre-train the lower layers.

Distribution of weight and initialization impact: Random initialization without pre-training has more error or deviation as compare to pre-training with first or second or both layer initializations.

Calculation Difficulties of Pre-training on Different Layers

An experiment setup in hybrid initialization value added to pre-trained weight application to different layers. If we compare pre-training to first layer, skip first layer pre-training to second layer, without pre-training results, we observed pre-training of first layer will provide better result than two others options. Pre-training to second layer skipping second layer directly may provide worst result than without pre-training [14]. So, pre-training must take place in lower layer but difficulties are, if we consider deep back propagation neural network where lower layers associated less informative and consequently ineffective results. Whereas second layer or upper layer more close to output layer more informative but may be with inherited error or with ineffective consequences.

Pre-training earlier layers must be used for better & effective output instead of training of supervised layers.

Discussion & Conclusion

Accepting and getting better deep architectures remains an open problem. Our confidence is that formulating and adopting approach for learning in deep architectures which requires a more reflective understanding of the situation.

It has been shown that pre-training put in robustness to a deep architecture. The same result can be performed by without pre-training but require more deep architecture and processing with respect to error finding, but may increase local maxima problem and that is increases optimization problem. For better generalization consistency is expected.

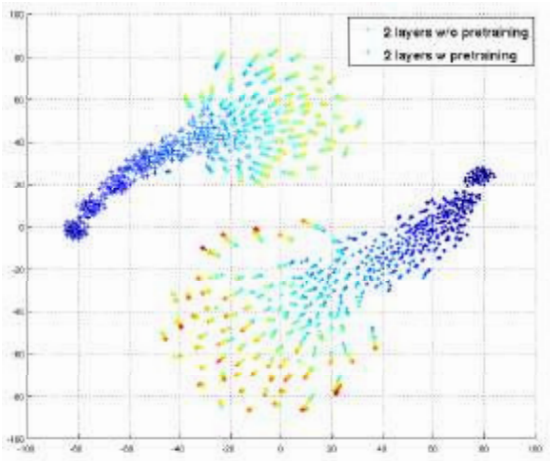

Fig. 6. Calculation over MNIST data for Pre-training and without pre-training behaviour with different colour dots.

We can see fig-6 which tells pre-training and unsupervised explanation with blue and yellow colour dotes. (MNIST, 50-50 network for each pre-trading and without pre-training).

Through this two experiment it can be seen that pre-training can reduce required training data set which behave like regularization with sufficient number of hidden layer (for systematic approach and training and adaptation requires sufficiently large number of layers.) In addition result says that pre-training has a better impact or result in higher order layers than lower order layers. Whereas in without pre-training higher order layer are with more error propagated.

Simply, pre-training has consistency, regularity, good initial marginal distribution and of course able to capture more dependencies.

References

1. Zhu, Chen & Yuille, (2009) "Unsupervised learning of probabilistic grammar-markov models" IEEE Transactions on Pattern Analysis and machine intelligence, pp.114-128.
2. Bengio, (2009) "Learning deep architectures for AI", IEEE transaction, pp. 11-17.
3. Weston, Ratle, (2008)"Deep learning via semi-supervised embedding", ICML, New York, pp. 1168-1175.

4. D. Haussler, (1994)"Unsupervised learning of distributions on binary vectors using two layer networks" Technical Report UCSC-CRL-94-25, University of California, Santa Cruz.
5. E. Hinton, S. Osindero, (2006) "A fast learning algorithm for deep belief nets", *Neural Computation*, pp. 1527–1554.
6. Collobert and J.Westo, (2008)"A unified architecture for natural language processing: Deep neural networks with multitask learning" ICML.
7. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. (2007) "An empirical evaluation of deep architectures on problems with many factors of variation",Twenty-fourth International Conference on Machine Learning, Omnipress, pp. 473–480.
8. G. E. Hinton, (2008)" Using deep belief nets to learn covariance kernels for Gaussian processes", MIT Press, Cambridge, MA.
9. Mnih& Hinton, (2009) "A scalable hierarchical distributed language model" NIPS 21, pp. 1081–1088.
10. Lee, Ekanadham, (2008) "Sparse deep belief net model for visual area", MIT Press, Cambridge.
11. S. Chopra, and Y. LeCun (2007)" Efficient learning of sparse representations with an energy-based model", MIT Press.
12. Larochelle, Y. Bengio, (2008) "Extracting and composing robust features with denoising autoencoders", Twenty-fifth International Conference on Machine Learning, pp. 1096–1103.
13. Erhan,Manzagol, (2009) "The difficulty of training deep architectures and theeffect of unsupervised pre-training" AISTATS,pp. 153–160.
14. Bengio,Simard& Frasconi, (1994) "Learning long-term dependencies with gradient descent is difficult" IEEE Transactions on Neural Networks, 5, pp. 157–166.