International Journal of Computer Engineering & Technology (IJCET)
Volume 9, Issue 3, May-June 2018, pp. 140–147, Article ID: IJCET_09_03_016
Available online at
http://www.iaeme.com/IJCET/issues.asp?JType=IJCET&VType=9&IType=3
Journal Impact Factor (2016): 9.3590(Calculated by GISI) www.jifactor.com
ISSN Print: 0976-6367 and ISSN Online: 0976–6375
© **IAEME** Publication.

STOCHASTIC BACK PROPAGATION FOR SCALABLE AND INFERENCE LEARNING

K. Thirupal reddy
Research scholar, Department of Computer Science and Engineering,
Rayalaseema University, Kurnool, A.P

Dr. T. Swarnalatha
Professor, Department of Computer Science and Engineering,
Narayana Engineering College, Nellore, A.P

ABSTRACT

Neural networks and deep learning currently provide the best solutions to many problems in image recognition, speech recognition, and natural language processing. This work presents a novel algorithm based on stochastic back propagation rules through stochastic variables. The proposed algorithm is scalable for inference and learning. In order to demonstrate the approach 2 networks were created with 100 hidden layers and trained using training set of different sizes. The proposed approach has been validated using MSIT data set and also though comparison of results available in the literature. The comparison is made in terms of average RMSE value. The result demonstrates the improved performance of the proposed approach.

Keywords: Back propagation, stochastic, Artificial Neural Networks, MSIT, RMSE.

Cite this Article: K. Thirupal reddy and Dr T. Swarnalatha, Stochastic Back propagation for Scalable and Inference Learning. *International Journal of Computer Engineering & Technology*, 9(3), 2018, pp. 140–147.
http://www.iaeme.com/IJCET/issues.asp?JType=IJCET&VType=9&IType=3

1. INTRODUCTION

Artificial Neural Networks are systems designed based on the inspiration drawn from the functioning of human brain. A typical Artificial Neural Network consists of collection of Arithmetic computing units called as nodes or neurons. Each of these nodes is connected together in a network of interconnected layers. Neural network (NN) models are parallel computing networks inspired by animal nervous system. They are adopted more commonly for forecasting and prediction in many fields. A neural network typically consists of input layer (with "" input neurons), one or many hidden layers (with "" number of hidden layers and "" number of hidden neurons), and an output layer (with "" number of output neurons).

Each layer will be interconnected with the weights (randomly generated). The information has to be feed-forwarded from each input neuron to all hidden neurons through these weights. Then, information processes use transfer function (linear or sigmoid) at each hidden neuron. Then, all the processed values have to be summed up at each hidden neuron and information to be passed on to the output neuron through connecting weights. Then again, the information is to be processed through transfer function at output neuron to get final value. Bias is considered in order to eliminate or offset the dominant solutions at hidden layer and at output layer. The whole process of feed-forward from input layer to output layer is termed as feed-forward process. The final observed value at the output layer is compared with the target value. The difference in error between the observed and predicted value is then evaluated. Then, a back propagation process is used to back-propagate errors until the weights are optimized to obtain minimum error between the observed and predicted value. In back propagation, partial derivatives with respect to the connected weights are calculated. Chain rule is used to get the updated weight.

The backpropagation algorithm was originally introduced in the 1970s, but its importance wasn't fully appreciated until a famous 1986 paper by David Rumelhart, Geoffrey Hinton, and Ronald Williams [1]. That paper describes several neural networks where backpropagation works far faster than earlier approaches to learning, making it possible to use neural nets to solve problems which had previously been insoluble. Today, the backpropagation algorithm is the workhorse of learning in neural networks. The ability of neuro emulation to imitate the behavior of real cases and capture nonlinearity has made it a suitable method for modeling. Back propagation learning algorithm using gradient (steepest descent) based approach is widely used in the neural network training. The training of NN is done by minimizing the error function (Mean Square Error or Root Mean Square Error) between the predicted and the observed value. However, a back propagation learning algorithm with gradient based approach in neural network training has numerous drawbacks such as the fact that performance depends on initial weights and that the likelihood of solution reaching global optimum is not assured. In order to overcome these limitations, it is essential to develop an efficient method to optimize the NN.

Recent efforts in machine learning research focused on developing scalable probabilistic models. Deep generative models are known to be able to generalize better to unknown data since the directed counter parts can better capture high level abstractions in the dataset. However, efficient inference algorithms for directed generative models have been a major problem. One of the core problems of modern statistics is to approximate difficult-to-compute probability densities. This problem is especially important in Bayesian statistics, which frames all inference about unknown quantities as a calculation about the posterior. Modern Bayesian statistics relies on models for which the posterior is not easy to compute and corresponding algorithms for approximating them.

The gold standard method for Bayesian learning in neural networks is Hamilton Monte Carlo (HMC) [2]. This is a batch method that can perform poorly on large data sets and also HMC requires problem-specific tuning parameters such as the length and number of integration steps. One alternative to MCMC inference in neural networks is the Laplace approximation [3]. However, the Laplace approximation requires computation of the inverse Hessian of the log likelihood, which can be infeasible to compute for large networks. Diagonal approximations to the Hessian are possible, but performance can deteriorate considerably. One alternative approach based on EP is described by Jylanki et al. [4]. This is a batch method that is not expected to scale to large data sets and, unlike PBP, it requires numerical quadrature. Jylanki keeps in memory several approximate factors for each data point, which is not feasible in large scale settings. Furthermore, by using latent variables, this method breaks each likelihood factor into additional sub-factors that are incorporated into the posterior approximation in multiple disconnected steps. PBP incorporates each likelihood factor in a single step, which is expected to be more accurate.

A scalable variational inference (VI) method for neural networks is described by Graves [5]. This method maximizes a lower bound on the marginal likelihood of the NN. The computation of this bound requires computing the expectation of the log of the numerator of the exact posterior under a factorized Gaussian approximation. This is intractable in general, and so Graves [5] proposes a Monte Carlo approximation to compute the lower bound, which is then optimized using a second approximation for stochastic gradient descent (SGD). While SGD is a common approach to optimization of neural networks, the initial approximation leads to inefficient use of data. As a result, the VI approach tends to generate poor solutions for larger data sets over which only a few passes are possible.

The technique that is most closely related to PBP is the expectation-back propagation (EBP) method described by Soudry et al. [6] which proposes an online EP technique for neural networks with sign activation functions and binary weights, with an extension to continuous weights. As with PBP, EBP also includes a forward propagation of probabilities followed by a backward propagation of gradients. However, there are three important contributions of PBP with respect to EBP. First, EBP can only model data with binary targets and cannot be applied when the $\boldsymbol{y_n}$ are continuous (as in regression), while PBP assumes continuous $\boldsymbol{y_n}$ and can be extended to binary targets using the same method as in EBP. Second, and more importantly, EBP with continuous weights only updates the mean parameters of the Gaussian posterior approximations.

The primary objective of the proposed work is to design and develop stochastic back propagation rules through stochastic variables to implement a novel algorithm that is suitable for scalable inference and learning. The approach borrows from Deep Neural Networks and Bayesian inference to obtain a generalized class of deep, directed generative models. In order to validate the proposed approach, MNIST data set has been used. In order to demonstrate the approach 2 networks were created with 100 hidden layers. One network was trained using a total of 6000 samples and the other one was trained using a total of 10000 samples. The results have been validated by comparing different sample sets of different sizes by computing the average RMSE value for each sample set. The result demonstrates the improved performance of the proposed approach and is compared with the results available in the literature.

2. EXTRACT INFORMATION

2.1. Proposed Approach

The proposed approach is explained in detail in this section, given data $\boldsymbol{D} = \{\boldsymbol{x_n}, \boldsymbol{y_n}\}_{\boldsymbol{n=1}}^{\boldsymbol{N}}$, made up of D-dimensional feature vectors $\boldsymbol{x_n} \in \mathbb{R}^{\boldsymbol{D}}$ and corresponding scalar target variables $\boldsymbol{y_n} \in \mathbb{R}$, we assume that each $\boldsymbol{y_n} = \boldsymbol{f}(\boldsymbol{x_n}; \boldsymbol{W}) + \in_{\boldsymbol{n}}$, where f (.; W) is the output of a multi-layer neural network with connections between consecutive layers and weights given by W. The evaluations of this NN are corrupted by additive noise variables $\in_{\boldsymbol{n}}$ where $\in_{\boldsymbol{n}} \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{\gamma^{-1}})$.

The NN has L layers, where $\boldsymbol{V_l}$ is the number of hidden units in layer $\boldsymbol{l}$, and $\boldsymbol{W} = \{\boldsymbol{W_l}\}_{\boldsymbol{l=1}}^{\boldsymbol{L}}$ is the collection of $\boldsymbol{V_l} \ \boldsymbol{X} \ (\boldsymbol{V_{l-1}} + \boldsymbol{1})$ weight matrices between the fully connected layers. The +1 is introduced here to account for the additional per-layer biases. We denote the outputs of the layers by vectors $\{\boldsymbol{z_l}\}_{\boldsymbol{l=0}}^{\boldsymbol{L}}$, where $\boldsymbol{z_0}$ is the input layer, $\{\boldsymbol{z_l}\}_{\boldsymbol{l=0}}^{\boldsymbol{L-1}}$ l=1 are the hidden units and $\boldsymbol{z_L}$ denotes the output layer, which is one-dimensional since the $\boldsymbol{y_n}$ are scalars. The input to the $\boldsymbol{l-th}$ layer is defined as $\boldsymbol{a_l} = \boldsymbol{W_l z_{l-1}} / \sqrt{\boldsymbol{V_{l-1}+1}}$, where the factor $\boldsymbol{1} / \sqrt{\boldsymbol{V_{l-1}+1}}$ keeps the scale of the input to each neuron independent of the number of incoming connections. The activation functions for each hidden layer are rectified linear units (ReLUs) [6], i.e., a(x) = max(x; 0).

Let y be an N-dimensional vector with the targets $\boldsymbol{y_n}$ and X be an N X D matrix of feature vectors $\boldsymbol{x_n}$ .The likelihood for the network weights W and the noise precision $\boldsymbol{\gamma}$, with data $\boldsymbol{D = (X, y)}$ ***is then***

$$p(y|W, X, \gamma) = \prod_{n=1}^{N} N\big(y_n | f(x_n; W), \gamma^{-1}\big) \tag{1}$$

To complete our probabilistic model, we specify a Gaussian prior distribution for each entry in each of the weight matrices in $\boldsymbol{W}$. In particular,

$$p(W|\lambda) = \prod_{l=1}^{L} \prod_{i=1}^{V_l} \prod_{j=1}^{V_{l-1}+1} N\big(w_{ij}, l|0, \lambda^{-1}\big) \tag{2}$$

Where $\boldsymbol{w_{ij.l}}$ is the entry in the i-th row and j-th column of $\boldsymbol{W_l}$and $\boldsymbol{\lambda}$ is a precision parameter. The hyper-prior for λ is chosen to be a gamma distribution, i.e.,$\boldsymbol{p(\lambda)} = \boldsymbol{Gam\big(\lambda|\alpha_0^\lambda, \beta_0^\lambda\big)}$ with shape $\boldsymbol{\alpha_0^\lambda = 6}$ and inverse scale $\boldsymbol{\beta_0^\lambda = 6}$. The values chosen for $\boldsymbol{\alpha_0^\lambda}$ ***and*** $\boldsymbol{\beta_0^\lambda}$ make this equivalent to having observed $\boldsymbol{v} =$ 12 samples from $\boldsymbol{N(0\,, \lambda^{-1})}$ with empirical variance equal to 1. This relatively low value for $\boldsymbol{v}$ compared to the large number N of observed data points makes this prior weakly informative. The prior for the noise precision $\boldsymbol{\gamma}$ is also gamma: $\boldsymbol{p(\gamma) = Gam\big(\gamma|\alpha_0^\gamma, \beta_0^\gamma\big)}$. We assume that the $\boldsymbol{y_n}$ have been normalized to have unit variance and, as above, we fix $\boldsymbol{\alpha_0^\gamma = 6}$ and $\boldsymbol{\beta_0^\gamma = 6}$.The posterior distribution for the parameters W,$\boldsymbol{\gamma}$ and λ can then be obtained by applying Bayes' rule:

$$p(W, \gamma, \lambda|D) = \frac{p(y|W,X,\gamma)p(W|\lambda)p(\lambda)p(\gamma)}{p(y|X)} \tag{3}$$

where p(y I X) is a normalization constant. Given a new input vector $\boldsymbol{x_*}$, we can then make predictions for its output $\boldsymbol{y_*}$ using the predictive distribution given by

$$p(y_*|x_*, D) = \int p(y_*|x_*, W, \gamma)\, p(W, \gamma, \lambda|D) d\gamma d\lambda dW, \tag{4}$$

Where $\boldsymbol{(y_*|x_*, W, \gamma) = N(y_*|f(x_*), \gamma)}$. However, the exact computation of $\boldsymbol{p(W, \gamma, \lambda|D)}$ and $\boldsymbol{p(y_*|x_*)}$ is not tractable in most cases. Therefore, in practice we have to resort to approximate inference methods. In the following section we describe a technique for approximate Bayesian inference in NN models that is both fast and also offers excellent predictive performance.

Having specified a probabilistic model for the posterior distribution over the inducing points can be written as $\boldsymbol{p(u|X, y) \propto p(u) \prod_n p(y_n|u, x_n)}$. This quantity can then be used for prediction of output given a test input, $\boldsymbol{p(y_*|x_*, X, y) = \int du p(u|X)\, p(y^*|u, x^*)}$. However, the posterior of u is not analytically tractable when there is more than one GP layer in the model. As such, approximate inference is needed; here we use Stochastic Expectation Propagation (SEP), a recently proposed modification to Expectation Propagation (EP) [7].

In SEP, the posterior $\boldsymbol{p(u|X, y)}$ is approximated by $\boldsymbol{(u) \propto p(u) g(u)^N}$, where the factor $\boldsymbol{g(u)}$ could be thought of as an *average* data factor that captures the average effect of a likelihood term on the posterior. The form chosen, though seems limited as first, in practice performs almost as well as EP in which there is a factor $\boldsymbol{g_n(u)}$ per data point, while significant reducing EP's memory footprint [7]. Specifically for our model, the memory complexity of EP is $\mathbf{O(NM^2)}$ as we need to store the mean and covariance matrix for each data factor; in contrast, such requirement for SEP is only O($\mathbf{M}^2$) regardless of the number of training points.

The SEP procedure involves looping through the dataset multiple times and performing the following steps:

1. Remove $\boldsymbol{g(u)}$ from the approximate posterior to form the cavity distribution.
2. Incorporate a likelihood term into the cavity to form the tilted distribution.
3. Moment match the approximate posterior to this distribution, and in addition to EP
4. Perform a small update to the average factor $\boldsymbol{g(u)}$. we choose a Gaussian form for both$\boldsymbol{q(u)}$ ***and*** $\boldsymbol{g(u)}$, and as a result steps 1 and 4 are analytically tractable.

3. MNSIT DATA SET

MNIST (Mixed National Institute of Standards and Technology) database is dataset for handwritten digits, distributed by Yann Lecun's THE MNIST DATABASE of handwritten digits website [8].The MNIST database was constructed from NIST's Special Database 3 and Special Database 1 which contain binary images of handwritten digits [9] . The MNIST training set is composed of 30,000 patterns from SD-3 and 30,000 patterns from SD-1. The 60,000 pattern training set contained examples from approximately 250 writers. The original black and white (bilevel) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. The images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field. A sample data from the data set is shown using figure (1) and a labeled visualization is given using figure (2).

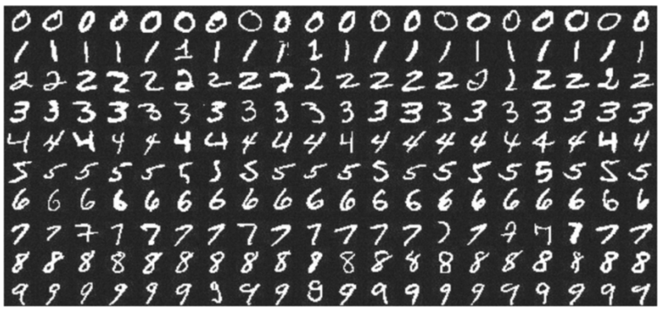

Figure 1 A sample subset of MNSIT data set for visualization

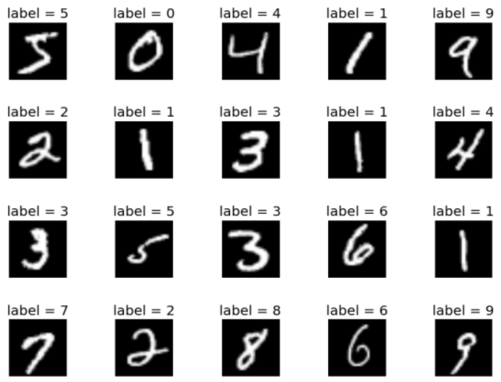

Figure 2 "Handwritten digit image" and its "label" from MNIST dataset.

4. RESULTS AND DISCUSSIONS

The proposed approach has been coded using Matlab R 2014 and in order to speed up the back end processing C++ mex files have been used for executing certain aspects of the proposed approach. The experimentation was carried in regard to the run time on Intel Core i7-2600 desktop with 64-bit Windows 7 OS and 16GB RAM.

In order to train the System, 3000 samples from SD1 and 3000 samples from SD2 have been considered for configuring Network 1 (Net-1) and 5000 Samples from SD1 and 5000 samples from SD2 have been considered for training and configuring Network 2 (Net-2).The training set and testing test from which the random samples were chosen from different writers. Both the networks comprised of 100 hidden layers.

In order to demonstrate and analyse the robustness of the approach none of the preprocessing steps deskewing, blurring, noise removal or pixel shifting have been implemented.The performance has been validated for randomly selected samples of sizes 25 (S25), 50 (S50), 75(S75), 100(S100) and 150(S150). The performance is evaluated in terms of average RMSE for each sample size.

The Root Mean Square Error (RMSE) (also called the root mean square deviation, RMSD) is a frequently used measure of the difference between values predicted by a model and the values actually observed from the environment that is being modeled. These individual differences are also called residuals, and the RMSE serves to aggregate them into a single measure of predictive power. The RMSE of a model prediction with respect to the estimated variable X_{model} is defined as the square root of the mean squared error:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_{obs,i} - X_{model,i})^2}{n}} \quad 5$$

Where X_{obs} is observed values and X_{model} is modeled values at time/place i.

The training time for the proposed approach for the configured network is tabulated using the table 1.0.

Table 1 Training time for the proposed network

Network	Number of training samples	Number of hidden layers	Number of training Epochs	Training time (Seconds)
Net -1	6000	100	30000	8280
Net-2	10000	100	30000	12120

It can be inferred from the table 1.0 that there is a considerable increase in the training time for Net-2. This can be attributed to the increased number of training samples in the training data set. The Net -2 has been trained using a data set which has 40 % more number of data points than the training data set used for net 1, subsequently the training time also shows 31.63 % increase in the training time when compared to the training time of Net -1 The average RMSE value for different randomly chosen test data sets are given in the following table 2.

Table 2 Average RMSE for different test sets

Network	S25	S50	S75	S100	S150
Net -1	1.32	1.17	1.06	0.99	0.86
Net-2	1.21	1.11	1.02	0.95	0.84

From the table 2 it can be observed that the average RMSE value decreases with the increase in number of samples, primarily because of the error being averaged out for higher number of samples. It can also be inferred from the table that there is a significant reduction in error for Net-1 when compared with Net-2. This can be attributed to the increased training samples, which was reflected in the increased training time as well. A comparative representation of errors between the two networks is illustrated using figure (3)

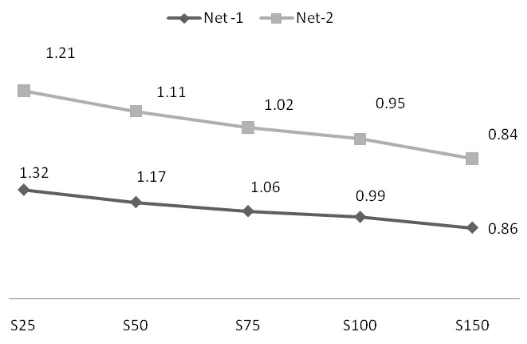

Figure 3 Average RMSE for different sample sets of different sizes

The percentage reduction in error for Net -2 over Net-1 is illustrated using the figure 4. It can be inferred from the figure that maximum reduction of 8.33 percentages is observed in the case of sample S25.

As the sample size increases there is a marked reduction in difference between the error percentages of the two networks. It is an interesting point to observe as it gives an indication of possible tradeoff between the training samples, which increases the training time and the error percentage.

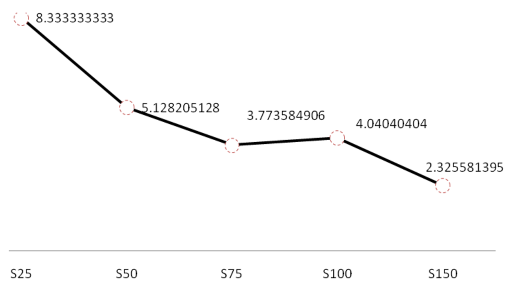

Figure 4 Percentage reduction in Average RMSE for Net -2 over Net-1

The results are validated by comparing, the results available in the literature for the same data set. The results are compared by least error reported in that literature. It can be inferred from the table 3, that the proposed approach delivers better results against the comparisons.

Table 3 Comparison of results available in the literature

S.No.	Author	Type of Classification / Network	Minimum Error Reported
1	Proposed	Stochastic Back Propagation	0.84
2	LeCun et.al. [10]	2-layer NN, 300 hidden units	4.7
3	LeCun et.al. [10]	Support Vector Machine (SVM)	1.1
4	Ranzato et.al. [11]	Convolution Neural Networks	0.89
5	Kegl et,al.[12]	Stumps on Haar features	1.02

5. CONCLUSIONS

The proposed approach being scalable is suited for high volume data sets and is suitable for big data By employing stochastic rules , joint optimization of all the parameters are ensured so that the chances of the solution getting trapped in the local minima is less. It can be inferred from the results the proposed approach delivers a performance that is better and comparable with the results presented in the literature. Similarly the results also illustrate the significance of training samples in reducing the error percentage.

REFERENCES

[1] David E. Rumelhart; Geoffrey E. Hinton; & Ronald J. Williams., Learning representations by back-propagating errors Nature volume 323, pages 533–536, 1986.

[2] Neal, Radford M. Bayesian learning for neural networks. PhD thesis, University of Toronto, 1995.

[3] MacKay, David J. C. A practical Bayesian framework for backpropagation networks. Neural computation, 4(3):448–472,1992c.

[4] Jylanki, Pasi, Nummenmaa, Aapo, and Vehtari, Aki. Expectation propagation for neural networks with sparsity-promoting priors. The Journal of Machine Learning Research, 15(1):1849–1901, 2014.

[5] Graves, Alex. Practical variational inference for neural networks. in Advances in Neural Information Processing Systems 24 , pp.2348–2356. 2011.

[6] Soudry, Daniel, Hubara, Itay, and Meir, Ron. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In Advances in Neural Information Processing Systems 27, pp. 963–971. 2014.

[7] A. Korattikara, V. Rathod, K. Murphy, and M. Welling, “Bayesian dark knowledge,” in Advances in Neural Information Processing Systems 29, 2015.

[8] http://yann.lecun.com/exdb/mnist/

[9] https://www.nist.gov/srd/nist-special-database-3

[10] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, 1998.

[11] Marc'Aurelio Ranzato, Fu-Jie Huang, Y-Lan Boureau and Yann LeCun: Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition, Proc. Computer Vision and Pattern Recognition Conference (CVPR'07), IEEE Press, 2007.

[12] Bal´azs K´egl and R´obert Busa-Fekete, Boosting products of base classifiers,ICML '09 Proceedings of the 26th Annual International Conference on Machine Learning Pages 497-504