APPROXIMATE GREATEST DESCENT IN NEURAL NETWORK OPTIMIZATION

KING HANN LIM∗, HONG HUI TAN∗
AND HENDRA G. HARNO**

∗Faculty of Engineering and Science
Curtin University Malaysia, Malaysia
**Department of Aerospace and Software Engineering
Gyeongsang National University, South Korea

Abstract. Numerical optimization is required in artificial neural network to update weights iteratively for learning capability. In this paper, we propose the use of Approximate Greatest Descent (AGD) algorithm to optimize neural network weights using long-term backpropagation manner. The modification and development of AGD into stochastic diagonal AGD (SDAGD) algorithm could improve the learning ability and structural simplicity for deep learning neural networks. It is derived from the operation of a multi-stage decision control system which consists of two phases: (1) when local search region does not contain the minimum point, iteration shall be defined at the boundary of the local search region, (2) when local region contains the minimum point, Newton method is approximated for faster convergence. The integration of SDAGD into Multilayered perceptron (MLP) network is investigated with the goal of improving the learning ability and structural simplicity. Simulation results showed that two-layer MLP with SDAGD achieved a misclassification rate of 9.4% on a smaller mixed national institute of national and technology (MNIST) dataset. MNIST is a database equipped with handwritten digits images suitable for algorithm prototyping in artificial neural networks.

1. Introduction. Deep structured learning has emerged as a new area of Artificial Neural Networks (ANN) research with its wide range of applications in signal and information processing. There are two general aspects often discussed in various high level descriptions of deep learning: i.e. (a) its structural models consisting of multiple layers or stages for nonlinear information processing, (b) optimization techniques for supervised or unsupervised learning of feature representation at sequentially deeper and larger abstract layers. However, non-linear higher level abstraction leads to the complex learning of good representation in the neural network without compensating with the information loss and confusion. In addition, it increases the cost of training problems such as ill-conditioning in weight initialization and local minima trapped in an objective function.

The learning process of ANN is achieved through neural network optimization. In general, optimization in mathematical term is the ability to minimize or maximize

2010 Mathematics Subject Classification. Primary: 58F15, 58F17; Secondary: 53C35.

Key words and phrases. Approximate greatest descent, two-phase optimization, neural network, multilayer perceptron, machine learning.

Thank you Professor Goh Bean San for in depth explanation on AGD.

∗ Corresponding author: King Hann Lim.
an objective function. Current optimization literature can be classified into two major aspects, (a) line search approach and (b) trust region approach. Line search approach computes the step direction based on the gradient information from the objective function followed by a heuristic choice of step length to minimize the error. On the other hand, trust region approach computes step direction via quadratic model that approximates a neighborhood of search region. Some of the well-known optimization techniques include, gradient descent (GD), Newton, Quasi-Newton (QN), Gauss-Newton (GN) and Levenberg-Marquardt (LM) methods.

GD [8] uses the gradient (first-order) information from the objective function to construct iteration steps. The computational cost in GD is much lower than the second-order derivative methods. However, GD often suffers from local minima problem because the gradient information does not fully interpret the actual geometry of an error surface. Hence, second-order derivative methods could achieve better performance than first-order derivative methods due to its better learning capability using the additional curvature information. One of second-order derivative examples is Newton method and it uses full Hessian [12] together with the gradient information to compute the change of update weights in the network. As a result, Newton method performs faster convergence rate if the objective function is quadratic and the initial guess of starting point is near to the solution. Nevertheless, due to the computation of inverse Hessian, Newton method is not recommended for function with many variables. It also suffers from singular Hessian problem when dealing with indefinite Hessian.QN [14] arises from Newton method to solve computation problem by using estimated Hessian. The most successful Hessian estimation scheme is the Broyden-Fletcher-Goldfard-Shanon (BFGS) [8] method but QN is suffering from storage problem due to the need to store large set of variables for estimation. GN [8] is another variances of Newton method which helps to solve heavy computational problems by using truncated Hessian. Although it does not need to deal with high storage cost it has suffered the problem of indefinite Hessian. LM [1] evolves from GN to resolve the indefinite Hessian problem by introducing a damping parameter. A relatively small value is added to truncated Hessian to avoid running into singular Hessian as the learning iterates. These problems become more significant when it is applied to the field of ANN parameters optimization. An efficient algorithm that avoids ill-conditioned weights and local optima while retaining optimal trajectory could help boosting the performance of ANN. Although there are plenty state-of-the-art learning algorithm available in ANN training, the strategy to seek for an optimal weights in ANN is yet to be discovered for all engineering applications.

The application of ANNs can be found in the wide range of industries from data analytic tools to future trend estimation and autonomous decision making. The performance of ANNs has drawn a huge attention from researchers due to its generalization to all applications and quick self-learning capability from real-life experience [9]. In the learning scheme of ANN, the weights are learned based on a set of labeled training samples with input patterns and targeted output using numerical optimization in the backpropagation (BP) manner. During BP, network errors are iteratively computed between network output and targeted values and backpropagated across the network for weights updating. Standard BP uses GDs approach to search for an optimum solution. However, ANN usually has a non-convex objective function with the presence of multiple local minimas [5]. Therefore, the searching
of optimal solution may stuck in the local minima and contribute to poor performance during network training. In addition, standard BP is more complex due to tedious selection of ideal step length and network controlling parameters [8, 3]. Some variances of GD are conjugate GD (CGD) [13] and stochastic GD (SGD) [4]. CGD avoids the use of linear algebra in iteration construction to reduce memory cost. It is suitable for function with many variables but the convergence rate is fairly comparable to other techniques. SGD introduces stochasticity to BP training samples with partial and noisy gradient. It is known to improve the convergence rate and networks generalizability. On top of that, second-order derivative method comprises the stochastic diagonal LM (SDLM). SDLM emerges from traditional LM with three approximations i.e. (a) dropping the off-diagonal terms of the Hessian matrices, (b) utilizing truncated Hessian in GN, and (c) estimating Hessian based on subset of entire training set [10, 2]. The Hessian approximation is aimed to reduce computational cost whilst remain the stochastic approach of SGD with second-order derivative method. Nonetheless, the choice of damping parameter has less theoretical justification [6].

Approximate Greatest Descent (AGD) [6, 7] is proposed in this paper as the learning algorithm of ANN. AGD utilizes the optimal control theory by segregating the learning trajectory into two phases. The implementation of AGD in optimizing the ANN parameters is not found in the literature review. This work serves as the pioneer in the field of ANN. In this paper, a modified version of AGD with stochastic diagonal approximation is proposed to iteratively compute errors and update the weights in the backpropagation manner. It incorporates the efficient learning of AGD algorithm with stochastic diagonal approximations to have more optimal ANN training and maintain low training time. We investigate the feasibility of stochastic diagonal AGD (SDAGD) in a Multilayered Perceptron (MLP) network in terms of missclassification rate and error rate. Subsequently, the aim of applying SDAGD is to improve the learning capability in neural network while maintain its network structure.

The presentation of this paper is outlined into five sections. Section 2 briefly presents the theory in neural networks. Section 3 shows the derivation of SDAGD into back propagation algorithm. In Section 4, an experimental setup is explained, followed by the discussion of SDAGD implementation with MLP. The last section contains a conclusion and future developments.

2. Backpropagation in Multilayer Perceptron. MLP [11] with BP learning scheme is represented Figure 1. MLP has the characteristic of nonlinear input-output mapping where function $M(X, W) = F^{(2)}(W^{(2)}(F^{(1)}(XW^{(1)})))$. $F^{(l)}(\cdot)$ is the nonlinear activation function of $l$-th layer, $X = \{X_1...X_m\}$ is the input sample, $W$ represents the adjustable weights and $Y = \{Y_1...Y_n\}$ is the output. An objective function $E_p = C(D, M(X, W))$, where $C(\cdot)$ is the difference between targeted output $D = \{D_1...D_n\}$ and the output from the network $M(X, W)$. In practice, the objective function in incremental neural network training is formulated as the squared error,

$$E = \frac{1}{2} (D - Y)^2.$$ (1)

The role of objective function is to find the optimum value of $W$ that minimizes $E(W)$, this is called the ANN optimization. For simplicity, the output of each layer is written as,
\[ X^{(l)} = F(Y^{(l)}) = F(W^{(l)}X^{(l-1)}). \] (2)

By applying chain rule, the classical BP equations are obtained as,

\[ \frac{\partial E}{\partial W^{(l)}} = \frac{\partial E}{\partial Y^{(l)}} \frac{\partial Y^{(l)}}{\partial W^{(l)}} = \frac{\partial E}{\partial Y^{(l)}} X^{(l-1)}, \] (3)

\[ \frac{\partial E}{\partial Y^{(l)}} = F'(Y^{(l)}) \frac{\partial E}{\partial X^{(l-1)}}, \] (4)

where \( F'(. \) \) is the partial derivatives of squashing function \( F(\cdot) \). The weight updates of a network can be computed iteratively using BP to find an optimal solution which minimizes the objective function \( E(W) \).

\[ W_{k+1} = W_k + \eta \frac{\partial E}{\partial W}, \] (5)

where \( W_{k+1} \) is the adjusted weight from \( W_k \), \( k \) represents the number of iteration and \( \eta \) is the learning rate. The iteration steps from BP can be generalized into a standard numerical optimization update rule. Learning rate \( \eta \) is equivalent to the step length \( \alpha_k \) and \( \frac{\partial E}{\partial W} \) represents the search direction \( p_k \) in

\[ W_{k+1} = W_k + \alpha_k p_k. \] (6)

Therefore, numerical optimization techniques are the key improvement to search for an optimal solution in neural network. The next section shows the derivation of second-order optimization in BP using SDAGD algorithm [16].

**Figure 1.** Structure of two layer multilayer perceptron.
3. **Backpropagation using SDAGD.** In order to adapt AGD in stochastic diagonal manner, Hessian is used for iteration computation instead of gradient information together with 3 approximations. Letting \(g(W) = \frac{\partial E}{\partial W}\) and \(H(W) = \frac{\partial^2 E}{\partial W^2}\), the exact Hessian formula for MLP can be written as [16],

\[
H(W) = \sum_{(i,j)} \sum_{(r,s)} \frac{\partial^2 E}{\partial w_{ij} \partial w_{rs}},
\]

(7)

where \(w_{ij}\) and \(w_{rs}\) are connection weight from unit \(j\) to unit \(i\) and unit \(s\) to unit \(r\) respectively. Unit \(i\), \(j\), \(r\) and \(s\) represents the nodes of an ANN structure. First approximation of SDAGD suggests to drop the off-diagonal terms of Hessian as there are no intra-connection in the same layer. Hence, the Hessian is written as,

\[
H(W) = \sum_{(i,j)} \frac{\partial^2 E}{\partial w^2_{ij}}.
\]

(8)

The local second-order derivative with respect to the sum of inputs of the downstream unit can be computed as,

\[
\frac{\partial^2 E}{\partial w^2_{ij}} = \frac{\partial^2 E}{\partial y^2_i} x^2_j.
\]

(9)

By utilizing product rule and chain rule, expanding \(\frac{\partial^2 E}{\partial y^2_i}\) yields,

\[
\frac{\partial^2 E}{\partial y^2_i} = F'(y_i)^2 \sum_r w^2_{ir} \frac{\partial^2 E}{\partial y^2_r} + F''(y_i) \frac{\partial^2 E}{\partial x_i}.
\]

(10)

As for the second approximation, truncated Hessian is used instead of full Hessian to save computation cost. It also help to solve the existence of negative terms in the second-order derivatives. Hence, the truncated Hessian is written as,

\[
\tilde{H}(W) = x^2_j F'(y_i)^2 \sum_r W^2_{r_i} \frac{\partial^2 E}{\partial y^2_r}.
\]

(11)

To further reduce the computation time for Hessian estimation, third approximation suggest to only use a smaller subset of training samples for Hessian estimation,

\[
\bar{H}(W) = \frac{1}{S} \sum_{i=1}^{S} \tilde{H}(W),
\]

(12)

where \(S\) is the number of training samples in the subset.

With the intention of adapting AGD as the learning algorithm, (6) is modified to cater the derivation of AGD. AGD arise from the concept of dynamical control analysis in a long-term optimal control trajectory manner. In regard to long-term optimal trajectory, the optimization problem is segregated into two phases to search for the minimum point \(W^*\). Phase 1 of AGD algorithm incorporates trust region like method by seeking minimum points based on a sequence of boundaries and approximate Newton method in phase 2. The illustration of AGD iteration in solving 2-dimensional problem is shown in Figure 2.

Let the local search region be \(Z_1, Z_2, ..., Z_{N-1}\), the minimum point lies at the boundary of search region. Whereas in the last boundary \(Z_N\), where \(N\) represents the last search boundaries, the minimum point occurs either in the interior or on the boundary of \(Z_N\). Thus, the iteration steps in AGD can be defined as an unit circle
as shown in Figure 3. The circular search region defines the search boundary used to connect iteration steps from \( W_k \) to \( W_{k+1} \). The step length and step direction is computed similar to (6) but via AGD iteration. Based on Figure 3, iteration steps can be computed via the radius of the circular boundary as,

\[
R_k = W_{k+1} - W_k = \alpha_k p_k.
\]  

(13)

Thus, the gradient of next iteration steps \( E(W_{k+1}) \) is minimized subjected to (14) to define the step direction \( p_k \).

\[
||W_{k+1} - W_k||^2 = R_k^2,
\]  

(14)

where \( R_k \) is the radius of the search region at iteration \( k \). Let \( \alpha_k \) be the step length and \( p_k \) be the step direction. With regards to seek for the minimum point in the local search region, Lagrangian expression \( L(W_k) \) is used and written as,
\[ L(W_k) = E(W_{k+1}) - \lambda_k (u_k^T u_k + R_k^2), \]  
\[ \frac{\partial L(W_k)}{\partial u_k} = \frac{\partial E}{\partial W_{k+1}} + 2\lambda_k u_k. \]  
\[ \text{According to optimal control theory, given } \frac{\partial L(W_k)}{\partial u_k} = 0 \text{ and substitute } u_k = \alpha_k p_k \text{ back into (16), the derivatives can be written as,} \]
\[ \frac{\partial E}{W_{k+1}} + 2\lambda_k \alpha_k p_k = 0. \]  
\[ \text{To simplify the equation, let } 2\lambda_k \alpha_k = 1. \text{ Since the next iteration step is still unknown and assuming the iteration steps is relatively small, the gradient of next local search region is approximated as the gradient of current local search region. Thus, the step direction is approximated as,} \]
\[ p_k = -\frac{\partial E}{\partial W_{k+1}} \approx -\frac{\partial E}{\partial W_k}. \]  
\[ \text{By applying Taylor series expansion, it yields,} \]
\[ p_k = -[I + \alpha_k \bar{H}(W_k)]^{-1} g(W_k). \]  
\[ \text{Let } u_k = \alpha_k p_k \text{ and } u_k^T u_k = R_k^2 \text{ referring to the constrain in equation (14), the relative step length of AGD is shown as follows,} \]
\[ \alpha_k^2 = \frac{R_k^2}{\|p_k\|^2} \rightarrow \alpha_k = \frac{R_k}{\|p_k\|}. \]  
\[ \text{Recalling } p_k = \frac{\partial E}{\partial W_k} = g(W_k) \text{ from (18), by letting } \mu_k = \alpha_k^{-1} \text{ yields,} \]
\[ \mu_k = \left(\frac{R_k}{\|p_k\|}\right)^{-1} = \frac{\|g(W_k)\|}{R_k}. \]  
\[ \text{The effect of } \mu \text{ is as follows: (a) for a large } \mu, \text{ it encourages linear convergence of AGD iteration, (b) for a small } \mu, \text{ it encourages the quadratic convergence of Newton method. Newton method requires only a single iteration to achieve convergence for quadratic equation [6]. Unlike LM method, AGD does not introduce any ad-hoc parameters and the Hessian is not restricted to be positive definite. In the best-case scenario, iterations should generate points on the constructed boundaries using trust region like approach and subsequently apply the Newton method at the final local search region to achieve quadratic convergence. Substituting relative step length into (6), the SDAGD formula can be written as,} \]
\[ W_{k+1} = W_k + \kappa \mu_k I + \bar{H}(W_k)]^{-1} g(W_k), \]  
where \( \kappa \) is the global learning rate.
4. Experimental Result. Equation (22) is implemented in MATLAB and compared with other existing optimization techniques in ANN. The network structure is constructed based on a two-layer MLP [15] with 784 input nodes, 700 hidden nodes and 10 output nodes. The squashing function is set to be sigmoid activation function. Weight initialization in the networks is set up using random weights distribution by normalizing the variance with $1/\sqrt{m}$, where $m$ is the number of input. The experiment setup is standardized for all optimization testing, so that all neurons in the network have initially identical weight distribution and empirically improves the rate of convergence. The two-layer MLP is tested with 3 configurations, i.e. (a) SGD algorithm with learning rate, $\eta = 0.005$, (b) SDLM algorithm with learning rate, $\eta = 0.0005$ and damping parameter, $\mu = 0.02$ and (c) SDAGD algorithm with global learning rate, $\kappa = 0.0005$ and the radius of local search region, $R = 1000$. The evaluated dataset is Mixed National Institute of Standards and Technologies (MNIST) [10] database for training and testing. MNIST consists of 60,000 training 10,000 testing hand written digits centered in a $28 \times 28$ dimension image. To have the quick prove of feasibility using SDAGD in neural networks, 5,000 training samples will be fed into the network at one epoch of training session and each training session are repeated for 50 epochs in a stochastic manner. All simulations are conducted on a computer with 3.6GHz Intel Core i7 4790 CPU and 32GB of DDR3 RAM.

The performance were evaluated based on two performance measurement metrics, i.e. Mean Squared Error (MSE) for learning characteristic and Misclassification Rate (MCR) for recognition rate comparison. MSE was computed via (23) while MCR was computed via (24). The results were then compared with two other existing optimization technique, i.e. Stochastic Gradient Descent (SGD) [4] and Stochastic Diagonal Levenberg-Marquardt (SDLM) [10].

\[ MSE = \frac{1}{P \times n} \sum_{i=1}^{P} \sum_{j=1}^{n} (d_{i,j} - x_{r,s})^2, \]  

(23)

where $d_{i,j}$ is the targeted output whereas $x_{r,s}$ is the output.

\[ MCR(\%) = (1 - \frac{\sum TP + \sum TN}{\sum TS}) \times 100\%, \]  

(24)

where TP represents true positive, TN represents true negative and TS represent total number of samples.

| Training Algorithm | Training MCR (%) | Testing MCR (%) | MSE |
|--------------------|------------------|----------------|-----|
| SGD                | 8.22             | 12.14          | 0.40|
| SDLM               | 8.86             | 10.19          | 0.32|
| SDAGD              | 6.46             | 9.40           | 0.21|

In Table 1, the second-order algorithms (SDLM and SDAGD) outperformed the first-order algorithm (SGD) in terms of MSE at 50 epochs. The reason of improvement by the second-order algorithms was due to the adaptation of curvature information in iteration steps construction during optimization which obtained more error surface information. In addition, SDAGD achieved faster roll-off rate and...
saturated at the earlier epoch as compared to SDLM. In terms of MCR, SDAGD achieved the lowest training and testing misclassification rate at 6.46% and 9.4% respectively among all optimization techniques. The outcomes demonstrated the robustness of a two-phase optimization technique by adaptively switching strategies with the control of relative step length.

5. Conclusion. This paper concluded the integration of AGD algorithm into ANN optimization with numerical derivation. The simulation result have empirically proven the proposed algorithm achieved 9.4% of testing misclassification rate compared to other existing optimization techniques. This was due to the adaptation of two-phase system designed to solve optimization problem in neural network via long-term optimal trajectory approach. The relative step length used in the proposed algorithm enabled the ability to adaptively switching strategies from AGD iteration to approximate Newton algorithm. This adaptive optimization technique suggested a better structured training scheme for ANNs and could potentially improve the learning capability and structural simplicity of deeper neural networks. In future, the proposed algorithm will be implemented into deeper neural networks to probe on the performance and robustness with large dataset.

Acknowledgments. This research work is supported by Ministry of Higher Education Malaysia (MOHE) under the Fundamental Research Grant Scheme (FRGS) with the project ID: FRGS/1/2015/TK04/CURTIN/02/1.

REFERENCES

[1] S. Amari, H. Park and K. Fukumizu, Adaptive method of realizing natural gradient learning for multilayer perceptron, Neural Comput., 12 (2000), 436–444.
[2] S. Becker and Y. LeCun, Improving the convergence of backpropagation learning with second order methods, Proc. of the Con. Models Summer School, (1988), 29–37.
[3] Y. Bengio, Learning deep architectures for AI, Foundations and trends in Machine Learning, 2 (2009), 1–127.
[4] L. Bottou, Large-scale machine learning with stochastic gradient descent, Proc. of COMPSTAT, (2010), 177–186.
[5] X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, Aistats, 9 (2010), 249–256.
[6] B. S. Goh, Greatest descent algorithms in unconstrained optimization, J. Optim. Theory Appl., 142 (2009), 275–289.
[7] B. S. Goh, Numerical method in optimization as a multi-stage decision control system, Latest Advances in Systems Science and Computational Intelligence, (2012), 25–30.
[8] Y. LeCun, L. Bottou, G. B. Orr and K. R. Müller, Efficient backprop, Neural Networks: Tricks of the Trade, Springer, (2012), 9–48.
[9] Y. LeCun, Y. Bengio and G. Hinton, Deep learning, Nature, 521 (2015), 436–444.
[10] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE, 86 (1998), 2278–2323.
[11] K. H. Lim, K. P. Seng, L. M. Ang and S. W. Chin, Lyapunov theory-based multilayered neural network, IEEE Transactions on Circuits and Systems II: Express Briefs, 4 (2009), 305–309.
[12] J. Nocedal and S. Wright, Numerical Optimization, 2nd ed., Springer, 2006.
[13] J. R. Shewchuk, An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, Tech. Rep. C. Carnegie-Mellon Univ., 1994.
[14] J. Sohl-Dickstein, B. Poole and S. Ganguli, Fast large-scale optimization by unifying stochastic gradient and quasi-Newton methods, Proc. 31st Int. Conf. Mach. Learn, (2014), 604–612.
[15] D. Stutz, Introduction to Neural Networks, Selected Topics in Human Language Technology and Pattern Recognition WS 12/14, 2014.
[16] H. H. Tan and K. H. Lim and H. G. Harno, Stochastic diagonal approximate greatest descent in neural networks, 2017 International Joint Conference on Neural Networks (IJCNN) (2017), 1895–1898.

Received April 2017; 1st revision December 2017; final revision December 2017.

E-mail address: glkhann@curtin.edu.my
E-mail address: tan.honghui@postgrad.curtin.edu.my
E-mail address: h.g.harno@gmail.com