IOP Conference Series: Materials Science and Engineering

PAPER • **OPEN ACCESS**

Review of second-order optimization techniques in artificial neural networks backpropagation

To cite this article: Hong Hui Tan and King Hann Lim 2019 *IOP Conf. Ser.: Mater. Sci. Eng.* **495** 012003

View the article online for updates and enhancements.

Recent citations

- PReLU and edgeaware filterbased image denoiser using convolutional neural network
Rini Smita Thakur *et al*

- Linearized Implicit Methods Based on a Single-Layer Neural Network: Application to Keller–Segel Models
M. Benzakour Amine

This content was downloaded from IP address 128.179.188.214 on 13/05/2021 at 15:46

IOP Conf. Series: Materials Science and Engineering **495** (2019) 012003 doi:10.1088/1757-899X/495/1/012003

Review of second-order optimization techniques in artificial neural networks backpropagation

Hong Hui Tan and King Hann Lim

Department of Electrical and Computer Engineering, Curtin University Malaysia, CDT 250, 98009, Miri, Sarawak, Malaysia

E-mail: tan.honghui@postgrad.curtin.edu.my, glkhann@curtin.edu.my

Abstract. Second-order optimization technique is the advances of first-order optimization in neural networks. It provides an addition curvature information of an objective function that adaptively estimate the step-length of optimization trajectory in training phase of neural network. With the additional information, it reduces training iteration and achieves fast convergence with less tuning of hyper-parameter. The current improved memory allocation and computing power further motivates machine learning practitioners to revisit the benefits of second-order optimization techniques. This paper covers the review on second-order optimization techniques that involve Hessian calculation for neural network training. It reviews the basic theory of Newton method, quasi-Newton, Gauss-Newton, Levenberg-Marquardt, Approximate Greatest Descent and Hessian-Free optimization. This paper summarizes the feasibility and performance of optimization techniques in deep neural network training. Comments and suggestions are highlighted for second-order optimization techniques in artificial neural network training in term of advantages and limitations.

1. Introduction

Artificial neural network contains numerous weights parameter connected in a deeper layer of network topology. Error signal is measured and back propagated to the network via optimization techniques as shown in Figure 1. Optimization serves as the backbone of artificial neural network learning. There are two categories of learning, i.e. first-order and second-order derivatives learning algorithms. First-order derivatives method uses gradient information to construct the next training iteration whereas second-order derivatives uses Hessian to compute the iteration based on the optimization trajectory. The first-order method relies only on the gradient information and it does not perform as comparable to the second-order method. It requires a series of fine-tuning on the hyperparameter to accommodate for an optimum trajectory across the error surface. Numerous improvements are made using adaptive approach on the first-order derivatives method to overcome the needs of extensive fine-tuning. Adaptive gradient algorithm (Adagrad) [1] obtains adaptive learning rate in optimization algorithm to reduce the needs of extensive hyperparameter fine-tuning. It performs an informative gradient-based learning derived based on the geometry of data of the earlier iterations. Adagrad will provides larger updates to infrequent training input while having smaller update for frequent training input. Adaptive learning rate algorithm (Adadelta) [2] is a per dimension learning rate method for gradient descent algorithm. It is designed to overcome the limitation of Adagrad algorithm which has a continuous decaying learning rate. Adaptive moment estimation (Adam) [3] is proposed


Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

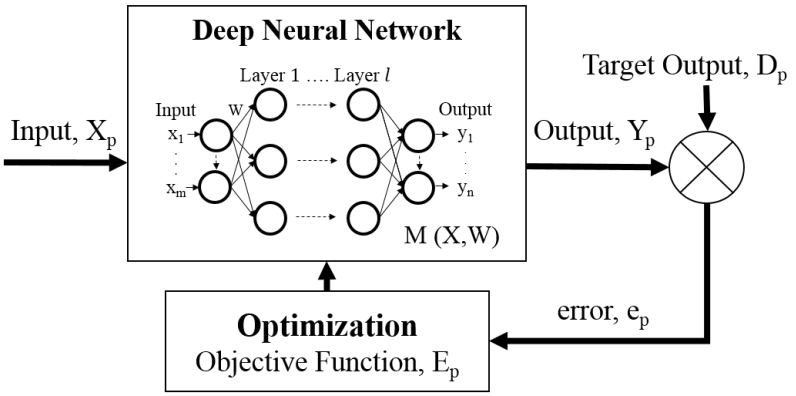

Figure 1. The block diagram of artificial neural network.

to enhance the optimization performance of Adadelta. It is a combination of per-parameter Adagrad and momentum adaptation. Unlike Adadelta, it stores the exponentially decaying average of past deltas just like the per-parameter momentum acceleration. Although most of the first-order derivatives adaptive method shows significant improvement in learning ability, there are still room of improvements to work with the current state-of-the-art optimization techniques. The second-order method is able to achieve better results due to the adaptation of curvature information. Hence, by applying the second-order derivatives in neural network training could adaptively adjusting the step-length throughout the training process of artificial neural networks. The current advances of computing power and memory allocation in the computing hardware has enabled the development of second-order derivative methods in the deep learning neural network optimization. This paper explores the possibility of applying second-order derivatives method into a high-order non-linear artificial neural networks to improve deep leaning neural networks learning ability and structural simplicity. The state-of-the-art of second-order derivatives methods is highlighted for performance evaluation. The advantages and limitations of the existing second-order derivatives methods are summarized in the paper. The Newton method is the first second-order optimization method proposed for neural networks training. It uses full Hessian in training and are prone to computation and memory issues. Quasi-Newton and Gauss-Newton are introduced to counter the drawback of Newton method with truncated Hessian and approximate Hessian respectively. Levenberg-Marquardt method is proposed to solve the indefinite matrix problem with the introduction of regularization parameter to the truncated Hessians. Consequently, approximate greatest descent utilizes control theory in developing a two phase long term and short term optimization approach. The Hessian-free method was then proposed as an alternative second-order optimization technique with the help of conjugate gradient algorithm.

2. Optimization in Artificial Neural Networks

In Figure 1, optimization techniques incorporate the backpropagation step, where error is measured from the targeted and computed output. Optimization in neural network context is the minimization of the objective function, E_p towards the solution, W^* for minimal error value. The iterative updates equation can be written as follows,

$$W_{k+1} = W_k + \alpha_k p_k, \tag{1}$$

where W_{k+1} is the updated weight matrix, k represents the iteration, α_k is the step length and p_k is the step direction. The most commonly used optimization techniques in artificial neural

network training is the gradient descent method. It is a first-order derivatives method. Gradient descent uses the negative gradient to propagate through the error surfaces. Thus, let step length, α_k be the learning rate, η and step direction, p_k be the negative gradient, $-\frac{\partial E}{\partial W} = -\nabla E$. The update equation can be written as,

$$W_{k+1} = W_k - \eta \frac{\partial E}{\partial W}. \tag{2}$$

However, the performance of gradient descent is largely affected by the choice of learning rate, η. An overly large η leads to drastic changes at each iteration and often failed to locate the minimum point, W^*. In contrast, an overly small took too small changes at each iteration and will take more training epochs to reach the minimum point [4]. Hence, it is a common practice to sweep through a few trials of η for hyperparameters fine-tuning. It is also known that ordinary gradient descent does not perform well in error surface in pathological curvature with multiple plateaus and saddle points. The second-order method with local curvature information is able to provide better trajectory across mountainous error surface commonly seen in artificial neural networks.

3. Second-Order Derivatives Methods

Newton method is the main second-order derivatives method but received less popularity due to heavy computation and memory usage. Hence, multiple alternative second-order derivatives methods are proposed throughout the years to accommodate the issues. The second-order derivatives methods provide better training trajectory across the local curvature of the error surface with the help of additional Hessian information. The use of Hessian matrices ease the fine-tuning of hyperparameter by adaptively switching the step size according to the different stages of learning. In the subsection, some second-order derivatives methods i.e. (a) Newton method (b) conjugate gradient; (c) quasi-Newton; (d) Gauss-Newton; (e) Levenberg-Marqaurdt, (f) Approximate greatest descent and (e) Hessian-free method will be covered.

3.1. Newton method

The profound second-order derivatives method is the Newton method [5]. It computes the next update step by incorporating inverse Hessian matrices, $\left(\frac{\partial^2 E}{\partial W^2}\right)^{-1} = H^{-1}$ and yields,

$$W_{k+1} = W_k - \eta \Big(\frac{\partial^2 E}{\partial W^2}\Big)^{-1} \frac{\partial E}{\partial W}. \tag{3}$$

Newton algorithm is very efficient and requires only a single iteration to reach the minimum point [6]. Nevertheless, the computational cost for Newton method is $\mathcal{O}(N^3)$ per iterations and requires large memory to store $N \times N$ matrices. Thus, it is impractical to apply Newton method directly to artificial neural networks with billions of variables for extreme cases. Instinctively, there are plethora of approximate inverse Hessian methods available for practical neural network usage.

3.2. Conjugate gradient method

The first approximate Hessian approach is the conjugate gradient [7]. Conjugate gradient uses Polak and Ribiere [5] method as the conjugate direction, β_k to adaptively alter the descent direction in each iteration. The conjugate direction can be computed using,

$$\beta_k = \frac{(\nabla E_k - \nabla E_{k-1})^T \nabla E_k}{(\nabla E_{k-1})^T \nabla E_{k-1}}. \tag{4}$$

By incorporating the conjugate direction into the update rules yields,

$$W_{k+1} = W_k - \eta(\nabla E_k + \beta_k p_{k-1}) \tag{5}$$

The used of conjugate direction reduces the needs of high computation capability as it is a $\mathcal{O}(N)$ method. This method did not use the actual Hessian directly by manipulating the current update with the update direction from previous iteration. On top of that, conjugate gradient uses a line search method to identify the step length and thus it is only suitable for batch learning.

3.3. Quasi-Newton method

Quasi-Newton [8] uses inverse Hessian estimation to compute each update iteration. There are a few estimation algorithms available but the most well-known is the Broyden-Fletcher-Golfarb-Shanno (BFGS) [4] algorithm. The estimate inverse Hessian, Q for BFGS is computed at each iteration with the following equation,

$$Q_k = \Big(I - \frac{\delta\phi^T}{\phi^T\delta}\Big)Q_{k-1}\Big(I - \frac{\phi\delta^T}{\phi^T\delta}\Big) + \frac{\delta\delta^T}{\phi^T\delta}, \tag{6}$$

where $\delta = W_k - W_{k-1}$ and $\phi = \nabla E_k - \nabla E_{k-1}$. By substituting the estimated inverse Hessian into the update equation yields,

$$W_{k+1} = W_k - \eta Q_k \frac{\partial E}{\partial w}. \tag{7}$$

Quasi-Newton indeed reduces majority of the computation cost with just $\mathcal{O}(N^2)$ computation compared to Newton method. However, it stills suffer from the needs of storing $N \times N$ matrices. There are limited memory version of quasi-Newton method dubbed as L-BFGS [9] developed with the aim of reducing the requires memory space for each iteration. Nevertheless, it is still outclassed by first-order method on deep and wide networks.

3.4. Gauss-Newton method

Gauss-Newton [4] method is another alternative to approximate inverse Hessian. Gauss-Newton uses only the squared Jacobian and ignore the second-term of entirely. The computation of squared Jacobian, or also known as truncated Hessian. Hence, the update equation written as,

$$W_{k+1} = W_k - \eta\Big(\frac{\partial E}{\partial W}^T \frac{\partial E}{\partial W} + \mu I\Big)^{-1} \frac{\partial E}{\partial W}. \tag{8}$$

Although the computation cost is slightly expansive with $\mathcal{O}(N^3)$, it is computed at each iteration and does not requires to store any information for the next iteration. In practical, this method is able to perform even with lower range of graphical processing unit. However, the ordinary form of Gauss-Newton is not able to deal with indefinite Hessian by halting the training altogether before making significant progress.

3.5. Levenberg-Marquardt method

The Levenberg-Marquardt (LM) [10] method was proposed in response to the drawback lies with Gauss-Newton method. The method introduces a regularization parameter, with $\mu > 0$ to contain the problem with indefinite Hessian. The Levenberg-Marquardt method is written as follows,

$$W_{k+1} = W_k - \eta\Big(\frac{\partial E}{\partial W}^T \frac{\partial E}{\partial W} + \mu I\Big)^{-1} \frac{\partial E}{\partial W}. \tag{9}$$

Nevertheless, the addition of regularization parameter is consider as an ad-hoc approach whilst introducing another parameter to be tuned for a promising learning trajectory.

3.6. Approximate Greatest Descent method

Approximate Greatest Descent (AGD) [11, 12, 13] acquired the concept of optimal control analysis for long-term optimal trajectory that really excels in formulating the optimal solution. AGD incorporate trust region method by seeking the minimum points based on sequence of boundaries. Unlike other methods, AGD used the actual objective function to formulate iteration steps instead of using an approximated function. Let the search boundaries be $Z_1, Z_2, ..., Z_N$, where N is the number of search boundaries. The minimum point on the initial iteration excluding the last iteration lies at the search boundary. Whereas in the last boundaries, the minimum point occurs either in the interior or on the boundary of Z_N. Based on AGD, a LM-like method derived from Taylors theorem is obtained as,

$$W_{k+1} = W_k - \eta(H + \mu I)^{-1}\frac{\partial E}{\partial W}, \tag{10}$$

where $\mu = ||\nabla E||/r$ is the relative step length. The relative step length is computed based on the radius, r of the search boundaries Unlike LM method, AGD do not introduce any ad hoc parameter. Besides that, AGD was demonstrated in solving the Rosenbrock function and Powells problem with the convergence moving to optimal point in lesser iterations [6].

3.7. Hessian-free method

Hessian-free [14] method also known as truncated-Newton method works by first computing a scaled down version of Hessian to determine the local curvature and implement conjugate gradient to optimize further. Since full Hessian is too expansive to compute at each iteration, Hessian-free method utilizes a scaled down version of Hessian matrix, $\widehat{H}$ with finite differences at the cost of a single extra gradient evaluation via the identity,

$$\widehat{H} = \lim_{\epsilon \to 0} \frac{\nabla E(w + \epsilon) - \nabla E(w)}{\epsilon}. \tag{11}$$

The Hessian approximation applied by Hessian-free method is ingenious as it only requires matrix-vector products in optimizing the quadratic objectives function. However, it works only with the help of conjugate gradient. The behavior of conjugate gradient will naturally make significant progress in minimization along with the training iterations.

4. Comparison and Analysis for Second-Order Derivatives Methods

There are multiple drawbacks exists in the current optimization techniques. For instance, gradient descent performs optimization via linear model often suffer from local minima issue. Gradient descent also demands for hyperparameter fine-tuning which sometimes requires more training time than involving second-order derivative approach. However, there are still room of improvement for current methods in artificial neural network context. Table 1 summarized the advantages and disadvantages of second-order derivatives methods. Newton method utilizes quadratic models that are suffer from high computing cost and requires to store large Hessian matrix for computation. The ordinary of Newton method is impossible to apply into artificial neural network training due to seemingly large number of parameters for a descent deep neural network. Conjugate gradient applies line-search approach for step length approximation only restrict the application only with batch training. Subsequently, quasi-Newton solves the computing problem with approximate Hessian. However, BFGS method requires gradient information from previous iteration which still prone to suffer from memory issues. Gauss-Newton excluded the need of storing information from previous iteration but are prone to exploding gradient if the matrix is not positive definite. LM method implemented an ad hoc approach by introducing the regularization parameter to contain the indefinite Hessian problem.

The fact the choice of regularization is not properly theorized, LM method still requires some hyperparameter fine-tuning. Hessian-free method proposed an ingenious approach on Hessian estimation but it will only work well with conjugate gradient as the second stage of algorithm. AGD is the latest advancement in numerical optimization for adaptive second-order method. It is derived and inspired by the multistage optimal control system. Hence, it is equipped with a two-phase spherical optimization strategy with relative step length. AGD is free from the vanishing and exploding gradient as the step length is changed according to the state of training. The relative step length is naturally an adaptive optimization approach which works as good as to the first-order optimization approaches. AGD utilizes the gradient information to construct an adaptive step length to deal with different stages of optimization problem that could ease the search for optimal trajectories.

Table 1. Summary of second-order derivative method.

Method	Summary
Newton [5]	● Quadratic convergence only if the searching point is close to the minimum solution. ● Requires high computing cost to compute inverse Hessian at each iteration. ● Requires to stores large Hessian matrix to compute the updates for each iteration.
Conjugate gradient [7]	● Utilizes conjugate direction to replace the needs of costly inverse Hessian computation. ● Requires line-search approach to approximate step size therefore only suitable for batch learning.
Quasi-Newton [8]	● Utilizes BFGS for inverse Hessian estimation to reduce computational cost. ● Requires decent amount of memory to store gradient and update direction information from previous iteration.
Gauss-Newton [4]	● Simplify the Hessian computaiton by using squared Jacobian. ● Does not requires to store information from previous iteration to compute the update direction. ● Struggle with indefinite Hessian and may halt the training without making significant training progress.
Levenberg-Marquardt [10]	● Solves the indefinite Hessian problem with regularization parameter. ● The choice of regularization parameter is heuristic.
Hessian-free [14]	● Compute approximate Hessian with finite differences of gradient evaluation. ● Requires conjugate gradient to complete the optimization.
Approximate Greatest Descent [13]	● Utilizes two-phase spherical optimization strategy to compute trajectory based on control theory. ● The step size is controlled by the relative step length derived based on the constructed spherical regions. ● It is an adaptive method.

5. Conclusion

The learning curve of artificial neural networks is always one of the well-studied area in machine learning. The main implication from the review is to identify the appropriate Hessian approximation approach which is suitable for artificial neural network training. Conjugate gradient and quasi-Newton both require to store information from previous iteration. Alternatively, Hessian-free method on Hessian approximation does not work as efficient without conjugate gradient algorithm. Although Gauss-Newton and Levenberg-Marquardt method requires more computation power per iterations, it can be solved with the help of parallel processing. Levenberg-Marquardt also uses less memory compared to other Hessian approximation approaches which enables the algorithm to run on lower-ends graphical processing unit. Nevertheless, Approximate greatest descent utilizes an adaptive two phase methods that which had proven to provide better trajectory towards the minimum.

Acknowledgments

This work is supported by Fundamental Research Grant Scheme (FRGS) from Ministry of Higher Education Malaysia under FRGS/1/2015/TK04/CURTIN/02/1.

References

[1] Duchi J, Hazan E and Singer Y 2011 *J. Mach. Learn. Res.* **12** 2121–2159 ISSN 1532-4435 URL http://dl.acm.org/citation.cfm?id=1953048.2021068
[2] Zeiler M D 2012 *CoRR* **abs/1212.5701** (*Preprint* 1212.5701) URL http://arxiv.org/abs/1212.5701
[3] Kingma D P and Ba J 2014 *CoRR* **abs/1412.6980** (*Preprint* 1412.6980) URL http://arxiv.org/abs/1412.6980
[4] LeCun Y, Bottou L, Orr G B and Müller K R 1998 *Efficient BackProp* (Berlin, Heidelberg: Springer Berlin Heidelberg) pp 9–50 ISBN 978-3-540-49430-0
[5] Nocedal J and Wright S 2006 *Numerical Optimization* Springer Series in Operations Research and Financial Engineering (Springer New York) ISBN 9780387303031
[6] Goh B S 2009 *Journal of Optimization Theory and Applications* **142** 275–289 ISSN 1573-2878
[7] Møller M F 1993 *Neural Networks* **6** 525 – 533 ISSN 0893-6080 URL http://www.sciencedirect.com/science/article/pii/S0893608005800565
[8] Sohl-Dickstein J, Poole B and Ganguli S 2013 *CoRR* **abs/1311.2115** (*Preprint* 1311.2115) URL http://arxiv.org/abs/1311.2115
[9] Xiao Y, Wei Z and Wang Z 2008 *Computers & Mathematics with Applications* **56** 1001 – 1009 ISSN 0898-1221 URL http://www.sciencedirect.com/science/article/pii/S0898122108001028
[10] Wilamowski B M and Yu H 2010 *IEEE Transactions on Neural Networks* **21** 930–937 ISSN 1045-9227
[11] Goh B S 1995 *American Control Conference, Proceedings of the 1995* vol 3 pp 2071–2074 vol.3
[12] Goh B S 2011 *Journal of Optimization Theory and Applications* **148** 505–527 ISSN 1573-2878
[13] Goh B S 2012 *Latest Advances in Systems Science and Computational Intelligence* 25–30
[14] Martens J 2010 *Proceedings of the 27th International Conference on International Conference on Machine Learning* 735–742 URL http://dl.acm.org/citation.cfm?id=3104322.3104416