International Journal of Engineering & Technology, 7 (4.36) (2018) 1194-1198

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Early Stopping Criteria for Levenberg-Marquardt Based Neural Network Training Optimization

[1]Azizah Suliman, [2]Batyrkhan Omarov

[1]*College of Computer Science & Information Technology, Universiti Tenaga Nasional, Kuala Lumpur, Malaysia*
[2]*International Information Technologies University, Almaty, Kazakhstan*
**Corresponding author E-mail: azizah@uniten.edu.my*

Abstract

In this research we train a direct distributed neural network using Levenberg-Marquardt algorithm. In order to prevent overtraining, we proposed correctly recognized image percentage based on early stop condition and conduct the experiments with different stop thresholds for image classification problem. Experiment results show that the best early stop condition is 93% and other increase in stop threshold can lead to decrease in the quality of the neural network. The correct choice of early stop condition can prevent overtraining which led to the training of a neural network with considerable number of hidden neurons.

Keywords: *Early Stop Condition, Levenberg-Marquardt Method, Neural Network, Overtraining.*

1. Introduction

Analysis of literatures shows that neural networks are effectively used in crucial applications such as pattern recognition [1], image classification [2, 8], speech recognition, natural language processing [1 - 3]. There are several key concepts that have been instrumental in the success of learning the neural networks, including gradient descent, parallel implementations, convolution neural networks, supervised and unsupervised pre-learning [3, 9].

Artificial neural network(ANN) methods are widely used in classification problems. Classification problem is a task to include the sample to one of several disjoint sets. When solving classification problems, ANN should include the existing object characteristics (observable data) to one or more specific classes.

One of the main challenges in implementing ANN is the significant amount of time needed in the training phase especially when solving complex problems. Depending on the growth of a number of hidden layers and neurons, the required time for ANN learning process and new instance assessment time, grows by leaps and bounds.

On the other hand, the rate of successful classification depends on the growth of a number of hidden layer and neurons. So, in general, the more training instances the network is guaranteed, the more effective result can be achieved. Ideally, it is very important to carry out training with a considerable number of neurons in the hidden layer and with a large number of training examples, but with a relatively low training time. So the challenge remains in improving the ANN by improving the training algorithms, selecting the best network topology, determining the number of hidden layers neurons, interpretation of weighting coefficients and bias, and their evaluation of optimality, etc.

The main goal of this research is developing an effective neural network training algorithm with keeping the hidden layers as minimum, without reducing recognition and classification accuracy. A significant improvement in performance can be achieved using second-order algorithms, such as Newton's algorithms, the conjugate gradient algorithm or the Levenberg-Marquardt (LM) algorithm. The Levenberg Marquardt algorithm was chosen as the training algorithm to be worked on in this research work because it gives a higher accuracy as compared to the other gradient algorithms [1].

It is believed that, despite the heuristic nature, the LM algorithm makes it possible to achieve the smallest error of the neural network, and, often with the least time. The algorithm provides an acceptable compromise between the convergence rate inherent in Newton's algorithms and the stability inherent in the gradient descent. The algorithm successfully combines the method of steepest descent (ie, minimization along the gradient) and Newton's method (that is, using a quadratic model to accelerate the search for a minimum of the function). LM provides fast convergence and regularization effect. It provides regularization to stabilize the ill-condition cases during training.

The paper is devoted to the investigation and improvement of one of the most effective algorithms for learning multi-layer perceptions - the Levenberg-Marquardt algorithm, to avoid of overtraining and get high classification rate with considerable number of hidden neurons.

This paper is organized as follows: Section 2 briefly introduces the Levenberg-Marquardt algorithm by detailing its associated mathematical model. Section 3 introduces the direct distributed neural network so readers can get a clearer picture of where the training algorithm is associated to the network and how it would later be manipulated to improve training. Section 4 discusses the recommended improvement, the early stop condition. The implementation of the algorithm is explained. Section 5 presents the experiment results to

demonstrate the improvement achieved. The subsection also describes the database used in the experiments. Section 6 concludes the paper and present its contributions.

2. Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm is one of the applications of the Newton's optimization strategy. The main expression of Newton's methods is the expression:

$$p_k = -[H(w_k)]^{-1} g(w_k), \quad (1)$$

where, p_k - direction that guarantees the achievement of the minimum of the object function for current step, $g(w_k)$ - gradient value at the point of the last solution of w_k, $H(w_k)$ - Hessian value at the point of the last solution of w_k.

When using the Levenberg-Marquardt algorithm, the exact value of the Hessian $H(w)$ in (1) is replaced with an approximate value g(w), that is calculated on the basis of information contained in the gradient, taking into account a certain regularization factor. To describe this method, we represent the objective function in a form corresponding to the existence of a single training sample,

$$E(w) = \frac{1}{2}\sum_{i=1}^{M}[e_i(w)]^2, \quad (2)$$

where, $e_i = [y_i(w) - d_i]$. When using notation

$$e(w) = \begin{bmatrix} e_1(w) \\ e_1(w) \\ \dots \\ e_1(w) \end{bmatrix}, \quad J(\theta) = \begin{bmatrix} \frac{\partial(e_1)}{\partial w_1} & \frac{\partial(e_1)}{\partial w_2} & \dots & \frac{\partial(e_1)}{\partial w_n} \\ \frac{\partial(e_2)}{\partial w_1} & \frac{\partial(e_2)}{\partial w_2} & \dots & \frac{\partial(e_2)}{\partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial(e_n)}{\partial w_1} & \frac{\partial(e_n)}{\partial w_2} & \dots & \frac{\partial(e_n)}{\partial w_n} \end{bmatrix} \quad (3)$$

The gradient vector and the approximated Hessian matrix corresponding to the objective function (2) are defined as:

$$g(w) = [J(w)]^T e(w) \quad (4)$$

$$G(w) = [J(w)]^T J(w) + R(w) \quad (5)$$

where, $R(w)$ components of Hessian ($H(w)$), that containing higher derivatives concerning w.

The essence of the Levenberg-Marquardt approach is to approximate $R(w)$ using a regularization factor vl in which a variable v called the Levenberg-Marquardt parameter is a scalar quantity that changes during optimization. Thus, the approximated Hessian matrix at the *k*-th step of the algorithm takes the following form:

$$G(w_k) = [J(w_k)]^T J(w_k) + v_k l \quad (6)$$

At the beginning of the learning process, when the actual value is still far from the desired solution, a parameter value much greater than the eigenvalue of the matrix $[J(w_k)]^T J(w_k)$. In this case, the Hessian is actually replaced by a regularization factor:

$$G(w_k) \cong v_k l \quad (7)$$

and the direction of minimization is chosen by the method of steepest descent:

$$p_k = -\frac{g(w_k)}{v_k} \quad (8)$$

As the error decreases and the approximation to the desired solution decreases, the parameter v_k decreases $[J(w)]^T J(w)$ in the equation (5) becomes more important. The efficiency of the algorithm is influenced by a competent selection of the value v_k. Too large initial value of v_k with the progress of optimization should decrease down to zero at achievement of the actual decision close to the required one. There are various ways of selecting this value, but we consider only one original technique that proposed by Marquardt:

$$\Delta\theta = (J^T J + \lambda I)^{-1} J^T F(\theta) \quad (9)$$

where *I* – is the identity matrix. It may be, as indicated in [4], that the curvature of the surface defined by the discrepancy can be "not identical" in all directions. For example, if there is a long and narrow trough on the surface of the discrepancy, the gradient component in the direction pointing along the base of the depression, it is very small and the gradient component along the trough walls is quite large. This leads to movement towards the walls of the trough, while the need to travel long distances along the base and a small - along its walls. To avoid this, in [5] it proposed to replace the identity matrix with a diagonal matrix of the approximate Hessian matrix. Then the formula (9) will take the next form of:

$$\Delta\theta = (J^T J + \lambda diag(J^T J))^{-1} J^T F(\theta) \quad (10)$$

This rule is used as follows: If the residual is reduces for the current iteration, (which means that the assumption of quadratic works), we decrease (usually 10 times) to reduce the effect of the gradient descent. On the other hand, if the residual increases, we must follow the direction of the gradient, and to increase (to the same amount).

3. Direct Distributed Neural Network

As a clarifying of the mathematical model of pattern recognition, direct distributed neural network can be represented as a vector function of vector argument [7]:

$$Y = Y(X, \theta) \quad (10)$$

Here $X = (x_1, ..., x_K)$ - input data, $\theta = (\theta_1, ..., \theta_M)$ – weights of a network, $Y = (y_1, ..., y_p)$ – network output. Then the network error for one period will be expressed by the formula

$$F(\theta)=\frac{1}{2}\sum_{i=1}^{L}\sum_{j=1}^{p}(y_{ij}-d_{ij})^2 \quad (11)$$

here d_{ij} - desired output of j-th output neuron for i-th element of a training set, L - number of elements of the training sample.

Let, $E=(e_{11,...,}e_{1p},e_{L1},...,e_{Lp})^T$ - discrepancy vector for a neural network, here $e_{ij}=y_{ij}-d_{ij}$. Then the formula (11) can be rewritten as:

$$F(\theta)=E^TE \quad (11)$$

and Jacobian matrix has the form:

$$J=\begin{bmatrix}\tilde{J}_1\\ \vdots\\ \tilde{J}_L\end{bmatrix},\quad \tilde{J}_i=\begin{bmatrix}\frac{\partial e_{i1}}{\partial\theta_1} & \frac{\partial e_{i1}}{\partial\theta_2} & \cdots & \frac{\partial e_{i1}}{\partial\theta_M}\\ \frac{\partial e_{i2}}{\partial\theta_1} & \frac{\partial e_{i2}}{\partial\theta_2} & \cdots & \frac{\partial e_{i2}}{\partial\theta_M}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial e_{ip}}{\partial\theta_1} & \frac{\partial e_{ip}}{\partial\theta_2} & \cdots & \frac{\partial e_{ip}}{\partial\theta_M}\end{bmatrix} \quad (12)$$

As described above, the increment of the weights of the neural network should be sought to the form of solutions of the equation (13).

$$\left(J^TJ+\lambda diag(J^TJ)\right)\Delta\theta=J^TE$$

Within the framework of the research we consider a neural network training which consists of three layers: input layer (n neurons), hidden layer (m neurons) and output layer (p neurons). In addition, each layer is present neuron, called threshold neurons that the output of which, in contrast to the usual neuron is always equal to one. The introduction of such neuron makes the learning algorithm of the neural network more flexible. This statement can be illustrated by a simple example. Let, a neural network consists of one neuron with a single input. For a neural network with one hidden layer formula (10) takes the form:

$$Y=Y(X,\theta)=\sigma\left(W^{(2)}\sigma\left(W^{(1)}X+B^{(1)}\right)+B^{(2)}\right) \quad (14)$$

Here $W^{(1)}$ is weight matrix of the hidden layer neurons, $W^{(2)}$ is weight matrix of output layer neurons, $B^{(1)}$ is weights of the hidden layer threshold neurons. $B^{(2)}$ is weights of the output layer threshold neurons, σ - activation function of a neuron.

Then, elements of Jacobian matrix will be the following: If θ is the weight of hidden layer neuron, that is $\theta_r\equiv w^{(1)}_{i'j'}$ then

$$\frac{\partial e_i}{\partial\theta_r}\equiv\frac{\partial e_i}{\partial w^{(1)}_{i'j'}}=w^{(2)}_{i'j'}\sigma'\left(\sigma\left(\sum_{j=1}^{n}w^{(1)}_{i'j}x_j\right)\right)x_{j'}\sigma'\left(\sum_{k=1}^{m}w^{(2)}_{ik}\bar{x}_k\right) \quad (15)$$

Here, $\bar{x}_k$ - output of k-th neuron of the hidden layer, $\sigma'(\cdot)$ - value of the derivative of activation function at the point.

If θ is the weight of output layer neuron, i.e. $\theta\equiv w^{(2)}_{i'j'}$ then

$$\frac{\partial e_i}{\partial\theta_r}\equiv\frac{\partial e_i}{\partial w^{(2)}_{i'j'}}=\begin{cases}\sigma'\left(\sum_{k=1}^{m}w^{(2)}_{ik}\bar{x}_k\right)\bar{x}_{j',i=i'}\\ 0,i\neq i'\end{cases} \quad (16)$$

If θ is the weight of hidden layer neuron, i.e. $\theta\equiv b^{(1)}_{i'}$, then

$$\frac{\partial e_i}{\partial\theta_r}\equiv\frac{\partial e_i}{\partial b^{(1)}_{i'}}=\partial w^{(2)}_{i'j'}\sigma'\left(\sigma\left(\sum_{j=1}^{n}w^{(1)}_{i'j}x_j\right)\right)\sigma\left(\sum_{k=1}^{m}w^{2}_{ik}\bar{x}_k\right) \quad (17)$$

If θ is weight of threshold neuron of the output layer, i.e. $\theta\equiv b^{(2)}_{i'}$, then

$$\frac{\partial e_i}{\partial\theta_r}\equiv\frac{\partial e_i}{\partial b^{(2)}_{i'}}=\begin{cases}\sigma'\left(\sum_{k=1}^{m}w^{(2)}_{ik}\bar{x}_k\right),j',i=i'\\ 0,i\neq i'\end{cases} \quad (18)$$

Thus, the neural network learning algorithm based on the LM algorithm will be as follows:

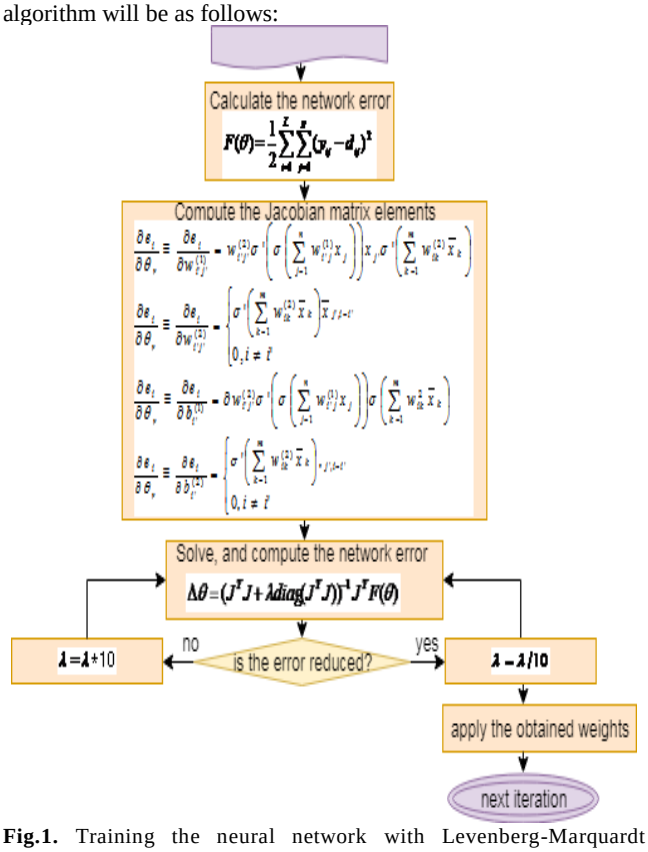


Fig.1. Training the neural network with Levenberg-Marquardt algorithm

4. Early Stop Condition

One of the most important moments in the training of neural networks is the selection of stop criteria for the training and evaluation of their effectiveness. To avoid losing the generalization properties and to reduce the number of epochs we

used so-called "early stop method". Its essence lies in the fact that the training set is divided into two types: the actual training set used to train and test set used to test the trained network. In the training process the neural network is constantly being tested using a test set. As soon as early stop condition is achieved, training stops.

Classical early stop condition is criterion of "no increase" error on the test set. Its essence is that the training is stopped as soon as the error on the test set will start to increase. This approach has a significant drawback. Neural network trained by an early stop to such criteria may have too many errors and, therefore, be of little use for practical calculations. As a rule, during training the neural network, the error on the test set is not monotonically decreasing. As a result, if we take as a stop criterion a simple increase in errors on a test set, training can stop when the neural network is not enough trained.

In [4] proposes three early stop criterion. Let $E_{tr}(t)$ is error of the neural network training set for the epoch t, $E_{va}(t)$ – the error on the test set (or a validation error) for the epoch t, $E_{opt}(t) = \min_{t'<t} E_{va}(t')$. The first criterion: Training must be stopped, when the loss of generalization exceeds a certain threshold.

$$GL(t) = 100\left(\frac{E_{va}(t)}{E_{opt}(t)} - 1\right) \tag{19}$$

However, during the growth of the generalization loss, neural network can overcome this, if the training error is reduced quickly enough. This speed can be estimated using learning progress for k epochs that defined by formula:

$$P_k(t) = 1000\left(\frac{\sum_{t-k+1}^{t} E_{tr}(t')}{k \min_{t'=t-k+1,t} E_{tr}(t')} - 1\right) \tag{20}$$

Second stop criteria may be the ratio of the loss to the generalization of training progress. Training must be stopped when this ratio exceeds a certain threshold:

$$PQ_k(t) = \frac{GL(t)}{P_k(t)} \tag{21}$$

Next, the training should stop after validation error increases for several epochs.

In our research, as a stop criterion we formulated the following rule: Training should be stopped when the error on the test set has decreased to a certain value. Recognition rate of the test set element selected as the error on the test set. Then the stop criteria can be expressed by the formula (22):

$$E_{va}(t) \le \mu, \quad \mu \in [0,100] \tag{22}$$

With this choice of stop criterion is important to choose the right value of μ. Too high a value can lead to a loss of generalization because of the retraining, and too little - to the "half-taught" neural network, of little use for practical tasks.

5. Experiment Results

To test the approach, object classification problem (human, car, and other objects) has been applied. To test the problem the data set was divided into two parts, a training set and a test set. The training set data were 300 instances of each species, and in the test were about 200 instances data.

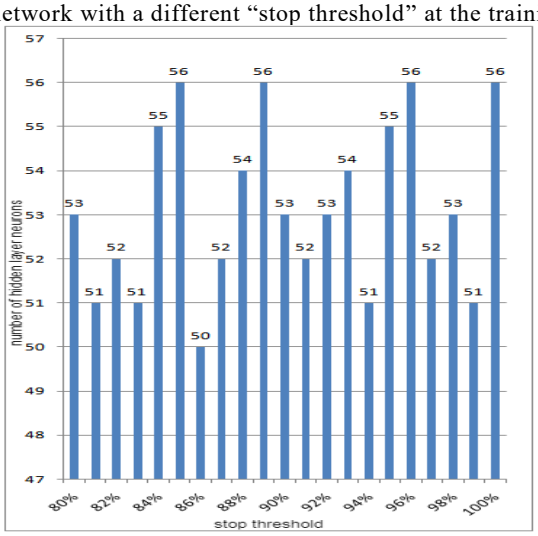
5.1. Data

The main purpose of the problem is to identify human, car from the images or video. Our data set consists of 1,500 instances, 500 of them are car images, 500 are human images, and the last 500 instances are other objects [6].

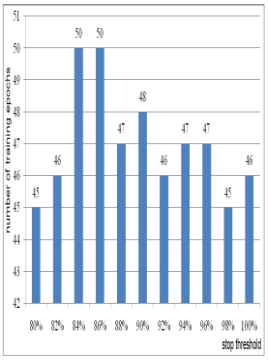
Fig. 2. Human, car, and other objects example

5.2. Results

In this calculation, we performed neural network training with "stop threshold" variable, to determine its best value. Stop threshold ranged in 80-100% diapason with 1% increment. As soon as the number of correctly recognized items exceeded the "stop threshold" the test set training stopped. After training, each neural network was tested on the same test together. The criteria for selecting the stop threshold, as is the case with the training sample was a minimum number of epoch of training and the best recognition accuracy. Figure 3a shows a comparison of the number hidden layer neurons and Figure 3b shows a comparison of the number of epochs of training a neural network with a different "stop threshold" at the training set.



(a) (**b**)



(b)

Fig. 3. Experiment results to determine the best stop threshold value

As can be seen from the figure, the change in stop threshold has practically no effect on the rate of neural network training, measured in the number of training epoch. However, changing the “stop threshold” affects the quality of the resulting neural network. Figure 4 illustrates a graph indicating the number of the recognized test cases from the test set, trained with different “stop threshold”. From the graph, we can conclude, that quality of the resulting neural network depends on the size of the “stop threshold”. However, maximization of the stop threshold does not lead to maximization of the quality of the neural network. Neural network trained with a “stop threshold” gave 93% best results where its graph lie above the other lines with the other stop thresholds. Further increasing of “stop threshold” reduces the quality of the neural network recognition. This can be explained by the fact that an excessive increase in the threshold of training leads to the fact that the neural network is "forced" to spend the training epoch to unjustified minimize errors on the test set, resulting in unnecessary iterations that the neural network can "forget" about the previously presented samples.

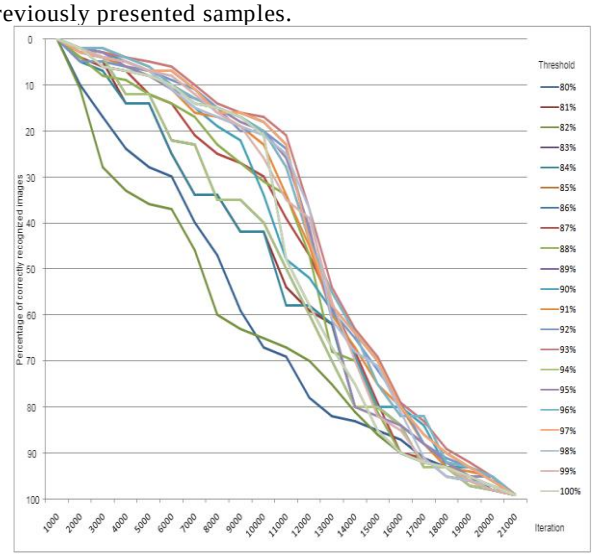


Fig. 4. The number of the recognized test cases from the test set, trained with different “stop threshold”

6. Conclusion

In this research, we considered Levenberg-Marquardt method as a neural network training algorithm for image classification problem. Based on the Levenberg-Marquardt method with early stop condition, direct distributed neural network was constructed. In order to determine the effectiveness of early stop condition, several experiments with different stop threshold were conducted. The most preferred value of hidden layer neurons and value of early stop threshold were determined. Impact of the number of hidden layer for the neural network, to the performance of the program complex was explored. The obtained results gave better results comparing the corresponding figures in the research of other authors. The proposed method performs better in classification task and also maintains a good trade-off between sensitivity and specificity. The proposed method is also computationally cost effective. Therefore, the proposed method can be a useful tool for classification.

References

[1] Omarov, B., Suliman, A., Kushibar, K. Face recognition using artificial neural networks in parallel architecture. Journal of Theoretical and Applied Information Technology 91 (2), pp. 238-248. (**2016**). Islamabad

[2] Omarov, B., Suliman, A., Tsoy, A. Parallel backpropagation neural network training for face recognition. Far East Journal of Electronics and Communications. Volume 16, Issue 4, December 2016, Pages 801-808. (**2016**)

[3] A. Altayeva, B. Omarov, H.C. Jeong, Y.I. Cho. Multi-step face recognition for improving face detection and recognition rate. Far East Journal of Electronics and Communications 16(3), pp. 471-491, **2016**

[4] Lutz P. 1998. Early Stopping-But When? Neural Networks: Tricks of the Trade. London, UK: Springer-Verlag

[5] Marquardt D. **1963**. An Algorithm for Least-Squares Estimation of Nonlinear Parameters SIAM Journal on Applied Mathematics. T. 11. № 2. C. 431-441

[6] Pinz A. **2016**. Human, car, other object database, Electronic resource, http://cvrg.iyte.edu.tr/datasets.htm.

[7] Xu J., Ho D.W.C., Zheng Y. **2004**. A Constructive Algorithm for Feedforward Neural Networks Control Conference. Shanghai: Inst, of Syst. Sei., East China Normal Univ., C. 659-664.

[8] Sattar, M.A., Achanta, S. “Development and validation of a simple method for simultaneous estimation of memantine and donepezil in pharmaceutical dosage forms by using RP-HPLC”, (2018) International Journal of Pharmaceutical Research, 10 (2), pp. 155-166.

[9] V. Franc and J. Cech, Learning CNNs for Face Recognition from Weakly Annotated Images, 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), Washington, DC, 2017, pp. 933-940. doi: 10.1109/FG.2017.115, **2017**