GAUSS-NEWTON APPROXIMATION TO BAYESIAN LEARNING

F. Dan Foresee* and Martin T. Hagan**

*Lucent Technologies
Oklahoma City, OK

**School of Electrical and Computer Engineering
Oklahoma State University
Stillwater, OK

email: fdf@lucent.com, mhagan@master.ceat.okstate.edu

Abstract

This paper describes the application of Bayesian regularization to the training of feedforward neural networks. A Gauss-Newton approximation to the Hessian matrix, which can be conveniently implemented within the framework of the Levenberg-Marquardt algorithm, is used to reduce the computational overhead. The resulting algorithm is demonstrated on a simple test problem and is then applied to three practical problems. The results demonstrate that the algorithm produces networks which have excellent generalization capabilities.

1. Introduction

The results described in this paper apply to multi-layer feedforward neural networks which are used for nonlinear regression. The networks are trained using supervised learning, with a training set of inputs and targets in the form $\{p_1, t_1\}, \{p_2, t_2\}, \ldots, \{p_n, t_n\}$. We assume that the targets are generated by $t_i = g(p_i) + \varepsilon_i$ where $g(p_i)$ is an unknown function and ε_i is independent Gaussian noise. The initial objective of the training process will be to minimize the sum of squared errors:

$$E_D = \sum_{i=1}^{n} (t_i - a_i)^2 \qquad (1)$$

where a_i represents the neural network response. This performance index will later be modified to improve network generalization.

The goal of neural network training is to produce a network which produces small errors on the training set, but which will also respond properly to novel inputs. When a network is able to perform as well on novel inputs as on training set inputs, we say that the network generalizes well. Our objective in this paper is to discuss a training algorithm which consistently produces networks with good generalization. This method for improving generalization constrains the size of the network weights and is referred to as regularization [1]. The idea is that the true underlying function is assumed to have a degree of smoothness. When the weights in a network are kept small, the network response will be smooth. With regularization, any modestly oversized network should be able to sufficiently represent the true function.

In the following section we will review regularization techniques. We will then apply David MacKay's Bayesian techniques to optimize regularization [1]. The optimal regularization technique requires the computation of the Hessian matrix. To minimize the computational overhead, we propose using a Gauss-Newton approximation to the Hessian matrix. This approximation is readily available when using the Levenberg-Marquardt algorithm for network training [2], [3]. Then, in the next section we will apply this new approximation to Bayesian regularization to a diverse set of three real world problems.

2. Regularization

Typically, training aims to reduce the sum of squared errors $F = E_D$. However, regularization adds an additional term; the objective function becomes $F = \beta E_D + \alpha E_W$, where E_W is the sum of squares of the network weights, and α and β are objective function parameters. The relative size of the objective function parameters dictates the emphasis for training. If $\alpha << \beta$, then the training algorithm will drive the errors smaller. If $\alpha >> \beta$, training will emphasize weight size reduction at the expense of network errors, thus producing a smoother network response.

The main problem with implementing regularization is setting the correct values for the objective function parameters. David MacKay [1] has done extensive work on the application of Bayes' rule to neural network training and to optimizing regularization. In the following we will show his main results as they apply to our problem.

In the Bayesian framework the weights of the network are considered random variables. After the data is taken, the density function for the weights can be updated according to Bayes' rule:

$$P(\mathbf{w} \mid D, \alpha, \beta, M) = \frac{P(D \mid \mathbf{w}, \beta, M) P(\mathbf{w} \mid \alpha, M)}{P(D \mid \alpha, \beta, M)} \qquad (2)$$

where D represents the data set, M is the particular neural network model used, and $\mathbf{w}$ is the vector of network weights. $P(\mathbf{w} \mid \alpha, M)$ is the prior density, which represents our knowledge of the weights before any data is collected. $P(D \mid \mathbf{w}, \beta, M)$ is the likelihood function, which is the probability of the data occurring, given the weights $\mathbf{w}$. $P(D \mid \alpha, \beta, M)$ is a normalization factor, which guarantees that the total probability is 1.

If we assume that the noise in the training set data is Gaussian and that the prior distribution for the weights is Gaussian, the probability densities can be written

$$P(D \mid \mathbf{W}, \beta, M) = \frac{1}{Z_D(\beta)} \exp(-\beta E_D) \text{ and}$$

$$P(\mathbf{w} \mid \alpha, M) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W), \qquad (3)$$

where $Z_D(\beta) = (\pi / \beta)^{n/2}$ and $Z_W(\alpha) = (\pi / \alpha)^{N/2}$. If we substitute these probabilities into Eq. (2), we obtain

$$\begin{aligned} P(\mathbf{w} \mid D, \alpha, \beta, M) &= \frac{\frac{1}{Z_W(\alpha)} \frac{1}{Z_D(\beta)} \exp(-(\beta E_D + \alpha E_W))}{\text{Normalization Factor}} \\ &= \frac{1}{Z_F(\alpha, \beta)} \exp(-F(\mathbf{w})) \end{aligned} \qquad (4)$$

In this Bayesian framework, the optimal weights should maximize the posterior probability $P(\mathbf{w} \mid D, \alpha, \beta, M)$. Maximizing the posterior probability is equivalent to minimizing the regularized objective function $F = \beta E_D + \alpha E_W$.

Optimizing the Regularization Parameters

Now we consider the application of Bayes' rule to optimizing the objective function parameters α and β. Here, we have

$$P(\alpha, \beta \mid D, M) = \frac{P(D \mid \alpha, \beta, M) P(\alpha, \beta \mid M)}{P(D \mid M)} \qquad (5)$$

If we assume a uniform prior density $P(\alpha, \beta \mid M)$ for the regularization parameters α and β, then maximizing the posterior is achieved by maximizing the likelihood function $P(D \mid \alpha, \beta, M)$. However, note that this likelihood function is the normalization factor for Eq. (2). Since all probabilities have a Gaussian form, we know the form for the posterior density of Eq. (2). It is show in Eq. (4). Now we can solve Eq. (2) for the normalization factor.

$$\begin{aligned} P(D \mid \alpha, \beta, M) &= \frac{P(D \mid \mathbf{w}, \beta, M) P(\mathbf{w} \mid \alpha, M)}{P(\mathbf{w} \mid D, \alpha, \beta, M)} \\ &= \frac{\left[\frac{1}{Z_D(\beta)} \exp(-\beta E_D)\right]\left[\frac{1}{Z_W(\alpha)} \exp(-\alpha E_W)\right]}{\frac{1}{Z_F(\alpha, \beta)} \exp(-F(\mathbf{w}))} \\ &= \frac{Z_F(\alpha, \beta)}{Z_D(\beta) Z_W(\alpha)} \cdot \frac{\exp(-\beta E_D - \alpha E_W)}{\exp(-F(\mathbf{w}))} = \frac{Z_F(\alpha, \beta)}{Z_D(\beta) Z_W(\alpha)} \end{aligned} \qquad (6)$$

Note that we know the constants $Z_D(\beta)$ and $Z_W(\alpha)$ from Eq. (3). The only part we do not know is $Z_F(\alpha, \beta)$. However, we can estimate it by Taylor series expansion. Since the objective function has the shape of a quadratic in a small area surrounding a minimum point, we can expand $F(\mathbf{w})$ around the minimum point of the posterior density $\mathbf{w}^{\mathrm{MP}}$, where the gradient is zero. Solving for the normalizing constant yields

$$Z_F \approx (2\pi)^{N/2} (\det((\mathbf{H}^{\mathrm{MP}})^{-1}))^{1/2} \exp(-F(\mathbf{w}^{\mathrm{MP}})) \qquad (7)$$

where $\mathbf{H} = \beta \nabla^2 E_D + \alpha \nabla^2 E_W$ is the Hessian matrix of the objective function. Placing this result into Eq. (6), we can solve for the optimal values for α and β at the minimum point. We do this by taking the derivative with respect to each of the log of Eq. (6) and set them equal to zero. This yields

$$\alpha^{\mathrm{MP}} = \frac{\gamma}{2E_W(\mathbf{w}^{\mathrm{MP}})} \text{ and } \beta^{\mathrm{MP}} = \frac{n - \gamma}{2E_D(\mathbf{w}^{\mathrm{MP}})}, \qquad (8)$$

where $\gamma = N - 2\alpha^{\mathrm{MP}} \mathrm{tr}(\mathbf{H}^{\mathrm{MP}})^{-1}$ is called the effective number of parameters, and N is the total number of parameters in the network. The parameter γ is a measure of how many parameters in the neural network are effectively used in reducing the error function. It can range from zero to N.

Gauss-Newton Approximation to the Hessian

The Bayesian optimization of the regularization parameters requires the computation of the Hessian matrix of $F(\mathbf{w})$ at the minimum point $\mathbf{w}^{MP}$. We propose using the Gauss-Newton approximation to Hessian matrix, which is readily available if the Levenberg-Marquardt optimization algorithm is used to locate the minimum point ([2]-[4]). The additional computation required for optimization of the regularization is minimal.

Here are the steps required for Bayesian optimization of the regularization parameters, with the Gauss-Newton approximation to Hessian matrix:

0. Initialize α, β and the weights. We choose to set $\alpha = 0$ and $\beta = 1$ and use the Nguyen-Widrow method of initializing the weighs [5]. After the first training step, the objective function parameters will recover from the initial setting.
1. Take one step of the Levenberg-Marquardt algorithm to minimize the objective function $F(\mathbf{w}) = \beta E_D + \alpha E_W$.
2. Compute the effective number of parameters $\gamma = N - 2\alpha \mathrm{tr}(\mathbf{H})^{-1}$ making use of the Gauss-Newton approximation to the Hessian available in the Levenberg-Marquardt training algorithm: $\mathbf{H} = \nabla^2 F(\mathbf{w}) \approx 2\beta \mathbf{J}^T\mathbf{J} + 2\alpha \mathbf{I}_N$, where $\mathbf{J}$ is the Jacobian matrix of the training set errors [2].
3. Compute new estimates for the objective function parameters $\alpha = \dfrac{\gamma}{2E_W(\mathbf{w})}$ and $\beta = \dfrac{n-\gamma}{2E_D(\mathbf{w})}$.
4. Now iterate steps 1 through 3 until convergence.

Bear in mind that with each reestimate of the objective function parameters the objective function is changing; therefore, the minimum point is moving. If traversing the performance surface generally moves toward the next minimum point, then the new estimates for the objective function parameters will be more precise. Eventually, the precision will be good enough that the objective function will not significantly change in subsequent iterations. Thus, we will obtain convergence.

When this Gauss-Newton approximation to Bayesian regularization (GNBR) algorithm is used, the best results are obtained if the training data is first mapped into the range [-1,1] (or some similar region). We typically scale both inputs and outputs.

After training, there are some casual checks that should be administered. First, if the final effective number of parameters γ is very close to the actual number of parameters N, then the neural network may not be large enough to properly represent the true function. In this case, simply add more hidden layer neurons and retrain. If the larger network has the same final γ, then the smaller network was large enough. Otherwise, more hidden layer neurons may need to be added. The second check is one of consistency of results. If the network is sufficiently large, then a second larger network will achieve comparable values for γ, E_D and E_W.

Now we will show an example of GNBR training. For this simple test we will use a triangular waveform. The training set consists of one hundred data points with Gaussian noise of zero mean and 0.01 variance added to sampled points of the triangle wave. Figure 1 shows the resulting functions for a 1-6-1 network without regularization and a 1-6-1 network trained with the GNBR algorithm. Note the overfitting that occurs without regularization.

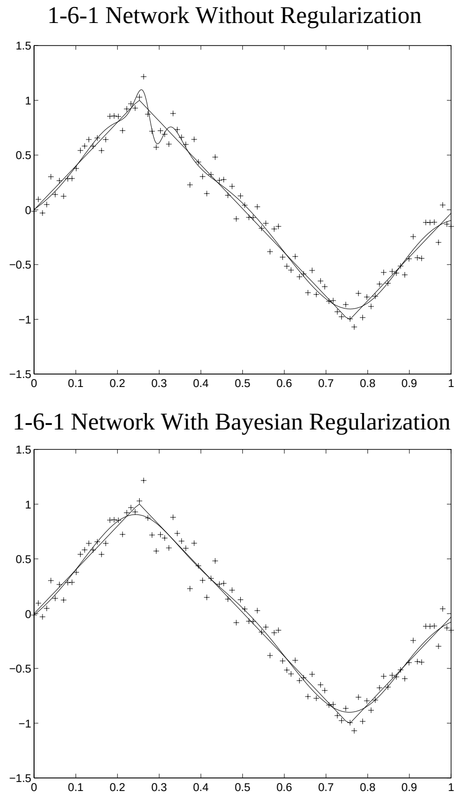


Figure 1 Approximation of the Triangle Wave

Table 1 summarizes the results for training many different networks of the 1-S-1 architecture. Notice how the effective number of parameters γ is constant for any network with at least 4 hidden layer neurons. This is the minimum size network required to properly represent the true function. The actual number of parameters N can increase to 43, providing thirty superfluous parameters, and yet we still get very consistent results. As noted before, the E_D and E_W functions remain constant as the size of network is increased. Also, because we know the true function, we can calculate the actual errors. This is designated as E_A in the table. This function is also very stable for any sufficiently complex network. The GNBR algorithm has produced optimal results for all networks of at least the minimal size. Thus, in contrast to some other techniques, the GNBR algorithm has the capability to produce optimal results the first time. Further, the cost of implementing the changes in the training algorithm was minimal since we are using the Gauss-Newton approximation to the Hessian for computation.

Table 1 Triangle wave results

S	E_D	E_W	E_A	N	γ
2	1.612	203.0	0.5031	7	5.659
3	1.214	187.8	0.1954	10	8.468
4	1.144	177.0	0.1080	13	9.843
5	1.143	177.2	0.1085	16	9.906
6	1.143	177.2	0.1088	19	9.908
8	1.143	177.1	0.1091	25	9.911
10	1.142	177.1	0.1093	31	9.913
14	1.142	177.0	0.1095	43	9.915

Experiments

To test the GNBR algorithm, we chose three real-world problems: a single-input/single-output regression, a time series prediction and an artificial chaotic series model.

Single-Input/Single-Output Regression

The first test set contains ages and weight-to-height ratios for preschool boys [6]. Figure 2 shows the normalized data “+” along with a trained neural network response. Table 2 summarizes the training results for several different networks of the 1-*S*-1 architecture. (E_T is the error on a test set containing 10% of the data which was held out from the training set.) Notice that for all models with $S \geq 4$ the errors and γ are the same. The GNBR algorithm produces consistent results, and the network response in Figure 2 clearly generalizes well.

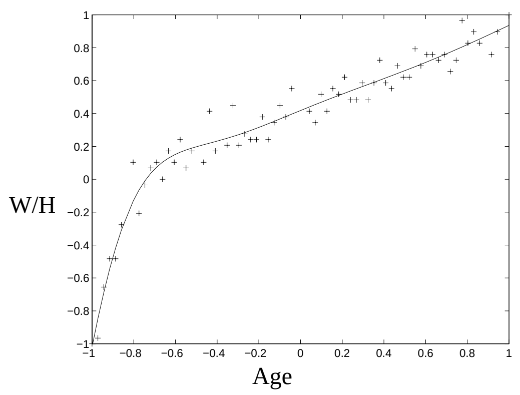


Figure 2 Normalized Preschool Boys’ Data

Table 2 Boy’s Data

S	E_D	E_W	E_T	N	γ
1	1.262	15.42	0.0475	4	3.055
2	0.443	45.65	0.0095	7	5.984
3	0.444	43.37	0.0087	10	6.059
4	0.441	41.63	0.0124	13	7.244
6	0.441	41.63	0.0124	19	7.244
8	0.441	41.63	0.0124	25	7.244
10	0.441	41.63	0.0124	31	7.244
20	0.441	41.63	0.0124	61	7.244

Chaotic Series

The Mackey-Glass chaotic equation [7] is

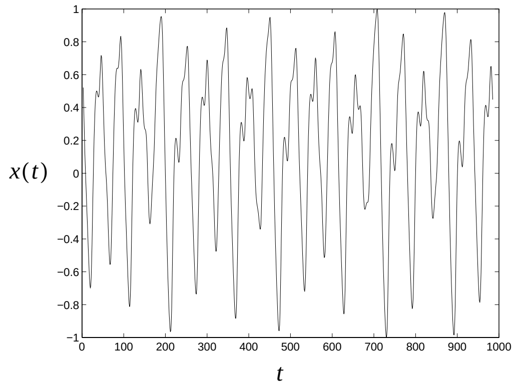
$$\dot{x}(t) = \frac{ax(t-\tau)}{1 + x(t-\tau)^{10}} - bx(t) \tag{9}$$

For our test we set the characteristic parameters to $a = 0.2$, $b = 0.1$ and $\tau = 17$. Letting $\Delta t = 1$, we iterated the equation to produce a time series. Skipping the first 1000 iterates (transient period), we captured the second 1000 points. The data is shown in Figure 3. We then normalized the data set and took the last 100 points (10%) as a testing set. A two-input/single-output network was then trained to predict the next time point from the current time point and the time point 17 time steps back.

Table 3 summarizes the training of 2-S-1 networks on the Mackey-Glass data. The effective number of parameters reached a maximum of 22 with the 2-7-1 network. Even though the actual number of parameters is increased, the effective number of parameters remained roughly constant, indicating that the 2-7-1 network is the smallest network with sufficient complexity to fit the data. In the lower graph of Figure 3 the predictions of the 2-7-1 network are indicated by “x” and the actual data points are indicated by “o”.

Table 3 Mackey-Glass Data

S	E_D	E_W	E_T	N	γ
1	2.5320	68.56	0.4012	5	4.077
2	0.0986	107.2	0.0147	9	8.178
3	0.0853	98.81	0.0124	13	11.82
4	0.0744	14.70	0.0108	17	16.71
5	0.0738	15.07	0.0108	21	19.47
6	0.0738	14.88	0.0108	25	20.91
7	0.0738	14.41	0.0108	29	22.07
8	0.0738	14.41	0.0108	33	22.08
12	0.0737	15.27	0.0108	49	21.61



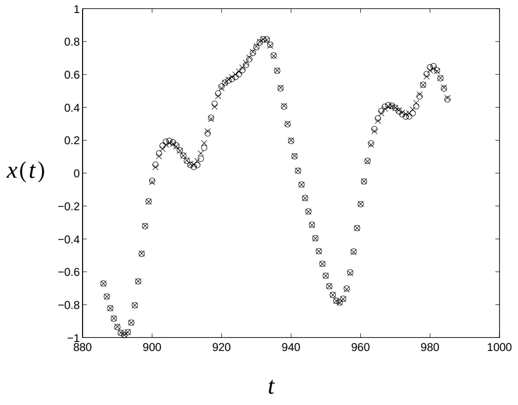
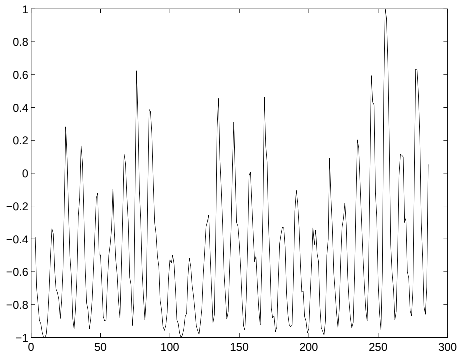


Figure 3 Mackey-Glass Data

Sunspot Data

The next experimental set is the annual average sunspot observations from 1700-1988 [8]. This data is displayed in Figure 4. Box and Jenkins [9] suggest a second order autoregressive model for this time series. Thus, networks with architecture 2-S-1 were used to predict the next year’s sunspot activity from the activity in the previous two years.

Table 4 summarizes the results of training on the sunspot data. The optimal network architecture is found to be 2-4-1. If we increase S beyond 4 the effective number of parameters remains constant. The sum of squared error on the training and test sets is consistent with the results described in [9].



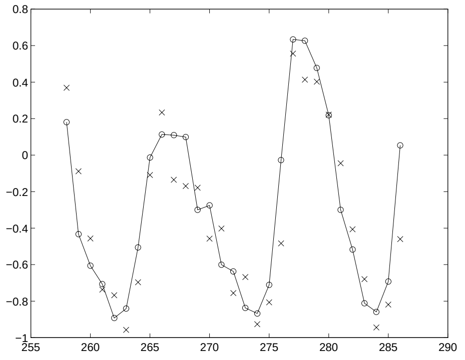


Figure 4 Sunspot Data

Table 4 Sunspot Data

S	E_D	E_W	E_T	N	γ
1	6.999	3.993	1.474	5	4.507
2	5.426	13.88	1.123	9	8.030
3	5.297	13.61	1.459	13	10.55
4	5.105	13.79	1.187	17	12.52
8	5.105	13.79	1.187	33	12.52
10	5.105	13.79	1.187	41	12.52
30	5.105	13.79	1.187	121	12.52

Summary and Conclusions

In this paper we have discussed the use of Bayesian regularization to prevent overfitting in neural network training. The Bayesian framework developed by David MacKay allows the optimal setting of the regularization parameters. One drawback of this Bayesian approach is that it requires the computation of the Hessian matrix of the performance index. In this paper we have introduced the GNBR algorithm, which uses a Gauss-Newton approximation to the Hessian matrix. The additional overhead of this Gauss-Newton approximation to Bayesian regularization is minimal when the Levenberg-Marquardt optimization algorithm is used to locate the optimal weights. Tests on several practical problems demonstrate that the GNBR algorithm provides good generalization in a number of settings.

References

[1] D. J. C. MacKay, "Bayesian Interpolation," *Neural Computation*, vol. 4, pp. 415-447, 1992.

[2] M. T. Hagan, H. B. Demuth and M. Beale, *Neural Network Design*, Boston: PWS Publishing Co., 1996.

[3] M. T. Hagan and M. Menhaj, "Training multilayer networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, 1994, pp. 989-993.

[4] F. D. Foresee, *Generalization and Neural Networks*, Ph.D. Dissertation, Oklahoma State University, 1996.

[5] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," *Proceedings of the IJCNN*, vol. 3, pp. 21–26, July 1990.

[6] E. S. Eppright, H. M. Fox, B. A. Fryer, G. H. Lamkin, V. M. Vivian and E. S. Fuller, "Nutrition of Infants and Preschool Children in the North Central Region of the United States of America," *World Rev. Nutrition and Dietetics*, vol. 14, pp. 269-332, 1972.

[7] M. Plutowski, G. Cottrell and H. White, "Experience with Selecting Exemplars from Clean Data," *Neural Networks*, vol. 9, pp. 273-294, 1996.

[8] H. Tong, *Non-linear Time Series: A Dynamical System Approach*, New York: Oxford University Press, 1990.

[9] G. E. P. Box and G. M. Jenkins, *Time Series Analysis Forecasting and Control*, Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1976.