On the Kalman Filtering Method in Neural-Network Training and Pruning

John Sum, Chi-sing Leung, Gilbert H. Young, and Wing-kay Kan

***Abstract*— In the use of extended Kalman filter approach in training and pruning a feedforward neural network, one usually encounters the problems on how to set the initial condition and how to use the result obtained to prune a neural network. In this paper, some cues on the setting of the initial condition will be presented with a simple example illustrated. Then based on three assumptions—1) the size of training set is large enough; 2) the training is able to converge; and 3) the trained network model is close to the actual one, an elegant equation linking the error sensitivity measure (the saliency) and the result obtained via extended Kalman filter is devised. The validity of the devised equation is then testified by a simulated example.**

***Index Terms*—Extended Kalman filter, multilayer perceptron, pruning training, weight saliency.**

I. Introduction

IN neural-network training, the most well-known online training method is the backpropagation algorithm (BPA) [18], which is a first-order stochastic gradient descent method. Its learning speed could be very slow. Many modified schemes inspired by the classical nonlinear programming technique [12], [7], using the approximation of the Hessian matrix of the error function, have been suggested in an attempted to speed up the training [14]. These schemes are usually computationally inexpensive compared with the full second-order methods; but they ofter involve a number of tuning parameters. A class of second-order descent methods inspired by the theory of system identification and nonlinear filtering [1] has recently been introduced to estimate the weights of a neural network. Two common examples are extended Kalman filter (EKF) which is applied to the training of multilayer perceptron [8], [19], [20], [22], [23] and recurrent neural network [24], [16], and the recursive least square (RLS) method which is applied to multilayer perceptron [3], [9], [11] and recurrent radial basis [4], [2]. The EKF approach is an online mode training in that the weights are updated immediately after the presentation of a training pattern. The training methods are useful in that they do not require the storage of the entire input–output history. With EKF algorithms, the learning speed is improved and the number of tuning parameters is reduced.

Manuscript received June 3, 1997.

J. Sum is with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong.

C. Leung is with the School of Applied Science, Nanyang Technological University, Singapore.

G. H. Young is with the Department of Computing, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong.

W. Kan is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, N.T., Hong Kong.

Publisher Item Identifier S 1045-9227(99)01004-8.

Another concern in neural networks is the size of a neural network for a given problem. If the size is too small, the network may not be trained to solve the given problem. On the other hand, if the size is too large, overfitting may usually occur [13] and also the resource is wasted. An approach to determine the size is hard pruning [17], such as optimal brain damage [10], optimal brain surgeon [6] or Finnoff–Hergert–Zimmermann pruning statistics [5]. In such methods, a large network is being trained first and some unimportant weights are removed later. In hard pruning, a Hessian matrix [10], [6] or a statistics [5] that is obtained by feeding the training samples or the testing samples into the trained network is required. In online control situation, the training samples or the testing samples may not be available after learning, the computational complexity of calculating such Hessian matrix could be substantial since the number of such samples could be very large.

As the EKF approach was shown to be a fast learning and some information such as the Kalman gain and error covariance matrix can be obtained during the progress of training, it would be interesting to inquire if there is any possibility of using EKF method together with hard pruning in order to speed up the learning process, as well as to determine the size of the trained network based on these information.

In this paper, we will present some results trying to connect extended Kalman filter and neural-network pruning. Specifically, we would like to present how can those result obtained by using extended Kalman filter training method be applied to measure the importance of a weight in a network and then give a simulation to illustrate and verify such idea. In the next section, the EKF training method will be introduced. After that, we will in section three describe how weight importance can be evaluated using the covariance matrix and the estimated weight vector which are being obtained via EKF method. The validity of the deduced equation will be described in Section IV and Section V will present the conclusion.

II. Kalman Filtering Method in Training

Let $y = f(x, \theta)$ be the transfer function of a single-layer feedforward neural network where $y \in R$ is the output, $x \in R^m$ is the input and $\theta \in R^n$ is its parameter vector. Given a set of training data $\{x(i), y(i)\}_{i=1}^{N}$, the training of a neural network can be formulated as a filtering problem [1], [20] assuming that the data are generated by the following noisy signal model:

$$\theta(t) = \theta(t-1) + v(t) \tag{1}$$

$$y(t) = f(x(t), \theta(t)) + \epsilon(t) \tag{2}$$

where $v(t)$ and $\epsilon(t)$ are zero mean Gaussian noise with variance $Q(t)$ and $R(t)$. A *good* estimation of the system parameter θ can thus be obtained by via the extended Kalman filter method [8], [19], [16], [22]

$$S(t) = H^T(t)[P(t-1) + Q(t)]H(t) + R(t) \tag{3}$$
$$L(t) = [P(t-1) + Q(t)]H(t)S^{-1}(t) \tag{4}$$
$$P(t) = (I_{n\times n} - L(t)H(t))P(t-1) \tag{5}$$
$$\hat{\theta}(t) = \hat{\theta}(t-1) + L(t)(y(t) - f(x(t), \hat{\theta}(t-1))) \tag{6}$$

where $H(t) = \frac{\partial f}{\partial \theta}|_{\theta=\hat{\theta}(t-1)}$.

III. Use of Matrix P in the Evaluation of Weight Saliency

It should note that after training, the only information that we have are 1) *the parametric vector* $\hat{\theta}$ and 2) the covariance matrix. For simplicity, we rewrite (5) in the following form:

$$P^{-1}(t) = [P(t-1) + Q(t)]^{-1} + H(t)R^{-1}H^T(t). \tag{7}$$

Suppose that the weight parameter and the covariance matrix P both converge, the matrix $Q(t)$ is constant diagonal matrix denoted by Q_0, where

$$Q_0 = qI$$

and

$$R = 1$$

we can readily establish the asymptotic behavior for matrix P

$$P_\infty^{-1} = [P_\infty + Q_0]^{-1} + H(t)H^T(t) \tag{8}$$

where

$$P_\infty = \lim_{t\to\infty} P(t).$$

Further assuming that there exist a time t_0 such that for all time $t > t_0$, all $P(t)$ will be very close to the limiting matrix P_∞, it is possible to deduce that

$$\begin{aligned}\frac{1}{N-t_0}\sum_{t=t_0+1}^{N} P_\infty^{-1} - [P_\infty + Q_0]^{-1} \\ = \frac{1}{N-t_0}\sum_{t=t_0+1}^{N} H(t)H(t)^T.\end{aligned} \tag{9}$$

When N is large, the second term in the right-hand side will approach to the expectation of $H(t)H^T(t)$ which is depended on the input x. Hence

$$P_\infty^{-1} = [P_\infty + Q_0]^{-1} + E[H(t)H^T(t)]. \tag{10}$$

By definition, Q_0 is a positive definite matrix, it is readily shown that P_∞ is also positive definite.

Furthermore, we can diagonal decompose the matrix P_∞ by

$$P_\infty = UDU^T$$

where D is the diagonal matrix with the eigenvalues of P_∞ as the elements, namely

$$D = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \lambda_{n_\theta} \end{bmatrix}$$

and the matrix U contains the corresponding eigenvectors. Under such decomposition, (10) can be rewritten as follows:

$$U[D^{-1} - (D+qI)^{-1}]U^T = E[H(t)H^T(t)]. \tag{11}$$

Here, $E[x]$ means the expectation of x. Now let β_k be the kth diagonal element of the matrix $[D^{-1} - (D+qI)^{-1}]$

$$\beta_k = \frac{1}{\lambda_k} - \frac{1}{\lambda_k + q} = \frac{q}{\lambda_k(\lambda_k+q)}. \tag{12}$$

Two special cases can thus be obtained

$$\beta_k \approx \begin{cases} \lambda_k^{-1}, & \text{if } q \gg \max\{\lambda_k\} \\ q\lambda_k^{-2}, & \text{if } q \ll \min\{\lambda_k\} \end{cases} \tag{13}$$

Empirically, we have found that λ_ks *are usually much larger than* q. So in the rest of the paper, we assume that $\beta_k \approx q/\lambda_k^2$.

It should note that P_∞^{-1} is equal to $UD^{-1}U^T$. Hence, putting the values of β_k into (10)–(11), it is readily to get an approximation for the factor $E[H(t)H^T(t)]$

$$\begin{aligned} E[H(t)H^T(t)] &\approx U\big[D^{-1} - (D+qI)^{-1}\big]U^T \\ &= qP_\infty^{-2}. \end{aligned}$$

Hence

$$P_\infty^{-2} \approx q^{-1}E[H(t)H^T(t)]. \tag{14}$$

Practically, we could not know $E[H(t)H^T(t)]$, so we approximate it by the mean average $\sum_t H(t)H^T(t)$. Putting back the definition of $H(t)$, we get that

$$E\left[\frac{\partial f}{\partial \theta_k}^2\right] \approx q\big(P_\infty^{-2}\big)_{kk}. \tag{15}$$

Here $(A)_{kk}$ denotes the kth diagonal element of a matrix A. With this, the expected change of error, denoted by E_k, due to the setting of θ_k being zero can be discussed.

A. Single Pruned Weight

Recall that the *true* function is defined as a nonlinear regressor, $y(x) = f(x, \theta_0) + \textit{noise}$, with parameter θ_0. After training, an approximation, $\hat{y}(x) = f(x, \hat{\theta})$ is obtained. We drop the subscript N from $\hat{\theta}$ for simplicity. Let $\hat{\theta}_1, \cdots, \hat{\theta}_{n_\theta}$ be the elements of $\hat{\theta}$, and σ^2 be the variance of the output noise, the expected predicted square error (EPSE) of this approximated function would be given by

$$E[(y(x) - \hat{y}(x))]^2 = E[(f(x,\theta_0) + \textit{noise} - f(x,\hat{\theta}))^2] \tag{16}$$
$$= E[(f(x,\theta_0) - f(x,\hat{\theta}))^2] + \sigma^2 \tag{17}$$

for $\hat{\theta} \to \theta_0$. It should be remarked that the expectation is taken over to future data.

Now consider that the kth element of $\hat{\theta}$ is being set to zero; let $\hat{y}_p(x)$ be the approximated function and $\hat{\theta}_p$ be the corresponding parametric vector, we can have the EPSE given by

$$E[(y(x) - \hat{y}_p(x))]^2 = E[(f(x,\theta_0) + \textit{noise} - f(x,\hat{\theta}_p))^2] \tag{18}$$

As the difference $(y(x) - \hat{y}_p(x))$ can also be decomposed as follows:

$$\begin{aligned} y(x) - \hat{y}_p(x) = {} & [f(x,\theta_0) + \textit{noise} - f(x,\hat{\theta})] \\ & + [f(x,\hat{\theta}) - f(x,\hat{\theta}_p)]. \end{aligned} \tag{19}$$

In case that $\hat{\theta}$ is already very close to θ_0, the first term on the right side will be a Gaussian noise term which is independent of the second term. This implies that

$$\begin{aligned} E[(y(x) - \hat{y}_p(x))^2] \approx {} & E[(f(x,\theta_0) - f(x,\hat{\theta}))^2] + \sigma^2 \\ & + E[(f(x,\hat{\theta}) - f(x,\hat{\theta}_p))^2]. \end{aligned} \tag{20}$$

When the magnitude of θ_k is small, the third term on the right-hand side of (20) can be expanded in Taylor series and thus the expected predicted square error can be approximated by

$$\begin{aligned} E[(y(x) - \hat{y}_p(x))^2] \approx {} & E[(f(x,\theta_0) - f(x,\hat{\theta}))^2] + \sigma^2 \\ & + E\left[\frac{\partial f(x,\hat{\theta})}{\partial \theta_k}^2\right]\hat{\theta}_k^2. \end{aligned} \tag{21}$$

Comparing (21) with (17), it is observed that the last term is the expected error increment due to pruning the kth element of $\hat{\theta}$. Using (15), we can now relate the matrix P_∞ and the parametric vector $\hat{\theta}$ in the following manner:

$$\Delta E_k = E\left[\frac{\partial f(x,\hat{\theta}_0)}{\partial \theta_k}^2\right]\hat{\theta}_k^2 \approx q\left(P_\infty^{-2}\right)_{kk}\hat{\theta}_k^2. \tag{22}$$

Here, we use the notation ΔE_k to denote the incremental change of the expected prediction error due to pruning the kth element, that is the expected prediction error sensitivity. Using (22), the weights' importance can be defined.

B. Multiple Pruned Weights

With the use of (22), the weight importance can thus be ranked in accordance with their ΔE_k and the ranking list is denoted by $\{\pi_1, \pi_2, \cdots, \pi_{n_\theta}\}$, where $\Delta E_{\pi_i} \le \Delta E_{\pi_j}$ if $i < j$. If we let $\hat{\theta}_a$ be a vector which $\pi_1, \cdots, \pi_k$ elements being zeros and other elements being identical to the corresponding elements in $\hat{\theta}$, we can estimate the incremental change of mean prediction error by the following formula:

$$\Delta E_{[\pi_1,\pi_k]} \approx q\hat{\theta}_a^T\left(P_\infty^{-2}\right)\hat{\theta}_a. \tag{23}$$

With this equation, we can thus estimate the number of weights (and which one) should be removed given that $\Delta E_{[\pi_1,\pi_k]} <$ *threshold*. It is extremely useful in pruning a neural network. As mentioned in [15], one problem in effective pruning is to determine which weights and how many weights should be removed simultaneously in one pruning step. Equation (23) sheds light on solving that problem as it is an estimation that amount of training error will be increased if the π_1st to π_kth weights are removed.

IV. Testing on the Approximation

To verify that ΔE_k (22) and $\Delta E_{[\pi_1,\pi_k]}$ (23) are good estimations of the change in training error, the generalized XOR problem is being solved. The function to be approximated is defined as follows:

$$y(x_1, x_2) = \mathrm{sign}(x_1)\,\mathrm{sign}(x_2).$$

A feedforward neural network with 20 hidden units is trained using the extended Kalman filter method with

$$P(0) = I.$$

The value of q is set to be 10^{-4} and 8000 training data are generated. The actual training error is defined as follows:

$$E_{\text{train}} = \frac{1}{8000}\sum_{t=1}^{8000}(y_t - \hat{y}_t)^2.$$

After training, another 100 pairs of data are passed to the network and the mean prediction error is defined as the mean squares testing error

$$E_{\text{test}} = \frac{1}{100}\sum_{t=1}^{100}(y_t - \hat{y}_t)^2.$$

The importance of the weights is ranked according to the ΔE_k, (22). The actual change of error is obtained by removing kth weight from the trained neural network and then passing 100 pairs of testing data to the pruned network. We calculate this testing error by the formula

$$E_p(k) = \frac{1}{100}\sum_{t=1}^{100}(y_t - \hat{y}_t)^2.$$

The actual change of error is thus evaluated by

$$\Delta E_k(\textit{actual}) = E_p(k) - E_{\text{test}}.$$

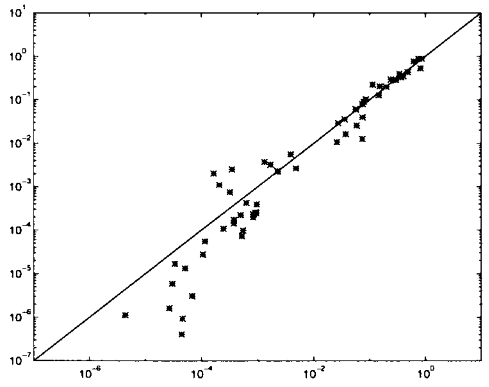
This error term is then compared with the estimate, ΔE_k, in Fig. 1(a). The x-axis corresponds to $E_p(k) - E_{\text{train}}$, i.e., the actual change of error while the y-axis corresponds to ΔE_k, the estimate.

In regard to the ranking list, the accumulative error is estimated via $\Delta E_{[\pi_1,\pi_k]}$ (23). Similarly, to evaluate the actual change of error, another 100 data pairs are generated. According to the ranking list, say $\{\pi_1, \cdots, \pi_{n_\theta}\}$, the π_1 up to π_k weights are removed. Then passing the 100 data pairs to the pruned network, we calculate the actual mean square error by the formulas

$$E_p[\pi_1, \pi_k] = \frac{1}{100}\sum_{t=1}^{100}(y_t - \hat{y}_t)^2.$$

And thus the actual change of error is $E_p[\pi_1, \pi_k] - E_{\text{test}}$. This error term is then compared with the estimate, $\Delta E_{[\pi_1,\pi_k]}$ and shown in Fig. 1(b).

Figs. 2 and 3 show the comparison between the estimated testing error $\Delta E_{[\pi_1,\pi_k]}$ and the actual $\Delta E_{[\pi_1,\pi_k]}$ against number of weights pruned for different values of q. The neural network has 20 hidden units. It is also found that the estimated


(a)

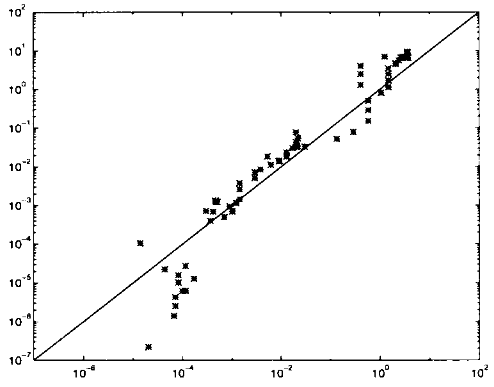
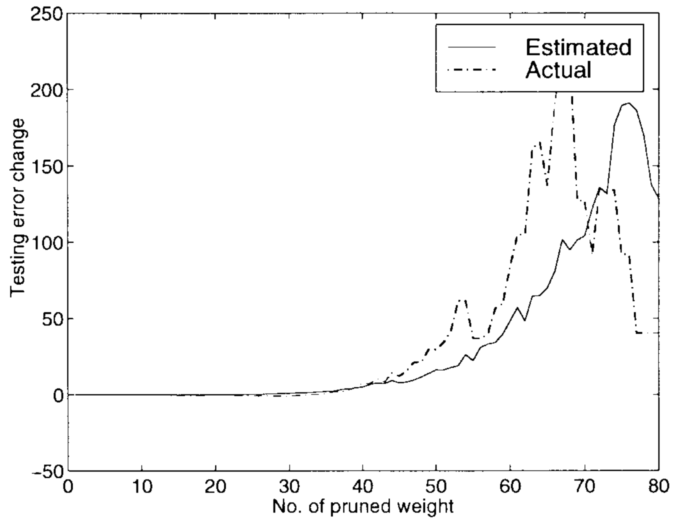
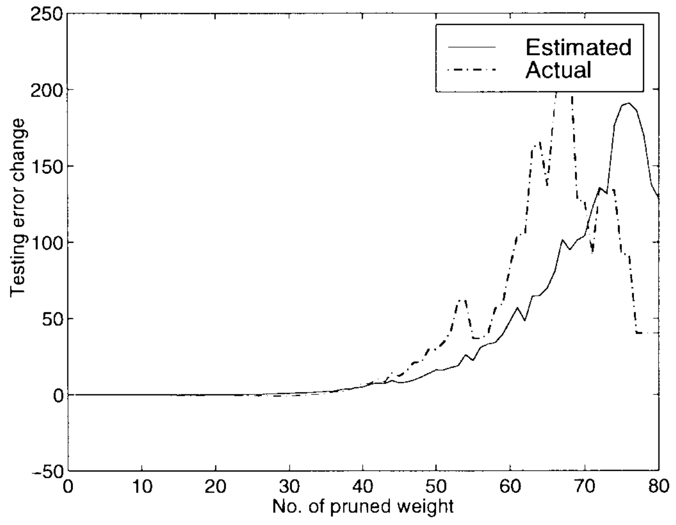

(b)

Fig. 1. Simulation results confirm that ΔE_k (22) and $\Delta E_{[\pi_1,\pi_k]}$ (23) are good estimations on the actual change of error. The vertical axis corresponds to the estimated error and the horizontal axis corresponds to the actual error. (a) ΔE_k for $n = 20$. (b) $\Delta E_{[\pi_1,\pi_k]}$ for $n = 20$.

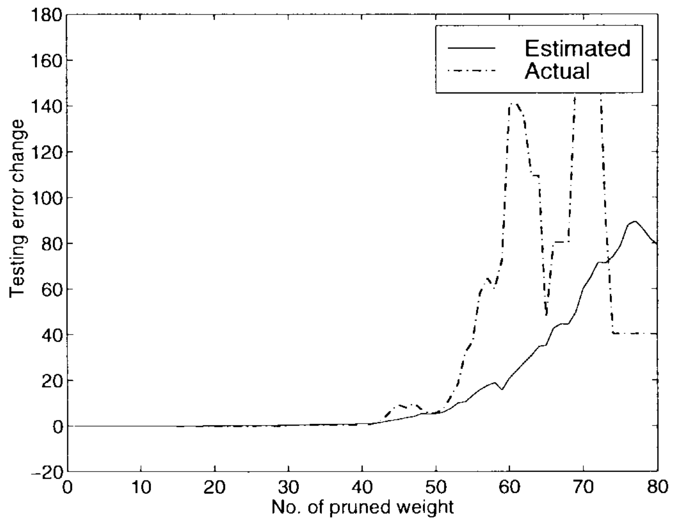


(a)

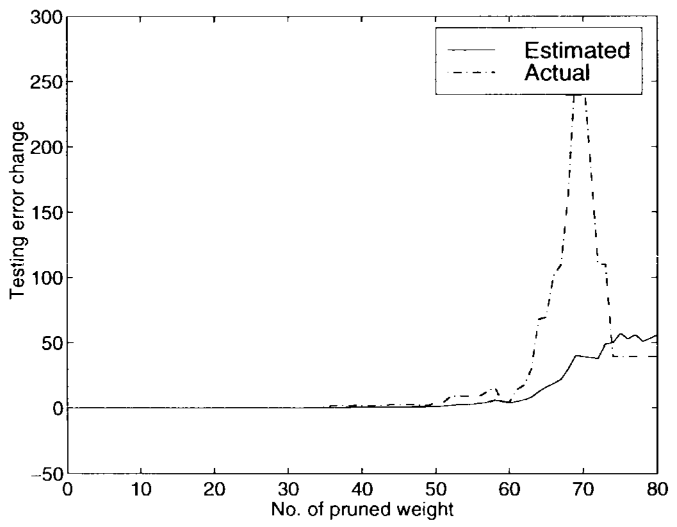


(b)

Fig. 2. Testing error change $\Delta E_{[\pi_1,\pi_k]}$ against number of weights pruned for q equals to (a) 0.001 and (b) 0.002.

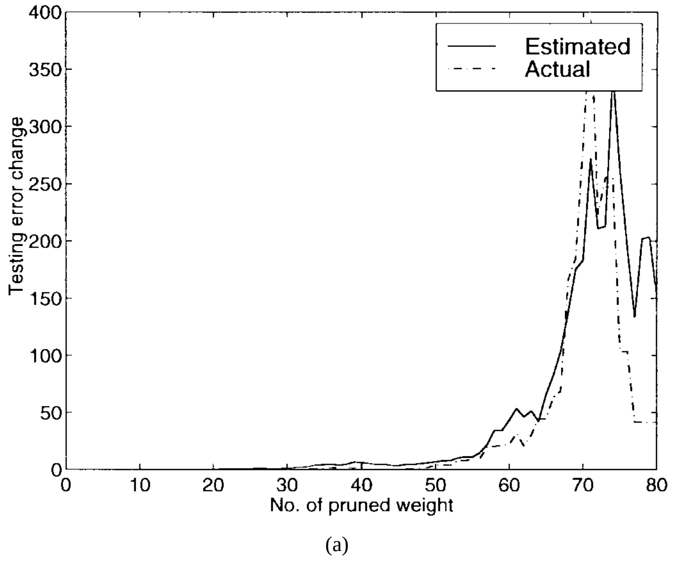


(a)

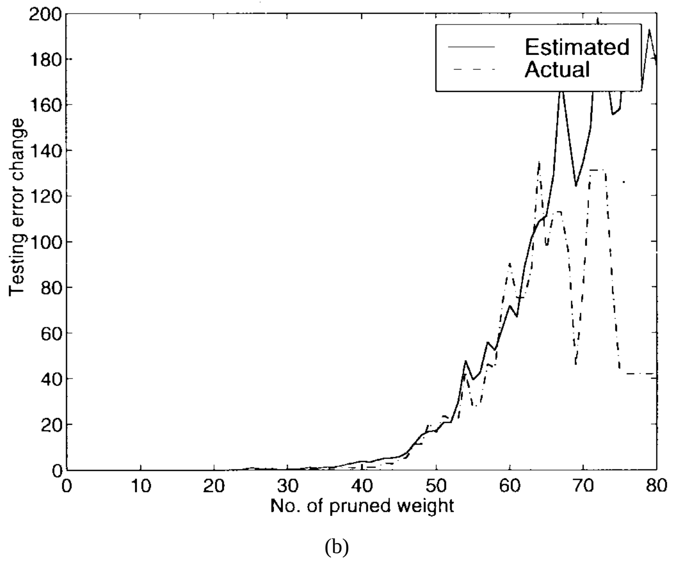


(b)

Fig. 3. Testing error change $\Delta E_{[\pi_1,\pi_k]}$ against number ofweights pruned for q equals to (a) 0.005 and (b) 0.010.

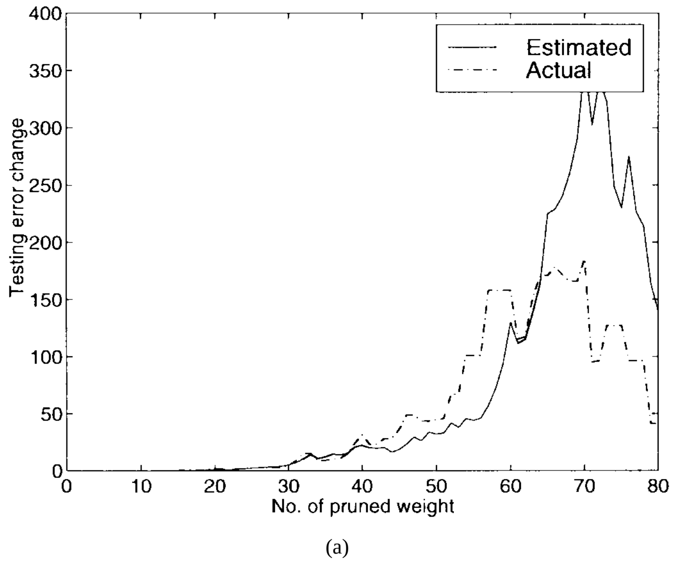


(a)

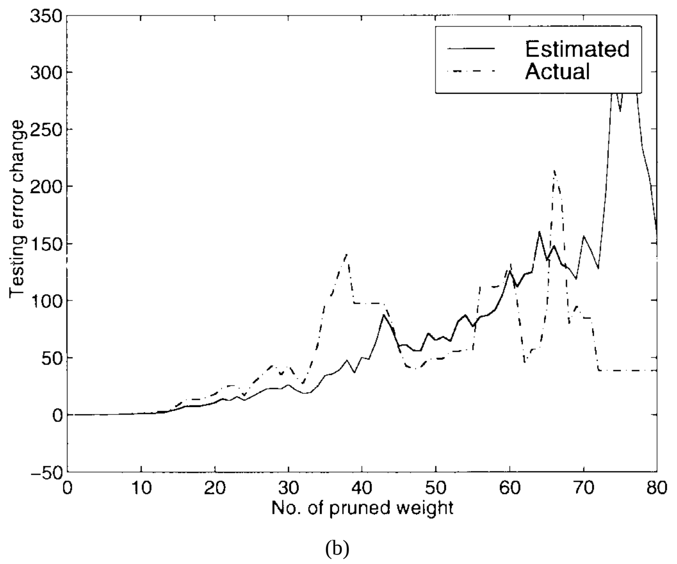


(b)

Fig. 4. Testing error change $\Delta E_{[\pi_1,\pi_k]}$ against number of weights pruned for α equals to (a) 0.001 and (b) 0.002.



(a)



(b)

Fig. 5. Testing error change $\Delta E_{[\pi_1,\pi_k]}$ against number of weights pruned for α equals to (a) 0.005 and (b) 0.010.

$\Delta E_{[\pi_1,\pi_k]}$ can closely estimate the actual $\Delta E_{[\pi_1,\pi_k]}$ for k up to 40.

The same experiment has also been carried out for the forgetting recursive least square (FRLS) training method. The weights are ranked in accordance with

$$S(\hat{\theta}_i(\text{FRLS})) = \alpha\frac{\hat{\theta}_i(\text{FRLS})^2}{2}\left(P_{\text{FRLS}}^{-1}(N) - P_{\text{FRLS}}^{-1}(0)\right)_{ii}.$$

Here, $\hat{\theta}(\text{FRLS})$ and P_{FRLS} are the weight vector and the covariance matrix obtained by using FRLS training method. Figs. 4 and 5 show the comparison between the estimated testing error $\Delta E_{[\pi_1,\pi_k]}$ and the actual $\Delta E_{[\pi_1,\pi_k]}$ against number of weights pruned for different values of α. It is found that for small α the estimated $\Delta E_{[\pi_1,\pi_k]}$ can closely estimate the actual $\Delta E_{[\pi_1,\pi_k]}$ for k around 50. For large α, the estimated $\Delta E_{[\pi_1,\pi_k]}$ can closely estimate the actual $\Delta E_{[\pi_1,\pi_k]}$ for k around 30.

[1]The formulation of FRLS-based pruning algorithm can be found in [21].

Comparing these two results, we can see that the EKF-based weight importance measure can closely approximate the actual incremental change of prediction error for k up to around $0.5n_\theta$, where n_θ is the total number of weights.

V. Conclusion

In this paper, we have presented a method on how to prune a neural network solely based on the results obtained by Kalman filter training such as the weight vector $\hat{\theta}$ and the $P(N)$ matrix. With the assumptions that 1) the training converges and 2) $\hat{\theta}$ is close to θ_0 and the size of the training data is large enough, we have derived an elegant equation expressing such relation. Making use of this equation, we are able to estimate the number of weight should be pruned and the which weight should be pruned away. Finally, it should noted that the estimated equation for the error sensitivity should also be applied to the detection of redundant input, as studied by

Zurada *et al.* in [25]. Investigation along this line is worthwhile for future research.

References

[1] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*. Englewood Cliffs, NJ: Prentice-Hall, 1979.

[2] S. A. Billings and C. F. Fung, "Recurrent radial basis function networks for adaptive noise cancellation," *Neural Networks*, vol. 8, no. 2, pp. 273–190, 1995.

[3] S. Chen, C. F. N. Cowan, S. A. Billings, and P. M. Grant, "Parallel recursive prediction error algorithm for training layered neural network," *Int. J. Contr.*, vol. 51, no. 6, pp. 1215–1228, 1990.

[4] S. Chen, S. A. Billing, and P. M. Grant, "A recursive hybrid algorithm for nonlinear system identification using radial basis function networks," *Int. J. Contr.*, vol. 55, no. 5, pp. 1051–1070, 1992.

[5] W. Finnoff, F. Hergert, and H. G. Zimmermann, "Improving model selection by nonconvergent methods," *Neural Networks*, vol. 6, pp. 771–783, 1993.

[6] B. Hassibi and D. G. Stork, "Second-order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems*, Hanson *et al.*, Eds., 1993, pp. 164–171.

[7] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: Macmillan, 1994.

[8] Y. Iiguni, H. Sakai, and H. Tokumaru, "A real-time learning algorithm for a multilayered neural network based on the extended Kalman filter," *IEEE Trans. Signal Processing*, vol. 40, pp. 959–966, 1992.

[9] S. Kollias and D. Anastassiou, "An adaptive least squares algorithm for the efficient training of artificial neural networks," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1092–1101, 1989.

[10] Y. LeCun *et al.*, "Optimal brain damage," *Advances Neural Inform. Processing Syst. 2*, D. S. Touretsky, Ed., pp. 396–404, 1990.

[11] C. S. Leung *et al.*, "On-line training and pruning for RLS algorithms," *Electron. Lett.*, No. 23, pp. 2152–2153, 1996.

[12] D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*. Reading, MA, Addison-Wesley, 1973.

[13] J. E. Moody, "Note on generalization, regularization, and architecture selection in nonlinear learning systems," in *1st IEEE-SP Wkshp. Neural Networks for Signal Processing*, 1991.

[14] O. S. P. Bojarczak and M. Stodolski, "Fast second-order learning algorithm for feedforward multilayer neural networks and its application," *Neural Networks*, vol. 9, no. 9, pp. 1583–1596, 1996.

[15] L. Prechelt, "Comparing adaptive and nonadaptive connection pruning with pure early stopping," *Progress in Neural Information Processing*, pp. 46–52, 1996.

[16] G. V. Puskorius and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks," *IEEE Trans. Neural Networks*, vol. 5, pp. 279–297, 1994.

[17] R. Reed, "Pruning algorithms—A survey," *IEEE Trans. Neural Networks*, vol. 4, pp. 740–747, 1993.

[18] D. Rumelhart, G. Hinton, and G. Williams, "Learning internal representations by error propagation," *Parallel Distributed Processing*. Cambridge, MA: MIT Press, vol. 1, 1986.

[19] S. Shah, F. Palmeieri, and M. Datum, "Optimal filtering algorithms for fast learning in feedforward neural networks," *Neural Networks*, vol. 5, pp. 779–787, 1992.

[20] S. Singhal and L. Wu, "Training multilayer perceptrons with the extended Kalman algorithm," in *Advances in Neural Inform. Processing Syst. I*, D. S. Touretzky, Ed., pp. 133–140, 1989.

[21] J. Sum, "Extended Kalman filter based pruning algorithms and several aspects of neural network learning," Ph.D. dissertation, Dept. Comput. Sci. Eng., Chinese Univ. Hong Kong, July 1998.

[22] E. A. Wan and A. T. Nelson, "Dual Kalman filtering methods for nonlinear prediction, smoothing, and estimation," to appear in *NIPS'96*, 1996.

[23] W. K. T. Fukuda and S. G. Tzafestas, "Learning algorithms of layered neural networks via extended Kalman filters," *Int. J. Syst. Sci.*, vol. 22, no. 4, pp. 753–768, 1991.

[24] R. J. Williams, "Training recurrent networks using the extended Kalman filter," in *Proc. IJCNN'92*, Baltimore, 1992, Vol. IV, pp. 241–246.

[25] J. M. Zurada, A. Malinowski, and S. Usui, "Perturbation method for deleting redundant inputs of perceptron networks," *Neurocomputing*, vol. 14, pp. 177–193, 1997.