Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 69 (2006) 2420–2424
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Letters

On-line estimation of the final prediction error via recursive least-squares method

John Sum[a,*], Kevin Ho[b,**]

[a]Department of Information Management, Chung Shan Medical University, Taichung 402, Taiwan
[b]Department of Computer Science and Communication Engineering, Providence University, Sha-Lu, Taiwan

Received 14 August 2005; received in revised form 7 February 2006; accepted 7 February 2006
Communicated by R.W. Newcomb
Available online 27 June 2006

Abstract

This paper starting from the very first principle presents a derivation of an equation estimating of the final prediction error for a neural network under the recursive least square framework. The equation is in the form:

$$\langle\langle PE\rangle_F\rangle_T = \langle TE\rangle_T \frac{N+d_1}{N-d_2},$$

where d_1 and d_2 are some values determined by the gradient of the nonlinear mapping at the true system parameter. A cheap way of estimating such prediction error based on the information obtained via the recursive least square training method is suggested.

Keywords: Feedforward neural network; Final prediction error; Recursive least square; On-line

1. Introduction

To alleviate the slow convergence of back-propagation algorithm (BPA) in training a feedforward neural network (FNN), the recursive least square (RLS) and the extended Kalman filter (EKF) [4,6] have been two popular methods to train a FNN efficiently in the last decade. The beauty of using the RLS and the EKF is that the network weights can be updated immediately after the presentation of a training pattern. This is important to many applications, such as real-time control and time series prediction. In a real-time environment, both training and prediction have to be done right after (or a few steps after) the arrival of a new data.

Kollias and Anastassiou [8] considered a FNN as a nonlinear system and directly applied the RLS method to train a FNN. Observe that the weights associated with a hidden neuron is loosely coupled with other hidden neurons, Shah et al. in [20] extended the work by simplifying the update equation for the weight covariance matrix. At the same time, Singhal and Wu [19], Watanabe et al. [23] and Iiguni et al. [5] independently formulated the training of a FNN as a filtering problem and then directly applied the EKF to train a neural network. Similarly, EKF method has also been applied in training other multi-layered FNN models for system control [18] and time series forecast [7]. In view of the effectiveness of using the RLS and the EKF as an on-line training method, Leung et al. [10–12], Sum et al. [22] and Chang et al. [3] extended such methods for on-line FNN pruning.

Whatever training method is applied, one will need to estimate the prediction error of such a trained FNN. Conventionally, this prediction error is normally measured after the training has been completed [2,13,15,17]. In this regard, extending the RLS method to estimate the prediction error adaptively would be very useful to real-time applications. In this paper, an on-line algorithm for obtaining the prediction error of a FNN, using the RLS method, will be presented. In the next section, the training objective of the RLS will be reviewed. Follow the approach

*Corresponding author.
**Also Corresponding author.
E-mail addresses: pfsum@csmu.edu.tw (J. Sum), ho@pu.edu.tw (K. Ho).

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.02.005

taken by Moody [15] and Murata et al. [17], the expected prediction error and the final prediction error (FPE) will be derived in Sections 3 and 4, respectively. The recursive algorithm will thus be presented in Section 5. We present the conclusions of this paper in Section 6.

2. Training objective for recursive least square

In this paper, we will consider a nonlinear neural network defined as follows:

$$y(x) = f(x,\theta) + \varepsilon, \tag{1}$$

where $y \in R$ is the output of the network, $x \in R$ is the input, $\theta \in R^n$ is the parametric vector and ε is a zero mean Gaussian noise with variance λ.[1] Given N training data $\{(x_t, y_t)\}_{t=1}^N$, the parametric vector θ can be estimated by the following recursive equations [8,10,11,20]:

$$P(t) = P(t-1) - \frac{P(t-1)H(t)H^{\mathrm{T}}(t)P(t-1)}{H^{\mathrm{T}}(t)P(t-1)H(t)+1}, \tag{2}$$

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{P(t-1)H(t)[y(x_t) - f(x_t,\hat{\theta}(t-1))]}{H^{\mathrm{T}}(t)P(t-1)H(t)+1}, \tag{3}$$

where

$$H(t) = \left.\frac{\partial f}{\partial \theta}\right|_{\theta=\hat{\theta}(t-1)}, \tag{4}$$

$$P(0) = \delta^{-1} I_{n\times n}.(\delta > 0). \tag{5}$$

The last equation for $P(0)$ can ensure that $P(t)$ does not trap in zero matrix. It is because zero matrix is a trivial solution of Eq. (2). The initial guess of the parametric vector $\hat{\theta}$ is set to θ_{ig}, i.e.

$$\hat{\theta}(0) = \theta_{\mathrm{ig}}.$$

The objective function of a recursive least square training is equivalent to the following penalty mean square error function [4]:

$$J(\hat{\theta}(N)) = \frac{1}{N}\sum_{t=1}^{N}(y(x_t) - f(x_t,\hat{\theta}(N)))^2 + \frac{\delta}{N}(\hat{\theta}(N)-\theta_{\mathrm{ig}})^{\mathrm{T}}(\hat{\theta}(N)-\theta_{\mathrm{ig}}). \tag{6}$$

The existence of the second term is due to Eq. (5). Now if we assume that the initial guess of the parameter θ is already close to the true parameter θ_0, this objective function could be locally approximated by a quadratic function,

$$J(\hat{\theta}(N)) \approx J(\theta_0) + \tfrac{1}{2}(\hat{\theta}(N)-\theta_0)^{\mathrm{T}}\nabla^2 J(\theta_0)(\hat{\theta}(N)-\theta_0). \tag{7}$$

[1] For a particular instance, say t, we also express the model as $y(x_t) = f(x_t,\theta) + \varepsilon_t$, where ε_t (for all $t = 1, 2, \ldots$) is a zero mean Gaussian noise with variance λ. $y(x_t)$ might even be written as y_t for notational simplicity.

Similar to the technique being used in [21], we let $\theta_{\mathrm{ig}} \approx \theta_0$. Then, $\hat{\theta}(N) \approx \theta_{\mathrm{ig}}$ and $(\hat{\theta}(N)-\theta_{\mathrm{ig}})^{\mathrm{T}}(\hat{\theta}(N)-\theta_{\mathrm{ig}}) \to 0$. For notational simplicity, we denote hereafter $\hat{\theta}(N)$ by θ. So,

$$\theta - \theta_0 \approx [\nabla^2 J(\theta_0)]^{-1}\nabla J(\theta) \tag{8}$$

$$= \left[\frac{1}{N}\sum_{t=1}^{N}\frac{\partial f(x_t,\theta_0)}{\partial\theta}\frac{\partial f^{\mathrm{T}}(x_t,\theta_0)}{\partial\theta} + \frac{\delta}{N}I\right]^{-1} \times \left[\frac{1}{N}\sum_{t=1}^{N}\varepsilon_t\frac{\partial f(x_t,\theta_0)}{\partial\theta}\right]. \tag{9}$$

Since ε_t is zero mean Gaussian random noise with variance λ_0, the variance of the estimated θ can be approximated as follows:

$$\langle(\theta-\theta_0)(\theta-\theta_0)^{\mathrm{T}}\rangle_T \approx \frac{\lambda_0}{N}\left[G+\frac{\delta}{N}I\right]^{-1} G\left[G+\frac{\delta}{N}I\right]^{-1}, \tag{10}$$

where the matrix G is given by

$$G = \frac{1}{N}\sum_{t=1}^{N}\frac{\partial f(x_t,\theta_0)}{\partial\theta}\frac{\partial f^{\mathrm{T}}(x_t,\theta_0)}{\partial\theta}. \tag{11}$$

The notation $\langle\cdot\rangle_T$ denotes that the expectation is taken over the training data set while the superscript T denotes matrix transpose.

3. The expected prediction error

Using Eq. (10), we are ready to derive an expression for the expected prediction error of the network. The result obtained here is identical to that derived in [14,21]. The reason of including this derivation is for the sake of completeness. The expected prediction error is defined as the expected error which the network will generate if an unseen data, denoted by x^F, is fed to the network after training. We use the superscript F to highlight that the input data is unseen (future data). The expected prediction error is given by

$$\langle PE\rangle_F = \langle(y(x^F) - f(x^F,\theta))^2\rangle_F \tag{12}$$

$$= \langle(f(x^F,\theta_0) + \varepsilon^F - f(x^F,\theta))^2\rangle_F. \tag{13}$$

Since θ is dependent on the training set, the random noise factor ε^F will be independent of the network parameter θ. The expected prediction error can further be simplified.

$$\langle PE\rangle_F \approx \left\langle(\varepsilon^F)^2 + \left(\frac{\partial f^{\mathrm{T}}(x^F,\theta_0)}{\partial\theta}(\theta-\theta_0)\right)^2\right\rangle_F = \lambda_0 + \langle\mathrm{tr}\{(\theta-\theta_0)(\theta-\theta_0)^{\mathrm{T}}G_t\}\rangle_F, \tag{14}$$

where

$$G_t = \frac{\partial f(x^F,\theta_0)}{\partial\theta}\frac{\partial f^{\mathrm{T}}(x^F,\theta_0)}{\partial\theta}.$$

The second term $\langle\mathrm{tr}\{(\theta-\theta_0)(\theta-\theta_0)^{\mathrm{T}}G_t\}\rangle_F$ is obtained by using the property that $a^{\mathrm{T}}b = \mathrm{tr}\{ba^{\mathrm{T}}\}$ for all $a, b \in R^n$.

Using the fact the θ is independent of the future data,

$$\langle PE\rangle_F = \lambda_0 + \mathrm{tr}\{(\theta-\theta_0)(\theta-\theta_0)^{\mathrm{T}}\langle G_t\rangle_F\}. \tag{15}$$

Here tr$\{A\}$ means the trace of matrix A. Suppose, we take the expectation of $\langle PE\rangle_F$ over the training set and assuming that the size of the training set is large enough, we can have the approximation,

$$\left\langle \frac{\partial f(x^F,\theta_0)}{\partial\theta}\frac{\partial f^{\mathrm{T}}(x^F,\theta_0)}{\partial\theta}\right\rangle_F \approx G. \tag{16}$$

By using the approximation given in Eq. (10),

$$\langle\langle PE\rangle_F\rangle_T \approx \lambda_0 + \frac{\lambda_0}{N}\left\{\left[G+\frac{\delta}{N}I\right]^{-1}G\left[G+\frac{\delta}{N}I\right]^{-1}G\right\}. \tag{17}$$

Let $\delta_1,\delta_2,\ldots,\delta_n$ be the eigenvalues of G and using the fact that the trace of a matrix is equal to the sum of its eigenvalues, $\langle\langle PE\rangle_F\rangle_T$ can be rewritten as follows:

$$\langle\langle PE\rangle_F\rangle_T \approx \lambda_0\left(1+\frac{1}{N}\sum_{k=1}^{n}\frac{\delta_k^2}{(\delta_k+\delta/N)^2}\right). \tag{18}$$

This result also indicates that the adding of $\delta>0$ (i.e. setting $P^{-1}(0)=\delta I$) acts like adding a smoothing regularizer or weight decay term [16] to the mean square errors. As the noise variance λ_0 is usually not known in advance [9], the estimation of the prediction error is not feasible unless we can find an estimator for this noise variance. Next, we would like to derive an estimation for that factor based on the training error (TE).

4. The expected training error and the FPE

Consider that the training error, denoted by TE, is defined as follows:

$$TE = \frac{1}{N}\sum_{t=1}^{N}(y(x_t)-f(x_t,\theta))^2. \tag{19}$$

According to the assumption that the estimated parametric vector θ is close to the true value θ_0, we can expand $f(x_t,\theta)$ in a Taylor series and ignore the second-order and higher-order term to get the following approximation:

$$TE = \frac{1}{N}\sum_{t=1}^{N}\varepsilon_t^2 - \frac{2}{N}\sum_{t=1}^{N}\varepsilon_t\frac{\partial f^{\mathrm{T}}(x_t,\theta_0)}{\partial\theta}(\theta-\theta_0) + \frac{1}{N}\sum_{t=1}^{N}\left(\frac{\partial f^{\mathrm{T}}(x_t,\theta_0)}{\partial\theta}(\theta-\theta_0)\right)^2. \tag{20}$$

Again, we consider that the size of the training set is large, the first term in the above equation will be equal to λ_0. Using Eq. (9), the second term becomes:

$$\frac{2}{N}\sum_{t=1}^{N}\varepsilon_t\frac{\partial f_t^{\mathrm{T}}}{\partial\theta}\left[G+\frac{\delta}{N}I\right]^{-1}\left[\frac{1}{N}\sum_{t=1}^{N}\varepsilon_t\frac{\partial f_t}{\partial\theta}\right] \tag{21}$$

which is equal to

$$\frac{2}{N}\mathrm{tr}\left\{\left[\frac{1}{N}\sum_{t=1}^{N}\varepsilon_t\frac{\partial f_t}{\partial\theta}\right]\left[\sum_{t=1}^{N}\varepsilon_t\frac{\partial f_t^{\mathrm{T}}}{\partial\theta}\right]\left[G+\frac{\delta}{N}I\right]^{-1}\right\}. \tag{22}$$

Here, we use the notation f_t representing $f(x_t,\theta_0)$ for simplicity. As ε_i and ε_j are independent, taking the expectation on the above expression would give expectation of the second term in Eq. (20):

$$\left\langle\frac{2}{N}\sum_{t=1}^{N}\varepsilon_t\frac{\partial f^{\mathrm{T}}(x_t,\theta_0)}{\partial\theta}(\theta-\theta_0)\right\rangle_T = \frac{2\lambda_0}{N}\mathrm{tr}\left\{G\left[G+\frac{\delta}{N}I\right]^{-1}\right\} = \frac{2\lambda_0}{N}\sum_{k=1}^{n}\frac{\delta_k}{\delta_k+\delta/N}. \tag{23}$$

Next consider the third term of the TE, we can have the following equalities:

$$\frac{1}{N}\sum_{t=1}^{N}\left(\frac{\partial f^{\mathrm{T}}(x_t,\theta_0)}{\partial\theta}(\theta-\theta_0)\right)^2 = \frac{1}{N}\sum_{t=1}^{N}\mathrm{tr}\left\{(\theta-\theta_0)(\theta-\theta_0)^{\mathrm{T}}\frac{\partial f(x_t,\theta_0)}{\partial\theta}\frac{\partial f^{\mathrm{T}}(x_t,\theta_0)}{\partial\theta}\right\} = \mathrm{tr}\{(\theta-\theta_0)(\theta-\theta_0)^{\mathrm{T}}G\}. \tag{24}$$

Taking the expectation over the training set and suppose the size of the training set is large, we can again obtain an approximation for this term:

$$\left\langle\frac{1}{N}\sum_{t=1}^{N}\left(\frac{\partial f^{\mathrm{T}}(x_t,\theta_0)}{\partial\theta}(\theta-\theta_0)\right)^2\right\rangle_T = \frac{\lambda_0}{N}\mathrm{tr}\left\{\left[G+\frac{\delta}{N}I\right]^{-1}G\left[G+\frac{\delta}{N}I\right]^{-1}G\right\} = \frac{\lambda_0}{N}\sum_{k=1}^{n}\frac{\delta_k^2}{(\delta_k+\delta/N)^2}. \tag{25}$$

Therefore, taking the expectation of TE over the training set and using Eqs. (23) and (25), we can have set up a relation for λ_0 and the TE:

$$\langle TE\rangle_T = \lambda_0\left[1-\frac{n}{N}+\frac{1}{N}\sum_{k=1}^{n}\frac{(\delta/N)^2}{(\delta_k+\delta/N)^2}\right]. \tag{26}$$

Note that n is the dimension of the parametric vector θ, i.e. the total number of parameters, while N is the total number of training data. Thus, we can have the equality between the expected prediction error and the expected TE:

$$\langle\langle PE\rangle_F\rangle_T = \langle TE\rangle_T\frac{N+d_1}{N-d_2}, \tag{27}$$

where

$$d_1 = \sum_{k=1}^{n}\frac{\delta_k^2}{(\delta_k+\delta/N)^2}, \quad d_2 = \sum_{k=1}^{n}\left(1-\frac{(\delta/N)^2}{(\delta_k+\delta/N)^2}\right).$$

This is the nonlinear extension of Akaike FPE [1]. Remember that δ_k is the kth eigenvalue of the matrix G, see Eq. (11). It is worth noting that in Akaike FPE [1], which is for the linear case, $d_1 = d_2 =$ number of parameters. Here, in the nonlinear case, they are generally not the same except when δ is zero. This equation is more useful than the one derived by [21], Eq. (18), in that Eq. (27) does not require the information of the system noise variance for the estimation of the prediction error.

In case the noise variance is known, by subtracting $\langle TE\rangle_T$ from $\langle\langle PE\rangle_F\rangle_T$, we can obtain Moody's general prediction error Eq. [15]:

$$\langle\langle PE\rangle_F\rangle_T = \langle TE\rangle_T + 2\frac{\lambda_0}{N}\sum_{k=1}^{n}\frac{\delta_k}{\delta_k+\delta/N}. \tag{28}$$

In case N is large enough, we can thus use the following equation to estimate the expected prediction error:

$$\langle PE\rangle_F = TE\frac{N+\hat{d}_1}{N-\hat{d}_2}. \tag{29}$$

The values $\hat{d}_1$ and $\hat{d}_2$ can be obtained once the eigenvalues $\hat{\delta}_k$ are estimated based on the N training data.

5. Evaluation of the nonlinear FPE

Let $\hat{\theta}(0)$ be the initial parametric vector and $P^{-1}(0) = \delta I$, the training of a feedforward neural network can be accomplished by the recursive equations, Eqs. (2) and (3). As a matter of fact, Eq. (2) is rewritten in the following form (by using the matrix inversion lemma [6]).

$$P^{-1}(t) = P^{-1}(t-1) + H(t)H^{\mathrm{T}}(t). \tag{30}$$

Therefore, once after N training data has been input, $P^{-1}(N)$ is given by

$$P^{-1}(N) = \delta I + \sum_{t=1}^{N} H(t)H^{\mathrm{T}}(t). \tag{31}$$

Suppose that the initial condition $\theta(0)$ is already close to the true parameter θ_0 and N is large enough,

$$P^{-1}(N) \approx \delta I + \sum_{t=1}^{N}\frac{\partial f(x_t,\theta_0)}{\partial\theta}\frac{\partial f^{\mathrm{T}}(x_t,\theta_0)}{\partial\theta} = \delta I + NG, \tag{32}$$

where G is defined in Eq. (11). Dividing both sides by N, we have the following equations: $G = N^{-1}P^{-1}(N) - \delta N^{-1}I$. So that the value of d_1 and d_2 can readily be obtained by the following equations.

$$d_1 = \mathrm{tr}\{(I-\delta P(N))^2\}, \quad d_2 = n - \mathrm{tr}\{\delta^2 P^2(N)\}. \tag{33}$$

And thus the FPE can be estimated once training is finished:

$$\langle PE\rangle_F = TE\frac{N+\mathrm{tr}\{(I-\delta P(N))^2\}}{N-n+\mathrm{tr}\{\delta^2 P^2(N)\}}. \tag{34}$$

The major advantage of using Eq. (34) to evaluate the FPE is that no second-order Hessian matrix has to be calculated. This will save a lot of computational cost. Besides, the values of matrix $P(N)$ and TE are readily obtained once the training is finished. This again introduces not much cost on the evaluation of d_1 and d_2. Together with the effectiveness of recursive least square in training a neural network, the evaluation of a network performance become relatively simple and yet on-line.

6. Conclusions

In this paper, (i) we have derived an equation to estimate the prediction error of a neural network model which is being trained by a recursive least square method and under the assumption that the initial guess of θ is already very close to θ_0. This equation can be treated as a nonlinear extension of the Akaike final prediction error. (ii) Furthermore we have devised an elegant method, Eq. (34), based on the covariance matrix $P(N)$ and the training error evaluated after each step of update to estimate this final prediction error. As the RLS is an effective on-line training method, algorithm (34) can be implemented on-line and yet without much computational cost introduced. Without loss of generality, the result derived in this paper can directly be extended to multiple-input–multiple-output (MIMO) systems as well.

Finally, it should be remarked that there are two major differences between the final prediction error equation derived in this paper and the one derived in [9]. First, our major focus is on the RLS training approach. It is an *on-line* training method. In [9], the concern is on *batch mode* training. Second, due to the use of the RLS, we are able to derive an on-line evaluation scheme for this final prediction error and this has not been investigated in [9].

References

[1] H. Akaike, Statistical predictor identification, Ann. Inst. Stat. Math. 22 (2) (1970) 203–217.
[2] A. Barron, Prediction squared error: a criterion for automatic model selection, in: S. Farlow (Ed.), Self-Organizing Methods on Modeling, Marcel Dekker, NY, 1984.
[3] S.J. Chang, A.C.S. Leung, K.W. Wong, J. Sum, A local training and pruning approach for neural networks, Int. J. Neural Syst. 10 (6) (2000) 425–438.
[4] G.C. Goodwin, K.S. Sin, Adaptive Filtering, Prediction and Control, Printice Hall, Englewood Cliffs, NJ, 1984.
[5] Y. Iiguni, H. Sakai, H. Tokumaru, A real-time learning algorithm for a multilayered neural network based on the extended Kalman filter, IEEE Trans. Signal Process. 40 (4) (1992) 959–966.
[6] R. Johansson, System Modeling and Identification, Prentice-Hall, Englewood Cliffs, NJ, 1993.
[7] A. Kimura, I. Arizono, H. Ohta, An improvement of a back propagation algorithm by extended Kalman filter and demand forecasting by layered neural networks, Int. J. Syst. Sci. 27 (5) (1996) 473–482.
[8] S. Kollias, D. Anastassiou, An adaptive least squares algorithm for the efficient training of artificial neural networks, IEEE Trans. Circuits Syst. 36 (8) (1989) 1092–1101.
[9] J. Larsen, L.K. Hansen, Generalization performance of regularized neural network models, Proceedings of the IEEE Workshop on

Neural Networks for Signal Processing, Ermioni, Greece, vol. IV, 1994, pp. 42–51.
[10] C.S. Leung, P.F. Sum, A.C. Tsoi, L.W. Chan, Several aspects of pruning methods in recursive least square algorithms for neural networks, in: K. Wong, I. King, D.Y. Yeung (Eds.), Theoretical Aspects of Neural, Computation: A Multidisciplinary Perspective, Singapore Pte. Ltd, Lecture Notes in Computer Science, Springer, Berlin, 1997, pp. 71–80.
[11] C.S. Leung, K.W. Wong, J. Sum, L.W. Chan, On-line training and pruning for RLS algorithms, Electron. Lett. 32 (23) (1996) 2152–2153.
[12] C.S. Leung, K.W. Wong, P.F. Sum, L.W. Chan, A pruning method for recursive least squared algorithm, Neural Networks 14 (2) (2001) 147–174.
[13] C.S. Leung, G.H. Young, J. Sum, W.K. Kan, On the regularization of forgetting recursive least square, IEEE Trans. Neural Networks 10 (6) (1999) 1842–1846.
[14] L. Ljung, J. Sjöberg, T. McKelvey, On the use of regularization in system identification, Technical Report, Department of Electronic Engineering, Linkoping University, Sweden, 1992.
[15] J. Moody, The effective number of parameters: an analysis of generalization regularization in nonlinear learning systems, in: R. Lippman, et al. (Eds.), Advances in Neural Information Processing Systems, vol. 4, Morgan Kaufmann, San Mateo, CA, 1992, pp. 847–854.
[16] J. Moody, Prediction risk and architecture selection for neural networks, in: V. Cherkassky, et al. (Eds.), From Statistics to Neural Networks, Springer, Berlin, 1994.
[17] N. Murata, S. Yoshizawa, S. Amari, Network information criterion–determining the number of hidden units for an artificial neural network model, IEEE Trans. Neural Networks 5 (6) (1994) 865–872.
[18] G.V. Puskorius, L.A. Feldkamp, Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks, IEEE Trans. Neural Networks 5 (2) (1994) 279–297.
[19] S. Singhal, L. Wu, Training feed-forward networks with the extended Kalman filter, Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Glasgow, Scotland, 1989, pp. 1187–1190.
[20] S. Shah, F. Palmieri, M. Datum, Optimal filtering algorithm for fast learning in feedforward neural networks, Neural Networks 5 (5) (1992) 779–787.
[21] J. Sjöberg, L. Ljung, Overtraining, regularization and searching for a minimum, with application to neural networks, Int. J. Control 62 (6) (1995) 1391–1407.
[22] J. Sum, C.S. Leung, G.H. Young, W.K. Kan, On the Kalman filtering method in neural network training and pruning, IEEE Trans. Neural Networks 10 (1) (1999) 161–166.
[23] K. Watanabe, T. Fukuda, S.G. Tzafestas, Learning algorithms of layered neural networks via extended Kalman filters, Int. J. Syst. Sci. 22 (4) (1991) 753–768.

John Sum received his B.Eng. in Electronic Engineering from Hong Kong Polytechnic University in 1992. Then, he received his M.Phil. and Ph.D. in Computer Science and Engineering from Chinese University of Hong Kong in 1995 and 1998, respectively. Before joining the Chung Shan Medical University (Taiwan), John has spent years teaching in several universities in Hong Kong, including Hong Kong Baptist University, Open University of Hong Kong and Hong Kong Polytechnic University. Currently, he is an assistant professor of the Department of Information Management in Chung Shan Medical University (Taiwan). His current research interests include neural computation, mobile sensor networks and scale-free network.

Kevin I-J Ho received the B.S. in Computer Engineering from National Chiao Tung University (Taiwan) in 1983. From 1985 to 1987, he was an assistant engineer of Institute of Information Industry, Taiwan. Then, he received his M.S. and Ph.D. in Computer Science from University of Texas at Dallas in 1990 and 1992, respectively. Currently, he is an associate professor of Department of Computer Science & Communication Engineering, Providence University (Taiwan). His current research interests include Image Processing, Algorithm Design and Analysis, Scheduling Theory, and Computer Networks.