ISSN 0040-5795, Theoretical Foundations of Chemical Engineering, 2012, Vol. 46, No. 3, pp. 274–278. © Pleiades Publishing, Ltd., 2012.

Extended and Unscented Kalman Filters for Artificial Neural Network Modelling of a Nonlinear Dynamical System[1]

A. Saptoro

Department of Chemical Engineering, Curtin University, Miri, Sarawak, Malaysia

e-mail: agus.saptoro@curtin.edu.my

Received January 14, 2010

Abstract—Recently, artificial neural networks, especially feedforward neural networks, have been widely used for the identification and control of nonlinear dynamical systems. However, the determination of a suitable set of structural and learning parameter value of the feedforward neural networks still remains a difficult task. This paper is concerned with the use of extended Kalman filter and unscented Kalman filter based feedforward neural networks training algorithms. The comparisons of the performances of both algorithms are discussed and illustrated using a simulated example. The simulation results show that in terms of mean squared errors, unscented Kalman filter algorithm is superior to the extended Kalman filter and back-propagation algorithms since there are improvements between 2.45–21.48% (for training) and 8.35–29.15% (for testing). This indicates that unscented Kalman filter based feedforward neural networks learning could be a good alternative in artificial neural network models based applications for nonlinear dynamical systems.

DOI: 10.1134/S0040579512030074

INTRODUCTION

In recent years, artificial neural networks (ANN) have been extensively used for identification and control of nonlinear chemical processes [1–3]. One of the most popular ANN for modelling of dynamical systems is the multilayered feedforward artificial neural networks (FANN). Despite FANN is popular and capable of describing any nonlinear dynamics, its application is limited by the difficulties in the training step.

It is well-known that finding the optimal synaptic weights of FANN is a very difficult numerical problem. When being solved with classical gradient descent optimization methods on longer input sequences, it usually behaves very slowly and poorly. This is the reason for searching for the more effective methods of FANN training.

In the last four decades, Kalman filter (KF) has become popular as an algorithm for state/parameter estimation. This is because it is easy to implement and exhibit computationally efficient calculation which is especially useful for real applications [4, 5]. However, continued development in the research community has established that the original KF is often not good enough for state estimation, especially for nonlinear systems. A variant of KF to deal with nonlinear system is the extended Kalman filter (EKF). This filter is basically an extension of the KF through a linearization procedure. The EKF provides this modification by linearizing all nonlinear models so the traditional KF can be applied. Unfortunately, the EKF has two important potential drawbacks. First, the derivation of the Jacobian matrices, the linear approximators to the nonlinear functions, can be complex causing implementation difficulties. Second, these linearisations can lead to filter instability if the timestep intervals are not sufficiently small [4, 5].

To address these limitations, Julier et al. [5] proposed the unscented Kalman filter (UKF) which is then further developed by Wan et al. [6, 7]. The basic difference between the EKF and UKF is that the UKF operates on the area that it is easier to approximate a Gaussian distribution than it is to approximate an arbitrary nonlinear functions. Instead of linearizing using Jacobian matrices, the UKF using a deterministic approach to capture the mean and covariance estimate with a minimal set of sample points. Moreover, when these points are propagated through the true nonlinear system, they capture the posterior mean and covariance accurately to second order (Taylor series expansion) for any nonlinearity. The EKF, in contrast, only achieves first order accuracy. No explicit Jacobian or Hessian calculations are necessary for the UKF. Remarkably, the computational complexity of the UKF is the same order as that of the EKF [5].

This paper presents the principles of two FANN training methods based on EKF and UKF that serve a better alternative to classic gradient descent methods. Even though, the uses of EKF and UKF based FANN training algorithms have been applied in many ANN problems, their accuracy comparison with other training algorithms have not been explored thoroughly. Moreover, to prove one learning algorithm to be superior to others, even in some limited domain, a compar-

[1] The article is published in the original.

ative study of the performances of EKF, UKF, gradient based and Levenberg-Marquardt based algorithms is required. Therefore, the contributions of this work are to explore the uses of hybrid EKF–FANN and UKF–FANN to a nonlinear dynamical system and then compare their performances with popular and widely used training algorithms such as gradient descent and Levenberg–Marquardt algorithms.

EXTENDED KALMAN FILTER

Consider the following nonlinear dynamical system described by the state-space model

$$x_{k+1} = f(k, x_k) + q_k, \quad (1)$$

$$y_k = h(k, x_k) + r_k, \quad (2)$$

where x_{k+1} and y_k represent the state vector and the measurement vector, w_k and v_k are independent zero-mean white Gaussian noise processes with covariance matrices Q_k and R_k, respectively. The functional $f(k, x_k)$ denotes a nonlinear transition matrix and likewise the functional $h(k, x_k)$ denotes a nonlinear measurement matrix.

Expanding $f(k, x_k)$ and $h(k, x_k)$ using the Taylor series, neglecting higher order terms at the point of filtered values $\hat{x}$ is the essential idea of EKF and its algorithm is outlined below [5]:

Initialisation at $k = 0$

$$\hat{x}_o = E[x_o],$$

$$P_o = E\left[(x_o - E[x_o])(x_o - E[x_o]^T)\right].$$

For $k = 1, 2, ..., \infty$

(1) Projection of state estimate propagation

$$\hat{x}_k^- = f(k, \hat{x}_{k-1}).$$

(2) Projection of error covariance propagation

$$P_k^- = F_{k,k-1} P_{k-1} F_{k,k-1}^T + Q_{k-1}.$$

(3) Calculation of the Kalman gain

$$K_k = P_{k,k-1} H_k^T (H_k P_{k,k-1} H_k^T + R_k)^{-1}.$$

(4) Updating state estimation

$$\hat{x}_k = \hat{x}_k^- + K_k y_k - h(k, \hat{x}_k^-).$$

(5) Updating the error covariance

$$P_k = (I - K_k H_k) P_k^-.$$

The quantities F and H are the state transition and observation matrices, respectively and they are defined to be the following Jacobians:

$$F_{k,k-1} = \left.\frac{\partial f(k,x)}{\partial x}\right|_{x = x_{k-1}}, \quad (3)$$

$$H_k = \left.\frac{\partial h(k,x)}{\partial x}\right|_{x = x_k^-}. \quad (4)$$

Now, consider a FANN's behaviour which can be described by the following nonlinear discrete models:

$$w_{k+1} = w_k + \omega_k, \quad (5)$$

$$o_k = h_k(w_k, u_k) + \nu_k. \quad (6)$$

Eq. (5) is the process equation where the state of the system w_k is the network's weights and ω_k is the process noise. Eq. (6) represents measurement equation where o_k is the network's desired output, u_k is the input vector and v_k is the random measurement noise.

With the EKF approach, the training of the FANN can be seen as state estimation for a nonlinear system. The EKF solution to the training problem is given by the following equations:

$$A_k = (R_k + H_k^T P_k H_k)^{-1}, \quad (7)$$

$$K_k = P_k H_k A_k, \quad (8)$$

$$\bar{w}_{k+1} = \bar{w}_k + K_k \xi_k, \quad (9)$$

$$P_{k+1} = P_k - K_k H_k^T P_k + Q_k. \quad (10)$$

The vector $\bar{w}_k$ represents the estimate of the weights of the network at update step k. This estimate is a function of the Kalman gain matrix K_k and the error vector $\xi k = o_k - \hat{o}_k$, where o_k is the target vector and $\hat{o}_k$ is the network's response at the kth presentation of a training pattern.

UNSCENTED KALMAN FILTER

Consider a FANN nonlinear mapping similar to Eq. (5):

$$w_{k+1} = w_k + \mu_k, \quad (11)$$

$$d_k = G(x_k, w_k) + e_k, \quad (12)$$

where w_k denotes the network weights, μ_k is the process noise, d_k is the desired response, $G(x_k, w_k)$ is the network output and e is the error $d_k - G(x_k, w_k)$.

The algorithm of the UKF based FANN is summarised as follows [5]: Initialisation at $k = 0$

$$\hat{w}_0 = E[w],$$

$$P_{w_0} = E[(w - \hat{w}_0)(w - \hat{w}_0)^T].$$

For $k = 1, 2, ..., ..., \infty$

(1) Updating time and sigma point calculation

$$\hat{w}_k = \hat{w}_{k-1},$$

$$P_{w_k}^- = (P_{w_{k-1}} + R_{k-1}^r),$$

$$W_{k|k-1} = [\hat{w}_k^- \hat{w}_k^- + \gamma\sqrt{P_{w_k}^-} \hat{w}_k^- - \gamma\sqrt{P_{w_k}^-}],$$

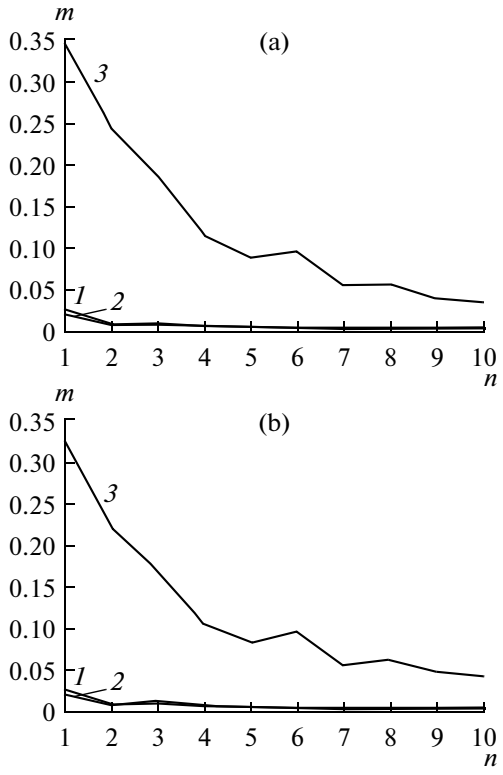

Fig. 1. Accuracy of trainekf (*1*), trainukf (*2*) and traingd (3): (a)—training process; (b)—testing process.

$$D_{k|k-1} = G(x_k, W_{k|k-1}),$$

$$d_k = G(x_k, \hat{w}_k^-).$$

(2) Updating measurement equations

$$P_{\tilde{d}k\tilde{d}k} = \sum_{i=0}^{2L} W_i^{(c)} (D_{i,k|k-1} - \hat{d}_k)(D_{i,k|k-1} - \hat{d}_k)^T + R_k^e,$$

$$P_{w_k d_k} = \sum_{i=0}^{2L} W_i^{(c)} (W_{i,k|k-1} - \hat{w}_k^-)(D_{i,k|k-1} - \hat{d}_k)^T,$$

$$K_k = P_{w_k d_k} P_{\tilde{d}k\tilde{d}k}^{-1},$$

$$\hat{w}_k = \hat{w}_k^- + K_k(d_k - \hat{d}_k),$$

$$P_{w_k} = P_{w_k}^- - K_k P_{\tilde{d}k\tilde{d}k} K_k^T.$$

In the algorithm above, $y = \sqrt{L + \lambda}$, λ is the composite scaling parameter, L is the dimension of the state, W_i is the calculated network's weights, R^r is the process noise covariance and R^e is the measurement noise covariance.

EXPERIMENTS

The main goal of our experiments was to compare the performance of four FANN training algorithms, namely gradient-descent (traingd), Levenberg–Marquardt (trainlm), EKF (trainekf) and UKF (trainukf). To achieve this goal, a synthetic benchmark problem is used as a simulation study.

The nonlinear system to be identified is expressed by

$$y = \sin(u), \tag{13}$$

where y is the output of the system and u is the system input. The system is stable at $u \in [0, 1]$. One hundred and twenty five input-output patterns were generated from the system and from this database. The datasets were divided into two subsets, 100 patterns of training data and 25 testing datasets. FANN models were then trained using the algorithms traingd, trainlm, trainekf and trainukf.

RESULTS AND DISCUSSION

The effectiveness of the algorithms was estimated by measuring the mean squared error of the training process (convergence) and the mean squared error of the testing process (generalization).

It is evident from the Fig. 1a and Fig. 1b that the performances of the trainekf and trainukf are far superior than the classical gradient-descent in both training convergence and generalization capability for any number of neurons in the hidden layer. This result appears to be meaningful as the nature of gradient–descent is very slow and poor in convergence and generalization. Closer observations of the performances of trainekf and trainukf (Fig. 2a and Fig. 2b) indicates that trainukf performs better than trainekf. This is consistent with the fact that UKF, instead of linearizing the nonlinear model using Jacobian matrices and first order Taylor series, performs on the basis of second order Taylor series, hence it is more accurate.

Interesting results are evident from the comparison of effectiveness of the trainekf, trainukf and trainlm as shown in Fig. 3a and Fig. 3b. Trainlm algorithm has much better training performance than both trainekf and trainukf. This appears to be true since trainlm is a fast algorithm which is quick to converge. Trainlm algorithm was designed based upon second order training without requiring Hessian matrix. However, in terms of generalization capability, trainlm is much inferior than trainekf and trainukf algorithms. The choice of better algorithm in this case will depend on the model developer. Nevertheless, generalization capability is more important than the training performance since the developed model may be used to deal with new patterns any time. For these reasons, trainekf and trainukf are also relatively better than the trainlm algorithm.

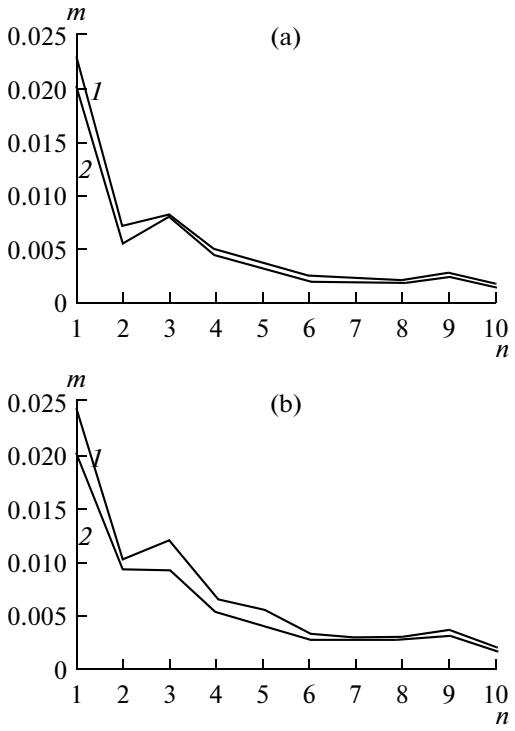

Fig. 2. Accuracy of trainekf (*1*) and trainukf (*2*): (a)—training process; (b)—testing process.

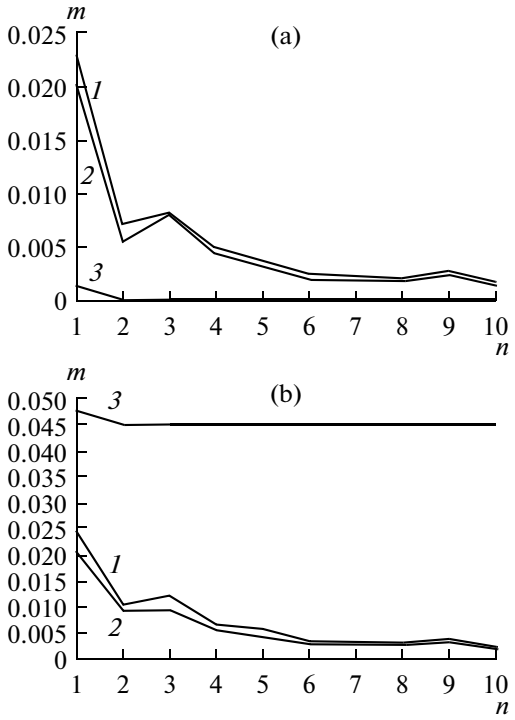

Fig. 3. Accuracy of trainekf (*1*), trainukf (*2*) and trainlm (*3*): (a)—training process; (b)—testing process.

CONCLUSIONS

We described thoroughly the basic principle of the EKF and UKF in this paper. The main contribution of this paper is the direct comparison of the effectiveness of the trainekf, trainukf, traingd and trainlm when they are used in FANN training of a nonlinear model. Results from the experiments on trainekf and trainukf algorithms indicate better performances compared to the classic backpropagation such as gradient descent (traingd). Trainukf algorithm itself shows superior performance than trainekf. On the other hand, compared with trainlm, trainekf and trainukf are inferior in the training performance but superior on their generalization capability. Based on these results, it can be concluded that trainukf (and also trainekf) appears to be a good alternative for training FANN model based chemical processes since there are improvements between 2.45–21.48% for training process and 8.35–29.15% for testing processes. A further study on the comparison of the efficiency of each training algorithms is required. This can be done by observing and comparing computational time requirements for several of the algorithms tested in this investigation.

NOTATION

D—desired response matrix;
d—desired response vector;
E—mean;
F—transition matrix;
f—nonlinear transition matrix function;
G—network's output matrix;
H—measurement matrix;
h—nonlinear measurement matrix function;
K—Kalman gain matrix or blending factor;
L—dimension of the state;
m—mean squared error;
n—number of hidden nodes;
o—network's desired output;
$\hat{o}$—network's response;
P—error covariance matrix of the state;
P^-—a priori estimate of the error covariance matrix;
Q—covariance matrix for process noise q;
q—independent zero–mean white Gaussian process noise;
R—covariance matrix for measurement noise r;
R^e—measurement noise covariance;
R^r—process noise covariance;
r—independent zero–mean white Gaussian measurement noise;
u—network's input vector;
v—network's random measurement noise;
W—calculated network's weights;
w—network's weights;
$\hat{W}$—weight estimate vector;

$\hat{W}^-$—a priori weight estimate vector;

x—state vector;

$\hat{x}$—state estimate vector;

$\hat{x}^-$—a priori state estimate vector;

λ—composite scaling parameter;

μ—process noise for artificial neural network model (extended Kalman filter training algorithm);

ξ—error vector;

ω—process noise for artificial neural network model (unscented Kalman filter training algorithm).

SUBSCRIPTS AND SUPERSCRIPTS

0—initial value;

i—index of state/network's weight;

k—discrete time/presentation of a training pattern.

REFERENCES

1. Quantrille, T.E. and Liu, Y.A., *Artificial Intelligence in Chemical Engineering*, Orlando: Academic, 1992.
2. Hussain, M.A., Review of the Applications of Neural Networks in Chemical Process Control—Simulation and Online Implementation, *Artif. Intell. Eng.*, 1999, vol. 13, p. 55.
3. Willis, M.J., Montague, G.A., Di Massimo, C., et al., Artificial Neural Networks in Process Estimation and Control, *Automatica*, 1992, vol. 28, no. 6, p. 1181.
4. Trebaticky, P., Recurrent Neural Network Training with the Extended Kalman Filter, *Proc. Student Research Conf. in Informatics and Information Technologies*, Bratislava, 2005, p. 57.
5. Haykin, S., *Kalman Filtering and Neural Networks*, New York: Wiley, 2001.
6. Wan, E.A., van der Merwe, R., and Nelson, A.T., Dual Estimation and the Unscented Transformation, *Advances in Neural Information Processing Systems*, Solla S.A., Leen T.K., and Muller K.R., Eds., Cambridge: MIT, 2000, vol. 12, p. 666.
7. Wan, E.A. and van der Merwe, R., The Unscented Kalman Filter for Nonlinear Estimation, *Proc. IEEE Symp. 2000 on Adaptive Systems for Signal Processing, Communication and Control*, Lake Louise, Alberta, Canada, 2000, p. 153.