Journal of Intelligent and Robotic Systems 3: 305–319, 1990.

Learning Algorithms for Neural Networks with the Kalman Filters

KEIGO WATANABE
*College of Engineering, Shizuoka University, Johoku 3-5-1, Hamamatsu 432, Japan**

SPYROS G. TZAFESTAS
Division of Computer Science, Department of Electrical Engineering, National Technical University of Athens, Zografou 15773, Athens, Greece

(Received: 1 February 1990)

Abstract. Based on various approaches, several different learning algorithms have been given in the literature for neural networks. Almost all algorithms have constant learning rates or constant accelerative parameters, though they have been shown to be effective for some practical applications. The learning procedure of neural networks can be regarded as a problem of estimating (or identifying) constant parameters (i.e. connection weights of network) with a nonlinear or linear observation equation. Making use of the Kalman filtering, we derive a new back-propagation algorithm whose learning rate is computed by a time-varying Riccati difference equation. Perceptron-like and correlational learning algorithms are also obtained as special cases. Furthermore, a self-organising algorithm of feature maps is constructed within a similar framework.

Key words. Neural nets, nonlinear filtering, Kalman filters, parameter estimation, learning systems, pattern recognition.

1. Introduction

In the study of artifical neural networks, one important problem is to develop an efficient learning algorithm (Lippmann, 1987). In general, the neural network can be viewed as a black box that learns to map input patterns to appropriate output patterns. The learning is accomplished by modifying the connection weights of neurons in such a way as to improve the desired mapping between input and output patterns.

There are three classes of learning rules (Matheus and Hohensee, 1987). The first class comprises the *correlational rules* in which individual weight changes are solely determined on the basis of levels of activity between connected units, as can be seen from the Hebbian learning (Hebb, 1949) or the Hopfield network (Hopfield, 1982). The second, which we call *error-correcting rules*, depends on external feedback about the desired signals of the output units (Widrow and Hoff, 1960; Rumelhart *et al.*, 1986a, b). The third class is the set of *unsupervised learning rules*, where learning occurs as self-adaptation to detected regularities in the input space, without direct feedback from a teacher or supervisor (Kohonen, 1982, 1987).

* Present address: Department of Mechanical Engineering, Faculty of Science and Engineering, Saga University, Honjomachi-1, Saga 840, Japan

It should be noted that the error-correcting and unsupervised learning rules are usually developed by using different ways, e.g. the Widrow–Hoff rule (Widrow and Hoff, 1960) that falls into the error-correcting rules is derived from the use of the gradient descent method, whereas Kohonen's self-organising algorithm (Kohonen, 1982, 1987), for feature maps, which is a similar form of unsupervised learning, is developed by making use of a modified Hebbian rule. Furthermore, the learning rate in almost all algorithms is constant, and a rational method for computing the time-varying version does not exist.

In this paper, the problem of learning neural networks is viewed as a problem of estimating constant parameters within the framework of Kalman filtering theory (Anderson and Moore, 1979). The well-developed Kalman filtering theory provides us with a unified treatment of computing the time-varying learning rate.

In Section 2 we review the back-propagation algorithm due to Rumelhart *et al.* (1986a, b), which is a representative of the error-correcting rules and is known as a powerful tool for attacking pattern recognitions like speech recognition (Plaut and Hinton, 1987; Gorman and Sejnowski, 1988) and learning controls for robotic manipulators (Kawato *et al.*, 1988). In Section 3, by using the extended Kalman filters, we provide a new back-propagation algorithm whose learning rate can be computed by a time-varying Riccati difference equation. In Section 4 we shall show how the perceptron-like learning and correlational rules can be derived, as special cases of the back-propagation algorithm, by using linear Kalman filters. We further extend the use of linear Kalman filtering to derive a self-organising algorithm for feature maps in Section 5. Finally, in Section 6, we give a simulation example using back-propagation algorithms.

2. Back-Propagation Algorithm via the Gradient Descent Method

In this section, we shall review the back-propagation algorithm which is a representative of the error-correcting rules. For a comprehensive discussion, the reader is referred to Rumelhart *et al.* (1986a, b) and Hinton (1987).

2.1. NEURAL MODEL

Consider a layered feedforward-type neural network. The network consists of M layers, in which the first layer denotes the input, the last M layer is the output, and the other layers are intermediate (or hidden) layers. It is assumed that the $(k-1)$th layer has N_{k-1} units.

The model of the network considered in Figure 1 is based on the following equations

$$i_j^k = \sum_{i=1}^{N_{k-1}} w_{ij}^{k-1,k} o_i^{k-1} + \theta_j^k, \tag{1}$$

$$o_j^k = f(i_j^k), \tag{2}$$

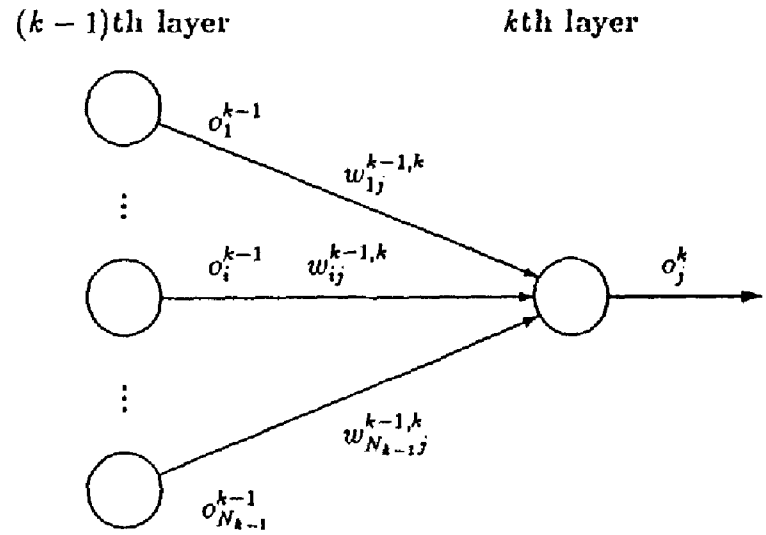

Fig. 1. Neural model.

where o_i^k represents the output of the ith neuron at the kth layer, $w_{ij}^{k-1,k}$ denotes the connection strength (or weight) from the ith neuron at the $(k-1)$ layer to the jth neuron at the kth layer, θ_j^k is a bias or the threshold of the jth neuron at the kth layer, and i_j^k is the total input to the jth neuron at the kth layer. The output of the jth neuron at the kth layer is generated through a nonlinear function given by the following logistic formula,

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

2.2. LEARNING PROCEDURE

The purpose of the learning procedure is to find a set of weights such that, when the network is presented with each input vector, the output vector produced by the network is the same as the desired output vector. If there is a fixed, finite set of input-output cases, the total error in the performance of the network with a particular set of weights can be computed by comparing the actual and desired output vectors for every case. The total error, J, is defined by

$$J = \frac{1}{2} \sum_p \sum_{j=1}^{N_M} (o_{jp}^M - y_{jp})^2, \tag{4}$$

where p is an index over cases (input-output pairs) and y is the desired state of an output unit.

A procedue which is called the *generalized delta rule* or *back-propagation algorithm* (Rumelhart *et al.*, 1986a, b) is applicable for networks with trainable hidden units. The back-propagation algorithm can be summarised in three equations as

follows:

$$\hat{w}_{ij}^{k-1,k} = \hat{w}_{ij}^{k-1,k} - \varepsilon\delta_j^k o_i^{k-1}, \tag{5}$$

$$\delta_j^M = (o_j^M - y_j) f'(i_j^M), \tag{6}$$

$$\delta_l^k = f'(i_l^k) \sum_{l=1} \hat{w}_{jl}^{k,k+1} \delta_l^{k+1}, \quad k = M-1, \ldots, 2, \tag{7}$$

where Equation (5) is the estimation equation of weights for any set of input output patterns, in which ε is a small constant and is called the *learning rate*. Equations (6) and (7) specify the error signal. If a unit is an output unit, its error signal is given by (6) as the standard delta rule, in which $f'(\cdot)$ is the derivative of the activation function which is given for (3) by

$$f'(i_j^k) = o_j^k(1 - o_j^k). \tag{8}$$

The error signal for hidden units for which there is no specified target is determined recursively in terms of the error signals of the units to which it is directly connected and the weights of those connections.

This learning procedure is based on the gradient descent method and requires only that the change in weight be proportional to $\partial E_p/\partial w$, i.e.

$$\Delta w_{ij}^{k-1,k} \propto -\varepsilon \partial E_p/\partial w, \tag{9}$$

where

$$E_p \triangleq \frac{1}{2} \sum_j (o_{jp}^M - y_{jp})^2. \tag{10}$$

Since the learning rate is constant, the larger this constant, the larger the changes in weights. For practical purposes, we choose a learning rate that is as large as possible without leading to oscillation. One way to increase the learning rate without leading to oscillation is to modify (9) to include a momentum term. This is given by the following rule

$$\Delta w_{ij}^{k-1,k}(t) = -\varepsilon\delta_j^k o_i^{k-1} + \alpha \Delta w_{ij}^{k-1,k}(t-1), \tag{11}$$

where t is the current iteration number and α is a constant valued in $0 < \alpha < 1$ which determines the effect of past weight changes on the current direction of movement in the weight space. Note here that the bias θ_j^k can also be learned by applying a similar equation to (11), i.e.

$$\hat{\theta}_j^k = \hat{\theta}_j^k + \Delta\theta_j^k(t), \tag{12}$$

$$\Delta\theta_j^k(t) = -\varepsilon\delta_j^k + \alpha\Delta\theta_j^k(t-1), \tag{13}$$

where we have imagined that θ_j^k is the weight from a unit that is always on, $o_i^{k-1} \equiv 1$.

3. Back-Propagation Algorithm via the Extended Kalman Filter

We have reviewed how the back-propagation algorithm essentially implements gradient descent in sum-squared error. It should be noted, however, that the learning rate is constant, so we may have to consume more time to obtain a sufficiently converged result, even though we can take into account a momentum term.

In this section, we regard the learning of network as an estimation (or identification) problem of constant parameters. As a result, we shall show how a time-varying learning rate can be computed by using the Kalman filter.

3.1. FOR THE LAST LAYER

We first consider the estimation of weights between the $(M-1)$th layer and the Mth layer. Consider the following model with a nonlinear observation equation:

$$w_{ij}^{M-1,M}(t+1) = w_{ij}^{M-1,M}(t) + \xi(t), \tag{14}$$

$$y_j(t+1) = o_j^M(t+1) + \eta(t+1), \tag{15}$$

where $\{\xi(t), \eta(t)\}$ are mutually independent, zero-mean white Gaussian noises with variances Q and R. Note that they can be considered as pseudo-noises for tuning the gains of the extended Kalman filter. To obtain the filtered state $\hat{w}_{ij}^{M-1,M}(t+1|t+1)$, the nonlinear function $o_j^M(t)$ in (15) must be expanded in Taylor series around the predicted state $\hat{w}_{ij}^{M-1,M}(t+1|t)$ with terms up to first order to yield the first-order extended Kalman filter. The expression with first-order terms is

$$o_j^M = o_j^M(\hat{w}_{ij}^{M-1,M}(t)) + H_{ij}^{M-1,M}(w_{ij}^{M-1,M} - \hat{w}_{ij}^{M-1,M}(t)) + \cdots, \tag{16}$$

$$H_{ij}^{M-1,M} \triangleq \left.\frac{\partial o_j^M}{\partial w_{ij}^{M-1,M}}\right|_{w_{ij}^{M-1,M}=\hat{w}_{ij}^{M-1,M}(t)} = \left.\frac{\partial o_j^M}{\partial i_j^M}\cdot\frac{\partial i_j^M}{\partial w_{ij}^{M-1,M}}\right|_{w_{ij}^{M-1,M}=\hat{w}_{ij}^{M-1,M}(t)} = f'(i_j^M)o_i^{M-1}, \tag{17}$$

where

$$\hat{w}_{ij}^{M-1,M}(t) \triangleq \hat{w}_{ij}^{M-1,M}(t+1|t) \equiv \hat{w}_{ij}^{M-1,M}(t|t)$$

as can be seen from (14). The filtered estimate of $w_{ij}^{M-1,M}$ at $t+1$ is therefore obtained by neglecting the highest-order terms

$$\hat{w}_{ij}^{M-1,M}(t+1) = \hat{w}_{ij}^{M-1,M}(t) - \frac{P_{ij}^{M-1,M}(t+1|t)f'(i_j^M)o_i^{M-1}[o_j^M - y_j]}{\{P_{ij}^{M-1,M}(t+1|t)[H_{ij}^{M-1,M}]^2 + R\}} \tag{18}$$

$$P_{ij}^{M-1,M}(t+1|t) = P_{ij}^{M-1,M}(t|t) + Q, \tag{19}$$

$$P_{ij}^{M-1,M}(t+1|t+1) = \left[1 - \frac{P_{ij}^{M-1,M}(t+1|t)[H_{ij}^{M-1,M}]^2}{\{P_{ij}^{M-1,M}(t+1|t)[H_{ij}^{M-1,M}]^2 + R\}}\right] P_{ij}^{M-1,M}(t+1|t), \tag{20}$$

with initial conditions $\hat{w}_{ij}^{M-1,M}(0) = \bar{w}_{ij}^{M-1,M}$ and $P_{ij}^{M-1,M}(0|0) = P_{ij}^{M-1,M}(0)$. It is worthwhile noting that the estimation equation (18) can be written as

$$\hat{w}_{ij}^{M-1,M}(t+1) = \hat{w}_{ij}^{M-1,M}(t) - \varepsilon_{ij}^{M-1,M}(t)(o_i^M - y_i)f'(i_j^M)o_i^{M-1}, \tag{21}$$

where

$$\varepsilon_{ij}^{M-1,M}(t) \triangleq \frac{P_{ij}^{M-1,M}(t+1|t)}{P_{ij}^{M-1,M}(t+1|t)[H_{ij}^{M-1,M}]^2 + R}, \tag{22}$$

and that

$$\varepsilon_{ij}^{M-1,M}(t)H_{ij}^{M-1,M} \equiv K_{ij}^{M-1,M}(t)\text{: Kalman gain.} \tag{23}$$

3.2. FOR HIDDEN LAYERS

We further extend the results obtained previously to the estimation of weights of hidden units. Consider similar models to (14) and (15) described by

$$w_{ij}^{k-1,k}(t+1) = w_{ij}^{k-1,k}(t) + \xi(t), \tag{24}$$

$$y_j^k(t+1) = o_j^k(t+1) + \eta(t+1), \tag{25}$$

where y_j^k denotes the fictitious desired output of an hidden unit.

In a way similar to that used in Section 3.1, the filtered estimates of $w_{ij}^{k-1,k}$, $k = M-1, \ldots, 2$, at $t+1$ are obtained by the following extended Kalman filters:

$$\hat{w}_{ij}^{k-1,k}(t+1) = \hat{w}_{ij}^{k-1,k}(t) - \varepsilon_{ij}^{k-1,k}(t)(o_j^k - y_j^k)f'(i_j^k)o_i^{k-1}, \tag{26}$$

$$P_{ij}^{k-1,k}(t+1|t) = P_{ij}^{k-1,k}(t|t) + Q, \tag{27}$$

$$H_{ij}^{k-1,k} = f'(i_j^k)o_i^{k-1}, \tag{28}$$

$$\varepsilon_{ij}^{k-1,k}(t) = \frac{P_{ij}^{k-1,k}(t+1|t)}{P_{ij}^{k-1,k}(t+1|t)[H_{ij}^{k-1,k}]^2 + R}, \tag{29}$$

$$P_{ij}^{k-1,k}(t+1|t+1) = \{1 - \varepsilon_{ij}^{k-1,k}(t)[H_{ij}^{k-1,k}]^2\}P_{ij}^{k-1,k}(t+1|t), \tag{30}$$

with initial conditions

$$\hat{w}_{ij}^{k-1,k}(0) = \bar{w}_{ij}^{k-1,k} \quad \text{and} \quad P_{ij}^{k-1,k}(0|0) = P_{ij}^{k-1,k}(0).$$

It should be noted, however, that the target state y_j^k at any hidden unit is actually not specified by the task. So, we must consider the replacement of y_j^k by any other available state. Fortunately, as can be seen from the results of the generalised delta rule, $(o_j^k - y_j^k)f'(i_j^k)$ in (26) can be replaced with δ_j^k given by (7). In such a case, the estimates of $w_{ij}^{k-1,k}$ are obtained by

$$\hat{w}_{ij}^{k-1,k}(t+1) = \hat{w}_{ij}^{k-1,k}(t) - \varepsilon_{ij}^{k-1,k}(t)\delta_j^k o_i^{k-1}. \tag{31}$$

Consequently, Box 1 summarizes the learning algorithm for weights via the extended Kalman filter. Also, Box 2 presents the learning algorithm for biases via the extended Kalman filter, where the error signal δ_j^k is given in Box 1.

Error Propagations:

$$\delta_j^k = o_j^k(1 - o_j^k) \sum_{l=1}^{N_l} \hat{w}_{lj}^{k,k+1}\delta_l^{k+1}, \quad k = M - 1, \ldots, 2,$$

$$\delta_j^M = (o_j^M - y_j)o_j^M(1 - o_j^M)$$

Learning Rates:

$$P_{ij}^{k,k}(t + 1|t) = P_{ij}^{k-1,k}(t|t) + Q,$$

$$P_{ij}^{k-1,k}(0|0) = P_{ij}^{k-1,k}(0),$$

$$H_{ij}^{k,k} = o_j^k(1 - o_j^k)o_i^{k-1},$$

$$\varepsilon_{ij}^{k-1,k}(t) = \frac{P_{ij}^{k-1,k}(t + 1|t)}{P_{ij}^{k-1,k}(t + 1|t)[H_{ij}^{k-1,k}]^2 + R}$$

$$P_{ij}^{k-1,k}(t + 1|t + 1) = \{1 - \varepsilon_{ij}^{k-1,k}(t)[H_{ij}^{k-1,k}]^2\}P_{ij}^{k,k}(t + 1|t)$$

Weight Estimates:

$$\hat{w}_{ij}^{k-1,k}(t + 1) = \hat{w}_{ij}^{k-1,k}(t) - \varepsilon_{ij}^{k-1,k}(t)\delta_j^k o_i^{k-1},$$

$$\hat{w}_{ij}^{k,k}(0) = \bar{w}_{ij}^{k-1,k}$$

Box 1. Back-propagation algorithm for weights via the extended Kalman filter

Learning Rates:

$$P_j^k(t + 1|t) = P_j^k(t|t) + Q, \quad P_j^k(0|0) = P_j^k(0),$$

$$H_j^k = o_j^k(1 - o_j^k),$$

$$\varepsilon_j^k(t) = \frac{P_j^k(t + 1|t)}{P_j^k(t + 1|t)[H_j^k]^2 + R},$$

$$P_j^k(t + 1|t + 1) = \{1 - \varepsilon_j^k(t)[H_j^k]^2\}P_j^k(t + 1|t)$$

Bias Estimates:

$$\hat{\theta}_j^k(t + 1) = \hat{\theta}_j^k(t) - \varepsilon_j^k(t)\delta_j^k,$$

$$\hat{\theta}_j^k(0) = \bar{\theta}_j^k$$

Box 2. Back-propagation algorithm for biases via the extended Kalman filter

Note that, as can be seen from the linearized observation matrix (17) (or 28) together with (8), the present learning algorithm often includes an unobservable estimation problem for the case when the input data consists of binary values 0 and 1. This implies that the problem becomes a mixed estimation problem including pure prediction and filtering. From this fact, to overcome the singularity in computing the time-varying learning rate, we will select $R = 1.0$ in a later simulation such as a recursive least-squares method.

4. Some Special Cases

4.1. PERCEPTRON-LIKE LEARNING ALGORITHM

In this subsection, we consider a network consisting of two layers indexed by $M-1$ and M, which is often called *a single perceptron*, where the layer indexed by $M-1$ can be regarded as an input layer. Furthermore, suppose that the model of the network is described by the following linear system

$$w_{ij}^{M-1,M}(t+1) = w_{ij}^{M-1,M}(t) + \xi(t), \tag{32}$$

$$y_j(t+1) = o_j^M(t+1) + \eta(t+1), \tag{33}$$

where

$$o_j^M(t+1) = \sum_{i=1}^{N_{M-1}} w_{ij}^{M-1,M}(t+1)o_i^M + \theta_j^M. \tag{34}$$

In this case, $f'(i_j^M) \equiv 1$, so that $H_{ij}^{M-1,M} = o_i^{M-1} \equiv i_i^{M-1}$. Consequently, we have the following recursive equation for estimating $w_{ij}^{M-1,M}$:

$$\begin{aligned} \hat{w}_{ij}^{M-1,M}(t+1) &= \hat{w}_{ij}^{M-1,M}(t) + \varepsilon_{ij}^{M-1,M}(t)(y_j - o_j^M)o_i^{M-1}, \\ \hat{w}_{ij}^{M-1,M}(0) &= \bar{w}_{ij}^{M-1,M}, \end{aligned} \tag{35}$$

where

$$\begin{aligned} P_{ij}^{M-1,M}(t+1|t) &= P_{ij}^{M-1,M}(t|t) + Q, \\ P_{ij}^{M-1,M}(0|0) &= P_{ij}^{M-1,M}(0), \end{aligned} \tag{36}$$

$$\varepsilon_{ij}^{M-1,M}(t) = \frac{P_{ij}^{M-1,M}(t+1|t)}{P_{ij}^{M-1,M}(t+1|t)[o_i^{M-1}]^2 + R}, \tag{37}$$

$$P_{ij}^{M-1,M}(t+1|t+1) = \{1 - \varepsilon_{ij}^{M-1,M}(t)[o_i^{M-1}]^2\}P_{ij}^{M-1,M}(t+1|t). \tag{38}$$

This can be looked upon as an extension of the Widrow–Hoff rule or least mean square (LMS) algorithm (Widrow *et al.*, 1976), or orthogonal learning (Amari, 1978). Note that the well-known perceptron convergence procedure is identical to the LMS algorithm, except the linear activation function is replaced by the hard limiter.

4.2. CORRELATIONAL RULE

We can further derive the correlational rules as a special case of the perceptron-like learning algorithm. The correlational rules are also classified into two types depending on whether a teacher is present or not.

(a) With teacher

When assuming that the innovation $(y_j - o_j^M)$ in (35) is equal to y_j, we can formally obtain the following algorithm

$$\begin{aligned} \hat{w}_{ij}^{M-1,M}(t+1) &= \hat{w}_{ij}^{M-1,M}(t) + \varepsilon_{ij}^{M-1,M}(t)y_j o_i^{M-1}, \\ \hat{w}_{ij}^{M-1,M}(0) &= \bar{w}_{ij}^{M-1,M}, \end{aligned} \tag{39}$$

where $\varepsilon_{ij}^{M-1,M}(t)$ is given by Equations (36)–(38). This is a correlational learning algorithm (Amari, 1978) having a time-varying learning rate when the teacher is present.

(b) Without teacher

By replacing the innovation $(y_j - o_j^M)$ by o_j^M, we further obtain the following Hebbian learning algorithm:

$$\hat{w}_{ij}^{M-1,M}(t+1) = \hat{w}_{ij}^{M-1,M}(t) + \varepsilon_{ij}^{M-1,M}(t) o_i^M o_i^{M-1},$$
$$\hat{w}_{ij}^{M-1,M}(0) = \bar{w}_{ij}^{M-1,M}, \tag{40}$$

where $\varepsilon_{ij}^{M-1,M}(t)$ is also given by Equations (36)–(38).

It should be noted here that Equations (39) and (40) are suboptimal, because the replacement of $(y_j - o_j^M)$ by y_j or o_j^M does not assure any longer that

$$E[(y_j - o_j^M)^2] = P_{ij}^{M-1,M}(t+1|t)[o_i^{M-1}]^2 + R$$

for any i.

5. Self-Organising Algorithm of Feature Maps

In this section, we consider a learning in artificial neural systems involving self-organisation in a completely unsupervised environment.

The neural responses are ordered spatially, different locations of the responses representing different feature values of the sensory signals (Kohonen, 1987). Kohonen demonstrated that such an order can be formed automatically in a self-organising process. Weights of the network gradually come to represent qualities or features of

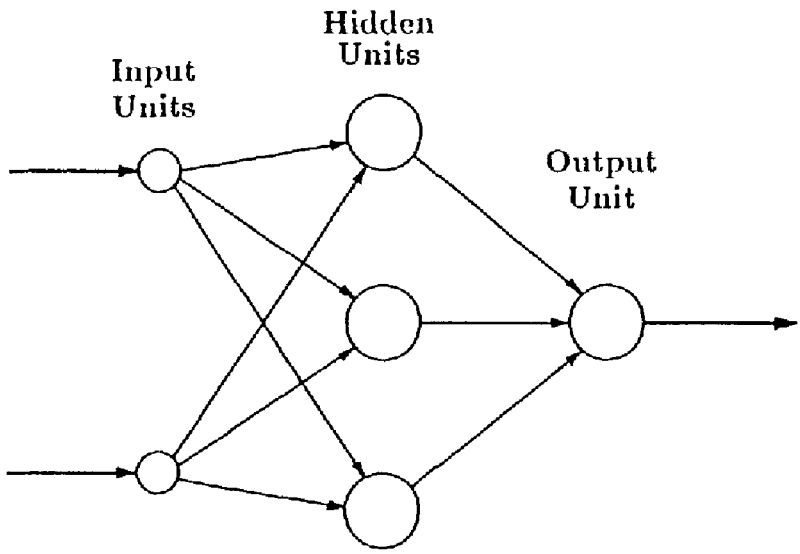

Fig. 2. A XOR network with three hidden units.

the input stream in such a way that topologically close units within the network respond similarly to similar input examples.

His algorithm creates a vector quantiser by adjusting weights w_{ij} from common inputs nodes to M output nodes arranged in a two-dimensional grid. During learning, a distance measure, d_j, is computed by using

$$d_j = \sum_{i=0}^{N-1} (x_i(t) - w_{ij}(t))^2, \tag{41}$$

where $x_i(t)$ is the input to node j at time t and $w_{ij}(t)$ is the weight from input node i to output node j at time t. The output node j^* with minimum distance is selected as a winner. Weights are updated for node j^* and all nodes in the neighbourhood defined by N_c, where N_c has a radius c from the centre unit j^*, and it starts large and slowly decreases in size over time. New weights are updated as follows:

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)[x_i(t) - w_{ij}(t)], \quad \text{for} \quad j \in N_c, \tag{42a}$$

$$w_{ij}(t+1) = w_{ij}(t) \quad \text{for} \quad j \notin N_c, \tag{42b}$$

where a gain $\eta(t)$ is taken to be $0 < \eta(t) < 1$ that decreases in time.

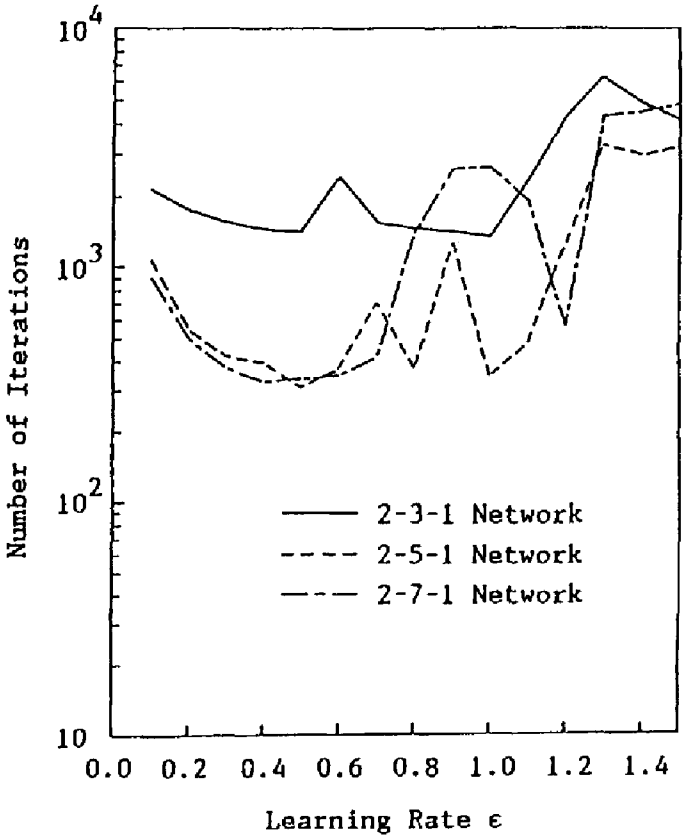

Fig. 3. The results of the number of iterations against the learning rate ε when the gradient descent method is applied.

It is interesting to note that the algorithm is linear with respect to weights. In fact, Kohonen derived the algorithm from the modified Hebbian law. However, it should be noted that, the gain $\eta(t)$ cannot be determined uniquely, but rather it is determined experimentally (Kohonen, 1986).

In this section, we regard the learning for the self-organising feature maps as a problem of estimating w_{ij} by using the 'observations' $x_i(t)$. Namely, we consider the following model with a linear observation equation:

$$w_{ij}(t+1) = w_{ij}(t) + \xi(t), \tag{43}$$

$$x_i(t+1) = w_{ij}(t+1) + \eta(t+1), \tag{44}$$

where $\{\xi(t), \eta(t)\}$ are defined in Section 3.

When applying the linear Kalman filtering to this model, we readily obtain the following recursive equations:

$$\hat{w}_{ij}(t+1) = \hat{w}_{ij}(t) + \eta_{ij}(t)[x_i(t+1) - \hat{w}_{ij}(t)], \quad \text{for } j \in N_c, \tag{45a}$$

$$\hat{w}_{ij}(t+1) = \hat{w}_{ij}(t), \quad j \notin N_c, \tag{45b}$$

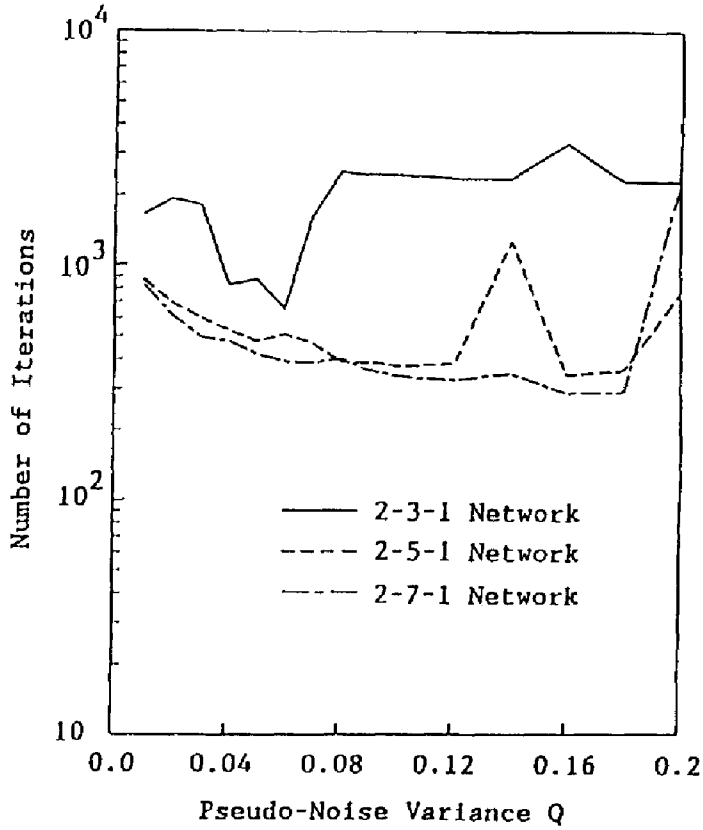

Fig. 4. The results of the number of iterations against the pseudo-noise variance Q when the extended Kalman filter method is applied.

where $\hat{w}_{ij}(0) = \bar{w}_{ij}$ and the Kalman gain $\eta_{ij}(t)$ is determined by

$$P_{ij}(t+1|t) = P_{ij}(t|t) + Q, \quad P_{ij}(0|0) = P_{ij}(0), \tag{46}$$

$$\eta_{ij}(t) = \frac{P_{ij}(t+1|t)}{P_{ij}(t+1|t) + R}, \tag{47}$$

$$P_{ij}(t+1|t+1) = [1 - \eta_{ij}(t)]P_{ij}(t+1|t), \quad \text{for } j \in N_c, \tag{48a}$$

$$P_{ij}(t+1|t+1) = P_{ij}(t+1|t), \quad \text{for } j \notin N_c. \tag{48b}$$

It is evident from Equation (47) that the range on the Kalman gain is $0 \leqslant \eta_{ij}(t) \leqslant 1$.

6. A Numerical Example

In this section, we simulate the classical XOR problem of Minsky and Papert (1988) by applying the back-propagation algorithms. We want to assign the input patterns to the output patterns as tabulated in Table I. We use a three-layered network having one hidden layer. Figure 2 shows one of the networks in which there are three hidden

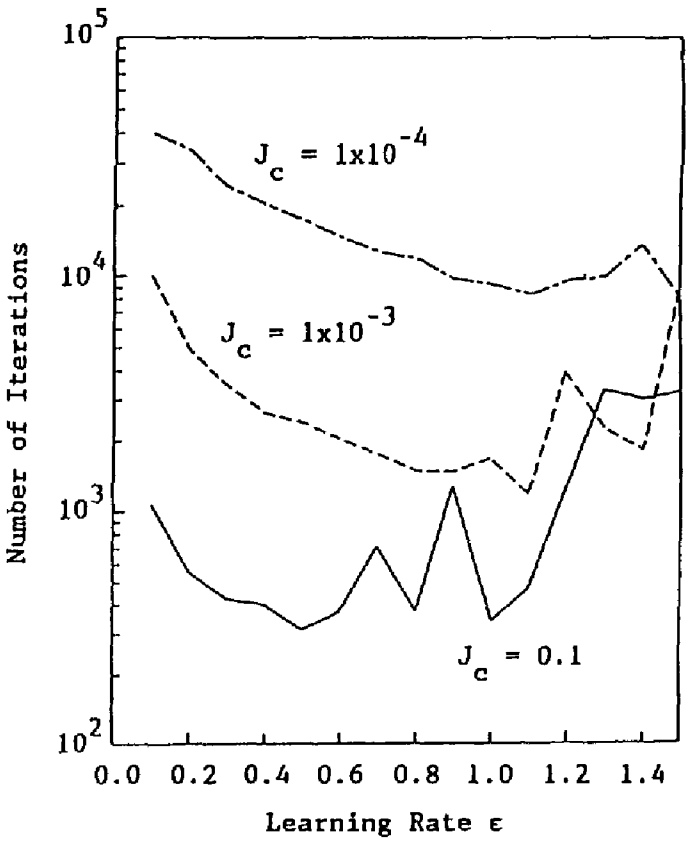

Fig. 5. The results of the number of iterations against the learning rate ε when the gradient descent method is applied for a 2-5-1 network.

Table I. XOR problem

Input patterns			Output patterns
0	0	→	0
0	1	→	1
1	0	→	1
1	1	→	0

units, where it is assumed that the biases at the input layer are set to be zero and other biases at any layers are set to be learned. Further, suppose that a set of input-output pattern is randomly presented to the network, and that we use a way of updating the weights after every input-output case (i.e. nonaccumulate way).

Figure 3 shows the results of the number of iterations against the learning rate ε when the gradient descent method is applied, where the initial weights were generated by the Gaussian random number subject to $N(0, Q_0^2)$, $Q_0 = 1.0$, and the results were averaged over 10 runs. Note, here, that in the works of Rumelhart *et al.* (1986a, b) the uniformly random numbers were used as initial weights. Further, the following

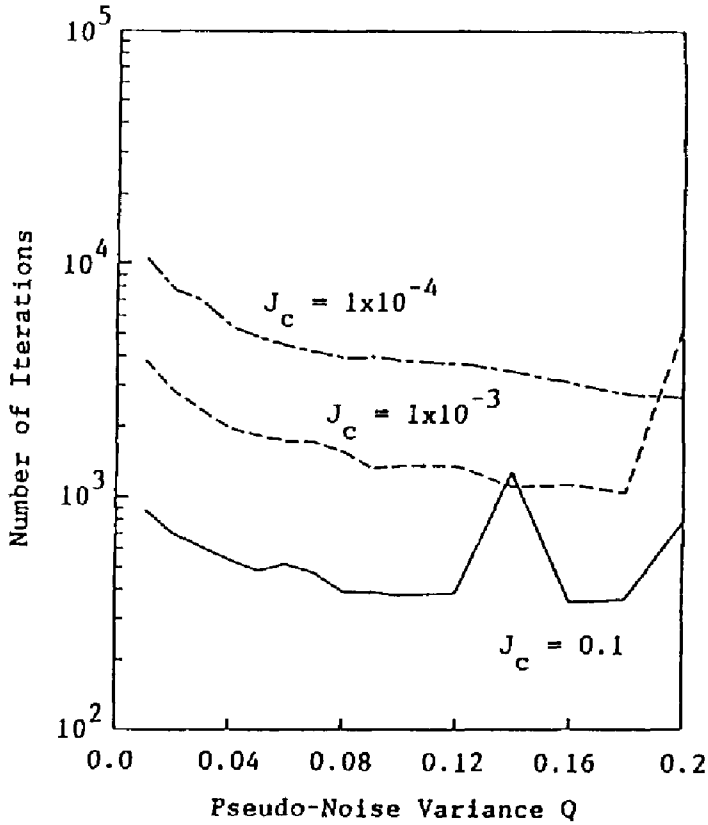

Fig. 6. The results of the number of iterations against the pseudo-noise variance Q when the extended Kalman filter method is applied for a 2-5-1 network.

conditions were set up: momentum factor $\alpha = 0.9$, truncated convergence error $J_c = 0.1$, and the upper limit of iteration was 1×10^4. It is observed from this figure that good performances are obtained at approximately $\varepsilon = 0.2 \sim 0.5$, and that the convergence speed of learning can be faster when increasing the number of hidden units.

Figure 4 shows the results of the number of iterations against Q due to the extended Kalman filter method, where the same conditions as used above were set, except that $R = 1.0$ and $P_0 = 10$. Observe from this figure that, for a 2-3-1 network, the present method gives a better performance at $Q = 0.04 \sim 0.06$ than the gradient descent method, while for 2-5-1 an 2-7-1 networks, the present method requires $300 \sim 400$ iterations to reach the desired convergence.

Figures 5 and 6 depict a comparison of both methods in changes of the truncated convergence error J_c for a 2-5-1 network, where it is assumed that, for cases $J_c = 1 \times 10^{-3}$ and $J_c = 1 \times 10^{-4}$, the upper limit of iteration is 4×10^4 and other conditions are the same as those used in Figures 3 and 4. It is found from these figures that the convergence speed of the present method is faster than that of the gradient descent method, if a sufficiently converged result is desired.

7. Conclusions

We have studied the learning algorithms for neural networks from the viewpoint of estimation (or identification) theory. The Kalman filtering approach has been adopted to estimate the connection weights of a network with a linear or nonlinear observation equation. This approach allows us to optimally or suboptimally compute a time-varying learning rate for any learning rule, whereas almost traditional algorithms have constant learning rates.

References

Amari, S (1978) Neural theory of association and concept-formation, *Biological Cybernetics* **26**, 175–185.

Anderson, B.D.O. and Moore, J.B. (1979) *Optimal Filtering*, Prentice-Hall, New Jersey.

Gorman, R.P. and Sejnowski, T.J. (1988) Analysis of hidden units in a layered network trained to classify sonar targets, *Neural Networks* **1**, 75–89.

Hebb, D.O. (1949) *Organization of Behavior*, Wiley, New York.

Hinton, G.E. (1987) Connectionists learning procedures, Carnegie-Mellon Technical Report CMU-CS-87-115.

Hopfield, J.J. (1982) Neural networks and physical system with emergent collective computational properties, *Proc. National Academy of Science USA*, Vol. 79, pp. 2554–2558.

Kawato, M., Uno, Y., and Suzuki, R. (1988) Adaptation and learning in control of voluntary movement II, *J. Robotic Soc. Japan* **6**(3), 50–58 (in Japanese).

Kohonen, T. (1982) Self-organized formation of topologically correct feature maps, *Biological Cybernetics* **43**, 59–69.

Kohonen, T. (1986) Self-organized sensory maps and associative memory, *Physics of Cognitive Processes 1986*, pp. 258–272.

Kohonen, T. (1987) Adaptive, associative, and self-organizing functions in neural computing, *Applied Optics* **26**(23), 4910–4916.

Lippmann, P. (1987) An introduction to computing with neural nets, *IEEE Acoustics Speech Signal Processing Magazine*, Vol. 4, April, pp. 4–22.

Matheus, C.J. and Hohensee, W.E. (1987) Learning in artificial neural systems, *Computational Intelligence* **3**(4), 263–294.

Minsky, M.L. and Papert, S.A. (1988) *Perceptrons*, Expanded Edition, MIT Press, Cambridge, MA.

Plaut, D.C. and Hinton, G.E. (1987) Learning sets of filters using back-propagation, *Computer Speech and Language* **2**, 35–61.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986a) Learning internal representation by error propagation, in D.E. Rumelhart and J.L. McClelland and the PDP research group (eds), *Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vol. I: Foundations*, MIT Press, Cambridge, MA.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986b) Learning representations by back-propagating errors, *Nature* **323**(9), 533–536.

Widrow, B. and Hoff, M.E. (1960) Adaptive switching circuits, *Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4*, pp. 96–104.

Widrow, B., McCool, J.M., Larimore, M.G., and Johnson, C.R. (1976) Stationary and nonstationary learning characteristics of the LMS adaptive filter, *Proc. IEEE* **64**(8), 1151–1162.