Probabilistic Universal Learning Networks and their Applications to Nonlinear Control Systems

Hirasawa, Kotaro
Department of Electrical and Electronic Systems Engineering, Kyushu University

Hu, Jinglu
Department of Electrical and Electronic Systems Engineering, Kyushu University

Murata, Junichi
Department of Electrical and Electronic Systems Engineering, Kyushu University

Jin, ChunZhi
Department of Control Engineering and Science, KyushuInstitute of Technology

https://doi.org/10.15017/1498351

出版情報：九州大学大学院システム情報科学紀要. 3 (2), pp.149-158, 1998-06-22. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

九州大学大学院
システム情報科学研究科報告
第3巻 第2号 平成10年9月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.3, No.2, September 1998

Probabilistic Universal Learning Networks and their Applications to Nonlinear Control Systems

Kotaro HIRASAWA*, Jinglu HU*, Junichi MURATA* and ChunZhi JIN**

(Received June 22, 1998)

Abstract: Probabilistic Universal Learning Networks (PrULNs) are proposed, which are learning networks with a capability of dealing with stochastic signals. PrULNs are extensions of Universal Learning Networks (ULNs). ULNs form a superset of neural networks and were proposed to provide a universal framework for modeling and control of nonlinear large-scale complex systems. A generalized learning algorithm has been devised for ULNs which can also be used in a unified manner for almost all kinds of learning networks. However, the ULNs can not deal with stochastic variables. Specific value of a stochastic signal can be propagated through a ULN, but the ULN does not provide any stochastic characteristics of the signals propagating through it. The PrULNs proposed here are equipped with machinery to calculate stochastic properties of signals and to train network parameters so that the signals behave with the pre-specified stochastic properties. The PrULNs will contribute to the solution of the following problems: (1) improving the generalization capability of the learning networks, (2) more sophisticated stochastic control than the conventional stochastic control, (3) designing problems for the complex systems such as chaotic systems. In this paper, PrULN is proposed and is applied to a nonlinear control system with noise.

Keywords: Learning networks, Neural networks, Probabilistic networks, Control systems, Complex systems, Nonlinear systems, Gradient method, Forward propagation, Backward propagation, Mean, Covariance, Stochastic signals

1. Introduction

Universal Learning Networks (ULNs) have been proposed as general and effective tools for modeling and control of nonlinear large-scale complex systems including physical, social and economical phenomena[1)2)3)]. ULNs are natural but farthest extensions of discrete-time recurrent neural networks; they consist of a number of inter-connected nodes where the nodes may have any continuously differentiable nonlinear functions in them and each pair of nodes can be connected each other by multiple branches with arbitrary (positive, zero, or even negative) time delays. The class of ULNs contains any particular type of neural networks as its sub-class; it includes static or dynamic networks, multi-layered or recurrent networks, and time delay neural networks (TDNNs).

For this generic type of learning networks, a generalized learning algorithm has been also proposed based on the gradient method where the gradient is calculated by either forward or backward propagation. Moreover, not only the first order derivatives (gradients) but also the higher order derivatives can be calculated systematically. This feature of ULNs has been successfully utilized in the design of robust control systems and chaotic control systems[4)5)6)].

However, the ULNs are not equipped with any machinery that can deal with stochastic variables. Specific values of a stochastic signal can be propagated through a ULN, but the ULN does not provide any stochastic characteristics of the signal propagating through it, such as distributions, means and covariances.

If the stochastic characteristics, e.g. means and covariances, of the output signals of ULNs are available based on the characteristics of its internal or external stochastic signals, it will extend the applicability of ULNs significantly. For example, if a ULN has small variances of its output variables in spite of relatively large variances of the input variables, it means that the input-output mapping is a smooth one, and it in turn leads to a high generalization ability of the network. Also, such ULNs that can deal with stochastic signals will realize more sophisticated stochastic control than the conventional ones that are based on the correlation functions.

Other useful applications will be found in design-

* Department of Electrical and Electronic Systems Engineering

** Department of Control Engineering and Science, Kyushu Institute of Technology

ing problems of the complex systems. Although active research on the complex systems and development of the chaos information theory based on entropy and mutual information are in progress, their main topics are only the analysis of the systems. In ULNs, large output variances for small initial value variances mean that the outputs vary significantly depending on the small changes in the initial values, in other words, that the outputs show chaotic behaviors. Thus, by changing the target values of the output variances, we will be able to design ULNs that produce chaotic or non-chaotic signals as we want.

In this paper, first, ULNs that can deal with stochastic signals, Probabilistic Universal Learning Networks (PrULNs), are presented. The complete characterization of a stochastic signal can be done by the probabilistic distribution and its dependency on time. However, the general treatment of the distribution is not easy. Therefore, here we characterize the stochastic property of a signal by several moments, such as mean and covariance.

Two issues are involved in the PrULNs, namely analysis and synthesis. In the analytical phase, the moments of the signals flowing through the networks are calculated. And synthesis means that, given the specification of moments of particular signals, we design a network that meets the moments specification.

Hitherto, a method for calculating central moments for nonlinear static mappings was studied[7], and recently it was applied to a new neural network-based electric load forecasting[8]. These address the analytic issues only, and treat stochastic signals in static neural networks. The PrULNs allow recurrent architectures and thus can deal with dynamic or nonstationary stochastic signals in both analytic and synthetic phases.

In the next section, the analytic issues are discussed where a generic method is described for calculating moments of signals flowing in static or dynamic networks. And in Section 3, the synthetic issues are addressed: the moments specification is incorporated in the criterion function, and parameter training is done so as to minimize it where the calculation method of higher order derivatives for ULNs[3] plays an essential role. The discussions concentrate on the first and second order moments, namely mean and covariance, for simplicity of description. However, the same arguments apply to the higher order moments.

In the last section, an application of PrULN to a nonlinear crane control system is presented, in which a controller is designed to reduce the effect of the noise added to the system.

2. Calculation Method for Means and Covariances of the Node Outputs in PrULNs

2.1 Node Outputs

The structure of a ULN is depicted in **Fig.A.1** in Appendix. Output of a node p in a recurrent ULN and a multi-layered ULN can be represented by the following two equations, respectively,

$$y_p(t) = Dynamic(x_1(t_1), x_2(t_2), \cdots, x_\ell(t_\ell)), \quad (1)$$

$$y_p = Static(x_1, x_2, \cdots, x_\ell), \quad (2)$$

where

- t : Time instant,
- $x_i(t_i)$: Node output at initial time $t_i \leq t$ or external input fed to the recurrent ULN at time t_i,
- x_i : External input to the multi-layered ULN,
- $Dynamic$: Mapping from initial values and external inputs to a particular node output of the recurrent ULN,
- $Static$: Mapping from external inputs to a particular node output of the multi-layered ULN.

The static systems are special cases of the dynamic systems, and therefore mainly the dynamic systems will be studied in the sequel.

The following symbols are used:

- $E[\cdot]$: Ensemble average of a random variable,
- $Cov[\cdot, \cdot]$: Covariance between two random variables,
- $Var[\cdot]$: Variance of a random variable,
- $\mu_3[\cdot, \cdot, \cdot]$: Third order central moment among three random variables.

2.2 Means

First, let us calculate the first order moment, that is the ensemble average $E[y_p(t)]$ of a node output $y_p(t)$.

The following equation is obtained by expanding $Eq.(1)$ in Taylor series up to the second order term around the point $(E[x_1(t_1)], \cdots, E[x_\ell(t_\ell)])$:

$$y_p(t) \simeq Dynamic(E[x_1(t_1)], E[x_2(t_2)], \cdots, E[x_\ell(t_\ell)]) + \sum_i \frac{\partial^+ y_p(t)}{\partial x_i(t_i)}(x_i(t_i) - E[x_i(t_i)])$$

$$+\frac{1}{2}\sum_i\sum_j \frac{\partial^{+2}y_p(t)}{\partial x_i(t_i)\partial x_j(t_j)}(x_i(t_i)-E[x_i(t_i)]) \cdot(x_j(t_j)-E[x_j(t_j)]), \quad (3)$$

where $\frac{\partial^+ y_p(t)}{\partial x_i(t_i)}$ and $\frac{\partial^{+2} y_p(t)}{\partial x_i(t_i)\partial x_j(t_j)}$ are the ordered derivatives proposed by Werbos[9] and are evaluated at $(E[x_1(t_1)],\cdots,E[x_\ell(t_\ell)])$.

Taking the ensemble average of the both sides of *Eq.*(3) with noting that the ensemble average of the second term in the right hand side is zero leads to the following equation,

$$E[y_p(t)] \simeq Dynamic(E[x_1(t_1)],E[x_2(t_2)],\cdots,E[x_\ell(t_\ell)]) +\frac{1}{2}\sum_i\sum_j \frac{\partial^{+2}y_p(t)}{\partial x_i(t_i)\partial x_j(t_j)}Cov[x_i(t_i),x_j(t_j)]. \quad (4)$$

Eq.(4) is the basic equation for computing the first order moment of the node output $y_p(t)$ from the first and second order moments of the node output initial values and the external inputs, $E[x_i(t_i)]$ and $Cov[x_i(t_i),x_j(t_j)]$. The second order derivatives that appear in the equation are calculated by the already proposed method[3] which is also described briefly in Appendix.

2.3 Covariances

Next let us calculate the covariance of $y_p(t)$ and $y_q(s)$. From *Eq.*(3) and (4), the following equation is obtained,

$$\begin{aligned} y_p(t)-E[y_p(t)] &\simeq \sum_i \frac{\partial^+ y_p(t)}{\partial x_i(t_i)}(x_i(t_i)-E[x_i(t_i)]) \\ &+\frac{1}{2}\sum_i\sum_j \frac{\partial^{\dagger 2} y_p(t)}{\partial x_i(t_i)\partial x_j(t_j)}(x_i(t_i)-E[x_i(t_i)]) \\ &\cdot(x_j(t_j)-E[x_j(t_j)]) \\ &-\frac{1}{2}\sum_i\sum_j \frac{\partial^{+2} y_p(t)}{\partial x_i(t_i)\partial x_j(t_j)} \\ &\cdot Cov[x_i(t_i),x_j(t_j)]. \end{aligned} \quad (5)$$

Similar equation can be obtained for $y_q(s)-E[y_q(s)]$.

Covariance of $y_p(t)$ and $y_q(s)$ is derived from *Eq.*(5) and the equation for $y_q(s)-E[y_q(s)]$ considering up to the third order terms as follows,

$$\begin{aligned} &Cov[y_p(t),y_q(s)] \\ &\simeq E[(y_p(t)-E[y_p(t)])(y_q(s)-E[y_q(s)])] \\ &=\sum_i\sum_j \frac{\partial^+ y_p(t)}{\partial x_i(t_i)}\frac{\partial^+ y_q(s)}{\partial x_j(t_j)}Cov[x_i(t_i),x_j(t_j)] \\ &+\frac{1}{2}\sum_i\sum_j\sum_k \frac{\partial^{+2} y_q(s)}{\partial x_i(t_i)\partial x_j(t_j)}\frac{\partial^+ y_p(t)}{\partial x_k(t_k)} \\ &\cdot\mu_3[x_i(t_i),x_j(t_j),x_k(t_k)] \\ &+\frac{1}{2}\sum_i\sum_j\sum_k \frac{\partial^{+2} y_p(t)}{\partial x_i(t_i)\partial x_j(t_j)}\frac{\partial^+ y_q(s)}{\partial x_k(t_k)} \\ &\cdot\mu_3[x_i(t_i),x_j(t_j),x_k(t_k)]. \end{aligned} \quad (6)$$

Eq.(6) is the basic equation for computing the covariance of $y_p(t)$ and $y_q(s)$ from the first, second and third moments of the node output initial values and the external inputs.

3. Learning Algorithm of Parameters

In this section, the synthetic issue is addressed, and a learning algorithm of PrULNs is proposed. PrULNs or the mappings *Dynamic* or *Static* which appear in *Eq.*(1) or (2) contain adjustable parameters λ_m. The parameters are trained so as to minimize a criterion function L by a gradient method. Here, the higher order derivatives play an important role again.

The criterion function should measure how well the PrULN under consideration generates stochastic signals with specified stochastic properties. As stated before, we use the first and second order moments in stochastic property specification. Therefore, let the criterion function L be represented by *Eq.*(7),

$$\begin{aligned} L_{MV} &= L_M + L_V \\ L_M &= \sum_p\sum_t (E[y_p(t)]-y_p^0(t))^2 \\ L_V &= \sum_p\sum_q\sum_t\sum_s \alpha_{pq}(t,s) \\ &\quad\cdot(Cov[y_p(t),y_q(s)]-y_{pq}^0(t,s))^2, \end{aligned} \quad (7)$$

where

$y_p^0(t)$: Target value of $E[y_p(t)]$,
$y_{pq}^0(t,s)$: Target value of $Cov[y_p(t),y_q(s)]$,
$\alpha_{pq}(t,s)$: Weighting coefficient.

By minimizing the criterion function, the parameters are adjusted so that the ensemble average of $y_p(t)$ approaches $y_p^0(t)$ and the covariance of $y_p(t)$ and $y_q(s)$ approaches $y_{pq}^0(t,s)$.

The training of parameters λ_m is done by the gradient method in the same way as parameter training of the Universal Learning Networks,

$$\lambda_m \leftarrow \lambda_m - \gamma \frac{\partial^+ L_{MV}}{\partial \lambda_m}, \tag{8}$$

$\gamma > 0$: learning coefficient.

The derivative $\frac{\partial^+ L_{MV}}{\partial \lambda_m}$ can be obtained by differentiating $Eq.(7)$ as follows,

$$\begin{aligned}
&\frac{\partial^+ L_{MV}}{\partial \lambda_m} \\
&= 2 \sum_p \sum_t (E[y_p(t)] - y_p^0(t)) \frac{\partial^+ E[y_p(t)]}{\partial \lambda_m} \\
&\quad + 2 \sum_p \sum_q \sum_t \sum_s \alpha_{pq}(t,s) \\
&\quad \cdot (Cov[y_p(t), y_q(s)] - y_{pq}^0(t,s)) \\
&\quad \cdot \frac{\partial^+ Cov[y_p(t), y_q(s)]}{\partial \lambda_m},
\end{aligned} \tag{9}$$

where $\frac{\partial^+ E[y_p(t)]}{\partial \lambda_m}$ and $\frac{\partial^+ Cov[y_p(t), y_q(s)]}{\partial \lambda_m}$ can be derived from $Eq.(4)$ and (6) as follows,

$$\begin{aligned}
&\frac{\partial^+ E[y_p(t)]}{\partial \lambda_m} \\
&= \frac{\partial^+ y_p(t)}{\partial \lambda_m} + \frac{1}{2} \sum_i \sum_j \frac{\partial^{+3} y_p(t)}{\partial x_i(t_i) \partial x_j(t_j) \partial \lambda_m} \\
&\quad \cdot Cov[x_i(t_i), x_j(t_j)],
\end{aligned} \tag{10}$$

$$\begin{aligned}
&\frac{\partial^+ Cov[y_p(t), y_q(s)]}{\partial \lambda_m} \\
&= \sum_i \sum_j \left[\frac{\partial^{+2} y_p(t)}{\partial x_i(t_i) \partial \lambda_m} \frac{\partial^+ y_q(s)}{\partial x_j(t_j)} \right. \\
&\quad \left. + \frac{\partial^+ y_p(t)}{\partial x_i(t_i)} \frac{\partial^{+2} y_q(s)}{\partial x_j(t_j) \partial \lambda_m} \right] \cdot Cov[x_i(t_i), x_j(t_j)] \\
&\quad + \frac{1}{2} \sum_i \sum_j \sum_k \left[\frac{\partial^{+3} y_q(s)}{\partial x_i(t_i) \partial x_j(t_j) \partial \lambda_m} \frac{\partial^+ y_p(t)}{\partial x_k(t_k)} \right. \\
&\quad \left. + \frac{\partial^{+2} y_q(s)}{\partial x_i(t_i) \partial x_j(t_j)} \frac{\partial^{+2} y_p(t)}{\partial x_k(t_k) \partial \lambda_m} \right] \\
&\quad \cdot \mu_3[x_i(t_i), x_j(t_j), x_k(t_k)] \\
&\quad + \frac{1}{2} \sum_i \sum_j \sum_k \left[\frac{\partial^{+3} y_p(t)}{\partial x_i(t_i) \partial x_j(t_j) \partial \lambda_m} \frac{\partial^+ y_q(s)}{\partial x_k(t_k)} \right. \\
&\quad \left. + \frac{\partial^{+2} y_p(t)}{\partial x_i(t_i) \partial x_j(t_j)} \frac{\partial^{+2} y_q(s)}{\partial x_k(t_k) \partial \lambda_m} \right] \\
&\quad \cdot \mu_3[x_i(t_i), x_j(t_j), x_k(t_k)].
\end{aligned} \tag{11}$$

The higher order derivatives in $Eq.(10)$ and $Eq.(11)$ can be calculated by the method of forward propagation Universal Learning Networks[3] as described in Appendix. And they are evaluated at $(E[x_i(t_i)], \cdots, E[x_\ell(t_\ell)])$. For example, the third order derivative $\frac{\partial^{+3} y_p(t)}{\partial x_i(t_i) \partial x_j(t_j) \partial \lambda_m}$ can be calculated using the iterative equation $Eq.(A.4)$ in Appendix, by putting $h_k(t) = y_p(t), \lambda_1(t_1) = x_i(t_i), \lambda_2(t_2) = x_j(t_j)$ and $\lambda_3 = \lambda_m$. The equation contains first and second order derivatives which can be calculated by $Eq.(A.2)$ and $Eq.(A.3)$, respectively, by substituting relevant variables and parameters into them.

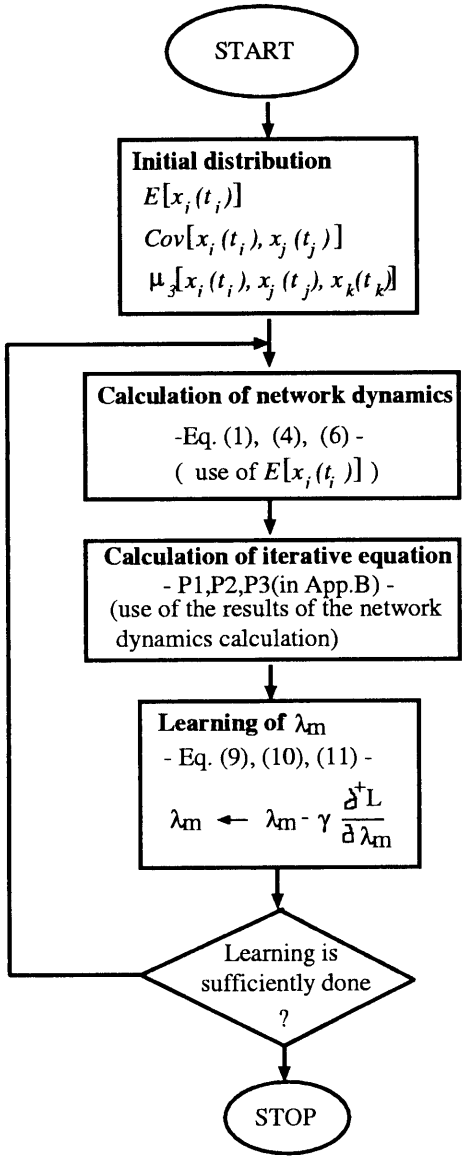


Fig.1 Flowchart of parameter training

In summary, the parameters are updated according to $Eq.(8)$, where necessary derivatives are given by $Eq.(9)$-(11) and $Eq.(A.2)$-$(A.4)$, and the node outputs are derived from the network mapping $Eq.(1)$. And thus, given the stochastic properties of the initial state of the network and the external inputs, and the target values for the stochastic properties of the node outputs, we can obtain a PrULN

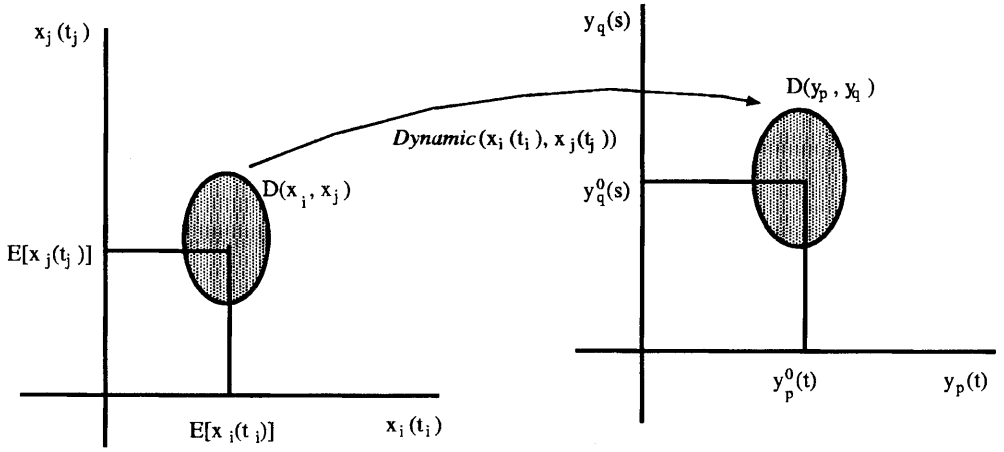


Fig.2 Transformation from initial value distribution to the desired output distribution by mapping $Dynamic(x_i(t_i), x_j(t_j))$

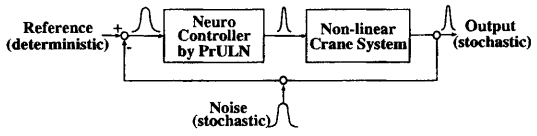


Fig.3 Control system with stochastic noise

whose stochastic properties are approximately equal to those specified.

In **Fig.1**, the flowchart of parameter training is shown.

Fig.2 shows the mapping which is to be realized by the PrULN for a two-dimensional unimodal distribution case. The ellipsoid $D(x_i, x_j)$ shows the contour in x_i–x_j space for a given probability, and $D(y_p, y_q)$ is the desired contour in y_p–y_q space for the same probability. What is done by the learning algorithm described so far is to train the PrULN so that it transforms the inside of $D(x_i, x_j)$ to the inside of $D(y_p, y_q)$. Therefore, by changing the target values $y_p^0(t)$, $y_q^0(s)$ and $y_{pq}^0(t, s)$, we can obtain a proper PrULN with the stochastic properties that we want.

4. Application of PrULN to Nonlinear Dynamic System Control

In this section, an application of PrULN to a nonlinear crane control system is presented, where the crane control system is contaminated with noise and an optimal controller should be designed to reduce the effect of the noise added to the system.

As is shown in **Fig.3**, the noise which is a stochastic signal with known mean and covariance is inserted in the output node of the crane control system. The problem is to design a neural network controller that gives the specified stochastic properties to the

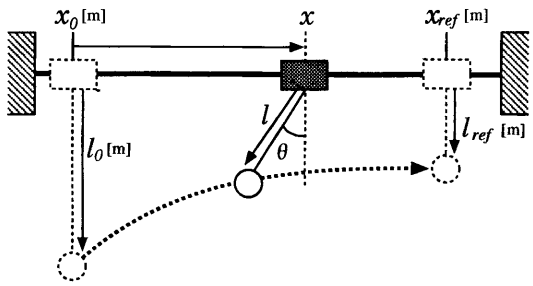


Fig.4 Nonlinear crane system

crane system.

4.1 Nonlinear Crane Control System

The nonlinear crane control system shown in **Fig.4** can be described by the following differential equations,

$$\begin{aligned}
\frac{d^2x}{dt^2} &= -\frac{mg}{M}\theta - \frac{D+G}{M}\frac{dx}{dx} + \frac{G}{M}u_1, \\
\frac{d^2\theta}{dt^2} &= -\frac{M+m}{\ell M}g\theta - \frac{D+G}{\ell M}\frac{dx}{dx} + \frac{G}{\ell M}u_1, \\
\frac{d^2\ell}{dt^2} &= -\frac{C+G_m}{M}\frac{d\ell}{dt} + \frac{G_m}{m}u_2.
\end{aligned} \tag{12}$$

In the above equations position of the crane stand, angle between the rope and vertical line, and position of the load are represented by x, θ, l respectively. u_1, u_2 are input voltages to the motor for moving the crane stand and for rolling up the load, and C, G, G_m, D, M, m, g are system parameters and gravity accelaration.

Equation(12) can be transformed into the discrete time network form shown in **Fig.5** which is commonly used in applications using Universal Learning Networks[1].

In **Fig.5**, $N_1, \cdots, N_6$ stand for the nodes of the

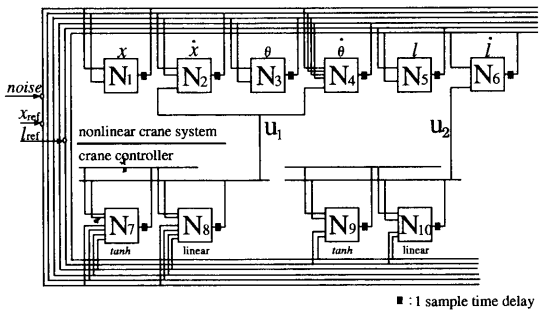


Fig.5 Control model of the crane system by ULN

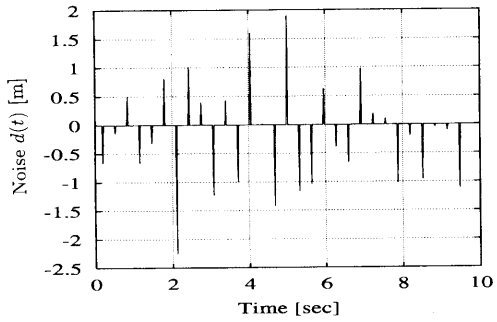


Fig.6 Sample time series of Gaussian noise

nonlinear crane system, and $N_7, \cdots, N_{10}$ represent the nodes of the crane controller. Each control input u_1 or u_2 is calculated by the linear function node N_8 or N_{10} which is connected recurrently with sigmoid function nodes N_7 or N_9.

In the simulations, the behavior of the system is calculated over a period of 10 [sec] which consists of a total of 500 sampling instants. An independent Gaussian noise $d(t)$ with $N(0,1)$ is added to the measurement of x at arbitrarily chosen 30 instants out of the 500. An example time series of the noise is shown in **Fig.6**.

The crane should be controlled so that position x of the crane stand and ℓ of the load become the reference value $x_{ref} = 1.0$[m] and $\ell_{ref} = 0.5$[m] as soon as possible and that angle θ is as small as possible, assuming initial positions of the crane stand and the load to be 0.0[m], 1.0[m] respectively.

Simulation conditions are as follows: $C = 0.42$[kg/sec], $G = 1700$[N/V], $G_m = 0.98$[N/V], $D = 300$[kg/sec], $M = 40$[kg], $m = 20$[kg], $g = 9.8$[m/sec^2], Δt (sampling period) $= 0.02$[sec].

4.2 Training of Parameters of Controllers

In this sub-section, training of parameters of two recurrent neural network controllers which consist of nodes N_7, N_8 and nodes N_9, N_{10} in **Fig.5** is presented.

Three kinds of criterion functions were studied to compare the performance of the controllers designed by PrULN and a conventional method.

4.2.1 Conventional Criterion Function E

Usage of the following criterion function E corresponds to the conventional parameter training of neural network controllers,

$$E = \frac{1}{2}\Bigg[\sum_{s\in T}\{Q_{n1}(x_{ref}-x(s))^2 + Q_{n3}\theta(s)^2 + Q_{n4}\dot{\theta}(s)^2 + Q_{n5}(\ell_{ref}-\ell(s))^2 + R_{n1}u_1(s)^2 + R_{n2}u_2(s)^2\} + Q_{n2}\dot{x}(t_f)^2 + Q_{n6}\dot{\ell}(t_f)^2\Bigg], \quad (13)$$

where

T : Set of time instants,
t_f : Final sampling instant,
$Q_{n1}, \cdots, Q_{n6} = 1.0, R_{n1} = R_{n2} = 0.001$
: Coefficients.

In this case, calculation of the dynamics $Eq.(1)$ with deterministic behavior is carried out supposing that a deterministic signal with value $E[d(t)]$ is inserted to the system instead of Gaussian noise. It is clear that $E[y_p(t)] = y_p(t)$, $Cov\,[y_p(t), y_q(s)] = 0$ in $Eq.(7) \sim Eq.(11)$. Therefore, only the first order derivative $\frac{\partial^+ y_p(t)}{\partial \lambda_m} = P_1(p,t,\lambda_m)$ is used to evaluate $\frac{\partial^+ L_{MV}}{\partial \lambda_m}$ in $Eq.(8)$.

4.2.2 Criterion Function Evaluating Covariance

The following criterion function L_{EV} is studied to investigate the effect of covariance control,

$$L_{EV} = E + L_V, \quad (14)$$

$$L_V = \sum_{s\in T}\{Q_{v1}Var[x(s)] + Q_{v2}Var[\dot{x}(s)] + Q_{v3}Var[\theta(s)] + Q_{v4}Var[\dot{\theta}(s)]\}, \quad (15)$$

where $Q_{v1} = Q_{v3} = 20.0$, $Q_{v2} = Q_{v4} = 4.0$ which were found to be appropriate.

For simplicity, this time only variance is considered and the target values of variances $y^0_{pq}(t,s)$ in $Eq.(7)$ are set to zero.

The criterion function L_{EV} means that parameters of the controller are trained so that variance of $x(t), \dot{x}(t), \theta(t)$ and $\dot{\theta}(t)$ caused by the Gaussian observation noise of $x(t)$ are minimized as well as the minimization of the criterion function E.

In the same way as described in 4.2.1, when calculating E in $Eq.(14)$, the dynamics using value $E[d(t)]$ instead of Gaussian noise was adopted.

4.2.3 Criterion Function Evaluating Mean and Covariance

The criterion function stated below corresponds to the original criterion function L_{MV} in *Eq.*(7), which means that the parameters are adjusted so that the ensemble average and covariance of the node output caused by the observation noise approach their target values,

$$L_{MV} = L_M + L_V, \tag{16}$$

$$\begin{aligned} L_M = \frac{1}{2}\Bigg[\sum_{s\in T} \{ & Q_{m1}(x_{ref} - E[x(s)])^2 \\ & + Q_{m3}(E[\theta(s)])^2 + Q_{m4}(E[\dot{\theta}(s)])^2 \\ & + Q_{m5}(l_{ref} - E[l(s)])^2 + R_{m1}(E[u_1(s)])^2 \\ & + R_{m2}(E[u_2(s)])^2 \} + Q_{m2}(E[\dot{x}(t_f)])^2 \\ & + Q_{m6}(E[\dot{l}(t_f)])^2 \Bigg], \end{aligned} \tag{17}$$

where the following coefficients were adopted as they were found to be appropriate,

$$\begin{aligned} & Q_{m1} = 4.0, Q_{m5} = 2.0, \\ & Q_{m2} = Q_{m3} = Q_{m4} = Q_{m6} = 1.0, \\ & R_{m1} = R_{m2} = 0.001, \\ & Q_{v1} = 8.0, Q_{v2} = 4.0, \\ & Q_{v3} = 6.0, Q_{v4} = 2.0, \end{aligned}$$

in *Eq.*(15) and (17).

Parameters of the controllers which consist of nodes N_7, N_8, N_9 and N_{10} in **Fig.5** can be trained by the gradient method minimizing one of the above three kinds of criterion functions using *Eq.*(8), $\cdots$, (11), and higher order derivatives of *Eq.*(10) and *Eq.*(11) can be calculated using interactive equations (*A*.2), $\cdots$, (*A*.4) in Appendix B where P_1, P_2 and P_3 are defined on not only neural network nodes but also the nodes which describe the nonlinear crane control system.

4.3 Simulation Results

Simulations were carried out five times per each criterion function with initial parameters being set randomly in $[-0.1, 0.1]$, and five learning curves after 100,000 times training converged to almost the same values.

From **Fig.7** showing the average learning curves for criteria E, L_{EV} and L_{MV}, it is shown that the criterion function E achieved the smallest value, on the other hand, the criterion function L_{MV} attained the largest value.

After training the parameters of the controller to minimize the criterion functions E, L_{EV} and L_{MV},

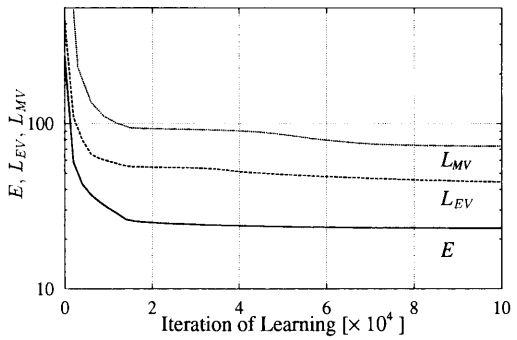
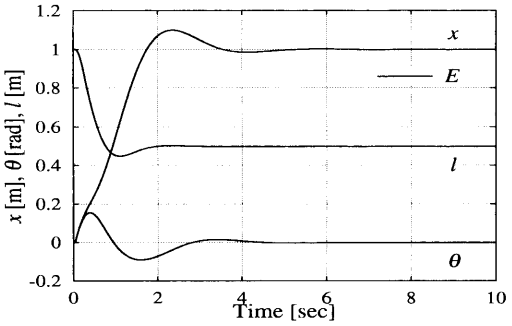


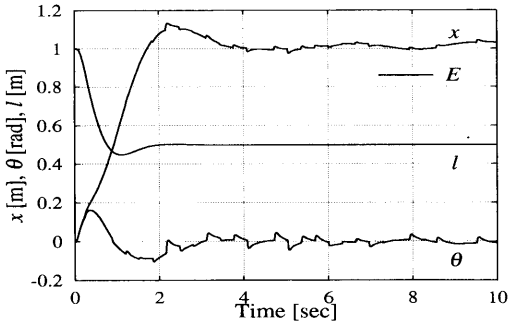
Fig.7 Learning curve of E, L_{EV}, L_{MV}

dynamics of the crane control system was calculated for two cases: the case where Gaussian noise $d(t)$ in **Fig.6** was inserted in the observation point of x and another case that the crane control system was noise free.

Fig.8, **Fig.9** and **Fig.10** show the crane dynamics without noise and dynamics contaminated with noise in the case of E, L_{EV} and L_{MV} respectively. When the criterion function E is used, which corresponds to the conventional design, the position x of the crane stand, the position l of the load and angle θ between the rope and vertical line show the desirable characteristics if the system is noise free,

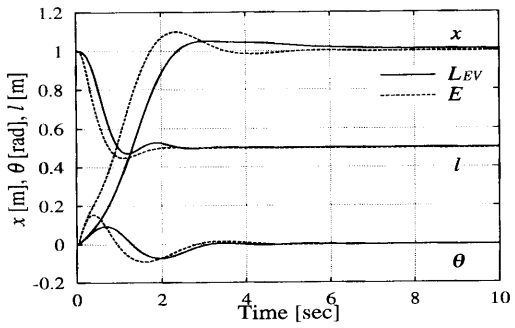


(a) Results without noise

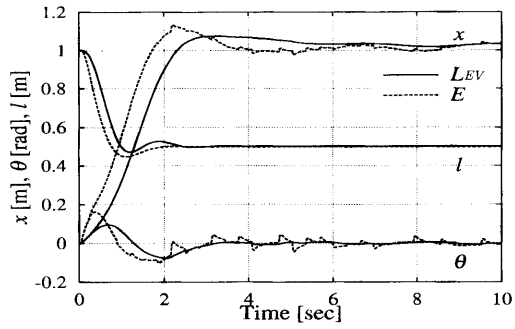


(b) Results with noise

Fig.8 Control results in the case of criterion E

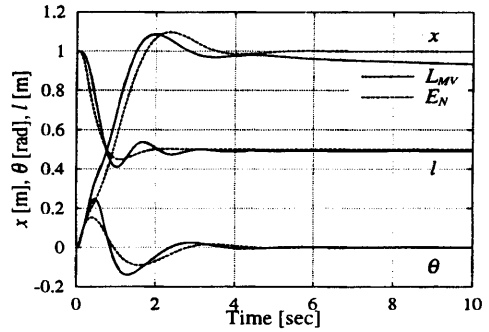


(a) Results without noise

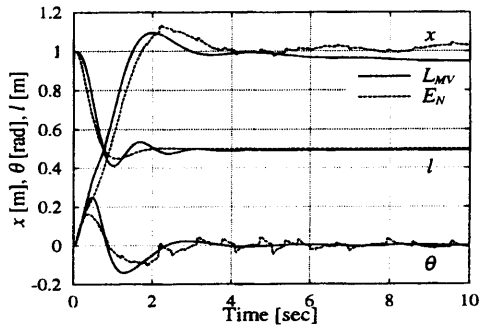


(b) Results with noise

Fig.9 Control results in the case of criterion L_{EV}



(a) Results without noise



(b) Results with noise

Fig.10 Control results in the case of criterion L_{MV}

but if the Gaussian noise $d(t)$ with $N(0,1)$ is inserted on the observation point of x, x fluctuates heavily because x_{ref} is changed by the noise $d(t)$, and θ which is related to x also fluctuates, while ℓ dose not vibrate because ℓ is not influenced by the dynamincs of x as shown in $Eq.(12)$.

Fig.9 shows the dynamics using the controller obtained by the criterion function L_{EV} compared with the criterion function E. In this case, parameters of the controller are trained so that the variances of $x(t), \dot{x}(t), \theta(t)$ and $\dot{\theta}(t)$ should be minimized, even through the noise is inserted to the system.

From **Fig.9** it is shown that the fluctuations of x and θ are decreased dramatically at the small sacrifice of steady state error, quick response and damping characteristics of x.

Fig.10 shows the dynamics of the crane control system when the criterion function L_{MV} is used. Almost the same dynamics are obtained by the criterion function L_{MV} as compared with the criterion function L_{EV}. The difference is that the steady state error of x in the case of L_{MV} is a little larger.

The conclusion is that PrULN can be utilized usefully to curb the effect caused by the noise, but careful choice of coefficients Q and R is necessary in order to meet the every kind of dynamic requirements such as samll steady state error, quick response and damping characteristic.

5. Conclusions

In this paper, Probabilistic Universal Learning Networks have been proposed. The PrULNs are extensions of conventional ULNs. They allow stochastic signals to be propagated through them, and they are equipped with machinery to calculate means and covariances of the stochastic signals and to train their parameters so that the signals behave with the pre-specified statistic properties.

The calculation of higher order derivatives that are necessary in the training are done based on the forward propagation algorithm which has been already devised for non-probabilistic ULNs. It has been shown from the simulation studies that an optimal controller of a nonlinear system which is contaminated with noise can be easily and effectively designed by utilizing the PrULNs. And, in future, many application systems of PrULN such as designing problems for stochastic systems will be developed.

Appendix

A Universal Learning Networks[1)2)3)]

A Universal Learning Network (ULN) is a discrete-time learning network where any kinds of nonlinearly operated nodes with a continuously differentiable function can be connected arbitrarily to each other by multiple branches that may have arbitrary time delays as shown in **Fig. A.1**.

The basic equation of the ULNs is represented as follows,

$$\begin{aligned} &h_j(t) \\ &= f_j(\{h_i(t - D_{ij}(p)) \mid i \in JF(j), p \in B(i,j)\}, \\ &\quad \{r_n(t) \mid n \in N(j)\}, \{\lambda_m(t) \mid m \in M(j)\}), \\ &\quad j \in J, t \in T, \end{aligned} \tag{A.1}$$

where

$h_j(t)$	:	Output value of node j at time t,
$r_n(t)$	:	Value of n-th external input variable at time t,
$\lambda_m(t)$	:	Value of m-th parameter at time t,
f_j	:	Nonlinear function of node j,
$D_{ij}(p)$	:	Time delay of p-th branch from node i to node j,
J	:	Set of node indices,
$JF(j)$	:	Set of indices of nodes which are connected to node j,
$JB(j)$	:	Set of indices of nodes which are connected from node j,
N	:	Set of indices of external input variables,
$N(j)$	:	Set of indices of external input variables which are fed to node j,
M	:	Set of parameter indices,
$M(j)$	:	Set of indices of parameters which directly affect output of node j,
$B(i,j)$	:	Set of indices of branches from node i to node j,
T	:	Set of time instants.

B Calculation of Higher Order Derivatives in ULNs [2)3)]

For simplicity, let us denote the derivative of node output $h_k(t)$ with respect to parameter $\lambda_1(t_1)$ by $P_1(k,t,\lambda_1(t_1))$. Likewise, we will use the symbols P_2 and P_3 to denote second and third order derivatives:

$$\begin{aligned} &P_2(k,t,\lambda_1(t_1),\lambda_2(t_2)) \\ &\quad = \frac{\partial^{+2} h_k(t)}{\partial\lambda_1(t_1)\partial\lambda_2(t_2)} \end{aligned}$$

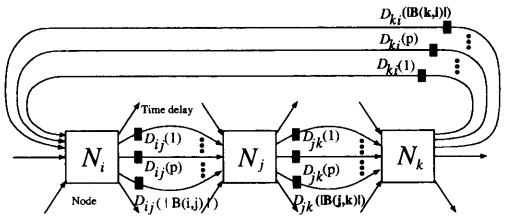


Fig.A.1 Architecture of a Universal Learning Network

$$\begin{aligned} &P_3(k,t,\lambda_1(t_1),\lambda_2(t_2),\lambda_3(t_3)) \\ &\quad = \frac{\partial^{+3} h_k(t)}{\partial\lambda_1(t_1)\partial\lambda_2(t_2)\partial\lambda_3(t_3)} \end{aligned}$$

The first, second and third order derivatives are obtained by the following equations,

$$\begin{aligned} P_1(k,t,\lambda_1(t_1)) = &\sum_{j\in JF(k)} \sum_{p\in B(j,k)} \\ &\left[\frac{\partial h_k(t)}{\partial h_j(t - D_{jk}(p))}\right. \\ &\left. P_1(j, t - D_{jk}(p)), \lambda_1(t_1)\right] \\ &+\frac{\partial h_k(t)}{\partial\lambda_1(t_1)}, \\ &k \in J, t \in T, t_1 \le t, \end{aligned} \tag{A.2}$$

$$\begin{aligned} P_2(k,t,\lambda_1(t_1),\lambda_2(t_2)) = &\sum_{j\in JF(k)} \sum_{p\in B(j,k)} \\ &\left[\frac{\partial^+\left(\frac{\partial h_k(t)}{\partial h_j(t-D_{jk}(p))}\right)}{\partial\lambda_2(t_2)}\right. \\ &\left. \cdot P_1(j, t - D_{jk}(p), \lambda_1(t_1))\right] \\ &+ \sum_{j\in JF(k)} \sum_{p\in B(j,k)} \left[\frac{\partial h_k(t)}{\partial h_j(t - D_{jk}(p))}\right. \\ &\left. \cdot P_2(j, t - D_{jk}(p), \lambda_1(t_1), \lambda_2(t_2))\right] \\ &+\frac{\partial^+\left(\frac{\partial h_k(t)}{\partial\lambda_1(t_1)}\right)}{\partial\lambda_2(t_2)}, \\ &k \in J, t \in T, t_1, t_2 \le t, \end{aligned} \tag{A.3}$$

$$
\begin{aligned}
&P_3(k,t,\lambda_1(t_1),\lambda_2(t_2),\lambda_3(t_3)) \\
&= \sum_{j\in JF(k)}\sum_{p\in B(j,k)} \\
&\quad \left[\frac{\partial^{+2}\left(\frac{\partial h_k(t)}{\partial h_j(t-D_{jk}(p))}\right)}{\partial\lambda_2(t_2)\partial\lambda_3(t_3)}\right. \\
&\quad \left.\cdot P_1(j,t-D_{jk}(p),\lambda_1(t_1))\right] \\
&\quad + \sum_{j\in JF(k)}\sum_{p\in B(j,k)} \\
&\quad \left[\frac{\partial^{+}\left(\frac{\partial h_k(t)}{\partial h_j(t-D_{jk}(p))}\right)}{\partial\lambda_2(t_2)}\right. \\
&\quad \left.\cdot P_2(j,t-D_{jk}(p),\lambda_1(t_1),\lambda_3(t_3))\right] \\
&\quad + \sum_{j\in JF(k)}\sum_{p\in B(j,k)} \\
&\quad \left[\frac{\partial^{+}\left(\frac{\partial h_k(t)}{\partial h_j(t-D_{jk}(p))}\right)}{\partial\lambda_3(t_3)}\right. \\
&\quad \left.\cdot P_2(j,t-D_{jk}(p),\lambda_1(t_1),\lambda_2(t_2))\right] \\
&\quad + \sum_{j\in JF(k)}\sum_{p\in B(j,k)}\left[\frac{\partial h_k(t)}{\partial h_j(t-D_{jk}(p))}\right. \\
&\quad \left.\cdot P_3(j,t-D_{jk}(p),\lambda_1(t_1),\lambda_2(t_2),\lambda_3(t_3))\right] \\
&\quad + \frac{\partial^{+2}\left(\frac{\partial h_k(t)}{\partial\lambda_1(t_1)}\right)}{\partial\lambda_2(t_2)\partial\lambda_3(t_3)}, \\
&\qquad k\in J, t\in T, t_1,t_2,t_3\le t. \qquad \text{(A.4)}
\end{aligned}
$$

The derivative of a node output with respect to another node output, e.g. $\partial h_k(t)\ /\partial h_j(t-D_{jk}(p))$, in these equations can be calculated by using the calculation results of the dynamics $Eq.(1)$ with its inputs being their mean values $E[x_i(t_i)]$.

References

1) K. Hirasawa, M. Ohbayashi and J. Murata: "Universal Learning Network and Computation of its Higher Order Derivatives", in *Proc. of IEEE International Conference on Neural Networks*, 1995, pp.1273–1277.
2) K. Hirasawa, M. Ohbayashi, M. Koga and M. Harada: "Forward Propagation Universal Learning Network", in *Proc. of IEEE International Conference on Neural Networks*, 1996, pp.353–358.
3) K. Hirasawa, J. Hu, M. Ohbayashi and J. Murata: "Computing Higher Order Derivatives in Universal Learning Networks", in *Journal of Advanced Computational Intelligence*, 1998 ,Vol.2, No.2.
4) M. Ohbayashi, K. Hirasawa, J. Murata and M. Harada: "Robust Learning Control using Universal Learning Network", in *Proc. of IEEE International Conference on Neural Networks*, 1996, pp.2208–2213.
5) M. Ohbayashi, K. Hirasawa, M. Hashimoto and J. Murata: "Robust Control using Second Order Derivative of Universal Learning Network", in *Proc. of IEEE International Conference on Systems, Man and Cybernetics*, 1996, pp.1184–1189.
6) M. Koga, K. Hirasawa, M. Ohbayashi and J. Murata: "Chaos Control using Second Order Derivatives of Universal Learning Network", in *Proc. of IEEE International Conference on Neural Networks*, 1995, pp.1287–1292.
7) G. J. Hahn and S. S. Shapiro: *Stability Models in Engineering*, Wiley, New York, 1967.
8) D. K. Ranaweera, G. G. Karady and G. G. Farmer: "Effect of Probabilistic Input on Neural Network–Based Electric Load Forecasting", *IEEE Transactions on Neural Networks*, Vol.7, No.6, November, 1996.
9) P. Werbos: *Beyond Regression "New Tools for Prediction and Analysis in the Behavioral Science"*, Ph.D. Dissertation, Harvard University, 1974.