International Journal of Network Dynamics and Intelligence Scilight

Article

Parameter Learning of Probabilistic Boolean Control Networks with Input-Output Data

Hongwei Chen [1,*], Qi Chen [1], Bo Shen [1], and Yang Liu [2]

[1] College of Information Science and Technology, Donghua University, Shanghai 201620, China
[2] School of Mathematical Sciences, Zhejiang Normal University, Jinhua 321004, China
[*] Correspondence: hongwei@dhu.edu.cn

Received: 23 September 2023
Accepted: 27 November 2023
Published: 26 March 2024

Abstract: This paper investigates the parameter learning problem for the probabilistic Boolean control networks (PBCNs) with input-output data. Firstly, an algebraic expression of the PBCNs is obtained by taking advantage of the semi-tensor product technique, and then, the parameter learning problem is transformed into an optimal problem to reveal the parameter matrices of a linear system in a computationally efficient way. Secondly, two recursive semi-tensor product based algorithms are designed to calculate the forward and backward probabilities. Thirdly, the expectation maximization algorithm is proposed as an elaborate technique to address the parameter learning problem. In addition, a useful index is introduced to describe the performance of the proposed parameter learning algorithm. Finally, two numerical examples are employed to demonstrate the reliability of the proposed parameter learning approach.

Keywords: probabilistic boolean control networks; parameter learning; semi-tensor product; forward and backward probabilities; expectation maximization algorithm

1. Introduction

The Boolean network (BN), as a typical logic model, was firstly put forward by Kauffman [1] with the aim of understanding the dynamics of nonlinear and complex biological systems. An important application of the BN is to simulate the dynamics of gene regulatory networks (GRNs) [2]. Understanding regulation mechanisms in GRNs plays a crucial role in practical applications. A new Boolean model, namely the Boolean control network (BCN), has been put forward in the pioneering work [3]. Recently, by taking advantage of the semi-tensor product (STP) of matrices, a novel method called the algebraic state space representation (ASSR) has been proposed to analyse the dynamics of BNs [4]. Consequently, the study of BNs has achieved great progress, such as controllability and observability [5, 6], disturbance decoupling [7, 8], synchronization [9, 10], optimal control [11, 12], output regulation [13, 14] and others [15−18].

A recent yet important discovery in the field of systems biology is that the gene expression process involves considerable uncertainties. Hence, a deterministic Boolean model might not be suitable for real applications. For the sake of simulating thee dynamics of actual GRNs accurately, the concept of probabilistic Boolean networks (PBNs) has been innovatively proposed by Shmulevich in [19]. The dynamics of PBCNs can naturally be regarded as a stochastic expansion of BCNs. Specifically, at each time step, the governing BCN is randomly selected from a collection of BCNs and endowed with a predetermined probability. In the past decades, much effort has been devoted to the study of PBNs and PBCNs, such as synchronization [20], finite-time stability [21], model evaluation [22], state feedback stabilization [23], optimal control [24], stability and stabilization [25, 26].

The parameter learning issue for BNs has emerged as a research topic of vital importance since it is helpful to reveal the regulatory principles of genes and uncover the regulatory process of GRNs. The key of the parameter learning problem lies in obtaining the parameters of BNs from time-series gene expression data. In other words, the identified BNs should match the given data as much as possible. In the past decades, a significant amount of attention has been focused on the parameter learning/identification issue of BNs from gene expression data. Based on the best-fit extension paradigm, some efficient algorithms have been proposed in [27] to identify the model structure, which are extremely useful in situations where the measurements of the gene expression are noisy. In the interesting

paper [28], a simplified search strategy has been introduced for the identification of BNs, which greatly reduces the time required to learn the networks. Based on a novel representation method of PBNs, an efficient learning algorithm has been designed in [29] for predicting the dynamic behavior of big-scale logic systems. It is worth noting that the aforementioned identification algorithms of Boolean models only use information on the occurrence of samples. For the purpose of extending the identifiable classes, a novel approach has been established in [30] to identify a PBN from samples, which also makes use of the information on the frequencies of different samples.

By resorting to the STP technique, a logic function can be represented as an algebraic form, and a BN can naturally be transformed into a discrete-time linear system. Then, the parameter identification problem of BNs can be converted into the problem of identifying the parameter matrices of the linear system that has been established, which definitely makes the learning problem more accessible. Note that the parameter learning issue for BNs has emerged as a research topic of great importance. In recent years, many interesting results have been achieved regarding the parameter learning issue of BNs. In the pioneering work [31], an STP-based method has been developed to construct the dynamic model of BNs by using the observed data. Later, some necessary and sufficient conditions have been established to identify a BCN via a series of input-output data [32]. In order to lower the data requirements, a position-transform mining technique has been proposed in the work [33] to improve data utilization during the identification procedure. In the interesting paper [34], an integer linear programming approach has been utilized to identify BNs based on the time-series gene expression data and the prior knowledge of the partial network structure and interactions between nodes.

It is worth mentioning that PBCNs contain several inputs and uncertainties, which makes it difficult to learn the network. To the best of the authors' knowledge, the parameter learning problem of PBCNs is still open and remains challenging. Therefore, the main motivation of this article is to shorten this gap by designing some effective parameter learning algorithms. The main technical contributions can be highlighted as follows: (1) the algebraic representation of PBCNs with input-output data is provided to simplify the learning problem; (2) two recursive STP-based algorithms are designed to calculate the forward and backward probabilities; (3) the expectation maximization (EM) algorithm is developed to learn the model parameters; and (4) a useful index is introduced to evaluate the performance of the proposed parameter learning algorithm.

The rest of this paper is arranged as follows. We introduce several fundamental preliminaries with respect to the ASSR of PBCNs and formulate the problem in Section 2. In Section 3, we present the main results of the parameter learning of PBCNs with input-output data and introduce an index to evaluate the performance of the developed parameter learning algorithm. In Section 4, two numerical examples are provided to demonstrate the validity of the proposed learning strategy. Section 5 gives a concise conclusion of this paper.

2. Preliminaries and Problem Statements

For convenience, we first introduce some necessary notations. $\mathbb{R}^p$ and $\mathbb{R}^{p\times q}$ stand for the p-dimensional Euclidean space and the set of all $p\times q$ real matrices, respectively. I_p is the identity matrix with degree p. $\mathbb{N}$ means the set of non-negative integers. δ_p^i represents the ith column of I_p. Δ_p refers to the delta set $\{\delta_p^i \mid i=1,2,\cdots,p\}$. $A\in\mathbb{R}^{p\times q}$ is called a logic matrix if $A=[\delta_p^{i_1}\ \delta_p^{i_2}\ \cdots\ \delta_p^{i_q}]$, where $i_1,i_2,\cdots,i_q\in\{1,2,\cdots,p\}$. For format compactness, the logic matrix A can alternatively be denoted as $\delta_p[i_1,i_2,\cdots,i_q]$. The set of all logic matrices of dimension $p\times q$ is denoted by $\mathcal{L}^{p\times q}$.

2.1. Mathematical Preliminaries

We present some indispensable preliminaries with respect to the STP technique, which will be quite instrumental in the subsequent study of the parameter learning problem.

Definition 1. [4] *Let matrices $A_1\in\mathbb{R}^{m\times n}$ and $A_2\in\mathbb{R}^{p\times q}$ be given. Then the STP of A_1 and A_2 can be defined by the following formula*:

$$A_1\ltimes A_2=(A_1\otimes I_{s/n})(A_2\otimes I_{s/p})$$

where s denotes the lowest common multiple of integers n and p.

Remark 1. *It is easy to see that the regular matrix multiplication can be regarded as a particular case of the STP of matrices. Therefore, the symbol "$\ltimes$" can be omitted if no confusions arise.*

Lemma 1. [4] *Some basic properties of the STP are listed as follows.*

(i) Let $A_1\in\mathbb{R}^p$ and $A_2\in\mathbb{R}^q$ be two column vectors. Then,

$$A_1\ltimes A_2=W_{[q,p]}\ltimes A_2\ltimes A_1.$$

(ii) Let the $p^2\times p$ logic matrix $M_p\triangleq[\delta_p^1\otimes\delta_p^1\ \ \delta_p^2\otimes\delta_p^2\ \ \cdots\ \ \delta_p^p\otimes\delta_p^p]$ and $\Gamma\in\Delta_p$ be given. Then,

$$\Gamma\ltimes\Gamma=M_p\Gamma.$$

For convenience, the logic variable $X \in \mathcal{D} = \{1,0\}$ is identified with the vector $x = \delta_2^{2-X}$. Based on Lemma 1, a matrix expression of the logical functions is given as follows.

Lemma 2. [4] *Let $g(x_1, x_2, \cdots, x_m): \mathcal{D}^m \to \mathcal{D}$ be a logical function. Then, there exists a unique matrix $\mathcal{G} \in \mathcal{L}^{2\times 2^m}$, namely, the structure matrix of g, such that*

$$g(x_1, x_2, \cdots, x_m) = \mathcal{G} \ltimes x_1 \ltimes \cdots \ltimes x_m = \mathcal{G} \ltimes_{i=1}^{m} x_i, \quad x_i \in \Delta_2.$$

Consider a PBCN with m control inputs, p output nodes and n state nodes. The state equation with a control input can be written as

$$X_i(t+1) = f_i^{\sigma_i}(U_1(t), \cdots, U_m(t), X_1(t), \cdots, X_n(t)) \tag{1a}$$

where $t \in \mathbb{N}$ denotes the discrete time instant. $U(t) \triangleq (U_1(t), \cdots, U_m(t))^\top \in \mathcal{D}^m$ represents the control input which is assumed to be known and deterministic, and $X(t) \triangleq (X_1(t), \cdots, X_n(t))^\top \in \mathcal{D}^n$ stands for the state vector. The logical functions $f_i^{\sigma_i}: \mathcal{D}^{n+m} \to \mathcal{D}$, $i = 1, 2, \cdots, n$, $\sigma_i = 1, 2, \cdots, l_i$, can be selected in the domain $\{f_i^1, f_i^2, \cdots, f_i^{l_i}\}$ with certain probability $\mathbb{P}_i^{\sigma_i}$, and $\sum_{\sigma_i=1}^{l_i} \mathbb{P}_i^{\sigma_i} = 1$. In this paper, the choose of each constituent BN is assumed to be independent. It is easy to verify that there are $L = \prod_{i=1}^{n} l_i$ possible realizations for the network, and the probability of selecting the realization $\mathbf{f}_l = (f_1^{\sigma_1}, f_2^{\sigma_2}, \cdots, f_n^{\sigma_n})$ is $\mathbb{P}_l = \prod_{i=1}^{n} \mathbb{P}_i^{\sigma_i}$. The observation output equation can be written as

$$Z_j(t) = h_j(X_1(t), \cdots, X_n(t), V_1(t), \cdots, V_r(t)) \tag{1b}$$

where the vector $Z(t) \triangleq (Z_1(t), \cdots, Z_p(t))^\top \in \mathcal{D}^p$ refers to the measurement output at discrete time instant t. $V(t) \triangleq (V_1(t), \cdots, V_r(t))^\top \in \mathcal{D}^r$ indicates the measurement noise obeying the Bernoulli distribution, i.e., $V_i(t) \sim B(1, p_i)$ where $i = 1, 2, \cdots, r$. The function $h_j: \mathcal{D}^{n+r} \to \mathcal{D}$, $j = 1, 2, \cdots, p$ denotes the time invariant logical function.

By taking advantage of the STP technique, the ASSR of system (1) can be obtained, which will essentially simplify the parameter learning problem investigated in this paper. Let $N \triangleq \Delta_{2^n}$, $M \triangleq \Delta_{2^m}$, $P \triangleq \Delta_{2^p}$ and $R \triangleq \Delta_{2^r}$. By denoting $x(t) = \ltimes_{i=1}^{n} x_i(t) \in \Delta_N$, $u(t) = \ltimes_{i=1}^{m} u_i(t) \in \Delta_M$, $z(t) = \ltimes_{i=1}^{p} z_i(t) \in \Delta_P$ and $v(t) = \ltimes_{i=1}^{r} v_i(t) \in \Delta_R$, one can obtain the componentwise algebraic form of system (1) as

$$\begin{cases} x_i(t+1) = F_i^{\sigma_i} u(t) x(t), & i = 1, \cdots, n, \\ z_j(t) = H_j v(t) x(t), & j = 1, \cdots, p. \end{cases}$$

Based on Lemmas 1-2, the corresponding ASSR of system (1) can be obtained as follows:

$$\begin{cases} x(t+1) = F_l u(t) x(t) \\ z(t) = H v(t) x(t) \end{cases}$$

where $\mathcal{F}_l = \mathcal{F}_1^{\sigma_1} \ltimes_{i=1}^{n} [(I_{MN} \otimes \mathcal{F}_i^{\sigma_i}) M_{m+n}] \in \mathcal{L}^{N\times MN}$ and $\mathcal{H} = \mathcal{H}_1 \ltimes_{j=1}^{p} [(I_{PN} \otimes \mathcal{H}_j) M_{p+n}] \in \mathcal{L}^{P\times PN}$. Here, M_{m+n} and M_{p+n} are the group power reducing matrices [4]. Let $Ex(t)$ be the expectation value of the state variable $x(t)$, we have

$$\begin{cases} Ex(t+1) = Fu(t) Ex(t) \\ z(t) = Hv(t) x(t) \end{cases} \tag{2}$$

where $F = \sum_{l=1}^{L} \mathbb{P}_l \mathcal{F}_l \in \mathcal{L}^{N\times MN}$.

Based on the ASSR of the PBCNs, we can obtain the following auxiliary result whose proof is straightforward, and hence, omitted here for brevity.

Lemma 3. *Consider the ASSR of system* (2).

(i) *For the given control input $u(t) = \delta_M^{u_t}$, the one-step state transition probability that drives the state form $x(t) = \delta_N^{i_t}$ to $x(t+1) = \delta_N^{i_{t+1}}$ can be computed as follows:*

$$p\left\{x(t+1) = \delta_N^{i_{t+1}} \middle| x(t) = \delta_N^{i_t}, u(t) = \delta_M^{u_t}\right\} = \left[F \ltimes \delta_M^{u_t}\right]_{i_{t+1}, i_t} \tag{3}$$

(ii) *The observation probability can be calculated as follows:*

$$p\left\{z(t) = \delta_P^{j_t} \middle| x(t) = \delta_N^{i_t}\right\} = \left[H\right]_{j_t, i_t} \tag{4}$$

where $H \triangleq \mathcal{H} \ltimes \left(\ltimes_{i=1}^{r} \begin{bmatrix} p_k & 1-p_k \end{bmatrix}^\top\right)$.

Remark 2. *Lemma 3 implies that the state process is Markov and the observations are conditionally independent given the states.*

2.2. Problem Formulation

For the PBCN (2), an important issue concerned in practical applications is to learn the model parameters which

greatly match the observed output sequence. Actually, the dynamics of system (2) can be depicted by the state transition probability matrix $\left[F \ltimes \delta_M^{u_t}\right]$ and the observation probability matrix H. Hereafter, the parameter learning problem of PBCNs is transformed into the problem of learning the above two probability matrices from the observed data. That is to say, instead of identifying system (1) directly, we are going to acquire its algebraic form (2) in the first step, and then, identify the corresponding system matrices. After that, it is possible to transfer the identified algebraic form (2) back into its logical form (1), see [35] for more details.

For convenience, the complete parameter set of the algebraic form (2) is denoted by $\theta = (F, H, \pi)$, where $\pi = (\pi_1, \pi_2, \cdots, \pi_N)$ represents the initial distribution of the state and $\pi_i \triangleq p\left\{x(0) = \delta_N^i\right\}$. The parameter θ can be derived by maximum-likelihood estimation as follows:

$$\hat{\theta} = \arg\max_{\theta} p\{z_0^T | \theta, u_0^{T-1}\} \tag{5}$$

where $z_0^T \triangleq (z(0), z(1), z(2), \cdots, z(T))$ represents the observation sequence and $u_0^T \triangleq (u(0), u(1), u(2), \cdots, u(T))$ represents the control input sequence.

Problem 1. *Given the observation sequence z_0^T and control sequence u_0^{T-1}, the aim of parameter learning of PBCNs with input-output data is to learn the parameter θ that maximizes the probability $p\left\{z_0^T \mid \theta, u_0^{T-1}\right\}$.*

Remark 3. *It is worth pointing out that the parameter learning problem for BNs can be addressed in two ways. The one is supervised learning, where many observation sequences and corresponding state sequences are known for inferring model parameters. The other is unsupervised learning, where only observation sequences are known for inferring model parameters.*

3. Main Results

The above proposed parameter learning problem will be studied in this section. We will firstly establish some auxiliary results for computing forward and backward probabilities to facilitate the parameter learning of PBCNs. Subsequently, the EM algorithm will be implemented to learn the system matrix of system (2) in a computationally efficient way. Finally, an important index will be put forward to evaluate the performance of the proposed parameter learning algorithm.

3.1. Forward and Backward Probabilities in PBCNs

For the ease of notations, the state sequence $(x(0), x(1), x(2), \cdots, x(T))$ is denoted by x_0^T. Without loss of generality, we assume that $x_0^T = (\delta_N^{i_0}, \delta_N^{i_1}, \cdots, \delta_N^{i_T})$, $u_0^{T-1} = (\delta_M^{u_0}, \delta_M^{u_1}, \cdots, \delta_M^{u_{T-1}})$ and $z_0^T = (\delta_P^{j_0}, \delta_P^{j_1}, \cdots, \delta_P^{j_T})$.

Theorem 1. *Consider system (2) with the control sequence u_0^{T-1} and observation sequence z_0^T. The joint probability of the system state $x(t) = \delta_N^{i_t}$ $(0 \leqslant t \leqslant T)$ and the partial observation sequence z_0^t on the condition of θ and u_0^{t-1} is denoted by*

$$\alpha_t(i_t) \triangleq p\left\{x(t) = \delta_N^{i_t}, z_0^t \middle| \theta, u_0^{t-1}\right\}. \tag{6}$$

Then, the forward probability $\alpha_t(i_t)$ can be computed in a recursive way as follows:

$$\alpha_{\kappa+1}(i_{\kappa+1}) = [H]_{j_{\kappa+1}, i_{\kappa+1}} \cdot \left[\sum_{i_\kappa=1}^{N} \left[F \ltimes \delta_M^{u_\kappa}\right]_{i_{\kappa+1}, i_\kappa} \cdot \alpha_\kappa(i_\kappa)\right], \quad \kappa = 0, 1, \cdots, t-1 \tag{7}$$

with the initial condition $\alpha_0(i_0) = \pi_{i_0} \cdot \left[H\right]_{j_0, i_0}$.

Proof of Theorem 1. According to the multiplication rule of probabilities and the definition of observation probabilities, the initial condition can be obtained as

$$\begin{aligned}
\alpha_0(i_0) &= p\left\{x(0) = \delta_N^{i_0}, z(0) = \delta_P^{j_0} \mid \theta\right\} \\
&= p\left\{x(0) = \delta_N^{i_0} \mid \theta\right\} \cdot p\left\{z(0) = \delta_P^{j_0} \mid x(0) = \delta_N^{i_0}, \theta\right\} \\
&= \pi_{i_0} \cdot \left[H\right]_{j_0, i_0}.
\end{aligned}$$

Then, we prove the validity of the recurrence formula (7). It is apparent from (6) that

$$\begin{aligned}
\alpha_{\kappa+1}(i_{\kappa+1}) &= p\left\{x(\kappa+1) = \delta_N^{i_{\kappa+1}}, z_0^{\kappa+1} \mid \theta, u_0^\kappa\right\} \\
&= p\left\{x(\kappa+1) = \delta_N^{i_{\kappa+1}}, z_0^\kappa \mid \theta, u_0^\kappa\right\} \cdot p\left\{z(\kappa+1) = \delta_M^{j_{\kappa+1}} \mid x(\kappa+1) = \delta_N^{i_{\kappa+1}}, z_0^\kappa, \theta, u_0^\kappa\right\} \\
&= p\left\{x(\kappa+1) = \delta_N^{i_{\kappa+1}}, z_0^\kappa \mid \theta, u_0^\kappa\right\} \cdot \left[H\right]_{j_{\kappa+1}, i_{\kappa+1}}.
\end{aligned} \tag{8}$$

According to the total probability theorem and the definition of the one-step transition probability (3), we obtain

$$
\begin{aligned}
p\left\{x(\kappa+1)=\delta_N^{i_{\kappa+1}}, z_0^{\kappa} \middle| \theta, u_0^{\kappa}\right\} &= \sum_{i_\kappa=1}^{N} p\left\{x(\kappa)=\delta_N^{i_\kappa}, x(\kappa+1)=\delta_N^{i_{\kappa+1}}, z_0^{\kappa} \middle| \theta, u_0^{\kappa}\right\} \\
&= \sum_{i_\kappa=1}^{N} p\left\{x(\kappa)=\delta_N^{i_\kappa}, z_0^{\kappa} \middle| \theta, u_0^{\kappa-1}\right\} \cdot p\left\{x(\kappa+1)=\delta_N^{i_{\kappa+1}} \middle| x(\kappa)=\delta_N^{i_\kappa}, \theta\right\} \\
&= \sum_{i_\kappa=1}^{N} \left[F \ltimes \delta_M^{u_\kappa}\right]_{i_{\kappa+1}, i_\kappa} \cdot \alpha_\kappa(i_\kappa). \qquad (9)
\end{aligned}
$$

By substituting (9) into (8), the recursion (7) is verified, which completes the proof of Theorem 1.

Theorem 2. *Consider system* (2) *with a control sequence* u_0^{T-1} *and an observation sequence* z_0^T. *The probability of observing* z_{t+1}^T *on the condition of* $x(t)=\delta_N^{i_t}$, *the model* θ *and the control sequence* u_t^{T-1} *is denoted by*

$$
\beta_t(i_t) \triangleq p\left\{z_{t+1}^T \middle| x(t)=\delta_N^{i_t}, \theta, u_t^{T-1}\right\}. \qquad (10)
$$

Then, the backward probability $\beta_t(i_t)$ *can be computed recursively as follows:*

$$
\beta_\kappa(i_\kappa) = \sum_{i_{\kappa+1}=1}^{N} \left[H\right]_{j_{\kappa+1}, i_{\kappa+1}} \cdot \beta_{\kappa+1}(i_{\kappa+1}) \cdot \left[F \ltimes \delta_M^{u_\kappa}\right]_{i_{\kappa+1}, i_\kappa}, \quad \kappa = T-1, T-2, \cdots, t \qquad (11)
$$

with the initial condition $\beta_T(i_T)=1$, $i_T=1,2,\ldots,N$.

Proof of Theorem 2. It follows from (6) and (10) that

$$
\begin{aligned}
\alpha_\kappa(i_\kappa)\cdot\beta_\kappa(i_\kappa) &= p\left\{x(\kappa)=\delta_N^{i_\kappa}, z_0^{\kappa} \middle| \theta, u_0^{\kappa-1}\right\} \cdot p\left\{z_{\kappa+1}^T \middle| x(\kappa)=\delta_N^{i_\kappa}, \theta, u_\kappa^{T-1}\right\} \\
&= p\left\{z_{\kappa+1}^T \middle| x(\kappa)=\delta_N^{i_\kappa}, \theta, u_\kappa^{T-1}\right\} \cdot p\left\{z_0^{\kappa} \middle| x(\kappa)=\delta_N^{i_\kappa}, \theta, u_0^{\kappa-1}\right\} p\left\{x(\kappa)=\delta_N^{i_\kappa} \middle| \theta, u_0^{\kappa-1}\right\} \\
&= p\left\{z_0^T \middle| x(\kappa)=\delta_N^{i_\kappa}, \theta, u_0^{T-1}\right\} \cdot p\left\{x(\kappa)=\delta_N^{i_\kappa} \middle| \theta, u_0^{\kappa-1}\right\} \\
&= p\left\{z_0^T, x(\kappa)=\delta_N^{i_\kappa} \middle| \theta, u_0^{T-1}\right\}.
\end{aligned}
$$

By letting $\kappa=T$, we have

$$
\beta_T(i_T) = \frac{p\left\{z_0^T, x(T)=\delta_N^{i_T} \middle| \theta, u_0^{T-1}\right\}}{\alpha_T(i_T)} = 1,
$$

which implies that $\beta_T(i_T)=1$, $i_T=1,\cdots,N$.

The next thing to do in the proof is to show the validity of the recursion (11). By resorting to the total probability theorem, we arrive at

$$
\begin{aligned}
\beta_\kappa(i_\kappa) &= \sum_{i_{\kappa+1}=1}^{N} p\left\{x(\kappa+1)=\delta_N^{i_{\kappa+1}}, z_{\kappa+1}^T \middle| x(\kappa)=\delta_N^{i_\kappa}, \theta, u_\kappa^{T-1}\right\} \\
&= \sum_{i_{\kappa+1}=1}^{N} p\left\{z_{\kappa+1}^T \middle| x(\kappa)=\delta_N^{i_\kappa}, x(\kappa+1)=\delta_N^{i_{\kappa+1}}, \theta, u_\kappa^{T-1}\right\} \cdot p\left\{x(\kappa+1)=\delta_N^{i_{\kappa+1}} \middle| x(\kappa)=\delta_N^{i_\kappa}, \theta, u_\kappa^{T-1}\right\} \\
&= \sum_{i_{\kappa+1}=1}^{N} p\left\{z_{\kappa+1}^T \middle| x(\kappa+1)=\delta_N^{i_{\kappa+1}}, \theta, u_\kappa^{T-1}\right\} \cdot p\left\{x(\kappa+1)=\delta_N^{i_{\kappa+1}} \middle| x(\kappa)=\delta_N^{i_\kappa}, \theta, u_\kappa^{T-1}\right\}.
\end{aligned}
$$

It follows from the definition of the backward probability that

$$
\begin{aligned}
\beta_\kappa(i_\kappa) &= \sum_{i_{\kappa+1}=1}^{N} p\left\{z(\kappa+1)=\delta_P^{j_{\kappa+1}} \middle| x(\kappa+1)=\delta_N^{i_{\kappa+1}}, \theta, u_\kappa^{T-1}\right\} \cdot p\left\{z_{\kappa+2}^T \middle| x(\kappa)=\delta_N^{i_\kappa}, \theta, u_{\kappa+1}^{T-1}\right\} \cdot \left[F \ltimes \delta_M^{u_\kappa}\right]_{i_{\kappa+1}, i_\kappa} \\
&= \sum_{i_{\kappa+1}=1}^{N} p\left\{z(\kappa+1)=\delta_P^{j_{\kappa+1}} \middle| x(\kappa+1)=\delta_N^{i_{\kappa+1}}, \theta, u_\kappa^{T-1}\right\} \cdot \beta_{\kappa+1}(i_{\kappa+1}) \cdot \left[F \ltimes \delta_M^{u_\kappa}\right]_{i_{\kappa+1}, i_\kappa} \\
&= \sum_{i_{\kappa+1}=1}^{N} \left[H\right]_{j_{\kappa+1}, i_{\kappa+1}} \cdot \beta_{\kappa+1}(i_{\kappa+1}) \cdot \left[F \ltimes \delta_M^{u_\kappa}\right]_{i_{\kappa+1}, i_\kappa}.
\end{aligned}
$$

This completes the proof of Theorem 2.

Remark 4. *The calculation process of the forward probability is based on the path of the state sequence. The key to the efficiency of the forward algorithm is to calculate the previous probability partially, and then use the path to recursively sum up the previous probability to avoid unnecessary repeated calculations. Specifically, we calculate N*

possible values of $\alpha_1(i_1)$ $(i_1 = 1,2,\cdots,N)$ *at the discrete time* $\kappa = 1$. *Similarly, we calculate* N *possible values of* $\alpha_{\kappa+1}(i_{\kappa+1})$ *by adding up all the previous probabilities* $\alpha_\kappa(i_\kappa)$ $(i_\kappa = 1,2,\cdots,N)$, *which avoids the redundancy of repeated calculations. At the same time, the reason for high computational efficiency of the backward algorithm is also the same. Each step only depends on the result of subsequent steps without repeated calculations. As shown in* Figure 1, *when calculating the probability of one or more states, if we can make full use of forward and backward algorithms at the same time, the calculation complexity will be greatly reduced.*

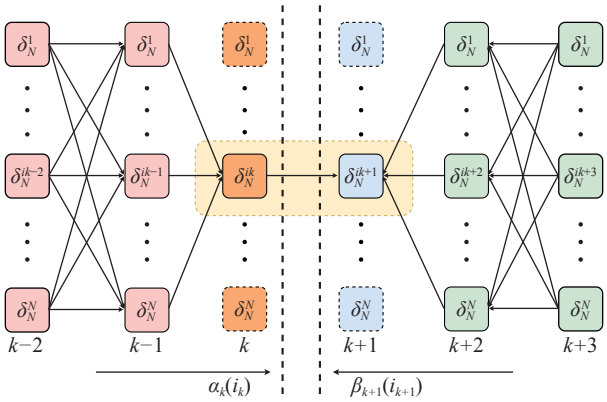


Figure 1. A trellis representation of the probability $\zeta_\kappa(i_\kappa, i_{\kappa+1})$.

Theorem 3. *By using the forward and backward probabilities, the specific formula for a single state probability can be obtained as follows.*

(i) *The probability of* $x(\kappa) = \delta_N^{i_\kappa}$ *on the condition of the observation sequence* z_0^T, *the model parameter* θ *and the known control sequence* u_0^{T-1} *is denoted by* $\gamma_\kappa(i_\kappa) \triangleq p\left\{x(\kappa) = \delta_N^{i_\kappa} \mid z_0^T, \theta, u_0^{T-1}\right\}$. *Then, the probability* $\gamma_\kappa(i_\kappa)$ *can be computed as follows:*

$$\gamma_\kappa(i_\kappa) = \frac{\alpha_\kappa(i_\kappa)\cdot\beta_\kappa(i_\kappa)}{\sum_{i_\kappa=1}^{N}\alpha_\kappa(i_\kappa)\cdot\beta_\kappa(i_\kappa)}. \tag{12}$$

(ii) *The joint probability of* $x(\kappa) = \delta_N^{i_\kappa}$ *and* $x(\kappa+1) = \delta_N^{i_{\kappa+1}}$ *on the condition of the observation sequence* z_0^T, *the model parameter* θ *and the known control sequence* u_0^T *is denoted by* $\zeta_\kappa(i_\kappa, i_{\kappa+1}) \triangleq p\left\{x(\kappa) = \delta_N^{i_\kappa}, x(\kappa+1) = \delta_N^{i_{\kappa+1}} \mid z_0^T, \theta, u_0^{T-1}\right\}$. *Then, the probability* $\zeta_\kappa(i_\kappa, i_{\kappa+1})$ *can be computed as follows:*

$$\zeta_\kappa(i_\kappa, i_{\kappa+1}) = \frac{\alpha_\kappa(i_\kappa)\left[F \ltimes \delta_M^{u_\kappa}\right]_{i_{\kappa+1},i_\kappa}\left[H\right]_{j_{\kappa+1},i_{\kappa+1}}\beta_{\kappa+1}(i_{\kappa+1})}{\sum_{i_\kappa=1}^{N}\sum_{i_{\kappa+1}=1}^{N}\alpha_\kappa(i_\kappa)\left[F \ltimes \delta_M^{u_\kappa}\right]_{i_{\kappa+1},i_\kappa}\left[H\right]_{j_{\kappa+1},i_{\kappa+1}}\beta_{\kappa+1}(i_{\kappa+1})}. \tag{13}$$

Proof of Theorem 3. According to the Bayesian formula, one has

$$\gamma_\kappa(i_\kappa) = \frac{p\left\{x(\kappa) = \delta_N^{i_\kappa}, z_0^T \mid \theta, u_0^{T-1}\right\}}{p\left\{z_0^T \mid \theta, u_0^{T-1}\right\}} = \frac{p\left\{x(\kappa) = \delta_N^{i_\kappa}, z_0^T \mid \theta, u_0^{T-1}\right\}}{\sum_{i_\kappa=1}^{N} p\left\{x(\kappa) = \delta_N^{i_\kappa}, z_0^T \mid \theta, u_0^{T-1}\right\}}. \tag{14}$$

By the definition of $\alpha_\kappa(i_\kappa)$ and $\beta_\kappa(i_\kappa)$, it is not difficult to verify that

$$\begin{aligned}
\alpha_\kappa(i_\kappa)\cdot\beta_\kappa(i_\kappa) &= p\left\{x(\kappa) = \delta_N^{i_\kappa}, z_0^\kappa \mid \theta, u_0^{\kappa-1}\right\}\cdot p\left\{z_{\kappa+1}^T \mid x(\kappa) = \delta_N^{i_\kappa}, \theta, u_k^{T-1}\right\} \\
&= p\left\{z_0^\kappa \mid x(\kappa) = \delta_N^{i_\kappa}, \theta, u_0^{\kappa-1}\right\}\cdot p\left\{x(\kappa) = \delta_N^{i_\kappa} \mid \theta, u_0^{\kappa-1}\right\}\cdot p\left\{z_{\kappa+1}^T \mid x(\kappa) = \delta_N^{i_\kappa}, \theta, u_\kappa^{T-1}\right\} \\
&= p\left\{z_0^T \mid x(\kappa) = \delta_N^{i_\kappa}, \theta, u_0^{T-1}\right\}\cdot p\left\{x(\kappa) = \delta_N^{i_\kappa} \mid \theta, u_0^{T-1}\right\} \\
&= p\left\{x(\kappa) = \delta_N^{i_\kappa}, z_0^T \mid \theta, u_0^{T-1}\right\}.
\end{aligned}$$

Taking (14) into consideration, we have (12).

We are now in the position to verify the correctness of (13). From the definition of $\zeta_\kappa(i_\kappa, i_{\kappa+1})$, we have

$$\zeta_\kappa(i_\kappa, i_{\kappa+1}) = \frac{p\left\{x(\kappa) = \delta_N^{i_\kappa}, x(\kappa+1) = \delta_N^{i_{\kappa+1}}, z_0^\kappa \mid \theta, u_0^{\kappa-1}\right\}}{\sum_{i_\kappa=1}^{N}\sum_{i_{\kappa+1}=1}^{N} p\left\{x(\kappa) = \delta_N^{i_\kappa}, x(\kappa+1) = \delta_N^{i_{\kappa+1}}, z_0^\kappa \mid \theta, u_0^{\kappa-1}\right\}}. \tag{15}$$

In the light of

$$p\left\{x(\kappa)=\delta_N^{i_\kappa}, x(\kappa+1)=\delta_N^{i_{\kappa+1}}, z_0^\kappa \mid \theta, u_0^{\kappa-1}\right\}=\alpha_\kappa(i_\kappa)\cdot\left[F\ltimes\delta_M^{u_\kappa}\right]_{i_{\kappa+1},i_\kappa}\cdot\left[H\right]_{j_{\kappa+1},i_{\kappa+1}}\cdot\beta_{\kappa+1}(i_{\kappa+1}), \tag{16}$$

which, together with (15), implies that (13) holds. The proof is complete.

Remark 5. The forward and backward probabilities allow for the efficient computation of the likelihood of observed data, which is a critical component of the EM algorithm that will be discussed below. Instead of recomputing the probabilities from scratch at each iteration, the existing calculations are reused to make the algorithm significantly faster with the help of the forward and backward probabilities.

3.2. Parameter Learning of PBCNs

The EM algorithm, originally put forward by Dempster, is an iterative technique for computing maximum likelihood estimation with incomplete data [36]. The EM algorithm consists of two steps: an expectation step (i.e., the E-step) and a maximization step (i.e., the M-step) Algorithm 1. Specifically, the expectation is with respect to the latent variables, which utilizes the present estimate of the parameters conditioned on the observed data. The M-step provides a novel estimation of parameters. The two steps are iterated until convergence. We first define the incomplete data as $D^{Inc}\triangleq(u_0^{T-1},z_0^T)$ and complete data as $D^c\triangleq(u_0^{T-1},z_0^T,x_0^T)$. Then, we can acquire the likelihood functions of the above two types of data, the likelihood function of the incomplete data $L_{Inc}(\theta,D^{Inc})\triangleq p\left\{z_0^T \mid u_0^{T-1},\theta\right\}$ and the likelihood function of the complete data $L_c(\theta,D^c)\triangleq p\left\{z_0^T,x_0^T \mid u_0^{T-1},\theta\right\}$.

Algorithm 1 EM algorithm for parameter learning of PBCNs

Input: the observation sequence z_0^T and the control sequence u_0^T

Output: the model parameters θ

Randomly initialize $\theta^{(0)}$
Set the maximum number of iterations M
Set the convergence threshold ϵ
for $t=1$ to M **do**
  if $\|\theta^{(t-1)}-\theta\|\geqslant\epsilon$ **then**
    for $t=0$ to T **do**
      E-step:
      compute $Q(\theta,\theta^{(t-1)})$ via (17)
      M-step:
      calculate $\arg\max_\theta Q(\theta,\theta^{(t-1)})$ via (21) and (22)
    end for
    Set new $\theta^*:=\theta^{(t)}$
  else
    Return optimal model parameters $\theta^{(t)}$
  end if
end for

The EM algorithm for the parameter identification problem of PBCNs can be conducted as follows (provided in Algorithm 1). First, the algorithm will initialize the system parameter $\theta^{(0)}$. After that, the E-step and the M-step are alternated until the change of θ is less than a given threshold. Algorithm 1 shows the general progress of estimating the optimal parameter θ^*. Each step of the iteration will make the log-likelihood function increase and the algorithm will make the likelihood function approach to a local maximum value. In Algorithm 1, the Q function is defined as follows:

$$\begin{aligned}Q(\theta,\theta^{(t-1)})&=E_{x_0^T}\left[\log L_c(\theta,D^c)\mid D^{Inc},\theta^{(t-1)}\right]\\&=E_{x_0^T}\left[\log p\left\{z_0^T,x_0^T \mid u_0^{T-1},\theta\right\}\mid u_0^{T-1},z_0^T,\theta^{(t-1)}\right]\\&=\sum_{x_0^T}\log p\left\{z_0^T,x_0^T \mid u_0^{T-1},\theta\right\}\cdot p\left\{x_0^T \mid u_0^{T-1},z_0^T,\theta^{(t-1)}\right\}\cdot\frac{1}{p\left\{z_0^T \mid \theta^{(t-1)},u_0^{T-1}\right\}}\end{aligned}$$

where $\theta^{(t-1)}$ means the present parameter estimate and θ denotes the latest parameter that increases the value of Q. It is obvious that x_0^T and $\theta^{(t-1)}$ can be seen as constants, and θ is the only variable required to be optimized. Note that

the last term in the right-hand side (RHS) of the above equation can be treated as a constant. The Q function can be simplified as

$$Q(\theta,\theta^{(t-1)}) = \sum_{x_0^T} \log p\left\{z_0^T, x_0^T \mid \theta, u_0^{T-1}\right\} \cdot p\left\{x_0^T \mid u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\}. \tag{17}$$

The next step is to find $\theta^{(t)}$ that maximizes the value of the Q function (M-step), that is

$$\theta^{(t)} = \arg\max_{\theta} Q(\theta,\theta^{(t-1)}). \tag{18}$$

Given the state sequence and control sequence, we have

$$p\left\{z_0^T, x_0^T \mid \theta, u_0^{T-1}\right\} = p\left\{x_0^T \mid \theta, u_0^{T-1}\right\} \cdot p\left\{z_0^T \mid x_0^T, \theta, u_0^{T-1}\right\} = \pi_{i_0} \cdot \prod_{t=0}^{T-1} \left[F \ltimes \delta_M^{u_t}\right]_{i_{t+1},i_t} \cdot \prod_{t=0}^{T} [H]_{j_t,i_t}.$$

By taking the logarithm of the above expression, the Q function can be rewritten as

$$\begin{aligned} Q(\theta,\theta^{(t-1)}) = & \sum_{x_0^T} \log \pi_{i_0} \cdot p\left\{x_0^T \middle| u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\} \\ & + \sum_{x_0^T} \left(\sum_{t=0}^{T-1} \log \left[F \ltimes \delta_M^{u_t}\right]_{i_{t+1},i_t} \right) \cdot p\left\{x_0^T \middle| u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\} \\ & + \sum_{x_0^T} \left(\sum_{t=0}^{T} \log [H]_{j_t,i_t} \right) \cdot p\left\{x_0^T \middle| u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\}. \end{aligned} \tag{19}$$

In view of the complexity of (19), we tend to, respectively, maximize the first term including π_{i_0}, the second term including $\left[F \ltimes \delta_M^{u_t}\right]_{i_{t+1},i_t}$ and the last term including $[H]_{j_t,i_t}$ since they are independently unrelated.

The first item of formula (19) can be rewritten as

$$\sum_{x_0^T} \log \pi_{i_0} \cdot p\left\{x_0^T \mid u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\} = \sum_{i_0=1}^{N} \log \pi_{i_0} \cdot p\left\{x(0) = \delta_N^{i_0} \mid u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\}.$$

Since the initial probability distribution satisfies $\sum_{i_0=1}^{N} \pi_{i_0} = 1$, we introduce the Lagrange multiplier λ to generate the following Lagrange function. By taking the partial derivative of the Lagrange function with respect to the variable π_{i_0}, we can obtain

$$\frac{\partial}{\partial \pi_{i_0}} \left[\sum_{i_0=1}^{N} \log \pi_{i_0} \cdot p\left\{x(0) = \delta_N^{i_0} \mid u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\} + \lambda \left(\sum_{i_0=1}^{N} \pi_{i_0} - 1 \right) \right] = 0.$$

Then, let the value of the formula to be zero after conducting the partial derivative, and we have

$$p\left\{x(0) = \delta_N^{i_0} \middle| u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\} + \lambda \pi_{i_0} = 0. \tag{20}$$

In the light of the fact that $\sum_{i_0=1}^{N} \pi_{i_0} = 1$, we take the summation of equation (20) from $i_0 = 1$ to N and obtain $\lambda = -p\left\{z_0^T \mid \theta^{(t)}, u_0^{T-1}\right\}$. Plugging the result into (20) yields

$$\pi_{i_0} = \frac{p\left\{x(0) = \delta_N^{i_0} \middle| u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\}}{p\left\{z_0^T \mid \theta^{(t-1)}, u_0^{T-1}\right\}}.$$

Moreover, combining the above result with equation (12), we have $\pi_{i_0} = \gamma_1(i_0)$.

Similarly, the middle item of formula (19) can be converted into the following form to determine $\left[F \ltimes \delta_M^{u_t}\right]_{i_{t+1},i_t}$.

$$\begin{aligned} & \sum_{x_0^T} \left(\sum_{t=0}^{T-1} \log \left[F \ltimes \delta_M^{u_t}\right]_{i_{t+1},i_t} \right) p\left\{x_0^T \mid u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\} \\ = & \sum_{i_t=1}^{N} \sum_{i_{t+1}}^{N} \sum_{t=0}^{T} \log \left[F \ltimes \delta_M^{u_t}\right]_{i_{t+1},i_t} \cdot p\left\{x(t) = \delta_N^{i_t}, x(t+1) = \delta_N^{i_{t+1}} \mid u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\}. \end{aligned}$$

Then, it can be obtained by introducing another Lagrange multiplier and multiplying the restraint condition $\sum_{i_{t+1}=1}^{N}\left[F \ltimes \delta_M^{u_t}\right]_{i_{t+1},i_t} = 1$ that

$$\left[F \ltimes \delta_M^{u_t}\right]_{i_{t+1},i_t} = \frac{\sum_{t=0}^{T}\left\{x(t)=\delta_N^{i_t}, x(t+1)=\delta_N^{i_{t+1}} \mid u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\}}{\sum_{t=0}^{T}\left\{x(t)=\delta_N^{i_t} \mid u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\}}.$$

After applying the expressions of $\gamma_t(i_t)$ and $\zeta_t(i_t, i_{t+1})$, we arrive at

$$\left[F \ltimes \delta_M^{u_t}\right]_{i_{t+1},i_t} = \frac{\sum_{t=0}^{T}\zeta_t(i_t, i_{t+1})}{\sum_{t=0}^{T}\gamma_t(i_t)}. \tag{21}$$

To find $\left[H\right]_{j_t,i_t}$, the third item of (19) can be transformed into

$$\sum_{x_0^T}\left(\sum_{t=0}^{T}\log\left[H\right]_{j_t,i_t}\right)p\left\{x_0^T \mid u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\} = \sum_{i_t=1}^{N}\sum_{t=0}^{T}\log\left[H\right]_{j_t,i_t}p\left\{x(t)=\delta_N^{i_t} \mid u_0^{T-1}, z_0^T, \theta^{(t-1)}\right\}.$$

We can analogously introduce a Lagrange multiplier under the condition of $\sum_{j_t=1}^{N}\left[H\right]_{j_t,i_t} = 1$. Thus, there holds

$$\left[H\right]_{j_t,i_t} = \frac{\sum_{t=0}^{T}\boldsymbol{\delta}\left[z(t), \delta_P^{j_t}\right]\cdot\gamma_t(i_t)}{\sum_{t=0}^{T}\gamma_t(i_t)} \tag{22}$$

where $\boldsymbol{\delta}\left[z(t), \delta_P^{j_t}\right]$ denotes the Kronecker delta function.

Remark 6. *In the following, we introduce the index ρ to describe the performance of the proposed parameter learning algorithm. Let the matrix G be the actual system parameter matrix and $\hat{G}$ be the learned parameter matrix, and both matrices have m rows and n columns. Then, the index of the performance of the parameter learning algorithm can be defined as follows:*

$$\rho = 1 - \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|\hat{G}_{ij} - G_{ij}\right|}{\sum_{i=1}^{m}\sum_{j=1}^{n}G_{ij}}$$

where G_{ij} is the entry located in the i-th row and j-th column of the matrix G. It is not difficult to infer from the above definition that the parameter learning performance index ρ is theoretically a number more than 0 and less than 1, without any unit. When the value of ρ is more closer to 1, it means that the performance of the parameter learning algorithm is better, and vice versa.

Remark 7. *In fact, the transition probability matrix in the network that we learn can further be used to reconstruct the system back to the logic expression* [35]. *At the same time, the reconstruction process from the probabilistic state transition matrix to the PBCNs is not unique, so it is worth further improving the reconstruction algorithm of the logical expression of the PBCNs. More deeply, the reconstructed network cannot be completely consistent with the real network, so there is still much optimization work to be done in the process of network reconstruction.*

4. Numerical Examples

Example 1. Consider a PBCN with 12 state nodes and 4 inputs. The network structure of the PBCN is shown in Figure 2, and the logic model is given as follows:

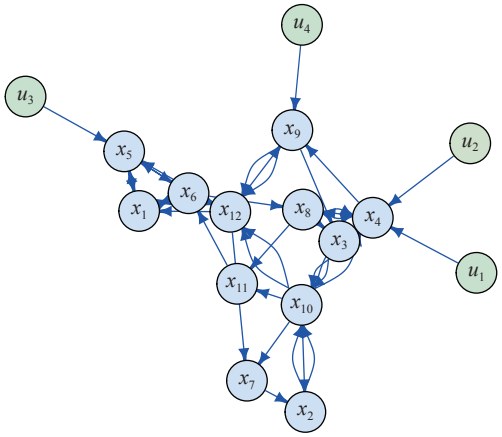


Figure 2. The model construction of the PBCN example in system (23).

$$\begin{cases} x_1(\kappa+1) = (\neg x_5(\kappa) \wedge (\neg x_6(\kappa) \wedge x_{12}(\kappa))) \vee (x_5(\kappa) \wedge (\neg x_6(\kappa) \vee x_{12}(\kappa))), \quad \mathbb{P}_1^{\sigma_1} = 1 \\ x_2(\kappa+1) = x_7(\kappa) \vee x_{10}(\kappa), \quad \mathbb{P}_1^{\sigma_2} = 1 \\ x_3(\kappa+1) = (\neg x_8(\kappa) \wedge (\neg x_4(\kappa) \wedge x_9(\kappa))) \vee (x_8(\kappa) \wedge (\neg x_4(\kappa) \vee x_9(\kappa))), \quad \mathbb{P}_1^{\sigma_3} = 1 \\ x_4(\kappa+1) = \begin{cases} x_8(\kappa) \wedge x_{10}(\kappa) \vee u_1(\kappa), & \mathbb{P}_1^{\sigma_4} = 0.6 \\ x_8(\kappa) \vee x_{10}(\kappa) \wedge u_2(\kappa), & \mathbb{P}_2^{\sigma_4} = 0.4 \end{cases} \\ x_5(\kappa+1) = (\neg x_{12}(\kappa) \wedge x_1(\kappa)) \vee u_3(\kappa), \quad \mathbb{P}_1^{\sigma_5} = 1 \\ x_6(\kappa+1) = (\neg x_{11}(\kappa) \wedge (\neg x_1(\kappa) \wedge x_5(\kappa))) \vee (x_{12}(\kappa) \wedge x_5(\kappa)), \quad \mathbb{P}_1^{\sigma_6} = 1 \\ x_7(\kappa+1) = x_{10}(\kappa) \vee x_{12}(\kappa), \quad \mathbb{P}_1^{\sigma_7} = 1 \\ x_8(\kappa+1) = x_4(\kappa) \wedge x_6(\kappa), \quad \mathbb{P}_1^{\sigma_8} = 1 \\ x_9(\kappa+1) = x_4(\kappa) \wedge x_{12}(\kappa) \vee u_4(\kappa), \quad \mathbb{P}_1^{\sigma_9} = 1 \\ x_{10}(\kappa+1) = \begin{cases} x_2(\kappa) \vee x_3(\kappa), & \mathbb{P}_1^{\sigma_{10}} = 0.7 \\ x_2(\kappa) \wedge x_3(\kappa), & \mathbb{P}_2^{\sigma_{10}} = 0.3 \end{cases} \\ x_{11}(\kappa+1) = \neg x_8(\kappa) \wedge x_{10}(\kappa), \quad \mathbb{P}_1^{\sigma_{11}} = 1 \\ x_{12}(\kappa+1) = \begin{cases} \neg x_6(\kappa) \wedge x_9(\kappa) \wedge x_{10}(\kappa), & \mathbb{P}_1^{\sigma_{12}} = 0.2 \\ \neg x_6(\kappa) \vee x_9(\kappa) \vee x_{10}(\kappa), & \mathbb{P}_2^{\sigma_{12}} = 0.8 \end{cases} \end{cases} \tag{23}$$

The corresponding output observation equation can be described as

$$\begin{cases} z_1(\kappa) = x_2(\kappa) \wedge x_7(\kappa) \vee v_1(\kappa) \\ z_2(\kappa) = \neg x_9\kappa) \wedge v_2(\kappa) \\ z_3(\kappa) = x_6(\kappa) \vee x_{10}(\kappa) \\ z_4(\kappa) = x_8(\kappa) \wedge (\neg x_5(\kappa) \vee x_3(\kappa)) \\ z_5(\kappa) = \neg x_{10}(\kappa) \vee x_7(\kappa) \wedge x_2(\kappa) \\ z_6(\kappa) = x_4(\kappa) \vee x_{12}(\kappa) \wedge v_3(\kappa) \\ z_7(\kappa) = x_{10}(\kappa) \wedge \neg x_9(\kappa) \\ z_8(\kappa) = \neg x_4(\kappa) \vee v_4(\kappa) \end{cases} \tag{24}$$

Based on Lemmas 1-2, the algebraic representation of model (23) can naturally be obtained as

$$\begin{cases} x_1(\kappa+1) = F^{\sigma_1} u(\kappa) x(\kappa) \\ x_2(\kappa+1) = F^{\sigma_2} u(\kappa) x(\kappa) \\ \quad \vdots \\ x_{12}(\kappa+1) = F^{\sigma_{12}} u(\kappa) x(\kappa) \end{cases} \tag{25}$$

where $F^{\sigma_i} \in \mathcal{L}^{2^{12} \times 2^{16}}$ $(i = 1,2,3,5,6,7,8,9,11)$ and $F^{\sigma_j} \in \left\{ F_1^{\sigma_j}, F_2^{\sigma_j} \right\} \in \mathcal{L}^{2^{12} \times 2^{16}}$ $(j = 4,10,12)$. Then, the following 8 structure matrices can be calculated as

$$\begin{cases} F_1 = F^{\sigma_1} \cdots F^{\sigma_3} * F_1^{\sigma_4} * F^{\sigma_5} \cdots F^{\sigma_9} * F_1^{\sigma_{10}} * F^{\sigma_{11}} * F_1^{\sigma_{12}} \\ F_2 = F^{\sigma_1} \cdots F^{\sigma_3} * F_1^{\sigma_4} * F^{\sigma_5} \cdots F^{\sigma_9} * F_1^{\sigma_{10}} * F^{\sigma_{11}} * F_2^{\sigma_{12}} \\ F_3 = F^{\sigma_1} \cdots F^{\sigma_3} * F_1^{\sigma_4} * F^{\sigma_5} \cdots F^{\sigma_9} * F_2^{\sigma_{10}} * F^{\sigma_{11}} * F_1^{\sigma_{12}} \\ F_4 = F^{\sigma_1} \cdots F^{\sigma_3} * F_1^{\sigma_4} * F^{\sigma_5} \cdots F^{\sigma_9} * F_2^{\sigma_{10}} * F^{\sigma_{11}} * F_2^{\sigma_{12}} \\ F_5 = F^{\sigma_1} \cdots F^{\sigma_3} * F_2^{\sigma_4} * F^{\sigma_5} \cdots F^{\sigma_9} * F_1^{\sigma_{10}} * F^{\sigma_{11}} * F_2^{\sigma_{12}} \\ F_6 = F^{\sigma_1} \cdots F^{\sigma_3} * F_2^{\sigma_4} * F^{\sigma_5} \cdots F^{\sigma_9} * F_2^{\sigma_{10}} * F^{\sigma_{11}} * F_2^{\sigma_{12}} \\ F_7 = F^{\sigma_1} \cdots F^{\sigma_3} * F_2^{\sigma_4} * F^{\sigma_5} \cdots F^{\sigma_9} * F_1^{\sigma_{10}} * F^{\sigma_{11}} * F_1^{\sigma_{12}} \\ F_8 = F^{\sigma_1} \cdots F^{\sigma_3} * F_2^{\sigma_4} * F^{\sigma_5} \cdots F^{\sigma_9} * F_2^{\sigma_{10}} * F^{\sigma_{11}} * F_1^{\sigma_{12}}. \end{cases}$$

The algebraic representation form of model (23) can be written as

$$\begin{cases} Ex(\kappa+1) = Fu(\kappa)Ex(\kappa) \\ z(\kappa) = Hv(\kappa)x(\kappa) \end{cases} \tag{26}$$

where the logic matrix $H \in \mathcal{L}^{2^8\times 2^{20}}$, the logic matrix $F = P_1 \cdot F_1 + P_2 \cdot F_2 + \cdots + P_{12} \cdot F_{12} \in \mathcal{L}^{2^{12}\times 2^{16}}$, and each probability satisfies

$$\begin{cases} P_1 = \mathbb{P}_1^{\sigma_1}\cdots\mathbb{P}_1^{\sigma_3}\cdot\mathbb{P}_1^{\sigma_4}\cdot\mathbb{P}_1^{\sigma_5}\cdots\mathbb{P}_1^{\sigma_9}\cdot\mathbb{P}_1^{\sigma_{10}}\cdot\mathbb{P}_1^{\sigma_{11}}\cdot\mathbb{P}_1^{\sigma_{12}} = 0.084 \\ P_2 = \mathbb{P}_1^{\sigma_1}\cdots\mathbb{P}_1^{\sigma_3}\cdot\mathbb{P}_1^{\sigma_4}\cdot\mathbb{P}_1^{\sigma_5}\cdots\mathbb{P}_1^{\sigma_9}\cdot\mathbb{P}_1^{\sigma_{10}}\cdot\mathbb{P}_1^{\sigma_{11}}\cdot\mathbb{P}_2^{\sigma_{12}} = 0.336 \\ P_3 = \mathbb{P}_1^{\sigma_1}\cdots\mathbb{P}_1^{\sigma_3}\cdot\mathbb{P}_1^{\sigma_4}\cdot\mathbb{P}_1^{\sigma_5}\cdots\mathbb{P}_1^{\sigma_9}\cdot\mathbb{P}_2^{\sigma_{10}}\cdot\mathbb{P}_1^{\sigma_{11}}\cdot\mathbb{P}_1^{\sigma_{12}} = 0.036 \\ P_4 = \mathbb{P}_1^{\sigma_1}\cdots\mathbb{P}_1^{\sigma_3}\cdot\mathbb{P}_1^{\sigma_4}\cdot\mathbb{P}_1^{\sigma_5}\cdots\mathbb{P}_1^{\sigma_9}\cdot\mathbb{P}_2^{\sigma_{10}}\cdot\mathbb{P}_1^{\sigma_{11}}\cdot\mathbb{P}_2^{\sigma_{12}} = 0.144 \\ P_5 = \mathbb{P}_1^{\sigma_1}\cdots\mathbb{P}_1^{\sigma_3}\cdot\mathbb{P}_2^{\sigma_4}\cdot\mathbb{P}_1^{\sigma_5}\cdots\mathbb{P}_1^{\sigma_9}\cdot\mathbb{P}_1^{\sigma_{10}}\cdot\mathbb{P}_1^{\sigma_{11}}\cdot\mathbb{P}_2^{\sigma_{12}} = 0.224 \\ P_6 = \mathbb{P}_1^{\sigma_1}\cdots\mathbb{P}_1^{\sigma_3}\cdot\mathbb{P}_2^{\sigma_4}\cdot\mathbb{P}_1^{\sigma_5}\cdots\mathbb{P}_1^{\sigma_9}\cdot\mathbb{P}_2^{\sigma_{10}}\cdot\mathbb{P}_1^{\sigma_{11}}\cdot\mathbb{P}_2^{\sigma_{12}} = 0.096 \\ P_7 = \mathbb{P}_1^{\sigma_1}\cdots\mathbb{P}_1^{\sigma_3}\cdot\mathbb{P}_2^{\sigma_4}\cdot\mathbb{P}_1^{\sigma_5}\cdots\mathbb{P}_1^{\sigma_9}\cdot\mathbb{P}_1^{\sigma_{10}}\cdot\mathbb{P}_1^{\sigma_{11}}\cdot\mathbb{P}_1^{\sigma_{12}} = 0.056 \\ P_8 = \mathbb{P}_1^{\sigma_1}\cdots\mathbb{P}_1^{\sigma_3}\cdot\mathbb{P}_2^{\sigma_4}\cdot\mathbb{P}_1^{\sigma_5}\cdots\mathbb{P}_1^{\sigma_9}\cdot\mathbb{P}_2^{\sigma_{10}}\cdot\mathbb{P}_1^{\sigma_{11}}\cdot\mathbb{P}_1^{\sigma_{12}} = 0.024. \end{cases}$$

The measurement of the output observation value is shown in Figure 3 by resorting to the previously proposed algorithms to learn the parameter F under the designated output observation sequence. By resolving the former problem (5), we conduct three sets of parameter learning experiments and take the mean and variance at each iteration step. The mean value of the parameter learning performance index ρ of matrices F and H are $\rho_F = 0.84$ and $\rho_H = 0.87$. As the iteration step increases, the parameter learning performance indexes of the two matrices continuously increase and finally stabilize at a certain high level shown in Figure 4.

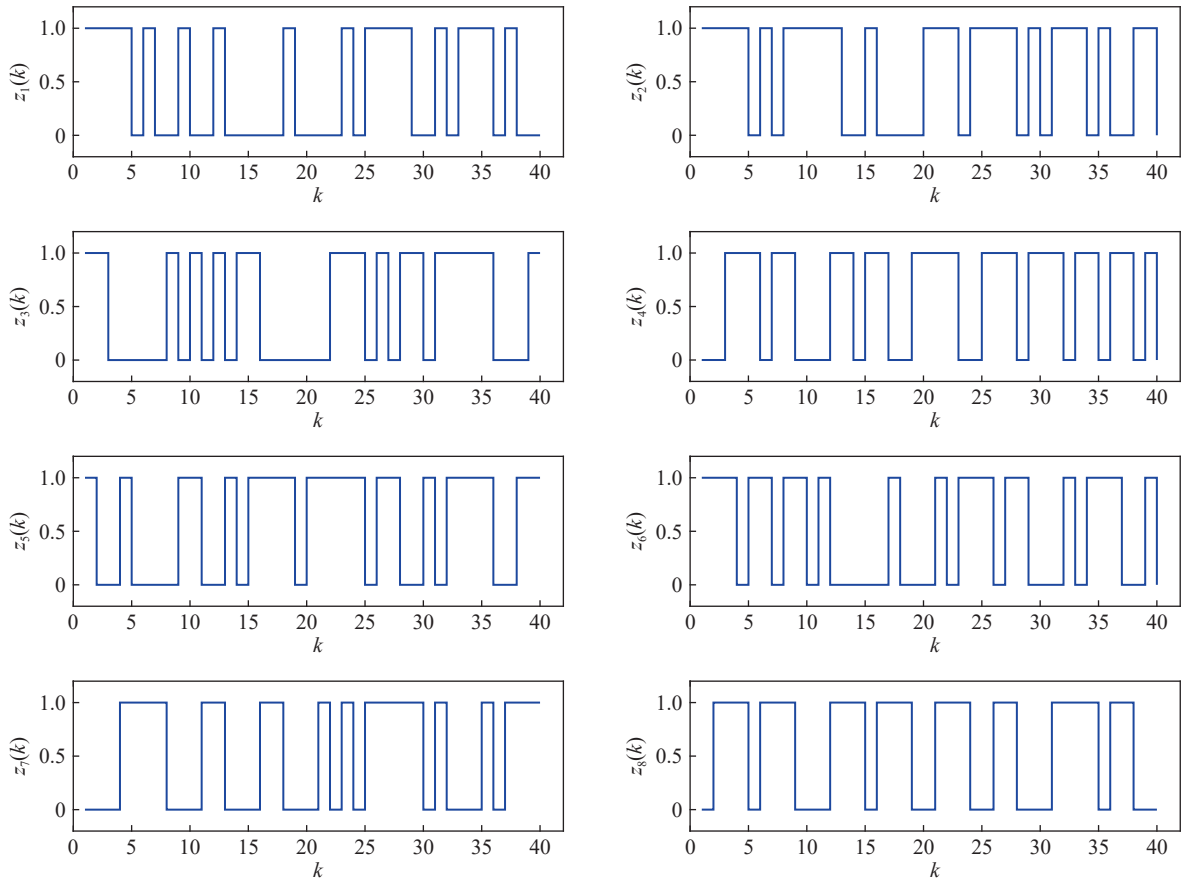


Figure 3. The measurement of the output observation value in system (24).

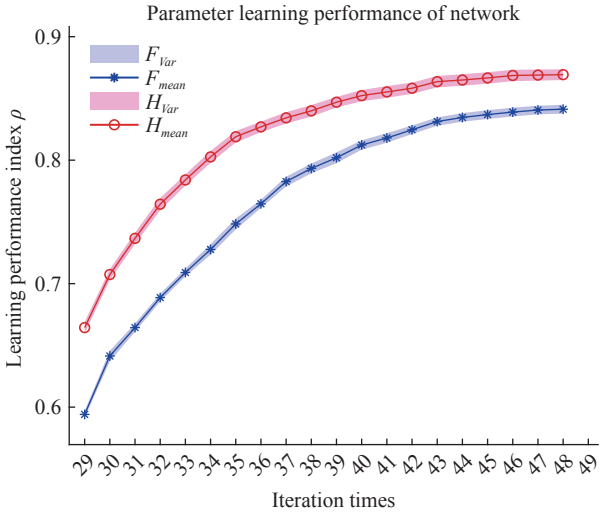


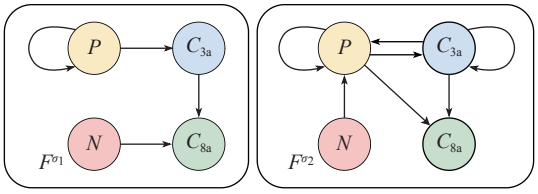
Figure 4. The parameter learning performance in system (26).

Example 2. We consider a PBCN originated from an actual *apoptosis network* model, which is an essential biological regulatory network that keeps a fully functional organism. A faulty apoptotic network can lead to a variety of diseases with either insufficient apoptosis or excessive apoptosis [37]. Then, the logic model of the *apoptosis network* can be described as follows:

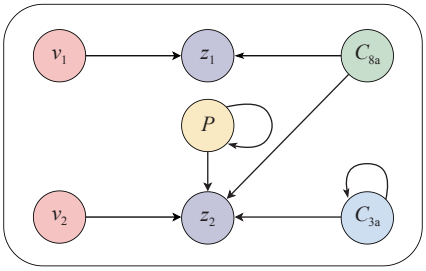
$$F^{\sigma_1}: \begin{cases} P(\kappa+1) = P(\kappa) \\ C_{3a}(\kappa+1) = \neg P(\kappa) \wedge C_{8a}(\kappa) \\ C_{8a}(\kappa+1) = C_{3a}(\kappa) \vee N(\kappa), \quad \mathbb{P}_1^{\sigma_1} = 0.2 \end{cases} \tag{27a}$$

$$F^{\sigma_2}: \begin{cases} P(\kappa+1) = (\neg P(\kappa) \wedge C_{3a}(\kappa)) \vee N(\kappa) \\ C_{3a}(\kappa+1) = P(\kappa) \wedge C_{3a}(\kappa) \\ C_{8a}(\kappa+1) = P(\kappa) \wedge C_{3a}(\kappa), \quad \mathbb{P}_2^{\sigma_2} = 0.8 \end{cases} \tag{27b}$$

where $C_{8a}(\kappa)$, $C_{3a}(\kappa)$ and $P(\kappa)$ indicate the active caspase 8 (C8a), active caspase 3 (C3a) and concentration degree in the inhibitor of apoptosis proteins, respectively; and $N(\kappa)$ stands for the tumor necrosis factor's concentration degree. For the ease of notations, let $x(\kappa)$ and $u(k)$ be, respectively, the state vector and control input vector of the apoptosis network, that is $x(\kappa) = (P(\kappa), C_{3a}(\kappa), C_{8a}(\kappa)) \in \mathcal{D}^3$ and $u(\kappa) = N(\kappa) \in \mathcal{D}$. System observations contain noise disturbances with $v(\kappa) = (v_1(\kappa), v_2(\kappa)) \in \mathcal{D}^2$, where $v_i(\kappa) \sim B(1, p_i)$ for $i = 1, 2$. The network structure of the PBCN (27) is shown in Figure 5. Each state node can freely switch its state in the range of network F^{σ_1} and F^{σ_2} with the probabilities $\mathbb{P}^{\sigma_1} = 0.2$ and $\mathbb{P}^{\sigma_2} = 0.8$.



(a) The network structural diagram of state nodes in system (27)



(b) The structural diagram of output observation nodes in system (28)

Figure 5. The network structure of PBCN in Example 4.

The output equation can be described as

$$\begin{cases} z_1(\kappa) = C_{8a}(\kappa) \vee v_1(\kappa) \\ z_2(\kappa) = (P(\kappa) \wedge \neg C_{3a}(\kappa)) \vee (\neg P(\kappa) \wedge v_2(\kappa)). \end{cases} \tag{28}$$

Let $x(\kappa) = P(\kappa) \ltimes C_{3a}(\kappa) \ltimes C_{8a}(\kappa)$, $u(\kappa) = N(\kappa)$, $z(\kappa) = \ltimes_{i=1}^{2} z_i(\kappa)$ and $v(\kappa) = \ltimes_{i=1}^{2} v_i(\kappa)$. With the help of the STP technique, the ASSR form of system (23) can be obtained as follows:

$$\begin{cases} Ex(\kappa+1) = Fu(\kappa)Ex(\kappa) \\ z(\kappa) = \mathcal{H}v(\kappa)x(\kappa) \end{cases}$$

where

$$F = 0.2\delta_8\left[3, 3, 3, 3, 5, 7, 5, 7, 3, 3, 4, 4, 5, 7, 6, 8\right] + 0.8\delta_8\left[1, 1, 4, 4, 4, 4, 4, 4, 5, 5, 8, 8, 4, 4, 8, 8\right]$$

and

$$\mathcal{H} = \delta_4\left[2, 4, 1, 2, 2, 3, 2, 1, 3, 4, 2, 3, 1, 2, 2, 4, 1, 2, 2, 4, 1, 3, 2, 1, 3, 4, 2, 2, 3, 1, 2, 4\right].$$

In the simulation, we take the control input sequence of the system as $u_1^{40} = \left[\delta_2^1, \delta_2^2, \delta_2^1, \cdots, \delta_2^2, \delta_2^1, \delta_2^2\right]^\top$ and $p_1 = p_2 = 0.1$. Then, the observation probability matrix H can be obtained as

$$H = \begin{bmatrix} 0.09 & 0 & 0.01 & 0 & 0.18 & 0.81 & 0 & 0.10 \\ 0.01 & 0.09 & 0.99 & 0.82 & 0.01 & 0.09 & 0.10 & 0 \\ 0.90 & 0 & 0 & 0.09 & 0.81 & 0.10 & 0 & 0 \\ 0 & 0.91 & 0 & 0.09 & 0 & 0 & 0 & 0.90 \end{bmatrix}.$$

Figure 6 displays the observed output sequence $z_{1:40}$ produced by the PBCN in (27) with the known observation probability matrix H and state transition probability F. The proposed algorithm is used to learn the matrices H and F under the designated output observation sequence. Let $\hat{F} = \left[\hat{F}_1 \ \hat{F}_2\right]$. The matrices can be estimated as follows:

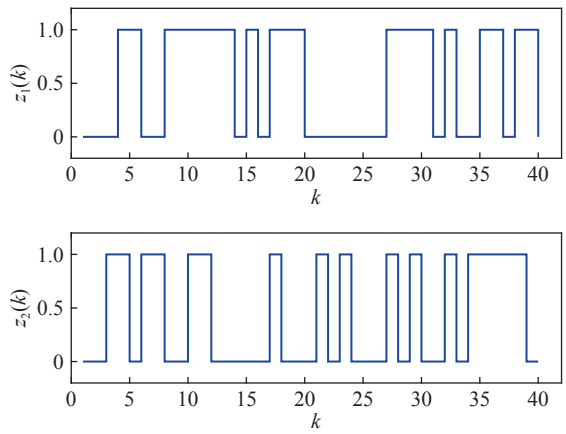


Figure 6. The measurement of the output observation value in system (28).

$$\hat{F}_1 = \begin{bmatrix} 0.74 & 0.75 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.17 & 0.25 & 0.19 & 0.21 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.75 & 0.77 & 0.81 & 0.77 & 0.75 & 0.80 \\ 0 & 0 & 0 & 0 & 0.16 & 0 & 0.21 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.18 & 0 & 0.21 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\hat{F}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.17 & 0.23 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0.26 & 0.73 & 0.86 & 0 & 0 \\ 0.78 & 0.75 & 0 & 0 & 0.22 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.17 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.22 & 0 & 0 \\ 0 & 0 & 0.79 & 0.94 & 0 & 0 & 0.85 & 0.89 \end{bmatrix}$$

and

$$\hat{H} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.21 & 0.79 & 0 & 0.11 \\ 0 & 0.09 & 0.97 & 0.82 & 0 & 0.07 & 0.12 & 0 \\ 0.89 & 0 & 0 & 0.12 & 0.83 & 0.12 & 0 & 0 \\ 0 & 0.94 & 0 & 0.10 & 0 & 0 & 0 & 0.91 \end{bmatrix}.$$

The parameter learning performance index ρ of matrices F and H are $\rho_F = 0.88$ and $\rho_H = 0.89$, respectively. The learning results of the above numerical example implies that our proposed algorithm performs well in identifying the model parameters under the real biological background.

5. Conclusion

In this paper, the parameter learning problem has been investigated for PBCNs with input-output data. By utilizing the STP technique, the ASSR of PBCNS has been obtained. Based on it, the parameter learning problem has naturally been converted into the one of identifying the corresponding parameters of a linear system, which definitely makes the learning process more mathematically accessible. Subsequently, the STP-based recursive forward and backward algorithms have been proposed. Then, the EM algorithm has been utilized to deal with the parameter learning problem. After that, an index has been introduced to describe the performance of the designed parameter learning algorithms. Finally, a regular logical model and a GRN model of the biological cell apoptosis network have been employed to show the effectiveness of the developed parameter learning algorithms.

Author Contributions: Hongwei Chen: writing-supervision, review and editing of writing, funding acquisition; **Qi Chen:** original draft writing; **Bo Shen:** writing-supervision, review and editing of writing, funding acquisition; **Yang Liu:** writing-supervision, review and editing of writing, funding acquisition. All authors have read and agree to the final version of the manuscript.

Funding: The research was partially supported by the National Natural Science Foundation of China under Grant 62003083, in part by the Shanghai Science and Technology Program under Grant 20JC1414500, in part by the Fundamental Research Funds for the Central Universities, and in part by the DHU Distinguished Young Professor Program under Grant 23D210401.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kauffman, S.A. Metabolic stability and epigenesis in randomly constructed genetic nets. J. Theor. Biol., **1969**, *22*: 437−467. doi: 10.1016/0022-5193(69)90015-0
2. Thomas, R. Boolean formalization of genetic control circuits. J. Theor. Biol., **1973**, *42*: 563−585. doi: 10.1016/0022-5193(73)90247-6
3. Akutsu, T.; Hayashida, M.; Ching, W.K.; *et al*. Control of Boolean networks: Hardness results and algorithms for tree structured networks. J. Theor. Biol., **2007**, *244*: 670−679. doi: 10.1016/j.jtbi.2006.09.023
4. Cheng, D.Z.; Qi, H.S.; Li, Z.Q. *Analysis and Control of Boolean Networks*: *A Semi-tensor Product Approach*; Springer: London, UK, 2011. doi: 10.1007/978-0-85729-097-7
5. Cheng, D.Z.; Qi, H.S. Controllability and observability of Boolean control networks. Automatica, **2009**, *45*: 1659−1667. doi: 10.1016/j.automatica.2009.03.006
6. Yu, Y.Y.; Meng, M.; Feng, J.E.; *et al*. Observability criteria for Boolean networks. IEEE Trans. Automat. Control, **2022**, *67*: 6248−6254. doi: 10.1109/TAC.2021.3131436
7. Feng, J.E.; Li, Y.L.; Fu, S.H.; *et al*. New method for disturbance decoupling of Boolean networks. IEEE Trans. Automat. Control, **2022**, *67*: 4794−4800. doi: 10.1109/TAC.2022.3161609
8. Zhao, R.; Feng, J.E.; Wang, B.; *et al*. Disturbance decoupling of Boolean networks via robust indistinguishability method. Appl. Math. Comput., **2023**, *457*: 128220. doi: 10.1016/j.amc.2023.128220
9. Li, R.; Chu, T.G. Complete synchronization of Boolean networks. IEEE Trans. Neural Netw. Learn. Syst., **2012**, *23*: 840−846. doi: 10.1109/TNNLS.2012.2190094
10. Chen, H.W.; Liang, J.L. Local synchronization of interconnected Boolean networks with stochastic disturbances. IEEE Trans. Neural Netw. Learn. Syst., **2020**, *31*: 452−463. doi: 10.1109/TNNLS.2019.2904978
11. Fornasini, E.; Valcher, M.E. Optimal control of Boolean control networks. IEEE Trans. Automat. Control, **2014**, *59*: 1258−1270. doi: 10.1109/TAC.2013.2294821
12. Laschov, D.; Margaliot, M. Minimum-time control of Boolean networks. SIAM J. Control Optim., **2013**, *51*: 2869−2892. doi: 10.1137/110844660
13. Li, H.T.; Xie, L.H.; Wang, Y.Z. Output regulation of Boolean control networks. IEEE Trans. Automat. Control, **2017**, *62*: 2993−2998. doi: 10.1109/TAC.2016.2606600

14. Zhang, X.; Wang, Y.H.; Cheng, D.Z. Output tracking of Boolean control networks. IEEE Trans. Automat. Control, **2020**, *65*: 2730−2735. doi: 10.1109/TAC.2019.2944903
15. Zhang, Q.L.; Feng, J.E.; Wang, B.; *et al*. Event-triggered mechanism of designing set stabilization state feedback controller for switched Boolean networks. Appl. Math. Comput., **2020**, *383*: 125372. doi: 10.1016/j.amc.2020.125372
16. Wang, Y.; Yang, Y.J.; Liu, Y.; *et al*. Fault detection and pinning control of Boolean networks. Appl. Math. Comput., **2022**, *429*: 127232. doi: 10.1016/j.amc.2022.127232
17. Li, X.; Liu, Y.; Lou, J.G.; *et al*. Robust minimal strong reconstructibility problem of Boolean control networks. Appl. Math. Comput., **2023**, *458*: 128209. doi: 10.1016/J.AMC.2023.128209
18. Li, H.T.; Xu, X.J.; Ding, X.Y. Finite-time stability analysis of stochastic switched Boolean networks with impulsive effect. Appl. Math. Comput., **2019**, *347*: 557−565. doi: 10.1016/j.amc.2018.11.018
19. Shmulevich, I.; Dougherty, E.R.; Kim, S.; *et al*. Probabilistic Boolean networks: A rule-based uncertainty model for gene regulatory networks. Bioinformatics, **2002**, *18*: 261−274. doi: 10.1093/bioinformatics/18.2.261
20. Chen, H.W.; Liang, J.L.; Lu, J.Q.; *et al*. Synchronization for the realization-dependent probabilistic Boolean networks. IEEE Trans. Neural Netw. Learn. Syst., **2018**, *29*: 819−831. doi: 10.1109/TNNLS.2017.2647989
21. Li, H.T.; Yang, X.R.; Wang, S.L. Perturbation analysis for finite-time stability and stabilization of probabilistic Boolean networks. IEEE Trans. Cybern., **2021**, *51*: 4623−4633. doi: 10.1109/TCYB.2020.3003055
22. Chen, H.W.; Wang, Z.D.; Shen, B.; *et al*. Model evaluation of the stochastic Boolean control networks. IEEE Trans. Automat. Control, **2022**, *67*: 4146−4153. doi: 10.1109/TAC.2021.3106896
23. Li, R.; Yang, M.; Chu, T.G. State feedback stabilization for probabilistic Boolean networks. Automatica, **2014**, *50*: 1272−1278. doi: 10.1016/j.automatica.2014.02.034
24. Wu, Y.H.; Guo, Y.Q.; Toyoda, M. Policy iteration approach to the infinite horizon average optimal control of probabilistic Boolean networks. IEEE Trans. Neural Netw. Learn. Syst., **2021**, *32*: 2910−2924. doi: 10.1109/TNNLS.2020.3008960
25. Li, F.F.; Xie, L.H. Set stabilization of probabilistic Boolean networks using pinning control. IEEE Trans. Neural Netw. Learn. Syst., **2019**, *30*: 2555−2561. doi: 10.1109/TNNLS.2018.2881279
26. Ding, X.Y.; Li, H.T.; Yang, Q.Q.; *et al*. Stochastic stability and stabilization of *n*-person random evolutionary Boolean games. Appl. Math. Comput., **2017**, *306*: 1−12. doi: 10.1016/j.amc.2017.02.020
27. Lähdesmäki, H.; Shmulevich, I.; Yli-Harja, O. On learning gene regulatory networks under the Boolean network model. Mach. Learn., **2003**, *52*: 147−167. doi: 10.1023/A:1023905711304
28. Nam, D.; Seo, S.; Kim, S. An efficient top-down search algorithm for learning Boolean networks of gene expression. Mach. Learn., **2006**, *65*: 229−245. doi: 10.1007/s10994-006-9014-z
29. Apostolopoulou, I.; Marculescu, D. Tractable learning and inference for large-scale probabilistic Boolean networks. IEEE Trans. Neural Netw. Learn. Syst., **2019**, *30*: 2720−2734. doi: 10.1109/TNNLS.2018.2886207
30. Akutsu, T.; Melkman, A.A. Identification of the structure of a probabilistic Boolean network from samples including frequencies of outcomes. IEEE Trans. Neural Netw. Learn. Syst., **2019**, *30*: 2383−2396. doi: 10.1109/TNNLS.2018.2884454
31. Cheng, D.Z.; Qi, H.S.; Li, Z.Q. Model construction of Boolean network via observed data. IEEE Trans. Neural Netw., **2011**, *22*: 525−536. doi: 10.1109/TNN.2011.2106512
32. Cheng, D.Z.; Zhao, Y. Identification of Boolean control networks. Automatica, **2011**, *47*: 702−710. doi: 10.1016/j.automatica.2011.01.083
33. Zhang, X.H.; Han, H.X.; Zhang, W.D. Identification of Boolean networks using premined network topology information. IEEE Trans. Neural Netw. Learn. Syst., **2017**, *28*: 464−469. doi: 10.1109/TNNLS.2016.2514841
34. Leifeld, T.; Zhang, Z.H.; Zhang, P. Identification of Boolean network models from time series data incorporating prior knowledge. Front. Physiol., **2018**, *9*: 695. doi: 10.3389/fphys.2018.00695
35. Cui, L.B.; Li, W.; Ching, W.K. On Construction of sparse probabilistic Boolean networks from a prescribed transition probability matrix. In *Proceedings of the Fourth International Conference on Computational Systems Biology*, *Suzhou*, *China*, *9–11 September 2010*; ORSC & APORC, 2010; pp. 227–234.
36. Dempster, A.P.; Laird, N.M.; Rubin, D.B. Maximum likelihood from incomplete data via the *EM* algorithm. J. Roy. Stat. Soc.: Ser. B (Methodol.), **1977**, *39*: 1−22. doi: 10.1111/j.2517-6161.1977.tb01600.x
37. Chaves, M. Methods for qualitative analysis of genetic networks. In *Proceedings of the European Control Conference*, *Budapest*, *Hungary*, *23–26 August 2009*; IEEE: New York, 2009; pp. 671–676. doi: 10.23919/ECC.2009.7074480

Citation: Chen, H.; Chen, Q.; Shen, B.; *et al*. Parameter Learning of Probabilistic Boolean Control Networks with Input-Output Data. *International Journal of Network Dynamics and Intelligence*. 2024, 3(1), 100005. doi: 10.53941/ijndi.2024.100005