Chapter 3

Learning Method for the Interval-Based Hybrid Dynamical System

In this chapter, we propose a two-step learning method for the interval-based hybrid dynamical system (interval system). The method consists of (1) an agglomerative clustering process of dynamics, which estimates approximate parameters of each dynamical system, and (2) a refinement process of overall system parameters, which include parameters of the constituting finite state automaton and dynamical systems, based on the approximated expectation-maximization (EM) algorithm.

3.1 Difficulties in Identification of Hybrid Dynamical Systems

Let us assume that only a large amount of multivariate sequences is given as training data. Then, in the most of hybrid systems, the system identification process that estimates all the system parameters becomes difficult because of its paradoxical nature. That is, the identification of the subsystems requires partitioned and labeled training data; meanwhile, the segmentation and labeling processes of training data require an identified set of subsystems. Moreover, the number of the subsystems is also unknown in general. Therefore, the parameter estimation problem requires us to simultaneously estimate temporal partitioning of training data (i.e., segmentation and labeling) and the set of subsystems (i.e., the number of the subsystems and their parameters). This problem also lies in the learning process of the interval system proposed in the previous chapter.

In the following paragraphs, we introduce relevant work as for the identification methods for various hybrid systems. We distinguish two cases when the number of subsystems is known or unknown.

3.1.1 In Case the Number of Subsystems is Known

The EM algorithm [DLR77] is one of the most common approaches to solve this kind of paradoxical problems when the number of subsystems is given. The algorithm estimates parameters based on the iterative calculation. In each step, the algorithm conducts model fitting to the training data using the model parameters that were updated in the previous step. Then, the parameters are updated based on the result of the current model fitting process.

Many of the hybrid system identification methods exploit the EM algorithm; for example, segmental models [ODK96], motion textures [LWS02], stochastic linear hybrid systems [BHJT04] (which follows the method in [LWS02]), and SLDSs [GH96, PRCM99, PRM00] applied this algorithm. A well-known identification technique for piecewise linear (PWL) models is introduced as k-mean like clustering [FMLM03]. This clustering technique can be regarded as an approximation (i.e., hard clustering version) of the EM algorithm, which is often referred to as soft clustering [DHS00].

Although the convergence of the EM algorithm is guaranteed, it strongly depends on the selection of initial parameters and often converges to a locally optimal solution, especially if the model has a large parameter space to search. As the alternative approach, an identification problem of PWL models can be recasted as a mixed integer programming (MIP), which find the globally optimal solution [RBL04, KHS$^+$04]. We can apply the method when the logical switching conditions between subsystems can be transformed into a set of inequalities; however, it is difficult to transform the dynamic switching conditions, which are modeled by an automaton. Another disadvantage of MIP lies in its computational complexity. The problem is well known to be NP-hard in the worst case [FMLM03, RBL04]. For these reasons, we exploit the EM algorithm rather than the MIP-based methods to identify the interval system.

3.1.2 In Case the Number of Subsystems is Unknown

Most of previous works in hybrid system identification assumed that the number of subsystems is given because the number often corresponds to that of manually defined operative conditions in the controlled object. In contrast, a set of dynamic primitives in human motion (e.g., primitive motion appeared in facial expressions) is undefined in most of cases. Whereas some heuristic sets of dynamic primitives are defined by human observation, they do not guarantee that the set is appropriate for man-machine interaction systems (e.g., automatic recognition of facial motion patterns). Hence, we should estimate the number of the subsystems (primitives) from a given training data set. The problem is often referred to as the cluster validation problem [DHS00] (see Subsection 3.3.5 for details).

In stochastic linear hybrid systems [LWS02, BHJT04], an online clustering based on a greedy algorithm is applied to determine the number of subsystems (i.e., linear dynamical systems) and initialize the EM algorithm [LWS02, BHJT04]. The drawback is that it depends on the order of data presentation, and is also sensitive to outliers in training data. Moreover, the algorithm requires deciding appropriate thresholds beforehand. As a result, the algorithm tends to return too many subsystems.

3.2 Two-Step Learning Method for the Hybrid Dynamical Systems

3.2.1 Parameters

The goal of the interval system identification is to estimate the parameters the following parameters from a large number of multivariate sequences:

- N : the number of dynamical systems (which corresponds to the number of discrete states)
- H, R : an observation matrix and an observation noise covariance matrix
- $F^{(i)}, g^{(i)}, Q^{(i)}\ (i = 1, ..., N)$: transition matrices, bias vectors, and process noise covariance matrices for each linear dynamical system
- $x_{\text{init}}^{(i)}, V_{\text{init}}^{(i)}\ (i = 1, ..., N)$: initial mean state vectors and initial state covariance matrices for each linear dynamical system

- $A_{ij}\ (i, j = 1, ..., N, i \neq j)$: a discrete-state transition matrix
- $\pi_i\ (i = 1, ..., N)$: an initial state distribution
- $h_m^{(i)}, h_v^{(i)}, h_c^{(i|j)}\ (i, j = 1, ..., N, i \neq j)$: parameters for two-dimensional duration distribution $h^{(i|j)}(l_k, l_{k-1})$ in adjacent intervals

We use the following notation to denote the set of parameters for convenience:

The parameter set of the overall system:

$$\Theta = \{\theta_i, A_{ij}, \pi_i, h_m^{(i)}, h_v^{(i)}, h_c^{(ij)} | i, j = 1, ..., N, i \neq j\}$$

The parameter set of constituting dynamical system $D_i (i = 1, ..., N)$:

$$\theta_i = \{F^{(i)}, g^{(i)}, Q^{(i)}, x_{\text{init}}^{(i)}, V_{\text{init}}^{(i)}\}$$

We assume that the observation matrix H and noise covariance matrix R are given. In case that the matrices are unknown, the subspace system identification techniques such as [OM95] can be applied by assuming a single transition matrix. Although the subspace identification technique has large sensitivity to noise, some research results show that this technique successfully identifies an observation matrix especially from visual feature sequences [DCWS03].

3.2.2 Two-Step Learning Method

As we described in Section 3.1, the EM algorithm-based parameter estimation method requires to solving two problems:

1. Initialization of the EM algorithm
2. Estimation of the number of subsystems

To overcome the problems, we propose a two-step learning method (see also Figure 3.1), which we describe in the following sections, that estimates parameters Θ of the interval system.

The key idea of the learning method is that we divide the estimation process into two steps: a clustering process of dynamical systems to estimate a set of required dynamical systems and a parameter refinement of the estimated dynamical systems. The parameters of automaton are also estimated in the second step.

Step 1 **Clustering of Dynamical Systems:** The first step is a clustering process that finds a set of dynamical systems: the number of the systems and their parameters. This step employs a set of typical sequences (i.e., a subset of the given training data set), and the sequences have been already mapped to the internal state space. Then, we apply an agglomerative hierarchical clustering technique, which we propose in this thesis, to the training data. This clustering technique estimate a set of dynamical systems (see Subsection 3.3 for details). After this process, we get the estimated number of dynamical systems N and approximate parameters of dynamical systems $\theta_i (i = 1, ..., N)$.

Step 2 **Refinement of the Parameters:** The second step is a refinement process of the system parameters based on the EM algorithm. The process is applied to all the given training data, whereas the clustering process is applied to a selected typical training set. Although the EM algorithm strongly depends on its initial parameters, the previous clustering step provides an initial parameter set that is relatively close to the optimum compared to a randomly selected parameter set. This algorithm also estimates the parameters of automaton such as a discrete-state transition probabilities and duration-length distributions associated with the state transition (see Subsection 3.4 for details). Finally, we get the estimated parameter set Θ of the overall interval system.

3.3 Hierarchical Clustering of Dynamical Systems

3.3.1 Overview of the Clustering Algorithm

Let us assume that a multivariate sequence $y_1^T \triangleq y_1, ..., y_T$ is given as a typical training data. We can consider a single training data without loss of generality because we do not estimate the order of transition of dynamical systems (i.e., discrete-state transition probabilities) in this clustering step. Then we simultaneously estimate a set of linear dynamical systems $\mathcal{D}$ (i.e., the number of linear dynamical system N and their parameters $\theta_1, ..., \theta_N$) with an interval set $\mathcal{I}$ (i.e., segmentation and labeling of the sequence), from the training sample y_1^T. The number of intervals K is also unknown.

We formulate the problem as the search of the linear dynamical system set $\mathcal{D}$

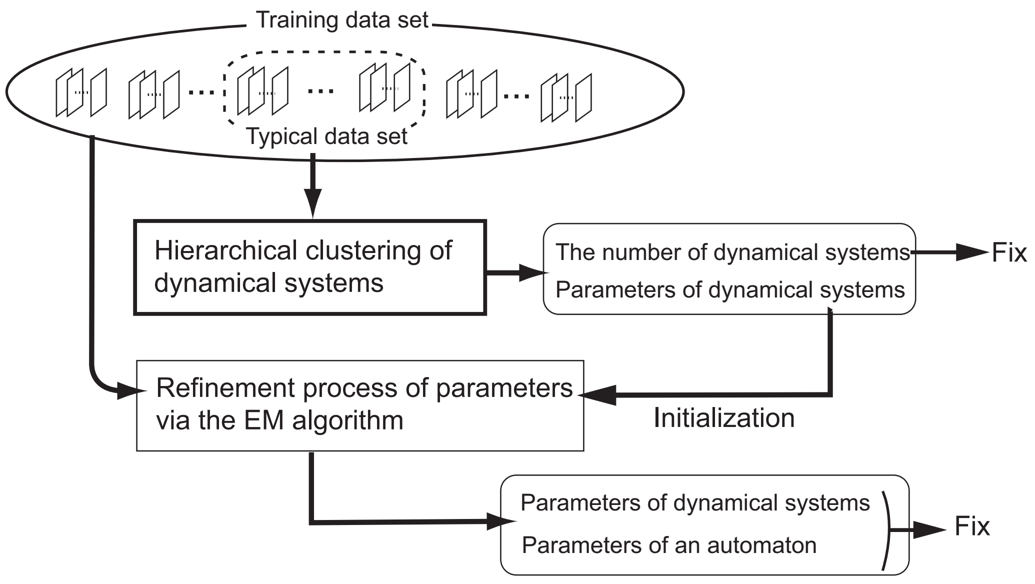

Training data set
Typical data set
Hierarchical clustering of dynamical systems
The number of dynamical systems
Parameters of dynamical systems
Fix
Refinement process of parameters via the EM algorithm
Initialization
Parameters of dynamical systems
Parameters of an automaton
Fix

Figure 3.1: A two-step learning for the interval system.

and the interval set $\mathcal{I}$ that maximizes the overall likelihood with respect to the training data: $\mathcal{L} \triangleq P(y_1^T|\mathcal{I}, \mathcal{D})$. Because the likelihood monotonically increases with an increase in the number of dynamical systems, we need to determine the right balance between the likelihood and the number N. A hierarchical clustering approach provides us an interface such as the history of model fitting errors in each merging steps to decide the number of dynamical systems (see Subsection 3.3.5 for details).

As we described in Subsection 3.2.1, we assume that the observation matrix H is given. That is, using the subspace system identification techniques such as [OM95] before the clustering step, we can simultaneously estimate observation matrix H and the corresponding internal state sequence $x_1^T \triangleq x_1, ..., x_T$ that can be mapped to the given observation sequence y_1^T by matrix H. In the following sections, we therefore focus on estimating parameters θ_i and the number of linear dynamical systems N from the given state sequence x_1^T.

The Hierarchical Clustering Algorithm of Dynamical Systems

The intuitive explanation of the hierarchical clustering algorithm is as follows:

1. Partition the training sequence into a group of very short sub-sequences and estimate a dynamical system that can model each sub-sequence respectively.

2. Compute the distance between each pair of estimated dynamical systems.

3. Merge the closest pair of dynamical systems; compute parameters of the merged dynamical system based on such sub-sequences that were modeled by the pair of dynamical systems to be merged.

4. Iterate the above agglomerating process until the closest distance between a pair of dynamical systems becomes greater than a pre-specified value.

After this process, we get the number of required dynamical systems N and approximate parameters of the dynamical systems. Note that the pre-specified value (i.e., threshold) required in step 4 determines the number of the estimated dynamical systems. It is however difficult to specify the value beforehand. Therefore, we first proceed with the algorithm until all the dynamical systems are agglomerated to a single dynamical system, and then estimate the number of the dynamical systems by evaluating the model fitting scores at each of the iteration steps.

Algorithm 5 shows the details of the clustering process. This algorithm requires initial partitioning of the training sequence, which we refer to as an initial interval set $\mathcal{I}_{\text{init}}$. We will discuss about the initialization in Subsection 3.3.2.

Algorithm 5 Agglomerative Hierarchical Clustering of Dynamical Systems.

```
for $I_i$ in $\mathcal{I}_{\text{init}}$ do
    $D_i \leftarrow \mathsf{Identify}\,(I_i)$
    insert $D_i$ to $\mathcal{D}$
end for
for all pair$(D_i,\ D_j)$ where $D_i,\ D_j \in \mathcal{D}$ do
    $\mathsf{Dist}\,(i,\ j) \leftarrow \mathsf{CalcDistance}\,(D_i,\ D_j)$
end for
while $N \geq 2$ do
    $(i^*,\ j^*) \leftarrow \arg\min_{(i,\ j)} \mathsf{Dist}\,(i,\ j)$
    $\mathcal{I}_{i^*} \leftarrow \mathsf{MergeIntervals}\,(\mathcal{I}_{i^*},\ \mathcal{I}_{j^*})$
    $D_{i^*} \leftarrow \mathsf{Identify}\,(\mathcal{I}_{i^*})$
    erase $D_j^*$ from $\mathcal{D}$
    $N \leftarrow N - 1$
    for all pair$(D_i^*,\ D_j)$ where $D_j \in \mathcal{D}$ do
        $\mathsf{Dist}(i^*,\ j) \leftarrow \mathsf{CalcDistance}\,(D_{i^*},\ D_j)$
    end for
end while
```

In the first step of the algorithm, dynamical systems are identified (which is denoted by Identify in Algorithm 5) from each interval in the initial interval set $\mathcal{I}_{\text{init}}$ individually based on the identification method proposed in Subsection 3.3.3. $\mathcal{I}_i$ is the interval set that comprises intervals labeled by D_i.

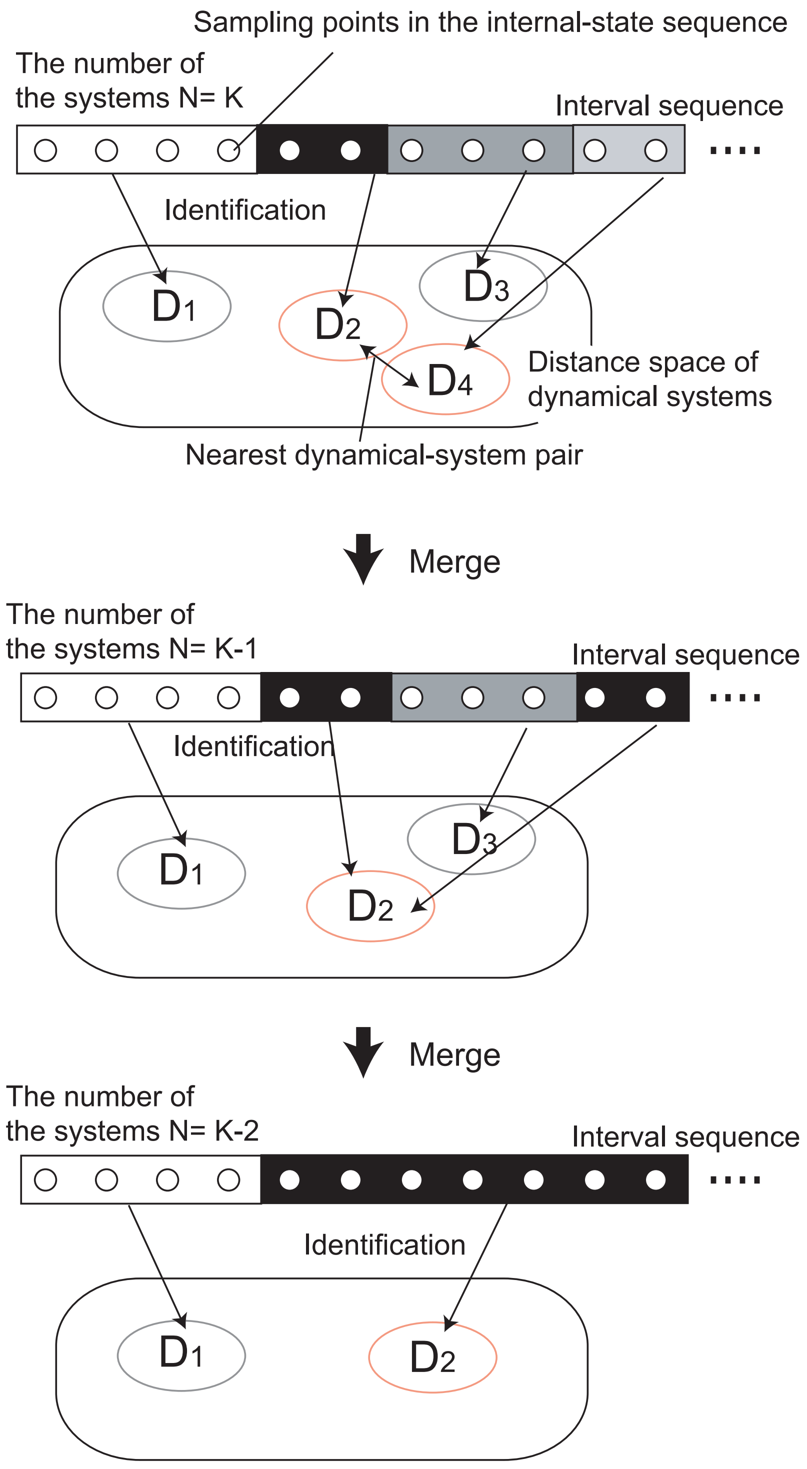

Sampling points in the internal-state sequence
The number of the systems N= K
Interval sequence
Identification
D1
D2
D3
D4
Distance space of dynamical systems
Nearest dynamical-system pair
Merge
The number of the systems N= K-1
Interval sequence
Identification
D1
D2
D3
Merge
The number of the systems N= K-2
Interval sequence
Identification
D1
D2

Figure 3.2: Hierarchical clustering of dynamical systems.

Then, we calculate the distances (denoted by CalcDistance) for all the dynamical system pairs based on the distance definition described in Subsection 3.3.4. After the distance calculation, the nearest dynamical systems are merged (which is denoted by MergeIntervals), and the parameter set of the merged system is estimated again. Here, two interval sets that belong to the nearest dynamical system pair are also merged in parallel. Figure 3.2 shows an example of the step in which the dynamical system D_2 and D_4 are merged.

Repeating the merging process in the previous paragraph, we get a single dynamical system in which all the dynamical systems are agglomerated.

3.3.2 Initialization of the Clustering Algorithm

The clustering algorithm proposed in the previous subsection iteratively merges intervals in the initial interval set $\mathcal{I}_{\text{init}}$. The final result of the clustering process is therefore affected by the estimation of set $\mathcal{I}_{\text{init}}$. In this section, we give some examples to determine $\mathcal{I}_{\text{init}}$.

Fixed-Length Segmentation

The simplest partitioning method of $\mathcal{I}_{\text{init}}$ is to divide the given training sequence into fixed length small intervals. However, this naive method is undesirable for real world problem such as signals from cameras and microphones because of its high computational cost. Let $|\mathcal{I}_{\text{init}}|$ be the number of initial intervals. Then, the actual computational cost of Algorithm 5 is proportional to $O(|\mathcal{I}_{\text{init}}|^2)$, which is due to the first distance calculation in the second block of Algorithm 5.

Zero-Velocity-Based Segmentation:

Now, we consider how to get the initial interval set that is appropriate for finding motion dynamics in human behavior. To reduce the computational cost, we first require $|\mathcal{I}_{\text{init}}|$ to be relatively smaller than the length of the training sequence. Second, we require that the initial intervals divide stationary pose from motion in the training sequence. This is useful for describing human motion based on simple primitives. Third, we require the intervals $I_i \in \mathcal{I}_{\text{init}}$ to be simple in the sense that the patterns in the initial intervals appear frequently in the training sequence.

To satisfy the conditions above, we exploit zero velocity of the signal. Let $\dot{x}_t$ be the velocity vector of given signal x_t $(t = 1, ..., T)$. If the Euclidean norm $||\dot{x}_t||$

becomes nearly zero (i.e., under the given threshold), we divide the signal at that temporal point. Then, if $||\dot{x}_t||^2$ becomes greater than the given threshold again, we also divide the signal at that point. Repeating this process, we get the initial intervals in which motion states and stationary states appear by turns.

Because x_t is a discrete-time signal, we use the following equation to calculate the norm of velocity $\dot{x}_t$:

$$||\dot{x}_t|| = \sqrt{\sum_{r=1}^{R} ||x_{t+r} - x_{t+r-1}||^2} \qquad (t = 1, ..., T - R), \tag{3.1}$$

where R is a smoothing parameter of velocity calculation.

3.3.3 Constrained System Identification Based on Eigenvalues

To identify the system parameters from only a small amount of training data, we have to use constraints for estimating desirable dynamics. In this chapter, we concentrate on extracting human motion primitives observed in such as facial motion, gaits, and gestures. Most of these motions are generated by the combination of intermittent muscle controls; therefore, constraints based on stability of dynamics might be suitable to find motion that converges to a certain state from an initial pose.

The key idea of estimating stable dynamics is to give constrains on the eigenvalues. As we see in Section 2.2.2, the dynamical system changes the state in a stable manner if all the eigenvalues are smaller than one. In the following paragraphs, we propose a novel system identification method that constrains all the eigenvalues of the system to be smaller than one. The method corresponds to Identify in Algorithm 5.

System Identification without Constraints

If the temporal range $[b, e]$ is represented by linear dynamical system D_i. Then, we can estimate the transition matrix $F^{(i)}$ and bias vector $g^{(i)}$ from the internal state sequence $x_b^{(i)}, .., x_e^{(i)}$ in the range $[b, e]$. This parameter estimation problem becomes a minimization problem of prediction errors.

If we have already estimated $F^{(i)}$ and $g^{(i)}$, we can predict a state vector at time

t from $x_{t-1}^{(i)}$ using Equation (2.1). We then get the prediction error vector:

$$\epsilon_t = x_t^{(i)} - (F^{(i)} x_{t-1}^{(i)} + g^{(i)}). \tag{3.2}$$

Thus, the summation for the squared norms of all the error vectors in the range $[b, e]$ becomes

$$\sum_{t=b+1}^{e} ||\epsilon_t||^2 = \sum_{t=b+1}^{e} ||x_t^{(i)} - (F^{(i)} x_{t-1}^{(i)} + g^{(i)})||^2. \tag{3.3}$$

Finally, we can estimate optimal $F^{(i)}$ and $g^{(i)}$ by solving the following least squares problem:

$$F^{(*i)}, g^{(*i)} = \arg \min_{F^{(i)}, g^{(i)}} \sum_{t=b+1}^{e} ||\epsilon_t||^2. \tag{3.4}$$

In the next paragraph, we show only the result of this estimation. As for details, refer to Appendix B.

Using notations

$$\begin{aligned} \hat{X}_0^{(i)} &= [x_b^{(i)} - m_0^{(i)}, ..., x_{e-1}^{(i)} - m_0^{(i)}], \\ \hat{X}_1^{(i)} &= [x_{b+1}^{(i)} - m_1^{(i)}, ..., x_e^{(i)} - m_1^{(i)}], \end{aligned}$$

we can calculate the estimated transition matrix and bias vector as follows:

$$F^{(i)*} = \hat{X}_1^{(i)} \hat{X}_0^{(i)\dagger}, \tag{3.5}$$

$$g^{(i)*} = m_1 - F^{(i)*} m_0, \tag{3.6}$$

where, m_0 and m_1 are mean vectors of columns in $X_0^{(i)}$ and $X_1^{(i)}$, respectively:

$$m_0^{(i)} \triangleq \frac{1}{l-1} \sum_{t=b}^{e-1} x_t^{(i)}, \quad m_1^{(i)} \triangleq \frac{1}{l-1} \sum_{t=b+1}^{e} x_t^{(i)}. \tag{3.7}$$

$X_0^{(i)\dagger}$ is a Moore-Penrose generalized inverse (pseudo inverse) of $X_0^{(i)}$. Inverse matrix $X^\dagger$ can be defined as:

$$X^\dagger = \lim_{\delta^2 \to 0} X^\top (XX^\top + \delta^2 I)^{-1} = \lim_{\delta^2 \to 0} (X^\top X + \delta^2 I)^{-1} X^\top. \tag{3.8}$$

See also [Alb72, Koh89] for details.

System Identification with an Eigenvalue Constraint

We now show the method to constrain the eigenvalues of transition matrix $F^{(i)}$. Using Equation (3.5) and Equation (3.8), we can get the following equation:

$$\begin{aligned} F^{(i)*} &= \lim_{\delta^2 \to 0} \hat{X}_1^{(i)} \hat{X}_0^{(i)\top} (\hat{X}_0^{(i)} \hat{X}_0^{(i)\top} + \delta^2 I)^{-1} \\ &= \lim_{\delta^2 \to 0} X_1^{(i)} \hat{X}_0^{(i)\top} (\hat{X}_0^{(i)} \hat{X}_0^{(i)\top} + \delta^2 I)^{-1}, \end{aligned} \tag{3.9}$$

where I is the unit matrix and δ is a nonzero real value. Here, we used $\hat{X}_1^{(i)} \hat{X}_0^{(i)\top} = X_1^{(i)} \hat{X}_0^{(i)\top}$ (see Appendix B).

For the constraint on the eigenvalues, we stop the limit in the Equation (3.9) before $\hat{X}_0^{(i)\top} (\hat{X}_0^{(i)} \hat{X}_0^{(i)\top} + \delta^2 I)^{-1}$ converges to its generalized inverse matrix. In other word, we use

$$F_{\delta^2}^{(i)} = X_1^{(i)} \hat{X}_0^{(i)\top} (\hat{X}_0^{(i)} \hat{X}_0^{(i)\top} + \delta^2 I)^{-1} \tag{3.10}$$

for the constrained transition matrix.

As we will see in the following paragraphs, we can determine the upper bound of eigenvalues in the given matrix from its elements based on Gershgorin's theorem (Appendix C). We can therefore make a constraint that all the eigenvalues to be smaller than one if we chose an appropriate nonzero value for δ, which controls the scale of elements in the matrix $F^{(i)*}$.

Moreover, this constraint method have the advantage of calculating the generalized inverse matrix in Equation (3.5) successfully even if the matrix $\hat{X}_0^{(i)} \hat{X}_0^{(i)\top}$ have zero eigenvalues (the matrix becomes singular), as we will see in the rest of this subsection.

Numerical Calculation for Searching δ

We now describe how to calculate δ that constrains the upper bound of eigenvalues. Here we omit the index i, which denotes the label of dynamical systems, to simplify the notation.

First, we decompose the matrix $\hat{X}_0$ based on the singular value decomposition (SVD) to facilitate the calculation of δ:

$$\hat{X}_0 = USV^\top, \tag{3.11}$$

where U is $n \times n_r$ and V is $(l-1) \times n_r$ matrix; here $n_r = \min(n, l-1)$. These matrices are column orthogonal; that is $U^\top U = I_{n_r}$ and $V^\top V = I_{n_r}$. The middle matrix $S = \mathrm{diag}[\sigma_1, ..., \sigma_{n_r}]$ is a $n_r \times n_r$ diagonal matrix; we assume that the diagonal elements (singular values) are sorted by descending order (i.e., $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{n_r}$). In addition, if $n \leq l-1$, then $n_r = n$ and U becomes an orthogonal square matrix that satisfies $U^\top U = UU^\top = I_n$; on the other hand, if $n \geq l-1$, then $n_r = l-1$ and V becomes an orthogonal square matrix that satisfies $V^\top V = VV^\top = I_{l-1}$. If the rank of matrix $\hat{X}_0$ becomes smaller than n_r, let the rank be $n_r{}'$, the smallest $n_r - n_r{}'$ singular values become zero.

Using this decomposition, we can rewrite Equation (3.10) as

$$\begin{aligned} F_{\delta^2} &= X_1 V S U^\top \left((USV^\top)(VSU^\top) + \delta^2 I \right)^{-1} \\ &= (X_1 V) S U^\top U (S^2 + \delta^2 I)^{-1} U^\top \\ &= Z S (S^2 + \delta^2 I)^{-1} U^\top \\ &= \sum_{k=1}^{n_r} \frac{\sigma_k}{\sigma_k^2 + \delta^2} z_k u_k^\top , \end{aligned} \tag{3.12}$$

where we assumed all the singular values are nonzero, and replaced $X_1 V$ with $Z = [z_1, ..., z_{n_r}]$ ($z_i \in \mathbf{R}^n$).

Suppose f_{rc} is an element in row r and column c of the transition matrix F. The upper bound of the eigenvalues $\mathcal{B}$ can be determined by the corollary of the Gershgorin's theorem (see Appendix C):

$$\mathcal{B} = \max_r \sum_{c=1}^{n} |f_{rc}|. \tag{3.13}$$

Here, we search the δ that satisfies $\mathcal{B} = 1$. Substituting Equation (3.12) into Equation (3.13) and set $\mathcal{B} = 1$, we obtain

$$\max_r \sum_{c=1}^{n} \left| \sum_{k=1}^{n_r} \frac{\sigma_k}{\sigma_k^2 + \delta^2} w_{rc}^{(k)} \right| = 1, \tag{3.14}$$

where $z_k u_k^\top = W^{(k)} = [w_{rc}^{(k)}]$. Solving this equation via an iterative numerical calculation, we can find the value of δ that constrains the eigenvalues to be smaller than one.

Since we need to take absolute values in the Equation (3.14), it is difficult to solve this equation using straight forward numerical analysis. We therefore use the following relation to divide the optimization into two stages:

$$\sum_{c=1}^{n}\left|\sum_{k=1}^{n_r}\frac{\sigma_k}{\sigma_k^2+\delta^2}w_{rc}^{(k)}\right| \leq \sum_{c=1}^{n}\sum_{k=1}^{n_r}\frac{\sigma_k}{\sigma_k^2+\delta^2}|w_{rc}^{(k)}| = \sum_{k=1}^{n_r}\frac{\sigma_k p_{rk}}{\sigma_k^2+\delta^2}, \tag{3.15}$$

where $p_{rk} = \sum_{c=1}^{n}|w_{rc}^{(k)}|$.

Stage1. In the first stage, we solve the following Equation (3.16) with respect to δ^2 based on Newton's method. For the initial value of δ, the smallest singular value seems to work well in most cases.

$$\mathcal{B}'_r(\delta^2) \triangleq \sum_{k=1}^{n_r}\frac{\sigma_k p_{rk}}{\sigma_k^2+\delta^2} = 1 \quad (r = 1, ..., n). \tag{3.16}$$

Stage2. The result of δ^2 from the first stage becomes larger than or equal to the solution of the following equation:

$$\mathcal{B}_r(\delta^2) \triangleq \sum_{c=1}^{n}\left|\sum_{k=1}^{n_r}\frac{\sigma_k}{\sigma_k^2+\delta^2}w_{rc}^{(k)}\right| = 1 \quad (r = 1, ..., n). \tag{3.17}$$

In the second stage, we therefore search the solution of the above equation based on a naive search; that is, we make δ^2 smaller than the previous step at each iteration step until one of $\mathcal{B}_r(\delta^2)(r = 1, ..., n)$ exceeds 1. For the initialization, we use the maximum of δ^2, $\arg\max_{\delta^2} \mathcal{B}'_r(\delta^2)$, in the first stage.

Despite that we need to calculate $\mathcal{B}_r(\delta^2)$ for all the row of F_{δ^2}, in our experiments, we select only one row in the beginning of search algorithm (before the first stage) using a criterion such as $\arg\max_r p_{r1}$. This criterion works well empirically, and reduces the computational cost drastically. More appropriate criterion might be to use $\arg\max_r \mathcal{B}'_r$ based on the result of the first stage.

Advantage of Constrained Identification in Singular Case

From Equation (3.12), the decomposition of unconstrained transition matrix in Equation (3.5) becomes

$$F^* = \lim_{\delta^2 \to 0} F_{\delta^2} = ZS^{-1}U^\top = \sum_{k=1}^{n_r}\frac{1}{\sigma_k}z_k u_k^\top. \tag{3.18}$$

We see that if one of the singular values becomes close to zero, every element of the estimated transition matrix get a large number (∞ if one of the singular value is zero). On the other hand, if we use the proposed constrained identification method, the function $1/\sigma_k$ is replaced by $\sigma_k/(\sigma_k^2 + \delta^2)$. Figure 3.3 shows the comparison of functions $1/\sigma_k$ and $\sigma_k/(\sigma_k^2 + \delta^2)$. We see that $\sigma_k/(\sigma_k^2 + \delta^2)$ is constrained to be in the desired range, which can be controlled by the value of δ^2; moreover, the value of $\sigma_k/(\sigma_k^2 + \delta^2)$ becomes zero when $\sigma_k = 0$. Thus, we can successfully estimate the transition matrix even if some singular values become zero.

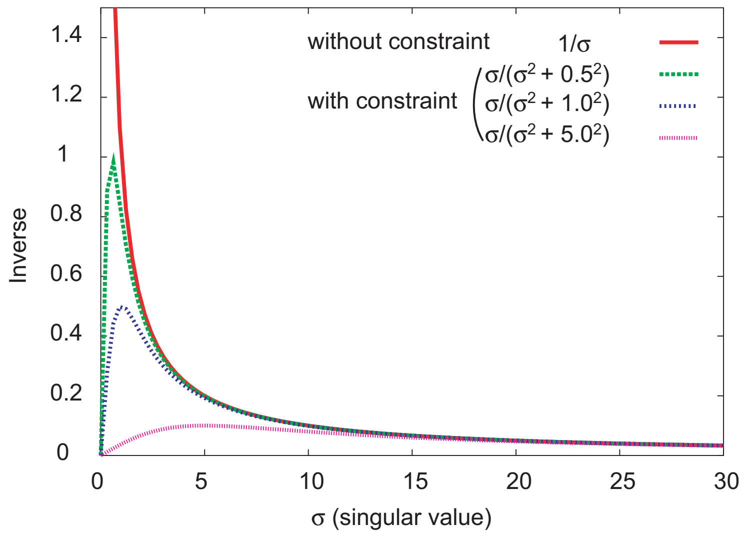


Figure 3.3: Functions $\frac{1}{\sigma}$ and $\frac{\sigma}{\sigma^2+\delta^2}$ (when $\delta^2 = 0.5, 1.0, 5.0$).

3.3.4 Distance Definition between Dynamical Systems

In order to determine a pseudo distance between two linear dynamical systems, we first compare the following three approaches: (a) direct comparison of model parameters, (b) decreased likelihood of the overall system after the merging of two models [Bra95], and (c) distance measure of distributions (e.g., KL divergence [JR85]).

Approach (a) is not desirable particularly with regards to our bottom-up approach because a linear dynamical system has a large parameter set that often causes over-fitting. Hence, the distance in the parameter space does not always represent the dissimilarity among dynamical systems.

Approach (b), which is often exploited with stepwise-optimal clustering [DHS00], performs well in the ideal condition, but it requires the computational cost of likelihood scores for all the combination of the linear dynamical system pairs.

From our preliminary experiments, we observed that (c) provides stable dissimilarity measure for the hierarchical clustering algorithm with a realistic computational cost. Therefore, we define the distance between linear dynamical system D_i and D_j as an average of two asymmetric divergences:

$$Dist(D_i, D_j) = \frac{KL(D_i||D_j) + KL(D_j||D_i)}{2},$$

where each of the divergences is calculated as an approximation of KL divergence normalized by the interval length:

$$\begin{aligned} KL(D_i||D_j) &\sim \sum_{I_k \in \mathcal{I}_i} P(I_k|D_i) \log \left(\frac{P(y_{b_k}^{e_k}|D_i)}{P(y_{b_k}^{e_k}|D_j)} \right)^{\frac{1}{|I_k|}} \\ &\sim \frac{1}{|\mathcal{I}_i|} \sum_{I_k \in \mathcal{I}_i} \left\{ \log P(y_{b_k}^{e_k}|D_i) - \log P(y_{b_k}^{e_k}|D_j) \right\}, \end{aligned} \qquad (3.19)$$

where $y_{b_k}, ..., y_{e_k}$ is an observation sequence partitioned by interval I_k. $|\mathcal{I}_i|$ is the summation of interval length in the interval set $\mathcal{I}_i$ that is labeled by a linear dynamical system D_i. Similarly, $|I_k| = e_k - b_k + 1$ is the length of interval I_k. We here approximated conditional probability of I_k to be $P(I_k|D_i) \sim |I_k|/|\mathcal{I}_i|$. We can use Equation (2.15) to calculate the likelihoods in Equation (3.19).

3.3.5 The Cluster Validation Problem

In real applications, it is an important problem to determine the appropriate number of dynamical systems (the number of clusters). The problem is often referred to as the cluster validation problem, which remains essentially unsolved. There are, however, several well-known criteria, which can be categorized into two types, to decide the number of clusters.

One of the types is defined based on the change of model fitting scores, such as log-likelihood scores and prediction errors (approximation of the log-likelihood scores), during the merging steps. If the score decreased rapidly, then the merging process is stopped [PH95]. In other words, it finds *knee* of the log-likelihood curve.

The other type is defined based on information theories, such as minimum description length and Akaike's information criterion. The information-theoretical criteria define the evaluation functions that consist of two terms: log-likelihood scores and the number of free parameters.

Although information-theoretical criteria work well in simple models, they tend to fail in evaluating right balance between the two terms, especially if the model becomes complex and has a large number of free parameters [LMZ98]. Because this problem also arises in our case, we use model fitting scores directly. First, we extract candidates for the numbers of the dynamical systems by finding peaks in the difference of model fitting errors between current and the previous steps. If the value exceeds a predefined threshold, then the number of dynamical systems in that step is added to the candidates. We consider that user should finally decide the appropriate number of the dynamical systems from the extracted candidates.

3.4 Parameter Refinement via the Expectation-Maximization Algorithm

This subsection describes a refinement process of the overall system parameter set Θ including the parameters of the automaton by exploiting the result of the clustering process. In the refinement process, the number of dynamical systems N is fixed (see Figure 3.1).

3.4.1 Overview of the Expectation-Maximization Algorithm

In order to refine the parameters, we apply an iterative estimation based on the EM algorithm. This algorithm starts from a given initial parameter set Θ_{init}, and repeats two steps: E (expectation) step and M (maximization) step. The E step fits the model to the given training samples based on the current model parameters, and calculates the expectation of log-likelihood with respect to all the given samples and to all the possible fitting instances. The M step updates the parameters to maximize the expectation of log-likelihood using the result of the statistics in the E step. After the iteration steps, the algorithm converges to the optimal parameters, and finds the result of the model fitting simultaneously.

Although the EM algorithm has been proved to converge, the obtained solutions are often trapped by local optima. This problem becomes significant especially if the size of the parameters increases. For a large parameter models, the algorithm therefore requires a parameter set that is relatively close to the optimum for the initialization.

The two-step learning method proposed in this chapter overcomes the initialization problem by exploiting the result of the clustering process applied for a typical training data set. From the viewpoint of the likelihood maximization, the clustering process estimates approximate parameters for the dynamical systems. To initialize remaining parameters such as the interval transition matrix $[A_{ij}]$ and interval duration distributions $h^{(ij)}(l_k, l_{k-1})$, we exploit Viterbi algorithm, which is introduced in Subsection 2.4.2, with some modification; that is, we set all the transition probabilities to be equal: $A_{ij} = 1/(N-1)$, and we also assume the interval duration $\hat{h}^{(ij)}(l_k, l_{k-1})$ to be uniform distributions in the range of $[l_{\min}, l_{\max}]$, where $i, j = 1, ..., N (i \neq j)$. As a result, Equation (2.29) of the Viterbi algorithm becomes

$$\delta_t(j,\tau) = P(y_{t-\tau+1}^{t} | s_t = q_j, l_t = \tau, f_t = 1) \max_{i(i \neq j), \tau_p} \{\delta_{t-\tau}(i, \tau_p)\}.$$

After the initialization above, parameter refinement iterations are applied to all the training data $\mathcal{Y} \triangleq \{y_1^{T_1}, ..., y_1^{T_M}\}$, where M is the number of the training sequences and $T_1, ..., T_M$ are the lengths of each sequence. Here, we approximate the EM algorithm because the original EM algorithm requires forward/backward inferences that often cause numerical underflow. We do not calculate the statistics for all the possible hidden variables (possible interval sequences), but use only a single interval sequence that gives the highest log-likelihood in each step. Thus, the total refinement process is described as follows:

1. Initialize $\Theta = \Theta_{\text{init}}$.
2. Repeat the following steps while the total likelihood changes greater than a given threshold ϵ:

   E step: search an optimal interval sequence, which is labeled by the discrete states, for all the training sequences based on the current parameter set Θ:

   $$\mathcal{I}^* = \arg\max_{\mathcal{I}} \log P(\mathcal{Y}, \mathcal{I} | \Theta),$$

   where $\mathcal{I}^*$ is the set of the searched interval sequences for all the training sequences.

M step: update the parameters based on the searched interval sequences in the E step:

$$\Theta = \arg\max_{\Theta'} \log P(\mathcal{Y}, \mathcal{I}^* | \Theta').$$

This algorithm is easily extended to use multiple interval sequences that give relatively high likelihood score rather than to use only an optimal interval sequence.

3.4.2 Parameter Estimation of the Automaton

The M step of the EM algorithm requires the estimation method of all the parameters in interval system Θ (see Subsection 3.2.1 for the details of the notation). Here, we show how to estimate the transition probabilities of discrete state, and the parameters of duration-length distributions. The parameter estimation of linear dynamical systems is similar to the method that we described in Subsection 3.3.3.

Discrete-State Transition Probabilities

Let $\mathcal{I}_1^*, ..., \mathcal{I}_M^*$ be the searched interval sequences for each training data $y_1^{T_1}, ..., y_1^{T_M}$ in the E step. Let s_k be the discrete state of interval $I_k^* \in \mathcal{I}_m^*$ $(m = 1, ..., M)$. The discrete-state transition probabilities A_{ij} $(i, j = 1, ..., N, i \neq j)$ are defined by Equation (2.19), we therefore estimate the probabilities by counting the frequency that the discrete state appears.

We first define some interval-label sets each of which specifies the subset in the interval set $\mathcal{I}_m^*$. Let $\mathcal{K}_m^{(i)}$ be the label set of intervals in which the discrete state is q_i; that is,

$$\mathcal{K}_m^{(i)} = \{k \,|\, s_k = q_i, I_k^* \in \mathcal{I}_m^*, k \geq 1\}. \tag{3.20}$$

Similarly, let $\mathcal{K}^{(i,j)}$ be the label set of intervals in which the discrete state is q_j and the discrete state of the previous interval is q_j; that is,

$$\mathcal{K}_m^{(i,j)} = \{k \,|\, s_k = q_j, s_{k-1} = q_i, I_k^* \in \mathcal{I}_m^*, k \geq 2\}. \tag{3.21}$$

Then, we can estimate the transition probability A_{ij} by the following equation:

$$P(s_k = q_j | s_{k-1} = q_i) = \frac{C(s_k = q_j, s_{k-1} = q_i)}{C(s_{k-1} = q_i)}, \tag{3.22}$$

where $C(s_k = q_j, s_{k-1} = q_i) = \sum_{m=1}^{M} |\mathcal{K}_m^{(i,j)}|$ is the frequency that states q_i and q_j

appears in the adjacent intervals in this order. $C(s_{k-1} = q_i) = \sum_{m=1}^{M} |\mathcal{K}_m^{(i)}|$ is the frequency that state q_i appeared in the interval sequence $\mathcal{I}^*$.

Initial Discrete-State Probabilities

The initial discrete-state probability π_i can be calculated by the similar method:

$$P(s_1 = q_i) = \frac{C(s_1 = q_i)}{M}, \tag{3.23}$$

where $C(s_1 = q_i)$ is the frequency that state q_i appears in the first interval of the searched interval sequences $\mathcal{I}_1^*, ..., \mathcal{I}_M^*$.

Parameters of Duration-Length Distributions

Let l_k be the duration length of interval $I_k^* \in \mathcal{I}_m^*$ $(m = 1, ..., M)$. As we described in Subsection 2.3.1, we assume two-dimensional Gaussian functions for duration-length distributions (see also Equation (2.17)):

$$P(l_k, l_{k-1} | s_k = q_j, s_{k-1} = q_i) = \mathcal{N}\left(h_m^{(i)}, \begin{bmatrix} h_v^{(i)} & h_c^{(ij)} \\ h_c^{(ij)} & h_v^{(j)} \end{bmatrix} \right) \tag{3.24}$$

We can estimate the parameters of the Gaussian distribution in Equation (3.24) by the following equations:

$$\begin{aligned}
h_m^{(i)} &= \frac{1}{\sum_{m=1}^{M} |\mathcal{K}_m^{(i)}|} \sum_{m=1}^{M} \sum_{k \in \mathcal{K}_m^{(i)}} l_k \qquad (i = 1, ..., N) \\
h_c^{(i)} &= \frac{1}{\sum_{m=1}^{M} |\mathcal{K}_m^{(i)}|} \sum_{m=1}^{M} \sum_{k \in \mathcal{K}_m^{(i)}} (l_k - h_m^{(i)})^2 \qquad (i = 1, ..., N) \\
h_v^{(ij)} &= \frac{1}{\sum_{m=1}^{M} |\mathcal{K}_m^{(ij)}|} \sum_{m=1}^{M} \sum_{k \in \mathcal{K}_m^{(ij)}} (l_k - h_m^{(j)})(l_{k-1} - h_m^{(i)}) \qquad (i, j = 1, ..., N,\ i \neq j)
\end{aligned}$$

3.5 Experiments

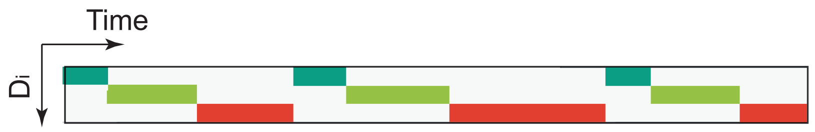
To evaluate the proposed parameter estimation methods, we first used simulated sequences for training data because it provides the ground truth of the estimated parameters. The first experiments shown in the Subsection 3.5.1 is for the verification of the clustering algorithm including the determination criterion of the number of clusters (linear dynamical systems), and the second experiments in Subsection 3.5.2 is for the evaluation of the overall two-step learning method. In Subsection 3.5.3, we see how the proposed method is applicable to handle real data.

3.5.1 Evaluation of the Clustering Algorithm Using Simulated Data

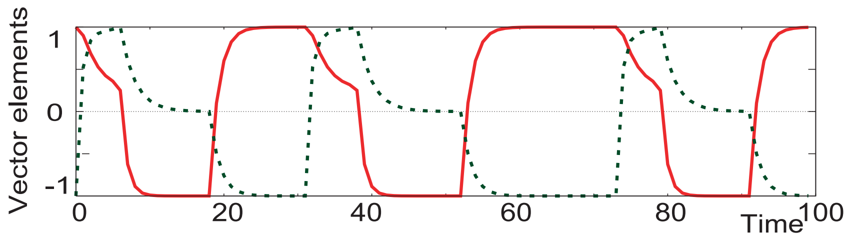
A multivariate sequence with length $T = 100$ was generated from the system that had the same parameters as the system in Section 2.5. In this sequence, three dynamic primitives were appeared in the temporal order of $D_1 \to D_2 \to D_3$, which repeats three cycles. The duration lengths of the discrete state are generated randomly based on the parameters of the duration-length distribution. Figure 3.4 (a) and (b) shows an example of the generated interval sequence and the two-dimensional observation sequence, respectively.

The hierarchical clustering algorithm proposed in Section 3.3 was applied to the sequence. Figure 3.4 (c) shows the overall model fitting errors between the original sequence Y and generated sequences $Y^{\text{gen}}(N)$ by the extracted N dynamical systems. The error in each merge iteration step was calculated by the Euclidean norm: $Err(N) = ||Y - Y^{\text{gen}}(N)|| = \sqrt{\sum_{t=1}^{T} ||y_t - y^{\text{gen}}(N)_t||^2}$, which is the approximation of the overall log-likelihood score. The segmentation result of each merging step is shown in Figure 3.5. We see that the segmentation results from $N = 6$ to $N = 2$ have interval patterns that repeats three times, which correspond to the cycle of transition between the linear dynamical systems. Especially, the segmentation result of $N = 3$ exactly corresponds to the interval sequence in Figure 3.4 (a), which was used to generate the input sequence in Figure 3.4 (b).

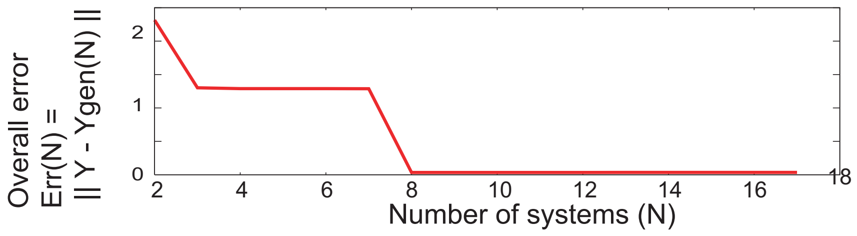
As we discussed in Subsection 3.3.5, we use the model fitting errors directly to extract candidates for the number of the dynamical systems. Figure 3.4 (d) shows the difference of the overall prediction error between current and previous steps $Err(N-1) - Err(N)$. We can find two peaks in $N = 6$ and $N = 3$ from this figure, where $N = 3$ is the ground truth and $N = 6$ is the largest number of dynamical systems in which the segmentation result has a cyclic pattern.



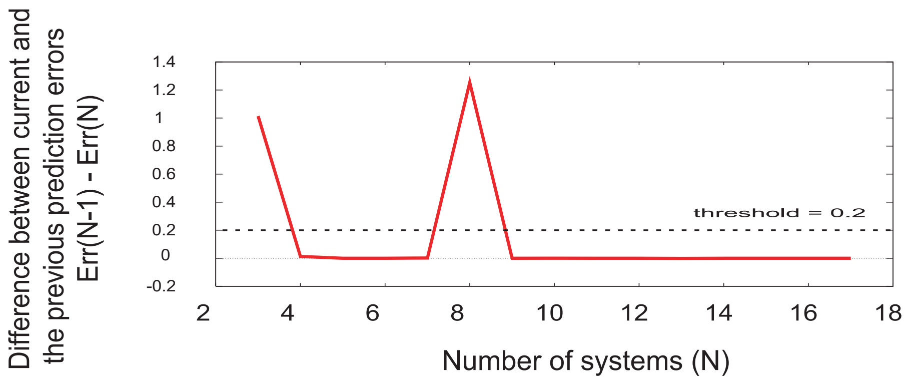
(a) Generated interval sequence of three dynamical systems



(b) Two-dimensional observation sequence generated from (a) (solid: the first element, dashed: the second element)



(c) Errors between the original and generated sequences



(d) Errors between the original and generated sequences

Figure 3.4: The Validation of the Clustering Algorithm (three-cycle input data).

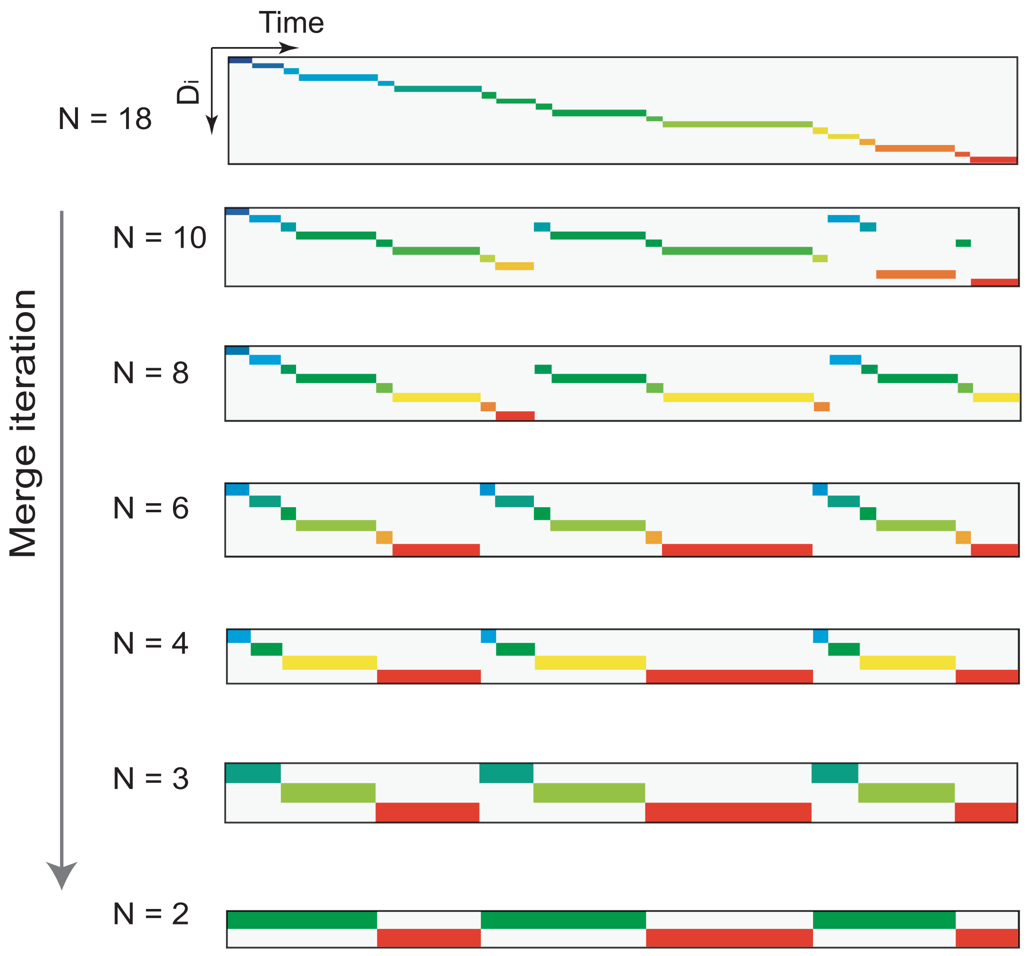


Figure 3.5: Segmentation result of each step in the clustering process.

3.5.2 Evaluation of the Refinement Process Using Simulated Data

[Step1] Clustering

Ten multivariate sequences were generated from the system. The parameters were same as the system in the previous experiments, except that the state transition probability A_{31} was set to zero to generate non-cyclic sequences.

In the sequences, three dynamic primitives were appeared in the temporal order of $D_1 \rightarrow D_2 \rightarrow D_3$, and only the length of the duration varied. Figure 3.6 (a) and (b) shows an example of the generated interval sequence and the two-dimensional observation sequence, respectively.

The proposed clustering process was evaluated by the sequence in Figure 3.6 (b). First, the initial interval set was determined by zero-crossing points of the first-order difference as we described in Subsection 3.3.2. Then, the initial dynamical systems were identified from the interval set. The number of the initial dynamical systems was $N = 6$. Figure 3.6 (c) shows the obtained interval sequences during the clustering process. The number of the dynamical systems was reduced from $N = 6$ to $N = 2$ by the iterative merging process of nearest dynamical system pairs. In each iteration step, two interval sets that belong to the nearest two dynamical systems were also merged.

The following parameters are the result of the clustering algorithm:

$$F^{(1)} = \begin{bmatrix} 0.01 & -0.14 \\ -0.10 & 0.20 \end{bmatrix}, g^{(1)} = \begin{bmatrix} 0.75 \\ 0.74 \end{bmatrix}, x_{\text{init}}^{(1)} = \begin{bmatrix} 1.0 \\ -1.0 \end{bmatrix}$$

$$F^{(2)} = \begin{bmatrix} 0.86 & 0.30 \\ -0.21 & -0.06 \end{bmatrix}, g^{(2)} = \begin{bmatrix} -0.30 \\ 1.12 \end{bmatrix}, x_{\text{init}}^{(2)} = \begin{bmatrix} 0.53 \\ 0.92 \end{bmatrix}$$

$$F^{(3)} = \begin{bmatrix} 0.49 & 0.09 \\ -0.11 & 0.75 \end{bmatrix}, g^{(3)} = \begin{bmatrix} 0.15 \\ -0.16 \end{bmatrix}, x_{\text{init}}^{(3)} = \begin{bmatrix} -0.63 \\ 0.60 \end{bmatrix}$$

[Step2] EM Algorithm

For the evaluation of the refinement process, we used the extracted dynamical systems in the clustering process. We manually selected $N = 3$ for the number of dynamical systems, which corresponds to the number of the original system that generated the typical sequence. Additional nine sequences were generated for the training data, and all the generated sequences including the typical sequence were used for the input of the EM algorithm.

The algorithm was initialized by the parameters found in the clustering process. Figure 3.7 (a) shows the searched interval sequences in the E step of each iteration step. We see that the partitioning of the intervals gradually converges to almost the same partitions as the original interval sequence shown in Figure 3.6 (a). The solid line in Figure 3.7 (b) shows the change of the overall log-likelihood of the system. We see that the algorithm almost converged at the ninth iteration. The dashed line in Figure 3.7 (b) shows the change of the overall log-likelihood when the duration distribution function $\hat{h}^{(ij)}$ was set to be uniform and the adjacent intervals was modeled to have no correlation. We see that the algorithm converges to a local optimum in this case.

The following parameters are the refined result via EM algorithms:

$$F^{(1)} = \begin{bmatrix} 0.60 & -0.10 \\ -0.10 & 0.20 \end{bmatrix}, g^{(1)} = \begin{bmatrix} 0.20 \\ 0.82 \end{bmatrix}, x_{\text{init}}^{(1)} = \begin{bmatrix} 1.00 \\ -1.00 \end{bmatrix}$$

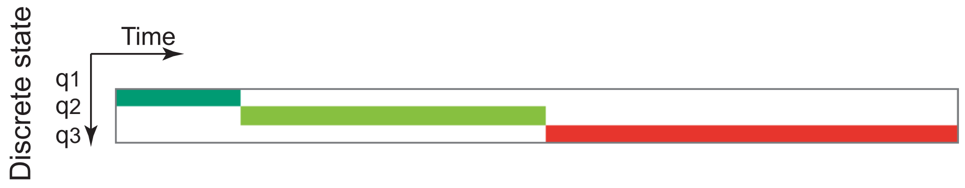
$$F^{(2)} = \begin{bmatrix} 0.32 & 0.02 \\ 0.06 & 0.52 \end{bmatrix}, g^{(2)} = \begin{bmatrix} -0.68 \\ 0.07 \end{bmatrix}, x_{\text{init}}^{(2)} = \begin{bmatrix} 0.25 \\ 1.00 \end{bmatrix}$$

$$F^{(3)} = \begin{bmatrix} 0.49 & 0.09 \\ -0.10 & 0.29 \end{bmatrix}, g^{(3)} = \begin{bmatrix} 0.60 \\ -0.60 \end{bmatrix}, x_{\text{init}}^{(3)} = \begin{bmatrix} 0.1 \\ -0.5 \end{bmatrix}.$$

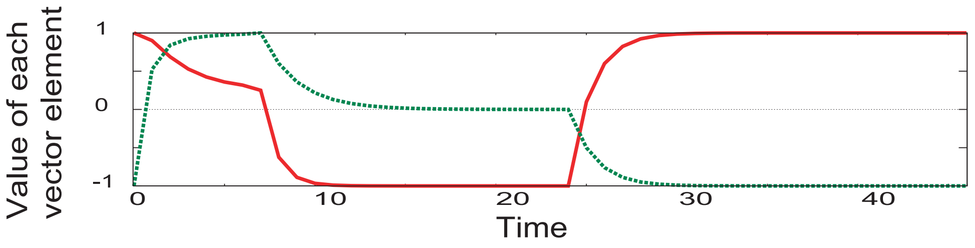
Consequently, the results of the clustering process become inaccurate if inappropriate sequences are selected for the typical training data. In spite of the inaccurate parameter estimation in the clustering process, the evaluation shows that the refinement process applied to all the training data recovers from the initial error. Especially, the meta-level features, such as modeling of duration lengths of intervals and relations between intervals, seem to work as constraints to the partitioning process in the EM algorithm.

3.5.3 Evaluation on Real Data

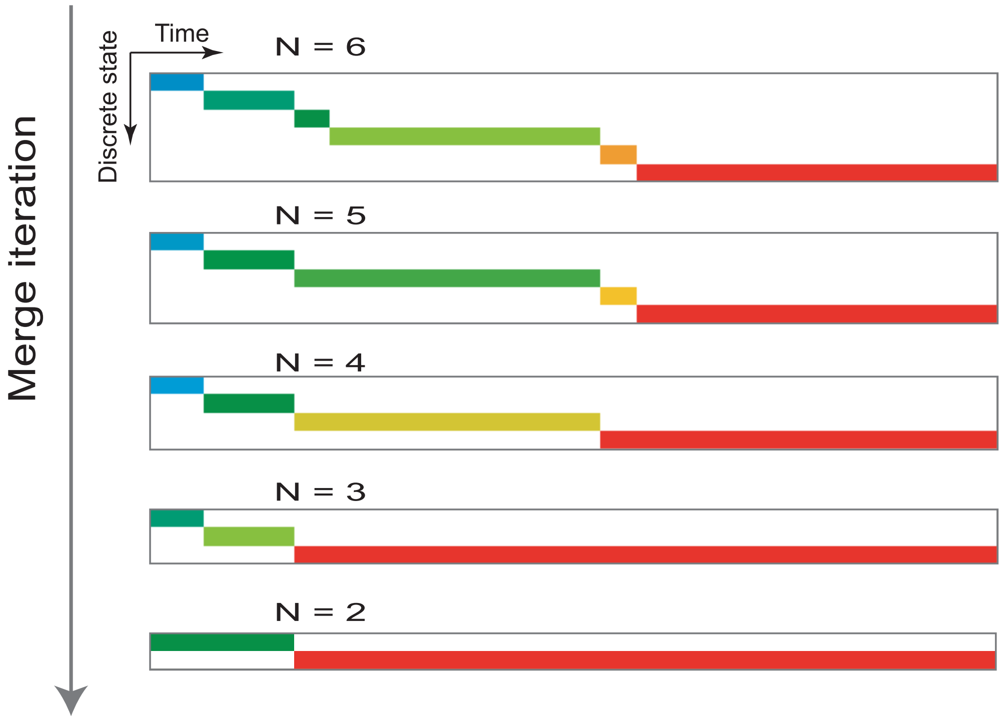
To evaluate the capability of the proposed method for real applications, we applied the clustering method to captured video data. A frontal facial image sequence was captured by 60fps camera during a subject was smiling four times. Facial feature points were tracked by the active appearance model [CET98, SEL03], and eight feature points around the right eye were extracted. The length



(a) Interval sequence generated for the typical training data

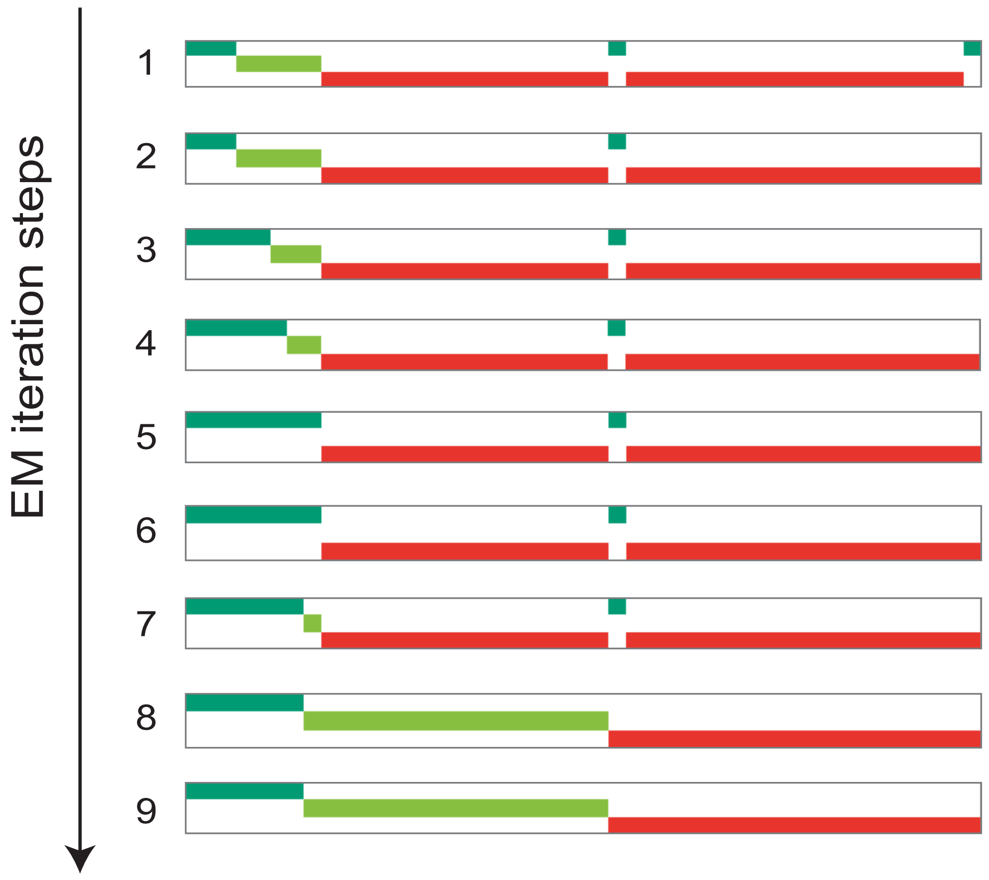


(b) Two-dimensional observation sequence selected for the typical data (solid: the first element, dashed: the second element)

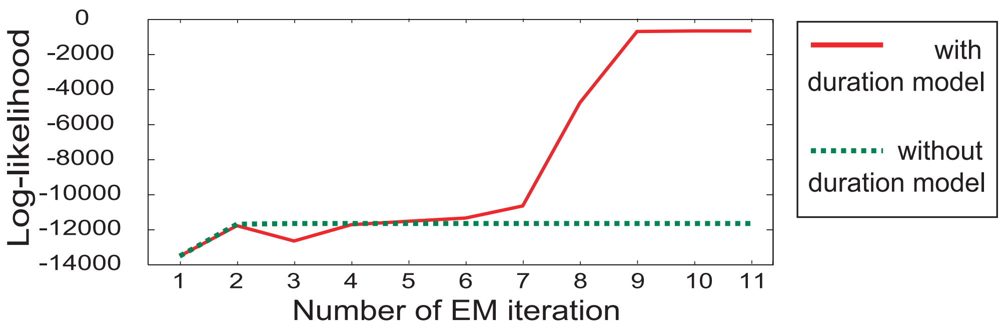


(c) Interval sequences partitioned by the clustered dynamical systems

Figure 3.6: The result of the clustering process applied to the typical training data shown in (b). The process was initialized by $N = 6$ and reduced the number of the dynamical systems (discrete states) by merging the nearest dynamical system pairs in each iteration steps.



(a) Searched interval sequence in each E step during the iteration



(b) The change of the overall log likelihood during the EM iteration

Figure 3.7: The result of the refinement process via the EM algorithm using all the training data.

of the sequence was $T = 1000$.

We applied the clustering method to the obtained 16-dimensional vector sequence that comprised x- and y-coordinates of the feature points (both coordinate coefficients were plotted together in Figure 3.8(a)). Figure 3.8(b) shows the overall model fitting errors between the original sequence Y and generated sequences $Y^{\text{gen}}(N)$ by the extracted N dynamical systems. The candidates of the number of the dynamical systems were determined as $N = 3$ and $N = 6$ by extracting the steps in which the difference $Err(N-1) - Err(N)$ exceeded given threshold (we used 0.01 in this experiment). The difference $Err(N-1) - Err(N)$ is shown in Figure 3.8(c).

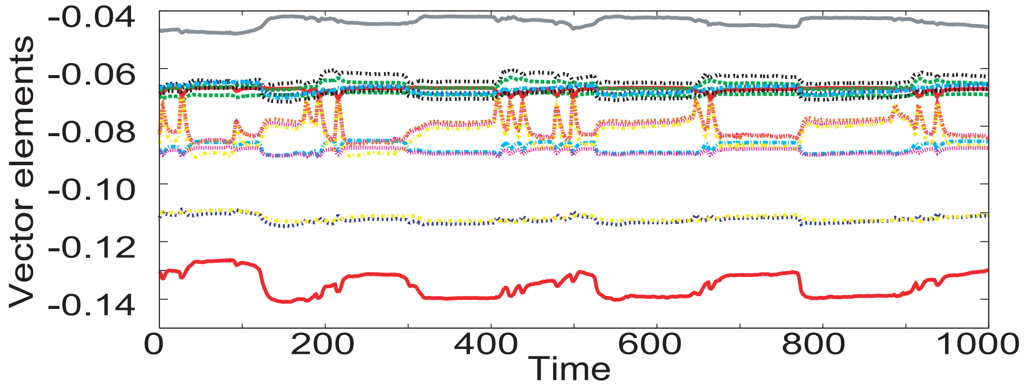
Figure 3.9(a) shows the intervals extracted from the clustering process when the number of dynamical systems was $N = 6$. Figure 3.9(b) shows the generated sequence from the extracted six dynamical systems, where each dynamical system was activated based on the partitioned intervals in Figure 3.9(a). The dominant dynamical systems D_3 and D_4 correspond to the intervals in which the eye had been remain closed and open, respectively. The other dynamical systems such as D_5 correspond to the eye blink motion.

Consequently, the history of model fitting errors during the clustering process helps us to decide the appropriate number of dynamical systems.

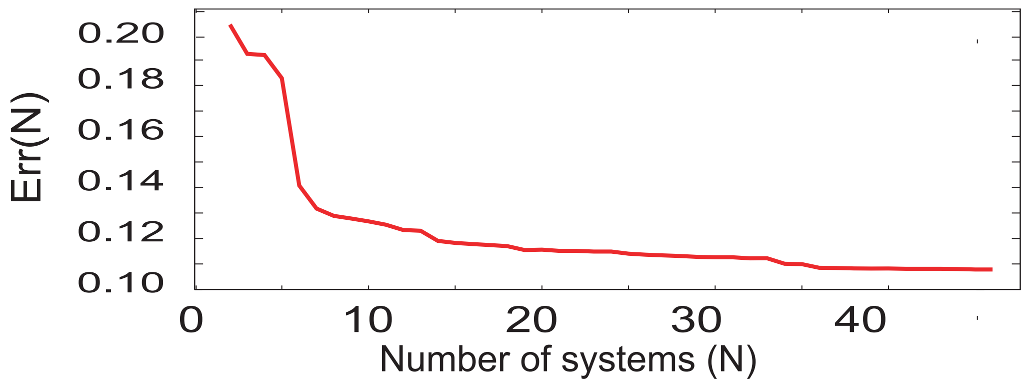
3.6 Discussion

In this chapter, we proposed a two-step learning algorithm to identify the interval-based hybrid dynamical system, which we introduced in the previous chapter. Especially, the hierarchical clustering of dynamical systems provides stable estimation technique of the number of dynamical systems and their parameters. This method can be regarded as one of model-based clustering methods [ZG03], however, the models are linear dynamical systems, which have a number of free parameters. We therefore proposed the constrained identification method of linear dynamical systems based on eigenvalues, which estimates stable dynamics from a small sample set of time-varying signals.

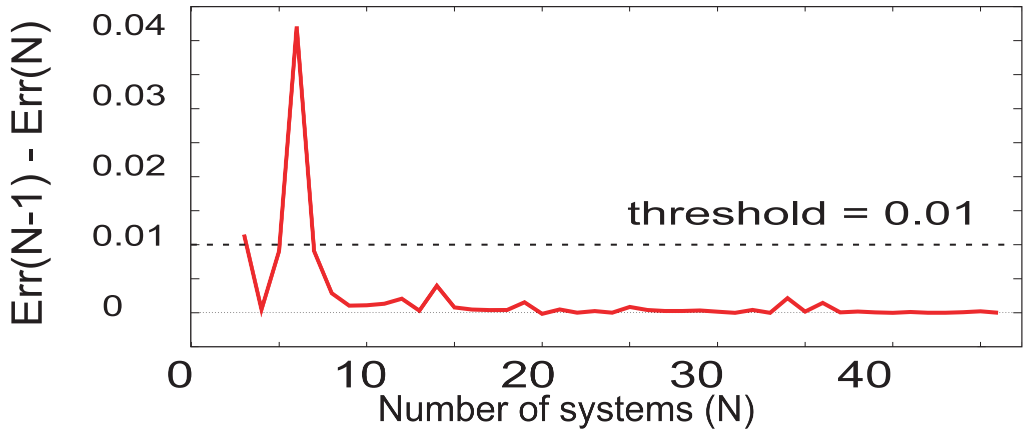
The experimental results on the simulated and real data show that the proposed parameter estimation method successfully finds a set of dynamical systems that is embedded in the training data and the transition probabilities between the dynamics with a modeling of adjacent interval duration lengths.



(a) Tracked feature points around the right eye

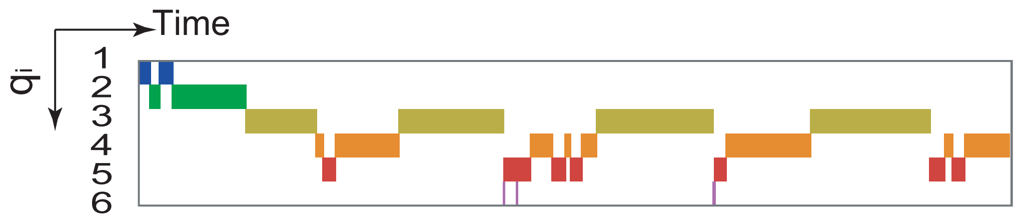


(b) Errors between original and generated sequences

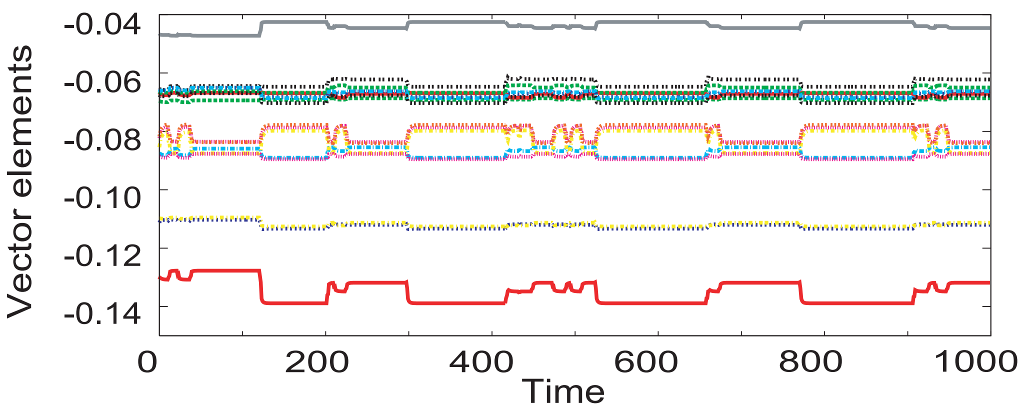


(c) Differences between current and the previous errors

Figure 3.8: Clustering results of the feature sequence around the right eye during a subject was smiling four times.



(a) Intervals partitioned by the dynamical systems



(b) Generated sequences from the clustered dynamical systems

Figure 3.9: Partitioned intervals during clustering and generated sequences from the clustered dynamical systems ($N = 6$).