RNNPB Introduction

1 Model overview

This document presents the main ideas behind our proposed model RNNPB. The main characteristic of the RNNPB is that chunks of spatio-temporal patterns of the sensory-motor flow can be represented by a vector of small dimensions. This vector plays the role of the bifurcation parameters of nonlinear dynamical systems. In other words, different vector values make the system generate different dynamic patterns. In our modeling, the nonlinear dynamical system is implemented by a Jordan-type recurrent neural network. The parametric biases (PB) that are allocated in the input layer function as the bifurcation parameters. The main advantage of utilizing the parameter bifurcation is that ideally the RNNPB can encode infinite number of dynamic patterns with modulating analog values of the PB vector.

The role of learning is to self-organize the mapping between the PB vector and behavioral spatio-temporal patterns. It is important to note that the PB vector for each learning pattern is self-determined in a non-supervised manner, without teacher signals. Another feature of the RNNPB is that the system works as both a behavior recognizer and generator as a mirror system after learning. When given a fixed PB vector, the RNNPB generates the corresponding dynamic patterns. On the other hand, when given target patterns to be recognized, the corresponding PB vectors are obtained through an iterative inverse computation.

In the learning phase, a set of movement patterns are learned through the forward model of the RNNPB by self-determining both the PB vectors, which are assigned differently for each movement pattern, and a synaptic weight matrix, which is common for all the patterns. The information flow of the RNNPB in the learning phase is shown in Figure 1(a). This learning is conducted using both target sequences of motor values m_t and the sensory values s_t. When given m_t and s_t in the input layer, the network predicts their values at the next time step in the output layer as $\hat{m_{t+1}}$ and $\hat{s_{t+1}}$. The outputs are compared with their target values m_{t+1} and s_{t+1} and the error generated is

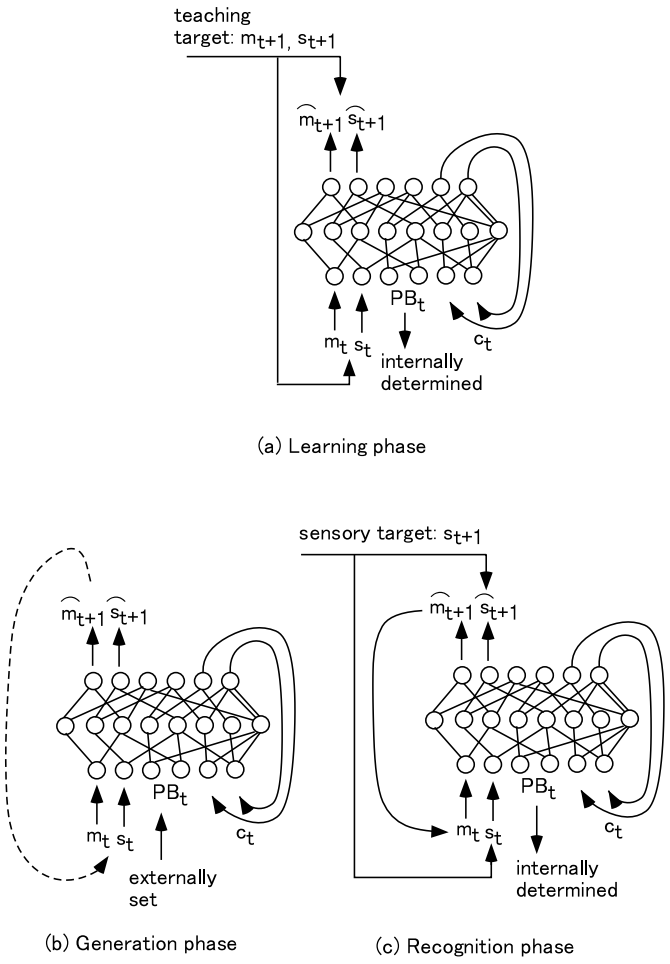

Figure 1: The system flow of RNNPB in learning phase (a) and testing phase (b).

back-propagated for the purpose of updating both the synaptic weights and PB vectors. Note that the determined synaptic weights are common to all learning patterns, but the PB vector is differently determined for each pattern. The manner of determining the PB vectors will be detailed in later sections. c_t represents the context units where the self-feedback loop is established from c_{t+1} in the output layer to c_t in the input layer. The context unit activations represent the internal state of the network.

After the learning is completed, the sensory-motor sequences can be generated by means of the forward dynamics of the RNNPB with the PB vectors fixed as shown in Figure 1(b). The PB vectors could be given from another network, as in the behavior-language association task described later, or self-determined through the recognition process, as in the imitative interaction task with the humanoid robot. In the generation phase, the RNNPB can be operated in a closed-loop mode where the next step's sensory-motor prediction outputs are fed back to the current step as inputs, as denoted by a dotted line on the left-hand side in Figure 1(b). Thus, the RNNPB can generate imaginary sensory-motor sequences without receiving the actual sensory inputs from the environment.

Figure 1(c) illustrates how the PB vectors can be inversely computed for the given target sensory sequences in the recognition phase. The RNNPB, when receiving the current sensory inputs s_t, attempts to predict their next vectors, $\hat{s_{t+1}}$, by utilizing the temporarily obtained PB vectors. The generated prediction error from the target value s_{t+1} is back-propagated to the PB units and the current PB vectors are updated in the direction of minimizing the error. The actual computation of the PB vectors is conducted by using the so-called regression window of the immediate past steps, by which the PB vectors can be modulated smoothly through the steps. (This mechanism will be detailed in the next section.) If pre-learned sensory sequence patterns are perceived, the PB vectors tend to converge to the values that were determined in the learning phase.

2 Computing the PB values

The PB vectors are determined through regression of the past sequence pattern. In the recognition phase, the regression is applied for the immediate past window steps L, by which the temporal profile of the PB, p_t from L steps before to the current step ct, is updated. The window for the regression shifts as time goes by while p_t is updated through the iterations. In the learning phase the regression is conducted for all steps of the training sequence patterns. (This means that the window contains the whole

sequence and it does not shift.)

The temporal profile of p_t in the sequence is computed via the back-propagation through time (BPTT) algorithm. In this computation ρ_t, the internal value of the parametric bias, is obtained first.

The internal value ρ_t changes due to the update computed by means of the error back-propagated to this parametric bias unit, which is integrated for a specific step length in the sequence. Then the parametric bias, p_t, is obtained by a sigmoid function of the output of the internal value. The utilization of the sigmoid function is just a way of computationally bounding the value of the parametric bias to a range of 0.0 to 1.0. In this way, the parametric bias is updated to minimize the error between the target and the output sequence.

For each iteration in the regression of the window, L steps of look-ahead prediction, starting from the onset step of the window, are computed by the forward dynamics of the RNN. Once the L steps of the prediction sequence are generated, the errors between the targets and the prediction outputs are computed and then back-propagated through time. The error back-propagation updates both the values of the parametric bias at each step and the synaptic weights. The update equations for the ith unit of the parametric bias at time t in the sequence are:

$$\delta {\rho_t}^i = k_{bp} \cdot \sum_{step=t-l/2}^{t+l/2} {\delta_t^{bp}}^i + k_{nb}(\rho_{t+1}^i - 2\rho_t^i + \rho_{t-1}^i) \quad (1)$$

$$\triangle \rho_{t(s+1)}^i = \epsilon \cdot \delta {\rho_t}^i + \eta \cdot \triangle \rho_{t(s)} \quad (2)$$

$$p_t^i = sigmoid(\rho_t) \quad (3)$$

In Eq. (1), $\delta\rho_t$, the delta component of the internal value of the parametric bias unit, is obtained from the summation of two terms. The first term represents the summation of the delta error, ${\delta_t^{bp}}^i$, in the parametric bias units for a fixed time duration l. ${\delta_t^{bp}}^i$, which is the error back-propagated from the output units to the ith parametric bias unit, is summed over the period from $t-l/2$ to $t+l/2$ time steps. By summing the delta error, the local fluctuations of the output errors will not affect the temporal profile of the parametric bias significantly. The parametric bias should vary only with structural changes in the target sequence. Otherwise it should become flat, or constant, over time.

The second term plays the role of a low pass filter through which frequent rapid changes of the parametric bias are inhibited. k_{nb} is the coefficient for this filtering effect. ρ_t is updated based on $\delta\rho_t$ obtained in Eq. (1). The actual update $\triangle\rho_{t(s+1)}$ at $s+1$ learning step from that at s learning step is computed by utilizing a momentum

term to accelerate convergence as shown in Eq. (2). Then, the current parametric bias p_t is obtained by means of the sigmoidal outputs of the internal values ρ_t in Eq. (3).