





Robot Arm Control Using Reward-Modulated Hebbian Learning

Koutaro Minato¹  and Yuichi Katori^{1,2}  

¹ Future University Hakodate, 116-2 Kamedanakano-cho,
Hakodate, Hokkaido 041-8655, Japan

katori@fun.ac.jp

² The Institute of Industrial Science, The University of Tokyo,
4-6-1 Komaba Meguro-ku, Tokyo 153-8605, Japan

Abstract. In recent years, soft robots with “softness” have been attracting much attention. Since soft robots have “softness”, they are expected to be able to perform delicate tasks that only humans can do. On the other hand, it is challenging to control. Therefore, in this research, we focused on reservoir computing with a biologically inspired learning algorithm. Reward-modulated Hebbian learning, one of the reservoir computing frameworks, is based on Hebbian learning rules and rewards and allows us to train the network without explicit teacher signals. The rewards are provided depending on the predicted and actual state of the environment influenced by the exploratory noise. We demonstrate that our model successfully controls the robot arm so that the tip position of the arm draws a given target trajectory.

Keywords: Reward-modulated Hebbian learning · Soft robotics · Reservoir computing

1 Introduction

In recent years, new research aimed at “softness” has been appeared in different science and technology fields. Traditional robotics has pursued the traditional engineering values of speed, power, precision, and certainty. However, it cannot make the “soft” motions that living things do. Soft robots are composed of soft materials, which are inspired by animals (e.g., squid, starfish, worms) that do not have hard internal skeletons [1]. Soft robots carry many advantages associated with mechanical softness [2, 3]. On the other hand, it is challenging to control soft robots compared to ordinary robots [4, 5]. Also, when a soft robot makes motion, its body produces diverse and complex dynamics. These are often high-dimensional, non-linear, and depend on the history of past stimuli.

Reservoir computing(RC) [6] is a kind of recurrent neural network. RC is constructed with a randomly connected neural network (dynamical reservoir), and only readout connections are learned. It is also suitable for handling non-linear and those containing past information. However, most RC models rely on supervised learning rules. In many motor tasks, including generation and planning

of motion, explicit teacher signal is not available. Reward-modulated Hebbian learning (RMHL) [7] is one of the frameworks of reservoir computing and is based on Hebbian learning rules and rewards obtained. This RMHL framework allows performing learning without using an explicit teacher signal.

In the present study, we address to control the complex dynamics of soft robots by using RMHL. We conduct an experiment using a robot arm of a 2-joint 6-muscle model of the forearm and upper arm, a muscle and skeletal system.

2 Methods

In the present study, we proposed the reservoir computing model for controlling the robot arm. The model is required to generate an appropriate control signal that causes tensions on muscles of the robot arm based on the given target time course of the tip of the arm. In the following subsections, we first explained the target of the control (robot arm), and then we describe the proposed model, including the network architecture and the network configuration procedure.

2.1 Robot Arm

The robot arm we use in the present study is a 2-joint, 6-muscle model of the forearm and upper arm, which is muscles and skeletal system (see Fig. 1)[8,9]. This robot arm was simulated using MuJoCo [10]. In this musculoskeletal system, muscles act as actuators that generate joint torque. In this model, the shoulder has a pair of monoarticular muscles and a pair of biarticular muscles for bending and stretching the shoulder. The elbow also has a pair of monoarticular muscles and a pair of biarticular muscles. This monoarticular muscle act only on one joint, and the biarticular muscles act on two joints.

In this robot arm model, each muscle is controlled with a control signal ranging from 0 to 1. 0 indicates that the muscle tension is 0, and 1 indicates that the muscle tension is maximum.

In this model, the robot arm is controlled using a feedback controller. The input signal to the feedback controller is based on the difference between the target angle of the robot arm joint and the actual angle of the robot arm joint. The output signal of the feedback controller is a control signal that causes tension on the muscles and moves the joint of the robot arm.

2.2 Network Architecture

The overall structure of the model used in the present research is shown in Fig. 2. The network model is composed of the reservoir and the feedback controller. The training of the reservoir is based on the concept of predictive coding [11], and the reward modulated Hebbian learning (RMHL). The reservoir generates the time course of the target joint angle and predicts the time course of the tip position of the arm. Training of the connections from the reservoir to the output layer that generates the target joint angle is achieved based on RMHL, while the training

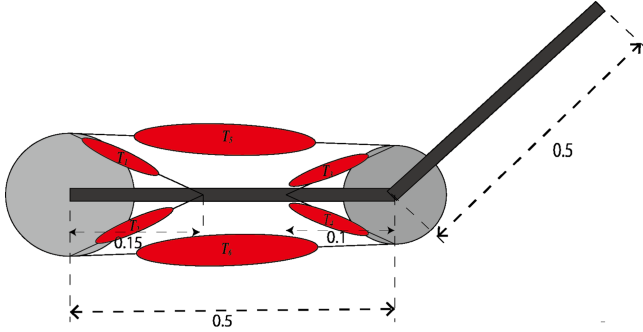


Fig. 1. Arm 2-joint 6-muscle model: Gray represents shoulder and elbow joints. Black represents forearm and upper arm. Red represents muscle. (Color figure online)

of the connection from the reservoir to the prediction layer is achieved by the FORCE algorithm [12]. The feedback controller generates the control signal for the muscles on the arm, and the controller is driven based on the differences between the target angle and the actual joint angles.

The reservoir is consisting of N sparsely and recurrently connected neurons. The firing rate of the neuron at time t is given by $\mathbf{r}(t) = \tanh[\mathbf{x}(t)]$. The dynamics of the internal state \mathbf{x} of the network is given by

$$\tau \dot{\mathbf{x}}(t) = -\mathbf{x}(t) + W^{in} \mathbf{c}(t) + W^{rec} \mathbf{r}(t) + W^e \mathbf{e}_{\mathbf{y}_a}(t), \quad (1)$$

where $\mathbf{e}_{\mathbf{y}_a}(t) = \mathbf{y}_t(t) - \mathbf{y}_a(t)$, and $\mathbf{c}(t)$ is the state of the context layer. This context information is passed from $\mathbf{c}(t)$ to the reservoir. Also, the context information is converted by the converter and passed to the target time course $\mathbf{y}_t(t)$, which is the tip position. $\mathbf{y}_p(t)$ is the predicted tip position, and $\mathbf{y}_a(t)$ is the actual tip position of the robot arm. τ is the time constant. $\mathbf{e}_{\mathbf{y}_a}(t)$ is the difference between target teacher signal $\mathbf{y}_t(t)$ and the actual tip position $\mathbf{y}_a(t)$. W^{rec} , W^{in} , and W^e denote the synaptic weights for recurrent connection with in the network, connections from context layer to the network, connections from $\mathbf{e}_{\mathbf{y}_a}(t)$ to the reservoir, respectively. These synaptic weights are configured with the following procedure.

The recurrent connection is configured sparsely and randomly with the following procedure. Firstly, generate a matrix of $N_x \times N_x$ with a connection rate β^{rec} , where N_x is the number of neurons in the reservoir. The elements of this matrix W_0^{rec} are randomly set to a non-zero value of -1 or 1 . Then, the spectral radius of this matrix ρ_0 is calculated. The synaptic connection of the recurrent connection W^{rec} is given as $W^{rec} = \alpha_r W_0^{rec} / \rho_0$ with the coefficient of synaptic strength α_r . Note that the spectral radius of W^{rec} equals α_r .

The connection for the contextual input W^{in} and the error feedback W^e are configured with the following procedure. First, generate a matrix of $N_x \times 2$ and $N_x \times 2$ with a connection rate 0.1 . The non-zero elements of these matrices W_0^{in} and W_0^e are set to, uniformly distributed random numbers from -1 to 1 .

W^{in} is given as $W^{in} = \alpha_i W_0^{in}$ with the synaptic strength α_i . W^e is given as $W^e = \alpha_e W_0^e$ with the synaptic strength α_e .

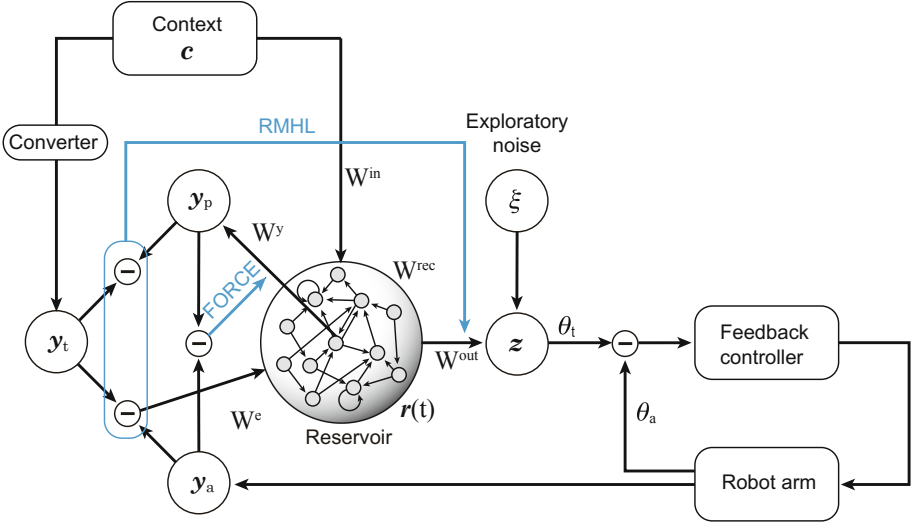


Fig. 2. Network architecture

The reservoir generate the target joint angle $\theta_t(t)$ in the output layer $z(t)$, and the output layer additionally receive the exploratory noise $\xi(t)$:

$$z(t) = W^{out} r(t) + \xi(t), \quad (2)$$

where W^{out} is the connections from the reservoir to the output layer $z(t)$. Here, we use the Ornstein-Uhlenbeck process as the exploratory noise. The Ornstein-Uhlenbeck process is the stochastic process given by the following stochastic differential equation

$$\dot{\xi}(t) = -\theta \xi(t) + \sigma \dot{W}(t), \quad (3)$$

where $W(t)$ presents the Wiener process, which has Gaussian increments, namely, the difference of the variable in a small-time step is normally distributed with a mean 0. Here, the exploratory noise is designed so that its amplitude is modulated with the e_{ya} : when the actual tip position $y_a(t)$ approaches the target tip position $y_t(t)$, the noise amplitude is decreased.

2.3 Reward-Modulated Hebbian Learning

The connection from the reservoir to the output layer is trained with reward-modulated Hebbian learning (RMHL). The reward is a scalar value utilized to

modulate the Hebbian learning on the output connection that strengthens the connection between correlated neurons.

In the present study, the reward is configured based on the comparison between the prediction and the actual tip position of the robot arm. The reward is provided when the actual tip position is closer to the target tip position than the predicted tip position. The reward $R(t)$ is defined with the following equation.

$$R(t) = \begin{cases} -\tanh[e_{ya}(t) - e_{yp}(t)] & \text{if } e_{ya}(t) - e_{yp}(t) < 0, \\ 0 & \text{if } e_{ya}(t) - e_{yp}(t) > 0, \end{cases} \quad (4)$$

where $e_{yp}(t) = |\mathbf{y}_t(t) - \mathbf{y}_p(t)|$ is the difference between the target tip position $\mathbf{y}_t(t)$ and the predicted tip position $\mathbf{y}_p(t)$. The reward $R(t)$ is provided when the $e_{ya} < e_{yp}$, and the weight change on the output connection $W^{out}(t)$ is given by

$$\Delta W^{out}(t) = R(t)\boldsymbol{\xi}(t)\mathbf{r}(t). \quad (5)$$

2.4 FORCE Learning

The connections from the reservoir to the prediction layer is trained with the first order reduced and controlled error (FORCE) learning, which is one of the online learning algorithms. According to the FORCE learning algorithm, W^y is updated with the following equations.

$$W^y(t + \Delta t) = W^y(t) + (\mathbf{y}_a(t) - \mathbf{y}_p(t))P(t)^T \mathbf{r}(t)^T. \quad (6)$$

The training progress so that the difference between the predicted tip position $\mathbf{y}_p(t)$ and the actual tip position $\mathbf{y}_a(t)$ is reduced. Where P is the running estimate of the inverse correlation among the output of the neurons and is updated with the following equation:

$$P(t + \Delta t) = P(t) - \frac{P(t)\mathbf{r}(t)\mathbf{r}(t)^T P(t)}{1 + \mathbf{r}(t)^T P(t)\mathbf{r}(t)}. \quad (7)$$

The initial value of P is $P(0) = I/\alpha_f$, where the matrix I is the identity matrix and α_f is the regularization parameter.

2.5 Feedback Controller

The feedback controller generates the control signal for the muscles on the arm, and the controller is driven based on the differences between the target angle and the actual joint angles. The feedback controller generates the control signal that causes tension on the muscles on the arm, based on the difference between the target joint angle $\boldsymbol{\theta}_t(t)$, which is the output of the reservoir, and the actual joint angle $\boldsymbol{\theta}_a(t)$. Based on the differences between the target and actual joint angle, the joint torque $\mathbf{F}(t)$ is determined using the proportional integral derivative (PID) control. The joint torque is given by

$$\mathbf{F}(t) = K_P(\boldsymbol{\theta}_t(t) - \boldsymbol{\theta}_a(t)) + K_I \int_0^t (\boldsymbol{\theta}_t(s) - \boldsymbol{\theta}_a(s))ds + K_D(\dot{\boldsymbol{\theta}}_t(t) - \dot{\boldsymbol{\theta}}_a(t)) \quad (8)$$

where, K_P , K_I , and K_D represent the gains of proportional control, integral control, and differential control, respectively.

The tension (control signal) of the robot arm can be obtained by using the pseudo-inverse matrix of matrix A and the joint torque [13]. The matrix A is determined by the moment arm of the robot arm. The moment arm A^T of the robot arm is determined as the follows based on the architecture of the robot arm shown in Fig. 1.

$$A^T = \begin{pmatrix} 0.15 & -0.15 & 0 & 0 & 0.5 & -0.5 \\ 0 & 0 & 0.1 & 0.1 & 0.5 & -0.5 \end{pmatrix} \quad (9)$$

The tension of the robot arm is given by

$$(T_1, T_2, T_3, T_4, T_5, T_6)^T = -(A^T)^\# \mathbf{F}(\mathbf{t}) \quad (10)$$

T_1 to T_6 are the tensions of muscles in the robot arm, which correspond to the muscles shown in Fig. 1. $(A^T)^\#$ is the pseudo inverse matrix of A^T . Since the value of the control signal is limited between 0 and 1, the tension is set to 0 if T_i is less than 0 and is set to 1 if T_i is more than 1.

The control signal from the feedback controller is sending to the robot arm, and the state of the robot arm is updated. The state of the robot arm can be obtained as actual joint angle $\boldsymbol{\theta}_a(t)$ and as actual tip position of the arm $\mathbf{y}_a(t)$.

The parameter values used in the present study are as follows. $N_x = 200$, $\tau = 0.2$, $\beta^{rec} = 0.1$, $\alpha_r = 0.8$, $\alpha_i = 0.1$, $\alpha_e = 0.1$, $\alpha_f = 0.1$, $\theta = 10$, $\sigma = 0.2$, $K_P = 4.0$, $K_I = 0.7$, $K_D = 0.3$.

3 Results

First, we evaluated the model with a task that requires the tip of the robot arm to draw a circle (see Fig. 3(A)). The experiment was performed in 5000 time-steps(25 s) per episode. In one episode, the target signal \mathbf{y}_t goes around the circle in Fig. 3(A) about 12 times. The experiment is repeated for 30 episodes.

Also, after each episode, the tip of the robot arm returns to the coordinates (1,0), which is the outstretched position. Learning was done from episodes 1 to 29, and learning was turn off in episode 30.

In the first episode, the robot arm did not draw a circle well, but as the learning progress, the robot arm could draw the circle (Fig. 3(C)(D)).

Figure 3(E)(F) shows the time course of the joint angles and the tip position. Upper panels in Fig. 3(E)(F) shows the target and actual joint angles $\boldsymbol{\theta}_t$ and $\boldsymbol{\theta}_a$ in the first and last episodes, respectively. Light blue and blue curves indicate the target $\boldsymbol{\theta}_t$ and the actual $\boldsymbol{\theta}_a$ angle of the shoulder joint, respectively. Purple and red curves indicate the target $\boldsymbol{\theta}_t$ and the actual $\boldsymbol{\theta}_a$ angle of the elbow joint, respectively. Lower panels in Fig. 3(E)(F) shows the target, actual, and predicted tip position \mathbf{y}_t , \mathbf{y}_a , and \mathbf{y}_p in the first and last episodes, respectively. Orange, red, and brown curves indicate \mathbf{y}_t , \mathbf{y}_a , and \mathbf{y}_p of y coordinate respectively. Blue, green, and purple curves indicate \mathbf{y}_t , \mathbf{y}_a , and \mathbf{y}_p of x coordinate respectively.

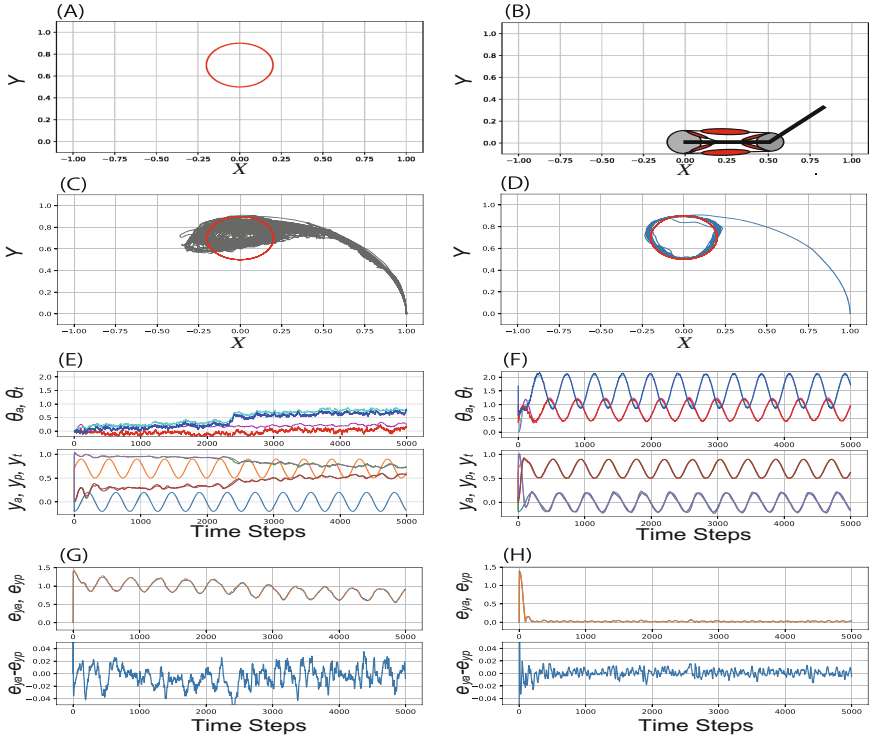


Fig. 3. (A) The circle that is required to be drawn by the arm tip. (B) The layout of the robot arm. The shoulder joint is fixed at the coordinates (0,0). (C): The orbits drawn by actual tip position \mathbf{y}_a in the first ten episodes. (D): The orbits drawn by \mathbf{y}_a in the last episode. (E) and (G): First episode. (F) and (H): Last episode. (Color figure online)

In the first episode, both joint angles exhibit slight fluctuation, and the tip position also fluctuates around the center of the target circle (Fig. 3 (E)). In the last episode, the orbit of the tip position roughly follows the target circle.

Figure 3(G)(H) shows the time courses of the differences in the actual, predicted, and target tip positions. Upper panels in Fig. 3(G) (H) shows the e_{ya} and e_{yp} in the first and last episodes. Blue curves represents e_{ya} and orange curves represents e_{yp} . Lower panels in Fig. 3(G) (H) shows the $e_{ya} - e_{yp}$ in the first and last episodes. The differences between the actual and the target tip position \mathbf{y}_a and between the predicted and the target tip position \mathbf{y}_p are larger in the early episodes, but after the learning, the differences are reduced. The difference $e_{ya} - e_{yp}$ is reflected to the reward value; if $e_{ya} - e_{yp}$ is negative, the reward is provided. The fluctuation of the difference $e_{ya} - e_{yp}$ is decreased as the learning progress. This indicates that the number of updates in W^{out} and amplitude of the exploratory noise become smaller as the learning progresses.

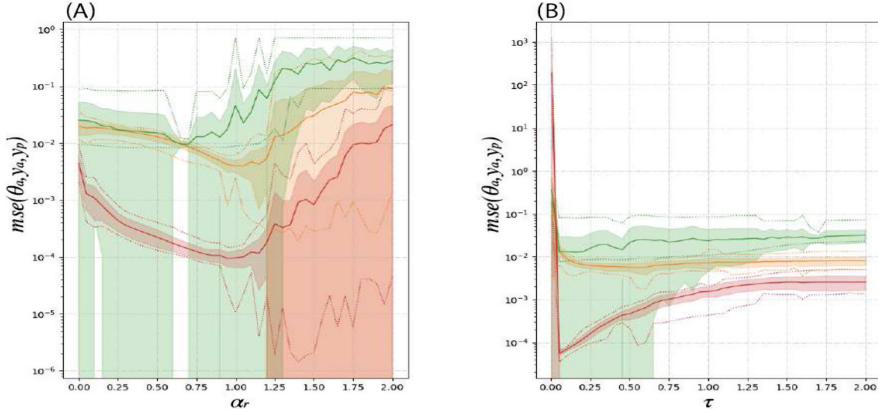


Fig. 4. The dependencies of the mean squared error (MSE) on the parameter values. (A) The dependencies of the MSEs on the strength of the recurrent connection α_r . (B) The dependencies of the MSEs on time constant of the reservoir τ . Red, orange, and green curves indicate the MSE of θ_t and θ_a , the MSE of y_t and y_a , and MSE of y_t and y_p , respectively. The bold curve indicates the mean value, the thin curve indicates the minimum and maximum values, and the filled area indicates the standard deviation. (Color figure online)

Figure 4 shows the dependencies of the mean squared error (MSE) of θ_t and θ_a , MSE of y_t and y_a , and MSE of y_t and y_p on the strength of the recurrent connection α_r and τ . The MSE between θ_t and θ_a is minimized around $\alpha_r = 0.8$. Also, when α_r exceeds 1, it starts to exhibit chaotic activity. Furthermore, the MSEs become small in the vicinity of 0.1 to 0.2 (Fig. 4(B)).

4 Conclusion

In the present study, we addressed controlling a robot arm with 2-joints and 6-muscles by the reservoir computing that generates the motor command (target joint angles) and predicts the state of the robot (tip position of the arm). We show that the model successfully controls the robot arm so that the tip position of the arm draws a given target trajectory.

This can be achieved by comparing the predicted arm motion with the actual arm motion. The actual arm motion is reflected in a realization of the exploratory noise. When the influence of the exploratory noise contributes to getting closer to the target motion than the predicted value, the learning progresses based on the realization of exploratory noise.

However, the actual orbit did not exactly match the target orbit, and there was a deviation. A possible reason for this deviation is that the feedback control does not perform completely redundant arm control. A possible solution to overcome the problem is to use an inverse static model and a model using a reverse

dynamics model together with PID control [13]. In the future, by incorporating these approaches into the present model, we expect to establish a further advantageous model for soft robot control.

Acknowledgements. This paper is based on results obtained from a project, JPNP16007, commissioned by the New Energy and Industrial Technology Development Organization (NEDO) and is supported by JSPS KAKENHI Grant Number 21H05163, 20H04258, 20H00596, and JST CREST(JPMJCR18K2).

References

1. Sheoherd, R.F., et al.: Multigait soft robot. *PNAS of USA* **108**(51), 20400–20403 (2011)
2. Kim, S., Laschi, C., Trimmer, B.: Soft robotics: a bioinspired evolution in robotics. *Trends Biotechnol.* **31**, 287–294 (2013)
3. Trivedi, D., Rahn, C.D., Kier, W.M., Walker, I.D.: Soft robotics: biological inspiration, state of the art, and future research. *Appl. Bionics Biomech.* **5**(3), 99–117 (2008)
4. Li, T., et al.: From the octopus to soft robots control: an octopus inspired behavior control architecture for soft robots. *Vie et Milieu*, **61**, 211–217 (2012)
5. Nie, X., Song, B., Ge, Y., Chen, W.W., Weerasooriya, T.: Dynamic tensile of soft materials. *Exp. Mach.* **49**(4), 451–458 (2009)
6. Jaeger, H.: Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the “echo state networks” approach. GMD report, German Nation. Res. Center Inf. Technol. **159**, 1–46 (2002)
7. Hoezer, G.M., Legenstein, R., Maass, W.: Emergence of complex computational structures from chaotic neural networks through reward-modulated hebbian learning. *Cerebr. Cortex* **24**(3), 677–690 (2012)
8. Izawa, J., Kondo, T., Ito, K.: Biological robot arm motion through reinforcement learning. *Proceed. 2002 IEEE Int. Conf. Robot. Autom.*, **4**, 3398–3403 (2002)
9. Kambara, H., Kim, K., Shin, D., Sato, M., Koike, Y.: Learning and generation of goal-directed arm reaching from scratch. *Neural Netw.* **22**(4), 348–361 (2009)
10. Todorov, E., Erez, T., Tassa, Y.: MuJoCo: a physics engine for model-based control. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033 (2012)
11. Katori, Y.: Network model for dynamics of perception with reservoir computing and predictive coding. In: Delgado-García, J.M., Pan, X., Sánchez-Campusano, R., Wang, R. (eds.) *Advances in Cognitive Neurodynamics (VI)*. ACN, pp. 89–95. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-8854-4_11
12. Sussillo, D., Abbott, L.F.: Generating coherent patterns of activity from chaotic neural networks. *Neuron*, **63**(4), 554–557 (2009)
13. Katayama, M., Kawato, M.: Virtual trajectory and stiffness ellipse during multi-joint arm movement predicted by neural inverse models. *Biol. Cybern.* **69**, 353–362 (1993)