Proceedings of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, June 14-19, 2009

A Motor Learning Neural Model based on Bayesian Network and Reinforcement Learning

Haruo Hosoya

Abstract— **A number of models based on Bayesian network have recently been proposed and shown to be biologically plausible enough to explain various phenomena in visual cortex. The present work studies how far the same approach can extend to motor learning, in particular, in combination with reinforcement learning, with the aim of suggesting a possible cooperation mechanism of cerebral cortex and basal ganglia. The basis of our model is BESOM, a biologically solid model for cerebral cortex proposed by Ichisugi, but extended with a reinforcement learning capability. We show how reinforcement learning can benefit from Bayesian network computations with unsupervised learning, in particular, in approximate representation of a large state-action space and detection of a goal state. By a simulation with a concrete BESOM network inspired by anatomically known cortical hierarchy to carry out a reach movement task, we demonstrate our model's stable and robust ability for motor learning.**

I. Introduction

In these few decades, an increasing number of evidences have shown that cerebral cortex may perform both hierarchical Bayesian inference [1]–[7] and unsupervised learning [8]–[11]. Among these, it has been shown that models based on Bayesian network can serve as a powerful and biologically plausible model and indeed have successfully explained various phenomena in visual cortex [1]–[3], [6]. However, it is not clear yet how the same argument can be extended to motor cortex, which has essentially the same hardware as visual cortex and therefore presumably has the same computational capability.

This work presents a rudimentary model study to address this question, in which we show how a Bayesian network framework can perform motor learning in combination with reinforcement learning. This combination is particularly important for motor learning since motor cortex has dense interconnection with basal ganglia [12], which appears to perform reinforcement learning from recent neurophysiological evidences [13]–[17]. Indeed, some researchers have already sketched a motor learning model based on such combination [16]. Our purpose here is to deepen this and propose a more detailed model with specific interaction mechanisms between these two computations, so as to suggest a possible cooperation strategy of cerebral cortex and basal ganglia.

Our approach is to adopt BESOM[1], which is a cerebral cortex model proposed by Ichisugi [7], and enhance it with reinforcement learning functionality, along the line sketched in [18]. The advantage of adopting the BESOM model is its strong connection with anatomical and physiological evidences. First, BESOM can naturally model a hierarchy of cortical areas [19] by a Bayesian network. Second, BESOM provides a variant of Pearl's belief propagation algorithm for probabilistic inference [20] that can straightforwardly be mapped to the six-layer structure of cerebral cortex and can explain various anatomical properties such as top-down and bottom-up connections that link specific cortical layers [19]. Thus, by constructing an instance of BESOM network to model a hierarchy of cortical regions, we can readily obtain some extent of biological plausibility. BESOM also provides a simple and powerful SOM-based learning algorithm to acquire conditional probabilities among nodes.

Our main contribution here is to show that a Bayesian network can provide useful computations that can be exploited by reinforcement learning. In particular, this allows for (1) approximate representation of a large state-action space by an "intermediate layer" that integrates sensory and motor signals and (2) measurement of a match between a prediction and an observation of certain key information by which we can detect a goal state and thereby calculate a reward. We will demonstrate that, with these mechanisms, our model can indeed achieve motor learning in a stable and robust manner, by using a simulation with a concrete BESOM network inspired by anatomically known cortical hierarchy to carry out a reach movement task. We will also discuss how plausibly the actual brain adopts these mechanisms and what benefits it would receive if this is the case.

The remainder of this paper is organized as follows. First, Section II gives an overview of BESOM and Section III presents its enhancement with reinforcement learning. Then, Section IV describes our reach movement model in BESOM and Section V shows our simulation results. Section VI relates our work with other and, finally, Section VII concludes this paper.

II. BESOM

This sections overviews the BESOM model. We first show the overall architecture and then describe two basic computations with BESOM: *recognition* based on Bayesian inference and *learning* based on SOM. Details of BESOM can be found in [7], [18].

The author is affiliated at Computer Science Department, University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033, Japan (email: hahosoya@is.s.u-tokyo.ac.jp).

The author would like to express a deepest thanks to Yuuji Ichisugi for his precious comments and suggestions.

[1] The name "BESOM" primarily stems from the English word "besom," which may look like the complex cortical hierarchy, but one may also consider the name as an acronym for BidirEctional SOM.

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 2, 2009 at 04:57 from IEEE Xplore. Restrictions apply.

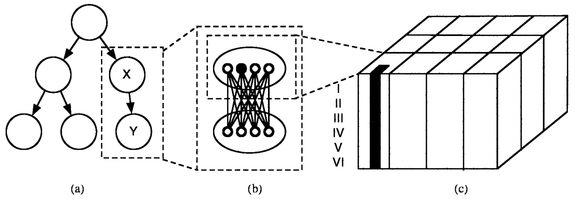


Fig. 1. The BESOM model. (a) A BESOM network (a whole or a part of cerebral cortex) is composed of multiple nodes (cortical areas or hypercolumns in them) and directed edges among them. (b) Each node has several units. We can consider that between all the units of a parent node to all the units of a child node are links representing conditional probabilities $P(Y|X)$. (c) A node corresponds to a hypercolumn (white block) and each unit (black block) of the node corresponds to a column in the hypercolumn.

A. Architecture

In BESOM, we consider a form of Bayesian network [20] as a neural model (Figure 1). That is, a BESOM network is a directed acyclic graph where each node, written X, represents a random variable. We assume that the range of each variable is finite (we write $|X|$ for the cardinality of X) and fixed at the same time as the structure of the network is given; we call each such value *unit*. Each edge linking between two variables represents dependency between these variables and is given conditional probabilities. That is, when an edge emanates from X to Y (in such case X is called parent of Y and conversely Y a child of X), we give the conditional probabilities $P(Y|X)$ to the edge. (Note that $P(Y|X)$ can thus be seen as a table of size $|Y| \times |X|$.) When an edge lacks between two nodes, there is, informally speaking, no direct dependency between these. A formal specification for the role of edges is that the joint probability distributions of all the nodes equals the product of every node's conditional probabilities given its parents: $P(X_1,\ldots,X_n) = \prod_i P(X_i|\mathrm{pa}(X_i))$ where $X_1,\ldots,X_n$ are the nodes in the network and $\mathrm{pa}(X)$ denotes the set of all the parents of node X.

Biologically, a BESOM network can be considered as a whole or a part of cerebral cortex. Each node represents a hypercolumn in a specific area and each unit of the node corresponds to a column within the hypercolumn. The direction of an edge represents the relationship between two areas in the cortical hierarchy, that is, an edge emanates from a node for a higher area to a node for a lower area. From these points of view, a probability distribution of a node's variable can be considered as a population of activities of the columns belonging to the hypercolumn corresponding to the node. Also, the conditional probabilities $P(Y|X)$ correspond to the weights of the connections from the columns for X to the columns for Y. Similar approaches can be found in the literature as "probabilistic population codes" [4].

B. Recognition by Bayesian inference

Let us consider using a BESOM network to get a form of interpretation of given inputs. Suppose that inputs are given to some of the nodes in the network in the way that each such node X_i is assigned a particular value x_i^e; we call such assignment *evidence*. Let e be the set of all evidences $\{X_i = x_i^e, \ldots\}$. From this, what we would like to obtain are the posterior probabilities $P(Y|e)$ of every node Y given the evidences; we call the posterior probabilities *beliefs* and write them by $\mathrm{BEL}(Y)$.

Pearl's belief propagation [20] is a well-known algorithm for computing beliefs. In this paper, we adopt it for our simulation and call it "recognition" from here on. Let us briefly overview the algorithm. First, we assume that each node X holds vectors $\mathrm{BEL}(X)$ ("beliefs"), $\lambda(X)$ ("retrospective support"), and $\pi(X)$ ("predictive support") each indexed with X's values, and each edge from X to Y holds vectors $\lambda_Y(X)$ ("bottom-up message") and $\pi_Y(X)$ ("top-down message") also indexed with X's values. Then, the algorithm works by bidirectionally passing around these messages among nodes. Specifically, we repeatedly apply the following update rules: for each node X with parents $U_1,\ldots,U_n$ and children $Y_1,\ldots,Y_m$,

$$\mathrm{BEL}(x) \leftarrow \nu_1 \lambda(x)\pi(x) \tag{1}$$

$$\lambda(x) \leftarrow \prod_j \lambda_{Y_j}(x) \tag{2}$$

$$\pi(x) \leftarrow \sum_{u_1,\ldots,u_n} P(x|u_1,\ldots,u_n) \prod_i \pi_X(u_i) \tag{3}$$

$$\lambda_X(u_i) \leftarrow \sum_x \lambda(x) \sum_{u_k:k\neq i} P(x|u_1,\ldots,u_n) \prod_{k\neq i} \pi_X(u_k) \tag{4}$$

$$\pi_{Y_j}(x) \leftarrow \nu_2 \pi(x) \prod_{k\neq j} \lambda_{Y_k}(x). \tag{5}$$

Here, ν_1 and ν_2 are the normalizing constants to make the sums $\sum_x \mathrm{BEL}(x)$ and $\sum_x \pi_{Y_j}(x)$ (respectively) 1. For a node X with evidence $X = x^e$, we use the rule

$$\lambda(x) \leftarrow \delta_{xx^e}$$

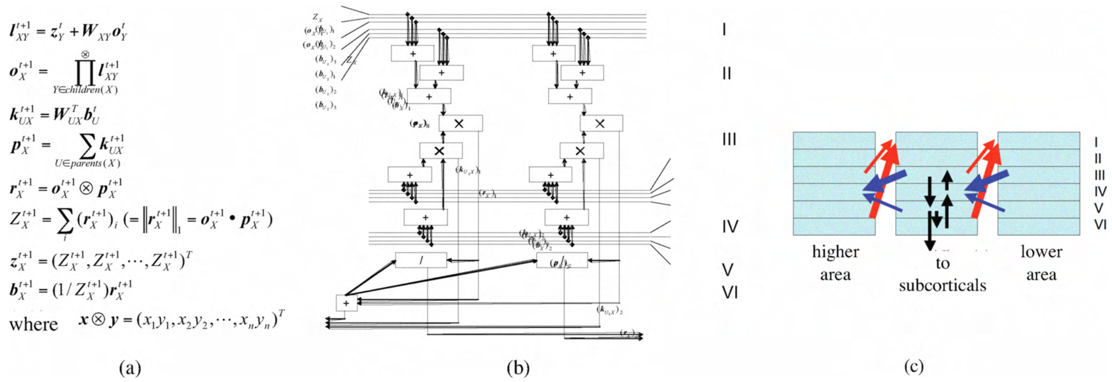
instead of (2) where $\delta_{xx'}$ is Kronecker's delta yielding 1 when $x = x'$ and 0 otherwise. For a root R, Rule (3) equals $\pi(r) = P(r)$, where $P(r)$ is the prior probability for the node R. We suppose throughout this paper that all nodes' priors are uniform.

If the network has no loop ignoring edge directions, it is known that a finite repetition of the above updates reaches a convergence and $\mathrm{BEL}(X)$ exactly yields the posterior probabilities [20]. For our purpose, however, the network may have a loop, in which case such convergence is not guaranteed and a finite repetition may only yield approximate posterior probabilities.

Note that Rules (3) and (4) need conditional probabilities $P(x|u_1,\ldots,u_n)$, whereas only available conditional probabilities in BESOM are $P(x|u_1),\ldots,P(x|u_n)$. For this reason, in this paper, we suppose that $U_1,\ldots,U_n$ are conditionally independent given X, by which, together with Bayes' theorem and uniformity of X's priors, we can use the following equation.

$$P(x|u_1,\ldots,u_n) = \prod_{i=1,\ldots,n} P(x|u_i)$$

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 2, 2009 at 04:57 from IEEE Xplore. Restrictions apply.



$$\boldsymbol{l}_{XY}^{t+1} = \boldsymbol{z}_Y^t + \boldsymbol{W}_{XY}\boldsymbol{o}_Y^t$$

$$\boldsymbol{o}_X^{t+1} = \prod_{Y \in children(X)}^{\otimes} \boldsymbol{l}_{XY}^{t+1}$$

$$\boldsymbol{k}_{UX}^{t+1} = \boldsymbol{W}_{UX}^T \boldsymbol{b}_U^t$$

$$\boldsymbol{p}_X^{t+1} = \sum_{U \in parents(X)} \boldsymbol{k}_{UX}^{t+1}$$

$$\boldsymbol{r}_X^{t+1} = \boldsymbol{o}_X^{t+1} \otimes \boldsymbol{p}_X^{t+1}$$

$$Z_X^{t+1} = \sum_i (\boldsymbol{r}_X^{t+1})_i \; (= \left\| \boldsymbol{r}_X^{t+1} \right\|_1 = \boldsymbol{o}_X^{t+1} \bullet \boldsymbol{p}_X^{t+1})$$

$$\boldsymbol{z}_X^{t+1} = (Z_X^{t+1}, Z_X^{t+1}, \cdots, Z_X^{t+1})^T$$

$$\boldsymbol{b}_X^{t+1} = (1/Z_X^{t+1})\boldsymbol{r}_X^{t+1}$$

$$\text{where} \quad \boldsymbol{x} \otimes \boldsymbol{y} = (x_1 y_1, x_2 y_2, \cdots, x_n y_n)^T$$

Fig. 2. Ichisugi's approximate belief propagation algorithm mapped to the neocortical laminar structure (quotation from [7] with slight modification). (a) An algorithm obtained by an approximation on Pearl's [20] is shown. (b) Each variable in the approximate belief propagation algorithm can be mapped to each neocortical layer in a straightforward way. (Two columns in a hypercolumn are shown.) The mapping shows a strong relationship to anatomically known connection properties in the actual neocortex as in (c). (c) Particular connection patterns can be found between specific layers, notably top-down connections from layer V to layer I and bottom-up ones from layer III to layer IV (cf. [21]).

Ichisugi has shown that, by certain approximation of Pearl's belief propagation algorithm, we can obtain an algorithm that can straightforwardly be mapped to the six-layer structure of the neocortex with strong anatomical and physiological plausibility in various respects (Figure 2) [7]. The approximation can be obtained by assuming a variant of noisy-OR property [20] (where each node satisfies $P(x|u_1, \ldots, u_n) \approx \sum_{i=1,\ldots,n} P(x|u_i)$) and removing the side conditions $i \neq k$ and $j \neq k$ from Rules (4) and (5). Though lack of space forces us to refer the reader to [7] for remaining details, the existence of such an algorithm forms a solid basis for biological connections of our model.

C. SOM-based learning

In BESOM, conditional probabilities $P(Y|X)$ are obtained by a SOM-based learning algorithm. First, for a given input, we perform the recognition algorithm described in the last subsection. Then, we invoke our learning rule given below for each combination of a node X and its child Y. For the ease of explanation, let us first present the most basic version of our learning rule. First, let the winners for both nodes $x^w = \mathrm{argmax}_x \mathrm{BEL}(x)$ and $y^w = \mathrm{argmax}_y \mathrm{BEL}(y)$. Then, update the conditional probabilities $P(Y|x^w)$ so as to incorporate the recognition results of the child nodes:

$$P(y|x^w) \leftarrow P(y|x^w) + \alpha \Big[u(y) - P(y|x^w) \Big] \qquad (6)$$

for each value y of Y, where α is a learning rate ($0 \leq \alpha \leq 1$) and the input vector $u(Y)$ is defined by:

$$u(y) = \delta_{yy^w}$$

It can be shown that, by using as learning rate an appropriate decreasing function of time, we can precisely obtain conditional probabilities $P(Y|X)$ during the learning [7].

Like in usual SOM, we incorporate neighborhood learning for obtaining a topographic map. For this, we first assume a topology among units by defining the distance $d(x_1, x_2)$ between every two units. In the sequel, we will consider 1D-nodes where the units are natural numbers or 2D-nodes where the units are pairs of natural numbers; in either case, Euclidean distance can be used.

In our neighborhood learning, we consider not only distances among outputs in the parent units (as usual), but also those among inputs in the child units as in [22]. To treat the first part, we adopt a usual learning rule:

$$P(y|x) \leftarrow P(y|x) + \alpha h(x, x^w) \Big[u(y) - P(y|x) \Big]$$

for each x and y. Here, $h(x, x^w)$ is a neighborhood function defined by the Gaussian function

$$h(x, x^w) = \exp\Big(- \frac{d(x, x^w)^2}{2\sigma^2} \Big)$$

where $\sigma > 0$ is a neighborhood width. The second part is implemented by using a blurred version of the input vector u:

$$u(y) = \nu h(y, y^w)$$

where ν is the normalizing constant $1/\sum_y h(y, y^w)$. The second part is needed since we will encode each quantity by a population of firing units and contiguity among such quantities crucially needs to be represented. For example, if we did not consider input distances, the map will not reflect the fact that a vector $(0, 1, 0, 0)$ (e.g., encoding a position) should be closer to $(0, 0, 1, 0)$ than $(0, 0, 0, 1)$.

As usual, in order to obtain a good topographic map, we take as σ a decreasing function of time. In the limit $\sigma \to 0$, we obtain exactly the basic learning rule (6) without neighborhood learning. Thus, even with neighborhood learning, we can expect that $P(Y|X)$ asymptotically approaches to the conditional probabilities during the learning.

Biological plausibility of SOM is rather a classical topic and discussed elsewhere in detail, e.g., [10]. We only point out here that SOM can be seen as an abstract version of a competitive learning algorithm that can be implemented by a hierarchical neural network with lateral inhibitory connections, which appears to exist in the neocortex [23].

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 2, 2009 at 04:57 from IEEE Xplore. Restrictions apply.

III. Extension with Reinforcement learning

We enhance the above described BESOM model with reinforcement learning. Our learning algorithm can be seen as a variant of Sarsa [15] adjusted to the BESOM framework.

In our model, we allow a node to have a reinforcement learning capability and call such node *RL-node*. We assume that the child nodes of an RL-node X can be divided to a set of *action nodes* A_i and a set of *state nodes* S_j. Since the RL-node X associates actions and states presented to these child nodes during the SOM-based learning, the units of X can be regarded as representing state-action pairs. However, this representation is approximate since there are usually less units than all combinations of states and actions. We also assume that the RL-node X holds a *value function* V^X from X's units to real numbers.

We also need to somehow determine a reward (in real number). We propose here a reward calculation mechanism based on "matching rate" information available from Bayesian network computation. The idea here is that we can consider that a goal state has arrived exactly when an expectation for certain "key" information matches an observation on the same information. For example, consider a reaching task where the key information is the position of the subject. Let us expect the subject to be at the goal position. If we observe that the subject is actually at the goal position, then we consider that it has reached the goal; otherwise, it has not. However, there are two issues: how to obtain such expectation and observation and how to measure matching between these. For the first issue, note that, if a Bayesian network contains a node, say Y, that represents the key information, both expectation and observation are already available at that node. That is, after performing the recognition algorithm given in Section II-B, the "predictive support" $\pi(Y)$ gives the expectation for the key information inferred from the surrounding context, whereas the "retrospective support" $\lambda(Y)$ gives the observation for the same information based on external inputs. For the second issue, we can calculate how well these two match by first normalizing $\lambda(Y)$ as $\lambda'(y) = \nu\lambda(y)$ (where ν is the normalizing constant $1/\sum_y \lambda(y)$) and then calculating the inner product of π and λ':

$$m = \sum_y \pi(y)\lambda'(y)$$

Let us call this *matching rate*. Then, the reward r is determined by the following:

$$r = \begin{cases} 1 & (m \geq m_{\text{th}}) \\ 0 & (\text{otherwise}) \end{cases}$$

where $0 \leq m_{\text{th}} \leq 1$ is a threshold.

Using this reward mechanism, we propose the following reinforcement learning algorithm. Initially, we set the value $V^X(x)$ of each unit to 1.0 in order to enforce exploration in early stages (a.k.a. optimistic strategy). Then, we repeat the following steps.

1) Select action:
   a) Input each current state s_j to the corresponding state node S_j.
   b) Perform the recognition algorithm (Section II-B) to obtain the beliefs $\mathrm{BEL}(X)$.
   c) Let $\hat{x} = \mathrm{argmax}_x \mathrm{BEL}(x) V^X(x)$.
   d) Choose each action $a_i = \mathrm{argmax}_{a_i} P(a_i|\hat{x})$ and execute it.
2) Obtain a reward r.
3) Update value:

$$V^X(\hat{x}_{\text{prev}}) \leftarrow V^X(\hat{x}_{\text{prev}}) + \alpha\Big[r + \gamma V^X(\hat{x}) - V^X(\hat{x}_{\text{prev}})\Big]$$

   where α is a learning rate ($0 \leq \alpha \leq 1$), γ is a discount rate ($0 \leq \gamma \leq 1$), and $\hat{x}_{\text{prev}}$ is the $\hat{x}$ determined in the previous pass. (We skip this "update value" step in the first pass, since $\hat{x}_{\text{prev}}$ has not been determined yet.)
4) Let $\hat{x}_{\text{prev}} \leftarrow \hat{x}$.

The basic structure of our algorithm is similar to the usual Sarsa algorithm except for the "select action" step. In this, we first perform Bayesian inference to figure out how likely each unit x of node X represents the current states (substeps (a) and (b)). We then take, among such units, the one $\hat{x}$ that has the best expected value (substep (c)), that is, the unit that maximizes its value multiplied by its likelihood. Finally, we choose, for each i, the best action a_i that is represented at $\hat{x}$ (substep (d)). The value for the selected approximate state-action pair $\hat{x}$ will be subject to change in the "update value" step.

Note in the above that, due to the approximation of the state-action space, our algorithm needs to involve probabilistic techniques to determine a state-action pair and an action to execute. If we assume that brain uses reinforcement learning and represents a state-action space in cerebral cortex, some kind of approximation of the space should be necessary because usable neurons are limited resources. Then, an important question is whether reinforcement learning can work even under such approximation. Answering this question is one of our purposes to conduct a simulation experiment.

Since our reward calculation mechanism based on matching rates is quite simple and general, one may ask whether some similar mechanism might be used in actual brain. Though it would anyway be a matter of speculation, we would like to point out that the matching rate information is explicitly present in Ichisugi's approximate belief propagation algorithm, namely, the variable Z_X in Figure 2 (a). Matching rates might then be used for reward calculation by, e.g., a path from neurons representing this variable (perhaps through prefrontal cortex) to dopamine neurons.

Though we have chosen Sarsa here, we do not claim at all that brain may use this algorithm. Indeed, although an increasing number of evidences indicate that basal ganglia may perform reinforcement learning, what specific algorithm is actually used there is still under active debate [13], [16], [17]. Under this situation, we have taken Sarsa just because it is one of the simplest known reinforcement learning algorithms and thus appropriate for studying a method to adapt it to Bayesian network. If some other algorithm later

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 2, 2009 at 04:57 from IEEE Xplore. Restrictions apply.

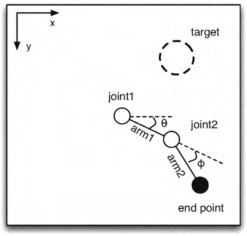


Fig. 3. Our reach movement task. See the text for details.

turns out to be the right one, we believe that some similar adaption could be used.

IV. BESOM Network for Reach Movement

In this section, we first specify our reach movement task and then describe our BESOM network and learning strategies to perform the task.

A. Task description

Suppose that a two-joint arm robot is placed on a horizontal plane, as shown in Figure 3. The lengths of arm1 and arm2 are equal, and joint1 is fixed at the center. Thus, the positional state of the robot can be described by two angles: θ (the angle between the x-axis and the arm1) and ϕ (the angle between arm1's extension and arm2). The ranges of these angles are restricted to $-90 \leq \theta < 180$ and $0 \leq \phi < 180$ (in degree). As motor commands, the arms' angle velocities $\Delta\theta$ and $\Delta\phi$ (more precisely, angle changes in a time unit) can be controlled. Their ranges are also restricted to $-4 \leq \Delta\theta < 4$ and $-4 \leq \Delta\phi < 4$ (in degree). As sensory information, "somatosensory" inputs θ and ϕ and "visual" inputs (x, y) can be used, where (x, y) tells the position of a visible object, in our case, either the arm end or the target.

Given a visually presented target on the horizontal plane, our goal is to control the arm so that its end point reaches the target. In addition, in order to study our mechanism for goal state detection, we consider a slightly complicated setting where, during the reach movement, the target is visible but the arm end is *not*. Such setting may naturally arise when we try to reach a lighting target in darkness or an object behind an obstacle (in the latter case, the target position would be remembered rather than visible). Note that the key in this setting is to infer the position of the arm end from the body position, which must be acquired before starting to learn reach movement.

B. Network structure

We use the BESOM network shown in Figure 4 to carry out our reach movement task. The network consists of seven nodes. We assume that V is a 2D-node and the other nodes are 1D. The pair of leaf nodes $M1$ and $M2$ represent motor commands $\Delta\theta$ and $\Delta\phi$ and the pair $S1$ and $S2$ represent positional states θ and ϕ. The leaf node V represents a visual position (x, y) of the arm end or the target. In addition, there are two parent nodes. One is PM node subjugating the two motor nodes $M1$ and $M2$ and the two somatosensory nodes $S1$ and $S2$, and the other is the PP node subjugating all sensory nodes $S1$, $S2$, and V. The PM node is an RL-node whose action nodes are $M1$ and $M2$ and whose state nodes are $S1$ and $S2$. The PP node is used for the association of somatosensory and visual inputs, by which we can convert between body and visual positions and measure matching between them, as we will detail below.

When actually representing a quantity in a node, we discretize it since each node can only have a finite number of units. For example, for the 1D-node $S1$, we assume that its units are integers s in the range $0 \leq s < n$, and then map an angle θ in the range $-90 \leq \theta < 180$ to the unit $s = \lfloor n(\theta + 90)/270 \rfloor$. We treat the 2D-node V similarly except that we assume that its units are integer pairs (t, u) and then map a position (x, y) to a unit (t, u) with each coordinate discretized. We also sometimes need a concrete value converted back from a unit representation. Since a concrete value corresponding to a unit is not unique, we take the midpoint of possible concrete values. For example, for $S1$, we use the conversion $\theta = 270(s + 0.5)/n - 90$. From now on, we assume that such conversion (in either direction) is done implicitly.

Our network is inspired by known hierarchical structure among several relevant primate cortical areas. Though we do not necessarily claim that these areas perform exactly as in our model, it would be worth speculating more specific correspondences.

- $M1$ and $M2$ might correspond to primary motor area (area 4), which directly emits motor commands [24, Chapter 38].
- PM might correspond to premotor area (area 6), which receives ascending projections from area 4 [19] and seems to be crucially involved in visually guided reach movement [24, Chapter 38].
- $S1$ and $S2$ might correspond to area 5, which is a higher somatosensory area that receives ascending projections from lower somatosensory areas and sends ones to area 6 [19] and have neurons that are known to provide various postural information [24, Chapter 23].
- V might represent some higher visual area.
- PP might correspond to area 7b, which is a posterior parietal area that receives ascending projections from both higher visual areas and area 5 [19] and have neurons that are known to to respond to visual stimuli presented in the same position as the hand [24, Chapter 23].

In the remainder of this section, we describe how reach movement can be learned with our network. It takes two phases: associative learning and reinforcement learning.

C. Phase 1: Associative learning

In this phase, we randomly move the arm and learn, at PM, association between motor commands ($\Delta\theta$ and $\Delta\phi$) and body positions (θ and ϕ) and, at PP, association between body positions and the visual position (x, y) of the arm end determined from the body positions.

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 2, 2009 at 04:57 from IEEE Xplore. Restrictions apply.

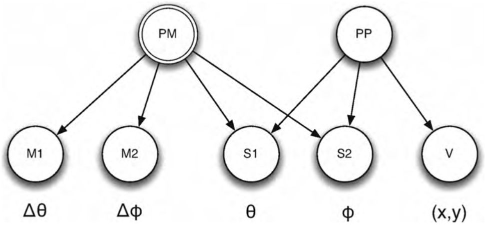


Fig. 4. A BESOM network to execute reach movement task: the PM node ("premotor") represents action-state pairs and, at the same time, performs reinforcement learning. The $M1$ and $M2$ ("primary motor") nodes represent actions $\Delta\theta$ and $\Delta\phi$; the $S1$ and $S2$ ("somatosensory") nodes represent states θ and ϕ; the V node ("visual") represents an (x, y)-position of the arm end or the target. The PP node ("posterior parietal") represents relationship between body-coordinate (in $S1$ and $S2$) and visual-coordinate (in V) positions.

Specifically, the learning repeats the following steps.

1) choose $\Delta\theta$, $\Delta\phi$, θ, and ϕ from uniform probability distributions. Further, determine the visual position (x, y) of the arm end from θ and ϕ.
2) input the quantities $\Delta\theta$, $\Delta\phi$, θ, ϕ, and (x, y) to the corresponding leaf nodes $M1$, $M2$, $S1$, $S2$, and V with the already mentioned discretization.
3) perform recognition (Section II-B).
4) perform SOM-learning (Section II-C).

After the learning, we can expect that each parent node yields a topographic map that represents combinations of inputs given to the child nodes. In the PM node, each unit represents tuples $(\Delta\theta, \Delta\phi, \theta, \phi)$. Since these four quantities are chosen uniformly and independently, the PM node attempts to represent all possible tuples in a uniform manner. However, since there are normally fewer units than all possible tuples, each unit is assigned to several tuples, thus approximating the whole tuple space. In the PP node, each unit, likewise, represents tuples $(\theta, \phi, (x, y))$. This time, however, since the visual position (x, y) is interdependent with the body position θ and ϕ, the units of the PP node are assigned to only tuples that are possible from this dependency. Thus, the PP node acquires relationship between (θ, ϕ) and (x, y).

The last point implies that the PP node can be used for converting between a body position (θ, ϕ) and a visual position (x, y). To infer the latter from the former, we first input body positions to the $S1$ and $S2$ nodes and then perform the recognition algorithm (Section II-B); the result appears as the posteriors $P(V|S1, S2)$ in the V node. The converse is similar.

This idea of using an "intermediate layer" for the association of visual and body positions and thereby performing coordinate transformation is rather common and has repeatedly been presented in the literature, together with numerous arguments of physiological plausibility (see [25] for a comprehensive review).

D. Phase 2: Reinforcement learning

In this phase, we perform reinforcement learning whose goal is to move the arm to reach the target, without using the visual information on the arm end. For reward calculation, we use the mechanism based on matching rates described in Section III. However, we need to choose what key information to expect and observe. The position of the arm end is not appropriate since it cannot be observed: the arm is invisible. Thus, we choose here the target position as the key information since the target is visible and its position can be inferred by using the body position of the arm end through the PP node. More precisely, if we give the body position to the nodes $S1$ and $S2$ and the target position to the node V, the recognition algorithm yields the inferred target position as $\pi(V)$ and the observed target position as the "retrospective support" $\lambda(V)$. Then, reward calculation follows the method described in Section III. (Alternatively, we could take the body position of the arm end as the key information, in which case we can infer the body position from the target position given to the node V and match it with the observed body position given to the nodes $S1$ and $S2$.)

With this reward rule, the reinforcement learning phase goes as follows. Each episode starts with placing the arm in a certain position and then simply performs the learning algorithm described in Section III where, as mentioned above, the actions are $\Delta\theta$ and $\Delta\phi$ given to the nodes $M1$ and $M2$ and the states are θ and ϕ given to the nodes $S1$ and $S2$. Each action execution simply performs $\theta \leftarrow \theta + \Delta\theta$ and $\phi \leftarrow \phi + \Delta\phi$, provided that these do not exceed their ranges.

Careful readers may have noticed, but our model works only for a target that is fixed to a single position throughout a simulation. This is because we have only a single value function that is not dependent on the target position. Handling any target position seems to require a substantial extension of our model. We continue to discuss this point in Section VII.

V. Simulation

We have constructed a computer simulation program for our reach movement model described in the last section. This section shows and discusses the results. First of all, we set the number of units for each node as $|M1| = |M2| = |S1| = |S2| = 9$, $|PM| = 80$, $|PP| = 40$, and $|V| = 36$ (we use a grid of size 6×6 for the 2D-node V).

We first ran the associative learning phase just as described in Section IV-C. We used the constant learning rate $\alpha = 0.05$ but the time-decreasing neighborhood width $\sigma(t) = 5000/(1000 + t)$. We repeated the learning step for 10000 times. Figure 5 shows the results of this phase obtained at the nodes PM and PP. We can see that each acquired a topographic map of the inputs to the child nodes where contiguities in these inputs are overall preserved. Since the inputs to PM's child nodes were chosen uniformly, the map was formed in a way that uniformly covers all input combinations. On the other hand, since the input (x, y) to V was interdependent with the inputs θ and ϕ to $S1$ and $S2$, this relationship was reflected in the map acquired at PP. Note that both parent nodes have much fewer units than all the combinations of the inputs and therefore

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 2, 2009 at 04:57 from IEEE Xplore. Restrictions apply.

those nodes significantly approximate the input space. In particular, PM has only 80 units whereas the whole state-action space has $9^4 = 6561$ combinations. Nonetheless, the remaining simulation yielded rather stable results. Moreover, the stability seemed robust against the number of units—the performance did not degrade much even when the number was changed by half or double. An additional note is that a smooth topographic map like in Figure 5 emerged only when we considered, in the SOM-learning, not only distances among output units but also those among input units. We observed that, without such smoothness, the reinforcement learning phase was extremely unstable, tending to get stuck in local minima.

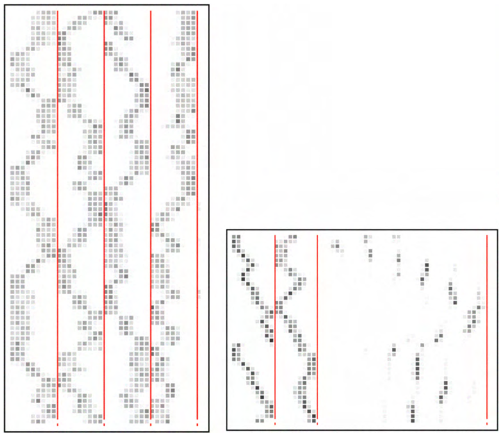


Fig. 5. The result of the associative learning phase: topographic maps were acquired at PM (left) and PP (right). In each map, each dot represents the conditional probability of each unit of a child node given a unit of a parent node, where a darker dot means a higher probability. The vertical axis represents the units of the parent node and the horizontal one the units of each child, where the units of the multiple child nodes are displayed in separated blocks in the same order as in Figure 4. For V (the right-most block of PP), its each unit (t, u) is placed at position $t + 6u$ (recall that $|V| = 6 \times 6$).

Next, we moved on to the reinforcement learning phase as described in Section IV-D. Before starting this phase, in order to make sure that our method to calculate rewards was reasonable, we looked into the relationship between the matching rate and the actual distance between the target and the arm end, varying the position of the target and the arm angles. Figure 6 shows the relationship. We can clearly see that the matching rate is significantly high for short distances and therefore provides a good criterion to measure how close the arm end is to the target. Based on the distribution, we set the threshold for giving a reward as $m_{\mathrm{th}} = 0.2$.

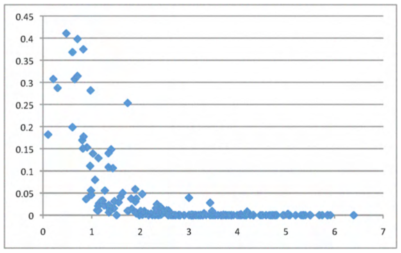


Fig. 6. The matching rates (vertical axis) against the distances (horizontal axis): a shorter distance tended to give a higher matching rate.

In the reinforcement learning phase, we took 500 steps for each episode. Each episode began with initializing the arm position to $\theta = 45$ and $\phi = 0$ and fixing the target to a position midway from the center to the northeast. We set the constant learning rate $\alpha = 0.05$ and discount rate $\gamma = 0.9$. Figure 7 shows the total rewards given in each episode. We can see that the learning hardly obtained rewards in early stages. However, after it found by chance where the target was (around episode 10), it quickly acquired how it could get there, obtaining increasingly high rewards, and finally reached a convergence (around episode 17).

The overall behavior of the algorithm seemed to be rather stable in the sense that every run eventually converged to a meaningful solution. However, although the particular run shown above found a good solution rather quickly, other runs did not necessarily; sometimes, the solution found was rather disappointing or the convergence was rather slow. We consider that this was because we used a rather simplistic reinforcement learning algorithm. We did not, however, pursue an effective learning algorithm since our focus here is to study a cooperation strategy of Bayesian network, unsupervised learning, and reinforcement learning and to suggest a possible interplay between cerebral cortex and basal ganglia.

VI. Related work

Several attempts have been made to combine SOM and reinforcement learning in the context of robotics. Touzet [26] has used SOM for the efficient representation of a large state-action-value triple space and Smith [27] has used it for mapping a continuous state or action to a discrete index space into a value look-up table. Relative to both, our novelty here is to adapt such combination to a Bayesian network framework, thus giving a probabilistic semantics. Further, we have pointed out that such Bayesian framework can also provide a way to detect a goal state and have discussed how brain would benefit from these Bayesian computations if it actually adopts them.

Though reach movement by itself is not our focus here, it is worth mentioning other work addressing this classical problem. In particular, some have formalized models for the computation of optimal trajectories of arm movement as optimization problems for certain objective functions [28], [29]. Others have proposed a supervised motor learning strategy called feedback-error-learning as a model for the execution of such trajectories [30]. These pieces of work mainly focus on how the brain acquires *feedforward* motor control. On the other hand, our model basically treats learning of *feedback* control since it generates motor commands always based on sensory inputs. As discussed in [30], although the brain must have a feedforward controller (presumably in cerebellum) for the efficient motor control, it must also have a feedback controller (presumably in cerebral cortex) that provides teacher signals to the feedforward controller. Indeed, considering studies on disorders in the human cerebellum, it seems that

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 2, 2009 at 04:57 from IEEE Xplore. Restrictions apply.

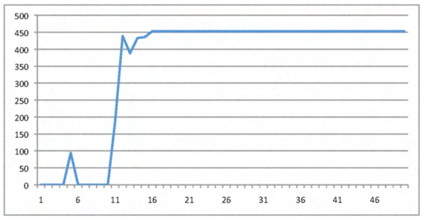


Fig. 7. The result of the reinforcement learning phase: the obtained total rewards (vertical axis) are shown for each episode (horizontal axis).

the use of only feedback control should be able to accomplish reach movement though the trajectory would not be smooth at all [24, Chapter 42]. In fact, the initial motivation for starting the present work was to investigate a mechanism for such feedback control, though further research is still needed to address this question.

VII. Conclusion

In this paper, we have studied a neural model that combines BESOM with reinforcement learning for the purpose of explaining a possible cooperation between cerebral cortex and basal ganglia. In this, we have shown that a Bayesian network can provide various computations useful for reinforcement learning, namely, approximate representation of state-action space and detection of a goal state. Through our simulation results with an anatomically inspired BESOM network for a reach movement task, we have demonstrated that our model can stably and robustly achieve motor learning.

Although brain could receive much computational benefit from such cooperation mechanisms between Bayesian network and reinforcement learning, much more physiological evidences must be collected in order to claim that brain really adopts these. Also, our current model needs further refinement. In particular, the model requires the target of the reaching task to be set at a fixed position and never changed throughout a simulation, which is clearly inconsistent with actual brain. This unsatisfaction is apparent from the fact that the PM node in our model does not receive any visual signal and therefore cannot take the target position as a part of the state. However, in actual cerebral cortex, premotor receives projections from posterior parietal association areas (which in turn receives visual inputs) [24, Chapter 38]. A possible way for the generalization might be to simply add an edge from PM to V, which could make the value function also dependent on the target position. However, our feeling is that this would mix up values of states for different positions of the target and considerably degrade the learning performance. Alternatively, we are currently considering extending the model so that it can select a separate value function depending on the target position, perhaps in the approach of "mixture of experts," though this would require a nontrivial extension to the BESOM model.

References

[1] T. S. Lee and D. Mumford, "Hierarchical bayesian inference in the visual cortex," *Journal of Optical Society of America A*, vol. 20, no. 70, pp. 1434–1448, 2003.

[2] R. P. Rao, "Bayesian computation in recurrent neural circuits," *Neural Computation*, vol. 16, pp. 1–38, 2004.

[3] ——, "Bayesian inference and attentional modulation in the visual cortex," *Neuroreport*, vol. 16, no. 16, pp. 1843–1848, 2005.

[4] W. J. Ma, J. M. Beck, P. E. Latham, and A. Pouget, "Bayesian inference with probabilistic population codes," *Nature Neuroscience*, vol. 9, no. 11, pp. 1432–1438, 2006.

[5] K. Doya, S. Ishii, A. Pouget, and R. P. N. Rao, Eds., *Bayesian Brain*. MIT Press, 2007.

[6] D. George and J. Hawkins, "A hierarchical Bayesian model of invariant pattern recognition in the visual cortex," in *Proceedings of International Joint Conference on Neural Networks*, vol. 3, 2005, pp. 1812–1817.

[7] Y. Ichisugi, "A cerebral cortex model that self-organizes conditional probability tables and executes belief propagation," *International Joint Conference on Neural Networks (IJCNN 2007)*, pp. 178–183, 2007.

[8] C. von der Malsburg, "Self-organization of orientation-sensitive cells in the striate cortex," *Biological Cybernetics*, vol. 14, pp. 85–100, 1973.

[9] H. Barlow, "Unsupervised learning," *Neural Computation*, vol. 1, pp. 295–311, 1989.

[10] T. Kohonen, *Self-Organizing Maps (3rd Edition)*. Springer, 2000.

[11] B. Olshausen and D. Field, "Emergence of simple-cell receptive field properties by learning sparse code for natural images," *Nature*, vol. 381, 1996.

[12] G. E. Alexander, M. R. DeLong, and P. L. Strick, "Parallel organization of functionally segregated circuits linking basal ganglia and cortex," *Annual Review of Neuroscience*, vol. 9, pp. 357–381, 1986.

[13] J. Houk, J.L.Davis, and D. Beiser, Eds., *Models of Information Processing in the Basal Ganglia*. MIT Press, 1995.

[14] W. Schultz, "Predictive reward signal of dopamine neurons," *Journal of Neurophysiology*, vol. 80, pp. 1–27, 1998.

[15] R. Sutton and A. Barto, *Reinforcement learning*. MIT Press, 1998.

[16] K. Doya, "What are the computations of the cerebellum, the basal ganglia and the cerebral cortex?" *Neural Networks*, vol. 12, pp. 961–974, 1999.

[17] K. Samejima, Y. Ueda, K. Doya, and M. Kimura, "Representation of action-specific reward values in the striatum," *Science*, vol. 310, no. 5752, pp. 1337–1340, 2005.

[18] Y. Ichisugi, "Current status of understanding the information processing priciples in the brain," National Institute of Advanced Industrial Science and Technology, Tech. Rep. AIST07-J00012, 2008, in Japanese.

[19] D. J. Felleman and D. C. V. Essen, "Distributed hierarchical processing in the primate cerebral cortex," *Cerebral Cortex*, vol. 1, pp. 1–47, 1991.

[20] J. Pearl, *Probabilistic Reasoning In Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1997.

[21] D. Pandya and E. Yeterian, "Architecture and connections of cortical association areas," in *Cerebral Cortex*, A. Peters and E. G. Jones, Eds. Plenum Press, 1985, vol. 4, pp. 3–61.

[22] Y. Ichisugi, "On structure learning of Bayesian network performed by neural networks in cerebral cortex," in *Proceedings of SIG-FPAI*. Japanese Society for Artificial Intelligence, Nov 2008, in Japanese.

[23] Z. F. Kisvárday and U. T.Eysel, "Functional and structural topography of horizontal inhibitory connections in cat visual cortex," *European Journal of Neuroscience*, vol. 5, pp. 1558–1572, 1993.

[24] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, *Principles of Neural Science (4Rev Edition)*. McGraw-Hill Medical, 2000.

[25] A. Pouget and L. H. Snyder, "Computational approaches to sensorimotor transformations," *Nature Neuroscience*, vol. 3, pp. 1192–1198, 2000.

[26] C. F. Touzet, "Neural reinforcement learning for behaviour synthesis," *Robotics and Autonomous Systems*, vol. 22, no. 3-4, pp. 251–281, 1997.

[27] A. J. Smith, "Applications of the self-organising map to reinforcement learning," *Neural Networks*, vol. 15, pp. 1107–1124, 2002.

[28] T. Flash and N. Hogan, "The coordination of arm movements: An experimentally confirmed mathematical model," *The Journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[29] Y. Uno, M. Kawato, and R. Suzuki, "Formation and control of optimal trajectory in human multijoint arm movement," *Biological Cybernetics*, vol. 61, pp. 89–101, 1989.

[30] M. Kawato, "Feedback-error-learning neural network for supervised motor learning," in *Advanced Neural Computers*, R. Eckmiller, Ed. Elsevier Science Publishers, 1990, pp. 365–371.

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 2, 2009 at 04:57 from IEEE Xplore. Restrictions apply.