NAIST-IS-DD0761010

Doctoral Dissertation

Goal-Oriented Representations of the External World: A Free-Energy-Based Approach

Makoto Otsuka

February 20, 2010

Department of Bioinformatics and Genomics
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Makoto Otsuka

A thing of beauty is a joy forever.

- John Keats

False-imagination teaches that such things as light and shade, long and short, black and white are different and are to be discriminated; but they are not independent of each other; they are only different aspects of the same thing, they are terms of relation, not of reality. Conditions of existence are not of a mutually exclusive character; in essence things are not two but one.

- Lankavatara Sutra

To my parents.

Acknowledgments

This work would never have materialized without the help of many people to whom I have the pleasure of expressing my appreciation and gratitude.

My deepest gratefulness is due to my supervisor, Shin Ishii, who accepted a student who came back from the United States with the intention to do research on concept formation from the viewpoint of optimality. Despite holding only a Bachelor's degree in psychology and cognitive science, he allowed me to dive into the field of machine learning. It was more than encouraging. Also, without his trust in my own judgment to study in Okinawa under the supervision of Kenji Doya, none of the rich experiences in this small island would have been possible.

I would like to extend my deepest gratitude to my co-supervisor, Kenji Doya, who gave me the opportunity to study in his lab in the early days of Okinawa Institute of Science and Technology (OIST). Through the countless activities in his lab, I received a lot of invaluable experience. From his lifestyle, I learned not only about science, but also about the importance of balance in life.

I would like to express my appreciation to Junichiro Yoshimoto and Eiji Uchibe for providing me guidance, suggestions and frequent discussions during my study. Their continuous one-to-one support allows me to break the finishing tape of this long distance race. I also want to offer my gratitude to the members of my thesis committee, Tomohiro Shibata and Kenji Sugimoto.

I extend my thanks to all members of Neural Computation unit. Especially, I appreciate Emiko Asato and Chikako Uehara for supporting my research in every aspect from business trips to heavily consumed hot coffee. Thanks to Tomofumi Inoue and Yasuhiro Inamine for setting up my computers and helping me out of numerous machine troubles. I deeply appreciate Takashi Nakano for being a great friend of mine. His ingenuous advice prevented me from getting sidetracked and helped me out of trouble numerous times. Thanks to Makoto Ito for showing a great example of enjoying each moment and doing the best in every aspect of life. Thanks to Stefan Elfwing for sharing his ideas in front of the whiteboard and implementing the proposed algorithms on the Cyber Rodent. Thanks to Viktor Zhumatiy for checking the final version of my thesis. Thanks to Kayoko Miyazaki, Katsuhiko Miyazaki, Masato Hoshino, Shinji Kimura, Aurelien Cassagnes, Alan Rodrigues and Mayumi Higa for being such kind and charming people. Their kindness has always been a strong support for me. Thanks to Tetsuro Morimura and Takuming Kamioka for taking the unpaved road of writing Ph.D. thesis before me. I would like to express my gratefulness to my favorite visitors to Neural Computation Unit, Medhi Khamassi, Akihiro Funamizu, Thomas Trappenberg, Byron Yu, and Thomas Strösslin.

I would like to further extend my thanks to all of the friends in OIST. Thanks to my diving, camping, drinking, and thesis-checking buddies, Tomas Pluskal, Cathy Vickers, Ben Torben-Nielsen, Gunner Wilken, and Klaus M. Stiefel, to name but a few. I would like to say thanks to all of my friends I met in Nara Institute of Science and Techonology (NAIST). Especially, thanks to Mizuki Ihara, Satoshi Morimoto, Melva Vidal (Jessica), Eri Saito, Atsunori Kanemura, Shinichi Maeda, Shigeyuki Oba, Wako Yoshida, Tsuyoshi Ueno, Yoko Kikuchi, Sachiko Yamazaki, Ritsuko Harima, Yasutaka Kamei, and Osamu Matsumoto for assisting me in many different aspects. Without their help, I would probably have gotten lost somewhere in the middle of this long-lasting journey.

I would also like to say thanks to my friends in Japan, especially in Okinawa. First of all, I would like to express my deepest gratitude to Shinnosuke Imazu. I feel really lucky to have met him. Thanks to Kishin Sakiyama, Shigeru Ahagon, and Tatsuya Morinaga at Kotorino for giving me a chance to touch the soil. I would like to thank all members of Okinawa Sudbury school: especially, Ayumi Tahara, Junko Hayashi, Yoshio and Mika Munakata, Midori Potter, and Simon Robinson. I am grateful to the following friends who gave me countless support and love: Akira and Nika Hidaka, Haruki Nakamura, Mihiro Matsuda, Yukikazu Hosokawa, Motoki Sueyoshi, Isamu Jahana, Taeko Okamura, Naomi Haraki, Yuko Ichimura, Yukari Takaya, Satoshi Oyama, Eri Kawano, Yoshimi Kurosawa, Hiroshi Ueda, and Yukikazu Hosokawa, to name but a few. Thanks to Makoto Ando for showing me a way to listen to the voice of nature inside and outside of us.

Much love and appreciation must go to my precious friends in Los Angeles. Thanks to Dr. Greg Stevens in UCLA for listening to my wild idea of concept formation as a creation of unit vector even before I knew the field of linear algebra. His supportive attitude really helped me, and I think my wild idea has somehow materialized in this thesis after a decade. Thanks to Dr. Mario Mata for paying attention to me when I lost my direction and confidence in the middle of nowhere. Thanks to Dr. Shinsuke Shimojo for accepting my visit to his office and giving me advice and encouragement. I deeply appreciate the love and kindness of Steve Chong So Lee, Penny Rosman, Manuel and Itzel Bejar, Daniel Bernstein, Irwan Yunus, Chumba Sichale, Tunde Johnson, Junko and Ronald Vigo, Carey Chan, Kaori Shimazaki, Ray Sato, Karen Fang, Rainbow Kan and Mini Kurian. Thanks to Akihiro Mori, Kanako Tsumagari, Yuri Kawano, Jun Kozuma, Koki Yamashita, Mayumi Oka, Wakana Hayashi, and Tomokazu Yokoyama for being such great friends for more than a decade.

Thanks to my first students, Nana and Aya, and their charming mother, Mia Kamimura. Thanks to many of my students and coworkers in Asahi Gakuen, who kindly allowed me to study in Japan. Especially thanks to Saburo Matsumoto who made a song for me, and Sakaguchi Yoichi who believed in me and gave me confidence to explore a new world over the horizon.

Finally, and most importantly, I would like to thank my parents–Yoshinobu and Emiko, my sisters–Michiko and Yoko, my oldest friend–Kyota Mikami, and my partner–Ayaka Mori for their faith in me, their love and support. Thanks from the bottom of my heart.

Goal-Oriented Representations of the External World: A Free-Energy-Based Approach[*]

Makoto Otsuka

Abstract

The central objective of living organisms or intelligent systems is not to model the world as accurately as possible but to act in their environments to achieve some goals. However, efficient decision making in the real world requires successful encoding of noisy, high-dimensional sensory inputs and representation of the implicit constraints in the environmental dynamics. In this dissertation, we explore methods for goal-directed sensory representations and decision making in partially observable Markov decision processes (POMDPs) with high-dimensional, noisy, multimodal sensory inputs and unknown dynamics.

First, we investigated whether free-energy-based reinforcement learning (FERL), which is known to handle Markov decision processes (MDPs) with high-dimensional states and actions, can handle an easy class of POMDPs, in which the detection of the true state behind the noisy, high-dimensional observation reduces the problem to MDPs. Using a novel "digit-floor" task, we found the reward- and action-dependent sensory coding in the distributed activation patterns of hidden units despite large variations in the sensory observations for the hidden state. Second, the FERL was combined with recurrent neural networks to handle POMDP problems that require a dynamic combination of sensory inputs. Using both low-dimensional bit patterns and high-dimensional binary images, we verified that the dynamic task-structure was implicitly reflected in the time-varying hidden unit activations.

[*]Doctoral Dissertation, Department of Bioinformatics and Genomics, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0761010, February 20, 2010.

These results show that our dynamic extension of FERL can construct a distributed representation of the external world autonomously while solving realistic sequential decision making problems. This approach, which is compatible with Friston's free-energy principle, provides a basis for biologically plausible models of representation learning in the brain.

Keywords:

Reinforcement learning, partially observable Markov decision process, goal-oriented representation, population coding, restricted Boltzmann machine, free energy

Contents

List of Figures

List of Tables

Symbols and Notation

Symbol	Meaning
$\triangleq$	left-hand side is defined by the right-hand side
t	discrete time, $t \in \{0, 1, 2, \dots\}$
z_t	time-dependent scalar
$\boldsymbol{z}_t,\ \boldsymbol{Z}_t$	time-dependent vector or matrix, respectively
$z_{i,t},\ Z_{ij,t}$	element of a time-dependent vector or matrix, respectively
w_{ik}^{sh}	connection weight from the node H_k to the node S_i
$\sigma(x)$	logistic sigmoid function $\sigma(x) \triangleq 1/(1+\exp(-x))$
$\sigma_\beta(x)$	logistic sigmoid function with a gain modification $\sigma_\beta(x) \triangleq 1/(1+\exp(-\beta x))$
$\mathbb{E}[\cdot]$	expectation operator
$E(\cdot)$	energy function
$F(\cdot)$	equilibrium free energy with a unity temperature
$F_\beta(\cdot)$	equilibrium free-energy with an inverse temperature β
$F(q, \cdot)$	variational free energy with a unity temperature and the trial distribution q
$F_\beta(q, \cdot)$	variational free energy with an inverse temperature β and the trial distribution q

Chapter 1

Introduction

1.1 Motivations and rationale

The central objective of living organisms is not to model the world as accurately as possible but to act in their environment to achieve some goal. All behaviors of living animals are goal-directed, and their goals are implicitly linked with reward or punishment signals. Some organisms use reflexes to perform stereotypical actions to avoid immediate danger, and others use higher cognitive capabilities to sequentially select actions to achieve long-term goals.

If an environment remains static over many generations and if animals can detect the critical states for optimal decision making using their sensory system, they can use genetically programmed, hardwired reflexes to achieve their goals. However, if task-relevant information is obscured in high-dimensional sensory inputs, agents need to extract the information first and build efficient spatial representations of the external world in order to achieve their goals. Even if the agent successfully extracts the task-relevant features from the current sensory inputs, the task might still be partially observable with the full set of extracted features. Then, the agents cannot solely rely on the current sensory inputs. They need to have a memory of past inputs as well. Here, the spatiotemporal representations of the environment now become relevant.

To perform tasks beyond reflexes, meaningful representations of the world become critical. Gradual changes in the neural response reflecting task demand have been observed in many regions in the cortex from the cellular (Freedman et al., 2001; Freedman and Assad, 2006; Froemke et al., 2007) to the network levels (Sigala and Logothetis, 2002; Jog et al., 1999; Hyman et al., 2006). Beyond the primary cortex, the brain appears to extract goal-directed representations automatically according to the task requirement. As these examples suggest, goal-directed representations lay the basis for the optimal decision making in a noisy and uncertain environment.

It is intriguing to find changing representations in the real brain while an animal's performance improves via trial and error. Unfortunately, beyond the cellular level, it is difficult for experimentalists to find changing representations in the real brain while the performance improves in a reinforcement learning task. There have been some attempts using a multichannel microelectrode array (MEA) in in vitro network (Dockendorf et al., 2009); however some experimental breakthrough is required to reveal the changing neural representation during the performance improvement of learning animals.

Instead, we employed a different approach. We first formulated the computational problem that animals solve as a mathematical optimization problem called a partially observable Markov decision process (POMDP). Then, we proposed a minimal architecture and learning rule that can solve the given problem in a biologically plausible manner. Finally, we investigated how the representation is modified and maintained in the model while the artificial agent's performance improves. This approach may provide an answer to the long-lasting epistemological question: "How do we represent the world?" (Churchland and Sejnowski, 1990).

This dissertation has two objectives. The first objective is to elucidate a biologically plausible minimal architecture that can efficiently solve POMDP problems. The second objective is to investigate how a goal-directed representation is dynamically modified and maintained in the model of an agent.

1.2 Representation learning as a scientific problem

Kawato's definition of computational neuroscience is (Kawato, 2008)

"to elucidate information processing of the brain to the extent that artificial machines, either computer programs or robots, can be built to solve the same computational problems that are solved by the human brain, using essentially the same principles."

We believe that the computational problems that the mammalian brain has evolved to solve can be captured by the POMDP with high-dimensional, noisy, multimodal sensory inputs without any prior knowledge of the environment. Our focus is a mammalian brain, which is usually equipped with general-purpose sensory systems such as a visual system.

Animals do not need to know the absolute coordinates in the universe to behave optimally. Instead, they require relative coordinates of the external world within themselves; this is sufficient to help them achieve their goals. Then, the following question arises: how should the brain create a relative coordinate system within itself? The guiding principle of representation learning is succinctly and beautifully captured by Friston's free-energy principle (Friston, 2009; Friston and Kiebel, 2009) that simply says, "any adaptive change in the brain will minimize free-energy."

Our approach is compatible with the free-energy principle. The main differences lies in the type of probabilistic model. Our approach uses an undirected graphical model (random Markov field), whereas Friston and the other (George and Hawkins, 2009) use a directed graphical model (Bayesian network). In addition, Friston's formulation of reinforcement learning as a minimization of observation prediction error (Friston et al., 2009) appears to be a supervised learning, not a real reinforcement learning, since an agent need to see the optimal trajectories before the free-energy-based action selection becomes possible. On the other hand, our approach handles real reinforcement learning problems using the free energy of a stochastic network.

1.3 Representation learning as an engineering problem

To learn how to behave in unknown environments, a reinforcement learning framework can be employed to obtain the optimal behavioral policy. For tasks that involve continuous sensory data, linear function approximators such as CMAC and RBF networks with fixed basis functions (Sutton and Barto, 1998) are commonly used because they theoretically guarantee an approximation of the true value function with an arbitrary precision if an appropriate set of basis functions are selected for a particular task. However, the selection of basis functions is task-dependent and needs to be done manually based on prior knowledge about the task domain. In addition, the linear function approximators invariably face the curse of dimensionality (Bellman, 1957) when the dimensionality of data increases.

In order to reduce the burden of feature designing, several methods of automatic feature extraction have been proposed in the field of reinforcement learning. These methods are roughly divided into two groups according to whether rewards affect the acquired basis functions. The first type constructs basis functions that capture statistical regularities in the sensory data (Mahadevan, 2005). The second type adjusts the basis functions depending on the acquired rewards (Keller et al., 2006; Santamaria et al., 1998; Sprague, 2007).

As an engineering problem, we focus on the problem of designing a state space $\mathcal{S}$ for reinforcement learning agents in a goal-directed fashion.

1.4 Contributions of this dissertation

The dissertation makes the following contributions:

- an analysis of reward- and action-dependent coding of sensory inputs in the hidden layer of a free-energy-based reinforcement learning agent.
- an analysis of noise tolerance of a free-energy-based-reinforcement learning (FERL) framework.
- a new task called the digit-floor task that is specifically designed to analyze the coding of true states embedded in the high-dimensional sensory inputs.
- a predictor-actor architecture that can solve a POMDP problem with high-dimensional noisy input without any prior knowledge of environmental dynamics and hidden states.

- a (digit) matching T-maze task that requires not only memory but also sensory inputs to act optimally. With some constraints, the task can be interpreted as a delayed-matching task that is commonly used in monkey experiments, and the predictor-actor model can be used for model-based analysis of neural data.

1.5 Contents of this dissertation

This dissertation is organized as follows. Chapter 2 provides a review of relevant background about energy-based models, reinforcement learning, POMDPs, and recurrent neural networks. In Chapter 3, we investigate the emergence of reward- and action-dependent sensory coding in FERL using a digit-floor task, which falls into a class of easy POMDPs. In Chapter 4, FERL is combined with RNNs to handle moderately difficult POMDPs without any prior knowledge of environmental dynamics and hidden states. In Chapter 5, the proposed architecture is improved to handle difficult POMDPs with noisy, high-dimensional inputs without the assumption of any prior knowledge of the environment. In Chapter 6, we conclude with an overall summary.

Chapter 2

Backgrounds

2.1 Energy-based models

Energy-based models (a.k.a., undirected graphical models, Markov random fields) are the most important tool in this dissertation. They provide a principled method to unify information theory, machine learning, and reinforcement learning.

2.1.1 Energy, probability, and information

An energy is an arbitrary function defined over the configuration of a stochastic system $\boldsymbol{\xi}$. Let us assume a system composed of N binary stochastic units $\boldsymbol{\xi} \in \Xi \triangleq \{0,1\}^N$. Let us further assume that the energy is given by an arbitrary function $E(\boldsymbol{\xi};\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents the parameters of the energy function.

Once the energy function and an inverse temperature, which controls an overall stochasticity of a system, are given for a certain stochastic system, we can obtain the probability distribution of the system being in a certain configuration in its thermal equilibrium $p_\beta(\boldsymbol{\xi};\boldsymbol{\theta})$. Note that the parameter $\boldsymbol{\theta}$ of the equilibrium distribution is the parameter of the energy function. This conversion of energy into probability involves two steps: exponential transformation and normalization. In the first step, energy is transformed into an unnormalized probability (ocasionally called intensity) by the exponential transformation

$$\tilde{p}_\beta(\boldsymbol{\xi};\boldsymbol{\theta}) \triangleq \exp\{-\beta E(\boldsymbol{\xi};\boldsymbol{\theta})\} \ ,$$

where β is the inverse temperature of the system. In the second step, the unnormalized probability $\tilde{p}_\beta(\boldsymbol{\xi}; \boldsymbol{\theta})$ is transformed into a probability $p_\beta(\boldsymbol{\xi}; \boldsymbol{\theta})$ by normalization

$$p_\beta(\boldsymbol{\xi}; \boldsymbol{\theta}) = \frac{\tilde{p}_\beta(\boldsymbol{\xi}; \boldsymbol{\theta})}{\sum_{\boldsymbol{\xi}'} \tilde{p}_\beta(\boldsymbol{\xi}'; \boldsymbol{\theta})} \, . \tag{2.1}$$

This probability is called the equilibrium distribution (also called the Gibbs distribution or Boltzmann distribution) of the stochastic system defined by the energy function $E(\boldsymbol{\xi}; \boldsymbol{\theta})$. Note that the parameters of the distribution are equivalent to those of the energy function. This probability distribution is called the *equilibrium* distribution because the stochastic system realizes a configuration $\boldsymbol{\xi}$ with probability $p_\beta(\boldsymbol{\xi}; \boldsymbol{\theta})$ after the system with an inverse temperature β reaches thermal equilibrium.

The summation appearing in the denominator implies that the sum is over all possible configurations of $\boldsymbol{\xi}' \in \Xi$. This denominator is called the partition function (or normalization constant) and is often expressed as $Z_\beta(\boldsymbol{\theta}) \triangleq \sum_{\boldsymbol{\xi}'} \tilde{p}_\beta(\boldsymbol{\xi}'; \boldsymbol{\theta})$. The computation of the partition function is often prohibitive due to the exponentially growing numbers of possible configurations as the number of nodes increases. For example, even in the simplest system with N binary nodes, which is our current focus, the computation of the partition function requires the summation of 2^N terms. Due to this partition function, we can rarely evaluate the probability in Eq. (2.1).

We can often draw samples from this probability distribution $p_\beta(\boldsymbol{\xi}; \boldsymbol{\theta})$ regardless of the computational complexity of calculating the probability. The single requirement for sampling, which is usually satisfied, is that the unnormalized probability $\tilde{p}_\beta(\boldsymbol{\xi}; \boldsymbol{\theta})$, or equivalently the energy function $E(\boldsymbol{\xi}; \boldsymbol{\theta})$, can readily be evaluated. Many sampling methods such as Gibb's sampling can be used to obtain samples from the probability distribution.

Thus far, we have established the relationship between the energy and the probability via an unnormalized probability. We can establish another relationship between the energy and the probability via a quantity called *information*. Using the notion of information measure first introduced by (Shannon, 1948), the information content (or surprise) of observing the configuration $\boldsymbol{\xi}$ in the stochastic system is defined as

$$I(\boldsymbol{\xi};\boldsymbol{\theta}) \triangleq -\ln p_\beta(\boldsymbol{\xi};\boldsymbol{\theta}) \ , \tag{2.2}$$

where the natural logarithm is used as a base. Using Eq. (2.1), the information content of $\boldsymbol{\xi}$ can be written as

$$I(\boldsymbol{\xi};\boldsymbol{\theta}) = -\ln \tilde{p}_\beta(\boldsymbol{\xi};\boldsymbol{\theta}) - \{-\ln Z_\beta(\boldsymbol{\theta})\} \ . \tag{2.3}$$

Dividing both side of Eq. (2.3) by the inverse temperature β yields

$$\frac{1}{\beta} I(\boldsymbol{\xi};\boldsymbol{\theta}) = \underbrace{-\frac{1}{\beta}\ln \tilde{p}_\beta(\boldsymbol{\xi};\boldsymbol{\theta})}_{E(\boldsymbol{\xi};\boldsymbol{\theta})} - \underbrace{\left\{-\frac{1}{\beta}\ln Z_\beta(\boldsymbol{\theta})\right\}}_{F_\beta(\boldsymbol{\theta})} \ . \tag{2.4}$$

The first term in the right hand side of Eq. (2.4) is the energy of configuration $\boldsymbol{\xi}$. The second term is interpreted as an important quantity called the (equilibrium) free energy of the stochastic system:

$$F_\beta(\boldsymbol{\theta}) \triangleq -\frac{1}{\beta}\ln Z_\beta(\boldsymbol{\theta}) \ . \tag{2.5}$$

We can also say that we obtained the relationship in Eq. (2.4) by applying the inverse of the exponential transformation[1] to Eq. (2.1). This implies that, by the exponential transformation, energy is mapped to the unnormalized probability and information is mapped to the probability. Eq. (2.4) provides a further insight in that the free energy is used to "normalize" the energy. After subtracting the free energy $F(\boldsymbol{\theta})$ from the energy $E(\boldsymbol{\xi};\boldsymbol{\theta})$, 0 becomes meaningful in the energy- or log-domain. In this context, the information $I(\boldsymbol{\xi};\boldsymbol{\theta})$ can be interpreted as a temperature-dependent *normalized energy*:

$$I(\boldsymbol{\xi};\boldsymbol{\theta}) = \beta\{E(\boldsymbol{\xi};\boldsymbol{\theta}) - F_\beta(\boldsymbol{\theta})\} \ .$$

[1]When the exponential transformation f is defined as $f(\cdot) = \exp\{-\beta\ \cdot\}$, the inverse of the exponential transformation is given by $f^{-1}(\cdot) = -\frac{1}{\beta}\ln(\cdot)$

2.1.2 Variational and equilibrium free energies

The entropy (or expected information) can be expressed as

$$\begin{aligned}
H[p_\beta;\boldsymbol{\theta}] &\triangleq -\sum_{\boldsymbol{\xi}} p_\beta(\boldsymbol{\xi};\boldsymbol{\theta}) \ln p_\beta(\boldsymbol{\xi};\boldsymbol{\theta}) && \text{(2.6a)}\\
&= \sum_{\boldsymbol{\xi}} p_\beta(\boldsymbol{\xi};\boldsymbol{\theta}) I(\boldsymbol{\xi};\boldsymbol{\theta}) && \text{(2.6b)}\\
&= \beta \sum_{\boldsymbol{\xi}} p_\beta(\boldsymbol{\xi};\boldsymbol{\theta}) \left\{E(\boldsymbol{\xi};\boldsymbol{\theta}) - F_\beta(\boldsymbol{\theta})\right\} && \text{(2.6c)}\\
&= \beta \underbrace{\sum_{\boldsymbol{\xi}} p_\beta(\boldsymbol{\xi};\boldsymbol{\theta}) E(\boldsymbol{\xi};\boldsymbol{\theta})}_{\langle E(\boldsymbol{\theta})\rangle_{p_\beta}} - \beta F_\beta(\boldsymbol{\theta}) && \text{(2.6d)}
\end{aligned}$$

where $\langle E(\boldsymbol{\theta})\rangle_{p_\beta}$ is called the expected energy (or internal potential) of the stochastic system. Reorganizing the terms in Eq. (2.6d) provides us with a new interpretation of equilibrium free energy: expected energy minus temperature-multiplied entropy,

$$\begin{aligned}
F_\beta(\boldsymbol{\theta}) &= \langle E(\boldsymbol{\theta})\rangle_{p_\beta} - \frac{1}{\beta} H[p_\beta;\boldsymbol{\theta}] && \text{(2.7)}\\
&= \sum_{\boldsymbol{\xi}} p_\beta(\boldsymbol{\xi};\boldsymbol{\theta}) E(\boldsymbol{\xi};\boldsymbol{\theta}) + \frac{1}{\beta} \sum_{\boldsymbol{\xi}} p_\beta(\boldsymbol{\xi};\boldsymbol{\theta}) \ln p_\beta(\boldsymbol{\xi};\boldsymbol{\theta}) \; . && \text{(2.8)}
\end{aligned}$$

Using an arbitrary variational distribution $q(\boldsymbol{\xi})$ instead of the equilibrium distribution $p_\beta(\boldsymbol{\xi};\boldsymbol{\theta})$ in (2.8) leads to a *variational* free energy:

$$\begin{aligned}
F_\beta(q,\boldsymbol{\theta}) &\triangleq \langle E(\boldsymbol{\theta})\rangle_q - \frac{1}{\beta} H[q] && \text{(2.9)}\\
&= \sum_{\boldsymbol{\xi}} q(\boldsymbol{\xi}) E(\boldsymbol{\xi};\boldsymbol{\theta}) + \frac{1}{\beta} \sum_{\boldsymbol{\xi}} q(\boldsymbol{\xi}) \ln q(\boldsymbol{\xi}) \; . && \text{(2.10)}
\end{aligned}$$

The lower bound of the variational free energy $F_\beta(q,\boldsymbol{\theta})$ is given by the equilibrium free energy $F_\beta(\boldsymbol{\theta})$, and this lower bound is achieved when the variational distribution coincides with the Boltzmann distribution :

$$F_\beta(\boldsymbol{\theta}) = \arg\min_q F_\beta(q,\boldsymbol{\theta}) \qquad \Leftrightarrow \qquad q(\boldsymbol{\xi}) = p_\beta(\boldsymbol{\xi};\boldsymbol{\theta}) \tag{2.11}$$

Changing the energy parameter $\boldsymbol{\theta}$ to decrease the equilibrium free energy has interesting consequences. This is achieved by decreasing the expected energy (first term in Eq. (2.8)) or increasing the entropy (second term in Eq. (2.8)). In short, changing $\boldsymbol{\theta}$ so as to lower the equilibrium free energy ensures the frequent realization of lower energy configurations (a constraint imposed by the expected energy term) while encouraging equal realization of all possible configurations (a constraint imposed by the entropy term). A self-organizing property arises as a consequence of this free energy minimization. The balance between these two constraints is controlled by the inverse temperature β.

2.1.3 Free energy and EM algorithm

Suppose that we can divide binary stochastic units $\boldsymbol{\xi}$ in the system into two groups: visible units $\boldsymbol{v}$ and hidden (latent) units $\boldsymbol{h}$. Then, the joint probability distribution can be written as $p_\beta(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is still a parameter of an energy function. Using this expression, the log-likelihood is written as a combination of three terms:

$$\ln p_\beta(\boldsymbol{v}; \boldsymbol{\theta}) = Q[q, p_{\text{joint}}] + H[q] + D_{\text{KL}}[q, p_{\text{post}}] \ , \tag{2.12}$$

where the first term is the expected complete data log-likelihood

$$Q[q, p_{\text{joint}}] \triangleq \sum_{\boldsymbol{h}} q(\boldsymbol{h}) \ln p_\beta(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) \ , \tag{2.13}$$

the second term is the entropy of the variational distribution over hidden nodes $\boldsymbol{h}$

$$H[q] \triangleq \sum_{\boldsymbol{h}} q(\boldsymbol{h}) \ln q(\boldsymbol{h}) \ , \tag{2.14}$$

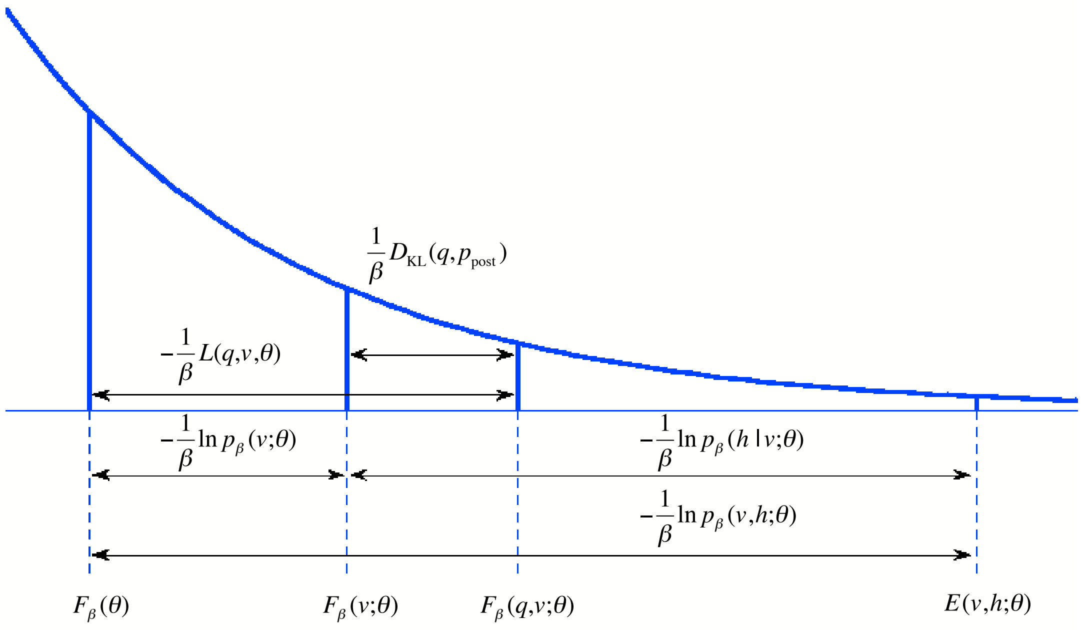
and the third term is the Kullback-Leibler (KL) divergence

$$D_{\text{KL}}[q, p_{\text{post}}] \triangleq \sum_{\boldsymbol{h}} q(\boldsymbol{h}) \ln \frac{q(\boldsymbol{h})}{p_\beta(\boldsymbol{h}|\boldsymbol{v}; \boldsymbol{\theta})} \ . \tag{2.15}$$

Eq. (2.12) can be further modified by combining the first two terms

$$\ln p_\beta(\boldsymbol{v}; \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{v}, \boldsymbol{\theta}) + D_{\text{KL}}[q, p_{\text{post}}] \ , \tag{2.16}$$

where $\mathcal{L}(q, \boldsymbol{v}, \boldsymbol{\theta})$ is called the lower-bound of the log-likelihood because the KL divergence always takes a non-negative value. A graphical interpretation of Eq. (2.16) is shown in Fig. 2.1.



$\frac{1}{\beta}D_{KL}(q,p_{post})$
$-\frac{1}{\beta}L(q,v,\theta)$
$-\frac{1}{\beta}\ln p_\beta(v;\theta)$
$-\frac{1}{\beta}\ln p_\beta(h|v;\theta)$
$-\frac{1}{\beta}\ln p_\beta(v,h;\theta)$
$F_\beta(\theta)$
$F_\beta(v;\theta)$
$F_\beta(q,v;\theta)$
$E(v,h;\theta)$

Figure 2.1. Relationship between an energy domain (horizontal axis) and an unnormalized-probability domain (vertical axis). The exponential curve represents $y = \exp(-\beta x)$, where x is some quantity in the energy domain. All quantities associated with double-headed arrows are some types of information (or surprise).

The parameter $\boldsymbol{\theta}$ cannot be changed directly to raise the log-likelihood due to the expectation in Eq. (2.13). Instead, an expectation-maximization (EM) algorithm raises its lower-bound in an iterative fashion. In the E-step, the variational distribution $q(\boldsymbol{h})$ is set to the posterior distribution $p_\beta(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\theta})$ to make the KL divergence between these two distributions $D_{\mathrm{KL}}[q, p_{\mathrm{post}}]$ vanish. In the M-step, the lower-bound $\mathcal{L}(q, \boldsymbol{v}, \boldsymbol{\theta})$, or equivalently the complete data log-likelihood $Q[q, p_{\mathrm{joint}}]$, is increased by changing the parameter $\boldsymbol{\theta}$. These two steps are repeated until convergence.

2.1.4 Boltzmann machine

Thus far, we have not defined the specific form for the parameters of the energy function. Let us assume that two binary visible nodes, V_i and V_j, in a visible layer $\boldsymbol{V} = \{V_1, \dots, V_N\}$ are connected by the symmetric weight $w^{vv}_{ij} \equiv [\boldsymbol{W}^{vv}]_{ij}$. In the same manner, two binary hidden nodes, H_k and H_l, in a hidden layer $\boldsymbol{H} = \{H_1, \dots, H_K\}$ are connected by a symmetric weight $w^{hh}_{kl} \equiv [\boldsymbol{W}^{hh}]_{kl}$. In addition, a visible node V_i is connected with a hidden node H_j by a symmetric weight $w^{vh}_{ij} \equiv [\boldsymbol{W}^{vh}]_{ij}$. The bias terms for the visible node V_i and hidden node H_j are $b^v_i \equiv [\boldsymbol{b}^v]_i$ and $b^h_j \equiv [\boldsymbol{b}^h]_j$, respectively. These parameters are collectively represented as $\boldsymbol{\theta} \triangleq \{\boldsymbol{W}^{vv}, \boldsymbol{W}^{vh}, \boldsymbol{W}^{hh}, \boldsymbol{b}^v, \boldsymbol{b}^h\}$. This parameterization creates a stochastic model called the Boltzmann machine. The stochastic behavior of the Boltzmann machine is characterized by the energy of the network:

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) \triangleq -\frac{1}{2}\boldsymbol{v}^\top \boldsymbol{W}^{vv} \boldsymbol{v} - \boldsymbol{v}^\top \boldsymbol{W}^{vh} \boldsymbol{h} - \frac{1}{2}\boldsymbol{h}^\top \boldsymbol{W}^{hh} \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{b}^v - \boldsymbol{h}^\top \boldsymbol{b}^h \ . \tag{2.17}$$

For learning in the Boltzmann machine, the EM algorithm described in Sect. 2.1.3 can be used. However, the exact evaluation of a posterior $p_\beta(\boldsymbol{h}|\boldsymbol{v}; \boldsymbol{\theta})$ is often computationally infeasible due to the mutual dependence between hidden nodes.

2.1.5 Restricted Boltzmann machine

A restricted Boltzmann machine (RBM) is a special type of Boltzmann machine with a restricted connectivity. The special feature of the RBM is the absence of intra-layer connections between nodes within the same layer. In short, $\boldsymbol{W}^{vv} = \boldsymbol{0}$ and $\boldsymbol{W}^{hh} = \boldsymbol{0}$. The energy of a particular configuration $(\boldsymbol{V} = \boldsymbol{v}, \boldsymbol{H} = \boldsymbol{h})$ in the RBM is defined by

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = -\boldsymbol{v}^\top \boldsymbol{W}^{vh} \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{b}^v - \boldsymbol{h}^\top \boldsymbol{b}^h \tag{2.18}$$

This restricted connectivity ensures that all hidden units are statistically decoupled when visible units are clamped to the observed values:

$$p_\beta(\boldsymbol{h}|\boldsymbol{v}; \boldsymbol{\theta}) = \prod_{k=1}^{K} p_\beta(h_k|\boldsymbol{v}; \boldsymbol{\theta}) \ . \tag{2.19}$$

This special feature of RBMs allows an easy computation of the expectation over the posterior distribution:

$$\hat{h}^\beta_k \equiv \langle h_k \rangle^\beta = \sum_{h_k \in \{0,1\}} h_k p_\beta(h_k|\boldsymbol{v}; \boldsymbol{\theta}) = \frac{1}{1 + \exp(-\beta\{\sum_i w^{vh}_{ik} v_i + b^h_k\})} \ .$$

Consequently, the evaluation of two important quantities becomes easy. One is the free energy of the network:

$$
\begin{aligned}
F_\beta(\boldsymbol{v},\boldsymbol{\theta}) &= \sum_{\boldsymbol{h}} p_\beta(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\theta}) E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}) + \frac{1}{\beta}\sum_{\boldsymbol{h}} p_\beta(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\theta}) \ln p_\beta(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\theta}) && (2.20)\\
&= -\sum_{i=1}^{N^v}\sum_{k=1}^{N^h} v_i W_{ik}^{vh}\hat{h}_k^\beta - \sum_{i=1}^{N^v} v_i b_i^v - \sum_{k=1}^{N^h}\hat{h}_k^\beta b_k^h && (2.21)\\
&\quad + \frac{1}{\beta}\left\{\sum_{k=1}^{N^h}\hat{h}_k^\beta \ln \hat{h}_k^\beta + \sum_{k=1}^{N^h}(1-\hat{h}_k^\beta)\ln(1-\hat{h}_k^\beta)\right\}\ , && (2.22)
\end{aligned}
$$

and the other is its derivatives:

$$
\begin{aligned}
\frac{\partial F_\beta(\boldsymbol{v},\boldsymbol{\theta})}{\partial w_{ik}} &= -v_i\hat{h}_k^\beta && (2.23)\\
\frac{\partial F_\beta(\boldsymbol{v},\boldsymbol{\theta})}{\partial b_i^v} &= -v_i && (2.24)\\
\frac{\partial F_\beta(\boldsymbol{v},\boldsymbol{\theta})}{\partial b_k^h} &= -\hat{h}_k^\beta\ . && (2.25)
\end{aligned}
$$

In a usual unsupervised setting, the focus is not on the above two quantities but on the log-likelihood of observed data $\mathcal{D}^N \triangleq \{\boldsymbol{v}_n\}_{n=1}^N$:

$$
\begin{aligned}
\ln p_\beta(\mathcal{D}^N;\boldsymbol{\theta}) &= \sum_{n=1}^{N}\ln p_\beta(\boldsymbol{v}_n;\boldsymbol{\theta}) && (2.26)\\
&= -\sum_{n=1}^{N}\beta\left\{F_\beta(\boldsymbol{v}_n,\boldsymbol{\theta}) - F_\beta(\boldsymbol{\theta})\right\}\ . && (2.27)
\end{aligned}
$$

Its derivative is given by

$$\begin{aligned}
\frac{\partial}{\partial w_{ik}} \ln p_\beta(\mathcal{D}^N; \boldsymbol{\theta}) &= \sum_{n=1}^{N} \frac{\partial}{\partial w_{ik}} \ln p_\beta(\boldsymbol{v}_n; \boldsymbol{\theta}) && (2.28)\\
&= -\sum_{n=1}^{N} \beta \left\{ \frac{\partial F_\beta(\boldsymbol{v}_n, \boldsymbol{\theta})}{\partial w_{ik}} - \frac{\partial F_\beta(\boldsymbol{\theta})}{\partial w_{ik}} \right\} && (2.29)\\
&= -\sum_{n=1}^{N} \beta \left\{ -v_{n,i} \langle h_{n,k} \rangle^\beta + \langle v_i h_k \rangle^\beta \right\} && (2.30)\\
&= \beta \left\{ \underbrace{\sum_{n=1}^{N} v_{n,i} \langle h_{n,k} \rangle^\beta}_{\langle v_i h_k \rangle^\beta_{\text{data}}} - \underbrace{\sum_{n=1}^{N} \langle v_i h_k \rangle^\beta}_{\langle v_i h_k \rangle^\beta_{\text{model}}} \right\} , && (2.31)
\end{aligned}$$

where

$$\langle v_i h_k \rangle^\beta_{\text{model}} = \sum_{v_i \in \{0,1\}} \sum_{h_k \in \{0,1\}} p_\beta(v_i, h_k; \boldsymbol{\theta}) v_i h_k \ .$$

Unfortunately, $\langle v_i h_k \rangle^\beta_{\text{model}}$ is difficult to compute due to the complex dependencies between all random variables in the system. The (alternating) Gibbs sampling can be used to sample from the joint probability distribution $p_\beta(v_i, h_k; \boldsymbol{\theta})$ and calculate this expectation. However, we have to run (alternating) Gibbs sampling an infinite number of times in order to calculate the true expectation.

To overcome this difficulty, Hinton (2002) proposed the Contrastive Divergence (CD) method and proved that it works well in practice. In the original CD method, called 1-step CD or CD-1, the model distribution is simply replaced by the reconstruction distribution.

$$\frac{\partial}{\partial w_{ik}} \ln p_\beta(\mathcal{D}^N; \boldsymbol{\theta}) \approx \beta \left\{ \langle v_i h_k \rangle^\beta_{\text{data}} - \langle v_i h_k \rangle^\beta_{\text{recon}} \right\} \tag{2.32}$$

The reconstruction is created by first stochastically updating all hidden nodes using a single observation $\boldsymbol{v}_n$. Then, the state of visible units is stochastically changed after clumping the hidden nodes to the updated value in the previous step. In this step, the RBM "reconstructs" the observation. Then, it updates the hidden variables stochastically using the "reconstructed" observations. If the update is terminated at this point, it is called CD-1. This layer-wise sampling procedure, called alternating Gibbs sampling, can be terminated after n-updates; this is called CD-n. It is known that CD-n ($n > 1$) works better than the original CD-1; however, it is computationally expensive due to the multiple steps involved in alternating Gibbs sampling. In addition, CD-1 does not estimate the true gradient of the log probability. Recently, Tieleman (2008) proposed a method that estimates the true gradient of the log probability. This method is called Persistent Contrastive Divergence (PCD).

Although the theoretical domain of the configuration in the visible layer is restricted to $\boldsymbol{V} \in \{0,1\}^N$, the restriction is relaxed to $\boldsymbol{V} \in [0,1]^N$ in many practical applications. In image processing, for example, the relaxation can be regarded as the mean-field approximation in the binarization process of gray-scale (or colored) pixels and it often makes the RBM work efficiently (Sutskever and Hinton, 2007). Because we employ a set of image data for our demonstration in the later section, we assume that this relaxation is acceptable in this dissertation. To preserve the theoretical validity of the RBM, it is possible to adopt Gaussian units as the input nodes (Welling et al., 2005); however, this type of the RBM is beyond the scope of this dissertation.

2.2 Reinforcement learning

Reinforcement learning is a subfield of machine learning that aims to learn the optimal behavioral policy autonomously using the sampled states, actions, and rewards collected stochastically through the interaction with the environment (Barto et al., 1983; Sutton and Barto, 1998). The properties of reinforcement learning are as follows:

- trial-and-error search
  - An agent does not know how the environment evolves (i.e., the agent possesses no prior knowledge of an environmental dynamics $\mathcal{P}^a_{ss'}$ and an expected reward function $\mathcal{R}^a_{ss'}$).
- delayed reward

	Observability of states $\mathcal{S}$	
	Fully observable	Partially observable
Non-interactable	Markov Chain (MC) $(\mathcal{S}, \mathcal{P}^0, \mathcal{P})$	Hidden Markov Model (HMM) $(\mathcal{S}, \mathcal{Z}, \mathcal{P}^0, \mathcal{P}, \mathcal{O})$
Interactable	Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{P}^0, \mathcal{P}, \mathcal{R})$	Partially Observable MDP (POMDP) $(\mathcal{S}, \mathcal{A}, \mathcal{Z}, \mathcal{P}^0, \mathcal{P}, \mathcal{R}, \mathcal{O})$

Table 2.1. Classification of four models in terms of observability and interactability.

– There exists a trade-off between an immediate and a distant reward.

The challenges of reinforcement learning are (1) a trade-off between exploration and exploitation and (2) setting or scheduling of meta-parameters such as the learning rate α, inverse temperature β, and discount factor γ. Reinforcement learning can be used to acquire the behavioral policy in an unknown environment (e.g., distant planet where the dynamics are unknown) where the correct action is difficult to know beforehand.

2.2.1 Markov decision processes

Definition Markov decision processes (MDPs) can be formally defined as a 5-tuple $\mathcal{M}_{\text{MDP}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}^0, \mathcal{P}, \mathcal{R})$ where

- $\mathcal{S}$ is the *state space.* $\mathcal{S} \triangleq \{1, 2, \ldots, |\mathcal{S}|\}$ is assumed to be a finite set of discrete states.
- $\mathcal{A}$ is the *action space.* $\mathcal{A} \triangleq \{1, 2, \ldots, |\mathcal{A}|\}$ is also assumed to be a finite set of discrete actions. If both the state space $\mathcal{S}$ and the action space $\mathcal{A}$ are finite and discrete, such an MDP is called a *finite* MDP.
- $\mathcal{P}^0 : \mathcal{S} \to [0, 1]$ is the *initial state distribution.* $\mathcal{P}^0_s$ denotes the probability of an episode starting from state s:

$$\mathcal{P}^0_s \triangleq p(s_0 = s) \ . \tag{2.33}$$

Because it is the probability distribution over the initial state, $\sum_{s\in\mathcal{S}}\mathcal{P}_s^0 = 1$ should be satisfied.

- $\mathcal{P}:\mathcal{S}\times\mathcal{A}\times\mathcal{S}\rightarrow[0,1]$ is the *state transition probability distribution.* $\mathcal{P}_{ss'}^a$ denotes the probability of reaching state s' given the condition of taking action a at state s:

$$\mathcal{P}_{ss'}^a \triangleq p(s_{t+1}=s'|s_t=s,a_t=a)\ . \tag{2.34}$$

  Because it is a probability distribution over the next state, $\sum_{s'\in\mathcal{S}}\mathcal{P}_{ss'}^a = 1$ should be satisfied.

- $\mathcal{R}:\mathcal{S}\times\mathcal{A}\times\mathcal{S}\rightarrow\mathbb{R}$ is the *expected reward function.* $\mathcal{R}_{ss'}^a$ denotes the expected reward obtained by reaching state s' after taking action a at state s:

$$\mathcal{R}_{ss'}^a \triangleq \mathbb{E}[r_{t+1}|s_t=s,a_t=a,s_{t+1}=s']\ . \tag{2.35}$$

  As a slight variant, the expected reward function can also be defined for a state-action pair (s,a):

$$\mathcal{R}_s^a \triangleq \mathbb{E}[r_{t+1}|s_t=s,a_t=a] \equiv \sum_{s'\in\mathcal{S}}\mathcal{P}_{ss'}^a\mathcal{R}_{ss'}^a\ . \tag{2.36}$$

The objective of MDPs is to find the behavioral policy π that maximizes a pre-specified scalar objective function. The policy π encodes how an agent selects an action at a certain state. The policy can be either stochastic or deterministic. A stationary stochastic policy $\pi(s,a)\triangleq p(a|s)$ is a map from the state to the distribution over all possible actions $\pi:\mathcal{S}\rightarrow\Omega(\mathcal{A})$, where $\Omega(\mathcal{A})$ is the set of all probability distributions over $\mathcal{A}$. A stationary deterministic policy $\pi(s)\triangleq\delta_{a'}(a)$ is a map from a state to an action $\pi:\mathcal{S}\rightarrow\mathcal{A}$, where $\delta_{a'}(a)$ is a Kronecker delta that takes a value 1 if $a=a'$ and 0 otherwise.

There are two commonly used objective functions for MDPs: expected discounted return and expected average reward. Depending on the objective function, there exist two formalisms. They are related in an intrinsic manner (Konda and Tsitsiklis, 2004).

Expected discounted return formalism: In this formalism, the expectation of the discounted return is used as an objective function. First, the *discounted return* or *discounted cumulative reward* from time t is defined as

$$R_t^\gamma \triangleq r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \ , \tag{2.37}$$

where γ is called a *discount factor* (or discount rate). The expected discounted return conditioned on state s is called the *state value* of s and is defined as

$$V^\pi(s) = \mathbb{E}^\pi \left[R_t^\gamma | s_t = s \right] \ , \tag{2.38}$$

where this expectation is obtained by taking an expectation over all possible paths generated by following the fixed behavioral policy π in the environment specified by the state transition probability $\mathcal{P}_{ss'}^a$ and expected reward function $\mathcal{R}_{ss'}^a$. $V^\pi(s)$ can be interpreted as a function of state s, and therefore, it is also called a *state value function* over $\mathcal{S}$. Similarly, the expected discounted return conditioned on both state s and action a is defined as

$$Q^\pi(s,a) = \mathbb{E}^\pi \left[R_t^\gamma | s_t = s, a_t = a \right] \ , \tag{2.39}$$

and is called a *state-action value* of (s,a) or a *state-action value function* over $\mathcal{S} \times \mathcal{A}$. The expected discounted return is then computed as

$$J^\gamma \triangleq \sum_{s \in \mathcal{S}} \mathcal{P}_s^0 V^\pi(s) \equiv \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mathcal{P}_s^0 \pi(s,a) Q^\pi(s,a) \ . \tag{2.40}$$

In the expected discounted return formalism, the objective function depends on the initial distribution $\mathcal{P}_s^0$. This formalism is commonly used in value-based methods.

2.2.2 Dynamic programming

Dynamic programming (DP), introduced by Richard Bellman, is a set of methods that can solve MDPs *with* a complete knowledge of the environment. Here, the environment implies the environmental dynamics $\mathcal{P}_{ss'}^a$ and the expected reward function $\mathcal{R}_{ss'}^a$. If we know the dynamics of the environment and expected reward function, we can exploit the Bellman equations to obtain the optimal value function using either a policy iteration method or a value iteration method.

Bellman equations

Bellman equations are recursive relationships of value functions. Depending on the value function, there exist the following types of equations.

- Bellman equation for a state value function V^π

$$\begin{aligned}
V^\pi(s) &= \mathbb{E}^\pi[R_t|s_t = s] && (2.41)\\
&= \mathbb{E}^\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\middle| s_t = s\right] && (2.42)\\
&= \mathbb{E}^\pi[r_{t+1}|s_t = s] + \gamma\mathbb{E}^\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_t = s\right] && (2.43)\\
&= \sum_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}}\pi(s,a)\mathcal{P}^a_{ss'}\mathcal{R}^a_{ss'} + \gamma\sum_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}}\pi(s,a)\mathcal{P}^a_{ss'}\underbrace{\mathbb{E}^\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_{t+1} = s'\right]}_{V^\pi(s')} && (2.44)\\
&= \sum_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}}\pi(s,a)\mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')\right] && (2.45)
\end{aligned}$$

- Bellman equation for state-action value function Q^π is

$$Q^\pi(s,a) = \sum_{s'\in\mathcal{S}}\mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma\sum_{a'\in\mathcal{A}}\pi(s',a')Q^\pi(s',a')\right] \qquad (2.46)$$

- Bellman equation for an optimal state value function $V^*(s)$

$$\begin{aligned}
V^*(s) &\triangleq \max_\pi V^\pi(s) && (2.47)\\
&= \max_\pi\sum_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}}\pi(s,a)\mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma V^*(s')\right] && (2.48)
\end{aligned}$$

- Bellman equation for an optimal state-action value function $Q^*(s,a)$

$$\begin{aligned}
Q^*(s,a) &\triangleq \max_\pi Q^\pi(s,a) && (2.49)\\
&= \max_\pi\sum_{s'\in\mathcal{S}}\mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma\sum_{a'\in\mathcal{A}}\pi(s',a')Q^\pi(s',a')\right] && (2.50)
\end{aligned}$$

2.2.3 Reinforcement learning algorithms

Reinforcement learning algorithms are a set of methods that can solve MDP problems *without* a knowledge of the environmental dynamics $\mathcal{P}^a_{ss'}$ and the expected reward function $\mathcal{R}^a_{ss'}$. An agent does not possess any information about the consequence of taking action a at state s without learning models of the environment: a forward model $\hat{\mathcal{P}}^a_{ss'}$ and a reward model $\hat{\mathcal{R}}^a_{ss'}$.

Depending on the maintenance of these environmental models, reinforcement learning methods can be divided into two types: model-free methods and model-based methods. Model-free methods do not learn the models of the environment to improve a policy. SARSA (Rummery and Niranjan, 1994) and Q-learning (Watkins and Dayan, 1992) are two of the most famous model-free algorithms. In contrast, model-based methods explicitly learn the prediction model of the next state $\hat{\mathcal{P}}^a_{ss'}$ and the next reward $\hat{\mathcal{R}}^a_{ss'}$. Note that the forward model $\hat{\mathcal{P}}^a_{ss'}$ predicts the Markov state and not the non-Markov observation.

2.3 Partially observable Markov decision processes

A partially observable Markov decision process (POMDP) is a variant of reinforcement learning that was originally designed to model the realistic decision making problems (Astrom, 1965; Smallwood and Sondik, 1973; Lovejoy, 1991). It has been extensively studied in the field of operations research and optimal control since the late 1960s and is still considered as one of the most difficult problems in the field of machine learning.

Due to its generality, POMDPs encompass wide ranges of problems with different difficulties. We first provide a formal definition of POMDP. Then, we classify POMDPs according to their difficulties in order to clarify the problems that we are going to focus on in this dissertation.

Definition The POMDP can be formally defined as a 7-tuple $\mathcal{M}_{\text{POMDP}} = (\mathcal{S}, \mathcal{A}, \mathcal{Z}, \mathcal{P}^0, \mathcal{P}, \mathcal{R}, \mathcal{O})$, where five constructs share the same definition with the MDP $\mathcal{M}_{\text{MDP}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}^0, \mathcal{P}, \mathcal{R})$, and the additional two constructs are defined as follows:

- $\mathcal{Z}$ is the *observation space.* $\mathcal{Z} \triangleq \{z_1, z_2, \ldots, z_{|\mathcal{Z}|}\}$ is assumed to be a finite set of discrete observations.
- $\mathcal{O} : \mathcal{S} \times \mathcal{A} \times \mathcal{Z} \rightarrow [0, 1]$ is the *observation distribution.*

These two additional constructs prevent an agent from directly detecting the true state of the world. The agent only receives partial information about the true state through stochastic observations.

Although the objective of POMDPs remains the same as that of MDPs, i.e., to find the behavioral policy π that maximizes a prespecified scalar objective function, this separation of an agent from the true environmental state makes the problem extremely difficult. This difficulty arises from the definition of the "state" on which the agent base its decision. The main difficulty in POMDPs is the construction of the Markovian "state." Exact algorithms construct Markovian states internally by either keeping all past experiences (no compression, instance-based approach) or maintaining sufficient statistics of them (lossless compression, belief-state-based approach). Approximate algorithms construct approximate Markov states by keeping a lossy memory of the past experiences (lossy compression, contextual-feature-based approach).

2.3.1 Classification of POMDPs with problem difficulties

No single POMDP algorithm solves all different types of POMDP problems. This is due to the varying difficulties of POMDP problems. Depending on their difficulties, we classify POMDP problems in this dissertation into three classes.

Class I POMDP A class I POMDP is solely characterized by the partial observability of the hidden environmental state. An agent receives only partial information about the true state (e.g., pixel images of a handwritten digit); however, the successful detection of the true environmental state (e.g., true class of handwritten digit) reduces the problem MDP. This class is usually not considered as a POMDP in literature because of the lack of "perceptual aliasing," which is the defining characteristic of conventional POMDPs. Perceptual aliasing implies that several environmental states are aliased to the same observation, and the agent cannot make an optimal decision solely depending on the immediate observation. However, we still consider this class as a POMDP due to its partial observability of true states. In addition, we assume no prior knowledge about the environment. An agent does not possess any prior knowledge about the environmental dynamics or about the state space of the environment (e.g., a class of possible digits).

Class II POMDP The conventional POMDPs described in literature belong to this category. The defining characteristics of class II POMDPs are (1) partial observability of environmental states, (2) perceptual aliasing, and (3) low-dimensional observation. Again, we assume no prior knowledge about the environment.

Class III POMDP The class III POMDP is an extension of the class II POMDP with high-dimensional observations. Assumptions of no prior knowledge about the environment and high-dimensional observations make this class particularly suitable for modeling decision making in the real world.

2.3.2 Incorporating memory

Identical sensory inputs can be differentiated with the help of memory traces in the agent. Some type of memory structure becomes critical to solve POMDP problems with perceptual aliasing (i.e., class II and III). There exist four approaches to incorporate past information into decision making. We use Whitehead's terminology to describe these approaches (Whitehead and Lin, 1995).

1. Instance-based approach:
    - All observations, actions, and rewards are stored as instances in memory. Then, a distance matrix is used to weigh the contribution of past instances in order to estimate the value function.
    - A collection of all instances forms a Markov state.
    - One drawback is the increasing number of instances in the memory. An online kernel sparsification method (Engel et al., 2002) can alleviate this problem.

- Design of a kernel function, or a distance matrix, drastically affects the learning performance.
- Gaussian processes reinforcement learning (GPRL) (Engel, 2005), kernel-based reinforcement learning (Ormoneit and Sen, 2002), and instance-based reinforcement learning (Santamaria et al., 1998) belong to this category.

2. Window-Q approach:
    - This approach pads the current, insufficient observation with few recent observations and actions within a fixed window size.
    - Padded observations form a Markov state.
    - It is difficult to select a fixed window size in advance. If the chosen window size is too small, the padded observations do not form a Markov state.
3. Recurrent-Q approach:
    - An Elman-type recurrent neural network is used to retain contextual features about the past observations and actions.
    - The combination of the current observation and memory (contextual features) forms a Markov state.
4. Recurrent-model approach:
    - One-step prediction module (predictor) and Q-learning module (actor) are combined to estimate the value function.
    - If its prediction becomes perfect, hidden nodes of a predictor (contextual features) should contain sufficient information about the past. Then, the combination of the current sensory inputs with contextual features forms a Markov state.

Both the recurrent-Q and the recurrent-model approaches use recurrent neural networks to retain past experiences. In the next section, we explain learning algorithms for RNNs.

2.4 Recurrent neural network

A recurrent neural network (RNN) is used to map input sequences to disired output sequences. Unlike a feedforward network, the RNN handles data with a temporal structure. Due to this ability, the RNN is commonly used for sequential prediction.

During the last few decades, several training algorithms of RNNs have been proposed. We can separate these algorithms into two types depending on the assumption of hidden connections. One type of training methods usually assumes a full recurrent connection. Examples of this first type are backpropagation through time (BPTT) (Werbos, 1990), real-time recurrent learning (RTRL) (Williams and Zipser, 1989), and extended Kalman filter (EKF) (Puskorius and Feldkamp, 1994). Another type of training methods assumes a sparse recurrent connection. Examples of this second type are reservoir computing methods, such as echo state network (ESN) (Jaeger and Haas, 2004) and liquid state machine (LSM) (Maass et al., 2002), and self-organizing recurrent network (SORN) (Lazar et al., 2009). In this section, we focus on the BPTT algorithm since it is used in the latter chapter.

2.4.1 Backpropagation through time

The network is composed of three types of layers–input, hidden and output layers–and connection weights and biases for nodes in these layers. We collectively denote the input, hidden, and output layer by $\boldsymbol{U} \triangleq \{U_i\}_{i=1}^{N^u}$, $\boldsymbol{M} \triangleq \{M_k\}_{k=1}^{N^m}$, and $\boldsymbol{X} \triangleq \{X_j\}_{j=1}^{N^x}$, where N^u, N^m, and N^x are the number of nodes in each layer, respectively. The realized values of these nodes are written in the lower case like $\boldsymbol{u}$, $\boldsymbol{m}$ and $\boldsymbol{x}$. The connection weight from the lth hidden node H_l to the kth hidden node H_k is denoted by w_{kl}^{mm}. Likewise, the connection weight from the ith input node X_i to the kth hidden node M_k is denoted by w_{ki}^{mu}. The connection weight from the kth hidden node M_k to the jth output node X_k is denoted by w_{jk}^{xm}. The hidden and output nodes have biases b_k^m and b_j^x, respectively. In order to predict the initial target before receiving the initial input, we put the special bias b_k^{m0} for the hidden nodes in the initial time step instead of b_k^m. The dynamics of the network is defined by,

$$m_{k,t} = \sigma\left\{\sum_{l=1}^{N^m} w_{kl}^{mm} m_{l,t-1} + \sum_{i=1}^{N^u} w_{ki}^{mu} u_{i,t} + b_k^m\right\}, \qquad t = 1, \ldots, T \tag{2.51a}$$

$$m_{k,t} = \sigma\left\{b_k^{m0}\right\}, \qquad t = 0 \tag{2.51b}$$

$$x_{j,t} = \sigma\left\{\sum_{k=1}^{N^m} w_{jk}^{xm} m_{k,t} + b_j^x\right\}, \qquad t = 0, \ldots, T\ . \tag{2.51c}$$

where $\sigma(z) \triangleq 1/(1 + \exp(-z))$ is a sigmoid function.

The objective of learning is to change the weight and bias parameters so that the output trajectories $\{\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_T\}$ follows a desired trajectories $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$. As a measure of RNN's performance, objective function is defined as the average of instantaneous losses E_t,

$$J(T) \triangleq \frac{1}{T+1} \sum_{t=0}^{T} E_t.$$

The instantaneous loss function is selected depending on the type of output nodes. For continuous output nodes, square error function is used,

$$E_t = \frac{1}{2} \sum_{j=1}^{J} c_j \left\{x_{j,t} - \hat{x}_{j,t}\right\}^2, \qquad t = 0, \ldots, T\ , \tag{2.52}$$

where c_j is the error weighting parameters which control the weighting of errors for each nodes. Its derivative with respect to the output $\hat{x}_j$ is given by,

$$\frac{\partial E_t}{\partial y_{j,t}} = -\{x_{j,t} - \hat{x}_{j,t}\}\ . \tag{2.53}$$

For binary output nodes, cross-entropy error function is used,

$$E_t = -\sum_{j=1}^{J} c_j \left[x_{j,t} \log \hat{x}_{j,t} + \{1 - x_{j,t}\} \log \{1 - \hat{x}_{j,t}\}\right], \qquad t = 0, \ldots, T, \tag{2.54}$$

Its derivative with respect to the output $\hat{x}_j$ is given by,

$$\frac{\partial E_t}{\partial y_{j,t}} = -\frac{x_{j,t} - \hat{x}_{j,t}}{\hat{x}_{j,t}\{1 - \hat{x}_{j,t}\}}\ . \tag{2.55}$$

The weights and biases are updated by

$$w_{ij}^{\cdot\cdot} := w_{ij}^{\cdot\cdot} - \alpha \frac{\partial J(T)}{\partial w_{ij}^{\cdot\cdot}} \tag{2.56a}$$

$$b_i^{\cdot} := b_i^{\cdot} - \alpha \frac{\partial J(T)}{\partial b_i^{\cdot}} \tag{2.56b}$$

where the dots in the superscript of weights and biases represent appropriate layer names. For the output weight w_{jk}^{xm}, the gradients in Eqs. (2.56) are defined by,

$$\frac{\partial J(T)}{\partial w_{jk}^{xm}} = \sum_{t=0}^{T} \delta_{j,t}^{x} m_{k,t} \tag{2.57}$$

where

$$\delta_{j,t}^{x} \triangleq \frac{\partial E_t}{\partial \hat{x}_{j,t}} \hat{x}_{j,t}\{1 - \hat{x}_{j,t}\}, \qquad t = 0, \ldots, T \ , \tag{2.58}$$

where $\partial E_t / \partial \hat{x}_{j,t}$ is given by Eq. (2.53) or Eq. (2.55). For the recurrent connection w_{kl}^{mm}, the gradient in Eqs. (2.56) are defined by,

$$\frac{\partial J(T)}{\partial w_{kl}^{mm}} = \sum_{t=0}^{T} \delta_{k,t}^{m} m_{l,t-1} \tag{2.59}$$

where

$$\delta_{k,t}^{m} \triangleq \frac{\partial J(T)}{\partial m_{k,t}} m_{k,t}\{1 - m_{k,t}\}, \qquad t = 0, \ldots, T \ , \tag{2.60}$$

$\partial J(T)/\partial m_{k,t}$ can be obtained by the following recursive form:

$$\frac{\partial J(T)}{\partial m_{k,T}} = \sum_{j} \delta_{j,T}^{x} w_{jk}^{xm}, \qquad t = T \tag{2.61}$$

and

$$\frac{\partial J(T)}{\partial m_{k,t}} = \sum_{j} \delta_{j,t}^{x} w_{jk}^{xm} + \sum_{l} \delta_{l,t+1}^{m} w_{lk}^{mm}, \qquad t = 0, \ldots, T-1 \tag{2.62}$$

For the input weight w_{ki}^{mu}, the gradients in Eqs. (2.56) are defined by,

$$\frac{\partial J(T)}{\partial w_{ki}^{mu}} = \sum_{t=0}^{T} \delta_{k,t}^{m} x_{k,t} \ . \tag{2.63}$$

The complete algorithm for BPTT is shown in the following section.

Back Propagation Through Time (Batch Training)

1. Set the step-size parameter

$$\alpha := (\text{appropriately small value}),$$

2. Initialize connection weights (biases are included in the weights):

$$\begin{aligned} w_{jk}^{xm} &:= (\text{at random}), & j = 1, \ldots, J;\ \ k = 1, \ldots, K+1. \\ w_{kl}^{mm} &:= (\text{at random}), & k = 1, \ldots, K;\ \ l = 1, \ldots, K+1. \\ w_{ki}^{mu} &:= (\text{at random}), & k = 1, \ldots, K;\ \ i = 1, \ldots, I+1. \\ b_{k}^{m0} &:= (\text{at random}), & k = 1, \ldots, K. \end{aligned}$$

3. Initialize hidden units and compute the initial output $\hat{x}(0)$.

$$\begin{aligned} m_{k,0} &:= \sigma(b_k^{m0}) & k = 1, \ldots, K. \\ \hat{x}_{j,0} &:= \sigma\left\{ \sum_{k=1}^{K+1} w_{jk}^{xm} m_{k,0} \right\}, & j = 1, \ldots, J \end{aligned}$$

4. For each $t = 1, \ldots, T$

   (a) Observe the current input $\boldsymbol{u}_t$.

   (b) Compute the prediction output $\hat{\boldsymbol{x}}_t$.

$$\begin{aligned} m_{k,t} &:= \sigma\left\{ \sum_{l=1}^{K+1} w_{kl}^{mm} m_{l,t-1} + \sum_{i=1}^{I} w_{ki}^{mu} u_{i,t} \right\}, & k = 1, \ldots, K \\ \hat{x}_{j,t} &:= \sigma\left\{ \sum_{k=1}^{K+1} w_{jk}^{xm} m_{k,t} \right\}, & j = 1, \ldots, J \end{aligned}$$

   (c) Evaluate the prediction error $e_{j,t}$.

$$e_j(t) := x_{j,t} - \hat{x}_{j,t}, \qquad j = 1, \ldots, J$$

5. Reset the gradients used for training

$$\begin{aligned} \frac{\partial J(T)}{\partial w_{jk}^{xm}} &:= 0, & j = 1, \ldots, J;\ \ k = 1, \ldots, K+1. \\ \frac{\partial J(T)}{\partial w_{kl}^{mm}} &:= 0, & k = 1, \ldots, K;\ \ l = 1, \ldots, K+1. \\ \frac{\partial J(T)}{\partial w_{ki}^{mu}} &:= 0, & k = 1, \ldots, K;\ \ i = 1, \ldots, I+1. \end{aligned}$$

6. Reset the effective error gain for error collection in the hidden layer.

$$\delta^m_{k,T+1} := 0, \qquad\qquad k = 1, \ldots, K.$$

7. For each $t = T, T-1, \ldots, 0$,

   (a) Evaluate the effective gain for error correction in the output layer.

   $$\delta^x_{j,t} := \begin{cases} -\hat{x}_{j,t}\,(1-\hat{x}_{j,t})\,e_{j,t}, & \text{(for continuous desired outputs)} \\ -e_{j,t} & \text{(for binary desired outputs)} \end{cases}, \quad j = 1, \ldots, J.$$

   (b) Update $\partial J(T)/\partial w^{xm}_{jk}$

   $$\frac{\partial J(T)}{\partial w^{xm}_{jk}} := \frac{\partial J(T)}{\partial w^{xm}_{jk}} + \delta^x_{j,t} m_{k,t}, \qquad j = 1, \ldots, J; \quad k = 1, \ldots, K+1$$

   (c) Propagate the effective gain for error correction to the hidden layer.

   $$\delta^m_{k,t} := \sum_{j=1}^{J} \delta^x_{j,t} w^{xm}_{jk} + \sum_{l=1}^{K} \delta^m_l (t+1) w^{mm}_{lk}, \qquad k = 1, \ldots, K.$$

   (d) Update $\partial J(T)/\partial w^{mm}_{kl}$ and $\partial J(T)/\partial w^{mu}_{ki}$

   $$\frac{\partial J(T)}{\partial w^{mm}_{kl}} := \frac{\partial J(T)}{\partial w^{mm}_{kl}} + \delta^m_{k,t} m_{l,t-1}, \qquad k = 1, \ldots, K; \quad l = 1, \ldots, K+1$$
   $$\frac{\partial J(T)}{\partial w^{mu}_{ki}} := \frac{\partial J(T)}{\partial w^{mu}_{ki}} + \delta^m_{k,t} x_{i,t}, \qquad k = 1, \ldots, K; \quad i = 1, \ldots, I+1$$

8. Update all connection weights

$$w^{xm}_{jk} := w^{xm}_{jk} - \alpha \frac{\partial J(T)}{\partial w^{xm}_{jk}}, \qquad j = 1, \ldots, J; \quad k = 1, \ldots, K+1.$$
$$w^{mm}_{kl} := w^{mm}_{kl} - \alpha \frac{\partial J(T)}{\partial w^{mm}_{kl}}, \qquad k = 1, \ldots, K; \quad l = 1, \ldots, K+1.$$
$$w^{mu}_{ki} := w^{mu}_{ki} - \alpha \frac{\partial J(T)}{\partial w^{mu}_{ki}}, \qquad k = 1, \ldots, K; \quad i = 1, \ldots, I+1.$$

Chapter 3

Reward- and Action-dependent Sensory Coding

3.1 Introduction

Recently, a reinforcement learning algorithm that is capable of handling high-dimensional inputs and actions was proposed by Sallans and Hinton (2000, 2004). This method exploits the statistical independence realized by the undirected graphical model called the Product of Experts (PoE), and it utilizes the negative free energy of the network state to approximate a state-action value function. The proposed method, which we call "free-energy-based reinforcement learning (FERL)" method, has been demonstrated to be able to solve a temporal credit assignment problem in Markov Decision Processes (MDPs) with large state and action spaces (Sallans and Hinton, 2000, 2004). In addition, compared to conventional RL algorithms with linear function approximators with prefixed basis functions such as CMAC or RBF networks (Sutton and Barto, 1998), the FERL can handle highly nonlinear value functions without feature engineering. Due to this ability to handle high-dimensional binary data without prior tweaking of features, the algorithm has a potential to bring a breakthrough for the real-world application of reinforcement learning. However, the characteristics of the FERL have not been fully investigated yet. In this chapter, we first demonstrate the noise tolerance obtained by the FERL under the newly proposed task called the "digit-floor" task. We then analyze the information coding in the activation patterns of hidden nodes in the network.

3.2 Free-energy-based reinforcement learning framework

In this dissertation, we used the RBM (Fig. 4.1(a)), which is the subtype of PoE using only binary random variables, as a model of an actor. The visible layer $\boldsymbol{V}$ is composed of binary state nodes $\boldsymbol{S}$ and binary action nodes $\boldsymbol{A}$ (i.e., $\boldsymbol{V} \triangleq \{\boldsymbol{S}, \boldsymbol{A}\}$) . The hidden layer is purely composed of binary hidden nodes $\boldsymbol{H}$. A state node S_i is connected to a hidden node H_k by a symmetric connection weight $w_{ik}^{sh} \equiv [\boldsymbol{W}^{sh}]_{ik}$, and an action node A_j is connected to a hidden node H_k by a symmetric connection weight $w_{jk}^{ah} \equiv [\boldsymbol{W}^{ah}]_{jk}$. All nodes also have bias terms: $b_i^s \equiv [\boldsymbol{b}^s]_i$ for state nodes, $b_j^a \equiv [\boldsymbol{b}^a]_i$ for action nodes, and $b_k^h \equiv [\boldsymbol{b}^h]_k$ for hidden nodes. Therefore, all the parameters of RBM-based agent can be collectively denoted as $\boldsymbol{\theta} \triangleq \{\boldsymbol{W}^{sh}, \boldsymbol{W}^{ah}, \boldsymbol{b}^s, \boldsymbol{b}^a, \boldsymbol{b}^h\}$. The equilibrium free energy of the system with inverse temperature β, which is the negative log-partition function of the posterior probability over $\boldsymbol{H}$ given a configuration ($\boldsymbol{S} = \boldsymbol{s}, \boldsymbol{A} = \boldsymbol{a}$) in thermal equilibrium, is given by

$$\begin{aligned}
F_\beta(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}) &\equiv -\frac{1}{\beta} \ln \sum_{\boldsymbol{h}} \exp(-\beta E(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{h}; \boldsymbol{\theta})) \\
&= \sum_{\boldsymbol{h}} p_\beta(\boldsymbol{h}|\boldsymbol{v}; \boldsymbol{\theta}) E(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{h}; \boldsymbol{\theta}) + \frac{1}{\beta} \sum_{\boldsymbol{h}} p_\beta(\boldsymbol{h}|\boldsymbol{v}; \boldsymbol{\theta}) \ln p_\beta(\boldsymbol{h}|\boldsymbol{v}; \boldsymbol{\theta}) \\
&= -\boldsymbol{s}^\top \boldsymbol{W}^{sh} \hat{\boldsymbol{h}} - \boldsymbol{a}^\top \boldsymbol{W}^{ah} \hat{\boldsymbol{h}} - \boldsymbol{s}^\top \boldsymbol{b}^s - \boldsymbol{a}^\top \boldsymbol{b}^a - \hat{\boldsymbol{h}}^\top \boldsymbol{b}^h \\
&\quad + \frac{1}{\beta} \left[\sum_{k=1}^{K} \hat{h}_k^\beta \log \hat{h}_k^\beta + \sum_{k=1}^{K} (1 - \hat{h}_k^\beta) \log(1 - \hat{h}_k^\beta) \right]
\end{aligned}$$

where

$$\begin{aligned}
\hat{h}_k^\beta &\equiv p(h_k = 1|\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}) && \text{(3.1a)} \\
&= \mathbb{E}[H_k|\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}] && \text{(3.1b)} \\
&= \sigma(\beta[\boldsymbol{W}^{hs}\boldsymbol{s} + \boldsymbol{W}^{ha}\boldsymbol{a} + \boldsymbol{b}^h]_k) && \text{(3.1c)}
\end{aligned}$$

is the conditional expectation of H_k given the configuration $(\boldsymbol{s}, \boldsymbol{a})$.

The network is trained so that the negative equilibrium free energy approximates the state-action value function, i.e., $Q(\boldsymbol{s}, \boldsymbol{a}) \approx -F_\beta(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta})$. Due to the existence of hidden nodes, this step is done by the EM algorithm. With the characteristics of RBM, the EM algorithm is massively simplified both analytically and computationally. The E-step is simply done by the evaluation of Eq. (3.1c). Due to the conditional independence of hidden variables, this evaluation vanishes the Kullback-Leibler divergence between variational distribution $q(\boldsymbol{h})$ and the posterior distribution $p(\boldsymbol{h}|\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta})$.

The error function to be minimized in the M-step is the average square temporal-difference (TD) error

$$J^{\mathrm{TD}} = \frac{1}{T}\sum_{t=0}^{T}\frac{1}{2}\delta_t(\boldsymbol{\theta})^2 \tag{3.2}$$

where TD error $\delta_t(\boldsymbol{\theta})$ takes one of the three following forms:

$$\begin{aligned}
\delta_t^{\mathrm{SARSA(0)}}(\boldsymbol{\theta}) &\triangleq r_{t+1} - \gamma F(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}; \boldsymbol{\theta}) + F(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\theta}) \\
\delta_t^{\mathrm{Q(0)}}(\boldsymbol{\theta}) &\triangleq r_{t+1} - \gamma \max_{\boldsymbol{a}\in\mathcal{A}} F(\boldsymbol{s}_{t+1}, \boldsymbol{a}; \boldsymbol{\theta}) + F(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\theta}) \\
\delta_t^{\mathrm{S(0)}}(\boldsymbol{\theta}) &\triangleq r_{t+1} - \gamma \sum_{\boldsymbol{a}\in\mathcal{A}} p(\boldsymbol{a}|\boldsymbol{s}_{t+1}; \boldsymbol{\theta}) F(\boldsymbol{s}_{t+1}, \boldsymbol{a}; \boldsymbol{\theta}) + F(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\theta}) \ ,
\end{aligned}$$

where γ is a discount factor for future rewards. First form of TD error $\delta_t^{\mathrm{SRASA(0)}}$ is used for the SARSA learning algorithm without an eligibility trace. The second form of TD error $\delta_t^{\mathrm{Q(0)}}$ is used for the Q-learning algorithm without eligibility trace. The third form of the TD error $\delta_t^{\mathrm{S(0)}}$ is used for the modified SARSA algorithm, which reduces the variance in the parameter update using the current policy. Since we can easily calculate the action-selection probability (or take samples from it) and the free energy for certain state-action pair, computation of this TD error $\delta_t^{\mathrm{S(0)}}$ is feasible.

Partial derivatives of the error function Eq. (3.2) give the episodic update rule of the parameters.

$$\Delta\boldsymbol{\theta} = -\alpha \nabla_{\boldsymbol{\theta}} J_T^{\mathrm{TD}} \tag{3.3}$$

where α is a learning rate and

$$\nabla_{\boldsymbol{\theta}} J_T^{\mathrm{TD}} = \frac{1}{T}\sum_{t=0}^{T}\delta_t \nabla_{\boldsymbol{\theta}} F(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\theta}) \ . \tag{3.4}$$

Due to the characteristics of RBM, the derivative of equilibrium free energy with respect to each parameter $\nabla_{\boldsymbol{\theta}} F(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta})$ in Eq. (3.4) can be compactly expressed as

$$\frac{\partial F(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})}{\partial w_{ik}^{sh}} = -s_i \hat{h}_k \tag{3.5}$$

$$\frac{\partial F(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})}{\partial w_{jk}^{ah}} = -a_j \hat{h}_k \tag{3.6}$$

$$\frac{\partial F(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})}{\partial b_i^{s}} = -s_i \tag{3.7}$$

$$\frac{\partial F(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})}{\partial b_j^{a}} = -a_j \tag{3.8}$$

$$\frac{\partial F(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})}{\partial b_k^{h}} = -\hat{h}_k \ . \tag{3.9}$$

This update concludes the M-step of the EM algorithm.

The online update rule for each parameter can be easily derived as

$$\Delta w_{ik}^{sh} = \alpha \delta_t s_{i,t} \hat{h}_{k,t} \tag{3.10a}$$

$$\Delta w_{jk}^{ah} = \alpha \delta_t a_{j,t} \hat{h}_{k,t} \tag{3.10b}$$

$$\Delta b_i^{s} = \alpha \delta_t s_{i,t} \tag{3.10c}$$

$$\Delta b_j^{a} = \alpha \delta_t a_{j,t} \tag{3.10d}$$

$$\Delta b_k^{h} = \alpha \delta_t \hat{h}_{k,t} \ . \tag{3.10e}$$

Actions can be selected using either a clamping or a sampling method depending on the number of possible actions. The clamping method works best for problems with a small number of alternative actions. With this method, free energy at state $\boldsymbol{S} = \boldsymbol{s}$ is calculated for each possible action by "clumping" action nodes $\boldsymbol{A}$ for each configurations $\boldsymbol{a} \in \mathcal{A}$. Then, To select an action at a given state $\boldsymbol{s}$, we used softmax action selection rule with inverse temperature β

$$\pi(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}, \beta) = p(\boldsymbol{a}|\boldsymbol{s}; \boldsymbol{\theta}, \beta) = \frac{\exp\{-\beta F(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})\}}{\sum_{\hat{\boldsymbol{a}}} \exp\{-\beta F(\boldsymbol{s}, \hat{\boldsymbol{a}}, \boldsymbol{\theta})\}} \ . \tag{3.11}$$

Alternatively, the action can be selected using the alternating Gibbs sampling. First, the observation nodes are set to the current inputs, and action nodes and

reinforcement learning $(-)$	information theory $(+)$	optimal control $(+)$
reward	energy	instantaneous cost
value function	free energy	cost-to-go function

Table 3.1. Parallelism between reinforcement learning, information theory, and optimal control.

hidden nodes are randomly initialized. Next, the binary states of hidden nodes are sampled from the posterior distribution $P(\boldsymbol{h}|\boldsymbol{s},\boldsymbol{a};\boldsymbol{\theta}) = \prod_k P(h_k|\boldsymbol{s},\boldsymbol{a};\boldsymbol{\theta})$. Then, the action nodes are updated using the states of hidden units $P(\boldsymbol{a}|\boldsymbol{h};\boldsymbol{\theta}) = \prod_j P(a_j|\boldsymbol{h};\boldsymbol{\theta})$. The previous two steps are repeated for a long time; then we can sample the action from the distribution $P(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})$.

In our experiment, instead of Gibbs sampling, we used the softmax action selection rule with an inverse temperature parameter β by directly calculating the free energies for each action due to the following two reasons: (i) the number of action is small in the current task, and (ii) action nodes has a constraint such that one of the action nodes can be active at a time.

3.3 Why free energy?

In FERL, a negative equilibrium free energy of an RBM is used to represent a state-action value function. The reason for choosing this seemingly unintuitive represetation becomes clear by realizing the parallelism between reinforcement learning, information theory, and optimal control (Table 3.1).

A parallelism between reinforcement learning and optimal control is easy to recognize. Reinforcement learning and optimal control use the opposite sign for functionally same quantities. As an instantaneous quantity received by taking an action at a certain state, a reward is used in reinforcement learning as opposed to an instantaneous cost in optimal control. The reward in reinforcement learning encourages a certain action, and the instantaneous cost discourages a certain action. In other words, an instantaneous cost is equivalent to a negative reward. In contrast, a value function in reinforcement learning and a cost-to-go function in optimal control theory capture the long-term stochastic consequences starting from a certain state (or a state-action pair) with a fixed policy. Regardless of the stochasticity, both the value function and the cost-to-go function take a scalar value and only depend on the starting state (or the state-action pair) due to the assumptions of infinite-horizon and the Markov property over states. These assumptions ensure that the state (or state-action) distribution reaches the equilibrium distribution. In other words, the path distribution starting from a certain state (or a state-action pair) given a certain policy is completely captured by the equilibrium distribution over states (and actions given by the policy for these states). Due to the existance of an equilibrium distribution over the state space (or the state-action space), the calculation of the expectation over spatio-temporal path can be replaced by the one of the expectation over the state (or state-action) space.

A parallelism between reinforcement learning and information theory is more difficult to interpret. Let us first recognize that a reward in reinforcement learning (or an instantaneous cost) is equivalent to an energy (or an unnormalized surprise) in the information theory with an opposite sign. In other words, the reward is the negative energy. Next point for understanding this parallelism is that the calculation of a value function can be interpreted as taking expectation with respect to the equilibrium distribution under the assumption of infinite-horizon and the Markov property over states. Due to the existance of an equilibrium distribution, the calculation of the expectation over spatio-temporal path can be replaced by the one of the expectation over space. This replacement of spatio-temporal expectation by the spatial expectation starting from a certain state (or state-action pair) is well captured by an equilibrium distribution of an RBM with the clumped state (or state-action) nodes. In other words, the equilibrium distribution of the RBM with clumped state-action nodes can capture distribution over paths starting from the given state-action pair. Therefore, the negative free energies of the RBM can be used to represent state-action values for MDPs. If a problem use a finite horizon setting, we need to think a path free energy explicitly (Kappen, 2005).

3.4 Digit-floor task

The digit-floor task, which is a variation of a grid maze task and classified as the class I POMDP (Section 2.3.1). This task is specifically designed to provide a common platform in the reinforcement learning community to investigate how the hidden structure of the task behind noisy and high-dimensional observations are extracted and represented by a reinforcement learning agent. In the digit-floor task, an agent receives noisy high-dimensional observation (i.e., an image of handwritten digit corrupted with white noise) in each grid instead of the index of grid used in the typical grid maze task. There are several points that makes this task unique and attractive.

First, the states of the POMDP task, i.e., the class labels of handwritten digits, are explicitly defined; therefore, we can use the class labels for analyzing or visualizing results. For example, we can compute the statistics in the activation patterns of hidden nodes induced by pixel images belonging to each class label. Besides, the statistics is useful for evaluating how clearly the agent can distinguish pixel images belonging two distinct classes.. Next, this task is easy to implement and also easy to replicate the simulation in the exactly same setting. Also, the optimal policy and/or return can be easily found. In this paper, we used the digit-floor task to test the noise tolerance of the FERL and to investigate the type of information coded in the activation patterns of a hidden layer.

The goal of an agent in the digit-floor task is to discover the "optimal" way to act on a "digit floor" autonomously. The "optimal" behavioral policy leads to the highest discounted return in the sense of expectation. At each time step, the agent decides the direction of movement based on previous experiences and the current noisy high-dimensional observation of a handwritten digit painted on the floor. Neither the agent can see the entire floor at once nor detect the true class labels of handwritten digits, so that the problem is partially observable. After the action is executed, the agent receives a numerical reward and moves toward its intended direction. When the agent reaches the pre-specified goal positions, the agent is randomly transported to a position other than the goal positions.[1]

[1]We can terminate the episode after visiting the goal state and make it an episodic task though we employed the continuing version of the task and defined the tiles with the digit of "4" as goal positions in this dissertation for convenience.

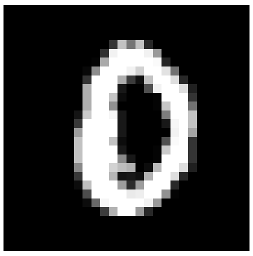
The MNIST database of handwritten digits, which contains 60,000 training images and 10,000 test images, is used for the patterns on the tiles. The original pixel intensity of the images in the MNIST data is represented by an integer between 0 and 255. By applying the mean-field approximation for the pixel, the intensity is linearly normalized within the range of $[0, 1]$. A “tile” is composed of 28×28 pixels of a handwritten digit as shown in Fig. 3.1(a). A “patch” is composed of 3×3 tiles positioned in a predefined arrangement as shown in Fig. 3.1(b). A “floor” comprises $L \times L$ patches, where for each patch, different tiles are selected randomly from the training data set. In other words, all patches on the floor share the same pre-specified arrangement though the handwritten digits on tiles are all different. The floor has a torus structure; hence, it has no “edge.” Fig. 3.1(c) shows a typical example of floor patterns with $L = 4$.

To investigate the properties of the FERL, two constraints are imposed in this study.[2] The first constraint is that the unit distance of movement is one tile. The other constraint is that the agent has an “imperfect” vision system with a visual field size of 28×28 pixels; here, “imperfect” implies that some of pixels observed in the current input are randomly replaced by white noise. Therefore, an agent never observes the exact same visual input even if it has visited the same tile many times. Examples of the noisy inputs are shown in the Fig. 3.2.

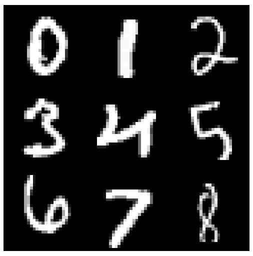
The digit-floor task is formally defined as a POMDP, which is commonly specified with a 7-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{Z}, \mathcal{P}^0, \mathcal{P}, \mathcal{R}, \mathcal{O})$. $\mathcal{S}$ is a set of hidden states on the floor, which is defined as a subset of all the possible types of tiles $\mathcal{S} \subset \mathcal{S}_0 \triangleq \{$“0”, ..., “9”, “N”$\}$[3]. A set of all the possible actions is denoted by $\mathcal{A} \equiv \{$‘N’, ‘E’, ‘S’, ‘W’$\}$, which indicates “step one tile to the North/East/South/West.” A set of all the possible observations is $\mathcal{Z} \equiv [0, 1]^{784}$. An initial distribution $\mathcal{P}^0 : \mathcal{S} \to \mathbb{R}$ is defined as the uniform distribution over all non-goal states. A state transition $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is deterministic, and the agent always moves toward the intended position. A default reward function $R_0 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as follows: the movements toward the tile “4”, which is shown as a blank tile in the center of the patch in Fig. 3.3, from neighboring tiles give a +10 reward. Clockwise movements along the boundary of each patch result in a +4 reward. All other movements

[2]These constraints can be removed in the general setting.

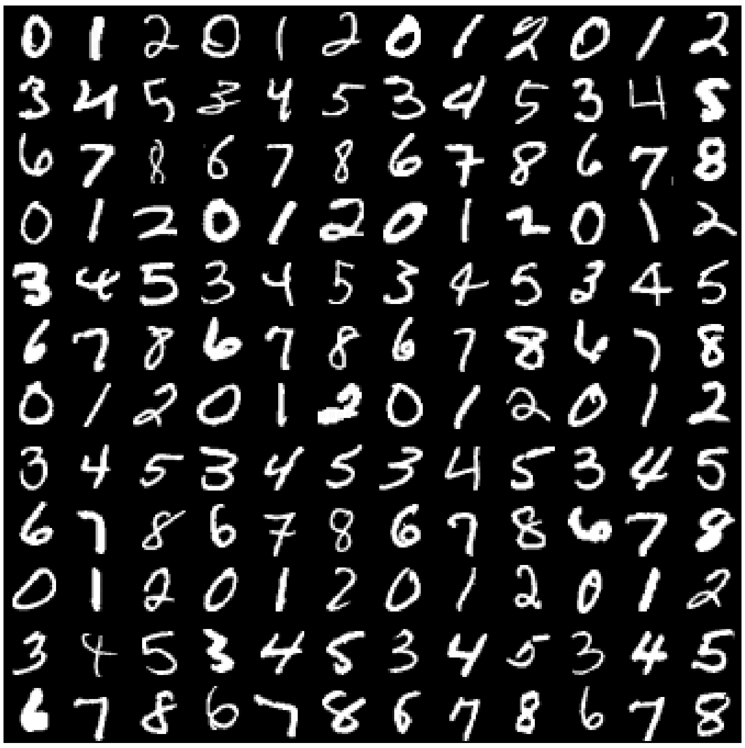
[3]A double quotation mark is used to indicate a certain digit class. Also “N” indicates that a tile is covered with white noise.



(a) Digit tile



(b) Digit patch



(c) Digit floor

Figure 3.1. Basic components of a digit floor: (a) is composed of 28×28 pixels; (b) is composed of 3×3 tiles; and (c) is composed of 4×4 patches.

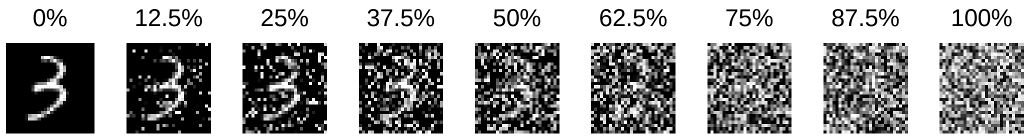


Figure 3.2. Examples of observations under noise. The number on the top of each image shows the noise level (the percentage of pixels replaced by white noise).

incur a -2 reward. The probability distribution over the observation space given the true state $P(\boldsymbol{z} \in \mathcal{Z}|\boldsymbol{s} \in \mathcal{S})$ is implicitly determined on the basis of the exact floor map and noise levels.

3.5 Simulation results

The basic setting of our simulation was as follows. A single simulation run, which is called "epoch," consisted of two phases. In the training phase, an agent acted on the predefined digit floor. In the testing phase, 400 new images of each digit, which had never appeared in the training phase, were shown to the agent, and the behaviors of the network were investigated.

The meta-parameters for learning were set as follows. A learning rate α in Eqs. (3.10) was exponentially decreased from 0.1 to 0.01 in the course of each epoch. In order to avoid the unwanted influence of the large number of connections on the speed of learning, the learning rate was divided by the square root of the number of total edges in the network. The discount rate γ in Eqs. (3.10) was set to 0.9. The inverse temperature β in Eq. (3.11), which is a parameter in the softmax action selection rule, was linearly increased from 0 to 2.

3.5.1 Noise tolerance

In order to test the influence of noise in both the training and testing phases, images were tainted with different levels of noise. We conducted tests for nine different noise levels incremented by 12.5%, or equivalently 98 noise pixels, from the noiseless image to the purely noisy image (Fig. 3.2).

An agent was trained for 100,000 steps per epoch. The agent had 20 hidden

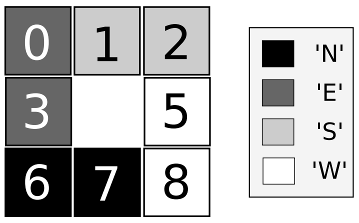


Figure 3.3. Optimal actions.

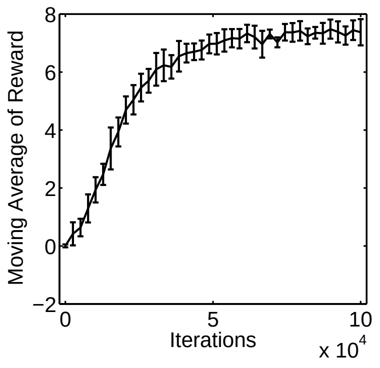


Figure 3.4. Moving average of the immediate rewards over 100 steps.

nodes. The optimal action at each hidden state is shown in Fig. 3.3. The course of training with 12.5% noise level is shown in Fig. 3.4. The vertical axis shows the moving average of the immediate rewards obtained in the previous 100 steps. The error bar along this axis represents the standard deviation of the moving average over 10 epochs with different initialization. As shown in the figure despite the 12.5% noise level in the training phase, the agent performs at a nearly optimal level after 60,000 steps.[4] The connection weights between observation nodes and hidden nodes before and after training are shown in Fig. 3.5.1.

Before and after each epoch of training, 400 test images with various noise levels were shown to the agent. Fig. 3.6(b) shows the count of the greedy actions induced by showing 400 noisy test images to the agent that is trained on a digit floor with 12.5% observation noise. The results for each digit are organized in such a manner that they match the pre-configured position of the digit floor. For example, the upper-left graph of Fig. 3.6(b) illustrates that most of the test images of “0” triggered the correct optimal action ‘E’ in all the 10 epochs after training. The same behavioral pattern was observed for all digits used in the task, and therefore the result supports the robustness of the FERL to noise. For comparison, the same 400 noisy test images were shown to the agent before training. The result in Fig. 3.6(a) indicates that an agent does not have a behavioral tendency for a certain digit.

[4]The upper-bound of theoretically expected reward per time step is $8 = ((4+10)+(10))/(2+1)$.

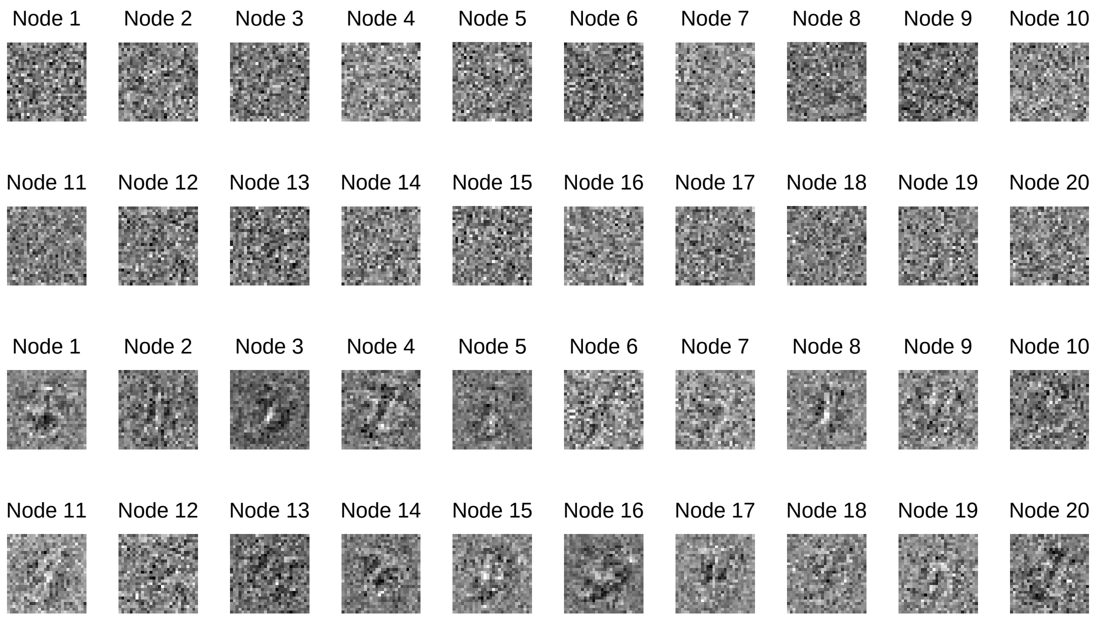


Figure 3.5. Connection weights from observations to 20 hidden nodes: weights before the training (Upper two rows) and weights after the training (lower two rows)

The agent was able to perform at the satisfactory level even with 37.5% noise level in a test set [5]. However, the The performance deteriorated after 50% of noise level in the test set, as shown in Fig. 3.6(c). When trained with 50% of noise, the performance did not deteriorate even with 50% of test noise, as shown in Fig. 3.6(d). The agent could even perform at the satisfactory level with 62.5% test noise level before the performance deteriorated at 75% of noise level. With 75 % noise level, even people find it difficult to detect the correct class of a given digit, as shown in Fig. 3.2. The results suggest that the tolerance to the noise get better as the noise level in training increases.

3.5.2 Outlier detection

400 test images of handwritten "1" were shown to the agent before and after the training. Then, images are sorted by the probability of selecting the optimal action for digit "1", which is south. The result is shown in Fig. 3.7 Before the training the agent did not show any preference for moving south by seeing typical one (3.7(a)). However, after the training, the agent exhibited the clear preference for moving south by seeing typical "1" and tries to move non-south direction by seeing atypical "1".

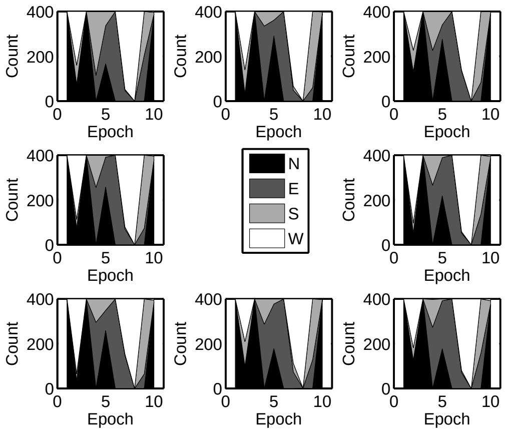
3.5.3 Population coding of reward-dependent and -invariant states

In order to analyze information coded in the activity patterns of hidden nodes, we modified the previous floor pattern. All the tiles in state "5" in Fig. 3.1(c) were replaced by purely noisy images that were statistically independent of one another. The example of the floor with the noise tile is shown in Fig. 3.8.

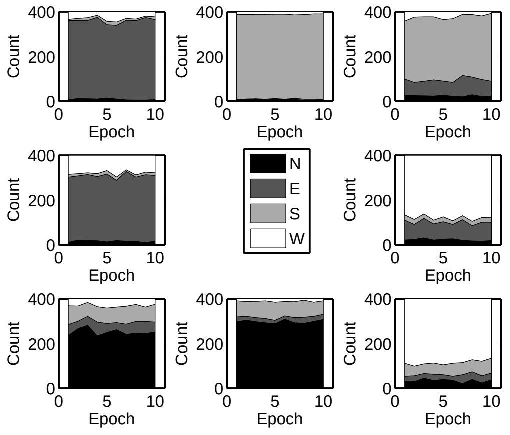
An agent was trained for 30,000 steps per epoch. We repeated the epoch for 10 times with different initializations for position and weights. The observation noise levels both in the training and test phases were fixed at 12.5%.

Two different reward functions were designed to verify reward-dependent coding. The first reward function was the same as the previously used one, which we call a clockwise (CW) condition. Similarly, the second reward function is called a

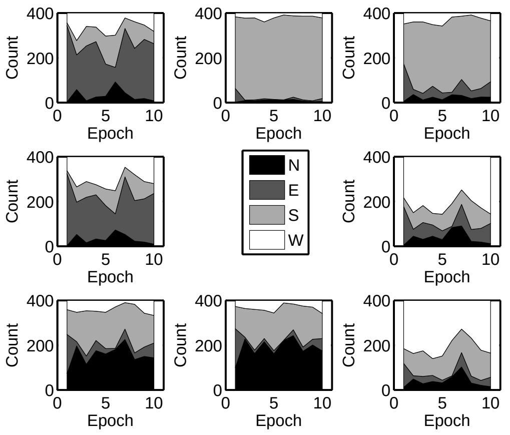
[5] Data not shown due to the limited space.



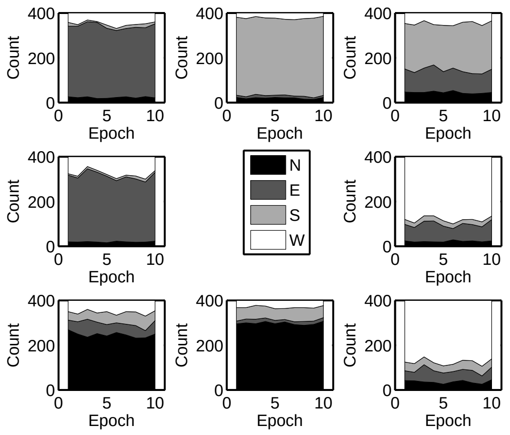
(a) (12.5%, 12.5%), before training



(b) (12.5%, 12.5%), after training

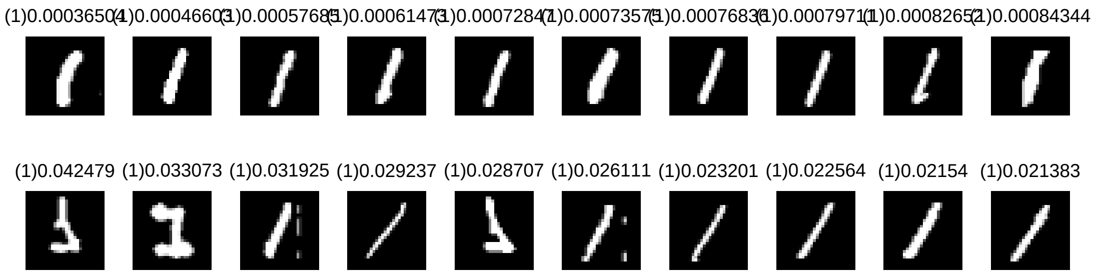


(c) (12.5%, 50%), after training

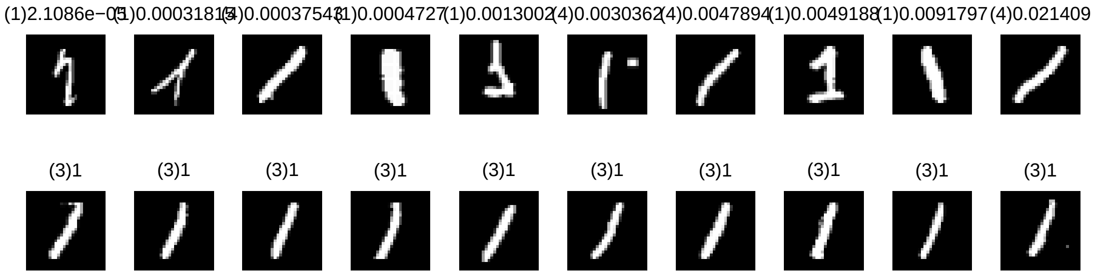


(d) (50%, 50%), after training

Figure 3.6. Count of greedy actions induced by 400 test images for different combinations of training and test noise levels. The two numbers in the parenthesis indicate the noise levels in the training and testing phases.



(a) Before the training.


(1)2.1086e−05(1)0.0003181(4)0.0003754(1)0.0004727(1)0.0013002(4)0.0030362(4)0.0047894(1)0.0049188(1)0.0091797(4)0.021409
(3)1
(3)1
(3)1
(3)1
(3)1
(3)1
(3)1
(3)1
(3)1
(3)1

(b) After the training.

Figure 3.7. Outlier detection: Images are sorted by the action-selection probability to go south (optimal action for the digit “1”). The probability is shown on top of each image. Top-left images induces the lowest probability and bottom-left images induced the highest probability.

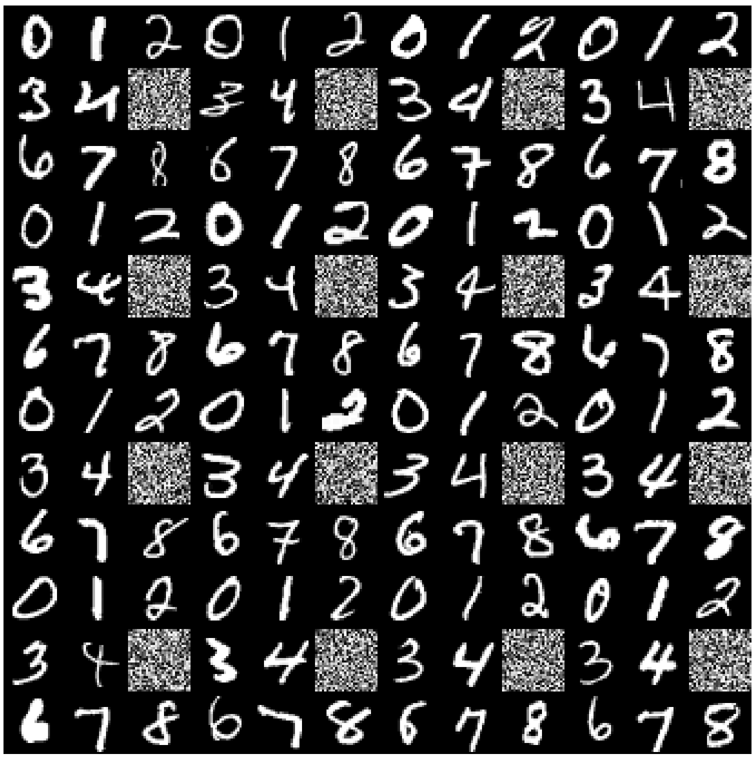
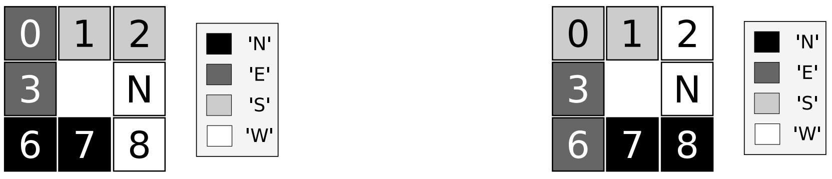


Figure 3.8. A digit floor with the noise tile

counterclockwise (CCW) condition. The magnitude of reward in both conditions were the same. The only difference was the optimal action at each corner of the patches.



(a) CW (b) CCW

Figure 3.9. Optimal actions under the clockwise (CW) and counterclockwise (CCW) conditions.

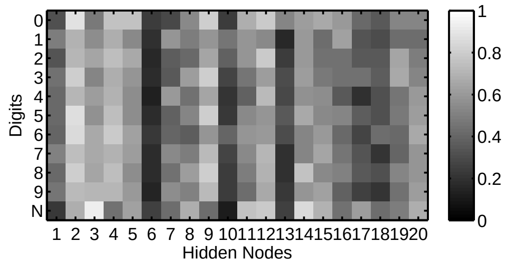
When 400 test images of each digit were presented to a pre- or post-trained agent, 20 hidden nodes showed differential activations. Figure 3.10(a) shows the median activation of 20 hidden nodes[6] when 400 images of each digit were presented for the pre-trained agent in the epoch 1. We can see several vertical bands in the figure. It indicates that corresponding hidden nodes do not show the particular preference to certain digits. This trend changes after training. Figure 3.10(b) and Fig. 3.10(c) shows the median activation of 20 hidden nodes given 400 test images of each digit to the agent trained in CW and CCW conditions, respectively. In both conditions, hidden nodes have differential activation patterns for distinct digits compared to the one without training. Also, if we let the agent walk over the floor freely without the delivery of reward, activation level of all hidden nodes become very low as shown in Fig. 3.10(d).

In order to quantify the similarity between the activation patterns induced by certain digits, we first calculated the median firing rate (MFR) of hidden nodes for each digit c. The criterion is given by

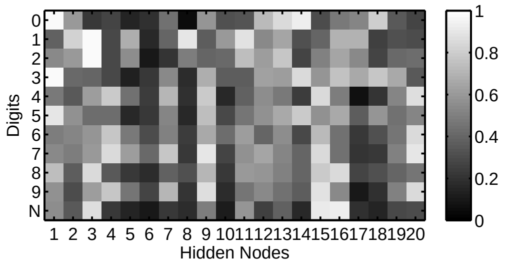
$$\mathrm{MFR}_c = \mathrm{Median}_{\boldsymbol{z}\in\mathcal{D}_c^{\mathrm{test}}}\left[\frac{1}{|\mathcal{A}|}\sum_{\boldsymbol{a}\in|\mathcal{A}|} P(\boldsymbol{h}|\boldsymbol{z},\boldsymbol{a})\right] , \tag{3.12}$$

where $\mathcal{D}_c^{\mathrm{test}}$ is a set of test data belonging to the digit c, and the operator $\mathrm{Median}_{\boldsymbol{z}\in\mathcal{D}_c^{\mathrm{test}}}$ means to take the median of the expression inside the bracket

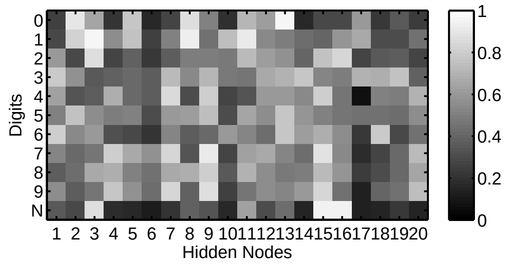
[6]Activation of a hidden node H_k for a given input $\boldsymbol{s}$ is defined as a marginal posterior: $\sum_{\boldsymbol{a}} P(h_k|\boldsymbol{s},\boldsymbol{a})P(\boldsymbol{a}|\boldsymbol{s})$.



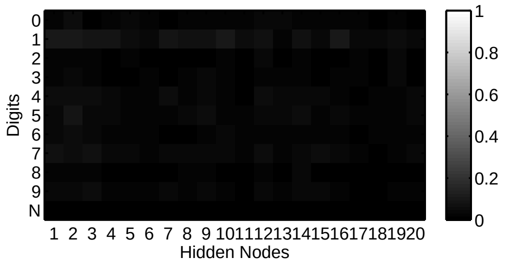
(a) Before training



(b) CW condition



(c) CCW condition



(d) No reward condition

Figure 3.10. The median activation of 20 hidden nodes when 400 test images of each digit are presented to an agent trained in different reward conditions.

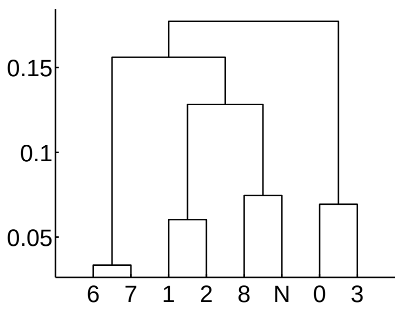
with respect to $\mathcal{D}_c^{\text{test}}$. The influence of action nodes was marginalized out by assuming the uniform prior over all admissible actions $\mathcal{A}$. For any pair of digits (i, j), the cosine angle between MFR_i and MFR_j, which are denoted by $\boldsymbol{G}_{ij}$, were used as the similarity measure of the MFRs. All the cosine angles were collectively represented by a Gram matrix $\boldsymbol{G}$. In order to visualize a structure captured by the Gram matrix, hierarchical clustering was employed. A distance matrix used for the hierarchical clustering was created from the Gram matrix using $\boldsymbol{D} = \boldsymbol{g}\mathbf{1}^\top + \mathbf{1}\boldsymbol{g}^\top - 2\boldsymbol{G}$, where $\boldsymbol{g}$ denotes the vectorized diagonal entries of $\boldsymbol{G}$.

In the CW condition, as shown in Fig. 3.9(a), the following pairs had the same optimal action: ("1", "2"), ("0", "3"), ("6", "7"), ("8", "N") where "N" denotes the white noise. As shown in Fig. 3.11(a), each pair with the same optimal action formed a distinct cluster. In the CCW condition, the following pairs had the same optimal action: ("0", "1"), ("3", "6"), ("7", "8"), ("2", "N"). Still, all the pairs formed separate clusters. These aggregation patterns could not be explained by the common statistical regularity between digit images because the digit closest to "N" was "8" in the CW condition as opposed to "2" in the CCW condition.

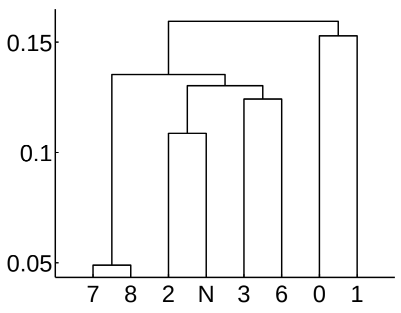
As shown in Fig. 3.11(c), without any reward, perceptually similar "1" and "7" were grouped together; in the same manner, "3" and "8" were grouped together. These subjective perceptual similarities are further confirmed by calculating the average mutual information between pair-wise class labels $C \equiv \{c_1, c_2\} \subset \mathcal{S}_0$ and a pixel intensity $Z_i \in [0, 1]$, which is an average of the value calculated for each pixel

$$I(Z_i; C) = H(Z_i) - \sum_{j=1}^{2} P(C = c_j) H(Z_i | C = c_j).$$

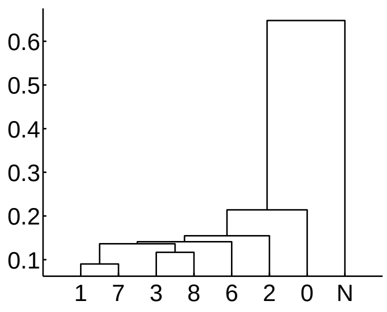
It measures how much we know about the class identity in average by just observing a single pixel. If the mutual information is low, it means a pixel value itself is not a good indicator of a class label; therefore, low mutual information indicates that two classes are perceptually similar. Fig. 3.12 shows the hierarchically clustered digits according to the mutual information calculated for each pair of digits in the test data set. Since the class "N" has a significantly higher mutual information than the rest of the data, this class is excluded from the figure in order to show the clustering pattern clearly. The major clustering trends



(a) CW, 20 nodes



(b) CCW, 20 nodes



(c) No reward, 20 nodes

Figure 3.11. Hierarchically clustered median firing patterns of 20 hidden nodes.

observed in Fig. 3.12 is same as the one in Fig. 3.11(c). Notice the clustering in Fig. 3.12 is created just by the test data, and therefore reflects the statistical regularities of the data set. This indicates that without a reward information, an agent extracts reward-independent statistical regularities of digits on the floor. This viewpoint is also supported by the large distance between "N," which did not possess any statistical regularities, and other digits.

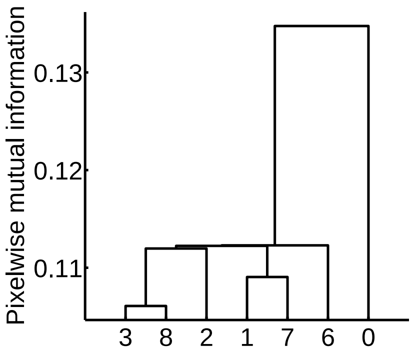
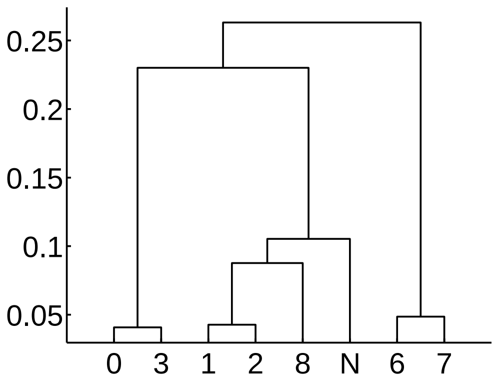


Figure 3.12. Hierarchically clustered digits according to the mutual information
.

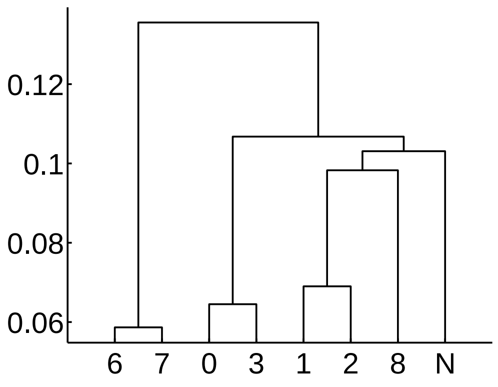
Here, the size of the hidden layer was varied to examine its impact on the reward-driven representation of hidden states. Two new conditions with 10 and 40 hidden nodes were inspected, and the results were compared with the default condition with 20 hidden nodes shown in Fig. 3.11(a). Figs. 3.13(a) and (b) are not clearly different as compared to Fig. 3.11(a). Therefore, we can safely state that the reward-dependent representations formed by FERL do not severely depend on the number of nodes in a hidden layer.

3.6 Summary and discussion

In this chapter, we showed that both the reward and the statistical regularity in the images of handwritten digits influenced the encoding of the states, i.e., class labels of digits. We found that the information about the class labels was robustly coded in the activation patterns of hidden nodes in the RBM. A new task called digit-floor task was introduced to test the properties of the FERL.



(a) CW, 10 nodes



(b) CW, 40 nodes

Figure 3.13. Hierarchically clustered median firing patterns of 10 and 40 hidden nodes.

In order to see the encoding of digits clearly, we used an easy class of POMDP (class I in section 2.3.1), which can be solved by the conventional reinforcement learning algorithms for MDPs if they could perfectly identify the digit class for any pixel image.

The use of a RBM for a reinforcement learning agent appears to be biologically plausible due to the following few reasons. First reason is the use of stochastic binary nodes and its unique learning rule (Eq. (3.10)), which can be interpreted as a Hebb rule modified with TD error. The second reason is the entropy maximization of posterior distribution over hidden nodes. It makes the efficient use of limited resources possible. The third reason is the gradual degradation of performance with noise.

Chapter 4

Solving POMDPs without Prior Knowledge of an Environment

4.1 Introduction

In the cases described in the previous chapter, past experiences were unnecessary for optimal decision making. Immediate observations contained enough information for optimal decision making. In this chapter, we focus on a conventional POMDP with perceptual aliasing.

In this chapter, we extend the free-energy-based reinforcement learning (FERL) framework to handle POMDPs using Whitehead's recurrent-model architecture (Whitehead and Lin, 1995). The proposed method can reliably solve POMDPs with low-dimensional observations without any prior knowledge of environmental dynamics. This architecture also has the potential to handle high-dimensional observations.

4.2 Model architecture

We employ Whitehead's recurrent-model architecture (Whitehead and Lin, 1995) to incorporate task-relevant memory for solving POMDP problems. Fig. 4.1(b) shows the proposed recurrent-model architecture consisting of two parts: an Elman-type recurrent neural network (RNN) for one-step prediction (predictor) and a RBM for Q-learning (actor).

The predictor module behaves so as to predict an upcoming observation $\boldsymbol{y}_t$ and an encoded reward $\boldsymbol{r}_t$[1] jointly based on the memory $\boldsymbol{m}_t$, which is supposed to summarize the history of all past events. The binarized reward $\boldsymbol{r}_t$ is used instead of a raw scalar reward r_t in order to share a single memory layer for both observation and reward predictions.

At the initial time step ($t = 0$), the prediction of the initial observation $\boldsymbol{y}_0$ is strongly controlled by the initial memory bias $\boldsymbol{b}^m_{\text{init}}$:

$$\boldsymbol{m}_0 = \sigma_{\beta_m}(\boldsymbol{b}^m_{\text{init}}) \tag{4.1}$$

$$\boldsymbol{y}_0 = \sigma_{\beta_y}(\boldsymbol{W}^{ym}\boldsymbol{m}_0 + \boldsymbol{b}^y) \ , \tag{4.2}$$

where β_m and β_y are gain parameters for updating a memory and outputting an observation, respectively. At each time step t, the internal potential of the memory is given by a linear transformation of the previous observation, action, and memory $(\boldsymbol{y}_{t-1}, \boldsymbol{a}_{t-1}, \boldsymbol{m}_{t-1})$. Then, the activation of the current memory layer, $\boldsymbol{m}_t$, is given by the application of a sigmoid function $\sigma(\cdot)$ on the internal potential:

$$\boldsymbol{m}_t = \sigma_{\beta_m}(\boldsymbol{W}^{my}(c_y\boldsymbol{y}_{t-1}) + \boldsymbol{W}^{ma}(c_a\boldsymbol{a}_{t-1}) + \boldsymbol{W}^{mm}\boldsymbol{m}_{t-1} + \boldsymbol{b}^m) \ , \tag{4.3}$$

where c_y and c_a are input scaling parameters for the previous observation and action, respectively. [2] The input scaling parameters control the the relative influence of the previous observation, action, and memory on the memory update in Eq. (4.3). Once the activation of the memory layer $\boldsymbol{m}_t$ is set, the network predicts ($\boldsymbol{y}_t$, $\boldsymbol{r}_t$) as the composite transformation of a linear mapping of $\boldsymbol{m}_t$ followed by a sigmoid function $\sigma(\cdot)$:

$$\boldsymbol{y}_t = \sigma_{\beta_y}(\boldsymbol{W}^{ym}\boldsymbol{m}_t + \boldsymbol{b}^y) \tag{4.4}$$

$$\boldsymbol{r}_t = \sigma_{\beta_r}(\boldsymbol{W}^{rm}\boldsymbol{m}_t + \boldsymbol{b}^r) \ , \tag{4.5}$$

[1]The notation r denotes a scalar reward, and the vector notation $\boldsymbol{r}$ denotes a bit coding of the scalar reward with respect to all possible rewards.

[2]Note that $\boldsymbol{W}^{my}$ is not equal to the transpose of $\boldsymbol{W}^{ym}$. $\boldsymbol{W}^{my}$ is the weight from the input observation layer $\boldsymbol{y}_{t-1}$ to the hidden memory layer $\boldsymbol{m}_t$. On the other hand, $\boldsymbol{W}^{ym}$ is the weight from the hidden memory layer $\boldsymbol{m}_t$ to the output observation layer $\boldsymbol{y}_t$.

where β_r is a gain parameter for outputting an encoded reward.

A cross entropy error function is used to measure the instantaneous prediction error:

$$\begin{aligned} E_t = &-\sum_{i=1}^{N^y} \rho_y \left[y_{i,t} \log \hat{y}_{i,t} + \{1 - y_{i,t}\} \log \{1 - \hat{y}_{i,t}\} \right] , \\ &-\sum_{i=1}^{N^r} \rho_r \left[r_{i,t} \log \hat{r}_{i,t} + \{1 - r_{i,t}\} \log \{1 - \hat{r}_{i,t}\} \right] , \qquad t = 1, \ldots, T_{\text{epi}} \ , \end{aligned} \tag{4.6}$$

where ρ_y and ρ_r are error weighting parameters for binary observations and encoded rewards, respectively. These two parameters are used to balance the relative importance between the observation and the reward prediction.

All weights and biases of the predictor are randomly initialized and trained by the backpropagation through time (BPTT) algorithm. Other parameters such as gain parameters, input scaling parameters, and error weighting parameters are adjusted manually for each task.

The *state layer* of the actor module is composed of the current observation and predictor's memory $(\boldsymbol{y}_t, \boldsymbol{m}_t)$. The actor is trained by the SARSA(0) algorithm with Eq. (3.10).

4.3 Experiments

We used a regular T-maze task in order to show the proposed model's ability to solve a POMDP without any prior knowledge of the environmental dynamics. We also designed a matching T-maze task in order to show the proposed model's ability to integrate memory and observation on the fly. PCA is applied to the activation patterns of functional modules (e.g., a predictor's memory layer, an actor's state, etc.) in order to investigate the changing trends of characteristic firing patterns during the course of performance improvement.

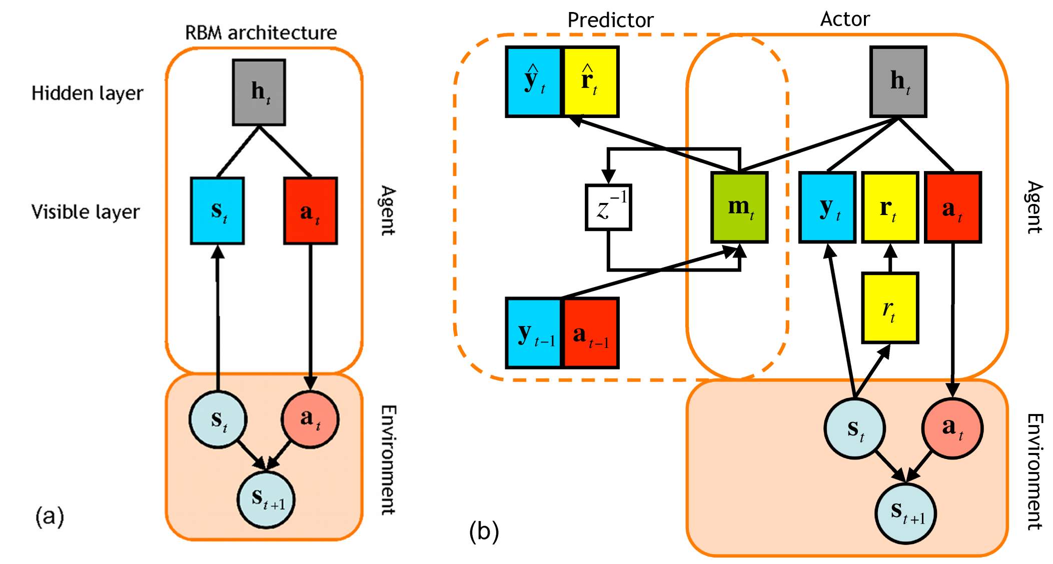


Figure 4.1. Models for handling high-dimensional inputs. (a) An actor-only architecture for MDPs. (b) A predictor-actor architecture for POMDPs.

4.3.1 Regular T-maze task

Task specification

A regular T-maze task is a non-Markovian grid-based T-maze task first introduced by Bakker (2002). The correct position of the goal depends on the initial signal at the start position. This task only requires an agent to remember the initial signal to enter the correct goal after it successfully reaches the T-junction.

An agent has four possible actions: go one step North, West, East, or South.

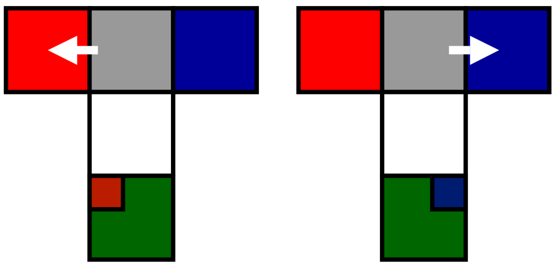


Figure 4.2. A regular T-maze with a corridor length of 3. The optimal action at the T-junction is indicated by the arrow for each signal condition.

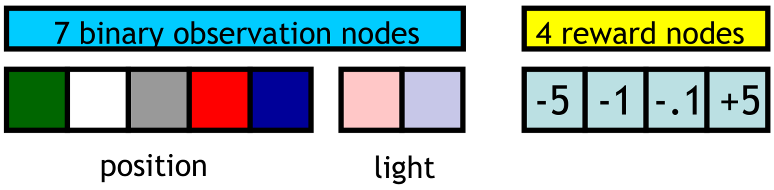


Figure 4.3. Bit coding of the position, signal, and reward signals

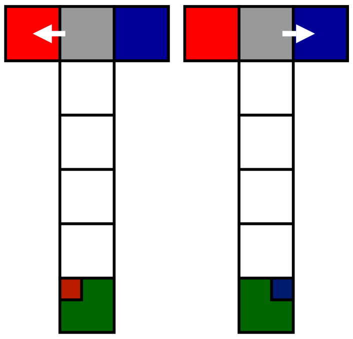


Figure 4.4. A regular T-maze with the corridor length of 6.

At each time step, an agent simultaneously receives the information about its position and a signal specifying the rewarding goal position. There are five types of information related to the position: (1) start position, (2) middle of the corridor, (3) T-junction, (4) left goal, and (5) right goal. The signals are only observed at the start position. Other than these positions, the agent receives no signal. As a result, all observations are represented by 7 binary nodes (5 for the position and 2 for the signal).

An episode ends either when the agent steps into the goal states or after the number of action selections exceeds the maximum number of steps allowed in a single episode. If the initial signal is on the right (or East on the figure), the agent receives a reward of +5 at the right goal and -5 at the left goal; otherwise, the reward condition is reversed. When the agent hits the wall, the underlying environmental state does not change, and the agent receives a reward of -1. Otherwise, the agent moves in the intended direction and receives a reward of -0.1. Upon each episode, an initial signal is independently and randomly selected and fixed. A position, a signal, and reward are all represented by orthogonal bits.

Parameters	Values	Parameters	Values	Parameters	Values
T_{min}	3	ρ_y	1	$\alpha_{\mathrm{predictor}}$	0.05
T_{max}	10	ρ_r	4	α_{actor}	0.05
N^y	7	c_y	1	β	$0 \rightarrow 10$
N^r	4	c_a	1	γ	0.95
N^m	20	β_m	1	N_{sample}	2000
N^h	20	β_y	1	N	10000
N^a	4	L_{training}	5	N_{epoch}	400

Table 4.1. Parameters for the regular/matching T-maze task.

Parameters

The parameters of the following two experiments in this chapter (regular T-maze task and matching T-maze task) are listed in Table. 4.1. The environment of the T-maze is specified by the following parameters. T_{min} is the minimum number of steps required to reach the goal or, equivalently, the length of the vertical corridor. T_{min} can be changed to create a long T-maze. An example with $T_{\mathrm{min}} = 6$ is shown in Fig. 4.4 ($T_{\mathrm{min}} = 3$ is the default value). T_{max} is the maximum number of steps in an episode ($T_{\mathrm{max}} = 10$ is the default value). All episodes with length T_{max} were prematurely terminated.

For the prediction module, we used 7 observation nodes $\boldsymbol{Y}$ ($N^y = 7$), 4 reward nodes $\boldsymbol{R}$ ($N^r = 4$), and 20 memory nodes $\boldsymbol{M}$ ($N^m = 20$). Error weighting parameters for observation and reward are set as $\rho_y = 1$ and $\rho_r = 4$, respectively. Input scaling parameters (c_y and c_a) and gain parameters (β_m and β_y) are all set to 1. The learning rate of the predictor ($\alpha_{\mathrm{predictor}}$) is fixed to 0.05. The predictor's weights and biases were initialized with a spherical Gaussian with a unit variance.

For the actor module, we additionally used 20 hidden nodes $\boldsymbol{H}$ ($N^h = 20$) and 4 action nodes $\boldsymbol{A}$ ($N^a = 4$). The learning rate of the actor (α_{actor}) is fixed to 0.05. The inverse temperature β is linearly increased from 0 to 10. The discount factor γ is set to 0.95. The actor's weights and biases were initialized with a spherical Gaussian with variance 1.

Training procedure

First, data were collected by randomly exploring the environment for 2000 episodes ($N_{\mathrm{sample}} = 2000$). Each episodic data have a variable step-length between $T_{\min}$ and $T_{\max}$. Data with long sequences were truncated to observe the generalization of the predictor's training. For the training of RNNs, data sequences with a step-length from $T_{\min}$ to $T_{\min} + L_{\mathrm{training}} - 1$ were used, where the parameter L_{training} specifies the maximum step-length of a training sequence. We used $L_{\mathrm{training}} = 5$ and used training data with step-lengths of 3, 4, 5, 6, and 7. Collected data were further balanced to obtain equal numbers of training instances for different step-lengths. 80% of balanced data were used as the training data for RNNs, and the remaining 20% of data were used as the test data. A set of training data with step-length T is collectively denoted as

$$\mathcal{D}^{T}_{\mathrm{train}} \triangleq \{\boldsymbol{y}^{(n)}_{0:T}, \boldsymbol{r}^{(n)}_{1:T}\}^{N_T}_{n=1} ,$$

where N_T is the number of instances in the set. The predictor is trained using BPTT with the pattern update mode. The entire set of training data with variable step-lengths is shown for 400 epochs ($N_{\mathrm{epoch}} = 400$). Finally, an FE-based actor is trained with the TD learning rule. The entire pseudocode is given in Algorithm 1.

Results

In order to show the performance improvement of the predictor, the average weighted absolute prediction errors were plotted in Fig. 4.5. The cross-entropy error, which was used for actual training, was not used to plot the changing prediction error because this value increased rapidly as the predictor's confidence in estimates increased. The average weighted absolute prediction errors is defined as

$$E^{T}_{t}(\boldsymbol{\theta}, \rho_y, \rho_r, \mathcal{D}^{T}_{\mathrm{train}}) = \sum_{n=1}^{N_T} \frac{1}{N_T} E^{\mathrm{abs}}_{t}(\boldsymbol{y}^{(n)}_{0:T}, \boldsymbol{r}^{(n)}_{1:T}, \rho_y, \rho_r; \boldsymbol{\theta}) , \tag{4.7}$$

where

$$E^{\mathrm{abs}}_{t}(\boldsymbol{y}_{0:T}, \boldsymbol{r}_{1:T}, \rho_y, \rho_r; \boldsymbol{\theta}) = \rho_y \sum_{i=1}^{N^y} |y_{i,t} - \hat{y}_{i,t}| + \rho_r \sum_{i=1}^{N^r} |r_{i,t} - \hat{r}_{i,t}| . \tag{4.8}$$

Algorithm 1 Entire training sequence for the one-predictor model.

```
Initialize all weights and biases.
for $n = 1$ to $N_{\text{sample}}$ do
  Collect episodic samples $\{\boldsymbol{y}_t, r_t, \boldsymbol{a}_t\}_{t=0}^{T_n}$ with a random policy.
end for
Create a training dataset $\{\mathcal{D}_{\text{train}}^{T}\}_{T=T_{\min}}^{T_{\max}}$ and a test dataset $\{\mathcal{D}_{\text{test}}^{T}\}_{T=T_{\min}}^{T_{\max}}$.
for $T = T_{\min}$ to $T_{\min} + L_{\text{training}} - 1$ do
  Train a predictor with $\mathcal{D}_{\text{train}}^{T}$ using a BPTT learing rule.
end for
for $n = 1$ to $N_{\text{epoch}}$ do
  Actor's weights are initialized.
  for $m = 1$ to $N$ do
    $t \leftarrow 0$.
    Predict the initial observation with $\hat{\boldsymbol{y}}_t$.
    Get the initial observation $\boldsymbol{y}_t$.
    repeat
      Select action $\boldsymbol{a}_t$.
      Predict the observation and reward with $\hat{\boldsymbol{y}}_{t+1}$ and $\hat{r}_{t+1}$.
      Get the an observation $\boldsymbol{y}_{t+1}$ and a reward $r_{t+1}$.
      Actor's weights are updated with a TD learning rule.
      $t \leftarrow t + 1$
    until The goal is reached or $t = T_{\max}$
  end for
end for
```

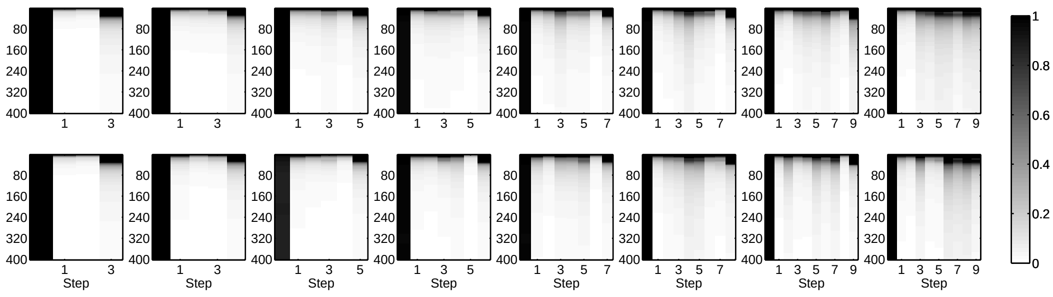


Figure 4.5. Average weighted prediction errors of the joint predictor of observations and rewards in the regular T-maze task. The measure of prediction errors is the absolute error. The top row in the figure shows the error for the training dataset $\mathcal{D}^T_{\text{train}}$, and the bottom row shows the error for the test dataset $\mathcal{D}^T_{\text{test}}$. Each column shows the error for data with variable lengths T. The left-most column used the data with 3 steps to the goal ($T = 3$), and the right-most column used the data with 10 steps ($T = 10$, premature termination). The vertical axis and horizontal axis of each panel indicate the training epoch of RNNs and steps t in a episode, respectively. A prediction error with step 0 implies the prediction error for the initial observation.

is a weighted absolute prediction error.

After training the prediction with $\mathcal{D}^{T}_{\text{train}}$ ($T = 3, 4, 5, 6, 7$), the predictor could estimate the predictable observations and encoded rewards precisely, as shown in Fig. 4.5. The prediction error rapidly decreased in the first few epochs. The learning is well generalized for the test dataset (bottom row in Fig. 4.5) and for longer sequences (three right-hand side panels in Fig. 4.5). The prediction errors for the initial observation (left-most column in each panel in Fig. 4.5) did not decrease as the learning proceeded. This is due to the unpredictability of the signal given at the beginning of the episode.

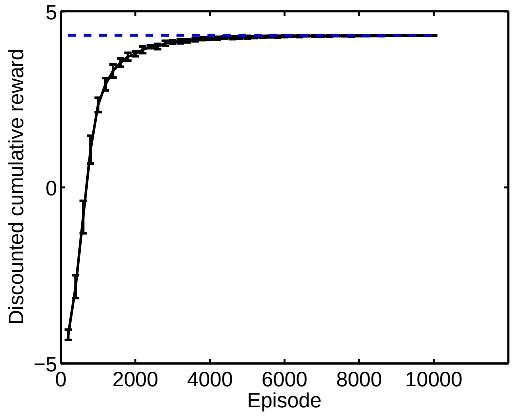
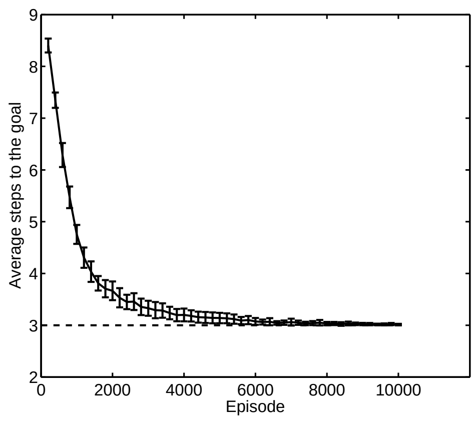
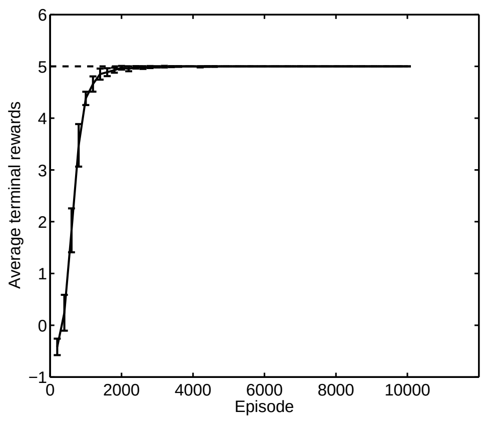


Figure 4.6. Moving average of discounted return in the regular T-maze task with corridor length 3 ($T_{\text{min}} = 3$). The window size is 200 episodes. Error bars in (a) show the standard error of the mean (s.e.m.) over 10 runs. The upper bound of the discounted return is indicated by the dotted line ($R_0 = 4.3175$ with $\gamma = 0.95$).

After the training of the predictor, the predictor's weights were fixed. Then, the actor was trained using SARSA(0). Because the correct decision at the T-junction is the only source of a positive reward (+5 terminal reward), the positive return itself ensures that an agent successfully solved the given POMDP task. A moving average of the discounted return over 10 runs in Fig. 4.6 indicates that this POMDP task is solved optimally. Fig. 4.7(a) and (b) show that the agent not only learned to reach the goal with optimal 3 steps but also learned to reach the correct goals in all 10 runs. Fig. 4.8(a) shows that the TD error decreases over the episodes in all runs. A closer look at the first run (Fig. 4.8(b)) reveals

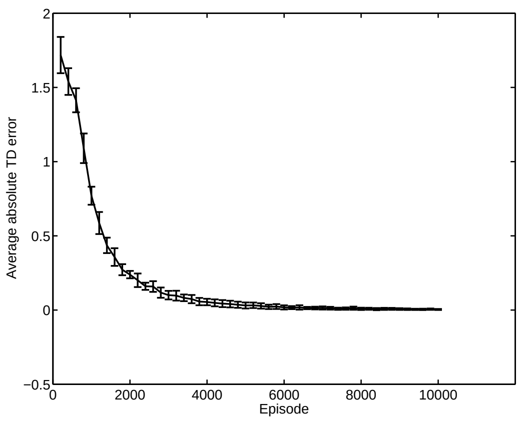


(a) Steps to the goal.

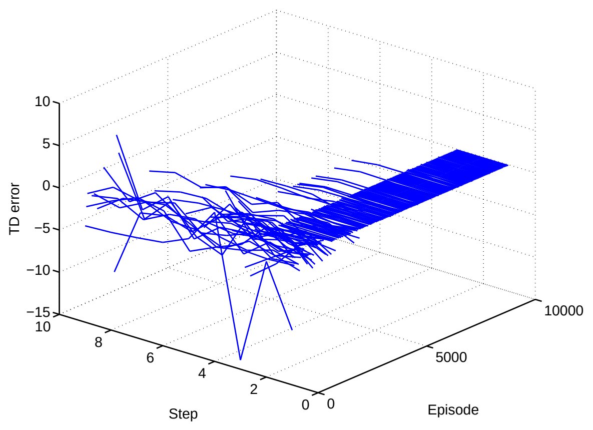


(b) Terminal rewards.

Figure 4.7. Performance measured in (a) the number of steps to the goal and (b) terminal rewards. The window size of the moving average is 200 episodes. Each measure is averaged over 10 runs. The optimal performance levels are indicated by a dotted line.



(a) Average over 10 runs



(b) Individual runs

Figure 4.8. Absolute TD error (a) Moving average of the mean absolute TD error ($T_{\min} = 3$). The episodic mean is taken first, and then, the moving average is calculated. The window size is 200 episodes. Error bars show the standard error of the mean (s.e.m.). (b) TD errors in the 1st run. Data are sparsely plotted every 50 episode starting from the first episode.

how stepwise TD error decreased as the learning proceeded. We can see a clear correlation between the magnitude of TD errors and the number of steps to the goal.

4.3.2 Matching T-maze task

Task specification

The matching T-maze task is a simple extension of a regular T-maze task (Bakker, 2002). In order to act optimally, an agent needs to use both the memory and the immediate observation. This new task is specifically designed to investigate how an agent integrates an observation and task-relevant memory traces on the fly.

Unlike the regular T-maze task, the signals are observed at the start position and the T-junction. The agent receives no signal at the other three positions.

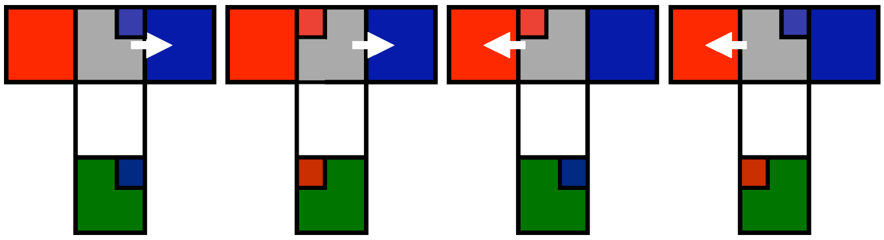


Figure 4.9. Matching T-maze task. Arrows indicate the optimal action at the T-junction.

An episode ends either when the agent steps into the goal states or after the number of action selections exceeds the maximum steps allowed in a single episode. If two signals at each end of the corridor are the same, the agent receives a reward of $+5$ at the right goal and -5 at the left goal; if two signals at each end of the corridor are not the same, the reward condition is reversed. When the agent hits the wall, the underlying environmental state does not change, and the agent receives a reward of -1. Otherwise, the agent receives a reward of -0.1. Upon each episode, two signals are independently and randomly selected and fixed. A position, a signal, and reward are all represented by orthogonal bits.

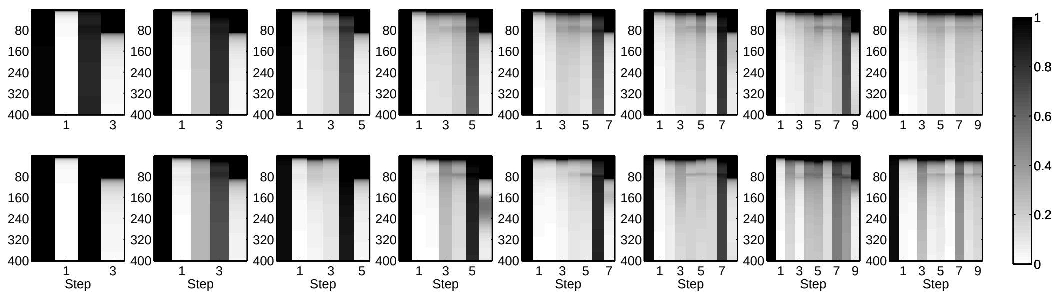


Figure 4.10. Average weighted prediction errors of the joint predictor of observations and rewards in the matching T-maze task. Refer to the caption of Fig. 4.5 for reading the figure.

Parameters

We used 7 observation nodes $\boldsymbol{Y}$, 4 reward nodes $\boldsymbol{R}$, 20 memory nodes $\boldsymbol{M}$ for the predictor module, and 20 hidden nodes $\boldsymbol{H}$.

Basic performance

As shown in Fig. 4.10, predictable observations and rewards are well estimated by the joint predictor after training with $\mathcal{D}^{T}_{\text{train}}$ ($T = 3, 4, 5, 6, 7$). The decrease in the prediction error occurred at the initial prediction, and then, it gradually propagated from shorter steps to longer steps in all conditions. The prediction errors for the last step, whose minimization is crucial for the task, always occurred after the prediction errors for the leading steps decreased. The learning was well generalized for the test dataset (bottom row in Fig. 4.10) and for longer sequences (three right-hand side panels in Fig. 4.10).

Prediction errors for $t = 0$ (pre-execution step) and $t = T - 1$ (step immediately before entering the goal) are high for all successfully terminated episodes ($T = 3, \ldots, 9$) due to the unpredictability of two signals. Interestingly, the prediction errors decrease as T increases for pre-terminal observations at $t = T - 1$. This is due to the increasing number of episodes with predictable pre-terminal observation. The pre-terminal observation becomes predictable if an agent hits the north-wall before entering the goal. Due to the difficulty of predicting the final reward (this requires the initial signal and the signal at the T-junction to

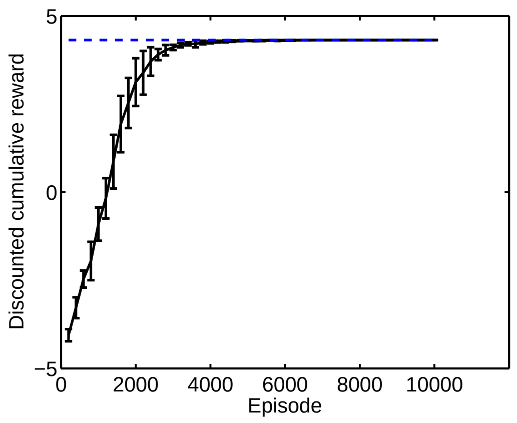
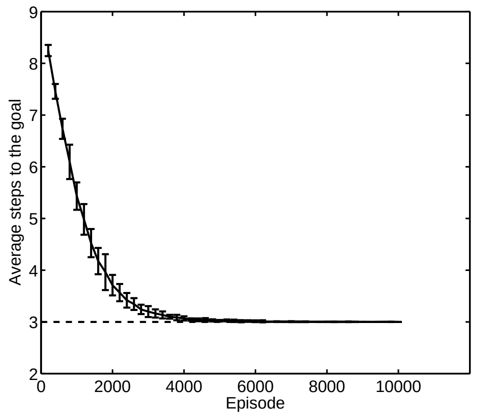


Figure 4.11. Moving average of discounted return in the matching T-maze task ($T_{\text{min}} = 3$). The window size is 200 episodes. Error bars show the standard error of the mean (s.e.m.). The upper bound of the discounted return is indicated by the dotted line ($R_0 = 4.3175$ with $\gamma = 0.95$).

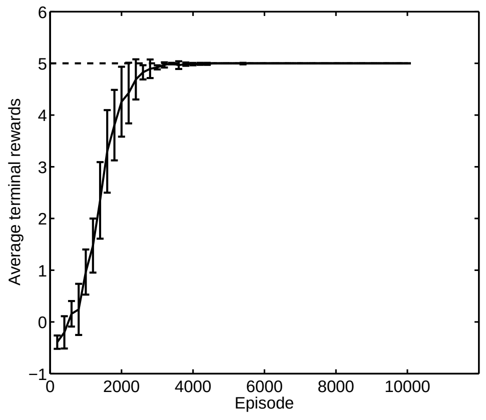
be combined), the prediction error for the terminal reward is high as compared to the previously obtained rewards. However, the prediction error for the terminal reward still decreased gradually over epochs. A cascading improvement of prediction performance from the early steps (small t) to later steps (large t) is also seen in all panels. This indicates the requirement of accurate prediction of a shorter sequence before predicting the longer sequence.

After training the predictor, its weights were fixed. Then, the actor was trained using SARSA(0). A moving average of the discounted return over 10 runs is shown in Fig. 4.11. It indicates that this POMDP task was solved optimally in all runs. This result is further confirmed with different performance measures: average number of steps to the goal (Fig. 4.12(a)) and average terminal reward (Fig. 4.12(b)).

Fig. 4.13(a) shows that the TD error decreases to zero in all runs. A closer look at the first run (Fig. 4.13(b)) reveals how the stepwise TD error decreases as the learning proceeds. We can see a clear correlation between the magnitude of TD errors and the number of steps to the goal. In addition, the TD error is always large at the last step in the early episodes due to the difficulty for predicting the final reward.

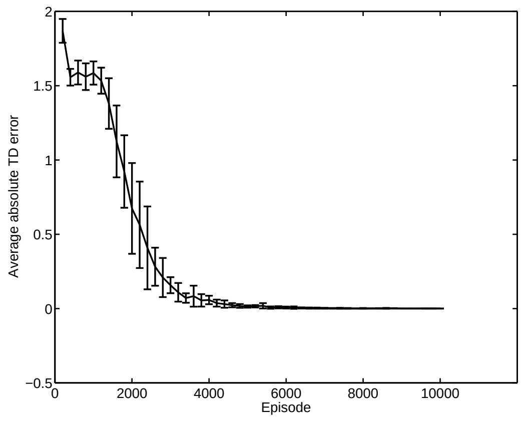


(a) Steps to the goal.

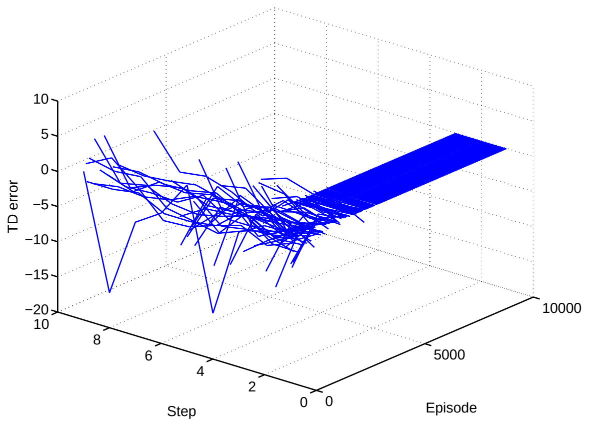


(b) Terminal rewards.

Figure 4.12. Performance measured in (a) the number of steps to the goal and (b) terminal rewards. The window size of the moving average is 200 episodes. Each measure is averaged over 10 runs. The optimal performance levels are indicated by a dotted line.



(a) Average over 10 runs

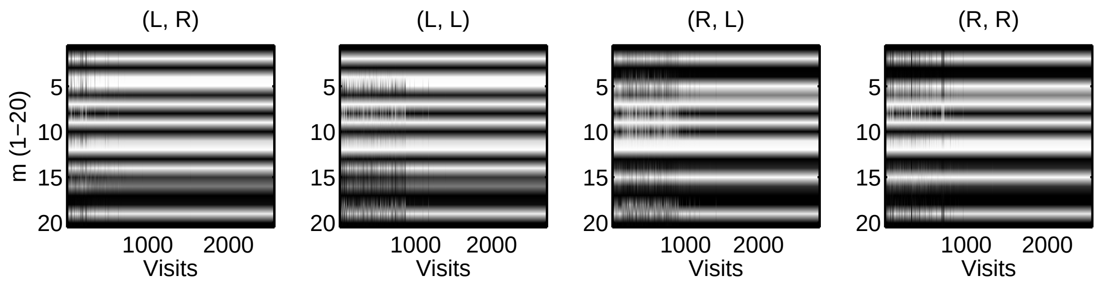


(b) Run 1

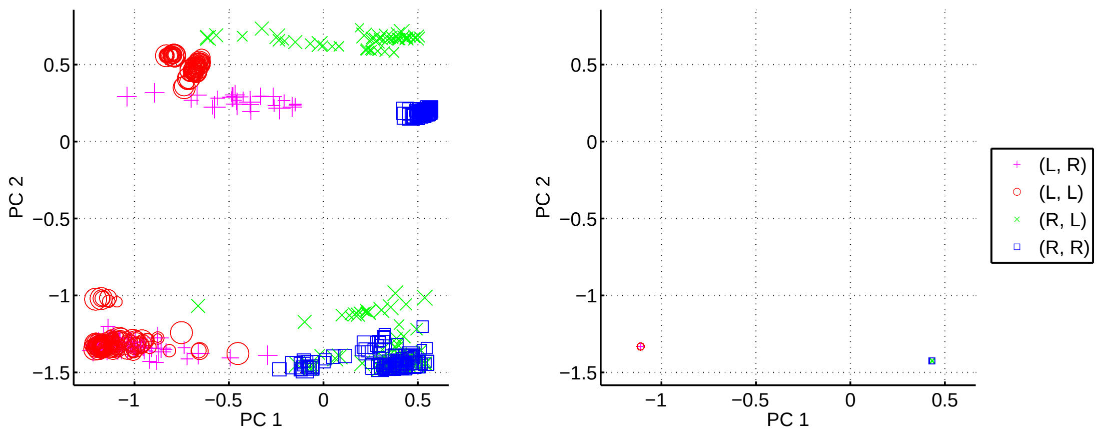
Figure 4.13. TD errors. (a) Mean absolute TD error. The window size of the moving average is 200 episodes. Error bars show the standard error of the mean (s.e.m.) over 10 runs. (b) TD errors in the 1st run. Data are sparsely plotted every 50 episode starting from the first episode.

Internal representations

We are interested in how sequentially collected observations are encoded and maintained in the agent throughout the decision making process. In this section, we analyze the activation patterns of the functional module in the agent.



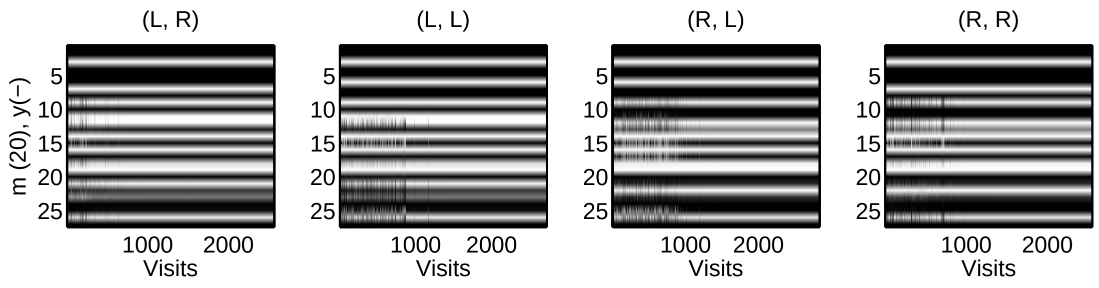
(a) All activations (initial signal, T-junction signal).



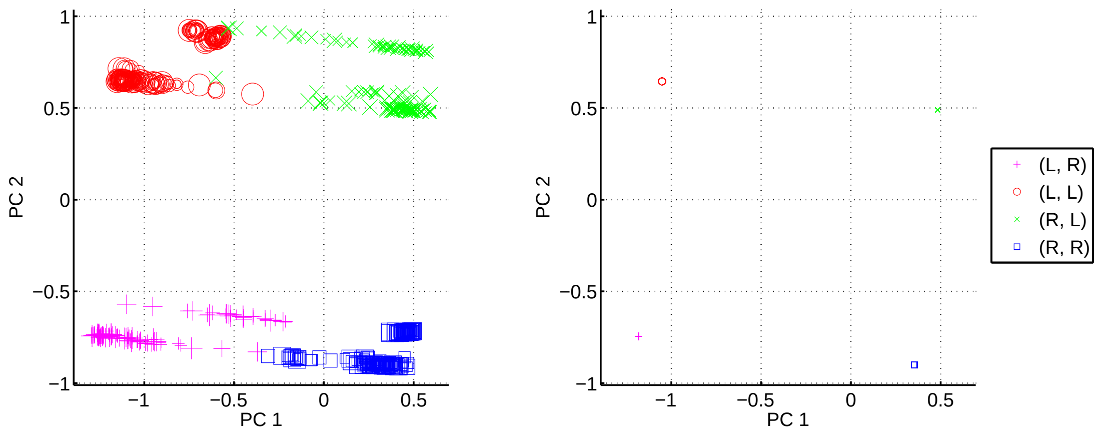
(b) PCA analysis: (Left) first 1000 episodes. (Right) last 1000 episodes.

Figure 4.14. Activation patterns of the predictor's memory nodes at the T-junction. These activations were used to predict the obtained rewards at the T-junction before entering the T-junction. (a) Activations for all visits to the T-junction are shown according to the four possible signal conditions (initial signal, T-junction signal). (b) PCA analysis of these activations. The size of the marker reflects the the number of steps to the goals. The smallest marker indicates 3 steps, and the largest marker indicates 10 steps.

First, we focus on the memory layers of the predictor. In order to predict the terminal reward correctly, the information about the initial signal should be



(a) All activations (initial signal, T-junction signal).



(b) PCA analysis: (Left) first 1000 episodes. (Right) last 1000 episodes.

Figure 4.15. Activation patterns of the actor's state nodes at the T-junction. These activations were used to select the next action at the T-junction.

encoded and maintained in the activation patterns of this layer. Fig. 4.14 suggests that the memory of initial signal is encoded and maintained in the predictor's memory layer when the agent reaches the T-junction. The activation patterns at the T-junction differ depending on the initial signal even before the learning starts (Fig. 4.14(a)). These trends are further confirmed by the PCA analysis in Fig. 4.14(b). It shows that, during both the first and the last 1000 episodes, the variations in the activations are mostly explained by the difference in the initial signal.

Next, the activation patterns of the actor's *state* nodes at the T-junction are investigated (Fig. 4.15). The actor's state is composed of 7-dimensional observation ($\boldsymbol{y}$) and 20-dimensional predictor's memory layer ($\boldsymbol{m}$). The PCA analysis

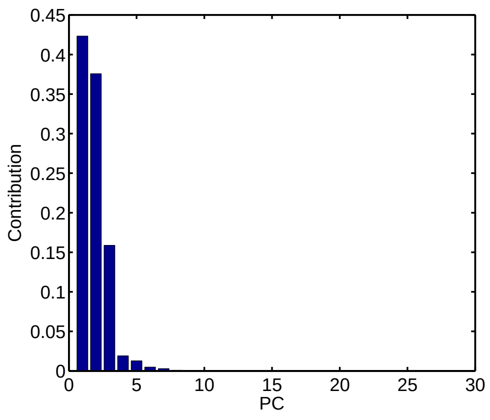
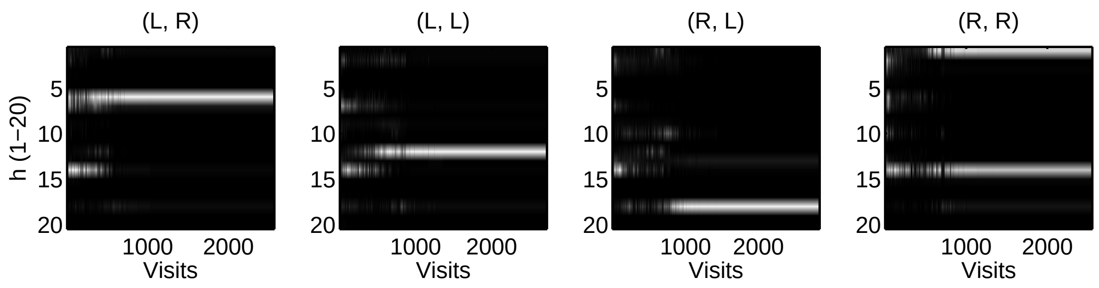


Figure 4.16. Contributions of each principal component (actor's state nodes).

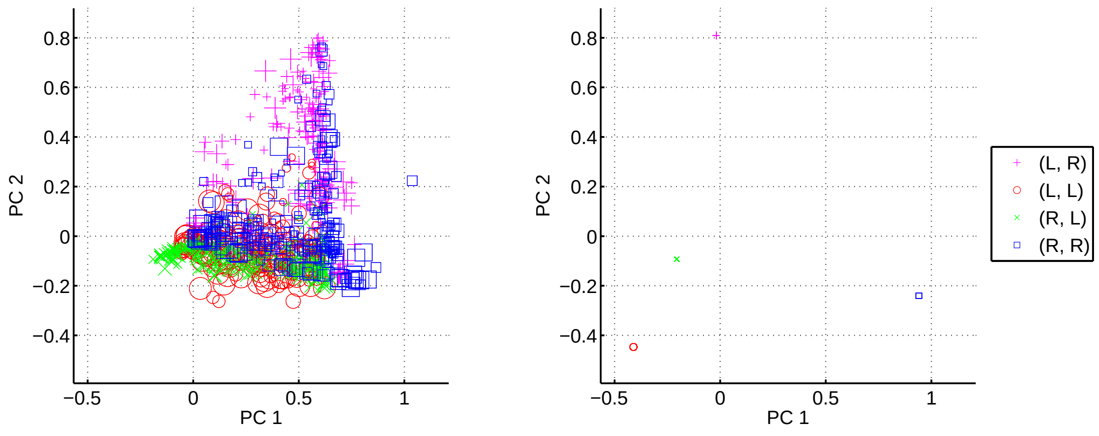
shown in Fig. 4.15 revealed that four conditions are separately represented in state nodes. Further analysis revealed that most of the variability in the actor's state nodes can be captured by the first two principal components (Fig. 4.16).

Finally, activation patterns of the actor's hidden nodes at the T-junction are investigated (Fig. 4.19). For each task condition, distinct firing patterns appear as the training proceeds (Fig. 4.17(a)). This trend is further confirmed in the PCA analysis of these activations. In Fig. 4.19, activations during the first and the last 1000 visits to the T-junction are projected onto the space spanned by the first two principal components. This figure shows the separation of firing patterns depending on the task condition. The dynamic separation in the learning process is clearly seen in Fig. 4.18 (Fig. 4.19 shows the first and the last 1000 episodes of Fig. 4.18). It reveals the gradual convergence of firing patterns depending on the task condition.

Further analysis revealed that most of the variability in the actor's hidden nodes can be captured by the first three principal components (Fig. 4.19(a)). The projection on this three-dimensional space revealed that the firing patterns for each task condition were not only separated in the three-dimensional space but were also orthogonalized (Fig. 4.19(b)).



(a) All activations (initial signal, T-junction signal).



(b) PCA analysis: (Left) first 1000 episodes. (Right) last 1000 episodes.

Figure 4.17. Activation patterns of the actor's hidden nodes at the T-junction. These activations were realized after selecting the action at the T-junction.

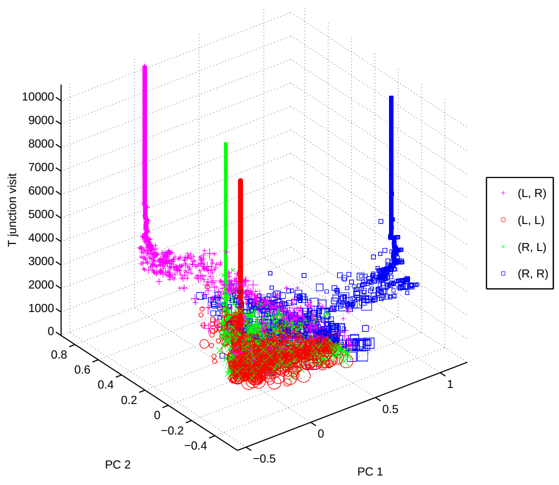
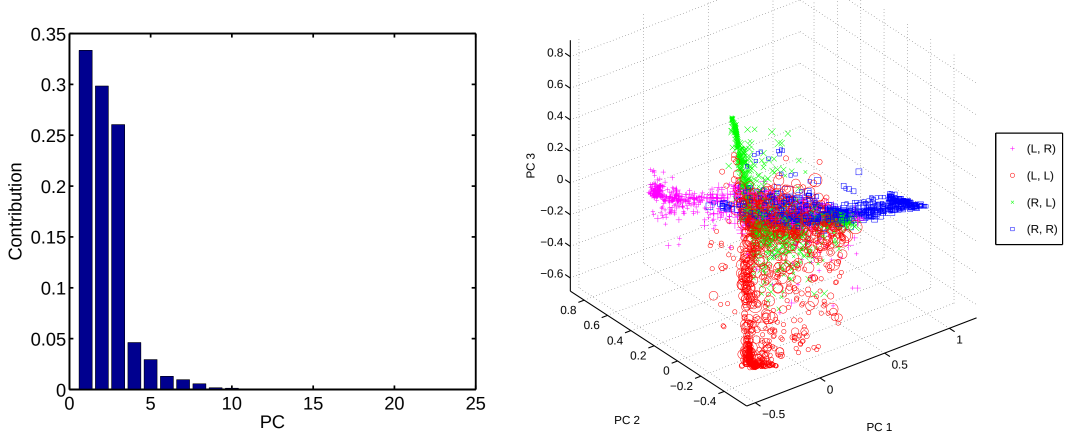


Figure 4.18. Changing activation patterns of the actor's hidden nodes at the T-junction.



(a) Contribution of each principal component

(b) Projection of all activations in the first three principal components

Figure 4.19. Analysis of the activation patterns of the actor's hidden nodes.

Parameters	Values	Parameters	Values	Parameters	Values
T_{min}	3	ρ_y	1	$\alpha_{\text{predictor}}$	0.05
T_{max}	10	ρ_r	4	α_{actor}	0.05
N^y	784	c_y	0.0089	β	$0 \rightarrow 10$
N^r	4	c_a	1	γ	0.95
N^m	20	β_m	1	N_{sample}	2000
N^h	20	β_y	1	N	10000
N^a	4	L_{training}	5	N_{epoch}	400

Table 4.2. Parameters for the digit matching T-maze task.

4.3.3 Digit matching T-maze task

The task was further extended to the digit matching T-maze task in order to test the proposed model's ability to handle high-dimensional sensory inputs. In the digit matching T-maze task, instead of orthogonal bit codes of the position and signal, 784-dimensional binarized images of handwritten digits are given to the agent as sensory inputs (Fig. 4.20). All the other task settings are the same as those in the matching T-maze task described in Section 4.3.2.

Here, we only focus on the ability of the proposed architecture to handle high-dimensional observations; therefore, observations are not stochastic but static. We only use a single instance of handwritten digits for each digit class shown in Fig. 4.20.[3]

The parameters used for the digit matching T-maze task are listed in Table. 4.2. Input scaling parameters are set to $c_y = 0.0089$ $(= 7/784)$ and $c_a = 1$ to balance each layer's contribution on the memory update. The prediction error weights are set as $\rho_y = 1$ and $\rho_r = 4$ in order to reflect the importance of the reward prediction in the given task. All the other parameters are the same as those used in the previous experiments.

The performance of the agent remained suboptimal as shown in Fig. 4.21(a); however, the agent still showed the tendency to select the correct goal as shown in Fig. 4.21(b). It indicates that the information about the initial signal was at least retained in the predictor's hidden nodes.

[3]This restriction is removed in the next chapter.

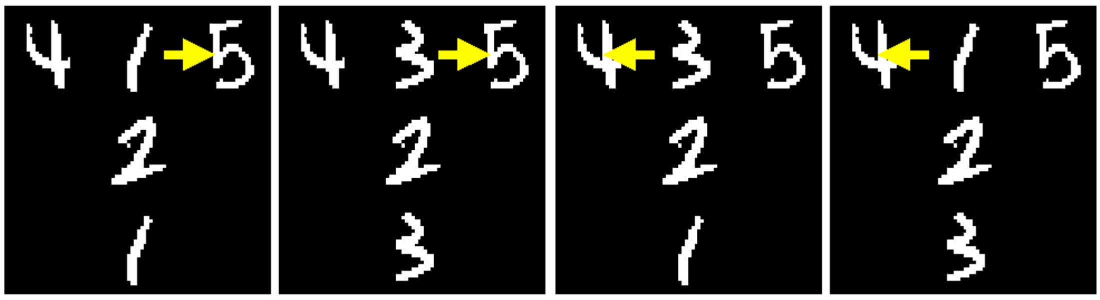
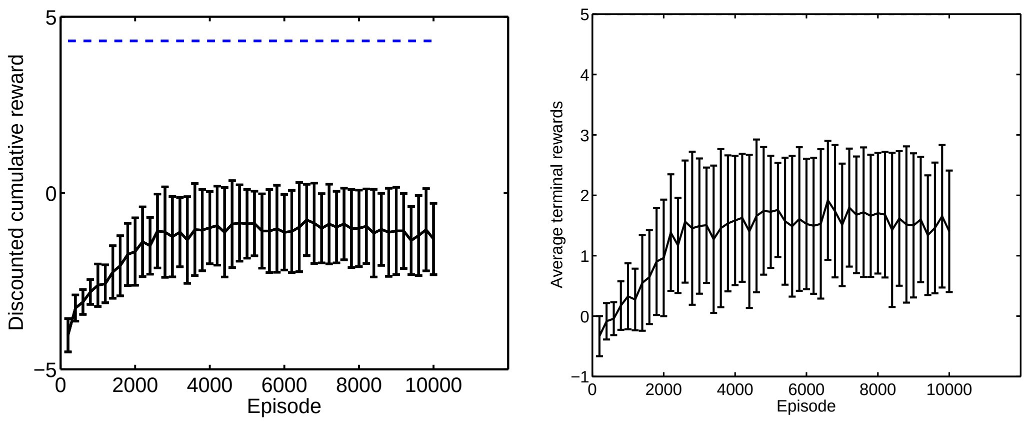


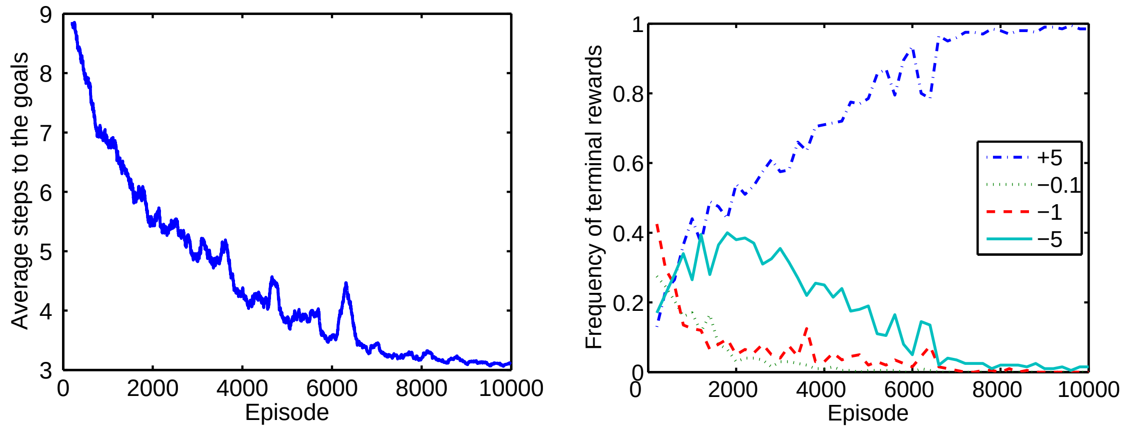
Figure 4.20. Digit matching T-maze task. The optimal action at the T-junction is indicated by arrows.



(a) Discounted return.

(b) Terminal reward.

Figure 4.21. Performance of the predictor-actor model in the digit matching T-maze task.



(a) Steps to the goal.　　(b) A predictor-actor

Figure 4.22. Performance of the predictor-actor model in digit T-maze task. (a) Moving average of number of steps to the goals over the past 200 episodes. (b) Frequency of terminal rewards in each 200-episode block.

Fig. 4.22 shows one successful run in the digit matching T-maze task. This success in a particular run is encouraging because the proposed architecture can potentially handle POMDP problems with high-dimensional observations. However, its performance is fairly unstable in different runs. In addition, its performance strongly depends on the tuning of parameters such as prediction error weights (ρ_y nad ρ_r) and input scaling parameters (c_y and c_a).

4.4 Summary and discussion

In this chapter, we proposed a new method to solve POMDP problems without a prior knowledge of an environment. Here, neither the state transition probability nor the true set of underlying states is given *a priori*. The agent selected the action solely based on the current observation and its own internal memory. Unlike the original recurrent-model architecture, which used a three-layer feedforward neural network to approximate state-action values, we employed an RBM-based actor.

The proposed method successfully solved two variants of T-maze tasks: regular T-maze task and matching T-maze task, which are listed in the order of increasing task complexity. There were a few success in the digit matching T-maze task; however these were too weak to conclude that this task can be solved with the proposed architecture.

In our simulation, the predictor estimated both the observation and the encoded reward for simplicity. However, this simplification introduced the complication of balancing these two terms with weighting parameters ρ_y and ρ_r. Observations and rewards should probably be predicted separately using different memory layers. With this separation, we can estimate a scalar reward directly instead of a binarized reward. The characteristics of FERL to handle different modalities may help to use the different memory layers jointly with observations.

The prediction of observation can also be improved using a preprocessing mechanism. This preprocessing might be achieved using a pre-trained deep belief network (Hinton, 2002) or a deep Boltzmann machine (Salakhutdinov and Hinton, 2009). By first extracting statistical regularities in the raw sensory data, an actor can use these extracted features as sensory inputs instead of raw sensory inputs. In addition, the predictor can predict these processed features instead of raw sensory data. This preprocessing may improve the robustness to noise and observation variabilities. In addition, this approach might also boost the learning speed if we use a smaller number of hidden nodes in the top hidden layer than the one of observation nodes. These advancements are likely to serve as building blocks of truly autonomous robots.

Chapter 5

Solving POMDPs with High-Dimensional Noisy Observations

5.1 Introduction

In the previous chapter, we introduced a novel architecture that can solve POMDP problems with unknown dynamics. Although the proposed architecture could solve POMDP problems with low-dimensional bits (regular and matching T-maze task), the performance in problems with high-dimensional observations (digit matching-T-maze task) was not satisfactory given the proven ability of the free-energy-based agent to handle noisy high-dimensional data in the MDPs (Chapter. 3).

In this chapter, we modify the architecture in order to improve the performance by incorporating the results obtained in the previous chapter. The key features in the improved architecture are (1) the use of RBMs for preprocessing high-dimensional noisy observations, (2) the use of separate predictors for observation and reward, (3) the use of a raw scalar reward instead of an encoded reward, and (4) the removal of unwanted parameters. As a result, POMDPs with high-dimensional noisy observations were solved without any prior knowledge of environmental dynamics.

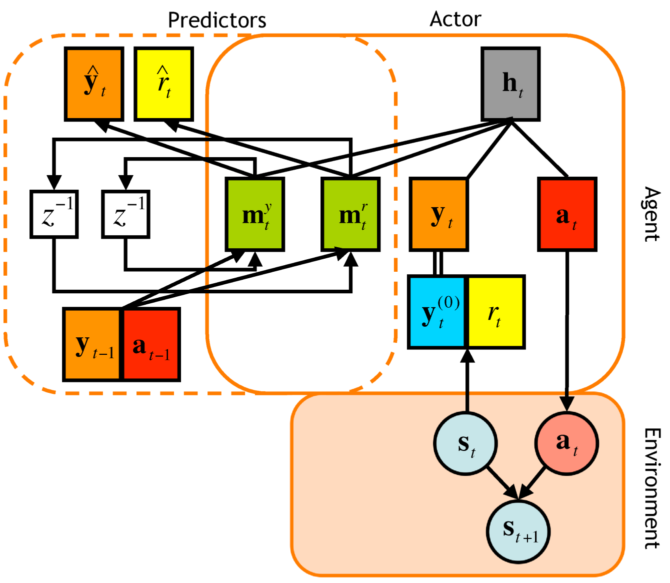


Figure 5.1. Improved predictor-actor architecture. This architecture includes an RBM–based observation preprocessor and separate predictors for processed observations and scalar rewards.

5.2 Model architecture

The improved recurrent-model architecture for solving POMDP problems with high-dimensional observations is shown in Fig. 5.1. As in the previous model (Fig. 4.1(b)), an agent is composed of two overlapping modules: a predictor and an actor. However, each module is reconstructed to overcome the limitations posed in the previous chapter. The major changes in the architecture are the introduction of observation preprocessing and the separation of predictors. The effects of these changes are further elaborated below.

Observation preprocessing In the improved architecture, observations are preprocessed using RBMs. In Fig. 5.1, the preprocessing module is represented by two boxes ($\boldsymbol{y}_t^{(0)}$ and $\boldsymbol{y}_t$) connected by a double line. Observation preprocessing affords many benefits.

The first is the reduction in the dimensionality of high-dimensional observations. As long as the same amount of information is preserved, a low-dimensional representation is less noisy and becomes a better target for the predictor than a high-dimensional representation.

The second is that the extracted codes or features can be used as a more useful representation than raw high-dimensional observations. RBMs not only learn the generative model of observations but also code the observations cohesively according to the observation distribution. In other words, observations with similar statistical characteristics are closely mapped in the feature space, and those with different statistical properties are orthogonally mapped in the feature space. Unlike mixture-based generative models (e.g., mixture of Gaussian, HMM, etc.), RBMs realize a distributed representation of data.

Third, these extracted codes are robust to noise. These three benefits make the processed observations ideal targets for the predictor. Given the same data, a low-dimensional denoised meaningful target is much easier to predict than a high-dimensional noisy target.

Finally, the freedom of choosing the number of hidden units in RBMs can be utilized to balance the importance between the observation and the memory. For example, in the architecture used in the previous chapter, for the agent seeing pixel-images, the memory is updated by 784-dimensional observation nodes, 20-dimensional memory nodes, and 4-dimensional action nodes. In order to alleviate this unbalanced contribution to the memory update, the input scaling parameters (c_y and c_a) have been introduced thus far. Instead of tweaking these parameters, we can control the importance between the observation and the memory on the memory update by setting the number of RBMs' hidden nodes and predictor's memory nodes accordingly.

Separation of predictors In the new model, the observation and reward are predicted by different predictors. This change allows the predictors to use different types of output nodes, or equivalently, different types of prediction error functions. In the previous chapter, a reward was encoded in a binary vector due to the requirement of using one type of error function in a single agent. In the new architecture, we do not need this limitation. Therefore, the error function for each predictor can be independently considered. For example, we can choose a cross entropy error function for binary processed observation nodes

$$E_t^y = -\sum_{i=1}^{N^y} \left[y_{i,t} \log \hat{y}_{i,t} + \{1 - y_{i,t}\} \log \{1 - \hat{y}_{i,t}\} \right] \ , \qquad t = 0, \ldots, T_{\text{epi}} \ , \tag{5.1}$$

and a square error function for linear reward nodes

$$E_t^r = \frac{1}{2}(r_t - \hat{r}_t)^2 \ , \qquad\qquad t = 1, \ldots, T_{\text{epi}} \ . \tag{5.2}$$

Note that the error weighting parameters (ρ_y and ρ_r in (4.6)), which made the predictor's learning rule complicated in the previous chapter, were dropped. In addition, the artificial binarization of reward is no longer necessary.

The separation of the predictors yield two memory layers. An actor uses both memory layers for decision making. In the new architecture, the actor's *state* is composed of a processed observation layer and two memory layers provided by the predictors.

Training procedure

The improved architecture has three learning phases. In the first learning phase, the preprocessor is trained using contrastive divergence. In the second phase, a predictor is trained with randomly collected data. In the third phase, an FE-based actor is trained using the TD learning rule. The pseudocode for the entire training sequence is given in Algorithm 2.

5.3 Experiments

5.3.1 Matching T-maze task

The same matching T-maze task as that described in Section 4.3.2 is solved using the new architecture.

Parameters

Table. 5.1 lists all parameters used for the two experiments described in this chapter: matching T-maze task and digit matching T-maze task. Several parameters

Algorithm 2 Entire training sequence for the improved model.

```
Initialize all weights and biases.
if Preprocessor is used then
  Train the preprocessor using a CD learning rule.
end if
for $n = 1$ to $N_{\text{sample}}$ do
  Collect episodic samples $\{\boldsymbol{y}_t, r_t, \boldsymbol{a}_t\}_{t=0}^{T_n}$ with a random policy.
end for
Create a training dataset $\{\mathcal{D}_{\text{train}}^T\}_{T=T_{\min}}^{T_{\max}}$ and a test dataset $\{\mathcal{D}_{\text{test}}^T\}_{T=T_{\min}}^{T_{\max}}$.
for $T = T_{\min}$ to $T_{\min} + L_{\text{training}} - 1$ do
  Train a predictor with $\mathcal{D}_{\text{train}}^T$ using a BPTT learing rule.
end for
for $n = 1$ to $N_{\text{epoch}}$ do
  Actor's weights are initialized.
  for $m = 1$ to $N$ do
    $t \leftarrow 0$.
    Predict the initial (processed-)observation with $\hat{\boldsymbol{y}}_t$.
    Get the initial observation $\boldsymbol{y}_t$.
    if Preprocessor is used then
      Process $\boldsymbol{y}_t$.
    end if
    repeat
      Select action $\boldsymbol{a}_t$.
      Predict the (processed-)observation and reward with $\hat{\boldsymbol{y}}_{t+1}$ and $\hat{r}_{t+1}$.
      Get the an observation $\boldsymbol{y}_{t+1}$ and a reward $r_{t+1}$.
      if Preprocessor is used then
        Process $\boldsymbol{y}_t$.
      end if
      Actor's weights are updated with a TD learning rule.
      $t \leftarrow 0$.
    until The goal is reached or $t = T_{\max}$
  end for
end for
```

Parameters	Values	Parameters	Values	Parameters	Values
$T_{\min}$	3	ρ_y	N/A	$\alpha_{\text{predictor}}$	0.05
$T_{\max}$	10	ρ_r	N/A	α_{actor}	0.05
N^y	7/20	c_y	N/A	β	$0 \rightarrow 10$
N^r	1	c_a	N/A	γ	0.95
N^m	N/A	β_m	N/A	N_{sample}	2000
N^h	20	β_y	N/A	N	10000
N^a	4	L_{training}	5	N_{epoch}	400/600
N^{m_y}	20	N^{m_r}	20		

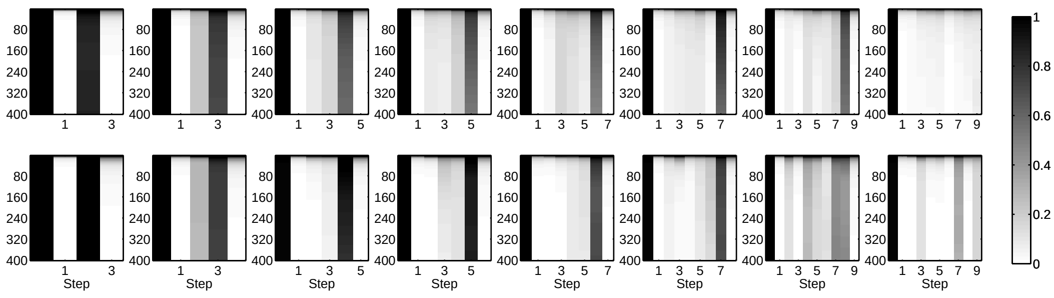
Table 5.1. Parameters for the matching T-maze task (bit/pixel observations). N/A indicates associated parameters are removed in the improved architecture. The newly included parameters are added at the bottom row.

were removed in the new architecture. These are marked with N/A in Table. 5.1. The added parameters are the number of observation-predictor's hidden nodes N^{m_y} and the number of reward-predictor's hidden nodes N^{m_r}. They were both set to 20. For the architecture using an RBM-based observation preprocessor, the number of RBM's hidden nodes are denoted by N^y. In this case, $N^{y^{(0)}}$ is used as the number of raw observation nodes.

Basic performance

As shown in Fig. 5.2, predictable observations and rewards are well estimated by two distinct predictors after training with $\mathcal{D}^T_{\text{train}}$ ($T = 3, 4, 5, 6, 7$). The prediction error rapidly decreased in the first few epochs. The learning was well generalized for the test dataset (bottom row in Fig. 5.2(a) and Fig. 5.2(b)) and for longer sequences (three right-hand side panels in Fig. 5.2(a) and Fig. 5.2(b)).

The performance of the processed-observation predictor is shown in Fig. 5.2(a). Prediction errors for the initial observation ($t = 0$) and for the observation immediately before entering the goal ($t = T-1$) are high for all successfully terminated episodes ($T = 3, \dots, 9$) due to the unpredictability of two signals. Interestingly, the prediction error decreases as T increases for the pre-terminal observation at $t = T - 1$. This is due to the increasing number of episodes with predictable


(a) Observation prediction error

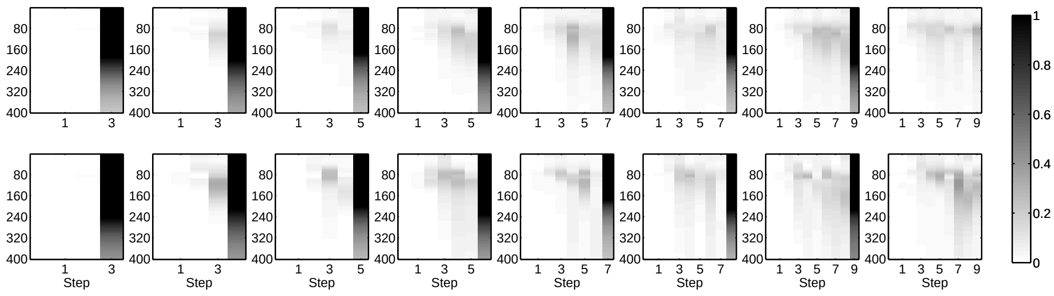
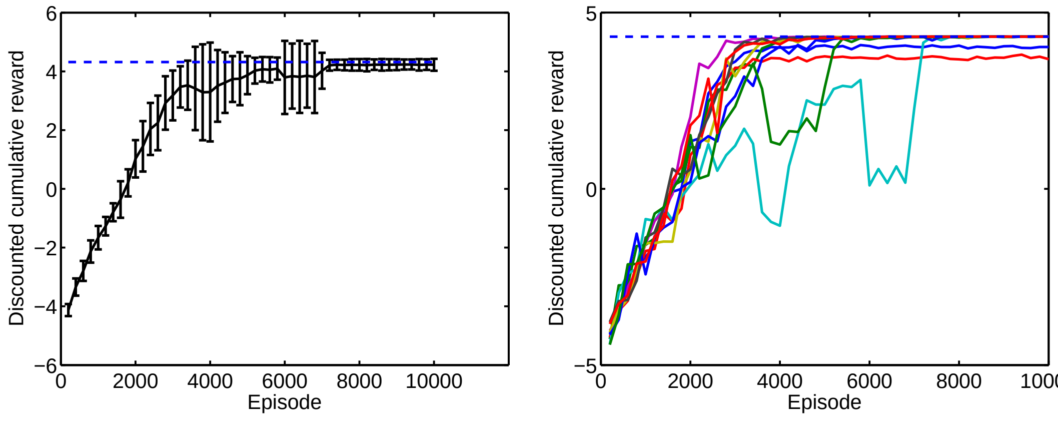

(b) Reward prediction error

Figure 5.2. Average prediction errors of (a) processed-observation predictor and (b) scalar-reward predictor. The measure of prediction errors are absolute error for (a) and squared error for (b). Refer to the caption of Fig. 4.5 for reading the figure.

pre-terminal observations. The pre-terminal observation becomes predictable if an agent hits the north-wall before entering the goal.

The performance of the scalar-reward predictor is shown in Fig. 5.2(b). Due to the difficulty of predicting the final reward (this requires the combination of the initial signal and the signal at the T-junction), the prediction error for the terminal reward is high as compared to the previously obtained rewards. However, the prediction error for the terminal reward still decreased gradually over epochs. A cascading improvement of prediction performance from earlier steps (smaller t's) to later steps (larger t's) is also seen in all panels other than $T = 3$. This indicates the requirement of accurate prediction of shorter sequence before predicting the longer sequence.



(a) Average over 10 runs (b) Individual runs

Figure 5.3. Moving average of discounted return in the matching T-maze task ($T_{\text{min}} = 3$). The window size is 200 episodes. Error bars in (a) show the standard error of the mean (s.e.m.). The upper bound of the discounted return is indicated by the dotted line ($R_0 = 4.3175$ with $\gamma = 0.95$).

After the training of the predictor, the predictor's weights were fixed. Then, the actor was trained using SARSA(0). The moving average of the discounted return over 10 runs shown in Fig. 5.3(a) indicates that this POMDP task was solved near-optimally. The large variance in the middle of the episodes and the remaining variance at the end in Fig. 5.3(a) are both explained by investigating the individual runs. Fig. 5.3(b) reveals that the agent deviated from a smooth

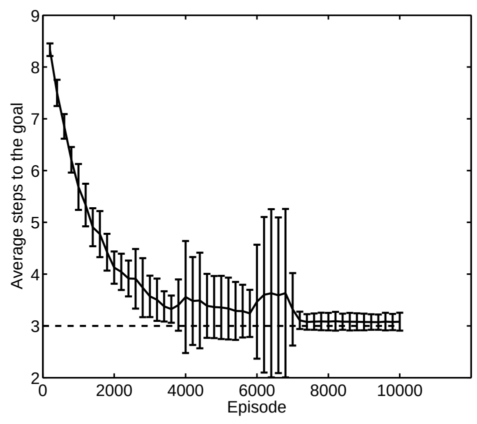


Figure 5.4. Average number of steps to the goal over 10 runs ($T_{\min} = 3$)

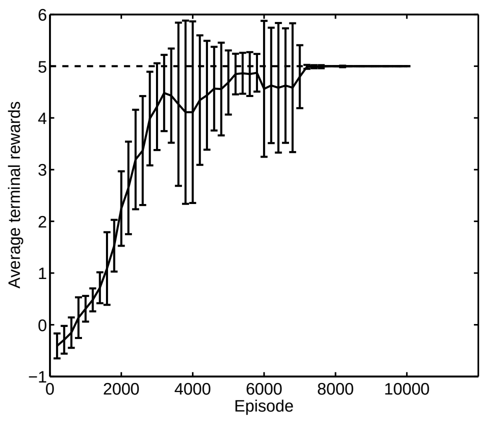
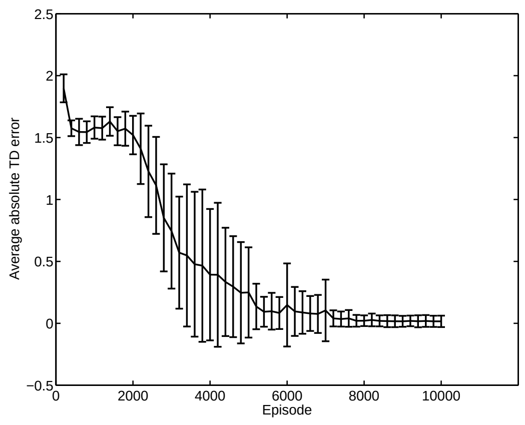


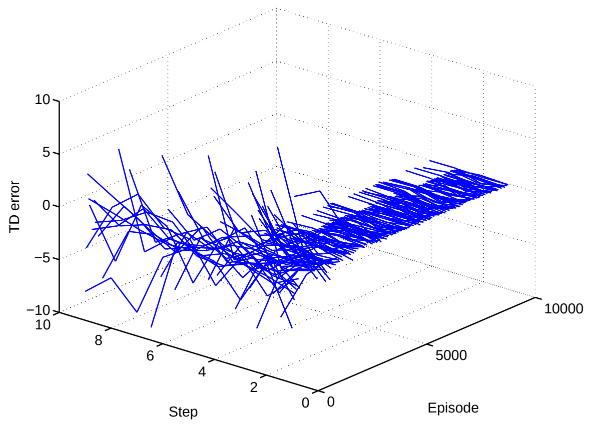
Figure 5.5. Average terminal reward over 10 runs ($T_{\min} = 3$)

performance improvement in 2 of 10 runs, which contributed to the large variance in the middle of the episodes. In addition, two other runs converged to a sub-optimal performance. In these runs, the agent successfully discriminated the correct goal at the T-junction (Fig. 5.5); however, it took redundant steps to reach the goal (Fig. 5.4).

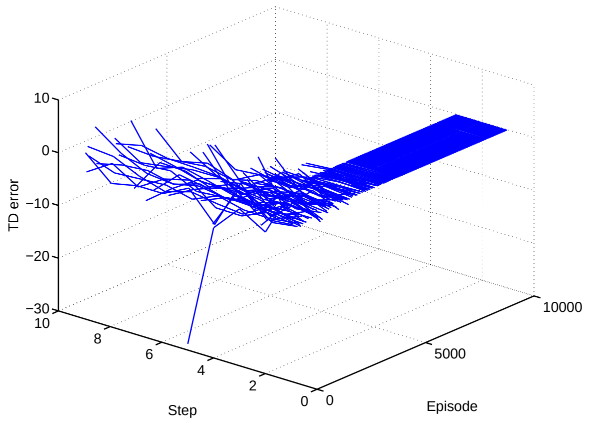
Fig. 5.6(a) shows that the TD error decreased over the episodes in all runs. Fig. 5.6(b) and (c) show how the TD error gradually decreased as the learning proceeded in two typical runs. The TD error jumps at the terminal step due to its difficulty in predicting the terminal reward. In addition, as shown in Fig. 5.6(b), the agent occasionally reached the goal in the sub-optimal 4 steps. This implies



(a) Absolute TD error



(b) Run 1



(c) Run 2

Figure 5.6. TD errors. (a) Mean absolute TD error. The window size of the moving average is 200 episodes. Error bars show the standard error of the mean (s.e.m.) over 10 runs. (b, c) TD errors in the (b) 1st run and (c) 2nd run. Data are sparsely plotted every 50 episodes starting from the first episode.

that this is one of the two runs that created the remaining variance in Fig. 5.3(a), Fig. 5.4, and Fig. 5.6(a).

Internal representations

We are interested in how sequentially collected observations are encoded and maintained in the agent throughout the decision making process. In this section, we analyze the activation patterns of the functional module in the agent.

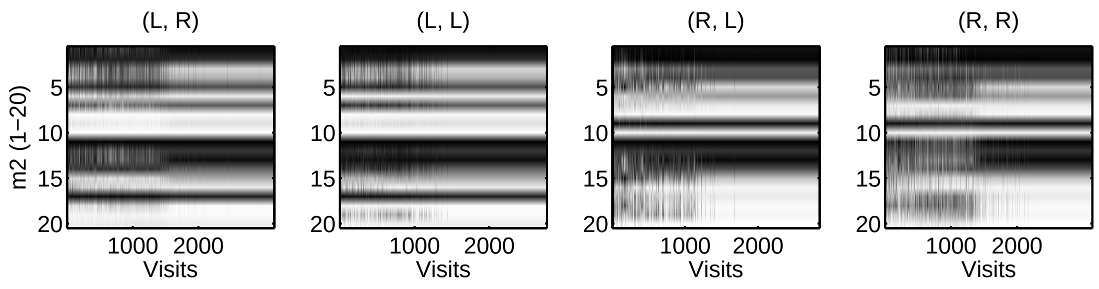
First, we focus on the memory layers of the reward predictor. In order to predict the terminal reward correctly, the information about the initial signal should be encoded and maintained in the activation patterns of this layer. Fig. 5.7 shows the maintenance of the initial signal in the reward-predictor's memory layer. The activation patterns at the T-junction differ depending on the initial signal even before the learning started (Fig. 5.7(a)). These trends are further confirmed by the PCA analysis in Fig. 5.7(b). This shows that, during both the first and the last 1000 episodes, the variations in the activations are mostly explained by the initial signal.

In order to see the difference between the reward predictor and the observation predictor, we analyze the activation patterns of the observation predictor's memory layer. The activation patterns at the T-junction did not get differentiated according to the initial signals (Fig. 5.8(a)). These trends are further confirmed by the PCA analysis shown in Fig. 5.8(b).

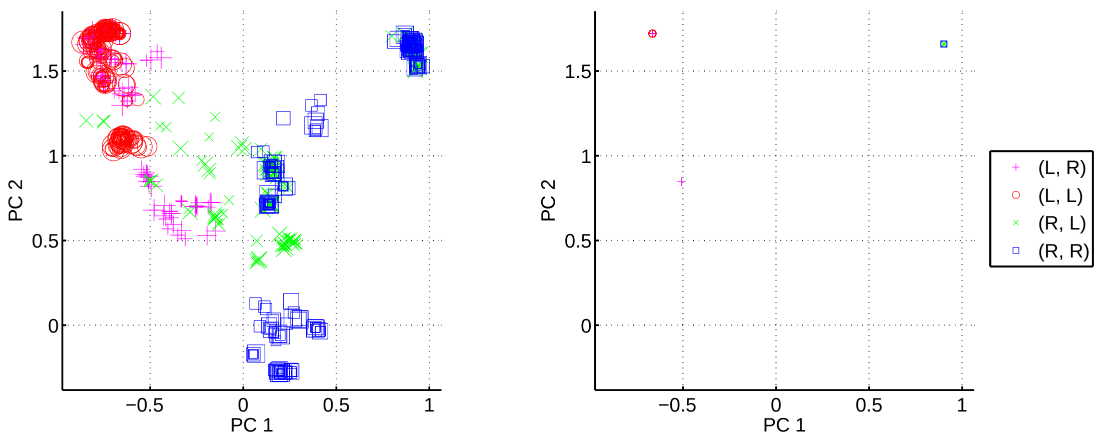
Next, the activation patterns of the actor's *state* nodes at the T-junction are investigated (Fig. 5.9-5.10). The actor's state is composed of 7-dimensional observation ($\boldsymbol{y}$) and 20-dimensional observation-predictor's memory layer ($\boldsymbol{m}^y$) and 20-dimensional reward predictor's memory layer ($\boldsymbol{m}^r$). The PCA analysis shown in Fig. 5.9 revealed that four conditions are separately represented in state nodes.

Further analysis revealed that most of the variability in the actor's state nodes can be captured by the first three principal components (Fig. 5.10(a)). The projection on this three-dimensional space revealed a clear separation of activations according to the task condition (Fig. 5.10(b)).

Finally, the activation patterns of the actor's hidden nodes at the T-junction are investigated (Fig. 5.11-5.13). For each task condition, distinct firing patterns

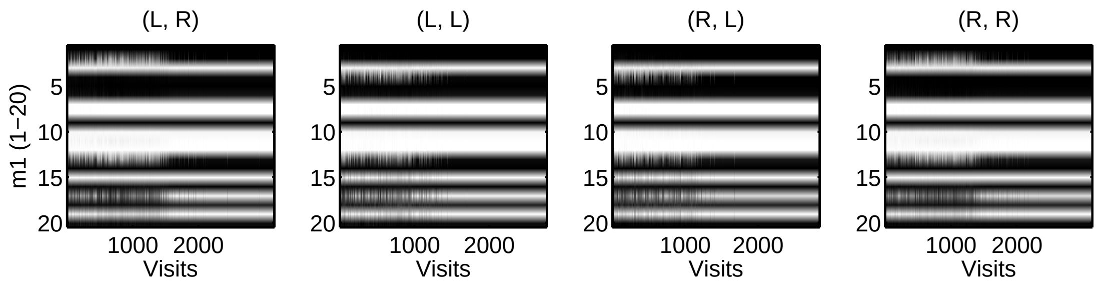


(a) All activations (initial signal, T-junction signal).

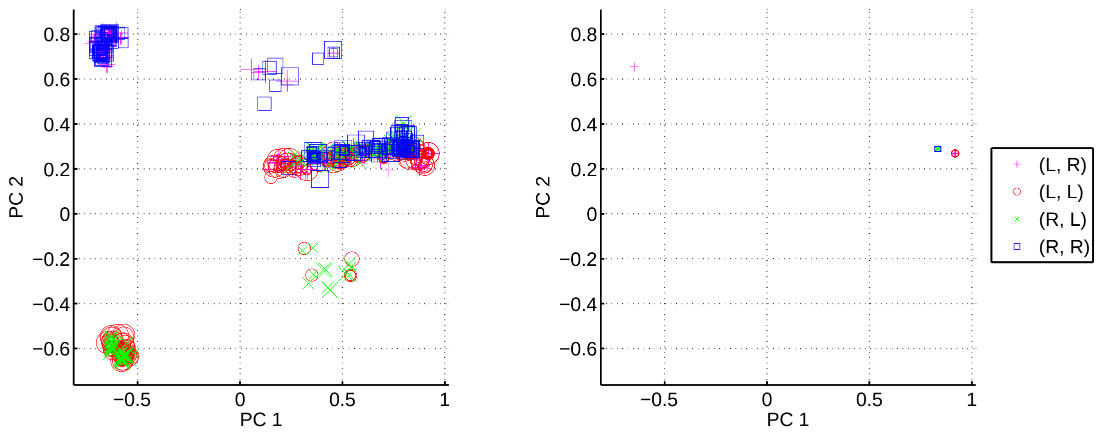


(b) PCA analysis: (Left) first 1000 episodes. (Right) last 1000 episodes.

Figure 5.7. Activation patterns of the reward predictor's memory nodes at the T-junction. These activations were used to predict the obtained rewards at the T-junction before entering the T-junction. (a) Activations for all visits to the T-junction are shown according to the episode conditions (initial signal, T-junction signal). (b) PCA analysis of these activations. The size of the marker reflects the number of steps to the goals. The smallest marker indicates 3 steps, and the largest marker indicates 10 steps.

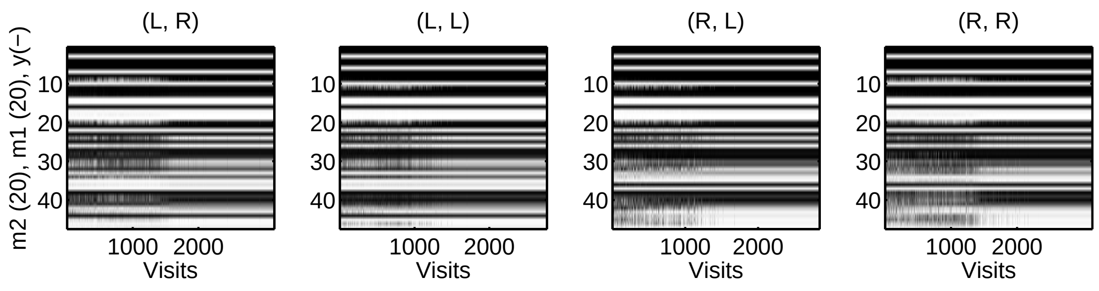


(a) All activations (initial signal, T-junction signal).

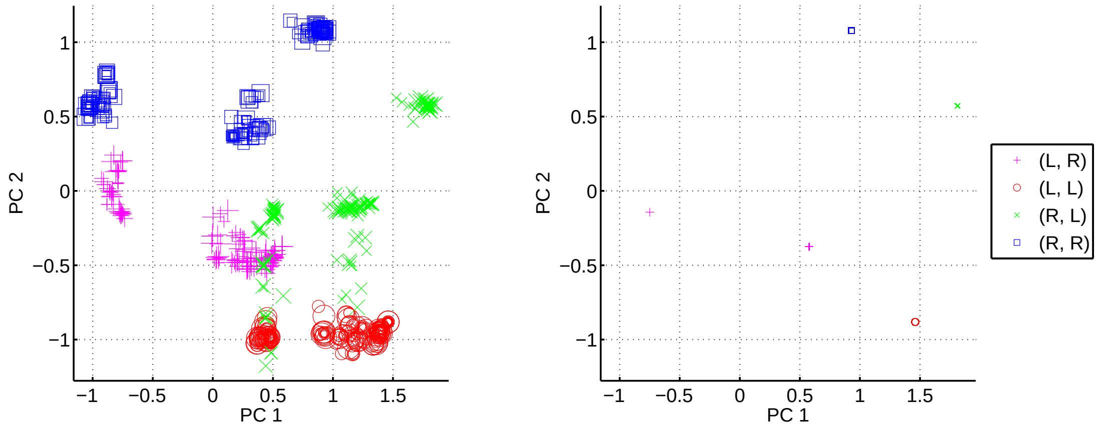


(b) PCA analysis: (Left) first 1000 episodes. (Right) last 1000 episodes.

Figure 5.8. Activation patterns of the observation predictor's memory nodes at the T-junction. These activations were used to predict the preprocessed observations observed at the T-junction before entering the T-junction. (a) Activations for all visits to the T-junction are shown according to the episode conditions (initial signal, T-junction signal). (b) PCA analysis of these activations.

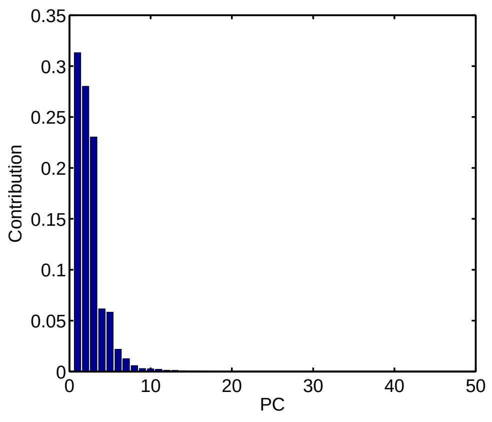


(a) All activations (initial signal, T-junction signal).

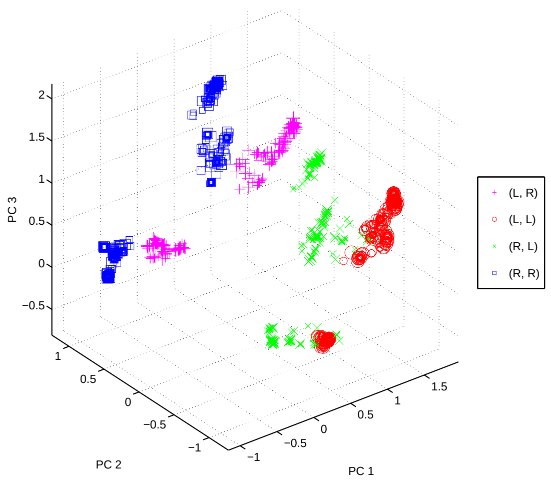


(b) PCA analysis: (Left) first 1000 episodes. (Right) last 1000 episodes.

Figure 5.9. Activation patterns of the actor's state nodes at the T-junction. These activations were used to select the next action at the T-junction.



(a) Contribution of each principal component



(b) Projection of all activations in the first three principal components

Figure 5.10. Analysis of the activation patterns of the actor's state nodes.

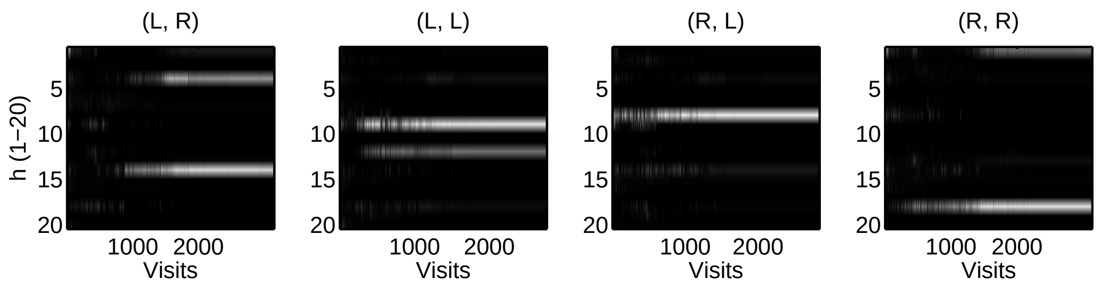
appear as the training proceeds (Fig. 5.11(a)). These trends are further confirmed in the PCA analysis of these activations. In Fig. 5.11, activations during the first and the last 1000 visits to the T-junction are projected onto the two principal components. This figure shows the separation of firing patterns depending on the task condition. The dynamic separation in the learning process is clearly seen in Fig. 5.12 (Fig. 5.11 shows the first and the last 1000 episodes of Fig. 5.12). It reveals the gradual convergence of firing patterns depending on the task condition.

Further analysis revealed that most of the variability in the actor's hidden nodes can be captured by the first three principal components (Fig. 5.13(a)). The projection on this three-dimensional space revealed that the firing patterns for each task condition were not only separated in the three-dimensional space but were also orthogonalized (Fig. 5.13(b)).

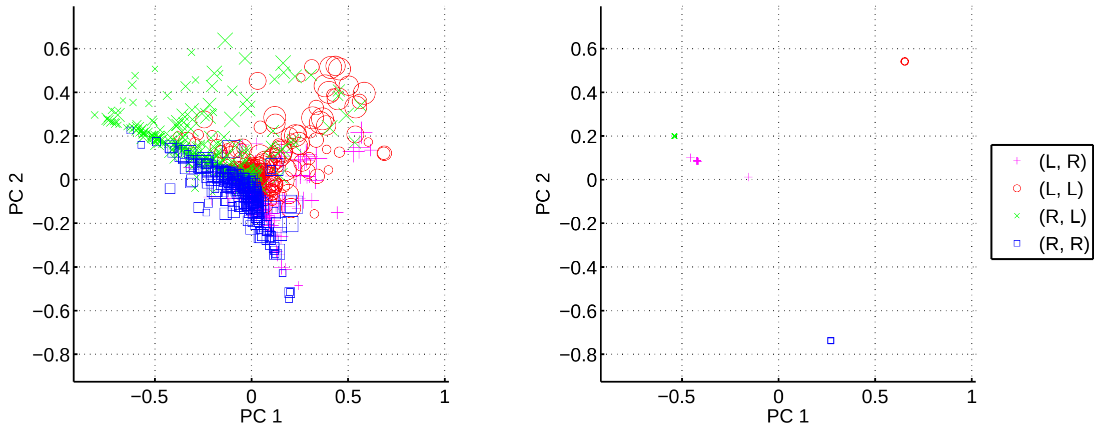
5.3.2 Digit matching T-maze task

Task setting

The digit matching T-maze task (Fig. 4.20) was extended further to use noisy pixel images. In this extended version, observations associated with a certain



(a) All activations (initial signal, T-junction signal).



(b) PCA analysis: (Left) first 1000 episodes. (Right) last 1000 episodes.

Figure 5.11. Activation patterns of the actor's hidden nodes at the T-junction. These activations were realized after selecting the action at the T-junction.

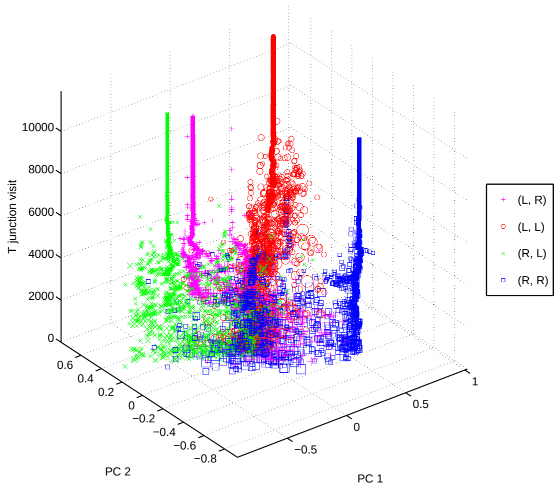
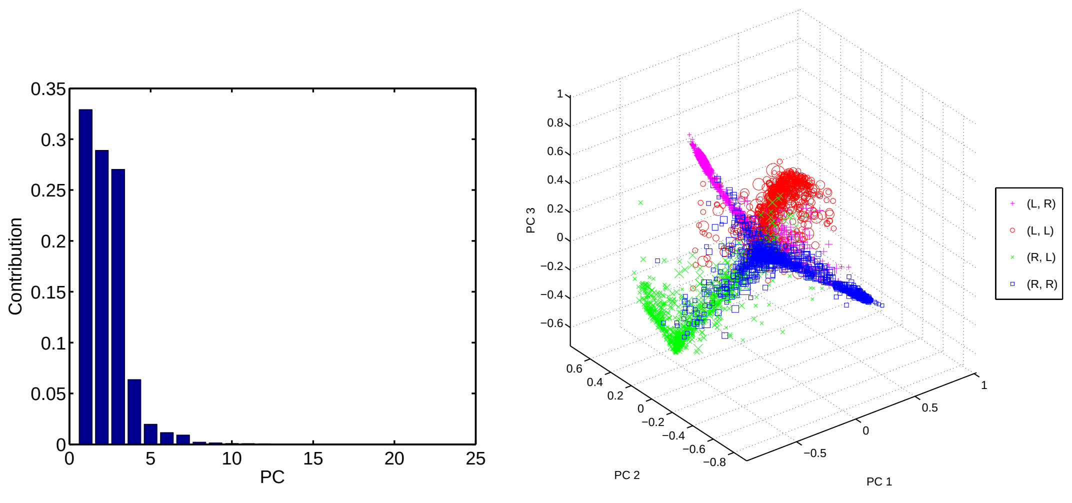


Figure 5.12. Changing activation patterns of the actor's hidden nodes at the T-junction.



(a) Distribution of each principal component (b) Projection of all activations in the first three principal components

Figure 5.13. Analysis of activation patterns of the actor's hidden nodes at the T-junction.

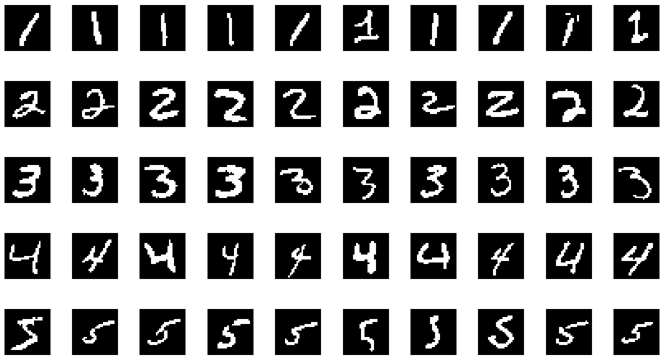


Figure 5.14. Examples of actual observations for each digit class. 10 out of 100 images are shown.

position are not fixed but are stochastically chosen from the fixed-size reservoir of handwritten digits. Observations change even if the agent visits the same position within the same episode. In order to simplify the analysis, we fixed the size of the handwritten digits in the reservoir to 100. Fig. 5.14 shows examples of actual handwritten digits used for observations. These handwritten images vary greatly even within the same digit class.

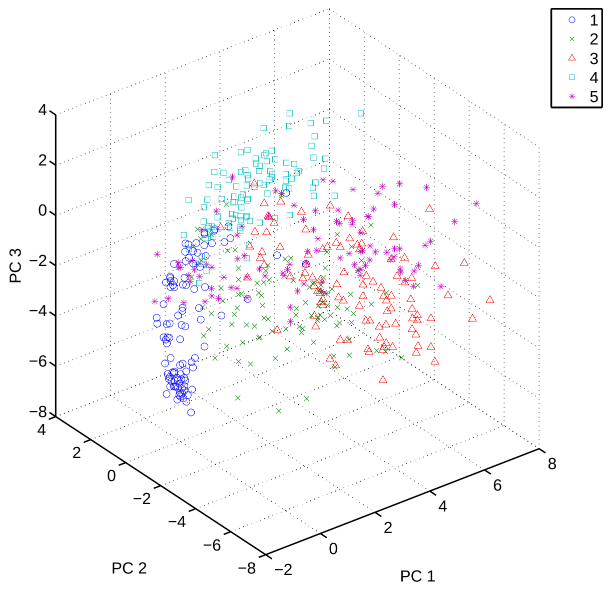
Fig. 5.15(a) shows the distribution of these five digits in the lower-dimensional space spanned by the first three principal components. The contribution ratio of the principal components has a long tail (Fig. 5.15(b)).

Observation preprocessor

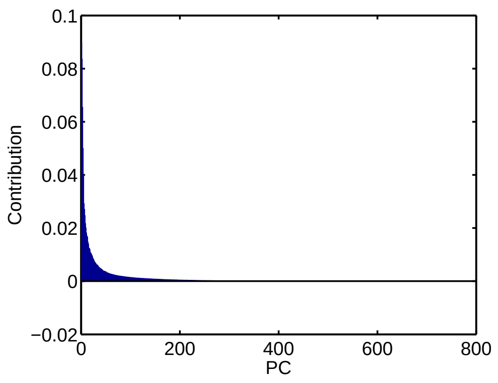
In the improved architecture, high-dimensional observations are preprocessed with RBMs. We used an RBM with 20 hidden nodes to preprocess 784-dimensional observations.

The RBM was trained with all 60000 training data in the MNIST dataset of handwritten digits (approximately 6000 images per digit class). The entire training dataset was swept 600 times for training using contrastive divergence with 3 iterations of Markov update (CD-3).

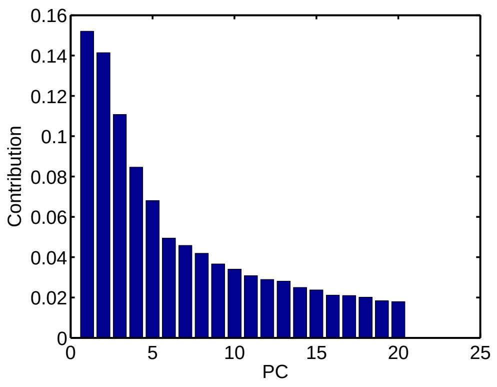
Fig. 5.16(a) shows the distribution of processed observations for five digits in the lower-dimensional space spanned by the first three principal components. As compared to Fig. 5.15(a), the distribution of each digit class is well preserved



(a) Projection of all possible raw observations on the subspace spanned by 3 principal components

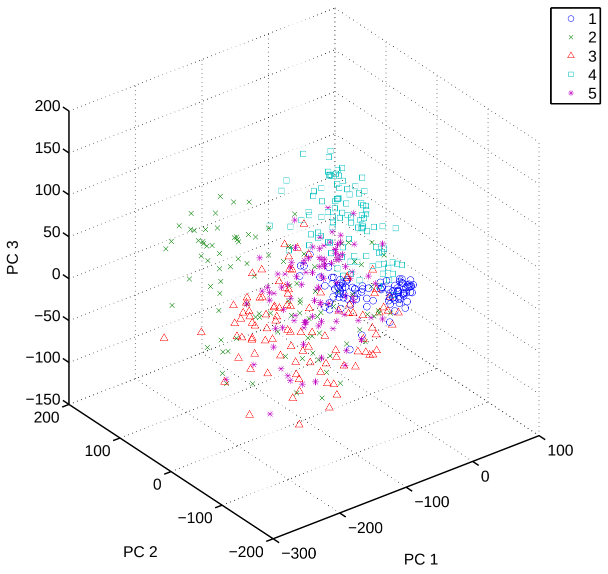


(b) Distribution of all 784 principal components

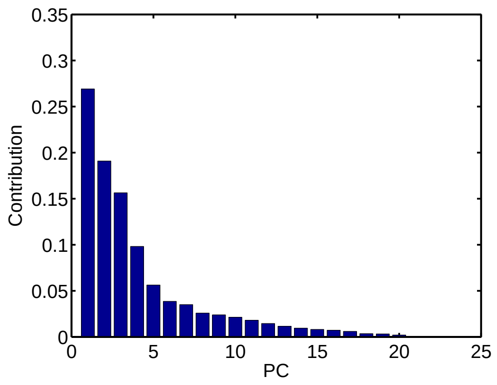


(c) Distribution of the first 20 principal components

Figure 5.15. PCA analysis of all possible observations (500 instances of 784-dimensional pixel images of handwritten digits) in the digit matching T-maze.

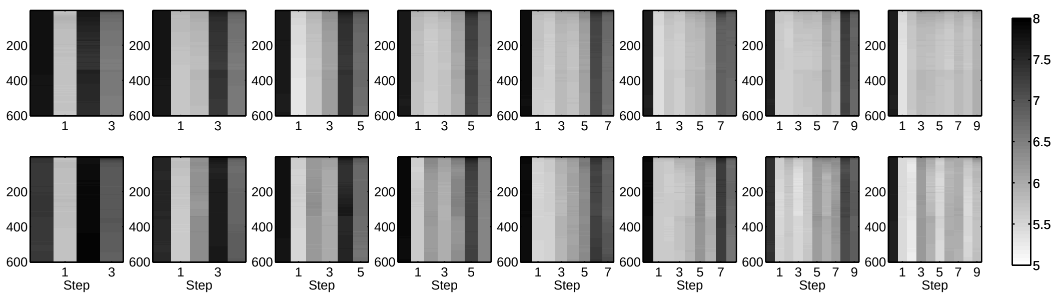


(a) Projection of processed observations on the subspace spanned by 3 principal components



(b) Distribution of all 20 principal components

Figure 5.16. PCA analysis of processed observations. The dimensionality is reduced from 784 to 20 using a trained RBM. 500 instances of 20-dimensional processed data are used for the analysis.


(a) Observation prediction error

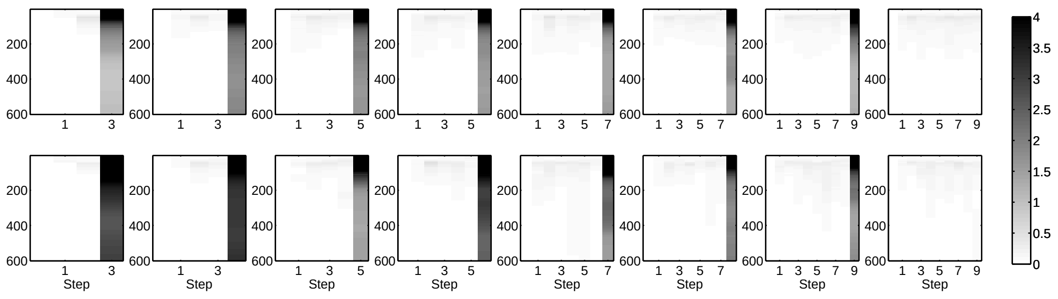
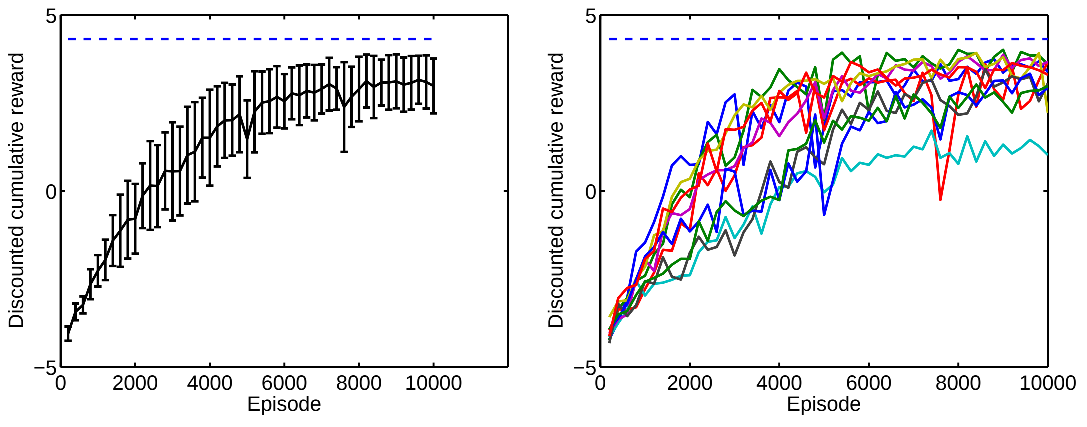

(b) Reward prediction error

Figure 5.17. Average prediction errors of (a) the processed-observation predictor and (b) scalar-reward predictor. The measure of prediction errors are absolute error for (a) and squared error for (b). Note the lower bound of the color map in (a) is not 0 but 5. Refer to the caption of Fig. 4.5 for reading the figure.

after this operation. In addition, the variability in the data was well captured by the first few principal components (Fig. 5.15(b)).

Basic performance

As shown in Fig. 5.17, the predictor's performance reflected the predictability of observations and rewards after training with $\mathcal{D}^T_{\text{train}}$ ($T = 3, 4, 5, 6, 7$). As compared to Fig. 5.2(a), the baseline of prediction errors is high in Fig. 5.17(a) due to the noise in the processed observations. The noise or variability in the inputs and targets of sequence prediction data makes the prediction task significantly difficult. Nevertheless, the processed-observation predictor exhibits a performance improvement in this difficult prediction task. Otherwise, the same overall trends seen in Fig. 5.2 are again observed in Fig. 5.17.

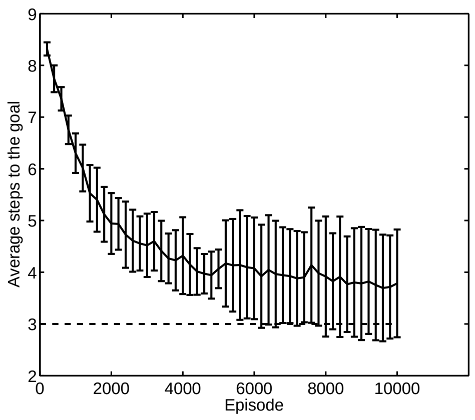


(a) Average over 10 runs (b) Individual runs

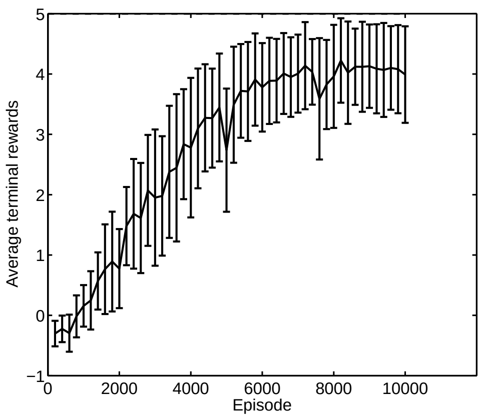
Figure 5.18. Average discounted return in the matching T-maze task ($T_{\min} = 3$). The window size is 200 episodes. Error bars show the standard error of the mean (s.e.m.). The upper bound of the discounted return is indicated by the dotted line ($R_0 = 4.3175$ with $\gamma = 0.95$).

After the training of the predictor, the predictor's weights were fixed. Then, the actor was trained using SARSA(0). A moving average of the discounted return over 10 runs is shown in Fig. 5.18(a). It indicates that this POMDP task was sufficiently solved regardless of the noise in the high-dimensional observations. Both the average number of steps to the goal (Fig. 5.19(a)) and the average terminal reward (Fig. 5.19(b)) improved toward their theoretical optimal values; however, they never reached these values due to the existing noise in the observations. This POMDP task with high-dimensional, noisy inputs without a prior knowledge of the environment is considered to be an extremely difficult problems in POMDP literature. Because the positive cumulative cannot be obtained consistently without the identification of the correct goal, the positive discounted return in all runs shown in Fig. 5.18(b) indicates that the agent sufficiently and consistently learned to discriminate the correct goal in all 10 runs regardless of the task difficulties.

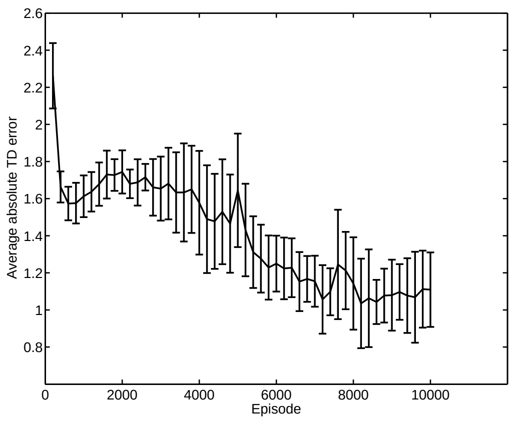
Fig. 5.19(c) shows that the TD error decreases over the episodes in all runs. A closer look at the first run (Fig. 5.19(d)) reveals how the TD error gradually decreased as the learning proceeded. Unlike Fig. 5.6(b), the TD error does not



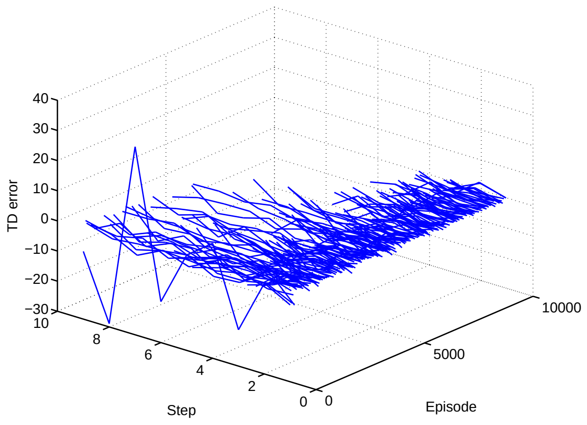
(a) Average number of steps to the goal over 10 runs ($T_{\min} = 3$)



(b) Average terminal reward over 10 runs ($T_{\min} = 3$)



(c) Average over 10 runs



(d) Run 1

Figure 5.19. TD errors in the digit-matching T-maze. Refer to the caption of Fig. 5.6 for reading the figure.

exhibit a large jump at the last step, but is distributed evenly for all time steps throughout episodes (except the first episode). This should be due to the noise in the observations. The remaining ambiguity about the exact position in the maze necessarily increases the TD errors (note that the magnitude of the z-axis is different in Fig. 5.6(b) and Fig. 5.19(d)).

Internal representations

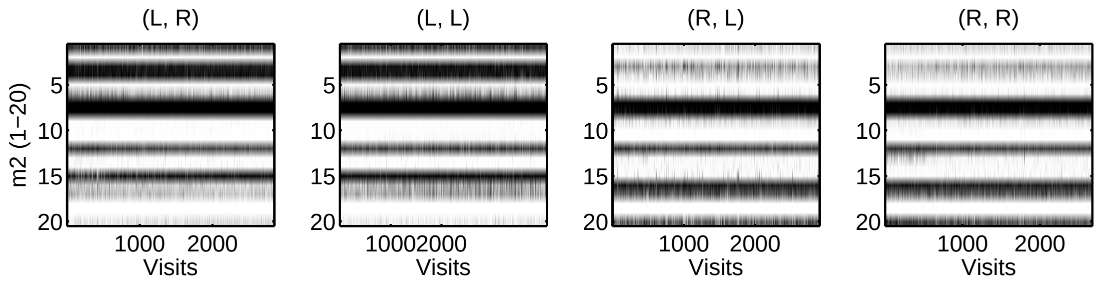
We now investigate how this artificial agent codes the external world, and how it is modified as the agent's performance improves.

First, we focus on the memory layers of the reward predictor. In order to predict the terminal reward correctly, the information about the initial signal should be encoded and maintained in the activation patterns of this layer. Fig. 5.20(a) shows the activation of the 20 nodes at the T-junction in all four signal conditions. The PCA analysis in Fig. 5.20(b) shows that the variation in the activations is explained by the initial signal even before the actor's learning starts. Fig. 5.20 clearly suggests that the maintenance of the initial signal in the reward-predictor's memory layer.

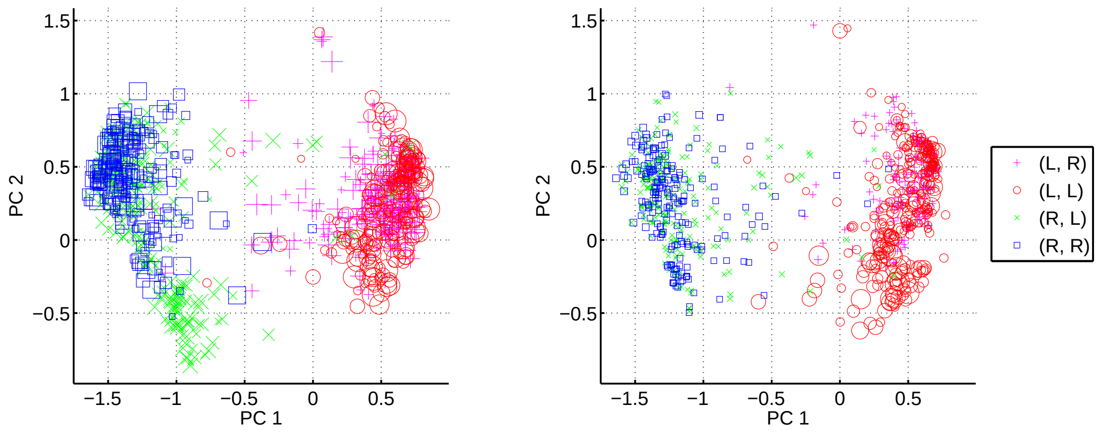
In order to see the difference between reward predictor and observation predictor, we also analyze the activation patterns of the observation predictor's memory layer. The activation patterns at the T-junction did not get differentiated according to the initial signals (Fig. 5.21(a)). These trends are further confirmed by the PCA analysis shown in Fig. 5.21(b).

Next, the activation patterns of the actor's *state* nodes at the T-junction are investigated (Fig. 5.23). The actor's state is composed of the 20-dimensional processed observation layer ($\boldsymbol{y}$), the 20-dimensional observation predictor's memory layer ($\boldsymbol{m}^y$), and the 20-dimensional reward predictor's memory layer ($\boldsymbol{m}^r$). The PCA analysis shown in Fig. 5.23 revealed that the activation patterns of state nodes are well separated in the two-dimensional projected space. These differential activation patterns were present even before the learning started.

Further analysis revealed that most of the variability in the actor's state nodes can be captured by the first three principal components (Fig. 5.23(a)). The projection on this three-dimensional space revealed the clear separation of activations according to the task condition (Fig. 5.23(b)).

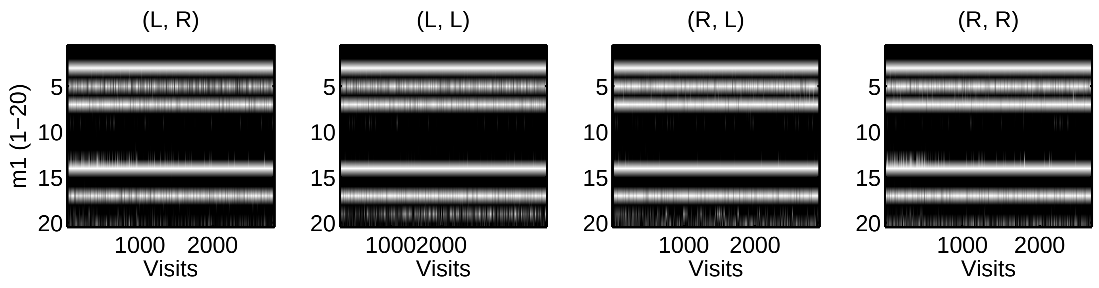


(a) All activations (initial signal, T-junction signal).

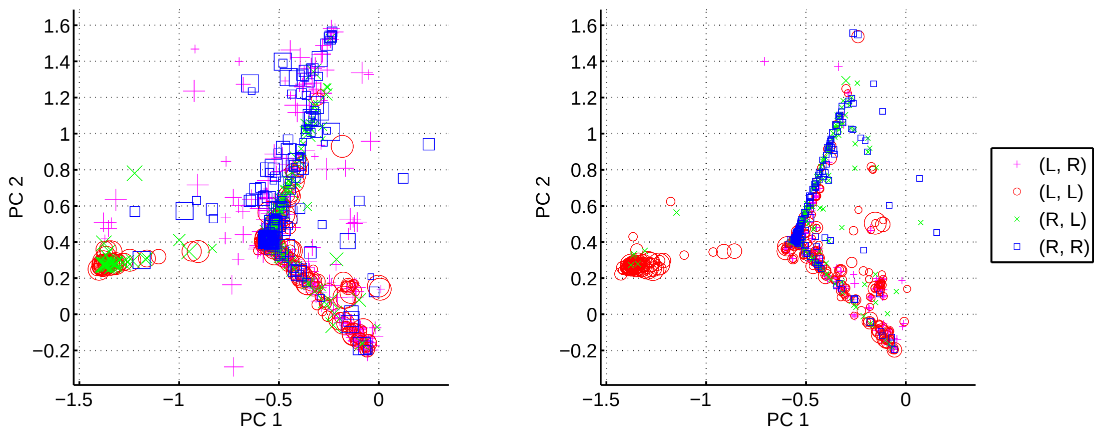


(b) PCA analysis: (Left) first 1000 episodes. (Right) last 1000 episodes.

Figure 5.20. Activation patterns of the reward predictor's memory nodes at the T-junction. These activations were used to predict the obtained rewards at the T-junction before entering the T-junction. (a) Activations for all visits to the T-junction are shown according to the episode conditions (initial signal, T-junction signal). (b) PCA analysis of these activations. The size of the marker reflects the number of steps to the goals. The smallest marker implies 3 steps, and the largest marker implies 10 steps.

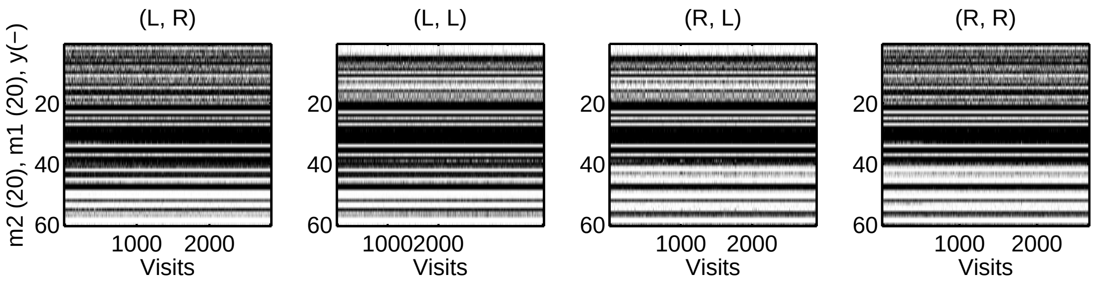


(a) All activations (initial signal, T-junction signal).

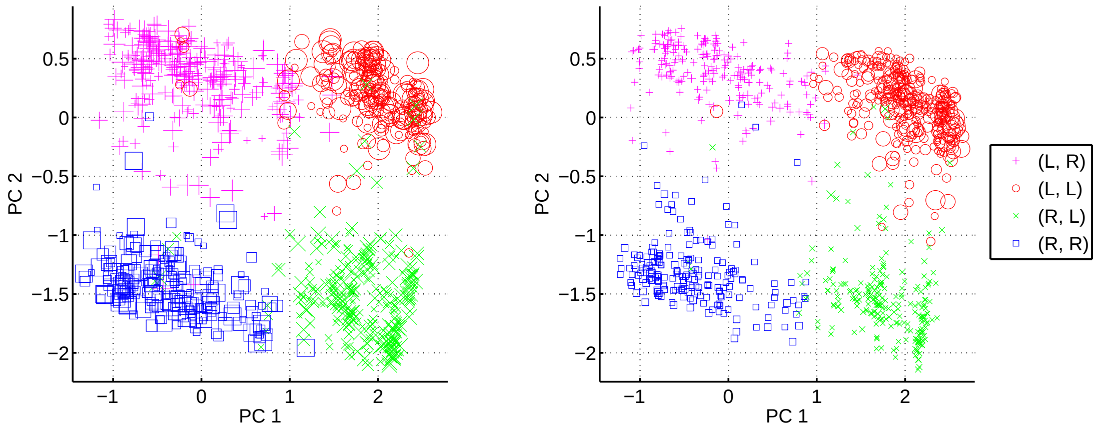


(b) PCA analysis: (Left) first 1000 episodes. (Right) last 1000 episodes.

Figure 5.21. Activation patterns of the observation predictor's memory nodes at the T-junction. These activations were used to predict the preprocessed observations observed at the T-junction before entering the T-junction. (a) Activations for all visits to the T-junction are shown according to the episode conditions (initial signal, T-junction signal). (b) PCA analysis of these activations.

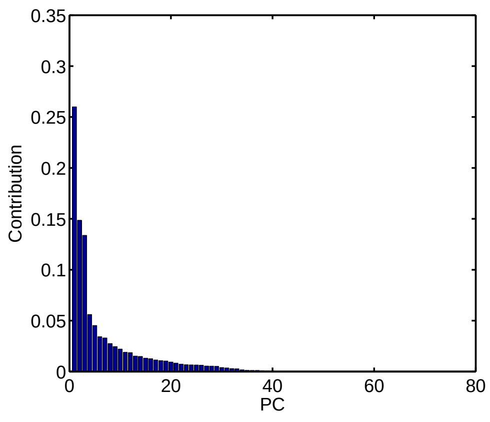


(a) All activations (initial signal, T-junction signal).

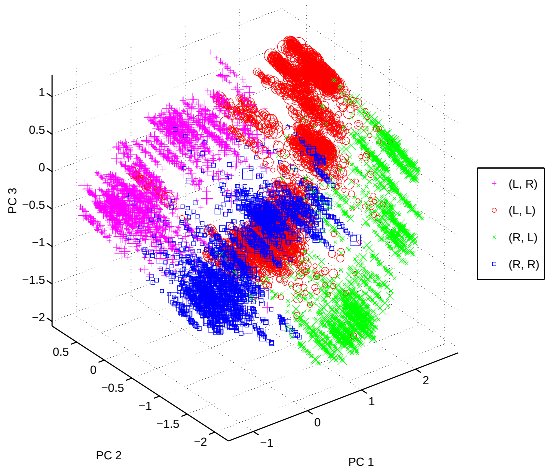


(b) PCA analysis: (Left) first 1000 episodes. (Right) last 1000 episodes.

Figure 5.22. Activation patterns of the actor's state nodes at the T-junction. These activations were used to select the next action at the T-junction.



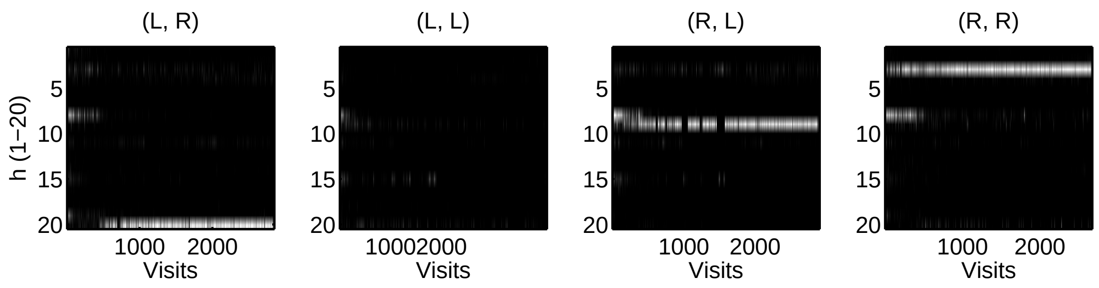
(a) Distribution of each principal component



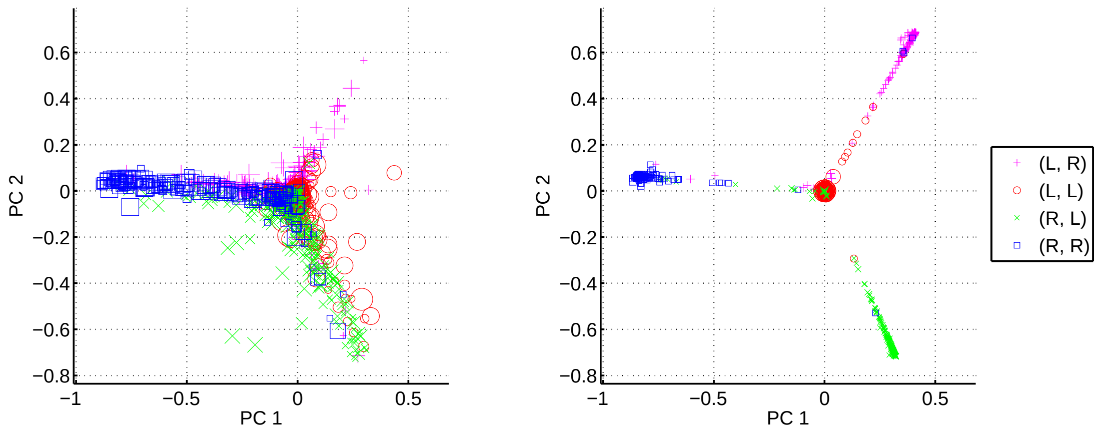
(b) Projection of all activations in the first three principal components

Figure 5.23. Analysis of activation patterns of the actor's state nodes.

Finally, the activation patterns of the actor's hidden nodes at the T-junction are investigated (Fig. 5.25). For each task condition, distinct firing patterns appear as the training proceeds (Fig. 5.24(a)). These trend are further confirmed in the PCA analysis of these activations. In Fig. 5.25, activations during the first and the last 1000 visits to the T-junction are projected onto the two principal components. This figure shows the separation of firing patterns depending on the task condition. The dynamic separation in the learning process is clearly seen in Fig. 5.25(a) (Fig. 5.25 shows the first and the last 1000 episodes of Fig. 5.25(a)). It reveals the gradual convergence of firing patterns depending on the task condition regardless of the noisy observation. In this run, (L, L) condition is represented with no activation in the actor's hidden nodes. Further analysis revealed that most of the variability in the actor's hidden nodes was already captured by the first two principal components (Fig. 5.25(b)). This is the reason why the firing patterns for each task condition were not only separated but were also orthogonalized in the two-dimensional space (Fig. 5.25(a)).

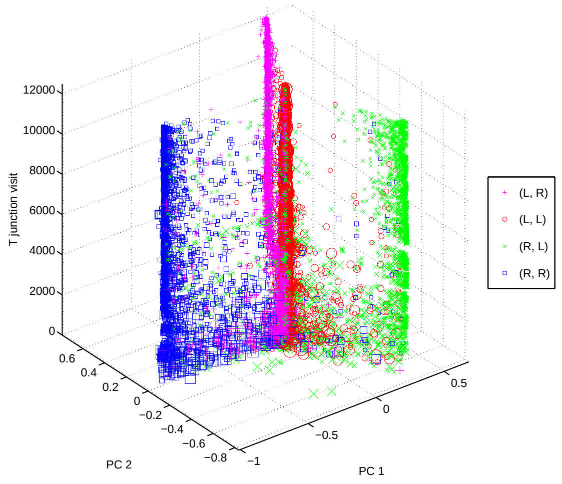


(a) All activations (initial signal, T-junction signal).

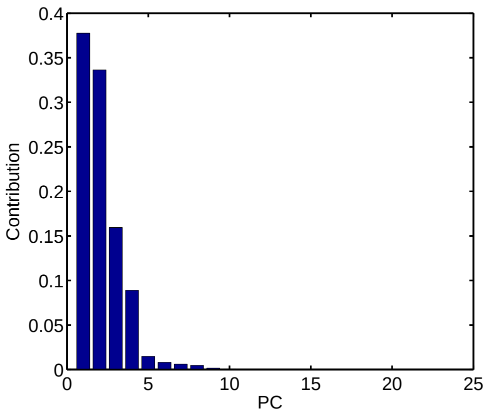


(b) PCA analysis: (Left) first 1000 episodes. (Right) last 1000 episodes.

Figure 5.24. Activation patterns of the actor's hidden nodes at the T-junction. These activations were realized after selecting the action at the T-junction.



(a) Changing activation patterns of the actor's hidden nodes at the T-junction.



(b) Distribution of each principal component.

Figure 5.25. Analysis of activation patterns of the actor's hidden nodes.

5.4 Summary and discussion

In this chapter, we modified the previously proposed architecture of an agent in order to solve a more difficult class of POMDP problems. The two new features introduced here were observation preprocessing based on RBMs and separate predictors for observations and rewards. This modification results in the removal of cumbersome reward encoding process and several unwanted parameters. The performance of the new architecture is reliable even in the POMDP task with high-dimensional noisy inputs. Regardless of the apparent difference between the two tasks solved in this chapter (matching T-maze task with 7-dimensional noiseless observations and digit matching T-maze task with 784-dimensional noisy observations), the same parameter sets were used for both experiments except the length of predictor training. The general coding patterns of different functional modules were similar in both tasks.

It is interesting to note that similar coding patterns were seen in both experiments. Although these two tasks used different observations (or more precisely, the different observation probabilities $p(\boldsymbol{o}|\boldsymbol{s})$ with different observation sets $\mathcal{O}$), the underlying MDP is exactly the same. It appears that coding reflects the underlying "true" MDP task regardless of the superficial difference in the observation. The task-dependent coding of high-dimensional observations realized in the proposed architecture matches the experimental finding in the monkey study (Amemori and Sawaguchi, 2006). Given the ability to handle multimodal, noisy, high-dimensional inputs, it is possible that the basic concept of the given architecture is implemented in the brain as a computational principle.

There are several ways to improve the the proposed architecture. First, we can mix the training phase of a predictor and an actor. When the predictor's performance is not satisfactory, an actor can explore the environment to collect data with a low inverse temperature β. As the predictor's performance improves, the inverse temperature β is increased to collect more useful data for task achievement. Prediction errors can also be added to the raw reward to boost exploration in an area with surprising observations. In this manner, we can balance exploration and exploitation.

Second, the use of DBNs may be beneficial to handle high-dimensional inputs from several modalities. In order to integrate information from different modalities, the dimensionality should be massively reduced in each modality. Because the hierarchical organization of DBNs can preserve more information given the fixed reduced dimensionality than single-layer RBNs, DBNs can provide better processed data for the actor than RBMs.

Third, a mechanism for treating high-dimensional actions can be incorporated. Fortunately, this is an easy step because the FERL framework was originally proposed to handle high-dimensional actions using Markov chain Monte Carlo methods (Sallans and Hinton, 2000).

Fourth, the predictor's performance can be improved in several directions. In order to completely avoid the difficulty of training RNNs, we can make use of concepts used in reservoir computing methods such as echo state networks (ESNs) (Jaeger and Haas, 2004) and liquid state machines (LSMs) (Maass et al., 2002). According to this framework, randomly initialized and properly adjusted sparsely connected recurrent weights can carry sufficient information for temporal prediction. In order to handle a task that has long-term dependencies, methods such as long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1996) and temporal kernel recurrent neural networks (TKRNN) (Sutskever and Hinton,

2010) can be used.

Fifth, continuous data can be handled with the code conversion through a pre-trained RBM (Nair and Hinton, 2008). A single RBM with Gaussian visible units and binary hidden units is first pre-trained using raw continuous data. Then, the agent can use these processed data as binary observations.

Finally, we can also improve the biological plausibility of the model. Although an actor implemented as an RBM appears to be biologically plausible, the BPTT used to train the predictor in Chapters 4 and 5 severely degrades the biological plausibility. This part can be modified using a spike-timing-dependent-plasticity (STDP) learning rule. Recently, Lazar et al. (2009) proposed a self-organizing recurrent network (SORN) that combined STDP and a homeostatic rule to train binary RNNs. This model appears to fit the predictor part of our model. In order to improve the biological plausibility, the homeostatic rule used in the SORN can be replaced by the Oja rule. Our newly proposed model can also be implemented as a spiking neural network. A firing rate model such as our proposed architecture becomes equivalent to the spiking neural network model under the assumption of an expoential synaptic kernel (Dayan et al., 2001).

Chapter 6

Conclusions

The general goal of this thesis was to understand how learning animals represent the external world. We believe that they represent the world to achieve some goals through decision making. We did not directly investigate the brain. Instead, we tried to solve using neurally realizable models the computational problems that our brains solve. We formalized this problem as a partially observable Markov decision process (POMDP) with high-dimensional noisy observations without knowledge of environmental dynamics or a state set with a Markov property. This problem by itself is considered to be notoriously difficult in the fields of machine learning and autonomous robotics.

As a building block, we employed a method originally proposed by Sallans and Hinton (2000, 2004), which we call a free-energy-based reinforcement learning (FERL) framework, due to its biological plausibility and its ability to handle MDPs with high-dimensional states and actions.

When the agent solves an easy POMDP task (Class I POMDP using the classification criterion in section 2.3.1) called a digit floor task without a prior knowledge of environmental dynamics and an underlying state set with a Markov proerpty, the underlying task structure was reflected in the activation patterns of the agent's hidden nodes in observation-, reward- and action-dependent fashion (Chapter. 3). After learning, noisy digit images that share the same optimal action induced similar activities in the agent's hidden nodes. Several experimental studies with monkeys reported the similar findings about the reward-modulated sensory coding (Freedman et al., 2001; Miller et al., 2003; Sigala and Logothetis, 2002). In addition, by giving a constant reward of 0 for any combination of states and actions, the activation patterns of the hidden nodes reflected a distribution of encountered observations. Reward-independent coding of observations in the early stages of sensory processing has been reported in many experimental studies and was modeled extensively in computational neuroscience literature (Olshausen and Field, 1996; Rao and Ballard, 1999; Hyvarinen et al., 2009). FERL's properties observed in our experiments such as the robustness to noise and the gradual degradation of performance with noise increase are also commonly seen in biological agents (Pelli and Farell, 1999; Rainer et al., 2001).

In chapter 4, FERL was combined with Elman-type recurrent neural networks to handle POMDP problems that require dynamic combination of sensory inputs. Although the biological plausibility is hampered by the use of BPTT, moderately difficult POMDP tasks (Class II POMDP) with low-dimensional bit patterns were solved due to this change. Knowledge of environmental dynamics and a state set were again not assumed. After the predictor's performance improved sufficiently, the state of the actor, which is a combination of the current observation and currently maintained memory, began to exhibit Markov properties even under the true POMDP setting. In other words, the current partial observation was enhanced enough to have an approximately Markov property by the rich information about the past carried in the predictor's hidden layer. In the matching T-maze task, the underlying task structure gradually emerged in the activation pattern of the actor's hidden nodes as the leaning proceeded. This gradual separation of the population's firing properties during performance improvement is believed to be a characteristic of real neuronal populations. However, the neural data is not yet available due to the difficulties involved in simultaneously recording from the multiple task-relevant neurons.

In Chapter 5, the architecture was modified to handle POMDP tasks with high-dimensional stochastic observations. Knowledge of environmental dynamics and a state set were again not assumed. This setting is much more realistic because we rarely see the same object in exactly the same way. The same images on the computer screen are likely to be projected as different images on the retina. Animals somehow need to use the perceptual similarity of the objects. This part is realized using a restricted Boltzmann machine (RBM) in the improved architecture. The RBM's known ability of dimensionality reduction and orthogonalization in the feature space made preprecessed observations better targets than raw observations. The results presented in this chapter confirmed that the improved architecture could reliably solve these difficult POMDP problems (Class III POMDP). The task-dependent encoding of observation sequences was found in the activation patterns of the actor's hidden units. Again, these trends in the population of nodes emerged as the learning proceeded.

Simulation results obtained in this thesis supported our claim that our dynamic extension of FERL could construct a distributed representation of the external world autonomously while solving realistic sequential decision making problems. This computational approach may provide a basis for biologically plausible models of representation learning in the brain. The results constitute my tentative answer to the symbol grounding problem and the long-lasting epistemological question asked by biological agents that resist a natural tendency to disorder: "How do we know?"

References

Amemori, Kenichi and Sawaguchi, Toshiyuki (2006) "Rule-dependent shifting of sensorimotor representation in the primate prefrontal cortex," *Eur. J. Neurosci.*, Vol. 23, pp. 1895–1909.

Astrom, K. J. (1965) "Optimal control of Markov decision processes with the incomplete state estimation," *Journal of Computer and System Sciences*, Vol. 10, pp. 174–205.

Bakker, B. (2002) "Reinforcement learning with long short-term memory," *NIPS*, Vol. 2, pp. 1475–1482.

Barto, A.G., Sutton, R.S., and Anderson, C.W. (1983) "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on systems, man, and cybernetics*, Vol. 13, No. 5, pp. 834–846.

Bellman, R. E. (1957) *Dynamic Programming*, Princeton: Princeton University Press.

Churchland, P.S. and Sejnowski, T.J. (1990) "Neural representation and neural computation," *Philosophical Perspectives*, Vol. 4, pp. 343–382.

Dayan, P., Abbott, L.F., and Abbott, L. (2001) *Theoretical neuroscience: Computational and mathematical modeling of neural systems*: MIT Press.

Dockendorf, K.P., Park, I., He, P., Principe, J.C., and DeMarse, T.B. (2009) "Liquid state machines and cultured cortical networks: The separation property," *BioSystems*, Vol. 95, pp. 90–97.

Engel, Y. (2005) “Algorithms and representations for reinforcement learning,” Ph.D. dissertation, The Hebrew University of Jerusalem.

Engel, Y., Mannor, S., and Meir, R. (2002) “Sparse online greedy support vector regression,” in *13 th European Conference on Machine Learning*, pp. 84–96: Springer.

Freedman, D.J. and Assad, J.A. (2006) “Experience-dependent representation of visual categories in parietal cortex,” *Nature*, Vol. 443, No. 7107, pp. 85–88.

Freedman, D.J., Riesenhuber, M., Poggio, T., and Miller, E.K. (2001) “Categorical representation of visual stimuli in the primate prefrontal cortex,” *Science*, Vol. 291, No. 5502, p. 312.

Friston, K. (2009) “The free-energy principle: a rough guide to the brain?,” *Trends in Cognitive Sciences*, Vol. 13, No. 7, pp. 293–301.

Friston, K. and Kiebel, S. (2009) “Cortical circuits for perceptual inference,” *Neural Networks*, Vol. 22, No. 8, pp. 1093–1104.

Friston, KJ, Daunizeau, J., Kiebel, SJ, and Sporns, O. (2009) “Reinforcement Learning or Active Inference,” *PLoS ONE*, Vol. 4, No. 7, p. e6421.

Froemke, R.C., Merzenich, M.M., and Schreiner, C.E. (2007) “A synaptic memory trace for cortical receptive field plasticity,” *Nature*, Vol. 450, No. 7168, pp. 425–429.

George, D. and Hawkins, J. (2009) “Towards a Mathematical Theory of Cortical Micro-circuits,” *PLoS Comput Biology*, Vol. 5, No. 10, pp. 1–26.

Hinton, G. E. (2002) “Training Products of Experts by Minimizing Contrastive Divergence.,” *Neural Computation*, Vol. 14, pp. pp 1771–1800.

Hochreiter, Sepp and Schmidhuber, Jurgen (1996) “LSTM can Solve Hard Long Time Lag Problems,” in *NIPS*, p. 473.

Hyman, S.E., Malenka, R.C., and Nestler, E.J. (2006) “Neural mechanisms of addiction: the role of reward-related learning and memory,” *Annu. Rev. Neurosci*, Vol. 29, pp. 565–598.

Hyvarinen, A., Hurri, J., and Hoyer., P. O. (2009) *Natural Image Statistics*: Springer-Verlag.

Jaeger, H. and Haas, H. (2004) “Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication,” *Science*, Vol. 304, No. 5667, p. 78.

Jog, M.S., Kubota, Y., Connolly, C.I., Hillegaart, V., and Graybiel, A.M. (1999) “Building neural representations of habits,” *Science*, Vol. 286, No. 5445, p. 1745.

Kappen, HJ (2005) “Path integrals and symmetry breaking for optimal control theory,” *Journal of statistical mechanics: theory and experiment*, Vol. 2005, p. P11011.

Kawato, M. (2008) “From Understanding the Brain by Creating the Brain towards manipulative neuroscience,” *Philosophical Transactions of the Royal Society B: Biological Sciences*, Vol. 363, No. 1500, p. 2201.

Keller, P. W., Mannor, S., and Precup, D. (2006) “Automatic Basis Function Construction for Approximate Dynamic Programming and Reinforcement Learning,” in *Proceedings of International Conference of Machine Learning*, pp. 449–456.

Konda, V.R. and Tsitsiklis, J.N. (2004) “On actor-critic algorithms,” *SIAM Journal on Control and Optimization*, Vol. 42, No. 4, pp. 1143–1166.

Lazar, A., Pipa, G., and Triesch, J. (2009) “SORN: a Self-organizing Recurrent Neural Network,” *Front. Comput. Neurosci*, Vol. 3, p. 23.

Lovejoy, William S. (1991) “A survey of algorithmic methods for partially observed Markov decision processes,” *Annals of Operations Research*, Vol. 28, No. 1, pp. 47–66.

Maass, W., Natschläger, T., and Markram, H. (2002) "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, Vol. 14, No. 11, pp. 2531–2560.

Mahadevan, Sridhar (2005) "Samuel Meets Amarel: Automating Value Function Approximation using Global State Space Analysis," in *Proceedings of the National Conference on Artificial Intelligence (AAAI-2005).*

Miller, E.K., Nieder, A., Freedman, D.J., and Wallis, J.D. (2003) "Neural correlates of categories and concepts," *Current Opinion in Neurobiology*, Vol. 13, No. 2, pp. 198–203.

Nair, V. and Hinton, G. (2008) "Implicit mixtures of restricted Boltzmann machines," *Advances in Neural Information Processing Systems*, Vol. 21.

Olshausen, BA and Field, DJ (1996) "Natural image statistics and efficient coding," *Network: computation in neural systems*, Vol. 7, No. 2, pp. 333–339.

Ormoneit, D. and Sen, S. (2002) "Kernel-based reinforcement learning," *Machine Learning*, Vol. 49, No. 2, pp. 161–178.

Pelli, D.G. and Farell, B. (1999) "Why use noise?," *Journal of the Optical Society of America A*, Vol. 16, No. 3, pp. 647–653.

Puskorius, GV and Feldkamp, LA (1994) "Neurocontrol of nonlinear dynamical systems with Kalman filtertrained recurrent networks," *IEEE Transactions on Neural Networks*, Vol. 5, No. 2, pp. 279–297.

Rainer, G., Augath, M., Trinath, T., and Logothetis, N.K. (2001) "Nonmonotonic noise tuning of BOLD fMRI signal to natural images in the visual cortex of the anesthetized monkey," *Current Biology*, Vol. 11, No. 11, pp. 846–854.

Rao, R.P.N. and Ballard, D.H. (1999) "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects," *nature neuroscience*, Vol. 2, pp. 79–87.

Rummery, G. and Niranjan, Mahesan (1994) "On-line Q-learning using connectionist systems,"Technical report, Cambridge Univeristy Engineering Department.

Salakhutdinov, R. and Hinton, G. (2009) "Deep Boltzmann Machines," in *Proceedings of The Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS09)*, Vol. 5, pp. 448–455.

Sallans, B. and Hinton, G. E. (2000) "Using Free Energies to Represent Q-values in a Multiagent Reinforcement Learning Task," in *NIPS*, pp. 1075–1081: MIT Press.

Sallans, B. and Hinton, G. E. (2004) "Reinforcement Learning with Factored States and Actions," *Journal of Machine Learning Research*, Vol. 5, pp. 1063–1088.

Santamaria, J., Sutton, R., and Ram, Ashwin (1998) "Experiments with reinforcement learning in problems with continuous state and action spaces," *Adaptive Behavior*, Vol. 6, pp. 163–217.

Shannon, C. E. (1948) "A Mathematical Theory of Communication," *The Bell System Technical Journal*, Vol. 27, pp. 379–423.

Sigala, N. and Logothetis, N.K. (2002) "Visual categorization shapes feature selectivity in the primate temporal cortex," *Nature*, Vol. 415, pp. 318–320.

Smallwood, R. D. and Sondik, E. J. (1973) "The optimal control of partially observable Markov processes over a finite horizon," *Operational Research*, Vol. 21, pp. 1071–1088.

Sprague, Nathan (2007) "Basis Iteration for Reward Based Dimensionality Reduction," in *the 2007 International Conference on Development and Learning in London, England.*

Sutskever, I. and Hinton, GE (2007) "Learning multilevel distributed representations for high-dimensional sequences," in *Proceeding of the Eleventh*

International Conference on Artificial Intelligence and Statistics, pp. 544–551.

Sutskever, Ilya and Hinton, Geoffrey (2010) "Temporal-Kernel Recurrent Neural Networks," *Neural Networks*, Vol. 23, pp. 239–243.

Sutton, R. S. and Barto, A. G. (1998) *Reinforcement Learning*: MIT Press.

Tieleman, T. (2008) "Training restricted Boltzmann machines using approximations to the likelihood gradient," in *Proceedings of the 25th international conference on Machine learning*, pp. 1064–1071, ACM New York, NY, USA.

Watkins, Charles and Dayan, Peter (1992) "Technical Note Q-Learning," *Machine Learning*, Vol. 8, p. 279.

Welling, M., Rosen-Zvi, M., and Hinton, G. (2005) "Exponential family harmoniums with an application to information retrieval," *Advances in neural information processing systems*, Vol. 17, pp. 1481–1488.

Werbos, P. J. (1990) "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, Vol. 78, No. 10, pp. 1550–1560.

Whitehead, S. D. and Lin, L. J. (1995) "Reinforcement learning of non-Markov decision processes," *Artificial Intelligence*, Vol. 73, pp. 271–306.

Williams, R.J. and Zipser, D. (1989) "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, Vol. 1, No. 2, pp. 270–280.