An Exploration of Predictive Representations of State

by

Chen Ma

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

The predictive representations hypothesis is that representing the state of the world in terms of predictions about the future will result in good generalization. In this thesis, good generalization is specifically quantified by good learning performance in both accuracy and speed when predicting future experiences of interest. We test the predictive representations hypothesis in different scenarios where predictions of interest vary. We observe that the predictive representations hypothesis does hold in specific scenarios. Inspired by this finding, we propose *Predictive State Update* (PSU), a state update rule that incrementally computes the next state from the current state, while being *aware* of current predictions of interest in addition to next increment of experiences. *Any* existing state representation approach can instantiate the PSU if it summarizes the past incrementally, updating the next state based on the current state and next increment of experiences. We empirically demonstrate that (i) the use of PSU can boost the generalization performance of existing state representation approaches, such as those based on simple recurrent neural networks, LSTM (Long Short-Term Memory) networks, and GRU (Gated Recurrent Units) networks, and (ii) these instantiations of PSU outperform approaches which represent states exclusively using predictions.

To my parents

Acknowledgements

I would like to thank my supervisor, Richard Sutton. His persistent enthusiasm in AI always encourages me to work harder in the face of difficulties. He teaches me how to turn an idea into a rigorous scientific study. He makes me realize that writing is thinking.

I would like to thank my examining committee, Michael Bowling and Adam White, for reading my thesis and providing insightful feedback.

I would like to thank Chenjun Xiao, Kris De Asis, Matthew Schlegel, and Huizhen Yu for improving my work through in-depth discussions.

I would like to thank Dylan Ashley, Alex Kearney, Eric Graves, Banafsheh Rafiee, Shibhansh Dohare, and Parash Rahman for polishing my thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Predictive knowledge represents information about the world by a set of predictions, the expected outcomes resulting from possible interactions. The *Predictive state representations* approach links predictive knowledge to state representations by directly representing the state in terms of predictions about the future. In this thesis, we explore the relationship between predictive knowledge and state representations. Our first contribution is a test of the hypothesis that predictive states are useful for acquiring predictive knowledge in the environment. Our second contribution is a novel approach utilizing predictions to boost the performance of a broad class of existing state representations approaches which seeks to acquire predictive knowledge.

1.1 Predictions as knowledge

Humans accumulate knowledge about the world through interactions with the environment. For example, when it comes to the question of what a ballpoint pen is, one may recall several manipulations and outcomes while interacting with it. From these experiences, one might have expectations, or *predictions*, about what the outcomes of these manipulations might be, and define a ballpoint pen in terms of them. If I trace a ballpoint pen on a piece of paper, there could be a mark left on the paper. If I release the ballpoint pen from

my hand, the pen could drop on the floor. This knowledge representation is referred to as *predictive knowledge*, and is described by a set of expected outcomes resulting from possible interactions.

1.2 State Representations

In reinforcement learning, we typically formalize an agent's interactions with an environment in a way that, at each time step, the agent is in some *environment state*. Given this state, the agent takes an *action* and ends up in a new environment state. In many real-world cases, the agent often can only acquire an *observation* containing partial information of the environment state at each step. Take an example of a mouse exploring a maze: only based on its current sense of surrounding obstacles, the mouse doesn't know for certain its exact location in the maze, and whether the path ahead can lead to an exit. If it could keep track of where it has previously been in the maze, it would have a better idea of where it is and where it should go next. This understanding of where one is in an environment can be formalized as a *state representation*, which summarizes past experiences in a way that contains sufficient information for an agent to base its decisions on. Instead of taking a whole trajectory of the past experiences as inputs, state representations often summarize the past *incrementally*, updating the next state based on the current state and next increment of experiences. State representations also refer to *agent states*, which can be informally understood as an agent's subjective approximation of the environment state. Unlike the environment state which has complete information for predicting *anything* about the environment, the agent state is only required to contain *sufficient* information for an agent's purpose.

1.3 Predictive Representations of State

The *predictive state representations* approach links predictive knowledge to state representations by directly representing the state of the environment in terms of predictions about the future. In this work, we focus on the problem of using state representations to acquire predictive knowledge of interest. Note that predictions making up the state generally can differ from the predictions of interest, but here, we only consider using predictions of interest to form the state to avoid the problem of discovering which set of predictions would be most useful to include in the state.

The *predictive representations hypothesis* suggests that representing the state of the world in terms of predictions about the future will result in good generalization. In this thesis, *representing the state of the world in terms of predictions* means using predictions explicitly in the state. *Good generalization* is measured by the learning performance in both accuracy and speed when making multiple predictions of interest. A good state representation should not only be able to capture sufficient information to compute the next agent state, which will be used to make the predictions of interest, but also avoid carrying redundant information that can slow down learning. Note that the use of *function approximators* may influence the generalization performance when using state representations. This is because function approximators are commonly used to acquire state representations and make predictions, and different function approximators may result in entirely different learning performance.

1.4 Contributions

Our first contribution is to test the predictive representations hypothesis. In the test, we compare the generalization performance of the predictive approach

with that uses *hidden* states in simple RNNs (Recurrent Neural Networks) as state representations. Here, simple RNNs are also known as Elman networks (Elman, 1990), which are probably the simplest form of RNNs whose hidden states are fed back into themselves during the next step of input. Since many factors may affect the generalization performance when using state representations, we develop several strategies to ensure the fairness of the test. (1) We design four scenarios where the predictions of interest vary, and especially consider the cases where the predictions making up the states is insufficient to compute the next agent states. (2) We emphasize the role of the state representations by performing the test in environments where the state of the world is not fully exposed to the agent. (3) We perform the test using a variety of function approximators, each of which has different capacities. This is to alleviate the effect of using function approximators for computing state representations and predictions. We observe that the predictive representations hypothesis does hold in specific scenarios.

Our second contribution is a novel state update rule, *Predictive State Update* (PSU), that incrementally computes the next state from the current state, while being *aware* of current predictions of interest in addition to next increment of experiences. *Any* existing state representation approach can instantiate the PSU if it summarizes the past incrementally, updating the next state based on the current state and next increment of experiences. We empirically demonstrate that (i) the use of PSU can boost the generalization performance of existing state representation approaches, such as those based on simple recurrent neural networks, LSTM (Long Short-Term Memory) networks, and GRU (Gated Recurrent Units) networks, and (ii) these instantiations of PSU outperform approaches which represent states exclusively using predictions.

1.5 Outline

In Chapter 2, we introduce how to learn multiple predictions using state representations. We then test the predictive representations hypothesis by comparing the predictive approach with the approach based on simple RNNs for learning predictions of interest in Chapter 3. Based on the results of the test, we introduce a novel state update rule PSU that incrementally updates the next state from the current state, while being aware of current predictions, and demonstrate PSU's learning performance when instantiated by state representations approaches based on simple RNNs, LSTM networks, and GRU networks, and additionally perform an ablation study of the algorithm in Chapter 4. In Chapter 5, we introduce existing works related to our exploration of predictive states in this thesis. In Chapter 6, we conclude the contributions of this thesis.

Chapter 2

Background

This chapter introduces the preliminary knowledge for the following chapters. In Section 2.1, we define prediction and explain how to learn predictions by temporal difference (TD) methods if environment states are available. In Section 2.2, we introduce the concept of interrelated predictions with an example and introduce how to represent and learn several interrelated predictions using TD networks and its extension, TD networks with options. In Section 2.3, we introduce how to construct the state through interactions with an environment if environment states are not accessible. In Section 2.4, we present how to learn interrelated predictions while constructing state representations from interactions with an environment.

2.1 Learning Predictions by TD methods

We formalize the interaction between the agent and the environment as a discrete dynamical system (DDS). At time step t, the agent in *state* $S_t \in \mathcal{S}$ takes an *action* A_t from action space $\mathcal{A}$ according to policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. At the next time step, the agent transits to the next state S_{t+1} according to a transition probability $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ of the environment and receives an *observation* $O_t \in \mathcal{O}$ according to an observation function $z : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \to [0, 1]$ of the environment. The state in DDS is also referred to as the *environment*

state. Note that the algorithms, such as temporal-difference methods and temporal difference networks with options, for learning predictions in RL are introduced in the case assuming the agent has access to the environment state. However, in many real-world cases, the environment state is not available to the agent. We will discuss this case in Section 2.3 and Section 2.4.

An agent expresses the prediction as discounted cumulative sensory signals following a policy. Here, we use *cumulant* $C_t \in \mathbb{R}$ as general form of observations. The prediction is accordingly defined as the *general value function* (GVF),

$$v_{\pi,\gamma,C}(s) \doteq \mathbb{E}\left[\sum_{k=t}^{\infty} C_{k+1} \prod_{i=t+1}^{k} \gamma(S_i) \middle| S_t = s, A_{t:\infty} \sim \pi\right],$$

where $\gamma : \mathcal{S} \to [0, 1]$ is a generalized form of discounting, which determines the horizon of the summation (Sutton *et al.*, 2011).

Temporal-difference (TD) method is a widely-used method to learn the prediction in a bootstrap way — the estimates are updated based on other estimates, without waiting until the final outcome (Sutton and Barto, 2018). We can approximate the prediction $v(s)$ by parametrizing it as $\hat{v}_\theta(s)$ and learning θ using the temporal difference update rule,

$$\theta \leftarrow \theta + \alpha[Z_t - \hat{v}_\theta(S_t)]\nabla_\theta(\hat{v}_\theta(S_t)), \tag{2.1}$$

where

$$Z_t = C_{t+1} + \gamma(S_{t+1})\hat{v}_\theta(S_{t+1}).$$

Here, the step size is denoted by α, and Z_t is the target for the state value $\hat{v}_\theta(S_t)$.

2.2 Learning Interrelated Predictions by TDO

Temporal difference networks with options (TDO) extend TD methods by considering the interrelationship among a set of predictions, rather than treating

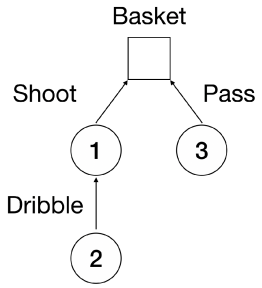


Figure 2.1: The question networks for the example of basketball game. Each circular node indicates one prediction. The square node indicates the cumulant, which is scoring a basket in this case. The directed edge indicates the compositionality conditional on a course of actions extending over several steps. The first prediction is based on the cumulant of scoring a basket if I shoot the ball. The second prediction is based on the first prediction if I dribble the ball up the court. The third prediction is based on the cumulant of scoring a basket if I shoot the ball.

each prediction independently (Rafols *et al.*, 2006). In this chapter, we firstly informally describe TDO with an example. Then we explain the formalization of TD networks, which is the precursor of TDO. Finally, we introduce how to use TDO to represent and learn multiple interrelated predictions.

Example Take an example (Rafols, 2006) from a basketball game. The predictive knowledge consists of three predictive questions as follows.

- If I shoot the ball, could I score a basket?
- If I dribble the ball up the court and then shoot it, could I score a basket?
- If I pass the ball, could I score a basket?

The second prediction is equivalent to if I dribble the ball up the court, what would the value of the first prediction be. That is, the prediction of the second question is based on the first prediction if I dribble the ball up the court. We term this relationship as *compositionality*. Note that compositionality can be extended to the cumulant. For example, the first prediction is based on the cumulant of scoring a basket if I shoot the ball.

Temporal difference networks with options (TDO) represent the interrelationship among the predictions using *question networks* (Rafols *et al.*, 2006). We illustrate the question networks for the example of the basketball game in Figure 2.1. Each circular node indicates one prediction: circular node one, two, and three correspond to the first, second, and third prediction, respectively. The square node indicates the cumulant, which is scoring a basket in this case. The directed edge indicates the compositionality conditional on a course of actions extending over several steps. The first prediction is based on the cumulant of scoring a basket if I shoot the ball. The second prediction is based on the first prediction if I dribble the ball up the court. The third prediction is based on the cumulant of scoring a basket if I shoot the ball. It is obvious that the first and third prediction can be learned through temporal difference methods. The second prediction can be learned by TDO in the way of treating the first prediction as the cumulant in the TD method.

TD Networks TD networks (Tanner and Sutton, 2005) is the precursor of TDO. TD networks represent the relationship of multiple predictions in *question networks*. Each circular node in question networks denotes a prediction and therefore the whole network represent multiple interrelated predictions.

When it comes to learning these interrelated predictions, TD networks use a TD update rule (2.1) with the target defined by the question networks, whereas TD methods use the value of the next state as the target. Suppose there are k interrelated predictions represented in the question network, we can approximate the i-th prediction of a state s by parametrizing it as $\hat{y}^i_\theta(s)$. The target to approximate the i-th prediction is

$$Z^i_t = u^i(O_{t+1}, \hat{y}^1_\theta(S_{t+1}), \hat{y}^2_\theta(S_{t+1}), ..., \hat{y}^k_\theta(S_{t+1})),$$

where $u^i : \mathcal{O} \times \mathbb{R}^k \to \mathbb{R}$ defines how the i-th prediction related to the observation and multiple predictions. Note that this interrelationship among

predictions is represented in question networks. The agent follows a *behaviour policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. The update is only performed when the current action matches the conditional action of the prediction. We can learn θ using the temporal difference update rule as follow,

$$\theta \leftarrow \theta + \alpha c^i(A_t)[Z_t^i - \hat{y}_\theta^i(S_t)]\nabla_\theta(\hat{y}_\theta^i(S_t)),$$

where $c^i : \mathcal{A} \to [0, 1]$ determines whether the i-th prediction is consistent with the agent's action, α is the step size, and Z_t^i is the target for the prediction $\hat{y}_\theta^i(S_t)$.

TD Networks with Options TD networks with options (TDO) extend TD networks in a way that each prediction is defined as the expected target conditioning on a course of actions rather than only one action. The option framework (Sutton *et al.*, 1999) provides a way to represent a course of actions extending over several steps, which is a generalized form of the action. The option framework assumes the agent has access to the environment state from the state space $\mathcal{S}$. The option consists of three components: a set of initiation states $\mathcal{I} \in \mathcal{S}$, which specifies the states where the option is available, a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, by which the agent will follow when the agent is in the option, and a termination condition $\beta : \mathcal{S} \to [0, 1]$, which determines whether the agent will exit the current option and start to select a new option. Note that the primitive action is a special case of the option where the initiation set is the whole state space, the policy is the primitive action for each state, and the termination condition is always 1.

When it comes to learning these interrelated predictions, TDO use TD update rule (2.1) with the target defined by the question networks. As a contrast, TD methods use the value of the next state as the target. Suppose there are k interrelated predictions represented in question networks, we can approximate the i-th prediction of a state s by parametrizing it as $\hat{y}_\theta^i(s)$. The

target Z_t^i to approximate the i-th prediction is as follows:

$$Z_t^i = \beta^i(O_{t+1}, S_t)U_t^i + [1 - \beta^i(O_{t+1}, S_t)]\hat{y}_\theta^i(S_{t+1}), \tag{2.2}$$

where

$$U_t^i = u^i(O_{t+1}, \hat{y}_\theta^1(S_{t+1}), \hat{y}_\theta^2(S_{t+1}), ..., \hat{y}_\theta^k(S_{t+1})). \tag{2.3}$$

Here, similarly to TD networks, $u^i : \mathcal{O} \times \mathbb{R}^k \to \mathbb{R}$ defines how the i-th prediction related to the observation and multiple predictions. Note that this interrelationship among predictions is represented in question networks. Whether the option is terminated at time step $t + 1$ is indicated by $\beta^i(O_{t+1}, S_t)$. If $\beta^i(O_{t+1}, S_t) = 1$, then the target is U_t^i that is related to the observation and multiple predictions. This relationship is specified in the question network. If $\beta^i(O_{t+1}, S_t) = 0$, then the target is the prediction itself at the next step. The agent follows a *behaviour policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. The update is only performed when the current action matches the option's policy of the prediction. We can learn θ using the temporal difference update rule as follow,

$$\theta \leftarrow \theta + \alpha\mu^i(A_t, S_t)[Z_t^i - \hat{y}_\theta^i(S_t)]\nabla_\theta(\hat{y}_\theta^i(S_t)),$$

where $\mu_t^i : \mathcal{A} \times \mathcal{S} \to [0, 1]$ determines whether the option of the prediction y_t^i is being followed (that is, whether the option's policy is consistent with the current action), the step size is α, and Z_t^i is the target for the prediction $\hat{y}_\theta^i(S_t)$.

2.3 State Representations

From preceding discussions, we introduced how to learn interrelated predictions by temporal difference networks with options in the case assuming the environment state is available to the agent. Specifically, the learned approximate value functions are written as functions of the environment state. In this section, we will firstly explain the concept of Markov states and environment states. We then describe the general form of state representations, which can

be used as the environment states when they are not available to an agent. Finally, we introduce two state representation approaches, the predictive representations of state approach, and the approach based on simple recurrent neural networks.

In many real-world cases, an agent often can only acquire an *observation* containing partial information of the environment state at each step. As a result, an agent will only experience a stream of interleaved actions and observations, $A_0, O_1, A_1, O_2, ...$ Take an example of a mouse exploring a maze: only based on its current sense of surrounding obstacles, the mouse doesn't know for certain its exact location in the maze, and whether the path ahead can lead to an exit.

The *Markov state* is a summary of the past experiences that contains sufficient information to predict anything about the environment. We will formally explain the concept of the Markov state as follow. The *past experiences* is a trajectory of interleaving actions and observations defined as history,

$$H_t = A_0, O_1, A_1, O_2, ..., A_{t-1}, O_t.$$

The Markov state *summarizes* the past experiences as a function of history $S_t = f(H_t)$. *To contain sufficient information to predict anything about the environment* means any two histories h and h', that are mapped to the same state with the function f, will have the same probability of arbitrary future experiences given the same sequence of actions as follow

$$\begin{aligned} & f(h) = f(h') \Longrightarrow \\ & \prod_{i=t+1}^{t+k} \Pr\{O_i = o_i | H_t = h, A_t = a_t, O_{t+1} = o_{t+1}, ..., A_{i-1} = a_{i-1}\} \\ = & \prod_{i=t+1}^{t+k} \Pr\{O_i = o_i | H_t = h', A_t = a_t, O_{t+1} = o_{t+1}, ..., A_{i-1} = a_{i-1}\}. \end{aligned} \tag{2.4}$$

In contrast with the Markov state, the environment state not only contains sufficient information to predict anything about the environment, but also can

contain more information other than a summary of history.

In the example of a mouse exploring a maze, if the mouse could maintain a summary of past observations as it rolls forward in time, that is, incrementally update the current summary of past based on the previous summary and previous observations, it would have a better idea of where it is, and where it should go next. This understanding of where one is in an environment can be formalized as a *state representation*, which summarizes past experiences incrementally in a way that contains sufficient information for an agent to base its decisions on. More specifically, state representations incrementally summarize the past in the way of updating the current summary of the past based on the previous summary of the past, the previous action, and the current observation, as in

$$S_t = u(S_{t-1}, A_{t-1}, O_t), \tag{2.5}$$

where the function u is called the *state-update* function. Note that a state update function is a special [1] way of representing the state as a summary of history $S_t = f(H_t)$. In this case, we can simplify (2.4) into one step future experience,

$$\begin{aligned} &f(h) = f(h') \Longrightarrow \\ &\Pr\{O_{t+1} = o_{t+1} | H_t = h, A_t = a_t\} = \Pr\{O_{t+1} = o_{t+1} | H_t = h', A_t = a_t\}. \end{aligned} \tag{2.6}$$

This can be proved by rolling out through future experiences while repeatedly applying (2.6) and (2.5). State representations are often referred to as *agent states*, which can be informally understood as an agent's subjective approximation of the Markov state. Unlike the Markov state which has complete

[1] For example, in an environment where the agent can only sense three observations denoted 1,2,and 3. In the case representing the state as a summary of history where $f(11) = f(12)$ and $f(113) \neq f(123)$, one can not represent it using a state-update function. This is because state-update function has a limitation the the next state representation must be the same if the current state representation and the next increment of experiences are the same. In contrast, representing states with history does not has such limitation.

information for predicting *anything* about the environment, the agent state is only required to contain *sufficient* information for an agent's purpose.

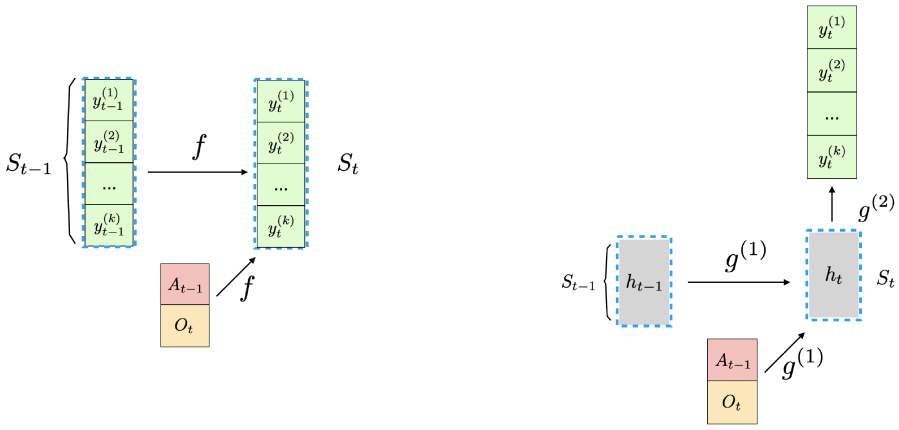
The *predictive state representations* approach directly represents the state of the environment in terms of predictions about the future. The idea of representing the state with predictions is based on the hypothesis that states with the same predictions about the future will be the same (Littman and Sutton, 2002). In this work, predictions making up the state are defined as interrelated general value functions in TDO, and we focus on the problem of using state representation to predict future experiences of interest. Note that predictions making up the state can generally differ from the predictions of interest, but here, we only consider using predictions of interest to form the state to avoid the problem of discovering which set of predictions would be most useful to include in the state. As illustrated in Figure 2.2a, predictive state S_{t-1} is represented by a concatenation of k predictions of interests y^i_{t-1} as follows:

$$S_{t-1} = [y^1_{t-1}, y^2_{t-1}, ..., y^k_{t-1}].$$

The initial state S_0 is set to a vector of zeros. The state S_t is a mapping of the previous state S_{t-1}, previous action A_{t-1}, and current observation O_t with the function f. Note that the state update proceeds simultaneously with making predictions y^i_t as follows:

$$y^i_t = f^i(S_{t-1}, A_{t-1}, O_t).$$

Alternatively, the *hidden* states in simple RNNs (Recurrent Neural networks) can be used as state representations. Here, simple RNNs are also known as Elman networks (Elman, 1990), which are probably the simplest form of RNNs whose hidden states are fed back into themselves during the next step of input. In the typical sequence modeling scenario, the hidden states in RNNs are often used to summarize history by minimizing the prediction error for



(a) Structure of Predictive State (b) Structure of simple RNNs

Figure 2.2: (a) An illustration of the predictive state approach. The predictive state S_{t-1} is represented by a concatenation of predictions y_{t-1}^i. The state S_t is a mapping of previous state S_{t-1}, previous action A_{t-1}, and current observation O_t with the function f. This state update proceeds simultaneously with making predictions y_t^i. (b) An illustration of simple RNNs. The hidden state S_t is a mapping of previous state S_{t-1}, previous action A_{t-1}, and current observation O_t with the function $g^{(1)}$. The predictions y_t^i is a mapping of the hidden state S_t with the function $g^{(2)}$.

the next observation. In contrast, here we use the hidden states in RNNs to learn accurate predictions of interest. As illustrated in Figure 2.2b, the hidden state S_t is a mapping of previous state S_{t-1}, previous action A_{t-1}, and current observation O_t with the function $g^{(1)}$

$$S_t = g^{(1)}(S_{t-1}, A_{t-1}, O_t).$$

The initial state S_0 is set to a vector of zeros. The predictions $y_t^{(i)}$ is a mapping of hidden state S_t with the function $g^{(2)}$ as follows:

$$y_t^i = g^{(2),i}(S_t).$$

2.4 Putting it All Together

In previous sections, we first explained how to learn interrelated predictions by TD methods when environment states are available to the agent, and then

we introduced how to represent the state when the environment state is not fully exposed to an agent. In this section, we put these two together and illustrate how to learn interrelated predictions of interest with the use of state representations. We can approximate the next state s_t by parametrizing it as $f_{\theta_2}(\hat{s}_{t-1}, a_{t-1}, o_t)$, and approximate each prediction $y^i(s)$ by parametrizing it as $\hat{y}^i_{\theta_1}(s)$. Accordingly, TDO learns interrelated predictions by replacing the environment state in (2.3) and (2.2) with the approximated state representation as follows:

$$U^i_t = u^i(O_{t+1}, \hat{y}^1_{\theta_1}(\hat{s}_{t+1}), \hat{y}^2_{\theta_1}(\hat{s}_{t+1}), ..., \hat{y}^k_{\theta_1}(\hat{s}_{t+1})),$$

$$Z^i_t = \beta^i(O_{t+1}, \hat{s}_t)U^i_t + [1 - \beta^i(O_{t+1}, \hat{s}_t)]\hat{y}^i_{\theta_1}(\hat{s}_{t+1}), \tag{2.7}$$

$$\theta_1 \leftarrow \theta_1 + \alpha\mu^i(A_t, \hat{s}_t)[Z^i_t - \hat{y}^i_{\theta_1}(\hat{s}_t)]\nabla_{\theta_1}(\hat{y}^i_{\theta_1}(\hat{s}_t)), \tag{2.8}$$

$$\theta_2 \leftarrow \theta_2 + \alpha\mu^i(A_t, \hat{s}_t)[Z^i_t - \hat{y}^i_{\theta_1}(\hat{s}_t)]\hat{y}^i_{\theta_1}(\hat{s}_t)\nabla_{\theta_2}(\hat{s}_t). \tag{2.9}$$

Here, u^i, β^i, and μ^i are defined in the same way as in Section 2.2. We illustrate the pseudo-codes in Algorithm 1.

Note that we need to compute gradient $\nabla_{\theta_2}(\hat{s}_t)$ in a recursive way, because state update is performed recursively as in $\hat{s}_t = f_{\theta_2}(\hat{s}_{t-1}, a_{t-1}, o_t)$. There are two major methods for revolving this issue. (1) Backpropagation through time (BPTT) directly computes gradient tracing back through time (Rumelhart *et al.*, 1985). The time complexity of BPTT is $O(n)$, if it is performed n steps backward through time. Additionally, performing BPTT along a long history is difficult (Bengio *et al.*, 1994). An approximate solution is to truncate the computing of the gradient into k time steps back in history, namely $k-$BPTT. (2) Real-time recurrent learning (RTRL) is proposed to compute the gradient without tracing backward through history (Williams and Zipser, 1989). However, RTRL has a space complexity of $O(n)$ and time complexity of $O(n^2)$, where n is the number of weights needed to compute the gradient. In the work, we focus on the problem of learning prediction in a time and

Algorithm 1 Learning interrelated predictions with state representations

Initial state s_0 and $t = 1$
Take action from behaviour policy, $a_0 \sim \pi$ and receive observation o_1 from environment
for each time step $t = 1, \cdots, T$ **do**
    Estimate current state, $s_t = f(s_{t-1}, a_{t-1}, o_t)$
    Compute predictions, $y_t^i = y^i(s_t)$
    Take action from behaviour policy $a_t \sim \pi$, and receive observation o_{t+1} from environment
    Estimate next state $s_{t+1} = f(s_t, a_t, o_{t+1})$
    Compute predictions $y_{t+1} = y(s_{t+1})$
    Compute termination function $\beta_{t+1}^i = \beta^i(o_{t+1}, s_t)$
    Compute target z_t^i according to (2.7).
    Perform update on θ_1, θ_2 according to (2.8) and (2.9).
    $a_{t-1} \leftarrow a_t$, $o_t \leftarrow o_{t+1}$
end for

space efficient way. Therefore, we compute the gradient using 1-BPTT that truncates the computing of the gradient into one time step back in history.

Chapter 3

Test of Predictive Representations Hypothesis

One of the main contributions of this thesis tests the *predictive representations hypothesis* that representing the state of the world in terms of predictions about the future will result in good generalization. Chapter 2 provides the preliminary knowledge for this chapter: it introduces the concept of interrelated predictions and how to learn these predictions with the use of state representations.

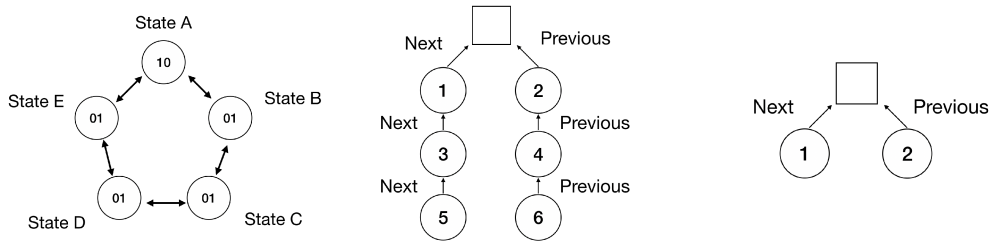
In Section 3.1, we elaborate on the predictive representations hypothesis and introduce strategies to ensure a fair test of the hypothesis. In Section 3.2, we describe four scenarios in which we perform the test. In Section 3.3, we define the metric for good generalization as it relates to the hypothesis. In Section 3.4 and Section 3.5, we explain the experiment details and analyze the results of the test. In Section 3.6, we study how the choice of hyperparameters influences the generalization performance of the approach that represents states explicitly with predictions.

3.1 Predictive Representations Hypothesis

The *predictive representations hypothesis* suggests that representing the state of the world in terms of predictions about the future will result in good gen-

eralization. In this chapter, *representing the state of the world in terms of predictions* means using predictions explicitly in the state. *Good generalization* is measured by the learning performance in both accuracy and speed when making multiple predictions of interest. A good state representation should not only be able to capture sufficient information to compute the next agent state, which will be used to make the predictions of interest, but also avoid carrying redundant information that can slow down learning. Note that the use of *function approximators* may influence the generalization performance when using state representations. This is because function approximators are commonly used to acquire state representations and make predictions, and different function approximators may result in entirely different learning performance.

In the test, we compare the generalization performance of the predictive approach with that uses hidden states in simple RNNs (Recurrent Neural Networks) as state representations. We name this the state representation approach based on simple RNNs. Here, simple RNNs are also known as Elman networks (Elman, 1990), which are probably the simplest form of RNNs whose hidden states are fed back into themselves during the next step of input. Since many factors may affect the generalization performance when using state representations, we develop several strategies to ensure the fairness of the test. (1) We design four scenarios where the predictions of interest vary, and especially consider the cases where the predictions making up the states is insufficient to compute the next agent states. (2) We emphasize the role of the state representations by performing the test in environments where the state of the world is not fully exposed to the agent. (3) We perform the test using a variety of function approximators, each of which have different capacities. This is to alleviate the effect of using function approximators for computing state representations and predictions.



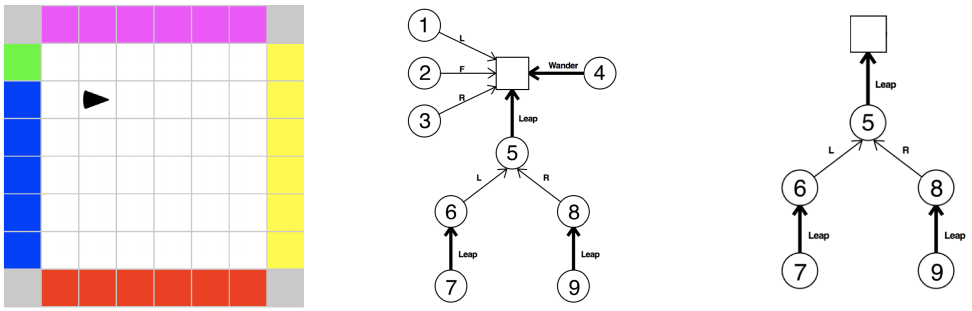
(a) 5 State Ring World (b) R-6 Question Networks (c) R-2 Question Networks

Figure 3.1: (a) Ring world contains five states of A, B, C, D and E. At each time step, the agent can take one step clockwise or counterclockwise as actions and received an observation of *01* or *10* accordingly. (b) The question networks consist of six round nodes, representing six predictions. The square indicates the cumulant, which is one bit of the observation *01* or *10*. The next and previous are primitive actions. (c) The question networks consist of two round nodes, representing two predictions. The square indicates the cumulant, which is one bit of the observation *01* or *10*. *Next* and *previous* are primitive actions.

3.2 Environments

We test the predictive representation hypothesis in the *Ring World* and *Compass World* environments (Rafols *et al.*, 2006), where the state of the world is only partially observable by the agent.

In the Ring world with five states, as shown in Figure 3.1a, an agent can take actions *next*, and *previous*. Action next causes the agent to move one step clockwise. The action previous causes the agent to move one step counterclockwise. At each time step, the agent receives a one-hot vector, which is either *01* or *10*, as an observation. In the *R-6* scenario, the goal for the agent is to make six predictions in the Ring World environment. We expect these six predictions can capture sufficient information to compute the next agent state. We illustrate these six predictions as six round nodes in question networks shown in Figure 3.1b. Node 1, 3, and 5 predict the expected observation when taking one step of next, two steps of next, and three steps of next, respectively. Similarly, nodes 2, 4, and 6 predict the



(a) Compass World (b) C-9 Question Networks (c) C-5 Question Networks

Figure 3.2: (a) An illustration of the Compass World environment. Compass world is the environment used for the experiments. The egocentric agent, indicated by the triangle, is able to take actions of left, right, and forward. The walls are colored cells, which the agent can't move into. (b) Question networks in the C-9 scenario. The question networks consist of nine nodes, representing nine predictions. The square indicates the cumulant, ranging from red, orange, yellow, green, and blue. L, F, and R are primitive actions left, forward, and right. Leap and Wander are options. (c) Question networks in the C-5 scenario. The question networks consist of five nodes, representing five predictions. The square indicates the cumulant, ranging from red, orange, yellow, and blue. L and R are primitive actions left and right. A leap is an option.

expected observation when taking one step of previous, two steps of previous, and three steps of previous, respectively. In the *R-2* scenario, we design the goal for an agent as making two predictions in the Ring World environment, and these two predictions are insufficient to compute the next agent state. We illustrate these two predictions as two round nodes in question networks shown in Figure 3.1c. Node 1 predicts the expected observation when taking one step of next, whereas node 2 predicts the expected observation when taking one step of the previous.

In Compass world (Rafols *et al.*, 2006), as shown in Figure 3.2a, an agent can take actions *forward*, *left*, and *right*. Action forward causes the agent to move one step forward unless it senses the wall, in which case it doesn't move. The action left, and right causes the agent to rotate 90 degrees to the left and right, respectively. At each time step, the agent receives a one-hot

vector indicating the cell color in front of it. There are six colors in total: red, green, blue, magenta, yellow, and white, as shown in Figure 3.2a. In the *C-9* scenario, we design the goal for an agent as making nine predictions in the Compass World environment, and these predictions may not be sufficient to compute the next agent state. We illustrate these nine predictions as nine nodes in question networks shown in Figure 3.2b. Node 1, 2, and 3 predict the expected observation when taking a primitive action left, forward, or right. Node 4 estimates the expected outcome if the agent follows the behavior policy and terminates when a wall is sensed or spontaneously with probability 0.5. Node 5 estimates the expected outcome if the agent takes a *leap*, during which it continuously takes action forward until a wall is sensed. Node 6, 7, 8, and 9 predict the expected outcome as the agent follows the sequence of actions left-leap, leap-left-leap, right-leap, and leap-right-leap, respectively. In the *C-5* scenario, we design the goal for an agent as making five predictions in the environment of Compass World, and expect these predictions are sufficient to compute the next agent state. We illustrate these predictions as five nodes in question networks shown in Figure 3.2c. We keep the prediction nodes of 5,6,7,8, and 9 in the scenario of *C-5* and replace the green cells of the wall by blue ones.

3.3 Error Metric

The generalization performance of a state representation approach is measured in terms of the learning performance in both accuracy and speed when making interrelated predictions of interest. The quality of estimated predictions is measured in comparison with the ground truth from an oracle. At each time step t, we compute Root Mean Square Error (RMSE) between the ground

truth and the estimated one as follows,

$$error_t = \sqrt{\frac{1}{m}\sum_i (y_t^i - y_{true,t}^i)^2},$$

where $y_{true,t}^i$ is the ground true for each prediction y_t^i and m is the number of interrelated predictions. We illustrate the performance of learning predictions in the aspects of mean and standard error as follow,

$$\mu_t = \frac{1}{n}\sum_{k=1}^{n} error_t^k$$

$$\sigma_t = \frac{1}{\sqrt{n}}\sqrt{\frac{\sum_{k=1}^{n}(error_t^k - \mu_t)^2}{n-1}},$$

where n is the number of repeats of the experiment.

3.4 Experiment Details

In the test, we compare the generalization performance of the predictive approach with the approach based on simple RNNs. We sweep over different combinations of step size and function approximator capacity, and we then select the one with the best performance for each algorithm. Specifically, we first compute the area under each learning curve by averaging the RMSE every one thousand time steps. Then we select the hyper-parameter with the lowest area under the learning curve. As shown in Figure 2.2a, the mapping f is a two-layer fully connected neural networks with a sigmoid activation function, which is also applied to the outputs layer. As shown in Figure 2.2b, the mapping $g^{(1)}, g^{(2)}$, are two separate single-layer fully connected neural networks followed by a sigmoid activation function. Note that the dimension of the hidden layer in f and the dimension of hidden states in simple RNNs will influence the capacity of function approximators. For convenience, we refer to this as the *dimension of the approximator*. Table 3.1 summarizes the combinations of the dimension of the approximator and step size used for the test in different

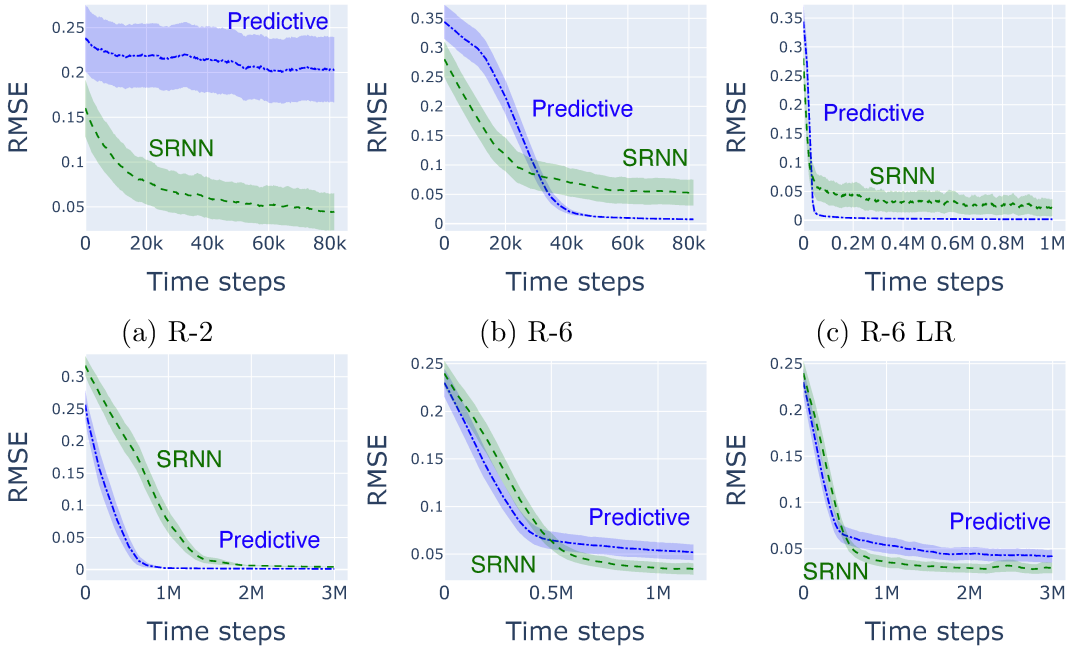
Scenario	Step Size	Dims of Approximators
Predictive, R-2	(0.01 0.1 1.0 10.0)	(6 20 39 65 96 135 230 350)
Simple RNNs, R-2	(0.01 0.1 1.0 10.0)	(5 10 15 20 25 30 40 50)
Predictive, R-6	(0.01 0.1 1.0 10.0)	(5 12 23 37 75 125 145 187 262)
Simple RNNs, R-6	(0.01 0.1 1.0 10.0)	(5 10 15 20 30 40 50 60)
Predictive, C-5	(0.01 0.1 1.0 10.0)	(7 18 32 50 71 95 189 313)
Simple RNNs, C-5	(0.01 0.1 1.0 10.0)	(0 10 20 30 40 50 60 90 120)
Predictive, C-9	(0.01 0.1 1.0 10.0)	(6 15 25 38 52 69 108 155 210)
Simple RNNs, C-9	(0.01 0.1 1.0 10.0)	(0 10 20 30 40 50 60 80 100 120)

Table 3.1: A summary of hyper-parameters used for learning predictions with the predictive approach and the approach based on simple RNNs. The underline indicates the hyper-parameter setting with the best performance for each algorithm.

scenarios. The underline indicates the hyper-parameter setting with the best performance for each algorithm.

In the ring world environment, the input observation for the agent at each time step is a two-dimensional one hot vector *01* or *10*, and the action is a two-dimensional one-hot vector, which indicates *next* and *previous*. The interrelated predictions are learned in an off-policy manner where the policy associated with the predictions is not consistent with the agent's behavior policy. The agent follows a random behavior policy μ with equal action probabilities for the actions next and previous. We perform the experiments under one million time steps of interaction with the environment. Each experiment is repeated 30 times.

In the compass world environment, the input observation for the agent at each time step is a 6-dimensional one hot vector indicating six colors as shown in Figure 3.2a, and the action is a 3-dimensional one hot vector, which indicates *forward*, *left* and *right*. The interrelated predictions are learned in an off-policy manner where the policy associated with the predictions is not consistent with the agent's behavior policy. The agent follows a random behavior policy μ with action probabilities of 0.5, 0.25, and 0.25 for the actions forward, left,



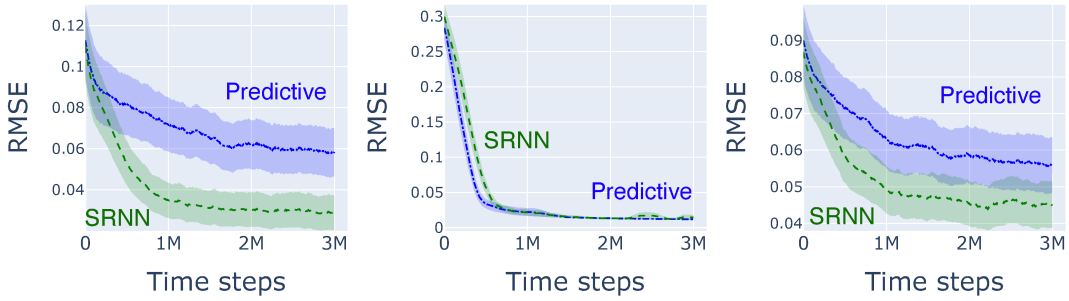
(a) R-2 (b) R-6 (c) R-6 LR

(d) C-5 (e) C-9 (f) C-9 LR

Figure 3.3: Generalization performance of the predictive approach in comparison with the approach based on simple RNNs in the R-2, R-6, C-5, and C-9 scenarios. Root mean square error is averaged over all predictions. The performance over the long run (LR) is also demonstrated if needed. Shading denotes standard error.

and right, respectively. We perform the experiments under three million time steps of interaction with the environment. Each experiment is repeated 30 times.

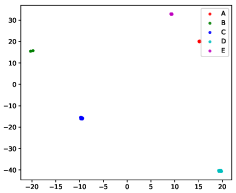
3.5 Results



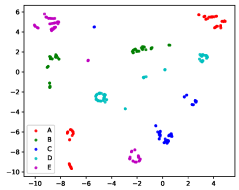
(a) C-9 Group One step (b) C-9 Group With Leap (c) C-9 Wander

Figure 3.4: Individual learning performance in the C-9 scenario. (a) Root mean square error averaged over one step predictions each of which predicts the expected observation when taking a primitive action left, forward, and right, respectively. (b) Root mean square error averaged predictions with the leap each of which predict the expected outcome as the agent follows the sequence of actions left-leap, leap-left-leap, right-leap, and leap-right-leap, respectively. (c) Root mean square error for the option of wander.

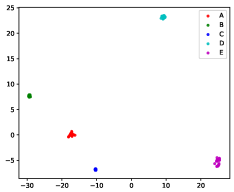
We demonstrate the generalization performance of the predictive approach in comparison with the approach based on simple RNNs in Figure 3.3. In the R-6 and C-5 scenarios, we observe the predictive representation hypothesis does hold. Why? Firstly, in these two scenarios, the predictions of interest making up the predictive state can capture sufficient information to compute the next agent state, which will be used to make the predictions of interest. Secondly, the predictive states only contain information from predictions of interest, whereas the hidden states in simple RNNs may contain redundant information that slows down learning. As an illustration, we visualize hidden states in simple RNNs and predictive states as training progresses in the R-6 scenario. We collect the data of state representations in a sequence of 500 consecutive training steps starting from step $35k$, $45k$, $65k$, and $95k$. Note that the predictive approach only outperforms the approach based on simple RNNs after $35k$ steps. We reduce the dimension of the state representations into two dimensions using *t-SNE* (Maaten and Hinton, 2008), which clusters the data points whose Euclidean distances are small with each other. As shown in Figure 3.5m, we observe that there are fewer clusters of the predictive states



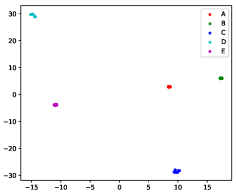
(a) Predictive 35k



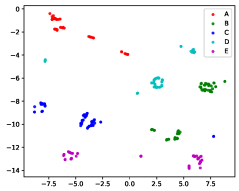
(b) Simple RNNs 35k



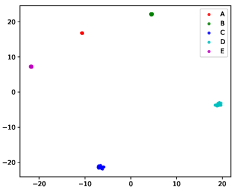
(c) Simple RNNs' Pred 35k



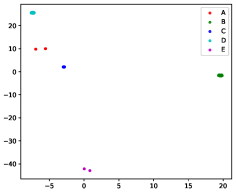
(d) Predictive 45k



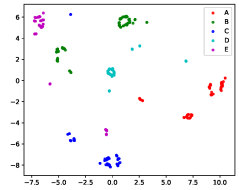
(e) Simple RNNs 45k



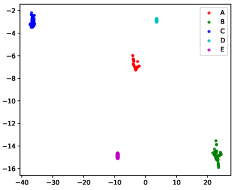
(f) Simple RNNs' Pred 45k



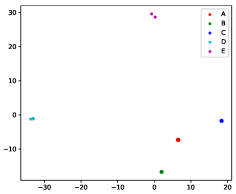
(g) Predictive 65k



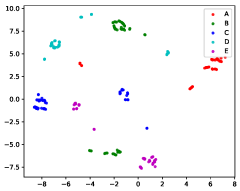
(h) Simple RNNs 65k



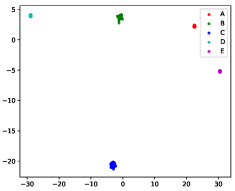
(i) Simple RNNs' Pred 65k



(j) Predictive 95k



(k) Simple RNNs 95k



(l) Simple RNNs' Pred 95k

(m) Visualization of predictive states, hidden states in simple RNNs, and predictions outputted from simple RNNs as training progress in the R-6 scenario. The data is collected in a sequence of 500 consecutive training steps starting from step $35k$, $45k$, $65k$, and $95k$. The legend of A, B, C, D, and E are five environment states in the Ring World, as shown in Figure 3.1a.

than those of the hidden states in simple RNNs. If one learns predictions in a tabular form, creating a table, each item of which corresponds to a cluster of state representations, the predictive approach can learn faster as it maintains a smaller table than the approach based on simple RNNs. Why are there fewer clusters of the predictive states? Suppose one groups the environment states by their predictions of interest. For each group of environment states, the predictive states are the same, whereas the hidden states in simple RNNs can carry extra information and thus can be completely different. In addition, any two predictive states for two different groups of environment states must be different. This also applies to the hidden states in simple RNNs.

In the R-2 and C-9 scenarios, we observe that the predictive representation hypothesis does not hold anymore, as shown in Figure 3.3. The predictions making up the predictive state can not capture sufficient information to compute the next agent state, whereas the hidden states in simple RNNs can adjust its space to obtain extra information to overcome this problem. For example in the R-2 scenario, if the predictions are learned correctly, the predictive states for the environment state C and D are the same. Here, states A, B, C, D, and E are five environment states in the Ring World environment, as shown in Figure 3.1a. After taking one step, clockwise, states C and D end up with states D and E, respectively. The predictions of interest for the state D and E differ. However, if one performs state update based on the predictive states and the action of one step clockwise, the updated predictive state of the states C and D are the same. In the C-9 scenario, if the Wander prediction is perfectly learned, predictive states do contain sufficient information for updating the next state. However, the estimate of the Wander prediction tends to have high variance and thus is not suitable to form the state.

We further analyze the learning performance averaged over a subset of predictions which share similar learning performance, referred to here as the

individual performance. We omit the individual results, which are consistent with the ones averaged over all predictions. In Figure 3.4, we illustrate individual performance for the C-9 scenario in three groups: *Group One Step*, *Group With Leap*, and *Wander*. (1) Group One Step averages result over one-step predictions, each of which predicts the expected observation when taking a primitive action left, forward, and right, respectively. (2) Group With Leap averages results over predictions with the leap, each of which predicts the expected outcome as the agent follows either the sequence of actions left-leap, leap-left-leap, right-leap, or leap-right-leap. (3) Wander is the result only for the option of Wander. It is no surprise that the approach based on simple RNNs generalizes better in one step predictions as its hidden states can adjust freely to carry extra information. In fact, predictions in the C-5 scenario share similar predictions as of Group With Leap in C-9. When comparing the individual performance of Group With Leap in the C-9 scenario, as shown in Figure 3.4b, with the overall performance in C-5, as shown in Figure 3.3d, the predictive approach's generalization performance is almost the same in these two scenarios, whereas the approach based on simple RNNs generalizes much better in the C-9 scenario. Because more predictions in C-9 can add strong supervision on the hidden states in simple RNNs and thus help it converge to decent state space.

3.6 Parameter Study

Hyper-parameters include combinations of step size and function approximator capacity. Here the capacity of the function approximator is determined by the dimension of the approximator, which refers to the dimension of the hidden layer in the predictive approach and dimension of the hidden states in simple RNNs as described in Section 3.4. To visualize the generalization performance over different hyper-parameter settings, we roughly measure the learning per-

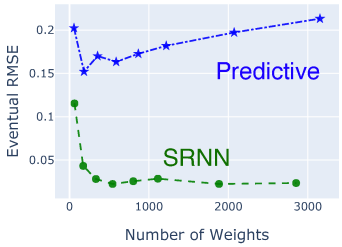
formance of speed and accuracy as the averaged RMSE during the first and last time period, respectively. More specifically, the first tenth and the last tenth of the period are selected for the ring world environment, whereas the first three tenths and the last tenth are chosen for the compass world environment. Since the predictive approach and the approach based on simple RNNs are using completely different function approximators, we summarize their performance along with the total number of weights of the approximators. The total number of weights for these two approaches is computed as,

$$N_{simpleRNNs} = (h + i + 1) * h + (h + 1) * n$$

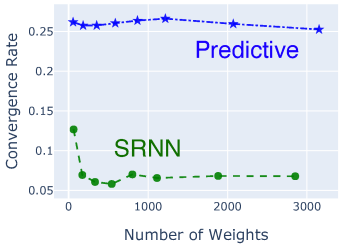
$$N_{predictive} = (n + i + 1) * h + (h + 1) * n$$

where n is number of predictions of interest, h is the dimension of approximators, and i is the dimension of the inputs, which includes one step action and observation.

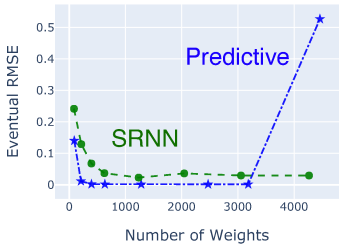
We first study how the capacity of function approximators influences the generalization performance of the predictive approach and the approach based on simple RNNs while step size is optimal. As shown in Figure 3.6, the predictive approach achieves decent generalization performance with less number of weights in function approximators. We also observe that the approach based on simple RNNs is less sensitive to the number of weights in function approximators. We then investigate how the step size affects the generalization performance of the predictive approach and the approach based on simple RNNs while the capacity of function approximators vary. As shown in Figure 3.7i and Figure 3.8, we observe that the predictive approach and the approach based on simple RNNs are comparably sensitive to the step size.



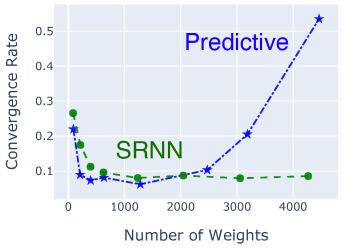
(a) R-2 Accuracy



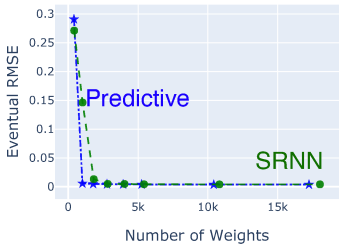
(b) R-2 Speed



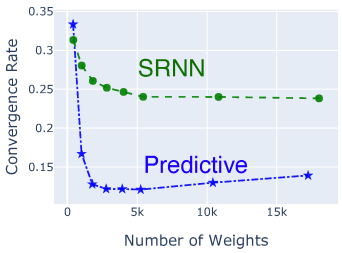
(c) R-6 Accuracy



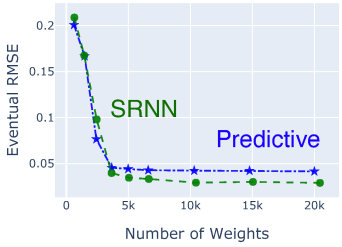
(d) R-6 Speed



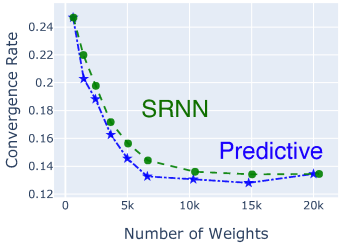
(e) C-5 Accuracy



(f) C-5 Speed

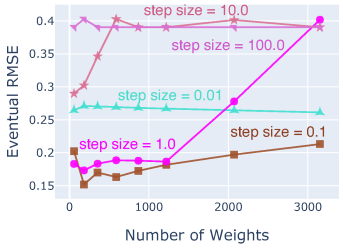


(g) C-9 Accuracy

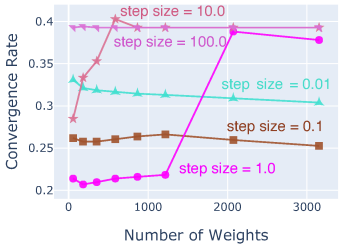


(h) C-9 Speed

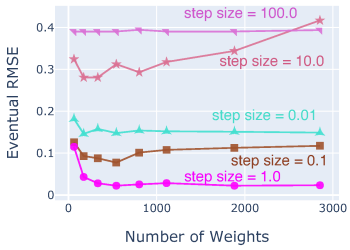
Figure 3.6: Generalization performance of the predictive approach and the approach based on simple RNNs while step size is optimal and capacity of function approximators varies. Generalization performance is roughly measured as the learning speed and accuracy that is the averaged RMSE during the first and last time period, respectively. The x-axis is total number of weights for function approximators with different capacity.



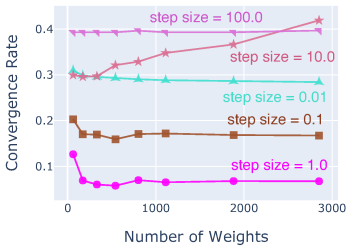
(a) R-2 Predictive Accuracy



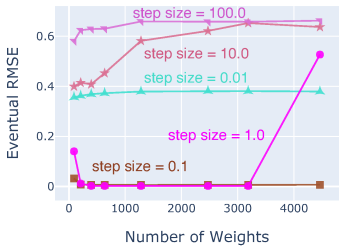
(b) R-2 Predictive Speed



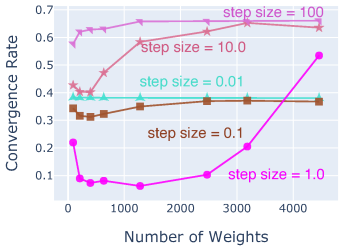
(c) R-2 RNN Accuracy



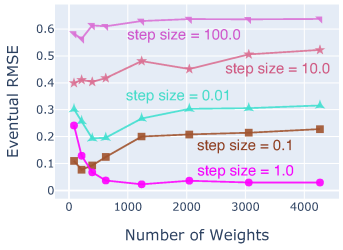
(d) R-2 RNN Speed



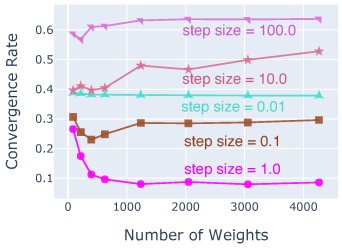
(e) R-6 Predictive Accuracy



(f) R-6 Predictive Speed

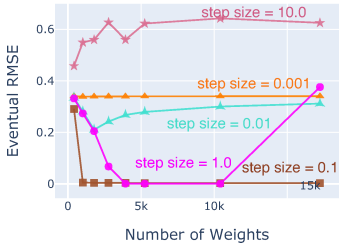


(g) R-6 RNN Accuracy

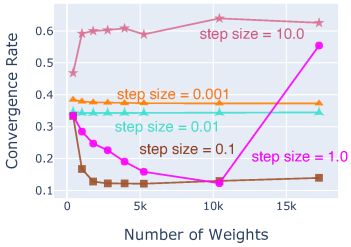


(h) R-6 RNN Speed

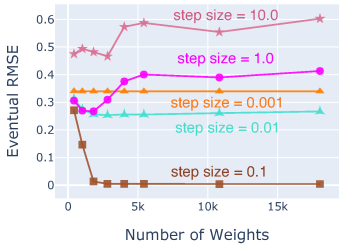
(i) Generalization performance of the predictive approach and the approach based on simple RNNs with different combinations of step size and the capacity of the function approximator. Generalization performance is roughly measured as the learning speed and accuracy that is the averaged RMSE during the first and last time period, respectively. Each line in the plot corresponds to the performance of one specific step size. The The x-axis is total number of weights for function approximators with different capacity.



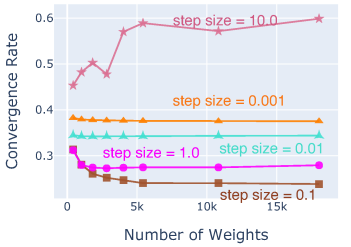
(a) C-5 Predictive Accuracy



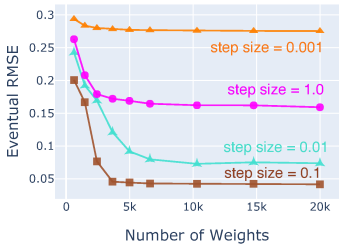
(b) C-5 Predictive Speed



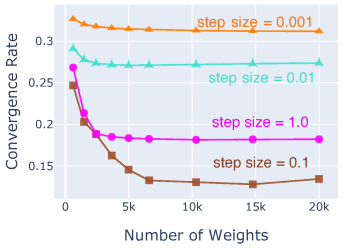
(c) C-5 RNN Accuracy



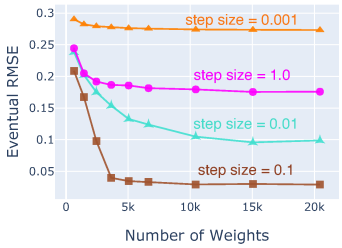
(d) C-5 RNN Speed



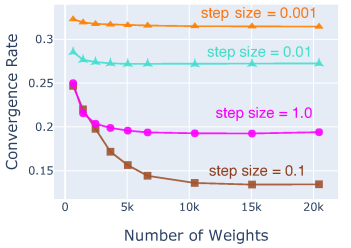
(e) C-9 Predictive Accuracy



(f) C-9 Predictive Speed



(g) C-9 RNN Accuracy



(h) C-9 RNN Speed

Figure 3.8: Generalization performance of the predictive approach and the approach based on simple RNNs with different combinations of step size and the capacity of the function approximator. Generalization performance is roughly measured as the learning speed and accuracy that is the averaged RMSE during the first and last time period, respectively. Each line in the plot corresponds to the performance of one specific step size. The x-axis is total number of weights for function approximators with different capacity.

Chapter 4

Predictive State Update

This chapter introduces *predictive state update* (PSU), a novel approach that utilizes predictions to boost the performance of a broad class of existing state representations that seeks to acquire predictive knowledge. PSU is one main contribution of this thesis. The idea of PSU comes from the results of testing the predictive representations hypothesis as discussed in Chapter 3.

In Section 4.1, we introduce PSU and how to use PSU to construct state representations for making predictions. In Section 4.2, we explain the experiment details when testing the generalization performance of the use of PSU. In Section 4.3, we analyze the generalization performance of using PSU with existing state representations approaches and additionally compare them with the predictive approach. In Section 4.4, we study how the choice of hyperparameters influences the generalization performance of the use of PSU. In Section 4.5, we perform an ablation study of PSU through an empirical study of an alternative approach combining predictions with state representations.

4.1 Predictive State Update

Inspired by the results that predictive representations hypothesis does hold in specific scenarios, we introduce a novel state update rule, *Predictive State Update* (PSU), that incrementally compute the next state from the current

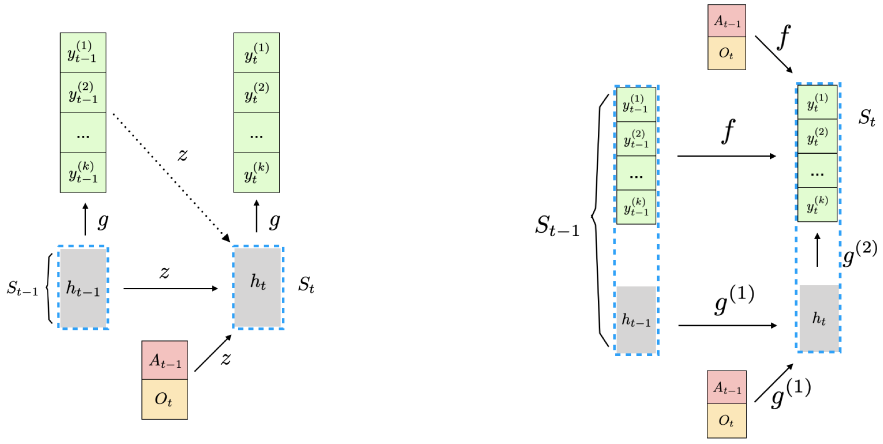
state, while being *aware* of current predictions of interest in addition to the next increment of experiences (the current action and the next observation). Conventionally, past experiences are only interleaved actions and observations, as in $H_t = A_0, O_1, A_1, O_2, ..., A_{t-1}, O_t$. State representations incrementally summarize the past with a state update rule that computes the next state based on the current state, the current action, and the next observation, as in $S_{t+1} = u(S_t, A_t, O_{t+1})$. In contrast, PSU constructs the next state while being aware of the current predictions y_t in addition to the next increment of experiences (A_t and O_{t+1}), as in $S_{t+1} = z(S_t, A_t, O_{t+1}, y_t)$.

Why does one need to be aware of predictions as part of history information to construct state representations? When state representations are used for making predictions of interest, current predictions of interest are computed from current state representations with a mapping function. Therefore, current predictions can be used together with current states to form the next states. On the other hand, when analyzing why predictive representations hypothesis does hold, we observe that for the same environment states, their predictions of interest must be the same, whereas their state representations can be completely different. Explicitly including current predictions to form next states introduces the interesting phenomenon that if predictions of interest are the same then the states are possibly to be the same even though the state representations differ.

We illustrate how to use PSU to construct state representations and make predictions of interest, as shown in Figure 4.1a. We firstly concatenate k predictions of interests y^i_{t-1} as in the predictive approach,

$$p_{t-1} = [y^1_{t-1}, y^2_{t-1}, ..., y^k_{t-1}].$$

We then form the current state $S_t \in \mathbb{R}^n$ by a mapping of previous predictions p_{t-1}, previous state S_{t-1}, previous action A_{t-1}, and current observation O_t



(a) Structure of PSU

(b) Structure of CPSR

Figure 4.1: (a) Illustration of PSU. The state S_t is a mapping of previous predictions y_{t-1}^i, previous state S_{t-1}, previous action A_{t-1}, and current observation O_t with function z. The predictions $y_t^{(i)}$ is a mapping of state S_t with function g. (b) Illustration of CPSR. The state h_t is a mapping of previous state h_{t-1}, previous action A_{t-1}, and current observation O_t with function $g^{(1)}$. The predictions $y_t^{(i)}$ is computed from two parts: one is the mapping of state h_t with function $g^{(2)}$, and another is the mapping of previous predictions y_{t-1}^i, previous action A_{t-1}, and current observation O_t with function f.

with the function z,

$$S_t = z(p_{t-1}, S_{t-1}, A_{t-1}, O_t).$$

The initial state S_0 is set to a vector of zeros. Each predictions y_t^i is a mapping of state S_t with the function g^i,

$$y_t^i = g^i(S_t).$$

Any existing state representation approach can *instantiate* PSU if it summarizes the past incrementally, updating the next state based on the current state and next increment of experiences. Here, instantiation specifically means replacing the update function in PSU (the mapping z in $S_{t+1} = z(S_t, A_t, O_{t+1}, y_t)$) by the update function of the existing state representations approach (the mapping u in $S_{t+1} = u(S_t, A_t, O_{t+1})$), and treating current predictions y_t as part of next increment of experiences (the current action and

the next observation). In the case where we instantiate PSU with the predictive state approach, if we use current predictions of interest to form the predictive state in the next step, then this instantiation of PSU degenerates to the predictive approach. In this thesis, we focus on instantiating PSU with three commonly used state representation approaches that are based on simple recurrent neural networks, LSTMs (Long Short-Term Memory networks), and GRUs (Gated Recurrent Units networks). We name these instantiations as PSU-SRNN, PSU-LSTM, and PSU-GRU. Here, LSTMs and GRUs are variants of RNNs that adopt different gating mechanisms to force the gradient directly flow over time and thus improve simple RNNs' ability to capture temporal information (Hochreiter and Schmidhuber *et al.*, 1997; Cho *et al.*, 2014).

4.2 Experiment Details

We study how the use of PSU affects the generalization performance of approaches based on simple RNNs, LSTMs, and GRUs in the R-2, R-6, C-5, and C-9 scenarios, as described in Section 3.2. We additionally compare these instantiations of PSU with the predictive approach. We sweep over different combinations of step size and function approximator capacity, and we then select the one with the best performance for each algorithm. Specifically, we first compute the area under each learning curve by averaging the RMSE every one thousand time steps. Then we select the hyper-parameter with the lowest area under the learning curve. As shown in Figure 4.1a, the mappings of g is one single-layer fully connected neural networks with a sigmoid activation function, and the mapping z is replaced by the update function used in the existing state representation approach. Note that the dimension of the hidden states in these existing state representations approaches and their instantiations of PSU will influence the capacity of the function approximator. For convenience, we call this the *dimension* of the approximator. Table 4.1

Scenario	Step Size	Dims of Approximators
R-2	(0.01 0.1 1.0 10.0 100.0)	(5 10 15 20 25 30 40 50)
R-6	(0.01 0.1 1.0 10.0 100.0)	(2 5 10 15 20 30 40 50 60)
C-5	(0.01 0.1 1.0 10.0 100.0)	(10 20 30 40 50 60 90 120)
C-9	(0.01 0.1 1.0 10.0 100.0)	(10 20 30 40 50 60 80 100 120)

Table 4.1: A summary of hyper-parameters used for learning predictions with approaches based on simple RNNs, LSTMs, GRUs, and their instantiations of PSU in different scenarios.

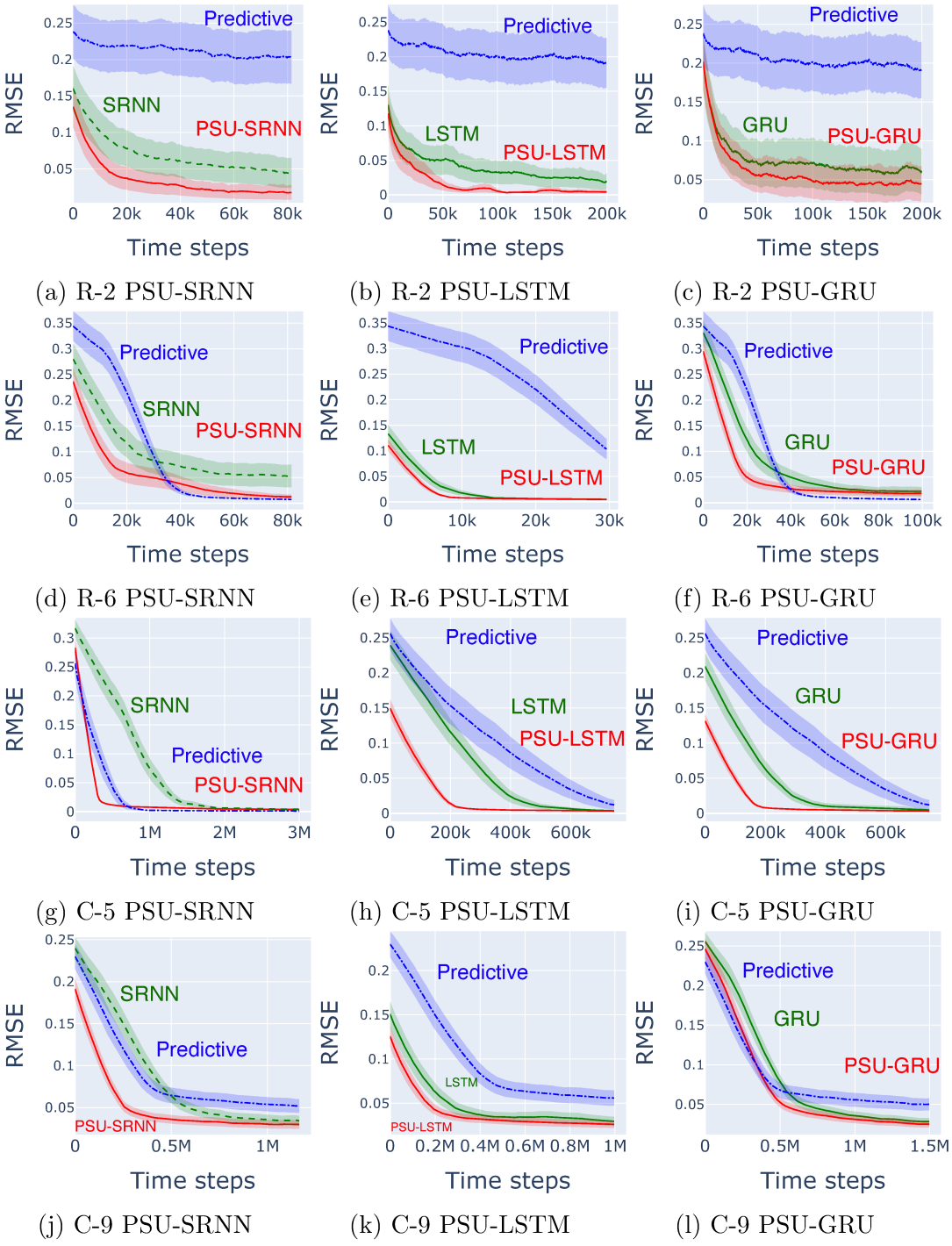
Scenario	PSU-SRNN	LSTM	PSU-LSTM	GRU	PSU-GRU
R-2	(1.0 30)	(1.0 15)	(1.0 30)	(0.1 20)	(0.1 15)
R-6	(1.0 15)	(1.0 30)	(1.0 40)	(0.1 10)	(0.1 15)
C-5	(0.1 60)	(0.1 90)	(0.1 120)	(0.1 120)	(0.1 90)
C-9	(0.1 120)	(0.1 120)	(0.1 120)	(0.01 120)	(0.01 100)

Table 4.2: A summary of the hyper-parameter setting with the best performance for each algorithm in each scenario.

summarizes the combinations of the dimension of the approximator and step size used for learning predictions with approaches based on simple RNNs, LSTMs, GRUs, and their instantiations of PSU in different scenarios. Table 4.2 summarizes the hyper-parameter setting with the best performance for each algorithm. Other experiment details are exactly the same as described in Section 3.4.

4.3 Results

As shown in Figure 4.2, we observe that the use of PSU can boost the generalization performance of approaches based on simple RNNs, LSTMs, and GRUs. Firstly, this result supports the intuition that current predictions of interest can be used as part of the history to form the next states. Secondly, this result implies that being aware of predictions can help the hidden states not be too flexible and thus leads to faster learning, as explicitly including current predictions to form next states introduces the interesting phenomenon that



(a) R-2 PSU-SRNN (b) R-2 PSU-LSTM (c) R-2 PSU-GRU

(d) R-6 PSU-SRNN (e) R-6 PSU-LSTM (f) R-6 PSU-GRU

(g) C-5 PSU-SRNN (h) C-5 PSU-LSTM (i) C-5 PSU-GRU

(j) C-9 PSU-SRNN (k) C-9 PSU-LSTM (l) C-9 PSU-GRU

Figure 4.2: Generalization performance of PSU instantiated by approaches based on simple RNNs, LSTMs, and GRUs in comparison with the predictive approach in the R-2, R-6, C-5, and C-9 scenarios. Shading denotes standard error.

if predictions of interest are the same then the states are possibly to be the same even though the state representations differ. We also observe that PSU-SRNN, PSU-LSTM, and PSU-GRU outperform the predictive approach. This is because the hidden state of these instantiations is capable of finding another state space, and thus they generalize better than the predictive approach.

4.4 Parameter Study

We study how the use of PSU affects the hyper-parameter sensitivity of approaches based on simple RNNs, LSTMs, and GRUs, through visualization. Hyper-parameters are combinations of step size and function approximator capacity. Here the capacity of the function approximator is determined by the dimension of the approximator, which refers to as the dimension of the hidden states in these existing state representations approaches and their instantiations of PSU. To visualize the generalization performance over different hyper-parameter settings, we roughly measure the generalization performance for each setting as the averaged RMSE during the first and last time period, respectively. More specifically, the first tenth and the last tenth of the period are selected for the ring world environment, whereas the first three tenths and the last tenth are chosen for the compass world environment. Since different state representation approaches are using different function approximators, we summarize their performance along with the total number of weights in the approximators. The total number of weights for PSU-SRNN, LSTM, PSU-LSTM, GRU, and PSU-GRU are computed as following,

$$N_{PSU-SRNN} = (h + i + n + 1) * h + (h + 1) * n$$

$$N_{LSTM} = 4 * h * (i + h + 1) + (h + 1) * n$$

$$N_{PSU-LSTM} = 4 * h * (i + h + 1 + n) + (h + 1) * n$$

$$N_{GRU} = 3 * h * (i + h + 1) + (h + 1) * n$$

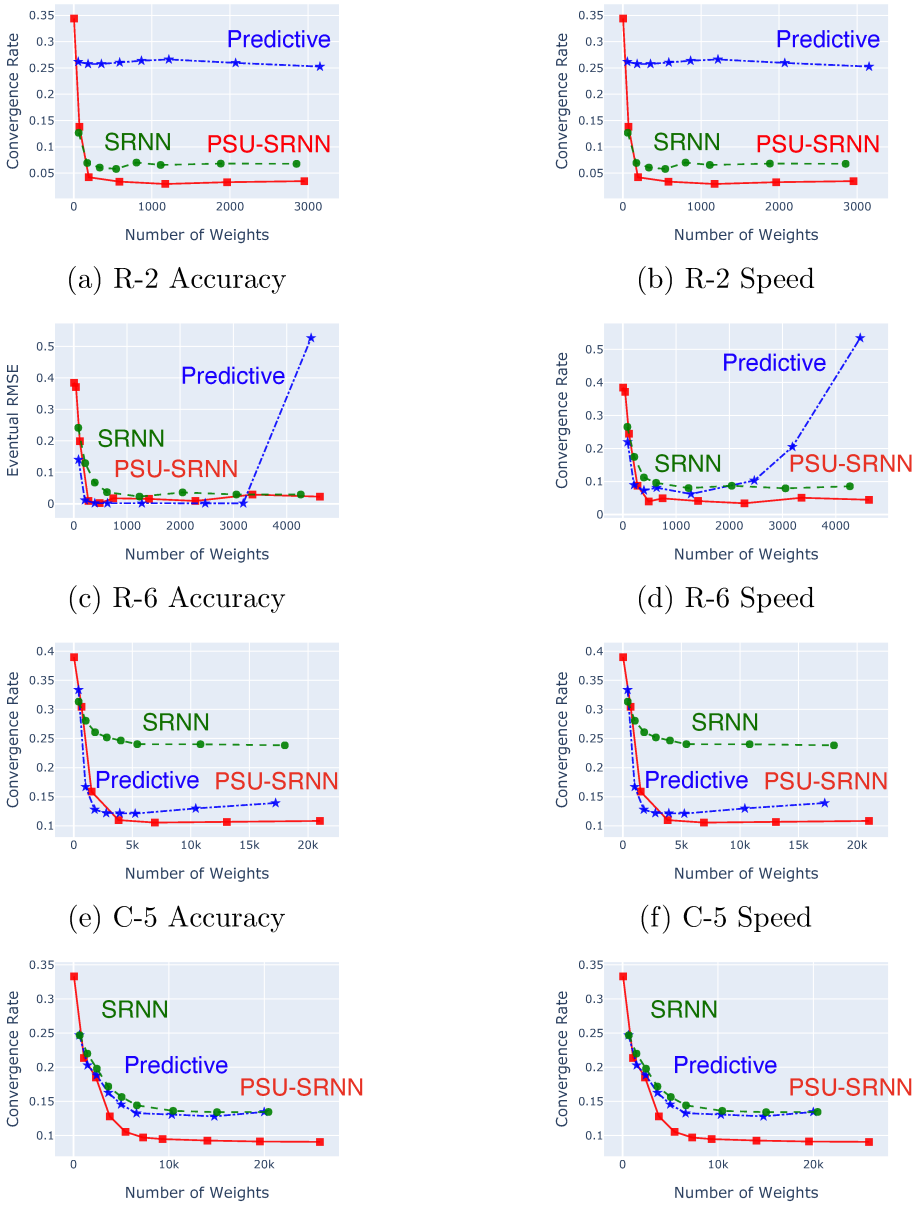
$$N_{PSU-GRU} = 3 * h * (i + h + 1 + n) + (h + 1) * n,$$

where n is number of predictions of interest, h is the dimension of the approximator, and i is the dimension of the inputs, which includes action and observation at each time step. Note that the number of weights for approaches based on simple RNNs, LSTMs, and GRUs is $\mathcal{O}(h*(i+h+n))$, and the number of weights only increases $\mathcal{O}(h*n)$ when instantiating PSU with these state representations approaches.

First, as shown in Figure 4.3, 4.4, and 4.5, the use of PSU can boost the generalization performance of approaches based on simple RNNs, LSTMs, and GRUs with comparable number of weights in function approximators. When comparing PSU-SRNN and the predictive approach in Figure 4.3, we observe that PSU-SRNN achieves decent generalization performance with fewer number of weights in function approximators and is more robust when this number of weights grows. Secondly, as shown in Figure 4.6, Figure 3.7i and Figure 3.8, we observe that the use of PSU does not affect the simple RNNs' sensitivity to the step size. This conclusion is also true for LSTM and GRU. For brevity, we omit the plots studying these algorithms' sensitivity to the step size.

4.5 Ablation Study

One naive solution to combining predictions and state representations is to concatenate predictive states and hidden states in simple RNNs, and compute predictions based on this concatenation. We name this method Concatenation of the Predictive approach and Simple RNNs (CPSR). In this section, we empirically demonstrate CPSR is not an effective solution towards combining predictions and state representations.



(a) R-2 Accuracy

(b) R-2 Speed

(c) R-6 Accuracy

(d) R-6 Speed

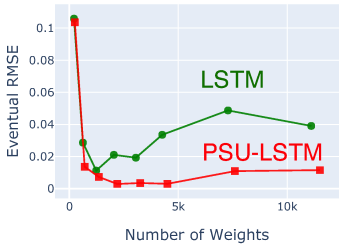
(e) C-5 Accuracy

(f) C-5 Speed

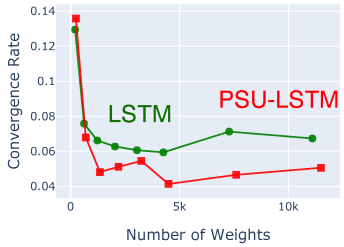
(g) C-9 Accuracy

(h) C-9 Speed

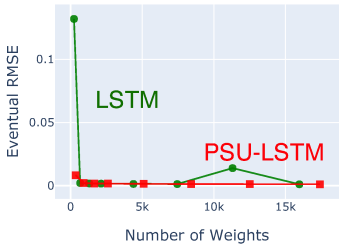
Figure 4.3: Generalization performance of PSU-SRNN, simple RNNs, and the predictive approach while step size is optimal and capacity of function approximators vary. Generalization performance is roughly measured as the learning speed and accuracy with respect to the averaged RMSE during the first and last time period. The x-axis is total number of weights for function approximators with different capacity.



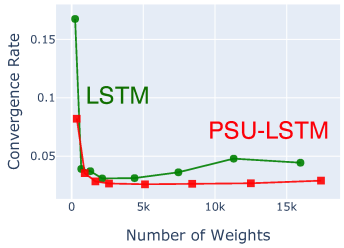
(a) R-2 LSTM Accuracy



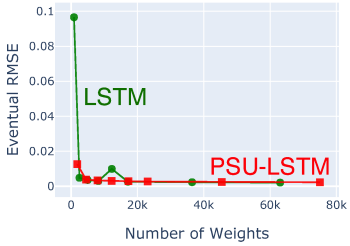
(b) R-2 LSTM Speed



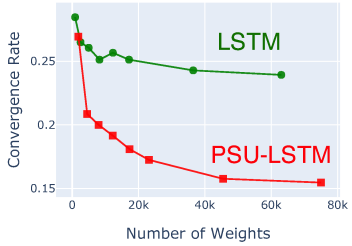
(c) R-6 LSTM Accuracy



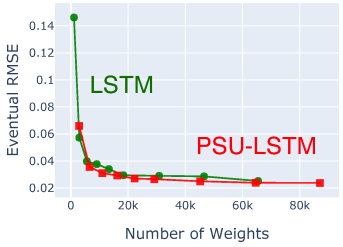
(d) R-6 LSTM Speed



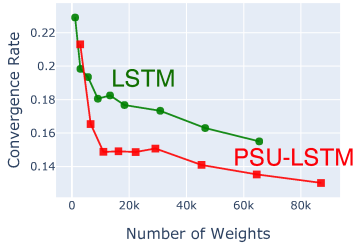
(e) C-5 LSTM Accuracy



(f) C-5 LSTM Speed

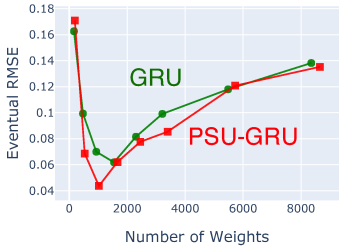


(g) C-9 LSTM Accuracy

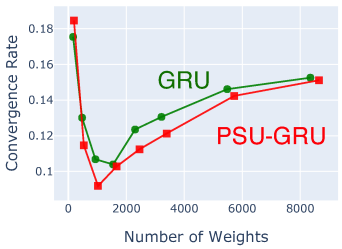


(h) C-9 LSTM Speed

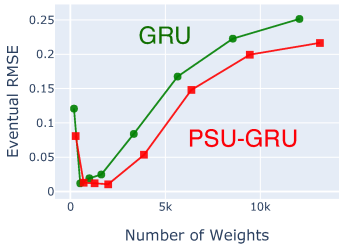
Figure 4.4: Generalization performance of LSTM and PSU-LSTM while step size is optimal and capacity of function approximators vary. Generalization performance is roughly measured as the learning speed and accuracy with respect to the averaged RMSE during the first and last time period. The x-axis is total number of weights for function approximators with different capacity.



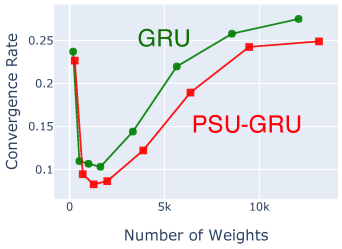
(a) R-2 GRU Accuracy



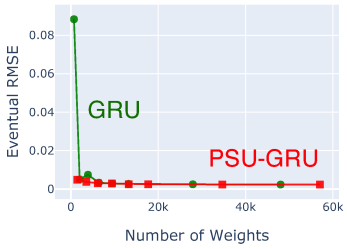
(b) R-2 GRU Speed



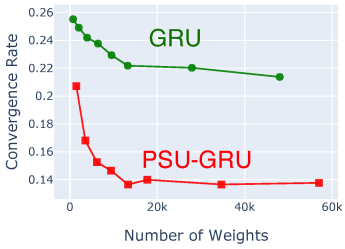
(c) R-6 GRU Accuracy



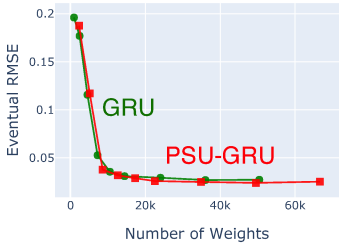
(d) R-6 GRU Speed



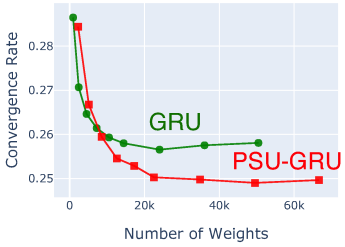
(e) C-5 GRU Accuracy



(f) C-5 GRU Speed

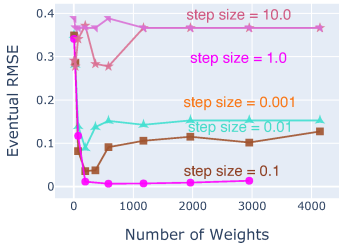


(g) C-9 GRU Accuracy

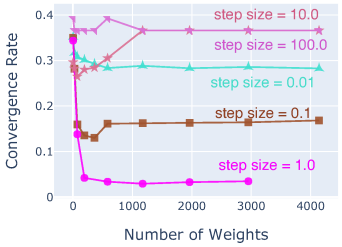


(h) C-9 GRU Speed

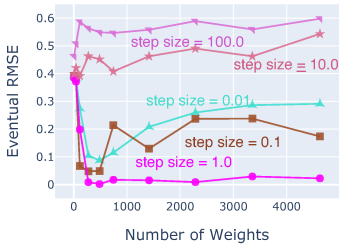
Figure 4.5: Generalization performance of GRU and PSU-GRU while step size is optimal and capacity of function approximators vary. Generalization performance is roughly measured as the learning speed and accuracy with respect to the averaged RMSE during the first and last time period. The x-axis is total number of weights for function approximators with different capacity.



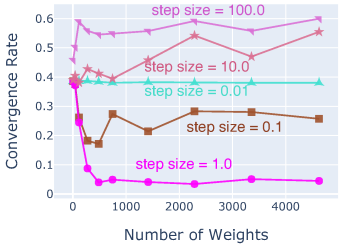
(a) R-2 PSU Accuracy



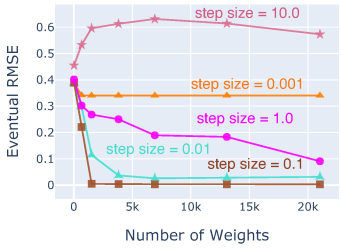
(b) R-2 PSU Speed



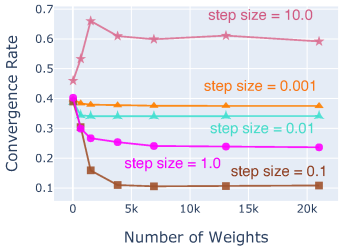
(c) R-6 PSU Accuracy



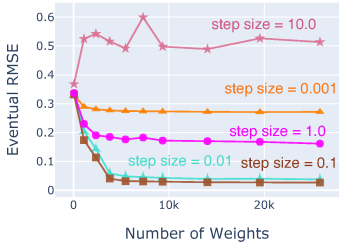
(d) R-6 PSU Speed



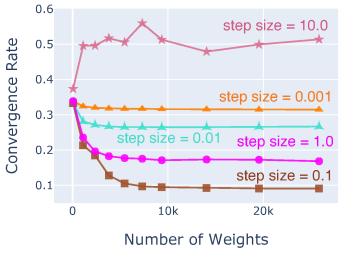
(e) C-5 PSU Accuracy



(f) C-5 PSU Speed



(g) C-9 PSU Accuracy



(h) C-9 PSU Speed

Figure 4.6: Generalization performance of PSU-SRNN with different combination of step size and capacity of function approximators. Generalization performance is roughly measured as the learning speed and accuracy with respect to the averaged RMSE during the first and last time period. Each line in the plot corresponds to the performance of one specific step size. The x-axis is total number of weights for function approximators with different capacity.

As shown in Figure 4.1b, we firstly concatenate k predictions of interests y^i_{t-1} as in the predictive approach,

$$p_{t-1} = [y^1_{t-1}, y^2_{t-1}, ..., y^k_{t-1}].$$

We then form the current hidden state $h_t \in \mathbb{R}^n$ by a mapping of previous hidden state h_{t-1}, previous action A_{t-1}, and current observation O_t with function $g_{(1)}$,

$$h_t = g_{(1)}(h_{t-1}, A_{t-1}, O_t)$$

The initial state S_0 is set to a vector of zeros. Each predictions y^i_t is computed from two parts: one is the mapping of hidden state h_t with function $g^i_{(2)}$, and another is the mapping of previous predictions y_{t-1}, previous action A_{t-1}, and current observation O_t with function f^i,

$$y^i_t = \sigma(f^i(p_{t-1}, A_{t-1}, O_t) + g^i_{(2)}(h_t)),$$

where σ is an activation function such as the sigmoid function.

We study CPSR's generalization performance in the R-2, R-6, C-5, and C-9 scenarios while comparing with the approach based on simple RNNs and the predictive approach. We sweep over different combinations of step size and function approximator capacity. As shown in Figure 2.2a, the mapping f is a two-layer fully connected neural networks with a sigmoid activation function, and the mapping $g^{(1)}$, and $g^{(2)}$ are two separate single-layer fully connected neural networks without an activation function. The dimension of hidden layers in the mapping f can be understood as a bottleneck of predictive state, which controls the amount of predictive information that can influence the next predictions. We call this the dimension of the predictive states. Analogously, the dimension of hidden states in simple RNNs controls their impact on the next predictions. It is obvious that if the dimension of one of these two states equals zero, then CPSR degenerates to another state approach.

Scenario	Step Size	Dims of Predictive	Dims of simple RNNs
R-2	(0.001 0.01 0.1 1.0 10.0)	(0 4 16 32 64)	(0 5 20 30 40 50)
R-6	(0.001 0.01 0.1 1.0 10.0)	(0 5 10 40 75 128)	(0 4 16 32 50)
C-5	(0.001 0.01 0.1 1.0 10.0)	(0 7 32 95 189)	(0 32 60 90 120)
C-9	(0.001 0.01 0.1 1.0 10.0)	(0 4 8 32 64 128)	(0 20 40 80 120)

Table 4.3: A summary of hyper-parameters swept over when learning predictions with CPSR.

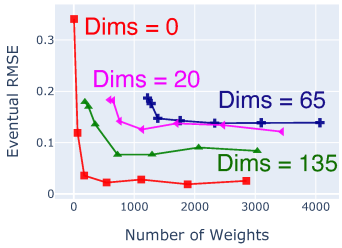
We summarize the combinations of the dimension of these two states and step size in different scenarios in Table 4.3. Other experiment details are exactly the same as described in Section 3.4.

To visualize the performance over a large amount of hyper-parameter settings, we roughly measure the generalization performance for each setting as the averaged RMSE during the initial and last period of time, respectively. More specifically, the first tenth and the last tenth of the period are selected for the ring world environment, whereas the first three tenths and the last tenth are chosen for the compass world environment. Since different state representation approaches are using different function approximators, we summarize their performance along with the total number of weights of the approximators. The total number of weights for CPSR is computed as

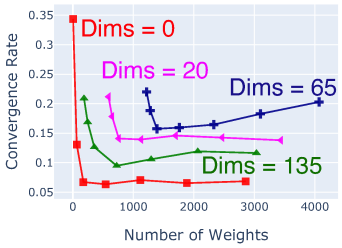
$$N_{CPSR} = (h + i + 1) * h + (n + i + 1) * n + (h + 1) * n$$

where n is number of predictions of interest, h is the dimension of approximators, and i is the dimension of the inputs, which includes action and observation at each time step.

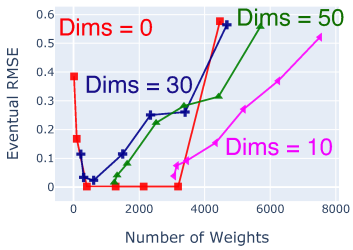
We demonstrate CPSR's generalization performance when the capacity of function approximators varies, but the step size is optimal in Figure 4.7. In the R-2 and C-9 scenarios, the best performance of CPSR is the same as the simple RNNs. In the R-6 and C-5 scenarios, the best performance of CPSR is the same as the predictive state approach. Thus, we can conclude that, the



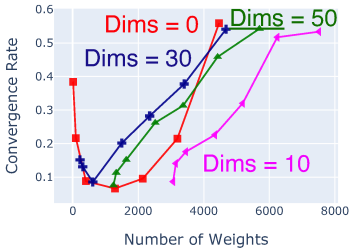
(a) R-2 Accuracy by Dims of Predictive



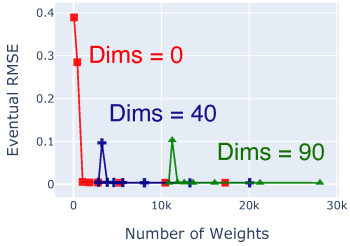
(b) R-2 Speed by Dims of Predictive



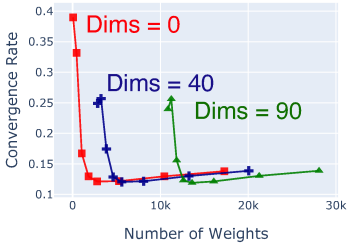
(c) R-6 Accuracy by Dims of SRNNs



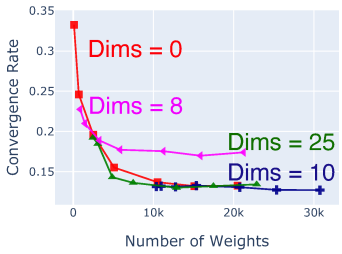
(d) R-6 Speed by Dims of SRNNs



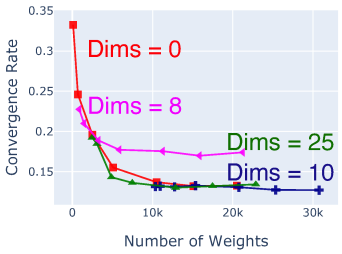
(e) C-5 Accuracy by Dims of SRNNs



(f) C-5 Speed by Dims of SRNNs



(g) C-9 Accuracy by Dims of Predictive



(h) C-9 Speed by Dims of Predictive

Figure 4.7: Generalization performance of CPSR while step size is optimal and capacity of function approximators vary. Generalization performance is roughly measured as the learning speed and accuracy with respect to the averaged RMSE during the first and last time period. The x-axis is the total number of weights of each function approximator.

naive concatenation of predictive states and hidden states in simple RNNs is not an effective way to combine these two approaches.

Chapter 5

Related Works

This chapter presents existing works related to our exploration of predictive representations of state in this thesis. In Section 5.1, we introduce literature using predictions as knowledge. In Section 5.2, we describe several attempts to learn and discover a set of predictions to form the state. In Section 5.3, we compare our interpretation of predictive representations hypothesis with those in existing works. In Section 5.4, we briefly summarize the principal works using recurrent neural networks (RNNs) to represent and learn predictions. In Section 5.5, we present a survey of works taking advantage of both RNN based and the predictive state representation approaches.

5.1 Predictions as Knowledge

Humans accumulate knowledge about the world via interactions with the environment. Predictive knowledge about the environment is represented by the expected outcome (predictions) of possible interactions. Cunningham (1972), and Drescher (1991) propose to ground knowledge in experiences, which is one major characteristic of predictive knowledge. Tanner and Sutton (2005) propose temporal difference networks for representing and learning interrelated predictions. These predictions are also used as state representations based on the intuition that predictions can capture knowledge about the environment.

Sutton *et al.* (2011) formalize knowledge about the environment as a large number of general value functions (GVFs), each of which captures a portion of world knowledge through its own policy, reward function, and termination function. GVFs can be informally understood as a general form of predictions.

Predictive knowledge frameworks have demonstrated their advantages in several real-life applications. In the domain of robotics, thousands of predictions about sensory inputs at multiple timescales can be learned in an effective and online fashion (Modayil *et al.*, 2014). White (2015) adopt advanced off-policy method gradient-TD (Sutton *et al.*, 2009) to learn predictions on a robot in parallel. Edwards *et al.* (2016) show that GVFs can be used to improve the control method in the domain of powered prosthetic arms.

5.2 Predictive Representations of State (PRS)

Predictive representations of state (PRS) form the state as a vector of predictions about the future - the occurrence of a set of action–observation sequences, termed as tests (Littman and Sutton, 2002). A set of tests that are sufficient to make any prediction about the future is named as *core tests*. If core tests are given beforehand, a *learning* algorithm is needed to estimate the core tests through interactions with the environment. Otherwise, the problem of finding core tests is referred to as *discovering* PRS.

Several attempts have been made to learn and discover core tests in PRS. Singh *et al.* (2003) present the first algorithm to estimate the core tests by updating one-step extensions of all the core tests and null test. Singh *et al.* (2004) propose the first procedure for discovering core tests in PRS. They introduce a system-dynamics matrix whose element indicate the prediction for a test given a history. The system-dynamics matrix can be estimated through sampling from the environment. The process for discovering core tests is to gradually increase the length of the test until the rank of the system-dynamics

matrix does not grow. Instead of measuring the predictions by the Monte Carlo method, McCracken and Bowling (2017a) present an algorithm to estimate the predictions by gradient descent. This algorithm can be used to learn and discover PRS in an online fashion. Bowling *et al.* (2006) introduce a method for developing a good explorative policy when estimating predictions by the Monte Carlo approach.

Spectral learning has recently been exploited for discovering core tests of PRS in high dimensional environments. Rosencrantz *et al.* (2004) adopt a principal-components-based algorithm to learn *transformed* PRS (a variant of PRS) from a large set of tests, which contains sufficient tests to form the state. Boots *et al.* (2011) use a spectral algorithm to learn transformed PRS in a high-dimensional robot environment and perform point-based planning in the learned model. Kulesza *et al.* (2015) present an algorithm to search for tests when a large set of tests with sufficient statistics is not available.

Value functions can be used as a generalized form of tests. Tanner and Sutton (2005) and Rafols *et al.* (2006) extend TD methods to learn interrelated value functions and use these predictions as state representations in a partially observable environment. Makino and Takagi (2008) propose an algorithm for discovering interrelated predictions in TD networks (Tanner and Sutton, 2005) in an online fashion. Sutton *et al.* (2011) formalize knowledge about the environment as a large number of general value functions (GVFs), each of which captures a portion of world knowledge through its own policy, reward function, and termination function. Schlegel *et al.* (2018) reformulate the interrelated value functions (Tanner and Sutton, 2005; Rafols *et al.*, 2006) in the language of GVFs.

In this work, we form the predictive state via a vector of interrelated GVFs, and focus on the problem of learning multiple interrelated GVFs of interest.

5.3 Predictive Representations Hypothesis

The predictive representations hypothesis suggests that representing the state of the world in terms of predictions about the future will result in good generalization.

In this thesis, *representing the state of the world* means explicitly using interrelated GVFs to form the state (Rafols *et al.*, 2005; Tanner and Sutton, 2005; Rafols *et al.*, 2006; Schaul *et al.*, 2013; Schlegel *et al.*, 2018). By contrast, there are several attempts that implicitly utilize predictions to learn state representations. Venkatraman *et al.* (2017) learn state representations through gradient descent with an objective of predicting the distribution of future observations. Jaderberg *et al.* (2016) learn a robust state representation through gradient descent while minimizing multiple predictions about the environment. Sun *et al.* (2016) represent the state using a filtered space of distribution of future observations.

Good generalization is usually quantified by good learning performance in both accuracy and speed when an agent achieves a goal. One possible goal for an agent is to maximum accumulative rewards while interacting with the environment (Rafols *et al.*, 2005; Schaul *et al.*, 2013; Jaderberg *et al.*, 2016). Another goal is to answer several predefined questions about future experiences in the form of GVFs (Tanner and Sutton, 2005; Rafols *et al.*, 2006; Schlegel *et al.*, 2018). Some works focus on one specific type of GVFs, which predict k observations in the future (Sun *et al.*, 2016; Venkatraman *et al.*, 2017). In this thesis, good generalization is specifically measured by good learning performance in accuracy and speed when learning interrelated GVFs of interest.

5.4 Recurrent Neural Networks (RNNs)

Instead of explicitly forming states from observations, the hidden states in recurrent neural networks (RNNs) can be used as state representations. The hidden states in RNNs summarize the history through gradient descent while minimizing prediction errors. There is a long history of using RNNs for sequence modeling. Jordan (1997) proposes a recurrent connection from predictions back to the hidden state, which is thus able to keep a dynamic memory. Alternatively, Elman (1990) feeds the hidden state back to itself in a recurrent way, which is the precursor to the development of recent RNNs. It is challenging for the hidden states in RNNs to capture information over extended time intervals through gradient descent. Different gating mechanisms, which force the gradient to directly flow over time, are commonly adopted to alleviate this problem (Hochreiter and Schmidhuber *et al.*, 1997; Cho *et al.*, 2014).

The use of RNNs achieves state-of-art results in sequential modeling for natural language processing (Bengio *et al.*, 2003), speech recognition (Graves and Jaitly, 2014) and handwriting recognition (Graves *et al.*, 2008). In reinforcement learning, RNNs are widely used especially in tasks where environment states are not accessible to the agent, such as robotic controlling (Duan *et al.*, 2016), and Atari games(Hausknecht and Stone, 2015).

In this work, we use the hidden states in RNNs to learn interrelated GVFs in an online fashion, whereas existing works usually require trajectories of history information.

5.5 PRS and RNNs

Several attempts have been made to draw insights from both PRS and RNNs based on the intuition that spectral algorithms for learning PRS pose appealing theoretical support, whereas RNNs have powerful expression but lack

probabilistic interpretation. In this line of works, the state representations are used for predicting the probability of future occurrences, and learning PRS requires access to trajectories of history information. Downey *et al.* (2017b) introduce a spectral initialization when learning PRS through predictive state inference machines (Sun *et al.*, 2016). This method is equivalent to a special case of the back-propagation through time algorithm in RNNs. Downey *et al.* (2017a) present a new architecture of RNNs that use a bilinear transfer function derived from the idea of state updates in PRS.

Predictions can be used as regularizers when learning the hidden states in RNNs. Venkatraman *et al.* (2017) add supervision to the hidden states in RNNs by encoding statistics of future observations into the hidden state. Jaderberg *et al.* (2016) use value functions for auxiliary tasks as regularizers to obtain robust state representations. Specifically, they learn the hidden states in RNNs while maximizing the rewards for both the main task and auxiliary tasks.

In this work, we use predictions learned by TD methods to boost the generalization performance of a broad class of state representation approaches (including but potentially not limited to RNN based approaches) without access to trajectories of history information.

Chapter 6

Conclusion

The *Predictive state representations* approach links predictive knowledge to state representations by directly representing the state of the environment in terms of predictions about the future. As our first contribution in this thesis, we empirically demonstrate that explicitly using predictions to represent the state does result in good generalization performance in specific scenarios. This inspires us to explore alternative approaches for relating predictions to state representations. Our second contribution is a novel state update rule, predictive state update (PSU), that computes the next state while being aware of current predictions in addition to current state and next increment of experiences. Our experiment shows that (i) the use of PSU can boost the generalization performance of existing state representation approaches, such as those based on simple recurrent neural networks, LSTM (Long Short-Term Memory) networks, and GRU (Gated Recurrent Units) networks, and (ii) these instantiations of PSU outperform approaches which represent states exclusively using predictions. Our work provides strong evidence supporting that utilizing predictions to form state representations will result in good generalization.

Bibliography

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157-166.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137-1155.

Boots, B., Siddiqi, S. M., and Gordon, G. J. (2011). Closing the learning-planning loop with predictive state representations. *The International Journal of Robotics Research*, 30(7):954-966.

Bowling, M., McCracken, P., James, M., Neufeld, J., and Wilkinson, D. (2006). Learning predictive state representations using non-blind policies. In *Proceedings of the 23rd International Conference on Machine learning*, pages 129-136.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724-1734.

Cunningham, M. (1972). *Intelligence: Its Organization and Development.* Academic Press.

Downey, C., Hefny, A., Boots, B., Gordon, G. J., and Li, B. (2017a). Predictive state recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 6053-6064.

Downey, C., Hefny, A., and Gordon, G. (2017b). Practical learning of predictive state representations. arXiv:1702.04121.

Drescher, G. L. (1991). *Made-up Minds: a Constructivist Approach to Artificial Intelligence.* MIT press.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the International Conference on Machine Learning*, pages 1329-1338.

Edwards, A. L., Hebert, J. S., and Pilarski, P. M. (2016). Machine learning and unlearning to autonomously switch between the functions of a myoelectric arm. In *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 514-521.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

Graves, A., and Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 1764-1772.

Graves, A., Liwicki, M., Bunke, H., Schmidhuber, J., and Fernández, S. (2008). Unconstrained on-line handwriting recognition with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 577-584.

Hausknecht, M., and Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*.

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735-1780.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. (2016). Reinforcement learning with unsupervised auxiliary tasks. arXiv:1611.05397.

Jordan, M. I. (1997). Serial order: A parallel distributed processing approach. In *Advances in Psychology*, volume 121, pages 471-495.

Kulesza, A., Jiang, N., and Singh, S. (2015). Spectral learning of predictive state representations with insufficient statistics. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Littman, M. L., and Sutton, R. S. (2002). Predictive representations of state. In *Advances in Neural Information Processing Systems*, pages 1555-1561.

Maaten, L. V. D., and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579-2605.

Makino, T., and Takagi, T. (2008). On-line discovery of temporal-difference networks. In *Proceedings of the 25th International Conference on Machine Learning*, pages 632-639.

McCracken, P., and Bowling, M. (2006). Online discovery and learning of predictive state representations. In *Advances in Neural Information Processing Systems*, pages 875-882.

Modayil, J., White, A., and Sutton, R. S. (2014). Multi-timescale nexting in a reinforcement learning robot. *Adaptive Behavior*, 22(2):146-160.

Rafols, E., Koop, A., and Sutton, R. S. (2006). Temporal abstraction in temporal-difference networks. In *Advances in Neural Information Processing Systems*, pages 1313-1320.

Rafols, E. J. (2006). *Temporal Abstraction in Temporal-diference Networks*. Ph.D. thesis, University of Alberta.

Rafols, E. J., Ring, M. B., Sutton, R. S., and Tanner, B. (2005). Using predictive representations to improve generalization in reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 835–840.

Rosencrantz, M., Gordon, G., and Thrun, S. (2004). Learning low dimensional predictive representations. In *Proceedings of the Twenty-First International Conference on Machine learning*, page 88.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report. California University San Diego, La Jolla Institute for Cognitive Science.

Schaul, T., and Ring, M. (2013). Better generalization with forecasts. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*.

Schlegel, M., White, A., Patterson, A., and White, M. (2018). General value function networks. arXiv:1807.06763.

Singh, S., James, M. R., and Rudary, M. R. (2004). Predictive state representations: A new theory for modeling dynamical systems. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 512–519.

Singh, S. P., Littman, M. L., Jong, N. K., Pardoe, D., and Stone, P. (2003).

Learning predictive state representations. In *Proceedings of the 20th International Conference on Machine Learning*, pages 712–719.

Sun, W., Venkatraman, A., Boots, B., and Bagnell, J. A. (2016). Learning to filter with predictive state inference machines. In *Proceedings of the International Conference on Machine Learning*, pages 1197–1205.

Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT press.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. (2009). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 993–1000.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, volume 2, pages 761–768.

Tanner, B., and Sutton, R. S. (2005). Temporal-difference networks with history. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 865–870.

Venkatraman, A., Rhinehart, N., Sun, W., Pinto, L., Hebert, M., Boots, B., Kitani, K., and Bagnell, J. (2017). Predictive-state decoders: Encoding the

future into recurrent networks. In *Advances in Neural Information Processing Systems*, pages 1172–1183.

White, A. (2015). *Developing a Predictive Approach to Knowledge*. Ph.D. thesis, University of Alberta.

Williams, R. J., and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280.