ABSTRACT
(Signal Processing)

PLANNING AND LEARNING IN PARTIALLY OBSERVABLE STOCHASTIC ENVIRONMENTS

by

Hui Li

Department of Electrical and Computer Engineering
Duke University

Date: ____________________________
Approved:

______________________________
Lawrence Carin, Supervisor

______________________________
Ronald Parr

______________________________
David Banks

______________________________
Jeffrey Krolik

______________________________
Rebecca Willett

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University

2006
PLANNING AND LEARNING IN PARTIALLY OBSERVABLE STOCHASTIC ENVIRONMENTS

by

Hui Li

Department of Electrical and Computer Engineering
Duke University

Date: __________________________

Approved:

______________________________
Lawrence Carin, Supervisor

______________________________
Ronald Parr

______________________________
David Banks

______________________________
Jeffrey Krolik

______________________________
Rebecca Willett

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University

2006
Abstract

In this thesis the first problem we address and solve is sequential decision-making for a single environment, based on partially observable Markov decision processes (POMDPs), assuming the POMDP models are given. We develop three new algorithms for finding the optimal policy for POMDPs.

The first is region-based value iteration (RBVI). The RBVI approximates the true polyhedral partition of the belief simplex in the POMDP model with an ellipsoidal partition, such that the optimal value function is linear in each ellipsoidal region. The position and shape of each region, as well as the gradient (alpha-vector) of the optimal value function in the region, are parameterized explicitly, and are estimated via efficient expectation maximization (EM) and variational Bayesian EM (VBEM), based on a set of selected sample belief points.

The other algorithm is incremental least-squared policy iteration (ILSPI). The ILSPI computes a basis representation of the infinite-horizon value function by minimizing the squared Bellman residual, and performs policy improvement at the sample belief points. A number of optimal basis functions are determined by the algorithm to minimize the Bellman residual incrementally, via efficient computations. We show that, by using optimally determined basis functions, the policy can be improved successively on a set of most probable belief points sampled from the reachable belief set.

The third algorithm is piecewise linear policy iteration (PLPI). Unlike the ILSPI, the PLPI represents the value function by a finite set of hyper-planes, which are estimated in the policy-evaluation step by an iterative application of the Bellman equation. The PLPI performs point-wise policy improvement at belief samples, like the ILSPI.
The three algorithms, though novel, still fall into the classical POMDP problem of finding the optimal policy given the POMDP model. In many applications, however, it is very difficult to obtain the POMDP model. What the agent has is the history of observations and actions obtained by interacting with the environment. The second problem we address in this thesis is to find a good policy given the history of observations and actions, without requiring knowing the model of the environment.

We propose a new parametric framework, called *regionalized policy representation* (RPR), for reinforcement learning in partially observable stochastic environments. The RPR yields a stochastic policy in the form of a distribution of action given the history of actions and observations. Learning of RPRs is based on the empirical value function, obtained through Monte-Carlo integration, using the experiences with the environments. Two learning algorithms are presented, namely, maximum-value (MV) estimation and variational Bayesian (VB) learning.

The stochastic policy representation by RRRs makes it possible to transfer similar experience from one environment when learning the policy in another. By learning simultaneously across multiple environments, the experience in each individual environment is augmented through the borrowing of experience from relevant environments. In the thesis, we develop a *multi-task-learning RPR framework* (RPR-MTL), where multiple RPRs are unified by a common Dirichlet process from which the prior distribution of each RPR is drawn. The RPR-MTL automatically identifies relevant environments and clusters experience from them to learn a RPR shared by the relevant environments. We develop the Gibbs sampling algorithm as well as an Gibbs-variational algorithm for learning the RPR-MTL. We also show the Gibbs-variational algorithm can be implemented by distributed computation.

The final contribution of the thesis is to apply the techniques of POMDPs to the problem of cost-sensitive sequential selection of sensors or features.
Keywords: sequential decision-making, agent, stochastic environment, partially observable Markov decision process (POMDP), horizon, expected future reward, value function, piecewise linear convexity, value iteration, policy iteration, Bellman residual, least-squares, incremental basis selection, reinforcement learning, episodes, Dirichlet process, multi-task learning (MTL), expectation-maximization (EM), variational Bayesian EM (VBEM)
Acknowledgements

I would like to express my gratitude to all who have helped me in the past five years to make this long journey pass smoothly.

First of all, I would like to thank my advisor, Dr. Lawrence Carin, for his guidance, inspiration, encouragement, and support given to this work. Without his extensive and patient support, I could not achieve what I have achieved during these five years. Not only the knowledge he imparted to me, but also the spirit of keeping working hard and never waste a second of time which encouraged me during the past five years and will continue affecting and impressing me throughout my whole career life.

I am grateful to Dr. Ronald Parr who helped me to broaden my research vision with his insightful thoughts and brilliant mind, and to Dr. David Banks who wrote me a reference letter and gave me many useful career suggestions. I am also grateful to Dr. Jeffrey Krolik and Dr. Rebecca Willett for serving on my prelim and final defense committee.

I would like to thank my colleague and friend Qiuhua Liu. We came to Duke in the same year and have since then worked as teammates in many course projects as well as research projects. Her affable nature, good attitude towards life, and smiling face have often lightened the steps in the five-year journey and made the life at Duke happier. I would also like to thank Dr. Yanting Dong, Dr. Nilanjan Dasgupta, Dr. Balaji Krishnapuram, Dr. Shaorong Chang, Dr. Dehong Liu, Dr. Shihao Ji, Dr. David Williams, Dr. Esther Dura, Dr. Jun Fang, Yuting Qi, Lihan He, Ya Xue, and Tong Zhou and many others for their generous help.

Special thanks go to my husband Dr. Xuejun Liao. He has been so patient and supportive and encouraging since I met him, and has given me the confidence to challenge difficulties in life. He has always been there with me to help me and
share the stress during the pursuit of my PhD degree. His true and ceaseless love for research and pursuit for perfection often shape me and change me.

Finally, I would like to express my deepest appreciation to my parents and my brothers, whose unselfish love and unconditional support finally have made me reach my short-term goal and achieve what I have achieved.
Contents

[Abstract] iv

[Acknowledgements] vii

[List of Tables] xiv

[List of Figures] xvi

1 Introduction 1

1.1 Sequential Decision Making 1

1.2 Markov Decision Processes (MDPs) 3

1.2.1 Formal definition of the MDP 5

1.2.2 Solutions to MDPs 6

1.3 Partially Observable Markov Decision Processes (POMDPs) 7

1.3.1 Formal definition of the POMDP 9

1.3.2 Belief states 10

1.3.3 Solutions to POMDPs 13

1.4 Model-free Reinforcement Learning 14

1.5 Structure of the Thesis 15

2 Region-Based Value Iteration for Partially Observable Markov Decision Processes 18

2.1 Introduction to Value Iteration for POMDPs 18

2.2 Overview of Value Iteration Algorithms for POMDPs 20

2.3 Region-based Value Iteration (RBVI) Algorithm for POMDP 21

2.3.1 Ellipsoidal approximation of the polyhedral piecewise linearity 22

2.3.2 Parameter estimation with expectation-maximization 24
3.7 Appendix: Analysis and Proof

3.7.1 Differences between two least-squares approximations .......................... 73
3.7.2 Proof of Theorem 4 .................................................................................. 75

4 Regionalized Policy Representation for Reinforcement Learning in POMDPs

4.1 Introduction ................................................................................................. 77
4.2 Mathematical Formulation of the RPR ........................................................ 78
4.3 Empirical Learning of the RPR ................................................................... 82
4.4 Maximum-Value Estimation of the RPR ....................................................... 85
  4.4.1 Calculation of $\{q_k^k(z_{0:t})\}$ ................................................................. 86
  4.4.2 Maximization of $LB(\hat{\Theta}|\Theta)$ .......................................................... 88
4.5 Variational Bayesian Learning of the RPR .................................................... 89
  4.5.1 Calculation of $\{q_k^k(z_{0:t})\}$ ................................................................. 92
  4.5.2 Calculation of $g(\Theta)$ ......................................................................... 93
  4.5.3 Computation of Lower Bound ............................................................... 94
4.6 Experimental Results ................................................................................... 95
  4.6.1 Performance as a Function of Number of Episodes ................................. 96
  4.6.2 Comparison to Other Algorithms ............................................................ 97
4.7 Conclusions on RPR ................................................................................... 98
4.8 Appendix ..................................................................................................... 100
5 Reinforcement Learning in Multiple Partially Observable Stochastic Environments

5.1 Introduction .................................................. 105

5.2 Dirichlet Process ............................................. 107

5.2.1 Some Properties of the RPR regarding Experience Transfer ............................................. 109

5.3 Joint RRRPs for Multiple Stochastic Environments ...................................................... 110

5.3.1 Basis Formulation ......................................... 110

5.3.2 Multi-RPR Learning by Gibbs Sampling ...................................................... 114

5.4 The Hybrid Gibbs-Variational Algorithm for Multi-RPR Learning ...................................... 119

5.4.1 The \((q, g)\)-scheme ........................................ 119

5.4.2 The Gibbs-Variational Algorithm ...................................................... 120

5.4.3 Distributive Computation ...................................... 122

5.5 Experimental Results ........................................ 123

5.5.1 Results on Multiple Partially Observable Stochastic Environments ...................................... 124

5.5.2 Results on Multi-task Multi-Aspect Classification ...................................................... 131

5.6 Conclusions for RPR-MTL ..................................... 138

6 Reward-Directed Bayesian Classification with POMDP ...................................................... 139

6.1 Introduction .................................................. 139

6.2 Intuitive Description of the Reward-Directed Classifier (RDBC) ............................................ 140

6.3 Mathematical Formulation of the RDBC ...................................................... 142

6.4 Learning of the RDBC ........................................ 144

6.5 Experimental Results ........................................ 145

6.6 Conclusions for Cost-Sensitive Classification ...................................................... 149
List of Tables

1.1 Value Iteration for MDP ........................................... 8
1.2 Policy Iteration for MDP ........................................... 8
2.1 The Proposed RBVI Algorithm with EM as a Parameter Estimator . 36
2.2 Results on the benchmark problems, where $T$ denotes time in seconds, $|\Gamma|$=number of $\alpha$-vectors, and n.v.=not available. The results marked with $*$ are those we obtained by coding the respective algorithms in Matlab; other results may have been coded in languages other than Matlab and executed on computer platforms different from ours. (To be continued on Table 2.3) ........................................... 39
2.3 Results on the benchmark problems (Continued from Table 2.2) . . . 40
2.4 Time Comparison in Big O Notation, where dim($\tilde{b}$) denotes the dimensionality of $\tilde{b} = U^T \tilde{b}$ and $U$ is a matrix whose orthonormal columns span the space of $\tilde{b}$, and $N_\Gamma$ denotes the number of EM or VBEM iterations. ........................................... 43
3.1 The ILSPI Algorithm: Policy Evaluation ........................................... 62
3.2 The ILSPI Algorithm: Main Loop ........................................... 63
3.3 Time and Memory Complexity of the ILSPI ........................................... 65
3.4 The Piecewise Linear Policy iteration Algorithm ........................................... 68
3.5 Results on the benchmark problems, where the time is displayed in seconds. The results marked with $*$ are those we obtained by coding the respective algorithms in Matlab; other results may have been coded in languages other than Matlab and executed on computer platforms different from ours. ........................................... 70
4.1 A comparison of the RPR to other reinforcement learning algorithms on Hallway2 ........................................... 100
5.1 The Gibbs sampling algorithm for multitask-RPR learning ........................................... 118
5.2 The hybrid Gibbs-variational algorithm for multitask-RPR learning . 122

6.1 Observation Cost of the Pima dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . 146
List of Figures

1.1 An MDP model. ........................................... 4
1.2 The belief state transformation $T(b|a,o)$ for the three-state case. .......... 12
1.3 The POMDP is decomposed into two components: state estimator (SE) and an policy ($\pi$). ........................................... 13
2.1 The finite-horizon optimal value function $V^n(b)$ is a maximization over a finite set of $\alpha$-vectors: an illustration for a two-state POMDP. .............. 20
2.2 Partition of the belief simplex into $K$ ellipsoidal regions, such that the optimal value function $V(b)$ is linear in $b$ in each region. Here it is assumed that $|S| = 3$ and the true piecewise linear convex partition are $3$ polygons. $K = 3$ ellipses are used to cover (probabilistically) the belief simplex and preserve the piecewise linear convexity of $V(b)$. .......................... 22
2.3 Illustration of Benchmark problems. ........................................... 38
2.4 The $\omega_k = p(z = k)$ estimated by VBEM. .......................... 41
2.5 The eigenvalues of the covariance matrix of the belief samples in $B$ for the four benchmark problems. .......................... 44
3.1 The policy iteration consists of two phases: policy evaluation and policy improvement. ........................................... 53
3.2 The value functions $V^\pi(b)$ of each ILSPI iteration for the tiger problem [Kaelbling et al. (1998)], which has $|S| = 2$. The horizontal axis is $b(1) = p(s = \text{“tiger on the left”}|\text{history})$. The vertical axis is $V$. The initial policy is a random one. Green cricle denotes the action “open left door”; blue plus denotes the action “open right door”; red dot denotes the action “listen”. .......................... 71
3.3 Comparison of the converged ILSPI policy with the exact policy, for the tiger problem. Green cricle denotes the action “open left door”; blue plus denotes the action “open right door”; red dot denotes the action “listen”. .......................... 72
4.1 Results of the RPR on the Hallway problem. The horizontal axis is log_{10} of the number of episodes $K$ used in learning the RPR. The horizontal axis in each sub-figure is (a) Goal rate (b) Discounted accumulative reward (c) Number of steps to reach the goal (d) Time in seconds for learning the RPR.

4.2 Results of the RPR on the Hallway2 problem. The horizontal axis is log_{10} of the number of episodes $K$ used in learning the RPR. The horizontal axis in each sub-figure is (a) Goal rate (b) Discounted accumulative reward (c) Number of steps to reach the goal (d) Time in seconds for learning the RPR.

5.1 Block diagram of the distributive Gibbs-variational algorithm for multitask-RPR learning.

5.2 The three distinct grid-world environments, where the goal is designated by the basket of food, each block indicates a cell in the grid world, and the two gray cells are occupied by a wall. The red dashed lines in (a) and (c) indicate the similar parts in the two environments. The agent locates himself by observing the openness of a cell in the four orientations. Both the motion and the observation are noisy.

5.3 Results on the problem of multiple stochastic environments (a) Average success rate for the agent to reach the goal within 15 steps (b) Average step for the agent reaching the target. The red solid line with stars denotes the results of RPR-MTL; the green dotted line with diamonds denotes the results of RPR-STL; and the blue dashed line with circles denotes the results of RPR-Pooling.

5.4 Results on the problem of multiple stochastic environments: Hinton diagrams for the between-task similarity matrix, where the number of training episodes is equal to (a) 3 (b) 10 (c) 60 (d) 120.

5.5 A typical configuration of multi-aspect classification of underwater objects.

5.6 Frequency-domain acoustic responses of the five underwater objects (a) Target-1 (b) Target-2 (c) Target-3 (d) Target-4 (e) Clutter.

5.7 (a) Average classification rate in the problem of shell target classification (b) Zoomed-out version of the figure in (a), when the number of episodes is equal or less than 40. Red solid line with stars denotes the results of RPR-MTL; green dotted line with diamonds denotes the results of RPR-STL and blue dashed line with circles denotes the results of RPR-Pooling.
5.8 Results on the problem of shell target classification: Hinton diagram for the 
between similarity matrix, where the number of training episodes is equal 
to (a) 10 (b) 30 (c) 110 (d) 150. ................................. 137

6.1 Representation of feature dependence for a given class in the RDBC. Each 
node is dependent on (and only on) the nodes that emanates a directed 
edge to it. Though the internal state \( s \) is Markovian, the observation \( o \) is 
not, therefore the dependence among \( o_1 \cdots o_n \) is well represented. ................................. 142

6.2 An illustration of state transitions in the proposed RDBC. A solid circle de-
notes a state of class 1; a hollow circle denotes a state of class 2; the diamond 
denotes the terminal state. A directed edge connecting two states denotes a 
transition from the initial state to the destination state; the number marked 
by the edge denotes the probability of the associated state transition; an 
edge with no numbers indicates that the associated transition probability 
is to be estimated from training data. ................................. 144

6.3 Correct classification rate as a function of false prediction cost. The mean 
and error bars are generated from 10 independent trials of random split of 
training and test instances. ................................. 147

6.4 Observation cost averaged over test instances, as a function of false pre-
diction cost. The mean and error bars are generated from 10 independent 
trials of random split of training and test instances. ................................. 148

6.5 Feature repetition rate averaged over test instances, as a function of false 
prediction cost. The mean and error bars are generated from 10 independent 
trials of random split of training and test instances. ................................. 148
Chapter 1

Introduction

1.1 Sequential Decision Making

In our daily life we frequently face the problem of making sequential decisions to maximize a certain optimality criterion. For example, politicians need to make policies to guarantee the long-term development of a country. A businessman may want to decide from time to time on how much money to invest on maintenance and how many products to produce so as to maximize future profits. A doctor needs to make decisions on what kinds of treatments should be given to his patients, such that diagnosis accuracy is guaranteed and yet unnecessary medical cost is avoided.

In the field of artificial intelligence (AI), sequential decision making plays a central role (Russell and Norvig 1995). For example, a board game player needs to calculate a sequence of moves to increase his chance of winning the game. A robot navigating in an environment needs to choose from available actions to reach an intended goal.

In a more formal formulation, the decision maker is called an agent, and the context in which the agent operates is called an environment. The agent perceives the environment through sensors and acts upon the environment through effectors (Russell and Norvig 1995), the former resulting in collection of observations, and the latter consisting of choosing from available actions and executing the action selected. The situation which changes from time to time (resulting from the actions of the agent) is called the state of the environment. The state is an important concept to characterize the aspects of the environment that are relevant to the decision-making. In the robot example, the state can represent the robot’s position in the environment, the remaining power in batteries, and the status of sensors equipped on the robot.
Different problems have different definitions of states, and a good definition makes the problem easier to solve. The immediate optimality of the agent’s performance is quantitatively measured by a special observation called reward, and the objective of the agent is to maximize the rewards accumulated in the long run. The reward is problem-dependent: it is the money earned in the businessman example and it indicates correct diagnosis as well as medical cost savings in the patient example. In the robot example, the reward is used to designate the goal state and the savings on battery usage.

In a stochastic environment, the result is not deterministic but probabilistic when the agent executes an action. Therefore, executing the same sequence of actions multiple times could lead to different paths in the state space and therefore different total rewards. The agent must take into account the uncertainty in the result of an action when choosing actions, and this is accomplished by maximizing the expected total reward in the long run. The maximization returns a policy, which gives the rule for choosing actions in all possible states.

The optimality of a policy depends on how far the agent looks into the future. If the agent does not look ahead and maximizes the immediate reward, the policy is known as myopic. If the agent looks $n$ steps ahead, and maximizes the expected total rewards accumulated over $n$ steps in the future, the policy is called a non-myopic policy with a horizon length $n$. In the extreme, the agent looks infinite steps ahead and obtains an infinite-horizon policy; in this case, future rewards are typically discounted to make the total reward bounded.

If a policy is invariant to time then the policy is stationary; otherwise, it is called a non-stationary policy. An finite-horizon policy is typically non-stationary since the policy’s optimality changes as the agent moves towards the end of its lifetime; on the contrary, an infinite-horizon policy is usually stationary since the agent always
assumes an infinite life before it (Kaelbling et al. 1998).

In many cases, the observation received in a given state does not fully characterize that state, and this situation arises when the agent has low-quality sensors or one part of the environment is intrinsically similar to another. When this happens, the environment becomes only partially observable to the agent, and the agent must take this into consideration when defining the optimality of the policy. In a partially observable stochastic environment, the decision making becomes very challenging, since the agent must consider all possible future possibilities given all possible histories.

The methods of sequential decision making can be classified into two broad categories — model-based methods and model-free models. In the first, the model of the environment is assumed known and given to the agent, and the agent calculates the optimal policy based on the model — this is called model-based planning. In the second case, the model is unknown and the agent learns the policy using a try-and-improve approach — this is called model-free learning or reinforcement learning. Of the following three subsections, Section 1.2 and 1.3 are concerned with model-based planning and Section 1.4 is concerned with model-free learning.

1.2 Markov Decision Processes (MDPs)

Markov decision processes (MDPs) (Bellman 1957; Puterman 1994; Kaelbling et al. 1996; Sutton and Barto 1998; Gordon 1999) are well-understood and intensively investigated models in sequential decision problems. In MDPs the states are fully observable and the state transitions satisfy the Markov property. The Markov property means that the state of the process at time \( t + 1 \) is only dependent on the state at time \( t \) and independent of all previous states. Despite of the simplicity, the MDP model provides very useful insights, concepts, and methods for studying a wider range of sequential decision making problems.
In the finite case, the MDP is a stochastic model that models the features of a dynamic system in terms of a finite set of observable states, with action dependent state-transition probabilities represented as a Markov process. After the next state is entered, the agent receives a reward that is dependent on the next state, the current state, and the action taken by the agent in the current state. Figure 1.1 shows that the agent takes the current state as an input and takes an action according to a policy, which in turn affects the next state of the world (environment). In the MDP framework, while there may be uncertainty about the effects of an agent’s action, there are no ambiguities about the state the agent arrives to when the action is completed — because the agent observes the state completely. By using the MDP formulation, the complexity of sequential decision making is greatly reduced and finding an optimal policy becomes manageable.

Figure 1.1: An MDP model.
1.2.1 Formal definition of the MDP

In this section, we give a formal definition of the MDP. The finite MDP is defined by a tuple \((S, A, T, R)\), where

- \(S\) is a finite set of states;
- \(A\) is a finite set of actions;
- \(T : S \times A \rightarrow \Pi(S)\) are action-dependent state-transition matrices with \(T_{ss'}^a = Pr(s_{t+1} = s' | s_t = s, a_t = a)\) denoting the probability that the agent lands in state \(s'\) by taking action \(a\) in state \(s\);
- \(R : S \times A \rightarrow R\) is the reward function, giving the expected immediate reward received by the agent for taking an action in each state; \(R(s, a) = \sum_{s' \in S} r_{ss'}^a T_{ss'}^a\) is the expected immediate reward for taking action \(a\) in state \(s\), and \(r_{ss'}^a\) is the immediate reward accrued when taking action \(a\) in state \(s\) results in entering state \(s'\).

The quantities \(T_{ss'}^a\) and \(R(s, a)\) (or equivalently \(r_{ss'}^a\)) completely specify the most important aspects of the dynamics of a finite MDP. Since the next state \(s'\) is only dependent on the current state \(s\) and current action \(a\) and the reward is also independent of all previous history given the current state \(s\) and action \(a\), all the previous changes in the environment are summarized in the current state. Given the current state, the decision about the action does not rely on the previous states or actions. Since the state at each time step in an MDP model is fully observable, the goal of the MDP is to find a policy \(\pi\) mapping any state \(s\) to an associated optimal action \(a\) with the objective of maximizing some measure of the rewards received in the long run. There are three measures that have been widely used \([Kaelbling et al., 1996]\).

Infinite-horizon expected sum of discounted rewards is a criterion in which the agent takes all the future rewards into account and acts optimally so as to maximize
the expected sum of future rewards $\mathbb{E}[\sum_{t=0}^{\infty}(\gamma^t r_t)]$ over an infinite horizon, but the rewards received in the future are geometrically discounted according to discount factor $\gamma$, (where $0 < \gamma < 1$). The larger the discount factor $\gamma$, the more effect future rewards have on the current decision.

*Finite-horizon expected sum of rewards* only considers the rewards for the next $n$ steps and acts optimally so as to maximize the expected sum of future rewards $\mathbb{E}[\sum_{t=0}^{n} (r_t)]$ over a finite horizon. Since the precise length of an agent’s life is not often known in advance, this criterion is not always appropriate. One may also include a discount factor if desired.

The third criterion is *average expected sum of rewards* $\lim_{n \to \infty} \mathbb{E}[\frac{1}{n} \sum_{t=0}^{n} (r_t)]$, in which the agent is supposed to take actions that maximize its long-term average reward. This criterion can be seen as the limiting case of *Infinite-horizon expected sum of discounted rewards* as the discount factor $\gamma$ approaches to 1.

These three optimality measures of long-run rewards have their applications in different problems. In this thesis, we will focus on the first criterion – *Infinite-horizon expected sum of discounted rewards*.

### 1.2.2 Solutions to MDPs

As we mentioned in Section 1.2.1, the goal of the MDP is to find a policy $\pi$, which assigns an action to each given state $s$, with the objective of maximizing the rewards received over a horizon $n$.

The standard algorithms for finding an optimal policy for an MDP involve the computation and storage of two arrays indexed by state: value function $V(s)$ and policy $\pi(s)$. The value function $V(s)$ is a measure of the goodness of an agent being in a given state $s$ in terms of the amount of expected sum of future rewards. The policy $\pi(s)$ yields an action for each state. Assume that we use the *infinite-horizon expected...*
sum of discounted rewards as the policy optimality criterion, then the infinite-horizon value function for a given policy $\pi$ is formally defined as (Sutton and Barto 1998)

$$V^\pi(s) = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right\}$$

(1.1)

and the value function satisfies the Bellman equations

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} T_{ss'}^{\pi(s)} V^\pi(s')$$

(1.2)

By the policy improvement theorem (Sutton and Barto 1998),

$$\pi'(s) := \arg \max_a \left[ R(s, a) + \sum_{s'} T_{ss'}^a V^\pi(s') \right]$$

(1.3)

is a policy that is improved over $\pi$, where the bracketed quantity is the action-value function.

The two standard MDP algorithms, value iteration and policy iteration, are summarized in Table 1.1 and Table 1.2. It is easily seen that both algorithms are based on (1.2) and (1.3). In fact, at each iteration in Table 1.1 (1.2) and (1.3) is each applied once and only once; at each iteration in Table 1.2 (1.2) is applied until convergence and then (1.3) is applied once. In (Puterman 1994), a third MDP algorithm called modified policy iteration, is described. The modified policy iteration is essentially a sequence of iterations, at each of which, (1.2) is applied multiple times (not waiting for convergence) and then (1.3) is applied once.

### 1.3 Partially Observable Markov Decision Processes (POMDPs)

In many cases, however, the strong assumption that the states are fully observable in MDPs is often not true and we can only have access to observations with state-
Table 1.1: Value Iteration for MDP
1. Initialize $V_0(s) = 0$, for all state $s \in S$; Let $n = 1$
2. Repeat until $\max_s |V_n(s) - V_{n-1}(s)| < \Delta$ (a small positive number)
   For $s = 1 : |S|$
   Compute $V_{n+1}(s) = \max_a [R(s, a) + \gamma \sum_{s'} T^{a}_{ss'} V_n(s')]$
   Increase $n$ by 1
4. Output the policy $\pi(s) := \arg \max_a [R(s, a) + \gamma \sum_{s'} T^{a}_{ss'} V_n(s')]$

Table 1.2: Policy Iteration for MDP
1. Randomly initialize $\pi(s)$, for all state $s \in S$
2. Policy Evaluation
   Let $\pi' = \pi$, $n = 1$, and randomly initialized $V(s)$;
   Repeat until $\max |V_{n+1}^\pi(s) - V_n^\pi(s)| < \Delta$
   For $s = 1 : |S|$
   Compute $V_{n+1}^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} P^\pi(s) V_n^\pi(s')$
3. Policy Improvement
   For $s = 1 : |S|$
   $\pi'(s) = \arg \max_a [R(s, a) + \gamma \sum_{s'} T^{a}_{ss'} V_n^\pi(s')]$
   If $\pi'(s) = \pi(s)$, output $\pi(s)$; Otherwise, go to 2

dependent statistics (while the underlying and hidden states are still Markovian).

Robot navigation is a typical example of such a problem. The robot is often unable
to produce the motion exactly as it intends to, because of the ruggedness of the terrain
and the low precision of motion; the robot cannot locate itself precisely, because of the
noise-corrupted observations or the ambiguities of the environment. The robot must
have the ability to plan and make decisions under these two uncertainties in order to
navigate in such an environment. The partially observable Markov decision process
(POMDP) (Sondik 1971; Kaelbling et al. 1998) is such a model, dealing with the
unobservable states using state-dependent observation statistics and the underlying
process is a discrete-time finite-state Markov system. The POMDP model provides a
rich mathematical framework by handling the uncertainty in a probabilistic manner
that facilitates the theoretical analysis of making optimal sequential decisions. In the
POMDP model, the agent need not know exactly what are the consequences of taking
an action, instead it maintains a distribution over all possible consequences; the agent
need not locate itself precisely, instead it maintains a conditional distribution over
all possible locations given its past experiences.

1.3.1 Formal definition of the POMDP

Consider a finite horizon and assume there are \( n \) control intervals remaining until the
horizon is reached. Since states are not fully observable, the agent has to select action
\( a \) based on the available information up to the present to maximize the expected sum
of future rewards. The action selected results in the transition from the current
(unobservable) state \( s \) to another \( s' \). The resulting state \( s' \) is not directly observable,
but a random observation \( o \) is made that depends probabilistically on the current
state \( s' \) and the action \( a \); at the same time an immediate reward \( r \) is obtained that
is a function of the observation \( o \), the current state \( s' \), the previous state \( s \), and the
action \( a \) taken in \( s \).

The partially observable Markov decision process (POMDP) is a six tuple \((S, A, T, O, \Omega, R)\) where \((S, A, T)\) are defined exactly the same as in the MDP and

- \( O \) denotes the finite set of possible observations that the agent can make;

- \( \Omega \) is defined by the probability \( \Omega_{s'ao}^a \) of making observation \( o \) after performing
  action \( a \) and transiting to state \( s' \);

- \( R \), the immediate expected reward, is also related to the observation \( o \), defined
  slightly different from that in the MDP; \( R(s, a) = \sum_{o \in O} \sum_{s' \in S} T_{ss'}^a \cdot r_{ss'}^a \cdot \Omega_{s'ao}^a \) is
  the expected immediate reward when taking action \( a \) in state \( s \); and \( r_{ss'}^a \) is the
  immediate reward, if by taking action \( a \) the agent makes a transition from state
  \( s \) to state \( s' \) and observes \( o \).
Two types of uncertainties are included in the POMDP model. First, given an action, the state transition is not certain but occurs with probabilities defined by $T$. Second, after landing in a state, the observation made is not certain but is governed by the probabilities as defined by $\Omega$.

1.3.2 Belief states

Though the POMDP is conceptually simple, finding the optimal policy $\pi$ for a POMDP is often much more challenging than for an MDP. The reason to which the difficulties are attributed, as we have stated several times, is that the state at each time step is not fully observable. The optimal policy we want to find for POMDPs should not be represented by a mapping from state to action, since the state is now unaccessible; instead it should be a mapping from history to action, because a history of actions and observations represents the information available to the agent. As history length increases, the number of possible histories increases exponentially. Such a huge number of histories with different lengths poses problems to CPU and memory. Fortunately, however, it has been proved in \cite{Smallwood1973} that the belief state is a sufficient statistics summarizing the whole history up to the present time. Instead of maintaining the history, the agent maintains a belief state, defined as the probability distribution over the states given past actions and observations (this assumes that the agent has knowledge of the underlying statistical model defined by $T_{ss'}^{a}$ and $\Omega_{s'o}^{a}$). With a new action and the resulting observation, the belief state is updated according to Bayes rule, which only depends on the last belief state. Since the belief state satisfies the Markovian property it constitutes a continuous-state Markov decision process \cite{Smallwood1973}, and this notion plays an important role in simplifying the search for the optimal policy for POMDPs.
For convenience of exposition, we denote by $h$ the history of information (actions and observations) available up to the present time step. We use the vector $b = [b(1), b(2), \ldots, b(N)]^T$ to denote the belief state at the present time step, where $b(i) = P(s = i|h)$ and the superscript $T$ denotes matrix (vector) transpose. If the next new action is $a$, which results in a new observation $o$, then the belief state is updated by Bayes rule

$$
\tilde{b}_o^a(s') = \frac{\sum_{s \in S} b(s) T_{ss'}^a \Omega_{s'o}^a}{p(o|b,a)}
$$

(1.4)

where

$$p(o|b,a) = \sum_{s' \in S} \sum_{s \in S} b(s) T_{ss'}^a \Omega_{s'o}^a
$$

(1.5)

is the normalization factor. Note we have used the subscript and superscript to explicitly indicate that $\tilde{b}_o^a$ is dependent on $a$ and $o$.

Equation (1.4) can be interpreted in two ways. The first interpretation is geometric, in which we say that (1.4) is a transform in the belief simplex — it transforms $b$ to $\tilde{b}_o^a$ given the new action $a$ and new observation $o$. This first interpretation is illustrated in Figure 1.2 for a three-state POMDP, where the equivalent expression $T(b|a,o) = \tilde{b}_o^a$ is used for notational convenience. It is important to note that the transform $T(\cdot|a,o)$ is a continuous transform, i.e., $T(b_2|a,o) \rightarrow T(b_1|a,o)$ as $b_2 \rightarrow b_1$.

The continuity of $T(\cdot|a,o)$ plays an important role in proving the piecewise linear convexity of the POMDP optimal value function (Smallwood and Sondik 1973).

The second interpretation is probabilistic, in which we say that given the new action $a$ and new observation $o$, the old belief state $b$ transits to the new belief state $\tilde{b}_o^a$ with probability probability $p(o|b,a)$. The second interpretation implies that although the belief state is continuous, their transition probability is discrete: from any given $b$, one can only make a transition to a finite number of new belief states.
\[ \{ b^a_0 : a \in \mathcal{A}, o \in \mathcal{O} \} \]. For any action \( a \in \mathcal{A} \), the belief transition probabilities are given by:

\[
P_T(b'|b, a) = \begin{cases} 
p(o|b, a), & \text{if } b' = b^a_0 \\ 0, & \text{otherwise} \end{cases} \tag{1.6}
\]

Since there are finite number of actions and observations, the total number of transitions is \( |\mathcal{A}| |\mathcal{O}| \).

Note that the belief state here is defined entirely by the actions and observations, and it is independent of the observed rewards; the rewards are used to quantify the value of the actions. The POMDP then can be decomposed into two parts, as shown in Figure 1.3.

The component SE is responsible for updating the belief state and it does not use the reward, and the policy is responsible for producing actions for each belief state and it is based on the reward. The goal of the POMDP planning problem is to find a policy mapping any belief state to an associated optimal action, with the objective
of maximizing the expected future reward of each belief state over a finite or infinite horizon.

1.3.3 Solutions to POMDPs

Although the introduction of the belief state greatly simplifies the POMDP problem and converts a POMDP into a continuous-state MDP, finding the optimal policy in an exact way is still computationally intractable for most POMDPs. Currently, most researchers in this field are putting the focus on developing approximate algorithms for POMDPs. In this thesis, part of our work concentrates on developing approximate algorithms for POMDPs. Previous POMDP algorithms were developed from three directions: value iteration, policy iteration and policy search, which will be described in detail in the following chapters.
1.4 Model-free Reinforcement Learning

The discussions in the previous sections assumed that the underlying statistical model of the environment is available to the agent. With the model at hand, the agent can predict the future possibilities when taking an action in any situation. By taking an expectation of all future possibilities the optimal policy can be calculated and this is a planning problem.

When the model is unknown, the agent cannot compute such quantities as the belief state. In this case, the agent must learn empirically, based on a sequence of interactions with the environment, these defining a set of experiences. There are two ways to use the experiences: either they can be used to estimate a model of the environment (which then is used to generate a policy as above) or they can be used to learn a policy directly.

Although both options use experiences, model estimation could require more experiences than direct policy learning. This is because the model is a complete description of the environment and a full exploration is required, whereas the policy is concerned with maximal future rewards and may only need a partial exploration of the environment. In direct policy learning, the rewards serve as reinforcements, i.e., the stimuli that encourage certain actions and discourage other actions, hence the learning is often termed reinforcement learning.

When the environment is fully observable, the agent has access to the states even though it is ignorant of the state transitions. Access to states makes reinforcement learning a lot easier, since a state then works like a memory cell that stores the reinforcement value associated with that state. The value function can be easily estimated by adding the rewards received at each step accumulatively to the corresponding state. This gives rise to many MDP reinforcement algorithms like TD-learning (Sutton 1978 1988), Q-learning (Watkins and Dayan), Sarsa (Sutton [14]...
and Barto [1998].

In a partially observable environment, however, reinforcement learning becomes much more complicated. This is because the memory cells for the agent to store the reinforcement values, the belief states, are now not available to the agent — the belief state is not computable due to the lack of a model of the environment.

Without belief states, the agent relies on the raw history to memorize the past. This is obviously a crude method to use. The key to reinforcement learning in partially observable environments is to find a good method to compactly represent the history.

1.5 Structure of the Thesis

This thesis consists of three primary parts. The first part includes Chapter 2 and Chapter 3, which focus on developing model-based planning algorithms for partially observable Markov decision processes (POMDP). The second part is composed of Chapter 4 and Chapter 5, where we develop model-free reinforcement learning algorithms for single as well as multiple partially observable stochastic environments. The third part of the thesis is Chapter 6, in which we develop a POMDP-based method for non-myopic sequential feature selection in Bayesian classification.

Each chapter in this thesis contains new algorithms that are relatively independent of other chapters, therefore each chapter has its own introduction, literature review, and conclusions. Nevertheless, the chapters are coherently organized by the same theme, which is: planning and learning under stochasticity and partial observability. The outline of each chapter is as follows.

In Chapter 2 we propose an efficient value iteration algorithm for POMDPs. The algorithm, called region-based value iteration (RBVI), approximates the polyhedral deterministic partition of the belief simplex with an ellipsoidal probabilistic parti-
tion, such that the optimal value function is linear in each ellipsoidal region. We present results in comparison against state-of-the-art POMDP algorithms, as well as algorithmic complexity analysis.

Chapter 3 contains two new policy iteration algorithms that we have proposed for POMDPs. The first one, called incremental least-square policy iteration (ILSPI), represents the infinite-horizon value function as a linear span of basis functions. The ILSPI performs policy evaluation using a simple incremental procedure and improves the policy via pointwise maximization, both making the ILSPI an efficient algorithm. The second algorithm proposed in Chapter 3 is called piecewise linear policy iteration (PLPI), which includes the well-known point-based value iteration (PBVI) as a special case. Experimental results as well as complexity analysis are presented for both the ILSPI and PLPI algorithms.

Chapter 4 is concerned with reinforcement learning, assuming the POMDP model is unknown to the agent. We propose a new framework, called regionalized policy representation (RPR), for representing the history-dependent stochastic policy. Learning of the RPR is based on episodic interactions with the environments, and the empirical value function derived from the episodes. We present two algorithms, maximum-value (MV) estimation and variational Bayesian (VB) learning, to learn the RPRs. We experimentally evaluate the RPR on grid-world examples and compare it to state-of-the-art reinforcement learning algorithms.

In Chapter 5, we present a multitask learning (MTL) framework of the RPR. The framework, abbreviated as “RPR-MTL” unifies multiple RPRs, each associated with an environment, by having their prior distributions drawn from a common Dirichlet (DP) process. The RPR-MTL automatically identifies similar experiences from different environments and mingles them to augment the experience of each individual environment. The experience augmentation enhances policy learning across the envi-
environments, which is particularly advantageous when each environment is experience-poor. We develop Gibbs sampling as well as Gibbs-variational algorithms for the RPR-MTL, and present experimental results on grid world problems and real multi-aspect classification problems.

Chapter 6 is concerned with reward-directed Bayesian classification (RDBC) based on POMDPs. Placing a cost on each feature as well as a cost/reward on each false/correct prediction of class labels, the RDBC aims to maximize/minimize the total reward/cost associated with each individual instance of prediction. The RDBC can be used to perform cost-sensitive sequential feature selection of sensors, tests, or features. Results on medical diagnosis are presented to the merits of the RDBC.

In Chapter 7 we summarize the work of this thesis and point out some interesting directions for future research.
2.1 Introduction to Value Iteration for POMDPs

Many POMDP policy-design algorithms developed to date belong to the family of value iteration, which updates the optimal value function using dynamic programming. Value function recursively computes the optimal value function for the length-\(n\) horizon using the optimal value function of the length-(\(n-1\)) horizon, starting from a zero-length horizon to a horizon of arbitrary length. \cite{Smallwood1973} proved that for a finite horizon the optimal value function of a POMDP is piecewise linear and convex. Thus, the key problem in value iteration becomes finding the gradients, called \(\alpha\)-vectors, of the optimal value function for the length-\(n\) horizon given the \(\alpha\)-vectors of the length-\((n-1)\) horizon.

Let \(V^n(b)\) denote the optimal value function representing the maximum expected future reward of any belief point \(b\) for a horizon of length \(n\), and let \(\gamma \in [0,1)\) be a discount factor defined in Section 1.2.1. Then \(V^n(b)\) can be expressed recursively as

\[
V^n(b) = \max_{a \in A} \left\{ \sum_{s \in S} b(s) \sum_{s' \in S} T^n_{ss'} \sum_{o \in O} \Omega^a_o [r^a_{ss'} + \gamma V^{n-1}(b')] \right\}
\]

\[
= \max_{a \in A} \left\{ \sum_{s \in S} b(s) R(s, a) + \gamma \sum_{o \in O} \sum_{s' \in S} b(s) T^n_{ss'} \Omega^a_o V^{n-1}(b') \right\} \tag{2.1}
\]

where \(R(s, a)\) is the expected immediate reward,

\[
R(s, a) = \sum_{s' \in S} T^n_{ss'} \sum_{o \in O} \Omega^a_o r^a_{ss'} \tag{2.2}
\]
and $b'$ is determined by (1.4). The expression in (2.1) is reformulated as

$$
V^n(b) = \max_{a \in A} \left\{ \sum_{s \in S} b(s)R(s, a) + \gamma \sum_{o \in O} p(o|b, a)V^{n-1}(b') \right\}
$$

$$= \max_{a \in A} \left\{ R(b, a) + \gamma \sum_{b'} p(b'|b, a)V^{n-1}(b') \right\}
$$

(2.3)

Smulwod and Sondik (1973) proved that the solution to (2.3) has a simple form

$$V^n(b) = \max_k \left[ \sum_{s \in S} \alpha^n_k(s)b(s) \right] = \max_k (b^T \cdot \alpha^n_k)
$$

(2.4)

where the superscript $T$ denotes matrix (vector) transpose and $\alpha^n_k = [\alpha^n_k(1), \alpha^n_k(2), \cdots, \alpha^n_k(|S|)]^T$ is called an $\alpha$-vector. Equation (2.4) implies that for any finite horizon $V^n(b)$ is piecewise linear and convex in $b$ and the gradient of its $k$-th hyperplane is $\alpha^n_k$. In light of (2.4), the value iteration problem becomes one of computing $\alpha^n$ from $\alpha^{n-1}$, which is solved by substituting (2.4) and (1.4) into (2.1),

$$V^n(b) = \max_{a \in A} \left\{ \sum_{s \in S} b(s)R(s, a) + \sum_{o \in O} p(o|b, a) \max_k (\alpha^{n-1}_k \cdot b') \right\}
$$

$$= \max_{a \in A} \left\{ \sum_{s \in S} b(s)R(s, a) + \sum_{o \in O} \max_k \sum_{s' \in S} \sum_{s \in S} b(s)T^a_{ss}, \Omega^a_{s'o}\alpha^{n-1}_k(s') \right\}
$$

$$= \max_{a \in A} \left\{ \sum_{s \in S} b(s)[R(s, a) + \sum_{o \in O} \sum_{s' \in S} T^a_{ss'}, \Omega^a_{s'o}\alpha^{n-1}_{l(b,a,o)}(s')] \right\}
$$

(2.5)

where $l(b, a, o) = \arg\max_k \{\sum_{s \in S} b(s) \sum_{s' \in S} T^a_{ss'}, \Omega^a_{s'o}\alpha^{n-1}_k(s')\}$. As each bracketed expression in the rightmost side of (2.5) produces a possible $\alpha^n$, the number of distinct $\alpha^n$ can be as large as $|\mathcal{A}||\{\alpha^{n-1}\}||\mathcal{O}|$, where $|\mathcal{A}|$ and $|\mathcal{O}|$ denote the size of $\mathcal{A}$ and $\mathcal{O}$, respectively, and $|\{\alpha^{n-1}\}|$ denotes the number of distinct $\alpha^{n-1}$. It is clear that the number of distinct $\alpha^n$ will quickly become intractable as the horizon length $n$ increases.
Figure 2.1: The finite-horizon optimal value function $V^n(b)$ is a maximization over a finite set of $\alpha$-vectors: an illustration for a two-state POMDP.

2.2 Overview of Value Iteration Algorithms for POMDPs

Many of the $\alpha^n$ produced in the rightmost side of (2.5) are inactive or useless, where $\alpha^n_k$ is inactive if $b^T \alpha^n_k \leq b^T \alpha^n_j \forall j \neq k \forall b$. Figure 2.1 gives us an illustration of $\alpha^n_k$ for a two-state POMDP. The optimal value function $V^n(b)$, denoted by thick black line segments, is a piecewise linear and convex function in $b$. Different colors represents different $\alpha$-vectors. It can be seen that not all $\alpha$-vectors (such as $\alpha^n_3$ and $\alpha^n_4$) are useful in constituting the value function $V^n(b)$. There is a unique action corresponding to each $\alpha$-vector. Exact value iteration algorithms such as the one-pass algorithm (Smallwood and Sondik 1973), the linear-support algorithm (Cheng 1988), and the witness algorithm (Kaelbling et al. 1998) seek to identify the complete set of active $\alpha$-vectors, pruning away the inactive ones. However, the pruning requires solving many linear programs and could be computationally very expensive.
Point-based methods (Lovejoy 1991; Brafman 1997; Poon 2001; Zhang and Zhang 2001; Pineau et al. 2003) alleviate the computational load by focusing only on a finite set of belief points \( \mathcal{B} \). In particular, Pineau et al. (2003) suggested constructing \( \mathcal{B} \) as a set of points most uniformly sampled from the belief points reachable by the POMDP model, and backing up an \( \alpha \)-vector for each point in \( \mathcal{B} \); hence it avoids the pruning procedures of inactive \( \alpha \)-vectors. The number of \( \alpha \)-vectors for horizon \( n \) then becomes \(|A||\{\alpha^{n-1}\}||O||\mathcal{B}||\), which does not exponentially increase with the horizon length. The resulting algorithm, point-based value iteration (PBVI), was proven to be a practical POMDP solution scaling up to large problems. The idea of focusing on a subset of belief points instead of the whole belief simplex is quite enlightening and has subsequently been pursued in a number of papers (Spaan and Vlassis 2004; Smith and Simmons 2004, 2005), where various heuristics were proposed to further improve the algorithmic efficiency.

2.3 Region-based Value Iteration (RBVI) Algorithm for POMDP

As implied by (2.4), the belief simplex is partitioned into a finite number of polyhedral regions, such that \( V^n(b) \) is linear in \( b \) in each region. In the proposed RBVI algorithm we approximate this polyhedral partition with an ellipsoidal partition, in which each polyhedral region is approximated by one or more ellipsoidal regions and a single \( \alpha \)-vector is estimated and backed up for each ellipsoidal region. We derive efficient estimators \( \Upsilon \) to jointly estimate the \( \alpha \)-vectors as well as the position and shape parameters of each ellipsoidal region, using the samples \( \{b, V^n(b)\}_{b \in \mathcal{B}} \). For completeness and comparison, we give two versions of \( \Upsilon \), based on expectation-maximization (EM) and variational Bayesian EM (VBEM). Variational Bayesian EM (VBEM) also provides a principle way to select the number of ellipsoidal regions.
2.3.1 Ellipsoidal approximation of the polyhedral piecewise linearity

For a horizon of length $n$, the proposed RBVI algorithm is designed to probabilistically partition the belief simplex into a number of ellipsoidal regions such that in each of the regions the optimal value function $V^n(b)$ is linear in $b$. See Figure 2.2 for an illustration and (2.6) for the mathematical formulation. This ellipsoidal partition is an approximation of the true polyhedral partition (Smallwood and Sondik 1973). The position and shape of each ellipsoidal region, along with the $\alpha$-vector (gradient) of $V^n(b)$ in each region, are estimated from the samples $\{b, V^n(b)\}_{b \in B}$.

For notational simplicity, we write the optimal value function as $V(b)$ in the remainder of this section as well as Sections 2.3.2 and 2.3.3, dropping the dependence on the horizon length $n$. The discussions apply to any finite $n$, unless otherwise indicated.

**Figure 2.2:** Partition of the belief simplex into $K$ ellipsoidal regions, such that the optimal value function $V(b)$ is linear in $b$ in each region. Here it is assumed that $|S| = 3$ and the true piecewise linear convex partition are 3 polygons. $K = 3$ ellipses are used to cover (probabilistically) the belief simplex and preserve the piecewise linear convexity of $V(b)$.

We now formally describe the proposed ellipsoidal partition of the belief simplex. Let $B_k$ represent the $k$-th ellipsoidal region in which the optimal value function $V(b)$ is linear in $b$. Assume $\{B_k : k = 1, \cdots, K\}$ form an approximate partition of the
belief simplex. Then \( V(b) \) can be modeled as

\[
V(b) = \begin{cases} 
\alpha_1^T b + e_1 & \text{if } b \in B_1 \\
\alpha_2^T b + e_2 & \text{if } b \in B_2 \\
\vdots & \vdots \\
\alpha_K^T b + e_K & \text{if } b \in B_K
\end{cases}, \quad e_k \sim \mathcal{N}(0, \mu_k^2)
\]

where \( e_k \) is introduced to reflect any noise or error due to the approximation, and it is assumed to be a zero-mean Gaussian distribution with variance \( \mu_k \). The Gaussian noise assumption produces a least-square (LS) solution of \( \alpha_k, k = 1, \cdots, K \), using the data from region \( k \) (probabilistically defined) as will be shown in (2.20), which is reasonable since all belief samples are equally important (the importance has been accounted for in the sampling — a more important belief point will be sampled more frequently, see Section 2.3.4).

Letting \( z(b) = k \) indicate that \( b \in B_k \), we write (2.6) equivalently as

\[
p(V(b)|b, z(b) = k, \alpha_k) = \frac{1}{\sqrt{2\pi\mu_k^2}} \exp\left\{ -\frac{(V(b) - \alpha_k^T b)^2}{2\mu_k^2} \right\} \quad (2.7)
\]

The \( k \)-th ellipsoidal region in the belief simplex is represented probabilistically as

\[
p(b|z(b) = k, c_k, D_k) = \frac{1}{(2\pi)^{\frac{d}{2}} |D_k|^{\frac{1}{2}}} \exp\left\{ -\frac{1}{2} (b - c_k)^T D_k^{-1} (b - c_k) \right\} \quad (2.8)
\]

where \( d = |S| \) is the number of states in the POMDP model and \( c_k \) is a \( d \)-dimensional column and \( D_k \) is a \( d \times d \) matrix. The \( c_k \) and \( D_k \) respectively represent the position and shape parameters of the \( k \)-th ellipsoidal region. As the elements of \( b \) sum to one, \( b \) can be uniquely represented by its first \( d - 1 \) elements. In this representation, \( c \) and \( D \) will be \( d - 1 \) dimensional; \( \alpha \) will still be \( d \) dimensional but its last element becomes an intercept. The equations in this chapter are expressed in the original \( d \) dimensional belief simplex. However, they are easily converted to expressions in the \( d - 1 \) dimensional representation.
For any $b$, the prior distribution of $z(b)$ over the $K$ ellipsoidal regions is assumed to be $p(z(b) = k) = \omega_k$. The joint probability of $\{b, V(b), z(b)\}$ is given by

$$p(V(b), b, z(b) = k|\omega_k, c_k, D_k, \alpha_k) = \omega_k p(b|z(b) = k; c_k, D_k) p(V(b)|b, z(b) = k; \alpha_k)$$  \hfill (2.9)$$

Integrating out $z$ yields the joint probability of $(b, V(b))$,

$$p(V(b), b|\omega_k, c_k, D_k, \alpha_k) = \sum_{k=1}^{K} \omega_k p(b|z(b) = k; c_k, D_k)p(V(b)|b, z(b) = k, \alpha_k)$$  \hfill (2.10)$$

The proposed RBVI aims to find the $\alpha$-vectors based on a finite set of samples $\{b, V(b)\}_{b \in B}$, construction of which is discussed in Section 2.3.4. In the following two sections, we focus on the estimation of $\{\omega_k, c_k, D_k, \alpha_k\}_{k=1}^{K}$, assuming that this sample set is given.

### 2.3.2 Parameter estimation with expectation-maximization

We consider a given set of belief points $B = \{b_1, b_2, \cdots, b_{|B|}\}$ and the set of associated optimal values $V = \{V_1, V_2, \cdots, V_{|B|}\}$, where $V_i = V(b_i)$, $i = 1, 2, \cdots, |B|$. Assuming the pairs $(b_i, V_i)$, $i = 1, 2, \cdots, |B|$, are independent \footnote{Once the $B$ and $V$ are computed, everything is restricted to the given horizon $n$, hence the belief points are independent of each other as are their values.}, and denoting $z_i = z(b_i)$, we have the likelihood function for $(V, B)$,

$$p(V, B|\omega, c, D, \alpha) = \prod_{i=1}^{(|B|)} \sum_{k=1}^{K} \omega_k p(b_i|z_i = k; c_k, D_k)p(V_i|b_i, z_i = k, \alpha_k)$$  \hfill (2.11)$$

where $\omega = \{\omega_k\}_{k=1}^{K}$, $c = \{c_k\}_{k=1}^{K}$, $D = \{D_k\}_{k=1}^{K}$, and $\alpha = \{\alpha_k\}_{k=1}^{K}$.

We seek the parameters $\omega, c, D, \alpha$ that maximize the logarithmic likelihood function $\ln p(V, B|\omega, c, D, \alpha)$. Since $\omega_1, \cdots, \omega_K$ represent a discrete probability distribution, they must be nonnegative and sum to one. Therefore we have a constrained
optimization problem,

\[
\max_{\omega,c,D,\alpha} \sum_{i=1}^{L} \sum_{k=1}^{K} \omega_k p(b_i|z_i=k, c_k, D_k) p(V_i|b_i, z_i=k, \alpha_k) 
\]  

subject to \( \sum_{k=1}^{K} \omega_k = 1 \)

\( \omega_k \geq 0, k = 1, \ldots, K \)  

By Jensen’s inequality, for any \( \{\delta^i_k\}_{k=1}^{K} \) satisfying \( \sum_{k=1}^{K} \delta^i_k = 1 \), it holds

\[
\ln \sum_{k=1}^{K} \omega_k p(b_i|z_i=k, c_k, D_k) p(V_i|b_i, z_i=k, \alpha_k) 
\]

\[
= \ln \sum_{k=1}^{K} \delta^i_k \omega_k p(b_i|z_i=k, c_k, D_k) p(V_i|b_i, z_i=k, \alpha_k) 
\]

\[
\geq \sum_{k=1}^{K} \delta^i_k \ln \omega_k p(b_i|z_i=k, c_k, D_K) p(V_i|b_i, z_i=k, \alpha_k) - \sum_{k=1}^{K} \delta^i_k \ln \delta^i_k 
\]  

where the inequality becomes an equality when

\[
\delta^i_k = \delta^i_k(\omega, c, D, \alpha) = \frac{\omega_k p(b_i|z_i=k, c_k, D_k) p(V_i|b_i, z_i=k, \alpha_k)}{\sum_{k=1}^{K} \omega_k p(b_i|z_i=k, c_k, D_k) p(V_i|b_i, z_i=k, \alpha_k)} 
\]  

Let

\[
Q(\hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha}|\omega, c, D, \alpha)
\]

\[
= \sum_{i=1}^{L} \sum_{k=1}^{K} \delta^i_k(\omega, c, D, \alpha) \ln \left[ \omega_k p(b_i|z_i=k, \hat{c}_k, \hat{D}_k) p(V_i|b_i, z_i=k, \hat{\alpha}_k) \right] 
\]  

Iteratively finding \( Q(\hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha}|\omega, c, D, \alpha) \) (E-Step) and maximizing \( Q(\hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha}|\omega, c, D, \alpha) \) with respect to \( \hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha} \) (M-Step) constitutes the Expectation-Maximization (EM) algorithm.

In the E-Step, we use the current estimates to compute \( \delta^i_k(\omega, c, D, \alpha) \). In the M-Step, we maximize \( Q(\hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha}|\omega, c, D, \alpha) \) with respect to \( \{\hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha}\} \), subject o
the constraints in (2.12), the solutions of which are given analytically by

\[
\hat{\omega}_k = \frac{1}{|B|} \sum_{i=1}^{|B|} \delta_{ik}^i
\]  
(2.17)

\[
\hat{c}_k = \frac{\sum_{i=1}^{|B|} \delta_{ik}^i b_i}{\sum_{i=1}^{|B|} \delta_{ik}^i}
\]  
(2.18)

\[
\hat{D}_k = \frac{\sum_{i=1}^{|B|} \delta_{ik}^i (b_i - \hat{c}_k)(b_i - \hat{c}_k)^T}{\sum_{i=1}^{|B|} \delta_{ik}^i}
\]  
(2.19)

\[
\hat{\alpha}_k = \left[\sum_{i=1}^{|B|} \delta_{ik}^i b_i b_i^T\right]^{-1} \sum_{i=1}^{|B|} \delta_{ik}^i V_i b_i
\]  
(2.20)

and the detailed derivations of (2.17), (2.18), (2.19), and (2.20) are shown in Appendix 2.7.1.

### 2.3.3 Parameter estimation with variational Bayesian EM

As an alternative to standard EM, we consider the variational Bayesian EM (VBEM), which is known to be less sensitive to local maxima and singularities of the likelihood function \cite{Corduneanu2001}. It is also used to select the model order (the number of hyperellipsoids). Unlike EM, which finds a maximum-likelihood (ML) estimate of \((\omega, c, D, \alpha)\), VBEM finds an approximate posterior of \((c, D, \alpha)\) for given \(\omega\) (\(\omega\) is treated as the model’s hyper-parameter that controls model selection). Given \(B = \{b_1, \ldots, b_{|B|}\}\) and \(V = \{V_1, \ldots, V_{|B|}\}\), the logarithmic marginal likelihood is given by

\[
\ln p(V, B|\omega) = \ln \int_{\Theta} \sum_{z_1 \cdots z_{|B|} = 1}^K p(V, B, z, \Theta|\omega)\]

\[
\geq \ln \int_{\Theta} \sum_{z_1 \cdots z_{|B|} = 1}^K q(z, \Theta) \frac{p(V, B, z, \Theta|\omega)}{q(z, \Theta)}\]

\[
\geq \ln \int_{\Theta} \sum_{z_1 \cdots z_{|B|} = 1}^K q(z, \Theta) \ln \frac{p(V, B, z, \Theta|\omega)}{q(z, \Theta)}\]

\[26\]
\[ L(V, B|\omega) = \int_\Theta \sum_{z_{1:|B|}} q(z) q(\Theta) \ln \frac{p(V, B, z, \Theta|\omega)}{q(z) q(\Theta)} \]

\[ = \ln p(V, B|\omega) - KL(q(z) q(\Theta) || p(z, \Theta|V, B, \omega)) \] (2.22)

where \( KL(q||p) \) denotes the Kullback-Leibler distance between two probability distributions \( q \) and \( p \). Given model \( \omega \), VBEM maximizes the lower bound given by (2.22) or, equivalently, minimizes the KL distance between the variational posterior \( q(z) q(\Theta) \) and the true posterior \( p(z, \Theta|V, B, \omega) \). The maximization is achieved by alternatively solving for \( q(z) \) and \( q(\Theta) \), keeping one fixed while solving for the other.

Taking into consideration that \( \sum_{z_{1:|B|}} q(z) = 1 \) and \( \int_\Theta q(\Theta) = 1 \), the solutions can be easily obtained as

\[ q(z) \propto \exp \left\{ \langle \ln p(V, B, z|\Theta) \rangle_{q(\Theta)} \right\} \] (2.23)

\[ q(\Theta) \propto p(\Theta) \exp \left\{ \langle \ln p(V, B, z|\Theta) \rangle_{q(z)} \right\} \] (2.24)

where \( \langle \cdot \rangle_{q(\Theta)} \) denotes expectation with respect to \( q(\Theta) \). Iteratively finding \( q(z) \) (VB-E Step) and finding \( q(\Theta) \) (VB-M Step) constitutes the VBEM algorithm (Jaakkola 2001; Beal 2003).

As our likelihood function in (2.11) has an exponential form, we choose conjugate-exponential priors for the model parameters \( \Theta = \{\omega, c, D\} \), assuming the model parameters are independent of each other \textit{a priori},

\[ p(\Theta) = p(c, D, \alpha) = \prod_{k=1}^{K} N(c_k) W(D_k^{-1}) N(\alpha_k) \] (2.25)
where $N$ and $W$ represent normal and Wishart distributions, respectively,

$$N(c_k|D_k) = \frac{\exp\left(-\frac{1}{2}(c_k - m_k^0)^T \beta_k^{-1} D_k^{-1} (c_k - m_k^0)\right)}{Z_{c_k}}$$

(2.26)

$$W(D_k^{-1}) = \frac{\exp\left(-\frac{1}{2} \text{tr}(S_k^{-1} D_k^{-1})|D_k^{-1}|^{(\alpha_k^0 - d - 1)/2}\right)}{Z_{D_k}}$$

(2.27)

$$N(\alpha_k) = \frac{\exp\left(-\frac{1}{2}(\alpha_k - \eta_k^0)^T \Sigma_k^{-1} (\alpha_k - \eta_k^0)\right)}{Z_{\alpha_k}}$$

(2.28)

and $Z_{D_k} = 2^{n_k^0 d/2} \pi^{d(d-1)/4} |S_k^0|^{n_k^0 d/2} \prod_{j=1}^d \Gamma\left(\frac{n_k^0 + 1 - j}{2}\right)$, $Z_{c_k} = (2\pi)^{d/2} |\beta_k D_k|^{1/2}$, and $Z_{\alpha_k} = (2\pi)^{d/2} |\Sigma_k^0|^{1/2}$ are normalization constants. The variational posterior $q(\Theta)$ will have the same form as the prior, with the hyper-parameters given by (2.33), (2.34), (2.35), (2.36), and (2.37).

**VB-E Step** Substituting (2.9) into (2.23) and assuming the pairs $(V_i, b_i), i = 1 \cdots |B|$ are independent, we obtain $q(z) = \prod_{i=1}^{|B|} q(z_i)$, where, for $k = 1, \cdots, K$,

$$q(z_i = k) \propto \exp\left(\langle \ln p(V_i, b_i, z_i = k|\Theta) \rangle_{q(\Theta)}\right)$$

$$= \exp\left\{ \langle \ln \pi_k \rangle_{\pi} - \frac{d}{2} \ln 2\pi + \frac{1}{2} \langle \ln |D_k^{-1}| \rangle_{D_k} \right.$$

$$- \frac{1}{2} \text{tr}\left(\langle D_k^{-1} ((b_i - c_k)(b_i - c_k)^T) \rangle_{D_k}\right)$$

$$- \frac{1}{2} \ln 2\pi \mu_k^2 + \left\langle -\frac{1}{2\mu_k^2} (V(b_i) - \alpha_k b_i)^2 \right\rangle_{\alpha_k} \right\}$$

$$= \exp\left\{ \psi(\lambda_k) - \psi(\sum_k \lambda_k) - \frac{d}{2} \ln 2\pi \right.$$}

$$+ \frac{1}{2} \sum_{j=1}^d \psi\left(\frac{n_k + 1 - j}{2}\right) + \frac{1}{2} d \ln 2 + \frac{1}{2} \ln |S_k|$$

$$- \frac{1}{2} (b_i - m_k)^T n_k S_k (b_i - m_k) - \frac{1}{2} \text{tr}(\beta I_d)$$

28
\[-\frac{1}{2} \ln 2\pi \mu_k^2 - \frac{1}{2\mu_k^2} ((V_i - \eta_k^T b_i)^2 + b_i^T \Sigma_k b_i) \right\} \\
= A_k \mathcal{N} (b_i; m_k, n_k^{-1} S_k^{-1}) \\
\times \exp \left\{ -\frac{1}{2} \ln 2\pi \mu_k^2 - \frac{1}{2\mu_k^2} ((V_i - \eta_k^T b_i)^2 + b_i^T \Sigma_k b_i) \right\} \quad (2.29)

where \(\psi\) is the digamma function and \(I_d\) is \(d\)-dimensional identity matrix, and

\[ A_k = \exp \left\{ \psi(\lambda_k) - \psi(\sum_k \lambda_k) \right\} \\
+ \frac{1}{2} \sum_{j=1}^{d} \psi \left( \frac{n_k + 1 - j}{2} \right) + \frac{1}{2} d \ln 2 - \frac{1}{2} d \ln n_k - \frac{1}{2} \text{tr}(\beta_k I_d) \right\} \quad (2.30)\]

As \(\sum_{k=1}^{K} q(z_i = k) = 1\), we must normalize the rightmost side of (2.29) to get the magnitudes of \(q(z_i = k)\). For notational simplicity, we will use the abbreviations \(\delta_k^i = q(z_i = k)\), therefore

\[ \delta_k^i = q(z_i = k) \]

\[ = \frac{A_k \mathcal{N}(b_i; m_k, n_k^{-1} S_k^{-1}) \exp \left\{ -\frac{1}{2} \ln 2\pi \mu_k^2 - \frac{1}{2\mu_k^2} ((V_i - \eta_k^T b_i)^2 + b_i^T \Sigma_k b_i) \right\}} {\sum_{k=1}^{K} A_k \mathcal{N}(b_i; m_k, n_k^{-1} S_k^{-1}) \exp \left\{ -\frac{1}{2} \ln 2\pi \mu_k^2 - \frac{1}{2\mu_k^2} ((V_i - \eta_k^T b_i)^2 + b_i^T \Sigma_k b_i) \right\}} \quad (2.31)\]

**VB-M Step** Substituting (2.9), (2.25), (2.26), (2.27), (2.28) into (2.24) and assuming the pairs \((V_i, b_i), i = 1 \cdots, |\mathcal{B}|\) are independent, we have

\[ q(c, D, \alpha) \propto p(c, D, \alpha) \prod_{i=1}^{|\mathcal{B}|} \exp \left\{ \ln p(V_i, b_i, z_i = k|\omega, c, D, \alpha) \right\} \]

\[ \propto \prod_{k=1}^{K} \exp \left\{ -\frac{1}{2} \text{tr}(S_k^{-1,0} D_k^{-1}) \right\} \exp \left\{ -\frac{1}{2} \sum_{i=1}^{|\mathcal{B}|} \delta_k^i (b_i - c_k)^T D_k^{-1} (b_i - c_k) \right\} \\
\times \exp \left\{ -\frac{1}{2} \sum_{i=1}^{|\mathcal{B}|} \frac{\delta_k^i}{\mu_k^2} (\alpha_k^T b_i b_i^T \alpha_k - 2\alpha_k^T b_i V_i) \right\} \exp \left\{ -\frac{1}{2} (\alpha_k - \eta_k^0)^T \Sigma_k^{-1,0} (\alpha_k - \eta_k^0) \right\} \]
\[
\exp \left( -\frac{1}{2} \text{tr} \left[ (c_k - m_k^0)(c_k - m_k^0)^T \beta_k^{-1,0} D_k^{-1} \right] \right) |D_k^{-1}|^{\frac{1}{2}(n_k^0 + \sum_{i=1}^{|B|} \delta_k^i - d - 1)}
\]

(2.32)

It is seen from (2.32) that the variational posterior \( q(c, D, \alpha) \) has the same form as the prior \( p(c, D, \alpha) \), with the hyper-parameters derived in Appendix 2.7.2 and given here by

\[
\beta_k^{-1, \text{new}} = \beta_k^{-1,0} + \sum_{i=1}^{|B|} \delta_k^i
\]

(2.33)

\[
m_k^{\text{new}} = \frac{\beta_k^{-1,0} m_k^0 + \sum_{i=1}^{|B|} \delta_k^i b_i}{\beta_k^{-1, \text{new}}}, n_k^{\text{new}} = n_k^0 + \sum_{i=1}^{|B|} \delta_k^i
\]

(2.34)

\[
S_k^{-1, \text{new}} = S_k^{-1,0} + \sum_{i=1}^{|B|} \delta_k^i b_i b_i^T + \beta_k^{-1,0} m_k^0 m_k^{0,T} - m_k^{\text{new}} m_k^{\text{new,T}} \beta_k^{-1, \text{new}}
\]

(2.35)

\[
\Sigma_k^{-1, \text{new}} = \Sigma_k^{-1,0} + \frac{\sum_{i=1}^{|B|} \delta_k^i b_i b_i^T}{\mu_k^2}
\]

(2.36)

\[
\eta_k = (\Sigma_k^{-1, \text{new}})^{-1} (\Sigma_k^{-1,0} \eta_k^0 + \sum_{i=1}^{|B|} b_i V_i \delta_k^i)
\]

(2.37)

At the end of the VBEM algorithm, we let \( \alpha_k = \eta_k \) for \( k = 1, \ldots, K \), and use them as the \( \alpha \)-vectors in the POMDP optimal value function for the horizon length considered currently.

**Estimation of Hyper-parameters \( \omega \)** Following the (approximate) type-II maximum likelihood approach (Corduneanu and Bishop 2001), we estimate the model hyper-parameters \( \omega \) by maximizing the the lower bound as given by (2.22), which is evaluated as

\[
\mathcal{L}(\mathcal{V}, \mathcal{B}|\omega) = \int_{\Theta} \sum_{z_1, \ldots, z_{|B|} = 1} q(z) q(\Theta) \ln \frac{p(\mathcal{V}, \mathcal{B}, z, \Theta)}{q(z) q(\Theta)}
\]

30
\[
\sum_{z_1 \cdots z_K = 1} \int_{\Theta} q(z) q(\Theta) \ln \frac{p(\Theta) \prod_{i=1}^{\lvert B \rvert} \omega_{z_i} p(V_i, b_i | z_i, \Theta)}{q(z) q(\Theta)} = \sum_{i=1}^{\lvert B \rvert} \sum_{k=1}^{K} q(z_i = k) \ln \omega_k + \sum_{i=1}^{\lvert B \rvert} \sum_{z_i = 1}^{K \sum_{z_i = 1}^{K}} q(c, D, \alpha) q(z_i) \ln p(V_i, b_i | c, D, \alpha, z_i)
\]

\[-KL(q(c, D, \alpha) | | p(c, D, \alpha)) - \sum_{i=1}^{\lvert B \rvert} \sum_{z_i = 1}^{K \sum_{z_i = 1}^{K}} q(z_i) \ln q(z_i) \]  \quad (2.38)

where the second term is calculated using (2.29) and (2.32) and is given by

\[
\sum_{i=1}^{\lvert B \rvert} \sum_{z_i = 1}^{K \sum_{z_i = 1}^{K}} \int_{c,D,\alpha} q(c, D, \alpha) q(z_i) \ln p(V_i, b_i | c, D, \alpha, z_i)
\]

\[
= \sum_{i} \sum_{k} q(z_i = k) \left( \int_{c,D} \ln p(b_i | z_i = k, c, D) q(c, D) dcdD + \int_{\alpha} \ln (V_i | b_i, z_i = k, \alpha) q(\alpha) d\alpha \right)
\]

\[
= \sum_{i} \sum_{k} q(z_i = k) \left\{ -\frac{d \ln 2\pi}{2} + \frac{1}{2} \sum_{j=1}^{d} \psi\left(\frac{n_k + 1 - j}{2}\right) + \frac{d \ln 2}{2} + \frac{\ln |S_k|}{2} - \frac{1}{2} \text{tr}(\beta I_d) \right. \\
- \frac{1}{2} (b_i - m_k)^T S_k (b_i - m_k) - \frac{1}{2} \ln 2\pi \mu_k^2 - \frac{1}{2\mu_k^2} \left( (V_i - \eta_k^T b_i)^2 + b_i^T \Sigma_k b_i \right) \right\} \quad (2.39)
\]

The third term in (2.38) can be obtained by substituting (2.25) and (2.32) into the KL distance definition, and is given by

\[
KL(q(\alpha_k) | | p(\alpha_k)) = KL(\eta_k^{\text{new}}, \Sigma_k^{\text{new}} | | \eta_k, \Sigma_k)
\]
\[
= - \frac{1}{2} \ln |\Sigma_k^{\text{new}}\Sigma_k^{0,-1}| - \frac{1}{2} d + \frac{1}{2} \text{tr}(\Sigma_k^{\text{new}}\Sigma_k^{0,-1}) \\
+ \frac{1}{2}(\eta_k^{\text{new}} - \eta_k^0)^T \Sigma_k^{0,-1} (\eta_k^{\text{new}} - \eta_k^0)
\]

(2.41)

\[
KL(q(D_k)||p(D_k)) = KL(n_k^{\text{new}}, \Sigma_k^{\text{new}}||n_k^0, \Sigma_k^0)
\]

\[
= \ln \frac{Z_{D_k}^{\text{old}}}{Z_{D_k}^{\text{new}}} + \frac{n_k^{\text{new}} - n_k^0}{2} \sum_{j=1}^d \psi\left(\frac{n_k^{\text{new}} + 1 - j}{2}\right) + d \ln 2 + \ln |\Sigma_k^{\text{new}}| \\
+ \frac{1}{2} n_k^{\text{new}} \text{tr}(\Sigma_k^{0,-1} \Sigma_k^{\text{new}} - I)
\]

(2.42)

where

\[
\frac{Z_{D_k}^{\text{old}}}{Z_{D_k}^{\text{new}}} = \frac{2n_k^0 d/2 \pi^{d(d-1)/4} |\Sigma_k^0|^{n_k^0/2} \prod_{j=1}^d \Gamma\left(\frac{n_k^0 + 1 - j}{2}\right)}{2n_k^{\text{new}} d/2 \pi^{d(d-1)/4} |\Sigma_k^{\text{new}}|^{n_k^{\text{new}}/2} \prod_{j=1}^d \Gamma\left(\frac{n_k^{\text{new}} + 1 - j}{2}\right)}
\]

(2.43)

Given \(q(c, D, \alpha)\) and \(q(z)\), we maximize \(L(V, B|\omega)\) with respect to \(\omega\). Considering \(\sum_{k=1}^K \omega_k = 1\), we obtain the solution as \(\omega_k = \frac{1}{|B|} \sum_{i=1}^{|B|} q(z_i = k)\). As the VBEM directly maximizes the lower bound \(L(V, B|\omega)\), it is guaranteed to increase monotonically and serves as a convergence criterion.

### 2.3.4 Selection of sample belief points \(B\)

Let \(B_0\) be a set of initial belief points at time \(t = 0\). For time \(t = 1, 2, \cdots\), let \(B_t = \{b' : p(o|b, a) > 0, b \in B_{t-1}, a \in A, o \in O\}\), where \(b'\) is given by (1.4). Then \(\bigcup_{t=0}^\infty B_t\) is the set of belief points reachable by the POMDP by starting from \(B_0\). It is known that for most practical POMDP models, \(\bigcup_{t=0}^\infty B_t\) covers only a subset of the belief simplex (Pineau et al. 2003). It is therefore sufficient to only find optimal actions for the belief points in \(\bigcup_{t=0}^\infty B_t\), instead of all points in the belief simplex.

However, the belief points in \(\bigcup_{t=0}^\infty B_t\) may still be infinite. To overcome this, we follow the idea in (Pineau et al. 2003) to perform pruning for \(B_t\) at every \(t\). For any
given \( b \in B_{t-1} \), let \( B_{t,b} = \{ b' : a \in A, \alpha \sim p(o|b,a) \} \), where \( b' \) is given by (1.4). Define \( \rho_t(b') \) as the \( L_1 \) distance of \( b' \) from \( \cup_{\tau=0}^{t-1} B_\tau \), i.e., \( \rho_t(b') = \min_{b'' \in \cup_{\tau=0}^{t-1} B_\tau} ||b' - b''||_1 \). Let \( b_{t,b,\text{max}}' = \arg \max_{b' \in B_{t,b}} \rho_t(b') \). We construct \( B_t = \{ b_{t,b,\text{max}}' : \rho_t(b_{t,b,\text{max}}') > \zeta, b \in B_{t-1} \} \), where \( \zeta \) is a threshold and is usually set to zero. That is, we prune down the points in \( B_t \) to retain only those that are no closer than \( \zeta \) to \( \cup_{\tau=0}^{t-1} B_\tau \) and that are farther away from \( \cup_{\tau=0}^{t-1} B_\tau \) than all other points expanded from the same point in \( B_{t-1} \). Since for every point in \( B_{t-1} \) we generate at most one point in \( B_t \), this pruning process produces a \( B_t \) that contains at most the same number of belief points as \( B_{t-1} \), for \( t = 1, 2, \cdots \). In fact, the number of points in \( B_t \) will begin to diminish at some \( t \) until finally at \( t^* \) we have zero points generated in \( B_{t^*} \), which indicates termination of the belief expansion. Clearly this can always be made to happen by using a large enough threshold \( \zeta \).

Upon termination of belief expansion at time \( t^* \), the set of sample belief points is obtained as \( B = \cup_{t=0}^{t^*} B_t \). From the belief pruning, it is clear that the belief points in \( B \) spread most uniformly among the belief points reachable by the POMDP and are therefore most representative.

Given any \( b \in B \), the associated optimal-value \( V^*(b) \) for the length-\( n \) horizon is computed from (2.5), using the \( \alpha^{n-1} \) estimated for the length-(\( n - 1 \)) horizon. The initial \( \alpha \)-vectors \( \alpha^0 \) represent the terminal rewards and are determined by the reward structure of the POMDP.

### 2.3.5 Belief space compression

Although the dimensionality of the belief space is very high in large problems, the belief points the agent most often encounters have much fewer degrees of freedom (Roy et al., 2005). This implies that the belief points in \( B \) most probably will be restricted to a subspace of the original belief simplex. Considering this, it is natural
to apply dimension reduction techniques to transform the belief points in $\mathcal{B}$ to some lower dimensional subspace.

To run a value iteration algorithm entirely in the subspace, one must also transform the POMDP model, including state transition matrices, observation functions, and the reward function, into this subspace. The value-directed compression (VDC) algorithm in (Poupart and Boutilier 2002) gives such a transform, by computing a low-dimensional representation of the POMDP model in the Krylov subspace (Golub and Loan 1989). A method is proposed in (Roy et al. 2005) to transform the POMDP into a nonlinear subspace using the exponential family principal component analysis (PCA) (Collins et al. 2002). The value function in the nonlinear subspace is no longer piecewise linear and convex, which could jeopardize the performance of resulting policy.

In the RBVI formulation presented here, the EM and VBEM algorithms could be performed in a lower-dimensional subspace and yet we do not require transforming the POMDP into the subspace. This is possible because the computation of $\alpha$ by EM or VBEM does not involve using the POMDP model. The low-dimensional $\alpha$ vectors produced by the EM and VBEM algorithms are easily projected back into the original belief space.

We perform the belief space compression by applying principle component analysis (PCA) (Duda and Hart 1973; Joliffe 1986) to transform the belief points in $\mathcal{B}$ into a lower-dimensional subspace. For a given set of belief points $\mathcal{B}$, we compress the belief points using the procedures as summarized in the following:

- Compute the covariance matrix, denoted by $C$, of the belief points in $\mathcal{B}$;
- Calculate the eigenvalues, denoted by $e = [e_1, e_2, \cdots, e_d]$, and the corresponding eigenvectors, denoted by $U = [u_1, u_2, \cdots, u_d]$, of the covariance matrix, where $d = |\mathcal{S}|$ is dimensionality of the original belief space and is equal to the number
of states.

- Choose the principal components and find the subspace. Let \( \Lambda = \{i : e_i \geq \max_i(e_i) \times 0.01\} \) and the belief subspace is spanned by \( \{u_i : i \in \Lambda\} \);

- Compress the belief dimension by using \( b' = U_{new}^Tb \), where \( U_{new} \) is a matrix with columns \( \{u_i : i \in \Lambda\} \);

The PCA transforms \( B \) into \( B' = \{b' : b' = U_{new}^Tb, b \in B\} \). The \( B' \) along with the associated values \( \mathcal{V} \) are fed into the estimators \( \Upsilon \) to obtain the lower-dimensional \( \alpha \)-vectors \( \alpha' \), which are then project back to get the \( \alpha \)-vectors in the original belief space, \( \alpha = U_{new} \star \alpha' \).

### 2.3.6 The complete RBVI algorithm

We now give the complete procedures of the proposed RBVI algorithm. Since we have two methods to estimate the \( \alpha \)-vectors and ellipsoidal parameters, the algorithm has two versions: RBVI-EM, RBVI-VBEM. The procedures of the RBVI-EM are summarized in Table 2.1, where \( \text{std}(\mathcal{V}) \) denotes standard deviation of the elements in \( \mathcal{V} \). For the RBVI-VBEM, we simply modify Steps 2.2 and 2.3, replacing (E-Step, M-Step) with (VB-E-Step, VB-M-Step, estimation of \( \omega \)). To execute the policy, we use the estimated \( \alpha \)-vectors in (2.5) to find the optimal action.

### 2.4 Experimental Results

We demonstrate the performance of the proposed RBVI on four benchmark problems. The first three, namely, Tiger-grid, Hallway, and Hallway2, were introduced in [Littman et al. (1995)] and have since been widely used to test scalable POMDP solutions. The fourth problem is the Tag problem introduced in [Pineau et al. (2003)], which is relatively new and has a larger problem size.
Table 2.1: The Proposed RBVI Algorithm with EM as a Parameter Estimator

**Input:** POMDP model \( (\mathcal{S}, \mathcal{A}, T, \mathcal{O}, \Omega, R) \), \( K \), initial \( \alpha \)-vectors \( \alpha^0 \), convergence criterion for EM, convergence criterion for the value function or maximum length of horizon \( T \);

**Output:** the \( \alpha \)-vectors \( \alpha^n \)

1. Obtain a set of belief points \( \mathcal{B} = \{ b_1, b_2, \cdots, b_|\mathcal{B}| \} \), as described in Section 2.3.4
2. Repeat until \( n = T \) or \( \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} V^n(b_i) \) converges in \( n \):
   1. Compute \( V^n(b) \) for original belief points, \( \forall b \in \mathcal{B} \), using (2.5);
   2. Initialize \( \{ \omega, c, D, \alpha \} \); let \( \mu_k = \text{std}(V) \), \( k = 1, 2, \ldots, K \).
3. Repeat until the logarithm of the likelihood function in (2.11) converges:
   1. E-Step: compute \( \delta_k^i \) using (2.15) along with (2.7) and (2.8).
   2. M-Step: compute the updated parameters \( \{ \hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha} \} \) using (2.18), (2.19), (2.20);
   3. Record \( \alpha^n = U_{\text{new}}\alpha \); let \( n = n + 1 \).

Since our algorithms are tested based on these benchmark problems, we first give a brief description of them. The four problems are illustrated in Figure 2.3. The first three problems, TigerGrid, Hallway and Hallway2, are navigation problems. In each one of these problems, a robotic agent navigates in a grid-world environment and tries to reach the goal (denoted by a “star” sign) within a minimum number of steps. Each grid-world environment has a particular configuration of rooms and each of the four orientations in a room is represented by a state, which is labeled by a unique number in Figure 2.3. The agent can take five different actions \{Stay, Walk Forward, Turn Right and Turn Left\}. The problems consists of 17 basic observations, each corresponding to one of 16 different combinations of the presence/absence of a wall in each of 4 directions, plus a “star”. The environments each has its own particulars. In TigerGrid, except there is a reward for reaching the goal, there are
two traps denoted by “-1”. If the agent falls into one of the traps, it will obtain a penalty. And unlike the other environments, in which the agent starts uniformly from any states, the agent in TigerGrid problem only starts with equal probability facing north in one or the other of the two rooms marked with a robot symbol in Figure 2.3. In the problem Hallway, except for the 17 basic observations, there are also three landmarks which are visible only when the agent faces south. The transitions and observations are extremely noisy. The three models can be downloaded from the website: http://www.cs.brown.edu/research/ai/pomdp/examples/index.html.

The TagAvoid problem is different from the first three problems. In TagAvoid problem, the goal of the agent is to search or tag a moving opponent. The moving opponent stochastically moves according to a fixed policy which is pre-defined by the model. The state space of TagAvoid is the cross product of two features: Robot(marked with R)=\{s_1, s_2, \ldots, s_{29}\}; Opponent(marked with O)=\{s_1, s_2, \ldots, s_{29}, s_{tagged}\}. The five different actions the robot can select are \{Move North, Move South, Move East, Move West, and Tag\}. The robot will obtain -1 reward for each move; the tag action results in +10 reward if the robot and opponent are in the same location; otherwise -10 reward is imposed. The position of the opponent is completely unobservable unless both agents are in the same cell. At each time-step, the opponent (with omniscient knowledge) moves away from the robot with the probability of 0.8 and stays in the original place with probability of 0.2. This model can be downloaded from the website: http://www.science.uva.nl/mtjspaann/pomdp. The numerical values of |S|, |A|, |O| of these problems are also summarized in Table 2.2.

The proposed RBVI is compared to six state-of-the-art POMDP algorithms: Grid \cite{brafman1997}, PBUA \cite{poon2001}, PBVI \cite{pineau2003}, BPI \cite{poupart2003}, Perseus \cite{spaann2004}, HSVI \cite{smith2004, smith2005}, where BPI is based on policy iteration while the others are point-based meth-
ods for value iteration. The comparison is made in terms of performance, policy computation time, and $|\Gamma|$ the number of $\alpha$-vectors upon convergence of the optimal value function. To replicate the experiments of previous authors, we measure the algorithms’ performance by the discounted-accumulative-reward averaged over $N_{test}$ independent tests. For Tiger-Grid, $N_{test} = 151$ and each test terminates after the agent takes 500 actions; during a test the agent is reset each time it reaches the goal. For Hallway and Hallway2, $N_{test} = 251$ and each test terminates when the agent reaches the goal or has taken a maximum of 251 actions. For Tag, $N_{test} = 1000$ and each test terminates when the agent successfully tags its component or has taken a maximum of 100 actions.

Figure 2.3: Illustration of Benchmark problems.
Table 2.2: Results on the benchmark problems, where $T$ denotes time in seconds, $|\Gamma|=\text{number of } \alpha\text{-vectors}$, and $\text{n.v.}=\text{not available}$. The results marked with * are those we obtained by coding the respective algorithms in Matlab; other results may have been coded in languages other than Matlab and executed on computer platforms different from ours. (To be continued on Table 2.3)

| Method          | Reward | $T$(s) | $|\Gamma|$ |
|-----------------|--------|--------|-----------|
| **Tiger-Grid**  | $|S|=33, |A|=5, |O|=17$ |
| GridBraffman    | 0.94   | n.v.   | 174       |
| PBUAPoon        | 2.30   | 12116  | 660       |
| PBVIPineau et al. (2003) | 2.25   | 3448   | 470       |
| PBVI (*)        | 2.23   | 2239   | 970       |
| BPIPoupart and Boutilier (2003) | 2.22   | 1000   | 120       |
| PerseusSpaan and Vlassis (2004) | 2.34   | 104    | 134       |
| HSVI Smith and Simmons (2004) | 2.35   | 10341  | 4860      |
| HSVI2Smith and Simmons (2005) | 2.30   | 52     | 1003      |
| RBVI-EM (*)     | 1.95   | 135    | 10        |
| RBVI-VBEM (*)   | 2.05   | 64     | 10        |
| **Hallway**     | $|S|=57, |A|=5, |O|=21$ |
| PBUAPoon        | 0.53   | 450    | 300       |
| PBVI [Pineau et al. (2003)] | 0.53   | 288    | 86        |
| PBVI (*)        | 0.54   | 1166   | 408       |
| BPI [Poupart and Boutilier (2003)] | 0.51   | 185    | 43        |
| PerseusSpaan and Vlassis (2004) | 0.51   | 35     | 55        |
| HSVI [Smith and Simmons (2004)] | 0.52   | 10836  | 1341      |
| HSVI2 [Smith and Simmons (2005)] | 0.52 | 2.4    | 147       |
| RBVI-EM (*)     | 0.54   | 88     | 10        |
| RBVI-VBEM (*)   | 0.54   | 84     | 10        |

For the RBVI, the EM is initialized as follows: $D = \{D_k = I_d\}_{k=1}^K$ where $I_d$ is a $d \times d$ identity matrix, $c = \{c_k\}_{k=1}^K \subset B$ are $K$ belief samples that are most separated from each other, and $\alpha = \{\alpha_k = \alpha_{LS}\}$ where $\alpha_{LS}$ is the least-squares (LS) estimate using $\{b, V(b)\}_{b \in B}$. The prior distribution in VBEM is constructed such that the resulting means of the parameters equal to the corresponding initial parameters in EM.

The experimental results of the proposed RBVI algorithm are summarized in Table 2.2 in comparison to the other six algorithms. It is demonstrated that for
Table 2.3: Results on the benchmark problems (Continued from Table 2.2)

| Method                  | Reward | \(T(s)\) | \(|\Gamma|\) |
|-------------------------|--------|-----------|-----------|
| **Hallway2**            |        |           |           |
| PBUA Poon (2001)        | 0.35   | 27898     | 1840      |
| PBVI Pineau et al. (2003) | 0.34   | 360       | 95        |
| PBVI (*)                | 0.35   | 2345      | 572       |
| BPI Poupart and Boutilier (2003) | 0.32   | 790       | 60        |
| Perseus Spaan and Vlassis (2004) | 0.35   | 10        | 56        |
| HSVI1 Smith and Simmons (2004) | 0.35   | 10010     | 1571      |
| HSVI2 Smith and Simmons (2005) | 0.35   | 1.5       | 114       |
| RBVI-EM (*)             | 0.30   | 90        | 15        |
| RBVI-VBEM (*)           | 0.31   | 103       | 15        |

<table>
<thead>
<tr>
<th>Tag</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PBUA Pineau et al. (2003)</td>
<td>-9.180</td>
<td>180880</td>
<td>1334</td>
</tr>
<tr>
<td>BPI Poupart and Boutilier (2003)</td>
<td>-6.65</td>
<td>250</td>
<td>17</td>
</tr>
<tr>
<td>Perseus Spaan and Vlassis (2004)</td>
<td>-6.17</td>
<td>1670</td>
<td>280</td>
</tr>
<tr>
<td>HSVI1 Smith and Simmons (2004)</td>
<td>-6.37</td>
<td>10113</td>
<td>1657</td>
</tr>
<tr>
<td>HSVI2 Smith and Simmons (2005)</td>
<td>-6.36</td>
<td>24</td>
<td>415</td>
</tr>
<tr>
<td>RBVI-EM (*)</td>
<td>-6.56</td>
<td>2481</td>
<td>15</td>
</tr>
<tr>
<td>RBVI-VBEM (*)</td>
<td>-6.34</td>
<td>2430</td>
<td>15</td>
</tr>
</tbody>
</table>

Each of the four problems the RBVI achieves performance competitive to the other algorithms and yet uses a significantly smaller number of \(\alpha\)-vectors. Moreover the RBVI is also one of the best algorithms in terms of the running time spent in policy computation. Since the running time is heavily dependent on computer platforms, we will also perform a more accurate time comparison in Section 2.5 using the Big O notation.

To examine why such a small number of \(\alpha\)-vectors is sufficient to represent the value function, we show in Figure 2.4 the model hyper-parameters \(\omega = \{\omega_k\}_{k=1}^K\) estimated by VBEM. It is seen that the \(\omega_3\) in Hallway, the \(\omega_{11}\) in Hallway2 and the \(\omega_{10}\) in TagAvoid are approximately zero, each indicating a spurious region and the associated \(\alpha\)-vector. This implies that \(K = 10\) for Hallway and \(K = 15\) for Hallway2 and TagAvoid is more than sufficient to represent the value function on the belief
samples. The $\omega$ for Tiger-Grid does not seem to show any spurious $\alpha$-vectors. This may partly explain why the performance of RBVI degrades a little on Tiger-Grid.

### 2.5 Time Comparison in Big O Notation

It is known that PBVI has a time complexity of $O(|B||S||\Gamma_{PBVI}|.|A||O|)$ (Pineau et al. 2003) for a single iteration, where $\Gamma_{PBVI}$ denotes the set of $\alpha$-vectors backed up by PBVI in the previous iteration. As PBVI backs up an $\alpha$-vector for each point in $B$, the complexity is equal to $O(|B|^2|S||A||O|)$. Clearly the size of $B$ greatly affects the efficiency of PBVI. In this chapter, we introduce a *region-based value iteration* (RBVI) algorithm, which backs up a single $\alpha$-vector for each of the convex regions.

---

Figure 2.4: The $\omega_k = p(z = k)$ estimated by VBEM.
over which the optimal value function is linear. The RBVI is a two-stage processing: first it computes \( V^n(b) \) \( \forall b \in B \) from \( \alpha^{n-1} \) using (2.5); second, it uses an estimator \( \Upsilon \) to estimate the region-based \( \alpha^n \) from the samples \( \{b, V^n(b)\}_{b \in B} \). Computation of \( \{V^n(b)\}_{b \in B} \) has a time complexity of \( O(|B||S||\Gamma_{RBVI}||A||O|) \), where \( \Gamma_{RBVI} \) denotes the set of \( \alpha \)-vectors backed up by RBVI in the previous iteration; \( \Upsilon \) has a complexity denoted by \( O(\Upsilon) \). Clearly, \( |\Gamma_{RBVI}| < |\Gamma_{PBVI}| \), hence the first stage of RBVI is always more efficient than PBVI. Therefore any \( \Upsilon \) satisfying \( O(\Upsilon) < O(|B||S||\Gamma_{PBVI}||A||O|) \) will yield a RBVI that computes more efficiently than PBVI.

As implied by (2.4), the belief simplex is partitioned into a finite number of polyhedral regions, such that \( V^n(b) \) is linear in \( b \) in each region. In the proposed RBVI algorithm we approximate this polyhedral partition with an ellipsoidal partition, in which each polyhedral region is approximated by one or more ellipsoidal regions and a single \( \alpha \)-vector is estimated and backed up for each ellipsoidal region. We derive efficient estimators \( \Upsilon \) to jointly estimate the \( \alpha \)-vectors as well as the position and shape parameters of each ellipsoidal region, using the samples \( \{b, V^n(b)\}_{b \in B} \). For completeness and comparison, we give two versions of \( \Upsilon \), based on expectation-maximization (EM) and variational Bayesian EM (VBEM). We show in Section 2.5 that the time complexity of either EM or VBEM is no worse than \( O(|B|\dim(U^Tb)^2|\Gamma_{RBVI}|N_{\Upsilon}) \), where \( N_{\Upsilon} \) is the number of EM or VBEM iterations, and \( \dim(U^Tb) \leq |S| \), with \( \dim \) denoting the dimensionality and \( U \) a matrix of full column rank.

At a given iteration of the optimal value function, the proposed RBVI has a two-stage processing: first, compute \( V(b) \) \( \forall b \in B \) from (2.5) using the \( \alpha \)-vectors estimated in the previous iteration; second, estimate the new \( \alpha \)-vectors for the present iteration. The time complexity of the RBVI therefore consists of two terms, corresponding to the two stages, as summarized in Table 3.3, where the PBVI time complexity \( \text{Pineau et al. (2003)} \) is also shown as a comparison.
Table 2.4: Time Comparison in Big O Notation, where \( \dim(\tilde{b}) \) denotes the dimensionality of \( \tilde{b} = U^T b \) and \( U \) is a matrix whose orthonormal columns span the space of \( \tilde{b} \), and \( N_T \) denotes the number of EM or VBEM iterations.

<table>
<thead>
<tr>
<th>Computation involving Eqn. (2.5)</th>
<th>EM or VBEM estimator (( \Upsilon_{\text{EM}} ) or ( \Upsilon_{\text{VBEM}} ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBVI ( O(</td>
<td>B</td>
</tr>
<tr>
<td>Proposed RBVI ( O(</td>
<td>B</td>
</tr>
</tbody>
</table>

In Table 2.4, we have shown the general case when we perform a linear transform \( \tilde{b} = U^T b \) (\( U \) is computed only once) and obtain \( \tilde{b} \) as a projection of \( b \) on \( \text{span}(U) \), the subspace spanned by the orthonormal columns of \( U \). Usually \( U \) is an identity matrix. However, it is possible \( \dim(\tilde{b}) = \dim(U^T b) \ll \dim(b) = |S| \), and replacing \( b \) with \( \tilde{b} \) can result in great reduction in the EM (VBEM) computations. This happens, for example, when the belief samples in \( B \) are sparse (which is the case in the Tag problem). To illustrate the reduction of belief space dimensionality, we plot the eigenvalues of the covariance matrix of belief points for the four benchmark problems in Figure 2.5. It is seen that for TigerGrid and Hallway2 problems, all eigenvalues are very large except one. Since this reduction is due to that the belief vector \( b \) has its elements sum to one and is rank deficient by one, there is effectively no dimensionality reduction. But for the problem TagAvoid, the degree of freedom of the reachable belief points is far less than the original dimensionality, therefore a great dimensionality reduction is obtained.

The most intensive computations are performed in equations (2.8), (2.19), and (2.20) for EM, and (2.29), (2.35), and (2.36) for VBEM. These equations all involve the belief samples in \( B \) which are fixed throughout the RBVI iterations, hence many computations can be computed beforehand only once. By avoiding redundant computations in these equations, it is possible to further bring down the EM/VBEM
computational complexity. The time complexity presented in Table 2.4 represents the case when we perform straightforward computation of these equations.

Since PBVI backs up an \( \alpha \)-vector \( \forall b \in \mathcal{B} \), \( |\Gamma_{PBVI}| \approx |\mathcal{B}| \), whereas \( |\Gamma_{RBVI}| \ll |\mathcal{B}| \) for RBVI. Therefore, the first stage of the RBVI is always computed more efficiently than PBVI, regardless of \( |S| \), \( |\mathcal{A}| \), and \( |\mathcal{O}| \). The time complexity of the second stage in RBVI has a squared dependence on \( \dim(\tilde{b}) = \dim(U^T b) \) and it is independent of \( |\mathcal{A}| \) and \( |\mathcal{O}| \). The EM and VBEM estimators in RBVI typically converges very fast, resulting in a small \( N_{\Upsilon} \). Therefore, the second stage of RBVI can be computed more efficiently than PBVI when: either (i) \( \frac{|S|}{\max(|\mathcal{A}|, |\mathcal{O}|)} \) is not too large, as in Tiger-Grid, Hallway, and Hallway2; or (ii) \( \dim(\tilde{b}) = \dim(U^T b) \ll \dim(b) = |S| \) as in Tag where.
2.6 Conclusions on RBVI

We have proposed an approximate region-based value iteration (RBVI) algorithm for computing the POMDP policy. By approximating the true polyhedral piecewise linearity of the optimal value function with an ellipsoidal piecewise linearity, we are able to find a small number of region-based $\alpha$-vectors efficiently. The compact representation of the value function brings significant computational savings, particularly when the number of states is comparable to the product of the number of actions and the number of observations, or when the belief samples exhibit some special structures like sparseness. The results on benchmark problems show the the proposed RBVI can produce performance that is competitive to state-of-the-art POMDP algorithms, by using a significantly smaller number of $\alpha$-vectors. The RBVI time complexity shown in the chapter is based on straightforward computation of the related equations. There may still exist fast algorithms to avoid redundant computations, and the complexity order of the RBVI can be further reduced.

2.7 Appendix: Parameter derivation of EM and VBEM

Some matrix identities often used in derivations in this chapter:

$\mathbf{A}, \mathbf{B}, \mathbf{C}$ are matrices and $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are vectors which are not dependent on $\mathbf{X}$

- \[ \mathbf{a}^T \mathbf{X} \mathbf{b} = \text{tr}(\mathbf{b} \mathbf{a}^T \mathbf{X}) = \text{tr}(\mathbf{X} \mathbf{b} \mathbf{a}^T) \]

- \[ \frac{\partial \ln |\mathbf{X}|}{\partial \mathbf{X}} = (\mathbf{X}^{-1})^T \]

- \[ \frac{\partial \text{tr}(\mathbf{X}^{-1} \mathbf{A})}{\partial \mathbf{X}} = \frac{\partial \text{tr}(\mathbf{A} \mathbf{X}^{-1})}{\partial \mathbf{X}} = -\mathbf{X}^{-1} \mathbf{A}^T \mathbf{X}^{-1} \]
\[ \frac{\partial \text{tr}(X^TAX)}{\partial X} = \frac{\partial \text{tr}(AXX^T)}{\partial X} = \frac{\partial \text{tr}(XX^TA)}{\partial X} = (A + A^T)X \]

### 2.7.1 Parameter derivation of the Expectation-Maximization (EM)

Substituting (2.7) and (2.8) into (2.10), we have

\[
Q(\hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha} | \omega, c, D, \alpha) = \sum_{i=1}^{\lvert B \rvert} \sum_{k=1}^{K} \delta_k^i \ln \left[ \omega_k p(b_i | z_i = k, \hat{c}_k, \hat{D}_k) p(V_i | b_i, z_i = k, \hat{\alpha}_k) \right] \\
= \sum_{i=1}^{\lvert B \rvert} \sum_{k=1}^{K} \delta_k^i \ln \omega_k + \sum_{i=1}^{\lvert B \rvert} \sum_{k=1}^{K} \delta_k^i \left( -\frac{1}{2} \ln(2\pi \mu_k^2) + \left\{ -(V(b) - \alpha_k^T b)^2 / (2\mu_k^2) \right\} \right) \\
+ \sum_{i=1}^{\lvert B \rvert} \sum_{k=1}^{K} \delta_k^i \left( -\frac{1}{2} \left( \ln |D_k| + d \ln(2\pi) \right) + \text{tr}\left\{ -(b - c_k)(b - c_k)^T D_k^{-1} \right\} \right) 
\]

(2.44)

In the M-Step, adding the Lagrange multiplier \( \lambda \), using the constraints in (2.12), and setting the derivative of \( Q(\hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha} | \omega, c, D, \alpha) \) equal to zero, we have

\[
\frac{\partial}{\partial \hat{\omega}_k} \left( Q(\hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha} | \omega, c, D, \alpha) - \lambda \left( \sum_{k=1}^{K} \hat{\omega}_k - 1 \right) \right) = 0 \\
\frac{\partial}{\partial \hat{c}_k} \left( Q(\hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha} | \omega, c, D, \alpha) - \lambda \left( \sum_{k=1}^{K} \hat{\omega}_k - 1 \right) \right) = 0 \\
\frac{\partial}{\partial \hat{D}_k} \left( Q(\hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha} | \omega, c, D, \alpha) - \lambda \left( \sum_{k=1}^{K} \hat{\omega}_k - 1 \right) \right) = 0 \\
\frac{\partial}{\partial \hat{\alpha}_k} \left( Q(\hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha} | \omega, c, D, \alpha) - \lambda \left( \sum_{k=1}^{K} \hat{\omega}_k - 1 \right) \right) = 0 
\]

(2.45)

Noting the parameters \{\( \hat{\omega}, \hat{\alpha}, (\hat{c}, \hat{D}) \}\) are decoupled in (2.44), the derivatives with respect to \{\( \hat{\omega}, \hat{\alpha}, \hat{c}, \hat{D} \)\} is easily computed,

\[
\frac{\partial}{\partial \hat{\omega}_k} \left( Q(\hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha} | \omega, c, D, \alpha) - \lambda \left( \sum_{k=1}^{K} \hat{\omega}_k - 1 \right) \right) 
\]
\[ \frac{\partial}{\partial \hat{\omega}_k} \left( \sum_{i=1}^{[B]} \delta_i^k \ln \hat{\omega}_k - \lambda \left( \sum_{k=1}^{K} \hat{\omega}_k - 1 \right) \right) = \sum_{i=1}^{[B]} \frac{\delta_i^k}{\hat{\omega}_k} - \lambda = 0 \]

\[ \hat{\omega}_k = \frac{\sum_{i=1}^{[B]} \delta_i^k}{\lambda} \text{, using the constraints in (2.12), we have} \]

\[ \hat{\omega}_k = \frac{\sum_{i=1}^{[B]} \delta_i^k}{|B|} \quad (2.46) \]

\[ \frac{\partial}{\partial \hat{\alpha}_k} \left( Q(\hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha} | \omega, c, D, \alpha) - \lambda \left( \sum_{k=1}^{K} \hat{\omega}_k - 1 \right) \right) = \frac{\partial}{\partial \hat{\alpha}_k} \left( \sum_{i=1}^{[B]} \sum_{k=1}^{K} \delta_i^k \left( - \frac{1}{2} \ln(2\pi\mu_k^2) + \{-(V_i - \alpha_k^T b_i)^2/(2\mu_k^2)\} \right) \right) \]

\[ = \frac{\partial}{\partial \hat{\alpha}_k} \left( \sum_{i=1}^{[B]} \delta_i^k (V_i - \alpha_k^T b_i)^2/(2\mu_k^2) \right) \]

\[ = \sum_{i=1}^{[B]} \delta_i^k b_i b_i^T \hat{\alpha}_k - \sum_{i=1}^{[B]} \delta_i^k V_i b_i = 0 \quad \text{then we have} \]

\[ \hat{\alpha}_k = \left[ \sum_{i=1}^{[B]} \delta_i^k b_i b_i^T \right]^{-1} \sum_{i=1}^{[B]} \delta_i^k V_i b_i \quad (2.47) \]

\[ \frac{\partial}{\partial \hat{c}_k} \left( Q(\hat{\omega}, \hat{c}, \hat{D}, \hat{\alpha} | \omega, c, D, \alpha) - \lambda \left( \sum_{k=1}^{K} \hat{\omega}_k - 1 \right) \right) = \frac{\partial}{\partial \hat{c}_k} \left( \sum_{i=1}^{[B]} \delta_i^k \left( \frac{1}{2} \ln |D_k| + d \ln(2\pi) + \text{tr}\{-(b_i - c_k)(b_i - c_k)^T D_k^{-1}\} \right) \right) \]

\[ = \frac{\partial}{\partial \hat{c}_k} \left( \sum_{i=1}^{[B]} \delta_i^k \left( \text{tr}\{-(b_i - c_k)(b_i - c_k)^T D_k^{-1}\} \right) \right) \]

\[ = (D_k^{-1} + (D_k^{-1})^T) \sum_{i=1}^{[B]} (b_i - \hat{c}_k) \delta_i^k = 0 \]

47
\[ \hat{c}_k = \frac{\sum_{i=1}^{|[\mathcal{B}]|} \delta_k^i b_i}{\sum_{i=1}^{|[\mathcal{B}]|} \delta_k^i} \]  

(2.48)

\[ \frac{\partial}{\partial D_k} \left( Q(\hat{\omega}, \hat{c}, \hat{\alpha} | \omega, c, D, \alpha) - \lambda \left( \sum_{k=1}^{K} \hat{\omega}_k - 1 \right) \right) = \frac{\partial}{\partial \hat{D}_k} \left( \sum_{i=1}^{|[\mathcal{B}]|} \delta_k^i \left( - \frac{1}{2} \left( \ln |D_k| + d \ln(2\pi) \right) + \text{tr} \left\{ - \frac{1}{2} (b_i - c_k)(b_i - c_k)^T D_k^{-1} \right\} \right) \right) \]

\[ = - \frac{1}{2} (D_k^{-1})^T \sum_{i=1}^{|[\mathcal{B}]|} \delta_k^i + \frac{1}{2} D_k^{-1} \left( \sum_{i=1}^{|[\mathcal{B}]|} \delta_k^i (b_i - c_k)(b_i - c_k)^T D_k^{-1} \right) = 0 \]

since the covariance matrix \( D_k^{-1} \) is symmetric, then

\[ \hat{D}_k = \frac{\sum_{i=1}^{|[\mathcal{B}]|} \delta_k^i (b_i - \hat{c}_k)(b_i - \hat{c}_k)^T}{\sum_{i=1}^{|[\mathcal{B}]|} \delta_k^i} \]

(2.49)

### 2.7.2 Parameter derivation of the variational Bayesian EM (VBEM)

**VB-M Step** Substituting (2.9), (2.25), (2.26), (2.27), (2.28) into (2.24) and assuming the pairs \( (V_i, b_i), i = 1 \cdots, |\mathcal{B}| \) are independent, we have

\[ q(c, D, \alpha) \propto p(c, D, \alpha) \prod_{i=1}^{|[\mathcal{B}]|} \exp \left\{ - \frac{1}{2} \ln |D_k| - \frac{1}{2} \text{tr} \left( D_k^{-1} (b_i - c_k)(b_i - c_k)^T \right) \right\} \]

\[ - \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln 2\pi \mu_k^2 - \frac{1}{2} (V(b_i) - \alpha_k^T b_i)^2 \bigg|_{z_i \equiv k} \]

\[ = p(c | D)p(D)p(\alpha) \prod_{i=1}^{|[\mathcal{B}]|} \exp \left\{ \sum_{k=1}^{K} \delta_k^i \left( - \frac{1}{2} \ln |D_k| - \frac{1}{2} (b_i - c_k)^T D_k^{-1} (b_i - c_k) \right) - \frac{V_i^2}{2\mu_k^2} - \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln 2\pi \mu_k^2 \right\} \]
\[ \alpha \propto p(c|D)p(D)p(\alpha) \prod_{i=1}^{B} \prod_{k=1}^{K} |D_k^{-1}|^{\tilde{\delta}_i/2} \exp \left( -\frac{1}{2} \tilde{\delta}_i (b_i - c_k)^T D_k^{-1} (b_i - c_k) \right) \times \exp \left( -\frac{1}{2\mu_k^2} \tilde{\delta}_i (\alpha_k^T b_i b_i^T \alpha_k - 2\alpha_k^T b_i V_i) \right) \]

\[ \propto \prod_k \exp \left( -\frac{1}{2} \text{tr} \left\{ (c_k - m_k^0)(c_k - m_k^0)^T \beta_k^{0,-1} D_k^{-1} \right\} \right) |D_k^{-1}|^{(n_k^0 - d - 1)/2} \]

\[ \times \exp \left( -\frac{1}{2} \text{tr} \left\{ S_k^{0,-1} D_k^{-1} \right\} \right) \exp \left( -\frac{1}{2} \sum_{i=1}^{|B|} \tilde{\delta}_i \frac{1}{\mu_k} \left( \alpha_k^T b_i b_i^T \alpha_k - 2\alpha_k^T b_i V_i \right) \right) \]

\[ = \prod_{k=1}^{K} |D_k^{-1}|^{(n_k^0 - d - 1)/2 + \frac{1}{2} \sum_{i=1}^{|B|} \tilde{\delta}_i} \exp \left( -\frac{1}{2} \text{tr} \left\{ S_k^{0,-1} D_k^{-1} \right\} \right) \]

\[ \times \exp \left( -\frac{1}{2} \sum_{i=1}^{|B|} \tilde{\delta}_i (b_i - c_k)^T D_k^{-1} (b_i - c_k) - \frac{1}{2} \text{tr} \left\{ (c_k - m_k^0)(c_k - m_k^0)^T \beta_k^{0,-1} D_k^{-1} \right\} \right) \]

\[ \times \exp \left( -\frac{1}{2} \sum_{i=1}^{|B|} \tilde{\delta}_i \frac{1}{\mu_k} \left( \alpha_k^T b_i b_i^T \alpha_k - 2\alpha_k^T b_i V_i \right) - \frac{1}{2} (\alpha_k - \eta_k^0)^T \Sigma_k^{0,-1} (\alpha_k - \eta_k^0) \right) \]

\[ \propto \prod_{k=1}^{K} \exp \left( -\frac{1}{2} \text{tr} \left\{ S_k^{-1,0} D_k^{-1} \right\} \right) \exp \left( -\frac{1}{2} \sum_{i=1}^{|B|} \delta_i (b_i - c_k)^T D_k^{-1} (b_i - c_k) \right) \]

\[ \times \exp \left( -\frac{1}{2} \sum_{i=1}^{|B|} \delta_i \frac{1}{\mu_k} \left( \alpha_k^T b_i b_i^T \alpha_k - 2\alpha_k^T b_i V_i \right) \right) \exp \left( -\frac{1}{2} (\alpha_k - \eta_k^0)^T \Sigma_k^{-1,0} (\alpha_k - \eta_k^0) \right) \]

\[ \times \exp \left( -\frac{1}{2} \text{tr} \left\{ (c_k - m_k^0)(c_k - m_k^0)^T \beta_k^{0,0} D_k^{-1} \right\} \right) |D_k^{-1}|^{\frac{1}{2}(n_k^0 + \sum_{i=1}^{|B|} \delta_i - d - 1)} \quad (2.50) \]

From (2.50), equations (2.33), (2.34), (2.35), (2.36) and (2.37) are obtained.
Chapter 3

Approximate Policy Iteration algorithms for Partially Observable Markov Decision Processes

3.1 Introduction to Policy Iteration for POMDPs

As we have discussed in Chapter [2], the value iteration method solves for the finite-horizon policy of a POMDP. To obtain the infinite-horizon policy, one must solve the policy over successively larger horizons until the value function converges to that of the infinite-horizon. When using value iteration to find the optimal policy for the POMDP, the value function is updated using dynamic programming, computing the optimal value function for the length-\( n \) horizon using the value function of the length-(\( n - 1 \)) horizon. The errors between the true optimal value function and the approximate value function can be carried from one iteration to the next. Thus the errors can accumulate as the iteration proceeds, which, in the worst case, would result in serious degradation in the value function approximation when the horizon length is large. Since the policy we find using the value iteration is often sensitive to the estimated value function, a small change in the estimated value may result in a great change in the policy. An alternative approach is to solve the infinite-horizon policy directly using policy iteration.

Before we go into the details of the policy iteration for POMDPs, here we quickly give a review of the policy iteration for MDPs. Besides the value function \( V(s) \) defined in (1.2) for a given policy \( \pi \), here we consider another function \( Q(s, a) \), an action-value
function (Sutton and Barto 1998), which is the expected future reward accumulated starting from the state $s$, taking action $a$, and following a given deterministic policy $\pi$ thereafter

$$Q^\pi(s, a) = E_\pi\{R_t|s_t = s, a_t = a\} = E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\}$$

$$= R(s, a) + \gamma \sum_{s'} T_{ss'}Q^\pi(s', \pi(s'))$$

$$= R(s, a) + \gamma \sum_{s'} T_{ss'}V^\pi(s')$$ \hspace{1cm} (3.1)

**Policy iteration** for MDPs (Howard 1960) is a method for finding the optimal policy by generating a sequence of successively improved policies. Each iteration $n$ consists of two phases: *policy evaluation* and *policy improvement*. Policy evaluation computes the action-value function $Q^{\pi_n}(s, a)$ of the current policy $\pi_n$ and policy improvement finds a new greedy policy

$$\pi_{n+1}(s) = \arg \max_{a \in A} Q^{\pi_n}(s, a) \hspace{1cm} (3.2)$$

which is guaranteed to be no worse than $\pi$, as shown by the policy improvement theorem (Howard 1960; Sutton and Barto 1998).

**Theorem 1. (Policy improvement for MDPs)** Let $\pi(s)$ and $\pi'(s)$ be a pair of deterministic policies and the policy $\pi'(s)$ is found according to (3.2) for all $s \in \mathcal{S}$, then the policy $\pi'$ must as good as, or better than $\pi$, i.e., it must obtain greater or equal expected reward from all states $s \in \mathcal{S}$,

$$V^{\pi'}(s) \geq V^\pi(s) \hspace{1cm} (3.3)$$

By (Smallwood and Sondik 1973), the POMDP can be treated as a continuous MDP in belief state $b$ and the policy is a mapping from $b$ to action, $a = \pi(b)$. Let
\( V^\pi(b) \) be the infinite-horizon value of \( b \) by following policy \( \pi \), then the value function satisfies the Bellman equation (Sondik 1978),

\[
V^\pi(b) = \sum_{s \in \mathcal{S}} b(s)R(s, \pi(b)) + \gamma \sum_{o \in \mathcal{O}} p(o|\pi(b))V^\pi(\tilde{b}^\pi_o(b)) \tag{3.4}
\]

where \( p(o|\pi(b)) \) is determined by (1.5) and \( \tilde{b}^\pi_o(b) \) is determined by (1.4).

For any stationary policy \( \pi \), we have the following policy improvement theorem for POMDPs (Blackwell 1965; Howard 1971; Sondik 1978).

**Theorem 2.** (Howard-Blackwell policy improvement for POMDPs) Let \( V^\pi(b) \) be the infinite-horizon value function of a stationary policy \( a = \pi(b) \). Define the \( Q \) function

\[
Q^\pi(b, a) = \sum_{s \in \mathcal{S}} b(s)R(s, a) + \gamma \sum_{o \in \mathcal{O}} p(o|b, \pi(b))V^\pi(\tilde{b}^a_o) \tag{3.5}
\]

where \( \tilde{b}^a_o \) is defined by (1.4), and the new policy

\[
\pi'(b) = \arg \max_a Q^\pi(b, a) \tag{3.6}
\]

Then \( \pi' \) is an improved policy over \( \pi \), i.e.,

\[
V^{\pi'}(b) \geq V^\pi(b) \tag{3.7}
\]

for any belief point (belief state) \( b \).

The two steps – policy evaluation and policy improvement are repeated until there is no change in policy in two consecutive iterations. When this happens, we can say that the policy has approximately often converged to the optimal policy. It is also found that the policy converges in a surprisingly small number of iterations. The block diagram of POMDP policy iteration is illustrated in Figure 3.1.
3.2 Overview of POMDP Algorithms Based on Policy Iteration

Although Theorem 2 provides the theoretical basis for POMDP policy iteration, there are two inherent difficulties in implementing the policy iteration procedures (Sondik 1978). First, unlike the MDP, which computes the $Q(s, a)$ by solving a finite number of Bellman equations, computation of $Q(b, a)$ is very expensive and even intractable, because the belief state $b$ is continuous and the value function $V^\pi(b)$ for an infinite horizon may not be expressed as a piecewise linear convex function of belief state $b$. Secondly, as the belief state $b$ is continuous, the requirement in the Theorem 2 that the maximization in (3.6) should be performed for every possible $b$ also poses great challenges to policy iteration.

Sondik (1978) simplifies the step of policy evaluation in POMDPs by limiting the choice of policies to be within a special class — the finite transient policies. A policy
\( \pi \) is deterministic and finite transient, if
\[
p^\pi(a|b) = \begin{cases} 
1; & \text{if } a = \pi(b) \\
0; & \text{otherwise}
\end{cases}
\] (3.8)
and there exists \( n < \infty \) such that the set of belief-states generated by starting from every belief-state and following \( \pi \) for \( n \) consecutive steps is disjoint with \( \{ b : \pi(b) \text{ is discontinuous at } b \} \). Sondik (1978) proves that if the optimal policy falls into the class of finite transient policies, the optimal value function \( V^\pi(b) \) is piecewise linear and convex in belief state \( b \). Sondik also shows that for those optimal policies which are not finitely transient, the optimal value function of POMDPs for infinite horizon can be arbitrary closely approximated by a piecewise linear and convex function. The step of policy evaluation is then reduced to finding the \( \alpha \)-vectors in each iteration by solving a finite number of linear equations
\[
\alpha_k(s) = R(s, a(k)) + \gamma \sum_{o \in O} \sum_{s' \in S} T_{ss'}^{a(k)} \Theta_{ss'}^{a(k)} \alpha_{\nu(k,o)}(s')
\] (3.9)
for \( k = 1, 2, \ldots, K \), where \( \alpha_k \) is as defined in Chapter 2 and represents the gradient of the \( k \)-th hyperplane; \( a(k) \) is the action taken in the \( k \)-th polyhedron in the belief simplex and \( \nu(k, o) \) is the index of successor \( \alpha \)-vector if \( o \) is observed. In the step of policy improvement (updating \( \nu(k, o) \)), Sondik represents each polyhedron by a set of linear inequalities, with each linear inequality corresponding to a possible linear boundary of the polyhedron (Smallwood and Sondik 1973; Sondik 1978), and performs policy improvement based on the representation of linear inequalities.

Hansen (1997) presents an improved algorithm for Sondik’s policy iteration. Hansen’s policy evaluation is essentially the same as that of Sondik, solving the \( \alpha \)-vectors using (3.9). Hansen interprets the representation of \( \alpha \)-vectors as a finite state controller or automaton. Each \( \alpha \)-vector defines a state of the automaton. In the policy improvement step, instead going back to the representation of linear inequalities,
Hansen directly uses the representation of $\alpha$-vectors to perform policy improvement. In particular, Hansen performs one-step value iteration and updates the automaton (mainly $\nu(k, o)$), by adding new dominant $\alpha$-vectors and removing the $\alpha$-vectors that are pointwise dominated by others. This integrates Sondik’s multiple policy representations into a single one and greatly reduces the computational complexity and makes it easier to implement. As in Sondik (1978), Hansen’s algorithm still focuses on the entire belief simplex and offers a guarantee of convergence of the value function over the whole belief simplex. The focus on the entire belief simplex, on the other hand, often results in the number of $|A||\Gamma||O|$ $\alpha$-vectors added in each iteration in the worst case, where $|A|$ is the number of actions, $|\Gamma|$ the size of the automaton in the last iteration and $|O|$ is the number of observations. The rapid increase in the size of the automaton makes Hansen’s algorithm very difficult to scale up to large problems.

Bounded policy iteration (BPI) proposed by Poupart and Boutilier (2003) tries to reduce the size of the automaton by introducing a stochastic transition between the states. In both Sondik and Hansen’s algorithms, $\nu(k, o)$ is deterministic, which means by making observation $o$, the previous state $k$ transits to a new state $k' = \nu(k, o)$ with probability one. A stochastic transition in the automaton means that the previous state $k$ transits to a new state $k'$ with probability $p(k'|k, o)$ after making observation $o$. In order to guarantee that each state transits to another one with probability one and each state has an associated action $a$, the number of states should be very large so as to closely approximate the optimal value function. A continuous belief point can be also treated as a state in the limiting case. The introduction of stochastic transitions greatly reduces the size of the automaton and relaxes the requirement of deterministic transitions between states. Poupart and Boutilier (2003) also proves that when an $\alpha$-vector (each corresponding to a state in the automaton) is jointly dominated by a group of $\alpha$-vector, there exists a pointwise-dominating convex combination of $\alpha$-vector.
in this group. This pointwise-dominating convex combination can be found by solving linear programs. When adding a new dominant $\alpha$-vector, an old $\alpha$-vector is removed which is either pointwise dominated by the new $\alpha$-vector or by a group of existing $\alpha$-vectors. By doing this, the size of the automaton is bounded throughout the policy iterations. This approach, though it avoids the problem that the automaton grows rapidly with iterations, brings new computations of solving a great number of linear programs.

3.3 The ILSPI Algorithm for POMDP

In this section we propose a new policy iteration algorithm for POMDPs. This algorithm, called incremental least squares policy iteration (ILSPI), concentrates only on the belief states that are actually reachable by the POMDP. As noted in Pineau et al. (2003), the reachable belief points of most practical POMDP models constitute only a subset of the belief simplex. It is therefore sufficient to only find optimal actions for the reachable belief points, instead of all points in the belief simplex. The ILSPI represents an extension (although not exact) of the least squares policy iteration (LSPI) for MDPs (Lagoudakis and Parr 2002).

Though both the algorithms of ILSPI and LSPI (Lagoudakis and Parr 2002) use the term “least squares”, there is a difference in their meanings. Our algorithm is based on Bellman residual minimization approximation, minimizing the squared Bellman error; and LSPI (Lagoudakis and Parr 2002) is based on least squares fixed point approximation. We will analyze the two different representations in detail in the Appendix of this chapter. Apart from the difference in error definitions, ILSPI extends the LSPI algorithm (Lagoudakis and Parr 2002) in two respects: (a) it solves the policy for a POMDP, including the MDP as a special case; (b) it determines the optimal basis functions to incrementally reduce the Bellman residual. As the ILSPI
is based on a finite set of belief points, it can be thought of as a point-based policy iteration algorithm.

As discussed earlier, a policy iteration algorithm iteratively applies the policy improvement theorem to obtain successively improved policies. Put more specifically here, policy iteration consists of two alternative steps — in policy evaluation, we compute the value function $V^\pi(b)$ by solving the Bellman equation (3.4); in policy improvement, we improve the policy according to Theorem 2. The following subsection describe how the two steps are implemented in the ILSPI, based on a finite set of belief points reachable by the POMDP.

### 3.3.1 Policy improvement in reachable belief states

Let $\mathcal{B}_0$ be a set of initial belief points (states) at time $t = 0$. For time $t = 1, 2, \cdots$, let $\mathcal{B}_t$ be the set of all possible $\tilde{b}_o^a$ in (1.4), $\forall \ b \in \mathcal{B}_{t-1}$, $\forall \ a \in \mathcal{A}$, $\forall \ o \in \mathcal{O}$, such that $p(o|b, a) > 0$. Then $\cup_{t=0}^\infty \mathcal{B}_t$ is the set of belief points reachable by the POMDP by starting from $\mathcal{B}_0$. It is known that for most practical POMDP models, $\cup_{t=0}^\infty \mathcal{B}_t$ constitutes only a subset of the belief simplex (Pineau et al. 2003). It is therefore sufficient to only improve policy on the belief points in $\cup_{t=0}^\infty \mathcal{B}_t$, since they are the belief states that the POMDP will encounter by starting from $\mathcal{B}_0$. It is reasonable to let $\mathcal{B}_0$ contain only the uniform (most uncertain) belief state, i.e., the centroid point in the belief simplex.

For any given initial belief states (points) $\mathcal{B}_0$, we have the following policy improvement theorem.

**Theorem 3.** (Policy improvement in reachable belief states) Let $\mathcal{B}_0$ be a set of initial belief points (states) at time $t = 0$. For time $t = 1, 2, \cdots$, let $\mathcal{B}_t$ be the set of all possible $\tilde{b}_o^a$ in (1.4), $\forall \ b \in \mathcal{B}_{t-1}$, $\forall \ a \in \mathcal{A}$, $\forall \ o \in \mathcal{O}$, such that $p(o|b, a) > 0$. Let $\pi$ be a
stationary policy and $V^\pi(b)$ its infinite-horizon value function. Define the $Q$ function

$$Q^\pi(b, a) = \sum_{s \in S} b(s) R(s, a) + \gamma \sum_{o \in O} p(o|b, a) V^\pi(\tilde{b}_o)$$

(3.10)

Then the new policy

$$\pi'(b) = \arg \max_a Q^\pi(b, a), \quad \forall b \in \bigcup_{t=0}^\infty B_t$$

(3.11)

improves over $\pi$ for any belief state (point) $b \in \bigcup_{t=0}^\infty B_t$, i.e.,

$$V^\pi'(b) \geq V^\pi(b), \quad \forall b \in \bigcup_{t=0}^\infty B_t$$

(3.12)

Proof: By (3.4), (3.10), and (3.11), we have, $\forall b \in \bigcup_{t=0}^\infty B_t$,

$$V^\pi(b) = Q^\pi(b, \pi(b)) \leq Q^\pi(b, \pi'(b))$$

(3.13)

This equation must hold for $\tilde{b}_o^{\pi'(b)}$ for any $o \in O$ because they all are members of $\bigcup_{t=0}^\infty B_t$. Therefore we can keep on expanding $Q^\pi(b, \pi'(b))$ until every $\pi$ appearing in it is replaced by $\pi'$, upon which time the rightmost side of (3.13) becomes $Q^\pi'(b, \pi'(b)) = V^\pi'(b)$ and we have $V^\pi(b) \leq V^\pi'(b)$, which completes the proof.

The reachable belief set $\bigcup_{t=0}^\infty B_t$ may still be infinitely large. We obtain a manageable belief set $\mathcal{B}$ by sampling from $\bigcup_{t=0}^\infty B_t$, using the procedures described in Section 2.3.4. The belief samples $\mathcal{B}$ thus produced yield a uniform representation of the belief states most probably visited by the POMDP. Those belief states that are scarcely visited have negligible contribution in the righthand side of (3.10) and can be ignored. The uniform sampling ensures that the $V^\pi$ computed by minimizing the Bellman residual on the samples will generalize well to the remaining belief points most probably visited. In addition, the optimally determined basis functions also enhance the generalization of $V^\pi$. 

58
3.3.2 Basis representation of the value function

We represent the infinite-horizon value function $V^\pi(b)$ as a linear combination of basis functions of $b$

$$V^\pi(b) = w^T \phi(b) \quad (3.14)$$

where $\phi(b) = [\phi^0(b), \phi^1(b), \ldots, \phi^N(b)]^T$ is a column containing $N + 1$ basis functions with $\phi^0(b) \equiv 1$ accounting for a constant bias term and each $\phi^n(b), n = 1, 2, \ldots, N$ is a basis function. Substituting (3.14) into the Bellman equation (3.4), we obtain

$$w^T \phi(b) = \sum_{s \in S} R(s, \pi(b)) b(s) + \gamma \sum_{o \in O} p(o|b, \pi(b)) w^T \phi(\tilde{b}_{\pi(b)}^o) \quad (3.15)$$

which is simplified to

$$w^T \phi(b) = \eta^\pi(b) + w^T \left[ \sum_{o \in O} \phi(\tilde{b}_{\pi(b)}^o) \rho^\pi_o(b) \right] \quad (3.16)$$

by defining

$$\eta^\pi(b) = \sum_{s \in S} R(s, \pi(b)) b(s) \quad (3.17)$$

$$\rho^\pi_o(b) = \gamma p(o|b, \pi(b)) = \gamma \sum_{s, s' \in S} b(s) \pi(b) T_{ss'}^\pi(b) O_{s' o}^\pi \quad (3.18)$$

Note there is an equation of the form of (3.16) for every belief point $b \in B$.

3.3.3 Policy evaluation by minimizing Bellman residual

We want to compute the value function $V^\pi$, which evaluates how good the policy $\pi$ is. This $V^\pi$ is computed by solving the Bellman equation which, under the basis representation of $V^\pi$, is given by (3.16). The problem reduces to finding the $\phi$ and $w$ that satisfy the Bellman equation as closely as possible. We achieve this by
minimizing the square of the Bellman residual (the difference between the left side and right side of (3.16)), accumulated over all \( b \in B \),

\[
e(\phi, w) = \sum_{b \in B} \left[ w^T \left( \phi(b) - \sum_{o \in O} \phi(\tilde{b}_{o}^{\pi(b)}) \rho_o^{\pi(b)} \right) - \eta^{\pi(b)} \right]^2
\]  

(3.19)

For given \( \phi \), the \( w \) is solved by setting the derivative with respect to \( w \) to zero

\[
\frac{\partial e}{\partial w} = 2 \sum_{b \in B} \left( w^T \phi(b) - \sum_{o=1}^{\mid O \mid} w^T \phi(\tilde{b}_{o}^{\pi(b)}) \rho_o^{\pi(b)} - \eta^{\pi(b)} \right)
\times \left( \phi(b) - \sum_{o \in O} \phi(\tilde{b}_{o}^{\pi(b)}) \rho_o^{\pi(b)} \right) = 0
\]  

(3.20)

\[
w = M^{-1} \sum_{b \in B} \eta^{\pi(b)} \left( \phi(b) - \sum_{o \in O} \phi(\tilde{b}_{o}^{\pi(b)}) \rho_o^{\pi(b)} \right)
\]  

(3.21)

where

\[
M = \sum_{b \in B} \left\{ \left[ \phi(b) - \sum_{o \in O} \phi(\tilde{b}_{o}^{\pi(b)}) \rho_o^{\pi(b)} \right] \left[ \phi(b) - \sum_{o \in O} \phi(\tilde{b}_{o}^{\pi(b)}) \rho_o^{\pi(b)} \right]^T \right\}
\]  

(3.22)

Substituting the solution of \( w \) in (3.21) back into (3.19) gives

\[
e(\phi) = \sum_{b \in B} \left\{ \left( \eta^{\pi(b)} \right)^2 - \eta^{\pi(b)} w^T \left[ \phi(b) - \sum_{o \in O} \phi(\tilde{b}_{o}^{\pi(b)}) \rho_o^{\pi(b)} \right] \right\}
\]  

(3.23)

where \( w \) is related to \( \phi \) by (3.21) and therefore \( e(\phi) \) is a functional with \( \phi \) as free variables. By minimizing \( e(\phi) \), we can determine the optimal \( \phi \) for the \( V^\pi \). Recalling \( \phi(b) = [1, \phi^1(b), \ldots, \phi^N(b)]^T \), this amounts to determining \( N \), the number of basis functions, and the functional form of each basis function \( \phi^n(\cdot) \), \( n = 1, \ldots, N \). We consider an incremental procedure and determine the optimal basis functions one after another. The following theorem gives an efficient algorithm for such an incremental procedure. The proof of the theorem is given in the Appendix of this chapter.
Theorem 4. Let \( \phi(b) = [1, \phi^1(b), \ldots, \phi^N(b)]^T \). Let \( \phi^{N+1}(b) \) be a single basis function. Assume the \( M \) matrices in (3.22), corresponding to \( \phi \) and \([\phi, \phi^{N+1}]^T\), are all non-degenerate. Then

\[
\delta e(\phi, \phi^{N+1}) = e(\phi) - e([\phi, \phi^{N+1}]^T) = \left( c^T w - \sum_{b \in B} \eta^\pi(b) \left[ \phi^{N+1}(b) - \sum_{o \in O} \phi^{N+1}(\tilde{b}_o^\pi(b)) \rho_o^\pi(b) \right] \right)^2 q^{-1}
\]

(3.24)

where \( w \) is given by (3.21), \( M \) is given by (3.22), and

\[
c = \sum_{b \in B} \left\{ \phi(b) - \sum_{o \in O} \phi(\tilde{b}_o^\pi(b)) \rho_o^\pi(b) \right\} \left[ \phi^{N+1}(b) - \sum_{o \in O} \phi^{N+1}(\tilde{b}_o^\pi(b)) \rho_o^\pi(b) \right]^T \quad (3.25)
\]

\[
d = \sum_{b \in B} \left[ \phi^{N+1}(b) - \sum_{o \in O} \phi^{N+1}(\tilde{b}_o^\pi(b)) \rho_o^\pi(b) \right]^2 \quad (3.26)
\]

\[
q = d - c^T M^{-1} c > 0 \quad (3.27)
\]

By (3.24) and (3.27), \( \delta e(\phi, \phi^{N+1}) \geq 0 \), thus adding \( \phi^{N+1} \) to \( \phi \) generally makes the squared Bellman residual decrease or unchanged (indicating convergence). The decrease \( \delta e(\phi, \phi^{N+1}) \) depends on \( \phi^{N+1} \). By selecting basis functions that bring the maximum decrease, we incrementally minimize the squared Bellman residual \( e(\phi) \).

The pseudo Matlab code for the ILSPI policy evaluation is given in Table 3.1.

3.3.4 Policy improvement by pointwise maximization of \( Q^\pi \)

Once the basis representation of the value function \( V^\pi \), i.e., \( \phi \) and \( w \), are found in the policy evaluation, we plug \( V^\pi(\tilde{b}_a^o) = w^T \phi(\tilde{b}_a^o) \) into (3.10) to obtain \( Q^\pi(b, a) \). Then we perform maximization as formulated in (3.11) to get the improved policy \( \pi' \). This maximization is performed in a pointwise manner, working on each \( b \in B \) one by one.
function \([w, \phi(\cdot)] = \text{ILS}(\Phi_B, \Phi_\overline{B}, \eta, \rho)\)

% \(\Phi_B = \{\psi(b) : b \in B, \psi \in \Phi\}\)
% \(\Phi_\overline{B} = \{\psi(b_{\pi(b)}^{\pi(b)}) : b \in B, o \in O, \psi \in \Phi\}\)
% Initialization:
\(N = 0, \phi(\cdot) = 1;\)
Compute \(M\) with (3.22), \(w\) with (3.21), and \(e_0\) with (3.19);
while the sequence \(\{\epsilon_n\}_{n=0:N}\) is not converged
for all basis function \(\phi^{N+1} = \psi \in \Phi\)
    Compute \(c\) with (3.25), \(d\) with (3.26), and \(q\) with (3.27);
    if \(q = 0\)
        \(\Phi = \Phi \setminus \{\psi\};\)  continue;
    else
        compute \(\delta e(\phi, \psi)\) using (3.24);
    end
end
Compute \(\psi^* = \arg\max_{\psi \in \Phi} \delta e(\phi, \psi)\);
% Update:
\(\phi(\cdot) = [\phi^T(\cdot), \psi^*(\cdot)]^T;\)  \(\Phi = \Phi \setminus \{\psi^*\}\)
\(M \leftarrow M^{\text{new}}\) and \(w \leftarrow w^{\text{new}}\), where \(M^{\text{new}}\) is computed with (3.44) and \(w^{\text{new}}\) with (3.46);
\(e_{N+1} = e_N - \delta e(\phi, \psi^*);\)  \(N = N + 1;\)
end

The ILSPI policy improvement is given in Table 3.2 as part of the main ILSPI loop, presented in the form of pseudo Matlab code.

It is noted that the ILSPI outputs \(w\) and \(\phi(\cdot)\), giving \(V^\pi(b) = w^T\phi(b)\), which is substituted into (3.10) to obtain \(Q^\pi(b, a)\) and produce the policy

\[\pi(b) = \arg\max_a Q^\pi(b, a)\]

### 3.3.5 Selection of the form of basis functions

Since the value function \(V^\pi(b)\) is represented as a linear combination of basis functions, the selection of basis functions plays an important role in the policy quality. Generally, there is a wide variety of basis functions available for function approxi-
The ILSPI Algorithm: Main Loop

\begin{table}[h]
\centering
\begin{tabular}{|l|}
\hline
function \([w, \phi(\cdot)] = \text{ILSPI}(\text{POMDP}, B, \Phi)\) \\
\% \(\Phi\) is a set of candidate basis functions \\
\% Pre-computation \\
Compute \(\tilde{b}_o^a\) with (1.4) and \(\rho_o^{(b)=a}\) with (3.18), \(\forall \ b \in B, \ \forall \ a \in A, \ \forall \ o \in O\); \\
Compute \(\Phi_B = \{\psi(b) : b \in B, \psi \in \Phi\}\); \\
while \(1/\|B\| \sum_{b \in B} V^\pi(b)\) does not converge \\
\% Policy Evaluation \\
Compute \(\eta^\pi(b)\) with (3.17), \(\forall \ b \in B, \ o \in O\); Compute \\
\(\Phi_B = \{\psi(\tilde{b}_o^a(b)) : b \in B, o \in O, \psi \in \Phi\}\); \\
\([w, \phi(\cdot)] = \text{ILS}(\Phi_B, \Phi_B, \eta, \rho)\); \\
\% Policy Improvement \\
Compute \(Q^\pi(b, a)\) with (3.10), (3.14), \(\forall \ b \in B, \ \forall \ a \in A\); \\
Update \(\pi(b) = \arg \max_{a \in A} Q^\pi(b, a), \forall \ b \in B;\)
\hline
\end{tabular}
\caption{The ILSPI Algorithm: Main Loop}
\end{table}

Where \(c_i\) and \(\sigma_i\) are the centroid and width of the \(i\)th basis function. A second order inhomogeneous polynomial has the form

\[\phi_i(b) = (b^T c_i + 1)^2\]  

where \(c_i\) is the coefficient vector and the intercept 1 indicates the inhomogeneity.

The radial basis functions are widely used in approximation, interpolation, and learning, and they have very good approximation abilities that are well known (Powell 1987; T. and F. 1987; Park and Sandberg 1991). Although the RBF functions do not take advantage of the piecewise linear convexity of the optimal value function, they
do have the ability of approximating the value function to an arbitrary accuracy if
the belief samples are properly selected.

We consider the second order inhomogeneous polynomial as suitable to value
approximation, because the second order polynomials are convex and match the
value function very well. We note that quadratic functions have been recently used
in (T. Wang and Schuurmans 2006) to compose a convex upper bound to the optimal
value function and promising results have been obtained.

Our results are presented based on the radial basis functions, though we observe
similar results by using the second order polynomials. In our algorithm, a set of
candidate radial basis functions are generated, the $i$th one of which has the form

$$
\phi_i(b) = \exp\left(-\frac{\|b - b_i\|^2}{2\sigma}\right)
$$

(3.30)

where the width $\sigma$ is a constant and the centroid $b_i$ is one of the belief samples in $\mathcal{B}$. Based on Theorem 4, we select a number of basis functions from the candidates to compose our approximate value function, in each policy evaluation step.

### 3.3.6 Time and Memory Complexity of the ILSPI

The time and memory complexity of the ILSPI is given in Table 3.3, where time refers
to the number of multiplications and memory refers to the number of intermediate
variables (not counting the input variables) involved the computation. A computation
task is performed outside the while-loop in Table 3.2 if it is marked with “out”
and inside the while-loop if marked with “in”. Some computations tasks can be
performed in or out the loop, at different time and memory costs. Table 3.2 represents
a specific combination of the “out” and “in”. The results presented in this chapter
were produced by computing $\tilde{b}_\pi^a$, $\rho_\pi^{(b)}$, $\phi(b)$ outside of the loop and all others inside
the loop.
In Table 3.3, \( u = M^{-1}c \), \( L \) is the number of ILSPI iterations (i.e., while-loops in Table 3.2) performed, and \( \Upsilon_\phi \) is the time of computing a single basis function \( \psi \in \Phi \) at a single belief point \( b \in \mathcal{B} \). For radial basis functions (RBF), which we use to produce the results in this chapter, \( \Upsilon_\phi = |S| \). The \( c \) is computed in time of \( O(|\mathcal{B}|\Phi NL) \) by storing the elements in \( c \) for every \( \psi \in \Phi \), which consumes a memory of \( O(|\Phi|N) \). The \( u = M^{-1}c \) is introduced as an intermediate variable to speed up computation of \( q = d + c^TM^{-1}c \). The \( u \) is stored for every \( \psi \in \Phi \) using a memory of \( O(|\Phi|N) \).

### 3.4 Piecewise Linear Policy Iteration (PLPI)

In Section 3.3 we used a linear combination of basis functions to approximate the value function \( V^\pi \) by automatically selecting the basis and adjusting the combination coefficients. However, considering the fact that the value function of the POMDP...
is either piecewise linear or can be arbitrarily closely approximated by a piecewise linear function, it is possible to perform policy iteration directly based on a piecewise linear approximation.

As we have discussed above, it is very difficult to find all $\alpha$-vectors in (3.9) over the entire belief simplex for large problems. Therefore we restrict ourselves to finding the $\alpha$-vectors over a finite set of belief samples $B$, and we associate each belief sample with a unique $\alpha$-vector. Since there is a one-to-one correspondence between the belief samples and the $\alpha$-vectors, we use $k$ to index the $k$th belief sample as well as the associated $\alpha$-vector. Restricted to the belief samples, we rewrite Equation (3.9) as

$$\alpha^{\text{new}}_k(s) = R(s, a(k)) + \gamma \sum_{o \in \mathcal{O}} \sum_{s' \in S} T^{a(k)}_{ss'} \Omega^{a(k)}_{s'o} \alpha^{\text{old}}_{l(b, a(k), o)}(s')$$

(3.31)

where $l(b, a(k), o) = \arg\max_j \left\{ \sum_{s \in S} b(s) \sum_{s' \in S} T^{a(k)}_{ss'} \Omega^{a(k)}_{s'o} \alpha^{\text{old}}_j(s') \right\}$, and $a(k)$ is the action at the $k$th belief sample $b_k \in B$.

The piecewise linear policy iteration (PLPI) is described as follows.

**Policy representation** The policy is represented by $\pi = \{ (\alpha_k, a(k)) : b_k \in \mathcal{B} \}$, where $\alpha_k$ is the $\alpha$-vector corresponding to the $k$th belief sample, and $a(k)$ is the associated action. The policy maps any belief point $b$ to the action

$$\pi(b) = \begin{cases} a(k^*) & \text{if } k^* = \arg\max_k b^T \alpha_k \end{cases}$$

(3.32)

**Policy evaluation** In policy evaluation, we compute the $\alpha$-vectors (local value functions) from the controls (actions) at belief samples, and the $\alpha$-vectors along with the controls complete the policy representation in (3.32). Given the action $a(k)$ at every sample $b_k \in \mathcal{B}$, the $\alpha$-vectors are computed by iterative application of Equation (3.31). We initialize the $\alpha$-vectors for the belief samples and denote the initialization...
by $\alpha^0 = \{\alpha^0_k : b_k \in B\}$. We plug the $\alpha$-vectors $\alpha^n$ into the right side of (3.31) to obtain the updates $\alpha^{n+1}$. We repeat this procedure with successively larger $n$, starting from $n = 0$. At each iteration, the value of each belief sample is computed by

$$V_n(b) = \max_k b^T \alpha^n_k$$

(3.33)

If $\sum_{b \in B} V_{n+1}(b) \approx \sum_{b \in B} V_n(b)$, we say that the policy evaluation converges.

However, we do not have to wait for the convergence. Suppose we stop the iteration at some $n$, we obtain a policy $\pi$, which is represented by $\pi = \{(\alpha_k^n, a(k)) : b_k \in B\}$.

**Policy improvement** As in the ILSPI, we perform pointwise maximization to improve the policy on the belief samples.

Once the basis representation $\pi = \{(\alpha_k^n, a(k)) : b_k \in B\}$ is found in policy evaluation, we plug $V^\pi(\tilde{b}^a_o) = \max_k (\alpha_k^n)^T \tilde{b}^a_o$ into (3.10) to obtain $Q^\pi(b, a)$. Then we perform maximization as formulated in (3.11) to get the improved controls at the belief samples, i.e., $a'(k) = \arg \max_a Q^\pi(b_k, a)$.

The improved controls at belief samples, $\{a'(b_k) : b_k \in B\}$, are fed into the policy evaluation iterations, to compute the $\alpha$-vectors, which along with $\{a'(b_k) : b_k \in B\}$ complete the representation of the improved policy. The complete piecewise linear policy iteration algorithm is given in Table 3.4.

The computational complexity of piecewise linear policy iteration is $O(|B|^2 |S| |O||IterN|_{PE}|IterN|_{PI})$, where $|IterN|_{PE}$ is the number of policy-evaluation iterations (inner loop) and $|IterN|_{PI}$ is the number of policy iterations (outer loop).

We find in experiments that we do not need to perform the policy evaluation until it converges. This gives rise to three versions of the PLPI algorithm:

- If we perform the iteration in policy-evaluation *multiple steps* but do not wait for convergence, and then perform policy improvement, we obtain an algorithm
Table 3.4: The Piecewise Linear Policy iteration Algorithm

<table>
<thead>
<tr>
<th>Function $\alpha = \text{alphaPI}(\text{POMDP}, B)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>While $\frac{1}{</td>
</tr>
<tr>
<td>% Policy Evaluation</td>
</tr>
<tr>
<td>While $\alpha$-vector for each belief point $b$ does not converge</td>
</tr>
<tr>
<td>Iteratively update $\alpha$-vectors with (3.31), $\forall b \in B, o \in O$;</td>
</tr>
<tr>
<td>% Policy Improvement</td>
</tr>
<tr>
<td>Compute $Q^\pi(b, a)$ with (3.10), (3.33), $\forall b \in B, \forall a \in A$;</td>
</tr>
<tr>
<td>Update $\pi(b) = \arg \max_{a \in A} Q^\pi(b, a)$, $\forall b \in B$;</td>
</tr>
</tbody>
</table>

that is similar to the modified policy iteration in MDPs (Puterman and Shin 1978).

- If we perform the iteration in policy-evaluation a single step (i.e., $|\text{IterN}|_{PE} = 1$), and then perform policy improvement, we obtain the point-based value iteration (Pineau et al. 2003).

- If we perform the iteration in policy-evaluation until convergence (i.e., $|\text{IterN}|_{PE} \rightarrow \infty$) and then perform policy improvement, the algorithm we obtain is the originally intended piecewise linear policy iteration.

There is a tradeoff between $|\text{IterN}|_{PE}$ and $|\text{IterN}|_{PI}$. The more iterations we have in policy-evaluation, the fewer iterations we need in the overall policy iteration — that is, the fewer times we need to improve the policy.

3.5 Experimental Results

We demonstrate the performance of the proposed ILSPI and PLPI algorithms on four benchmark problems. The four benchmark problems are described in Section 2.4. The proposed algorithms are compared to five point-based algorithms of value itera-
tion: Grid Brafman (1997), PBUA Poon (2001), PBVI Pineau et al. (2003), Perseus Spaan and Vlassis (2004), and HSVI Smith and Simmons (2004, 2005), in terms of performance and policy computation time. To replicate the experiments of previous authors, we measure algorithm performance by the accumulative-discounted-reward averaged over $N_{\text{test}}$ independent tests, as in Section 2.4.

The experimental results of the proposed ILSPI algorithm and PLPI algorithm are summarized in Table 3.5, in comparison to the other five algorithms. It is demonstrated that for each of the four problems the ILSPI and PLPI algorithms compare competitively to its value-iteration counterparts in terms of both performance and computational time.

The time efficiency of ILSPI can be attributed to two facts: (a) ILSPI leverages the simplicity of the least squares criterion (squared Bellman residual) and uses matrix tricks to speed up the computation; (b) ILSPI typically converges within a smaller number of iterations than value iteration methods. This is not surprising as fast convergence is generally observed in policy iteration algorithms Sutton and Barto (1998). And the time efficiency of piecewise linear policy iteration owes to two facts: (a) piecewise linear policy iteration is based on a subset of belief points instead of on the whole belief simplex; (b) there exists an optimal choice of $|\text{IterN}|_{PE}$ in piecewise linear policy iteration.

To demonstrate the accuracy of the ILSPI algorithm as well as its fast convergence, we apply the algorithm to the tiger problem Kaelbling et al. (1998). Figure 3.2 shows the evolution of the value function $V^\pi(b)$ over the ILSPI iterations, starting from a random policy. Figure 3.3 compares the $V^\pi(b)$ of the converged ILSPI policy with that of the exact policy. It is seen from the figures that ILSPI converges after only three iterations and that the converged policy is almost identical to the exact policy.
Table 3.5: Results on the benchmark problems, where the time is displayed in seconds. The results marked with * are those we obtained by coding the respective algorithms in Matlab; other results may have been coded in languages other than Matlab and executed on computer platforms different from ours.

<table>
<thead>
<tr>
<th>Method</th>
<th>Reward</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tiger-Grid</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grid</td>
<td>0.94</td>
<td>n.v.</td>
</tr>
<tr>
<td>PBUA (2001)</td>
<td>2.30</td>
<td>12116</td>
</tr>
<tr>
<td>PBVI (2003)</td>
<td>2.25</td>
<td>3448</td>
</tr>
<tr>
<td>PBVI (*)</td>
<td>2.23</td>
<td>2239</td>
</tr>
<tr>
<td>Perseus</td>
<td>2.34</td>
<td>104</td>
</tr>
<tr>
<td>Smith and Simmons (2004)</td>
<td>2.35</td>
<td>10341</td>
</tr>
<tr>
<td>Smith and Simmons (2005)</td>
<td>2.30</td>
<td>52</td>
</tr>
<tr>
<td>ILSPI (*)</td>
<td>2.21</td>
<td>136</td>
</tr>
<tr>
<td>PLPI (*)</td>
<td>2.04</td>
<td>62</td>
</tr>
<tr>
<td><strong>Hallway</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBUA (2001)</td>
<td>0.53</td>
<td>450</td>
</tr>
<tr>
<td>PBVI (2003)</td>
<td>0.53</td>
<td>288</td>
</tr>
<tr>
<td>PBVI (*)</td>
<td>0.54</td>
<td>1166</td>
</tr>
<tr>
<td>Perseus</td>
<td>0.51</td>
<td>35</td>
</tr>
<tr>
<td>Smith and Simmons (2004)</td>
<td>0.52</td>
<td>10836</td>
</tr>
<tr>
<td>Smith and Simmons (2005)</td>
<td>0.52</td>
<td>2.4</td>
</tr>
<tr>
<td>ILSPI (*)</td>
<td>0.54</td>
<td>66</td>
</tr>
<tr>
<td>PLPI (*)</td>
<td>0.54</td>
<td>119</td>
</tr>
<tr>
<td><strong>Hallway2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBUA (2001)</td>
<td>0.35</td>
<td>27898</td>
</tr>
<tr>
<td>PBVI (2003)</td>
<td>0.34</td>
<td>360</td>
</tr>
<tr>
<td>PBVI (*)</td>
<td>0.35</td>
<td>2345</td>
</tr>
<tr>
<td>Perseus</td>
<td>0.35</td>
<td>10</td>
</tr>
<tr>
<td>Smith and Simmons (2004)</td>
<td>0.35</td>
<td>10010</td>
</tr>
<tr>
<td>Smith and Simmons (2005)</td>
<td>0.35</td>
<td>1.5</td>
</tr>
<tr>
<td>ILSPI (*)</td>
<td>0.30</td>
<td>206</td>
</tr>
<tr>
<td>PLPI (*)</td>
<td>0.35</td>
<td>300</td>
</tr>
<tr>
<td><strong>Tag</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perseus</td>
<td>-6.17</td>
<td>1670</td>
</tr>
<tr>
<td>Smith and Simmons (2004)</td>
<td>-6.37</td>
<td>10113</td>
</tr>
<tr>
<td>Smith and Simmons (2005)</td>
<td>-6.36</td>
<td>24</td>
</tr>
<tr>
<td>ILSPI (*)</td>
<td>-12.3</td>
<td>737</td>
</tr>
<tr>
<td>PLPI (*)</td>
<td>-6.34</td>
<td>3121</td>
</tr>
</tbody>
</table>
Figure 3.2: The value functions $V^\pi(b)$ of each ILSPI iteration for the tiger problem [Kaelbling et al. (1998)], which has $|S| = 2$. The horizontal axis is $b(1) = p(s = \text{"tiger on the left"}|\text{history})$. The vertical axis is $V$. The initial policy is a random one. Green circle denotes the action "open left door"; blue plus denotes the action "open right door"; red dot denotes the action "listen".

### 3.6 Conclusions on ILSPI and PLPI

We have presented two new policy iteration algorithms, incremental least squares policy iteration (ILSPI) and piecewise linear policy iteration (PLPI), for solving infinite-horizon stationary policies of partially observable Markov decision processes (POMDPs).

The ILSPI represents the policy with optimally determined basis functions and computes the value function by minimizing the square error between the left-hand
Figure 3.3: Comparison of the converged ILSPI policy with the exact policy, for the tiger problem. Green circle denotes the action “open left door”; blue plus denotes the action “open right door”; red dot denotes the action “listen”.

The exact value function and optimal policy provide a baseline for evaluating the performance of the ILSPI algorithm. The value function and policy of ILSPI are compared against the exact policy to show the effectiveness of the algorithm.

Side and righthand side of the Bellman equation (Bellman residual). In the policy improvement step, the ILSPI improves the policy in the belief states reachable by the POMDP. Like policy iteration in general, the ILSPI converges to the optimal policy within a small number of iterations. In addition, the simplicity of the least squares is leveraged to make the ILSPI policy evaluation efficient.

The PLPI algorithm utilizes the piecewise linearity of the value function. By finding $\alpha$-vectors for a finite set of belief samples and performing the policy improvement on belief samples, we avoid the difficulties encountered in (Sondik 1978) and (Hansen 1997) and significantly reduced the computation and memory consumptions.

The ILSPI and PLPI algorithms are applied to four benchmark problems and are
demonstrated to be competitive to value iteration algorithms, in terms of performance and computational efficiency.

3.7 Appendix: Analysis and Proof

3.7.1 Differences between two least-squares approximations

Bellman residual minimization Approximation  According to the Bellman equation, the value function is represented

$$V_\pi = R + \gamma P_\pi V_\pi$$  \hspace{1cm} (3.34)

where $V$ and $R$ is a $|B| \times 1$ vector, $P$ is a $|B| \times |B|$ transition matrix of belief points.

We can approximate the value function $V_\pi$ by minimizing the squares of the Bellman residual ($L_2$ norm of the difference between the left hand side and right hand side of (3.34)), which is defined as

$$e = \|V_\pi - R - \gamma P_\pi V_\pi\|^2$$  \hspace{1cm} (3.35)

For ease of exposition, we let $\phi(b) = [\phi^0(b), \phi^1(b), \ldots, \phi^N(b)]$ be a row containing $N + 1$ basis functions with $\phi^0(b) \equiv 1$ accounting for a constant bias term. Each column $\phi^n(b), n = 1, 2, \ldots, N$ is a basis function. Then $\Phi$ is a $|B| \times (N + 1)$ matrix.

The value function is represented as

$$V_\pi = \Phi w$$  \hspace{1cm} (3.36)

On the one hand, we substitute (3.36) into (3.35) and set the derivative $\frac{\partial e}{\partial w}$ to zero to solve $w$ as

$$\nabla w = \frac{\partial e}{\partial w} = 2(\Phi - \gamma P_\pi \Phi)^T(\Phi w - R - \gamma P_\pi \Phi w) = 0$$  \hspace{1cm} (3.37)

$$(\Phi - \gamma P_\pi \Phi)^T(\Phi - \gamma P_\pi \Phi)w = (\Phi - \gamma P_\pi \Phi)^T R$$

73
\[ w = [(\Phi - \gamma P_\pi \Phi)^T (\Phi - \gamma P_\pi \Phi)]^{-1} (\Phi - \gamma P_\pi \Phi)^T R \]  
(3.38)

On the other hand, we substitute (3.36) into (3.34) to write
\[ \Phi w = R + \gamma P_\pi \Phi w \]
\[ (\Phi - \gamma P_\pi \Phi)w = R \]  
(3.39)

Let \( \Theta = \Phi - \gamma P_\pi \Phi \). Since \( N + 1 \) is much smaller than \(|\mathcal{S}|\), (3.39) constitutes an over-constrained linear system (Lagoudakis and Parr 2003). The vector \( R \) may not be in the space spanned by the columns of \( \Theta \), in which case we can not find a solution \( w \) to (3.39). We then project the vector \( R \) onto the space spanned by the columns of \( \Theta \), to yield
\[ R_\perp = \Theta (\Theta^T \Theta)^{-1} \Theta^T R \]  
(3.40)

where \( \Theta (\Theta^T \Theta)^{-1} \Theta^T \) is the orthogonal projection matrix. Then \( w \) is approximated by the projection \( R_\perp \)
\[ \Theta w = R_\perp = \Theta (\Theta^T \Theta)^{-1} \Theta^T R \]
\[ w = (\Theta^T \Theta)^{-1} \Theta^T R \]  
(3.41)

which is seen to be exactly the same as (3.38), by substituting back \( \Theta = \Phi - \gamma P_\pi \Phi \).

**Least squares fixed point Approximation (Lagoudakis and Parr 2003)**

Least squares fixed point approximation is based on the algorithm of least squares temporal difference learning (LSTD) (Boyan 2002; Bradtke and Barto 1996; Sutton and Barto 1998). LSTD learning is to approximate true value of \( V^\pi \) with \( V_t \) which is represented by \( \Phi w \). Minimizing the mean squares error \((V^\pi - V_t)^2\) with respect to parameter \( w \), we obtain \( \nabla w = (V^\pi - V_t)\Phi \). Since it is impossible to obtain the true value of \( V^\pi \), LSTD uses the one-step update \( R + \gamma PV_t \) to approximate it. We have
\[ \nabla w = (R + \gamma PV_t - V_t)\Phi \]  
(3.42)
Comparing (3.42) to (3.37), we observe that the difference between (3.42) and (3.37) results from the approximation of $V^\pi$ in LSTD. Since $V^\pi$ is approximated using $R + \gamma PV$, $V^\pi$ should also be a function of parameter $w$, which is not considered when minimizing the mean square error in LSTD. Based on this LSTD approximation, $w$ is solved

$$\Phi^T(\Phi - \gamma P_\pi \Phi)w = \Phi^T R$$

$$w = [\Phi^T(\Phi - \gamma P_\pi \Phi)]^{-1} \Phi^T R$$

(3.43)

From the analysis of Lagoudakis and Parr (2003), since $\Phi w = R + \gamma P_\pi \Phi w$ constitutes an over-constrained system, $R + \gamma P_\pi \Phi w$ may also not in the space spanned by vectors of $\Phi$ (not $\Theta$). Projecting the vector $R + \gamma P_\pi \Phi w$ (which is not a constant vector but varies with parameter $w$) into the space spanned by the columns of $\Phi$, $w$ is obtained as in (3.43) (Lagoudakis and Parr 2003).

In Lagoudakis and Parr (2003) and Munos (2003), the authors provide further analysis on these two approximations and the disadvantages and advantages of each approximation.

### 3.7.2 Proof of Theorem 4

For any belief state $b$, we define the $\varphi$ notation $\varphi(b) = \phi(b) - \sum_{o \in O} \phi(\tilde{b}^{\pi(b)}_o) \pi(b)$ and $\varphi^{N+1}(b) = \phi^{N+1}(b) - \sum_{o \in O} \phi^{N+1}(\tilde{b}^{\pi(b)}_o) \pi(b)$. We use $\varphi$ and $\varphi^{N+1}$ in this proof to simplify the equations. Let $\phi^{new} = [\phi, \phi^{N+1}]^T$, which is transformed to the $\varphi$ nation as $\varphi^{new} = [\varphi, \varphi^{N+1}]^T$. By (3.22), the $M$ matrix corresponding to $\phi^{new}$ is

$$M^{new} = \sum_{b \in B} \begin{bmatrix} \varphi(b) \\ \varphi^{N+1}(b) \end{bmatrix}^T = \begin{bmatrix} M \\ c^T \\ d \end{bmatrix}^T$$

(3.44)

where $M$, $c$, and $d$ are as in (3.22), (3.25), and (3.26), respectively. By the conditions of the theorem, the matrices $M$ and $M^{new}$ are all full rank. Using the block matrix
inversion formula, we get

$$(M_{\text{new}})^{-1} = \begin{bmatrix} M^{-1} + M^{-1}cq^{-1}c^TM^{-1} & -M^{-1}cq^{-1} \\ -q^{-1}c^TM^{-1} & q^{-1} \end{bmatrix}$$  \hspace{1cm} (3.45)$$

where $q$ is given in (3.27). By (3.21), the $w_{\text{new}}$ corresponding to $[\phi^T, \phi^{N+1}]^T$ are

$$w_{\text{new}} = [M_{\text{new}}]^{-1} \sum_{b \in B} \eta^{\pi(b)} \begin{bmatrix} \varphi(b) \\ \varphi^{N+1}(b) \end{bmatrix} = \begin{bmatrix} w + M^{-1}cq^{-1}g \\ -q^{-1}g \end{bmatrix}$$  \hspace{1cm} (3.46)$$

with

$$g = c^Tw - \sum_{b \in B} \eta^{\pi(b)}\varphi^{N+1}(b)$$  \hspace{1cm} (3.47)$$

Hence,

$$[\varphi_{\text{new}}(b)]^Tw_{\text{new}} = [\varphi(b)]^Tw + \left\{ [\varphi(b)]^TM^{-1}c - \varphi^{N+1}(b) \right\}gq^{-1}$$  \hspace{1cm} (3.48)$$

By (3.23),

$$e(\phi_{\text{new}}) = \sum_{b \in B} \left\{ (\eta^{\pi(b)})^2 - \eta^{\pi(b)}(w_{\text{new}})^T\varphi_{\text{new}}(b) \right\}$$  \hspace{1cm} (3.49)$$

Substituting (3.48), and applying (3.21), (3.23), and (3.47),

$$e(\phi_{\text{new}}) = e(\phi) - \left\{ c^Tw - \sum_{b \in B} \eta^{\pi(b)}\varphi^{N+1}(b) \right\}^2 q^{-1}$$  \hspace{1cm} (3.50)$$

which is (3.24) in the $\varphi$ notation. By the conditions of the theorem, $M_{\text{new}}$ is full rank and is positive definite by construction. By (3.45), $q^{-1}$ is a diagonal element of $(M_{\text{new}})^{-1}$, hence $q^{-1} > 0$ and $q > 0$. The proof is thus completed.
Chapter 4

Regionalized Policy Representation for Reinforcement Learning in POMDPs

In the previous chapters, we assumed the POMDP model of the environment is given and available to the agent, hence the problems dealt with model-based planning. In this chapter and in Chapter 5, we consider the model-free reinforcement learning, assuming that the model of the environment is unavailable to the agent and the agent learns the policy empirically from experiences.

4.1 Introduction

Model-free reinforcement in partially observable stochastic domains is a challenging problem. One difficulty in this problem arises from the fact that the belief state which summarizes the history is not available when the POMDP model is unknown. Carrying a long history of actions and observations is cumbersome and inefficient. To solve the representational issue, many methods have been proposed to compress the history and express it in a compact manner, these including the reactive policy and history truncation (Jaakkola et al. 1995; Baxter and Bartlett 2001), finite policy graphs (Meuleau et al. 1999), finite state controllers (Aberdeen and Baxter 2002), predictive state representations (Littman et al. 2002; Singh et al. 2003, 2004; James and Singh 2004; Wiewiora 2005; Wolfe et al. 2005; Bowling et al. 2006; McCracken and Bowling 2006) and utile distinction HMMs (Wierstra and Wiering 2004).

In this chapter we introduce a new parametric model, called regionalized policy representation (RPR), for representing a stochastic policy in the absence of a POMDP model. The RPR characterizes the dynamics of decision states instead of world-states.
alone and it has a direct correspondence to the piecewise constancy of optimal actions for a finite horizon (Smallwood and Sondik 1973). The dynamics of decision states are driven jointly by actions and observations, the former capturing the dynamics of world-states and the latter reflecting the perceptual aliasing. We develop a policy-dependent value function empirically evaluated on the available experience collected from the agent-environment interaction. We learn the RPR by seeking maximization of the empirical value function with respect to the RPR parameters. We present both maximum-value learning and variational Bayesian learning of the RPR.

4.2 Mathematical Formulation of the RPR

Since the belief state is not observable in the model-free case, we treat the belief-state as a hidden (latent) variable and marginalize it to yield a stochastic POMDP policy that is purely dependent on the history of actions and observations. The belief state dynamics, as well as the optimal control in each state, is learned empirically from experiences, instead of being computed from a POMDP model. In general, we could learn the dynamics and control in the continuous space of belief state. For the convenience of exposition, however, we restrict ourselves mainly to the case when the continuous belief-state space is quantized into a finite set of disjoint regions. With the quantization, we then learn the dynamics and control of the belief region, instead of individual belief state. The resulting framework is termed regionalized policy representation to reflect the fact that the policy of action-choosing is expressed through the dynamics of belief regions as well as the local controls in each region. We also use the term decision state as a synonym of belief region, because it is an elementary unit to encode the decisions of action-choosing. Now we give a formal definition of the proposed representation.

Definition 5. A regionalized policy representation (RPR) is a tuple $\langle A, O, Z, W, \mu, \pi \rangle$
specified as follows. The $\mathcal{A}$ and $\mathcal{O}$ are respectively a finite set of actions and observations. The $\mathcal{Z}$ is a finite set of decision states (belief regions). The $W$ are decision-state transition matrices with $W(z, a, o', z')$ denoting the probability of transiting from $z$ to $z'$ when taking action $a$ in decision state $z$ results in observing $o'$. The $\mu$ is the initial distribution of decision states with $\mu(z)$ denoting the probability of initially being in decision state $z$. The $\pi$ are state-dependent stochastic policies with $\pi(z, a)$ denoting the probability of taking action $a$ in decision state $z$.

The stochastic formulation of $W$ and $\pi$ in Definition 5 is fairly general and subsumes two special cases.

1. If $z$ shrinks down to a single belief-state $b$, $z = b$ becomes a sufficient statistic of the POMDP (Smallwood and Sondik 1973) and there is a unique action associated with it, thus $\pi(z, a)$ is deterministic and the local policy can be simplified as $a = \pi(b)$. In addition, $W(z, a, o', z') = W(b, a, o', b')$ is given by (1.6) and the transition of $z'$ is also deterministic.

2. If the belief regions form a Markov partition of the belief-state space, i.e., $\mathcal{B} = \cup_{z \in \mathcal{Z}} B_z$, then the action choice in each region is constant and one region transits completely to another (or itself). In this case, both $W$ and $\pi$ are deterministic and, moreover, the policy yielded by the RPR (see (4.4)) is deterministic and finite transient (Sondik 1978).

In both of the two special cases, each $z$ has one action choice $a = \pi(z)$ associated with it, and one can write $W(z, a, o', z') = W(z, \pi(z), o', z')$, thus the transition of $z$ is driven solely by $o$. In general, each $z$ represents multiple individual belief-states, and the belief region transition is driven jointly on $a$ and $o$. The action-dependency captures the state dynamics of the POMDP, and the observation-dependency reflects the partial observability of the state (perception aliasing).
We now turn to discussing the policy yielded by the RPR. To make notation simple, the following conventions are observed in the discussion and throughout this chapter and Chapter 5:

The elements of \( \mathcal{A} \) are enumerated as \( \mathcal{A} = \{1, 2, \cdots, |\mathcal{A}|\} \), where \( |\mathcal{A}| \) denotes the cardinality of \( \mathcal{A} \). Similarly, \( \mathcal{O} = \{1, 2, \cdots, |\mathcal{O}|\} \) and \( \mathcal{Z} = \{1, 2, \cdots, |\mathcal{Z}|\} \).

A sequence of actions \( (a_0, a_1, \cdots, a_T) \) is abbreviated as \( a_{0:T} \), where the subscripts index discrete time steps. Similarly a sequence of observations \( (o_1, o_2, \cdots, o_T) \) is abbreviated as \( o_{1:T} \), and an sequence of decision states \( (z_0, z_1, \cdots, z_T) \) is abbreviated as \( z_{0:T} \), etc.

A history \( h_t \) is the set of actions executed and observation received up to time step \( t \), i.e., \( h_t = \{a_{0:t-1}, o_{1:t}\} \).

Let \( \Theta = \{\pi, \mu, W\} \) denote the parameters of the RPR. Given a history of actions and observations, \( h_t = (a_{0:t-1}, o_{1:t}) \), collected up to the time step \( t \), the RPR yields a joint probability distribution of \( z_{0:t} \) and \( a_{0:t} \)

\[
p(a_{0:t}, z_{0:t}|o_{1:t}, \Theta) = \mu(z_0)\pi(z_0, a_0) \prod_{\tau=1}^t W(z_{\tau-1}, a_{\tau-1}, o_{\tau}, z_{\tau})\pi(z_{\tau}, a_{\tau}) \tag{4.1}
\]

where application of local controls \( \pi(z_t, a_t) \) at every time step implies that \( a_{0:t} \) are all drawn according to the RPR. The decision states \( z_{0:t} \) in (4.1) are hidden variables and we marginalize them to get

\[
p(a_{0:t}|o_{1:t}, \Theta) = \sum_{z_{0:t}} \mu(z_0)\pi(z_0, a_0) \prod_{\tau=1}^t W(z_{\tau-1}, a_{\tau-1}, o_{\tau}, z_{\tau})\pi(z_{\tau}, a_{\tau}) \tag{4.2}
\]

It follows from (4.2) that

\[
p(a_{0:t-1}|o_{1:t}, \Theta) = \sum_{a_{t-1}} p(a_{0:t}|o_{1:t}, \Theta)
\]

80
\[
\sum_{z_0, \ldots, z_{t-1}=1} |Z| \mu(z_0) \pi(z_0, a_0) \prod_{\tau=1}^{t-1} W(z_{\tau-1}, a_{\tau-1}, o_\tau, z_\tau) \pi(z_\tau, a_\tau)
\]
\[
\times \sum_{a_1} \sum_{z_t=1} |A| \sum_{a_t=1} |Z| \sum_{z_{t-1}} W(z_{t-1}, a_{t-1}, o_t, z_t) \pi(z_t, a_t)
\]
\[
= 1
\]
\[
= p(a_{0:t-1}|o_{1:t-1}, \Theta) \tag{4.3}
\]

which implies that observation \(o_t\) does not influence the actions before \(t\), which is agreement with our expectation. From (4.2) and (4.3), we can write the history-dependent distribution of action choices

\[
p(a_\tau|h_\tau, \Theta) = p(a_\tau|a_{0:\tau-1}, o_{1:\tau}, \Theta) = \frac{p(a_0, a_1, \ldots, a_\tau, o_1, \ldots, o_\tau, \Theta)}{p(a_{0:\tau-1}|o_{1:\tau-1}, \Theta)} \tag{4.4}
\]

which gives a stochastic RPR policy for choosing the action \(a_t\), given the historical actions and observations. The policy is purely history-dependent, with the unobservable belief regions \(z\) integrated out.

The history \(h_t\) forms a Markov process with deterministic transitions driven by actions and observations: \(h_t = h_{t-1} \cup \{a_{t-1}, o_t\} \). Applying this recursively, we get

\[
\prod_{\tau=0}^{t} p(a_\tau|h_\tau, \Theta) = \left[ \prod_{\tau=0}^{t-2} p(a_\tau|h_\tau, \Theta) \right] p(a_{t-1}|h_{t-1}, \Theta) p(a_t|h_{t-1}, a_{t-1}, o_t, \Theta)
\]
\[
= \left[ \prod_{\tau=0}^{t-2} p(a_\tau|h_\tau, \Theta) \right] p(a_{t-1}|h_{t-1}, o_t, \Theta)
\]
\[
= \left[ \prod_{\tau=0}^{t-3} p(a_\tau|h_\tau, \Theta) \right] p(a_{t-2}|h_{t-2}, \Theta) p(a_{t-1}|h_{t-2}, a_{t-2}, o_{t-1}, o_t, \Theta)
\]
\[
= \left[ \prod_{\tau=0}^{t-3} p(a_\tau|h_\tau, \Theta) \right] p(a_{t-2}|h_{t-2}, o_{t-1}, \Theta)
\]
\[
\vdots
\]

81
where we have used $p(a_\tau|h_\tau, o_{\tau+1:t}) = p(a_\tau|h_\tau)$ and $h_0 = \text{null}$. The rightmost side of (4.5) is the observation-conditional probability of joint action-choosing at multiple time steps $\tau = 0, 1, \cdots, t$. Equation (4.5) can be verified directly by multiplying (4.4) over $\tau = 0, 1, \cdots, t$

\[
\prod_{\tau=0}^{t} p(a_\tau|h_\tau, \Theta)
\]

\[
= p(a_0|o_1, \Theta) \frac{p(a_0:1|o_1, \Theta) p(a_0:2|o_1:2, \Theta) \cdots p(a_{0:t-1}|o_{1:t-1}, \Theta)}{p(a_0:1|o_1, \Theta) p(a_0:2|o_1:2, \Theta) p(a_0:t-1|o_{1:t-1}, \Theta)}
\]

\[
= p(a_0:t|o_{1:t}, \Theta)
\] (4.6)

It should be noted that (4.2) is obtained by assuming all the actions up to $t$ have been drawn according to the RPR policy. In the case that some of the historical action result from following other policies that are independent of the RPR, we should modify the equation as

\[
p(a_0:t|o_{1:t}, \Theta) = \sum_{z_0, \cdots, z_t=1}^{|Z|} \left[ \mu(z_0) \left[ \pi(z_0, a_0) \right] \prod_{\tau=1}^{t} W(z_{\tau-1}, a_{\tau-1}, o_{\tau}, z_{\tau}) \left[ \pi(z_{\tau}, a_{\tau}) \right]^{\varsigma_{\tau}} \right]
\] (4.7)

where $\varsigma_{\tau} \in \{0, 1\}$ are indicator variables such that $\varsigma_{\tau} = 1$ if $a_\tau$ is drawn by the RPR policy and $\varsigma_{\tau} = 0$ if $a_\tau$ is drawn by some policy other than the RPR. Substituting the modification into (4.4) gives the history-dependent RPR policy when the historical actions are partly produced by other policies.

### 4.3 Empirical Learning of the RPR

We are interested in empirical learning of the RPR, based on the experiences (data) that are collected by an agent who interacts with the environment in question by fol-
following a policy. Assuming the interaction is episodic, i.e., it breaks into subsequences called episodes (Sutton and Barto 1998), we can represent the experiences by a set of episodes.

**Definition 6.** An *episode* is a sequence of agent-environment interactions terminated in an absorbing state that transits to itself with zero rewards (Sutton and Barto 1998). An episode is denoted by \((a_0^k r_0^k o_1^k a_1^k r_1^k \cdots o_T^k a_T^k r_T^k)\), where the subscripts are discrete times, \(k\) indexes the episodes, and \(o\), \(a\), and \(r\) are respectively observations, actions, and immediate rewards.

Without loss of generality, we assume in this chapter and Chapter 5 that the immediate rewards \(r\)'s are non-negative, which can always be achieved by adding a constant; this results in a constant added to the value function (the value function of a POMDP is linear in immediate reward) and does not influence the policy.

The objective in learning the RPR is to maximize, with respect to \(\Theta\), the expected sum of future reward accrued by following the policy in (4.4). We give an optimality criterion in Definition 7, and we then prove that the criterion approaches the expected sum of future reward, as the number of episodes approaches infinity.

**Definition 7.** Let \(D^{(K)} = \{(a_0^k r_0^k o_1^k a_1^k r_1^k \cdots o_T^k a_T^k r_T^k)\}_{k=1}^K\) be a set of episodes obtained by an agent who interacts with the environment by following policy \(\Pi\) to select actions, where \(\Pi\) is an arbitrary stochastic policy with action-selecting distributions \(p^\Pi(a_t|h_t) > 0, \forall \text{ action } a_t, \forall \text{ history } h_t\). The *RPR optimality criterion* is defined as

\[
\hat{V}(D^{(K)}; \Theta) \overset{\text{def.}}{=} \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \gamma^l r_t^k \prod_{\tau=0}^t p^\Pi(a^k_\tau|h^k_\tau) \prod_{\tau=0}^t p(a^k_\tau|h^k_\tau, \Theta) \tag{4.8}
\]

where \(h^k_t = a_0^k o_1^k a_1^k \cdots o_t^k\) is the history of actions and observations up to time \(t\) in the \(k\)-th episode, \(0 < \gamma < 1\) is the discount, and \(\Theta\) denotes the parameters of the RPR.
Theorem 8. Let $D^{(K)} = \{(a_{0}^{k}, \cdots, a_{t}^{k}, r_{t}^{k})\}_{k=1}^{K}$ be a set of episodes obtained by an agent who interacts with the environment by following $\Pi$ to choose actions, where $\Pi$ is an arbitrary stochastic policy whose action-selecting distribution $p^{\Pi}(a_{t}|h_{t}) > 0 \ \forall \ a_{t}$ and $\forall \ h_{t}$. Then

$$\lim_{K \to \infty} \hat{V}(D^{(K)}; \Theta) = \lim_{K \to \infty} \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_{k}} \frac{\gamma^{t} r_{t}^{k}}{\prod_{\tau=0}^{t} p^{\Pi}(a_{\tau}^{k}|h_{\tau}^{k})} \prod_{\tau=0}^{t} p(a_{\tau}^{k}|h_{\tau}^{k}, \Theta)$$

is the expected sum of discounted rewards (value function) by following the RPR policy parameterized by $\Theta$, over an infinite horizon.

Theorem 8 shows that the optimality criterion given in Definition 7 is indeed the true value function in the limiting case when there are infinite number of episodes. In reality, we can only have a finite set of episodes and base the learning on this finite set.

Now, the learning problem becomes maximization of $\hat{V}(D^{(K)}; \Theta)$ with respect to $\Theta$, for a given finite set of episodes $D^{(K)}$. Substituting (4.2) and (4.5) into (4.8), we obtain

$$\hat{V}(D^{(K)}; \Theta) = \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_{k}} \tilde{r}_{t}^{k} \sum_{z_{0}^{k}, \cdots, z_{t}^{k}=1} p(a_{0:t}^{k}, z_{0:t}^{k}|o_{1:t}^{k}, \Theta)$$

where

$$\tilde{r}_{t}^{k} = \frac{\gamma^{t} r_{t}^{k}}{\prod_{\tau=0}^{t} p^{\Pi}(a_{\tau}^{k}|h_{\tau}^{k})}$$

is introduced to simplify the notation.

Given the episodes $D^{(K)}$, either a single optimal RPR can be learned by maximizing the empirical value function (4.10) or a posterior distribution of the RPR...
can be learned following a Bayesian learning procedure. The $\Theta$ resulting from value maximization is called a Maximum-Value (MV) estimate.

A single maximum-value RPR is preferred when the number of episodes is large, in which case the single RPR is expected to approach the optimal. A posterior distribution is preferred when we the episodes are insufficient, in which case a single RPR is plagued by over-fitting and less robust, or when a prior is desired for transferring the RPR to a future environment.

In Section 4.3, we present the algorithm to maximize (4.10), with respect to $\Theta = \{\mu, W, \pi\}$, this giving the maximum-value estimate of $\Theta$. In Section 4.5, we present Bayesian learning for the RPR, which yields a posterior distribution of $\Theta$.

We assume $r_t \geq 0$, which can always be achieved by adding a constant to $r_t$; this results in a constant added to the value function (the value function of a POMDP is linear in immediate reward) and does not influence the policy.

### 4.4 Maximum-Value Estimation of the RPR

We develop an iterative procedure for value maximization, which is presented in the following theorem.

**Theorem 9.** Let

$$q^k_t(z^k_0:|\Theta^{(n)}) = \frac{\tilde{r}^k_t}{V(D(K); \Theta)} p(a^k_0:t, z^k_0:|o^k_1:t, \Theta(n))$$

(4.12)

for $z^k_0, z^k_1, \ldots, z^k = 1, 2, \ldots, |Z|$ and $k = 1, 2, \ldots, K$, and

$$\Theta^{(n+1)} = \arg \max_{\Theta \in \mathcal{X}} \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z^k_0, \ldots, z^k = 1} q^k_t(z^k_0:|\Theta^{(n)}) \ln \frac{\tilde{r}^k_t p(a^k_0:t, z^k_0:|o^k_1:t, \Theta(n))}{q^k_t(z^k_0:|\Theta^{(n)})}$$

(4.13)
where

\[ \mathcal{F} = \left\{ \Theta = (\mu, \pi, W) : \sum_{i=1}^{\mid Z \mid} \hat{\mu}(i) = 1, \sum_{a=1}^{\mid A \mid} \hat{\pi}(i, a) = 1, \sum_{j=1}^{\mid Z \mid} \hat{W}(i, a, o, j) = 1, \right. \]

\[ \left. j = 1, 2, \ldots, \mid Z \mid, a = 1, 2, \ldots, \mid A \mid, o = 1, 2, \ldots, \mid O \mid \right\} \quad (4.14) \]

is the set of feasible parameters for the RPR in question. Let \( \{ \Theta^{(0)}, \Theta^{(1)}, \ldots, \Theta^{(n)} \ldots \} \) be a sequence yielded by iteratively applying (4.12) and (4.13), staring from \( \Theta^{(0)} \). Then

\[ \lim_{n \to \infty} \hat{V}(D^{(K)}; \Theta^{(n)}) \]

exists and the limit is a maxima of \( \hat{V}(D^{(K)}; \Theta) \).

### 4.4.1 Calculation of \( \{ q^k_t(z^k_{0:t} | \Theta) \} \)

We rewrite (4.12) to get

\[ q^k_t(z^k_{0:t} | \Theta) = \frac{\hat{r}_t^k p(a^k_{0:t} | o^k_{1:t}, \Theta)}{\hat{V}(D^{(K)}; \Theta)} p(z^k_{0:t} | a^k_{0:t}, o^k_{1:t}, \Theta) \quad (4.15) \]

Only some marginals of \( p(z^k_{0:t} | a^k_{0:t}, o^k_{1:t}, \Theta) \) will be used in evaluating the objective function in (4.13), in particular,

\[ \xi^k_{t, \tau}(i, j) = p(z^k_{\tau} = i, z^k_{\tau+1} = j | a^k_{0:t}, o^k_{1:t}, \Theta) \quad (4.16) \]

\[ \phi^k_{t, \tau}(i) = p(z^k_{\tau} = i | a^k_{0:t}, o^k_{1:t}, \Theta) \quad (4.17) \]

The following lemma gives a factorized form of these marginal distributions which can be used to facilitate computation.
Lemma 10. Let

\[
\alpha^k_\tau(i) = p(z^k_\tau = i|a^k_0, o^k_1, \Theta) = \frac{p(z^k_\tau = i, a^k_0|o^k_1, \Theta)}{\prod_{\tau' = 0}^\tau p(a^k_{\tau'}, h^k_{\tau'}, \Theta)} \tag{4.18}
\]

\[
\beta^k_{t,\tau}(i) = \frac{p(a^k_{t+1}|z^k_\tau = i, o^k_{\tau+1}, \Theta)}{\prod_{\tau' = \tau}^t p(a^k_{\tau'}, h^k_{\tau'}, \Theta)} \tag{4.19}
\]

Then

\[
\xi^k_{t,\tau}(i, j) = \alpha^k_\tau(i)W(z^k_\tau = i, a^k_\tau, o^k_{\tau+1}, z^k_{\tau+1} = j)\pi(a^k_{\tau+1}|z^k_\tau = i)\beta^k_{\tau+1}(j) \tag{4.20}
\]

\[
\phi^k_{t,\tau}(i) = \alpha^k_\tau(i)\beta^k_{t,\tau}(i)p(a^k_\tau|h^k_{\tau+1}) \tag{4.21}
\]

The \( \alpha \) and \( \beta \) variables can be computed recursively as

\[
\alpha^k_\tau(i) = \begin{cases} 
\frac{\mu(z^k_0 = i)\pi(a^k_0|z^k_0 = i)}{p(a^k_0|h^k_0, \Theta)}, & \tau = 0 \\
\frac{\sum_{j=1}^{\vert Z \vert} \alpha^k_{\tau-1}(j)W(z^k_{\tau-1} = j, a^k_{\tau-1}, o^k_{\tau}, z^k_{\tau} = i)\pi(a^k_{\tau}|z^k_\tau = i)}{p(a^k_{\tau}|h^k_{\tau}, \Theta)}, & \tau > 0
\end{cases} \tag{4.22}
\]

\[
\beta^k_{t,\tau}(i) = \begin{cases} 
\frac{1}{p(a^k_\tau|h^k_{\tau}, \Theta)}, & \tau = t \\
\frac{\sum_{j=1}^{\vert Z \vert} W(z^k_\tau = i, a^k_{\tau}, o^k_{\tau+1}, z^k_{\tau+1} = j)\pi(z^k_{\tau+1} = j, a^k_{\tau+1})\beta^k_{\tau+1}(j)}{p(a^k_\tau|h^k_{\tau}, \Theta)}, & \tau < t
\end{cases} \tag{4.23}
\]

for \( t = 0, \cdots, T_k \) and \( k = 1, \cdots, K \).

Since \( \sum_{i=1}^{\vert Z \vert} \alpha^k_\tau(i) = 1 \), it follows from (4.22) that

\[
p(a^k_\tau|h^k_{\tau}, \Theta) = \begin{cases} 
\sum_{i=1}^{\vert Z \vert} \mu(z^k_0 = i)\pi(a^k_0|z^k_0 = i), & \tau = 0 \\
\sum_{i=1}^{\vert Z \vert} \sum_{j=1}^{\vert Z \vert} \alpha^k_{\tau-1}(j)W(z^k_{\tau-1} = j, a^k_{\tau-1}, o^k_{\tau}, z^k_{\tau} = i)\pi(a^k_{\tau}|z^k_\tau = i), & \tau > 0
\end{cases} \tag{4.24}
\]
4.4.2 Maximization of $\text{LB}(\hat{\Theta}|\Theta)$

We rewrite (4.63) to get

$$
\text{LB}(\hat{\Theta}|\Theta) = \frac{1}{K} \sum_{k=1}^{K} T_k \sum_{t=0}^{T_k} \sum_{|Z|} q^k_t(z_{0:t}^k|\Theta^{(n)}) \ln \frac{\hat{p}_t^k(a_{0:t}^k, z_{0:t}^k|\Theta^k, \hat{\Theta})}{q^k_t(z_{0:t}^k|\Theta^{(n)})} 
$$

$$
= \frac{1}{K} \sum_{k=1}^{K} T_k \sum_{t=0}^{T_k} \sum_{|Z|} q^k_t(z_{0:t}^k|\Theta^{(n)}) \ln p(a_{0:t}^k, z_{0:t}^k|\Theta^k, \hat{\Theta}) 
+ \text{constant} 
$$

where the “constant” collects all the terms irrelevant to $\hat{\Theta}$. Substituting (4.15) gives

$$
\text{LB}(\hat{\Theta}|\Theta) = \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{|Z|} \hat{p}_t^k(a_{0:t}^k|\Theta^k, \Theta) \left\{ \sum_{i=1}^{|Z|} \phi^k_{t,0}(i) \ln \hat{\mu}(i) + \sum_{\tau=1}^{t} \sum_{i=1}^{|Z|} \phi^k_{t,\tau}(i) \ln \hat{\pi}(i, a^k_\tau) 
+ \sum_{\tau=1}^{t} \sum_{i,j=1}^{|Z|} \xi^k_{t,\tau}(i, j) \ln \hat{W}(i, a^k_{\tau-1}, a^k_{\tau}, j) \right\} + \text{constant} 
$$

It is not difficult to show that $\hat{\Theta} = \arg\max_{\Theta \in \mathcal{F}} \text{LB}(\hat{\Theta}|\Theta)$ is given by

$$
\hat{\mu}(i) = \frac{\sum_{k=1}^{K} \sum_{t=0}^{T_k} \rho^k_t \phi^k_{t,0}(i)}{\sum_{i=1}^{\sum_{k=1}^{K} \sum_{t=0}^{T_k} \rho^k_t \phi^k_{t,0}(i)}} 
$$

$$
\hat{\pi}(i, a) = \frac{\sum_{k=1}^{K} \sum_{t=0}^{T_k} \rho^k_t \sum_{\tau=1}^{t} \phi^k_{t,\tau}(i) \delta(a^k_\tau, a)}{\sum_{a=1}^{\sum_{k=1}^{K} \sum_{t=0}^{T_k} \rho^k_t \sum_{\tau=1}^{t} \phi^k_{t,\tau}(i) \delta(a^k_\tau, a)}} 
$$

$$
\hat{w}(i, a, o, j) = \frac{\sum_{k=1}^{K} \sum_{t=0}^{T_k} \rho^k_t \sum_{\tau=1}^{t-1} \xi^k_{t,\tau}(i, j) \delta(a^k_\tau, a) \delta(o^k_{\tau+1}, o)}{\sum_{j=1}^{\sum_{k=1}^{K} \sum_{t=0}^{T_k} \rho^k_t \sum_{\tau=1}^{t-1} \xi^k_{t,\tau}(i, j) \delta(a^k_\tau, a) \delta(o^k_{\tau+1}, o)}} 
$$

for $i, j = 1, 2, \ldots, |Z|$, $a = 1, \ldots, |A|$, and $o = 1, \ldots, |O|$, where

$$
\delta(a, b) = \begin{cases} 
1, & a = b \\
0, & a \neq b 
\end{cases}
$$
and

$$ρ_t^k = \tilde{r}_t^k p(a_{0:t}^k | o_{1:t}, \Theta) = \frac{\gamma_t r_t^k}{\prod_{\tau=0}^{t} p(h_{\tau}^k | h_{\tau}^k, \Theta)} \prod_{\tau=0}^{t} p(h_{\tau}^k | h_{\tau}^k, \Theta)$$

$$= \gamma_t r_t^k \prod_{\tau=0}^{t} \frac{p(h_{\tau}^k | h_{\tau}^k, \Theta)}{p^{\Pi}(a_{0:t}^k | h_{\tau}^k)} \tag{4.30}$$

where $p(h_{\tau}^k | h_{\tau}^k, \Theta)$ is computed from the $\alpha$ variables by (4.24).

Given a history of actions and observations $(a_{0:T-1}, o_{1:T})$ collected up to time step $T$, the single RPR yields a distribution of $a_T$ as given by (4.4). The optimal choice for $a_T$ can be obtained by either sampling from this distribution or taking the action that maximizes the probability.

### 4.5 Variational Bayesian Learning of the RPR

In the Bayesian framework, we assume a prior distribution $p(\Theta)$ of $\Theta$ and aim to compute the posterior distribution of $\Theta$. What is unusual here is that we lack a likelihood function. In order to proceed, we let the empirical value function in (4.8) take the place of the usual likelihood function. Thus assumed, we take the expectation of (4.10) with respect to the prior $p(\Theta)$, and obtain

$$\hat{V}(D^{(K)}) = \int p(\Theta) \hat{V}(D^{(K)}; \Theta) \, d\Theta$$

$$= \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} |\mathcal{Z}| \sum_{z_{0,t}^k, \cdots, z_{k-1,t}^k=1} p(\Theta) p(a_{0,t}^k, z_{0,t}^k | o_{1:t}, \Theta) \, d\Theta \tag{4.31}$$

We now use a variational method to find an approximate posterior of $\Theta$. For any $q_t^k(z_{0,t}^k) \geq 1, k = 1, \cdots, K$, and $g(\Theta) \geq 1$ that satisfy $\int g(\Theta) d\Theta = 1$ and

$$\frac{1}{K} \sum_{k=1}^{K} \sum_{t=1}^{T_k} \sum_{z_{0,t}^k, \cdots, z_{k-1,t}^k=1} q_t^k(z_{0,t}^k) = 1 \tag{4.32}$$
we have

\[
\ln \hat{V}(\mathcal{D}^{(K)}) = \ln \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z_0^t, \ldots, z_t^t = 1} |Z| \int q_t^k(z_0^t)g(\Theta) \frac{\hat{r}_t^k p(\Theta) p(a_{0,t}^k, z_{0:t}^k | o_{1:t}^k, \Theta)}{q_t^k(z_0^t)g(\Theta)} d\Theta
\]

\[
\geq \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z_0^t, \ldots, z_t^t = 1} |Z| \int q_t^k(z_0^t)g(\Theta) \ln \frac{\hat{r}_t^k p(\Theta) p(a_{0,t}^k, z_{0:t}^k | o_{1:t}^k, \Theta)}{q_t^k(z_0^t)g(\Theta)} d\Theta
\]

\[
= \ln \hat{V}(\mathcal{D}^{(K)}) - \text{KL} \left( \{ q_t^k(z_0^t)g(\Theta) \} \parallel \left\{ \frac{\hat{r}_t^k p(\Theta) p(a_{0,t}^k, z_{0:t}^k | o_{1:t}^k, \Theta)}{\hat{V}(\mathcal{D}^{(K)})} \right\} \right)
\]

\[
= \text{LB} \left( \{ q_t^k \}, g(\Theta) \right) \tag{4.33}
\]

where

\[
\text{KL} \left( \{ q_t^k(z_0^t)g(\Theta) \} \parallel \left\{ \frac{\hat{r}_t^k p(\Theta) p(a_{0,t}^k, z_{0:t}^k | o_{1:t}^k, \Theta)}{\hat{V}(\mathcal{D}^{(K)})} \right\} \right)
\]

\[
= \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z_0^t, \ldots, z_t^t = 1} |Z| \int q_t^k(z_0^t)g(\Theta) \ln \frac{q_t^k(z_0^t)g(\Theta)}{\hat{r}_t^k p(\Theta) p(a_{0,t}^k, z_{0:t}^k | o_{1:t}^k, \Theta)} d\Theta \tag{4.34}
\]

and \(\text{KL}(q||p)\) denotes the Kullback-Leibler (KL) distance between two probability distributions. Because

\[
\frac{\hat{r}_t^k p(\Theta) p(a_{0,t}^k, z_{0:t}^k | o_{1:t}^k, \Theta)}{\hat{V}(\mathcal{D}^{(K)})} = \frac{\hat{r}_t^k p(\Theta) p(a_{0,t}^k, z_{0:t}^k | o_{1:t}^k, \Theta)}{\hat{V}(\mathcal{D}^{(K)})} = \frac{\hat{r}_t^k p(a_{0,t}^k | o_{1:t}^k)}{\hat{V}(\mathcal{D}^{(K)})} p(z_{0:t}, \Theta | a_{0,t}^k, o_{1:t}^k) \tag{4.35}
\]

we can rewrite the lower bound in (4.33) as

\[
\text{LB} \left( \{ q_t^k \}, g(\Theta) \right)
\]

\[
= \ln \hat{V}(\mathcal{D}^{(K)}) - \text{KL} \left( \{ q_t^k(z_0^t)g(\Theta) \} \parallel \left\{ \frac{\hat{r}_t^k p(a_{0,t}^k | o_{1:t}^k)}{\hat{V}(\mathcal{D}^{(K)})} p(z_{0:t}, \Theta | a_{0,t}^k, o_{1:t}^k) \right\} \right) \tag{4.36}
\]

90
Since \( \ln\hat{V}(D^{(K)}) \) is independent of \( \Theta \), it is clear that maximization of the lower bound \( \text{LB}\left(\{q_k^k\}, g(\Theta)\right) \) is equivalent to minimization of the KL distance between the variational distributions \( \{q_k^k(z_{0:t})g(\Theta)\} \) and the weighted posterior distribution

\[
\left\{ \frac{r_t^k p(a_{0:t}^k|o_{1:t}^k)}{V(D^{(K)})} p(z_{0:t}^k, \Theta|a_{0:t}^k, o_{1:t}^k) \right\},
\]

where the weight manifest the value associated with each action sequence.

The lower bound maximization is accomplished by solving \( \{q_k^k\}_{k=1}^{K} \) and \( q_\Theta \) alternatively, keeping one fixed while solving for the other.

**Theorem 11.** Iteratively applying the following two equations produces a sequence of monotonically increasing lower bounds \( \text{LB}\left(\{q_k^k\}, g(\Theta)\right) \), which converges to a maxima.

\[
q_t^k(z_{0:t}) = C_z^{-1} \tilde{r}_t^k \exp \left\{ \int g(\Theta) \ln p(a_{0:t}^k, z_{0:t}^k|o_{1:t}^k, \Theta) d\Theta \right\} \tag{4.37}
\]

\[
g(\Theta) = C_\Theta^{-1} p(\Theta) \exp \left\{ \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z_{0:t}^k} q_t^k(z_{0:t}^k) \ln \tilde{r}_t^k p(a_{0:t}^k, z_{0:t}^k|o_{1:t}^k, \Theta) \right\} \tag{4.38}
\]

where

\[
C_z = \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z_{0:t}^k} \tilde{r}_t^k \exp \left\{ \int g(\Theta) \ln p(a_{0:t}^k, z_{0:t}^k|o_{1:t}^k, \Theta) d\Theta \right\} \tag{4.39}
\]

\[
C_\Theta = \int p(\Theta) \exp \left\{ \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z_{0:t}^k} q_t^k(z_{0:t}^k) \ln \tilde{r}_t^k p(a_{0:t}^k, z_{0:t}^k|o_{1:t}^k, \Theta) \right\} d\Theta \tag{4.40}
\]

are normalization constants such that

\[
\frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z_{0:t}^k} q_t^k(z_{0:t}^k) = 1 \text{ and } \int g(\Theta) d\Theta = 1.
\]
4.5.1 Calculation of \( \{ q_k^k(z_{0,t}^k) \} \)

Our choice of the prior distribution for \( \Theta \) is specified as follows,

\[
p(\Theta) = p(\mu|\nu)p(\pi|\rho)p(W|\omega)
\]

Our choice of the prior distribution for \( \Theta \) is specified as follows,

\[
p(\Theta) = p(\mu|\nu)p(\pi|\rho)p(W|\omega)
\]

\[ p(\mu|\nu) = \text{Dir}\left( \mu(1), \ldots, \mu(|Z|) \big| \nu \right) \quad (4.41) \]

\[ p(\pi|\rho) = \prod_{i=1}^{[A]} \text{Dir}\left( \pi(i,1), \ldots, \pi(i,|A|) \big| \rho_i \right) \quad (4.42) \]

\[ p(W|\omega) = \prod_{a=1}^{[A]} \prod_{o=1}^{[O]} \prod_{i=1}^{[Z]} \text{Dir}\left( W(i,a,o,1), \ldots, W(i,a,o,|Z|) \big| \omega_{i,a,o} \right) \quad (4.43) \]

where \( \text{Dir} \) denotes a Dirichlet distribution, \( \rho_i = \{ \rho_{i,m} \}_{m=1}^{[A]} \), \( \nu = \{ \nu_i \}_{i=1}^{[Z]} \), and \( \omega_{i,a,o} = \{ \omega_{i,a,o,j} \}_{j=1}^{[Z]} \). Initially, we let \( g(\Theta) = p(\Theta) \), substituting which into (4.37), we obtain

\[
q_k^k(z_{0,t}^k) = \frac{\hat{r}_k^k}{C_z} \exp \left\{ \langle \ln \pi(z_t^k, a_t^k) \rangle_{g(\pi)} + \langle \ln \mu(z_0^k) \rangle_{g(\mu)} + \sum_{\tau=1}^{t} \langle \ln W(z_{\tau-1}^k, a_{\tau-1}^k, o_{\tau-1}^k, z_0^k) \rangle_{g(W)} \right\}
\]

\[ = \frac{\hat{r}_k^k}{C_z} \tilde{\pi}(z_t^k, a_t^k) \tilde{\mu}(z_0^k) \prod_{\tau=1}^{t} \tilde{W}(z_{\tau-1}^k, a_{\tau-1}^k, o_{\tau-1}^k, z_0^k) \quad (4.45) \]

where \( \langle \cdot \rangle_{g(\mu)} \) denotes taking expectation with respect to \( g(\mu) \), etc, and

\[
\tilde{\pi}(i, m) = \exp \left\{ \langle \ln \pi(i, m) \rangle_{g(\pi)} \right\} = \exp \left\{ \psi(\rho_{i,m}) - \psi \left( \sum_{m=1}^{[A]} \rho_{i,m} \right) \right\} \quad (4.46)
\]

\[
\tilde{\mu}(i) = \exp \left\{ \langle \ln \mu(i) \rangle_{g(\mu)} \right\} = \exp \left\{ \psi(\nu_i) - \psi \left( \sum_{i=1}^{[Z]} \nu_i \right) \right\} \quad (4.47)
\]

\[
\tilde{W}(i,a,o,j) = \exp \left\{ \langle \ln W(i,a,o,j) \rangle_{g(W)} \right\}
\]

92
\[
\begin{align*}
\psi(\omega_{i,a,o,j}) - \psi(\sum_{j=1}^{[A]} \omega_{i,a,o,j}) \\
= \exp \left\{ \psi(\omega_{i,a,o,j}) - \psi(\sum_{j=1}^{[A]} \omega_{i,a,o,j}) \right\} \tag{4.48}
\end{align*}
\]

where \(\psi(\cdot)\) is the digamma function (Beal 2003). Using (4.1), we can write (4.45) equivalently as

\[
q_k^k(z_{0,t}) = q_k^k(z_{0,t} | \tilde{\Theta}) = \exp \left\{ \sum_{k=1}^{K} \sum_{t=0}^{T_k} \gamma_k^k(i, j) \ln W(i, a_k^k, o_k^k, j) \right\} \tag{4.49}
\]

which has the same form as (4.15) by replacing \(\Theta\) with \(\tilde{\Theta} = \{\tilde{\pi}, \tilde{\mu}, \tilde{W}\}\). Therefore we can use (4.20), (4.21), (4.22), (4.23), (4.24) to compute the \(\gamma\) and \(\xi\) variables.

### 4.5.2 Calculation of \(g(\Theta)\)

Substituting (4.1) and (4.49) into (4.38), employing (4.20), (4.21), (4.22), (4.23), (4.24), we obtain

\[
g(\Theta) = \frac{p(\Theta)}{C_\Theta} \exp \left\{ \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \gamma_k^k(i, j) \ln \mu(i) \right. \\
+ \sum_{\tau=1}^{t} \sum_{i=1}^{[Z]} \phi_{k,i}^{\tau}(i) \ln \pi(i, a_k^\tau) + \sum_{\tau=1}^{t} \sum_{i,j=1}^{[Z]} \xi_{k,i}^{\tau}(i, j) \ln W(i, a_k^\tau, o_k^\tau, j) \left. \right\} \\
= \frac{p(\Theta)}{C_\Theta} \prod_{k=1}^{K} \prod_{t=0}^{T_k} \left\{ \prod_{i=1}^{[Z]} [\mu(i)]^{\sigma_{k,i}^{\gamma,0}(i)} \prod_{\tau=1}^{t} \prod_{i,j=1}^{[Z]} [\pi(i, a_k^\tau)]^{\sigma_{k,i}^{\gamma,\gamma}(i)} \right. \\
\times \prod_{\tau=1}^{t} \prod_{i,j=1}^{[Z]} [W(i, a_k^\tau, o_k^\tau, j)]^{\sigma_{k,i}^{\gamma,\gamma}(i,j)} \left. \right\} \\
= p(\mu | \hat{\theta}) p(\pi | \hat{\rho}) p(W | \hat{\Sigma}) \tag{4.50}
\]
where

\[
\sigma_{k}^{t} = \frac{\hat{r}^{k}_{t}p(a_{0:t}^{k}|a_{t}^{k}, \Theta)}{V(D^{(K)}; \Theta)} = \frac{\gamma_{k}^{t}r_{k}^{t}}{P_{\Theta}(\Theta)} \prod_{t=0}^{T_{k}} p(h_{t}^{k}|h_{t}^{k}, \Theta) \prod_{t=0}^{T_{k}} p(\hat{h}_{t}^{k}|h_{t}^{k}) \prod_{t=0}^{T_{k}} p(h_{t}^{k}|h_{t}^{k}, \Theta)
\]

\[
\hat{V}(D^{(K)}; \Theta) = \frac{\gamma_{k}^{t}r_{k}^{t}}{P_{\Theta}(\Theta)} \prod_{t=0}^{T_{k}} p(h_{t}^{k}|h_{t}^{k}, \Theta) \prod_{t=0}^{T_{k}} p(h_{t}^{k}|h_{t}^{k}, \Theta)
\]

(4.51)

and \(p(\mu|\hat{\nu}), p(\pi|\hat{\rho}), p(W|\hat{\omega})\) are the same as in (4.42), (4.43), (4.44), respectively, with the hyper-parameters given by

\[
\hat{\nu}_{i} = v_{i} + \sum_{k=1}^{K} \sum_{t=0}^{T_{k}} \sigma_{k}^{t} \gamma_{k}^{t}(i)
\]

(4.52)

\[
\hat{\rho}_{i,a} = \rho_{i,a} + \sum_{k=1}^{K} \sum_{t=0}^{T_{k}} \sigma_{k}^{t} \gamma_{k}^{t}(i) \delta(a_{t}^{k}, a)
\]

(4.53)

\[
\hat{\omega}_{i,a,o,j} = \omega_{i,a,o,j} + \sum_{k=1}^{K} \sum_{t=0}^{T_{k}} \sigma_{k}^{t} \xi_{k}^{t}(i,j) \delta(a_{t-1}^{k}, a) \delta(o_{t}^{k}, o)
\]

(4.54)

### 4.5.3 Computation of Lower Bound

The lower bound \(\text{LB}(\{q_{k}^{t}\}_{k=1}^{K}, q_{\Theta})\) is required to check the convergence of the variational algorithm. Taking the logarithm of (4.37), we obtain

\[
\ln q_{t}^{k}(\gamma_{0:t}^{k}) = \ln C_{z}^{-1} r_{t}^{k} \exp \left\{ \int g(\Theta) \ln p(a_{0:t}^{k}, \gamma_{0:t}^{k}|\Theta) d\Theta \right\}
\]

\[
= -\ln C_{z} + \int g(\Theta) \ln r_{t}^{k} p(a_{0:t}^{k}, \gamma_{0:t}^{k}|a_{0:t}^{k}, \Theta) d\Theta
\]

(4.55)

which is substituted into the right side of (4.67) in the appendix (Section 4.8.4) to cancel the first term, yielding

\[
\text{LB} \left( \{ q_{k}^{t} \} , g(\Theta) \right) = \ln C_{z} - \int g(\Theta) \ln \frac{g(\Theta)}{p(\Theta)} d\Theta
\]

94
\[ \ln C_z - \text{KL}\left(g(\Theta) || p(\Theta)\right) \]  

Given a history of actions and observations \((a_{0:T-1}, o_{1:T})\) collected up to time step \(T\), one has two options in computing the optimal action \(a_T\) from the ensemble of RPRs. In the first one computes the posterior mean \(\overline{\Theta} = \int \theta q(\theta) d\theta\) and treats it as a single RPR, computing \(p(a_T|a_{0:T-1}, o_{1:T}, \overline{\Theta})\) by using (4.4). In the second, one computes the marginal likelihood \(p(a_T|a_{0:T-1}, o_{1:T}) = \int p(a_T|a_{0:T-1}, o_{1:T}, \Theta)q(\Theta) d\Theta\), by integrating out the RPR parameters. In either case, the optimal choice for \(a_T\) can be obtained by sampling from the distribution or taking the maximum probability. The integration \(\int p(a_T|a_{0:T-1}, o_{1:T}, \Theta)q(\Theta) d\Theta\) can be computed by applying the VB algorithm above to a single decision example \((R_T|a_T|a_{0:T-1}, o_{1:T})\) with \(R_T = 1\). Since the posterior of a single example is approximately equal to the prior, the lower bound reduces to \(\text{LB} \approx \ln C_1\), which yields \(\int p(a_T|a_{0:T-1}, o_{1:T}, \Theta)q(\Theta) d\Theta \approx p(a_T|a_{0:T-1}, o_{1:T}, \Theta)\), with \(\Theta\) given by (4.46), (4.47), and (4.48).

4.6 Experimental Results

We consider the benchmark examples Hallway and Hallway2 (Littman et al. 1995), which have been described in Section 2.4. Though the Hallway and Hallway2 problems were originally designed to test algorithms based on a given POMDP model, and they have recently been employed as benchmark domains for testing model-free reinforcement algorithms (Bakker 2004; Wierstra and Wiering 2004). We follow the standard procedure to test the performance of an algorithm on Hallway and Hallway2 — i.e., a total of 251 independent trials are performed, and each trial is terminated when either the goal is reached or a maximum budget of 251 steps is consumed. For each trial, three performance measures are recorded: (a) the discounted accumulative reward, i.e., the sum of exponentially discounted rewards received over the \(N_{te} \leq 251\)
steps (b) the goal rate, which is the percentage of the 251 trials that have reached
the goal (c) the median number of the steps that have actually been taken in the 251
trials.

4.6.1 Performance as a Function of Number of Episodes

First we investigate the performance of the RPR, as a function of the number of
episodes used in the learning. We learn a RPR from a set of $K$ episodes $D^{(K)}$
that are generated by following a policy $\Pi$, and the learning follows the procedures
described in Section 4.4. The conditions for the policy $\Pi$, as given in Theorem 8,
are very mild, and are satisfied by even a random policy. However, a purely random
policy produces long episodes (it takes a long time for a random agent to reach the
goal), which makes the learning very slow. To accelerate learning, we here use a semi-
random policy $\Pi$, which is simulated by the rule that, with probability 0.5, $\Pi$ uses the
PBVI algorithm (Pineau et al. 2003) to choose the action and, with probability 0.5,$\Pi$
chooses the action by sampling uniformly from $\mathcal{A}$. Since PBVI is an approximately
good policy, the semi-random $\Pi$ thus simulated ensures reaching the goal within a
reasonable number of steps.

Each $D^{(K)}$ produces a RPR, which is then evaluated following the standard testing
procedure. The number of decision states $|\mathcal{Z}|$ can be selected via the marginal value
function $\hat{V}(D^{(K)})$ in (4.31), which results from an integration over $\Theta$ and therefore
serves as an evidence of how good the choice of $|\mathcal{Z}|$ fits to the episodes. According
to the Occam Razor principle, the minimum $|\mathcal{Z}|$ fitting the episodes has the best
generalization. In practice, however, we find that the performance of the RPR is
robust to the choice of $|\mathcal{Z}|$. This is because learning of the RPR is a process of
allocating counts over $z$ — as more $z$’s are included, they simply share the counts
that otherwise would have been allocated to a single $z$. Provided the sharing of counts
is consistent with $\mu$, $\pi$, and $W$, the policy will not change.

The results on Hallway and Hallway2 are summarized in Figures 4.1 and 4.2, where we present each of the three performance measures plus the learning time, as a function of $\log_{10}$ of the number of episodes $K$ used in the learning. Each curve results from an average over 20 independently generated $D^{(K)}$ and the error bars show the standard deviations. Since the numbers $K$ are distributed non-uniformly, we have used the logarithmic values of $K$ in the horizontal axis.

It is clearly shown in Figures 4.1 and 4.2 that the performance of the RPR becomes better and better as the number of episodes $K$ used to learn it increases; moreover, the error bars shrinks with $K$, indicating increased certainty about the performance; the time increase since the algorithm has more data to process. As recalled from Theorem 8, the empirical value function $\hat{V}(D^{(K)}; \Theta)$ approaches the exact value function as $K$ goes to infinity. Assuming the RPR has enough memory (decision states) and the algorithm finds the global maxima, the RPR will approach the optimal policy as $K$ increases. Therefore, Figures 4.1 and 4.2 serves as an experimental verification of Theorem 8.

4.6.2 Comparison to Other Algorithms

To demonstrate the competitiveness of the RPR, we compare its performance against those of two state-of-the-art reinforcement learning algorithms, namely, RL-LSTM (Bakker 2004) and UDHMM (Wierstra and Wiering 2004). The comparison is made in terms of goal rate and number of steps to the goal, for which the results of RL-LSTM and UDHMM are available from the literature.

The comparison is summarized in Table 4.1, where the results of the RPR is taken from the 16-th point in figure 4.2 and the results of other algorithms are cited from (Loch and Singh 1998; Bakker 2004; Wierstra and Wiering 2004). It is seen that the
Figure 4.1: Results of the RPR on the Hallway problem. The horizontal axis is \( \log_{10}(K) \) of the number of episodes \( K \) used in learning the RPR. The horizontal axis in each sub-figure is (a) Goal rate (b) Discounted accumulative reward (c) Number of steps to reach the goal (d) Time in seconds for learning the RPR.

RPR outperforms the most best competing algorithms, although the UDHMM used more than 450 episodes, while the RPR used 355 episodes.

4.7 Conclusions on RPR

In this chapter we have presented the regionalized policy representation (RPR) to yield a history-dependent stochastic policy for environments characterized by a partially observable Markov decision process (POMDP). The RPR is learned in a model-
free manner, based on a set of experiences collected through interaction with the environment. We have developed algorithms for learning the maximum-value RPR and the variational posterior PRP. The latter offers the ability for selecting the number of decision states based on the Occam Razor principle and potentially more robustness when the experiences are meager. The results demonstrated on benchmark problems show that the RPR is a powerful model-free policy representation that yields policies competitive with those of state-of-the-art algorithms.
Table 4.1: A comparison of the RPR to other reinforcement learning algorithms on Hallway2

<table>
<thead>
<tr>
<th>Method</th>
<th>Goal (%)</th>
<th>Median Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Walk</td>
<td>26</td>
<td>&gt; 251</td>
</tr>
<tr>
<td>SARSA(λ) [Loch and Singh 1998]</td>
<td>77</td>
<td>73</td>
</tr>
<tr>
<td>RL-LSTM [Bakker 2004]</td>
<td>94</td>
<td>61</td>
</tr>
<tr>
<td>UDHMM [Wierstra and Wiering 2004]</td>
<td>92</td>
<td>62</td>
</tr>
<tr>
<td>RPR</td>
<td>97</td>
<td>46</td>
</tr>
</tbody>
</table>

4.8 Appendix

4.8.1 Proof of Theorem 8

According to [Kaelbling et al. 1998], the expected sum of exponentially discounted reward (value function) over an infinite horizon can be written as

\[ V = E \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right] \]  

where \(0 < \gamma < 1\) is the discount factor. Let \(E\) denote the environment in question and \(\mathcal{P}_E\) the corresponding probabilistic model (POMDP). Let \(\Theta\) be the parameters specifying the the RPR, the expectation in our situation here is \(E_{\text{episodes}|E,\Theta}\). Thus

\[
V = E_{\text{episodes}|E,\Theta} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right] 
= \sum_{a_0,a_1,a_2 \cdots} p(a_0) p(a_1) \cdots p(r_0|a_0,a_1,a_2 \cdots) \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right] 
= \sum_{a_0,a_1,\cdots} p(a_0,a_1,\cdots|\Theta) \mathcal{E}_{r_0,r_1,r_2,\cdots|a_0,a_1,\cdots} \mathcal{P}_E \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right]
\]
where

\[ \text{where} \]

We begin our derivation by writing the empirical value function in its logarithm

\[ \text{(Importance sampling (Robert and Casella 1999))} \]

\[
\sum_{a_0,a_1 \cdots} p^\Pi(a_0a_1 \cdots)p(a_0a_1 \cdots | \Theta) E_{r_{0a_1}r_{1a_2} \cdots \sim \mathcal{P}_E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right]
\]

\[
= \sum_{a_0,a_1 \cdots} p^\Pi(a_0a_1 \cdots) E_{r_{0a_1}r_{1a_2} \cdots \sim \mathcal{P}_E} \left[ \sum_{t=0}^{\infty} \frac{\gamma^t r_t}{\prod_{\tau=0}^{t} p^{\Pi}(a_{\tau} | h_{\tau})} \prod_{\tau=0}^{t} p(a_{\tau} | h_{\tau}, \Theta) \right]
\]

\[
= E_{a_0,a_1 \cdots \sim p^\Pi, r_{0a_1}r_{1a_2} \cdots \sim \mathcal{P}_E} \left[ \sum_{t=0}^{\infty} \frac{\gamma^t r_t}{\prod_{\tau=0}^{t} p^{\Pi}(a_{\tau} | h_{\tau})} \prod_{\tau=0}^{t} p(a_{\tau} | h_{\tau}, \Theta) \right]
\]

\[
\lim_{K \to \infty} \frac{1}{K} \sum_{k=1}^{K} \left[ \sum_{t=0}^{T_k} \frac{\gamma^t r_t}{\prod_{\tau=0}^{t} p^{\Pi}(a_{\tau} | h_{\tau})} \prod_{\tau=0}^{t} p(a_{\tau} | h_{\tau}, \Theta) \right] = \lim_{K \to \infty} \hat{V}(D^{(K)}; \Theta) \quad (4.58)
\]

where the last equality results because \( r_t^k = 0 \) for \( t > T_k \) according to Definition 6.

Q.E.D.

4.8.2 Proof of Theorem 9

We begin our derivation by writing the empirical value function in its logarithm

\[
\ln \hat{V}(D^{(K)}; \Theta) = \ln \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \prod_{i=1}^{\mid Z \mid} p(a_{0:t}, z_{0:t}^k | a_{1:t}, \Theta)
\]

\[
= \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z_{0:t}^k} \prod_{i=1}^{\mid Z \mid} q_{t}^k(z_{0:t}^k) \frac{a_{0:t}, z_{0:t}^k | a_{1:t}, \Theta}{q_{t}^k(z_{0:t}^k)} \quad (4.59)
\]

where

\[
q_{t}^k(z_{0:t}^k) \geq 0
\]

\[
\frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z_{0:t}^k} q_{t}^k(z_{0:t}^k) = 1 \quad (4.60)
\]
Applying Jensen’s inequality to (4.59), we obtain

\[
\ln \hat{V}(D^{(K)}; \Theta) \geq \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z^k_0, \ldots, z^k_T=1}^{|Z|} q^k_t(z^k_0; \Theta) \ln \frac{\hat{r}^k_t p(a^k_{0:t}, z^k_{0:t}|a^k_{1:t}, \Theta)}{q^k_t(z^k_0; \Theta)}
\]  

(4.61)

The lower bound is maximized when

\[
q^k_t(z^k_0; \Theta) = \frac{\hat{r}^k_t}{\hat{V}(D^{(K)}; \hat{\Theta})} p(a^k_{0:t}, z^k_{0:t}|a^k_{1:t}, \hat{\Theta})
\]

(4.62)

which turns the inequality into an equality. Define

\[
\text{LB}(\hat{\Theta}|\Theta) = \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z^k_0, \ldots, z^k_T=1}^{|Z|} q^k_t(z^k_0; \Theta) \ln \frac{\hat{r}^k_t p(a^k_{0:t}, z^k_{0:t}|a^k_{1:t}, \hat{\Theta})}{q^k_t(z^k_0; \Theta)}
\]

(4.63)

By (4.61), \( \text{LB}(\hat{\Theta}|\Theta) \leq \text{LB}(\hat{\Theta}|\hat{\Theta}) = \ln \hat{V}(D^{(K)}; \hat{\Theta}) \) holds for any \( \Theta \) and \( \hat{\Theta} \). Therefore, when \( \hat{\Theta} = \arg \max_{\Theta \in \mathcal{F}} \text{LB}(\hat{\Theta}|\Theta) \), we have

\[
\ln \hat{V}(D^{(K)}; \Theta) = \text{LB}(\Theta|\Theta) \leq \text{LB}(\hat{\Theta}|\Theta) \leq \text{LB}(\hat{\Theta}|\hat{\Theta}) = \ln \hat{V}(D^{(K)}; \hat{\Theta})
\]

Starting from \( \Theta^{(0)} \) we compute

\[
\Theta^{(1)} = \arg \max_{\Theta \in \mathcal{F}} \text{LB}(\hat{\Theta}|\Theta^{(0)})
\]

\[
\Theta^{(2)} = \arg \max_{\Theta \in \mathcal{F}} \text{LB}(\hat{\Theta}|\Theta^{(1)})
\]

\[\vdots\]

which satisfy \( \hat{V}(D^{(K)}; \Theta^{(0)}) \leq \hat{V}(D^{(K)}; \Theta^{(1)}) \leq \hat{V}(D^{(K)}; \Theta^{(2)}) \leq \cdots \). Since the value function is upper bounded, this monotonically increasing sequence must converge, which happens at a maxima of \( \hat{V}(D^{(K)}; \Theta) \). Q.E.D.
4.8.3 Proof of Lemma

Substituting (4.18) and (4.19), we have

$$\text{Right side of (4.20)} = \frac{p(z_j^k = i, z_{j+1}^k = j, a_{0t}^k|o_{1t}^k, \Theta)}{\prod_{\tau_r=0} p(a_{\tau_r}^k|h_{\tau_r}^k)}$$

(4.64)

Since the denominator is equal to \(p(a_{0t}^k|o_{1t}^k, \Theta)\) by (4.5), we have

$$\text{Right side of (4.20)} = p(z_j^k = i, z_{j+1}^k = j|a_{0t}^k, o_{1t}^k, \Theta)$$

= \(\xi_{i,j}^k(t, \Theta)\)

(4.65)

Similarly,

$$\text{Right side of (4.21)} = p(z_j^k = i, a_{0t}^k|o_{1t}^k, \Theta)$$

= \(p(z_j^k = i|a_{0t}^k, o_{1t}^k, \Theta)\)

= \(\phi_{i,j}^k(t, \Theta)\)

(4.66)

This completes the proof.

Q.E.D.

4.8.4 Proof of Theorem

We rewrite the lower bound in (4.33) as

$$\text{LB} \left( \{ q_{\Theta}^k \}, g(\Theta) \right) = \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{\sum_{z_0^k, \ldots, z_{t-1}^k}^{Z}} q_{\Theta}^k(z_0^k, \ldots, z_{t-1}^k) g(\Theta) \ln q_{\Theta}^k(z_0^k, \ldots, z_{t-1}^k, \Theta) d\Theta$$

$$- \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{\sum_{z_0^k, \ldots, z_{t-1}^k}^{Z}} q_{\Theta}^k(z_0^k, \ldots, z_{t-1}^k) \ln q_{\Theta}^k(z_0^k, \ldots, z_{t-1}^k)$$

$$- \int g(\Theta) \ln \frac{g(\Theta)}{p(\Theta)} d\Theta$$

(4.67)

We alternatively find the \(q\) and \(g\) that maximizes the lower bound, keeping one fixed while finding the other.
Keeping $g$ fixed, we solve $\max_q LB \left( \{ q^k_t \}, g(Θ) \right)$ subject to the constraints in (4.32). We construct the Lagrangian
\[
\ell_q = LB \left( \{ q^k_t \}, g(Θ) \right) - λ \left( 1 - \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z_0^k, \cdots, z_t^k=1}^{\mid Z \mid} q^k_t(z^k_{0:t}) \right) \quad (4.68)
\]
Differentiating $\ell(q)$ with respect to $q$ and setting the result to zero, we obtain
\[
\frac{∂\ell_q}{∂(q^k_t(z^k_{0:t}))} = \int g(Θ) \ln \tilde{r}^k_t p(a^k_{0:t}, z^k_{0:t} | q^k_{1:t}, Θ) dΘ - \ln q^k_t(z^k_{0:t}) - 1 + λ
\]
which is solved to give
\[
q^k_t(z^k_{0:t}) = e^{λ-1} \tilde{r}^k_t \exp \left\{ \int g(Θ) \ln p(a^k_{0:t}, z^k_{0:t} | q^k_{1:t}, Θ) dΘ \right\} \quad (4.70)
\]
By using the constraints in (4.32), we arrive at (4.37) with $e^{λ-1} = C_z^{-1}$.

Keeping $q$ fixed, we solve $\max_g LB \left( \{ q^k_t \}, g(Θ) \right)$ subject to the constraint that $\int g(Θ)dΘ = 1$. Construct the Lagrangian
\[
\ell_g = LB \left( \{ q^k_t \}, g(Θ) \right) - λ \left( 1 - \int g(Θ)dΘ \right) \quad (4.71)
\]
Differentiating $\ell_g$ with respect to $g$ and setting the result to zero, we obtain
\[
\frac{∂\ell_g}{∂(g(Θ))} = \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z_0^k, \cdots, z_t^k=1}^{\mid Z \mid} q^k_t(z^k_{0:t}) \ln \tilde{r}^k_t p(a^k_{0:t}, z^k_{0:t} | q^k_{1:t}, Θ) - 1 - \ln p(Θ) + λ
\]
which is solved to give
\[
g(Θ) = e^{λ-1} p(Θ) \exp \left\{ \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{z_0^k, \cdots, z_t^k=1}^{\mid Z \mid} q^k_t(z^k_{0:t}) \ln \tilde{r}^k_t p(a^k_{0:t}, z^k_{0:t} | q^k_{1:t}, Θ) \right\} \quad (4.73)
\]
By using the constraint $\int g(Θ)dΘ = 1$, we arrive at (4.38) with $e^{1-λ} = C_Θ^{-1}$. Q.E.D.
Chapter 5

Reinforcement Learning in Multiple Partially Observable Stochastic Environments

In this chapter we develop the regionalized policy representation (RPR) for reinforcement learning in multiple partially observable stochastic environments. The RPR multi-task learning (RPR-MTL) algorithm is based on a Dirichlet process formulation. The RPR-MTL algorithm automatically identifies useful experiences from similar environments based on the clustering property of Dirichlet processes, and directly learns the policies for these environments based on all relevant experiences.

5.1 Introduction

All discussions in Chapter 4 were concerned with learning a policy for a single environment, while in many real-world settings there could be multiple environments for which policies are desired. There are two special cases: in the first, a single agent has collected experiences from previous environments and he wants to borrow from previous experiences when learning the policy for a new environment. In the second case, multiple agents are distributed across multiple environments, and they want to communicate with each other and share experiences such that their respective performances are enhanced. In either case, the experiences in one environment should be properly exploited to benefit learning in another environment (Guestrin et al., 2003). The positive transfer of experiences between multiple environments and the joint learning in multiple policies will ideally save time, improve policy quality, and enhance generalization to new environments, especially when the experiences from
each individual environment are scarce (Thrun 1996). We employ the concept of “multi-task learning” (Caruana 1997) for learning multiple policies simultaneously, treating the learning in each individual environment as a task. The policies are formulated by multiple RPRs that are coupled by a common Dirichlet process (DP) prior — such a formulation is referred to as the multitask RPR (RPR-MTL) throughout the chapter.

The transferability of experiences between tasks depends on whether the tasks are correlated. If the tasks are independent of each other, experience transfer will deteriorate the performance on each individual task. If the tasks are identical to each other, pooling all the experiences together is the best way to share information. In practice we do not know the degree to which the tasks are related. How to identify the common structure shared across multiple environments and how to appropriately represent the sharing are the key issues that need be addressed when developing an MTL algorithm.

To date there has been much work addressing the sharing structure. Most of the work follows a hierarchical Bayesian approach, which assumes that the parameters (models) for each task are sampled from a common prior distribution, such as a Gaussian distribution specified by unknown hyperparameters (Lawrence and Platt 2004; Yu et al. 2003). The parameters as well as the hyperparameters are estimated simultaneously in the learning phase. In (Bakker and Heskes 2003) a single Gaussian prior is extended to a Gaussian mixture, each task is given a corresponding Gaussian prior and related tasks are allowed to share a common Gaussian prior. Such a formulation for information sharing is more flexible than a single common prior, but still has limitations. First, the form of the prior distribution must be specified a priori, such as Gaussian distribution or other forms of distributions. Secondly, the number of mixture components must also be pre-specified. Note that each environment is
very complex and imposing a parametric common prior distribution across all environments is often too restrictive and possibly misleading. As a solution, we adopt a nonparametric approach by employing the Dirichlet process (DP) as our prior (Yu et al. 2004; Xue et al. 2006). The nonparametric prior does not assume a specific form and each environment has its own prior distribution drawn from a common Dirichlet process. The number of distinct prior distributions need not be specified in advance, but varies adaptively. Sharing information in this way keeps the uniqueness of each task and meanwhile employs the similarities between relevant tasks.

Although we are not the first to use the nonparametric DP prior for multi-task learning, this chapter represents the first attempt to apply the DP prior to reinforcement learning in multiple partially observable stochastic environments. The information transfer or sharing is performed directly at the policy level, via the regionalized policy representation.

The remainder of this chapter is organized as follows. We first give an introduction to Dirichlet processes in Section 5.2, and discuss some properties of the RPR regarding experience transfer in Section 5.2.1. In Section 5.3 we formulate the RPR-MTL and present the Gibbs sampling algorithm for the RPR-MTL, following which, we then give a hybrid Gibbs-variational algorithm in Section 5.4 for fast learning of the RPR-MTL. We present experimental results on grid worlds and multi-aspect classification in Section 5.5 and summarize the chapter in Section 5.6.

5.2 Dirichlet Process

Let $\mathcal{X}$ be a space and $\mathcal{Y}$ be the set of subsets of $\mathcal{X}$. Let $G_0$ be a base probability measure of $X \in \mathcal{X}$ and $\alpha$ be a positive scalar. Let $B_1, B_2, \ldots, B_k$ be a partition of $\mathcal{X}$ satisfying $B_i \cap B_i = \emptyset \forall i \neq j$ and $\bigcup_{i=1}^{k} B_i = \mathcal{X}$. A stochastic process $G$ is called
a Dirichlet process (Ferguson 1973) if, for any partition \( B_1, B_2, \ldots, B_k \) of \( \mathcal{X} \),

\[
(G(X \in B_1), G(X \in B_2), \ldots, G(X \in B_k)) \\
\sim \text{Dir}(\alpha G_0(X \in B_1), \alpha G_0(X \in B_2), \ldots, \alpha G_0(X \in B_k))
\]

(5.1)

where \( \text{Dir} \) denotes the Dirichlet distribution.

**Some properties of the Dirichlet Process**

- The expectation of \( G \) is given by the base
  \[
  E(G) = G_0
  \]
  (5.2)

- The posterior distribution of \( G \) given observations \( X_1, X_2, \ldots, X_n \) is also a
  Dirichlet process

  \[
p(G|X_1, X_2, \ldots, X_n) = \text{Dir}\left(\alpha + n, \frac{\alpha}{\alpha + n} G_0 + \frac{1}{\alpha + n} \sum_{i=1}^{n} \delta_{X_i}\right)
  \]
  (5.3)

  where \( \delta_{X_i} \) is a discrete probability measure concentrated at \( X_i \).

- Given \( X_1, X_2, \ldots, X_n \), the conditional probability of a new observation \( X_{n+1} \) is
  given by

  \[
p(X_{n+1}|X_1, X_2, \ldots, X_n, \alpha, G_0) = \delta_{X_i} \text{ with probability } \frac{n_i}{\alpha + n} \quad \text{G_0 with probability } \frac{\alpha}{\alpha + n}
  \]
  (5.4)

  The property (5.4) shows that given \( X_1, X_2, \ldots, X_n \), the new sample drawn
  from the prior \( G_0 \) with probability \( \frac{\alpha}{\alpha + n} \) and takes one of the previous \( n \) samples
  with probability \( \frac{1}{\alpha + n} \). Let \( X_1^*, X_2^*, \ldots, X_K^* \) denote the distinct samples from
  \( X_1, X_2, \ldots, X_n \), the next sample is drawn according to

  \[
  X_{n+1}|X_1, X_2, \ldots, X_n, \alpha, G_0 \sim \begin{cases}
  \delta_{X_i}^* & \text{with probability } \frac{n_i}{\alpha + n} \\
  G_0 & \text{with probability } \frac{\alpha}{\alpha + n}
  \end{cases}
  \]
  (5.5)
where \( n_i \) is the number of instances of \( X_i^* \) in \( X_{1:n} \). Blackwell and MacQueen (1973) showed that (5.5) has exactly the same property as the \( \text{Pólya urn scheme} \). The new sample \( X_{n+1} \) tends to take a previous sample that has occurred with a higher frequency. The scalar \( \alpha \) controls the probability of sampling from the base distribution \( G_0 \). The larger \( \alpha \) is, the higher the probability the new sample \( X_{n+1} \) will be drawn from the base distribution \( G_0 \). In the extreme cases, when \( \alpha \) tends to infinity, the new sample \( X_{n+1} \) will always be drawn from \( G_0 \); when \( \alpha \) tends to zero, the new sample \( X_{n+1} \) will be taken from among \( X_{1:n} \).

### 5.2.1 Some Properties of the RPR regarding Experience Transfer

Two environments differing in their world-state dynamics may have similar decision dynamics. As the RPR represents the dynamics of decision states, it captures the fundamental characteristics of the environment that affect the optimal decisions. Environments may differ in many ways: they may have different number of world states, observations, and actions; their reward structures may be different. Since the RPR does not assume access to the world states, it can naturally handle policy transfer between environments with different world states. When two environments have only a subset of observations or actions in common, the RPR takes as its actions (or observations) the union of the actions (or observations) from all possible environments. When learning in a particular environment, the RPR will simply ignore those \( W(\cdot, a, o, \cdot) \) where the \( a \) and \( o \) are not allowed in the environment. Another important feature of the RPR is that it is insensitive to the absolute rewards; this is because the RPR backs up a history-dependent distribution of actions and the action can stay invariant even when the value changes. Thus the RPR can be easily transferred between environments provided the difference in their reward structures
do not significantly change the decision regions.

As a result, the RPR policy learned for a previous environment can be directly applied to a new environment provided the actions and observations of the previous subsume those of the new. This is not possible for model-based policies unless the environments have fully shared world-states – model-based policies are expressed in terms of belief state which changes as the definition of world-states changes. Moreover the RPR allows experience with previous environments to be conveniently incorporated into policy learning for a new environment, via an appropriate sharing structure such as Dirichlet processes (DP).

5.3 Joint RRPs for Multiple Stochastic Environments

5.3.1 Basis Formulation

Consider $M$ partially observable and stochastic environments indexed by $m = 1, 2, \cdots, M$, each of which is apparently different from others but may actually share fundamental common characteristics. Assume we have a set of episodes collected from each environment, $D^{(K_m)}_m = \{(a_{0m,k}^{m,k}, o_1^{m,k}, a_1^{m,k}, r_1^{m,k}, \cdots, o_{T_{m,k}}^{m,k}, a_{T_{m,k}}^{m,k}, r_{T_{m,k}}^{m,k})\}_{k=1}^{K_m}$, for $m = 1, 2, \cdots, M$, where $T_{m,k}$ represents the length of episode $k$ in environment $m$. For the $m$-th environment, we have the empirical value function

$$\hat{V}(D^{(K_m)}_m; \Theta_m) = \frac{1}{K_m} \sum_{k=1}^{K_m} \sum_{t=0}^{T_{m,k}} \hat{r}_t^{m,k} \cdot p(a_{0t}^{m,k}, o_{1t}^{m,k}, \Theta_m)$$

(5.6)

where $\Theta_m = \{\pi_m, \mu_m, W_m\}$ are the RPR parameters for the $m$-th individual environment. For given episodes, the empirical value $\hat{V}(D^{(K_m)}_m; \Theta_m)$ is a function of $\Theta$. One way of learning the RPR for the $m$-th environment is to seek the $\Theta_m$ that maximizes
the empirical value. An alternative is to compute

\[
p(\Theta_m|D^{(K_m)}_m) = \frac{\hat{V}(D^{(K_m)}_m; \Theta_m)p(\Theta_m)}{\int \hat{V}(D^{(K_m)}_m; \Theta_m)p(\Theta_m)d\Theta_m} \tag{5.7}
\]

which is literally a normalized product of the empirical value function and a prior distribution \(p(\Theta_m)\). Since \( \int p(\Theta_m|D^{(K_m)}_m)d\Theta_m = 1 \), (5.7) yields a valid probability distribution, which we call the posterior distribution of \(\Theta_m\) given the episodes \(D^{(K_m)}_m\).

It is noted that (5.7) would be the Bayes rule, if \(\hat{V}(D^{(K_m)}_m; \Theta_m)\) were a likelihood function. Since \(\hat{V}(D^{(K_m)}_m; \Theta_m)\) is a value function in our case, (5.7) is a somewhat non-standard use of the Bayes rule. However, like the classic Bayes rule, (5.7) indeed gives a distribution whose shape incorporates both the prior information about \(\Theta_m\) and the empirical information from the episodes.

We follow a nonparametric framework for specifying the prior distribution of \(\Theta_m\), and let \(\Theta_m, m = 1, 2, \cdots, M\), each have its own prior distribution. We do not presume any parametric form for these priors; instead, we let these priors be drawn from a common Dirichlet process (Ferguson 1973; Blackwell and MacQueen 1973; Antoniak 1974; Sethuraman 1994) with base distribution \(G_0\) and precision \(\alpha\). Thus

\[
\Theta_m|G \sim G
\]

\[
G|\alpha, G_0 \sim DP(\alpha, G_0) \tag{5.8}
\]

Since the priors are drawn from the same Dirichlet process, they have the chance of being similar to each other, and this imposes a clustering structure — the \(\Theta_m\)’s are encouraged to cluster together if doing so does not conflict with the empirical evidence from the episodes. The clustering mechanism of the Dirichlet process can also be equivalently interpreted by an explicit construction of \(G\), which has the form of a infinite random mixture — the location of each component is drawn from \(G_0\) and the mixing proportions are determined by a stick-breaking process parameterized by \(\alpha\).
In the stick-breaking formulation, the $\Theta_m$'s can either share only the location or both the location and the mixing proportion, of a component. When the explicit $G$ is not preferred, one marginalizes out $G$ and obtains the probability of $\Theta_m$ that is directly conditional on $G_0$ and $\alpha$, as shown in (5.4) in the general case, and rewritten below for $\Theta_m$

$$p(\Theta_m|\Theta_{-m}, \alpha, G_0) = \frac{\alpha}{\alpha + M - 1} G_0(\Theta_m) + \frac{1}{\alpha + M - 1} \sum_{j=1, j\neq m}^M \delta(\Theta_m - \Theta_j)$$ (5.9)

where the probability is conditioned on $\Theta_{-m} = \{\Theta_1, \ldots, \Theta_{m-1}, \Theta_{m+1}, \ldots, \Theta_M\}$. Equation (5.9) gives the full conditionals of the joint prior distribution $p(\Theta_1, \Theta_2, \cdots, \Theta_M)$ and by performing Gibbs sampling (Robert and Casella 1999) one obtains the samples from the joint prior.

It should be noted that the formulation in (5.8) places a DP prior directly on $\Theta$. Because of the discrete nature of $G$, this implies that when parameters $\Theta$ are shared between different environments, they are shared exactly. This may be too restrictive for some problems, since environment $i$ and $j$ are similar, we may desire the associated parameters $\Theta_i$ and $\Theta_j$ to be similar, but not exactly the same. This may be accommodated, for example, via the following modification to (5.8)

$$\Theta_m|\Psi_m \sim F(\Theta_m|\Psi_m)$$
$$\Psi_m|G \sim G$$
$$G|\alpha, G_0 \sim DP(\alpha, G_0)$$ (5.10)

The formulation in (5.10) results in an infinite mixture model of $\Psi$. When environment $i$ and $j$ share, $\Theta_i$ and $\Theta_j$ share a component of this infinite mixture, but the specific draws of $\Theta_i$ and $\Theta_j$ will generally differ from each other — this can provide greater flexibility. For the work in this Chapter, we employ the formulation in (5.8)
for simplicity, and our results extend to the formulation in (5.10) in a straightforward way.

Thus far we have only addressed the prior and have not yet taken the episodes into consideration. We next combine the prior and empirical evidence from the episodes, to compute the posterior of the \( \Theta_m \)'s. Our goal is to find the distribution of \( \Theta_m \) conditional on \( \Theta_{-m}, D_m^{(K_m)} \), \( G_0 \), and \( \alpha \). From (5.7) and (5.9), we write

\[
\begin{align*}
p(\Theta_m | \Theta_{-m}, D_m^{(K_m)}, \alpha_0, G_0) &\propto \frac{\hat{V}(D_m^{(K_m)}; \Theta_m) (\alpha G_0(\Theta_m) + \sum_{j=1, j \neq m}^M \delta(\Theta_m - \Theta_j))}{C} \\
&= \frac{\alpha \hat{V}(D_m^{(K_m)}; \Theta_m) G_0(\Theta_m) + \sum_{j=1, j \neq m}^M \hat{V}(D_m^{(K_m)}; \Theta_j) \delta(\Theta_m - \Theta_j)}{C} \\
&= \frac{\alpha p(\Theta_m | D_m^{(K_m)}) \int G_0(\Theta) \hat{V}(D_m^{(K_m)}; \Theta) d\Theta + \sum_{j=1, j \neq m}^M \hat{V}(D_m^{(K_m)}; \Theta_j) \delta(\Theta_m - \Theta_j)}{C}
\end{align*}
\]

(5.11)

where \( p(\Theta_m | D_m^{(K_m)}) \) is as defined by (5.7) and

\[
C = \alpha \int G_0(\Theta) \hat{V}(D_m^{(K_m)}; \Theta) d\Theta + \sum_{j=1, j \neq m}^M \hat{V}(D_m^{(K_m)}; \Theta_j)
\]

(5.12)

is the normalization constant. The simple algorithm for sampling from \( p(\Theta_m | \Theta_{-m}, D, \alpha_0, G_0) \) is thus

\[
\Theta_m | \Theta_{-m}, D_m^{(K_m)}, \alpha_0, G_0 \sim \begin{cases} 
\delta(\Theta_m - \Theta_j) \text{ with probability } \frac{\hat{V}(D_m^{(K_m)}; \Theta_j)}{C} \\
p(\Theta_m | D_m^{(K_m)}) \text{ with probability } \frac{\int G_0(\Theta) \hat{V}(D_m^{(K_m)}; \Theta) d\Theta}{C}
\end{cases}
\]

(5.13)

We can apply a Gibbs sampler to (5.13) to obtain the samples of \( (\Theta_1, \Theta_2, \cdots, \Theta_M) \). Upon convergence of the Gibbs sampling, these samples represent the samples from the joint posterior \( p(\Theta_1, \Theta_2, \cdots, \Theta_M | \{D_m^{(K_m)}\}_{m=1}^M, \alpha_0, G_0) \).
5.3.2 Multi-RPR Learning by Gibbs Sampling

Equation (5.13) provides the posterior full conditionals, based on which Gibbs sampling can be performed to jointly learn the RPRs for the $M$ environments. However, Gibbs sampling based directly on (5.13) is very inefficient because it always updates a $\Theta$ by using the episodes from a single environment and it never updates the $\Theta$ by pooling the episodes from the relevant environments which share the same $\Theta$. When the probability for sharing is large and the sharing environments are not allowed to congregate their episodes, the mixing of $\Theta$’s will be slow and the convergence of the Gibbs sampler will be greatly impeded.

This situation can be avoid by introducing a vector of indicator variables (Neal 1998). Let $\overline{\Theta} = \{\overline{\Theta}_n : n = 1, \ldots, N\}$ be the set of distinct $\Theta_m$, where $N \leq M$ is the number of distinct elements in $\Theta = \{\Theta_1, \Theta_2, \ldots, \Theta_M\}$. Let $c = \{c_1, c_2, \ldots, c_M\}$ denote the vector of indicator variables defined by $c_m = n$ iff $\Theta_m = \overline{\Theta}_n$, $n \in \{1, 2, \ldots, N\}$ and $m = 1, 2, \ldots, M$. The prior conditional distribution $p(c_m|c_{-m})$ that arises from the polya-urn representation of the Dirichlet process is as follows

$$p(c_m|c_{-m}, \alpha) = \frac{\alpha}{\alpha + M - 1} \delta(c_m - N - 1) + \sum_{n=1}^{N} \frac{l_{-m,n}}{\alpha + M - 1} \delta(c_m - n) \quad (5.14)$$

where $l_{-m,n}$ denotes the number of elements in $\{i : c_i = n, i \neq m\}$ and $c_m = N + 1$ indicates a new sample is drawn from the base $G_0$. It is implicitly assumed that the samples from the Dirichlet process are exchangeable (i.e., the joint distribution of the samples is independent of the order of the samples). Given $c_m$ and $\overline{\Theta}$, the distribution of $\Theta_m$ is given by

$$p(\Theta_m|c_m, \overline{\Theta}, \alpha, G_0) = \delta(c_m - N - 1)G_0(\Theta_m) + \sum_{n=1}^{N} \delta(c_m - n)\delta(\Theta_m - \overline{\Theta}_n) \quad (5.15)$$

We now incorporate the empirical evidence from the episodes and compute the
the posterior conditional distribution of $c_m$. To obtain a usable Gibbs sampler as discussed shortly, we must also make the posterior conditional on the hidden decision states $z$. Let $z^m = \{z^{m,k}_{0:T_m,k} : k = 1, 2, \cdots, K_m\}$, we have

$$
p(c_m | c_{-m}, G, D^{(K_m)}, z^m, \alpha) = \frac{\int \hat{V}(D^{(K_m)}_m, z^m; \Theta_m)p(\Theta_m | c_m, \Theta, G_0)p(c_m | c_{-m}, \alpha)d\Theta_m}{\sum_{c_m=1}^{N+1} \int \hat{V}(D^{(K_m)}_m, z^m; \Theta_m)p(\Theta_m | c_m, \Theta, G_0)p(c_m | c_{-m}, \alpha)d\Theta_m}
$$

$$
= \beta_m^0 \delta(c_m - N - 1) + \sum_{n=1}^{N} \beta_n^m \delta(c_m - n) \quad (5.16)
$$

where

$$
\beta_0^m = \frac{\alpha C_m^{\alpha} G_0}{\alpha C_m^{\alpha} G_0 + \sum_{n=1}^{N} l_{-m,n} \hat{V}(D^{(K_m)}_m, z^m; \Theta_n)} \quad (5.17)
$$

$$
\beta_j^m = \frac{l_{-m,j} \hat{V}(D^{(K_m)}_m, z^m; \Theta_j)}{\alpha C_m^{\alpha} G_0 + \sum_{n=1}^{N} l_{-m,n} \hat{V}(D^{(K_m)}_m, z^m; \Theta_n)} \quad (5.18)
$$

$$
C_m^{\alpha} G_0 = \int \hat{V}(D^{(K_m)}_m, z^m; \Theta_m)G_0(\Theta_m)d\Theta_m \quad (5.19)
$$

and

$$
\hat{V}(D^{(K_m)}_m, z^m, \Theta_m) = \sum_{k=1}^{K_m} \sum_{t=0}^{T_{m,k}} \hat{r}_{m,k}^t p(t_{0:t}, n_{0:t} | o_{1:t}, \Theta_m)
= \sum_{k=1}^{K_m} \sum_{t=0}^{T_{m,k}} \mu_m(z^{m,k}_0) \pi_m(z^{m,k}_0, a^{m,k}_0) \prod_{\tau=1}^{t} W_m(z^{m,k}_\tau, a^{m,k}_\tau, o^{m,k}_\tau, \pi_m(z^{m,k}_\tau, a^{m,k}_\tau)) \quad (5.20)
$$

From (5.17) and (5.18), we can see the the indicator $c_m$ tends to equal $n$ if $\hat{V}(D^{(K_m)}_m, \Theta_n)$ is large, which occurs when the $n$-th distinct RPR produces a high
empirical value (discounted accumulative reward) in the $m$-th environment. If none of the existing RPRs produces a high empirical value in the $m$-th environment, $c_m$ will tend to be equal to $N + 1$, which means a new $\Theta$ will be drawn from the base $G_0$. As with the direct Gibbs sampler in (5.13), the Gibbs sampler (5.16) does not impose a fixed number of distinct RPRs (represented by $\Theta$) but allows it to vary from iteration to iteration. The Gibbs sampler in (5.16), however, converges much faster than that in (5.13), because once multiple environments share the same $\Theta$ (their indicator variables are equal), the episodes from these environments are pooled together, and the corresponding $\Theta_n$ is updated based on this pool.

To complete the computation of $C_{\hat{G}_0}^{m,k,t}$ in (5.19), we need to specify the form of the base $G_0$. As our $\hat{V}(D_m^{(K_m)}, z^m, \Theta_m)$ in (5.20) has an exponential form, we choose conjugate-exponential base distribution $G_0$, assuming the model parameters $\{\mu, \pi, W\}$ are independent of each other a priori

$$G_0(\Theta) = p(\mu|\upsilon)p(\pi|\rho)p(W|\omega)$$

(5.21)

$$p(\mu|\upsilon) = \text{Dir}(\mu(1), \ldots, \mu(|Z|)|\upsilon)$$

(5.22)

$$p(\pi|\rho) = \prod_{i=1}^{\lvert Z \rvert} \text{Dir}(\pi(i, 1), \ldots, \pi(i, |A|)|\rho_i)$$

(5.23)

$$p(W|\omega) = \prod_{a=1}^{|A|} \prod_{o=1}^{|O|} \prod_{i=1}^{|Z|} \text{Dir}(W(i, a, o, 1), \ldots, W(i, a, o, |Z|)|\omega_{i,a,o})$$

(5.24)

Substituting (5.21) and (5.20) into (5.19), we find the analytical form of the integral

$$C_{\hat{G}_0}^{m,k,t} = \sum_{k=1}^{K_m} \sum_{t=0}^{T_{m,k}} \tau_t^{m,k} \left[ \prod_i \Gamma(\gamma_{i,m,k,t}^{m,k,t}) \prod_i \Gamma(\gamma_{i,m,k,t}^{m,k,t}) \prod_i \Gamma(\gamma_{i,m,k,t}^{m,k,t}) \prod_i \Gamma(\gamma_{i,m,k,t}^{m,k,t}) \prod_i \Gamma(\gamma_{i,m,k,t}^{m,k,t}) \prod_i \Gamma(\gamma_{i,m,k,t}^{m,k,t}) \prod_i \Gamma(\gamma_{i,m,k,t}^{m,k,t}) \prod_i \Gamma(\gamma_{i,m,k,t}^{m,k,t}) \right]$$

(5.25)
with
\[
\hat{\upsilon}^{m,k,t}_i = \upsilon^m_i + \delta(z^{m,k,t}_0 - i) \tag{5.26}
\]
\[
\hat{\rho}^{m,k,t}_{i,a} = \rho^m_{i,a} + \sum_{t=0}^{T_{m,k}} \delta(z^{m,k}_t - i)\delta(a^{m,k}_t - a) \tag{5.27}
\]
\[
\hat{\omega}^{m,k,t}_{i,a,o,j} = \omega^m_{i,a,o,j} + \sum_{t=1}^{T_{m,k}} \delta(z^{m,k}_{t-1} - i)\delta(a^{m,k}_{t-1} - a)\delta(o^{m,k}_t - o)\delta(z^{m,k}_t - j) \tag{5.28}
\]

Once the samples of the indicator variables are given, we put together the episodes from the environments whose indication variables are equal, and update the \(\Theta\)'s. Let \(I_n(c) = \{m : c_m = n\}\) for \(n = 1, \cdots, N\). If \(I_n(c)\) is an empty set, remove \(\Theta_n\); otherwise, draw \(\Theta_n\) according to
\[
p(\Theta_n|c, D, z) = \frac{\sum_{m \in I_n(c)} V(D^{(K_m)}_m, z^m; \Theta_n) G_0(\Theta_n)}{\int \sum_{m \in I_n(c)} V(D^{(K_m)}_m, z^m; \Theta_n) G_0(\Theta_n) d\Theta_n}
\]
\[
= \sum_{m \in I_n(c)} \left[ \sum_{k=1}^{K_m} \sum_{t=0}^{T_{m,k}} \hat{\beta}^{m,k,t}_0 \hat{G}^{m,k,t}_0(\Theta_n) \right] \tag{5.29}
\]
where
\[
\hat{G}^{m,k,t}_0(\Theta) = p(\mu|\hat{\upsilon}^{m,k,t}) p(\pi|\hat{\rho}^{m,k,t}) p(W|\hat{\omega}^{m,k,t}) \tag{5.30}
\]
has the same form as \(G_0(\Theta)\) in (5.21) but with \(\upsilon, \rho, \omega\) respectively replaced by \(\hat{\upsilon}^{m,k,t}\), \(\hat{\rho}^{m,k,t}\), \(\hat{\omega}^{m,k,t}\) as given in (5.26), (5.27), and (5.28), and
\[
\hat{\beta}^{m,k,t}_0 = \frac{C^{m,k,t}_{G_0}}{\sum_{m \in I_n(c)} \sum_{k=1}^{K_m} \sum_{t=0}^{T_{m,k}} C^{m,k,t}_{G_0}} \tag{5.31}
\]
with \(C^{m,k}_{G_0}\) given by (5.19).

Given \(\Theta_m\), the hidden decision states \(z^{m,k}_t, t = 0, 2, \cdots, T_{m,k}\) are sampled by using the initial state probabilities \(\mu_m(z^{m,k}_0)\) and state transition probabilities \(W_m(z^{m,k}_{t-1}, a^{m,k}_{t-1}, o^{m,k}_t, z^{m,k}_t)\).
The complete Gibbs sampling algorithm for the multitask-RPR learning is summarized in Table 5.1. Upon convergence of Gibbs sampling, we obtain the samples \((\Theta, z, c)\) drawn from the joint posterior distribution \(p(\Theta, z, c | D, \alpha, G_0)\), conditional on the empirical evidence from the episodes \(D\) congregated across all the environments. From these samples, we recover \(\Theta_m = \Theta_{c_m}\) for \(m = 1, 2, \ldots, M\). By simple post-processing of the samples, we obtain the mean or mode of each \(\Theta_m\), which gives a single RPR for each environment and yields the history-dependent policy as given by (4.4). Alternatively, we can keep the samples of each \(\Theta_m\) and use them to produce an ensemble of RPRs for each environment. The RPR ensemble gives multiple history-dependent policies, which are marginalized (averaged) to yield the final choice for the action.

Table 5.1: The Gibbs sampling algorithm for multitask-RPR learning

1. Initialization: \(N = M\), \(c_m = m\) for \(m = 1, 2, \ldots, M\), initialize \(\{\Theta_n\}_{n=1}^{N}\).
2. Repeat until convergence:

   1. For \(m = 1, 2, \ldots, M\),
      1.1 \(\Theta_m = \Theta_{c_m}\)
      1.2 Draw hidden decision states
          \(z_{0:T_m,k} \sim p(z_{0:T_m,k} | a_{0:T_m,k}, \Theta_m)\)
          for \(k = 1, 2, \ldots, K_m\).
      1.3 Draw indicator variables
          \(c_m \sim p(c_m | c_{-m}, \Theta, D_m(K_m), z_m, \alpha)\)
          according to (5.16), and using (5.17), (5.18), (5.25).

3. For \(n = 1, 2, \ldots, N\),
   3.1 Find the \(n\)-th cluster: \(I_n(c) = \{m : c_m = n\}\) for \(n = 1, \cdots, N\)
   3.2 If \(I_n(c)\) is an empty set, delete the \(n\)-th cluster;
       otherwise, draw the distinct parameters
       \(\Theta_n \sim p(\Theta_n | c, D, z)\)
       according to (5.29).
5.4 The Hybrid Gibbs-Variational Algorithm for Multi-RPR Learning

5.4.1 The \((q, g)\)-scheme

We have obtained improved convergence of Gibbs sampling by introducing indicator variables \(c\) to allow explicit merging of episodes from sharing environments and by using a conjugate base \(G_0\) that permits the exact solution of \(C_{m}^{m}\). However, in practice we find that hundreds of thousands of iterations are still required to observe the Gibbs sampler’s convergence. There are two particular reasons to which the slow convergence of Gibbs sampling can be attributed in our case.

First, the conjugacy of \(G_0\), and therefore the analytic solution of \(C_{m}^{m}\) in (5.25), are conditional on the samples of the hidden decision states \(z\). As a result, the update of \(\Theta\) at each Gibbs iteration is made for these \(z\) samples alone, as seen in the update equations (5.26), (5.27), and (5.28). Since there is a \(z_{i}^{m,k}\) for every time step \(t\) in every episode \(k\) from every environment \(m\), it could take an intolerably large number of iterations for a mixing of the \(z\) samples.

Second, our empirical value function in (5.6) is a mixture, each component corresponding to a single time step in a single episode. Typically there are many episodes and many time steps in each episode, implying the mixture can be very large. This gives rise to an equally large or even larger (when the episodes of multiple environments are merged) mixture in the posterior of \(\Theta\), as shown in (5.29). To draw a sample of \(\Theta\) from such a mixture, one first makes a draw to decide the component and then draws \(\Theta\) from this component. Therefore the update of \(\Theta\) at each Gibbs iteration is de facto based on using the empirical evidence from a single episode involving the reward at a single time — instead of based on all episodes at all time steps. Gibbs sampling in our case lacks the mechanism of simultaneously using all
available empirical evidences, and thus has a very slow mixing of the samples of \( \Theta \).

Our method of ameliorating the Gibbs sampler in Section 5.3.2 is based on the idea that, instead of alternatively updating samples of \( z \) and \( \Theta \), we alternatively update \( q(z) \) and \( g(\Theta) \), the factorized variational posteriors introduced in Section 4.5 of Chapter 4. Just as alternative sample-updating makes the samples converge to the draws from \( p(\Theta, z|D) \), alternative \( (q, g) \)-updating makes the product \( q(z)g(\Theta) \) converge to the variational lower bound of \( p(\Theta, z|D) \). The \( (q, g) \)-scheme can be compared to a block-style Gibbs sampler, with each \( q \) and \( g \) corresponding to a block of variables — each \( q \) represents the \( z \)’s of all episodes from an environment and each \( g \) represents the RPR parameters of one or multiple environments. The \( (q, g) \)-scheme collapses multiple variables into a single block and manipulates these variables simultaneously, thus the mixing is performed at the level of blocks, instead of the level of variables. For this reason, the \( (q, g) \)-scheme has better mixing than the Gibbs sampler in Section 5.3.2 and converges much faster.

From an intuitive perspective, the \( (q, g) \)-scheme allows one to use soft counts instead of hard counts — i.e., one replaces the Kronecker \( \delta \) (which represent hard counts) in (5.26), (5.27), and (5.28) with soft variational counts which indicates the credits allocated to each hidden decision state \( z \). By using soft counts, one can visit all decision states at a single Gibbs iteration and update \( \Theta \) simultaneously for all states.

### 5.4.2 The Gibbs-Variational Algorithm

At a single Gibbs iteration, we use the \( (q, g) \)-scheme to process \( z \) and \( \Theta \) efficiently, keeping the remaining part of the Gibbs sampler intact, thus leading to a hybrid Gibbs-variational algorithm.

As we recall from Section 4.5 in Chapter 4, the \( (q, g) \)-scheme is based on a pair
of analytic forms: one is $q$ for given $g$, the other one is $g$ for given $q$. In the context of multiple environments, these analytic expressions are rewritten as follows

$$q^{m,k}_t(z^{m,k}_{0:t}) = C_{z,t}^{-1} z^{m,k}_{0:t} \exp \left\{ \int g_m(\Theta_m) \ln p(a^{m,k}_{0:t}, z^{m,k}_{0:t} | o^{m,k}_{1:t}, \Theta_m) \, d\Theta_m \right\} \tag{5.32}$$

$$g_m(\Theta_m) = C_{\Theta,m}^{-1} p(\Theta_m) \exp \left\{ \frac{1}{K_m} \sum_{k=1}^{K_m} \sum_{t=0}^{T_{m,k}} \sum_{Z^{m,k}_{0:t}} q^{m,k}_t(z^{m,k}_{0:t}) \ln r^{k}_t p(a^{m,k}_{0:t}, z^{m,k}_{0:t} | o^{m,k}_{1:t}, \Theta_m) \right\} \tag{5.33}$$

where $C_{z,m}$ and $C_{\Theta,m}$ are normalization constants such that $\int g_m(\Theta_m) \, d\Theta_m = 1$ and

$$\frac{1}{K_m} \sum_{k=1}^{K_m} \sum_{t=0}^{T_{m,k}} \sum_{Z^{m,k}_{0:t}} q^{m,k}_t(z^{m,k}_{0:t}) = 1$$

The hybrid Gibbs-variational algorithm is summarized in Table 5.2. This algorithm can be viewed from two perspectives as discussed below.

In the first view, the algorithm is a single variational-Bayes (VB) algorithm applied to the episodes congregated across all environments, and an internal task-grouping is performed adaptively over the iterations. Upon convergence, task-grouping becomes stationary and the algorithm splits into $N$ independent VB algorithms, which can be implemented in parallel.

In the second view, the algorithm consists of $M$ dependent VB algorithms, and each algorithm interacts with other algorithms by exchanging intermediate computational results. Upon convergence, there is no new information to exchange and the interaction is cut off, in which case the $M$ VB algorithms run independently of each other. The second view suggests an distributive implementation of the algorithm, which is further discussed in Section 5.4.3.
Table 5.2: The hybrid Gibbs-variational algorithm for multitask-RPR learning

1 Initialization: $N = M$; $c_m = m$ for $m = 1, 2, \cdots, M$; initialize $\{g_n(\cdot)\}_{n=1}^N$;
   $\overline{\Theta}_n = \int \Theta g_n(\Theta) d\Theta$ for $n = 1, 2, \cdots, N$.

2 Repeat until convergence:
   1.1 For $m = 1, 2, \cdots, M$,
      1.1.1 $g_m(\cdot) = \overline{g}_m(\cdot), \Theta$
      1.1.2 Update variational posteriors of hidden decision states
           $q_{m,k}^{m,k}(z_{0:t}^m) = C_{m,k}^{-1} z_{0:t}^m \tilde{r}_{m,k}^{m,k} \ln p(a_{0:t}, z_{0:t}^m, \varphi_k^m, \Theta_m) d\Theta_m$
           for $k = 1, 2, \cdots, K_m$.
      1.1.3 Draw indicator variables
           $c_m \sim p(c_m|c_m, \overline{\Theta}, D_m(K_m), z_m, \alpha)$
           according to (5.16), using (5.17) and (5.18), with $C_m = C_{\overline{\Theta}_m}$
           given by the normalization constant in (5.33).

1.2 For $n = 1, 2, \cdots, N$,
   1.2.1 Find the $n$-th cluster: $I_n(c) = \{m : c_m = n\}$ for $n = 1, \cdots, N$
   1.2.2 If $I_n(c)$ is an empty set, delete the $n$-th cluster;
      otherwise, update variational posteriors of the distinct parameters according to
      $g(\overline{\Theta}_n) \propto p(\overline{\Theta}_n) K_n \sum_{m \in I_n(c)} \sum_{k=1}^{K_m} \sum_{t=0}^{T_m,k} \sum_{i=1}^{z_{0:t}^m} \sum_{i=1}^{z_{0:t}^m} \phi_{m,k,t}^m(\overline{\Theta}_n)$
      $\overline{K}_n = \sum_{m \in I_n(c)} K_m$
      $\varphi_{m,k,t}^m(\overline{\Theta}_n) = q_{m,k}^{m,k}(z_{0:t}^m) \ln \tilde{r}_{m,k} \ln p(a_{0:t}, z_{0:t}^m, \varphi_k^m, \overline{\Theta}_n)$
      $\overline{\Theta}_n = \int \Theta g_n(\Theta) d\Theta$

5.4.3 Distributive Computation

Assume that there are $M$ agents, with one agent in each of the $M$ environments and
there is a central node that communicates with the agents and performs global
computations. Each agent collects episodes by interacting with the local environment
and performs computations based on the local episodes.

Each agent uses a $(q, g)$-scheme to alternatively update the posteriors based on
local episodes. The $q^m$, computed by the $m$-th agent, represents the re-evaluation
of rewards received in the local episodes, as well as the posterior distribution of
hidden decision states $z$. Based on $q^m$, the $m$-th agent computes the probability
mass function (pmf) of the indicator variables $c_m$ using (5.16) and makes a draw of $c_m$ according to this pmf.

The $q^m$ and $c_m$ are sent to the central node, at which the environments are grouped according to $c$ and the reevaluated rewards from the same group are assembled. Finally the rewards are allocated to the hidden decision states $z$ according to the posterior distribution of $z$, completing the update of $\overline{g}_n$ for each group $n$. The updated $\overline{g}$'s are sent back to the agents and the next iteration begins.

Figure 5.1 shows the block diagram of the distributive Gibbs-variational algorithm.

5.5 Experimental Results

We evaluate the performance of the multitask-RPR algorithm on two problems. The first is learning to reach goals in multiple partially observable stochastic environments. The second is multi-task multi-aspect classification.
5.5.1 Results on Multiple Partially Observable Stochastic Environments

Problem Description

In this problem, there are $M = 10$ environments and each environment is a grid-world, i.e., an array of rectangular cells. Of the ten environments, three are distinct and are shown in Figure 5.2, the remaining are duplicated from the three distinct ones. Specifically, the first three environments are duplicated from a distinct one, the following three environments are duplicated from another distinct one, and the last four environments are duplicates from the third distinct one. We assume 10 sets of episodes, with the $m$-th set collected from the $m$-th environment. Since the 10 environments are not all the same, we do not expect a single RPR will suffice to represent a good policy across all the environments. Meanwhile, the 10 environments are not all different, hence we do not think having a separate RPR for each environment is a good choice either. The approach we take to handle this problem is to apply the multitask-RPR algorithms developed in previous sections and let the algorithms automatically find out which environments should share a RPR and which should not.

In each one of the distinct environments shown in Figure 5.2, the agent can take five actions $\{\text{move forward, move backward, move left, move right, stay}\}$. In each cell of the grid-world environments, the agent can only observe the openness of the cell in the four directions. The agent then has a total of 16 possible observations indicating the $2^4 = 16$ different combinations of the openness of a cell in the four orientations. The actions (except the action stay) taken by the agent are not accurate and have some noises. The probability of arriving the correct cell by taking a move action is 0.7 and the probability of arriving at other neighboring cells is 0.3. The perception is not accurate either and has a certain level of noise. The probability of correctly
Figure 5.2: The three distinct grid-world environments, where the goal is designated by the basket of food, each block indicates a cell in the grid world, and the two gray cells are occupied by a wall. The red dashed lines in (a) and (c) indicate the similar parts in the two environments. The agent locates himself by observing the openness of a cell in the four orientations. Both the motion and the observation are noisy. Observing the openness is 0.8 and the probability of making a mistaken observation is 0.2. The agent will receive a unit reward if he reaches the goal and zero reward otherwise. The agent does not know how many cells there are in the grid world, nor which cell he is in; the only information available to him comprises the sequences of actions, observations, and rewards (i.e., the episodes). The objective of the agent in each environment is to take actions to maximize the discounted cumulative reward.
According to our specification of the rewards, the objective is equivalent to reaching the goal (indicated by a basket of food in the figures) within the minimum number of steps.

**Algorithms Learning and Evaluation**

For each environment \( m = 1, 2, \cdots, 10 \), there is a set of episodes \( D_m \), which are collected by simulating the agent-environment interaction using the models described above and a policy \( \Pi \) that the agent follows to determine his actions. The conditions for the policy \( \Pi \), which is given by Theorem 4 Chapter 4, are very mild, and are satisfied by even a random policy. However, a purely random policy produces long episodes (it takes a long time for a random agent to reach the goal), which makes the learning very slow. To accelerate learning, we here use a semi-random policy \( \Pi \), which is simulated by the rule that, with probability 0.5, \( \Pi \) uses the PBVI algorithm (Pineau et al. 2003) to choose the action and, with probability 0.5, \( \Pi \) chooses the action by sampling uniformly from \( \mathcal{A} \). Since PBVI is an approximately good policy, the semi-random \( \Pi \) thus simulated ensures reaching the goal within a reasonable number of steps.

Applying the algorithm in Table 5.2 to the 10 sets of episodes, we obtain a set of RPRs, which includes a local policy in each hidden decision state and the dynamics of the decision states — these dynamics are the internal representation of an environment as perceived by the agent, taking into account the perception aliasing. Here 6 hidden decision states are used for all environments — other numbers give similar results. After obtaining the RPR policy for each environment, we test it by executing the policy 1000 independent times, each time starting randomly from a cell in the grid worlds. The performance of the algorithm is evaluated by two performance measures: (a) the average success rate for the agent to reach the goal within 15 steps, and (b)
the average steps for reaching the goal. Each performance measure is computed from
the 1000 instances of policy execution, and is averaged over 20 independent trials — each trial consists of simulating independent $D_m$, $m = 1, 2, \ldots, 10$, learning a set of RPRs from these episodes, and executing each RPR policy in its corresponding environment.

Taking the average over the trials, we obtain one point on each curve in Figure 5.3(a) and Figure 5.3(b). Different points correspond to different numbers of episodes for each environment. We consider 16 different numbers of episodes in our results and they are 3, 4, 12, 24, 60, 120, 240. From Equations (5.17) and (5.18), the choice of the precision parameter $\alpha$ influences the probability of sampling a new cluster; it hence influences the resulting number of distinct RPR parameters $\Theta$. West (1992) has pointed out that the choice of $\alpha$ is closely related to the number of episodes. Despite of the influences of $\alpha$ over the number of distinct $\Theta$, the choice of $\alpha$ is not so delicate in our problem. We only need set $\alpha$ approximately around the numerical range of $C_m^o/G_0$ and the results will not change much. Here the results are obtained by setting the $\alpha$ as $10 \times 3^2, 10 \times 3^3, \ldots, 10 \times 3^{15}$ for the different numbers of episodes.

In the results as summarized in Figures 5.3(a) and 5.3(b), we compare three algorithms — the multitask-RPR (RPR-MTL), the RPR for single-task learning (RPR-STL), and the RPR by a simple pooling of all tasks (RPR-Pooling). As before, a task here refers to “learning the RPR in a single environment”. The RPR-MTL is the multitask-RPR algorithm presented in Table 5.2, the RPR-STL refers to 10 independent RPRs each learned based on the episodes from one of ten environments, and the RPR-Pooling refers to a single RPR learned from a simple pooling of the episodes across all ten environments. It is clear that the RPR-STL treats the 10 tasks as independently of each other, the RPR-Pooling assumes the 10 tasks are identical. In each of the figures, the results of the RPR-MTL are denoted by the red solid lines.
Figure 5.3. Results on the problem of multiple stochastic environments (a) Average success rate for the agent to reach the goal within 15 steps (b) Average step for the agent reaching the target. The red solid line with stars denotes the results of RPR-MTL; the green dotted line with diamonds denotes the results of RPR-STL; and the blue dashed line with circles denotes the results of RPR-Pooling.

with stars; the results of RPR-STL are denoted by green dotted lines with diamonds; and the results of the RPR-Pooling are denoted by the blue dashed lines with circles.
Figures 5.3(a) and 5.3(b) show that when the number of training episodes is very small, pooling episodes from all environments is helpful since the episodes from each environment are so scarce that they do not provide enough information for learning a good RPR. When the number of training episodes increases, pooling them together begins to show disadvantages since the ten environments are not all the same. The RPR-MTL algorithm automatically finds out appropriate sharing among the environments to improve the overall performance. Figures 5.3(a) and 5.3(b) show this point. The general performance of the RPR-MTL is much better than that of the RPR-STL and the RPR-Pooling except several points at the beginning, where the episodes are so extremely scarce that it is difficult to find even the sharing structure.

Analysis of the Sharing Mechanism

We further investigate the sharing mechanism of the RPR-MTL by plotting Hinton diagrams. The Hinton diagram (Hinton and Sejnowski 1986) is a qualitative diagram of the elements of a data matrix. Each element is represented by a square whose size is proportional to the magnitude. In our case here, the data matrix is the between-task similarity matrix (Xue et al. 2006) defined as follows: the between-task similarity matrix is a symmetric matrix of size $N_{task} \times N_{task}$ (where $N_{task}$ denotes the number of tasks and $N_{task} = 10$ in the present experiment), the $(i, j)$-th element measuring the frequency that task $i$ and task $j$ belong to the same cluster (i.e., they result in the same RPR). In order to avoid the bias due to any specific set of episodes, we perform 20 independent trials and average the similarity matrix over the 20 trials. In each trial, if tasks $i$ and $j$ belong to one cluster, we add 1 at $(i, j)$ and $(j, i)$ of the matrix. We select four typical points with the number of training episodes equal to 3, 10, 60, 120 and compute the between-task similarity matrices for these points. The Hinton diagrams for the three matrices are plotted in Figures 5.4(a) and 5.4(b).
Figure 5.4: Results on the problem of multiple stochastic environments: Hinton diagrams for the between-task similarity matrix, where the number of training episodes is equal to (a) 3 (b) 10 (c) 60 (d) 120.

As mentioned at the beginning of this section, environments 1,2,3 are duplicates of the distinct environment in Figure 5.2(a), environments 4,5,6 are duplicates of the distinct one in Figure 5.2(b) and environments 7,8,9,10 are duplicates of the distinct one in Figure 5.2(c). We expect that a good algorithm should automatically find out the right environments to share a RPR — i.e., only those that are similar to each other should share.

The Hinton diagrams Figures 5.4(a) and 5.4(b) show that when the number of training episodes is small, environments 1, 2, 3, 7, 8, 9, 10 have higher frequency
of sharing the same RPR. This sharing can be intuitively justified by first recalling that these environments are duplicates of Figures 5.2(a) and 5.2(c) and then noting that the parts circumscribed by red dashed lines in Figures 5.2(a) and 5.2(c) are quite similar. Meanwhile the Hinton diagrams also show a weak sharing between environments 4,5,6,7,8,9,10, which are duplicates of Figures 5.2(b) and 5.2(c). This is probably because the episodes are very few at this stage, and pooling episodes from environments that are not so relevant to each other could also be helpful. This explains why, in Figure 5.3(a), the performance of the RPR-Pooling is as good as that of the RPR-MTL when the number of episodes is small.

As the number of episodes progressively increases, the ability of the RPR-MTL to identify the correct sharing becomes better and better and, as seen in Figures 5.4(b) and 5.4(c) only those episodes from relevant environments are pooled together to enhance the performance — a simple pooling of all episodes together deteriorates the performance. This explains why the RPR-MTL greatly outperforms the RPR-Pooling with the increase of episodes. Meantime, the RPR-STL does not perform well either when the episodes are not enough. However, when there are more and more episodes from each environment, the RPR-STL learns and performs steadily better until it outperforms the RPR-Pooling and becomes comparable to the RPR-MTL.

5.5.2 Results on Multi-task Multi-Aspect Classification

Problem Description

Multi-aspect classification refers to the problem of identifying the class label of an object using observations from a sequence of viewing angles. This problem is generally found in applications where the object responds to interrogations in an angle-dependent manner. In such cases, an observation at a single viewing angle carries the information specific to only that angle and the nearby angles, and one requires
Figure 5.5: A typical configuration of multi-aspect classification of underwater objects.

observations at many viewing angles to fully characterize the object.

More importantly, the observations at different viewing angles are not independent of each other, and correlated in a complicated and yet useful way. The specific form of the angle-dependency is dictated by the physical constitution of the object as well as the nature of the interrogator — typically electromagnetic or acoustic waves. By carefully collecting and processing observations sampled at densely spaced angles, it is possible to form an image, based on which classification can be performed. An alternative approach that is well-recognized and widely used is treat the observations as a sequence and characterize the angle-dependency by a hidden Markov model (HMM) (Runkle et al. 1999).

In this section, we consider multi-aspect classification of underwater objects based on acoustic responses of the objects. Figure 5.5 shows a typical configuration, where the cylinder represents an underwater object, and the agent moves angularly to change the viewing angle $\varphi$. We assume the angular motion is one-dimensional, i.e., the agent moves clockwise or counterclockwise on the page, but does not move out of the page. Then the state space consists of any angle between $[0^\circ, 360^\circ]$. In practice the continuous space is discretized into a number of angular sectors, leading to a finite set of states $S$. Assuming that the agent moves by a constant angular step
Figure 5.6: Frequency-domain acoustic responses of the five underwater objects (a) Target-1 (b) Target-2 (c) Target-3 (d) Target-4 (e) Clutter.

and that he is uniformly at any angle in a state, the state transition probabilities can be computed using simple geometry (Runkle et al. 1999).

In our experiment, there are a total of five objects, four of them are targets of interest and one of them represents the clutter. The frequency-domain acoustic responses of these objects are shown in Figure 5.6. We aim to distinguish each target from the clutter, and this gives a multi-task problem where each task is a multi-aspect classification.
The RPR for Multi-Aspect Classification

We apply the RPR to the multi-aspect classification problem. Our approach is distinct from the HMM approach (Runkle et al. 1999) in two major respects. First, the RPR is a control model and it aims to optimize the value function, instead of the likelihood function. Since the RPR takes into account a reward structure, it can be more flexible in specifying the learning objective. Second, the RPR embraces all objects in the same representation, instead of having a separate model for each individual object. As a result, it is a discriminative model instead of a generative model.

The RPR does not manipulate the angular states — it works directly with observations. Since classification is treated as a control problem in the RPR, we need two extra components, actions and rewards, to complete the specification. We consider three actions \{declare as target, declare as clutter, move and sense\}. When the agent takes action move and sense, it moves 5° clockwise and collects an observation. The reward structure is specified as follows. A correct declaration receives a reward of 100 units, a false declaration receives a reward of zero, and the action move and sense receives a reward of 10 units.

The episodes used in learning the RPR consist of a number of observation sequences, each observation is associated with the action move to sense and the terminal action in each episode is the correct declaration. The correction declaration is available because the episodes in this problem are the training data in standard classification, hence the ground truth of class labels is known.

The Classification Results

The raw data are shown in Figure 5.6 for the five objects we are considering. Each datum is the response of an object is measured uniformly at every 1°. Each raw datum is converted into a feature vector using matching pursuit (McClure and L. Carin), and
Figure 5.7: (a) Average classification rate in the problem of shell target classification (b) Zoomed-out version of the figure in (a), when the number of episodes is equal or less than 40. Red solid line with stars denotes the results of RPR-MTL; green dotted line with diamonds denotes the results of RPR-STL and blue dashed line with circles denotes the results of RPR-Pooling.

the feature vectors are further discretized by vector quantization (Gersho and Gray 1992) to produce a finite code-book. As mentioned earlier, we have a total of four
tasks, each task is to distinguish each of the four targets from the clutter.

As in the problem of multiple environments, we compare three algorithms, the RPR-MTL, the RPR-STL, and the RPR-Pooling, the details of which are described on page 127 in Section 5.5.1. The performance is evaluated by the correct classification rate. Figure 5.7 summarizes the performance as a function of the number of training episodes. Each point in the figure is an average from 20 independent trials, in each of which we start the training episodes at random initial angles and collect 10 observations for each episode.

Figure 5.7 shows that the RPR-MTL achieves a performance comparable to the best between the RPR-STL and the RPR-Pooling. Although the RPR-STL and the RPR-Pooling each performs well when the number of training episodes is favorable to them, the RPR-MTL constantly performs well since it adaptively adjusts the sharing among tasks as the number of training episodes changes, such that the sharing is appropriate regardless of the number of episodes.

**Analysis of the Sharing Mechanism**

The Hinton diagram of the between-task similarity matrix is shown in Figures 5.8(a), 5.8(b), 5.8(c), for the cases when there is 10, 30, 110, 150 training episodes, respectively.

Figure 5.8(a) shows that when the episodes are scarce, all tasks tend to share information with each other. With the increase of episodes, the sharing patterns begin to change. Shown in Figure 5.8(b), task 1 and task 2 tend to cluster, and so do task 3 and task 4, as the number of episodes increases to 30. When the number increases to 110, shown in Figure 5.8(c), the two clusters in Figure 5.8(b) are further enhanced. Shown in Figure 5.8(d), as more episodes are added, task 1 and task 2 splits, and the other cluster still remains.
Figure 5.8: Results on the problem of shell target classification: Hinton diagram for the between similarity matrix, where the number of training episodes is equal to (a) 10 (b) 30 (c) 110 (d) 150.

Now let us check if the task clusters shown in Figure 5.8 are correct. First, we note from Figure 5.6 that target 1 and 2 are similar and so are target 3 and 4. Since each task is to distinguish one of the four targets from the clutter, it can be equivalently said that task 1 and 2 are similar and so are task 3 and 4. Therefore the two clusters in Figures 5.8(b) and 5.8(c) are justified. The reason for task 1 and 2 to split in 5.8(d) can be attributed to the fact the features of target 1 and 2 are very effective (probably because of a low level of noise) and therefore, when the training episodes are enough, the two corresponding tasks become less dependent — i.e., each task can perform well based on its own data.
5.6 Conclusions for RPR-MTL

In this chapter we have extended the regionalized policy representation (RPR) to multitask learning, where the RPRs for multiple environments are learned simultaneously under a unified framework. The sharing is achieved by placing a Dirichlet process (DP) prior on the RPRs across all environments. Both the Gibbs sampling algorithm and a hybrid Gibbs-variational algorithm are presented for learning the RPRs empirically from the episodes across the environments.

The multitask-RPR is demonstrated by experimental results, which show that while the two competing algorithms are sensitive to the number of episodes, the multitask-RPR constantly performs the best regardless of the number of episodes. The good performance is attributed to the ability of the multitask-RPR to automatically identify useful experiences from related environments to enhance the performance. The multitask-RPR adaptively adjusts sharing patterns to offset the changes in the episodes and hence has addressed the problem of how to positively transfer the information from related environments to the benefit of improving learning in the present environment.
Chapter 6

Reward-Directed Bayesian Classification with POMDP

In this chapter we employ POMDPs as a tool to perform active sequential feature selection in classification problems. It extends the classic Bayesian classifier to the case in which the feature selection as well as label prediction is reward-directed. The resulting reward-directed Bayesian classifier presents a novel application of POMDPs.

6.1 Introduction

A traditional Bayesian classifier can be viewed as a 4-tuple \(\langle C, \mathcal{X}, \mathcal{O}, \Omega_{c,o_1,o_2,...,o_d}\rangle\), where \(C\) is a finite set of class labels and \(\mathcal{X} = \{x_1, x_2, \cdots, x_d\}\) is a finite set of class features, \(\mathcal{O} = \mathcal{O}_1 \times \mathcal{O}_2 \times \cdots \times \mathcal{O}_d\) with \(\mathcal{O}_i\) defining the set of possible observations of \(x_i\), and \(\Omega\) is the observation function with \(\Omega_{c,o_1,o_2,...,o_d}\) denoting the probability of observing \([o_1, o_2, \cdots, o_d] \in \mathcal{O}\) given class label \(c \in C\). The goal of Bayesian classification is to correctly predict the class label of any given observation vector in \(\mathcal{O}\). Denoting by \(p(c)\) the prior distribution of class labels, its posterior distribution is computed by Bayes rule,

\[
p(c|o_1, o_2, \cdots, o_d) = \frac{p(o_1, o_2, \cdots, o_d|c)p(c)}{\sum_{c' \in C} p(o_1, o_2, \cdots, o_d|c')p(c')}
\]

(6.1)

A traditional Bayesian classifier makes predictions based on observations of all features in \(\mathcal{X}\), with no mechanism for selecting the features to observe.

In many applications such as medical diagnosis, observing a feature may entail expensive instrumental measurement and time-consuming analysis. Given a limited budget, time, or other resources, it may not possible to observe all features. Moreover,
some features may not be as helpful to diagnosis as others. Selectively observing the most useful features is important in minimizing the cost (negative reward). In other respects, some diseases may be more serious and require more accurate prediction than others. In such scenarios the classifier must jointly maximize prediction accuracy and observation reward (negative cost) by quantifying the reward/cost in a unified manner. In this chapter we refer to this type of classification as reward-directed classification.

The problem of reward-directed classification has been investigated previously by Bonet and Geffner (1998) and Guo (2003), under the naive Bayes assumption that the features \([x_1, x_2, \ldots, x_d]\) are independent conditional on the class label, i.e., \(p(o_1, o_2, \ldots, o_d|c) = \prod_{i=1}^d p(o_i|c)\) for all \([o_1, o_2, \ldots, o_d] \in \mathcal{O}\). This assumption is very strong and can result in serious degraded classification performance in real applications, where the assumption is often violated.

We propose a reward-directed classification algorithm in which the naive Bayes assumption is relaxed. The key idea is to use a Markov chain as an internal representation of feature dependence. We demonstrate using real medical data that a Markov chain with a moderate number of states can significantly improve the classification accuracy as well as reduce observation cost.

### 6.2 Intuitive Description of the Reward-Directed Classifier (RDBC)

Before proceeding to the mathematical formulation, we give an intuitive description of the RDBC, emphasizing the aspects in which it is different from the traditional Bayesian classifier.

The features used by the RDBC for prediction are not given \textit{a priori} and the RDBC is responsible to choose the features to use, from a given feature set \(\mathcal{X}\). The
features are selected and observed sequentially. Assume the RDBC is instructed to observe \( n \) features and a given feature can be repeatedly observed. At the time of making the \( i \)-th observation, the RDBC has collected a list of past observations and the associated feature indices \( \varepsilon_i = [a_0 o_1, \cdots, a_{i-2} o_{i-1}] \), where \( o_j \) is an observation of feature \( x_{a_{j-1}}, j = 1 \cdots i - 1 \). See Figure [6.1] for a graphical illustration of the relations of \( o \) and \( a \). In choosing \( a_{i-1} \) (the feature index of \( o_i \)), the RDBC takes into account the list \( \varepsilon_i \) and the conditional distribution \( p(o_i o_{i+1} \cdots o_n | \varepsilon_i, a_{i-1} a_i^* \cdots a_{n-1}^*) \), where \( a_j^*, i + 1 \leq j \leq n \), is the optimal feature index for \( o_j \) given the RDBC is instructed to observe \( n - j + 1 \) features \( [o_j \cdots o_n] \). A policy of feature selection is learned with the goal of simultaneously maximizing the reward of correct prediction and minimizing the cost of observation and false prediction.

The RDBC uses an internal Markov chain to represent the feature dependence of a given class. Let \( o_1, \cdots, o_n \) be the observations of \( n \) features \( x_{a_0}, \cdots, x_{a_{n-1}} \in X \), respectively. The RDBC expresses the class-conditional probability as

\[
p(o_1, \cdots, o_n | c, a_0, \cdots, a_{n-1})
\]

\[
= \sum_{s_0 \cdots s_n \in S_c} p(o_1, \cdots, o_n, s_0, \cdots, s_n | a_0, \cdots, a_{n-1})
\]

(6.2)

where \( s_i \) is the internal state of \( o_i \), \( i = 1 \cdots n \), \( S_c \) is a finite set of internal states defined for class \( c \), and \( s_0 \) is an initial state. See Figure [6.1] for a graphical illustration of the relations of \( s, o \), and \( a \).

It is clear that such a representation is sensitive to the order of \( \{o_1 \cdots o_n\} \) and the associated \( \{a_0 \cdots a_{n-1}\} \), which implies that different permutations of \( \{(a_0 o_1), \cdots, (a_{n-1} o_n)\} \) appear different to the representation. This order information is necessary in the sequential feature selection process. However, the order sensitivity may make \( p(o_1 \cdots o_n | c, a_0 \cdots a_{n-1}) \) different for different permutations of \( \{(a_0 o_1), \cdots, (a_{n-1} o_n)\} \), which is harmful as this probability is being treated as the joint probability of \( \{o_1, \cdots, o_n\} \) conditional on \( \{c, a_0, \cdots, a_{n-1}\} \) and should remain invariant.
regardless of the order. To preserve the order-invariance, training of this representation (i.e., estimation of its state transition probabilities and observation probabilities) must be based on a sufficient number of permutations of each \{ \((a_0 o_1), \cdots, (a_{n-1} o_n)\) \} to make the permutations equally probable in the resulting representation.

\[ \text{state: } s_0 \to s_1 \to \cdots \to s_n \]

\[ \text{observation: } o_1 \to o_2 \to \cdots \to o_n \]

\[ \text{Feature index: } a_0 \to a_1 \to \cdots \to a_{n-1} \]

**Figure 6.1:** Representation of feature dependence for a given class in the RDBC. Each node is dependent on (and only on) the nodes that emanates a directed edge to it. Though the internal state \(s\) is Markovian, the observation \(o\) is not, therefore the dependence among \(o_1 \cdots o_n\) is well represented.

### 6.3 Mathematical Formulation of the RDBC

The proposed RDBC can be formulated as a Partially Observable Markov Decision Process (POMDP) \cite{kaelbling1998} with a specialized state structure. Specifically, the RDBC is defined as a 8-tuple \( \langle C, X, \mathcal{O}, S, A, T_{sa}, \Omega_{ao}, R \rangle \), where \(C\) is a finite set of class labels and \(X = \{x_1, x_2, \cdots, x_d\}\) is a finite set of class features; the remaining 6 elements are the elements in a standard POMDP and they are specified below.
The $O$ is a union of disjoint sets $O_1, O_2, \ldots, O_d$, with $O_i$ denoting the set of possible observations of $x_i$. The $S$ is a union of disjoint sets $S_1, S_2, \ldots, S_{|C|}$, and $\{t\}$, with $S_c$ the set of internal states for class $c$, $t$ the terminal state, and $|C|$ denoting the cardinality of $C$. The $A = \{1, \ldots, d, d + 1, \ldots, d + |C|\}$ is the set of possible actions; letting $a$ be an action variable, $a = i$ denotes “observing feature $x_i$” and $a = d + c$ denotes “predicting as class $c$”.

The $T$ are the state-transition matrices with $T_{ss'}^a$ denoting the probability of transiting to state $s'$ by taking action $a$ in state $s$. The RDBC prohibits transition between internal states of different classes, therefore $T_{ss'}^a = 0$, $\forall a \in A$, $s \in S_c$, $s' \in S_{c'}, c \neq c'$. In addition, the RDBC has a probability-one transition from any non-terminal state to the terminal state when the action $a$ is “predicting”, i.e., $T_{ss'}^a = 1$, $\forall d + 1 \leq a \leq d + |C|$, $s \neq t$, $s' = t$; and it has a uniformly random transition from the terminal state to an internal state of any class when the action $a$ is “observing a feature”, i.e., $T_{ss'}^a = 1/(|S_c||C|)$, $\forall 1 \leq a \leq d$, $s = t$, $s' \in S_c$. The state transitions in the RDBC are illustrated in Figure 6.2 for a two-class problem ($|C| = 2$), with two internal states defined for class 1 ($|S_1| = 2$) and three internal states defined for class 2 ($|S_2| = 3$).

The $\Omega$ are the observation functions with $\Omega_{s's_o}^a$ denoting the probability of observing $o$ after performing action $a$ and transiting to state $s'$. The $R$ is the reward function with $R(s, a)$ specifying the expected immediate reward that is received by taking action $a$ in state $s$.

Using the definitions of the RDBC, we have the expansion

\[
p(o_1 \cdots o_n, s_0 \cdots s_n | a_0 \cdots a_{n-1}) = p(s_0) \prod_{i=1}^n T_{s_i s_{i-1}}^{a_{i-1}} \Omega_{a_{i-1}}^{o_i} \quad (6.3)
\]

where we assume that given class $c$ the initial state is uniformly distributed in $S_c$, i.e., $p(s_0) = \frac{1}{|S_c|}$ given class $c$. When $|S_c| = 1$, we have $p(s_i | s_{i-1}, a_{i-1}) = 1$ and consequently $p(o_1 \cdots o_n, s_0 \cdots s_n | a_0 \cdots a_{n-1}) = p(s_0) \prod_{i=1}^n p(a_i | s_i, a_{i-1})$, which is substi-
Equation (6.4) shows that the distribution of observations conditional class \( c \) reduces to a naive Bayes expression when a single state is defined for class \( c \). This demonstrates that in order to capture feature dependence of a class, multiple states must be defined for the class.

### 6.4 Learning of the RDBC

There are two steps in the learning of the RDBC. One is to learn the POMDP model, which is a six tuple \((S, A, T, O, \Omega, R)\) as defined in Section 1.3.1. The other is to
find the optimal policy based on the POMDP model we have learned.

The finite set of actions \( \mathcal{A} \) and observation \( \mathcal{O} \) are defined in Section 6.3 and the finite set of states \( \mathcal{S} \) is the union of \(|S_c|\), the number of internal states for each class \( c \). We estimate the transition matrices \( T \) and observation functions \( \Omega \) from a training data set, using the standard Expectation-Maximization (EM) method (Koenig and Simmons 1996; Rabiner 1989). The determination of \(|S_c|\) can be implemented using variational Bayesian approach (An et al. 2006). For the purpose of the completion of the thesis, the POMDP model learning via EM and Variational Bayesian EM (VBEM) are given in the Appendix of this chapter. Both the POMDP learning algorithms using EM and VBEM only learn the transition matrices \( T \) and observation functions \( \Omega \) and does not consider the learning of the reward function \( R \). Fortunately, however, the reward functions in this problem, which is given a priori or manually set, is only dependent on action \( a \) and independent of the hidden states \( s \). Then the traditional POMDP model learning (Koenig and Simmons 1996; Rabiner 1989) can be used here. Upon completion of these, one obtains the RDBC representation of the class-conditional distribution of observations, as given by (6.2) and (6.3).

Once we obtain the complete POMDP model, we need find an optimal policy maximizing the expected future reward (value) (Kaelbling et al. 1998). In the previous chapters, we have discussed many approximate model-based methods for the POMDP planning, which can be equivalently used here to find the policy. The point-based value iteration (PBVI) (Pineau et al. 2003) is used in our experiment.

### 6.5 Experimental Results

We evaluate the performance of the proposed RDBC on the Pima Indians Diabetes dataset (Turney 1995), a public data set available at http://www.ics.uci.edu/mlearn/MLSummary.html. The dataset consists of 768 medical instances for diabetes di-
agnosis. Each instance consists of 8 features, representing 8 distinct medical measurements. The observation costs of the 8 features, which are summarized in Table 6.1, are based on information from the Ontario Ministry of Health (1992)\cite{OntarioMinistryOfHealth1992}. Each feature is quantized into 5 uniform bins, yielding a set of $8 \times 5 = 40$ possible observations, i.e., $|\mathcal{O}| = 40$. Each instance has a diagnostic result of either “healthy” or “diabetes”, which are referred to class 1 and class 2 in our results. The 768 instances are randomly split into a training set of 512 instances and a testing set of 256 instances. For each experimental setting, we perform 10 independent trials of the random split and generate the mean and standard deviation of the results from the 10 trials.

Table 6.1: Observation Cost of the Pima dataset

<table>
<thead>
<tr>
<th>Feature Index</th>
<th>Feature Description</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>number of times pregnant</td>
<td>$1.00</td>
</tr>
<tr>
<td>2</td>
<td>glucose tolerance test</td>
<td>$17.61</td>
</tr>
<tr>
<td>3</td>
<td>diastolic blood pressure</td>
<td>$1.00</td>
</tr>
<tr>
<td>4</td>
<td>triceps skin fold thickness</td>
<td>$1.00</td>
</tr>
<tr>
<td>5</td>
<td>serum insulin test</td>
<td>$22.78</td>
</tr>
<tr>
<td>6</td>
<td>body mass index</td>
<td>$1.00</td>
</tr>
<tr>
<td>7</td>
<td>diabetes pedigree function</td>
<td>$1.00</td>
</tr>
<tr>
<td>8</td>
<td>age in years</td>
<td>$1.00</td>
</tr>
</tbody>
</table>

There are 10 actions (i.e., $|\mathcal{A}| = 10$), including 8 observation actions and 2 prediction actions. We consider three configurations of internal states for the two classes. In the first configuration, class 1 has 6 internal states and class 2 has 5; in the second configuration, both classes have 10 internal states; in the third configuration, both classes have 1 internal state, which is the naive Bayes case. For a given state configuration, the reward function $R(s, a)$ is constructed as follows: when action $a$ is one of the 8 observation actions, $R(s, a) = -(\text{cost of } x_a)$ regardless of $s$; when action $a$ is one of the 2 prediction actions, $R(s, a) = 50$ if $s \in \mathcal{S}_a$ (correct prediction) and $R(s, a) = -\lambda$ if $s \notin \mathcal{S}_a$ (false prediction), where $\lambda$ is the cost of a false prediction.
We vary $\lambda$ in the range [$0, $200$] and present each result as a function of $\lambda$.

The state transition probabilities involving the terminal state are computed analytically as in Section 6.3. The remaining entries of $T_{ss'}$ as well as $\Omega_{os}$ are estimated from the training data set. For each training instance, the 8 observations (of 8 features), denoted $o_1, o_2, \cdots, o_8$, are randomly permuted to produce 20 permuted versions of $\{(a_0 o_1), (a_1 o_2), \cdots, (a_7 o_8)\}$ (where $a_0 = 1$, $a_1 = 2$, $\cdots$, $a_7 = 8$). The 512 training instances yield $512 \times 20$ permutations in total, which are used to estimated $T_{ss'}$ and $\Omega_{os}$. The PBVI [Pineau et al. (2003)] is used to learn the policy.

![Figure 6.3](image.png)

**Figure 6.3:** Correct classification rate as a function of false prediction cost. The mean and error bars are generated from 10 independent trials of random split of training and test instances.

In testing, the policy is followed until a prediction action is selected and executed to make $s$ transit to the terminal state to complete the present prediction phase. We compute three performance indexes at the end of each prediction phase: correct classification rate, observation cost accumulated, and feature repetition rate. Assume that at the end of a prediction phase, $n$ observations are made of $m < n$ features.
Figure 6.4: Observation cost averaged over test instances, as a function of false prediction cost. The mean and error bars are generated from 10 independent trials of random split of training and test instances.

Figure 6.5: Feature repetition rate averaged over test instances, as a function of false prediction cost. The mean and error bars are generated from 10 independent trials of random split of training and test instances.
(some features are observed more than once), then the feature repetition rate is computed as $(n - m)/n$.

The results obtained on the Pima data are summarized in Figures 6.3, 6.4, and 6.5. In each of the figures, black solid line denotes RDBC with 10 internal states for each class, red dashed line denotes RDBC with 6 internal states for class 1 and 5 internal states for class 2, green dotted line denotes the RDBC with 1 internal state for each class (the naive Bayes case).

Figures 6.3 and 6.4 show that, with a larger number of internal states for each class, higher correct classification rates are achieved at lower observation costs. This striking comparison can be explained by Figure 6.5, which shows that with increased internal states, the feature repetition rate is reduced. In the Pima data set, the features are noise free, so there is no sense in observing a given feature multiple times. The only reason that could lead to repetitive observation of the same feature is that the classifier is memoryless and does not remember that it has observed a feature before. It is obvious that a single state for each class does not provide memory to the classifier and therefore the naive Bayes classifier has the highest feature repetition rate. In contrast, the RDBC with 10 states for each class has the best memory, which gives it the lowest feature repetition rate. Repetitively observing the same feature is harmful in the Pima data: it increases cost and yet provides no new information to improve classification. This explains Figures 6.3 and 6.4.

**6.6 Conclusions for Cost-Sensitive Classification**

We have presented a reward-directed Bayesian classifier (RDBC) that preserves the feature dependence in its internal states. The proposed RDBC is formulated as a POMDP. The results on a diabetes dataset show the RDBC with a moderate number of states significantly improves over the naive Bayes classifier, both in prediction
accuracy and observation parsimony. It is also demonstrated that the RDBC performs better by using more states to increase its memory.

### 6.7 Appendix: POMDP Model Learning via EM and VBEM

For general purpose, we here give the derivation of the POMDP Mixture model learning via expectation and maximization (EM) and variational Bayesian EM (VBEM). When the number of mixture models is equal to one, it is reduced to a classical POMDP model. Some notations used in the derivation are listed here:

- $K$: number of POMDP mixture models;
- $N_k$: number of states for the $k$-th POMDP model;
- $cb$: size of code book;
- $O^n$: the $n$-th observation sequence with length $T_n$;
- $a^n$: the $n$-th action sequence with length $T_n - 1$;
- $T_n$: length of the $n$-th observation sequence;
- $\mu_k$: mixture coefficient;
- $\pi_k$: initial state distribution for the $k$-th POMDP model;
- $A^k$: action-dependent of transition matrix for the $k$-th POMDP model;
- $B^k$: action-dependent of observation matrix for the $k$-th POMDP model;

Definition of forward variable in the POMDP

$$
\alpha_t(i) = p(o_1, o_2, \ldots, o_t, s_t = i|a_1, a_2, \ldots, a_{t-1}, \Theta) \quad (6.5)
$$

The forward variable can be inductively solved, exactly the same as HMM (Rabiner 1989)

1) Initialization

$$
\alpha_1(i) = \pi_i B_{0i}(a_1) \quad (6.6)
$$
2) Induction
\[ \alpha_{t+1}(j) = \left( \sum_i \alpha_t(i) A_{ij}(a_t) \right) B_{j_{t+1}}(a_t) \]  \hspace{1cm} (6.7)

3) Termination
\[ p(o_1, \ldots, o_T|a_1, \ldots, a_{T-1}, \Theta) = \sum_i \alpha_T(i) \]  \hspace{1cm} (6.8)

Definition of backward variable in the POMDP
\[ \beta_t(i) = p(o_{t+1}, o_{t+2}, \ldots, o_T|s_t = i, a_t, a_{t+1}, \ldots, a_{T-1}, \Theta) \]  \hspace{1cm} (6.9)

The backward variable can be inductively solved (exact the same as HMM)
1) Initialization
\[ \beta_T(i) = 1 \]  \hspace{1cm} (6.10)

2) Induction
\[ \beta_t(i) = \sum_j A_{ij}(a_t) B_{j_{t+1}}(a_t) \beta_{t+1}(j) \]  \hspace{1cm} (6.11)

We define the variable
\[ \gamma_t(i) = p(s_t = i|o_{1:T}, a_{1:T-1}, \Theta) \]  \hspace{1cm} (6.12)

which can be expressed in terms of forward-backward variable
\[ \gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_i \alpha_t(i)\beta_t(i)} \]  \hspace{1cm} (6.13)

We define another variable
\[ \xi_t(i, j) = p(s_t = i, s_{t+1} = j|o_{1:T}, a_{1:T-1}, \Theta) \]  \hspace{1cm} (6.14)

which can be expressed in terms of forward-backward variable
\[ \xi_t(i, j) = \frac{\alpha_t(i)A_{ij}(a_t)B_{j_{t+1}}(a_t)\beta_{t+1}(j)}{\sum_i \alpha_t(i)\beta_t(i)} \]  \hspace{1cm} (6.15)
6.7.1 Learning the POMDP mixture model via expectation-maximization (EM)

Assume the data the robot collects from the whole environments are \( \{ O^n, a^n, 1 < n \leq N \} \), where \( \{ O^n, a^n, 1 \leq n \leq N \} \) are i.i.d, and each \( O^n, a^n \) is long sequences of observations and actions, respectively. We want to use a POMDP mixture model to model the data set. Thus, we have

\[
p(O^n|a^n, \Theta) = \sum_{k=1}^{K} \mu_k p_k(O^n|a^n, \Theta_k)
\]

(6.16)

and

\[
p(O^{1:N}|a^{1:N}, \Theta) = \prod_{n=1}^{N} \sum_{k=1}^{K} \mu_k p_k(O^n|a^n, \Theta_k)
\]

(6.17)

We seek the parameter \( \Theta = \{ \mu_k, \pi_k, A_k, B_k, k = 1, \ldots, K \} \) that maximize the logarithmic likelihood function \( \ln p(O^{1:N}|a^{1:N}, \Theta) \). Since \( \mu_k \geq 0 \) and \( \sum_{k=1}^{K} \mu_k = 1 \), we have a constrained problem,

\[
\max_{\mu, \pi, A, B} \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \mu_k p_k(O^n|a^n, \Theta) \quad \text{subject to} \quad \sum_{k=1}^{K} \mu_k = 1, \mu_k \geq 0, k = 1, \cdots, K
\]

(6.18)

Two sets of hidden variables are introduced here. One is \( z_n = k \), denoting the data sequence \( O^n \) is generated from the \( k \)-th POMDP model. The other is the \( s^n \), denoting the hidden state sequence for observation sequence \( O^n \). Then we have,

\[
\sum_{n=1}^{N} \ln \sum_{k=1}^{K} \mu_k p_k(O^n|a^n, \Theta) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} q(z_n = k) \frac{\mu_k p_k(O^n|a^n, \Theta, z_n = k)}{q(z_n = k)}
\]

152
Since

\[ \sum_{n=1}^{N} \sum_{k=1}^{K} q(z_n = k) \ln \mu_k p(O^n | a^n, \Theta, z_n = k) / q(z_n = k) \]

\[ = \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \ln \mu_k + \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \ln p(O^n | a^n, \Theta, z_n = k) - \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \ln \delta_n^k \]

Jensen’s inequality

\[ \geq \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \ln \mu_k + \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \sum_{s^n \in Q} q(s^n) \ln p(O^n, s^n | a^n, \Theta, z_n = k) - \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \ln \delta_n^k \]

Jensen’s inequality

\[ \geq \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \ln \mu_k + \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \sum_{s^n \in Q} q(s^n) \ln p(O^n, s^n | a^n, \Theta, z_n = k) - \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \ln \delta_n^k \]

(6.19)

where the first inequality in (6.19) becomes an equality when

\[ \delta_n^k \overset{df}{=} q(z_n = k) = p(\delta_n = k | O^n, a^n, \Theta) \]

\[ = \frac{p(O^n | \delta_n = k, a^n, \Theta) p(\delta_n = k | \Theta)}{\sum_{k=1}^{K} p(O^n | \delta_n = k, a^n, \Theta) p(\delta_n = k | \Theta)} = \frac{\sum_{i=1}^{N_k} \alpha_{T_n(i)}^k \mu_k}{\sum_{k=1}^{K} \sum_{i=1}^{N_k} \alpha_{T_n(i)}^k \mu_k} \]

(6.20)

The second inequality becomes an equality when \( q(s^n) = p(s^n | a^n, a^n, z_n = k, \Theta) \). Let

\[ Q(\hat{\Theta} | \Theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \ln \mu_k + \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \sum_{s^n \in Q} p(s^n | a^n, a^n, z_n = k, \Theta) \ln p(O^n, s^n | a^n, \Theta, z_n = k) \]

(6.21)

Since

\[ p(O^n, s^n | a^n, \Theta) = \pi_{s_1}^k B_{s_1, a_1}^k (a_1) \prod_{l=2}^{T_n} A_{s_{l-1}, s_l}^k (a_l) B_{s_l, a_l}^k (a_l) \]

(6.22)

153
substituting (6.22) to (6.21), we have

\[
Q(\hat{\mu}, \hat{A}, \hat{B}, \hat{\pi}|\mu, A, B, \pi) = \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \ln \mu_k
\]

\[
+ \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \sum_{s^n \in Q} p(s^n|a^n, a^n, z_n = k, \Theta) \ln p(O^n, s^n|a^n, \Theta, z_n = k)
\]

\[
= \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \ln \mu_k + \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{s^n \in Q} \delta_n^k p(s^n|a^n, a^n, z_n = k, \Theta) \ln \pi_{s_n}^k
\]

\[
+ \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{s^n \in Q} \delta_n^k p(s^n|a^n, a^n, z_n = k, \Theta) \ln \prod_{t=2}^{T_n} A_{s_{t-1}a_t}^k(a_t)
\]

\[
+ \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{s^n \in Q} \delta_n^k p(s^n|a^n, a^n, z_n = k, \Theta) \ln \prod_{t=1}^{T_n} B_{s_{t}a_t}^k(a_t)
\]

\[
= \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_n^k \ln \mu_k + \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{s^n \in Q} \delta_n^k p(s^n|a^n, a^n, z_n = k, \Theta) \ln \pi_{s_n}^k
\]

\[
+ \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{s^n \in Q} \delta_n^k p(s^n|a^n, a^n, z_n = k, \Theta) \ln \prod_{t=2}^{T_n} A_{s_{t-1}a_t}^k(a_t)
\]

\[
+ \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{s^n \in Q} \delta_n^k p(s^n|a^n, a^n, z_n = k, \Theta) \ln \prod_{t=1}^{T_n} B_{s_{t}a_t}^k(a_t)
\]

Iteratively finding \(Q(\hat{\mu}, \hat{A}, \hat{B}, \hat{\pi}|\mu, A, B, \pi)\) (E-Step) and maximizing \(Q(\hat{\mu}, \hat{A}, \hat{B}, \hat{\pi}|\mu, A, B, \pi)\) with respect to \(\hat{\mu}, \hat{A}, \hat{B}, \hat{\pi}\) (M-Step) constitute the Expectation-Maximization (EM) algorithm.

In the E-Step, we use the current estimates to compute \(\delta_n^k\). In the M-Step, we maximize \(Q(\hat{\mu}, \hat{A}, \hat{B}, \hat{\pi}|\mu, A, B, \pi)\) with respect to \(\{\hat{\mu}, \hat{A}, \hat{B}, \hat{\pi}\}\), the solutions of which are given analytically by
**M-step:** The first term of (6.23) becomes

\[
(1) = \sum_{k=1}^{K} \sum_{n=1}^{N} \sum_{s^n \in Q} \log \mu_k \delta_n^k \quad s.t. \sum_{k=1}^{K} \mu_k = 1
\]  
(6.24)

Introducing the Lagrange multiplier \( \gamma \), and setting the derivative equal to zero, we obtain

\[
\frac{\partial}{\partial \mu_k} \left( \sum_{k=1}^{K} \sum_{n=1}^{N} \log \mu_k \delta_n^k + \gamma \left( \sum_{k=1}^{K} \mu_k - 1 \right) \right) = 0
\]  
(6.25)

and

\[
\hat{\mu}_k = \frac{\sum_{n=1}^{N} \delta_n^k}{\sum_{k=1}^{K} \sum_{n=1}^{N} \delta_n^k}
\]  
(6.26)

The second term of (6.23) becomes

\[
(2) = \sum_{k=1}^{K} \sum_{n=1}^{N} \sum_{s^n \in Q} \log(\pi_{s_0}^k)p(s^n|o^n, a^n, z_n = k, \Theta) \quad s.t. \sum_{i=1}^{N_K} \pi_i^k = 1
\]

Introducing the Lagrange multiplier \( \lambda \), and setting the derivative equal to zero, we get:

\[
\frac{\partial}{\partial \pi_i^k} \left( \sum_{k=1}^{K} \sum_{n=1}^{N} \sum_{s^n \in Q} \log(\pi_{s_0}^k)p(s^n|o^n, a^n, z_n = k, \Theta) + \lambda \left( \sum_{i=1}^{N_K} \pi_i^k - 1 \right) \right) = 0
\]  
(6.27)

Then, we have

\[
\hat{\pi}_i^k = \frac{\sum_{n=1}^{N} \gamma_{k,n}^i \delta_n^k}{\sum_{i=1}^{N_K} \sum_{n=1}^{N} \gamma_{k,n}^i \delta_n^k}
\]  
(6.28)

also, taking the derivative of the third term of (6.23) with respect to \( A_{s_{t-1}s_t}^k \), with the restraint of \( \sum_{j=1}^{N_K} A_{i,j}^k(a) = 1 \), we have

\[
\hat{A}_{i,j}^k(a) = \frac{\sum_{n=1}^{N} \sum_{0 \leq t < T_a} \xi_{t}^{k,n}(i,j) \delta_n^k}{\sum_{j=1}^{N_K} \sum_{n=1}^{N} \sum_{s.t.a_{t}^n = a} \xi_{t}^{k,n}(i,j) \delta_n^k}
\]  
(6.29)
In the same way, we can obtain

$$\hat{B}_{i,c}^k(a) = \frac{\sum_{n=1}^N \sum_{\substack{0 \leq t \leq T_n \\text{s.t.} \ \gamma_{t}^k(i) = \delta_n}} \sum_{\substack{s.t.o_n^q = c \\text{s.t.} \ a_n^q = a}} K_n(i)}{\sum_{j=1}^{N_k} \sum_{n=1}^N \sum_{\substack{0 \leq t \leq T_n \\text{s.t.} \ \gamma_{t}^k(i) = \delta_n}} K_n(i)}$$

(6.30)

### 6.7.2 Learning the POMDP mixture model via variational Bayesian EM (VBEM)

The logarithmic marginal likelihood for all sequences is lower bounded by

$$\ln \prod_{n=1}^N p(O^n|a^n)$$

$$= \sum_{n=1}^N \ln \int_\Theta \sum_{s_n \in Q} \sum_{z_n=1}^K p(\Theta)p(O^n, s^n, z_n, \Theta|a^n)$$

Jensen’s inequality

$$\geq \sum_{n=1}^N \int_\Theta \sum_{s_n \in Q} \sum_{z_n=1}^K q(s^n, z_n, \Theta) \ln \frac{p(O_{1:T_n}^n, \Theta, s_{1:T_n}^n, z_n|a_{1:T_n-1}^n)}{q(s^n, z_n, \Theta)}$$

$$= \mathcal{L}(O|a)$$

(6.31)

where $\sum_{s_n \in Q} \sum_{z_n=1}^K \int_\Theta q(s^n, z_n, \Theta) = 1$. Assuming factorization $q(s^n, z_n, \Theta) = q(s^n, z_n) q(\Theta)$, we have

$$\mathcal{L}(O|a) = \sum_{n=1}^N \left\{ \ln p(O^n|a^n) - \text{KL} \left( q(s^n, z_n, \Theta) || p(s^n, z_n, \Theta|O^n, a^n) \right) \right\}$$

$$\approx \sum_{n=1}^N \left\{ \ln p(O^n|a^n) - \text{KL} \left( q(s^n, z_n) q(\Theta) || p(s^n, z_n, \Theta|O^n, a^n) \right) \right\}$$

(6.32)

VBEM maximizes the lower bound given by (6.31) or, equivalently, minimizes the KL distance between the variational posterior $q(s^n, z_n) q(\Theta)$ and the true posterior $p(s^n, z_n, \Theta|O^n, a^n)$. The maximization is achieved by alternatively solving for $q(s^n, z_n)$
and \(q(\Theta)\), keeping one fixed while solving for the other. Taking into consideration that \(\sum_{z_n=1}^K \sum_{s^n \in Q} q(z_n, s^n) = 1\) and \(\int_\Theta q(\Theta) = 1\), the solutions are can be easily obtained as

\[
q(s^n, z_n) \propto \exp \left\{ \langle \ln p(O^n, s^n, z_n|a^n, \Theta) \rangle_{q(\Theta)} \right\} \tag{6.33}
\]

\[
q(\Theta) \propto p(\Theta) \exp \left\{ \langle \ln p(O^n, s^n, z_n|a^n, \Theta) \rangle_{q(s^n, z_n)} \right\} \tag{6.34}
\]

where \(\langle \cdot \rangle_{q(\Theta)}\) denotes expectation with respect to \(q(\Theta)\). Since the complete data likelihood is

\[
p(O^n, s^n, z_n = k|a^n, \Theta) = p(z_n = k|a^n, \Theta)p(s^n, O^n|a^n, \Theta, z_n = k) = \sum_{s^n \in Q} q(s^n, z_n) \frac{p(O^n, s^n, z_n|a^n, \Theta)}{q(z_n, s^n)}
\]

\[
= \mu_k^k \prod_{i=1}^{N_k} (\pi_k^i)^{s^n_{1,i}} \left( \prod_{t=2}^{T_n} \prod_{i=1}^{N_k} \prod_{j=1}^{N_k} (A_k^{ij}(a))^{s^n_{t-1,i}, s^n_{t,j}} \right) \times \left( \prod_{t=1}^{T_n} \prod_{i=1}^{N_k} \prod_{c=1}^{cb} (B_k^{ic}(a))^{s^n_{t,i}, o^n_{t,c}} \right) \tag{6.35}
\]

where \(s^n_{t,i}\) denotes that at time \(t\), the hidden state \(s_t = i\) for the \(n\)th sequence. Then we have

\[
\ln p(O^n, s^n, z_n = k|a^n, \Theta) = \sum_{t=1}^{T_n} (S_{t-1}^n)^T \ln A_k^T S_t^n + \sum_{t=1}^{T_n} (S_{t}^n)^T \ln B_k^T a_t^n \tag{6.36}
\]

**VBM-step:** Substituting (6.36) into (6.34)

\[
\ln q(\Theta) = \ln p(\Theta) + \langle \ln p(z_n = k, s^n|a^n, \Theta) q(z_n = k, s^n) \rangle_{q(z_n = k, s^n)} + \langle \ln p(\pi^k) + \ln p(A^k) + \ln p(B^k) + \langle \ln \mu + \ln p(A + \ln B) \rangle_{q(s^n, z_n = k)} \rangle_{q(z_n = k)} \tag{6.37}
\]

For the POMDP mixture model, we choose conjugate exponential priors for the model parameters \(\{\mu, A, B, \pi\}\), assuming the model parameters are independent of
each other \textit{a priori},

\[ p(\Theta) = p(\mu) \prod_{k=1}^{K} p(\pi^k)p(A^k)p(B^k) \]  

(6.38)

where

\[ p(\mu) = \text{Dir}(\mu_1, \mu_2, \ldots, \mu_K|w_1^{(\mu)}, w_2^{(\mu)}, \ldots, w_K^{(\mu)}) \]  

(6.39)

\[ p(\pi^k) = \text{Dir}(\pi_1^k, \pi_2^k, \ldots, \pi_N^k|w_1^{(\pi^k)}, w_2^{(\pi^k)}, \ldots, w_N^{(\pi^k)}) \]  

(6.40)

\[ p(A^k) = \prod_{\alpha}^{N_k} \prod_{i=1}^{N_k} \text{Dir}(A_{i1}^k(a), \ldots, A_{iN_k}^k(a)|w_{i1}^{(A^k)}(a), \ldots, w_{iN_k}^{(A^k)}(a)) \]  

(6.41)

\[ p(B^k) = \prod_{\alpha}^{N_k} \prod_{i=1}^{N_k} \text{Dir}(B_{i1}^k(a), \ldots, B_{icb}^k(a)|w_{i1}^{(B^k)}(a), \ldots, w_{icb}^{(B^k)}(a)) \]  

(6.42)

and \text{Dir} represents Dirichlet distribution. Then we have

\[ w_{\mu,\text{new}}^k = w_{\mu}^k + \sum_{n=1}^{N} \delta_n^k \]  

(6.43)

\[ w_{i}^{\pi^k,\text{new}} = w_{i}^{\pi^k} + \sum_{n=1}^{N} \delta_k^n q(s^n_1 = i) \]  

(6.44)

\[ w_{ij}^{A^k,\text{new}}(a) = w_{ij}^{A^k}(a) + \sum_{n=1}^{N} \sum_{2 \leq t \leq T_n} \delta_n^k q(s^n_{t-1} = i, s^n_t = j) \]  

(6.45)

\[ w_{ic}^{B^k,\text{new}}(a) = w_{ic}^{B^k}(a) + \sum_{n=1}^{N} \sum_{2 \leq t \leq T_n} \delta_n^k q(s^n_t = i) \]  

(6.46)

\textbf{VBE-step}

\[ \ln q(s^n, z_n = k) \propto \langle \ln p(o^n, z_n = k, s^n|a^n, \Theta)q(\Theta) = \delta_n^k \langle \ln \mu \rangle_{q(\mu)} + (s^n_1)^T (\ln \pi^k)_{q(\pi^k)} \]
\[
+ \sum_{t=2}^{T_n} (S_{t-1}^n)^T \langle \ln A^k \rangle_{q(A^k)} (S_t^n)
\]

\[
+ \sum_{t=1}^{T_n} (S_t^n)^T \langle \ln B^k \rangle_{q(B^k)} \theta_t^n
\]

(6.47)

Comparing (6.36) and (6.47), we find that these two equations are identical except that the expectations are taken of the logarithm of the parameters. The natural parameter vector \( \phi(\Theta) \) is given by

\[
\Theta = (\mu, \pi^k, A^k, B^k)
\]

(6.48)

\[
\phi(\Theta) = (\ln \mu, \ln \pi^k, \ln A^k, \ln B^k)
\]

(6.49)

The expected natural parameter \( \bar{\phi} \) is given by

\[
\bar{\phi} = \langle \phi(\Theta) \rangle_{q(\Theta)}
\]

= \((\ln \mu)_{q(\mu)}, (\ln \pi^k)_{q(\pi^k)}, (\ln A^k)_{q(A^k)}, (\ln B^k)_{q(B^k)}\)

(6.50)

The updated parameter vector \( \tilde{\Theta} \) is computed by

\[
\tilde{\Theta} = \phi^{-1}(\langle \phi(\Theta) \rangle_{q(\Theta)})
\]

= \((\exp((\ln \mu)_{q(\mu)}), \exp((\ln \pi^k)_{q(\pi^k)}), \exp((\ln A^k)_{q(A^k)}), \exp((\ln B^k)_{q(B^k)}))\)

= \((\tilde{\mu}, \tilde{\pi}^k, \tilde{A}^k, \tilde{B}^k)\)
Then, we have

\[ \tilde{\mu} = \{ \tilde{\mu}_k \} = \exp \left( \psi(w^{(\mu)}_k) - \psi(\sum_{k=1}^{K} w^{(\mu)}_k) \right) \]  

(6.51)

\[ \tilde{\pi}^k = \{ \tilde{\pi}^k_i \} = \exp \left( \psi(w^{(\pi^k)}_i) - \psi(\sum_{i=1}^{N_k} w^{(\pi^k)}_i) \right) \]  

(6.52)

\[ \tilde{A}^k = \{ \tilde{A}^k_{ij}(a) \} = \exp \left( \psi(w^{(A^k)}_{ij}(a)) - \psi(\sum_{j=1}^{N_k} w^{(A^k)}_{ij}(a)) \right) \]  

(6.53)

\[ \tilde{B}^k = \{ \tilde{B}^k_{ic}(a) \} = \exp \left( \psi(w^{(B^k)}_{ic}(a)) - \psi(\sum_{c=1}^{cb} w^{(B^k)}_{ic}(a)) \right) \]  

(6.54)

and

\[ \delta^n_k = \frac{\tilde{\mu}_k p(O^n|a^n, z_n = k, \tilde{\Theta})}{\sum_{k=1}^{K} \tilde{\mu}_k p(O^n|a^n, z_n = k, \tilde{\Theta})} \]  

(6.55)
Chapter 7

Summary and Future Work

7.1 Contributions of the Thesis

In this thesis, we have proposed a number of new methods, frameworks, and algorithms, for model-based planing and model-free learning in partially observable stochastic environments. The contributions of the thesis can be divided into three parts, which are summarized in Subsections 7.1.1, 7.1.2, and 7.1.3, respectively.

7.1.1 Efficient POMDP algorithms

We have proposed three efficient new algorithms, namely the RBVI (Chapter 2), the ILSPI and the PLPI (Chapter 3), for planning in partially observable Markov decision processes (POMDPs), assuming the POMDP models are given.

The RBVI is an approximate value iteration algorithm. Based on the piecewise linearity of the finite-horizon optimal value function, the RBVI performs a piecewise linear regression, based on a finite set of belief samples and optimal values at the samples as computed with the Bellman equation. With properly selected belief samples, the RBVI can estimate the gradient (α-vector) of each hyper-plane with very high accuracy and represent the value function with just the right number of hyper-planes. Compared to pruning of spurious hyper-planes by linear programming, the regression is a much cheaper method. When the belief simplex is under-sampled, the RBVI can introduce errors into the estimated value function, like any other sample-based methods (PBVI, for example). Though we have not obtained an error analysis for the RBVI, the performance of the RBVI has been demonstrated to be comparable to that of PBVI and other state-of-the-art POMDP algorithms.
Technically, the RBVI makes an approximation in the belief simplex partition. The ellipsoidal approximation to the true polyhedral partition enables the RBVI to estimate the value functions efficiently. By maximizing the marginal likelihood function approximated with a variational approach, the RBVI finds the right number of hyper-planes to represent the value function. Experimental studies on benchmark problems show that the RBVI uses a significantly smaller number of hyper-planes to obtain the similar results to those of state-of-the-art POMDP algorithms.

The ILSPI is an approximate policy iteration algorithm for finding infinite horizon stationary policies for POMDPs. The ILSPI iteratively improves the control (actions) that policy $\pi$ produces at finite belief samples, and generalizes the control from belief samples to the entire belief simplex. The generalization is accomplished in the policy-evaluation step, by parameterizing the value function of $\pi$ as a linear span of basis functions, and estimating the parameters by minimizing the Bellman residual at belief samples. In the policy-improvement step, the parametric value function in plugged into the Bellman equation to yield the Q function $Q^\pi(b,a)$, which is used to improve the control at belief samples. The ILSPI derives its generalization ability from optimally selected basis functions, which is implemented by an efficient incremental procedure.

The PLPI is another approximate policy iteration algorithm. Like the ILSPI, the PLPI performs policy improvement at finite belief samples. However, instead of using a basis representation, the PLPI uses finite number of hyper-planes to represent the value function. The policy at belief samples is generalized to the entire belief simplex, through the $\alpha$-vectors that are estimated by an iterative application of the Bellman equation. At each iteration, the old $\alpha$-vectors are plugged into the righthand side and the left-hand side to output the new $\alpha$-vectors.

Both the ILSPI and the PLPI have been demonstrated on benchmark problems
and compared to state-of-art POMDP algorithms. The results have shown that the ILSPI and the PLPI are among the best algorithms, in terms of performance and CPU time.

7.1.2 New policy representations and algorithms for single and multiple partially observable stochastic environments

In the second part of this thesis, we have proposed the RPR framework for a single environment (Chapter 4) and multiple environments (Chapter 5), assuming the environments are partially observable stochastic and the models of the environments are unknown to the agents.

The RPR is a parametric framework for representing a stochastic policy, i.e., the probability of action given observed histories. The history in the RPR plays the role of belief state in POMDPs. The RPR memorizes histories via a Markov chain of hidden decision states driven jointly by actions and observations. The agent explores the environment via episodic interactions, and the episodes collected form the experiences that are fed into the learning algorithms to estimate the RPR parameters. Two learning algorithms have been presented, namely, maximum-value estimation (MV) and variational Bayesian (VB) learning. The RPR generalizes to unseen histories through the estimated parameters.

The RPR has been evaluated experimentally on benchmark problems and shown to compare favorably to other state-of-the-art model-free algorithms.

The RPR-MTL in Chapter 5 is a suite of RPRs unified by a common Dirichlet process (DP) that generates the prior distribution for each individual RPR. Each RPR is associated with an environment, and the inherent clustering properties of DP enable the transfer of episodes between the environments, by identifying relevance
of one environment to another. Episodes transfer augments the experience in each individual environment and enhances policy learning across all. We have presented the Gibbs sampling algorithm and the Gibbs-variational algorithm to jointly learn the suite of RPRs. The Gibbs-variational algorithm is easily implemented via distributed computation, in which the agents in different environments exchange intermediate variables, but not data (episodes) need be exchanged.

The RPR-MTL has been applied to both grid-world problems as well as real multiple-aspect classification problems, to demonstrate the improvements due to episodes transfer and experience sharing.

7.1.3 Application of partially observable Markov decision processes

The last part of this thesis is an application of POMDPS (Chapter 6). In this part, we have considered a classification problem wherein the class features are not given a priori. The classifier is responsible for selecting the features, to minimize the cost of observing features while also maximizing the classification performance. We have proposed a reward-directed Bayesian classifier (RDBC) to solve this problem. The RDBC features an internal state structure for preserving the feature dependence, and is formulated as a partially observable Markov decision process (POMDP). The results on a diabetes dataset have shown that the RDBC with a moderate number of states significantly improves over the naive Bayes classifier, both in prediction accuracy and observation parsimony. We have also demonstrated that the RDBC performs better by using more states to increase its memory.
7.2 Directions for Future Work

Based on the findings in the thesis, there can be several directions that one can take to pursue further research.

7.2.1 New basis functions for the ILSPI

The ILSPI is not restricted to the specific form of basis functions and is applicable to any types of basis. The right choice of basis functions is important and leads to a more accurate approximation of the value function. We have discussed in Section 3.3.5 the relevance of radial basis functions (RBF) and second-order polynomials in value function approximation.

However, RBF and polynomials are not the only choice. An alternative choice of basis functions is diffusion wavelets (Coifman et al. [2005a,b]). Diffusion wavelets yields a multi-resolution representation on discrete graphs and subsets of a Euclidean space, which may be relevant to value function approximation. Diffusion wavelets have recently been used to approximate value functions in MDPs (Mahadevan and Maggioni [2005]; Maggioni and Mahadevan [2006]). But for POMDPs, it is still under investigation. How to apply diffusion wavelets in the ILSPI is an interesting and promising research topic that deserves careful studies. The basic idea could be to use diffusion wavelets to represent the value function on belief samples, where the diffusion wavelets are constructed on a graph induced by a random walk over the belief samples, given a policy.

7.2.2 Multi-agent learning with RPRs

The RPR has been proposed in the context of a single agent. Though we have considered multiple agents in multiple environments in Chapter 5, we have assumed there that each agent is only concerned with its own environment.
In a broader context, consider the individual local environments connected to form a big global environment. Then the global environment becomes complicated and huge, and it is very difficult for one agent to implement a task in a short period. Employing multiple agents to achieve a common goal becomes a natural way to deal with complicated situations. Since the agents act in the same (global) environment, more coordinations are required to them to obtain the joint reward. This means the agents could be related in a more complicated way, other than sharing experience.

For example, consider two agents finding multiple goals in a grid-world environment. The optimal action of each agent not only depends on its own history, but also depends on the history of the other agent. In this case, each agent might still keep a separate RPR, but the two RPRs are now learned in a more coordinated way, involving more complicated information exchange.
Bibliography


172


Biography

Hui Li was born on Nov. 23, 1974, in Jiangxi, China. She received the B.S. degree in electrical engineering from Hunan University, Changsha, China in 1996 and M.S. degree in electrical engineering from Xi’dian University, Xi’an, China in 2001. She also received the M.S. degree and Ph.D degree in the Department of Electrical and Computer engineering from Duke University in 2003 and 2006 respectively.

Publications


Technical Reports


Professional Activities

Professional Membership: IEEE;