Journal of Machine Learning Research 10 (2009) 1955-1988 Submitted 9/08; Revised 3/09; Published 8/09

Provably Efficient Learning with Typed Parametric Models

Emma Brunskill EMMA@CSAIL.MIT.EDU
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02143, USA

Bethany R. Leffler BLEFFLER@CS.RUTGERS.EDU
Lihong Li LIHONG@CS.RUTGERS.EDU
Michael L. Littman MLITTMAN@CS.RUTGERS.EDU
Department of Computer Science
Rutgers University
Piscataway, NJ 08854, USA

Nicholas Roy NICKROY@MIT.EDU
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02143, USA

Editor: Sham Kakade

Abstract

To quickly achieve good performance, reinforcement-learning algorithms for acting in large continuous-valued domains must use a representation that is both sufficiently powerful to capture important domain characteristics, and yet simultaneously allows generalization, or sharing, among experiences. Our algorithm balances this tradeoff by using a stochastic, switching, parametric dynamics representation. We argue that this model characterizes a number of significant, real-world domains, such as robot navigation across varying terrain. We prove that this representational assumption allows our algorithm to be probably approximately correct with a sample complexity that scales polynomially with all problem-specific quantities including the state-space dimension. We also explicitly incorporate the error introduced by approximate planning in our sample complexity bounds, in contrast to prior Probably Approximately Correct (PAC) Markov Decision Processes (MDP) approaches, which typically assume the estimated MDP can be solved exactly. Our experimental results on constructing plans for driving to work using real car trajectory data, as well as a small robot experiment on navigating varying terrain, demonstrate that our dynamics representation enables us to capture real-world dynamics in a sufficient manner to produce good performance.

Keywords: reinforcement learning, provably efficient learning

1. Introduction

Reinforcement learning (RL) (Sutton and Barto, 1998) has had some impressive real-world successes, including model helicopter flying (Ng et al., 2004) and expert software backgammon players (Tesauro, 1994). Two of the key challenges in reinforcement learning are scaling up to larger, richer domains, and developing principled approaches for quickly learning to perform well. Our interest lies in developing algorithms for large continuous-valued environments, including problems such as

learning the best route to drive to work, or how a remote robotic rover can learn to traverse different types of terrain. To perform learning efficiently in such environments, we will assume that the world dynamics can be compactly described by a small set of simple parametric models, such as one for driving on highways and another for driving on small roads. We will prove that this assumption allows our algorithm to require an amount of experience that only scales polynomially with the state space dimension. We will also empirically demonstrate that these assumptions are realistic for several real world data sets, indicating that our Continuous-Offset Reinforcement Learning (CORL) algorithm may be well suited for large, high-dimensional domains.

A critical choice in the construction of a reinforcement-learning algorithm is how to balance between actions that gather information about the world environment (exploration) versus actions that are expected to yield high reward given the agent's current estimates of the world environment (exploitation). In early work, algorithms such as Q-learning were shown to perform optimally in the limit of infinite data (Watkins, 1989), but no finite-sample guarantees were known. More recently there have been three main branches of model-based reinforcement learning research concerned with the exploration problem. The first consists of heuristic approaches, some of which perform very well in practice, but lack performance guarantees (for example Jong and Stone 2007). The second branch strives to perform the action that optimally balances exploration and exploitation at each step. Such Bayesian approaches include the model parameters inside the state space of the problem. Poupart et al. (2006) assumed a fully observed discrete state space and modeled the underlying model parameters as hidden states, effectively turning the problem into a continuous-state partially observable Markov decision process (POMDP). Castro and Precup (2007) also assumed a fully observed discrete state space but represented the model parameters as counts over the different transitions and reward received, thereby keeping the problem fully observable. Doshi et al. (2008) considered a Bayesian approach for learning when the discrete state space is only partially observable, and Ross et al. (2008) considered learning in a partially-observed continuous-valued robot navigation problem. Approaches in the Bayesian RL framework run into inherent complexity problems and typically produce algorithms that only approximately solve their target optimality criteria.

In our work we will focus on achieving near optimality, making precise guarantees on when, and with what probability, it will be achieved. This type of approach to reinforcement learning was commenced by Kearns and Singh (2002) and Brafman and Tennenholtz (2002) who created algorithms that were guaranteed to achieve near optimal performance on all but a small number of samples, with high probability. We will refer to work in this line of research as "probably approximately correct" (PAC-MDP), as introduced by Strehl et al. (2006), and will discuss it further in the sections that follow. One of the appealing aspects of this area over a Bayesian RL approach is that it allows one to make precise statements about the efficiency and performance of algorithms: if the MDP or POMDP used in the Bayesian RL approach could be solved exactly with an informative prior, then this approach would likely outperform PAC-MDP approaches. However, when a Bayesian RL problem is only approximately solved or when the prior information is incorrect, it is unknown how far the resulting solution is from the optimal behavior. Our work lies within this third PAC-MDP approach, and draws upon the past advances made in this subfield, including our own initial work in this area (Brunskill et al., 2008). The current work makes a significant theoretical generalization of our initial results which requires different proof techniques, and presents a number of new experiments and discussions.

Within the PAC-MDP line of research, there has been little work on directly considering continuous-valued states. One exception is the work of Strehl and Littman (2008), who considered learning in continuous-valued state-action spaces. Their work assumed that a single dynamics representation was shared among all states, and that the noise parameter of the dynamics representation was known. The focus of their paper was slightly different than the current work, in that the authors presented a new online regression algorithm for determining when enough information was known to make accurate predictions.

An alternate approach to handling continuous state spaces is to discretize the space into a grid. This step enables prior PAC-MDP algorithms such as R-max (Brafman and Tennenholtz, 2002) to be applied directly to the discretized space. However, their representation of the world may not fully exploit existing structure. In particular, such a representation requires that the dynamics model for each state-action tuple is learned independently. Since each state-action can have entirely different dynamics, this approach has a great deal of representational power. However, as there is no sharing of dynamics information among states, it has a very low level of generalization. In contrast, the work of Strehl and Littman (2008) and the classic linear quadratic Gaussian regulator model (Burl, 1998) assume that the dynamics model is the same for all states, greatly restricting the representational power of these models in return for higher generalization and fast learning.

Recently, there have been several approaches that explore the middle ground of representational power and generalization ability. Jong and Stone (2007) assumed that the dynamics model between nearby states was likely to be similar, and used an instance-based approach to solve a continuous-state RL problem. Their experimental results were encouraging but no theoretical guarantees were provided, and the amount of data needed would typically scale exponentially with the state-space dimension. A stronger structural assumption is made in the work of Leffler et al. (2007), which focused on domains in which the discrete state space is divided into a set of types. States within the same type were assumed to have the same dynamics. The authors proved that a typed representation can require significantly less experience to achieve good performance compared to a standard R-max algorithm that learns each state-action dynamics model separately.

Our work draws on the recent progress and focuses on continuous-state, discrete-action, typed problems. By using a parametric model to represent the dynamics of each of a discrete set of types, we sacrifice some of the representational power of prior approaches (Leffler et al., 2007; Brafman and Tennenholtz, 2002) in return for improved generalization, but still retain a much more flexible representation than approaches that assume a single dynamics model that is shared across all states. In particular, we prove that restricting our representational power enables our algorithm to have a sample complexity that scales *polynomially* with the state-space dimension. An alternate approach is to place a uniformly spaced grid over the state space and solve the problem using the existing algorithms from Leffler et al. (2007) or Brafman and Tennenholtz (2002). However, this strategy results in an algorithm whose computational complexity scales exponentially with the state-space dimension.

Our algorithm involves a subroutine for solving a continuous-state MDP using the current model estimates. Outside of special cases like the linear Gaussian quadratic regulator problem (Burl, 1998), planning cannot be performed exactly for generic continuous-state MDPs. Therefore we explicitly incorporate the error introduced by approximate planning in our sample complexity bounds. This is in contrast to prior PAC-MDP approaches, which typically assume the estimated MDP can be solved exactly.

In particular, our dynamics representation is a simple noisy offset model, where the next state is presumed to be a function of the prior state, plus an offset and some Gaussian distributed noise. The offset and Gaussian parameters are assumed to be specified by the type t of the state and action a, thereby allowing all states of the same type to share dynamics parameters. More formally,

$$s' = s + \beta_{at} + \varepsilon_{at}, \tag{1}$$

where s is the current state, s' is the next state, $\varepsilon_{at} \sim \mathcal{N}(0, \Sigma_{at})$ is drawn from a zero-mean Gaussian with covariance Σ_{at}, and β_{at} is the offset.

In our experimental section we first demonstrate our algorithm on the standard RL PuddleWorld problem of Boyan and Moore (1995). We next illustrate the importance of learning the variance of different types by an example of an agent with a hard time deadline. The third example is a simulated decision problem in which an agent is trying to learn the best route for driving to work. The simulator uses real car-trajectory data to generate its trajectories. In the final experiment, a real robot car learns to navigate varying terrain. These experiments demonstrate that the noisy offset dynamics model, while simple, is able to capture real world dynamics for two different domains sufficiently adequately to allow the agent to quickly learn a good strategy.

At a high level, our work falls into the category of model-based reinforcement-learning algorithms in which the MDP model (Equation 1) can be *KWIK-learned* (Li et al., 2008; Li, 2009), and thus it is efficient in exploring the world. The Knows Whats It Knows (KWIK) framework is an alternate learning framework which incorporates characteristics of the Probably Approximately Correct (PAC) learning framework, which will be discussed further below, and the mistake bound framework. Though our theoretical development will follow a PAC-style approach, the KWIK framework provides another justification of the soundness and effectiveness of our algorithm.

The focus of this paper is on the sample complexity of the CORL algorithm. CORL assumes an approximate MDP planner to solve the current estimated MDP, and several such approximate planners with guarantees on the resulting solution involve a discretizaton that results in an exponential tiling of the state space. In such cases the computational complexity of CORL will scale exponentially with the number of dimensions. However, the experimental results demonstrate that CORL exhibits computational performance competitive with or better than existing approaches.

The rest of the paper proceeds as follows. In Section 2, we will briefly discuss the background to our work and then present the CORL algorithm. Section 3 presents our theoretical analysis of our algorithm. In Section 4 we present experimental results, and in Section 5 we conclude and discuss future work.

2. A Continuous-state Offset-dynamics Reinforcement Learner

This section introduces terminology and then presents our algorithm, CORL.

2.1 Background

The world is characterized by a continuous-state discounted MDP $M = \langle S, A, p(s'|s,a), R, \gamma \rangle$ where $S \subseteq \mathbb{R}^N$ is the N-dimensional state space, A is a set of discrete actions, $p(s'|s,a)$ is the transition dynamics, $\gamma \in [0,1)$ is the discount factor and $R : S \times A \to [0,1]$ is the reward function. In addition to the standard MDP formulation, each state s is associated with a single observable type $t \in T$. The total number of types is N_T and the mapping from states to types $S \to T$ is assumed to be known.

Algorithm 1 CORL

Input: N (dimension of the state space),$|A|$ (number of actions), N_T (number of types), R (reward model), γ (discount factor), N_{at} (minimum number of samples per state-action pair)
Set all type-action tuples $\langle t,a\rangle$ to be unknown and initialize the dynamics models (see text) to create an empirical known-type MDP model $\hat{M}_K$.
Start in a state s_0.
loop
    Solve MDP $\hat{M}_K$ using approximate solver and denote its optimal value function by Q_t.
    Select action $a = \text{argmax}_a Q_t(s,a)$.
    Increment the appropriate n_{at} count (where t is the type of state s).
    Observe transition to the next state s'.
    If n_{at} exceeds N_{at} then mark $\langle a,t\rangle$ as "known" and estimate the dynamics model parameters for this tuple.
end loop

The dynamics of the environment are determined by the current state type t and action a taken:

$$p(s'|s,a) = \mathcal{N}(s';s+\beta_{at},\Sigma_{at}).$$

Therefore, types partition the state space into regions, and each region is associated with a particular pair of dynamics parameters.

In this work, we focus on when the reward model is provided[1] and the dynamics model parameters are hidden. The parameters of the dynamics model, β_{at} and Σ_{at}, are assumed to be unknown for all types t and actions a at the start of learning. This model is a departure from prior related work (Abbeel and Ng, 2005; Strehl and Littman, 2008), which focuses on a more general linear dynamics model but assumes a single type and that the variance of the noise Σ_{at} is known. We argue that in many interesting problems, the variance of the noise is unknown and estimating this noise may provide the key distinction between the dynamics models of different types.

In reinforcement learning, the agent must learn to select an action a given its current state s. At each time step, it receives an immediate reward r based on its current state.[2] The agent then moves to a next state s' according to the dynamics model. The goal is to learn a policy $\pi : S \to A$ that allows the agent to choose actions to maximize the expected total reward it will receive. The value of a particular policy π is the expected discounted sum of future rewards that will be received from following this policy, and is denoted $V^\pi(s) = E_\pi[\sum_{j=0}^{\infty}\gamma^j r_j|s_0 = s]$, where r_j is the reward received on the j-th time step and s_0 is the initial state of the agent. Let π^* be the optimal policy, and its associated value function be $V^*(s)$.

2.2 Algorithm

Our algorithm (*c.f.*, Algorithm 1) is derived from the R-max algorithm of Brafman and Tennenholtz (2002). We first form a set of $\langle t,a\rangle$ tuples, one for each type-action pair. Note that each tuple

1. As long as the reward can be KWIK-learned (Li et al., 2008) then the results are easily extended to when the reward is unknown. KWIK-learnable reward functions include, for instance, Gaussian, linear and tabular rewards.
2. For simplicity, the reward is assumed to be only a function of state in this paper. It is straightforward to extend our results to the case when the reward function also depends on the action taken.

corresponds to a particular pair of dynamics model parameters, $\langle \beta_{at}, \Sigma_{at} \rangle$. A tuple is considered to be "known" if the agent has been in type t and taken action a a number N_{at} times. At each time step, we construct a new MDP $\hat{M}_K$ as follows, using the same state space, action space, and discount factor as the original MDP. If the number of times a tuple has been experienced, n_{at}, is greater than or equal to N_{at}, then we estimate the parameters for this dynamics model using maximum-likelihood estimation:

$$\tilde{\beta}_{at} = \frac{\sum_{i=1}^{n_{at}} (s'_i - s_i)}{n_{at}}, \tag{2}$$

$$\tilde{\Sigma}_{at} = \frac{\sum_{i=1}^{n_{at}} (s'_i - s_i - \tilde{\beta}_{at})(s'_i - s_i - \tilde{\beta}_{at})^T}{n_{at}} \tag{3}$$

where the sum ranges over all state-action pairs experienced for which the type of s_i was t, the action taken was a, and s'_i was the successor state. Note that while Equation 3 is a biased estimator, it is also popular and consistent, and becomes extremely close to the unbiased estimate when the number of samples n_{at} is large. We choose it because it makes our later analysis simpler.

Otherwise, we set the dynamics model for all states and the action associated with this type-action tuple to be a transition with probability 1 back to the same state. We also modify the reward function for all states associated with an unknown type-action tuple $\langle t_u, a_u \rangle$ so that all state-action values $Q(s_{t_u}, a_u)$ have a value of $V_{\max}$ (the maximum value possible, $1/(1-\gamma)$). We then seek to solve $\hat{M}_K$. This MDP includes switching dynamics with continuous states, and we are aware of no planners guaranteed to return the optimal policy for such MDPs in general. CORL assumes the use of an approximate solver to provide a solution for a MDP. There are a variety of existing MDP planners, such as discretizing or using a linear function approximation, and we will consider particular planner choices in the following sections. At each time step, the agent chooses the action that maximizes the estimate of its current approximate value according to Q_t: $a = \text{argmax}_a Q_t(s,a)$. The complete algorithm is shown in Algorithm 1.

3. Learning Complexity

In this section we will first introduce relevant background and then provide a formal analysis of the CORL algorithm.

3.1 Preliminaries and Framework

When analyzing the performance of an RL algorithm $\mathcal{A}$, there are many potential criteria to use. In our work, we will focus predominantly on sample complexity with a brief mention of computational complexity. Computational complexity refers to the number of operations executed by the algorithm for each step taken by the agent in the environment. We will follow Kakade (2003) and use *sample complexity* as shorthand for the *sample complexity of exploration*. It is the number of time steps at which the algorithm, when viewed as a non-stationary policy π, is not ε-optimal at the current state; that is, $Q^*(s,a) - Q^\pi(s,a) > \varepsilon$ where Q^* is the optimal state-action value function and Q^π is the state-action value function of the non-stationary policy π. Following Strehl et al. (2006), we are interested in showing, for a given ε and δ, that with probability at least $1-\delta$ the sample complexity of the algorithm is less than or equal to a polynomial function of MDP parameters. Note that we only consider the number of samples to ensure the algorithm will learn and execute a near-optimal

policy with high probability. As the agent acts in the world, it may be unlucky and experience a series of state transitions that poorly reflect the true dynamics due to noise.

Prior work by Strehl et al. (2006) provided a framework for analyzing the sample complexity of R-max-style RL algorithms. This framework has since been used in several other papers (Leffler et al., 2007; Strehl and Littman, 2008) and we will also adopt the same approach. We first briefly discuss the structure of this framework.

Strehl et al. (2006) defined an RL algorithm to be greedy if it chooses its action to be the one that maximizes the value of the current state s ($a = \text{argmax}_{a \in A} Q(s,a)$). Their main result goes as follows: let $\mathcal{A}(\varepsilon, \delta)$ denote a greedy learning algorithm. Maintain a list K of "known" state-action pairs. At each new time step, this list stays the same unless during that time step a new state-action pair becomes known. MDP $\hat{M}_K$ is the agent's current estimated MDP, consisting of the agent's estimated models for the known state-action pairs, and self loops and optimistic rewards (as in our construction described in the prior section) for unknown state-action pairs. MDP M_K is an MDP which consists of the true (underlying) reward and dynamics models for the known state-action pairs, and again self loops and optimistic rewards for the unknown state-action pairs. To be clear, the only difference between MDP $\hat{M}_K$ and MDP M_K is that the first uses the agent's experience to generate estimated models for the known state-action pairs, and the second uses the true model parameters. π is the greedy policy with respect to the current state-action values $Q_{\hat{M}_K}$ obtained by solving MDP $\hat{M}_K$: $V^{\pi}_{\hat{M}_K}$ is the associated value function for $Q_{\hat{M}_K}$ and may equivalently be viewed as the value of policy π computed using the estimated model parameters. $V^{\pi}_{M_K}$ is the value of policy π computed using the true model parameters. Assume that ε and δ are given and the following three conditions hold for all states, actions and time steps:

1. $Q^*(s,a) - Q_{\hat{M}_K}(s,a) \leq \varepsilon$.
2. $V^{\pi}_{\hat{M}_K}(s) - V^{\pi}_{M_K}(s) \leq \varepsilon$.
3. The total number of times the agent visits a state-action tuple that is not in K is bounded by $\zeta(\varepsilon, \delta)$ (the *learning complexity*).

Then, Strehl et al. (2006) show for any MDP M, $\mathcal{A}(\varepsilon, \delta)$ will follow a 4ε-optimal policy from its initial state on all but N_{total} time steps with probability at least $1 - 2\delta$, where N_{total} is polynomial in the problem's parameters $(\zeta(\varepsilon, \delta), \frac{1}{\varepsilon}, \frac{1}{\delta}, \frac{1}{1-\gamma})$.

The majority of our analysis will focus on showing that our algorithm fulfills these three criteria. In our approach, we will define the known state-action pairs to be all those state-actions for which the type-action pair $\langle t(s), a \rangle$ is known. We will assume that the absolute values of the components in Σ_{at} are upper bounded by a known constant B_σ which is, without loss of generality, assumed to be greater than or equal to 1. This assumption is often true in practice. We denote the determinant of matrix D by $\det D$, the trace of a matrix D by $\text{tr}(D)$, the absolute value of a scalar d by $|d|$ and the p-norm of a vector v by $\|v\|_p$. Full proofs, when omitted, can be found in the Appendix.

3.2 Analysis

Our analysis will serve to prove the main result:

Theorem 1 *For any given δ and ε in a continuous-state noisy offset dynamics MDP with N_T types where the covariance along each dimension of all the dynamics models is bounded by $[-B_\sigma, B_\sigma]$,*

on all but N_{total} time steps, our algorithm will follow a 4ε-optimal policy from its current state with probability at least $1-2\delta$, where N_{total} is polynomial in the problem parameters $(N, |A|, N_T, \frac{1}{\varepsilon}, \frac{1}{\delta}, \frac{1}{1-\gamma}, \frac{1}{\lambda_N}, B_\sigma)$ where λ_N is the smallest eigenvalue of the dynamics covariance matrices.

Proof To prove this, we need to demonstrate that the three criteria of Strehl et al. (2006) hold. The majority of our effort will focus on the second criterion. This criterion states that the value of states under the estimated known-state MDP $\hat{M}_K$ must be very close to the value of states under the known-state MDP M_K that uses the true model parameters for all known type-action pairs. To prove this we must bound how far away the model parameters estimated from the agent's experience can be from the true underlying parameters, and how this relates to error in the resulting value function. We must also consider the error induced by approximately solving the estimated MDP $\hat{M}_K$. Achieving a given accuracy level in the final value function creates constraints on how close the estimated model parameters must be to the true model parameters. We will illustrate how these constraints relate to the amount of experience required to achieve these constraints. This in turn will give us an expression for the number of samples required for a type-action pair to be known, or the learning complexity for our algorithm. Once we have proved the second criterion we will discuss how the other two conditions are also met.

Therefore we commence by formally relating how the amount of experience (number of transitions) of the agent corresponds to the accuracy in the estimated dynamics model parameters.

Lemma 2 *Given any $\varepsilon, \delta > 0$, then after $T = \frac{12N^2B_\sigma^2}{\varepsilon^2\delta}$ transition samples (s,a,s') with probability at least $1-\frac{2\delta}{3}$, the estimated offset parameter $\tilde{\beta}$, computed by Equation 2, and estimated covariance parameters $\tilde{\sigma}_{ij}$, computed by Equation 3, will deviate from the true parameters β and σ_{ij} by at most ε: $\Pr(\|\tilde{\beta}-\beta\|_2 \leq \varepsilon) \geq 1-\frac{\delta}{3}$ and $\Pr(\max_i |\tilde{\sigma}_{ij}-\sigma_{ij}| \leq \varepsilon) \geq 1-\frac{\delta}{3}$.*

Proof T will be the maximum of the number of samples to guarantee the above bounds for the offset parameter β and the number of samples needed for a good estimate of the variance parameter. We first examine the offset parameter:

Lemma 3 *Given any $\varepsilon, \delta > 0$, define $T_\beta = \frac{3N^2B_\sigma}{\varepsilon^2\delta}$. If there are T_β transition samples (s,a,s'), then with probability at least $1-\frac{\delta}{3}$, the estimated offset parameter $\tilde{\beta}$, computed by Equation 2, will deviate from the true offset parameter β by no more than ε along any dimension d; formally, $\Pr(\max_d \|\tilde{\beta}_d - \beta_d\|_2 \geq \frac{\varepsilon}{\sqrt{N}}) \leq \frac{\delta}{3N}$.*

Proof From Chebyshev's inequality, we know

$$P(|(s'_{id} - s_{id}) - \beta_d| \geq \frac{\varepsilon}{\sqrt{N}}) \leq \frac{\sigma_d^2 N}{\varepsilon^2},$$

where s_{id} and σ_d^2 are the value of the i-th state and variance of the offset along dimension d, respectively. Using the fact that the variance of a sum of T_β i.i.d. variables is just T_β multiplied by the variance of a single variable, we obtain

$$\begin{aligned} \Pr(|\sum_{i=1}^{T_\beta}(s'_{id} - s_{id}) - T_\beta\beta_d| \geq T_\beta\frac{\varepsilon}{\sqrt{N}}) &\leq \frac{T_\beta\sigma_d^2 N}{T_\beta^2\varepsilon^2} \\ \Pr(|\tilde{\beta}_d - \beta_d| \geq \frac{\varepsilon}{\sqrt{N}}) &\leq \frac{\sigma_d^2 N}{T_\beta\varepsilon^2}. \end{aligned}$$

We require the right-hand side above be at most $\frac{\delta}{3N}$ and solve for T_β:

$$T_\beta = \frac{3\sigma_d^2 N^2}{\delta\varepsilon^2}.$$

We know that the variance along any dimension is bounded above by B_σ so we can substitute this in the above expression to derive a bound on the number of samples required:

$$T_\beta \geq \frac{3B_\sigma N^2}{\delta\varepsilon^2}.$$

■

Lemma 3 immediately implies a bound on the L_2 norm between the estimated offset parameter vector and the true offset parameter vector, as follows:

Lemma 4 *Given any* $\varepsilon, \delta > 0$, *if* $\Pr(\max_d |\tilde{\beta}_d - \beta_d| \geq \frac{\varepsilon}{\sqrt{N}}) \leq \frac{\delta}{3N}$, *then* $\Pr(\|\tilde{\beta} - \beta\|_2 \geq \varepsilon) \leq \frac{\delta}{3}$.

Proof By a union bound, the probability that any of the dimensions exceeds an estimation error of at most $\frac{\varepsilon}{\sqrt{N}}$ is at most $\frac{\delta}{3}$. Given this, with probability at least $1 - \frac{\delta}{3}$ all dimensions will simultaneously have an estimation error of less than $\frac{\varepsilon}{\sqrt{N}}$ and from the definition of the L2 norm this immediately implies that $\|\tilde{\beta} - \beta\|_2 \leq \varepsilon$. ■

We next analyze the number of samples needed to estimate the covariance accurately.

Lemma 5 *Assume* $\max_d |\tilde{\beta}_d - \beta_d| \leq \varepsilon$ *for* $\varepsilon < 1/4$. *Given any* $\delta > 0$, *define* $T_\sigma = \frac{12N^2 B_\sigma^2}{\delta\varepsilon^2}$. *If there are* T_σ *transition samples* (s, a, s'), *then with probability at most* $\frac{\delta}{3}$, *the estimated covariance parameter* $\tilde{\sigma}_{ij}$, *computed by Equation 3, deviates from the true covariance parameter* σ_{ij} *by more than* ε *over all entries* ij; *formally,* $\Pr(\max_{i,j} |\tilde{\sigma}_{ij} - \sigma_{ij}| \geq \varepsilon) \leq \frac{\delta}{3}$.

We provide the proof of Lemma 5 in the appendix: briefly, we again use Chebyshev's inequality which requires us to bound the variance of the sample covariance.

Combining Lemmas 4 and 5 gives a condition on the minimum number of samples necessary to ensure, with high probability, that the estimated parameters of a particular type-action dynamics model are close to the true parameters. Without loss of generality, assume $B_\sigma \geq 1$, then

$$T = \max\{T_\beta, T_\sigma\} = \max\left\{\frac{3N^2 B_\sigma}{\varepsilon^2\delta}, \frac{12N^2 B_\sigma^2}{\varepsilon^2\delta}\right\} = \frac{12N^2 B_\sigma^2}{\varepsilon^2\delta}.$$

■

From Lemma 2 we now have an expression that relates how much experience the agent needs in order to have precise estimates of each model parameter. We next need to establish the distance between two dynamics models which have different offset and covariance parameters. This distance will later be important for bounding the value function difference between the estimated model MDP $\hat{M}_K$ and the true model MDP M_K.

Following Abbeel and Ng (2005), we choose to use the variational distance between two dynamics models P and Q:

$$d_{var}(P(x),Q(x)) = \frac{1}{2}\int_{\mathcal{X}} |P(x)-Q(x)|dx.$$

In our algorithm, β_1 and Σ_1 are the true offset parameter and covariance matrix of the Gaussian distribution, and β_2 and Σ_2 are the offset parameter and covariance matrix estimated from data. Since we can guarantee that they can be made arbitrarily close (element-wise), we will be able to bound the variational distance between two Gaussians, one defined with the true parameters and the other with the estimated parameters. The real-valued, positive eigenvalues of Σ_1 and Σ_2 are denoted by $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N > 0$ and $\lambda'_1 \geq \lambda'_2 \geq \cdots \geq \lambda'_N > 0$, respectively. Because of symmetry and positive definiteness of Σ_1 and Σ_2, λ_i and λ'_i must be real as well as positive. Since all eigenvalues are positive, they are also the singular values of their respective matrices.

Lemma 6 *Assume* $\max_{i,j} |\Sigma_1(i,j) - \Sigma_2(i,j)| \leq \varepsilon$, *and* $N\varepsilon \left\|\Sigma_1^{-1}\right\|_\infty < 1$, *then,*

$$d_{var}(\mathcal{N}(s'-s|\beta_1,\Sigma_1),\mathcal{N}(s'-s|\beta_2,\Sigma_2)) \leq \frac{||\beta_1-\beta_2||_2}{\sqrt{\lambda_N}} + \sqrt{\frac{N^2\varepsilon}{\lambda_N} + \frac{2N^3 B_\sigma \varepsilon}{\lambda_N^2 - N^{1.5}\varepsilon\lambda_N}}.$$

Proof We will use $\mathcal{N}(\beta,\Sigma)$ as an abbreviation for $\mathcal{N}(s'-s|\beta,\Sigma)$. Then

$$\begin{aligned} d_{var}(\mathcal{N}(\beta_1,\Sigma_1),\mathcal{N}(\beta_2,\Sigma_2)) &\leq d_{var}(\mathcal{N}(\beta_1,\Sigma_1),\mathcal{N}(\beta_2,\Sigma_1)) + d_{var}(\mathcal{N}(\beta_2,\Sigma_1),\mathcal{N}(\beta_2,\Sigma_2)) \\ &= \frac{||(\mathcal{N}(\beta_1,\Sigma_1),\mathcal{N}(\beta_2,\Sigma_1))||_1}{2} + \frac{||(\mathcal{N}(\beta_2,\Sigma_1),\mathcal{N}(\beta_2,\Sigma_2))||_1}{2} \\ &\leq \sqrt{2\mathrm{d_{KL}}(\mathcal{N}(\beta_1,\Sigma_1)\,\|\,\mathcal{N}(\beta_2,\Sigma_1))} + \sqrt{2\mathrm{d_{KL}}(\mathcal{N}(\beta_2,\Sigma_1)\,\|\,\mathcal{N}(\beta_2,\Sigma_2))} \end{aligned}$$

where $\mathrm{d_{KL}}(\|)$ is the Kullback-Leibler divergence. The first step follows from the triangle inequality and the last step follows from Kullback (1967) (included for completeness in Lemma 14 in the appendix).

The KL divergence between two N-variate Gaussians has the closed form expression

$$\mathrm{d_{KL}}(\mathcal{N}(\beta_1,\Sigma_1)\,\|\,\mathcal{N}(\beta_2,\Sigma_2)) = \frac{1}{2}\left((\beta_1-\beta_2)^T\Sigma_1^{-1}(\beta_1-\beta_2) + \ln\frac{\det\Sigma_2}{\det\Sigma_1} + \mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) - N\right).$$

Substituting this expression into the above bound on d_{var} we get

$$d_{var}(\mathcal{N}(\beta_1,\Sigma_1),\mathcal{N}(\beta_2,\Sigma_2)) \leq \sqrt{(\beta_1-\beta_2)^T\Sigma_1^{-1}(\beta_1-\beta_2)} + \sqrt{\ln\left(\frac{\det\Sigma_2}{\det\Sigma_1}\right) + \mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) - N}. \quad (4)$$

Our proof relies on bounding both terms of Equation 4. Note that this expression reduces (up to a constant) to the bound proved by Abbeel and Ng (2005) when the variance is known.

We now start with the first term of Equation 4:

Lemma 7

$$(\beta_1-\beta_2)^T\Sigma_1^{-1}(\beta_1-\beta_2) \leq \frac{1}{\lambda_N}||\beta_1-\beta_2||_2^2.$$

Proof First note that since Σ_1^{-1} is a Hermitian matrix,

$$\frac{(\beta_1-\beta_2)^T\Sigma_1^{-1}(\beta_1-\beta_2)}{||\beta_1-\beta_2)||_2^2}$$

is a Rayleigh quotient which is bounded by the maximum eigenvalue of Σ_1^{-1}. The eigenvalues of Σ_1^{-1} are precisely the reciprocals of the eigenvalues of Σ_1. Therefore, the Rayleigh quotient above is at most $\frac{1}{\lambda_N}$:

$$(\beta_1-\beta_2)^T\Sigma_1^{-1}(\beta_1-\beta_2) \leq \frac{||\beta_1-\beta_2||_2^2}{\lambda_N}.$$

■

We now provide lemmas that bound the components of the second term of Equation 4: proofs are provided in the appendix.

Lemma 8 *If* $\max_{i,j}|\Sigma_1(i,j)-\Sigma_2(i,j)| \leq \varepsilon$ *for any* $1 \leq i,j \leq N$, *then*

$$\left|\ln\frac{\det\Sigma_2}{\det\Sigma_1}\right| \leq N\varepsilon\left(\frac{1}{\lambda_1}+\frac{1}{\lambda_2}+\cdots+\frac{1}{\lambda_N}\right) \leq \frac{N^2\varepsilon}{\lambda_N}.$$

Lemma 9 *If* $\max_{i,j}|\Sigma_1(i,j)-\Sigma_2(i,j)| \leq \varepsilon$ *and* $N\varepsilon\left\|\Sigma_1^{-1}\right\|_1 < 1$, *then*

$$\mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right)-N \leq \frac{2N^3\varepsilon B_\sigma}{\lambda_N^2-(N)^{1.5}\lambda_N\varepsilon}.$$

Combining the results of Lemmas 7, 8, and 9 completes the proof of Lemma 6. ■

Note this bound is tight when the means and the variances are the same.

At this point we can relate the number of experiences (samples) of the agent to a distance measure between the estimated dynamics model (for a particular type-action) and the true dynamics model.

We now bound the error between the state-action values of the true MDP model M_K solved exactly and the approximate state-action values of our estimated model MDP $\hat{M}_K$ obtained using an approximate planner, as a function of the error in the dynamics model estimates. This is a departure from most related PAC-MDP work which typically assumes the existence of a planning oracle for choosing actions given the estimated model.

Lemma 10 ***(Simulation Lemma)*** *Let* $M_1 = \langle S,A,p_1(\cdot|\cdot,\cdot),R,\gamma\rangle$ *and* $M_2 = \langle S,A,p_2(\cdot|\cdot,\cdot),R,\gamma\rangle$ *be two MDPs*[3] *with dynamics as characterized in Equation 1 and non-negative rewards bounded above by 1. Given an* ε *(where* $0 < \varepsilon \leq V_{\max}$*), assume that for all state-action tuples* (s,a), $d_{var}(p_1(\cdot|s,a),p_2(\cdot|s,a)) \leq (1-\gamma)^2\varepsilon/(2\gamma)$ *and the error incurred by approximately solving a MDP, defined as* ε_{plan} *is also at most* $(1-\gamma)^2\varepsilon/(2\gamma)$ *(to be precise,* $\varepsilon_{plan} = ||V^*-\tilde{V}^*||_\infty \leq (1-\gamma)^2\varepsilon/(2\gamma)$ *where* $\tilde{V}^*$ *is the value computed by the approximate solver). Let* π *be a policy that can be applied to both* M_1 *and* M_2. *Then, for any stationary policy* π, *for all states s and actions a,* $|Q_1^\pi(s,a) - \tilde{Q}_2^\pi(s,a)| \leq \varepsilon$, *where* $\tilde{Q}_2^\pi$ *denotes the state-action value obtained by using an approximate MDP solver on MDP* M_2 *and* Q_1^π *denotes the true state-action value for MDP* M_1 *for policy* π.

3. For simplicity we present the results here without reference to types. In practice, each dynamics parameter would be subscripted by its associated MDP, type, and action.

Proof Let $\Delta_Q = \max_{s,a} |Q_1^\pi(s,a) - \tilde{Q}_2^\pi(s,a)|$ and define $\tilde{V}_2^\pi$ to be the approximate value of policy π computed using an approximate MDP solver on MDP M_2, and V_1^π be the exact value of policy π on MDP M_1. Note that since we are taking the max over all actions, Δ_Q is also equal to or greater than $\max_s |V_1^\pi(s) - \tilde{V}_2^\pi(s)|$. Let $Lp_2(s'|s,a)$ denote an approximate backup for MDP M_2.

Since these value functions are the fixed-point solutions to their respective Bellman operators, we have for every (s,a) that

$$
\begin{aligned}
&|Q_1^\pi(s,a) - \tilde{Q}_2^\pi(s,a)| \\
&= \left| \left(R(s,a) + \gamma \int_{s' \in S} p_1(s'|s,a) V_1^\pi(s') ds' \right) - \left(R(s,a) + \gamma \int_{s' \in S} Lp_2(s'|s,a) \tilde{V}_2^\pi(s') ds' \right) \right| \\
&\leq \gamma \left| \int_{s' \in S} p_1(s'|s,a) V_1^\pi(s') - Lp_2(s'|s,a) \tilde{V}_2^\pi(s') ds' \right| \\
&\leq \gamma \left| \int_{s' \in S} p_1(s'|s,a) V_1^\pi(s') - p_1(s'|s,a) \tilde{V}_2^\pi(s') + p_1(s'|s,a) \tilde{V}_2^\pi(s') - Lp_2(s'|s,a) \tilde{V}_2^\pi(s') ds' \right| \\
&\leq \gamma \left| \int_{s' \in S} \left[p_1(s'|s,a)(V_1^\pi(s') - \tilde{V}_2^\pi(s')) + p_1(s'|s,a) \tilde{V}_2^\pi(s') - p_2(s'|s,a) \tilde{V}_2^\pi(s') \right. \right. \\
&\quad \left. \left. + p_2(s'|s,a) \tilde{V}_2^\pi(s') - Lp_2(s'|s,a) \tilde{V}_2^\pi(s') \right] ds' \right| \\
&\leq \gamma \left| \int_{s' \in S} p_1(s'|s,a)(V_1^\pi(s') - \tilde{V}_2^\pi(s')) ds' \right| + \gamma \left| \int_{s' \in S} (p_1(s'|s,a) - p_2(s'|s,a)) \tilde{V}_2^\pi(s') ds' \right| \\
&\quad + \gamma \left| \int_{s' \in S} p_2(s'|s,a) \tilde{V}_2^\pi(s') - Lp_2(s'|s,a) \tilde{V}_2^\pi(s') ds' \right|
\end{aligned}
$$

where the final expression was obtained by repeatedly adding and subtracting identical terms and using the triangle inequality. This expression must hold for all states s and actions a, so it must also hold for the maximum error over all states and actions:

$$
\begin{aligned}
\max_s \max_a |Q_1^\pi(s,a) - \tilde{Q}_2^\pi(s,a)| &\leq \gamma \int_{s' \in S} p_1(s'|s,a) \Delta_Q ds' + \gamma \left| \int_{s' \in S} (p_1(s'|s,a) - p_2(s'|s,a)) \tilde{V}_2^\pi(s') ds' \right| \\
&\quad + \gamma \left| \int_{s' \in S} \left(p_2(s'|s,a) \tilde{V}_2^\pi(s') - Lp_2(s'|s,a) \tilde{V}_2^\pi(s') \right) ds' \right| \\
\Delta_Q &\leq \gamma \Delta_Q + \gamma \left| \int_{s' \in S} (p_1(s'|s,a) - p_2(s'|s,a)) \tilde{V}_2^\pi(s') ds' \right| \\
&\quad + \gamma \left| \int_{s' \in S} \left(p_2(s'|s,a) \tilde{V}_2^\pi(s') - Lp_2(s'|s,a) \tilde{V}_2^\pi(s') \right) ds' \right| \\
&\leq \gamma \Delta_Q + \gamma V_{max} \left| \int_{s' \in S} p_1(s'|s,a) - p_2(s'|s,a) ds' \right| \\
&\quad + \gamma \left| \int_{s' \in S} \left(p_2(s'|s,a) \tilde{V}_2^\pi(s') - Lp_2(s'|s,a) \tilde{V}_2^\pi(s') \right) ds' \right| \\
&\leq \gamma \Delta_Q + \gamma V_{max} d_{var}(p_1(s'|s,a), p_2(s'|s,a)) + \gamma \varepsilon_{plan}
\end{aligned}
$$

where we have again used the triangle inequality. Therefore

$$\begin{aligned}\Delta_Q &\leq \gamma\Delta_Q + \gamma V_{\max} d_{var} + \gamma\varepsilon_{plan} \\ &= \frac{\gamma\frac{d_{var}}{1-\gamma}}{1-\gamma} + \frac{\gamma\varepsilon_{plan}}{1-\gamma},\end{aligned}$$

where we have used d_{var} as shorthand for $d_{var}(p_1(s'|s,a), p_2(s'|s,a))$

We have now expressed the error in the value function as the sum of the error due to the model approximation and the error due to using an approximate MDP planner. Using the assumptions in the lemma, the result immediately follows. ■

We can now use the prior lemmas to prove Theorem 1. First we need to examine under what conditions the two assumptions of the Simulation Lemma hold. The first assumption requires that $d_{var}(p_1(\cdot|s,a), p_2(\cdot|s,a)) \leq (1-\gamma)^2\varepsilon/(2\gamma)$ for all state-action tuples. From Lemma 6 this holds for a particular type-action tuple (which encompasses all state-action tuples where the state belongs to that type) if

$$\frac{||\beta_2 - \beta_1||_2}{\sqrt{\lambda_N}} + \sqrt{\frac{N^2\varepsilon}{\lambda_N} + \frac{2N^3 B_\sigma \varepsilon}{\lambda_N^2 - (N)^{1.5}\max_{ij}|\tilde{\sigma}_{ij} - \sigma_{ij}|\lambda_N}} \leq \frac{(1-\gamma)^2\varepsilon}{2\gamma} \tag{5}$$

and

$$\max_{ij}|\tilde{\sigma}_{ij} - \sigma_{ij}| \leq \varepsilon. \tag{6}$$

We can ensure Equation 5 holds by splitting the error into three terms:

$$\begin{aligned}\frac{\|\beta_2 - \beta_1\|_2}{\sqrt{\lambda_N}} &\leq \frac{(1-\gamma)^2\varepsilon}{4\gamma} \\ \frac{N^2\varepsilon}{\lambda_N} &\leq \frac{(1-\gamma)^4\varepsilon^2}{32\gamma^2} \\ \frac{2N^3 B_\sigma \varepsilon}{\lambda_N^2 - (N)^{1.5}\max_{ij}|\tilde{\sigma}_{ij} - \sigma_{ij}|\lambda_N} &\leq \frac{(1-\gamma)^4\varepsilon^2}{32\gamma^2}.\end{aligned}$$

Given these three equations, and Equation 6, we can obtain bounds on the error in the dynamics parameter estimates:

$$\|\tilde{\beta} - \beta\|_2 \leq \frac{(1-\gamma)^2\varepsilon\lambda_N^{0.5}}{4\gamma} \tag{7}$$

$$\max_{ij}|\tilde{\sigma}_{ij} - \sigma_{ij}| \leq \frac{(1-\gamma)^4\varepsilon^2\lambda_N}{32\gamma^2 N^2} \tag{8}$$

$$\max_{ij}|\tilde{\sigma}_{ij} - \sigma_{ij}| \leq \frac{(1-\gamma)^4\varepsilon^2\lambda_N^2}{16\gamma^2 N^3 B_\sigma + (1-\gamma)^4\varepsilon^2(N)^{1.5}\lambda_N}. \tag{9}$$

Assume[4] that $\lambda_N \leq 1$, $N > 1$ and $B_\sigma \geq 1$. In this case the upper bound in Equation 9 will be at least as small as the upper bounds in Equations 7 and 8.

4. This is just a simplifying assumption and it is trivial to show the bounds will have a similar polynomial dependence on the parameters if the assumptions do not hold.

We therefore require that the error ε in the model parameters be bounded by

$$\varepsilon \leq \frac{(1-\gamma)^4\varepsilon^2\lambda_N^2}{16\gamma^2N^3B_\sigma + (1-\gamma)^4\varepsilon^2(N)^{1.5}\lambda_N} \tag{10}$$

(from Equation 9). Lemma 2 provides a guarantee on the number of samples $T = \frac{12N^2B_\sigma^2}{\varepsilon^2 g}$ required to ensure with probability at least $1-g$ that all the model parameters have error of most ε. In order to ensure that the model parameters for all actions and types simultaneously fulfill this criteria with probability δ, it is sufficient to require that $g = \delta/(|A|N_T)$, from the union bound. We can then substitute this expression for g and Equation 10 into the expression for the number of samples T:

$$T = \frac{12N^2B_\sigma^2}{\left(\frac{(1-\gamma)^4\varepsilon^2\lambda_N^2}{16\gamma^2N^3B_\sigma+(1-\gamma)^4\varepsilon^2(N)^{1.5}\lambda_N}\right)\frac{\delta}{|A|N_T}} = \frac{12N^2|A|N_TB_\sigma^2(16\gamma^2N^3B_\sigma + N^1.5\lambda_N(1-\gamma)^4\varepsilon^2)^2}{(1-\gamma)^8\varepsilon^2\delta\lambda_N^4}.$$

Given this analysis, the first assumption of the Simulation Lemma holds with probability at least $1-\delta$ after

$$O\left(\frac{N^8|A|N_TB_\sigma^4}{(1-\gamma)^8\varepsilon^4\delta(\lambda_N)^4}\right)$$

samples.

The second assumption in the Simulation Lemma requires that we have access to an MDP planner than can produce an approximate solution to our typed-offset-dynamics continuous-state MDP. At least one such planner exists if the reward model is Lipschitz continuous; under a set of four conditions, Chow and Tsitsiklis (1991) proved that the optimal value function V_ε of a discrete-state MDP formed by discretizing a continuous-state MDP into $\Theta(\varepsilon)$-length (per dimension)[5] grid cells is an ε-close approximation of the optimal continuous-state MDP value function, denoted by V^*:

$$||V_\varepsilon - V^*||_\infty \leq \varepsilon.$$

The first condition used to prove the above result is that the reward function is Lipschitz-continuous. In our work, the reward function is assumed to be given, so this condition is a prior condition on the problem specification. The second condition is that the transition function is piecewise Lipschitz continuous. In other words, the transition model is Lipschitz-continuous over each of a set of finite subsets that cover the state space, and that the boundary between each subset region is piecewise smooth. For each type and action our transition model is a Gaussian distribution, which is Lipschitz-continuous, and there are a finite number of different types so it is piecewise Lipschitz-continuous. As long as our domain fulfills our earlier stated assumption that there are a finite number of different type regions, and the boundaries between each are piecewise smooth, then Chow and Tsitsiklis's second assumption is satisfied. The third condition is that the dynamics probabilities represent a true probability measure that sums to one ($\int_{s'} p(s'|s,a) = 1$), though the authors show that this assumption can be relaxed to $\int_{s'} p(s'|s,a) \leq 1$ and the main results still hold.

5. More specifically, the grid spacing h_g must satisfy $h_g \leq \frac{(1-\gamma)^2\varepsilon}{K_1+2KK_2}$ and $h_g \leq \frac{1}{2K}$ where K is the larger of the Lipschitz constants arising from the assumptions discussed in the text, and K_1 and K_2 are constants discussed in Chow and Tsitsiklis (1991). For small ε any h_g satisfying the first condition will automatically satisfy the second condition.

In our work, our dynamics models are defined to be true probability models. Chow and Tsitsiklis's final condition is that there must be a bounded difference between any two controls. In our case we consider only finite controls, so this property holds directly. Assuming the reward model fulfills the first condition, our framework satisfies all four conditions made by Chow and Tsitsiklis, and we can use their result.

By selecting fixed grid points at a regular spacing of $\frac{(1-\gamma)^2\varepsilon}{2\gamma}$, and by requiring that there exist at least one grid point placed in each contiguous single-type region, we can ensure that the maximum error in the approximate value function compared to the exactly solved value function is at most $\frac{(1-\gamma)^2\varepsilon}{2\gamma}$. This provides a mechanism for ensuring the second assumption of the Simulation Lemma holds. In other words, if the grid width used is the minimum of $\frac{(1-\gamma)^2\varepsilon}{2\gamma}$ and the minimum contiguous length of a single-type region, then the resulting value function using this discrete approximate planner is no more than $\frac{(1-\gamma)^2\varepsilon}{2\gamma}$-worse in value than the optimal exact planner for the continuous-state MDP.[6]

Type-action tuples with at least T samples are defined to be "known." From the analysis above, the estimated dynamics model for such types have a d_{var} value from the true known type-action dynamics model of at most $(1-\gamma)^2\varepsilon/(2\gamma)$. All unknown type-action tuples are defined to be self-loops. Therefore the dynamics models of our known-type, estimated dynamics MDP $\hat{M}_K$ relative to a known-type MDP with the true dynamics parameters M_K have a d_{var} of zero for all the unknown type-action tuples (since these are always defined as self loops) and at most $(1-\gamma)^2\varepsilon/(2\gamma)$ for all the known type-action tuples. Hence the first assumption of the Simulation Lemma holds. The second assumption of the Simulation Lemma is fulfilled given the analysis in the prior paragraph. Given these two assumptions are satisfied, the Simulation Lemma guarantees that the approximate value of our known-type MDP $\hat{M}_K$ under its greedy policy π ($\pi(s) = \text{argmax}_a Q_{\hat{M}_K}(s,a)$) is ε-close to the optimal value of the known-type MDP with the true dynamics parameters M_K under policy π: $||\tilde{V}^{\pi}_{\hat{M}_K} - V^{\pi}_{M_K}||_\infty \leq \varepsilon$. This fulfills condition 2 of Strehl et al. (2006).

The first condition of Strehl et al. (2006) can be re-expressed as:

$$Q^*(s,a) - Q_{\hat{M}_K}(s,a) = (Q^*(s,a) - Q_{M_K}(s,a)) + (Q_{M_K}(s,a) - Q_{\hat{M}_K}(s,a)) \leq \varepsilon.$$

We start by considering the first expression, $Q^*(s,a) - Q_{M_K}(s,a)$. If all type-action pairs are known, then M_K is the same as the original MDP, and this expression equals 0. If some type-action pairs are unknown, then the value of states of that type, associated with that action, becomes $V_{\max}$ under MDP M_K. As all known type-action pairs have the same reward and dynamics model as the original MDP, this implies that the value Q_{M_K} must be either equal or greater than Q^*, since all the value of all unknown state-actions is at least as great in Q_{M_K} as their real value Q^*. For this reason, $Q^*(s,a) - Q_{M_K}(s,a)$ is always less than or equal to 0.

We next consider $Q_{M_K}(s,a) - Q_{\hat{M}_K}(s,a)$. The variational distanced_{var} between the dynamics models of M_K and $\hat{M}_K$ for all unknown type-action tuples is zero, because all the dynamics of unknown tuples are self loops. As discussed above, the d_{var} between all known type-action tuples is at most $(1-\gamma)^2\varepsilon/(2\gamma)$. We can then apply the Simulation Lemma to guarantee that $|Q^{\pi}_{M_K}(s,a) - Q^{\pi}_{\hat{M}_K}(s,a)| \leq \varepsilon$. As a result, the first condition of Strehl et al. (2006) holds.

The third condition limits the number of times the algorithm may experience an unknown type-action tuple. Since there are a finite number of types and actions, this quantity is bounded above by

6. The condition of the extent of a typed region is a requirement in order to ensure that the discrete-representation doesn't skip over a smaller region of a different type, that may have a different optimal policy.

$N_{at}N_T|A|$, which is a polynomial in the problem parameters $(N, |A|, N_T, \frac{1}{\varepsilon}, \frac{1}{\delta}, \frac{1}{1-\gamma}, \frac{1}{\lambda_N}, B_\sigma)$. Therefore, our algorithm fulfills the three specified criteria and the result follows. ■

3.3 Discussion

Prior PAC-MDP work has focused predominantly on discrete-state, discrete-action environments. The sample complexity of the R-max algorithm by Brafman and Tennenholtz (2002) scales with the number of actions and the square of the number of discrete states, since a different dynamics model is learned for each discrete state-action tuple. In environments in which states of the same type share the same dynamics, Leffler et al. (2007) proved that the sample complexity scales with the number of actions, number of discrete states, and number of types. Assuming the number of types is typically much less than the number of states, this can result in significantly faster learning, as Leffler et al. (2007) demonstrate empirically. However, a naïve application of either technique to a continuous-state domain involves uniformly discretizing the continuous-state space. This procedure that results in a number of states that grows exponentially with the dimension of the state space. In this scenario the approaches of both Brafman and Tennenholtz (2002) and Leffler et al. (2007) will have a sample complexity that scales exponentially with the state space dimension, though the approach of Leffler et al. (2007) will scale better if there are a small number of types.

In contrast, the sample complexity of our approach scales polynomially in the numbers of actions and types as well as state space dimension, suggesting that it is more suitable for high dimensional environments. Our results follow the results of Strehl and Littman (2008), who gave an algorithm for learning in continuous-state and continuous-action domains that has a sample complexity that is polynomial in the state space dimension and the action space dimension. Our work demonstrates that we can get similar bounds when we use a more powerful dynamics representation (allowing states to have different dynamics, but sharing dynamics within the same types), learn from experience the variance of the dynamics models, and incorporate the error due to approximate planning.

Our analysis presented so far considers the discounted, infinite-horizon learning setting. However, our results can be extended to the H-step finite horizon case fairly directly using the results of Kakade (2003). Briefly, Kakade considers the scenario where a learning algorithm $\mathcal{A}$ is evaluated in a cycling H-step periods. The H-step normalized undiscounted value U of an algorithm is defined to be the sum of the rewards received during a particular H-step cycle, divided by H. Kakade defines $\mathcal{A}$ to be ε-optimal if the value over a state-action trajectory, until the end of the current H-step period, is within ε of the optimal value U. Using this alternate definition of value requires a modification of the Simulation Lemma which results in a bound on Δ_Q of $(H-1)V_{max}d_{var} + (H-1)\varepsilon_{plan}$. In short, the $\gamma/(1-\gamma)$ terms has been replaced with $H-1$. The earlier results on the number of samples required to obtain good estimates of the model parameters are unchanged, and the final result follows largely as before, except we now have a polynomial dependence on the horizon H of the undiscounted problem, compared to a polynomial dependence on the discount factor $1/\gamma$.

Finally, though our focus is on sample complexity, it is also important to briefly consider computational complexity. To ensure the approximate planner produces highly accurate results, our algorithm's worst-case computational complexity is exponential in the number of state dimensions. While this fact prevents it from being theoretically computationally efficient, in the next section we

present experimental results that demonstrate our algorithm performs well empirically compared to a related approach in a real-life robot problem.

4. Experiments

First we demonstrate the benefit of using domain knowledge about the structure of the dynamics model on the standard reinforcement-learning benchmark, PuddleWorld (Boyan and Moore, 1995). Then we illustrate the importance of explicitly learning the variance parameters of the dynamics in a simulated "catching a plane" experiment. This is in contrast to some past approaches which assume the variance parameters to be provided, such as Strehl and Littman (2008).

Our central hypothesis is that offset dynamics are a simple model that reasonably approximate several real world scenarios. In our third experiment, we applied our algorithm to a simulated problem using in which an agent needs to learn the best route to drive to work, and used date collected from real cars to simulate the dynamics. In our final experiment we applied our algorithm to a real-world robot car task in which the car most cross varying terrain (carpet and rocks) to reach a goal location.

CORL requires a planner to solve the estimated continuous-state MDP. Planning in continuous-state MDPs is an active area of research in its own right, and is known to be provably hard (Chow and Tsitsiklis, 1989). In all our experiments we used a standard technique, Fitted Value Iteration (FVI), to approximately solve the current MDP. In FVI, the value function is represented explicitly at only a fixed set of states. In our experiments these fixed states are uniformly spaced in a grid over the state space. Planning requires performing Bellman backups for each grid point; the value function over points not in this set is computed by function interpolation. We used Gaussian kernel functions as the interpolation method. Using sufficiently small kernel widths relative to the spacing of the grid points will make the approach equivalent to using a nearest neighbour standard discretization. However, there are some practical advantages in coarse grids to more smooth methods of interpolation. We discuss this issue in more depth in the next section.

In each experiment, N_{at} was tuned based on informal experimentation.

4.1 Puddle World

Puddle world is a standard continuous-state reinforcement-learning problem introduced by Boyan and Moore (1995). The domain is a two-dimensional square of width 1 with two oval puddles, which consist of the area of radius 0.1 around two line segments, one from $(0.1, 0.75)$ to $(0.45, 0.75)$ and the other from $(0.45, 0.4)$ to $(0.45, 0.8)$. The action space consists of the four cardinal directions. Upon taking an action the agent moves 0.05 in the specified cardinal direction with added Gaussian noise $\mathcal{N}(0, 0.001 * I)$, where I is a two-by-two identity matrix. The episode terminates when the agent reaches the goal region which is defined as the area in which $x + y \geq 1.9$. All actions receive a reward of -1 unless the agent is inside a puddle, in which case it then receives a reward of -400 times the distance inside the puddle. Figure 1 provides a graphical depiction of the puddle world environment.

We expect CORL will outperform prior approaches on this problem for two reasons. The first is that we assume the reward is given. However even if the reward model is provided, most past work still has to learn the dynamics model for this world, which is still a significant undertaking. The second reason is a feature of the CORL algorithm: its dynamics model uses the additional information that the dynamics for one action for all states of the same type are identical. Here there

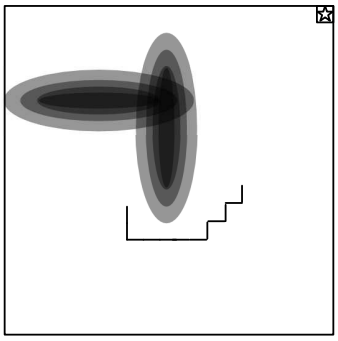


Figure 1: Puddle world. The small square with a star in it denotes the goal region. The black ovals represent the puddles and the black line represents a sample partial trajectory of an agent navigating around this world.

is only a single type, so CORL must learn only 4 sets of transition parameters, one for each action. Our goal here is simply to demonstrate that this additional information can lead to significantly faster learning over other more general techniques.

Puddle world has previously been used in reinforcement-learning competitions, and our evaluation follows the procedure of the second bakeoff (Dutech et al., 2005). We initially generated 50 starting locations, and cycled through these when initializing each episode. Each episode goes until the agent reaches the goal, or has taken 300 actions. We report results from taking the average reward within fifty sequential episodes (so the first point is the average reward of episodes 1-50, the second point is the average of episodes 51-100, etc.). We set N_{at} to be 15, and solved the approximate MDP model using Fitted Value Iteration with Gaussian kernel functions. The kernel means were spaced uniformly in a 20x20 grid across the state space (every 0.05 units), and their standard deviation was set to 0.01. Note that in the limit as the standard deviation goes to 0 the function interpolation becomes equivalent to nearest neighbour. Nearest neighbour is the interpolation method used in the approximate continuous-state MDP solver by Chow and Tsitsiklis (1991) which provides guarantees on the resulting value function approximation. However, since computational complexity scales exponentially with the grid discretization, practical application can require the use of coarse grids. In this case, we found that using a smoother function interpolation method empirically outperformed a nearest neighbour approach. In particular, we found that a kernel width, which is a measure of the standard deviation, of 0.01 gave the best empirical performance. This value lies in the middle of kernel widths which are smaller than the variance of the dynamics models and the grid spacing and those widths larger than the grid spacing and dynamics models.

We compare our results to the reported results of Jong and Stone (2007)'s Fitted R-max and to Lagoudakis and Parr (2003)'s Least Squares Policy Iteration (LSPI).[7] Fitted R-max is an instance-based approach that smoothly interpolates the dynamics of unknown states with previously observed transitions, and takes a similar approach for modeling the reward function. LSPI is a policy iteration approach which uses a linear basis function representation of the state-action values, and uses a set of sample transitions to compute the state-action values.

7. Both results reported come from the paper of Jong and Stone (2007).

Algorithm	Number of episodes			
	50	100	200	400
CORL	−19.9	−18.4	−20.3	−18.7
Fitted R-max	−130	−20	20	−20
LSPI	−500	−330	−310	−80

Table 1: PuddleWorld results. Each entry is the average reward/episode received during the prior 50 episodes. Results for the other algorithms are as reported by Jong and Stone (2007).

On the first 50 episodes, Fitted R-max had an average reward of approximately −130, though on the next 50 episodes it had learnt sufficiently good models that it reached its asymptotic performance of an average reward of approximately −20. In contrast, CORL learned a good model of the world dynamics within the very first episode, since it has the advantage of knowing that the dynamics across the entire state space are the same. This meant that CORL performed well on all subsequent episodes, leading to an average reward on the first 50 episodes of −19.9. Least Squares Policy Iteration (Lagoudakis and Parr, 2003) learned slower than Fitted R-max. Results are summarized in Table 1.

It is worth a short remark on the comparability of the reported results, as LSPI and Fitted R-max were run without knowing the reward model. LSPI's performance on a deterministic reward reinforcement-learning problem such as PuddleWorld will be identical to its performance in the known-reward case, as the rewards in the sampled transitions will be the same in either situation. Given this, assuming known reward, as we do for CORL, will not change the LSPI results. In contrast we do expect that Fitted R-max will be slightly faster if it does not have to learn the reward model. This is because a wider interpolation width can be selected if the reward is known and only the dynamics are unknown, since all states share the same dynamics. However, even in this case, Fitted R-max will be approximating the Gaussian dynamics by a set of observed transitions, and so it appears likely that CORL will still be faster than Fitted R-max since CORL assumes the (true) parametric representation of the transition model.

In summary, given the reward function, CORL can learn a good dynamics model extremely quickly in PuddleWorld, since the dynamics are typed-offset with a single type. This additional information enables CORL to learn a good policy for puddle world much faster than Fitted R-max and LSPI. This experiment illustrates the advantage of CORL when the transition dynamics are known to be identical across the state space, and to follow a noisy offset model.

4.2 Catching a Plane

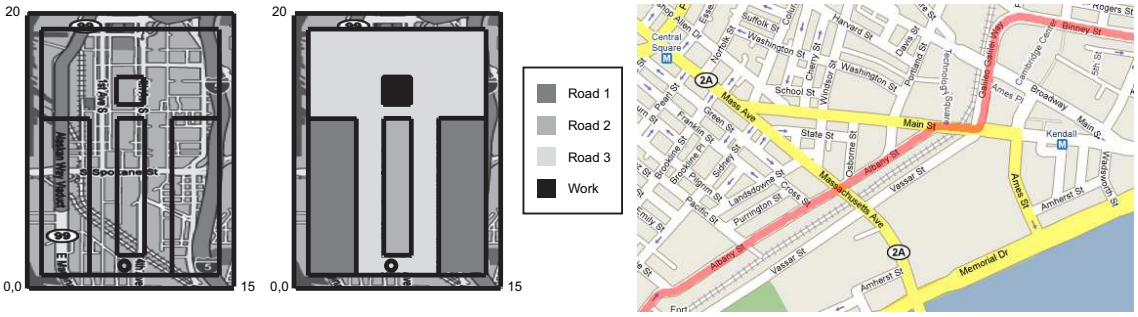
We next consider some examples with multiple types. Thousands of people now rely on internet maps or GPS units to do efficient trip planning, and better traffic prediction is an area of recent research (Horvitz et al., 2005). In many street systems, there exist a number of different classes of road types according to the designated speed limits associated with these roads. These different road types are often also associated with different variances. For example, while highways have mean speeds that are faster than small side streets, highways typically have a very large variance; rush hour highway speeds may often be slower than smaller side streets.

Figure 2: Motivating example where an agent must drive from a start location (designated with an S) to a goal area (shown with a highlighted rectangle) in time for a deadline, and can choose to take large high speed and variance roads, or slower small variance roads.

In our first experiment we considered a scenario in which learning this variance is critical: an agent must learn the best way to drive to an airport in time to catch a plane. A similar real-world environment is depicted in Figure 2. The agent starts from home and can either drive directly along a small side street, or cross over to the left or right to reach a main highway. The agent goes forward more quickly on the highway (with a mean offset of 2 units), but the highway also has a high variance of 0.49. In contrast, on the small side streets the agent goes forward more slowly (a mean offset of 1 unit) but with very small variance (0.00001). The state space is four dimensional, consisting of the agent's current x and y location, its orientation, and the amount of time that has passed. On each step five minutes pass. The agent can drive in a 14 by 19 region: outside of this is considered to be too far away.[8] If the agent exits this region it receives a reward of -1. The cost for each step is -0.05, the reward for reaching the airport in time for check in is $+1$ and the cost for not making the airport in time is -1. The discount factor is set to 1.0. N_{at} was set to 10. An episode starts when the agent takes its first step and lasts until the agent reaches the goal, exits the allowed region, or reaches the time at which check in for the plane closes. The agent starts at location $7,3$ facing north, and must figure out a way to reach the goal region which spans $[6.5-8.5, 15.5\text{-}17.5]$. To solve the underlying MDP, fitted value iteration was used. The fixed points were regularly spaced grid points with 15 across the x dimension, 20 across the y dimension, 4 orientation angles, and 21 time intervals. This yielded over 25,000 fixed points.

The correct path to take depends on the amount of time left. Here we explored three different deadline scenarios: when the agent has 60 minutes remaining (*RunningLate*), 70 minutes remaining (*JustEnough*), or 90 minutes remaining until check in closes for the plane (*RunningEarly*). In all three scenarios, the agent learned a good enough model of the dynamics within 15 episodes to compute a good policy. Note that using a naïve discretization of the space with the FVI fixed points as states and applying R-max would be completely intractable, as a different model would have to be learned for each of over 25,000 states. We display results for all three scenarios in Table 2. In

8. Note that this could be a reasonable assumption in some cases, such as when doctors had to stay within a certain radius of a hospital when on call.



(a) (b) (c)

Figure 3: (a) Simulated world map showing example environment. (b) World divided into a set of discrete types. Agent starts at driveway (black circle) and tries to learn a good route to reach work. (c) An example section of a car trajectory (constructed from a set of GPS locations).

the *RunningLate* scenario the agent learned that the side streets are too slow to allow it to ever reach the airport in time. Instead it took the higher variance highway which enables it to reach the airport in time in over half the episodes after N_{at} is reached for all types: its average reward is -0.4833 and it takes on average 56.62 minutes for it to reach the airport. In *JustEnough* the agent learned that the speed of side streets is sufficiently fast for the agent to reach the airport consistently in time, whereas the higher speed and variance highway would result in the agent failing to reach the check in time in some cases. Here the agent always reached the goal and receives an average reward of 0.45. In the *RunningEarly* scenario, the agent has enough time so that it can take either route and reliably reach the airport in time. In this scenario it learned to always take the highway, since in expectation that route will be faster. The average reward here was 0.4629.

This simulation serves to illustrate that our algorithm can quickly learn to perform well, in situations in which learning the variance is critical to ensure good performance.

4.3 Driving to Work

In our second experiment we again consider a simulated trip routing problem, but we now generate transitions in the simulator by sampling from real traffic data distributions. Here an agent must learn the best series of actions to drive from home to work in a small simulated world (see Figure 3(a) and 3(b)). The state consists of the current coordinates (x, y) and the orientation of the agent. There are three road types and each road type is associated with a different distribution of speeds. The

Scenario	Deadline (min)	Mean Reward/Episode	Mean Time to Reach Goal (min)
RunningLate	60	-0.4833	56.62
JustEnough	70	0.45	60
RunningEarly	90	0.4629	58.6

Table 2: Catching a plane: results after N_{at} has been reached for all types.

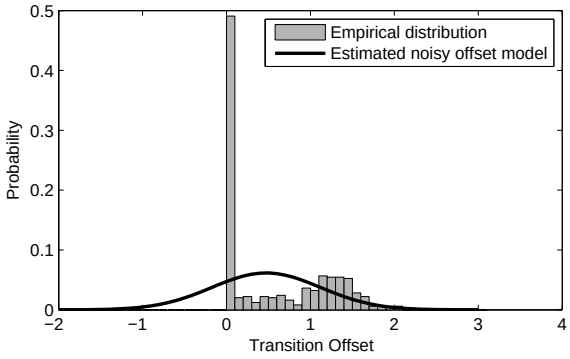
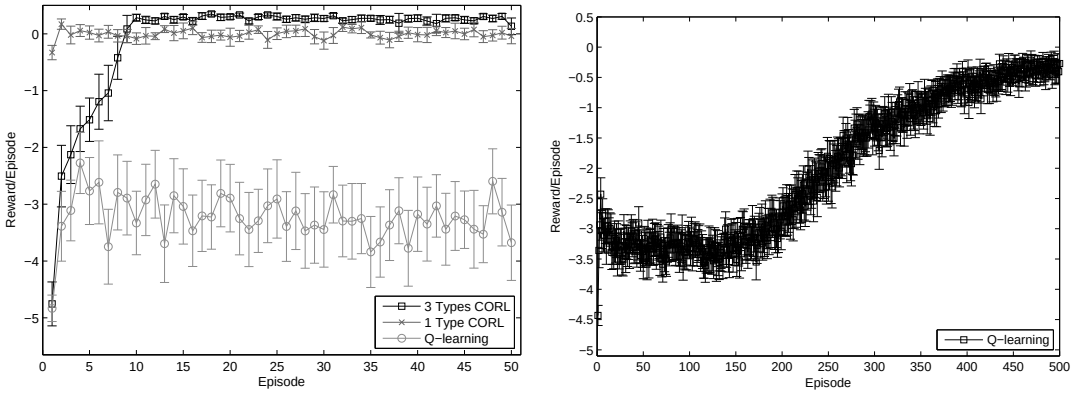


Figure 4: A histogram of car speeds on small roads that is used to generate transitions on road type 2 and the estimated dynamics model parameters found during the experiment

distributions were obtained from the CarTel project (Eriksson et al., 2008), which consists of a set of car trajectories from the Boston, Massachusetts area. GPS locations and time stamps are stored approximately every second from a fleet of 27 cars.[9] A section from one car trajectory is shown in Figure 3(c). Using this data set we extracted car trajectories on an interstate highway, small side streets and a local highway: these constitute types 1, 2 and 3 respectively in the simulation world. Each car trajectory consisted of a set of D GPS+time data points, which was converted into a set of $D-1$ transitions. Each transition in the simulation was sampled from these real-world transitions; for example, transitions in the simulator on road type 2 were sampled from real-world transitions on small side streets. Transitions from all three road types were all rescaled by the same constant in order to make the distances reasonable for the simulated world.[10] Figure 4 displays a histogram of rescaled transitions associated with small side streets. This figure shows that the speed distribution for small side streets was not Gaussian: the speed distribution for the other two street types was also not Gaussian. In particular, in no trajectories used does the car ever go backwards, whereas in some Gaussian models there will be small probability of this occurring. In this experiment we sought to investigate how well a noisy offset model could function in this environment, and the benefit of directly modelling different types of roads. Each transition in the simulated environment was sampled from the histogram of speeds associated with the road type at the agent's current position. Therefore, the data from the simulator is closer to the real environment than to the Gaussian distributions assumed by the learning algorithm.

The agent received a reward of 1 for reaching the work parking lot, -0.05 for each step, and -1 if it left the local area. Each episode finished when the agent either reached the goal, left the local area, or had taken 100 steps. An agent can go left, right or straight at each step. The transition induced by a straight action was determined by the road type as specified in the prior paragraph, and going left or right changed the orientation of the agent by 90 degrees with a very small amount of noise. The number of samples needed until a type-action tuple is known, N_{at}, was set to be 20. The discount factor was 1. The agent was always started in the same location and was allowed to learn across a set of 50 episodes. Results were averaged across 20 rounds of 50 episodes per

9. See more information about the project at `http://cartel.csail.mit.edu/`.
10. We also removed outlier transitions, as extremely fast speeds/transitions were likely to be errors in the log file.



(a) (b)

Figure 5: Reward versus episode. (a) Compares CORL with 3 types and 1 type to Q-learning. Results are averaged over 20 rounds (50 episodes per round). Error bars show 95% confidence intervals. (b) Shows Q-learning with 500 episodes per round, averaged over 100 rounds.

round. In one experiment the agent was given full knowledge of the three world types, and learned a different dynamics model for each type and action. In the second experiment the agent assumed there was only a single type and learned a dynamics model for each action. We also compared our approach to *Q*-learning over a uniformly-spaced discrete grid over the environment with an ε-greedy policy. We used a discretization that was identical to the fixed points used in the fitted value iteration planner of CORL. Points were mapped to their nearest neighbors. Q-learning requires specifying two parameters: the learning rate α which determines how much to adjust the state-action value estimates after each update, and ε which specifies how often to take a random action instead of the action that maximizes the current Q values. In this experiment α was set to 1.0 and decreased by multiplying by a factor of 0.9999 at each step.[11] We set ε to be 0.1.

The CORL results are displayed in Figure 5(a). This figure displays three encouraging results. The first is that in both CORL algorithms the agent learned to consistently reach the goal: the only way that the agent can receive a reward greater than -1 is to reach the goal, and all confidence intervals lie above -1 for all episodes after 10, indicating that the agent in both cases was successfully reaching the goal. This is promising because even though the underlying dynamics models were not exactly Gaussian noisy offset dynamics, a noisy offset model approximation was sufficient for the agent to learn a good policy in this environment. The estimated parameters computed for one type and action are displayed in Figure 4.

The second result is that the policy found by the agent that models all three types differently resulted in significantly higher reward than modeling the world with a single type: its performance suffered initially because it takes longer to learn a model of the world dynamics, but from about episode 10-50 modelling all types separately resulted in significantly higher reward per episode than modelling all types as the same. Table 3 displays the average reward of both approaches on episodes 10-50. These results demonstrate that traffic data does exhibit different speed distributions

11. We tried different decay factors for the α parameter but found that this worked better than decaying α more rapidly.

Algorithm	Average reward/episode
CORL with 3 types	0.27
CORL with 1 type	0.00
Q-learning	-3.2485

Table 3: Average reward on episodes 10-50 for the driving to work example.

on different types of roads, and that by considering such differences CORL can improve route directions even in a small simulated example.

The third result is that both CORL algorithms significantly outperformed *Q*-learning: again see Table 3 for comparing the short term performance of *Q*-learning to the CORL algorithm. This is not surprising since *Q*-learning is a model-free approach that trades off speed of computation per step in return for not requiring consistency between its state values through learned reward and dynamics models. Here in particular there is a large amount of structure in the domain that *Q*-learning cannot use. *Q*-learning does eventually begin to consistently reach the goal but this is only after about 500 episodes, more than an order of magnitude longer than the CORL algorithms took to find a good policy. These results are displayed in Figure 5(b). Such results argue that in situations where data is costly to gather, using a model can be extremely helpful.

4.4 Robot Navigation Over Varying Terrain

We also tried our algorithm in a real-life robotic environment involving a navigation task where a robotic car must traverse multiple surface types to reach a goal location. This experiment is a second example where a noisy offset dynamics model provides a sufficiently good representation of the real-world dynamics to allow our algorithm to learn good policies. We compared to the RAM-Rmax algorithm (Leffler et al., 2007), a provably efficient RL algorithm for learning in discrete-state worlds with types. The authors demonstrated that, by explicitly representing the types, they could get a significant learning speedup compared to R-max, which learns a separate dynamics model for each state. The RAM-Rmax algorithm represents the dynamics model using a list of possible next outcomes for a given type. CORL works directly with continuous-valued states, resulting in the improved sample complexity discussed earlier. This is achieved through assuming a fixed parametric representation of the dynamics, which is a less flexible model than the one used in RAM-Rmax. In this experiment we were interested in whether our representation was still rich enough to capture the real world dynamics involved in varying terrain traversal. We also investigated the computational load of CORL compared to RAM-Rmax, since by restricting our representation size we hoped to also achieve computational savings.

In this experiment we ran a LEGO® Mindstorms NXT robot (see Figure 6(b)) on a multi-surface environment. A tracking pattern was placed on the top of the robot and an overhead camera was used to determine the robot's current position and orientation. The domain, shown in Figure 6(a), consisted of two types of terrain: rocks embedded in wax and a carpeted area. The goal was for the agent to begin in the start location (indicated in the figure by an arrow) and end in the goal without going outside the environmental boundaries. The rewards were -1 for going out of

(a) (b)

Figure 6: (a) Image of the environment. The start location and orientation is marked with an arrow. The goal location is indicated by the circle.(b) LEGO® robot.

bounds, $+1$ for reaching the goal, and -0.01 for taking an action. Reaching the goal and going out of bounds ended the episode and resulted in the agent getting moved back to the start location.[12]

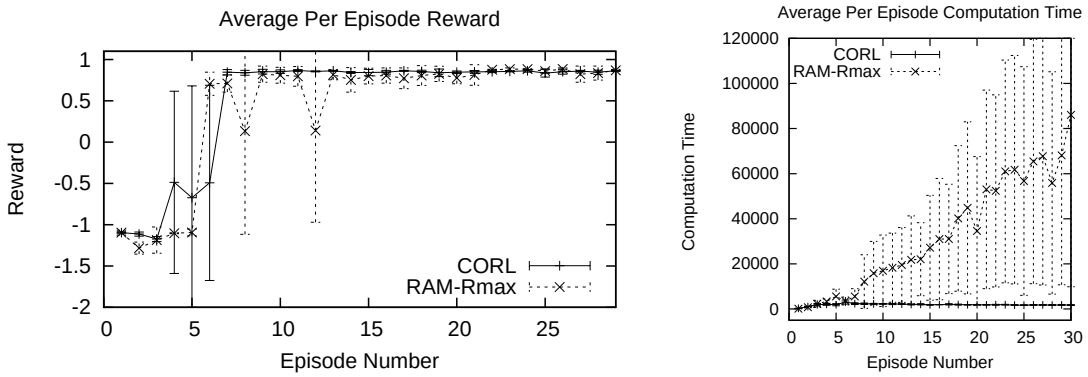
Due to the close proximity of the goal to the boundary, the agent needs an accurate dynamics model to reliably reach the goal. Part of the difficultly of this task is that the actions were going forward, turning left, and turning right. Without the ability to move backwards, the robot needed to approach the goal accurately to avoid falling out of bounds.

For the experiments, we compared our algorithm ("CORL") and the RAM-Rmax algorithm ("RAM"). The fixed points for the fitted value iteration portion of our algorithm were set to the discretized points of the RAM-Rmax algorithm. Both algorithms used an EDISON image segmentation system to uniquely identify the current surface type. The reward function was provided to both algorithms.

The state space is three dimensional: x, y position and orientation. Our algorithm implementation for this domain used a full covariance matrix to model the dynamics variance. For the RAM-Rmax agent, the world was discretized to a forty-by-thirty-by-ten state space. In our algorithm, we used a function approximator of a weighted sum of Gaussians, as described in Section 2. We used the same number of Gaussians to represent the value function as the size of the state space used in the discretized algorithm, and placed these fixed Gaussians at the same locations. The variance over the x and y variables was independent of each other and of orientation, and was set to be 16. To average orientation vectors correctly (so that -180° degrees and 180° do not average to 0) we converted orientations θ to a Cartesian coordinate representation $x_\theta = \cos(\theta), y_\theta = \sin(\theta)$. The variance over these two was set to be 9 for each variable (with zero covariance). For our algorithm and the RAM-Rmax algorithm, the value of N_{at} was set to four and five, respectively, which was determined after informal experimentation. The discount factor was set to 1.

Figure 7(a) shows the average reward with standard deviation for each of the algorithms over three runs. Both algorithms are able to receive near-optimal reward on a consistent basis, choosing

12. A video of the task can be seen at `http://people.csail.mit.edu/emma/corl/SuccessfulRun.mov` and `http://people.csail.mit.edu/emma/corl/SuccessfulRun.wmv`.



(a) (b)

Figure 7: (a) Reward received by algorithms averaged over three runs. Error bars show one standard deviation. (b) Total time taken by algorithms averaged over three runs. Error bars show one standard deviation.

similar paths to the goal. Our dynamics representation is sufficient to allow our algorithm to learn well in this real-life environment.

In addition, by using a fixed size (parametric) dynamics representation, the computational time per episode of our algorithm is roughly constant (Figure 7(b)). In the implementation of RAM-Rmax, the computational time grew with the number of episodes due to its dynamics model representation. This suggests that using a fixed size dynamics representation can have significant computation benefits. Overall CORL performed well in this domain, both in terms of reward achieved and computation required.

5. Conclusion and Future Work

In this paper we have presented CORL, an algorithm for efficiently learning to act in typed, continuous-state environments. CORL has a sample complexity that scales polynomially with the state space dimension and the number of types: this bound also directly incorporates the error due to approximate planning. Experiments on a simulated driving example using real world car data, and a small robot navigation task, suggest that noisy offset dynamics are a sufficiently rich representation to allow CORL to perform well in some real-world environments.

Due to the approximate MDP planning, we cannot currently guarantee both polynomial sample complexity and polynomial computational complexity. There are a number of recent advances in continuous-state MDP planning (Kocsis and Szepesvári, 2006; Kveton and Hauskrecht, 2006; Marecki and Tambe, 2008) as well as alternate approaches such as forward search techniques. In the future it would be interesting to investigate whether there exist alternate MDP planners that can provide ε-close approximations to the exact solutions with a computational complexity that scales polynomially with the number of state dimensions. Such approaches would enable CORL to achieve the appealing goal of polynomial dependence on the number of state dimension for both sample complexity and computational complexity.

Finally, the bounds provided remain overly large for many practical applications. We are broadly interested in developing techniques that can tighten the gap between the theoretical bounds and those needed for practical performance in real-world reinforcement learning.

Acknowledgments

B. Leffler, L. Li and M. Littman were partially supported by NSF DGE 0549115, a DARPA Transfer Learning grant and a National Science Foundation (NSF) Division of Information and Intelligent Systems (IIS) grant. E. Brunskill and N. Roy were supported by NSF IIS under Grant #0546467. Special thanks to Jacob Eriksson, Hari Balakrishnan, Samuel Madden, and the rest of the MIT CarTel team for generously providing access to their data set. We also appreciate the gracious time of the reviewers in helping us improve this work.

Appendix A.

Lemma 5 *Assume* $\max_d |\tilde{\beta}_d - \beta_d| \leq \varepsilon$ *for* $\varepsilon < 1/4$. *Given any* $\delta > 0$, *define* $T_\sigma = \frac{12N^2B_\sigma^2}{\delta\varepsilon^2}$. *If there are* T_σ *transition samples* (s,a,s'), *then with probability at most* $\frac{\delta}{3}$, *the estimated covariance parameter* $\tilde{\sigma}_{ij}$, *computed by Equation 3, deviates from the true covariance parameter* σ_{ij} *by more than* ε *over all entries* ij; *formally,* $\Pr(\max_i |\tilde{\sigma}_{ij} - \sigma_{ij}| \geq \varepsilon) \leq \frac{\delta}{3}$.

Proof First recall that σ_{ij} represents the covariance between dimensions i and j. We are interested in the probability that the estimated covariance $\tilde{\sigma}_{ij}$ differs from the true parameter σ_{ij}: $\Pr(|\tilde{\sigma_{ij}} - \sigma_{ij}| \geq \varepsilon)$. From Chebyshev's inequality, we can bound this expression as

$$\Pr(|\tilde{\sigma_{ij}} - \sigma_{ij}| \geq \varepsilon) \quad \leq \quad \frac{Var(\tilde{\sigma}_{ij})}{\varepsilon^2}, \tag{11}$$

where $Var(\tilde{\sigma}_{ij})$ is the variance of the sample variance.

We therefore require an upper bound on the variance of the sample covariance. We will derive a bound on this below in the general case of the covariance between two variables x and y both of which are Gaussian distributed.

$$\begin{aligned} Var(\tilde{\sigma}_{xy}) &= E[(\tilde{\sigma}_{xy} - \sigma_{xy})^2] \\ &= E\left[\left(\frac{1}{T_\sigma}\sum_{k=1}^{T_\sigma}(x_k - \bar{x})(y_k - \bar{y}) - \sigma_{xy}\right)^2\right] \end{aligned}$$

where $\bar{x}$ and $\bar{y}$ are the respective sample means, and in the second line we have written out the definition of the sample covariance. We can then use the linearity of expectation to derive

$$\begin{aligned} Var(\tilde{\sigma}_{xy}) &= \frac{1}{T_\sigma^2}\sum_{k=1}^{T_\sigma}\sum_{m=1}^{T_\sigma} E[(x_k - \bar{x})(x_m - \bar{x})(y_k - \bar{y})(y_m - \bar{y})] \\ &\quad -2\sigma_{xy}\frac{1}{T_\sigma}\sum_{k=1}^{T_\sigma} E[(x_m - \bar{x})(y_m - \bar{y})] + E[(\sigma_{xy})^2] \\ &= \frac{1}{T_\sigma^2}\sum_{k=1}^{T_\sigma}\sum_{m=1}^{T_\sigma} E[(x_k - \bar{x})(x_m - \bar{x})(y_k - \bar{y})(y_m - \bar{y})] - (\sigma_{xy})^2 \end{aligned}$$

where the second line follows from the definition of the covariance σ_{xy}. We next divide the summation into two expressions, when $m = k$ and when $m \neq k$, and use the property that the expectation of independent variables is the product of their expectations:

$$\begin{aligned} Var(\tilde{\sigma}_{xy}) &= \frac{1}{T_\sigma} E[(x_k - \bar{x})^2 (y_k - \bar{y})^2] + \frac{T_\sigma (T_\sigma - 1)}{T_\sigma^2} E[(x_k - \bar{x})(y_k - \mu_k)] E[(x_m - \bar{x})(y_m - \bar{y})] - (\sigma_{xy})^2 \\ &= \frac{1}{T_\sigma} E[(x_k - \bar{x})^2 (y_k - \bar{y})^2] + \frac{T_\sigma (T_\sigma - 1)}{T_\sigma^2} (\sigma_{xy})^2 - (\sigma_{xy})^2. \end{aligned}$$

We can now use the Cauchy-Schwarz inequality on the first term to get

$$\begin{aligned} Var(\tilde{\sigma}_{xy}) &\leq \frac{1}{T_\sigma} \sqrt{E[(x_k - \bar{x})^4] E[(y_k - \bar{y})^4]} + \frac{T_\sigma (T_\sigma - 1)}{T_\sigma^2} (\sigma_{xy})^2 - (\sigma_{xy})^2 \\ &= \frac{1}{T_\sigma} \sqrt{E[(x_k + \mu_x - \mu_x - \bar{x})^4] E[(y_k + \mu_y - \mu_y - \bar{y})^4]} + \frac{T_\sigma (T_\sigma - 1)}{T_\sigma^2} (\sigma_{xy}^2)^2 - (\sigma_{xy}^2)^2 \\ &= \frac{1}{T_\sigma} \sqrt{(3\sigma_{xx}^4 + 6\sigma_{xx}^2 (\bar{x} - \mu_x)^2 + (\bar{x} - \mu_x)^4)(3\sigma_{yy}^4 + 6\sigma_{yy}^2 (\bar{y} - \mu_y)^2 + (\bar{y} - \mu_y)^4)} \\ &\quad + \frac{T_\sigma (T_\sigma - 1)}{T_\sigma^2} (\sigma_{xy})^2 - (\sigma_{xy})^2 \end{aligned}$$

where we have used the fact that the fourth central moment of a Gaussian distribution is $3\sigma_{xx}^2$ in the final line. Next we make use of the assumptions that B_σ is an upper bound to all covariance matrix elements and the bound on the maximum error in the parameter offset estimates:

$$\begin{aligned} Var(\tilde{\sigma}_{xy}^2) &\leq \frac{(\varepsilon^4 + 6\varepsilon^2 B_\sigma + 3B_\sigma^2)}{T_\sigma} + \frac{T_\sigma (T_\sigma - 1)}{T_\sigma^2} (\sigma_{xy})^2 - (\sigma_{xy})^2 \\ &\leq \frac{4B_\sigma^2}{T_\sigma} \end{aligned}$$

where the last line follows because $\varepsilon < 1/4$ and $B_\sigma \geq 1$. We can then substitute this result into Equation 11 which yields

$$P(|\tilde{\sigma}_{ij} - \sigma_{ij}| \geq \varepsilon) \leq \frac{4B_\sigma^2}{\varepsilon^2 T_\sigma}.$$

To ensure that this bound holds simultaneously with probability $\frac{\delta}{3}$ for all N^2 covariance matrix elements it suffices by the union bound to require that each covariance entry exceeds its expected value by more than ε with probability at most $\frac{\delta}{3N^2}$:

$$\frac{4B_\sigma^2}{\varepsilon^2 T_\sigma} \leq \frac{\delta}{3N^2}.$$

Re-arranging yields the bound for the required number of samples:

$$T_\sigma \geq \frac{12N^2 B_\sigma^2}{\delta \varepsilon^2}.$$

■

Lemma 8 *If* $\max_{i,j}|\Sigma_1(i,j)-\Sigma_2(i,j)| \le \varepsilon$ *for any* $1 \le i,j \le N$, *then*

$$\left|\ln\frac{\det\Sigma_2}{\det\Sigma_1}\right| \le N\varepsilon\left(\frac{1}{\lambda_1}+\frac{1}{\lambda_2}+\cdots+\frac{1}{\lambda_N}\right) \le \frac{N^2\varepsilon}{\lambda_N}.$$

Proof Define $E = \Sigma_2 - \Sigma_1$. Clearly, E is symmetric since both Σ_1 and Σ_2 are symmetric. Its eigenvalues are denoted by $\psi_1 \ge \psi_2 \ge \cdots \ge \psi_N$, which are real (but can be negative or positive). First, it is known that

$$\det\Sigma_1 = \prod_{i=1}^{N}\lambda_i \quad \& \quad \det\Sigma_2 = \prod_{i=1}^{N}\lambda_i'.$$

Therefore,

$$\ln\frac{\det\Sigma_2}{\det\Sigma_1} = \ln\prod_{i=1}^{N}\frac{\lambda_i'}{\lambda_i} = \sum_{i=1}^{N}\ln\frac{\lambda_i'}{\lambda_i}.$$

From Geršgorin's theorem (Horn and Johnson, 1986, Theorem 6.1.1), the eigenvalues of E must be small as the elements of E are small. Specifically, each ψ_i must lie in one of the n Geršgorin discs:

$$\forall 1 \le j \le N : D_j = \{x \in R \mid |x - E(j,j)| \le \sum_{j' \ne j}\left|E(j,j')\right|\}.$$

It follows immediately that

$$|\psi_i| \le \sum_{j=1}^{N}|E(i,j)| \le N\varepsilon$$

as every component in E lies in $[-\varepsilon,\varepsilon]$.

On the other hand, from Weyl's theorem (Horn and Johnson, 1986, Theorem 4.3.1), we have

$$\psi_1 \ge \lambda_i' - \lambda_i \ge \psi_N.$$

We have just proved that both $|\psi_1|$ and $|\psi_N|$ are at most $N\varepsilon$, and thus

$$\left|\lambda_i' - \lambda_i\right| \le N\varepsilon.$$

Consequently,

$$\frac{\lambda_i'}{\lambda_i} \le \frac{\lambda_i + N\varepsilon}{\lambda_i} = 1 + \frac{N\varepsilon}{\lambda_i}.$$

Therefore, we have

$$\ln\frac{\det\Sigma_2}{\det\Sigma_1} = \sum_{i=1}^{N}\ln\frac{\lambda_i'}{\lambda_i} \le \sum_{i=1}^{N}\ln\left(1+\frac{N\varepsilon}{\lambda_i}\right) \le \sum_{i=1}^{N}\frac{N\varepsilon}{\lambda_i} \le \frac{(N)^2\varepsilon}{\lambda_N}$$

where the second to last inequality uses the inequality $\ln(1+x) \le x$ for $x \ge 0$. ■

The following lemmas will be useful to prove Lemma 9.

Lemma 11 ***(Lemma 2.7.1 and Theorem 2.7.2 from Golub and Van Loan 1996)*** *Suppose $Ax = b$ and $(A+\Delta A)y = b+\Delta b$ with $\|\Delta A\| \leq \varepsilon\|A\|$ and $\|\Delta b\| \leq \varepsilon\|b\|$. If $\varepsilon\kappa(A) < 1$, then $A+\Delta A$ is nonsingular, and*

$$\frac{\|y-x\|}{\|x\|} \leq \frac{2\varepsilon\kappa(A)}{1-\varepsilon\kappa(A)},$$

where $\|\cdot\|$ can be any ℓ_p matrix/vector norm, and $\kappa(A) = \|A\|\left\|A^{-1}\right\|$ is the corresponding condition number.

Lemma 12 ***(A trace inequality of von Neumann 1937)*** *Let A and B be two symmetric matrices of order n, whose singular values are $\xi_1 \geq \xi_2 \geq \cdots \geq \xi_n \geq 0$ and $\zeta_1 \geq \zeta_2 \geq \cdots \geq \zeta_n \geq 0$, respectively. Then*

$$|\mathrm{tr}\,(AB)| \leq \sum_{i=1}^{n} \xi_i\zeta_i.$$

Lemma 13 *Suppose the covariance matrix Σ_1 is non-singular; that is its eigenvalues $\lambda_1 : \lambda_N > 0$. Then*

$$\begin{aligned}
\mathrm{tr}\left(\Sigma_1^{-1}\right) &= \sum_{i=1}^{N} \frac{1}{\lambda_i} \leq \frac{N}{\lambda_N} \\
\max_{ij} |\Sigma_1^{-1}(i,j)| &\leq ||\Sigma_1^{-1}||_1 \\
||\Sigma_1^{-1}||_1 \leq \sqrt{N}||\Sigma_1^{-1}||_2 &= \frac{\sqrt{N}}{\lambda_N}.
\end{aligned}$$

Proof We prove the three upper bounds one by one:

1. It is a known fact that the trace of a matrix equals the sum of its eigenvalues. The first equality follows from the observation that the eigenvalues of Σ_1^{-1} are $\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \ldots, \frac{1}{\lambda_N}$.

2. This inequality follows from the definition of $||\Sigma_1^{-1}||_1$: it is the maximum absolute row sum of the matrix Σ_1^{-1}, and therefore is not less than the largest absolute component of the matrix.

3. It is known that $||A||_1 \leq \sqrt{N}||A||_2$ for any $N \times N$ matrix A (see, eg. theorem 5.6.18 in Horn and Johnson 1986). On the other hand, $||\Sigma_1^{-1}||_2$ equals the largest eigenvalue of Σ_1^{-1}, which is $\frac{1}{\lambda_N}$.

■

Lemma 9 *If $\max_{i,j} |\Sigma_1(i,j) - \Sigma_2(i,j)| \leq \varepsilon$ and $N\varepsilon\left\|\Sigma_1^{-1}\right\|_1 < 1$, then*

$$\mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) - N \leq \frac{2N^3\varepsilon B_\sigma}{\lambda_N^2 - (N)^{1.5}\lambda_N\varepsilon}.$$

Proof The i-th row (or column) of Σ_1^{-1} is the solution to the system of linear equations: $\Sigma_1 x = e_i$ where e_i has $N-1$ zero components except a 1 in the i-th component. Similarly, the i-th row (or

column) of Σ_2^{-1} is the solution to $\Sigma_2 y = e_i$. Since Σ_1 and Σ_2 differ by at most ε in every component, we have

$$\frac{\|\Sigma_1 - \Sigma_2\|_1}{\|\Sigma_1\|_1} \leq \frac{N\varepsilon}{\|\Sigma_1\|_1}.$$

For convenience, denote the right-hand side above by ε'. It follows from Lemma 11 that

$$\|x - y\|_1 \leq \frac{2\varepsilon'\kappa(\Sigma_1)\|x\|_1}{1 - \varepsilon'\kappa(\Sigma_1)}.$$

The above inequality holds for all N possible e values. Note that $\|x-y\|_1$ is the absolute sum of the i-th row (or column) of $\Sigma_1^{-1} - \Sigma_2^{-1}$ for e_i. Let $\psi_1 \geq \psi_2 \geq \cdots \geq \psi_N \geq 0$ be the singular values of $\Sigma_1^{-1} - \Sigma_2^{-1}$. From Geršgorin's theorem, it follows that for all i,

$$\psi_i \leq \max_e \|x-y\|_1 \leq \frac{2\varepsilon'\kappa(\Sigma_1)}{1-\varepsilon'\kappa(\Sigma_1)} \max_e \|x\|_1 = \frac{2\varepsilon'\kappa(\Sigma_1)}{1-\varepsilon'\kappa(\Sigma_1)} \left\|\Sigma_1^{-1}\right\|_1 \tag{12}$$

where $\kappa(\Sigma_1) = \|\Sigma_1\| \left\|\Sigma_1^{-1}\right\|$ the condition number of Σ_1. We can now complete the proof:

$$\begin{aligned}
\operatorname{tr}\left(\Sigma_2^{-1}\Sigma_1\right) - N &= \operatorname{tr}\left((\Sigma_2^{-1} - \Sigma_1^{-1})\Sigma_1\right) && (13)\\
&\leq \sum_{i=1}^{N} \psi_i \lambda_i && (14)\\
&\leq \frac{2\varepsilon'\kappa(\Sigma_1)\left\|\Sigma_1^{-1}\right\|_1}{1-\varepsilon'\kappa(\Sigma_1)} \sum_{i=1}^{N} \lambda_i && (15)\\
&= \frac{2\varepsilon'\kappa(\Sigma_1)\left\|\Sigma_1^{-1}\right\|_1}{1-\varepsilon'\kappa(\Sigma_1)} \operatorname{tr}(\Sigma_1) && (16)\\
&= \frac{2N\varepsilon\left\|\Sigma_1^{-1}\right\|_1^2}{1-N\varepsilon\left\|\Sigma_1^{-1}\right\|_1} \operatorname{tr}(\Sigma_1) && (17)\\
&= \frac{2N^2 B_\sigma \varepsilon\left\|\Sigma_1^{-1}\right\|_1^2}{1-N\varepsilon\left\|\Sigma_1^{-1}\right\|_1}, && (18)\\
&\leq \frac{2N^3\varepsilon B_\sigma}{\lambda_N^2 - (N)^{1.5}\lambda_N\varepsilon}. && (19)
\end{aligned}$$

The first equality (Equation 13) is due to the identity $\operatorname{tr}\left(\Sigma_1^{-1}\Sigma_1\right) = \operatorname{tr}(I) = N$, and the first inequality (Equation 14) is a direct application of von Neumann's inequality (Lemma 12) which can be used since the eigenvalues λ_i are also the singular values in this case. The second inequality (Equation 15) follows from the result of Equation 12, the second equality (Equation 16) follows by the definition of matrix traces, and the third equality (Equation 17) is obtained by noting that $\kappa(\Sigma_1) = \|\Sigma_1\|_1 \left\|\Sigma_1^{-1}\right\|_1$. Since each term in the covariance matrix is known to be bounded by B_σ then the trace is bounded by NB_σ which allows us to generate the fourth equality (Equation 18). The final result is obtained using the result of Lemma 13.

■

Lemma 14 ***(Theorem from Kullback 1967)*** *Let p_1 and p_2 be two probability density functions defined over $\mathcal{X}$. Define*

$$\Omega = \{x \in \mathcal{X} \mid p_1(x) \geq p_2(x)\}.$$

If p_1 and p_2 are both measurable (integrable) over Ω, then

$$\mathrm{d}_{\mathrm{KL}}(p_1 \,\|\, p_2) \geq \frac{1}{8} \|p_1 - p_2\|_1^2.$$

References

Pieter Abbeel and Andrew Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 1–8, 2005.

Justin Boyan and Andrew Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems (NIPS) 7*, pages 369–376, 1995.

Ronen I. Brafman and Moshe Tennenholtz. R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

Emma Brunskill, Bethany R. Leffler, Lihong Li, Michael L. Littman, and Nicholas Roy. CORL: A continuous-state offset-dynamics reinforcement learner. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 53–61, 2008.

Jeffrey B. Burl. *Linear Optimal Control*. Prentice Hall, 1998. ISBN 9780201808681.

Pablo Castro and Doina Precup. Using linear programming for Bayesian exploration in Markov decision processes. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2437–2442, 2007.

Chee-Seng Chow and John N. Tsitsiklis. The complexity of dynamic programming. *Journal of Complexity*, 5(4):466–488, 1989.

Chee-Seng Chow and John N. Tsitsiklis. An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE Transactions on Automatic Control*, 36(8):898–914, 1991.

Finale Doshi, Joelle Pineau, and Nicholas Roy. Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 256–263, 2008.

Alain Dutech, Timothy Edmunds, Jelle Kok, Michail Lagoudakis, Michael L. Littman, Martin Riedmiller, Bryan Russell, Bruno Scherrer, Richard Sutton, Stephan Timmer, Nikos Vlassis, Adam White, and Shimon Whiteson. Reinforcement learning benchmarks and bake-offs II. In *Advances in Neural Information Processing Systems (NIPS) 17 Workshop*, 2005.

Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: A WiFi-Based Vehicular Content Delivery Network. In *Proceedings of the 14th Conference on Mobile Computing and Networking (MOBICOM)*, pages 199–210, 2008.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996. ISBN 0-801-85414-8.

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1986. ISBN 0-521-38632-2.

Eric Horvitz, Johnson Apacible, Raman Sarin, and Lin Liao. Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 275–283, 2005.

Nicholas K. Jong and P. Stone. Model-based exploration in continuous state spaces. In *Proceedings of the 7th Symposium on Abstraction, Reformulation, and Approximation (SARA)*, pages 258–272, 2007.

Sham Kakade. *On the sample complexity of reinforcement learning*. PhD thesis, University College London, 2003.

Michael J. Kearns and Satinder P. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2–3):209–232, 2002.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Proceedings of the 17th European Conference on Machine Learning (ECML)*, pages 282–293, 2006.

Solomon Kullback. A lower bound for discrimination in terms of variation. *IEEE Transactions on Information Theory*, 13(1):126–127, January 1967.

Branislav Kveton and Milos Hauskrecht. Solving factored MDPs with exponential-family transition models. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 114–120, 2006.

M.G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

Bethany R. Leffler, Michael L. Littman, and Timothy Edmunds. Efficient reinforcement learning with relocatable action models. In *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI)*, pages 572–577, 2007.

Lihong Li. *A Unifying Framework for Computational Reinforcement Learning Theory*. PhD thesis, Rutgers University, New Brunswick, NJ, 2009.

Lihong Li, Michael L. Littman, and Thomas J. Walsh. Knows what it knows: A framework for self-aware learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 568–575, 2008.

Janusz Marecki and Milind Tambe. Towards faster planning with continuous resources in stochastic domains. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*, pages 1049–1055, 2008.

Andrew Ng, H.Jin Kim, Michael Jordan, and Shankar Sastry. Autonomous helicopter flight via reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS) 16*, pages 799–806, 2004.

Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 697–704, 2006.

Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayesian reinforcement learning in continuous POMDPs with application to robot navigation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 2845–2851, 2008.

Alexander L. Strehl and Michael L. Littman. Online linear regression and its application to model-based reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS) 20*, pages 1417–1424, 2008.

Alexander L. Strehl, Lihong Li, and Michael L. Littman. Incremental model-based learners with formal learning-time guarantees. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 485–492, 2006.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

Gerald J. Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

John von Neumann. Some matrix-inequalities and metrization of matrix-space. *Tomsk University Review*, 1:286–300, 1937.

Christopher J.C.H. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, University of Cambridge, United Kingdom, 1989.