Online Feature Selection for Model-based Reinforcement Learning

Trung Thanh Nguyen NTTRUNG@COMP.NUS.EDU.SG
Zhuoru Li LIZHUORU@COMP.NUS.EDU.SG
Tomi Silander SILANDER@COMP.NUS.EDU.SG
Tze-Yun Leong LEONGTY@COMP.NUS.EDU.SG
School of Computing, National University of Singapore, Singapore, 117417

Abstract

We propose a new framework for learning the world dynamics of feature-rich environments in model-based reinforcement learning. The main idea is formalized as a new, factored state-transition representation that supports efficient online-learning of the relevant features. We construct the transition models through predicting how the actions change the world. We introduce an online sparse coding learning technique for feature selection in high-dimensional spaces. We derive theoretical guarantees for our framework and empirically demonstrate its practicality in both simulated and real robotics domains.

1. Introduction

In model-based reinforcement learning (RL), factored state representations, often in the form of dynamic Bayesian networks (DBNs), are deployed to exploit structures of the world dynamics. This allows the agent to plan and act in large state spaces without actually visiting every state. However, learning the world dynamics of a complex environment is very difficult and often computationally infeasible. Most recent work in this area is based on the RMAX framework (Brafman & Tennenholtz, 2003), and focuses on sample-efficient learning of the optimal policies. This approach incurs heavy computational costs for maximizing information gain from every interaction, even in carefully designed, low-dimensional spaces.

We propose a variant formulation of the factored Markov decision process (MDP) that incorporates a principled way to compactly factorize the state space, while capturing comprehensive transition and reward dynamics information. We also propose an online multinomial logistic regression method with group lasso to automatically learn the relevant structure of the world dynamics model. While the regression models cannot capture the full conditional distributions like DBNs, their simplicity allows fast, online learning in very high dimensional spaces. Online feature selection is implemented with operating the regression algorithm in our variant MDP formulation.

In the rest of the paper, we will first introduce an MDP representation that captures the world dynamics as action effects. We then present an online learning algorithm for identifying relevant features via sparse coding, and show that in theory our framework should lead to computationally efficient learning of a near optimal policy. Due to the space limit, full proofs are placed in the supplementary materials. To back up the theoretical claims, we conduct experiments in both simulated and real robotics domains.

2. Method

In RL, a task is typically modelled as an MDP defined by a tuple (S, A, T, R, γ), where S is a set of states; A is a set of actions; $T : S \times A \times S \to [0, 1]$ is a transition function, such that $T(s, a, s') = P(s'|s, a)$ specifies the probability of transiting to state s' upon taking an action a at state s; R is a reward function indicating immediate expected reward after the state transition $s \xrightarrow{a} s'$; and future reward that occurs t time steps in future is discounted by γ^t. The agent's goal is to learn a policy π that specifies an action to perform at each state s, so that the expected discounted, cumulative future reward starting from s is maximized. In model-based RL, the optimal policy is estimated from the transition model T and the reward model R. In this paper we concentrate on learning the transition model, with a known reward model. The reward model, however, can be learned in a similar way.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

In a factored MDP, each state is represented by a vector of n state-attributes. The transition function for the factored states is commonly expressed using dynamic Bayesian networks (DBNs) in which $T(s,a,s') = \prod_{i=1}^{n} P(s'_i|Pa_i^a(s),a)$, where Pa_i^a indicates a subset of state-attributes in s called the parents of s'_i (Fig.1a). Learning T requires learning the subsets Pa_i^a and the parameters for conditional distributions, or the DBN local structures in other words.

Learning DBN structures of the transition function online, i.e., while the agent is interacting with the environment, however, is computationally prohibitive in most domains. On the other hand, recent studies (Xiao, 2009; Yang et al., 2010) have shown encouraging results in learning the structure of logistic regression models, which can effectively serve as local structures in the DBNs even in high dimensional spaces. While these regression models cannot fully capture the conditional distributions, their expressive power can be improved by augmenting low dimensional state representation with non-linear features of the state vectors. We introduce an online sparse multinomial logistic regression method that supports efficient learning of the structured representation of the transition function.

2.1. CMDP - a factored MDP with feature-variables and action effects

We present a variant of the factored MDP that defines a "compact but comprehensive" factorization of the transition function and supports efficient learning of the relevant features. We consider two major approaches to modeling world dynamics: predicting changes and differentiating features.

First, we predict the relative changes of states instead of directly specifying the next states in a transition. Mediating state changes via action effects is a common strategy in situation calculus (McCarthy, 1963). Since the number of relative changes or action effects is usually much smaller than the size of the state space, the corresponding prediction task should be easier. The learning problem can then be expressed as a multi-class classification task of predicting the action effects.

Second, we differentiate the roles of attributes or features that characterize a state. In a regular factored MDP, the state-attributes or features serve to both define the state space and capture information about the transition model. For example, two state-attributes, the (x,y)-coordinates uniquely identify a state and compactly factorize the state space in a grid-world. A policy can be learned on this factored space. The transition dynamics or action effects, however, may depend on other features of the state, such as the surface material at the location (state). Such features are often carefully included in the state representations. While essential in formulating the transition or reward models, these features may complicate the planning or learning processes by increasing the size and complexity of the state space.

We separate the state identifying state-attributes from the "merely" informative state-features in our representation. This way, we can apply an efficient feature selection method on a large number of state features to capture the transition dynamics, while maintaining a compact state space.

More formally, a *situation Calculus MDP* (CMDP) is defined by a tuple $(S, f, A, T, E, R, \gamma)$, where S, A, T, R, γ have the same meaning as in regular MDP. $S = \langle S_1, S_2, .., S_n \rangle$ is the state space implicitly represented by vectors of n *state-attributes*. The function $f : S \to \mathbb{R}^m$ extracts m *state-features* from each state. E is an *action effect* variable such that the transition function can be factored as

$$\begin{aligned} T(s,a,s') &= P(s' \mid s,a) = \prod_{i=1}^{n} P(s'_i \mid s, f(s), a) \\ &= \prod_{i=1}^{n} \sum_{e \in E} P(s'_i \mid e, s) P(e \mid s, f(s), a). \end{aligned}$$

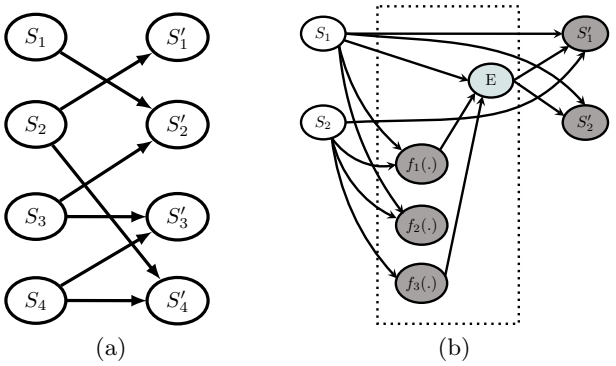
Figure 1b shows an example of this decomposition. The agent uses the feature function f to identify the relevant features, and then use both state attributes and features to predict the action effects. We also assume that the effect e and current state s determine the next state s', thus $P(s'|e,s)$ is either 0 or 1. This defines the semantic meaning of the effect which is assumed to be known by the agent. The remaining task is to learn the $P(e|s,a) = P(e|x(s),a)$, where $x(s) = (s, f(s))$, which is a classification problem; we solve this problem using multinomial logistic regression methods.

2.2. Online multinomial logistic regression with group lasso

We introduce a regularized online multinomial regression method with group lasso that allows us to learn a probabilistic multi-class classifier with online feature selection. We also show that the structure and parameters of the learnt classifier are likely to converge to those of the optimal classifier.

2.2.1. Multinomial logistic regression

Multinomial logistic regression is a simple yet effective classification method. Assuming K classes of d-



(a) (b)

Figure 1. a.) Standard DBN. b.) Our customized DBN for CMDP.

dimensional vectors $x \in \mathbb{R}^d$, we represent each class k with a d-dimensional prototype vector W_k. Classification of an input vector x is based on how "similar" it is to the prototype vectors. Similarity is measured with inner product $\langle W_k, x\rangle = \sum_{i=1}^{d} W_{ki}x_i$, where x_i denotes feature i. The log probability of a class is defined by $\log P(y = k|x; W_k) \propto \langle W_k, x\rangle$. The parameter vectors of the model form the rows of a matrix $W = (W_1, ..., W_K)^T$.

Let $l_t(W^t) = -\log P(y^t|x^t; W^t)$ denote the item-wise log-loss of a model with coefficient matrix W^t predicting a data point (y^t, x^t) observed at time t. A typical objective of an online learning system is to minimize the total loss by updating its W^t over time. However, the resulting model will often be very complicated and over-fitting. To achieve a parsimonious model, we express our a priori belief that most features are irrelevant or superfluous by introducing a regularization term $\Psi(W) = \lambda \sum_i^d \sqrt{K}||W_{\cdot i}||_2$, where $||W_{\cdot i}||_2$ denotes the 2-norm of the i^{th} column of W, and λ is a positive constant. This regularization is similar to that of group lasso. It communicates the idea that it is likely that a whole column of W has zero values (especially, for large λ). A column of all zeros suggests that the corresponding feature is not necessary for classification.

The objective function can now be written as

$$L(T) = \sum_{t=1}^{T} l_t(W^t) + \Psi(W^t)$$
$$= \sum_{t=1}^{T} -\log \frac{e^{\left\langle W^t_{y^t}, x^t\right\rangle}}{\sum_k e^{\left\langle W^t_k, x^t\right\rangle}} + \lambda \sum_i^d \sqrt{K}||W^t_{\cdot i}||_2,$$

where W^t is the coefficient matrix learned using $t-1$ previously observed data items. The quality of a sequence of parameter matrices $W^t, t \in (1, \ldots, T)$ with respect to a fixed parameter matrix W can be measured by the amount of extra loss, or regret

$$R_T(W) = L(T) - L_W(T)$$
$$= \sum_{t=1}^{T} (l_t(W^t) + \Psi(W^t)) - \sum_{t=1}^{T} (l_t(W) + \Psi(W))$$

We want to learn a series of parameters W^t to achieve small regret with respect to a good model W that has a small loss $L_W(T)$.

2.2.2. Online learning for regularized multinomial logistic regression

We introduce an update function, *mDAGL-update* (Algorithm 1) to extend the efficient dual averaging method (Xiao, 2009) for solving lasso and group lasso (Yang et al., 2010) logistic regression on binary classification to the multi-class case.

Let $h(W)$ be a strongly convex function with modulus 1; $W^0 = \arg\min_W h(W)$, and let $W^{t=1}$ be initialized to W^0. Let G^t_{ki} be the partial derivatives of function $l_t(W)$ with respect to W_{ki} at W^t ($G^t_{ki} = \frac{\partial l_t}{\partial W_{ki}}(W^t)$). We define $\bar{G}^t$ to be a matrix of average partial derivatives, i.e., $\bar{G}^t_{ki} = \frac{1}{t}\sum_{\tau=1}^{t} G^\tau_{ki}$, where

$$G^\tau_{ki} = -x^\tau_i (I(y^\tau = k) - P(k|x^\tau; W^\tau)). \quad (1)$$

For any data observed at time t, we update the coefficient matrix via

$$W^{t+1} = \arg\min_W \left(\left\langle \bar{G}^t, W \right\rangle + \Psi(W) + \frac{\beta_t}{t} h(W) \right), \quad (2)$$

where β_t is a non-negative, non-decreasing constant sequence, and $\langle \cdot, \cdot \rangle$ denotes an inner product between two matrices; $\left\langle \bar{G}^t, W \right\rangle = \sum_{k,i} \bar{G}^t_{ki} W_{ki}$.

Theorem 1 (Update Rule) *Given* $h(W) = \frac{1}{2}||W||_2^2$, *a* $K \times d$ *average gradient matrix* $\bar{G}^t$, *and a regularization parameter* $\lambda > 0$, *the optimal solution of (2) is achieved column-wise as follows*

$$W^{t+1}_{\cdot i} = \begin{cases} \overrightarrow{0} & \text{if} \quad ||\bar{G}^t_{\cdot i}||_2 \le \lambda\sqrt{K}, \\ \frac{t}{\beta_t}\left(\frac{\lambda\sqrt{K}}{||G^t_{\cdot i}||_2} - 1\right)\bar{G}^t_{\cdot i} & \text{otherwise.} \end{cases} \quad (3)$$

This rule dictates that when the length of the average gradient matrix column is small enough, the corresponding parameter column should be truncated to zero. This amounts to feature selection.

The following regret analysis confirms that the solution will converge and that the average maximal regret asymptotically approaches zero with rate $O(\frac{1}{\sqrt{t}})$.

Algorithm 1 The mDAGL update

Input: $t, y^t, x^t, W^t, \bar{G}^{t-1}, \lambda, \alpha$
$G^t \leftarrow$ use equation 1 with $(y^t, x^t), W^t$
$\bar{G}^t \leftarrow \frac{t-1}{t}\bar{G}^{t-1} + \frac{1}{t}G^t$
$W^{t+1} \leftarrow$ use equation 3 with $\bar{G}^t, \beta_t = \alpha\sqrt{t}, \lambda$
return $(W^{t+1}, \bar{G}^t)$

Theorem 2 (Regret Bound) *Let the sequence of* $\{W^t\}_{t\geq 1}$ *be generated by the update rule (3), and assume that there exists a constant* G *such that* $||\bar{G}^t||_2^2 \leq G^2, \forall t \geq 1$. *If we choose* $\beta_t = \alpha\sqrt{t}$ *where* $\alpha > 0$, *then for any* $t \geq 1$ *and for any* W *that satisfies* $h(W) \leq D^2$ *where* D *is a constant, the average regret is bounded as*

$$\frac{R_t(W)}{t} \leq \frac{\Delta}{\sqrt{t}}, \quad t = 1, 2, 3.., \tag{4}$$

where $\Delta = \left(\alpha D^2 + \frac{G^2}{\alpha}\right)$.

[**Proof Sketch**] The item-wise loss function $l_t(W)$ of multinomial logistic regression is convex, thus the techniques used for binary case (Xiao, 2009) can be applied for multinomial case as well.

Since the average regret goes asymptotically to zero, it may look very feasible that the sequence (W^t) also converges to some optimal W^*. However, the regret analysis is valid for any sequence of data, and without additional assumptions about the data generating process there may not be any asymptotically optimal classifier W^*, thus convergence is not meaningful. To study convergence, we assume the data is to be sampled independently from some joint distribution p for data vector (y,x). In this case we try to find a W that minimizes the expected loss $E_p[l(W)] + \Psi(W)$. Now assuming that the optimal solution W^* is sparse, and some other technical assumptions, it is indeed possible to show that

$$P(||W^t - W^*||_2 > \epsilon) < \left[\epsilon^{-1}(\epsilon^{-1} + r^{-1}) + \frac{2}{c}\Delta\right] t^{-\frac{1}{4}}, \tag{5}$$

where r and c are constants (see Lemma 13 in (Lee & Wright, 2012) for the result and its assumptions).

2.3. Model-based RL with feature selection

Our main task is to turn transition model learning into the learning of conditional distributions $P(E \mid s, f(s), a)$ using multinomial logistic regression for which attention to relevant features can be efficiently implemented online via mDAGL.

The key steps of our method, called loreRL (RL with regularized logistic regression), are presented in Algorithm 2. Inputs to loreRL are the CMDP components (except the transition function), regularization parameters λ and α of mDAGL algorithm, and the ϵ that determines the probability of taking a random action. We first initialize logistic regression parameters W_a and the average gradient matrices $\bar{G}_a$ for each action $a \in A$. We also randomly select a starting state s^0.

Algorithm 2 The loreRL algorithm

Input: mDAGL regularization parameters λ, α, CMDP variables S, f, A, E, R, γ, exploration ϵ.
Let $W = (W_1, W_2, \ldots, W_{|A|}) = (W^0, W^0, \ldots, W^0)$
Let $\bar{G} = (\bar{G}_1, \bar{G}_2, \ldots, \bar{G}_{|A|}) = (\vec{0}, \vec{0}, \ldots, \vec{0})$
$s_0 \leftarrow$ random initial state
for $t = 1, 2, 3, ...$ **do**
  $\pi \leftarrow$ Solve MDP using transition model $T(\bar{W})$
  $a \leftarrow \pi(s^t, \epsilon)$ #ϵ-greedy action selection
  Take action a yielding effect e, next state s^{t+1}
  $(W_a, \bar{G}_a) \leftarrow mDAGL(t, e, x(s^t), W_a, \bar{G}_a, \lambda, \alpha)$
end for

At each time step, a random action a is chosen with a small probability ϵ, but otherwise we calculate the optimal policy π for an MDP with the transition model $T(W)$ is based on the current effect predictors. While we have used value iteration (like in *Rmax*) for finding the optimal policy, any other model-based RL technique can be used as well. We do not focus on the planning part of RL here, but Dyna-Q or Prioritized Sweeping can be deployed for a more scalable algorithm. After performing an action a in state s^t and observing its effect e, the experience $(e, s^t, f(s^t))$ will be presented to the mDAGL algorithm that updates the parameter matrix W_a and the gradient matrix $\bar{G}_a$.

As we just do ϵ-greedy random sampling, it is impossible to guarantee PAC convergence to an optimal policy. Assuming that observed data is i.i.d, we can prove that difference in optimal value functions of two CMDPs with different logistic regression based transition functions is bounded by the difference in their parameters. This leads to a corollary for convergence to near optimal policy.

Theorem 3 (Difference in Value Function) *Let* $M_1 = (S, f, A, T(W^{M_1}), E, R, \gamma)$ *and* $M_2 = (S, f, A, T(W^{M_2}), E, R, \gamma)$ *be two CMDPs with optimal policies* π_1 *and* π_2 *respectively. Let us denote by* V_π^M *the value function for policy* π *in CMDP* M. *Let*

$$\epsilon_1 = 2\sqrt{\max_{a\in A, e\in E} ||W_e^{(a),M_1} - W_e^{(a),M_2}||_1 \sup_s ||x(s)||_1},$$

then $\max_{s \in S} \left(V_{\pi_1}^{M_2}(s) - V_{\pi_2}^{M_2}(s) \right) \leq \frac{2\gamma V_{\max} \epsilon_1}{1-\gamma}$,

where $W_e^{(a),M_1}$ *and* $W_e^{(a),M_2}$ *refer to the vector of coefficients corresponding to class* $E = e$ *under action* a *in model* M_1 *and* M_2 *respectively,* $||\cdot||_1$ *is the 1-norm of vector, and* V_{max} *is the maximum value of any state for any policy in either of the CMDPs.*

By taking M_2 to be an CMDP based on the optimal W^* and M_1 an estimated CMDP based on mDAGL, the vanishing bound given in equation (5) can be translated into a vanishing bound for value difference of policies. In case the true transition model is representable by a sparse W^*, we would most probably converge to a near optimal policy.

When we cannot express the true transition dynamics as logistic regression based on available state features, it is hard to give guarantees of performance. However, we can still have some confidence in doing well. The logistic regression model P_l^* closest (in Kullback-Leibler distance) to the true model P_{true} (possibly not a logistic regression model) is the one[1] that has the smallest expected log-loss. While our optimality criterion is the expected regularized log-loss, we expect the regularized log-loss optimal model P_Ψ^* to be close to P_l^* thus almost as close to P_{true} as we can get. This relatively small KL-distance can be converted to relatively small distances in actual transition probabilities, which can then further be converted to a relatively small bound on value differences by the same arguments used in proving Theorem 3. Therefore, since our model would very likely converge close to P_Ψ^*, we can expect to do almost as well as P_Ψ^*.

3. Related work

DBN has been a popular choice for factoring and approximating transition models. In DBN learning feature selection is equivalent to picking the parents of the state variables from the previous time slice. Recent studies have led to improvements in sample complexity for learning optimal policy. Those studies assume maximum number of possible parents for a node (Strehl et al., 2007), (Diuk et al., 2009), or knowledge of a planning horizon that satisfies certain conditions (Chakraborty & Stone, 2011). However, the improvements in sample complexity are achieved at the expense of actual computational complexity since these methods have to search through a large number of parent sets. Hence, these methods appear feasible only in manually designed, low-dimensional state-spaces.

Instead of searching for an optimal model with a minimal number of samples at almost any cost, our method attempts to save costs from early on, and gradually improve the model acknowledging that the true model may actually be unattainable. In this spirit the structure learning study by Degris et al. (Degris et al., 2006) resembles our work, but they do not address online learning with large feature sets. Ross et al. (Ross & Pineau, 2008) have used Markov chain Monte Carlo to sample from all possible DBN structures. However, the Markov Chain used has a very long burn-in period and slow mixing time, making sampling computationally prohibitive in large problems.

Kroon and Whiteson (2009) present a feature selection method for learning state values. This method, however, assumes that the DBN structures of transition and reward models are known. Our work, on the other hand, does not make such assumptions.

Leffler et al. (2007) also suggest to predict relative changes in states, which corresponds to the action effects in our formulation. However, they manually select important features to aggregate information from similar states for action effect predicting. Our work focuses on learning those features automatically. Hester and Stone (2009; 2012) later employ Quinlan's C4.5 (Quinlan, 1993) to learn a decision tree for predicting relative changes of every state variable. This works better than the method by Degris et al. Despite adapting C4.5 for online learning, the method is still very slow as a costly tree induction procedure has to be repeated many times in a large feature space. In addition, all the data needs to be stored for the purpose, which is undesirable in some applications.

Strehl and Littman (2007), and Walsh et al. (2009) have also proposed an online linear regression method with L2-regularization to approximate the transition model in continuous MDP. L2, however, does not implement feature selection.

4. Experiments

We present empirical evaluation of *loreRL* on both simulated and real robotic domains. The experiments aim to demonstrate that *loreRL* can a) generalize and approximate the transition model to achieve fast convergence to near optimal policy, and b) with feature selection, perform well in complex, feature rich environments. We also want to see if theoretical promises derived under assumption of i.i.d sampling can be realized in practice. We compare accumulated rewards of *loreRL* with factored Rmax (*fRmax*), in which the net-

[1]Such model may not always exist since the parameter set is open. However, for our argument, any model with almost infimum distance to the true model will do.

work structures of transition models are known (Strehl et al., 2007), and with factored ϵ-greedy (*fEpsG*), in which the optimistic Rmax exploration of *fRmax* is replaced by ϵ-greedy strategy. We also compare our method with RL-DT (Hester & Stone, 2009) and LSE-Rmax (Chakraborty & Stone, 2011), which are the state of the art model-based RL algorithms for learning transition models.

4.1. Grid-world domain

In this domain, the agent tries to reach its goal in the grid-world world consuming as little energy as possible. Each cell in the world has one of five surface materials: sand, soil, water, brick, and fire; there may be walls between cells. Surface and walls are features that determine the stochastic dynamics of the world. In addition, to test the variable selection aspect, we attach hundreds of random binary features to the environment; it has to learn to focus on the relevant features to quickly achieve its goal. The agent can perform four actions (move up, down, left, right), which will lead it to one of the four states around it or leave it to its current state. Effects of the actions are captured in five outcomes (moved up, left, down, right, did not move). The states are defined by the (x,y)-coordinates of the agent. To perform an action, agent will spend 0.01 units of energy. It loses 1 unit if falling into a state of fire, but gains 1 unit when successfully reaching an exit door. A task ends when agent reaches a terminal state, i.e., any exit door or state with fire.

We generated the environment transition models from four random multinomial logistic distributions (one for each action); every different combination of cell surfaces and walls around the cell will lead to different transition dynamics at the cell. The probability of going through a wall is rounded to zero and the freed probability mass is evenly distributed to other effects. The agent's starting position is randomly picked in each episode.

We ran the experiments with *loreRL* having $\alpha = 1.5$, $\lambda = 0.05$, $\gamma = 0.95$, exploration $\epsilon = 0.05$, parameter $m = 10$ for *fRmax*, $m = 5$ for *Rmax* ($m = 5$ is small for *Rmax*, but increasing it did not yield better result), fixed $m = 10, \sigma = 0.99$ for *LSE-Rmax* (similar to the author's report). All results are averaged over 20 runs, and we report the 95% confidence intervals.

Generalization and convergence. We first show that when the feature space is small, *loreRL* performs as efficiently as the state of the art methods. *RL-DT* employs a decision tree to generalize transition dynamics knowledge over states, but it is implemented with ϵ-greedy exploration strategy. *LSE-Rmax* appears to be the best structure learning in ergodic factored MDPs (Chakraborty & Stone, 2011). *fRmax* and *fEpsG* have correct DBN structures provided by an oracle. All the methods are implemented with our customized DBN to utilize domain knowledge. *Rmax* is included as a reference to show the effect of knowledge generalization.

As seen in Figure 2a, *loreRL* can approximate the world dynamics using samples in all the states, thus it converges as fast as *fEpsG*, and *RL-DT* to near optimal policy. Although *fRmax* is provided with the correct DBN structure, its accumulated reward is lower due to aggressive exploration to find the optimal model. After exploration the policy is guaranteed to be near optimal, but it may still take a long time (or forever) to catch up with *loreRL*. While *LSE-Rmax* follows the *Rmax* scheme, it starts with a simple model and explores a bit less aggressively than *fRmax*, gaining some advantage in early episodes. However, *LSE-Rmax* appears to require much more data to choose a more complex model. Its accumulated reward drops below *fRmax* after 150 episodes, and the angle of the curve suggests that its DBN structure is still not correct. We did not run *LSE-Rmax* for more episodes, as the algorithm has very high computational demand (Table 1).

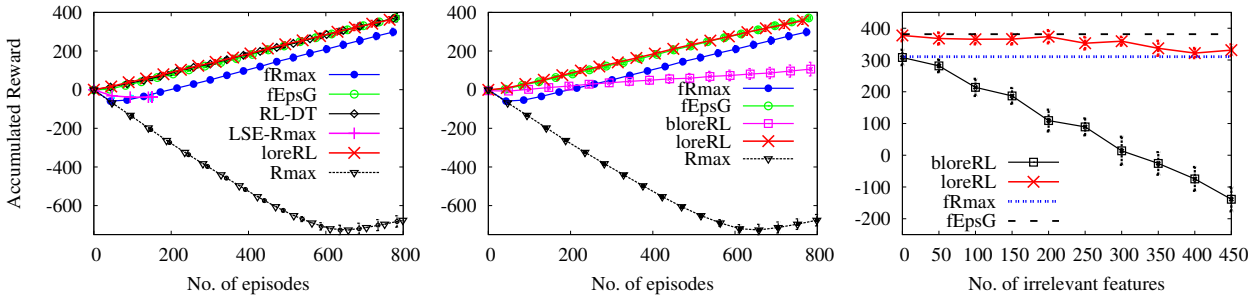
When the feature set has many irrelevant features (Figure 2b), *loreRL* is able to learn the relevant ones and still gain nearly as high accumulated reward as *fEpsG* which has relevant features provided by oracle. Also *loreRL*'s running time is not much longer than *fRmax*'s or *fEpsG*'s (Table 1). Other methods are too slow to be run in this high-dimensional environment.

These results also suggest that with ϵ-greedy exploration and random restarts, near optimal policy can be found even without i.i.d data sampling.

Table 1. Average running time per episode in 800 episodes when acting in an environment with 210 features. (Slow *RL-DT, LSE-Rmax* could only be run with 10 features.) Run on Intel Xeon CPU 2.13GHz, 32GB RAM.

Algorithm	fRmax	fEpsG	RL-DT	LSE-R.	bloreRL	**loreRL**
Time (sec.)	0.26	0.25	9.09	67.53	4.3	0.55

Feature selection. To model real-life situations, the feature space is usually exponentially large. The ability to focus only on the (most) relevant features is required to achieve effective learning. To understand the role of feature selection, we focus on comparing *loreRL* with a *bloreRL* that is based on multinomial logistic regression without feature selection (without the regularization term). *fEpsG* and *fRmax* are base lines.



(a) CMDP with 10 features. (b) Extra 200 irrelevant features. (c) Rewards after 800 episodes.

Figure 2. Accumulated rewards in a 900 state CMDP for various model-based RL-methods.

Figure 2b shows the accumulated reward when the environment has 200 irrelevant binary features. As seen, *loreRL* is still able to converge fast to optimal policy, and outperforms *fRmax* and *bloreRL*. Figure 2c shows performances of *loreRL* and *bloreRL* after 800 episodes as a function of the number of irrelevant features. Only minimally affected by the actual number of irrelevant features, *loreRL* can quickly select the relevant features and outperform *bloreRL*. *loreRL* does not lose much to *fEpsG* either. While *fRmax* may find an optimal policy before *loreRL* due to agressive exploration, its accumulated reward is still lower than *loreRL*'s. In our experiments, we also observed that *loreRL*, that selects a small set of features, is much faster than *bloreRL* (Table 1).

4.2. Robotics domain

Our second experiment is conducted in a real environment where we cannot expect the effect of the actions to follow logistic regression model. The domain (Figure 3) consists of a 5×5-feet environment made of various materials such as beans, soil, hay, leaves, and carpet that cause agent's actions, *move-forward*, *turn-left*, and *turn-right*, to have different effects at different locations. The environment also contains a blue target ball, and there are marks painted in green, blue, and red colors on cardboard surface. The three wheel robot was built with LEGO Mindstorms NXT kit and a camera was installed above the area so that the robot can fully observe the environment.

To learn the transition function for the robot, we discretized the environment into a state space of 8×8 (x,y)-locations and 8 different orientations of a robot, which yields a state space of 512 states. The actions may change robot's relative location in four different ways and orientation in five different ways resulting in total of 20 different effects. However, in different states the actions' tendency to produce effects may be different due to the differences in surface materials. To capture this variation, the agent describes each state with a long vector of binary features. The "green" binary indicator $f_i^G(s)$ of a state s is set to 1 iff there is a green mark that is further than i units but closer than $i+1$ units from the xy-center of the state s ($i \in \{0, \ldots, 99\}$). Similar features are defined for blue marks and to the blue target ball yielding 300 binary features. Eight indicators for different robot orientations are also included in the feature-base together with four intentionally redundant "there is/is-not a green/blue mark in the state grid"-bits. All together these yield 312 binary features per state. The intuition behind these features is that they serve as proxies to surface materials, slopes on the surfaces, obstacles, etc., which are likely to be important factors determining the dynamics in the environments, but which the robots sensors cannot capture. Although only few among these 312 features are important for modeling robots actions, the robot does not know those critical features. The robot has to learn to select them based on feedback while interacting with the environment.

The robot's task is to travel in the environment from a random start and reach the *blue ball*, which will earn it a reward of 2 points. The robot will receive -1 point if it falls out of the area or into the death places marked with orange rectangles, and -0.05 points for an action at any other states. An episode ends if the robot reaches a terminal state, or gets stuck for four consecutive actions.

The robot battery did not allow us to compare our algorithm with the slow *RL-DT* and *LSE-Rmax*; we could only compare with the fine-tuned *fRmax*, *fEpsG*, and *man-loreRL* algorithms in which we manually selected important features and specified the DBN-structures for the transition models. *man-loreRL* is based on multinomial logistic regression models with 12 manually selected features. We ran the experiments with *loreRL* and *man-loreRL* having $\alpha = 0.5$, $\lambda = 0.05$, $\gamma = 0.95$, exploration $\epsilon = 0.05$, parameter $m = 10$ for

fRmax. All results are averaged over 10 runs, and we report the 95% confidence intervals.

Figure 3. A real environment.

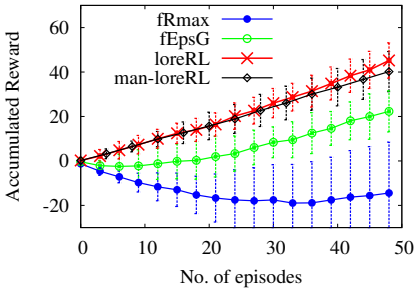


Figure 4. Accumulated rewards of various methods.

In Figure 4, *loreRL* appears to quickly capture the environment dynamics and outperform other methods. Even with manually selected features, *fRmax* and *fEpsG* require more exploration to learn the dynamics. *man-loreRL* gains rewards a bit faster, but in the end it slightly loses to *loreRL* possibly due to the (unforeseen) insufficiency of the manually selected features. Table 2 further shows that *loreRL* is fast. Its average running time per episode with 312-features is only slightly slower than with 12 manually selected features.

Table 2. Average running time per episode in 50 episodes. Run on Intel Centrino Duo T2400 (1.83GHz), 1GB RAM.

Algorithm	fRmax	fEpsG	man-loreRL	**loreRL**
Time (sec.)	134.37	127.69	93.54	108.10

5. Conclusions

We have demonstrated how online multinomial logistic regression with group lasso can be used to quickly obtain a parsimonious transition model in model based RL. The method leads to fast learning since a single transition model can be learnt using samples from all the states with a small set of features.

The efficiency is gained, however, at the expense of losing generality. Not all transition functions can be accurately represented as predicting action effects using state features via logistic regression. Nevertheless, we believe that this compromise between scalability and generality is often a useful one. The generality problem may also be alleviated by introducing non-linear features that are combinations of the original ones.

Other generalizations such as stochastic features and vector valued effects are also possible but are left for future work. The proposed framework also opens an opportunity for knowledge transfer. While different environments often have different states, the effects of actions are likely to persist from environment to environment. *mDAGL* would allow an agent to learn transferrable sparse action models. For instance, this method could be incorporated into the *TES* framework (Nguyen et al., 2012) to implement an intelligent agent that can learn, accumulate and transfer knowledge automatically between environments.

Acknowledgments

We thank PhD. Haiqin Yang for helpful comments on our sparse multinomial logistic regression. This research is supported by Academic Research Grants: MOE2010-T2-2-071 and T1 251RES1005 from the Ministry of Education in Singapore.

References

Brafman, Ronen I. and Tennenholtz, Moshe. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *In Journal of Machine Learning Research*, 3:213–231, 2003.

Chakraborty, Doran and Stone, Peter. Structure learning in ergodic factored MDPs without knowledge of the transition function's in-degree. In *Proceedings of the International Conference on Machine Learning*, ICML'11, pp. 737–744, 2011.

Degris, Thomas, Sigaud, Olivier, and Wuillemin, Pierre-Henri. Learning the structure of factored Markov decision processes in reinforcement learning problems. In *Proceedings of the International Conference on Machine Learning*, ICML'06, pp. 257–264, 2006.

Diuk, Carlos, Li, Lihong, and Leffler, Bethany R. The adaptive k-meteorologists problem and its application to structure learning and feature selection in reinforcement learning. In *Proceedings of the Interna-*

tional Conference on Machine Learning, ICML'09, pp. 249–256, 2009.

Hester, Todd and Stone, Peter. Generalized model learning for reinforcement learning in factored domains. In *Proceedings of International Conference on Autonomous Agents and Multiagent Systems*, volume 3 of *AAMAS'09*, pp. 717–724, 2009.

Hester, Todd and Stone, Peter. Texplore: real-time sample-efficient reinforcement learning for robots. *Machine Learning*, pp. 1–45, 2012.

Kroon, Mark and Whiteson, Shimon. Automatic feature selection for model-based reinforcement learning in factored MDPs. In *Proceedings of the International Conference on Machine Learning and Applications*, ICMLA'09, 2009.

Lee, S. and Wright, S. Manifold identification of dual averaging methods for regularized stochastic online learning. *In Journal of Machine Learning Research*, 13:1705–1744, 2012.

Leffler, Bethany R., Littman, Michael L., and Edmunds, Timothy. Efficient reinforcement learning with relocatable action models. In *Proceedings of the National Conference on Artificial Intelligence*, AAAI'07, pp. 572–577, 2007.

McCarthy, John. Situations, actions, and causal laws. Technical Report Memo 2, Stanford Artificial Intelligence Project, Stanford University, 1963.

Nguyen, Trung T., Silander, Tomi, and Leong, Tze Yun. Transferring expectations in model-based reinforcement learning. In *Proceedings of the Advances in Neural Information Processing Systems*, NIPS'12, 2012.

Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. ISBN 1-55860-238-0.

Ross, Stéphane and Pineau, Joelle. Model-based bayesian reinforcement learning in large structured domains. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, UAI'08, pp. 476–483, 2008.

Strehl, Er L. and Littman, Michael L. Online linear regression and its application to model-based reinforcement learning. In *Proceedings of the Advances in Neural Information Processing Systems*, NIPS'07, pp. 737–744, 2007.

Strehl, Er L., Diuk, Carlos, and Littman, Michael L. Efficient structure learning in factored-state MDPs. In *Proceedings of the National Conference on Artificial Intelligence*, AAAI'07, 2007.

Walsh, Thomas J., Szita, István, Diuk, Carlos, and Littman, Michael L. Exploring compact reinforcement-learning representations with linear regression. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, UAI'09, pp. 591–598, 2009.

Xiao, Lin. Dual averaging methods for regularized stochastic learning and online optimization. In *Proceedings of the Advances in Neural Information Processing Systems*, NIPS'09, 2009.

Yang, Haiqin, Xu, Zenglin, King, Irwin, and Lyu, Michael R. Online learning for group lasso. In *Proceedings of the International Conference on Machine Learning*, ICML'10, pp. 1191–1198, 2010.