Switching between Representations in Reinforcement Learning

Harm van Seijen, Shimon Whiteson and Leon Kester

Abstract This chapter presents and evaluates an on-line representation selection method for factored MDPs. The method addresses a special case of the feature selection problem that only considers certain sub-sets of features, which we call *candidate representations*. A motivation for the method is that it can potentially deal with problems where other structure learning algorithms are infeasible due to a large degree of the associated dynamic Bayesian network (DBN). Our method uses switch actions to select a representation and uses off-policy updating to improve the policy of representations that were not selected. We demonstrate the validity of the method by showing for a contextual bandit task and a regular MDP that given a feature set containing only a single relevant feature, we can find this feature very efficiently using the switch method. We also show for a contextual bandit task that switching between a set of relevant features and a subset of these features can outperform the performance of both individual representations, since the switch method combines the fast performance increase of the small representation with the high asymptotic performance of the large representation.

1 Introduction

In *reinforcement learning* (RL) [7, 13], an agent seeks an optimal control policy for a sequential decision problem. When the sequential decision problem is modeled as a *Markov decision process* (MDP) [2], the agent's policy can be represented as

Harm van Seijen and Leon Kester
TNO Defense, Safety and Security, Oude Waalsdorperweg 63, 2597 AK The Hague, The Netherlands, e-mail: harm.vanseijen@tno.nl, leon.kester@tno.nl

Shimon Whiteson
University of Amsterdam, Science Park 107, 1098 XG Amsterdam, The Netherlands, e-mail: s.a.whiteson@uva.nl

a mapping from each state it may encounter to a probability distribution over the available actions.

RL suffers from what is often called the 'curse of dimensionality': the state space grows exponentially as function of the number of problem parameters and it becomes quickly infeasible to solve. For the sub-domain of factored MDPs, where the state space is described through a set of features, one can often considerably reduce the complexity of the problem by making use of its structure. Boutilier et al. [3] showed for the planning case that by representing the transition and reward functions as a dynamic Bayesian network (DBN) and representing the associated conditional probability tables (CPTs) as a tree, it becomes possible to avoid the explicit enumeration of states. This allows the computation of value estimates and policy to be performed on partitions instead of individual states. They later extended this work by using an even more compact way to represent the CPTs, algebraic decision diagrams [6], and by combining it with approximate dynamic programming techniques [11].

The success of this approach inspired other people to use the idea of representing the transition and reward function as a DBN to speed up learning of the model in model-based RL. Although these methods still use explicit state enumeration, since the model is more efficiently learned also the sample efficiency is improved. The early work assumed the structure of the DBN is known, and only focussed on learning the values of the decision trees [8]. Strehl et al. [12] were able to relax the prior knowledge by showing how the structure of the DBN could be efficiently learned with only the degree of the DBN as prior knowledge. This work recently got extended by combining it with the KWIK framework [9] resulting in a method with better bounds [4].

Although these recent methods can learn the structure efficiently when the degree is small, the representation size grows exponentially in the degree N of the DBN [12, 4, 1], therefore for large values of N learning the DBN model becomes infeasible.

We present in this chapter an alternative approach that can potentially better deal with cases, where the current structure learning algorithms are infeasible. Our approach is different in two major ways. First, our method searches for the feature set that has currently the best performance, instead of searching for the feature set that has the highest performance in the limit. A second difference is that our method relies on knowledge in the form of relevant feature combinations. The advantage of this form of prior knowledge is that it allows for representing more specific domain knowledge than just the degree of the DBN, effectively reducing the problem size.

The general principle behind the algorithm is a very intuitive one. Each subset of features forms a *candidate representation*. At the start of each episode the agent selects one candidate representation and uses its policy to perform action-selection and policy updating till the end of the episode. If the episode is finished the value of the representation as a whole is updated.

Selection of the representation suffers from the same exploration-exploitation dilemma as regular action selection, i.e. the agent can choose either the representation that has the highest current estimated value, or it can choose a suboptimal representation in order to improve its policy and its estimate of that policy. Due to

this exploration cost there is a trade-off between the performance gain by using a more compact representation and the performance decrease due to the exploration of sub-optimal representations. To reduce the exploration cost we defined a number of conditions under which off-policy updating is possible of the policies of the unselected representations and of the value of the representation as a whole. This can increase the performance considerably.

The rest of this chapter is organized as follows. In section 2 we give a formal definition of a factored MDP. In section 3 we define different types of features and relate these to valid candidate representations. Section 4 describes the algorithm the case of a contextual bandit problem, a variation to the classical multi-armed bandit problem where the outcome of pulling an arm depends on context information. In section 5 we extend the ideas from section 4 to the case of an MDP. In section 6 we provide empirical results for the contextual bandit case and the full MDP case. We finish this chapter with a conclusion and a future work section.

2 Background

2.1 Factored MDP

In this section we will describe the framework of a factored MDP. We will adopt the notation from [5]. A MDP is formally defined as a 4-tuple $\langle X, A, T, R \rangle$ where

- X is the set of all states the agent can encounter
- A is the set of all actions available
- $T(x,a,x') = P(x'|x,a)$ is the transition function
- $R(x,a) = E(r|x,a)$, is the reward function.

An MDP satisfies the equations for the *Markov property*:

$$P(x_{t+1}, r_{t+1}|x_t, a_t) = P(x_{t+1}, r_{t+1}|x_t, a_t, r_t, x_{t-1}, a_{t-1}, ..., r_1, x_0, a_0) \tag{1}$$

for all x_{t+1}, r_{t+1} and all possible values of $x_t, a_t, r_t, ..., r_1, x_0, a_0$.

For a factored MDP, the state space is described using a set of state variables or *f*eatures: $X = \{X_1, ..., X_n\}$, where each X_i takes on values in some domain $Dom(X_i)$. We will only consider discrete and finite domains throughout this chapter. We will use the notation $\mathscr{D}$ to indicate the set of values with non-zero probablity. As an example consider a feature set X consisting of two indentical features X_1 and X_2, for the size of X the following holds:

$$|Dom(X)| = |Dom(X_1)| \cdot |Dom(X_2)|$$

while

$$|\mathscr{D}(X)| = |\mathscr{D}(X_1)| = |\mathscr{D}(X_2)|$$

A state x defines a value $x(i)$ for each variable X_i. We will refer to the set of variables that describes a state-space as a *representation*. For an instantiation $y \in \mathscr{D}(Y)$ and a subset of these variables $Z \subset Y$, we use $y[Z]$ to denote the value of the variables Z in the instantiation y.

The goal of a RL agent is to find an optimal policy $\pi^* = P(a|x)$, which maximizes the expected discounted return $\langle R_t \rangle$ for state s_t, with R_t defined as:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2}$$

where γ is a discount factor with $0 \leq \gamma \leq 1$.

3 Feature Selection in Factored MDPs

In this section we define several classes of features and show how these relate to constructing a *valid candidate representation*.

3.1 Feature Types

Definition 1. A feature $X_k \subseteq X$ is *irrelevant* with respect to Y if the following holds for all x^-_{t+1}, y^+_t, a_t

$$P(x^-_{t+1}, r_{t+1}|y^-_t, a_t) = P(x^-_{t+1}, r_{t+1}|y^+_t, a_t) \tag{3}$$

with

$$\begin{aligned} x^-_{t+1} &\in \mathscr{D}(X \setminus X_k) \\ y^+_t &\in \mathscr{D}(Y \cup X_k) \\ y^-_t &= y^+_t[Y \setminus X_k] \end{aligned}$$

Informally spoken, an irrelevant feature is a feature whos feature value doesn't effect the next value of any other feature (except potentially its own value), nor the reward received when a representation is used with all features from $Y \cup X_k$ are present.

The complement class is the class of relevant features:

Definition 2. A feature $X_k \subseteq X$ is *relevant* with respect to Y if it is not irrelevant with respect to Y.

We can devide the irrelevant feature class into three subclasses: constant, empty and redundant features.

Definition 3. A *constant* feature $X_k \subseteq X$ is a feature with $|\mathscr{D}(X_k)| = 1$.

A constant feature is a feature that never changes value. Note that a feature that stays constant during an episode, but changes value in between episodes is not constant according to this definition nor is it irrelevant. The difference between the two is that a feature with $|\mathscr{D}(X_k)| = 1$ can be removed from a representation, without effecting the markov property (as we will proof in section 3.2), while removing a feature of the second type can cause a violation of the Markov property. The Markov property can be violated since the history might reveal something about the current value of the feature and therefore effect the transition probability.

Since if we leave out a feature of the second type, the history might reveal something about its value and therefore effect the transition probability.

Definition 4. An *empty* feature $X_k \subseteq X$ is a feature for which the following holds

$$|\mathscr{D}(X_k)| > 1$$

$$P(x^-_{t+1}, r_{t+1}) = P(x^-_{t+1}, r_{t+1}|x(k)_t)$$

for all $x^-_{t+1} \in \mathscr{D}(X \setminus X_k), r_{t+1}, x(k)_t$.

An empty feature, is a feature that contains no usefull information, i.e. it is irrelevant with respect to the empty set.

Definition 5. A feature $X_k \subseteq X$ is *redundant* with respect to Y if it irrelevant with respect to Y and non-emtpy.

A redundant feature does contain usefull information, however this information is shared with some other features in Y. By removing some feature-set Z from Y, feature X_i can become a relevant feature with respect to $Y \setminus Z$.

Apart from the relevant/irrelevant classification, we define two other classifications: dependent and independent features.

Definition 6. A feature $X_k \subseteq X$ is *independent* if for all x_{t+1}, x_t, a_t the following holds

$$P(x_{t+1}(k)) = P(x_{t+1}(k)|x^-_{t+1}, r_{t+1}, x_t, a_t) \tag{4}$$

with

$$x_{t+1}(k) = x_{t+1}[X_k]$$
$$x^-_{t+1} = x_{t+1}[X \setminus X_k]$$

So the value of an independent feature doesn't depend on the previous state or current values of other features or the reward just received. Note that this doesn't prevent an independent feature to influence the next feature value of other features or the next reward. An independent feature is unique in the sense that it can contain relevant information, but leaving the feature out still gives a Markov representation as we will proof in the next subsection. Therefore, the regular methods still converge, although the resulting policy is not optimal in X. However, since we are primarily interested in the best online performance instead of the optimal performance, omitting independent features can play an important role in finding the best representation.

For sake of completeness, we also define the counterpart of an independent feature:

Definition 7. A feature $X_k \subseteq X$ is *dependent* if it is not independent.

3.2 Candidate Representation

A subset of features $Y \subseteq X$ forms a candidate representation. We will now define what we mean by a *valid* candidate representation.

Definition 8. Consider the MDP $M = \langle X, A, T, R \rangle$. A subset of features $Y \subseteq X$ is a *valid candidate representation* if the Markov property applies to it, i.e. if the following conditions hold :

$$P(y_{t+1}, r_{t+1} | y_t, a_t) = P(y' | y_t, a_t, r_t, y_{t-1}, a_{t-1}, ..., r_1, y_0, a_0) \tag{5}$$

for all $y_{t+1} \in \mathscr{D}(Y), r_{t+1}$ and all possible values of $y_t, a_t, r_t, ..., r_1, y_0, a_0$.

The full feature set X is per definition Markov. The following theorem shows how different feature sets can be constructed from X that maintain the Markov property, i.e. that are valid candidate representations. We will only consider valid candidate representations in the rest of this chapter.

Theorem 1. *Consider the MDP $M = \langle X, A, T, R \rangle$. A subset of features $Y \subset X$ is a valid candidate representation if for the set of missing features the following holds*

$$\forall X_i \notin Y : X_i \textit{ is irrelevant w.r.t. } Y \textit{ or } X_i \textit{ is an independent feature.}$$

Proof. To proof the theorem above it suffices to proof that if for an arbitrary set Z with $Y \subseteq Z \subseteq X$ the markov property holds, it also holds for $Z \setminus Z_k$ if Z_k is irrelevant w.r.t. Y or independent.

For the proof we will make use the following formulas that can be easily deduced from the Bayesian statistics rules:

$$P(a|b,c) \cdot P(b|c) = P(a,b|c) \tag{6}$$

$$P(a|b) = P(a|b,c,d) \Rightarrow P(a|b) = P(a|b,c) \tag{7}$$

$$P(a,b_i|c) = P(a,b_i|c,d) \quad \text{for all } i \Rightarrow P(a|c) = P(a|c,d) \tag{8}$$

with $P(b_i, b_j) = 0$ for all $i \neq j$ and $\sum_i P(b_i) = 1$.

First we will proof that the Markov property is conserved if Z_k is an irrelevant feature.

$$P(z_{t+1}, r_{t+1}|z_t, a_t) = P(z_{t+1}, r_{t+1}|z_t, a_t, r_t, z_{t-1}, a_{t-1}, ..., r_1, z_0, a_0)$$
$$P(z^-_{t+1}, r_{t+1}|z_t, a_t) = P(z^-_{t+1}, r_{t+1}|z_t, a_t, r_t, z_{t-1}, a_{t-1}, ..., r_1, z_0, a_0) \quad \text{using rule (8)}$$
$$P(z^-_{t+1}, r_{t+1}|z^-_t, a_t) = P(z^-_{t+1}, r_{t+1}|z_t, a_t, r_t, z_{t-1}, a_{t-1}, ..., r_1, z_0, a_0) \quad \text{using rule (3)}$$
$$P(z^-_{t+1}, r_{t+1}|z^-_t, a_t) = P(z^-_{t+1}, r_{t+1}|z^-_t, a_t, r_t, z^-_{t-1}, a_{t-1}, ..., r_1, z^-_0, a_0) \quad \text{using rule (7)}$$

Now for an indepedent feature:

$$P(z_{t+1}, r_{t+1}|z_t, a_t) = P(z_{t+1}, r_{t+1}|z_t, a_t, r_t, z_{t-1}, a_{t-1}, \ldots, r_1, z_0, a_0)$$
$$P(z^-_{t+1}, r_{t+1}|z_t, a_t) = P(z^-_{t+1}, r_{t+1}|z_t, a_t, r_t, z_{t-1}, a_{t-1}, \ldots, r_1, z_0, a_0) \quad \text{using rule (8)}$$

We now multiply both sides with $P(z(k)_t)$ and rewrite the left part as

$$\begin{aligned} &P(z^-_{t+1}, r_{t+1}|z_t, a_t) \cdot P(z(k)_t) \\ &\qquad = P(z^-_{t+1}, r_{t+1}|z(k)_t, z^-_t, a_t) \cdot P(z(k)_t|z^-_t, a_t) \quad \text{using rule (4)} \\ &\qquad = P(z^-_{t+1}, r_{t+1}, z(k)_t|z^-_t, a_t) \quad \text{using rule (6)} \end{aligned}$$

and the right part as:

$$\begin{aligned} &P(z^-_{t+1}, r_{t+1}|z_t, a_t, r_t, z_{t-1}, a_{t-1}, ..., r_1, z_0, a_0) \cdot P(z(k)_t) \\ &\qquad = P(z^-_{t+1}, r_{t+1}|z(k)_t, z^-_t, a_t, r_t, z_{t-1}, a_{t-1}, ..., r_1, x_0, a_0) \\ &\qquad \cdot P(z(k)_t|z^-_t, a_t, r_t, z_{t-1}, a_{t-1}) \quad \text{using rule (4)} \\ &\qquad = P(z^-_{t+1}, r_{t+1}|z(k)_t, z^-_t, a_t, r_t, z_{t-1}, a_{t-1}, ..., r_1, x_0, a_0) \\ &\qquad \cdot P(z(k)_t|z^-_t, a_t, r_t, z_{t-1}, a_{t-1}, ..., r_1, z_0, a_0) \quad \text{using rule (5)} \\ &\qquad = P(z^-_{t+1}, r_{t+1}, z(k)_t|z^-_t, a_t, r_t, z_{t-1}, a_{t-1}, ..., r_1, z_0, a_0) \end{aligned}$$

Combining these results gives:

$$P(z^-_{t+1}, r_{t+1}, z(k)_t|z^-_t, a_t) = P(z^-_{t+1}, r_{t+1}, z(k)_t|z^-{}_t, a_t, r_t, z_{t-1}, a_{t-1}, ..., r_1, z_0, a_0)$$
$$P(z^-_{t+1}, r_{t+1}|z^-_t, a_t) = P(z^-_{t+1}, r_{t+1}|z^-_t, a_t, r_t, z^-_{t-1}, a_{t-1}, ..., r_1, z^-_0, a_0) \text{ using (7),(8)}$$

□

4 Representation Selection for a Contextual Bandit

In this section we describe our representation selection method for a contextual bandit, a subclass of MDP problems where only a single action has to be taken. We will start by introducing an example contextual bandit task that will serve as a reference for the rest of this section. In section 4.2 we will explain how formally

the selection problem can be seen as solving a derived MDP. Section 4.3 shows how a representation can be evaluated. Section 4.4 describes the performance of our method can be increased by off-policy updating the Q-values of the representations and of the switch actions. Note that for a contextual bandit, each subset of features forms a valid candidate representation, since there is no history to consider and therefore the Markov property is never violated.

4.1 A Contextual Bandit Example

The standard multi-armed bandit problem represents a class of RL tasks, based on a traditional slot-machine but with multiple arms. Pulling an arm results in a reward drawn from a distribution associated with that arm. The goal of an agent is to maximize its reward over iterative pulls. The agent has no prior knowledge about the distributions associated with each arm. The dilemma the agent faces when performing this task is to either pull the arm that has currently the highest expected reward or to improve the estimates of the other arms. This is often referred to as the exploration versus exploitation dilemma. The contextual bandit problem is a variation of the multi-armed bandit problem where the agent observes context information that can affect the reward distributions of the arms. We can connect the bandit problem to an MDP, by interpreting the arms as actions and the context as the state. When the context is described through features, we can interpret the problem as a factored MDP where episodes have length 1.

We will now describe an example of a contextual bandit that will be used as a reference for the next sections. Consider a contextual bandit with two arms: a_0 and a_1. The context is described by two binary features: X and Y. The expected reward for a_0 is 0 for all context states, while the expected reward for a_1 can be either positive or negative depending on the context state. Figure 1(a) shows the expected reward for arm a_1 for all feature value combinations. The agent is given two candidate representations to choose from on this task. Representation S_X consists of only feature X, while representation S_Y consists of only feature Y. Both representations partition the original state space into two abstract states, one corresponding to each feature value. We can calculate the expected reward for these states from the full table in 1(a) and the prior probabilities of the feature values, which we assume to be 0.5 for both values of X and Y. Figures 1(b) and (c) show the expected reward of arm a_1 for representation S_X and S_Y respectively. Note, that the optimal policy of representation S_X has an expected reward of 1.5 (action a_0 will be chosen when $X = 1$), while the optimal policy for representation S_Y has an expected reward of 0.5.

In the next section, we will show for this example task how the concept of representation selection can be interpreted as solving a derived MDP.

X	Y	R
0	0	+4
0	1	+2
1	0	-2
1	1	-4

(a)

X	R
0	+3
1	-3

(b)

Y	R
0	+1
1	-1

(c)

Fig. 1 Expected return of arm a_1 for the full representation (a) and representation S_X (b) and S_Y (c) assuming a prior probability of 0.5 for the features-values 0 and 1 of feature X and Y.

4.2 Constructing the Switch Representation

Formally, the representation is one of the elements that defines an MDP task. Therefore, an agent that can choose between multiple representations, actually chooses between multiple MDP tasks, each of them having a different state space X, transition function T and reward function R, but with equal action set A. We can combine these multiple MDP tasks into a single 'switch' MDP task by defining *switch actions*, actions that have no effect on the environment but merely select the subtask the agent will try to solve.

In figure 2 we can see how this switch MDP is constructed from the individual MDPs for the example task described earlier. The agent can choose between an MDP that uses representation S_X, based on feature X, and S_Y, which is based on feature Y. Switch actions are defined that let the agent use either representation S_X or S_Y. The state space of the switch MDP consists of 6 states, a single start state from which the representation selection occurs, the non-terminal states from representations S_X and S_Y and one terminal state. Note that by including switching actions, the derived MDP has episodes of length 2. The switch actions are stochastic actions with 0 reward.

Defining a single start state from which switch actions are selected, guarantees that the Q-values of the switch actions are set independent from the feature values. We will motivate this approach, by demonstrating what happens for our bandit example if the representation is selected based on the Q-values of the candidate representations. Assume that the current state is defined by features values (x = 1, y = 0). From figure 1(a) we can see that the expected reward is -2. Representation S_X, which only sees feature X, predicts a reward of -3 in this case, while representation S_Y predicts a reward of +1. Therefore, if the representation selection would be based on the Q-values of the representations under consideration, S_Y would be selected in this case, and arm a_1 would be pulled which, despite the prediction of +1, would result in a reward drawn from a distribution with a mean of -2. On the other hand, if we had selected S_X, action a_0 would be the optimal action, which has an expected reward of 0. This illustrates that using the Q-values of the individual representations can lead to incorrect behavior. By defining switch-actions with specific Q-values

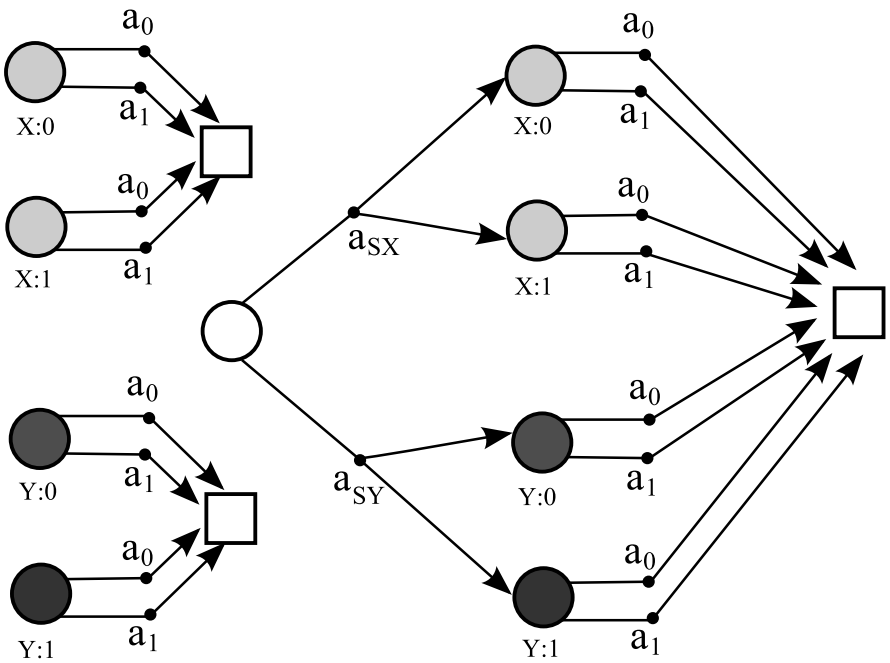

Fig. 2 The individual MDPs for the contextual bandit example and the derived switch MDP. The circles indicate specific states, the square indicates the end state and the black dots indicate actions. Stochastic actions have multiple arrows.

this problem is avoided, since these Q-values represent the average performance of a representation, which is 1.5 for S_X and 0.5 for S_Y.

The switch MDP is an MDP with special properties. One of its properties is that not all the actions are external actions. Although the MDP framework does not distinguish between internal or external actions, for our performance criteria it does matter, since we want to optimize only with respect to the external actions. In the next sections we will see how this property can be exploited to increase the on-line performance.

4.3 Evaluation of a Representation

In the example bandit task, the agent has to choose between only two equally sized representations. In the general case however, there can be many candidate representations of different sizes and information content. The policy of a representation is not the same at each time step, but it will improve over time until all of its Q-values have converged. Since small representations converge faster, a good on-line strategy can be to use a small representations for the early phase of learning and switch to

a larger representations with more information content, at a later stage. To be able to switch from one representation to another at the right moment, the agent needs to know the *current expected return* for each representation, which we define as follows:

Definition 9. The *current expected return of a representation* is the expected return of the start state of that representation, assuming the current policy of the representation is used. If the start state is drawn from a probability distribution of start states, then the current expected return refers to the weighted average of the expected returns of the start states, weighted according to their probability.

The Q-value of a switch action gives an estimate of the current expected return of the corresponding representation. To get a meaningful estimate, this Q-value cannot be updated by bootstrapping from the Q-values of the representation, since these converge too slowly and give not an accurate estimate for the current policy. Instead we use a monte-carlo update, which uses the complete return. Since the policy of a representations changes during learning, the most recent return gives the least biased estimate for determining the *current* expected return. However, since the variance on the return is very high it is necessary to average over multiple returns to get an accurate estimate. Tuning the learning rate is therefore essential for a good trade-off between the error due to variance and the error due to bias. The learning rate of the switch-actions is therefore set independently from the learning rate for the representations.

4.4 Improving Performance by Off-Policy Updating

Since the agent has to consider multiple representations, the exploration cost increases. Fortunately, the exploration cost can be reduced considerably by using off-policy updating techniques to simultaneously improve the policy of all representations. We can reduce the exploration cost further by also using off-policy updating for the switch actions. Off-policy updating means that the policy used to generate the behavior, i.e. the *behavior policy*, is different from the policy we try to evaluate, i.e. the *evaluation policy*.

4.4.1 Updating of the Unselected Representations

At the start of an episode, the agent selects via a switch action the representation it will use to base its external action selection on. If we assume that the agent can observe the full feature set, it can also observe what state it would end up in had it taken a different switch action. Therefore, parallel to the real experience sequence from the selected representation, an experience sequence for the other candidate representations can be constructed. These sequences can be used to perform extra updates.

Given our example task, assume that the context is described by feature values $x = 0$ and $y = 1$ and assume that representation S_X is selected for action selection. A possible experience sequence is then

$s_{start} \rightarrow a_{S_X} \rightarrow s_{x:0} \rightarrow a_1 \rightarrow r, s_{end}$

With this sequence we can update $Q(s_{x:0}, a_1)$. By observing feature Y we can create an experience sequence for S_Y

$s_{start} \rightarrow a_{S_Y} \rightarrow s_{y:1} \rightarrow a_1 \rightarrow r, s_{end}$

This parallel sequence can be used to update $Q(s_{y:1}, a_1)$. We have to take into account however that the update is biased since the action selection is based on representation S_X.

Underlying this bias is that state $s_{y:1}$ is actually an aggregation of two states from the full features set: $(x = 0, y = 1)$ and $(x = 1, y = 1)$. The expected reward for state-action pair $(s_{y:1}, a_1)$ is a weighted average of the expected rewards of these two underlying states. Since representation S_X aggregates the states from the full features set in a different way, the two states underlying $s_{y:1}$ corresponds to two different states in representation S_X. Therefore, if the selection probability of a_1 for those states is different, the rewards are not properly weighted to estimate the expected reward for $(s_{y:1}, a_1)$.

This bias can be avoided by only updating an unselected representation under certain conditions.

Theorem 2. *Assume that action selection occurs according to representation S_1 and we want to update representation S_2 based an a parallel experience sequence. We can perform an unbiased update of S_2 if one of the following two conditions hold:*

1. *if the feature set of S_1 is a subset of the feature set of S_2*
2. *if the action was an exploratory action under an exploration scheme that does not depend upon the specific state, like an ε-greedy exploration scheme*

Proof (sketch). Under the first condition, a single state of S_2 always corresponds to a single state of S_1, and therefore the states aggregated by S_2 never get incorrectly weighted. For the second case, remember that both S_1 and S_2 are constructed from X by removing irrelevant and independent features. Therefore, all the features that are in S_1, but not in S_2 are either irrelevant or independent. If they are irrelevant, the one-step model is the same regardless of the feature value and therefore using a different weighting of the aggregated states doesn't matter. If the feature is an independent feature, the one-step model should be weighted according to the feature value probability. When taking an exploration action, the selection probability of an action is the same regardless the underlying state, and therefore this garantees that we correctly weigh the aggregated states.

By off-policy updating of the unselected representations, we can improve the policy of a representation even if we do not select it, resulting in an overall better performance for the switching method. Besides updating the Q-values of the unselected representations, we can also update the Q-values of the switch actions corresponding to the unselected representations. This is discussed in the next section.

4.4.2 Updating the Unselected Switch-actions

Since the switch actions are updated using Monte Carlo updates we cannot apply the conditions from the previous section for off-policy updating of the switch actions. Off-policy Monte Carlo updating can be achieved using a technique called importance sampling [10].

To understand the difference with regular (on-policy) Monte Carlo updates consider that we determine the Q-value of a state-action pair (x,a) by simply taking the average of all returns seen so far:

$$Q(x,a) = \frac{\sum_{i=1}^{N} R_i(x,a)}{N} \tag{9}$$

where $R_i(x,a)$ is the return followed by the i-th visit of state-action pair (x,a) and N is the total number of returns observed for (x,a). A similar off-policy version can than be made by taking the weighted average:

$$Q(x,a) = \frac{\sum_{i=1}^{N} w_i(x,a) \cdot R_i(x,a)}{\sum_{i=1}^{N} w_i(x,a)} \tag{10}$$

where $w_i(x,a)$ is the weight assigned to the i-th return for (x,a). The value of $w_i(x,a)$ is computed as follows. Let $p(x,a)$ be the probability of the state action sequence following (x,a) occurring under the estimation policy π and $p'(x,a)$ be the probability of it occurring under the behavior policy π'. Than the weight $w_i(x,a)$ is equal to the relative probability of the observed experience-sequence of occurring under π and π', i.e. by $p(x,a)/p'(x,a)$. These probabilities can be expressed in their policy probabilities by:

$$w(x_t,a_t) = \frac{p(x_t,a_t)}{p'(x_t,a_t)} = \prod_{k=t+1}^{T-1} \frac{\pi(x_k,a_k)}{\pi'(x_k,a_k)} \tag{11}$$

For a deterministic evaluation policy π the weight w is non-zero only when all actions taken under π' match the action that would have been taken under π. If this is the case, the above equation simplifies to:

$$w(x_t,a_t) = \prod_{k=t+1}^{T-1} \frac{1}{\pi'(x_k,a_k)} \qquad \text{if } \pi(x_k) = a_k \text{ for all } k \geq t+1 \tag{12}$$

where $\pi(x_k)$ refers to the action the agent would take at timestep k (with probability 1) when following this deterministic policy.

We will now consider again our example contextual bandit task and the state action sequence from the unselected representation:

$s_{start} \rightarrow a_{S_Y} \rightarrow s_{y:1} \rightarrow a_1 \rightarrow r, s_{end}$

Since the state-action pair that requires off-policy updating, (s_{start}, a_{S_Y}), is followed by only a single action, and since we use a deterministic evaluation policy, the weight expression is reduced even further to

$$w = \frac{1}{\pi'(s_{x:0}, a_1)} \qquad \text{if } \pi(s_{x:0}) = a_1 \tag{13}$$

Given that we use an ε-greedy behavior policy and the condition that $\pi(s_{x:0}) = a_1$, the weight w is a fixed value and can be scaled to 1 in this case.

If the external action is an optimal action, we can perform the following update of the switch action

$$Q(s_0, a_{S_Y}) = (1-\alpha)\, Q(s_0, a_{S_Y}) + \alpha \cdot r \tag{14}$$

To update the switch action of an unselected representation S_i, two conditions have to hold

1. the reward should be unbiased with respect to S_i, i.e., one of the two conditions of section 4.4 should hold
2. the selected action is an optimal action according to S_i

5 Representation Selection for an MDP

Although the framework introduced in the previous section is not strictly limited to the bandit case, applying it to the more general MDP task brings up some complications. As explained in the previous section, for a contextual bandit problem any subset of features from the total feature set forms a valid candidate representations. This is not generally the case for a full MDP, since partial observability often leads to non-Markov behavior. For a valid candidate representation theorem 1 should hold. We will only consider valid candidate representations in this section. In the subsections below we explain how off-policy updating works in case an MDP.

5.1 Off-Policy Updating of the Unselected Representations

The method to off-policy update an unselected representation in the MDP case is very similar to the bandit case. The only difference is that for the bandit case the update target consists only of the reward, while for the MDP case the update target consists of a reward and next state. For updating an unselected candidate representations, this next state is the state according to that candidate representations. So if we have consider the representations S_X and S_Y and select S_X as the representation for action selection, we use the sequence

$\ldots \rightarrow r_t, s_{x_t} \rightarrow a_t \rightarrow r_{t+1}, s_{x_{t+1}} \rightarrow \ldots$

To perform the update

$$Q(s_{x_t}, a_t) = (1-\alpha)\, Q(s_{x_t}, a_t) + \alpha \left(r_{t+1} + \gamma \max_a Q(s_{x_{t+1}}, a)\right) \tag{15}$$

And, if one of the conditions for unbiased updates from section 4.4 holds we use the parallel sequence of S_Y

$\ldots \rightarrow r_t, s_{y_t} \rightarrow a_t \rightarrow r_{t+1}, s_{y_{t+1}} \rightarrow \ldots$

to perform the update

$$Q(s_{y_t}, a_t) = (1-\alpha)\, Q(s_{y_t}, a_t) + \alpha\,(r_{t+1} + \gamma \max_a Q(s_{y_{t+1}}, a)) \tag{16}$$

5.2 Off-Policy Updating of the Unselected Switch-actions

For the contextual bandit case, it was possible to perform very efficient off-policy Monte Carlo updates of the unselected switch-actions. Therefore, exploration of the switch-actions is not necessary for a contextual bandit problem.

Off-policy Monte Carlo updates for the MDP case are much less efficient. For a deterministic evaluation policy π, all the actions of an episode generated using the behavior policy π' have to be equal to π to get a non-zero weight (see (13)). Since for our switch method, the behavior policy is based on a different representation, this rarely is the case. If an stochastic evaluation policy would be used, the weights will always be non-zero, but the difference in weight values will be very large, also causing the off-policy updates to be inefficient. Therefore for the MDP case, we only perform updates of switch action corresponding to the selected representation and use exploration to ensure all switch actions are updated.

6 Experimental Results and Discussion

6.1 Contextual Bandit Problem

In the first experiment we compare the performance of the switching method against the performance of the full representation given prior knowledge that only one feature from a set of features is a relevant feature, while the others are empty features. The feature values are randomly drawn from their domain values after each pull of an arm. We make this comparison for a feature set of 3 and of 4 features. Each feature has 8 feature values. The bandit has two arms with opposite expected reward. Depending on the context, one has an expected reward of +1, while for the other arm it is -1. The reward is drawn from a normal distribution with a standard deviation of 2. For half of the feature values of the information-carrying feature the first arm has the +1 expected reward. Therefore without this feature the expected reward is 0.

The candidate representations for the switch algorithm consist in this case of one representation per feature. We used a learning rate of 0.01 for the representation evaluation and selected the representation greedy with respect to this evaluation. The exploration scheme is ε-greedy with $\varepsilon = 0.2$ for all methods. To kick-start the

switch method we used only exploration for the first 50 episodes. Since all candidate representations are updated during this exploration phase, this has a positive effect on the total reward.

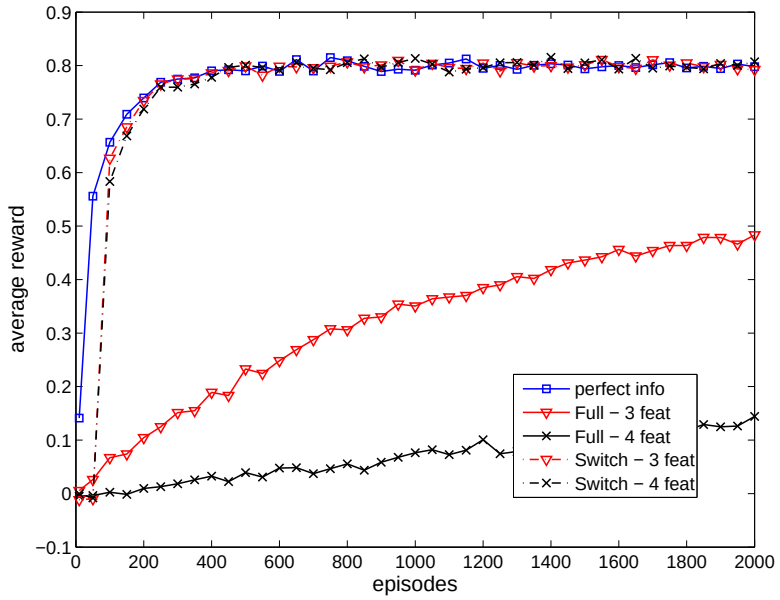

Fig. 3 Comparison of the switch method with the full feature set for a feature set of size 3 and size 4 for a contextual bandit problem. The perfect info graph shows what the performance would have been had the agent known beforehand what the relevant feature would be.

Figure 3 shows the results. To get an upperbound we also plotted the performance of a representation that consists only of the relevant feature. All results are averaged over 10.000 independent runs and smoothed.

The advantage of the switching method is very clear for this task. While the full representation grows exponentially with the number of features, with 512 context states for a set of 3 features and 4096 for a set of 4, the total number of states for the switching methods grows linear, with 24 states for a set of 3 features and 32 states a set of 4. The simultaneous updating of the representations on exploration increases the performance even further and brings it very close to the representation containing only the information carrying feature.

For our second experiment we consider a different contextual bandit task. For this bandit task we have 3 information containing features, but one of them contains the most information. We will demonstrate with this task that using a small incomplete representation for the initial learning phase can improve the overall performance for certain tasks. We compared in this case the performance of a representation containing only the most important feature (REP-SMALL), with the representation containing all 3 relevant features (REP-LARGE) and the switch method that has both

representations as candidate set (SWITCH). We used a learning rate of 0.001 for the representation evaluation and greedy representation selection. The exploration scheme is again ε-greedy with $\varepsilon = 0.2$ for all methods. The switch method uses only exploration for the first 100 episodes. The results are averaged over 10.000 independent runs and smoothed.

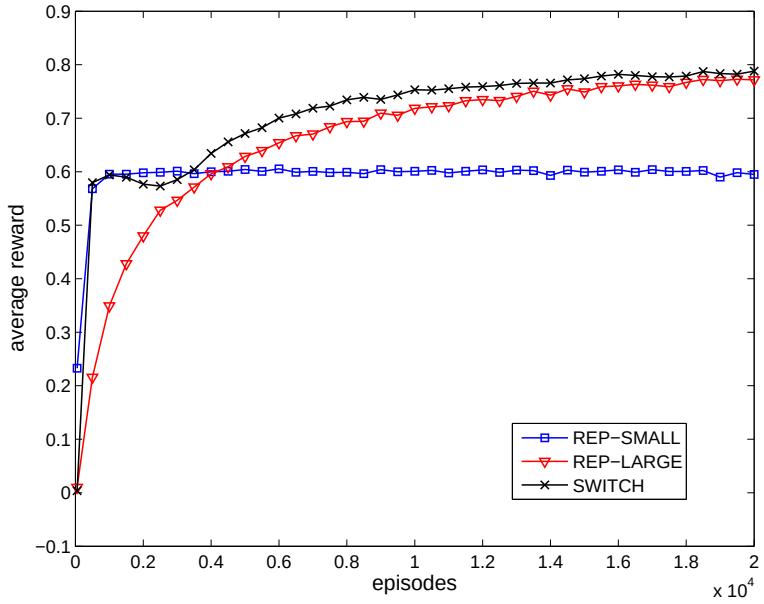

Fig. 4 Average reward for a contextual bandit when switching between a large and a small representation.

Figure 4 shows the average reward for the first 20.000 episodes. The performance of representation REP-SMALL is about 3/4 of that of representation REP-LARGE. After about 500 episodes, REP-LARGE has seen enough samples to outperform REP-SMALL. After the initial exploration phase, the switch method catches up very quickly with REP-SMALL, then the performance shows a small ditch before it starts climbing to new performance levels. We explain the ditch as following: while REP-SMALL is quickly recognized as the better representation we keep improving the Q-values of REP-LARGE as well as the Q-values of the switch actions by off-policy updating. Once the Q-value of the selection action for REP-LARGE approaches the Q-value of the selection action for REP-SMALL, for some of the runs the estimates will prematurely indicate REP-LARGE as the better one, causing a small ditch in the performance. Then, when the Q-values of REP-LARGE further improve, it will really outperform REP-SMALL causing the climb in performance. Interesting about the performance of the switch method is that it outperforms REP-LARGE at each point during learning. We explain this as following, the exact point where REP-LARGE outperforms REP-SMALL is different for each run. Since the switch method uses an

up-to-date estimate of the expected reward for each representation, it simply makes longer use of REP-SMALL for those runs where the Q-values of REP-LARGE are improved more slowly than average. Therefore, once REP-SMALL has been properly learned its performance forms a lower boundary for the remaining episodes. This lower boundary is not present when using only REP-LARGE (which performance boundary is 0) and therefore on average the switch method will do better at each point during learning.

6.2 MDP task

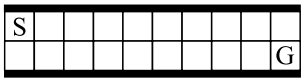

Fig. 5 The Corridor Task. The agent has to move from S to G, while avoiding bumping into the wall (thick black line).

The task we will consider is an episodic navigational task, where the agent has to move through a narrow corridor while avoiding bumping into the walls. The agent can take any of four movement actions: up, down, left and right. On top of this, the agent moves an additional step in either the up or the down direction (think of the corridor as being on a rocking ship). The information about the direction of the additional step is useful, since if the additional step would cause the agent to hit the wall, it can prevent this by making a step in the opposite direction. Whether the additional step is in the up or down direction, depends on the context, specified by a set of independent features that change value at each timestep. Only one of those features contains useful information, but the agent does not know which one. This creates an MDP equivalent of the first contextual bandit problem. Each step results in a reward of -1 plus an additional reward of -10 if the agent hits a wall. The environment is stochastic causing the agent to make, with a probability of 20%, a step in a random direction instead of the direction corresponding to its action. The discount factor is 0.95.

We compare the on-line performance of the switch method against the performance of the full feature set for a feature set that consists, besides the position feature, of 3 or 4 features describing the direction of the additional step. The candidate representations for the switch method consist of the position feature and one additional feature. We used a learning rate of 0.05 for all states and an ε-greedy policy with $\varepsilon = 0.1$ for the switch actions as well as regular actions.

Figure 6 shows the on-line performance for the first 10.000 episodes for a set of 3 features and a set of 4. As a reference we also show the performance of a representation consisting of only the position feature and the relevant additional feature. The results are averaged over 100 independent runs and smoothed.

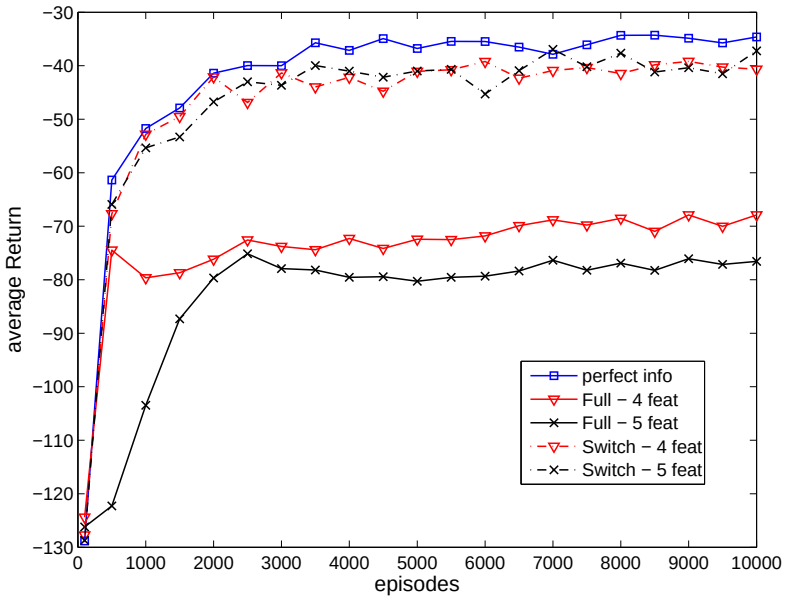

Fig. 6 Comparison of the switch method with the full representation on the corridor task for a feature set size of 4 and 5. The perfect info graph shows what the performance would have been had the agent known beforehand what the relevant feature would be.

The performance of the full representations increases fast during the initial learning phase, but after that increases only very slowly. As expected, the switch method performs a lot better and is close to the performance of the reference representation. That the switch method converge to a performance slightly lower than the reference representation is due to the additional exploration of the switch actions, causing sub-optimal representations to be selected.

7 Conclusions

We have presented a method for on-line representation selection for factored MDPs. The method addresses a special case of the feature selection problem, that only considers certain sub-sets of features, called *candidate representations*. The problem of representation selection is formalized by defining switch actions that select the representation to be used. The switch actions combine the MDP tasks corresponding to the candidate representations into a single switch MDP, that is than solved using standard RL techniques. Since the agent can observe the full feature set, parallel experience sequence can be constructed corresponding to the unselected representations. These parallel sequences can be used to off-policy update the Q-values of unselected representations. To update the Q-values of the switch actions Monte

Carlo updates are used since this gives a better estimate of the current performance of a representation than bootstrapping from the representations Q-values.

We demonstrated the validity of the method by demonstrating for a contextual bandit task and a regular MDP that given a feature set containing only a single relevant feature, we can find this feature very efficiently using the switch method. We also showed for a contextual bandit task that switching between a set of relevant features and a subset of these features our method can outperform both individual representations, since it combines the fast performance increase of the small representation with the high asymptotic performance of the large representation.

8 Future Work

Having a set of candidate representations, also contains some structure information. For example, the feature set size of the largest candidate representation gives an upper-limit for the degree of a DBN structure representation. This way, prior knowledge about features can be translated into structural parameters. We would like to compare our feature-based algorithm with algorithms learning the structure based on this translation. We expect that for problems with limited learning time, our algorithm has a higher online-performance, since from the start of an episode the exploration-exploitation dilemma is taken into account, whereas structure-learning algorithms typically perform exploration until they have an accurate model of the environment. We also would like to see if we can combine some of our ideas about online estimation of the current value of a feature set with structure learning.

Throughout this chapter we only considered valid candidate representations, i.e. representations for which the Markov property holds. We expect however that the method will also perform well if there are some candidate representations among the total set of candidate representations that are non-Markov, since their lower average performance will refrain the agent from selecting them. We would like to test this intuition on a number of benchmark problems.

Acknowledgements The authors would like to thank Christos Dimitrakakis for discussions about this paper.

References

1. Abbeel, P., Koller, D., Ng, A.: Learning factor graphs in polynomial time and sample complexity. The Journal of Machine Learning Research **7**, 1743–1788 (2006)
2. Bellman, R.E.: A Markov decision process. Journal of Mathematical Mechanics **6**, 679–684 (1957)
3. Boutilier, C., Dearden, R., Goldszmidt, M.: Exploiting structure in policy construction. In: International Joint Conference on Artificial Intelligence, vol. 14, pp. 1104–1113 (1995)

4. Diuk, C., Li, L., Leffler, B.: The adaptive k-meteorologists problem and its application to structure learning and feature selection in reinforcement learning. In: Proceedings of the 26th Annual International Conference on Machine Learning. ACM New York, NY, USA (2009)
5. Guestrin, C., Koller, D., Parr, R., Venkataraman, S.: Efficient solution algorithms for factored mdps. Journal of Artificial Intelligence Research **19**, 399–468 (2003)
6. Hoey, J., St-Aubin, R., Hu, A., Boutilier, C.: Spudd: Stochastic planning using decision diagrams. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pp. 279–288. Morgan Kaufmann (1999)
7. Kaelbling, L.P., Littman, M.L., Moore, A.P.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research **4**, 237–285 (1996)
8. Kearns, M., Koller, D.: Efficient reinforcement learning in factored mdps. In: International Joint Conference on Artificial Intelligence, vol. 16, pp. 740–747 (1999)
9. Li, L., Littman, M., Walsh, T.: Knows what it knows: a framework for self-aware learning. In: Proceedings of the 25th international conference on Machine learning, pp. 568–575. ACM New York, NY, USA (2008)
10. Siegmund, D.: Importance sampling in the monte carlo study of sequential tests. Annals of Statistics **4**, 673–684 (1976)
11. St-Aubin, R., Hoey, J., Boutilier, C.: Apricodd: Approximate policy construction using decision diagrams. In: Proceedings of Advances in Neural Information Processing Systems, pp. 1089–1095. MIT Press (2000)
12. Strehl, A., Diuk, C., Littman, M.: Efficient structure learning in factored-state mdps. In: Proceedings of the Twenty-Second National Conference on Artificial Intelligence, vol. 22, p. 645. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2007)
13. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, Massachussets (1998)