Simultaneous Learning of Structure and Value in Relational Reinforcement Learning

Scott Sanner SSANNER@CS.TORONTO.EDU
Department of Computer Science, University of Toronto, Toronto, ON M5S 3H5, CANADA

Abstract

We introduce an approach to model-free relational reinforcement learning in finite-horizon, undiscounted domains with a single terminal reward of success or failure. We represent the value function as a relational naive Bayes network and show that both the value (parameters) and structure of this network can be learned efficiently under a minimum description length (MDL) framework. We describe the SVRRL and FAA-SVRRL algorithms for efficiently performing simultaneous structure and value learning and apply FAA-SVRRL to the domain of Backgammon. FAA-SVRRL produces a high-performance agent in very few training games and with little computational effort, thus demonstrating the efficacy of the SVRRL approach for large relational domains.

1. Introduction

The field of relational reinforcement learning (RRL) has emerged in recent years as a major area of focus in the reinforcement learning community (Tadepalli et al., 2004; van Otterlo & Kersting, 2004). While RRL is an attractive approach for learning from delayed reward in a relational state representation, its generality does not come without severe representational and computational drawbacks:

- As the number of ground domain objects and the arity of the relations increase, there is a combinatorial explosion in the number of ground relations that describe a state. This results in an extremely large state space that can quickly become unmanageable, even if there are only a few relations in the problem specification.
- One must carefully decide on the hypothesis space from which a value function or policy may be selected. If too simple of a space is used, the learner may not be able to obtain a good representation of the value function. And if too complex of a space is used, the learner may never be able to find a good representation or obtain enough data to achieve a low-variance estimate of the value function.
- Finding a relational structure for a value function or policy that is optimal for all domain instantiations is extremely computationally difficult. Although a few approaches have provided algorithms for exact representations of relational value functions (Boutilier et al., 2001; Hölldobler & Skvortsova, 2004; Kersting et al., 2004), these techniques have only been applied successfully to relatively simple problem descriptions. In practice, exact value function representations are difficult to obtain and the driving research question is how to efficiently find good approximations of a value function (or policy).

In this paper we attempt to address the above issues by introducing an approach to model-free relational reinforcement learning that induces structure as it learns. This approach has the advantage of allowing a learner to start with a simple relational representation and augment it as needed to learn structure that is useful for predicting the value function. We apply this learning technique to Backgammon and demonstrate that it can produce a high-performance agent with very little computational effort and in very few training games in comparison to other state-of-the-art Backgammon learning algorithms.

2. Background and Related Work

2.1. Relational Reinforcement Learning

There are two major approaches to RRL: model-based and model-free. In model-based RRL, one usually as-

Appearing in *Proceedings of the ICML'05 Workshop on Rich Representations for Reinforcement Learning*, Bonn, Germany, 2005.

sumes the problem is modelled *explicitly* as a Markov decision process (MDP) (Puterman, 1994) with relational state space structure (RMDP). Then it is possible to use a relational generalization of an MDP solution algorithm to solve for the value function or policy of the RMDP. In contrast, model-free RRL usually assumes an *implicit* underlying RMDP and attempts to learn the parameters (and possibly structure) of a value function via direct experience without access to the underlying model. Since we focus on model-free approaches in this work, we refer the reader to recent work this area (Sanner & Boutilier, 2005) that includes a discussion of related work.

While all of the model-free RRL approaches are too numerous to mention, many algorithms are variants of an approach that approximates the value or Q-function with logical regression trees (Dzeroski et al., 1998). While this learning approach is top-down in that it recursively partitions the state space into finer value distinctions, more recent work (Walker et al., 2004; Croonenborghs et al., 2004) has taken a bottom-up approach to finding useful features for predicting value and combining them to make a predictive estimation of state value. While our SVRRL algorithm takes this latter approach, it focuses specifically on a relational naive Bayes net representation of the value function and presents efficient techniques for learning both the value (parameters) and structure of this network under a minimum description length (MDL) framework.

2.2. Bayes Net Structure Learning

Since our goal is to learn relational naive Bayes net structure, the most relevant work along these lines is in the field of Bayes net structure learning. Friedman and Goldschmidt (1996) evaluate the tree-augmented naive Bayes (TAN) network structure for learning classifiers in propositional domains. While we opt for a naive relational Bayes network rather than a propositional TAN network, we do leverage a similar theoretical framework in our approach. In the area of learning structure in relational Bayes nets, Friedman *et al* (1999) propose techniques for learning a general class of probabilistic relational models (PRMs) from a fixed dataset. While this explicit search-based approach seems too computationally expensive to perform on-line in an RRL agent, future extensions of SVRRL could incorporate some of these ideas.

3. Relational Reinforcement Learning Framework

In this paper, we restrict ourselves to undiscounted, finite-horizon domains with a single terminal reward

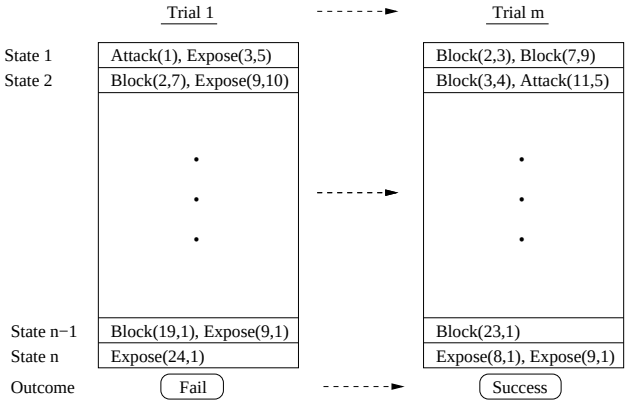

Figure 1. A diagram depicting training of the RRL agent in the domain of Backgammon (see Figure 2 for a detailed explanation of the relations). During each trial, the agent keeps track of all positive (i.e. true) ground atoms of these templates at every intermediate state (we've reduced the arity of the relations for readability). Once a terminal win or loss has been reached, the prior $P(W)$ and the conditional probability tables for the relational features $P(F_i|W)$ appearing during the trial are updated. This progresses for m trials.

of success or failure. Additionally, we assume that the state is described using a fixed set of relations $R = \{R_1, \ldots, R_i\}$, each having some finite arity. Each relation argument is assigned an attribute type from a set $A = \{A_1, \ldots, A_j\}$ where each attribute is itself a set of legal values the attribute can take. We refer to a relation and the attribute specification for its arguments as a *relation template*.

To make this more concrete, we provide a simple example domain with a single relation template $R_1(A_1, A_2)$ where $A_1 = \{a, b\}$ and $A_2 = \{1, 2\}$ so that there are four possible ground instantiations of this template: $R(a, 1)$, $R(a, 2)$, $R(b, 1)$, and $R(b, 2)$. Treating each ground atom as a binary proposition, the state is given by the full truth assignment (i.e. true or false) to each of the four propositions. Thus, in this simple relational domain, there are 2^4 or 16 possible states.

In the following discussion, we use binary propositions F_i to denote generic ground atoms (a.k.a. features) which can take on the value *true* denoted by f_i and the value *false* denoted by $\bar{f}_i$. For representational efficiency, we assume the state is represented by only the positive (true) atoms, which we arbitrarily label $\{f_1, \ldots, f_p\}$. Then, given that that there are a total of n ground atoms in a problem domain, we use absence-as-negation to infer that the remaining atoms $\{\bar{f}_{p+1}, \ldots, \bar{f}_n\}$ are false. We let $F = \{F_1 \ldots F_n\}$ (all ground atoms) and represent a state instantiation $f \in F$ as $f = \{f_1, \ldots, f_p \bar{f}_{p+1}, \ldots, \bar{f}_n\}$ (a truth assignment to all ground atoms). The omission of negative

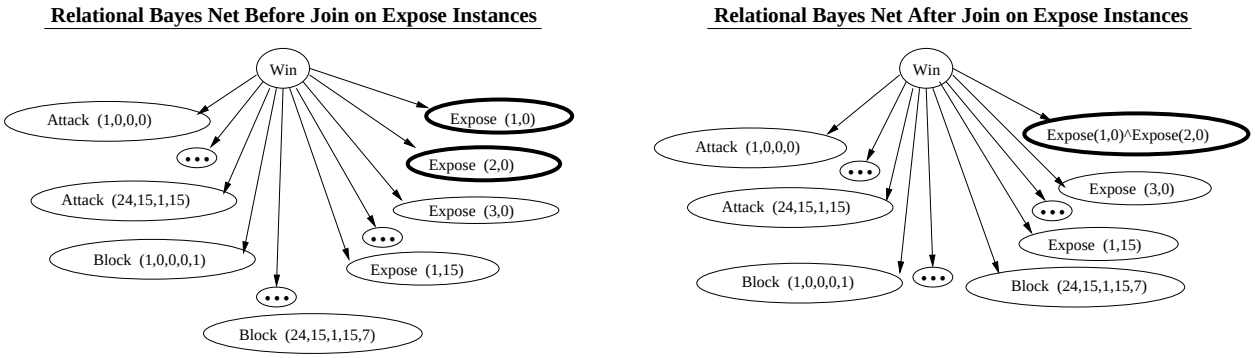

Figure 2. The relational naive Bayes net representation of the value function in Backgammon. In this domain, there are 24 points on a Backgammon board where a player's pieces can be placed (each player is assigned 15 pieces in the beginning and this number decreases as they manage to successfully bear each piece off the board). There is also a *bar* position where pieces can be placed when they have been blotted (i.e. *attacked* by the opponent because they were *exposed* by themselves on a point) and must wait to reenter the board. We use five attribute types, $PT = \{1, \ldots, 24\}$ for point locations, $OP = \{1, \ldots, 15\}$ for a count of opponents ahead of a point, $ON = \{1, \ldots, 15\}$ for a count of opponents within 7 points, $OB = \{1, \ldots, 15\}$ for a count of opponents on the bar, and $SZ = \{1, \ldots, 15\}$ as the number of consecutive points with at least two of a player's pieces (a *block* in Backgammon). From this, we define three relation templates $Attack(PT, OP, OB, ON)$, $Expose(PT, OP, OB, ON)$, and $Block(PT, OP, OB, ON, SZ)$. Here we have a child node for each ground atom derived from these templates. As SVRRL progresses, the system keeps track of the prior over winning $P(W)$ and the conditional probability tables $P(F_i|W)$ for each ground child node. As the system learns, it may decide to join two ground features as is done above for two ground atoms of the *Expose* relation.

atoms from the state is efficient since we expect the number of positive atoms to be small (and easily identifiable) in comparison to the total number of atoms ($p \ll n$). And as we will show, it is also computationally efficient for comparison of state values.

Figure 1 shows the learning task. Given a number of trials, each involving some *finite* number of time steps, the learner is presented with a relational specification of the positive state features $\{f_1, \ldots, f_p\}$ and chooses an action according to a fixed policy. This is repeated in each trial until the terminal state is reached and the terminal reward is received. If we model the underlying process as a finite-horizon MDP with a terminal reward of 1 for success/win and 0 for failure/loss, and a discount factor $\gamma = 1$ (i.e. no discount), then it is straightforward to show that the value function w.r.t. a fixed policy is simply the conditional probability of success/winning given the state, $P(w|f)$.

Now, the question we must answer is how to estimate $P(w|f)$. Even very small RRL domains can have hundreds of ground atoms and it would be impossible to represent the exact distribution, which in its fully enumerated form would require roughly one probability entry for every distinct truth assignment to ground atoms. For 100 ground atoms, this would require approximately 2^{100} distinct probability entries, which is clearly intractable. Thus, we need to focus on a compact, factored representation of $P(w|f)$ and one common way to do this is by using a Bayes net. In our case, we specifically choose to use the naive Bayes net representation given in Figure 2 since we need only record 2 probability entries $P(f_i|w)$ and $P(f_i|\bar{w})$ for each ground atom and 1 entry $P(w)$ for the prior over winning. For our previous example of a relational domain with 100 ground atoms, we need only record 201 probability entries to approximate the value of $P(w|f)$. While this is only an approximation, we show that we can "patch up" this simple representation through structure learning. But, first we focus on how to learn the value (parameters) of this network.

Now that we can compactly represent the value function as a Bayes net, we need to efficiently learn it from data. Since it is well-known that the max-likelihood parameters of Bayes net are simply the observed frequencies for each conditional probability table (CPT), we can efficiently approximate the value function by keeping track of the observed frequencies (denoted by $\hat{P}$) for each CPT. This allows us to compute the max-likelihood value for $P(w|f)$:

$$\hat{P}(w|f) = \frac{\hat{P}(f|w)\hat{P}(w)}{\hat{P}(f)} \tag{1}$$
$$= \frac{\hat{P}(w)\prod_{i=1}^{p}\hat{P}(f_i|w)\prod_{i=p+1}^{n}\hat{P}(\bar{f}_i|w)}{\sum_{o\in\{w,\bar{w}\}}\hat{P}(o)\prod_{i=1}^{p}\hat{P}(f_i|o)\prod_{i=p+1}^{n}\hat{P}(\bar{f}_i|o)}$$

As noted previously, the number of ground atoms (and therefore children in the naive Bayes network) is very

large. Yet even in the presence of an *infinite* number of negative features, we can still efficiently determine the best next state or after-state[1] given a finite set of the positive features for each state. Based on the fact that the state f which maximizes $P(w|f)$ will also maximize the log winning odds $\log(\frac{P(w|f)}{P(\bar{w}|f)})$, we obtain the following representation of the log winning odds of a state:

$$\log \frac{P(w|f)}{P(\bar{w}|f)} = \log \frac{P(w)}{P(\bar{w})} + \sum_{i=1}^{p} \log \frac{P(f_i|w)}{P(f_i|\bar{w})} + \sum_{i=p+1}^{n} \log \frac{P(\bar{f}_i|w)}{P(\bar{f}_i|\bar{w})} \quad (2)$$

Now, if we let $C = \log \frac{P(w)}{P(\bar{w})} + \sum_{i=1}^{n} \log \frac{P(\bar{f}_i|w)}{P(\bar{f}_i|\bar{w})}$, then we can express the log winning odds of a state described by *only the set of active feature instances*:

$$\log \frac{P(w|f)}{P(\bar{w}|f)} = C + \sum_{i=1}^{p} \left(\log \frac{P(f_i|w)}{P(f_i|\bar{w})} - \log \frac{P(\bar{f}_i|w)}{P(\bar{f}_i|\bar{w})} \right) \quad (3)$$

Since C is a constant *common to all states*, we can ignore it during comparisons of log winning odds of states. Thus, even in a relational naive Bayes net with a large number of negative features, it is still possible to efficiently determine the highest-valued state.

As one final practical consideration, we typically use smoothing and non-parametric techniques (e.g. nearest-neighbor, etc...) as in Sanner *et al* (2000) to efficiently store and estimate the CPTs for all ground atoms of a relation template. This allows us to leverage natural similarity measures between attribute values to obtain more robust estimates of the CPTs.

4. Structure Learning

Given the previous framework for learning the parameters of a fixed relational naive Bayes net value function in relational reinforcement learning, we now proceed to determine how to learn structure in this value function. We restrict our attention to two types of joint feature learning which we outline next. In the following examples, note that we are looking for two ground atoms F_a and F_b that we may want to join:

Feature Attribute Augmentation (FAA) When we first initialize our relational naive Bayes net for a problem domain, we obtain a child node for every ground feature, e.g. one child node may be $Expose(5, 3, 0, 2)$. Consequently, we must use the observed frequency counts to estimate the probabilities for this child node's CPT, i.e. $P(Expose(5, 3, 0, 2)|w)$ and $P(Expose(5, 3, 0, 2)|\bar{w})$. If the relation arity is high and the number of attribute choices is large, we can expect to obtain very little data for each potential ground atom of a relation template. Aside from non-parametric learning techniques for mitigating the effects of sparse data, we choose to initially approximate the above probabilities by assuming that all relation attributes are independent. For example, we estimate $P(E(5, 3, 0, 2)|w)$ (abbreviating Expose as 'E'), by $P(E(5, \cdot, \cdot, \cdot), E(\cdot, 3, \cdot, \cdot), E(\cdot, \cdot, 0, \cdot), E(\cdot, \cdot, \cdot, 2)|w)$ where $\cdot$ in an attribute slot indicates a *don't care*. This may give us a more accurate low-variance estimate in the presence of sparse data, but as we gain more experience (i.e. data) over time, we may want to relax this approximation. For example, we can let $F_a = E(5, \cdot, \cdot, \cdot)$, $F_b = E(\cdot, 3, \cdot, \cdot)$, and attempt to determine if the join $F_{a,b} = E(5, 3, \cdot, \cdot)$ is more informative than the independent features.[2]

Feature Conjunction (FC) In contrast to feature attribute augmentation, where in some sense we are simply dealing with approximations of probabilities *within the CPTs corresponding to each Bayes net child node*, we may also want to ask whether there is any additional information gained by *joining the CPTs for two arbitrary child nodes*. For example, we could let $F_a = Expose(5, 3, 0, 2)$ and $F_b = Attack(10, 3, 0, 1)$ and ask whether the joint probability of the conjunction of both atoms, $P(F_a, F_b|W)$, is more informative than the product of the probabilities given by the naive Bayes assumption, $P(F_a|W) \cdot P(F_b|W)$.[3]

In general, a learner can use both FAA and FC structure learning techniques which we denote generically as the SVRRL algorithm, or just FAA structure learn-

[1] An after-state (Sutton & Barto, 1998) is simply the state resulting from an agent's action before any other agent, if present, has chosen its respective action.

[2] For FAA-learning we are only looking at joining the attribute probability estimates in one ground atom, but it is easy to look at joining the attribute probabilities of *each* ground atom of a relation template. This latter learning approach is more relational in nature and is what we refer to by FAA-learning.

[3] For FC-learning, we are simply looking at joining two ground atoms so it would be misleading to think of this as relational learning. However, learning arbitrary conjunctions of relations for Bayes nets proves problemantic since there is no direct correspondence between a relational join and a manipulation of the underlying ground Bayes net. While relational learning of this sort has been done for Markov random fields (MRFs), we note that learning the optimal parameters for MRFs has no closed-form solution and must be done iteratively. So, for now, full FC relational learning in the naive Bayes net framework is the subject of future research. We note that full FC relational learning would also enable the use of variable unification and quantification in relational joins to reduce the number of ground atoms. Providing the RRL agent with such an expressive relational-learning space is one of our ultimate research goals.

ing which we denote as FAA-SVRRL. For whatever algorithm is chosen, the learner must maintain distributions for each individual feature $P(F_i|W)$ and potential FAA and FC joint feature instances $P(F_a, F_b|W)$. This requires a quadratic amount of work in the number of active features during max-likelihood parameter updating of the relational naive Bayes network. If this proves to be too computationally intensive for generic SVRRL, then FAA-SVRRL should be used.

Given the probabilities for our joint feature estimates, *our goal is to add relational structure to the network so that it maximizes the log-likelihood of the joint probability of the Bayes net.* Thus, given the current network structure, our goal is to ask which two feature atoms F_a and F_b to greedily combine via the FAA or FC methods of structure learning. We let V_{all} denote the set of all Bayes net binary variables $\{W, F_1, \ldots, F_p, F_{p+1}, \ldots, F_n\}$ and let $\vec{x} \in V_{all}$ be shorthand for the set of instances $\vec{x} \in \{W \times F_1 \times \cdots \times F_p \times F_{p+1} \times \cdots \times F_n\}$. Let $N(\vec{x})$ be the number of times that state instantiation $\vec{x}$ has occurred in the data. We express the log-likelihood of the naive Bayes net with parameters θ and M data samples D as the following:

$$l(\theta|D) = \sum_{\vec{x} \in V_{all}} N(\vec{x}) \Bigg(\log P(W) + \log P(F_a, F_b|W) + \sum_{i=1, i \notin \{a,b\}}^{n} \log P(F_i|W) \Bigg)$$

Now, given that we know the maximum likelihood parameters for this fixed naive Bayes network structure are simply the observed probabilities, and adding and subtracting the same $\log(\hat{P}(F_a|W)\hat{P}(F_b|W))$ term, we can express the maximum likelihood as the following:

$$\begin{aligned} l^*(\theta|D) &= M \sum_{\vec{x} \in V_{all}} \hat{P}(\vec{x}) \Bigg(\log \hat{P}(W) + \log \frac{\hat{P}(F_a, F_b|W)}{\hat{P}(F_a|W)\hat{P}(F_b|W)} + \sum_{i=1}^{n} \log \hat{P}(F_i|W) \Bigg) \\ &= M \sum_{W} \hat{P}(W) \log \hat{P}(W) + \\ &\quad M \sum_{c=1}^{n} \sum_{F_c, W} \hat{P}(F_c, W) \log \hat{P}(F_c|W) + \\ &\quad \sum_{F_a, F_b, W} \hat{P}(F_a, F_b, W) \log \frac{\hat{P}(F_a, F_b|W)}{\hat{P}(F_a|W)\hat{P}(F_b|W)} \\ &= M(H(W) + \sum_{i=1}^{n} H(F_i|W) + I(F_a, F_b|W)) \end{aligned}$$

Finally, we let $C = M(H(W) + \sum_{i=1}^{n} H(F_i|W))$ (M times the entropy of W plus the sum of the conditional entropy of every feature F_i in the network, which we note is constant *no matter what features F_a and F_b are chosen*). Thus, we arrive at the following pleasing result expressing the maximum log-likelihood of a relational naive Bayes network under a single feature join as a constant plus the the conditional mutual information values of those joined feature nodes, i.e. $l^*(\theta|D) = C + M \cdot I(F_a, F_b|W)$). If, as for FAA-learning, we want to look at the effects of joining multiple pairs of features, it is obvious that we need only sum the mutual information values of each pair being joined. Thus, FAA and FC-learning require local evaluations only, thereby leading to a highly efficient structure learning framework.

Since random noise almost always guarantees non-zero mutual information, we need a principled way to control the amount of structure learned. For this purpose, we choose the minimum description length principle (MDL) commonly used in Bayes net learning (Lam & Bacchus, 1994) to allow the SVRRL algorithm to balance the amount of training experience with the complexity of the network structure. At each update, we check whether to add a feature by determining if it minimizes the following MDL score where $|B|$ is the number of parameters in the naive relational Bayes network B:

$$MDL(B|D) = \frac{1}{2} log(M|B|) - l^*(\theta|D) \qquad (4)$$

Since this computation involves only a negation and constant addition to the log-likelihood, the update check to determine whether a joint feature should be added can be computed quite efficiently.

Finally, we can briefly summmarize the full SVRRL algorithm: 1) Initialize the naive Bayes network for a domain with a child for each ground atom, begin by estimating the probability for each child CPT by treating the individual relational attributes independently within each node (see FAA-learning); 2) For the current trial, execute the policy[4] to obtain data and keep track of all individual and joint feature occurrences; 3) When the MDL principle permits, augment the child CPTs (FAA) or child nodes (FC) with the feature joins which maximize the MDL score; 4) When a terminal state is reached, update all probability estimates (individual and joint) for features encountered during the trial, and goto step 2 to begin the next trial.

[4]The policy can be determined on-line as the best state w.r.t. the current value function. Although convergence is not guaranteed for such a non-stationary policy, this approach often works well in practice.

Player	Winning PCT	# Training Games
TD-Gammon 1-Ply (est)	66.0 % ± ???	1,500,000
FAA-SVRRL	51.2 % ± 0.02	5,000
Pubeval	50.0 % ± 0.00	unknown
HC-Gammon	40.0 % ± 3.46	100,000

Table 1. Asymptotic winning percentage of various Backgammon programs vs. Pubeval.

5. Empirical Evaluation

Since the SVRRL algorithm is intended to handle domains with only terminal rewards of success or failure, such an approach is appropriate for learning in goal-oriented tasks such as games where the outcome is simply a win or a loss. We choose Backgammon as a testbed for empirical evaluation since it has a rich relational feature space, a high branching factor, and is heavily stochastic. This makes it an extremely difficult game to solve via model-based techniques, making it a good candidate domain for putting SVRRL to the test.[5]

We used FAA-SVRRL as our initial implementation of a Backgammon learning agent. While we lack space to show the graphs, we note that FAA-SVRRL learns more quickly and asymptotically outperforms an algorithm using random structure learning in place of the greedy-optimal structure learning outlined previously. Table 1 gives the asymptotic performance and number of training games of the converged FAA-SVRRL learning algorithm vs. PubEval (trained linear neural net) in comparison to results obtained for an estimate of TD-Gammon 2.1 with 1-ply search (Tesauro, 1992) (expert level)[6], PubEval, and HC-Gammon (Pollack et al., 1996), a neural net learned via genetic coevolution. We note that SVRRL not only converges in the least number of training games, but achieves a commendable level of performance, coming in second only to expert-level TD-Gammon.

[5]In support of our efficiency claims, we note that our FAA-SVRRL learning algorithm completed 5000 training games in under 10 minutes of computation time on a 1 GHz Pentium III with 128 Mb of RAM. The best converged learner required only 240 features (or less than 10 Kb of RAM) to store the full non-parametric representation of the conditional probability tables. This is a reasonably compact representation of a value function for a game estimated to have over 10^{18} distinct states.

[6]The TD-Gammon 1-Ply value is estimated from (Galperin & Viola, 1998) assuming that the *Lin-3* opponent referenced in this paper performs comparably to Pubeval. This seems to be a reasonable assumption since both are reasonably strong players based on a single-layer linear neural net evaluator.

6. Concluding Remarks

In this paper, we have motivated and examined the problem of learning both structure and value in a relational reinforcement learning framework. This algorithm is not only extremely efficient, involving simple updates and no search, but by learning only the structure that maximizes the log likelihood of the relational naive Bayes net under a minimum description length framework, the computational burden on the learning agent is minimized. We have applied our FAA-SVRRL agent to the domain of Backgammon and have shown that it learns useful structure in an extremely efficient manner while achieving a commendable asymptotic performance level.

Acknowledgments

The author would like to thank Martijn van Otterlo and the anonymous reviewers for their comments and suggestions regarding earlier versions of this paper.

References

Boutilier, C., Reiter, R., & Price, B. (2001). Symbolic dynamic programming for first-order MDPs. *IJCAI-2001.*

Croonenborghs, T., Ramon, J., & Bruynooghe, M. (2004). Towards informed reinforcement learning. *ICML-2004 Workshop on Relational Reinforcement Learning.*

Dzeroski, S., de Raedt, L., , & Blockeel, H. (1998). Relational reinforcement learning. *ICML-1998.*

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. *IJCAI-1999.*

Friedman, N., & Goldszmidt, M. (1996). Building classifiers using bayesian networks. *AAAI-1996.*

Galperin, G., & Viola, P. (1998). *Machine learning for prediction and control* (Technical Report). MIT.

Hölldobler, S., & Skvortsova, O. (2004). A logic-based approach to dynamic programming. *AAAI-2004 Workshop on Learning and Planning in Markov Processes.*

Kersting, K., van Otterlo, M., & de Raedt, L. (2004). Bellman goes relational. *ICML -2004.*

Lam, W., & Bacchus, F. (1994). Learning bayesian belief networks: An approach based on the mdl principle. *Computational Intelligence*, *10*, 269–294.

Pollack, J., Blair, A., & Land, M. (1996). Coevolution of a backgammon player. *Fifth Artificial Life Conference.* Nara, Japan.

Puterman, M. (1994). *Markov Decision Processes-Discrete Stochastic Dynamic Programming.* New York, NY: John Wiley and Sons, Inc.

Sanner, S., Anderson, J. R., Lebiere, C., & Lovett, M. (2000). Achieving efficient and cognitively plausible learning in backgammon. *ICML-2000.*

Sanner, S., & Boutilier, C. (2005). Approximate linear programming for first-order mdps. *UAI-2005.*

Sutton, R. S., & Barto, A. (1998). *Reinforcement learning: An introduction.* MIT Press.

Tadepalli, P., Givan, R., & Driessens, K. (2004). Relational reinforcement learning: An overview. *ICML-2004 Workshop on Relational Reinforcement Learning.*

Tesauro, G. (1992). Practical issues in temporal difference learning. *NIPS-92.*

van Otterlo, M., & Kersting, K. (2004). Challenges for relational reinforcement learning. *ICML-04 Workshop on Relational Reinforcement Learning.*

Walker, T., Shavlik, J., & Maclin, R. (2004). Relational reinforcement learning via sampling the space of first-order conjunctive features. *ICML-04 Workshop on Relational Reinforcement Learning.*