Enabling MDP Solution Transfer by Abstraction

Abstract

We consider the problem of transferring a learned optimal policy between MDPs. We describe a method that views the task of representing the solution as a supervised learning problem. Our method determines the new abstract policy by Decision Tree learning over values of features on various levels of abstraction. We provide empirical results that show acceleration in learning due to the solution instantiation in the target problem.

1. Introduction

Recent efforts (Sherstov & Stone 2005, Gabel & Riedmiller 2005, von Hessling & Goel 2005) on transferring learned solution knowledge between Markov Decision Processes (MDPs) acknowledge the benefits of this re-use, such as the ability to handle previously unseen, similar problems. A naive approach is to transfer the learned optimal policy of the *source problem* to the *target problem* by directly mapping these problems to each other. However, this requires that every state/action pair in the target problem must have an identical counterpart in the source problem in order to assume the policy's optimality label. In this paper, we propose to generate a mapping on a higher level of abstraction between the two problems. Our goal is to reformulate the optimal policy to take on a form that is based on aspects pertaining to the entire problem class rather than merely to instances of this class. To achieve this effect, we use Decision Tree learning over environmental feature values on various levels of abstraction in order to identify expressive and appropriate features for describing the solution. If a sufficiently abstract solution to a representative problem instance is found, it can be instantiated in the new problem by classification, thus completing the policy mapping process. Using the transferred policy on new problems can significantly speed up subsequent learning.

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

2. Example Domain and Experiment Setup

The turn-based strategy game "C-Evo" (Gerlach 2005) serves as the testbed for our experiments. The agent takes the role as a player whose task is to solve one problem involving the moving of a ground unit, and to transfer solution knowledge to a similar, but unidentical target problem. In the first illustrative experiment, the agent is to steer a randomly placed land unit to the goal on map tile with number *583* by moving it to any adjacent non-water tile in the North, East, South or West in a deterministic fashion. Once the agent has learned the solution to the source problem depicted in figure 1, it is supposed to apply this knowledge on the similar target problem shown in figure 2.

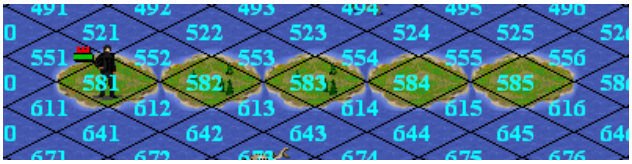

Figure 1. The illustrative source problem.

Figure 2. The illustrative target problem

For reasons that become apparent in subsequent sections we select the Policy Iteration algorithm (Sutton & Barto 1998) to solve the source problem. It requires the knowledge engineer to create the agent's mental model of the problem - an MDP defined by a set of

states $\{s\}$, actions $\{a\}$, a transition model *T(s, a, s')* as well as terminality and reward values *R(s)* for every state. While states and actions are usually enumerated explicitly, we take a feature-based perspective on this process: the engineer selects environmental features which distinguish between elements in the state space. The same is true for the elements of the action space, and we call all these features *defining features*. In our scenario, it is reasonable to select the *Map Tile Number* on which the unit is located as a criterion to distinguish between states. Similarly, the motion's *Compass Direction* is an intuitive choice for distinguishing actions. Positive reward is only obtained when the unit reaches the goal tile. Given these definitions, the agent can explore the environment and *solve* the source problem by assigning every state-action pair a boolean optimality value, where every state has exactly one optimal action that maximizes the expected discounted cumulative reward. In this paper, we focus on fully observable, infinite-horizon and discounted MDPs in discrete-valued domains.

3. Transferring Abstract Solutions between MDPs

3.1. Recording Additional Feature Values

We view both states and actions as having a set of features associated with them. While *state features* describe static aspects of the agent's situation, *action features* specify dynamic aspects - the perceived changes in environmental values when executing a particular action. In that sense, we endow the agent with a redundant knowledge representation that acknowledges the fact that states and actions may have aspects on various levels of abstraction. We assume that the agent can perceive or infer these additional values of so-called *non-defining features*. Instead of influencing the process of solving the MDP like defining features do, non-defining features facilitate the knowledge transfer between similar MDPs. All feature values for both states and actions are recorded every time the agent executes an action during the exploration of the environment, and some examples of those recordings, or *snapshots*, are shown in table 1. The four columns are headed by the the names of the four features, consisting of two state features followed by two action features. The three snapshots themselves are described by the values of defining features (boldfaced) and some manually selected non-defining features. While defining features may describe primarily problem-instance specific aspects of the environment, non-defining features should be designed to state expressions that are typical to the particular problem class (here: moving to the goal tile). It is up to the knowledge engineer to come up with a vocabulary of non-defining features that might "make a difference" for the entire problem class. In the simple move-to-goal scenario, the change in distance between unit and goal tile is an obvious choice.

Table 1. Three exemplary snapshots.

Map Tile No.	Relative Position ToGoal	**Compass Direction**	Change Distance ToGoal
581	WestOf	**GoWest**	MoveAway
584	EastOf	**GoNorth**	DontChange
584	EastOf	**GoWest**	MoveCloser

3.2. Abstracting Solution Representations using Decision Tree Learning

Decision Trees can be used for abstracting the solution representation based on the vocabulary of the recorded feature values. As stated above, solving the source MDP results in a boolean optimality label for each state-action pair. As every snapshot is a subset of only one state-action pair, it is possible to assign exactly one optimality value to every snapshot in the source problem (1:n mapping). This allows to associate alternative *explanations* (i.e. set of features) with the optimality labels. Within the space of possible explanations of the optimal policy, it is desirable that this reformulation step produces a form which is applicable to similar MDPs in order to support policy transfer. The goal is to find abstract explanations that can be transferred from source to target problem. This problem can be posed as a classification task to select a set of features that can sufficiently predict the optimality label. In this paper, we use Quinlan's ID3 algorithm with *Gain Ratio* (Quinlan 1993) as the ranking criterion, which supports our goal of simple explanations. As opposed to the commonly used Information Gain criterion, Gain Ratio discourages susceptible distinctions by features with many uniformly distributed values. In addition to that, a cost function discourages features whose values are only present in the source but not the target problem. As a result, the Decision Tree tends to incorporate those features whose values in the target problem have identical counterparts in the source problem. The subsequent section describes how this enables the instantiation of the solution. But first, we illustrate how this ranking criterion affects the solution representation for our exemplary scenario.

Table 2. An exemplary subset of labeled snapshots in the source problem.

Map Tile No.	Relative Position ToGoal	**Compass Direction**	Change Distance ToGoal	Optimality Label
581	WestOf	**GoEast**	MoveCloser	OPT.
581	WestOf	**GoWest**	MoveAway	SUBOPT.
581	WestOf	**GoNorth**	DontChange	SUBOPT.
581	WestOf	**GoSouth**	DontChange	SUBOPT.

training data is the set of all snapshots in the source problem, labeled according to their degree of optimality. A subset of the scenario's source problem dataset is depicted in table 2. The resulting Decision Tree reads like "whatever state the agent is in, moving closer to the goal is the (only) optimal action". For our example scenario, this solution representation shows that this modified ID3 algorithm suits the intended purpose for a variety of reasons. First of all, it is independent of the original MDP representation involving map tile numbers and compass directions. Secondly, the solution is concise not only in that it includes a few expressive features (e.g. *ChangeDistanceToGoal* is better than the combination of *MapTileNumber* and *CompassDirection*), but also those with fewer values (e.g. *RelativePositionToGoal* is preferred over *MapTileNumber*). Last but not least, the cost function demotes problem-instance specific features such as *MapTileNumber*, because the range overlap of the perceived feature values in source and target problem is very small. Thus, although lacking the guarantee of global optimality, our ranking criterion tends to prefer simple and expressive explanations that are common to both source and target problem.

3.3. A note on Relational Reinforcement Learning

The learning of abstract solution representations bears some similarity to the field of Relational Reinforcement Learning (RRL), so a short note on this relationship is in order. RRL is interested in generalization over objects and relations (Kaelbling et al. 2001, Van Otterlo and Kersting 2004). Originally, Dzeroski, De Raedt and Blockeel (1998) adopted an inductive approach to RRL, in which a relational regression tree is used to learn an abstract Q-function from the sampled traces. More recently, Guestrin et al (2003) have used a probabilistic model-based approach which models an abstraction over the relational MDP and uses samples to learn parameters of the models.

While RRL approaches learn a solution for MDPs whose states and actions are described by objects and relations, our work determines an optimal policy for propositional MDPs and subsequently converts it into a more abstract and perhaps relational representation.

3.4. Instantiating Abstracted Solutions in Other MDPs

To complete the policy transfer process, each state/action pair in the target MDP needs to obtain the proper optimality label from the Decision Tree. Thus, the set of snapshots is recorded in the target problem and each element is classified by its degree of optimality. The degree of overlap between the feature values in the Decision Tree and in the set of snapshots is one of the factors determining transfer success - in case the Decision Tree's feature does not contain the value in a snapshot a random optimality value is assigned to the snapshot. A Nearest Neighbor re-discretization approach can be employed for non-nominal features to reduce the steep labeling accuracy drop-off for small differences. The set of labeled snapshots can then be generalized to labeled state/action pairs. As this operation is an n:1 mapping, we selected majority voting to handle discrepancies in the suggested optimality values. Once a policy has been devised for the target MDP, it can be used as an initial solution approximation for the Policy Iteration algorithm. The success of the policy transfer step can then be expressed in terms how much this initial approximation speeds up the subsequent solving of the target MDP as compared to a random policy. In our illustrative target problem the instantiation step labels optimal exactly those state-action pairs which move the agent closer to the goal tile. This produces an optimal policy for the target problem - the agent can find the goal tile in an optimal manner, without being required to execute the Policy Iteration algorithm for the target MDP. Obviously, this hand-crafted scenario has prototypical results for illustrative purposes, but in the following evaluation section we show that our approach can also enable speed-up in more complicated problems.

4. Empirical Evaluation

To show the benefits of our approach in a more complex example with an indeterministic transition model, we created a combat scenario in which the agent's unit's task is to defeat a stationary enemy unit. In this scenario, the unit may move multiple times in one turn, hereby depleting the amount of movement points it has remaining for this turn, which again de-

creases the unit's strength for carrying out an attack in this turn. The enemy unit, on the other hand, remains passively at its location, waiting for the agent to attack by moving onto its map tile. This combat scenario requires abstracting the solution in order to perform knowledge transfer, because the problems are distinct in terrain and continent characteristics and the type of enemy unit. As opposed to the previous example, the source and target problem setup both additionally include the defining features *Remaining Movement Points, Combat Outcome, End Turn Now* and a multitude of non-defining features such as *Distance to Enemy*. To make the transfer problem more challenging, the environment exploration is stopped after approximately 130 attacks each for source and target problem. This allows the agent to perceive the two problems (particularly their indeterministic transition models) only to a limited extent. However, the algorithm is able to generate and instantiate an abstract solution that significantly speeds up the solving of the target attack-problem as figure 3 demonstrates. While the sum of state utilities of the random policy is only 10.69% of the global optimum, the transferred solution creates a policy that has 91.49% of the optimal policy's value. Although Policy Iteration converges in both cases to the global optimum, initializing it with the transferred solution requires only 11 iterations instead of 20 iterations in the random policy case. In order to examine the kind of abstract solution the agent might come up with for the source-attack problem, it was allowed to fully explore the environment in another run of this experiment. This enabled the subsequent abstraction process to generate a relatively simple solution that essentially says "close in on the enemy, but rest in the tile adjacent to it, in case you have only few movement points left".

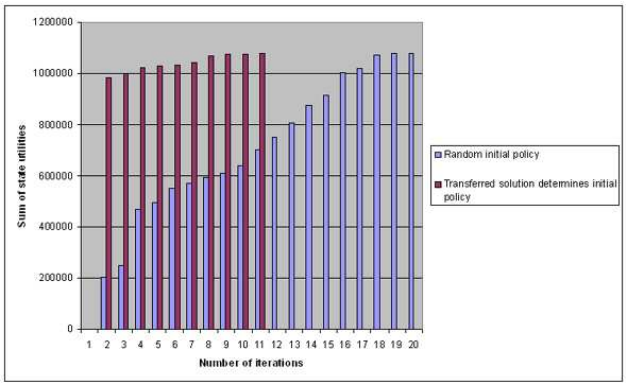

Figure 3. Progress comparison of solving the target attack-problem using a random initial policy versus utilizing the transferred solution.

5. Conclusion

The feature-based view on MDPs enables novel possibilities for solution transfer. Most importantly, features are used for a purpose, and the distinction between the two purposes of "solving the source problem" and "transferring the solution" let to the discrimination between defining and non-defining features. Our approach transfers the reformulated optimality labels between MDPs, and relies on the modified ID3 algorithm to perform the solution abstraction. We have shown that this idea can enable speed-up for problems which are dissimilar on a problem-instance level, but are akin on a problem-class level. In general, if the knowledge engineer can devise appropriate problem-class wide features in addition to formulating an adequate MDP representation, and the two problems are sufficiently similar, then our algorithm can successfully transfer the solution.

References

Dzeroski S., DeRaedt, L., & Blockeel, H. (1998). Relational reinforcement learning. *Proceedings of ICML* (pp. 136–143).

Gabel, T., & Riedmiller, R. (2005). Cbr for state function approximation in reinforcement learning. *Proceedings of ICCBR*.

Gerlach, S. http://c-evo.org/. .

Guestrin, C., K. D. G. C., & Kanodia, N. (2003). Generalizing plans to new environments in relational mdps. *Proceedings of IJCAI*.

Kaelbling, L., O. T. H. N., & Finney, S. (2001). Learning in worlds with objects. *The AAAI Spring Symposium*.

Quinlan, R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.

Sherstov, A., & Stone, P. (2005). Improving action selection in mdp's via knowledge transfer. *Proceedings of NCAI*.

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

Van Otterlo, M., & Kersting, K. (2004). Challenges for reinforcement learning. *Proceedings of ICML*.

von Hessling, A., & Goel, A. (2005). Extracting reusable cases from reinforcement learning. *Proceedings of ICCBR*.