Abstracting Complex Domains Using Modular Object-Oriented Markov Decision Processes

Shawn Squire and **Marie desJardins**
University of Maryland, Baltimore County
1000 Hilltop Circle, Baltimore, MD 21250
ssquire1, mariedj@umbc.edu

Abstract

We present an initial proposal for modular object-oriented MDPs, an extension of OO-MDPs that abstracts complex domains that are partially observable and stochastic with multiple goals. Modes reduce the curse of dimensionality by reducing the number of attributes, objects, and actions into only the features relevant for each goal. These modes may also be used as an abstracted domain to be transferred to other modes or to another domain.

Introduction

Complex domains that approach the scale of the real world have many hurdles for an intelligent agent to overcome. First, a state may be partially observable, leading to uncertainty with hundreds of state features which may be either observable or latent. The agent may also have dozens of actions available at any given opportunity, leading to a large branching factor and myriad potential paths to a solution. Finally, the agent may possess multiple goals that must be satisfied, some of which may be directly contradictory (Natarajan and Tadepalli 2005). Many, if not most, of these variables and actions can be irrelevant in the accomplishment of one particular goal. For example, an automated car equipped with multiple sensors should ignore variables such as current temperature and also ignore actions for lane changing if its goal is to navigate cars in a parking lot.

In this paper, we propose a method to automatically divide and filter an Object-Oriented Markov Decision Process into distinct modes, allowing the agent to handle smaller sub-problems that are directly related to finding the optimal solution for a specific goal. Each mode contains the most relevant features, actions, and objects for reaching a goal, ignoring irrelevant details that would increase learning time. Additionally, these modes may be easily isolated for training an agent on a specific task, or abstracted to facilitate learning transfer between modes in the same or different domains (Topin et al. 2015). One of the key benefits of using modular OO-MDPs is in domains where the designer can not fully define a set of predicates or structures for hierarchical learning.

Background

A Markov decision process (MDP) is a stochastic model that describes the states of a domain and the transition model between the states. An MDP is represented as a 4-tuple, $\{\mathcal{S}, \mathcal{A}, \mathcal{P}(\cdot,\cdot), \mathcal{R}(\cdot,\cdot)\}$, respectively consisting of a set of states, a set of actions, the transition probability function, and a reward function. The agent selects an action from state s by following a policy function, $\pi(s)$, which attempts to maximize the reward over multiple steps. Since the transition probability function and reward function may be stochastic, an MDP allows for reliability in probabilistic domains.

An Object-Oriented MDP (OO-MDP) (Diuk, Cohen, and Littman 2008) is designed to facilitate learning in domains with large state spaces. It extends the traditional MDP model by representing the domain as a set of object classes, $\mathcal{C}$, each with a set of pre-defined attributes, $Att(C)$. An object, o, belongs to a single object class and has values assigned for each attribute associated with the class. The state, s, of a domain is defined as the union of the attribute values for all objects in the domain, $s = \bigcup_{i=1}^{\mathcal{O}} \text{state}(o_i)$ where $O = \{o_1, o_2, \ldots, o_n\}$ for n objects in the domain.

This representation gives several benefits over an MDP that represents state as a single vector of features. First, it provides a capability of attribute the relevance of objects or object classes to specific goals and actions. Second, objects that are instantiated differently but with equal values (i.e., given objects o_1 and o_2 of object class c, $o_1 = o_2 \iff \forall a \in Att(c) : o_1.a = o_2.a$) are considered equivalent, so they may be handled interchangeably.

Modular Object-Oriented MDP

A *mode* in an OO-MDP is an abstracted representation of the domain for a specific goal g where a subset of *relevant* attributes, objects, and actions for that goal are considered. The relevance of an attribute, object, or action is defined as the perceived necessity of the respective value for reaching g from any state. Therefore, a mode for goal g is defined as the function $Mode(g) = \{Att(C)', \mathcal{O}', \mathcal{A}'\}$. $Att(C)' \subseteq Att(C)$ is the set of relevant attributes for g across all object classes. $\mathcal{O}'$ is an abstracted set of k relevant objects such that $\mathcal{O}' = \{o'_1, o'_2, \ldots, o'_k\}$, and o'_i is the union of abstracted attributes $Att(o_i.class)'$. Finally, $\mathcal{A}' \subseteq \mathcal{A}$ is the set of relevant actions for g.

Since the state of an OO-MDP is defined as the union of attributes for all objects, abstract state s' can be derived from $Att(C)'$ and $\mathcal{O}'$. Since multiple states may be abstracted into the same s', similar states that differ only in irrelevant variables are treated as the same state. Similarly, states that contain no relevant objects are empty states that are ignored in the mode. Therefore, the complexity of planning or learning in an abstracted domain is guaranteed to be less than or equal to the complexity of planning or learning in the grounded domain. (Note that since objects and attributes may be relevant for multiple goals and some objects and attributes may be irrelevant for all goals, modes are neither mutually exclusive nor exhaustive.)

Learning a mode begins with finding $Att(C)'$ and $\mathcal{O}'$. First, an optimal policy, π, is assumed to exist in the grounded domain for goal g. A mapping is produced between the grounded domain and all possible abstracted $Att(C)'$ and $\mathcal{O}'$ for $Mode(g)$. That is, a set of possible $Att(C)'$ and $\mathcal{O}'$ for $Mode(g)$ are generated by all possible combinations of attributes and objects. Next, a scoring mechanism is used to find the best possible mapping for a mode by finding the minimum number of attributes and objects that still lead to finding the best possible reward. Since this exhaustive method may be intractable with large state spaces, a Monte Carlo method for finding the best optimization may be used.

An action $a \in \mathcal{A}'$ exists in the mode if and only if $\exists s'_1, s'_2[P(s'_2|s'_1, a) \wedge s'_1 \neq s'_2]$. That is, an action is relevant if there exists a state in the mode in which the action may be used to reach another unique state in the mode.

Related Work

(Topin et al. 2015) provides a framework, Portable Option Discovery (POD), for creating portable mappings between different OO-MDPs. POD leverages the benefit of options in reinforcement learning (Sutton, Precup, and Singh 1999) to autonomously abstract partial policies from one domain, which can be subsequently grounded into a partial policy in a different domain. However, current implementations are limited to value iteration over tabular learning, and do not account for relevance of individual attributes for objects. Additionally, the framework provides no measure for domains with multiple and / or competing goals.

(Abel et al. 2015) uses affordances to add knowledge to an OO-MDP which prunes actions based on multiple goals. An affordance is a mapping from a predicate "precondition" and abstract goal description to a set of relevant possible actions. If the predicate is true for a given state, and the goal description matches the agent's current goal, then the affordance is "activated" and only the relevant actions are used. While this method of pruning irrelevant actions is similar to modular learning, affordances require an additional set of predicates to be defined for relations between objects in the domain; those predicates require additional design or expert knowledge. Additionally, the abstraction of states in Modular OO-MDPs allow many similar states to be handled equivalently to one another, decreasing complexity and alleviating the curse of dimensionality for learning methods such as reinforcement learning.

(Dietterich 1999) introduces the MAXQ method of hierarchical reinforcement learning to decompose a policy's value function into a set of subtasks. This method proves highly effective for dividing a complex state space for reinforcement learning into identifiable goals similar to modular OO-MDP. However, the MAXQ algorithm requires identifying the actions responsible for completing each subgoal, which requires knowledge that may not be provided by the designer. Modular OO-MDPs automatically selects the actions from the set of all actions given the relevance.

(Boutilier et al. 1995) explores a general algorithm for MDPs using temporal Bayesian networks to ascertain rules about independence for states without enumerating them. The regularities and structure of the domain can be exploited to group states that have same estimated value. However, the domains in this paper are assumed to have hidden variables and partial observability.

Conclusion

In this paper, we propose a means to simplify and abstract highly complex, stochastic, partially observable domains defined as an object-oriented MDP. This allows for reduced complexity when learning or planning in a domain, as well as facilitating existing transfer algorithms.

References

Abel, D.; Barth-Maron, G.; MacGlashan, J.; and Tellex, S. 2015. Affordance-Aware Planning.

Boutilier, C.; Dearden, R.; Goldszmidt, M.; and others. 1995. Exploiting structure in policy construction. In *IJCAI*, volume 14, 1104–1113.

Dietterich, T. G. 1999. Hierarchical reinforcement learning with the MAXQ value function decomposition. *arXiv preprint cs/9905014*.

Diuk, C.; Cohen, A.; and Littman, M. L. 2008. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, 240–247. ACM.

Natarajan, S., and Tadepalli, P. 2005. Dynamic Preferences in Multi-criteria Reinforcement Learning. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, 601–608. New York, NY, USA: ACM.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112(1):181–211.

Topin, N.; Haltmeyer, N.; Squire, S.; and Winder, J. 2015. Portable option discovery for automated learning transfer in object-oriented markov decision processes. *24th International Joint Conference on Artificial Intelligence*.