Model-Based Reinforcement Learning

Soumya Ray and Prasad Tadepalli
School of EECS, Oregon State University

July 10, 2009

1 Synonyms

Indirect Reinforcement Learning

2 Definition

Model-based Reinforcement Learning refers to learning optimal behavior indirectly by learning a model of the environment by taking actions and observing the outcomes that include the next state and the immediate reward. The models predict the outcomes of actions and are used in lieu of or in addition to interaction with the environment to learn optimal policies.

3 Motivation and Background

Reinforcement Learning (RL) refers to learning to behave optimally in a stochastic environment by taking actions and receiving rewards [1]. The environment is assumed Markovian in that there is a fixed probability of the next state given the current state and the agent's action. The agent also receives an immediate reward based on the current state and the action. Models of the next-state distribution and the immediate rewards are referred to as "action models" and, in general, are not known to the learner. The agent's goal is to take actions, observe the outcomes including rewards and next states, and learn a policy or a mapping from states to actions that optimizes some performance measure. Typically the performance measure is the expected total reward in episodic domains, and the expected average reward per time step or expected discounted total reward in infinite-horizon domains.

The theory of Markov Decision Processes (MDPs) implies that under fairly general conditions, there is a stationary policy, i.e., a time-invariant mapping from states to actions, that maximizes each of the above reward measures. Moreover, there are MDP solution algorithms, e.g., value iteration and policy iteration [2], which can be used to solve the MDP exactly given the action models. Assuming that the number of states is not exceedingly high, this suggests a straightforward approach for model-based reinforcement learning. The models can be learned by interacting with the environment by taking actions, observing the resulting states and rewards, and estimating the parameters of the action models through maximum likelihood methods. Once the models are estimated to a desired accuracy, the MDP solution algorithms can be run to learn the optimal policy.

One weakness of the above approach is that it seems to suggest that a fairly accurate model needs to be learned over the entire domain to learn a good policy. Intuitively it seems that we should be able to get by without learning highly accurate models for suboptimal actions. A related problem is that the method does not suggest how best to explore the domain, i.e., which states to visit and which actions to execute to quickly learn an optimal policy. A third issue is one of scaling these methods, including model learning, to very large state spaces with billions of states.

The remaining sections outline some of the approaches explored in the literature to solve these problems.

4 Theory and Methods

Systems that solve MDPs using value-based methods can take advantage of models in at least two ways. First, with an accurate model, they can use offline learning algorithms that directly solve the modeled MDPs. Second, in an online setting, the can use the estimated models to guide exploration and action selection. Algorithms have been developed that exploit MDP models in each of these ways. We describe some such algorithms below.

Common approaches to solving MDPs given a model are value or policy iteration [3, 1]. In these approaches, the algorithms start with a randomly initialized value function or policy. In value iteration, the algorithm loops through the state space, updating the value estimates of each state using Bellman backups, until convergence. In policy iteration, the algorithm calculates the value of the current policy and then loops through the state space, updating the current policy to be greedy with respect to the backed up values. This is repeated until the policy converges.

When the model is unknown but being estimated as learning progresses, we could use value or policy iteration in the inner loop: after updating our current model estimate using an observed sample from the MDP, we could solve the updated MDP offline and take an action based on the solution. However, this is computationally very expensive. To gain efficiency, algorithms such as Adaptive Real-time Dynamic Programming (ARTDP) [4] and Dyna [5] perform one or more Bellman updates using the action models after each real-world action and corresponding update to either a state-based or state-action based value function. Other approaches, such as prioritized sweeping [6] and Queue-Dyna [7], have considered the problem of intelligently choosing which states to update after each iteration.

A different approach to discovering the optimal policy is to use algorithms that calculate the gradient of the utility measure with respect to some adjustable policy parameters. The standard policy gradient approaches that estimate the gradient from immediate rewards suffer from high variance due to the stochasticity of the domain and the policy. Wang and Dietterich propose a model-based policy gradient algorithm that alleviates this problem by learning a partial model of the domain [8]. The partial model is solved to yield the value function of the current policy and the expected number of visits to each state, which are then used to derive the gradient of the policy in closed form. The authors observe that their approach converges in many fewer exploratory steps compared to model-free policy gradient algorithms in a number of domains including a real-world resource-controlled scheduling problem.

One of the many challenges in model-based reinforcement learning is that of efficient exploration of the MDP to learn the dynamics and the rewards. In the "Explicit Explore and Exploit" or E^3 algorithm, the agent explicitly decides between exploiting the known part of the MDP and optimally trying to reach the unknown part of the MDP (exploration) [9]. During exploration, it uses the idea of "balanced wandering," where the least executed action in the current state is preferred until all actions are executed a certain number of times. In contrast, the R-Max algorithm implicitly chooses between exploration and exploitation by using the principle of "optimism under uncertainty" [10]. The idea here is to initialize the model parameters optimistically so that all unexplored actions in all states are assumed to reach a fictitious state that yields maximum possible reward from then on regardless of which action is taken. Both these algorithms are guaranteed to find models whose approximate policies are close to the optimal with high probability in time polynomial in the size and mixing time of the MDP.

Since a table-based representation of the model is impractical in large state spaces, efficient model-based learning depends on compact parameterization of the models. Dynamic Bayesian networks offer an elegant way to represent action models compactly by exploiting conditional independence relationships, and have been shown to lead to fast convergence of models [11]. In some cases, choosing an appropriate prior distribution over model parameters can be important and lead to faster learning. In recent work, the acquisition of a model prior has been investigated in a multi-task setting [12]. In this work, the authors use a hierarchical Bayesian model to represent classes of MDPs. Given observations from a new MDP, the algorithm uses the model to infer an appropriate class (creating a new class if none seem appropriate). It then uses the distributions governing the inferred class as a prior to guide exploration in the new MDP. This approach is able to significantly

speed up the rate of convergence to optimal policy as more environments are seen.

In recent work, researchers have explored the possibility of using approximate models coupled with policy gradient approaches to solve hard control problems [13]. In this work, the approximate model is used to calculate gradient directions for the policy parameters. When searching for an improved policy, however, the real environment is used to calculate the utility of each intermediate policy. Observations from the environment are also used to update the approximate model. The authors show that their approach improves upon model-based algorithms that only used the approximate model while learning.

5 Applications

In this section, we describe some domains where model-based reinforcement learning has been applied.

Model-based approaches have been commonly used in RL systems that play two-player games [14, 15]. In such systems, the model corresponds to legal moves in the game. Such models are easy to acquire and can be used to perform lookahead search on the game tree. For example, the TD-LEAF(λ) system [15] uses the values at the leaves of an expanded game tree at some depth to update the estimate of the value of the current state. After playing a few hundred chess games, this algorithm was able to reach the play level of a US Master.

Model-based reinforcement learning has been used in a spoken dialog system [16]. In this application, a dialog is modeled as a turn-based process, where at each step the system speaks a phrase and records certain observations about the response and possibly receives a reward. The system estimates a model from the observations and rewards and uses value iteration to compute optimal policies for the estimated MDP. The authors show empirically that, among other things, the system finds sensible policies and is able to model situations that involve "distress features" that indicate the dialog is in trouble.

It was shown that in complex real-world control tasks such as pendulum swing-up task on a real anthropomorphic robot arm, model-based learning is very effective in learning from demonstrations [17]. A model is learned from the human demonstration of pendulum swing-up, and an optimal policy is computed using a standard approach in control theory called linear quadratic regulation. Direct imitation of the human policy would not work in this case due to the small differences in the tasks and the imperfections of the robot controller. On the other hand, model-based learning was able to learn successfully from short demonstrations of pendulum swing up. However, on a more difficult swing-up task that includes pumping, model-based learning by itself was inadequate due to the inaccuracies in the model. They obtained better results by combining model-based learning with learning appropriate task parameters such as the desired pendulum target angle at an intermediate stage where the pendulum was at its highest point.

In more recent work, model-based RL has been used to learn to fly a remote-controlled helicopter [18]. Again, the use of model-free approaches is very difficult, because almost any random exploratory action results in an undesirable outcome (i.e., a crash). To learn a model, the system bootstraps from a trajectory that is observed by watching an expert human fly the desired maneuvers. In each step, the system learns a model with the observed trajectory and finds a controller that works in simulation with the model. This controller is then tried with the real helicopter. If it fails to work well, the model is refined with the new observations and the process is repeated. Using this approach, the system is able to learn a controller that can repeatedly perform complex aerobatic maneuvers, such as flips and rolls.

Model-based RL has also been applied to other domains, such as robot juggling [19] and job-shop scheduling [20]. Some work has also been done that compares model-free and model-based RL methods [21]. From their experiments, the authors conclude that, for systems with reasonably simple dynamics, model-based RL is more data efficient, finds better policies, and handles changing goals better than model-free methods. On the other hand, model-based methods are subject to errors due to inaccurate model representations.

6 Future Directions

Representing and learning richer actions models for stochastic domains that involve relations, numeric quantities, and parallel, hierarchical, and durative actions is a challenging open problem. Efficient derivation of optimal policies from such rich representations of action models is another problem that is partially explored in symbolic dynamic programming. Constructing good policy languages appropriate for a given action model or class of models might be useful to accelerate learning near-optimal policies for MDPs.

7 See Also

Efficient Exploration in Reinforcement Learning, Symbolic Dynamic Programming, Adaptive Real-time Dynamic Programming, Bayesian Reinforcement Learning, Autonomous Helicopter Flight Using Reinforcement Learning, and Model-free Reinforcement Learning.

References

[1] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)

[2] Puterman, M.L.: Markov Decision Processes: Discrete Dynamic Stochastic Programming. John Wiley (1994)

[3] Kaelbling, L.P., Littman, M.L., Moore, A.P.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research **4** (1996) 237–285

[4] Barto, A.G., Bradtke, S.J., Singh, S.P.: Learning to act using real-time dynamic programming. Artificial Intelligence **72**(1) (1995) 81–138

[5] Sutton, R.S.: Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In: Proceedings of the Seventh International Conference on Machine Learning, Morgan Kaufmann (1990) 216–224

[6] Moore, A.W., Atkeson, C.G.: Prioritized sweeping: Reinforcement learning with less data and less real time. Machine Learning **13** (1993) 103–130

[7] Peng, J., Williams, R.J.: Efficient learning and planning within the dyna framework. Adaptive Behavior **1**(4) (1993) 437–454

[8] Wang, X., Dietterich, T.G.: Model-based policy gradient reinforcement learning. In: Proceedings of the 20th International Conference on Machine Learning. (2003) 776–783

[9] Kearns, M., Singh, S.: Near-optimal reinforcement learning in polynomial time. Machine Learning **49**(2/3) (2002) 209–232

[10] Brafman, R.I., Tennenholtz, M.: R-MAX — a general polynomial time algorithm for near-optimal reinforcement learning. Journal of Machine Learning Research **2** (2002) 213–231

[11] Tadepalli, P., Ok, D.: Model-based average-reward reinforcement learning. Artificial Intelligence **100** (1998) 177–224

[12] Wilson, A., Fern, A., Ray, S., Tadepalli, P.: Multi-task reinforcement learning: A hierarchical bayesian approach. In: Proceedings of the 24th International Conference on Machine Learning, Omni Press (2007) 1015–1022

[13] Abbeel, P., Quigley, M., Ng, A.Y.: Using inaccurate models in reinforcement learning. In: Proceedings of the 23rd International Conference on Machine Learning. (2006) 1–8

[14] Tesauro, G.: Temporal difference learning and TD-Gammon. Communications of the ACM **38**(3) (1995) 58–68

[15] Baxter, J., Tridgell, A., Weaver, L.: TDLeaf(λ): Combining temporal difference learning with game-tree search. In: Proceedings of the Ninth Australian Conference on Neural Networks (ACNN'98). (1998) 168–172

[16] Singh, S., Kearns, M., Litman, D., Walker, M.: Reinforcement learning for spoken dialogue systems. In: Advances in Neural Information Processing Systems. Volume 11. (1999) 956–962

[17] Atkeson, C.G., Schaal, S.: Robot learning from demonstration. In: Proceedings of the Fourteenth International Conference on Machine Learning. Volume 4., Morgan Kaufmann (1997) 12–20

[18] Abbeel, P., Coates, A., Quigley, M., Ng, A.Y.: An application of reinforcement learning to aerobatic helicopter flight. In: Advances in Neural Information Processing Systems. Volume 19., Cambridge, MA, MIT Press (2007) 1–8

[19] Schaal, S., Atkeson, C.G.: Robot juggling: implementation of memory-based learning. IEEE Control Systems Magazine **14**(1) (1994) 57–71

[20] Zhang, W., Dietterich, T.G.: A reinforcement learning approach to job-shop scheduling. In: Proceedings of the International Joint Conference on Artificial Intellience. (1995) 1114–1120

[21] Atkeson, C.G., Santamaria, J.C.: A comparison of direct and model-based reinforcement learning. In: Proceedings of the International Conference on Robotics and Automation. (1997) 20–25