Policy Iteration on Continuous Domains using Rapidly-exploring Random Trees

S S Manimaran Balaraman Ravindran

Reinforcement Learning(RL) deals with learning optimal control strategies through interactions with the world. One of the major hindrances in extending existing RL techniques to real world scenarios, is their inability to deal with continuous valued variables. Techniques such as variable discretization and using function approximators in place of lookup tables in the original discrete algorithms exist. Least Squares Policy Iteration(LSPI) [4] is an offline method that uses a set of experiences and linear function approximators. Although LSPI is a fundamentally sound algorithm, the question of how exactly the experiences should be generated exists. Attempts have been made to address this problem of online exploration in large domains by combining LSPI with R-MAX, a technique that does efficient exploration in discrete domains[6]. Several implementation issues arise in translating this to continuous domains, such as tuning exploration parameters and generalizing state visitation counters. We propose an algorithm - RRTPI that combines ideas from the domain of continuous path planning to clearly provide a method to generate sufficiently representative samples and ensure exploration. We briefly introduce rapidly-exploring random trees(RRTs)[5] and their properties that allow us to use them for solving optimal control problems in deterministic continuous domains. A RRT is grown as follows: (i) Sample a state s from a given state space S according to a sampling distribution f_s (ii) Calculate nearest node $s_{nearest}$ in the tree as measured by a distance metric $|\ .\ |_V$ (iii) Extend the tree from $s_{nearest}$ towards s as given by the $|\ .\ |_V$ and repeat till required size is reached. RRTs are sampling based methods that have been used to solve for feasible trajectories from a given starting to goal state in kinodynamic spaces, which is a problem that is at least PSPACE-hard. RRTs have good space filling properties and possess *asymptotic completeness*, i.e. they eventually find a solution if one exists. RRT* an extended version of RRTs have been developed that also guarantee *asymptotic optimality*, i.e. they eventually converge to an optimal solution as more samples are drawn [3]. These methods tend to perform poorly in domains that are underactuated and underpowered. It has been shown that RRTs explore the space efficiently only when the distance metric $|\ .\ |_V$ reflects the true cost to go[2], or the value function as they are known in RL literature, and not the euclidean distance. Several techniques have been developed that use LQR based approximations to the value function for efficient exploration. We use fitted policy evaluation on the samples generated by RRTs to estimate the value function. We then generate close to optimal trajectories by biasing the RRT using this estimated value function and then iteratively improves this by using the new samples to re-estimate the value function and so on. Given a model that allows us to sample transitions and rewards, we can build a sample set $\mathcal{D}$ consisting of (s, s', r) where s and s' are the current and next state and r is the reward. A procedure $FPE(.)$,

given a set of samples, returns the value function estimate V using some fitted policy evaluation technique. The distance metric is then defined as $\mid x - y \mid_V = V(x) - V(y)$. An outline of the RRTPI algorithm is as follows.

1. Use some initial distance function and repeat steps 2 and 3 till convergence.
2. Construct a RRT and a dataset $\mathcal{D}$ using $\mid . \mid_V$
3. Evaluate the value function V using *FPE*

This method has been implemented on the continuous gridworld, mountain car(under-powered) and acrobot(under-actuated) domains. It is interesting to note that steps 2 and 3 resemble policy evaluation and policy improvement, and hence the name RRT based Policy Iteration. Thus RRTPI is a sound algorithm that gives specific ways to generate samples such that it has guaranteed exploration properties. Moreover it can be shown that the generated samples are fast-mixing using properties of random geometric graphs along the lines of Karaman et al.[3]. This forms a basis for an analysis on sample complexity as provided by Antos et. al.[1], where the samples need to be sufficiently "representative". The entire algorithm is free from learning rates and exploration control parameters. Compared to LQR based approximations, our method does not need access to a closed form expression of the state dynamics and is more generic as it does not make assumptions of linearity of the state transition and quadracity of the reward function. Thus it can work on complex underactuated and underpowered domains.

References

[1] A. Antos, C. Szepesvari, and R. Munos, *Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path*, Machine Learning, 71 (2006), pp. 89–129.

[2] P. Cheng and S. M. Lavalle, *Reducing metric sensitivity in randomized trajectory design*, in Proceedings of IEEE International Conference on Intelligent Robots and Systems, 2001, pp. 43–48.

[3] S. Karaman and E. Frazzoli, *Sampling-based algorithms for optimal motion planning*, International Journal of Robotics Research, 30 (2011), pp. 846–894.

[4] M. G. Lagoudakis and R. Parr, *Least-squares policy iteration*, Journal of Machine Learning Research, 4 (2003), pp. 1107–1149.

[5] S. M. Lavalle, *Rapidly-exploring random trees: A new tool for path planning*, tech. rep., Computer Science Dept, Iowa State University, 1998.

[6] L. Li, M. L. Littman, and C. R. Mansley, *Online exploration in least-squares policy iteration*, in Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2, 2009, pp. 733–739.