Basis Function Adaptation with Least-Squares Policy Iteration for Continuous MDPs

Kenneth Marino Georgia Institute of Technology

My interest in robotics really begins with the media coverage of the DARPA driverless car Grand Challenge. Tasked with developing a completely autonomous car to navigate 240 km in the Mojave Desert in 2004, the most successful vehicle, Carnegie Mellon's Sandstorm, managed to travel 12 km before being caught on an embankment. Just one year later, five teams completed a similar course and just three years later six teams completed the DARPA "Urban Challenge," following traffic laws and avoiding other vehicles. It was the coverage of these events, while I was in middle school and high school, that got me seriously interested in robotics and artificial intelligence.

A little over a year ago, I started working with Professor Charles Isbell in the Lab for Interactive Artificial Intelligence on machine learning approaches to artificial intelligence. In particular I began working on state space representation, developing high-level features for the game of Go with deep networks, and on reinforcement learning.

Reinforcement learning is essentially a set of algorithms and methods to solve Markov Decision Processes (MDPs). In an MDP, the environment is modeled by a set of states S (such as coordinates on a 2D surface) and a set of possible actions A in those states [1]. There are also rewards R(s) that the agent (a robot for instance) receives for arriving in certain states. The goal of reinforcement learning is to determine a policy π, an action for every state $a = \pi(s)$, to maximize the rewards received by the agent [3]. Two commonly used methods for this are value iteration, which attempts to either learn the true value of a state V(s) or of state action pairs Q(s,a), and policy iteration which learns the value of a state $V_\pi(s)$ under policy π and then updates the policy by choosing the action in each state that maximizes the expected value of the next state under the policy $V_\pi(s')$ [2].

The formulation of MDPs lends itself particularly well to robotics problems. The set of states S is usually a continuous space (depending on the problem) and the action space A can be either a vector representing the robot's path or the control signals that determine the robot's motion. Reinforcement learning allows robots to learn optimal behavior through trail-and-error without explicitly detailing the solution to a particular problem [3]. The applications for RL in range from control to path-planning.

The usefulness of reinforcement learning for robotics comes with its own set of challenges. The first is that almost all domains of interest are continuous. In robot path-planning for example, you have might want to find the fastest path to a particular area on a 2D map with obstacles and hindrances. The robot can be anywhere within the map, not just at discrete x and y locations. The actions are also continuous: either a control signal for the robot's actuators or motion if we are using the assumption that the robot is a point particle. Control problems also usually involve continuous control and input signals.

One common way of avoiding the continuous problem is to discretize the state and action spaces– that is divide the space into a grid and call each grid a discrete state [1]. There are merits and drawbacks to this approach, but the biggest problem is that it leaves you with a potentially large number of states, which can be impractical. An alternative method is to use a function approximation method to learn the MDP. Rather than learn an exact value function V(s) or Q(s,a) for each state or each state-action pair, you approximate these functions.

One particularly efficient such approximation is the linear approximation architecture. This model maps states or state-action pairs to a low-dimensional feature vector using basis functions $\varphi(s)$ and uses a feature vector ω to map this to the value function: $V(s) = \omega * \varphi(s)$ [2]. Linear Architectures are particularly good for RL because they guarantee convergence and performance under certain conditions. Another advantage of linear architectures is that they learn efficiently from a limited number of experiences. This is particularly helpful for robotics applications as it is often more difficult to collect real data [3].

Basis Function Adaptation with Least-Squares Policy Iteration for Continuous MDPs

Kenneth Marino Georgia Institute of Technology

A current topic in linear architectures is the choice of basis functions. Most architectures use a class of functions called Radial Basis Functions (RBFs). The key detail of RBFs is that the value of the RBF function depends only on the distance of the input from the center c. RBFs are parameterized by the center c and the width W which roughly corresponds to the size of the function or how close a value has to be to the center to activate the RBF. The most common procedure is to choose a large number of RBFs and place them evenly over the state space. While this often works in simpler domains, for many domains it is not sufficient. There may be obstacles that divide the placed RBFs, causing large changes to the value function in a small space.

My research is focused on developing the best basis functions to describe a state space. Menache, Mannor, Shimkin (2005) [2] showed that using cross entropy (CE), the initial RBF basis functions can be optimized to provide the most accurate value function V(s) for a fixed policy. After reproducing their results, I determined that the natural course of action was to optimize the RBFs to determine the most accurate Q(s,a) function using an algorithm called Least Squares Policy Iteration (LSPI) [4]. This would allow one to determine the optimal policy in a domain: the goal of reinforcement learning.

The problems in using reinforcement learning for robotics problems make my approach potentially very helpful. I have shown in my experiments so far that the method works for smaller discrete domains, so the next step is to use the approach to solve a path-planning problem in robotics. The set of states S and A are both continuous, and the robot has to learn the best action a in any arbitrary state s. Because the RBFs are already able to take in continuous values, this approach is already suited to robotic path planning. Rather than having to discretize the state space, this function approximation allows you to extend the algorithm into continuous state and action spaces [1]. The basis function adaptation also allows for a more compact state representation, thus lessening the problems of dimensionality and sampling.

Sebastian Thrun, who worked on the Stanford robotic car for the 2005 and 2007 Grand Challenges as well as the Google Driverless Car, describes his motivation working in robotics in very personal terms. When he was 18, his best friend died in a car accident. I share his hope and optimism that research being done now in robotics and artificial intelligence bring about a better world. Over 20,000 Americans die in car crashes each year. Autonomous driving has the potential to prevent car accidents.

The development of computers and the Internet has been the last great technological revolution. I believe that robotics could be the next great revolution. Intelligent automatization could revolutionize every industry, dropping prices for consumers and raising the standard of living everywhere. Intelligent robotics for healthcare and for the care of the elderly could be a possible solution to the problems of an aging society and could replace humans on dangerous search and rescue missions. I believe that the development of more intelligent machines will be the next great boon for mankind.

References

[1] H. Hasselt, M. Wierning, "Reinforcement Learning in Continuous Action Spaces," in *Proceedings of the 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2007, pp. 272-279.

[2] I. Menache, S. Mannor, N. Shimkin, "Basis Function Adaptation in Temporal Difference Reinforcement Learning," *Annals of Operations Research*, vol. 134, pp. 215-238, 2005.

[3] J. Kober, J. Bagnell, J. Peters, "Reinforcement Learning in Robotics," *International Journal of Robotics Research*, 2013.

[4] M. Lagoudakis, R. Parr, "Least-Squares Policy Iteration," *Journal of Machine Learning Research*, vol. 4, pp.1107-1149, 2003.