A Tutorial on Linear Function Approximators for Dynamic Programming and Reinforcement Learning

Alborz Geramifard
Massachusetts Institute of Technology

Thomas J. Walsh
Massachusetts Institute of Technology

Stefanie Tellex
Massachusetts Institute of Technology

Girish Chowdhary
Massachusetts Institute of Technology

Nicholas Roy
Massachusetts Institute of Technology

Jonathan P. How
Massachusetts Institute of Technology
Editorial Scope

Topics

Foundations and Trends® in Machine Learning publishes survey and tutorial articles on the theory, algorithms and applications of machine learning, including the following topics:

- Adaptive control and signal processing
- Applications and case studies
- Behavioral, cognitive, and neural learning
- Bayesian learning
- Classification and prediction
- Clustering
- Data mining
- Dimensionality reduction
- Evaluation
- Game theoretic learning
- Graphical models
- Independent component analysis
- Inductive logic programming
- Kernel methods
- Markov chain Monte Carlo
- Model choice
- Nonparametric methods
- Online learning
- Optimization
- Reinforcement learning
- Relational learning
- Robustness
- Spectral methods
- Statistical learning theory
- Variational inference
- Visualization

Information for Librarians

Foundations and Trends® in Machine Learning, 2013, Volume 6, 4 issues. ISSN paper version 1935-8237. ISSN online version 1935-8245. Also available as a combined paper and online subscription.
A Tutorial on Linear Function Approximators for Dynamic Programming and Reinforcement Learning

Alborz Geramifard
MIT LIDS
agf@mit.edu

Thomas J. Walsh
MIT LIDS
twalsh@mit.edu

Stefanie Tellex
MIT CSAIL
stefie10@csail.mit.edu

Girish Chowdhary
MIT LIDS
girishc@mit.edu

Nicholas Roy
MIT CSAIL
nickroy@mit.edu

Jonathan P. How
MIT LIDS
jhow@mit.edu
# Contents

1 Introduction 2

2 Dynamic Programming and Reinforcement Learning 6
   2.1 Markov Decision Processes (MDPs) . . . . . . . . . . . 6
   2.2 MDP Solvers at a Glance . . . . . . . . . . . . . . . . 9
   2.3 Dynamic Programming . . . . . . . . . . . . . . . . . 10
   2.4 Approximate Dynamic Programming . . . . . . . . . . 14
   2.5 Trajectory Based Value Iteration . . . . . . . . . . . 18
   2.6 Approximate Dynamic Programming in Matrix Format . 20
   2.7 Reinforcement Learning . . . . . . . . . . . . . . . . . 31
   2.8 Big Picture . . . . . . . . . . . . . . . . . . . . . . . 38

3 Representations 40
   3.1 Tabular Representation . . . . . . . . . . . . . . . . . 42
   3.2 Fixed Sparse Representation . . . . . . . . . . . . . . 43
   3.3 Radial Basis Functions (RBFs) . . . . . . . . . . . . . 44
   3.4 Incremental Feature Dependency Discovery (iFDD) . . . 45

4 Empirical Results 47
   4.1 Algorithms . . . . . . . . . . . . . . . . . . . . . . . 48
   4.2 Domain Descriptions . . . . . . . . . . . . . . . . . . 48
4.3 Simulation Framework ........................................ 52
4.4 Simulation Results ........................................... 55
4.5 Discussion .................................................. 62

5 Summary ....................................................... 65

Acknowledgements ............................................... 68

References ....................................................... 69
Abstract

A Markov Decision Process (MDP) is a natural framework for formulating sequential decision-making problems under uncertainty. In recent years, researchers have greatly advanced algorithms for learning and acting in MDPs. This article reviews such algorithms, beginning with well-known dynamic programming methods for solving MDPs such as policy iteration and value iteration, then describes approximate dynamic programming methods such as trajectory based value iteration, and finally moves to reinforcement learning methods such as Q-Learning, SARSA, and least-squares policy iteration. We describe algorithms in a unified framework, giving pseudocode together with memory and iteration complexity analysis for each. Empirical evaluations of these techniques with four representations across four domains, provide insight into how these algorithms perform with various feature sets in terms of running time and performance.
Designing agents to act near-optimally in stochastic sequential domains is a challenging problem that has been studied in a variety of settings. When the domain is known, analytical techniques such as dynamic programming (DP) [Bellman 1957] are often used to find optimal policies for the agent. When the domain is initially unknown, reinforcement learning (RL) [Sutton and Barto 1998] is a popular technique for training agents to act optimally based on their experiences in the world. However, in much of the literature on these topics, small-scale environments were used to verify solutions. For example, the famous taxi problem has only 500 states [Dietterich 2000]. This contrasts with recent success stories in domains previously considered unsailable, such as 9×9 Go [Silver et al. 2012], a game with approximately $10^{38}$ states. An important factor in creating solutions for such large-scale problems is the use of linear function approximation [Sutton 1996, Silver et al. 2012, Geramifard et al. 2011]. This approximation technique allows the long-term utility (value) of policies to be represented in a low-dimensional form, dramatically decreasing the number of parameters that need to be learned or stored. This tutorial provides practical guidance for researchers seeking to extend DP and RL techniques to larger domains through linear value function approximation. We introduce DP and RL techniques in a unified frame-
work and conduct experiments in domains with sizes up to $\sim 150$ million state-action pairs.

Sequential decision making problems with full observability of the states are often cast as Markov Decision Processes (MDPs) \cite{Puterman}. An MDP consists of a set of states, set of actions available to an agent, rewards earned in each state, and a model for transitioning to a new state given the current state and the action taken by the agent. Ignoring computational limitations, an agent with full knowledge of the MDP can compute an optimal policy that maximizes some function of its expected cumulative reward (which is often referred to as the expected return \cite{SuttonBarto}). This process is known as planning. In the case where the MDP is unknown, reinforcement learning agents learn to take optimal actions over time merely based on interacting with the world.

A central component for many algorithms that plan or learn to act in an MDP is a value function, which captures the long term expected return of a policy for every possible state. The construction of a value function is one of the few common components shared by many planners and the many forms of so-called value-based RL methods. In the planning context, where the underlying MDP is known to the agent, the value of a state can be expressed recursively based on the value of successor states, enabling dynamic programming algorithms \cite{Bellman} to iteratively estimate the value function. If the underlying model is unknown, value-based reinforcement learning methods estimate the value function based on observed state transitions and rewards. However, in either case, maintaining and manipulating the value of every state (i.e., a tabular representation) is not feasible in large or continuous domains. In order to tackle practical problems with such large state-action spaces, a value function representation is needed that 1) does not require computation or memory proportional to the size of the number of states, and 2) generalizes learned values from data across states (i.e., each new piece of data may change the value of more than one state).

One approach that satisfies these goals is to use linear function approximation to estimate the value function. Specifically, the full set of states is
projected into a lower dimensional space where the value function is represented as a linear function. This representational technique has succeeded at finding good policies for problems with high dimensional state-spaces such as simulated soccer [Stone et al., 2005b] and Go [Silver et al., 2012]. This tutorial reviews the use of linear function approximation algorithms for decision making under uncertainty in DP and RL algorithms. We begin with classical DP methods for exact planning in decision problems, such as policy iteration and value iteration. Next, we describe approximate dynamic programming methods with linear value function approximation and “trajectory based” evaluations for practical planning in large state spaces. Finally, in the RL setting, we discuss learning algorithms that can utilize linear function approximation, namely: SARSA, Q-learning, and Least-Squares policy iteration. Throughout, we highlight the trade-offs between computation, memory complexity, and accuracy that underlie algorithms in these families.

In Chapter 3, we provide a more concrete overview of practical linear function approximation from the literature and discuss several methods for creating linear bases. We then give a thorough empirical comparison of the various algorithms described in the theoretical section paired with each of these representations. The algorithms are evaluated in multiple domains, several of which have state spaces that render tabular representations intractable. For instance, one of the domains we examine, Persistent Search and Track (PST), involves control of multiple unmanned aerial vehicles in a complex environment. The large number of properties for each robot (fuel level, location, etc.) leads to over 150 million state-action pairs. We show that the linear function approximation techniques described in this tutorial provide tractable solutions for this otherwise unwieldy domain. For our experiments, we used the RLPy framework [Geramifard et al., 2013a] which allows the reproduction of our empirical results.

There are many existing textbooks and reviews of reinforcement learning [Bertsekas and Tsitsiklis, 1996, Szepesvári, 2010, Buşoniu et al., 2010, Gosavi, 2009, Kaelbling et al., 1996, Sutton and Barto, 1998]. This tutorial differentiates itself by providing a narrower focus on the use of linear value function approximation and introducing many DP and RL techniques in a unified framework, where each algorithm is derived from the general concept of policy evaluation/improvement (shown in Figure 2.1). Also, our extensive
empirical evaluation covers a wider range of domains, representations, and algorithms than previous studies. The lessons from these experiments provide a guide to practitioners as they apply DP and RL methods to their own large-scale (and perhaps hitherto intractable) domains.


References


References


References


B. Scherrer. Should one compute the temporal difference fix point or minimize the bellman residual? the unified oblique projection view. In International Conference on Machine Learning (ICML), 2010.


