Stable Linear Approximations to Dynamic Programming for Stochastic Control Problems with Local Transitions

Benjamin Van Roy and John N. Tsitsiklis
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139
e-mail: bvr@mit.edu, jnt@mit.edu

Abstract

We consider the solution to large stochastic control problems by means of methods that rely on compact representations and a variant of the value iteration algorithm to compute approximate cost-to-go functions. While such methods are known to be unstable in general, we identify a new class of problems for which convergence, as well as graceful error bounds, are guaranteed. This class involves linear parameterizations of the cost-to-go function together with an assumption that the dynamic programming operator is a contraction with respect to the Euclidean norm when applied to functions in the parameterized class. We provide a special case where this assumption is satisfied, which relies on the locality of transitions in a state space. Other cases will be discussed in a full length version of this paper.

1 INTRODUCTION

Neural networks are well established in the domains of pattern recognition and function approximation, where their properties and training algorithms have been well studied. Recently, however, there have been some successful applications of neural networks in a totally different context – that of sequential decision making under uncertainty (stochastic control).

Stochastic control problems have been studied extensively in the operations research and control theory literature for a long time, using the methodology of dynamic programming [Bertsekas, 1995]. In dynamic programming, the most important object is the *cost-to-go (or value) function*, which evaluates the expected future

cost to be incurred, as a function of the current state of a system. Such functions can be used to guide control decisions.

Dynamic programming provides a variety of methods for computing cost–to–go functions. Unfortunately, dynamic programming is computationally intractable in the context of many stochastic control problems that arise in practice. This is because a cost–to–go value is computed and stored for each state, and due to the curse of dimensionality, the number of states grows exponentially with the number of variables involved.

Due to the limited applicability of dynamic programming, practitioners often rely on *ad hoc* heuristic strategies when dealing with stochastic control problems. Several recent success stories – most notably, the celebrated Backgammon player of Tesauro (1992) – suggest that neural networks can help in overcoming this limitation. In these applications, neural networks are used as compact representations that approximate cost–to–go functions using far fewer parameters than states. This approach offers the possibility of a systematic and practical methodology for addressing complex stochastic control problems.

Despite the success of neural networks in dynamic programming, the algorithms used to tune parameters are poorly understood. Even when used to tune the parameters of linear approximators, algorithms employed in practice can be unstable [Boyan and Moore, 1995; Gordon, 1995; Tsitsiklis and Van Roy, 1994].

Some recent research has focused on establishing classes of algorithms and compact representation that guarantee stability and graceful error bounds. Tsitsiklis and Van Roy (1994) prove results involving algorithms that employ feature extraction and interpolative architectures. Gordon (1995) proves similar results concerning a closely related class of compact representations called *averagers*. However, there remains a huge gap between these simple approximation schemes that guarantee reasonable behavior and the complex neural network architectures employed in practice.

In this paper, we motivate an algorithm for tuning the parameters of linear compact representations, prove its convergence when used in conjunction with a class of approximation architectures, and establish error bounds. Such architectures are not captured by previous results. However, the results in this paper rely on additional assumptions. In particular, we restrict attention to Markov decision problems for which the dynamic programming operator is a contraction with respect to the Euclidean norm when applied to functions in the parameterized class. Though this assumption on the combination of compact representation and Markov decision problem appears restrictive, it is actually satisfied by several cases of practical interest. In this paper, we discuss one special case which employs affine approximations over a state space, and relies on the locality of transitions. Other cases will be discussed in a full length version of this paper.

2 MARKOV DECISION PROBLEMS

We consider infinite horizon, discounted Markov decision problems defined on a finite state space $S = \{1, \ldots, n\}$ [Bertsekas, 1995]. For every state $i \in S$, there is a finite set $U(i)$ of possible control actions, and for each pair $i, j \in S$ of states and control action $u \in U(i)$ there is a probability $p_{ij}(u)$ of a transition from state i to state j given that action u is applied. Furthermore, for every state i and control action $u \in U(i)$, there is a random variable c_{iu} which represents the one-stage cost if action u is applied at state i.

Let $\beta \in [0, 1)$ be a discount factor. Since the state spaces we consider in this paper

are finite, we choose to think of cost–to–go functions mapping states to cost–to–go values in terms of cost-to-go vectors whose components are the cost-to-go values of various states. The optimal cost-to-go vector $V^* \in \Re^n$ is the unique solution to Bellman's equation:

$$V_i^* = \min_{u \in U(i)} \Big(E[c_{iu}] + \beta \sum_{j \in S} p_{ij}(u) V_j^* \Big), \qquad \forall i \in S. \tag{1}$$

If the optimal cost–to–go vector is known, optimal decisions can be made at any state i as follows:

$$u^* = \arg \min_{u \in U(i)} \Big(E[c_{iu}] + \beta \sum_{j \in S} p_{ij}(u) V_j^* \Big), \qquad \forall i \in S.$$

There are several algorithms for computing V^* but we only discuss the value iteration algorithm which forms the basis of the approximation algorithm to be considered later on. We start with some notation. We define the *dynamic programming operator* as the mapping $T : \Re^n \mapsto \Re^n$ with components $T_i : \Re^n \mapsto \Re$ defined by

$$T_i(V) = \min_{u \in U(i)} \Big(E[c_{iu}] + \beta \sum_{j \in S} p_{ij}(u) V_j \Big), \qquad \forall i \in S. \tag{2}$$

It is well known and easy to prove that T is a maximum norm contraction. In particular,

$$\|T(V) - T(V')\|_\infty \leq \beta \|V - V'\|_\infty, \qquad \forall V, V' \in \Re^n.$$

The value iteration algorithm is described by

$$V(t+1) = T(V(t)),$$

where $V(0)$ is an arbitrary vector in $\Re^n$ used to initialize the algorithm. It is easy to see that the sequence $\{V(t)\}$ converges to V^*, since T is a contraction.

3 APPROXIMATIONS TO DYNAMIC PROGRAMMING

Classical dynamic programming algorithms such as value iteration require that we maintain and update a vector V of dimension n. This is essentially impossible when n is extremely large, as is the norm in practical applications. We set out to overcome this limitation by using compact representations to approximate cost-to-go vectors. In this section, we develop a formal framework for compact representations, describe an algorithm for tuning the parameters of linear compact representations, and prove a theorem concerning the convergence properties of this algorithm.

3.1 COMPACT REPRESENTATIONS

A *compact representation* (or *approximation architecture*) can be thought of as a scheme for recording a high–dimensional cost-to-go vector $V \in \Re^n$ using a lower–dimensional parameter vector $w \in \Re^m$ ($m \ll n$). Such a scheme can be described by a mapping $\tilde{V} : \Re^m \mapsto \Re^n$ which to any given parameter vector $w \in \Re^m$ associates a cost–to–go vector $\tilde{V}(w)$. In particular, each component $\tilde{V}_i(w)$ of the mapping is the ith component of a cost-to-go vector represented by the parameter vector w. Note that, although we may wish to represent an arbitrary vector $V \in \Re^n$, such a scheme allows for exact representation only of those vectors V which happen to lie in the range of $\tilde{V}$.

In this paper, we are concerned exclusively with linear compact representations of the form $\tilde{V}(w) = Mw$, where $M \in \Re^{n \times m}$ is a fixed matrix representing our choice of approximation architecture. In particular, we have $\tilde{V}_i(w) = M_i w$, where M_i (a row vector) is the ith row of the matrix M.

3.2 A STOCHASTIC APPROXIMATION SCHEME

Once an appropriate compact representation is chosen, the next step is to generate a parameter vector w such that $\tilde{V}(w)$ approximates V^*. One possible objective is to minimize squared error of the form $\|Mw - V^*\|_2^2$. If we were given a fixed set of N samples $\{(i_1, V^*_{i_1}), (i_2, V^*_{i_2}), ..., (i_N, V^*_{i_N})\}$ of an optimal cost-to-go vector V^*, it seems natural to choose a parameter vector w that minimizes $\sum_{j=1}^{N}(M_{i_j}w - V^*_{i_j})^2$. On the other hand, if we can actively sample as many data pairs as we want, one at a time, we might consider an iterative algorithm which generates a sequence of parameter vectors $\{w(t)\}$ that converges to the desired parameter vector. One such algorithm works as follows: choose an initial guess $w(0)$, then for each $t \in \{0, 1, \ldots\}$ sample a state $i(t)$ from a uniform distribution over the state space and apply the iteration

$$w(t+1) = w(t) - \alpha(t)\Big(M_{i(t)}w(t) - V^*_{i(t)}\Big)M^T_{i(t)}, \tag{3}$$

where $\{\alpha(t)\}$ is a sequence of diminishing step sizes and the superscript T denotes a transpose. Such an approximation scheme conforms to the spirit of traditional function approximation – the algorithm is the common stochastic gradient descent method. However, as discussed in the introduction, we do not have access to such samples of the optimal cost-to-go vector. We therefore need more sophisticated methods for tuning parameters.

One possibility involves the use of an algorithm similar to that of Equation 3, replacing samples of $V^*_{i(t)}$ with $T_{i(t)}(V(t))$. This might be justified by the fact that $T(V)$ can be viewed as an improved approximation to V^*, relative to V. The modified algorithm takes on the form

$$w(t+1) = w(t) - \alpha(t)\Big(M_{i(t)}w(t) - T_{i(t)}(Mw(t))\Big)M^T_{i(t)}. \tag{4}$$

Intuitively, at each time t this algorithm treats $T(Mw(t))$ as a "target" and takes a steepest descent step as if the goal were to find a w that would minimize $\|Mw - T(Mw(t))\|_2^2$. Such an algorithm is closely related to the TD(0) algorithm of Sutton (1988). Unfortunately, as pointed out in Tsitsiklis and Van Roy (1994), such a scheme can produce a diverging sequence $\{w(t)\}$ of weight vectors even when there exists a parameter vector w^* that makes the approximation error $V^* - Mw^*$ zero at every state. However, as we will show in the remainder of this paper, under certain assumptions, such an algorithm converges.

3.3 MAIN CONVERGENCE RESULT

Our first assumption concerning the step size sequence $\{\alpha(t)\}$ is standard to stochastic approximation and is required for the upcoming theorem.

Assumption 1 *Each step size $\alpha(t)$ is chosen prior to the generation of $i(t)$, and the sequence satisfies $\sum_{t=0}^{\infty}\alpha(t) = \infty$ and $\sum_{t=0}^{\infty}\alpha^2(t) < \infty$.*

Our second assumption requires that $T : \Re^n \mapsto \Re^n$ be a contraction with respect to the Euclidean norm, at least when it operates on value functions that can be represented in the form Mw, for some w. This assumption is not always satisfied, but it appears to hold in some situations of interest, one of which is to be discussed in Section 4.

Assumption 2 *There exists some $\beta' \in [0, 1)$ such that*

$$\|T(Mw) - T(Mw')\|_2 \leq \beta'\|Mw - Mw'\|_2, \qquad \forall w, w' \in \Re^m.$$

The following theorem characterizes the stability and error bounds associated with the algorithm when the Markov decision problem satisfies the necessary criteria.

Theorem 1 *Let Assumptions 1 and 2 hold, and assume that M has full column rank. Let $\Pi = M(M^TM)^{-1}M^T$ denote the projection matrix onto the subspace $\mathcal{X} = \{Mw | w \in \Re^m\}$. Then,*
(a) With probability 1, the sequence $w(t)$ converges to w^*, the unique vector that solves:

$$Mw^* = \Pi T(Mw^*).$$

(b) Let V^* be the optimal cost-to-go vector. The following error bound holds:

$$\|Mw^* - V^*\|_2 \leq \frac{(1+\beta)\sqrt{n}}{1-\beta'}\|\Pi V^* - V^*\|_\infty.$$

3.4 OVERVIEW OF PROOF

Due to space limitations, we only provide an overview of the proof of Theorem 1.

Let $s : \Re^m \mapsto \Re^m$ be defined by

$$s(w) = E\left[\Big(M_i w - T_i(Mw(t))\Big)M_i^T\right],$$

where the expectation is taken over i uniformly distributed among $\{1, \ldots, n\}$. Hence,

$$E[w(t+1)|w(t), \alpha(t)] = w(t) - \alpha(t)s(w(t)),$$

where the expectation is taken over $i(t)$. We can rewrite s as

$$s(w) = \frac{1}{n}\Big(M^TMw - M^TT(Mw)\Big),$$

and it can be thought of as a vector field over $\Re^m$. If the sequence $\{w(t)\}$ converges to some w, then $s(w)$ must be zero, and we have

$$\begin{aligned} M^TMw &= M^TT(Mw) \\ Mw &= \Pi T(Mw). \end{aligned}$$

Note that

$$\|\Pi T(Mw) - \Pi T(Mw')\|_2 \leq \beta'\|Mw - Mw'\|_2, \qquad \forall w, w' \in \Re^m,$$

due to Assumption 2 and the fact that projection is a nonexpansion of the Euclidean norm. It follows that $\Pi T(\cdot)$ has a unique fixed point $w^* \in \Re^m$, and this point uniquely satisfies

$$Mw^* = \Pi T(Mw^*).$$

We can further establish the desired error bound:

$$\begin{aligned} \|Mw^* - V^*\|_2 &\leq \|Mw^* - \Pi T(\Pi V^*)\|_2 + \|\Pi T(\Pi V^*) - \Pi V^*\|_2 + \|\Pi V^* - V^*\|_2 \\ &\leq \beta'\|Mw^* - V^*\|_2 + \|T(\Pi V^*) - V^*\|_2 + \|\Pi V^* - V^*\|_2 \\ &\leq \beta'\|Mw^* - V^*\|_2 + (1+\beta)\sqrt{n}\|\Pi V^* - V^*\|_\infty, \end{aligned}$$

and it follows that

$$\|Mw^* - V^*\|_2 \leq \frac{(1+\beta)\sqrt{n}}{1-\beta'}\|\Pi V^* - V^*\|_\infty.$$

Consider the potential function $U(w) = \frac{1}{2}\|w - w^*\|_2^2$. We will establish that $(\nabla U(w))^T s(w) \geq \gamma U(w)$, for some $\gamma > 0$, and we are therefore dealing with a

"pseudogradient algorithm" whose convergence follows from standard results on stochastic approximation [Polyak and Tsypkin, 1972]. This is done as follows:

$$\begin{aligned}(\nabla U(w))^T s(w) &= \frac{1}{n}\Big(w - w^*\Big)^T M^T \Big(Mw - T(Mw)\Big) \\ &= \frac{1}{n}\Big(w - w^*\Big)^T M^T \Big(Mw - \Pi T(Mw) - (I - \Pi)T(Mw)\Big) \\ &= \frac{1}{n}\Big(Mw - Mw^*\Big)^T \Big(Mw - \Pi T(Mw)\Big),\end{aligned}$$

where the last equality follows because $M^T\Pi = M^T$. Using the contraction assumption on T and the nonexpansion property of projection mappings, we have

$$\begin{aligned}\|\Pi T(Mw) - Mw^*\|_2 &= \|\Pi T(Mw) - \Pi T(Mw^*)\|_2 \\ &\leq \beta' \|Mw - Mw^*\|_2,\end{aligned}$$

and applying the Cauchy–Schwartz inequality, we obtain

$$\begin{aligned}(\nabla U(w))^T s(w) &\geq \frac{1}{n}(\|Mw - Mw^*\|_2^2 - \|Mw - Mw^*\|_2 \|Mw^* - \Pi T(Mw)\|_2) \\ &\geq \frac{1}{n}(1 - \beta')\|Mw - Mw^*\|_2^2.\end{aligned}$$

Since M has full column rank, it follows that $(\nabla U(w))^T s(w) \geq \gamma U(w)$, for some fixed $\gamma > 0$, and the proof is complete.

4 EXAMPLE: LOCAL TRANSITIONS ON GRIDS

Theorem 1 leads us to the next question: are there some interesting cases for which Assumption 2 is satisfied? We describe a particular example here that relies on properties of Markov decision problems that naturally arise in some practical situations.

When we encounter real Markov decision problems we often interpret the states in some meaningful way, associating more information with a state than an index value. For example, in the context of a queuing network, where each state is one possible queue configuration, we might think of the state as a vector in which each component records the current length of a particular queue in the network. Hence, if there are d queues and each queue can hold up to k customers, our state space is a finite grid $\mathcal{Z}_k^d$ (i.e., the set of vectors with integer components each in the range $\{0, \ldots, k-1\}$).

Consider a state space where each state $i \in \{1, \ldots, n\}$ is associated to a point $x^i \in \mathcal{Z}_k^d$ $(n = k^d)$, as in the queuing example. We might expect that individual transitions between states in such a state space are local. That is, if we are at a state x^i the next visited state x^j is probably close to x^i in terms of Euclidean distance. For instance, we would not expect the configuration of a queuing network to change drastically in a second. This is because one customer is served at a time so a queue that is full can not suddenly become empty.

Note that the number of states in a state space of the form $\mathcal{Z}_k^d$ grows exponentially with d. Consequently, classical dynamic programming algorithms such as value iteration quickly become impractical. To efficiently generate an approximation to the cost–to–go vector, we might consider tuning the parameters $w \in \Re^d$ and $a \in \Re$ of an affine approximation $\tilde{V}_i(w, a) = w^T x^i + a$ using the algorithm presented in the previous section. It is possible to show that, under the following assumption

concerning the state space topology and locality of transitions, Assumption 2 holds with $\beta' = \sqrt{\beta^2 + \frac{6}{k-3}}$, and thus Theorem 1 characterizes convergence properties of the algorithm.

Assumption 3 *The Markov decision problem has state space* $S = \{1, \ldots, k^d\}$, *and each state* i *is uniquely associated with a vector* $x^i \in \mathcal{Z}_k^d$ *with* $k \geq 6(1-\beta^2)^{-1} + 3$. *Any pair* $x^i, x^j \in \mathcal{Z}_k^d$ *of consecutively visited states either are identical or have exactly one unequal component, which differs by one.*

While this assumption may seem restrictive, it is only one example. There are many more candidate examples, involving other approximation architectures and particular classes of Markov decision problems, which are currently under investigation.

5 CONCLUSIONS

We have proven a new theorem that establishes convergence properties of an algorithm for generating linear approximations to cost–to–go functions for dynamic programming. This theorem applies whenever the dynamic programming operator for a Markov decision problem is a contraction with respect to the Euclidean norm when applied to vectors in the parameterized class. In this paper, we have described one example in which such a condition holds. More examples of practical interest will be discussed in a forthcoming full length version of this paper.

Acknowledgments

This research was supported by the NSF under grant ECS 9216531, by EPRI under contract 8030-10, and by the ARO.

References

Bertsekas, D. P. (1995) *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, MA.

Boyan, J. A. & Moore, A. W. (1995) Generalization in Reinforcement Learning: Safely Approximating the Value Function. In J. D. Cowan, G. Tesauro, and D. Touretzky, editors, *Advances in Neural Information Processing Systems 7.* Morgan Kaufmann.

Gordon, G. J. (1995) Stable Function Approximation in Dynamic Programming. Technical Report: CMU-CS-95-103, Carnegie Mellon University.

Polyak, B. T. & Tsypkin, Y. Z., (1972) Pseudogradient Adaptation and Training Algorithms. *Avtomatika i Telemekhanika*, 3:45-68.

Sutton, R. S. (1988) Learning to Predict by the Method of Temporal Differences. *Machine Learning*, 3:9-44.

Tesauro, G. (1992) Practical Issues in Temporal Difference Learning. *Machine Learning*, 8:257-277.

Tsitsiklis, J. & Van Roy, B. (1994) Feature–Based Methods for Large Scale Dynamic Programming. Technical Report: LIDS-P-2277, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology. Also to appear in *Machine Learning.*