MS&E339/EE337B Approximate Dynamic Programming Lecture 1 - 3/31/2004

Introduction

Lecturer: Ben Van Roy *Scribe: Ciamac Moallemi*

1 Stochastic Systems

In this class, we study stochastic systems. A stochastic system consists of 3 components:

- *State* x_t - the underlying state of the system.
- *Noise* w_t - random disturbance from the environment.
- *Decision* u_t - control decision.

The state x_t evolves over time according to the equation

$$x_{t+1} = F(x_t, u_t, w_t), \tag{1}$$

for some deterministic function $F(\cdot, \cdot, \cdot)$.

A policy π is a sequence of mappings $(\mu_0, \mu_1, \ldots)$, where, at time t, the mapping μ_t determines the control decision u_t to take given that the state is x_t. The objective in this course is to determine policies that are "good."

2 Examples

In order to understand our framework, we consider a number of examples which can be viewed in the context of stochastic systems.

2.1 Tetris

Tetris is a computer game in which falling pieces must be positioned on a two-dimensional grid (see Figure 1). The goal is to form contiguous rows of blocks, at which point such rows are deleted. Falling pieces are selected at random and the game is over when the height of the pieces exceeds the height of the board. Tetris can be viewed as a stochastic as follows:

$$\text{state } x_t = \begin{pmatrix} \text{board configuration} \\ \text{shape of current falling piece} \end{pmatrix},$$

$$\text{decision } u_t = (\text{where to put the falling piece, i.e. orientation/translation}),$$

$$\text{noise } w_t = (\text{next falling piece}).$$

2.2 Linear Systems

One common example of a stochastic system system is a linear system. Here, $x_t \in \mathbb{R}^n$, $u_t \in \mathbb{R}^m$, and $w_t \in \mathbb{R}^n$ are vectors, and the system evolves according to the linear update

$$x_{t+1} = Ax_t + Bu_t + w_t,$$

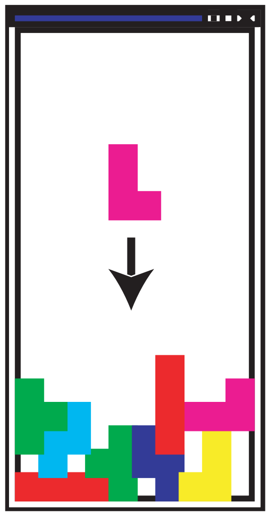


Figure 1: The computer game Tetris.

where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$. In such instances, linear policies are of particular interest. In a linear policy, the decision is a linear function of current state, in other words,

$$u_t = Kx_t,$$

for some $K \in \mathbb{R}^{m \times n}$. Aside from their simplicity, linear policies are interesting because in many cases the optimal policy will be a linear policy.

2.3 Dynamic Asset Allocation

Consider the following simplified dynamic asset allocation problem. At every time t, an individual can choose to invest portions of his wealth in either:

1. Money market - constant rate of return ρ (that is, ρ dollars returned at time $t+1$ for every one dollar invested at time t).
2. Mutual fund - rate of return r_t which is a random variable.

In the context of stochastic systems, we have

$$\text{state } x_t = (\text{wealth at time } t) \in \mathbb{R}_+,$$

$$\text{decision } u_t = (\text{fraction of wealth to invest in mutual fund at time } t) \in [0,1],$$

$$\text{noise } r_t = (\text{mutual fund rate of return at time } t) \in \mathbb{R}_+.$$

The system evolves according to the equation

$$x_{t+1} = \rho(1-u_t)x_t + r_t u_t x_t.$$

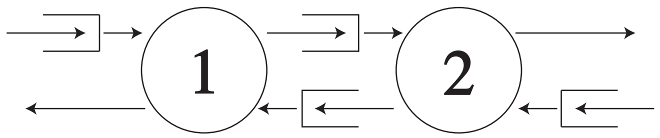


Figure 2: A queuing network.

2.4 Queuing

Consider the queuing network in Figure 2. Here, there two classes of products to be manufactured by servers 1 and 2. For the first class of products, produced along the top path, raw materials enter from the left, are first processed by server 1, then processed by server 2, and finally exit on the right. The second class of products are produced in an analogous fashion right-to-left along the bottom path. There are four queues in the system which buffer materials in different stages that are waiting for a busy server. Raw materials arrive to the input buffers at random times. For simplicity, assume that each server can serve one item per time step. Hence, if a server has items waiting in both of its buffers, it must make an allocation decision of which buffer to process an item from. This queuing network can be viewed as a stochastic system as follows:

$$\text{state } x_t = (\text{queue lengths})\,,$$

$$\text{decision } u_t = (\text{allocation of servers 1 and 2})\,,$$

$$\text{noise } w_t = (\text{random arrivals of raw materials})\,.$$

3 Transition Probabilities

In general, stochastic systems may evolve in discrete or continuous time, and their state/decison/noise variables may take values in finite or continuous state spaces. For simplicity, we will initially focus on discrete-time, finite-state stochastic systems. Note that the ideas we will present can be extended to other frameworks as well.

In particular, assume that

$$x_t \in S = \{1, \ldots, n\},$$

$$u_t \in U(x_t),$$

where for each $x \in S$, $U(x)$ is a finite set of available decisions or actions given state $x_t = x$. For $x, y \in S$, and $u \in U(x)$, we can define transition probabilities

$$P_{xy}(u) = \mathbb{P}\left\{ x_{t+1} = y \middle|\, x_t = x, u_t = u \right\}. \tag{2}$$

Transition probabilities provide an alternative to the state evolution equation (1) to specify the random structure of a stochastic system. In particular, we can relate (1) and (2) by

$$P_{xy}(u) = \mathbb{P}\left\{ F(x, u, w) = y \right\}.$$

Both viewpoints will be useful in the study of stochastic systems.

4 Optimality Criteria

We would like to determine the "best" policy for a given stochastic system. In order for this question to be well-posed, we must define a criteria for optimality. The most general formulation that may be considered is

$$\min_{\pi} \mathbb{E}\left[g(x_0, u_0, x_1, u_0, \ldots)\right]. \tag{3}$$

Here, we attempt to minimize the expected value of a cost function $g(\cdot)$, which is a function of the entire sample path $(x_0, u_0, x_1, u_1, \ldots)$. While this criteria is the most general, the optimization problem (3) will not be tractable in general.

A more tractable criteria is to assume that the cost function in (3) decomposes additively across time, resulting in

$$\min_{\pi} \mathbb{E}\left[\sum_{t=0}^{N} g(x_t, u_t, x_{t+1})\right]. \tag{4}$$

Here, $g(x_t, u_u, x_{t+1})$ is the cost of selecting decision u_t at time t, resulting in next state x_{t+1}, given current state x_t. Note that the evolution of the system is considered only up to time N. Hence, (4) is called the *finite horizon* problem.

In some systems, there is no natural time horizon N to pick. For these systems, the *total cost* objective may be more appropriate,

$$\min_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} g(x_t, u_t, x_{t+1})\right]. \tag{5}$$

The total cost objective is fine for systems that eventually terminate and generate zero cost thereafter. For systems that generate ongoing cost, however, the total cost will be infinite and will not provide a useful mechanism for differenting amongst policies. For such systems, one alternative is the *discounted cost* objective,

$$\min_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \alpha^t g(x_t, u_t, x_{t+1})\right]. \tag{6}$$

Here, $\alpha \in (0, 1)$ is a *discount factor* which weighs the relative contribution of costs in the near and long-term future to the total cost. The discounting also guarantees that the sum in (6) is finite.

Another alternative for the infinite horizon case is the *average cost* criteria,

$$\min_{\pi} \limsup_{N\to\infty} \frac{1}{N} \mathbb{E}\left[\sum_{t=0}^{N-1} g(x_t, u_t, x_{t+1})\right]. \tag{7}$$

Here, we are looking at the long-term average expected cost. The averaging ensures finite values even when the system generates ongoing cost. However, if the system eventually terminates the average cost will be zero, hence the average cost objective is not appropriate and the total cost objective (5) should be used.

Given an optimality criteria, we need a way to determine the optimal policy. One easy way out is to select at time t the decision u_t which minimizes the expected cost over the next time step, in other words

$$\min_{u_t} \mathbb{E}\left[g(x_t, u_t, x_{t+1}) \mid x_t\right].$$

This strategy, while simple to implement, does not factor in the effect of the current decision u_t on future costs. Hence, it is *myopic* and not generally optimal.

An alternative is to select the current decision u_t by the optimization problem

$$\min_{u_t} \mathbb{E}\left[g(x_t, u_t, x_{t+1}) + J(x_{t+1}) \mid x_t\right].$$

Here, the value $J(x_{t+1})$ captures the future cost that is incurred as a result of being in state x_{t+1} at time $t+1$. The field of dynamic programming provides methods for choosing a value function $J(\cdot)$ so as to result in an optimal policy.

In practical problems, number of possible values that x_t can take is enormous. For these problems, computing the value function $J(\cdot)$ by dynamic programming or even storing such a $J(\cdot)$ is infeasible. We will focus on approximate methods to find good policies. In particular, there are two broad classes of such methods:

1. *Value function approximation.* In a spirit similar to regression, we will consider a parameterized family of value functions $J_\theta(\cdot)$, where the number of parameters θ is tractable. We will develop methods for tuning the value of θ to result in good policies.

2. *Policy gradient methods.* Here, the class of possible policies will be parameterized, and the parameter will similarly be tuned to yield a good policy.

MS&E339/EE337B Approximate Dynamic Programming Lecture 2 - 4/5/2004

Dynamic Programming Overview

Lecturer: Ben Van Roy *Scribe: Vassil Chatalbashev and Randy Cogill*

1 Finite Horizon Problems

We distinguish between *finite horizon problems*, where the cost accumulates over a finite number of stages, say N, and *infinite horizon problems*, where the cost accumulates indefinitely. First we consider the finite horizon problems.
The dynamic system evolves according to the following mapping:

$$x_{t+1} = f(x_t, \mu_t(x_t), w_t) \in S,\ t = 0, 1, \cdots, N-1.$$

where $|S| < \infty$, $\mu_t(x_t)$ is the decision at time t. A policy π is $\pi = \{\mu_0, \mu_1, ...\mu_{N-1}\}$, where $\mu_t(x) \in U(x)$, and $U(x)$ is the set of all legal decisions in state x. Our goal is to minimize the expected cost:

$$\min_\pi \mathbb{E}\left[\sum_{t=0}^{N-1} g(x_t, \mu_t(x_t), x_{t+1}) \middle| x_0\right].$$

We will now define a useful *cost-to-go* function $J_k(x_k)$, which represents the remaining cost to be incurred starting at $t = k$ and assuming that at time k we are in state x_k:

$$J_k(x_k) = \min_{\mu_k, \cdots \mu_{N-1}} \mathbb{E}\left[\sum_{t=k}^{N-1} g(x_t, \mu_t(x_t), x_{t+1}) \middle| x_k\right]$$

Then, our goal would be to calculate the cost-to-go function $J_0(x_0)$, which is the overall cost to be incurred in the finite horizon of N steps. We now state a simple theorem, which will be useful later:

Theorem 1

$$J_k(x) = \min_{u \in U(x)} \sum_{y \in S} p_{xy}(u)(g(x, u, y) + J_{k+1}(y))$$

where $J_N(x) \equiv 0$. *Also, a policy* $\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$ *is optimal if and only if*

$$\mu_k^*(x) \in \arg\min_{u \in U(x)} \sum_{y \in S} p_{xy}(u)(g(x, u, y) + J_{k+1}(y)).$$

The proof is easy, simply use the definition of $J_k(x)$ and split up the minimization and the expectation.

2 Discounted Dynamic Programming

2.1 Discounted Finite Horizon Problems

Before talking about discounted infinite horizon problems, let's look at discounted finite horizon problems. Assume we have a discount factor $\alpha (\in (0, 1))$. The cost-to-go function now becomes:

$$J_k(x_k) = \min_{\mu_k, \cdots \mu_{N-1}} \mathbb{E}\left[\sum_{t=k}^{N-1} \alpha^{t-k} g(x_t, \mu_t(x_t), x_{t+1}) \middle| x_k\right]$$

and again, our goal is to find $J_0(x_0)$, and the minimizing policy. We can re-state the above theorem, by simply accounting for the discount factor appropriately. No proof is provided.

Theorem 2
$$J_k(x) = \min_{u \in U(x)} \sum_{y \in S} p_{xy}(u)(g(x,u,y) + \alpha J_{k+1}(y))$$
where $J_N(x) \equiv 0$. *Also, a policy* $\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$ *is optimal if and only if*
$$\mu_k^*(x) \in \arg\min_{u \in U(x)} \sum_{y \in S} p_{xy}(u)(g(x,u,y) + \alpha J_{k+1}(y)).$$

2.2 Discounted Infinite Horizon Problems

Now we can make the appropriate definitions for infinite horizon problems. Everything is the same as before, only now $t = 0, 1, \ldots$ and $\pi = \{\mu_0, \mu_1, \ldots\}$. Let's redefine the cost-to-go function to account for that:
$$J_k(x_k) = \min_{\mu_k, \mu_{k+1}, \ldots} \mathbb{E}\left[\sum_{t=k}^{\infty} \alpha^{t-k} g(x_t, \mu_t(x_t), x_{t+1}) \middle| x_k \right]$$
Let's also define the cost function with respect to a particular policy $\pi = \{\mu_0, \mu_1, \cdots\}$:
$$J_\pi(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \alpha^{t} g(x_t, \mu_t(x_t), x_{t+1}) \middle| x_0 = x \right]$$
Note that the expectation above converges because $\alpha \in (0,1)$ and also because g is bounded, because we assume a finite state space.
In discounted infinite horizon problems, our goal is to find the optimal policy and its associated cost:
$$J^*(x) = \inf_\pi J_\pi(x)$$

3 The Dynamic Programming Operator

We now define the Dynamic Programming Operator (DP Operator) T.
$$(TJ)(x) = \min_{u \in U(x)} \sum_{y \in S} p_{xy}(u)(g(x,u,y) + \alpha J(y))$$
Also let's define the operator T_μ with respect to a fixed μ:
$$(T_\mu J)(x) = \sum_{y \in S} p_{xy}(\mu(x))(g(x,u,y) + \alpha J(y))$$
Notice that with the above definitions we can restate Theorem 1 compactly as:

Theorem 3 $J_k = TJ_{k+1}$ *and* $\pi^* = \{\mu_0^*, \ldots, \mu_n^*\}$ *is optimal if and only if*
$$T_{\mu_k^*} J_{k+1} = TJ_{k+1}$$

Now let's prove an interesting property of the operator and the optimal value function J^*.

Theorem 4
$$J^* = \lim_{N \to \infty} T^N J.$$

Proof Let's look at $J_\pi(x_0)$ and split up the expectation in it in two parts:

$$J_\pi(x) = \mathbb{E}\left[\sum_{t=0}^{N-1} \alpha^t g(x_t, \mu_t(x_t), x_{t+1}) \middle| x_0 = x\right] + \mathbb{E}\left[\sum_{t=N}^{\infty} \alpha^t g(x_t, \mu_t(x_t), x_{t+1}) \middle| x_0 = x\right]$$

Let's look at the second term. Notice that its absolute value is less than $\frac{\alpha^N}{1-\alpha}M$, where M is a constant such that $|g(x,u,y)| < M$.

Recall that

$$(T^N J)(x_0) = \min_{\mu_0, \dots \mu_{N-1}} \mathbb{E}\left[\sum_{t=0}^{N-1} \alpha^t g(x_t, \mu_t(x_t), x_{t+1}) + \alpha^N J(x_N) \middle| x_0\right]$$

Now using our bound on the absolute value of the second term, and the above, we can write the following inequalities:

$$J_\pi(x_0) - \frac{\alpha^N}{1-\alpha}M - \alpha^N \|J\|_\infty \leq \mathbb{E}\left[\sum_{t=0}^{N-1} \alpha^t g(x_t, \mu_t(x_t), x_{t+1}) + \alpha^N J(x_N) \middle| x_0\right] \leq J_\pi(x_0) + \frac{\alpha^N}{1-\alpha}M + \alpha^N \|J\|_\infty$$

Let's minimize each term w.r.t π:

$$J^*_\pi(x_0) - \frac{\alpha^N}{1-\alpha}M - \alpha^N \|J\|_\infty \leq T^N J \leq J^*_\pi(x_0) + \frac{\alpha^N}{1-\alpha}M + \alpha^N \|J\|_\infty$$

Clearly as $n \to \infty$, $\alpha^N \to 0$. Since N was arbitrary it follows that $J^* = \lim_{n\to\infty} T^N J$.
The above proof used our assumption of finite state space to get an upper bound M on $g()$. It needs additional assumptions to work with infinite state spaces. $\square$

We can also show that the operator T has the following additional properties:

Theorem 5 *(Max-norm contraction) T is a maximum norm α-contraction. That is, $\|TJ - T\overline{J}\|_\infty \leq \alpha\|J - \overline{J}\|_\infty$ for all $J, \overline{J}$.*

Proof For arbitrary functions $g, h : A \to \mathbb{R}$, where A is some arbitrary set, the following property holds:

$$\left|\min_a g(a) - \min_a h(a)\right| \leq \max_a |g(a) - h(a)|.$$

Using this property we get

$$\begin{aligned} |(TJ)(x) - (T\overline{J})(x)| &= \left|\min_u \left(\sum_{y\in S} p_{xy}(u)(g(x,u,y) + \alpha J(y)\right) - \min_u \left(\sum_{y\in S} p_{xy}(u)(g(x,u,y) + \alpha \overline{J}(y)\right)\right| \\ &\leq \max_u \alpha \sum_{y\in S} p_{xy}(u)|J(y) - \overline{J}(y)| \\ &\leq \alpha\|J - \overline{J}\|_\infty. \end{aligned}$$

Since $\|TJ - T\overline{J}\|_\infty = \max_x |(TJ)(x) - (T\overline{J})(x)|$, the previous inequality implies $\|TJ - T\overline{J}\|_\infty \leq \alpha\|J - \overline{J}\|_\infty$. $\square$

Theorem 6 *(Monotonicity) If $J \geq \overline{J}$, then $TJ \geq T\overline{J}$.*

Proof Suppose $J \geq \overline{J}$. Then

$$\sum_{y \in S} p_{xy}(u) J(y) \geq \sum_{y \in S} p_{xy}(u) \overline{J}(y)$$

for all $x \in S$ and $u \in U(x)$. By multiplying both sides by α and adding the term $\sum_{y \in S} p_{xy}(u) g(x,u,y)$ to both sides of the inequality, we get

$$\sum_{y \in S} p_{xy}(u)(g(x,u,y) + \alpha J(y)) \geq \sum_{y \in S} p_{xy}(u)(g(x,u,y) + \alpha \overline{J}(y))$$

for all $x \in S$ and $u \in U(x)$. The above inequality implies $T_\mu J \geq T_\mu \overline{J}$ for any decision rule μ. Suppose μ^* is such that $T_{\mu^*} J = TJ$. Then $TJ \geq T_{\mu^*} \overline{J}$. Also, it is clear that $T_{\mu^*} \overline{J} \geq T\overline{J}$. Therefore $TJ \geq T\overline{J}$. □

Theorem 7 *(Offset property) Let e be such that $e(x) = 1$ for all $x \in S$. Then $T(J + ce) = TJ + \alpha ce$ for all $c \in \mathbb{R}$.*

Proof

$$\begin{aligned} T(J+ce)(x) &= \min_{u \in U(x)} \sum_{y \in S} p_{xy}(u)(g(x,u,y) + \alpha(J(y) + ce(y))) \\ &= \min_{u \in U(x)} \sum_{y \in S} (p_{xy}(u)(g(x,u,y) + \alpha J(y)) + \alpha c \\ &= (TJ)(x) + \alpha ce(x) \end{aligned}$$

□

4 Contractions

As was shown in the previous section, the dynamic programming operator T is an α-contraction in the max-norm. In this section we will prove some useful properties of contractions, and discuss some of their implications for dynamic programming. Throughout this section we will let F be an α-contraction with respect to some norm $\|\cdot\|$. For simplicity we will assume $F : \mathbb{R}^n \to \mathbb{R}^n$.

Theorem 8 *The sequence $\{F^N J\}$ converges for any J.*

Proof Since $F : \mathbb{R}^n \to \mathbb{R}^n$, it will suffice to show that $\{F^N J\}$ is a Cauchy sequence. Since F is an α-contraction, $\|FJ - F^2 J\| \leq \alpha \|J - FJ\|$. In general, $\|F^N J - F^{N+1} J\| \leq \alpha^N \|J - FJ\|$. To show that $\{F^N J\}$ is a Cauchy sequence, we need to show that for any $\epsilon > 0$, there exists some K such that $\|F^M J - F^N J\| \leq \epsilon$ for all $M, N \geq K$. For any K and $M, N \geq K$,

$$\begin{aligned} \|F^M J - F^N J\| &= \left\| \sum_{i=M}^{N-1} (F^i J - F^{i+1} J) \right\| \\ &\leq \sum_{i=M}^{N-1} \|(F^i J - F^{i+1} J)\| \\ &\leq \sum_{i=M}^{N-1} \alpha^i \|(J - FJ)\| \\ &\leq \frac{\alpha^K}{1-\alpha} \|(J - FJ)\| \end{aligned}$$

For any $\epsilon > 0$, we can find K such that

$$\frac{\alpha^K}{1-\alpha}\|(J - FJ)\| \leq \epsilon,$$

hence $\{F^N J\}$ is a Cauchy sequence. $\square$

Theorem 9 *F has a unique fixed point.*

Proof The sequence $\{F^N J\}$ converges to a fixed point of F, so at least one fixed point exists. Now suppose J_1 and J_2 are both fixed points of F. Since $FJ_1 = J_1$ and $FJ_2 = J_2$, this implies

$$\|FJ_1 - FJ_2\| = \|J_1 - J_2\|,$$

contradicting the contractive property of F. Therefore, the fixed point of F is unique. $\square$

Recall that the dynamic programming operator T is a max-norm α-contraction and that $T^N J \to J^*$ as $N \to \infty$. By the previous two theorems, we can conclude that J^* is the unique solution to the equation

$$J^* = TJ^*.$$

This is known as *Bellman's equation.* We can also use the fact that T_μ is a max-norm α-contraction for any μ to establish the following result:

Theorem 10 *A* stationary policy $\pi = \{\mu, \mu, \mu, \ldots\}$ *is optimal among all policies if and only if* $TJ^* = T_\mu J^*$.

Proof First suppose that the stationary policy described by μ is optimal. Let J_μ be the cost-to-go function under this policy. Since this policy is optimal, $J^* = J_\mu$. Also, the equation $J = T_\mu J$ is uniquely solved by J_μ. So $J_\mu = T_\mu J_\mu \Longrightarrow J^* = T_\mu J^* \Longrightarrow TJ^* = T_\mu J^*$.
Now suppose $TJ^* = T_\mu J^*$. This implies $J^* = T_\mu J^*$. Since J_μ is the unique solution of the equation $J = T_\mu J$, $J^* = J_\mu$, so the stationary policy described by μ is optimal. $\square$

5 Homework

The homework is due on Wednesday, April 14. You are allowed to work on the homework assignments in small groups.

1. Suppose F is an α-contraction with respect to the norm $\|\cdot\|$ and has fixed point J^*. Suppose $\overline{F}$ satisfies $\|FJ - \overline{F}J\| \leq \epsilon$ for all J and $\overline{F}^k J \to \overline{J}$. Show that
$$\left\|J^* - \overline{J}\right\| \leq \frac{\epsilon}{1-\alpha}$$

2. Define the operator T_x such that
$$(T_x J)(y) = \begin{cases} (TJ)(y) & \text{if } y = x \\ J(y) & \text{otherwise} \end{cases}$$
Consider $J_{k+1} = T_n T_{n-1} \cdots T_2 T_1 J_k$. Prove that $J_k \to J^*$.

3. Consider the dynamic programming algorithm $J_{k+1} = TJ_k$. Which converges faster, this algorithm or the algorithm using the operator T_x described in problem 2? Be prepared to answer this question in a 3 minute presentation in class with your group. You do not need to prepare a detailed argument, just give some intuition on which algorithm converges faster.

MS&E339/EE337B Approximate Dynamic Programming Lecture 3 - 4/7/2004

Policy Iteration and Rollouts

Lecturer: Ben Van Roy *Scribe: Jose Blanchet and Su-In*

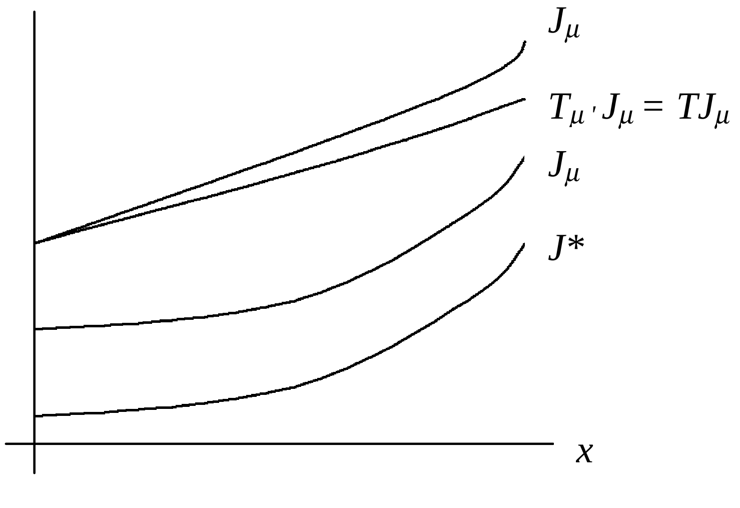
1 Policy Iteration

In contrast to the value iteration algorithm, which implies that $T^N J \to J^*$ as $N \to \infty$, the policy iteration algorithm generates a sequence of policies such that their associated cost-to-go functions $J_k = J^*$ for all k large enough; where J^* is the optimal cost-to-go function (the only fixed point of the dynamic operator T).

In other words, policy iteration generates an optimal stationary policy. The policy iteration algorithm proceeds as follows: given a policy μ_k, choose μ_{k+1} such that $T_{\mu_{k+1}} J_{\mu_k} = TJ_{\mu_k}$. Recall our notation, J_{μ_k} satisfies $J = T_{\mu_k} J$, where T_{μ_k} is the corresponding dynamic operator with respect to a problem with only one policy, namely, the stationary policy generated by μ_k. The next result constitutes the most important theorem of this lecture.

Theorem 1 *There exists a natural N_0 such that for all $k > N_0$ we have that $J_{\mu_k} = J^*$.*

Proof The next diagram shows what is happening at every iteration in the procedure.



First, note that, by optimality of J^*, we can write $J_{\mu_k} \geq J^*$. Also, by definition of μ_{k+1} and T, we have

$$T_{\mu_{k+1}} J_{\mu_k} = TJ_{\mu_k} \leq T_{\mu_k} J_{\mu_k} = J_{\mu_k}.$$

Thus, summarizing, $T_{\mu_{k+1}} J_{\mu_k} \leq J_{\mu_k}$, which implies

$$J_{\mu_k} \geq T^N_{\mu_{k+1}} J_{\mu_k} \to J_{\mu_{k+1}}.$$

Here, we have used standard properties of the dynamic programming operator i.e. monotonicity and contractive properties, in this case applied to the dynamic programming operator corresponding to the problem with only one policy, μ_{k+1}. Now, if $TJ_{\mu_k} = J_{\mu_k}$ then $J_{\mu_k} = J^*$ (because T has only fixed point), otherwise we must have that $T_{\mu_{k+1}} J_{\mu_k} \neq J_{\mu_k}$ and at least one improved can be made. The procedure terminates in a finite number of steps (when no improvement can be made) because there are finitely many decisions and the algorithm converges to J^* as it is the only fixed point of T. $\square$

A very interesting feature of the the policy improvement method is that it seems to work remarkably well in practice; in fact, a complete satisfactory explanation for which policy iteration has the mentioned fast convergence characteristics is still an open problem. The best bound known in the rate of convergence for a general problem with n states and 2 decisions is of order $2^n/n$. However, in practice policy iteration seems to converge in very few iterations, perhaps fewer than 10-15 iterations, for even very large problems. The worst known cases involve problem instances with n states (for small n) and require $n+2$ iterations.

2 Rollouts

In this algorithm the idea is to carry out a single policy iteration step from a given (heuristic) policy, while estimating the cost-to-go function of the given policy via simulation directly from the performance. For example, let's suppose we are studying a discounted dynamic programming problem. Fix a policy μ and a state x, then the idea is to estimate

$$\tilde{\mu}(x) = \min_u \mathbb{E}\left[\sum_{k=0}^{\infty} \alpha^t g(x_t, u_t, x_{t+1}) \middle| x_0 = x, u_0 = u, u_t = \mu(x_t)\right]$$

by

$$\tilde{\mu}(x) \approx \min_u \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{k=0}^{\infty} \alpha^t g(x_t, u_t, x_{t+1}) \middle| x_0 = x, u_0 = u, u_t = \mu(x_t)\right],$$

where N is the number of paths to simulate (of course, one has to truncate the infinite horizon in the previous computation).

The complexity of this method, naturally, grows exponentially as the computational power needed for the sequence of update may be enormous at each state. However, this procedure can work well if one has some good heuristic for μ and the controller is interested in how does the value (corresponding to the selected policy μ) amplifies in very few iterations.

MS&E339/EE337B Approximate Dynamic Programming Lecture 4 - 4/12/2004

Rollout Review, Linear Programming, and Real-Time DP

Lecturer: Ben Van Roy *Scribe: Mark Peters and Michael Rotkowitz*

1 Rollout Review

Suppose that we have a heuristic h such that $h(x_t)$ specifies an action for each x_t. To implement a rollout, we would consider each possible action at each state that the system reaches and calculate the expected cost-to-go based on taking each action then using the heuristic from that point onward. Simulation is used to estimate the expected cost-to-go for each considered action. The rollout will select the optimal action based on the estimated cost-to-go function and implement this action at the current state. If the dynamic program path is infinite, we can simply truncate it when the discounted value drops below some threshold.

Rollout is some kind of a real time policy iteration, it carries out a single policy improvement step, and it updates the policy only when the state is visited. We can think of rollout as a black box, the inputs are the heuristic policy h and current state x, the output is action a. Since policy iteration always converges to the optimal policy in a few steps (say 10-15), rollout usually improves the heuristic policy a lot. Additionally, we can implement a rollout with multiple iterations. After implementing a 1 iteration rollout h' (as shown on the right side of Figure 1), we can apply the rollout procedure to h' (basically treat h' as the heuristic). This creates a 2 iteration rollout. Increasing the number of iterations will exponentially increase the compute time for the rollout. Practically, though, we will often see large improvements with just a few iterations.

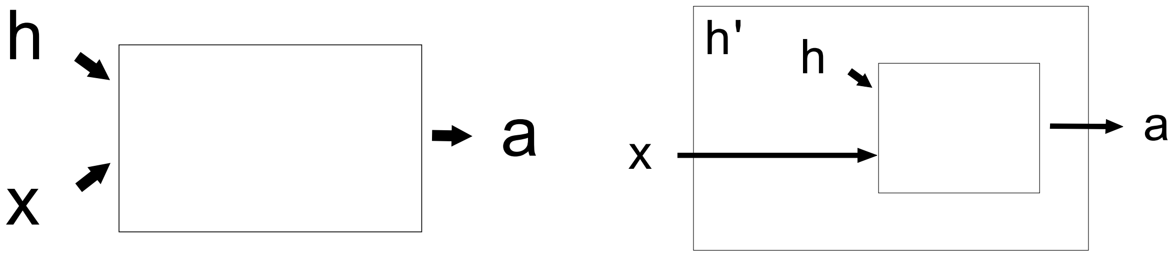


Figure 1: Schematic of Rollout Procedure

1.1 Example

Suppose we have the dynamics $x_{t+1} = f(x_t, u_t, w_t)$ where $w_t \sim q(\cdot)$ is some arbitrary distribution. Further assume that $u_t \in \{0, 1\}$, so that there are two possible actions for each time t. We have a heuristic $\mu : S \to \{0, 1\}$, which when applied yields the closed-loop dynamics

$$x_{t+1} = f(x_t, \mu(x_t), w_t)$$

To improve on this, we try the following rollout procedure.

$$(a) \qquad \text{Estimate} \quad \mathbb{E}\left[\sum_{t=0}^{\infty} \alpha^t g(x_t, u_t, x_{t+1}) \;\middle|\; x_0 = x, u_0 = 0, u_t = \mu(x_t)\ \forall\ t > 0\right]$$

$$(b) \qquad \text{Estimate} \quad \mathbb{E}\left[\sum_{t=0}^{\infty} \alpha^t g(x_t, u_t, x_{t+1}) \;\middle|\; x_0 = x, u_0 = 1, u_t = \mu(x_t)\ \forall\ t > 0\right]$$

The rollout simply chooses the lesser of the two. Two steps often used to make this estimation are truncation and simulation as shown for (a)

$$\begin{aligned} & \mathbb{E}\left[\sum_{t=0}^{\infty} \alpha^t g(x_t, u_t, x_{t+1}) \;\middle|\; x_0 = x, u_0 = 0, u_t = \mu(x_t)\ \forall\ t > 0\right] \\ \approx\ & \mathbb{E}\left[\sum_{t=0}^{M} \alpha^t g(x_t, u_t, x_{t+1}) \;\middle|\; x_0 = x, u_0 = 0, u_t = \mu(x_t)\ \forall\ t > 0\right] \\ \approx\ & \frac{1}{K}\sum_{k=0}^{K-1}\sum_{t=0}^{M} \alpha^t g(x_t^{(k)}, u_t^{(k)}, x_{t+1}^{(k)}) \text{ where } x_0 = x, u_0 = 0, u_t = \mu(x_t)\ \forall\ t > 0 \end{aligned}$$

We choose M large enough so that α^M is negligible, and simulate by the following steps

- Sample $w_t^{(k)}$ according to $q(\cdot)$
- $u_t^{(k)} = \begin{cases} 0 & \text{if } t = 0 \\ \mu(x_t^{(k)}) & \text{if } t > 0 \end{cases}$
- $x_{t+1}^{(k)} = f(x_t^{(k)}, u_t^{(k)}, w_t^{(k)})$

2 Linear Programming

We examine another algorithm for computing J^*. Consider the following optimization problem.

$$\begin{aligned} \text{maximize} \quad & \sum_{x \in S} J(x) \\ \text{subject to} \quad & TJ \geq J \end{aligned} \tag{1}$$

T is a nonlinear operator, so we seek to convert the constraint into several linear constraints.

$$(TJ)(x) = \min_{u \in U(x)} \sum_{y \in S} p_{xy}(u)\left(g(x_t, u, y) + \alpha J(y)\right) \ \geq\ J(x)$$

is equivalent to

$$\sum_{y \in S} p_{xy}(u)\left(g(x_t, u, y) + \alpha J(y)\right) \ \geq\ J(x) \qquad \forall u \in U(x)$$

Theorem 1 *Problem* (1) *is uniquely optimized by* J^*.

Proof If J is feasible, then $TJ \geq J$, and then by the monotonicity of T,

$$J \leq TJ \leq T^2 J \leq \cdots \leq J^*$$

So any feasible J satisfies $J \leq J^*$. Then J^* is feasible and dominates all other feasible points when maximizing the sum. So J^* solves Problem (1), and it's unique. $\square$

Remark This proof holds if we instead maximize $\sum c(x)J(x)$ for any c such that $c(x) > 0$ for every $x \in S$.

3 Real Time Dynamic Programming

3.1 Gauss-Seidel Value Iteration

This algorithm updates one component of J at a time, as in HW#1. It can be shown that the entire cycle of updates (to all components of J) is a max norm contraction. The homework problem asks that we prove convergence to J^* using the Gauss-Seidel value iteration.

3.2 Asynchronous Value Iteration

Asynchronous value iteration picks out an infinite sequence of states $(x^{(0)}, x^{(1)}, x^{(2)}, x^{(3)}, ...)$ such that every state occurs infinitely often. Consider the following algorithm:

$$J_{k+1}(x) = \begin{cases} (TJ_k)(x) & \text{if} \quad x^{(k)} = x \\ J_k(x) & \text{otherwise} \end{cases}$$

We can easily show that this will converge. First, look at states until we reach $x^{(l_1)}$ which is the first time that every state has been reached at least once. In other words, let l_1 be the first time at which $x_t = i$ for every state i for at least one $t \leq l_1$. Similarly, let l_2 be the lowest possible value such that $x_t = i$ for every possible state i for at least one t such that $l_1 < t \leq l_2$, and so forth. Then,

$$\|J_{l_1+1} - J^*\|_\infty \leq \alpha \|J_0 - J^*\|_\infty$$

and repeating for $x^{(l_2)}$

$$\|J_{l_2+1} - J^*\|_\infty \leq \alpha \|J_{l_1+1} - J^*\|_\infty \leq \alpha^2 \|J_0 - J^*\|_\infty$$
$$\vdots$$

Thus J_t converges to J^*.

The name for asynchronous value iteration was derived from asynchronous computation since, if you had each processor work on a component then you could have the component completion times correspond to when J is updated for that component. This justifies parallelizing the process.

3.3 Real Time Value Iteration

This is actually just a special case of asynchronous value iteration. This algorithm simply calls for a sequence of states to be generated by simulating the underlying system. Convergence of real time value iteration doesn't follow from the convergence of asynchronous value iteration due to the fact that you are not guaranteed to update each state an infinite number of times.

Example:

$$\text{Generate } x_0, x_1, x_2, x_3, \dots$$
$$\text{with corresponding } J_0, J_1, J_2, J_3, \dots$$

Method for generating the x_t's

- x_0 is arbitrary
- sample w_t from $q(\cdot)$
- Let $u_t \in \arg\min_{u \in U(x_t)} \sum_{y \in S} p_{x_t y}(u) \left(g(x_t, u, y) + \alpha J_t(y) \right)$.
- $x_{t+1} = f(x_t, u_t, w_t)$

Theorem 2 *If $J_0 \leq J^*$ then RTDP converges, and there exists t_0 such that u_t is optimal for every $t \geq t_0$.*

Proof Next lecture. □

Notes:

- J_0 doesn't necessarily correspond to a policy.
- t_0 is finite with probability 1 (but it is a random variable).
- J_t doesn't necessarily converge to J^* but the action derived will be optimal.
- $J_0 \leq J^*$ means we are optimistic about the cost-to-go function at each state, so we have incentives to go to each state and we are unlikely to get stuch in a bad loop.
- From the monotonicity of the DP operator, all functions will be less than J^*

$$\begin{gathered} J_0 \leq J^* \\ TJ_0 \leq TJ^* = J^* \\ \vdots \\ \text{Then } J_t \leq J^* \text{ for all } t \end{gathered}$$

MS&E339/EE337B Approximate Dynamic Programming Lecture 5 - 4/14/2004

Real Time Dynamic Programming, and Q-Functions

Lecturer: Ben Van Roy *Scribe: Rick Johnston and Brad Null*

1 Real Time Dynamic Programming

1.1 Overview of the Process

- *Simulate the dynamics of the System:*

$$x_{t+1} = f(x_t, u_t, w_t)$$

- *Select decision:*

$$u_t \in \arg\min_{u \in U(x_t)} \sum_{y \in S} p_{xy}(u)(g(x,u,y) + \alpha J_t(y))$$

- *Update J according to:*

$$(J_{t+1})(x) = \begin{cases} \sum_{y \in S} p_{xy}(u_t)(g(x,u,y) + \alpha J_t) & \text{if } x = x_t \\ J_t(x) & \text{otherwise} \end{cases}$$

Theorem 1 *If $J_0 \leq J^*$ then $\exists \bar{t}$ such that u_t are optimal for all $t \geq \bar{t}$ (where $\bar{t} < \infty$ with probability 1)*

Proof First, recall that $J_t \leq J^*$ $\forall t$ (established previously).

Partition the state space S into two sets. Let V be the set of states visited infinitely often and $\bar{V}$ be the complement of V in S. Thus $S = V \cup \bar{V}$.

If we consider a sample path, $\exists \hat{t}$ such that $p_{xy}(u_t) = 0$ $\forall y \in \bar{V}$, $\forall t \geq \hat{t}$ with probability 1.

At all times subsequent to $\hat{t}$, the probability of travelling to $\bar{V}$ is zero. (This follows from the fact that we have a finite state and decision space). Intuitively, this makes sense because each of the states in $\bar{V}$ are only visited a finite number of times. So there must be a time $\hat{t}$ after which the system remains in V. Subsequently, only states in V are relevant and as we are updating each infinitely often, we know we will converge on this set (i.e. for $x \in V$, $J_t(x) \to J_V^*(x)$ for some $J_V^*(x)$ which includes only decisions that keep us in V).

We also know that for $x \in V$, $J_V^*(x) = J_\pi(x)$ where J_π is the cost-to-go function for some legitimate policy π. As J^* is the minimum cost-to-go over all policies, $J_\pi(x) \geq J^*(x)$ $\forall x \in V$. This fact, in addition to the fact that $J_t \leq J^*$ $\forall t$ implies:

$$J_V^*(x) = J^*(x) \forall x \in V$$

When the algorithm terminates, we are left with $J_V^*(x) = J^*(x)$ $\forall x \in V$ and $J_{\bar{V}}(x) \leq J^*(x)$ $\forall x \in \bar{V}$. Under J_V^*, all $x \in V$ have corresponding decisions that ensure we never visit $\bar{V}$ using $J_{\bar{V}}(x)$ as the cost-to-go for all $x \in \bar{V}$. Thus, if we increase all cost-to-go functions for $x \in \bar{V}$ to $J^*(x)$ this would leave J_V^* unchanged, but would increase the cost-to-go for policies containing decisions with positive probability of returning to $\bar{V}$ from V. Thus, J_V^* would still correspond to an optimal policy over V. □

This theorem implies that eventually you will be making optimal decisions. However the process could become stuck in a bad part of the state space. To expand the theory to show global optimality one must

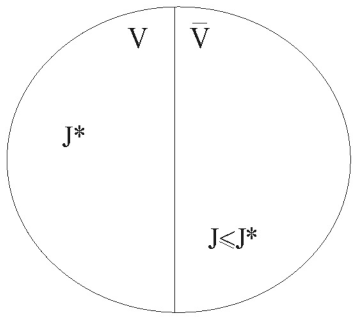


Figure 1: State Space for RTDP

add additional conditions specifying communication exists between all states, i.e. providing it is possible to get from any $x \in S$ to any $y \in S$.

According to this theorem, one valuable application of this algorithm might be to problems with very large state space (e.g. 100 billion states). For such problems we couldn't possibly store J values for each state. However, this theorem suggests that we might be able to use this approach to find an optimal policy among a small subset of the total state space.

2 Q-Functions

Suppose you knew J^*. To make optimal decisions, you would still need to solve

$$\min_{u \in U(x)} \sum_{y \in S} p_{xy}(u)(g(x,u,y) + \alpha J^*(y)) \ \forall x \tag{1}$$

and in doing so, you would have to take expectations for every possible decision $u \in U(x)$ for all states x.

As an alternative, you might define

$$Q^*(x,u) = \sum_{y \in S} p_{xy}(u)(g(x,u,y) + \alpha J^*(y)). \tag{2}$$

and choose optimal decisions by

$$\min_u Q^*(x,u) \tag{3}$$

So how do we compute Q^*? First observe that Q^* satisfies the equation $J^*(x) = \min_u Q^*(x,u) = Q^*(x, u^*(x))$. Substituting into equation 2 above we obtain:

$$Q^*(x,u) = \sum_{y \in S} p_{xy}(u)(g(x,u,y) + \alpha \min_{\bar{u}} Q^*(y,\bar{u})) \tag{4}$$

Q^* is the unique optimal solution of this equation.

For compactness we can define the operator F - similar to the dynamic programming operator T - such that:

$$(FQ)(x,u) = \sum_{y \in S} p_{xy}(u)(g(x,u,y) + \alpha \min_{\bar{u}} Q(y,\bar{u})) \tag{5}$$

Thus, we can rewrite equation 3 as $Q^* = FQ^*$, the Q-equivalent to Bellman's equation.

As we would expect, the following properties also hold for F:

- *F is a maximum norm α-contraction*
- *F has a unique fixed point which is Q^**
- *All of the following algorithms applied in Q will converge to Q^*: Q-Value Iteration (where $Q_{k+1} = FQ_k$), Gauss-Seidel, and Asynchronous Value Iteration*

2.1 Real Time Dynamic Programming using the Q operator

We can also apply Real Time Dynamic Programming (RTDP) to Q rather than J by in any iteration updating only the state-action pair currently under consideration. The following is an overview of this process:

- *Simulate the dynamics of the System:*
$$x_{t+1} = f(x_t, u_t, w_t)$$
- *Select decision:*
$$u_t \in arg \min_{u \in U(x_t)} Q_t(x_t, u)$$
- *Update Q according to:*
$$Q_{t+1}(x,u) = \begin{cases} (FQ_t)(x,u) & \text{if } (x,u) = (x_t, u_t) \\ Q_t(x,u) & \text{otherwise} \end{cases}$$

The characteristics of Q lead to the following theorem (analogous to theorem 1) which we will not prove.

Theorem 2 *If $Q_0 \leq Q^*$ then $\exists \bar{t}$ after which the estimate will be optimal.*

The advantage of Q-RTDP over standard RTDP is that only one expectation for each state-action pair has to be calculated. This means that individual iterations are executed more quickly. However convergence may be much slower under Q-RTDP.

MS&E339/EE337B Approximate Dynamic Programming Lecture 6 - 4/20/2004

Real Time Dynamic Programming, and Q-Functions

Lecturer: Ben Van Roy *Scribe: Lykomidis Mastroleon and Jeffrey Randall Sadowsky*

1 Real Time Dynamic Programming

1.1 Overview of the Real Time Dynamic Programming Algorithm

- *Simulate the dynamics of the System:*

$$x_{t+1} = f(x_t, u_t, w_t)$$

- *Select decision:*

$$u_t \in \arg \min_{u \in U(x_t)} \sum_{y \in S} p_{xy}(u)(g(x,u,y) + \alpha J_t(y))$$

- *Update J according to:*

$$(J_{t+1})(x) = \begin{cases} \sum_{y \in S} p_{xy}(u_t)(g(x,u,y) + \alpha J_t) & \text{if } x = x_t \\ J_t(x) & \text{otherwise} \end{cases}$$

1.2 Real Time Dynamic Programming using the Q operator

We can also apply Real Time Dynamic Programming (RTDP) to Q rather than J by in any iteration updating only the state-action pair currently under consideration. The following is an overview of this process:

- *Simulate the dynamics of the System:*

$$x_{t+1} = f(x_t, u_t, w_t)$$

- *Select decision:*

$$u_t \in arg \min_{u \in U(x_t)} Q_t(x_t, u)$$

- *Update Q according to:*

$$Q_{t+1}(x,u) = \begin{cases} (FQ_t)(x,u) & \text{if } (x,u) = (x_t,u_t) \\ Q_t(x,u) & \text{otherwise} \end{cases}$$

In evaluating the above update for Q_{t+1}, the following expectation has to be calculated :

$$(FQ)(x,u) = \sum_{y \in S} p_{xy}(u)(g(x,u,y) + \alpha \min_{\bar{u}} Q(y,\bar{u})) \tag{1}$$

1.3 Q-Learning

As an alternative to the computationally expensive calculation of the previously mentioned expectation, the following update algorithm can be used:

$$Q_{t+1}(x,u) = \begin{cases} \gamma_t(g(x_t,u_t,y_{t+1}) + \alpha Q(x_{t+1},u_{t+1})) + (1-\gamma_t)Q(x_t,u_t) & \text{if } (x,u) = (x_t,u_t) \\ Q_t(x,u) & \text{otherwise} \end{cases}$$

This is known as the Q-learning update rule. Provided that every state is visited infinitely often, the Q-learning update can be proven to converge to Q^*. However, this proof requires basic results from stochastic approximation theory.

2 Stochastic Approximation Theory

2.1 Strong Law of Large Numbers

The following theorem is known as the **Strong Law of Large Numbers (SSLN)**:

Theorem 1 *Let X_1, X_2, …, X_k be a sequence of i.i.d. random variables, each having a finite mean of $\mathbb{E}[X_i]$. Then:*

$$Y_k = \frac{1}{k}\sum_{i=1}^{k} X_i = r_k \rightarrow r^* = \mathbb{E}[X_i] \text{ w.p. } 1 \tag{2}$$

2.2 Recursive Version of Strong Law of Large Numbers and A Generalization

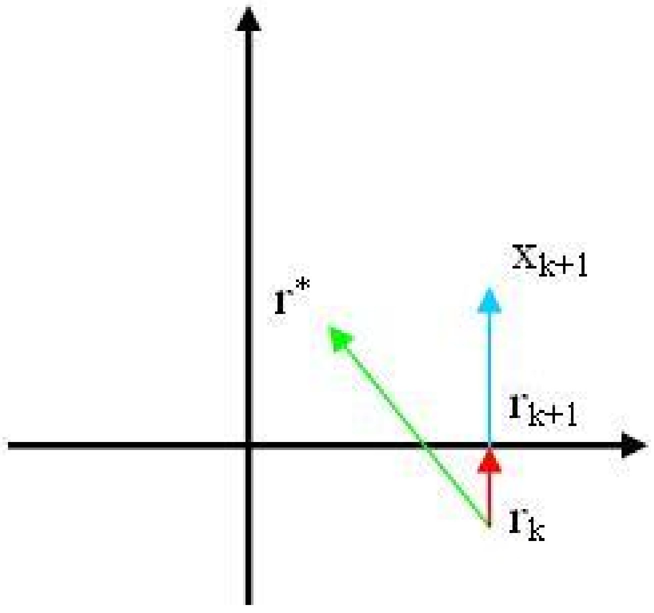


Figure 1: Graphical Interpretation of the SLLN generalization.

Based on the left hand equations in (2) we can write the following regarding r_k:

$$r_{k+1} = (1 - \frac{1}{k+1})r_k + \frac{1}{k+1}x_{k+1} = r_k + \frac{1}{k+1}(x_{k+1} - r_k) = r_k + \gamma_t(x_{k+1} - r_k) \quad (\gamma_k = \frac{1}{k+1}) \tag{3}$$

However, it can be proven that r_k converges to r^* for a range of γ_t. More specifically the following theorem is true:

Theorem 2

$$\begin{cases} \sum_{k=1}^{\infty} \gamma_k = \infty \\ \sum_{k=1}^{\infty} \gamma_k^2 < \infty \end{cases} \Rightarrow r_{k+1} = r_k + \gamma_k(x_{k+1} - r_k) \rightarrow r^*$$

An intuitive interpretation of this theorem is presented in Figure 1.

Now suppose F is an α-contraction with respect to the Euclidean-norm $\|\ldots\|_2$ and furthermore, assume that it has a fixed pint r^* ($r^* = Fr^*$). Then, we know that $r_{k+1} = Fr_k \rightarrow r^*$. We will now prove the following theorem:

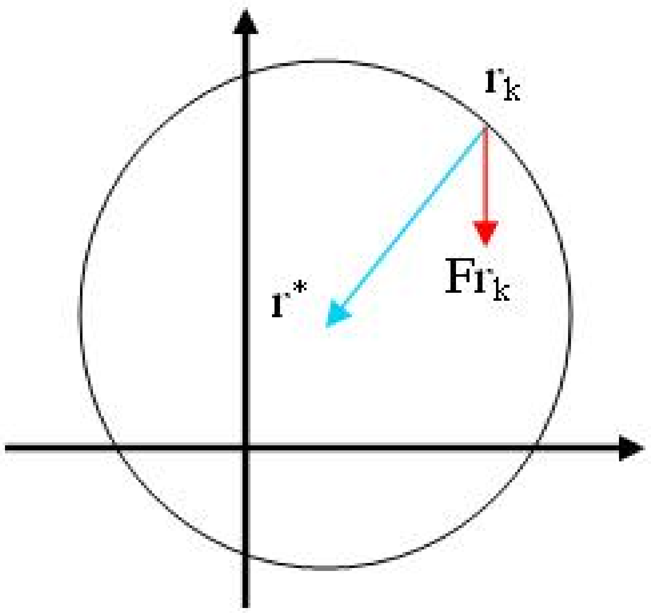


Figure 2: Graphical Interpretation of the SLLN generalization involving α-contractions.

Theorem 3 *Define* $r_{k+1} = r_k + \gamma_k(Fr_k + w_k - r_k)$ *with* w_t *i.i.d. such that* $\mathbb{E}[w_k] = 0$*.Assume:*

1. $\sum_{t=1}^{\infty} \gamma_k = \infty$
2. $\gamma_k \in (0, 1)$

Then $r_k \to r^*$.

Proof

(An intuitive interpretation of this theorem is presented in Figure 2.)

First let $w_k = 0$. Then $r_{k+1} = r_k + \gamma_k(Fr_k - r_k)$. We will first prove for this case that $r_k \to r^*$.

$$\|r_{k+1} - r^*\|_2 = \|r_k + \gamma_k(Fr_k - r_k) - r^*\|_2 \leq$$
$$\leq (1 - \gamma_k)\|r_k - r^*\|_2 + \gamma_k\|Fr_k - r^*\|_2 \leq$$
$$\leq (1 - \gamma_k)\|r_k - r^*\|_2 + \gamma_k\alpha\|r_k - r^*\|_2 \leq$$
$$\leq \|r_k - r^*\|_2 - \gamma_k(1 - \alpha)\|r_k - r^*\|_2 \leq$$

Define $\varepsilon_k = \|r_k - r^*\|_2$. Obviously:

$$\varepsilon_{k+1} \leq \varepsilon_k - \gamma_k(1 - \alpha)\varepsilon_k$$

Thus ε_k is non-increasing. Since $\varepsilon_k = \|r_k - r^*\|_2 \geq 0$ ε_k is also bounded below. Therefore we conclude that ε_k converges. Assume it converges to $\bar{\varepsilon}$.

If $\bar{\varepsilon} > 0$ because :

$$\varepsilon_{k+1} \leq \varepsilon_k - \gamma_k(1 - \alpha)\varepsilon_k \leq \varepsilon_k - \gamma_k(1 - \alpha)\bar{\varepsilon}$$

so by summing the previous inequalities we conclude that:

$$\bar{\varepsilon} \leq \varepsilon_1 - (1-\alpha)\bar{\varepsilon}\sum_{t=1}^{\infty}\gamma_k \leq -\infty$$

which leads to a contradiction. So it must be true that $\bar{\varepsilon} = 0$ □

In the next section we will build the general proof for noisy w_k by introducing additional concepts like Lyapunov functions.

3 Stochastic Approximation with Noise

Now let $w_k \neq 0$. Let's consider the more general case where $r_{k+1} = r_k + \gamma_k s(r_k, w_k)$. Also let $\overline{s}(r_k) = \mathbb{E}\left[s(r_k, w_k)\right]$. A Lyapunov function is a function $V : R^n \to R$ that satisfies the following conditions:

1. $V(r) \geq 0$
2. $\nabla_r V(r^\star) = 0$
3. $(\nabla_r V(r))^T \overline{s}(r) < 0$, for all $r \neq r^\star$

Theorem 4 *$V(r) = \frac{1}{2}\|r^\star - r\|_2^2$ is a Lyapunov function corresponding to $\overline{s}(r) = Fr - r$*

Proof (1) and (2) hold trivially. To show (3) we can see that:

$$(\nabla_r V(r))^T \overline{s}(r) = (r - r^\star)^T(Fr - r) = (r - r^\star)^T(r^\star - r) + (r - r^\star)^T(Fr - r^\star) \leq$$

$$-\|r - r^\star\|_2^2 + \alpha\|r - r^\star\|_2^2 = -(1-\alpha)\|r - r^\star\|_2^2 < 0 \text{ if } r \neq r^\star$$

where we have used that: $\|(r - r^\star)^T(Fr - r^\star)\| \leq \|(r - r^\star)\|\|(Fr - r^\star)\| \leq$

$$\leq \|(r - r^\star)\|\alpha\|(r - r^\star)\| = \alpha\|(r - r^\star)\|^2$$

□

Theorem 5 *If $V(r) = \frac{1}{2}\|r^\star - r\|_2^2$ is a Lyapunov function and the following conditions hold:*

1. $(r^\star - r)^T\overline{s}(r) \geq C_1\|r^\star - r\|_2^2$

2. $E_w[\|s(r, w)\|_2^2] \leq C_2(1 + \|r^\star - r\|_2^2)$

3. $\sum\gamma_k = \infty$, $\sum\gamma_k^2 < \infty$, $\gamma_k > 0$ and γ_k diminishing.

Then $r_k \to r^\star$ w.p. 1.

Before proving this theorem let's state without proof three standard convergence results from Probability Theory:

(a) Consider $X_k \geq 0$, s.t $E_k[X_{k+1}] \leq X_k$, then $X_k \to \overline{X} \geq 0$ w.p. 1. This result is known as the Supermartingale Convergence Theorem.

(b) Consider $X_k, Y_k \geq 0$, s.t $\sum Y_k < \infty$ and $E_k[X_{k+1}] \leq X_k + Y_k$, then $X_k \to \overline{X} \geq 0$ w.p. 1.

(c) Consider $X_k, Y_k, Z_k \geq 0$, s.t $\sum Y_k < \infty$ and $E_k[X_{k+1}] \leq X_k + Y_k - Z_k$, then $X_k \to \overline{X} \geq 0$ and $Z_k \to 0$ w.p. 1.

Proof Let $\epsilon_{k+1} = \|r_{k+1} - r^\star\|_2^2 = \|r_k + \gamma_k s(r_k, w_k) - r^\star\|_2^2$. By doing the inner product ϵ_{k+1} could be rewritten as:

$$\epsilon_{k+1} = \epsilon_k + \gamma_k^2 \|s(r_k, w_k)\|_2^2 - 2\gamma_k (r^\star - r_k)^T s(r_k, w_k)$$

Taking conditional expectations with respect to k and using conditions (1) and (2) of the statement of the theorem we get:

$$E_k[\epsilon_{k+1}] \leq \epsilon_k + \gamma_k^2 C_2 (1 + \|r^\star - r\|_2^2) - 2\gamma_k C_1 \|r - r^\star\|_2^2$$

$$E_k[\epsilon_{k+1}] \leq \epsilon_k + \epsilon_k(\gamma_k^2 C_2 - 2\gamma_k C_1) + C_2 \gamma_k^2$$

If we let $X_k = \epsilon_k$, $Y_k = C_2\gamma_k^2$ and $Z_k = \epsilon_k(2\gamma_k C_1 - \gamma_k^2 C_2)$ we are ready to use our convergence result (c), since we can trivially see that $X_k, Y_k \geq 0$. Moreover, $Z_k = \epsilon_k(2\gamma_k C_1 - \gamma_k^2 C_2)$ will be greater or equal to zero eventually by condition (3).

Hence, applying our convergence result (c) we have that: $\epsilon_k \to \bar{\epsilon}$ and $\epsilon_k(2\gamma_k C_1 - \gamma_k^2 C_2) \to 0$ w.p. 1 as $k \to \infty$. Since $\epsilon_K \geq 0$ for all k then $\bar{\epsilon} \geq 0$.

Now suppose $\bar{\epsilon} > 0$. $\forall \delta < \bar{\epsilon}$, there exists K such that $\epsilon_k > \bar{\epsilon} - \delta$, for all $k > K$. So for large enough k

$$E_k[\epsilon_{k+1}] \leq \epsilon_k + (\bar{\epsilon} - \delta)(\gamma_k^2 C_2 - 2\gamma_k C_1) + C_2\gamma_k^2$$

Taking conditional expectation we have

$$E_k[\epsilon_{k+l+1}] \leq E_k[\epsilon_{k+l}] + (\bar{\epsilon} - \delta)(\gamma_{k+l}^2 C_2 - 2\gamma_{k+l} C_1) + C_2\gamma_{k+l}^2, \forall l \geq 0$$

So

$$E_k[\epsilon_{k+l+1}] \leq \epsilon_k + (\bar{\epsilon} - \delta)(C_2 \sum_{i=0}^{l} \gamma_{k+i}^2 - 2C_1 \sum_{i=0}^{l} l\gamma_{k+i}) + C_2 \sum_{i=0}^{l} \gamma_{k+i}^2, \forall l \geq 0$$

Let $l \to \infty$ we get

$$\lim_{l \to \infty} E_k[\epsilon_{k+l+1}] \leq -\infty$$

which is impossible since $\epsilon_i \geq 0$. So we must have $\bar{\epsilon} = 0$. □

4 Homework

Consider $\|r\|_D^2 = r^T D r$, where D is diagonal and positive definite. If $V(r) = \frac{1}{2}\|r^\star - r\|_D^2$ is a Lyapunov function and the following conditions hold:

1. $(r^\star - r)^T D \bar{s}(r) \geq C_1 \|r^\star - r\|_D^2$
2. $E_w[\|s(r, w)\|_D^2] \leq C_2(1 + \|r^\star - r\|_D^2)$
3. $\sum \gamma_k = \infty$, $\sum \gamma_k^2 < \infty$, $\gamma_k > 0$ and γ_k diminishing.

Then $r_k \to r^\star$ w.p 1.

MS&E339/EE337B Approximate Dynamic Programming Lecture 7 - 4/26/2004

Asynchronous Stochastic Approximation, and Q-Learning

Lecturer: Ben Van Roy *Scribe: Shahriar Azizpour and Amirpouya Kavousian*

1 Review

In our previous discussion, different types of sequences in the following form were considered and their convergence were showed analytically:

$$r_{t+1} = r_t + \gamma_t s(r_t, w_t)$$

This method is in general like the *Gradient Method.* Although last time we showed the convergence of r_t with a special type of Lyapunov function $V(r) = \|r - r^*\|^2$, but it works for some other Lyapunov functions as well. In general consider:

$$w_t \perp_{r_t} (r_{t-1}, r_{t-2}, \ldots, w_{t-1}, w_{t-2}, \ldots)$$

$$\bar{s}(r) = \mathbb{E}\left[s(r_t, w_t) | r_t = r\right]$$

The following theorem is similar to what we had:

Theorem 1 *Suppose $\exists V : R^n \to R$ such that:*

1. $V(r) \geq 0$
2. *V is continuously differentiable, and $\exists L$ s.t. $\|\nabla V(r) - \nabla V(\bar{r})\|_2 \leqslant L\|r - \bar{r}\|_2$*
3. *$\exists c$ s.t. $c\|\nabla V(r)\|^2 \leqslant -\nabla V(r)^T \bar{s}(r)$*
4. *$\exists k_1, k_2,$ s.t. $\bar{s}(r) \leqslant k_1 + k_2\|\nabla V(r)\|^2$*

Then: if

$$\gamma_t > 0, \sum_{t=1}^{\infty} \gamma_t = \infty, \sum_{t=1}^{\infty} {\gamma_t}^2 < \infty$$

1. *$V(r_t)$ converges*
2. $\lim_{t\to\infty} \nabla V(r_t) = 0$
3. *All limit points satisfy $\nabla V(r) = 0$*

In the quadratic Lyapunov, we had only one optimal point. But here we see a more general case: you may have more than one limit point or you may have no limit at all (infinity).

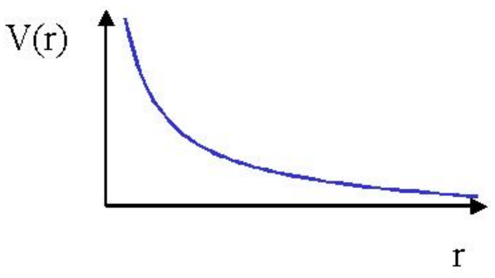


Figure 1: No limit case

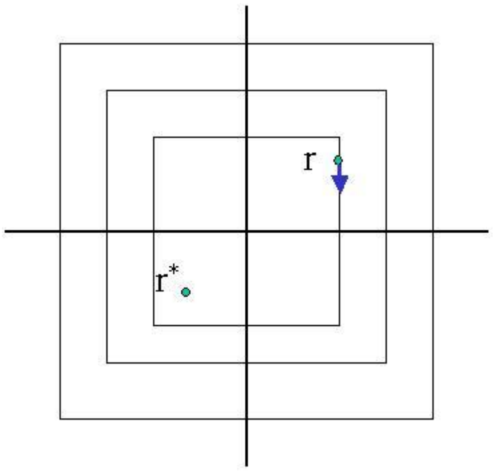


Figure 2: V(r) with Box level sets

2 Asynchronous Stochastic Approximation

Now consider:

$$r_{t+1}(i_t) = r_t(i_t) + \gamma_t s_{i_t}(r_t, w_t)$$

Where r_t is a vector and we update one component of r_t at each time. If $i_0, i_1, \cdots$ samples each index infinite number of times, then r_t converges. Because we are only updating one component at each time, we need maximum norm.

Note that using the $\|.\|_2$ may cause some problems (divergence instead of convergence).

Now, let's relate this with Q-Learning.

3 Q-Learning

The Q-Learning updating rule is:

$$Q_{t+1}(x,a) = (1-\gamma_t)Q_t(x,a) + \gamma_t(g(x,a,y) + \alpha \min_{\bar{a}} Q_t(y,\bar{a}))$$

We can relate Q-Learning with asynchronous stochastic approximation by the following substitution:

$$\begin{aligned} r_t &= Q_t \\ s_{(x,a)}(Q,y) &= g(x,a,y) + \alpha \min_{\bar{a}} Q_t(y,\bar{a}) - Q(x,a) \\ \bar{s}_{(x,a)}(Q) &= \sum_y P_{xy}(a)(g(x,a,y) + \alpha \min_{\bar{a}} Q_t(y,\bar{a}) - Q(x,a)) \\ &= FQ - Q \\ V(Q) &= \|Q - Q^*\|_\infty \end{aligned}$$

4 Approximation of J^*

Now suppose that instead of looking for a function $J(\cdot)$ in general and updating its values by TJ (value iteration), we want to approximate J^* (our limit, cost-to-go function) by some basis functions $\{\phi_1, \cdots, \phi_K\}$:

$$J^* \simeq \sum_{k=1}^{K} r_k \phi_k$$

where $\phi_1, \ldots, \phi_K : S \to R$.

In this algorithm, first we need to find good basis functions, then we need to calculate the coefficient r.

Example 1: Tetris

For example in Tetris, the J^* function is a function of the current state (configuration of the board) which shows that how bad or good will be the rest of our game (cost-to-go function). One choice of basis functions is:
ϕ_1 = max(height)
ϕ_2 = absolute difference between heights of columns 1 and 2
ϕ_3 = height of column 1
ϕ_4 = height of column 2
$\vdots$

4.1 Q-Learning (one possible action in each state)

Suppose that at each state there is only one legal action, i.e. there are no decisions to be made. Consider our Q-learning setting. Substitute Q_t's with J_t's

$$Q_{t+1}(x) = (1-\gamma_t)Q_t(x) + \gamma_t(g(x,a,y) + \alpha Q_t(y)))$$

or

$$J_{t+1}(x) = (1-\gamma_t)J_t(x) + \gamma_t(g(x,a,y) + \alpha J_t(y)))$$

This is a Markov Chain and we want to approximate J^* . Simulate the trajectory $x_0, x_1, \cdots$ and update J by

$$J_{t+1}(x_t) = (1-\gamma_t)J_t(x_t) + \gamma_t(g(x_t,a,x_{t+1}) + \alpha J_t(x_{t+1})))$$

or

$$J_{t+1}(x_t) = J_t(x_t) + \gamma_t(g(x_t,x_{t+1}) + \alpha J_t(x_{t+1}) - J_t(x_t))$$

where $J_k = \phi r_k$.

$$\phi = \begin{pmatrix} | & | & & | \\ \phi_1 & \phi_2 & \dots & \phi_K \\ | & | & & | \end{pmatrix}$$

We update r_k's by gradient method. So we will have:

$$r_{t+1} = r_t + \gamma_t \bigtriangledown_r (\phi r_t)(x_t)(g(x_t, x_{t+1}) + \alpha(\phi r_t)(x_{t+1} - \phi r_t)(x_t))$$

Note that r_t is a vector and $\bigtriangledown_r(\phi r_t)(x_t)$ is the direction of maximum impact. Because ϕr_t is a linear function w.r.t. r_t, so we can substitute the gradient:

$$r_{t+1} = r_t + \gamma_t \phi(x_t)(g(x_t, x_{t+1}) + \alpha(\phi r_t)(x_{t+1}) - (\phi r_t)(x_t))$$

where $\phi(i)$ is the ith row of ϕ.

4.2 Approximation Value Iteration

We already had the setting of value iteration:

$$J_{t+1} = TJ_t$$

It's natural to combine this value iteration with approximation of J, i.e., update values of r_t's according to

$$\phi r_{t+1} = \Pi T \phi r_t$$

where Π is the projecting operator

$$\Pi f = \arg\min_{\phi r} \|f - \phi r\|_2$$

Note that, first a T-operator (which is a α -contraction) and then a projector will be applied. This algorithm looks similar to the value iteration, so one might expect similar convergence results here. But unfortunately, the error of this algorithm doesn't go to 0. In fact, it grows exponentially. We will discuss this in the next lecture.

MS&E339/EE337B Approximate Dynamic Programming Lecture 8 - 4/28/2004

Approximate Value Iteration and Refinements

Lecturer: Ben Van Roy *Scribe: Eymen Errais and Donald Lee*

1 Approximate Value Iteration

1.1 Example of iteration divergence

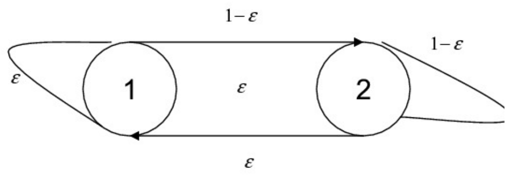


Figure 1: Markov Chain Diagram

Consider the autonomous (one action only) Markov Chain depicted above. We set $\alpha \in (0,1)$ and all costs to zero i.e. $g(1) = g(2) = 0$, hence $J^* = (0,0)$. Let $\Phi = (1,2)$ form the basis for our approximations, so all approximations of the value function take the form Φr. An update using the approximate VI yields

$$\Phi r_{k+1} = \Pi T \Phi r_k$$

$$\Pi J = \arg\min_{\Phi r} \|\Phi r - J\|_2$$

$$(TJ)(i) = \alpha\varepsilon J(1) + \alpha(1-\varepsilon)J(2) \text{ for } i\text{=1,2}$$

Hence

$$(T\Phi r_k)(i) = \alpha(2-\varepsilon)r_k$$

$$r_{k+1} = \arg\min_r \left((r - (T\Phi r_k)(1))^2 + (2r - (T\Phi r_k)(2))^2\right) = \frac{3}{5}\alpha(2-\varepsilon)r_k$$

If $\varepsilon \approx 0$ and $\alpha \approx 1$ then r_k grows to infinity.

2 Refinements

2.1 Some intuition

In stationarity, the Markov Chain above spends only ε-proportion of the time in state 1. Therefore it seems sensible to put more weight on state 2 by using a different norm, namely

$$\|\Phi r - J\|_D^2 = \varepsilon(r - J(1))^2 + (1-\varepsilon)(r - J(2))^2$$

With this modification, it can be shown that the iterations converge to zero no matter where we start.

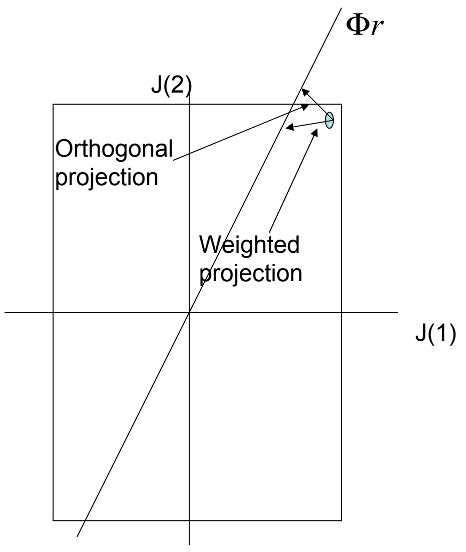


Figure 2: Intuition of divergence and convergence

2.2 Convergence issues

It is well known that $\|\Pi\|_D \leq 1$, which implies

$$\|\Pi T J - \Pi T \overline{J}\|_D \leq \|TJ - T\overline{J}\|_D$$

So if the DP operator is a $\|\cdot\|_D$ α-contraction, then so is the approximate VI operator ΠT.

Theorem 1 *For autonomous (one action only), irreducible and aperiodic Markov Chains with stationary distribution π,*

$$\|TJ - T\overline{J}\|_D \leq \alpha \|J - \overline{J}\|_D$$

where

$$\|x\|_D^2 = x^T D x$$

and

$$D = \begin{pmatrix} \pi_1 & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \pi_n \end{pmatrix}$$

Proof That the transition matrix P satisfies $\|P\|_D \leq 1$ follows from

$$\|PJ\|_D^2 = \sum_{i=1}^{n} \pi_i (PJ)_i^2 = \sum_{i=1}^{n} \pi_i (E[J(X_{t+1}) \mid X_t = i])^2$$

$$\overset{Jensen}{\leq} \sum_{i=1}^{n} \pi_i E[J(X_{t+1})^2 \mid X_t = i] = EJ(X_{t+1})^2 = \|J\|_D^2$$

which implies that

$$\|TJ - T\overline{J}\|_D^2 = \|\alpha PJ - \alpha P\overline{J}\|_D^2 = \alpha^2 \|P(J - \overline{J})\|_D^2 \leq \alpha^2 \|J - \overline{J}\|_D^2$$

$\square$

Since the approximate VI operator is a contraction, it follows that $\Phi r_k \to \Phi r^*$ for any starting value r_0. In general, this limit is not necessarily the projection of J^* onto span(Φ), i.e. $\Phi r^* \neq \Pi J^*$. However we can still provide a bound on the distance of Φr^* from J^* in terms of the minimum distance between J^* and the plane spanned by Φ:

Theorem 2

$$\|\Phi r^* - J^*\|_D \leq \frac{1}{\sqrt{1-\alpha^2}} \|\Pi J^* - J^*\|_D$$

Proof

$$\|\Phi r^* - J^*\|_D^2 = \|\Pi T \Phi r^* - \Pi J^* + \Pi J^* - J^*\|_D^2$$

$$\overset{Pythagoras}{=} \|\Pi T \Phi r^* - \Pi J^*\|_D^2 + \|\Pi J^* - J^*\|_D^2$$

$$\leq \|T \Phi r^* - J^*\|_D^2 + \|\Pi J^* - J^*\|_D^2$$

$$\overset{J^* = TJ^*}{\leq} \alpha^2 \|\Phi r^* - J^*\|_D^2 + \|\Pi J^* - J^*\|_D^2$$

$\square$

MS&E339/EE337B Approximate Dynamic Programming Lecture 9 - 5/5/2004

Temporal Difference Learning

Lecturer: Ben Van Roy *Scribe: Yirong Shen and Chih-Han Yu*

1 Review

Previously, we saw that approximate value iteration,

$$\Phi r_{t+1} = \Pi \mathrm{T} \Phi r_t$$

where Π is the projection operator

$$\Pi J = \arg\min_{\Phi r} \|J - \Phi r\|$$

could diverge. In particular, we saw an example of an autonomous Markov chain in which r_t grows unbounded. However, convergence could be achieved if we use the norm $\|\cdot\|_D$ in the projection operator, where D is the diagonal matrix with $D_{ii} = \pi(i)$ and π is the steady state distribution for the Markov chain, i.e.

$$\Pi J = \arg\min_{\Phi r} \|J - \Phi r\|_D$$

$$\|J\|_D^2 = \sum_x \pi(x) J^2(x)$$

where $\pi(x)$ is steady state distribution. In this case the result is that $(\Pi T)^k J \to \Phi \tilde{r}$ where $\tilde{r}$ is unique and

$$\|\Phi \tilde{r} - J^k\|_D \leq \frac{1}{\sqrt{1-\alpha^2}} \min_r \|J^* - \Phi r\|_D$$

2 Approximate Projection

For large state spaces, the projection operation Π is difficult to compute. Projection matrix $\Pi = \Phi(\Phi^T D \Phi)^{-1} \Phi^T D$

$$\Pi J = \arg\min_{\Phi r} \|J - \Phi r\|_D = \Phi(\Phi^T D \Phi)^{-1} \Phi^T D J$$

The difficulty of computing Φ is it involves summing/integrating over the state space. So instead of summing/integrating over the entire state space, we use Monte Carlo simulation to compute an approximate projection as follows:

1. Sample $x_1, x_2, \ldots, x_k$ from $\pi(x)$
2. Compute the approximate projection using least squares

$$\min_r \frac{1}{k} \sum_{i=1}^{k} (J(x_i) - (\Phi r)(x_i))^2$$

   when k goes to infinity, it will be equivalent to $\min_r \|J - \Phi r\|_D^2$

For a good approximation, the computation required depends on the number of basis functions and NOT on the size of the state space. Hence, even though exact projection is hard to compute, it can be approximated easily.

3 Temporal Difference

Now we describe the temporal difference method. Assume that through simulation, we can obtain a sequence of states $x_0, x_1, x_2, \ldots$. We update r as follows:

$$r_{t+1} = r_t + \gamma_t \phi(x_t) \overbrace{(\underbrace{g(x_t) + \alpha(\Phi r_t)(x_{t+1})}_{\text{improved estimate}} - \underbrace{(\Phi r_t)(x_t)}_{\text{old estimate}})}^{\text{temporal difference}}$$

To understand the above update formula, we note that the gradient of $(\Phi r)(x) = \phi(x)^T r$ with respect to r is

$$\nabla_r (\Phi r)(x) = \phi(x)$$

where $\phi(x)$ is the x-th row of Φ. The temporal difference is just the difference between the new and old estimates of the cost-to-go function. Hence, the update used to obtain r_{t+1} from r_t is proportional to the temporal difference in length and is in the direction of the gradient with respect to r_t at x_t in order to maximize the impact of estimation.

4 Proof of Convergence for TD

We can view TD in the framework of stochastic approximations:

$$r_{t+1} = r_t + \gamma_t s(r_t, x_t, x_{t+1})$$

The expected update step is

$$\bar{s}(r) = \mathbb{E}[s(r_t, x_t, x_{t+1}] = \sum_{x,y} \pi(x) \mathrm{P}_{xy}\, s(r, x, y)$$

We note that the expectation above is with respect to the steady state distribution of the Markov chain, which is different from cases that we saw in previous lectures where the randomness was due to i.i.d. noise, and the expectation was take with respect to the noise distribution.

For the convergence proof, we use the Lyapunov function

$$V(r) = \frac{1}{2}\|r - \tilde{r}\|^2$$

Since $\nabla V(r) = r - \tilde{r}$, all we need to do (plus verifying some technical conditions) is to show that the expected update step is a descent direction, i.e.

$$(r - \tilde{r})^T \bar{s}(r) < 0 \text{ for } r \neq \tilde{r}$$

First, we note that

$$\begin{aligned} \bar{s}_k(r) &= \sum_x \pi(x) \sum_y \mathrm{P}_{xy} \phi_k(x)[g(x) + \alpha(\Phi r)(y) - (\Phi r)(x)] \\ &= \sum_x \pi(x) \phi_k(x)((\mathrm{T}\Phi r)(x) - (\Phi r)(x)) \\ &= \phi_k^T D(\mathrm{T}\Phi r - \Phi r) \end{aligned}$$

Hence, we have $\bar{s}(r) = \Phi^T D(\mathrm{T}\Phi r - \Phi r)$. Now

$$\begin{aligned}
(r-\tilde{r})^T \bar{s}(r) &= (r-\tilde{r})^T \Phi^T D(\mathrm{T}\Phi r - \Phi r) \\
&= (\Phi r - \Phi\tilde{r})^T D(\Pi\mathrm{T}\Phi r - \Pi\Phi r) \\
&= (\Phi r - \Phi\tilde{r})^T D(\Pi\mathrm{T}\Phi r - \Phi r) \\
&= (\Phi r - \Phi\tilde{r})^T D(\Pi\mathrm{T}\Phi r - \Phi\tilde{r}) - \phi_k^T D(\Phi r - \Phi\tilde{r}) \\
&\leq \|\Phi r - \Phi\tilde{r}\|_D \cdot \|\Pi\mathrm{T}\Phi r - \Phi\tilde{r}\|_D - \|\Phi r - \Phi\tilde{r}\|_D \\
&\leq \alpha\|\Phi r - \Phi\tilde{r}\|_D^2 - \|\Phi r - \Phi\tilde{r}\|_D \\
&< 0
\end{aligned}$$

The first inequality follows from the Cauchy-Bunyakovsky-Schwarz inequality. The second inequality is due to the fact that $\Pi\mathrm{T}$ is a D-norm contraction. The last inequality is because α is less than 1. Hence, the expected update step is a descent direction for our chosen Lyapunov function. This combined with some suitable technical conditions, guarantees convergence.

The update rule from r_t to r_{t+1} can be written as the following forms:

$$\begin{aligned}
r_{t+1} &= r_t + \gamma_t\phi(x_t)(g(x_t) + \alpha(\Phi r_t)(x_{t+1}) - (\Phi r_t)(x_t)) \\
r_{t+1} &= r_t + \gamma_t\phi(x_t)((T\Phi r_t)(x_t) - (\Phi r_t)(x_t)) \\
r_{t+1} &= r_t + \gamma_t\phi(x_t)(J^*(x_t) - (\Phi r_t)(x_t)) \\
r_{t+1} &= r_t + \gamma_t \cdot \frac{1}{2}\nabla_r(J^*(x_t) - (\Phi r_t)(x_t))^2
\end{aligned}$$

MS&E339/EE337B Approximate Dynamic Programming Lecture 10 - 05/10/2004

Temporal Difference Learning (Continued)

Lecturer: Ben Van Roy *Scribe: Vishal Kasera and Priyendra Deshwal*

1 Topics Covered

This lecture talks about the following:

1. Convergence of the TD algorithm
2. TD-λ algorithm
3. The Optimal Stopping Problem

2 Convergence of the TD algorithm

We shall try to understand the convergence of the TD algorithm under a more general framework. Specifically, we shall pose the following variant of the TD learning algorithm.

$$r_{k+1} = r_k + \gamma_k \phi(x_k)(g(x_k) + \alpha(\Phi r_k)(y_k) - (\Phi r_k)(x_k)) \tag{1}$$

where, $x_k \sim q(\cdot), y_k \sim P_{x_k y_k}$.
The salient difference between this version and the previous version of the algorithm is that earlier we were simulating a complete Markov chain, whereas here we are resampling x_k from the distribution $q(\cdot)$ (that can be any arbitrary distribution) at each time step.

Theorem 1 *If $q = \pi$, where π is the steady state distribution, then $r_k \to \tilde{r}$. Also $\exists q \neq \pi$ such that $\|r_k\| \to \infty$.*

The proof for the convergence for the case of $q = \pi$ is similar to the one discussed in the previous lecture. It would seem that since in the original version of the TD algorithm, there would be some correlation in the sampled values (as we are sampling one trajectory), it should be different from the algorithm under consideration. However, since this is a stochastic approximation scheme, what matters is the steady state distribution.
Let us now argue that it is always possible to choose q such that the algorithm diverges. In particular, let us look at the expected steps that the values of r_k take.

$$\begin{aligned} \bar{s}(r) &= \mathbb{E}_{x,y}\left[\phi(x)(g(x) + \alpha(\Phi r)(y) - (\Phi r)(x))\right] & (2)\\ &= \sum_x q(x)\phi(x)\left[(T\Phi r)(x) - (\Phi r)(x)\right] & (3)\\ &= \Phi Q(T\Phi r - \Phi r) & (4) \end{aligned}$$

where Q is a diagonal matrix defined as $Q \in \Re^{n\times n}, Q_{ii} = q^{(i)}$. In (3), we have explicitly written out the expectation over x while the expectation over y has implicitly been taken, giving us the term $(T\Phi r)(x)$. Note that this expression is essentially the same as the one we encountered before - however since we can choose to sample from any arbitrary distribution $q(\cdot)$, we have freedom of choosing the matrix Q which we did not have earlier.

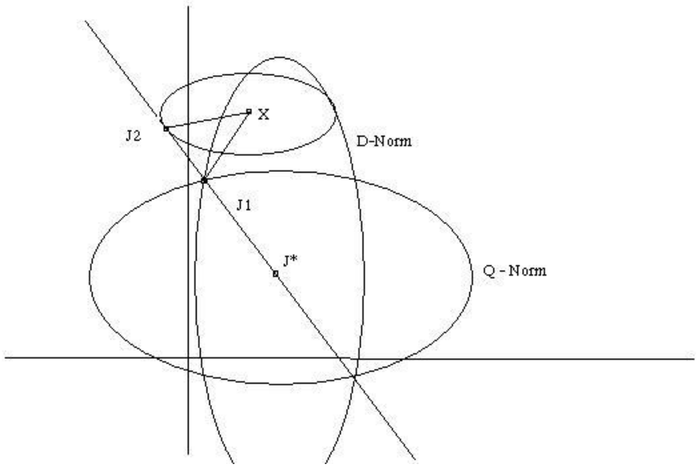


Figure 1: Illustration of Divergence

Now in order to show that the sequence converges, it is sufficient to show that the expected step has a negative inner product with the gradient of the Lyapunov function. On the other hand, if we can come up with a setting in which this inner product is always positive, then we will have shown that the sequence diverges. We shall now show such a case. Let us take our Lyapunov function to be $V(r) = \frac{1}{2}\|r - \tilde{r}\|^2$. Assume that $\exists \tilde{r} : J^* = \Phi\tilde{r}$. We have,

$$\begin{aligned}(\nabla_r V(r))^T \bar{s}(r) &= (r - \tilde{r})^T \Phi Q(T\Phi r - \Phi r) && (5)\\ &= (\Phi r - \Phi\tilde{r})^T Q(T\Phi r - \Phi r) && (6)\\ &= (\Phi r - \Phi\tilde{r})^T Q(\Pi_Q T\Phi r - \Phi r) && (7)\end{aligned}$$

where $\Pi_Q J = \arg\min_{\Phi r} \|J - \Phi r\|_Q$ is the standard projection operator.

Figure1 shows why the above expression is always positive. In the figure, $J^* = \Phi\tilde{r}$, $J1 = \Phi r$, $X = T\Phi r$ and $J2 = \Pi_Q T\Phi r$. We know that T is a contraction in the D-norm and *not* in the Q-norm. Hence, if we start at $J1$, then after applying the T operator, we may end up at X which is outside the Q-norm but inside the D-norm. Projecting X, we get $J2$. However, $J2$ is further away from J^* than $J1$ was. It is also easy to see that the expression in equation 7 will be positive. Hence, we can see that divergence is definitely possible.

3 TD-λ

TD-λ is a generalization of the normal TD algorithm where instead of using the derivative of $\Phi(x_t)$ in the update equation, we use some general function Z_t. So our new update equation is:

$$r_{t+1} = r_t + \gamma_t Z_t(g(x_t) + \alpha(\Phi r_t)(x_{t+1}) - (\Phi r_t)(x_t)) \tag{8}$$

An example of Z_t is:

$$\begin{aligned}Z_t &= \sum_{\tau=0}^{t} (\alpha\lambda)^{t-\tau}\phi(x_\tau) && (9)\\ Z_{t+1} &= (\alpha\lambda)Z_t + \phi(x_t) && (10)\end{aligned}$$

It is easy to see that when $\lambda = 0$ then $Z_t = \phi(x_t)$ and this is the normal TD algorithm.

Theorem 2 *When $\lambda = 0$, then $\|J^* - \Phi\tilde{r}\|_D \leq \frac{1}{\sqrt{1-\alpha^2}} \min_r \|J^* - \Phi r\|_D$. Moreover, when $\lambda = 1$, then $\|J^* - \Phi\tilde{r}\|_D = \min_r \|J^* - \Phi r\|_D$.*

The convergence properties of TD-λ are still under development. However, TD-λ works very well in lots of applications, and there are evidence that generally there exists some λ such that TD-λ converges faster than TD(0) and TD(1). So TD-λ is quite popular. For example, read the paper about Backgammon in the handout section.

4 Optimal Stopping

Our discussion so far has focussed on discounted problems, where there are no termination states nor stopping decisions. But in some cases, like the American option pricing problem, the most important decision is when to stop. So we will now take a look at the optimal stopping problem. It turns out that we can prove interesting results in this problem, please have a look at Prof. Van Roy's paper on this topic in the handout section.

Consider Markov chain $x_0, x_1, \ldots x_t$ where you can decide to stop at any time step t. The pay-off for stopping at time step t is denoted by $G(x_t)$ (note that the pay-offs here are not accumulative but terminal). We can adapt the Bellman equation to this problem as:

$$J^*(x) = \max(G(x), \alpha \sum_{y \in S} P_{xy} J^*(y)) \tag{11}$$

The Bellman operator can now be written as:

$$TJ(x) = \max(G(x), \alpha \sum_{y \in S} P_{xy} J(y)) \tag{12}$$

$$TJ = \max(G, \alpha PJ) \tag{13}$$

where P is the transition matrix for the Markov chain. T is a weighted Euclidean norm contraction.

Theorem 3 $\|TJ - T\bar{J}\|_D \leq \alpha \|J - \bar{J}\|_D$

Clearly $|\max(a, b) - \max(a, c)| \leq |b - c|$. $\|TJ - T\bar{J}\|_D \leq \|\alpha PJ - \alpha P\bar{J}\|_D \leq \alpha \|J - \bar{J}\|_D$

5 Homework (due on Wednesday 05/24)

5.1 Question 1

In this problem we shall try to generalize the proof of the TD-λ algorithm to work with $Z_t = \sum_{\tau=0}^{t} \alpha^{t-\tau} C_{t-\tau} \phi(x_\tau)$ where $\sum_{k=0}^{\infty} C_k = 1, C_k \geq 0$. In particular, define a Lyapunov function $V(r)$, and compute $\bar{s}(r)$ and show that $(\nabla_r V(r))^T \bar{s}(r) < 0$ except at limit.

5.2 Question 2

Consider ODE $\dot{J}_t = FJ_t - J_t$. Assume $\exists J : J = FJ$. J is finite dimensional. F can be any (possibly non-linear) operator.

5.2.1 Warmup:

Prove that J_t converges if $\forall J, \bar{J}$, $\|FJ - F\bar{J}\|_2 \leq \|J - \bar{J}\|_2$

5.2.2 Challenge:

Prove that J_t converges if $\forall J, \bar{J}$, $\|FJ - F\bar{J}\|_\infty \leq \|J - \bar{J}\|_\infty$

MS&E339/EE337B Approximate Dynamic Programming Lecture 11 - 5/12/2004

Approximate Linear Programming Approach

Lecturer: Ben Van Roy *Scribe: Erick Delage and Penka Markova*

1 Bounds for Approximate Value Iteration

In Lecture 8 we proved that for approximate value iteration, i.e.

$$\Phi r_{k+1} = \Pi T \Phi r_k$$

we have convergence: $r_k \to \tilde{r}$, and

$$\|\Phi\tilde{r} - J^*\|_{2,\pi} \leq \frac{1}{\sqrt{1-\alpha^2}} \min_r \|\Phi r - J^*\|_{2,\pi} = O(\min_r \|\Phi r - J^*\|_{2,\pi}) \tag{1}$$

where $\|J\|_{2,\pi} = (\sum_x \pi(x) J^2(x))^{1/2}$. From this $\tilde{r}$ we can derive the greedy policy $\tilde{u}$:

$$T_{\tilde{u}} \Phi \tilde{r} = T \Phi \tilde{r}$$

and get an error bound on the greedy policy

$$E[J_{\tilde{u}}(x_t) - J^*(x_t)] = O(\|\Phi\tilde{r} - J^*\|)_{2,\pi}$$

Implementation using Temporal Difference learning on an autonomous system or in an optimal stopping problem was proven to have similar bounds but no such bounds could be proven for problems with a bigger set of policies. On the other hand, the Approximate Linear Programming approach provides similar results.

2 Approximate Linear Programming Approach

Recall how the DP equation:

$$J^* = \min_{u \in U(x)} \sum_{y \in S} p_{xy}(u)(g(x,u,y) + \alpha J^*(y))$$

is solved with LP:

$$\begin{aligned} \text{maximize} \quad & c^T J \\ \text{subject to} \quad & TJ \geq J \end{aligned} \tag{2}$$

where $c(x) > 0, \forall x$. In order to reduce the dimension of the problem, we consider the following approximate LP:

$$\begin{aligned} \text{maximize} \quad & c^T \Phi r \\ \text{subject to} \quad & T\Phi r \geq \Phi r \end{aligned} \tag{3}$$

In a problem where the state space is very large, we are still required to reduce the amount of computation:

- *Evaluating* $c^T \Phi r$ - This summation is of the size of the state space.
  A close enough approximation can be obtained from considering c as a probability distribution, sampling x according to c and computing $\sum_x \Phi r(x)$
- *Constraining to* $T\Phi r \geq \Phi r$ - A constraint for every state, so there are $O(|\mathcal{S}|)$ constraints.
  The solution is to sample a bunch and ignore others. We will discuss this issue in the next lecture.

In this lecture we assume we can solve the approximate LP (3), and focus on the properties of the solution.

2.1 Feasibility of Φr in the LP

Since Φr only ranges a subspace in the value function space, it is possible that none of the reachable Φr are in the feasibility region of the LP (3). This region is presented in figure 1.

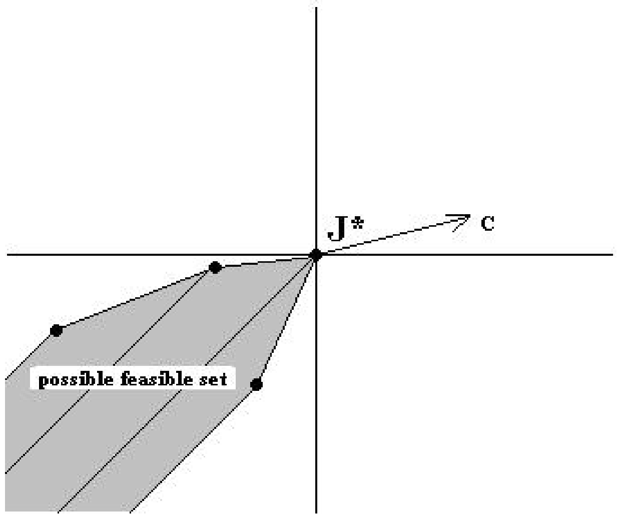


Figure 1: Feasibility region for LP

This region is necessarily included in the space for which $J \leq J^*$ as we discussed in Lecture 4 Theorem 1. The feasible space can then be sketched from a set of J known to be in the feasible space.
If J_0 and J_1 is in the set then all J's on the segment between J_0 and J_1 is in the set because of convexity. Also, if J_0 is in the set, then $J_0 - \gamma e$ is in the set, since

$$TJ_0 \geq J_0 \Rightarrow T(J_0 - \gamma e) = TJ_0 - T\gamma e = TJ_0 - \alpha\gamma e \geq J_0 - \gamma e$$

where $e(x) = 1$, $\forall x$ and $\gamma > 0$.

In the case where the span of Φ does not intersect with the LP feasible space, then the LP cannot be solved. Figure 2 shows how such a situation could occur.

Theorem 1 *Problem* (3) *has a feasible set if* $\exists r$ *such that* $\Phi r = e$ *where* $e(x) = 1, \forall x$.

Proof If $\Phi \hat{r} = e$ then $\Phi r = \Phi \bar{r} - \gamma e$ where $\bar{r} = r + \gamma \hat{r}$. The LP problem becomes: Let $c \geq 0$,

$$\begin{aligned} \text{maximize} \quad & c^T(\Phi\bar{r} - \gamma e) \\ \text{subject to} \quad & T(\Phi\bar{r} - \gamma e) \geq \Phi\bar{r} - \gamma e \end{aligned} \tag{4}$$

The feasible set of this problem is not empty since:

$$T(\Phi\bar{r} - \gamma e) \geq \Phi\bar{r} - \gamma e$$

$$T\Phi\bar{r} - \alpha\gamma e \geq \Phi\bar{r} - \gamma e$$

$$T\Phi\bar{r} \geq \Phi\bar{r} - (1 - \alpha)\gamma e$$

And there always exists a γ that will validate the inequality. □

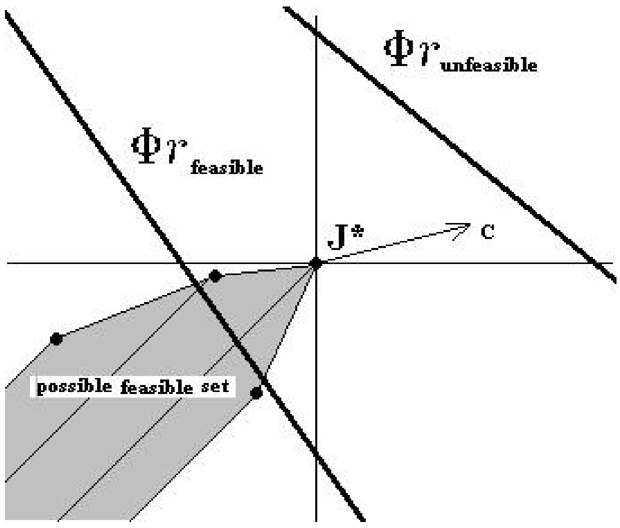


Figure 2: Approximate Value Function feasibility

2.2 Error Bond

Once the feasibility of Φr has been established, we can go back to determining the error bound for (3). The following result can be shown:

Theorem 2 *If $\Phi r = e$ for some r, and $\hat{r}$ is the optimal solution to LP* (3)*, then*

$$\|J^* - \Phi\hat{r}\|_{1,c} \leq \frac{2}{1-\alpha} \min_r \|\Phi r - J^*\|_\infty \tag{5}$$

where $c(x) > 0, \sum_x c(x) = 1, \|J\|_\infty = max_x|J(x)|$, and $\|J\|_{1,c} = \sum_x c(x)|J(x)|$.

Proof Let $r^* \in \arg\min_r \|\Phi r - J^*\|_\infty$ and $\varepsilon = \|\Phi r^* - J^*\|_\infty$. Then

$$\|T\Phi r^* - J^*\|_\infty \leq \alpha\|\Phi r^* - J^*\|_\infty \ = \alpha\varepsilon (\text{by}\alpha\text{-contraction, and by the definition of } \varepsilon).$$

$$\Longrightarrow T\Phi r^* \geq J^* - \alpha\varepsilon e \tag{6}$$

Since $T(J - \gamma e) = TJ - \alpha\gamma e$,

$$\begin{aligned} T(\Phi r^* - \gamma e) \ &= T\Phi r^* - \alpha\gamma e \\ &\geq J^* - \alpha\varepsilon e - \alpha\gamma e \text{ (by (6))} \\ &\geq \Phi r^* - \varepsilon e - \alpha\varepsilon e - \alpha\gamma e \\ &= \Phi r^* - \gamma e + ((1-\alpha)\gamma - (1+\alpha)\varepsilon)e \end{aligned} \tag{7}$$

Since e is in the span of Φ, hence there exists $\tilde{r}$ such that

$$\Phi\tilde{r} = \Phi r^* - \frac{(1+\alpha)}{1-\alpha}\varepsilon e.$$

Then by (7)

$$T\Phi\tilde{r} \geq \Phi\tilde{r}.$$

Going back to the result we are proving, we get

$$\begin{aligned}
\|J^* - \Phi\tilde{r}\|_{1,c} &\leq \|J^* - \Phi\tilde{r}\|_\infty \text{ (by the definitions of } \|\cdot\|_{1,c} \text{ and } \|\cdot\|_\infty\text{, and the restrictions on } c(x)) \\
&\leq \|J^* - \Phi r^*\|_\infty + \|\Phi r^* - \Phi\tilde{r}\|_\infty \text{ (by simple algebra and properties of max)} \\
&= \|J^* - \Phi r^*\|_\infty + \frac{(1+\alpha)\varepsilon}{1-\alpha} \\
&= \frac{2}{1-\alpha}\varepsilon
\end{aligned}$$

Lastly, $T\Phi r \geq \Phi r$ implies $\Phi r \leq J^*$, so maximizing $c^T\Phi r$ is equivalent to minimizing $c^T(J^* - \Phi r)$. So $\|J^* - \Phi\hat{r}\|_{1,c} \leq \|J^* - \Phi\tilde{r}\|_{1,c}$, and the result follows. □

While this is the type of bound we are looking for, there are still some problems with this result (including measuring error in terms of the max norm, the relationship between the $\|\cdot\|_{1,c}$ type of norm and performance loss).

MS&E339/EE337B Approximate Dynamic Programming Lecture 12 - 5/17/2004

Constraint Sampling

Lecturer: Ben Van Roy *Scribe: Jiarui Han and Ciamac Moallemi*

In the linear programming approach to DP, we either consider the exact LP

$$\begin{array}{ll} \text{maximize} & c^T J, \\ \text{subject to} & TJ \geq J, \end{array}$$

or the approximate LP

$$\begin{array}{ll} \text{maximize} & c^T \Phi r, \\ \text{subject to} & T\Phi r \geq \Phi r. \end{array}$$

Although the dimension of problem is reduced in the approximate version, the number of the constraints does not change and can be very large (or even infinite). Thus, it is still very hard to solve. In the constraint sampling approach, we solve the approximate LP using a randomly sampled subset of the constraints. This will yield reasonable policies, if we can prove the following two properties:

1. If we sample some reasonable number of constraints, then "almost all" others will be satisfied.
2. The constraints that are not satisfied don't distort the solution too much.

In this lecture, we will focus on the first property. The second property will be considered in the next lecture.

Consider the following general linear programming problem:

$$\begin{array}{ll} \text{maximize} & c^T x, \\ \text{subject to} & Ax \leq b. \end{array} \tag{1}$$

where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $m \gg n$ (or even $m = \infty$). Given a probability distribution μ over $\{1, \cdots, m\}$, sample the sequence $\{i_1, i_2, \ldots\}$ in an IID fashion according to μ. Define $\hat{x}_N$ as the optimal solution of the following LP

$$\begin{array}{ll} \text{maximize} & c^T x, \\ \text{subject to} & A_{i_j} x \leq b_{i_j}, \text{ for } j = 1, 2, \ldots, N, \end{array} \tag{2}$$

where A_{i_j} is the i_jth row of the matrix A.

We would like to establish the following theorem.

Theorem 1 *For arbitrary $\epsilon, \delta > 0$, if $N \geq n/(\epsilon\delta) - 1$, then*

$$\mathbb{P}\left\{\mu(\{i | A_i \hat{x}_N > b_i\}) \leq \epsilon\right\} \geq 1 - \delta, \tag{3}$$

where the probability is taken over the random sampling of constraints.

Here, ϵ represents a tolerance or control on how many constraints are allowed to be violated, and $1 - \delta$ represents a confidence level. The theorem states that given an ϵ and δ, the number of constraints we need for (3) to hold is linear in n, and, remarkably, does not depend on m.

Theorem 1 was originally established by de Farias and Van Roy in the context of approximate DP. The original derivation required sophisticated results from Vapnik-Chervonenkis theory. Subsequently, the result was established using the the idea of *support constraints* by Calafiore and Campi. This is the approach we will follow. This method is self-contained and extends to a broader class of convex optimization problems. These results are of independent interest to the convex optimization community in the context of robust optimization problems.

Definition 2 *Given an LP, a constraint is called a support constraint if the optimal objective value is changed if the constraint is relaxed.*

For example, in the left side of Figure 1, none of L1, L2, and L3 are support constraints; in the right side of Figure 1, L5 and L6 are support constraints while L4 is not. Notice that a support constraint must be an active constraint, but, in the case of degenerate vertices, active constraints need not be support constraints.

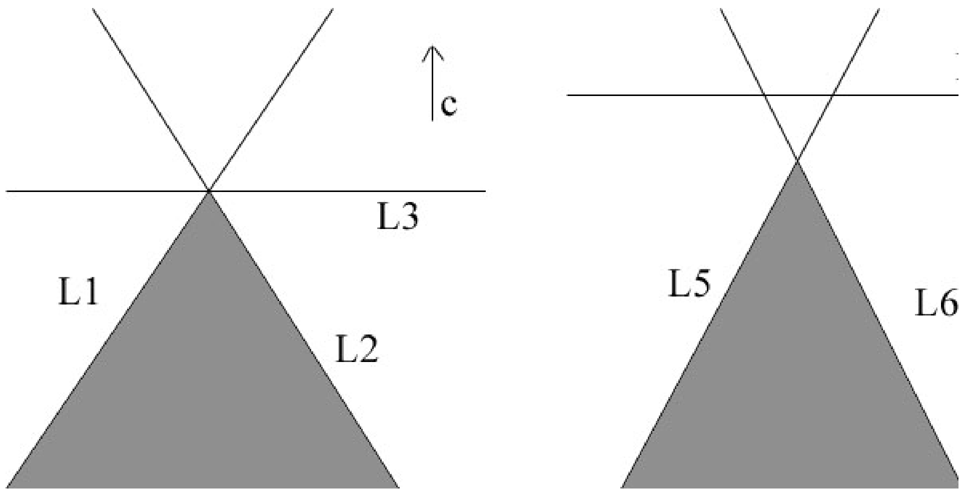


Figure 1: Support constraints.

The following theorem is a straightforward linear programming result that provides a bound on the number of support constraints.

Theorem 3 *If there are n variables in an LP, which is bounded and feasible, then there are at most n support constraints.*

Proof We'll use sensitivity analysis. Consider the dual of LP (1)

$$\begin{array}{lll} \text{minimize} & b^T y, & (y \in \mathbb{R}^m) \\ \text{subject to} & A^T y = c, & (n \text{ constraints}) \\ & y \geq 0. & (m \text{ constraints}) \end{array}$$

Notice that every vertex of the dual LP must have at least $m-n$ components which are zero. Since the LP is bounded and feasible, hence the dual always has an optimal solution y^* occuring at a vertex. If constraint i is a support constraint, then $y_i^* > 0$. So, the fact that there are at most n positive components in y^* implies that there are at most n support constraints. □

The following theorem will provide the fundamental bound necessary in the proof Theorem 1.

Theorem 4 *If $\hat{x}_N$ is the solution to the sampled LP (2), then*

$$\mathbb{E}\left[\mu\left(\{i : A_i \hat{x}_N > b_i\}\right)\right] \leq \frac{n}{N+1},$$

where the expectation above is taken over the random sampling of constraints.

Proof Given a sequence $\{i_1, \ldots, i_N, i_{N+1}\}$ sampled IID according to the dirstibuted μ, define $\hat{x}_N^k$ to be the solution of the LP

$$\begin{array}{ll} \text{maximize} & c^T x, \\ \text{subject to} & A_{i_j} x \le b_{i_j}, \text{ for } j \in \{1, 2, \ldots, N+1\} - \{k\}. \end{array}$$

Note that $\hat{x}_N^{N+1} = \hat{x}_N$. Then, by symmetry,

$$\mathbb{P}\left\{A_{i_{N+1}} \hat{x}_N > b_{i_{N+1}}\right\} = \mathbb{P}\left\{A_{i_k} \hat{x}_N^k > b_{i_k}\right\}.$$

Now, consider the event that $A_{i_k} \hat{x}_N^k > b_{i_k}$. If we consider the original sampled LP (2) for $\hat{x}_{N+1}$, notice that this event occurs only if i_k corresponds to a support constraint for $\hat{x}_{N+1}$. Since at most n of the $N+1$ constraints are support constraints, we have

$$\mathbb{P}\left\{A_{i_{N+1}} \hat{x}_N > b_{i_{N+1}}\right\} = \mathbb{P}\left\{A_{i_k} \hat{x}_N^k > b_{i_k}\right\} \le \frac{n}{N+1}.$$

Let P_N denote the distribution of $\hat{x}_N$. Then,

$$\begin{aligned} \mathbb{P}\left\{A_{i_{N+1}} \hat{x}_N > b_{i_{N+1}}\right\} &= \sum_{j=1}^m \int_{\mathbb{R}^n} \mu(j) 1_{\{A_j x > b_j\}} P_N(dx) \\ &= \int_{\mathbb{R}^n} \mu(\{i | A_i x > b_i\}) P_N(dx) \\ &= \mathbb{E}\left[\mu\left(\{i : A_i \hat{x}_N > b_i\}\right)\right]. \end{aligned}$$

The result follows. $\square$

We are ready to prove Theorem 1. Observe that if $N > n/(\epsilon\delta) - 1$, using Markov's Inequality,

$$\begin{aligned} \mathbb{P}\left\{\mu(\{i | A_i \hat{x}_N > b_i\}) > \epsilon\right\} &\le \frac{1}{\epsilon} \mathbb{E}\left[\mu(\{i | A_i \hat{x}_N > b_i\})\right] \\ &\le \frac{n}{\epsilon(N+1)} \\ &\le \frac{1}{\epsilon} \frac{n}{(n/(\epsilon\delta) - 1) + 1} = \delta. \end{aligned}$$

Homework Problem: Notice that the lower bound for N is Theorem 1 was order $O(n/(\epsilon\delta))$. Using Theorem 1, prove a lower bound for N of order

$$O\left(\frac{1}{\epsilon}\left[n \ln \frac{1}{\epsilon} + \ln \frac{1}{\delta}\right]\right).$$

MS&E339/EE337B Approximate Dynamic Programming Lecture 13 - 5/19/2004

Constraint Sampling for the ADP Convex Problems

Lecturer: Ben Van Roy *Scribe: Mark Peters and Mike Rotkowitz*

Last time, we considered the general LP

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \le b \end{array} \tag{1}$$

where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $m \gg n$. Given a probability distribution μ over $\{1, \cdots, m\}$, we sampled the sequence $\{i_1, i_2, \ldots, i_N\}$ of constraints in an IID fashion according to μ. Defining $\hat{x}_N$ as the optimal solution of the ensuing LP,

$$\begin{array}{lll} \text{maximize} & c^T x & \\ \text{subject to} & A_{i_j} x \le b_{i_j} & \text{for all } j = 1, 2, \ldots, N \end{array} \tag{2}$$

where A_{i_j} is the i_jth row of the matrix A, we established the following result.

Theorem 1 *For arbitrary $\epsilon, \delta > 0$, if $N \ge \frac{n}{\epsilon\delta} - 1$, then*

$$\mathbb{P}\left\{\mu\big(\{i \mid A_i \hat{x}_N > b_i\}\big) \ \le\ \epsilon\right\} \ \ge\ 1 - \delta, \tag{3}$$

where the probability is taken over the random sampling of constraints.

In this lecture, we'd like to leverage this result for approximate dynamic programming. First we note that the proof in the previous lecture did not depend on the constraints being linear, only convex. So we can generalize this result. Consider the convex optimization problem

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & g(x) \le b \end{array} \tag{4}$$

where $x \in \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^m$ convex, and $m \gg n$.

Similarly sample the sequence $\{i_1, i_2, \ldots, i_N\}$ of constraints in an IID fashion according to μ. Defining $\hat{x}_N$ as the optimal solution of the ensuing convex problem,

$$\begin{array}{lll} \text{maximize} & c^T x & \\ \text{subject to} & g_i(x) \le b_i & \text{for all } i \in \{i_1, i_2, \ldots, i_N\} \end{array} \tag{5}$$

where g_i is the ith convex constraint. We then have the following generalized result.

Theorem 2 *For arbitrary $\epsilon, \delta > 0$, if $N \ge \frac{n}{\epsilon\delta} - 1$, then*

$$\mathbb{P}\left\{\mu\big(\{i \mid g_i(\hat{x}_N) > b_i\}\big) \ \le\ \epsilon\right\} \ \ge\ 1 - \delta, \tag{6}$$

where the probability is taken over the random sampling of constraints.

We now consider the ADP LP

$$\begin{array}{lll} \text{maximize} & c^T \Phi r & \\ \text{subject to} & (T\Phi r)(x) \ge (\Phi r)(x) & \text{for all } x \in S \end{array} \tag{7}$$

and also consider this problem with sampled constraints and an additional constraint

$$\begin{array}{lll} \text{maximize} & c^T \Phi r & \\ \text{subject to} & (T\Phi r)(x) \ge (\Phi r)(x) & \text{for all } x \in \{x_1, \ldots, x_N\} \\ & r \in \mathcal{N} & \end{array} \tag{8}$$

where $\mathcal{N}$ is a bounded convex set which will prevent the optimization from taking too much advantage of excluded constraints. Let $\tilde{r}$ be the solution to problem (7) and let $\hat{r}$ be the solution to problem (8).

We recall that the weighted 1-norm of a vector is defined as $\|J\|_{1,c} = \sum c(x)|J(x)|$. Last week we derived a bound for $\|J^* - \Phi\tilde{r}\|_{1,c}$, and now we would like to have a bound for $\|J^* - \Phi\hat{r}\|_{1,c}$.

Define the probability distribution π_α as

$$\pi_\alpha = (1-\alpha)c^T(I - \alpha P_{\mu^*})^{-1}$$

Where μ^* is the optimal policy and P_{μ^*} is the transition matrix of the Markov chain under the optimal policy. Also, note that π_α can also be expressed as

$$\pi_\alpha = (1-\alpha)\sum_{t=0}^{\infty} \alpha^t c^T P_{\mu^*}^t$$

Further, we know that $\sum_{t=0}^{\infty} \alpha^t = \frac{1}{1-\alpha}$.

Note that $c^T P_{\mu^*}^t$ is the distribution of the Markov chain at time t when it starts from the distribution c^T. Thus, π_α can be viewed as an expected distribution of the Markov chain where the impact of the initial distribution c^T is being weighted by the value of α^t in the summation. Thus if α is close to zero, then the early terms in the summation will dominate the later terms and c^T will play a large role in determining π_α. However, as $\alpha \to 1$, then c^T will have less impact on the value of π_α because the future terms will have roughly the same weight as the early terms. Thus, the stationary distribution of P_{μ^*} will dominate.

One major problem here is that we are employing circular logic since π_α depends on the optimal policy. Unfortunately, in general, we can't use another distribution and have the result hold. However, there probably are some classes of problems where another distribution could be safely substituted.

Letting the constraints in problem (8) be sampled according to π_α, we get the following result.

Theorem 3 *If* $N \geq \frac{4K}{(1-\alpha)\epsilon\delta} \frac{\sup_{r\in\mathcal{N}}\|J^*-\Phi r\|_\infty}{c^T J^*}$ *then*

$$\|J^* - \Phi\hat{r}\|_{1,c} \leq \|J^* - \Phi\tilde{r}\|_{1,c} + \epsilon\,\|J^*\|_{1,c} \qquad \textit{with probability } 1-\delta$$

Proof We let g be the standard vector of cost as a function of state such that $J^* = (I - \alpha P_{\mu^*})^{-1} g = \sum_{t=0}^{\infty} \alpha^t P_{\mu^*}^t g$. We make use of the notation

$$x^+ = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \qquad x^- = \begin{cases} |x| & \text{if } x \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

such that $x^+ + x^- = x + 2x^-$.

$$\begin{aligned}
\|J^* - \Phi\hat{r}\|_{1,c} &= c^T|J^* - \Phi\hat{r}| \qquad \text{where the absolute value is taken component-wise} \\
&\leq c^T(I-\alpha P_{\mu^*})^{-1}|g - (I-\alpha P_{\mu^*})\Phi\hat{r}| \qquad \text{since the inverse has only non-negative elements} \\
&= c^T(I-\alpha P_{\mu^*})^{-1}\big((g-(I-\alpha P_{\mu^*})\Phi\hat{r})^+ + (g-(I-\alpha P_{\mu^*})\Phi\hat{r})^-\big) \\
&= c^T(I-\alpha P_{\mu^*})^{-1}\big((g-(I-\alpha P_{\mu^*})\Phi\hat{r}) + 2(g-(I-\alpha P_{\mu^*})\Phi\hat{r})^-\big) \\
&= c^T(J^* - \Phi\hat{r}) + 2c^T(I-\alpha P_{\mu^*})^{-1}(T_{\mu^*}\Phi\hat{r} - \Phi\hat{r})^- \qquad \text{where } (T_{\mu^*}\Phi\hat{r} = g + \alpha P_{\mu^*}\Phi\hat{r}) \\
&\leq c^T(J^* - \Phi\tilde{r}) + 2c^T(I-\alpha P_{\mu^*})^{-1}(T_{\mu^*}\Phi\hat{r} - \Phi\hat{r})^-
\end{aligned}$$

since $\hat{r}$ has fewer constraints and thus $c^T\Phi\hat{r} > c^T\Phi\tilde{r}$

$$\leq \|J^* - \Phi\tilde{r}\|_{1,c} + 2c^T(I-\alpha P_{\mu^*})^{-1}(T_{\mu^*}\Phi\hat{r} - \Phi\hat{r})^-$$

Then, focusing on the right-hand side of the last term of the inequality, we have

$$2c^T(I - \alpha P_{\mu^*})^{-1}(T_{\mu^*}\Phi\hat{r} - \Phi\hat{r})^- = \frac{2}{1-\alpha}\pi^T(T_{\mu^*}\Phi\hat{r} - \Phi\hat{r})^-$$

We note that $(T_{\mu^*}\Phi\hat{r} - \Phi\hat{r})^- = 0$ if $T_{\mu^*}\Phi\hat{r} \geq \Phi\hat{r}$ and thus $(T_{\mu^*}\Phi\hat{r} - \Phi\hat{r})^- = 0$ if $T\Phi\hat{r} \geq \Phi\hat{r}$ as well. Thus, this will equal zero if a constraint is being satisfied.

$$2c^T(I - \alpha P_{\mu^*})^{-1}(T_{\mu^*}\Phi\hat{r} - \Phi\hat{r})^- \leq \frac{2}{1-\alpha}\pi^T(T\Phi\hat{r} - \Phi\hat{r})^-$$

This is non-zero for each violated constraint and its magnitude is bounded by our bounding box

$$2c^T(I - \alpha P_{\mu^*})^{-1}(T_{\mu^*}\Phi\hat{r} - \Phi\hat{r})^- \leq \frac{2}{1-\alpha}\pi(\text{violated constraint}) \sup_{r\in\mathcal{N}} \|T\Phi r - \Phi r\|_\infty$$

Now, we would like to convert this inequality into a probabilistic statement so we will refer back to Theorem 2 and insert the following substitution.

$$\pi(\text{violated constraint}) \leq \epsilon_1 \qquad \text{with probability } 1-\delta \text{ where}$$

$$\epsilon_1 = \frac{(1-\alpha)\epsilon}{4}\frac{c^T J^*}{\sup_{r\in\mathcal{N}}\|J^* - \Phi r\|_\infty}$$

Furthermore, using the triangle inequality and the fact that T is an α contraction, we will also make this substitution

$$\sup_{r\in\mathcal{N}} \|T\Phi r - \Phi r\|_\infty \leq (1+\alpha)\sup_{r\in\mathcal{N}}\|J^* - \Phi r\|_\infty$$

After making these two substitutions, the following inequality will hold with probability $1-\delta$.

$$\begin{aligned} 2c^T(I - \alpha P_{\mu^*})^{-1}(T_{\mu^*}\Phi\hat{r} - \Phi\hat{r})^- &\leq \frac{2}{1-\alpha}\frac{(1-\alpha)\epsilon}{4}\frac{c^T J^*}{\sup_{r\in\mathcal{N}}\|J^* - \Phi r\|_\infty}(1+\alpha)\sup_{r\in\mathcal{N}}\|J^* - \Phi r\|_\infty \\ &\leq \epsilon c^T J^* \\ &= \epsilon \|J^*\|_{1,c} \end{aligned}$$

Thus, we have shown

$$\|J^* - \Phi\hat{r}\|_{1,c} \leq \|J^* - \Phi\tilde{r}\|_{1,c} + \epsilon\|J^*\|_{1,c} \qquad \text{with probability } 1-\delta$$

□

MS&E339/EE337B Approximate Dynamic Programming Lecture 12 - 5/24/2004

Average Cost Problems

Lecturer: Ben Van Roy *Scribe: Benjamin Van Roy*

1 Introduction

We have studied a linear program that approximates the optimal cost-to-go function for a discounted problem:

$$\begin{array}{ll} \text{maximize} & c^T \Phi r, \\ \text{subject to} & T\Phi r \geq \Phi r. \end{array}$$

We established an error bound for a parameter vector $\tilde{r}$ that attains the optimum of this linear program: if there is a vector r such that $\Phi r = e$ then,

$$\|J^* - \Phi \tilde{r}\|_{1,c} \leq \frac{2}{1-\alpha} \min_{r \in \Re^K} \|J^* - \Phi r\|_\infty.$$

Though this is an interesting result that provides some confidence in the approximation algorithm, it fails to address two important issues:

1. We are ultimately interested in *performance* of the resulting greedy policy, not just *error* in approximating J^*. In particular, if we use a greedy policy

$$\tilde{\mu}(x) \in \arg\min_u \sum_y p_{xy}(u) \left(g(x) + \alpha(\Phi \tilde{r})(y) \right),$$

what can we say about its cost-to-go $\tilde{J}_{\tilde{\mu}}$ relative to the optimal cost-to-go J^*?

2. What role does c play? In the result, c influences the metric with which we are assessing approximation error. But how should this metric be chosen so that small approximation error translates to good performance of the resulting policy?

In order to address these issues in an elegant manner, we will work with an average cost formulation. We could also treat these issues in a discounted cost framework, but its messier, so we choose to work with average cost. Since not everyone in the class is familiar with average cost dynamic programming, we develop the framework in this lecture.

2 Problem Formulation and Notation

Consider a stochastic system with dynamics characterized by transition probabilities $p_{xy}(u)$. There is a cost $g(x) \geq 0$ incurred when as state x. Under each policy μ, the system evolves as a Markov chain. We assume that for each μ, the resulting Markov chain has a unique steady-state distribution π_μ, which is strictly positive; i.e., $\pi_\mu(x) > 0$ for all x. The average cost associated with a policy μ is denoted by λ_μ. Note that

$$\lambda_\mu = \sum_x \pi_\mu(x) g(x) = \pi_\mu^T g.$$

The problem is to find a policy that attains minimal average cost:

$$\min_\mu \lambda_\mu.$$

We define a dynamic programming operator

$$(TJ)(x) = \min_u \sum_y p_{xy}(u)(g(x) + J(y)).$$

Note that this is the same as the dynamic programming operator defined in the context of average cost problems, but now the discount factor is set to 1.

3 The Primal LP

We begin by introducing a simple linear programming approach for generating an optimal policy. The LP decision variables are state-action probabilities $\rho(x,u)$. Note that there is one decision variable per state-action pair. Each $\rho(x,u)$ represents the fraction of time, in steady-state, that the system is in state x and action u is selected. Given state-action frequencies ρ, one can define a (possibly randomized) policy that attains the state action frequencies:

$$\Pr\{u_t = u | x_t = x\} = \frac{\rho(x,u)}{\sum_{\overline{u}} \rho(x,\overline{u})}.$$

Note that this construction makes sense only if the probability of being in state x is positive. This is implied by our assumption that every deterministic policy results in positive state probabilities.

The linear program optimizes average cost over the space of feasible state-action frequencies:

$$\begin{array}{lll} \min_\rho & \sum_{x,u} \rho(x,u) g(x) & \\ \text{s.t.} & \sum_{x,u} \rho(x,u) p_{xy}(u) = \sum_u \rho(y,u) & \forall y \\ & \sum_{x,u} \rho(x,u) = 1 & \\ & \rho(x,u) \geq 0 & \forall x, u. \end{array}$$

The last two sets of constraints ensure that ρ is a probability distribution. The first set of constraints restrict ρ based on the transition probabilities. It is easy to see that any optimal solution provides state action frequencies that minimize average cost, and that the optimal objective value is $\min_\mu \lambda_\mu$.

As with any linear program, the feasible set of ours forms a polytope. It is easy to show that, in our particular case, this polytope is bounded and that the vertices correspond to deterministic strategies. In particular, at any vertex ρ, for any state x, there exists an action $\overline{u}$ such that $\rho(x,\overline{u}) > 0$ and $\rho(x,u) = 0$ for $u \neq \overline{u}$.

Let ρ^* be an optimal vertex.

4 The Dual LP

The dual of the linear program introduced in the previous section can be written as

$$\begin{array}{lll} \max_{\lambda,h} & \lambda & \\ \text{s.t.} & g(x) - \lambda + \sum_y p_{xy}(u) h(y) \geq h(x) & \forall x, u. \end{array}$$

Let (λ^*, h^*) be an optimal solution. By the duality theorem, $\lambda^* = \min_\mu \lambda_\mu$.

Note that each (x,u)th constraint in the dual corresponds to a variable $\rho(x,u)$ in the primal. If $\rho(x,u) > 0$, then the corresponding constraint in the dual is binding. It follows that

$$\min_u \left(g(x) - \lambda^* + \sum_y p_{xy}(u) h^*(y) \right) = h^*(x).$$

In other words, (λ^*, h^*) satisfies a version of Bellman's equation:

$$Th^* - \lambda^* e = h^*.$$

Further, since $\rho(x,u) > 0$ if and only if action u is an optimal decision at state x, action

$$\overline{u} \in \arg\min_u \left(g(x) - \lambda^* + \sum_y p_{xy}(u) h^*(y) \right),$$

if and only if $\overline{u}$ is an optimal action at state x. In other words, a policy μ^* is optimal if and only if

$$T_{\mu^*} h^* = Th^*.$$

5 The Differential Cost Function

How should we interpret the function h^*? Well, we know that

$$h^* = Th^* - \lambda^* e = T_{\mu^*} h^* - \lambda^* e,$$

for an optimal policy μ^*. Hence, letting P_{μ^*} denote the transition matrix of the optimal policy μ^*, we have

$$h^* = g - \lambda^* e + P_{\mu^*} h^*.$$

Now what is the set of solutions h to the equation

$$h = g - \lambda^* e + P_{\mu^*} h?$$

Well, the larges eigenvalue of P_{μ^*} is equal to one, and the corresponding right eigenvector is e. The absolute value of every other eigenvalue is strictly less than one. Hence, the set of solutions h is the one-dimensional affine subspace $H = \{h^* + \gamma e | gamma \in \Re\}$. Further, for any h that solves this equation, (λ^*, h) is a feasible and therefore optimal solution to the dual linear program. Hence, the set of optimal solutions to the dual linear program is $(lambda^*, h) | h \in H\}$.

We started by taking (λ^*, h^*) to be an arbitrary optimal solution to the dual linear program. Since there are many possibilities for h^*, in order to avoid ambiguity, let h^* be the element of H for which $\pi_{\mu^*}^T h^* = 0$, for a distinguished optimal policy μ^*. Since

$$h^* = g - \lambda^* e + P_{\mu^*} h^*,$$

we have

$$h^* = \sum_{t=0}^{T-1} P_{\mu^*}^t (g - \lambda^* e) + P_{\mu^*}^T h^*.$$

The fact that $\pi_{\mu^*}^T g = \lambda^*$ and $\pi_{\mu^*}^T h^* = 0$ imply that $P_{\mu^*}^t (g - \lambda^* e)$ and $P_{\mu^*}^t h^*$ each converge to zero at an exponential rate. It follows that the limit

$$\lim_{T\to\infty} \sum_{t=0}^{T-1} P_{\mu^*}^t (g - \lambda^* e) + P_{\mu^*}^T h^*,$$

is well-defined and finite, and that

$$h^* = \sum_{t=0}^{\infty} P_{\mu^*}^t (g - \lambda^* e).$$

Another way of writing this is

$$h^*(x) = \lim_{T \to \infty} E\left[\sum_{t=0}^{T} (g(x_t) - \lambda^*) \middle| x_0 = x, u_t = \mu^*(x_t) \right].$$

For this reason, h^* is called the *differential cost function*. It represents the optimal sum of future costs, where each future cost is offset by subtracting the long-term average.

MS&E339/EE337B Approximate Dynamic Programming Lecture 15 - 5/26/2004

Average Cost & Discounted Average Cost Problems

Lecturer: Ben Van Roy *Scribe: Erick Delage and Lykomidis Mastroleon*

1 Average Cost Dynamic Programming

In the previous lecture we examined the average cost dynamic programming formulation and we introduced a simple linear programming approach for generating an optimal policy. More specifically:

- We assumed all policies generated irreducible and aperiodic Markov Chains
- There was a cost function $g(x)$
- The transition probabilities were $p_{xy}(\mu)$
- The dual LP was :$\{ \begin{array}{ll} \max & \lambda \\ \text{s.t.} & Th - \lambda \geq h \end{array}$

Given that (λ^*, h^*) is an optimal solution of the dual LP we showed that:

1. $\lambda^* = \min_\mu \lambda_\mu$ (using the duality theorem)
2. $h^* = Th^* - \lambda^* e$
3. μ^* optimal iff $Th^* = T_{\mu^*} h^*$

2 Interpretation of h^*

Based on the results of the previous section we have:

$$h^* = Th^* - \lambda^* e = T_{\mu^*} h^* - \lambda^* e \tag{1}$$

Notice that the right hand of equation (1) is basically a set of linear equations (as T_{μ^*} is a linear operator). More specifically:

$$h^* = g + P_{\mu^*} h^* - \lambda^* e \tag{2}$$

Now lets define the following set:

$$H = \{h | h = g + P_{\mu^*} h - \lambda^* e\} \tag{3}$$

where P_{μ^*} is the transition matrix for the irreducible and aperiodic Markov chain.
To continue our analysis we will need to use Perron-Frobenius theory. In particular we will need to use the following theorem (no proof is provided):

Theorem 1 *If P is irreducible and aperiodic then the following statements are true:*

- *The maximum eigenvalue of P is* 1
- *All the other eigenvalues of P are strictly less than* 1
- *There exists a right eigenvector $e = (1, 1, ..., 1)^\top$ such that $Pe = e$*

With this theorem in mind we can look again equation (3) and rewrite it in the following form:

$$H = \{h | (I - P_{\mu^*})h = g - \lambda^* e\} \tag{4}$$

We can make certain observations regarding the nullspace $N\{(I - P_{\mu^*})\}$ of $(I - P_{\mu^*})$:

- $(I - P_{\mu^*})e = e - P_{\mu^*}e = 0 => e \in N\{(I - P_{\mu^*})\}$
- $(I - P_{\mu^*})u_i = u_i - \lambda_i u_i \neq 0$ since $\lambda_i < 1 => u_i$ not in $N\{I - P_{\mu^*})\}$ (where u_i is any eigenvector of P_{μ^*} other than e and λ_i is the corresponding eigenvalue)
- $N\{(I - P_{\mu^*})\} = \{\gamma e | \gamma \in \mathbb{R}\}$ as it can be easily seen from the 2 previous observations

So if $h^* \in H$ then $(h^* + \gamma e) \in H \; \forall \; \gamma \in \mathbb{R}$. It's trivial to verify that any $h \in H$ is a feasible solution for the dual LP and so the solution h^* we actually receive from the dual LP is arbitrary. It will be convenient for us to select h^* as the element of this space that has the following property:

$$\pi_{\mu^*}^{\tau} h^* = 0 \tag{5}$$

Now if iterate equation (2) then we get the following:

$$\begin{aligned} h^* &= g + P_{\mu^*}h^* - \lambda^* e \\ &= g + P_{\mu^*}(g + P_{\mu^*}h^* - \lambda^* e) - \lambda^* e \\ &= g + P_{\mu^*}g + P_{\mu^*}^2 h^* - 2\lambda^* e \\ &= (g - \lambda^* e) + P_{\mu^*}(g - \lambda^* e) + P_{\mu^*}^2 h^* \\ &= \sum_{t=0}^{\tau-1} P_{\mu^*}^t (g - \lambda^* e) + P_{\mu^*}^{\tau} h^* \\ &= \sum_{t=0}^{\infty} P_{\mu^*}^t (g - \lambda^* e) \end{aligned}$$

where we used the fact that $lim_{\tau\to\infty} P_{\mu^*}^{\tau} h^* = \pi_{\mu^*}^T h^* = 0$.

Now we can write : $h^*(x) = E\{\sum_{t=0}^{\infty}(g(x_t) - \lambda^*) | x_0 = x, \mu^*\}$

$h^*(x)$ is often called "Differential cost-to-go function" as it shows how much the cost-to-go increases when the process is started in state x compared to when it is started in the steady state distribution.

3 The "Restart" Perturbation

From now on, we will be working with a perturb version of the average cost dynamic problem. In this version, the system is considered to have probability $(1 - \alpha)$ of getting into a restart distribution at each transition step. The new problem is entirely similar to the previous one with a minor change to the transition probability matrix.

$$P_{xy}^{\alpha,c}(\mu) = (1 - \alpha)c(y) + \alpha P_{xy}(\mu)$$

Obviously, we recover the traditional average cost problem when α is equal to 1.

Now we would like to a bound on how far from the original problem this perturbation brings us. The measure of difference we are interested in is the average cost when following a policy, since this tells us how likely we are of following the wrong policy and a larger average cost than necessary.

Theorem 2

$$\forall \alpha \in (0,1), \mu, c, \quad |\lambda_{\alpha,c,\mu} - \lambda_\mu| \leq z\frac{1-\alpha}{1-\alpha\nu}$$

This limits the disturbance to $O(1-\alpha)$.

Proof First:

$$\begin{aligned} P_\mu^t &\rightarrow e\pi_\mu^t \quad \text{all other eigenvalues vanish to 0} \\ c^T P_\mu^t &\rightarrow c^T e\pi_\mu^t \\ c^T P_\mu^t g &\rightarrow \pi_\mu^t g \quad \text{, since } c \text{ is a probability distribution and sums to 1} \end{aligned}$$

Therefore:

$$|c^T P_\mu^t g - \pi_\mu^t g| \leq z\nu^t \text{ for some } \nu < 1, \forall \mu, t \tag{6}$$

Notice also that the following expression holds:

$$\pi_{\alpha,c,\mu} = \sum_{t=0}^{\infty}(1-\alpha)\alpha^t c^T P_\mu^t \tag{7}$$

Using equations (6) and (7), we can know construct a bound on $|\lambda_{\alpha,c,\mu} - \lambda_\mu|$:

$$\begin{aligned} |\lambda_{\alpha,c,\mu} - \lambda_\mu| &= |\pi_{\alpha,c,\mu}^T g - \pi_\mu^T g| \\ &= |\sum_{t=0}^{\infty}(1-\alpha)\alpha^t c^T P_\mu^t g - \pi_\mu^T g| \quad \text{, using (7)} \\ &\leq \sum_{t=0}^{\infty}(1-\alpha)\alpha^t |c^T P_\mu^t g - \pi_\mu^T g| \quad \text{, from the triangle inequality} \\ &\leq (1-\alpha)\sum_{t=0}^{\infty}\alpha^t z\nu^t \\ &= \frac{1-\alpha}{1-\alpha\mu}z \end{aligned}$$

□

4 The Perturbed Average Cost Approximate LP

Remember the Approximate Linear Program for the discounted dynamic program:

$$\begin{aligned} \text{maximize} \quad & c^T \Phi r \\ \text{subject to} \quad & T\Phi r \geq \Phi r \end{aligned} \tag{8}$$

In the case of the perturbed average cost dynamic problem, we can approximate the optimal average cost function by solving the linear program:

$$\begin{aligned} \text{minimize} \quad & \lambda \\ \text{subject to} \quad & T_{c,\alpha}\Phi r - \lambda e \geq \Phi r \end{aligned} \tag{9}$$

Theorem 3 *Let $(\tilde{\lambda}, \tilde{r})$ be an optimal solution of the approximate linear program.*
Also let $\tilde{\mu}$ satisfy $T_{c,\alpha,\tilde{\mu}}\Phi r = T_{c,\alpha}\Phi r$.

Then,

$$|\lambda_{c,\alpha,\tilde{\mu}} - \lambda_{c,\alpha,\mu^*}| \leq \frac{2\theta}{1-\alpha} \min_r \|h^*_{c,\alpha} - \Phi r\|_\infty$$

Where $\theta = \max_{(r,\lambda)\ feasible} \frac{c^T(T_{c,\alpha}\Phi r - \Phi r - \lambda e)}{\pi^T_{\alpha,c,\mu}(T_{c,\alpha}\Phi r - \Phi r - \lambda e)}$

The proof of this bound will be develop in the next lecture. However, we have already interesting comments to make about the nature of this bound.

- $|\lambda_{\tilde{\mu}} - \lambda^*| \leq |\lambda_{c,\alpha,\tilde{\mu}} - \lambda_{c,\alpha,\mu^*}| + O(1-\alpha)$
  We are already interested in making α close to one to reduce this bound
- θ *value*
  Although the equation that defines θ is complicated we will find a way if making *theta* ≈ 1 by choosing c appropriately
- $\|\cdot\|_\infty$
  We have seen that the infinity norm is not satisfying limiting bounds. There is ways of getting to a similar inequality without this type of norm.
- $\frac{1}{1-\alpha}$
  Here we notice that we have contradictive objectives for α: setting it close 1 as stated earlier will make this fraction grow out of proportions. There are no solution yet for this problem.