Optimal Planning
Wednesday, November 19, 2014
9:48 AM

OPTIMAL PLANNING

Overview

When the MDP dynamics ($P^a_{ss'}$, $R^a_{ss'}$) are known, the Bellman (optimality) equations may be applied iteratively to compute optimal value functions, from which we may determine an optimal policy. These iterative algorithms are examples of dynamic programming algorithms.

Policy Evaluation

Let π be an arbitrary policy. Define a sequence of functions $V_0, V_1, \ldots$ according to

$$V_{k+1}(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s') \right].$$

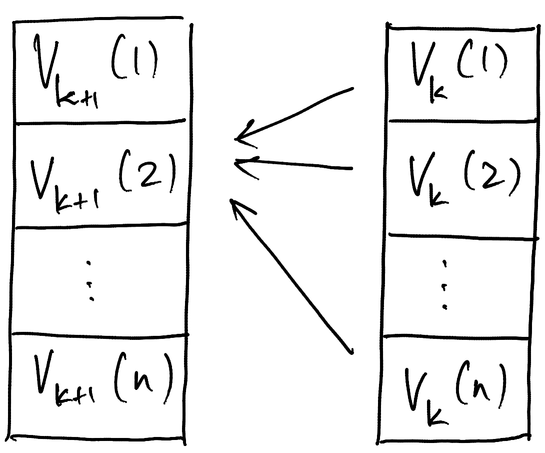
It can be shown that $\{V_k\}$ converges to V^π. When $|S|$ is large, this procedure is much more efficient than solving the Bellman equations directly.

V_0 may be arbitrary except that we require

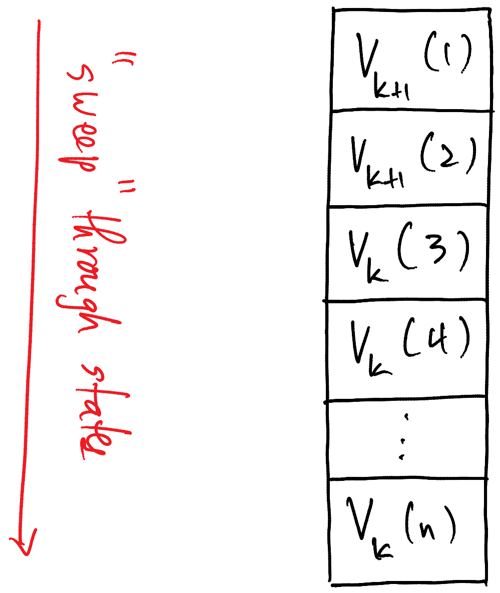
$$V_0(\text{terminal state}) = 0$$

for episodic tasks.

To implement the algorithm, either one array or two may be used — convergence is guaranteed in either case.



two arrays



one array

"in place" updates

$\boxed{\text{Policy Improvement}}$

If we can compute V^π, then we can compute

$$Q^\pi(s,a) = \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V^\pi(s') \right].$$

If $Q^\pi(s,a) > V^\pi(s)$ for some a, then it would seem that π is suboptimal, and can be improved upon by selecting action a when in state s. Formally, define

$$\pi' = \arg\max_a Q^\pi(a,s).$$

We have the following result, known as the policy improvement theorem.

Theorem $\pi' \geq \pi$, i.e., $V^{\pi'}(s) \geq V^\pi(s)$ $\forall s$.

Furthermore, if π is not optimal, then $\exists s$ s.t. $V^{\pi'}(s) > V^\pi(s)$.

Proof:
$$\begin{aligned}
V^\pi(s) &\leq Q^\pi(s, \pi'(s)) \\
&= \mathbb{E}_{\pi'}\{r_{t+1} + \gamma \underbrace{V^\pi(s_{t+1})}_{\text{Note: this is just a constant}} \mid s_t = s\} \\
&\leq \mathbb{E}_{\pi'}\{r_{t+1} + \gamma Q^\pi(s_{t+1}, \pi'(s_{t+1})) \mid s_t = s\} \\
&= \mathbb{E}_{\pi'}\{r_{t+1} + \gamma \mathbb{E}_{\pi'}\{r_{t+2} + \gamma V^\pi(s_{t+2})\} \mid s_t = s\} \\
&= \mathbb{E}_{\pi'}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 V^\pi(s_{t+2}) \mid s_t = s\} \\
&\leq \mathbb{E}_{\pi'}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V^\pi(s_{t+3}) \mid s_t = s\} \\
&\vdots
\end{aligned}$$

$$\mathbb{E}_{\pi'}\{ r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots \mid s_t = s \}$$

$$= V^{\pi'}(s)$$

To establish the second statement, we show the contrapositive. Thus, suppose $V^{\pi}(s) = V^{\pi'}(s) \ \forall s$. This implies $Q^{\pi}(s,a) = Q^{\pi'}(s,a) \ \forall s, a$. Then $\forall s$

$$\begin{aligned} V^{\pi}(s) &= V^{\pi'}(s) \\ &= Q^{\pi'}(s, \pi'(s)) \\ &= Q^{\pi}(s, \pi'(s)) \\ &= \max_a Q^{\pi}(s,a) \\ &= \max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^{\pi}(s') \right]. \end{aligned}$$

Since V^{π} satisfies the Bellman optimality equations, π is optimal. $\blacksquare$

Policy Iteration

Policy iteration is when we apply policy evaluation and improvement iteratively.

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \cdots$$

Note the each E is itself an iterative process.

For a finite MDP, policy iteration is guaranteed to converge to π^* in a finite (and often small) number of steps.

Here is an example from Sutton and Barto.

Example 4.2: Jack's Car Rental Jack manages two locations for a nationwide car rental company. Each day, some number of customers arrive at each location to rent cars. If Jack has a car available, he rents it out and is credited \$10 by the national company. If he is out of cars at that location, then the business is lost. Cars become available for renting the day after they are returned. To help ensure that cars are available where they are needed, Jack can move them between the two locations overnight, at a cost of \$2 per car moved. We assume that the number of cars requested and returned at each location are Poisson random variables, meaning that the probability that the number is n is $\frac{\lambda^n}{n!}e^{-\lambda}$, where λ is the expected number. Suppose λ is 3 and 4 for rental requests at the first and second locations and 3 and 2 for returns. To simplify the problem slightly, we assume that there can be no more than 20 cars at each location (any additional cars are returned to the nationwide company, and thus disappear from the problem) and a maximum of five cars can be moved from one location to the other in one night. We take the discount rate to be $\gamma = 0.9$ and formulate this as a continuing finite MDP, where the time steps are days, the state is the number of cars at each location at the end of the day, and the actions are the net numbers of cars moved between the two locations overnight. Figure 4.4 shows the sequence of policies found by policy iteration starting from the policy that never moves any cars. ■

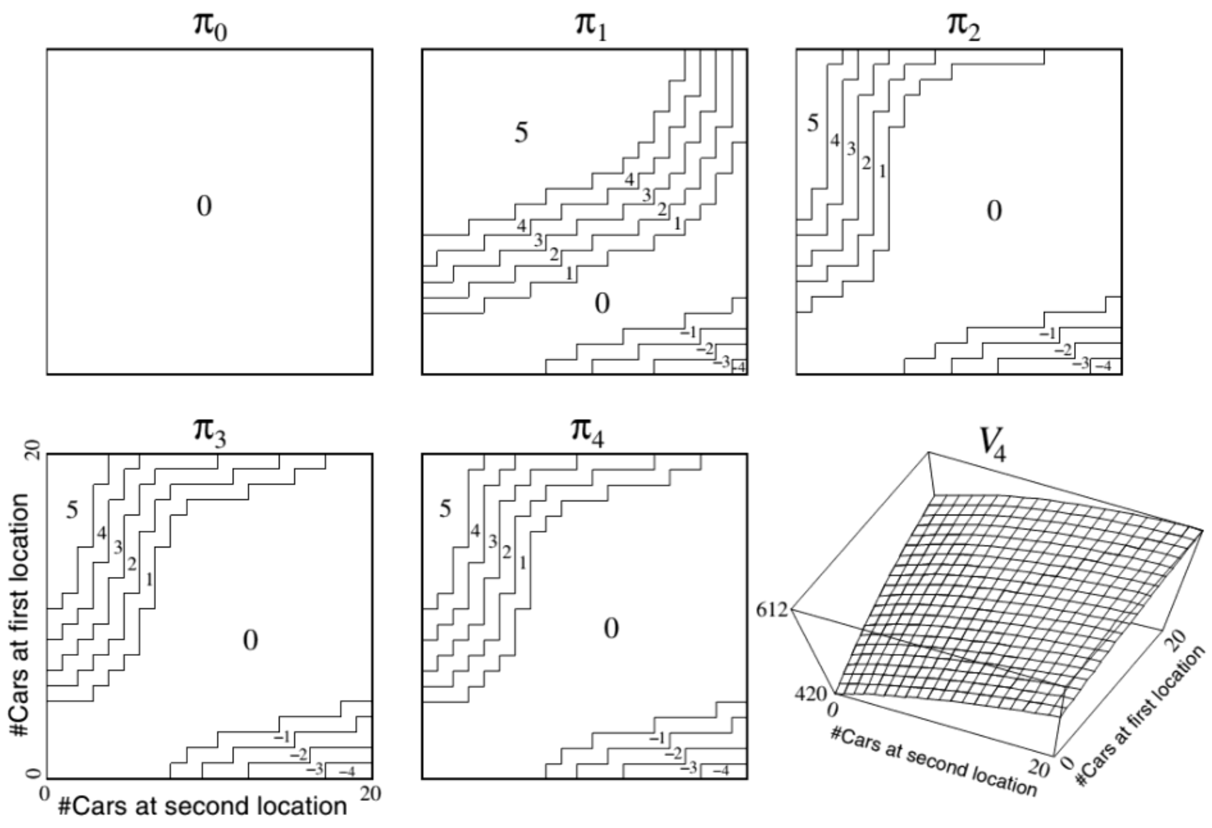


Figure 4.4 The sequence of policies found by policy iteration on Jack's car rental problem, and the final state-value function. The first five diagrams show, for each number of cars at each location at the end of the day, the number of cars to be moved from the first location to the second (negative numbers indicate transfers from the second location to the first). Each successive policy is a strict improvement over the previous policy, and the last policy is optimal.

Value Iteration

Value iteration is like policy iteration, except only one iteration of policy evaluation is applied at each E step.

Thus, value iteration produces a sequence $V_1, V_2, \ldots$ satisfying

$$V_{k+1}(s) = \max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s') \right].$$

To see this, let $Q_k(s,a)$ denote the state-action value function associated to V_k, i.e.

$$Q_k(s,a) = \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s') \right].$$

Now policy π_{k+1} is defined by

$$\pi_{k+1}(s) = \arg\max_a Q(s,a) \qquad \forall s$$

Combining this with one step of policy evaluation, we can can now see the claimed update.

In essence, value iteration uses the Bellman optimality equation as an update rule. Like policy iteration, value iteration converges to the optimal value function and an optimal policy.

More generally, converges holds if we apply any finite number of iterations of policy evaluation at each step. Different choices will lead to different rates of convergence.

Another example from Sutton and Barto:

Example 4.3: Gambler's Problem A gambler has the opportunity to make bets on the outcomes of a sequence of coin flips. If the coin comes up heads, he wins as many dollars as he has staked on that flip; if it is tails, he loses his stake. The game ends when the gambler wins by reaching his goal of \$100, or loses by running out of money. On each flip, the gambler must decide what portion of his capital to stake, in integer numbers of dollars. This problem can be formulated as an undiscounted, episodic, finite MDP. The state is the gambler's capital, $s \in \{1, 2, \ldots, 99\}$ and the actions are stakes, $a \in \{1, 2, \ldots, \min(s, 100 - s)\}$. The reward is zero on all transitions except those on which the gambler reaches his goal, when it is $+1$. A state-value function then gives the probability of winning from each state. A policy is a mapping from levels of capital to stakes. The optimal policy maximizes the probability of reaching the goal. Let p denote the probability of the coin coming up heads. If p is known, then the entire problem is known and it can be solved, for instance, by value iteration. Figure 4.6 shows the change in the value function over successive sweeps of value iteration, and the final policy found, for the case of $p = 0.4$. ■

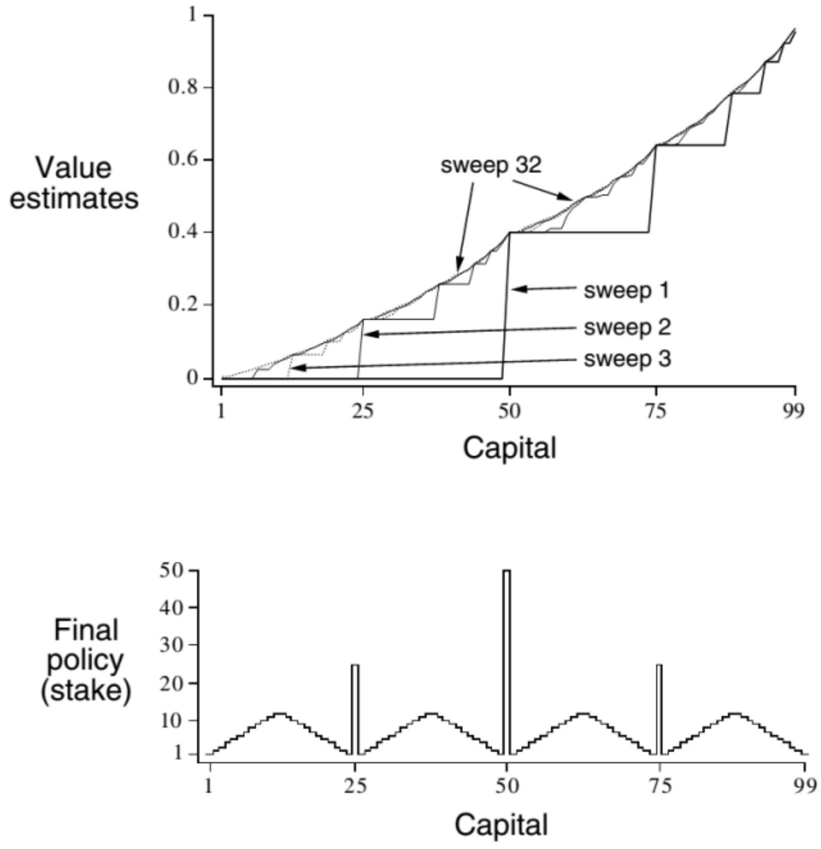


Figure 4.6 The solution to the gambler's problem for $p = 0.4$. The upper graph shows the value functions found by successive sweeps of value iteration. The lower graph shows the final policy.

$\boxed{\text{Asynchronous Updates}}$

So far we have assumed that every iteration of policy evaluation or policy improvement conducts a full "sweep" through the state space. If the state space is too large, however, this may be infeasible.

In asynchronous dynamic programming, some states are updated more often than others. Convergence theorem can still be established as long as each state is updated infinitely often. With asynchronous updates, we can still improve our policy without making a full sweep by focus more on the "important states".