Dynamic Programming

Week #4

Introduction

- *Dynamic Programming (DP)*
  - refers to a collection of algorithms
  - has a high computational complexity
  - assumes a perfect model of environment
  - hence is of limited utility in RL
  - is crucial because of its theoretical foundation.
- *Key idea in DP*: using the value functions to organize and structure the search for good policies.

Set up

- We assume the environment is a finite MDP.
- Finite set of states: *S*
- Finite set of actions: *A(s)*
- Dynamics of environment given by:
  - a set of transition probabilities, and

$$\boxed{P^a_{s_i s_j} = P(s_{t+1} = s_j \mid s_t = s_i, a_t = a)}$$

  - the expected immediate reward

$$\boxed{R^a_{s_i s_j} = E(r_{t+1} \mid s_t = s_i, a_t = a, s_{t+1} = s_j)}$$

- for all $s_i \in S$, $s_j \in S^+$ and a $\in A(s)$.

Policy Evaluation

- *Policy evaluation* (also called *prediction problem*) is where the state-value function V^π is computed for an arbitrary policy π.

$$
\begin{aligned}
V^{\pi}(s^i) &= E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s^i \right\} = \\
&= E_{\pi}\left\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_t = s^i \right\} = \qquad (5.1) \\
&= E_{\pi}\left\{ r_{t+1} + \gamma V^{\pi}(s_{t+1}) \middle| s_t = s^i \right\} \\
&= \sum_{a} \pi(s^i, a) \sum_{j} P^a_{s^i s^j} \left[R^a_{s^i s^j} + \gamma V^{\pi}(s_j) \right]
\end{aligned}
$$

- *π(s,a)* is the probability of taking action *a* at state *s*.

Policy Evaluation

- If *environment dynamics* (*transition probabilities and immediate returns for all states*) are *completely known*, then last line of (5.1) is a system of |S| simultaneous linear equations with |S| unknowns.
- Iterative methods are suitably used for computation of $V^{\pi}(s)$.

$$V_{k+1}(s^i) = E_{\pi}\left\{ r_{t+1} + \gamma V_k(s_{t+1}) \middle| s_t = s^i \right\}$$

$$= \sum_a \pi(s^i, a) \sum_j P^a_{s^i s^j}\left[R^a_{s^i s^j} + \gamma V_k(s_j) \right]$$

- As $k \to \infty$, V_k converges to V^{π}. This algorithm is called *iterative policy evaluation*.

Policy Evaluation

- Two ways to compute V_k values
  - *Full backup*: At time *k+1*, values of *all* states are updated using the value of all states at time *k*.
  - "*In place*" computation. At any time *k*, *one* state is updated. That is, to update the value of a state *s*, we use the current value(s) of relevant state(s) (i.e., states which can be arrived at from *s* or destination states of *s*).

Algorithm for Policy Evaluation

- *Input π, the policy to be evaluated*
- *Initialize, $V(s_i)=0$, for all $s_i \in S^+$*
- *Loop*
  - *$\Delta \leftarrow 0$*
  - *for each $s_i \in S$:*
    - *$v \leftarrow V(s_i)$*

$$V(s^i) = \sum_a \pi(s^i, a) \sum_j P^a_{s^i s^j} \left[R^a_{s^i s^j} + \gamma V_k(s_j) \right]$$

    - *$\Delta \leftarrow \max\{\Delta, |v - V(s_i)|\}$*
- *Until $\Delta < \theta$*
- *Output $V \approx V^\pi$*

Policy Improvement… (1)

- *Assumption*: we have determined V^{π} for an arbitrary deterministic policy π.
- *Question*: for some state s, should we change the policy to deterministically choose an action $a \neq \pi(s)$ so as to end up with a better value of $V(s)$?

$$Q^{\pi}(s^i, a) = E_{\pi}\left\{ r_{t+1} + \gamma V^{\pi}(s_{t+1}) \middle| s_t = s^i, a_t = a \right\} =$$

$$= \sum_j P^a_{s^i s^j} \left[R^a_{s^i s^j} + \gamma V^{\pi}(s_j) \right]$$

Policy Improvement… (2)

- Here, we check to see whether choosing action $a = \pi'(s) \neq \pi(s)$ *once* at state *s* and continue thereafter with the existing policy is better (i.e., results in a higher value $V^{\pi}(s)$ of state *s*).
- If so, then *choosing action* ***a*** *everytime at state* **s** *may mean a better policy overall*.

Policy Improvement... (3)

- Assertion: Let π and π' be any two deterministic policies such that for all $s \in S$,

$$Q^{\pi}\big(s, \pi'(s)\big) \geq V^{\pi}(s)$$

- Then π' must be as good as, or better than π. That is:

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

- Further, for any state s, a strict inequality in the red equation results in a strict inequlity in the blue equation below.

Proof of Policy Improvement Theorem

$$
\begin{aligned}
V^{\pi}(s) &\le Q^{\pi}\left(s,\pi'(s)\right) \\
&= E_{\pi'}\left\{r_{t+1}+\gamma V^{\pi}(s_{t+1}) \middle| s_t = s\right\} \\
&\le E_{\pi'}\left\{r_{t+1}+\gamma Q^{\pi}(s_{t+1},\pi'(s_{t+1})) \middle| s_t = s\right\} \\
&= E_{\pi'}\left\{r_{t+1}+\gamma E_{\pi'}\left\{r_{t+2}+\gamma V^{\pi}(s_{t+2})\right\} \middle| s_t = s\right\} \\
&= E_{\pi'}\left\{r_{t+1}+\gamma r_{t+2}+\gamma^2 V^{\pi}(s_{t+2}) \middle| s_t = s\right\} \\
&\le E_{\pi'}\left\{r_{t+1}+\gamma r_{t+2}+\gamma^2 r_{t+3}+\gamma^3 V^{\pi}(s_{t+3}) \middle| s_t = s\right\} \\
&\vdots \\
&\le E_{\pi'}\left\{r_{t+1}+\gamma r_{t+2}+\gamma^2 r_{t+3}+\gamma^3 r_{t+4}+\cdots) \middle| s_t = s\right\} \\
&= V^{\pi'}(s)
\end{aligned}
$$

This proof is received from reference [1].

Policy Improvement… (5)

- The next step here is to find the action *a* at each state *s* that appears best regarding $Q^{\pi}(s,a)$. The new greedy policy *π'* is:
- $$\begin{aligned}\pi'(s^i) &= \arg\max_a Q^{\pi}(s^i,a) \\ &= \arg\max_a E\left[r_{t+1} + \gamma V^{\pi}(s_{t+1}) \middle| s_t = s^i, a_t = a\right] \\ &= \arg\max_a \sum_j P^a_{s^i s^j}\left[R^a_{s^i s^j} + \gamma V^{\pi}(s^j)\right]\end{aligned}$$
- The process of making a new policy that improves on an original policy, by making it greedy or nearly greedy with respect to the value function of the original policy, is called *policy improvement*.

Policy Iteration… (1)

- *Idea*: Improve policy π using V^{π} to a better policy π_1 and compute (evaluate) V^{π_1}. Next, improve V^{π_1} to yield an even better policy π_2 and evaluate V^{π_2}. This two step iteration goes on with *strict improvements* until an optimal policy is encountered. For a finite MDP has only a finite number of policies, the process must *converge to an optimal policy and optimal value function in a finite number of iterations*.

Policy Iteration: Diagram

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

In the above diagram denoting the policy iteration the transition *E* and *I* stand for the policy evaluation and policy improvements phases of the poliy iteration steps in respective order [1].

The figure is an exact replication
of the figure on page 97 of [1].

Policy Iteration: Algorithm

- *Initialization*
- *$V(s) \in$ R, $\pi(s) \in$A(s) arbitrarily for all s$\in$ S*
- *Policy Evaluation*
- *Repeat*
  - *$\Delta \leftarrow 0$*
  - *For each s $\in$ S:*
    - *$v \leftarrow V(s_i)$*
    - $V(s) = \sum_j P_{ss^j}^{\pi(s)} \left[R_{ss^j}^{\pi(s)} + \gamma V(s^j) \right]$ ***// π(s) arbitrarily selected***
    - *$\Delta \leftarrow \max\{\Delta, |v - V(s)|\}$*
- *Until $\Delta < \theta$ (a small positive number)*
- *Policy Improvement*
- *Policy-stable $\leftarrow$true*
- *For each s $\in$ S:*
  - *$b \leftarrow \pi(s)$*

$$\pi(s) \leftarrow \arg\max_a \sum_j P_{s^i s^j}^a \left\{ R_{s^i s^j}^a + \gamma V^\pi(s^j) \right\}$$

  - *If $b \neq \pi(s)$, then policy-stable$\leftarrow$false*
- *If policy-stable, then stop; else go to policy evaluation*

This is the algorithm given in
Figure 4.3 on page 98 of [1].

Value Iteration... (1)

- *<u>Idea</u>*: Policy evaluation step itself in policy iteration may require iterative computations where *multiple sweeps through the state set* are performed. But we observe in many examples the *possibility of cutting short the policy evaluation step without losing the convergence guarantee of policy iteration*. A special case in this effort is *stopping policy evaluation after a single sweep* (one backup of each state). This algorithm is called *value iteration*.

$$V_{k+1}(s^i) = \max_a E\left\{ r_{t+1} + \gamma V_k(s_{t+1}) \middle| s_t = s^i, a_t = a \right\}$$

$$= \max_a \sum_j P^a_{s^i s^j} \left[R^a_{s^i s^j} + \gamma V_k(s^j) \right]$$

Value Iteration: Algorithm

- *Initialize V arbitrarily; e.g.,* *$V(s^i)=0$ for all $s^i \in S$*
- *Repeat*
  - *$\Delta \leftarrow 0$*
  - *For each $s^i \in S$:*
    - *$v \leftarrow V(s^i)$*

$$V(s^i) = \max_a \sum_j P^a_{s^i s^j} \left\{ R^a_{s^i s^j} + \gamma V(s^j) \right\}.$$

    - *$\Delta \leftarrow \max\{\Delta, |v - V(s^i)|\}$*
- *Until $\Delta < \theta$ (a small positive number)*

Output a deterministic policy, π, such that

$$\pi(s) \leftarrow \arg\max_a \sum_j P^a_{s^i s^j} \left[R^a_{s^i s^j} + \gamma V(s^j) \right]$$

This is the algorithm given in Figure 4.5 on page 102 of [1].

Asynchronous Dynamic Programming (ADP)

- *Idea*: ADP algorithms are *in-place iterative algorithms* that *update* (back up) *values of states in an unorganized manner*. This results in a non-uniform session of update of state values. At a certain point of time, some states may get their values updated several times while some others may have their values not updated even once. Once eliminating this property to correctly converge to an optimal policy, ADP algorithms provide for great flexibility for the selection of states to which backup operations are applied. This may be eliminated, for instance, by backing up the value of only one state s_k, on each step k, using the value iteration backup. In short:

Update only state s_k at time k !...

Generalized Policy Iteration (GPI)... (1)

- *Idea*: We studied policy iteration to involve *two interacting processes*: *policy evaluation* finding the value of states using the current policy and *policy improvement* looking in a *greedy* manner for a new policy wrt the current value function.
- These processes that we have seen to follow each other in a closed loop are *not required to operate in succession*. In fact, we have shown in *value iteration* that, between any two policy improvement session, a *policy evaluation session of a single sweep* has been sufficient. Further, *ADP algorithms* did provide a *higher degree of interleaving by updating the value of a single state at a policy evaluation session*.
- The general idea in GPI is the interaction of policy evaluation and improvement independent of how fine-grained the policy evaluation occurs and of any other details of these two processes.

Generalized Policy Iteration... (2)

$$\begin{tikzcd}
\boxed{\pi} \arrow[rrr, bend left, "{\substack{\textit{evaluation} \\ V \rightarrow V^{\pi}}}"] & & & \boxed{V} \arrow[lll, bend left, "{\substack{\pi \rightarrow \textit{greedy}(V) \\ \textit{improvement}}}"]
\end{tikzcd}$$

$$\vdots$$

$$\pi^* \leftrightarrow V^*$$

where π^* and V^* denote optimal policy and state values, respectively.

References

- [1] Sutton, R. S. and Barto A. G., "*Reinforcement Learning: An introduction*," MIT Press, 1998