Lecture 6: Algorithms for solving the Bellman equation

Course: Reinforcement Learning Theory
Instructor: Lei Ying
Department of EECS
University of Michigan, Ann Arbor

Algorithms for solving the Bellman equation

Overview of algorithms:

- Value iteration
- Policy iteration
- Linear programming

Value iteration

Recall that T is a contraction mapping. Starting from $\hat{J}_0$, we can iteratively compute

$$\hat{J}_{k+1} = T(\hat{J}_k)$$

Value iteration

Recall that T is a contraction mapping. Starting from $\hat{J}_0$, we can iteratively compute

$$\hat{J}_{k+1} = T(\hat{J}_k)$$

Claim: Under a contraction mapping,

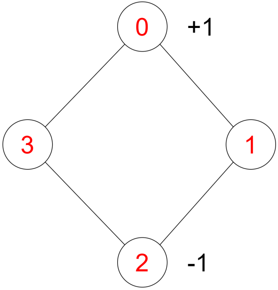
$$\hat{J}_k \to J^* \text{ s.t. } J^* = T(J^*)$$

After finding J^*, we can find the optimal policy by solving

$$\mu^*(i) \in \arg\max_u E[r(i,u)] + \alpha \sum_j P_{ij}(u) J^*(j)$$

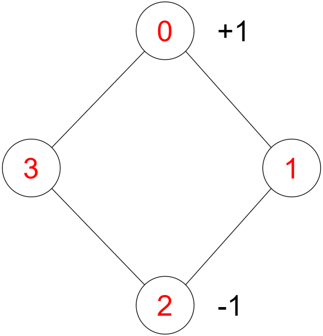
Value iteration

Example:



- Actions: clockwise (c) or counter-clockwise (cc). An action is correctly executed with probability 0.6 and moves to the opposite direction with probability 0.4.
- Discount factor $\alpha = 0.9$.

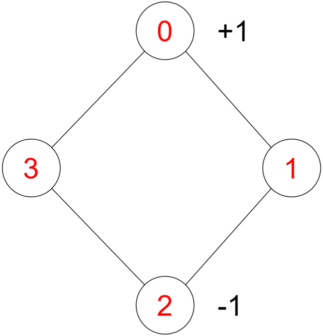
Value iteration



$$\hat{J}_0 = \begin{pmatrix} +1 \\ 0 \\ -1 \\ 0 \end{pmatrix}$$

$$\begin{aligned} \hat{J}_1(0) &= \max\{0.6(0+\alpha\hat{J}_0(1)) + 0.4(0+\alpha\hat{J}_0(3)), \\ &\qquad 0.4(0+\alpha\hat{J}_0(1)) + 0.6(0+\alpha\hat{J}_0(3))\} \\ &= \max\{0,0\} = 0 \end{aligned}$$

$$\begin{aligned} \hat{J}_1(1) &= \max\{0.6(-1+0.9(-1)) + 0.4(1+0.9(1)), \\ &\qquad 0.6(1+0.9(1)) + 0.4(-1+0.9(-1))\} \\ &= \max\{-0.38, 0.38\} = 0.38 \end{aligned}$$



Similarly,

$$\hat{J}_1 = \begin{pmatrix} 0 \\ 0.38 \\ 0 \\ 0.38 \end{pmatrix}$$

$$\hat{J}_2 = \begin{pmatrix} 0.342 \\ 0.2 \\ 0.342 \\ 0.2 \end{pmatrix}$$

... And so on.

Policy iteration

1. Choose an initial policy μ, set $k = 0$
2. Compute the value function of the policy J_{μ_k} by solving

$$J_{\mu_k} = T_{\mu_k}(J_{\mu_k})$$

3. Compute a new policy μ_{k+1} as

$$\mu_{k+1}(i) \leftarrow \arg\max_u E[r(i,u)] + \alpha \sum_j P_{ij}(u) J_{\mu_k}(j)$$

$$k \leftarrow k + 1$$

4. Repeat if $J_{\mu_{k+1}} \neq J_{\mu_k}$

Policy iteration

Claim: $J_{\mu_{k+1}} \geq J_{\mu_k}$, i.e. the value function improves at each step before the algorithm terminates.

Proof:

$$T_{\mu_{k+1}}(J_{\mu_k})(i) = E[r(i, \mu_{k+1}(i))] + \alpha \sum_j P_{ij}(\mu_{k+1}(i)) J_{\mu_k}(j)$$

From the definition of μ_{k+1},

$$\begin{aligned} T_{\mu_{k+1}}(J_{\mu_k})(i) &\geq E[r(i, \mu_k(i))] + \alpha \sum_j P_{ij}(\mu_k(i)) J_{\mu_k}(j) \\ &= T_{\mu_k}(J_{\mu_k})(i) \\ &= J_{\mu_k}(i) \end{aligned}$$

Policy iteration

From monotonicity,

$$
\begin{aligned}
T^n_{\mu_{k+1}}(J_{\mu_k}) &\geq J_{\mu_k} \\
J_{\mu_{k+1}} = \lim_{n\to\infty} T^n_{\mu_{k+1}}(J_{\mu_k}) &\geq J_{\mu_k}
\end{aligned}
$$

■

Policy iteration

Claim: If $J_{\mu_{k+1}} = J_{\mu_k}$, then μ_{k+1} is optimal.

Proof: By definition of policy iteration, we have

$$T_{\mu_{k+1}}(J_{\mu_k}) = T(J_{\mu_k})$$

Thus,

$$T_{\mu_{k+1}}(J_{\mu_{k+1}}) \underset{\text{Step 2}}{=} J_{\mu_{k+1}} \underset{J_{\mu_k} = J_{\mu_{k+1}}}{=} T(J_{\mu_{k+1}})$$

Since $J_{\mu_{k+1}} = T(J_{\mu_{k+1}})$, μ_{k+1} is optimal. ■

Value iteration vs. policy iteration

Policy iteration converges in a finite number of iterations, but each iteration can be costly since we need to solve $J_{\mu_k} = T(J_{\mu_k})$.

Linear programming

Recall that J^* satisfies

$$J^*(i) = \max_u \bar{r}(i,u) + \alpha \sum_j P_{ij}(u) J^*(j)$$

$$\Longleftrightarrow$$

$$J^*(i) \geq \bar{r}(i,u) + \alpha \sum_j P_{ij}(u) J^*(j) \; \forall u$$

$$\text{and } J^*(i) = \bar{r}(i,u) + \alpha \sum_j P_{ij}(u) J^*(j) \text{ for some } u$$

Linear programming

We consider the following LP:

$$\min_J \sum_i J(i)$$

$$J(i) \geq \bar{r}(i,u) + \alpha \sum_j P_{ij}(u) J(j), \quad \forall u$$

Claim: The optimal solution to the LP is J^*.

Linear programming

Proof:

- Note that any feasible J (vector) satisfies $J \geq T(J)$.
- By the monotonicity of mapping T, we have

$$T(J) \geq T^2(J) \implies J \geq T^2(J)$$

Continuing this argument, we have

$$J \geq T^n(J) \to J^* \text{ as } n \to \infty$$

So we have $J \geq J^* \; \forall$ feasible J, while J^* is a feasible solution by its definition.

Challenges of Applying These Algorithms

- Model-based algorithms: need to know $\bar{r}(i, u)$ and $P_{ij}(u)$
- Scalability: only work for finite state space and finite action space

Reference

- This lecture is based on R. Srikant's lecture notes on *Value Iteration/Policy Iteration/LP Solution* available at `https://sites.google.com/illinois.edu/mdps-and-rl/lectures?authuser=1`

Acknowledgements: I would like to thank Alex Zhao for helping prepare the slides, and Honghao Wei and Zixian Yang for correcting typos/mistakes.