Introduction to Reinforcement Learning: Part III

Dr. Mayank S JHA

Maitre de Conférences (Associate Professor),
CRAN,
UMR 7039, CNRS,
Polytech Nancy,
Office: C 225,
Université de Lorraine
France.

Table of Contents

Table of Contents

Policy Iteration

Consider a current policy $\pi(x, u)$,

- **Policy Evaluation:** Its *value* can be determined by solving the Bellman equation.
- **Policy Improvement:** Given the value for some policy $\pi(x, u)$, find another policy that is better, or at least no worse.

Policy Iteration

Consider a current policy $\pi(x, u)$.

- **Policy Evaluation:** Its *value* can be determined by solving the Bellman equation.

$$V^{\pi}(x) = \sum_{u} \pi(x, u) \sum_{x'} P^{u}_{xx'} \left[R^{u}_{xx'} + \gamma V^{\pi}(x') \right]$$

for all $x \in S \subseteq X$.

where S is a suitably selected subspace of the state space (to discuss later).

- **Policy Improvement:** Given the value for some policy $\pi(x, u)$, find another policy that is better, or at least no worse.

$$\pi'(x, u) = \arg\min_{\pi} \sum_{x'} P^{u}_{xx'} \left[R^{u}_{xx'} + \gamma V^{\pi}(x') \right]$$

for all $x \in S \subseteq X$

Policy Iteration

- It can be shown that $V^{\pi'}(x) \leq V^{\pi}(x)$.
- The policy determined as in (22) is said to be greedy with respect to value function $V^{\pi}(x)$.

If, $V^{\pi'}(x) = V^{\pi}(x)$, then $V^{\pi'}(x), \pi'(x,u)$ satisfy the Bellman Equations. Therefore $\pi'(x,u) = \pi(x,u)$ is the optimal policy and $V^{\pi'}(x) = V^{\pi}(x)$ the optimal value.

That is, an optimal policy, and only an optimal policy, is greedy with respect to its own value.

some properties

- At each step of such algorithms, a policy is obtained that is no worse than the previous policy.
- Proof of convergence under fairly mild conditions to the optimal value and optimal policy.
    - proofs are based on the Banach fixed point theorem.
- Bellman Optimality Eq is fixed point equation for $V^*(\cdot)$.
- Policy Evaluation and Improvement define a contraction Map.

Table of Contents

PI Algorithm

Select an initial policy $\pi_0(x, u)$.
Starting with $j = 0$, iterate on j until convergence:

PI

Policy Evaluation (PE)

$$V_j(x) = \sum_u \pi_j(x, u) \sum_{x'} P^u_{xx'} \left[R^u_{xx'} + \gamma V_j\left(x'\right) \right],$$

for all $x \in X$.

Policy improvement:

$$\pi_{j+1}(x, u) = \arg\min_{\pi} \sum_{x'} P^u_{xx'} \left[R^u_{xx'} + \gamma V_j\left(x'\right) \right]$$

for all $x \in X$

Policy Evaluation (PE)

At each step j, the policy evaluation algorithm determines the solution of the Bellman equation to compute the value $V_j(x)$ of using the current policy $\pi_j(x, u)$.
This value corresponds to the infinite sum for the current policy.
For reminder:

$$V^{\pi}(x) = E_{\pi}\left\{J_k \mid x_k = x\right\} = E_{\pi}\left\{\sum_{i=k}^{\infty} \gamma^{i-k} r_i \mid x_k = x\right\}$$

Then the policy is improved.
The steps are continued until there is no change in the value or the policy.

Policy Evaluation (PE)

- Note that j is not the time or stage index k but a policy iteration step iteration index.
- The policy iteration algorithm must be suitably initialized to converge. The initial policy $\pi_0(x, u)$ is *stabilising* .

Note: Policy iteration can be implemented for dynamical systems online in real time by observing data measured along the system trajectories. Data for multiple times k are needed to solve the Bellman equation (25) at each step j.

Policy Evaluation solution as Iterative procedure

- For finite MDP with N states, the policy evaluation equation is a system of N simultaneous linear equations, one for each state.
- Instead of directly solving the Bellman equation (PE), it can be solved by an iterative policy evaluation procedure.
- Note that (PE) is a fixed point equation for $V_j(\cdot)$ that defines the iterative policy evaluation map, (contraction map).

$$V_j^{i+1}(x) = \sum_u \pi_j(x,u) \sum_{x'} P^u_{xx'} \left[R^u_{xx'} + \gamma V_j^i\left(x'\right) \right], \quad i = 1, 2, \ldots,$$

Policy Evaluation solution as Iterative procedure

- Note that (PE) is a fixed point equation for $V_j(\cdot)$ that defines the iterative policy evaluation map, (contraction map).

$$V_j^{i+1}(x) = \sum_u \pi_j(x,u) \sum_{x'} P_{xx'}^u \left[R_{xx'}^u + \gamma V_j^i(x') \right], \quad i = 1, 2, \ldots,$$

- The iteration can be initialized at any non-negative value of $V_j^1(\cdot)$ and the iteration converges to the solution of PE $\rightarrow$ this solution is unique.
- A suitable initial value choice is the value function $V_{j-1}(\cdot)$ from the previous step $j-1$. On close enough convergence, set $V_j(\cdot) = V_j^i(\cdot)$ and proceed to apply PE.

Policy Evaluation solution as Iterative procedure

- The index $j \to$ step number of the policy iteration algorithm.
- The index $i \to$ is an iteration index to solve Policy Evaluation step (PE).

Dr. Mayank S JHA, mayank-shekhar.jha@univ-lorraine.fr Polytech Nancy, CRAN, University of Lorraine, France

Table of Contents

Policy Iteration for Linear DT

MDP is deterministic and satisfies the state transition equation

$$x_{k+1} = Ax_k + Bu_k,$$

with the discrete time index k. The associated infinite-horizon performance index has deterministic stage costs and is

$$J_k = \frac{1}{2}\sum_{i=k}^{\infty} r_i = \frac{1}{2}\sum_{i=k}^{\infty}\left(x_i^T Q x_i + u_i^T R u_i\right)$$

Here: state space $X = R^n$ and action space $U = R^m$ are infinite and continuous.

Select a policy $u_k = \mu(x_k)$ and write the associated value function as

$$V(x_k) = \frac{1}{2}\sum_{i=k}^{\infty} r_i = \frac{1}{2}\sum_{i=k}^{\infty}\left(x_i^T Q x_i + u_i^T R u_i\right)$$

An equivalent difference equation is

$$\begin{aligned} V(x_k) &= \frac{1}{2}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}\sum_{i=k+1}^{\infty}\left(x_i^T Q x_i + u_i^T R u_i\right) \\ &= \frac{1}{2}\left(x_k^T Q x_k + u_k^T R u_k\right) + V(x_{k+1}). \end{aligned}$$

- The solution $V(x_k)$ to this equation that satisfies $V(0) = 0$, is the value given above.
- **This is exactly the Bellman equation for the LQR.**

Policy Evaluation

Iterative policy evaluation (PE) .....

$$V_j(x) = \sum_u \pi_j(x,u) \sum_{x'} P^u_{xx'} \left[R^u_{xx'} + \gamma V_j\left(x'\right)\right],$$

for all $x \in X$.

.....applied on "Bellman Equation for the Discrete-Time LQR, the Lyapunov Equation
yields:

$$V^{j+1}\left(x_k\right) = \frac{1}{2}\left(x_k^T Q x_k + u_k^T R u_k\right) + V^{j+1}\left(x_{k+1}\right)$$

Assum value is quadratic in the state for some for some kernel matrix P, $V^j(x_k) = \frac{1}{2}x_k^T P^j x_k$
yields the Bellman equation form

$$x_k^T P^{j+1} x_k = x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P^{j+1} x_{k+1},$$

Assuming a constant, that is, stationary, state feedback policy $u_k = \mu(x_k) = -K^j x_k$ for some stabilizing gain K^j leads to:

$$x_k^T P^{j+1} x_k = x_k^T Q x_k + x_k^T {K^j}^T R K^j x_k + x_k^T (A - BK^j)^T P^{j+1} (A - BK^j) x_k.$$

Policy Evaluation

Since this equation holds for all state trajectories, we have **the Lyapunov Equation** as:

$$0 = \left(A - BK^j\right)^T P^{j+1}\left(A - BK^j\right) - P^{j+1} + Q + \left(K^j\right)^T RK^j$$

To solve this Lyapunov Equation, given a fixed policy K^j, the iterative equation is:

$$P^{i+1} = (A - BK^j)^T P^i (A - BK^j) + Q + {K^j}^T RK^j.$$

This recursion converges to the solution of the Lyapunov equation i.e. as $i \to \infty$, $P^i \to P^{j+1}$, with $P^{j+1} = \left(A - BK^j\right)^T P^{j+1}\left(A - BK^j\right) + Q + \left(K^j\right)^T RK^j$ if $(A - BK)$ is stable, for any choice of initial value $P^{i=0}$

Policy Improvement

The policy improvement step is

$$\mu^{j+1}(x_k) = K^{j+1} x_k$$
$$= \arg\min \left(x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P^{j+1} x_{k+1} \right)$$

which can be written explicitly as

$$K^{j+1} = -\left(B^T P^{j+1} B + R \right)^{-1} B^T P^{j+1} A.$$

Dr. Mayank S JHA, mayank-shekhar.jha@univ-lorraine.fr Polytech Nancy, CRAN, University of Lorraine, France

Observations

- The policy iteration algorithm relies on repeated solutions of Lyapunov equations at each step.
- called Hewer's algorithm $\rightarrow$ proven to converge to the solution of the Riccati equation in "The Bellman Optimality Equation for Discrete-Time LQR Is an Algebraic Riccati Equation."
- this is offline algorithm
- requires complete knowledge of the system dynamics (A, B) to find the optimal value and control.
- the algorithm requires that the initial gain K^0 be stabilizing.

Dr. Mayank S JHA, mayank-shekhar.jha@univ-lorraine.fr Polytech Nancy, CRAN, University of Lorraine, France

Algorithm

References I