ORF 505: Statistical Analysis of Financial Data

Tuesday Precept 10: Dynamic Programming

Nov 19, 2024

Lecturer: Qishuo Yin *Scribe: Qishuo Yin*

1 Policy Evaluation

By the updating rule in Bellman equation for v_π, the successive approximation of the state-value function can be computed as:

$$\begin{aligned} v_{k+1}(s) &= \mathbb{E}_\pi \left[R_{t+1} + \gamma v_k(S_{t+1})\right] \\ &= \sum_{a \in A(s)} \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_k(s')\right] \end{aligned}$$

Iterative policy evaluation: the sequence v_k can be shown in general to converge to v_π as $k \to \pi$ under the same conditions that guarantee the existence of v_π.

Algorithm 1 Policy Evaluation

Input: π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in S^+$
repeat
  $\Delta \leftarrow 0$
  for each $s \in S$ **do**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end for
until $\Delta < \theta$ (a small positive number)
Output: $V \approx v_\pi$

2 Policy Improvement

Theorem 2.1. *Policy improvement theorem.*
Let π and π' be any pair of deterministic policies such that, for all $s \in \mathcal{S}$,

$$q_\pi(s, \pi'(s)) \geq v_\pi(s)$$

Then the policy π' must be as good as, or better than, π. That is, it must obtain greater or equal expected return from all states $s \in \mathcal{S}$:

$$v'_\pi(s) \geq v_\pi(s)$$

Consider a new greedy policy, π':

$$\begin{aligned}
\pi'(s) &= \arg\max_a q_\pi(s, a) \\
&= \arg\max_a \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a\right] \\
&= \arg\max_a \sum_{s', r} p(s', r|s, a)\left[r + \gamma v_\pi(s')\right]
\end{aligned}$$

The process of making a new policy that improves on an original policy, by making it greedy with respect to the value function of the original policy, is called policy improvement.

Given v_π, the policy improvement theorem suggests that a better policy can be found by acting greedily with respect to v_π:

$$\pi'(s) = \arg\max_a \sum_{s', r} p(s', r|s, a)\left[r + \gamma v_\pi(s')\right] \tag{1}$$

3 Policy Iteration

Policy iteration alternates between policy evaluation and policy improvement until convergence:

1. **Policy Evaluation:** Compute v_π for the current policy π.
2. **Policy Improvement:** Update the policy based on the new value function.

The complete policy iteration algorithm can be given as:

4 Value Iteration

One drawback to policy iteration is that each of its iterations involves policy evaluation, which may itself be a protracted iterative computation requiring multiple sweeps through the state set. In fact, the policy evaluation step of policy iteration can be truncated in several ways without losing the convergence guarantees of policy iteration. This algorithm is called value iteration.

Value iteration simplifies policy iteration by combining policy evaluation and improvement in a single step:

$$v_{k+1}(s) = \max_a \sum_{s', r} p(s', r|s, a)\left[r + \gamma v_k(s')\right] \tag{2}$$

The value iteration algorithm can be given as:

More information can be found in section 4 of [1]

Algorithm 2 Policy Iteration

1. Initialization:
Initialize $V(s) \in \mathbb{R}$ and $\pi(s) \in A(s)$ arbitrarily for all $s \in S$
2. Policy Evaluation:
repeat
  $\Delta \leftarrow 0$
  for each $s \in S$ **do**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))[r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end for
until $\Delta < \theta$ (a small positive number)
3. Policy Improvement:
$policy\_stable \leftarrow$ true
for each $s \in S$ **do**
  $a \leftarrow \pi(s)$
  $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
  if $a \neq \pi(s)$ **then**
    $policy\_stable \leftarrow$ false
  end if
end for
if $policy\_stable$ **then**
  return V and π
else
  go to 2
end if

Algorithm 3 Value Iteration

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in S^+$)
repeat
  $\Delta \leftarrow 0$
  for each $s \in S$ **do**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end for
until $\Delta < \theta$ (a small positive number)
Output: a deterministic policy, π, such that

$$\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$$

References

[1] R. S. Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.