FPT Education
FPT UNIVERSITY

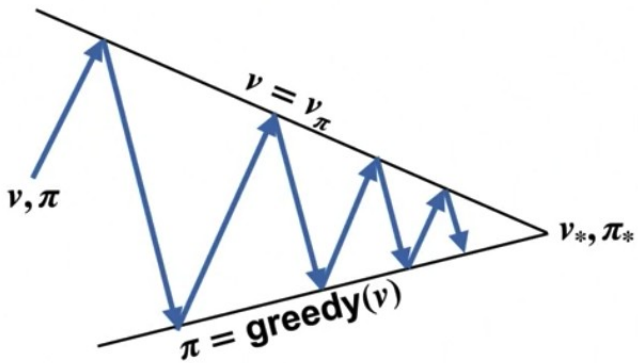
Generalized Policy Iteration

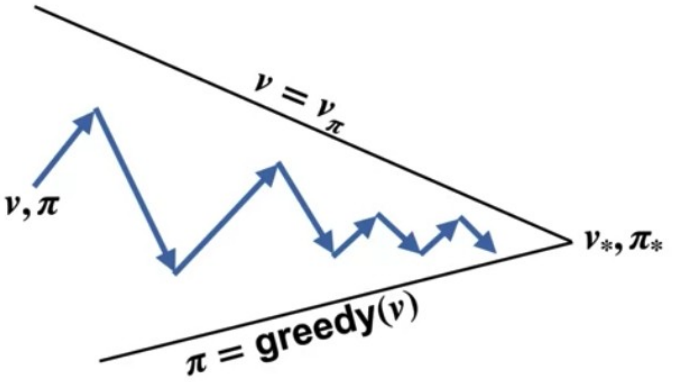
Objectives

- Understand the framework of generalized policy iteration
- Understand the distinction between synchronous and asynchronous dynamic programming methods

Generalized Policy Iteration

- Generalized Policy Iteration (GPI) is a framework in reinforcement learning (RL) that unifies various algorithms for policy evaluation and improvement.
- It aims to find an optimal policy by iteratively updating policies and value functions in a coordinated





Generalized Policy Iteration

- The key idea behind GPI is to alternate between two main processes:
  - Policy Evaluation: the current policy is evaluated to estimate its value function. The value function represents the expected cumulative reward obtained by following the policy from each state.

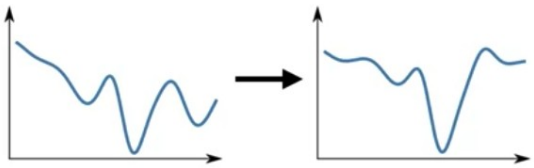
Generalized Policy Iteration

- Policy Improvement: After evaluating the current policy, the agent attempts to improve it by selecting actions that are expected to yield higher rewards.
- These two steps are repeated iteratively, with each iteration potentially improving the overall policy until convergence is achieved.

Value Iteration

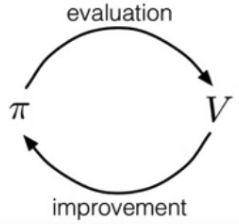
- We still sweep over all the states and greedify with respect to the current value function. However, we do not run policy evaluation to completion. We perform just one sweep over all the states. After that, we

Evaluation Step:



Improvement Step:

$\pi \leftarrow \textbf{greedy}(v)$



Value Iteration

- Value Iteration for estimating p~ p*

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \text{argmax}_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

Dynamic Programming

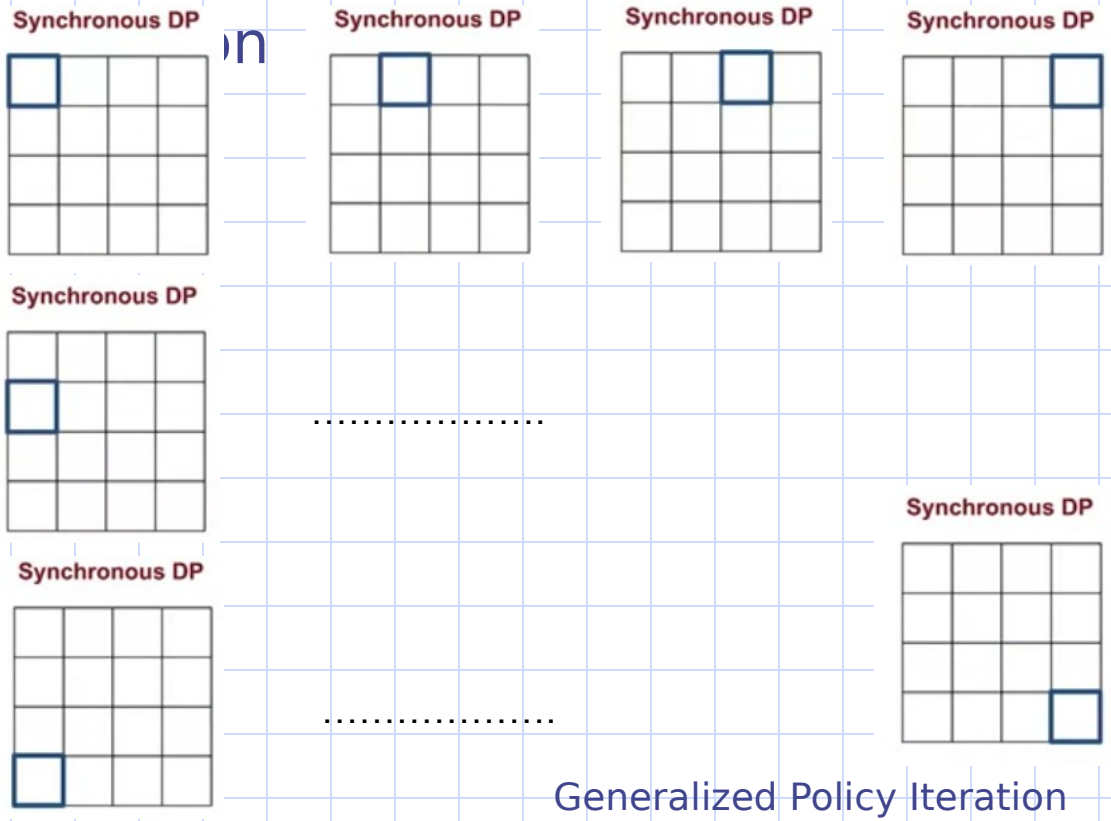
- Synchronous Dynamic Programming:
  - The value functions are updated simultaneously for all states or state-action pairs in each iteration.
  - The updates are synchronized across all states or state-action pairs.
  - Value iteration, which is a classic dynamic programming algorithm for finding optimal value functions.

Dynamic Programming

- Synchronous Dynamic Programming:
  - Synchronous dynamic programming can be computationally efficient when implemented in environments with a relatively small number of states or state-action pairs.

Dynamic Programming

- Synchronous DP: Value iteration sweeps the entire state space on each iteration just like policy



Dynamic Programming

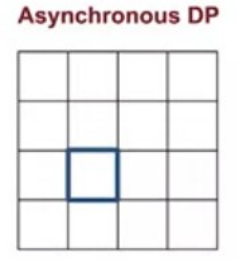
- Asynchronous Dynamic Programming:
  - The value functions are updated one at a time, either randomly or according to some specific order.
  - The updates are not synchronized across states or state-action pairs, and each update may use the most recent value estimates available at the time.
  - It is particularly useful when dealing with large-scale or continuous-state spaces, where updating all values simultaneously may be computationally infeasible.

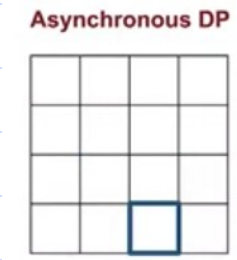
Dynamic Programming

- Asynchronous Dynamic Programming:
  - There are various strategies for selecting which values to update and in what order, such as prioritized sweeping, TD(λ), or online Q-learning.
  - Asynchronous dynamic programming methods often exhibit more flexibility and scalability compared to synchronous methods but may require careful tuning of update schedules and convergence criteria.

Dynamic Programming

- Asynchronous dynamic programming algorithms update the values of states in any order, they do not perform systematic sweeps.





Summary

- Understand the framework of generalized policy iteration
- Understand the distinction between synchronous and asynchronous dynamic programming methods

Q & A