Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – III (Reinforcement Learning)

AY: 2023 - 24

Experiment 6

Name: Kresha Shah

SAP ID: 60009220080

Aim: To solve Grid World using Policy Iteration and Value Iteration techniques of Dynamic Programming.

Theory:

Within Reinforcement Learning, dynamic programming (DP) can be described as a collection of algorithms that can be used to compute optimal policies iteratively, given a perfect model of the environment as a Markov Decision Process (MDP).

In Markov Decision Process (MDP), we assume that the state, action, and reward sets, S, A(s), and R, for s ∈ S, are finite, and that its dynamics are given by a set of probabilities p(s,r|s,a), for all s ∈ S, a ∈ A(s), r ∈ R, and s∈ S+ (S+ is S plus a terminal state if the problem is episodic).

The key idea of DP, and of reinforcement learning generally, is the use of value functions to organize andstructure the search for good policies.

DP algorithms are obtained by turning Bellman equations into assignments, that is, into update rules for improving approximations of the desired value functions.

Policy Iteration

In policy iteration, we start by choosing an arbitrary policy. Then, we iteratively evaluate and improvethe policy until convergence. In the beginning, we don't care about the initial policy being optimal or not. During the execution, we concentrate on improving it on every iteration by repeating policy evaluation and policy improvement steps. Using this algorithm we produce a chain of policies, where each policy is an improvement over the previous one. We conduct policy evaluation and policy improvement steps until the policy doesn't improve anymore.

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

Formulae used in Policy Iteration:

1. Policy Evaluation:

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

2. Policy Improvement:

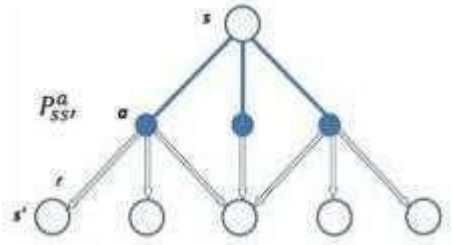
$$\pi(s) = arg\max_a \sum_{s',r'} p(s', r|s, a)[r + \gamma V(s')]$$

Value Iteration:

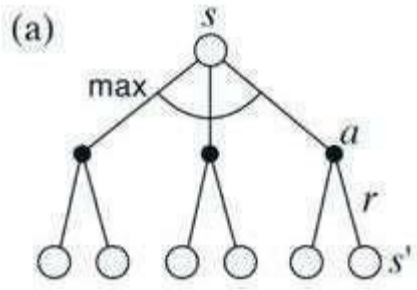
In value iteration, we compute the optimal state value function by iteratively updating the estimate. Westart with a random value function. At each step, we update it. Hence, we look-ahead one step and go over all possible actions at each iteration to find the maximum.

Formula used in Value Iteration:

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

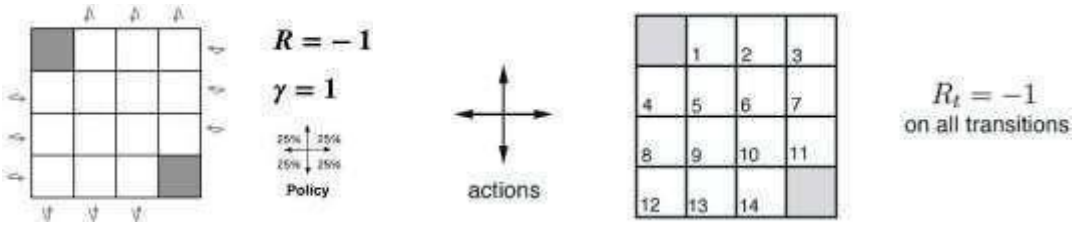


Policy Iteration



Value Iteration

Grid World:



Department of Computer Science and Engineering (Data Science)

Simple Grid World via Sutton and Barto RL Book

In this problem, we are given a grid (4 x 4 in this case). The goal is to reach either the top-left or thebottom-right square (grey coloured) from any other square on the grid, with maximum reward.

- You can jump one square in either of the North, South, East, or West directions from any given square.
- Each jump (in any direction) earns a reward of -1 except from the grey squares where the reward is 0.
- Jumping off the grid earns you -1, but you stay in the same square.
- Once you reach the goal, the game is over, and you cannot jump anywhere (i.e., even on jumping, you stay in the same square with a 0 reward).

Algorithm:

1. Policy Iteration

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
      $\Delta \leftarrow 0$
      For each $s \in \mathcal{S}$:
         $v \leftarrow V(s)$
         $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\left[r + \gamma V(s')\right]$
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      $a \leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
      If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return V and π; else go to 2

Department of Computer Science and Engineering (Data Science)

2. Value Iteration

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  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 $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
  $\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

Department of Computer Science and Engineering (Data Science)

Lab Assignment to do:

1. Generate a class for defining the GridWorld (4x4) Environment and its variables. (You can referto the code given below)
2. Implement Iterative Policy Evaluation and calculate the number of iterations needed toconverge the values for the final answer. (O)
3. Implement Policy Iteration using Policy evaluation
4. Implement Value Iteration (get similar results to the Policy Iteration Step)

Colab link:
https://colab.research.google.com/drive/1F6h_hPFfyOkqwmkHBLGsw_oj11h0z7s2?usp=sharing