Topics in RL – Project 1

February 2025

Common Instructions

For each project, you should implement any 4 of the following 5 algorithms using only python to find the optimal policy:

1. Value iteration
2. Policy iteration
3. MC control (on and/or off)
4. TD control using SARSA(λ)
5. Q learning

Do not use any RL packages or ready made commands/codes for any of the above algorithms. We want to see a code from scratch. For the first and second algorithm, assume transition probabilities and rewards are known and then show that the algorithms converge to the optimal value function/policy. For rest of the algorithms, use the transition probabilities to only generate the episodes online. For algorithms 3-5, plot the total discounted reward per episode and cumulative rewards over episodes. Also plot the instantaneous and cumulative regrets for the RL algorithms defined as follows.

Instantaneous episodic regret: It is the difference between the total discounted reward earned by your RL algorithm in the current episode and the expected cumulative discounted reward earned by the optimal policy in an episode (essentially $V_\alpha(s)$ if the episode started in state s). It measures how much reward you are losing choosing a sub-optimal policy.

Cumulative regret: At the current iteration/episode, this is running/cumulative sum of all the previous instantaneous episodic regrets till now.

Value and Policy iteration for the finite time horizon: Recall from the class that VI and PI were used for infinite horizon discounted problem settings. In fact, in such settings deterministic, stationary and Markovian policies are optimal and so in policy iteration we always chose deterministic and stationary policy. We also know that for a finite horizon problem, the optimal policy is non-stationary. In that case, how will you adapt that equation to solve the finite time horizon problem? Give it a thought! The following paragraph with Q learning for finite horizon will act as a hint.

Q-learning for finite horizon: Q-learning in the standard form assumes deterministic and stationary policy for an infinite horizon MDP. In many of the projects, the problem only makes sense for finite horizons. You can adapt Q-learning to finite horizon as shown in this link. Instead of assuming stationary policy and using $Q(s, a)$, use the notation $Q_t(s, a)$, and make the update to it in a similar fashion. See the following algorithm from the above paper.

Algorithm 1 Finite Horizon Q-Learning

Notation:

$Q_n^m(i,a)$: Q-value at state i, action a, stage n, recursion m.

$a(m)$: step-size at recursion index m

$Q_N(i,a)$: Q-value for state i and action a at terminal stage (N).

$g_n(i,a,j)$: Single stage reward for stage n where current state is i, action is a and next state is j. $g_N(i)$: Terminal reward at the N^{th} stage when terminal state is i.

$A(j)$: Set of feasible actions in state j.

$\eta(i,a)$: Sampling function taking input (i,a) as state-action pair and returns the next state.

Input: Samples of the form $\big(i$ (current state), a (action), r (reward), j (next state)$\big)$.

Output: Updated Q-value $Q_n^{m+1}(i,a)$ estimated after m iterations of the algorithm.

Initialization: $Q_n^0(i,a) = 0, \forall (i,a), n = 0,\ldots,N-1,$ and $Q_N^0(i,a) = g_N(i), \forall (i,a)$

procedure FINITE HORIZON Q-LEARNING:

$a(m) = \left\lceil \frac{1}{(m+1)/10} \right\rceil$

$j = \eta(i,a)$ (from samples)

$Q_n^{m+1}(i,a) = \big(1 - a(m)\big)\Big(Q_n^m(i,a)\Big) + a(m)$

$\times\Big(g_n\big(i,a\big) + \min_{b\in A(j)} Q_{n+1}^m\big(j,b\big)\Big), n = 0,1,\ldots,N-1,$

$Q_N^{m+1}(i,a) = g_N(i), \forall (i,a)$ tuples.

return $Q_n^{m+1}(i,a)$

Project 1: Inventory Management problem (Team 5)

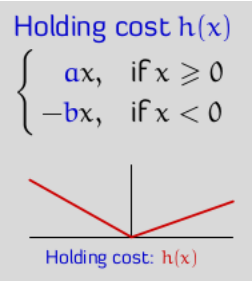
Retail stores stockpile products in warehouses to meet the random demand. Additional stocks are procured at regular intervals. Let X_t denote the amount of stock before the t-th procurement. In this example, time denotes the number of additional stock procurements. At time t, the store may procure an additional stock U_t $(\leq U)$ units for price p per unit. Thus the total procurement cost is pU_t.

The random demand W_t is i.i.d. with distribution P_W. The stock available at the next time is $X_{t+1} = X_t + U_t - W_t$, where a negative stock denotes backlogged demand.

The holding cost for the stock is given by $h(x)$ where a is the per-unit storage cost and b is the per-unit backlog cost.

Per-stage cost is $c(X_{t+1}, U_t) = h(X_{t+1}) + pU_t$. Find the optimal inventory control strategy to minimize the expected total cost over a finite horizon.

Parameters: Take P_W as uniform[0,10].



Reference: Page 14 of Aditya Mahajan Slides on Markov Decision Processes: Sequential decision-making with perfect observation. You can also enrich the problem based on the inventory control problem.

Project 2: Call options (Team 4)

An investor has a *call option* to buy one share of a stock at a fixed price p and has T days to *exercise* this option. For simplicity, assume that the investor makes a decision at the beginning of each day.

The investor may decide not to exercise the option but if he does exercise the option when the stock price is s, he effectively gets $(s - p)$.

Assume that the price of the stock varies with independent increments, *i.e.*, the price on day $t + 1$ is

$$S_{t+1} = S_t + W_t$$

where $\{W_t\}_{t\geq 1}$ is an i.i.d. process.

Your task is to create an agent that decides when to exercise (or not exercise) the call option in the given span of T days. Assume $p \in \mathbb{N}$ and assume W_t to be (discrete) uniformly distributed with endpoints $-\epsilon$ and $+\epsilon$ for some $\epsilon \in \mathbb{N}$. For example, for $\epsilon = 5$, $W_t \sim \mathcal{U}\{-5, 5\}$. You can spice up the problem by relaxing some of these assumptions, using different noise models, working with real data etc etc. Also see this link

References:

- Call Option Wiki
- Discrete Uniform Distribution Wiki
- Aditya Mahajan Notes on Call Option
- Page 23 of Aditya Mahajan Slides on Markov Decision Processes: Sequential decision-making with perfect observation

Project 3: A machine replacement model (Team 8)

Consider a manufacturing process, where the machine used for manufacturing deteriorates over time. Let $\mathcal{S} = \{0, 1, \ldots, n\}$ represent the condition of the machine. The higher the value of s, the worse the condition of the equipment.

A decision maker observes the state of the machine and has two options: continue operating the machine or replace it with a a new and identical piece of equipment. Operating the machine in state s costs $h(s)$, where $h(\cdot)$ is a weakly increasing function; replacing the machine costs a constant amount K.

When the machine is operated, it's state deteriorates according to

$$S_{t+1} = \min(S_t + W_t, n)$$

where $\{W_t\}_{t\geq 1}$ is an i.i.d. discrete process.

You can assume $W_t \sim$ Bernoulli(p) for some $p \in (0, 1)$. Also, take $K \in (h(s_i), h(s_{i+1}))$ for some $s_i \in S$. And $h(x) = x^2$

Reference: Aditya Mahajan Notes on Machine Replacement

Project 4: A House Selling Example (Team 6)

An individual wants to sell his house and an offer comes in at the beginning of each day. Assume that the successive offers are independent and an offer is j with probability P_j, $j = 0, 1, \ldots, N$ (here is j is "ranking" of offers). We suppose, however, that any offer not immediately accepted is not lost but may be accepted at any later date. Also, a maintenance cost of C is incurred each day the house remains unsold.

The state at time t will be the largest offer received up to t (including the offer at t). Therefore, (i is currently the best offer "rank", j is the next day offer "rank")

$$P_{ij} = \begin{cases} 0 & j < i \\ \sum_{k=0}^{i} P_k & j = i \\ P_j & j > i \end{cases}$$

Assume N = 25, i = 10, and P to follow a uniform distribution. Find the optimal strategy.

Reference: Applied Probability Models with Optimization Application, Sheldon Ross (Chapter 6).

Project 5: Windy Gridworld (Stochastic Wind) (Team 2)

Figure 1 shows a standard grid world, with start and goal states, but with one difference: there is a crosswind running upward through the middle of the grid. Assume eight possible actions: `up`, `down`, `right`, and `left` and also the diagonal moves.

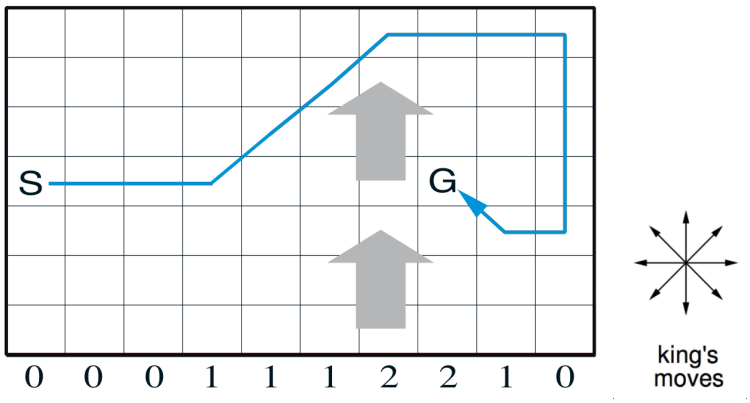


Figure 1: Windy Gridworld

In the middle region, the resultant next states are shifted upward by a "wind", the strength of which varies from column to column. Assume that the effect of the wind, if there is any, is stochastic, sometimes varying by 1 from the mean value given below each column. That is, a third of the time you move exactly according to these values but also a third of the time you move one cell above that, and another third of the time you move one cell below that. For example, if you are one cell to the right of the goal and you move left, then one-third of the time you move one cell above the goal, one-third of the time you move two cells above the goal, and one-third of the time you move to the goal. Note there's no wind in the columns that have the value 0 given below them.

Your task is to learn the optimal policy for this setting.

References:

- Pages 130, 131 of Sutton and Barto - Reinforcement Learning: An Introduction (2nd Edition)
- 49:43 timestamp onwards of RL Course by David Silver - Lecture 5: Model Free Control

Project 6: Riverswim (Team 1)

The RiverSwim MDP models an agent swimming in a river who can choose to swim either left or right. The MDP consists of six states arranged in a chain with the agent starting in the leftmost state (s = 1). If the agent decides to move left i.e with the river current then it is always successful but if it decides to move

right it might fail with some probability. If the final state of the agent is s_1, the agent receives an award of 0.005. If the final state of the agent is s_6, the agent receives an award of 1. Each episode is reset every H = 20 steps. The MDP state diagram is shown below:

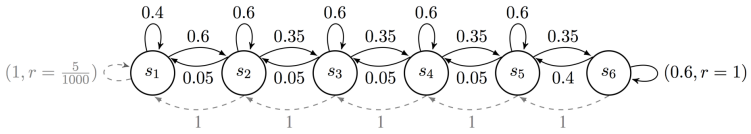


Figure 2: RiverSwim

In the above figure, Solid and dotted arrows denote the transition probabilities for actions 'right' and 'left', respectively.

Reference: Strehl and Littman - An analysis of model-based interval estimation for Markov decision processes

Project 7: Portfolio Optimization with single asset (Team 3)

Consider the portfolio optimization problem at this link. You will have to consider one and two assets in your portfolio that you have to optimize. This is discussed in chapter 3 and 4. In the illustrated examples, where are only 3 weights $\{-1, 0, 1\}$ that are considered. You can consider -0.5 and 0.5 in addition as well.

Project 8: Finite inventory pricing (Team 7)

Assume you operate a chartered plane of K seats with a booking window of T days.i.e., the plane leaves from A to B after every T days. Bookings can be made only for that particular segment and for the next departing flight and not for future departures. As an operator, you want to come up with an optimal pricing policy $p_t, t = 1, 2, \ldots, T$ that will maximize your total revenue over T days. Time is discrete, and on day t, the demand for seats at price p_t, denoted by $d(p_t)$ is a random variable supported on positive integers. Some examples are Bernoulli, Poisson, Geometric random variables. Note that if available seats are less than the demand on a day, then some demand is lost (but there is no cost for lost sales). First perform VI or PI to get the optimal pricing policy. Now assume that the demand function is unknown and use RL algorithms to see if the optimal policy is learnt.