CME241 Assignment 13

Quinn Hollister

February 2022

1 Implement Tabular MC Control algorithm

```
def glie_mc_tabular_control(
    mdp,
    states,
    approx_0,
    gamma,
    eps_as_func_of_episodes,
    episode_length_tolerance = 1e-6
) -> Iterator[QValueFunctionApprox[S,A]]:

    q: QValueFunctionApprox[S,A] = approx_0 #Should be a [(s,a): q] dict
    p: Policy[S,A] = epsilon_greedy_policy_tab(q, mdp, 1.0)
    yield q

    counts = defaultdict(lambda: 0)


    num_episodes = 0
    while True:
        trace = mdp.simulate_actions(states, p)
        num_episodes += 1
        for step in returns(trace, gamma, episode_length_tolerance):
            counts[(step.state, step.action)] += 1
            q[(step.state, step.action)] += (1/counts[(step.state, step.action)])* /
            (step.return_ - q[(step.state, step.action)])
        p = epsilon_greedy_policy_tab(q, mdp, eps_as_func_of_episodes(num_episodes))
        yield q

def greedy_policy_from_qvf_tab(q, actions):
    def optimal_action(s):
        q_vals = {a: q[(NonTerminal(s),a)] for a in actions(NonTerminal(s))}
        return max(q_vals, key = q_vals.get)
    return DeterministicPolicy(optimal_action)

def epsilon_greedy_policy_tab(q, mdp, epsilon):
    def explore(s, mdp = mdp):
        return mdp.actions(NonTerminal(s))
    return RandomPolicy(Categorical(
        {UniformPolicy(explore): epsilon,
         greedy_policy_from_qvf_tab(q, mdp.actions): 1 - epsilon}
    ))
```

2 Implement Tabular Q-Learning

```
def tabular_q_learning(
    mdp,
    policy_from_q,
    states,
    approx_0,
    gamma,
    max_episode_length
) -> Iterator[QValueFunctionApprox[S,A]]:
    q = approx_0
    yield q

    counts = defaultdict(lambda: 0)
    alpha = 0.03
    H = 1000
    beta = 0.5

    while True:
        state = states.sample()
        steps = 0
        while isinstance(state, NonTerminal) and steps < max_episode_length:
            policy = policy_from_q(q, mdp)
            action = policy.act(state).sample()
            next_state, reward = mdp.step(state, action).sample()
            next_return = max(
                q[(next_state, a)]
                for a in mdp.actions(next_state)
                ) if isinstance(next_state, NonTerminal) else 0.
            counts[(state,action)] += 1
            alpha_n = alpha / (1 + ((counts[(state,action)] - 1)/H)**beta)
            q[(state, action)] += (alpha_n)*(reward + gamma*next_return - q[(state,action)
            yield q
            state = next_state
```

3 Simple Inventory MDP Cap Example

Below, I share the optimal value functions and optimal policies achieved by the two approximation methods and compare them to the analytic solution.

Analytic:

NonTerminal(state=InventoryState(on$_hand = 1, on_order = 1)) : -28.991900091403522,$
$NonTerminal(state = InventoryState(on_hand = 0, on_order = 1)) : -27.660960231637496,$
$NonTerminal(state = InventoryState(on_hand = 1, on_order = 0)) : -28.660960231637496,$
$NonTerminal(state = InventoryState(on_hand = 0, on_order = 2)) : -27.991900091403522,$
$NonTerminal(state = InventoryState(on_hand = 2, on_order = 0)) : -29.991900091403522,$
$NonTerminal(state = InventoryState(on_hand = 0, on_order = 0)) : -34.89485578163003$
$ForStateInventoryState(on_hand = 0, on_order = 0) : DoAction1$
$ForStateInventoryState(on_hand = 0, on_order = 1) : DoAction1$
$ForStateInventoryState(on_hand = 0, on_order = 2) : DoAction0$
$ForStateInventoryState(on_hand = 1, on_order = 0) : DoAction1$
$ForStateInventoryState(on_hand = 1, on_order = 1) : DoAction0$
$ForStateInventoryState(on_hand = 2, on_order = 0) : DoAction0$

MC Control

NonTerminal(state=InventoryState(on$_hand = 0, on_order = 0)) : -35.503215118052246,$
$NonTerminal(state = InventoryState(on_hand = 2, on_order = 0)) : -30.336675438964534,$
$NonTerminal(state = InventoryState(on_hand = 1, on_order = 1)) : -29.35315787037659,$
$NonTerminal(state = InventoryState(on_hand = 1, on_order = 0)) : -28.939052659730574,$
$NonTerminal(state = InventoryState(on_hand = 0, on_order = 1)) : -27.943732795253123,$
$NonTerminal(state = InventoryState(on_hand = 0, on_order = 2)) : -28.335770099725305$
$ForStateInventoryState(on_hand = 0, on_order = 0) : DoAction2$
$ForStateInventoryState(on_hand = 0, on_order = 1) : DoAction1$
$ForStateInventoryState(on_hand = 0, on_order = 2) : DoAction0$
$ForStateInventoryState(on_hand = 1, on_order = 0) : DoAction1$
$ForStateInventoryState(on_hand = 1, on_order = 1) : DoAction0$
$ForStateInventoryState(on_hand = 2, on_order = 0) : DoAction0$

Q Learning

NonTerminal(state=InventoryState(on$_hand = 0, on_order = 0)) : -34.57887962360952,$
$NonTerminal(state = InventoryState(on_hand = 2, on_order = 0)) : -30.16224382407824,$
$NonTerminal(state = InventoryState(on_hand = 1, on_order = 1)) : -28.693013053803757,$
$NonTerminal(state = InventoryState(on_hand = 1, on_order = 0)) : -28.35708328317616,$
$NonTerminal(state = InventoryState(on_hand = 0, on_order = 1)) : -27.329398206917812,$
$NonTerminal(state = InventoryState(on_hand = 0, on_order = 2)) : -27.579792381783722$
$ForStateInventoryState(on_hand = 0, on_order = 0) : DoAction1$
$ForStateInventoryState(on_hand = 0, on_order = 1) : DoAction1$
$ForStateInventoryState(on_hand = 0, on_order = 2) : DoAction0$
$ForStateInventoryState(on_hand = 1, on_order = 0) : DoAction1$
$ForStateInventoryState(on_hand = 1, on_order = 1) : DoAction0$
$ForStateInventoryState(on_hand = 2, on_order = 0) : DoAction0$

Commentary:
It looks like our answers are reasonable when we compare them to the DP solution above, with each value function within 2%. However, it seems that the optimal action differs for one state, the (0,0) state. The fact that the optimal value functions don't differ that much seem to tell me that the optimal policy choice of Action 1 over Action 2 leads to very negligible differences in the reward structure, and since Tabular MC Control doesn't have the best convergence properties, that we've reached a "good enough" optimal policy, given the facts. With Q-learning, we've achieved the same optimal policy and have a very good approximation to the true value function.