Problem Set 2 (Approximate Dynamic Programming)

Due on May 27th. Use your favorite programming language & send a report along with the code in any easily accessible format. (E.g., a report in pdf, html, and the code separately in a .py file, or all together in a jupyter notebook.)

The goal of this problem set is to study various aspects of TD and LSTD methods in a simple Markov decision process. The process we study is the same controlled queueing system as considered in the first problem set. The definition is repeated for convenience below:

The state space consists of the integers $0, 1, 2, \ldots, N-1$, with $N = 100$, corresponding to the length of the queue.

The action space consists of two actions: action a_{low} corresponding to a low service rate of $q(a_{\text{low}}) = q_{\text{low}} = 0.51$ and action a_{high} corresponding to a high service rate of $q(a_{\text{high}}) = q_{\text{high}} = 0.6$.

The reward function assigns a reward of $r(x, a) = -(x/N)^2 - c(a)$, with $c(a)$ being the cost of action a. This cost is defined for the two actions as $c(a_{\text{low}}) = 0$ and $c(a_{\text{high}}) = 0.01$.

The transition function describes the dynamics of the queue as follows. The arrival rate $p = 0.5$ models the rate at which new requests arrive into the queue and the controlled service rate models the rate at which requests leave the queue. In each round, the queue length increases by 1 with probability p and decreases by 1 with probability $q(a)$ corresponding to the action a taken. Precisely, letting x_t be the current queue length, a_t be the action taken by the agent, we define $I_t \in \{0, 1\}$ as the increment such that $\mathbb{P}[I_t = 1] = p$ and $S_t \in \{0, 1\}$ as the decrement such that $\mathbb{P}[S_t = 1] = q(a_t)$, the queue length is updated in each round as

$$x_{t+1} = \text{trunc}\,(x_t + I_t - S_t)\,,$$

where the "trunc" operator truncates the value to the interval $[0, N-1]$: $\text{trunc}(x) = \min\{N-1, \max\{x, 0\}\}$.

The discount factor will be set as $\gamma = 0.9$.

The task will be to evaluate some simple policies and find a near-optimal policy via temporal-difference learning methods and linear function approximation. We will study three feature maps:

A fine feature map: a $N+1$-dimensional feature vector, with its i-th component defined as $\phi_i^{\text{fine}}(x) = \mathbf{1}\{x = i\}$ for each $i = 0, 1, 2, \ldots, N$. This feature map can represent all functions on the state space.

A coarse feature map: an $N/5$-dimensional feature vector with its i-th component defined as

$$\phi_i^{\text{coarse}}(x) = \mathbf{1}\{x \in [5(i-1), 5i-1]\}.$$

This feature map can represent piecewise constant functions on the state space.

A piecewise linear feature map: an $2 \times N/5$-dimensional feature vector with its first $N/5$ entries being equal to the previous feature map (i.e., $\phi_i^{\text{pwl}} = \phi_i^{\text{coarse}}$ for $i = 1, 2, \ldots, N/5$) and the second $N/5$ entries defined as

$$\phi_{N/5+i}^{\text{pwl}}(x) = \mathbf{1}\{x \in [5(i-1), 5i-1]\} \cdot (x - 5(i-1))/5,$$

for $i = 1, 2, \ldots, N/5$. The normalization constant $1/5$ is present to make sure that all features are bounded in $[0, 1]$. This feature map can represent piecewise linear functions on the state space.

Problem 1: Approximate policy evaluation

The first problem is to study the performance of the same simple policies as in the first problem set, but this time using TD(0) and LSTD with 10^4, 10^5, 10^6, and 10^7 sample transitions, under the three feature maps. Sample the transitions from one single episode starting from initial state $x_0 = N$ (a full queue). Plot the resulting value functions and compare them to the results you have obtained in the first problem set.
For completeness, the two policies to study are:

Lazy policy: the policy that always uses the low service rate: $\pi(x) = a_{\text{low}}$.

Aggressive policy: the policy that uses the low service rate for short queues ($\pi(x) = a_{\text{low}}$ for $x < 50$) and the high service rate for long queues ($\pi(x) = a_{\text{high}}$ for $x \geq 50$).

Hints: For TD(0), set a sequence of decreasing stepsize parameters with $\alpha_t = \frac{a}{t+b}$ for some tuning parameters a, b. (Relatively large values of both a and b such as $a = b = 10^5$ are recommended so that the stepsize decay is not too fast.) For LSTD, you can use a Moore–Penrose pseudoinverse (e.g., `numpy.linalg.pinv`) or a regularized matrix inverse $(M + \sigma I)^{-1}$ instead of M^{-1} for a small σ (e.g., 10^{-10}) if A_T is singular.

Problem 2: Approximate policy iteration

Using the LSTD implementation, implement an approximate policy iteration method where the following steps are repeated in each iteration k:

Policy evaluation: Evaluate the current policy π_k using LSTD and 10^5 sample transitions, and let $\widehat{V}_k$ denote the result.

Policy improvement: Define the Q-function estimate as

$$\begin{aligned}\widehat{Q}_k(x,a) = & r(x,a) \\ & + \gamma(1-p)\big(q(a)\,\widehat{V}_k(x-1) + (1-q(a))\,\widehat{V}_k(x)\big) \\ & + \gamma p\big(q(a)\,\widehat{V}_k(x) + (1-q(a))\,\widehat{V}_k(x+1)\big)\end{aligned}$$

and set the new policy as the greedy policy with respect to $\widehat{Q}_k$:

$$\pi_{k+1}(x) = \arg\max_a \widehat{Q}_k(x,a).$$

Plot the value function $\widehat{V}_k$ after 10 and 100 iterations and compare it with the value functions obtained in the previous problem. Also plot the service rates assigned by final policy π_k as a function of x (i.e., $q(\pi_k(x))$ for $x = 1,2,\ldots,N$) and interpret the results, comparing them to the optimal value functions you have obtained in the first problem set. Did the final policy robustly improve on the initial policy?