Reinforcement Learning May 30, 2021

Exercise 4

Lecturer: Yishay Mansour *TA: Aviv Rosenberg*

Theory

Question 1: Bandits with side information.

Consider a stochastic 2-armed bandit where each arm's i's reward sequence is generated independently from a Bernoulli distribution with parameter μ_i ($i \in \{1,2\}$). There are also two known constants, $0 < \mu_A < \mu_B < 1$, where $\mu_1, \mu_2 \in \{\mu_a, \mu_b\}$ and $\mu_1 \neq \mu_2$ (so the only uncertainty is which arm has μ_b as expected reward and therefore optimal). Let $\Delta := \mu_b - \mu_a$. The goal is to have low regret, $R_T := T\mu_b - E[\sum_{t=1}^{T} \mu_{I_t}]$, where $I_t \in \{1,2\}$ is the chosen arm in round t.
As you've seen in class, the UCB algorithm achieves a regret bound of $O(\log T/\Delta)$. But this completely ignores the (potentially huge) side information about the arms' rewards which is known beforehand. The problem asks you to analyze a bandit algorithm whose regret does not scale with T!

Consider the following algorithm.

1. Play each arm once (same as UCB algorithm- $I_1 \leftarrow 1, I_2 \leftarrow 2$).
2. For $t \geq 3$:
   if there exists an arm whose current empirical mean so far exceeds $\frac{\mu_a+\mu_b}{2}$,
   - play the arm with the highest empirical mean.
   else,
   - play both arms one after another, i.e., $I_t \leftarrow 1$ and $I_{t+1} \leftarrow 2$.

Let $\hat{\mu}_{i,T_i(t)}$ be the current empirical mean of arm i at time t, where $T_i(t)$ is the number of times arm i has been played up to time t i.e., $\hat{\mu}_{i,T_i(t)} = \frac{1}{T_i(t)} \sum_{\tau \in \{I_\tau = i,\, \tau \leq T_i(t)\}} X_\tau$.
Without loss of generality, let arm 1 be the optimal arm.
We can split the times when arm 2 is played according to the value of its current empirical mean into two sets:
(i) times when $\hat{\mu}_{2,T_2(t)} \geq \frac{\mu_a+\mu_b}{2}$.
(ii) times when $\hat{\mu}_{2,T_2(t)} \leq \frac{\mu_a+\mu_b}{2}$.
Together with $t = 2$, we can split the times that arm 2 is played into two sets:

$$\{I_t = 2, t \geq 3\} = \{t \geq 3, I_t = 2, \hat{\mu}_{2,T_2(t)} > \frac{\mu_a + \mu_b}{2}\} \bigcup \{t \geq 3, I_t = 2, \hat{\mu}_{2,T_2(t)} \leq \frac{\mu_a + \mu_b}{2}\}$$

(a) Bound (from above) the sum of probabilities of playing arm 2 at times when event (i) occurs, using Hoeffding's inequality*.
Hints: recall that (1) $\sum_{j=1}^{T} f(j) \leq \sum_{j=1}^{\infty} f(j)$ for any f such that $\forall_j f(j) \geq 0$ and (2) that $e^x \geq 1 + x$.
(b) Bound (from above) the sum of probabilities of playing arm 2 at all times when event (ii) occurs by using the definition of the algorithm and relating event (ii) to an event involving the empirical mean of arm 1. Obtain a bound by applying Hoeffding as before.
(c) Put together the conclusions of the previous parts and derive a regret bound that depends only on Δ (independent of T).
***Hoeffding's inequality**: For i.i.d random variables X_i bounded in $[0,1]$ with $\mu := E[X_i]$, then $\Pr[\sum_{i=1}^{n} X_i \geq n(\mu + \epsilon)] \leq \exp(-2n\epsilon^2)$ and $\Pr[\sum_{i=1}^{n} X_i \leq n(\mu - \epsilon)] \leq \exp(-2n\epsilon^2)$.

Question 2: Policy Gradient with the exponential distribution

The probability density function of the exponential distribution is defined for any $x \geq 0$:

$$f(x) = \lambda \cdot e^{-\lambda x}$$

Where λ is called the rate parameter. We use the exponential distribution to sample actions $a \geq 0$ as follows:

- Each state s is encoded by a vector $\phi(s)$.
- A policy π is characterized by a vector $\theta \in R^d$
- At each state s, we sample an action from an exponential distribution with parameter $\lambda = \exp(\theta^\intercal \phi(s))$.

(a) What is the probability of sampling an action $a \in [0,1]$? the answer should be a function of θ and $\phi(s)$.
(b) Calculate $\nabla_\theta \log(\pi(a|s;\theta))$.
(c) What is the REINFORCE update of the exponential distribution?

Programming Assignment

In this assignment you need to write a program that will learn to play the card game 21. You can implement your own code for the game or use an exisiting one (e.g., https://gist.github.com/mjhea0/5680216)

Game description: There is a deck of cards, total 52 cards. There are two players, the *house* and the *gambler*. The winner in the game is the player with the most number of points, which are less than (or equal to) 21.

When counting the points, each number card (2 to 10) has a value which is his number. Each face card (J, Q or K) is 10 points. The value of an ace A is 11 points (simplifying the rule of the real game where A can be either 1 or 11.)

At the beginning each player gets two cards, one is faced up (which you see) while the other is faced down (which you don't see). The gambler look at all its cards (and the house open card) and need to decide whether to ask for a another card (`hit`) or end (`stop`).

We fix the strategy of the house as follows. If the sum of the cards is 15 or less, the house perform `hit`, and if the sum is 16 or more it performs `stand`.

We model the game as an MDP whose states are labeled by the sum of the card of the gambler.

Task 1: Implement TD(0) and use it to compute the probability that the gambler wins, given that his policy is: If the sum is 18 or more then `stand` else `hit`.

Task 2: Implement SARSA, and compute an optimal policy against the house policy we consider. (Give as an output of the optimal action in each state, and the probability of winning from that state.)