Exercise 2.2 Bandit Example

Lenin G. Falconí[*]

2025-03-31

1 A 4 armed bandid problem

> *Exercise 2.2: Bandit example* Consider a k-armed bandit problem with k = 4 actions, denoted 1, 2, 3, and 4. Consider applying to this problem a bandit algorithm using $ϵ$-greedy action selection, sample-average action-value estimates, and initial estimates of Q_1(a) = 0, for all a. Suppose the initial sequence of actions and rewards is A_1 = 1, R_1 = -1, A_2 =2, R_2 =1, A_3 =2, R_3 = -2, A_4 =2, R_4 =2, A_5 =3, R_5 =0. On some of these time steps the ϵ case may have occurred, causing an action to be selected at random. On which time steps did this definitely occur? On which time steps could this possibly have occurred?

A k arm bandit algorithm is:

Algorithm 1 ϵ-Greedy k-Armed Bandit Algorithm

Input: Action set $A = \{1, 2, \ldots, k\}$, exploration rate ϵ
Initialize for all $a \in A$: $Q_1(a) \leftarrow 0$ and count $N_1(a) \leftarrow 0$
for $t = 1, 2, \ldots$ **do**
    if a random number in $[0, 1]$ is less than ϵ **then**
        Choose a random action $a_t \in A$
    else
        Select $A_t = \underset{a}{\operatorname{argmax}}(Q_t(a))$
    end if
    Execute action a_t, observe reward R_t
    Increment: $N_{t+1}(a_t) \leftarrow N_t(a_t) + 1$
    Update: $Q_{t+1}(a_t) \leftarrow Q_t(a_t) + \frac{1}{N_t(a_t)}\Big(R_t - Q_t(a_t)\Big)$
end for

*lenin.falconi@epn.edu.ec

1.1 Analyzing Greedy

Equation 2.3 in book has no subscripts but n seems to mean time step or step and the equation seems to be about a single state. Because of that I make those adjustments in the equation in the algorithm. Is that correct? The problem does not define any ϵ.

$t = 1$, $a = 1$:
$N_1 = [0, 0, 0, 0]$
$Q_1 = [0, 0, 0, 0]$
$A = [1, 2, 3, 4]$
$A_1 = 1$, $R_1 = -1$

$$N_1(1) = 0 + 1$$
$$Q_2(1) = Q_1(1) + \frac{1}{N_1(1)}[R_1 - Q_1(1)]$$
$$Q_2(1) = 0 + [-1]$$
$$Q_2(1) = -1$$

for $t = 1$, $a \in \{2, 3, 4\}$: $N_1(2) = N_1(3) = N_1(4) = 0$ Because none of those actions is taken. As a consequence $Q_1(2) = Q_1(3) = Q_1(4) = 0$. And the Q_1 vector would be:

$Q_1 = [-1, 0, 0, 0]$

So we continue with $t = 2, a = 2$. I will use to index the values of the vector starting in 1 $t = 2$, $a = 2$:
$N_2 = [1, 0, 0, 0]$
$Q_2 = [-1, 0, 0, 0]$

Getting $\underset{a}{\operatorname{argmax}}(Q_1) \neq 1$

$A = [1, 2, 3, 4]$
$A_1 = 2$, $R_1 = 1$

$$N_2(2) = 0 + 1$$
$$Q_3(2) = Q_2(2) + \frac{1}{N_2(2)}[R_2 - Q_2(2)]$$
$$Q_3(2) = 0 + [1]$$
$$Q_3(2) = 1$$

In the same way as before, I am reasoning that $Q_2(3) = Q_2(4) = 0$. As a consequence for $t = 3$, the agent chooses $a = 2$ $t = 3$, $a = 2$:
$N_3 = [1, 1, 0, 0]$
$Q_3 = [-1, 1, 0, 0]$

Getting $\underset{a}{\operatorname{argmax}}(Q_1) \neq 1$

$A = [1, 2, 3, 4]$
$A_1 = 2$, $R_1 = 1$

$$N_3(2) = 1 + 1$$
$$Q_4(2) = Q_3(2) + \frac{1}{N_3(2)}[R_3 - Q_3(2)]$$
$$Q_4(2) = 1 + 1/2 * (-2 - 1)$$
$$Q_4(2) = -1/2$$

Writing a python program. As I understand the problem, the idea is to estimate when an ϵ - greedy decision may have occurred given the sequence of states. Should I try to see what happens if I force to write the *bandit program* assuming an ϵ value. But in that case, do I need the other sequences or can they be ignored?

```
import numpy as np
A = [1,2,3,4]
Q = np.array([0,0,0,0], dtype=float)
N = np.array([0,0,0,0])
T = np.arange(6)
epsilon = 0.1
sequence = [(1,-1),(2,1),(2,-2), (2,2), (3,0)]
for n, (action, reward) in enumerate(sequence):
    print(f"step: {n+1}")
    print(f"current action: {action}")
    N[action-1]+=1
    print(f"N:{N}")
    Q[action-1] += 1/N[action-1]*(reward-Q[action-1])
    print(f"Q_{n+1+1}:{Q}")
    print("==========")
print(f"N:{N}")
print(f"Q:{Q}")
```

```
step: 1
current action: 1
N:[1 0 0 0]
Q_2:[-1.  0.  0.  0.]
==========
step: 2
current action: 2
N:[1 1 0 0]
Q_3:[-1.  1.  0.  0.]
==========
step: 3
current action: 2
N:[1 2 0 0]
Q_4:[-1.  -0.5  0.   0. ]
==========
```

```
step: 4
current action: 2
N:[1 3 0 0]
Q_5:[-1.          0.33333333  0.          0.        ]
==========
step: 5
current action: 3
N:[1 3 1 0]
Q_6:[-1.          0.33333333  0.          0.        ]
==========
N:[1 3 1 0]
Q:[-1.          0.33333333  0.          0.        ]
```

The first step gets a Q-value negative. Then, the argmax result could be either actions 2, 3 and 4, that have a value of 0. To break the tie, the agent chooses action 2. The ϵ case happens again in step 2, because $Q_3(2) = 1$. I think step 3 remains greedy choosing 2, but the result of the prediction for next state gives a negative value. A greedy approach would recommend actions 3 or 4 for step 4. Here greedy did not occur since for step 4 again action 2 is taken. Step 5 does not use greedy since $a_5 = \underset{a}{\operatorname{argmax}}(Q_5) = 2$.

1.2 Sample-averaged action value estimates

In this case I think we have to apply the equation:

$$Q_t(a) = \frac{\sum_{t=1}^{t-1} R_t \cdot \mathbb{1}_{A_t=a}}{\sum_{t=1}^{t-1} \mathbb{1}_{A_t=a}}$$

In this case I think we just calculate the product of Reward and Action for each action of the sequence:

$$Q_t(1) = \frac{\sum_{t=1}^{5} R_t \cdot \mathbb{1}_{A_t=1}}{\sum_{t=1}^{5} \mathbb{1}_{A_t=1}} = \frac{1}{1}$$

$$Q_t(2) = \frac{\sum_{t=1}^{5} R_t \cdot \mathbb{1}_{A_t=2}}{\sum_{t=1}^{5} \mathbb{1}_{A_t=2}} = \frac{0 + 2(1) + 2(-2) + 2(2)}{3} = \frac{2}{3}$$

$$Q_t(3) = \frac{\sum_{t=1}^{5} R_t \cdot \mathbb{1}_{A_t=3}}{\sum_{t=1}^{5} \mathbb{1}_{A_t=3}} = \frac{0 + 0 + 0 + 0 + 3(0)}{1} = 0$$

As a consequence $Q = [-1, 0.666, 0, 0]$

1.3 ϵ - Greedy Program

```
import numpy as np
A = [1,2,3,4]
Q = np.array([0,0,0,0], dtype=float)
N = np.array([0,0,0,0])
T = np.arange(6)
epsilon = 0.1
```

```
def bandit(action):
    sequence = [(1,-1),(2,1),(2,-2), (2,2), (3,0), (4,0)]
    rewards = [reward for act, reward in sequence if act == action]
    if not rewards:
        raise ValueError(f"No reward for action {action}")
    return np.random.choice(rewards)

for n in T:
    roll_dice = np.random.random()
    print(f"roll_dice: {roll_dice}")
    if roll_dice < epsilon:
        # explore
        action = np.random.randint(1,5)
        print(f"action explore:{action}")

    else:
        # exploit
        action = np.argmax(Q)+1
        print(f"action exploit:{action}")


    reward = bandit(action)
    print(f"step: {n+1}")
    print(f"current action: {action}")
    N[action-1]+=1
    print(f"N:{N}")
    Q[action-1] += 1/N[action-1]*(reward-Q[action-1])
    print(f"Q_{n+1+1}:{Q}")
    print("==========")
print(f"N:{N}")
print(f"Q:{Q}")
```

```
roll_dice: 0.41547343809500925
action exploit:1
step: 1
current action: 1
N:[1 0 0 0]
Q_2:[-1.  0.  0.  0.]
==========
roll_dice: 0.9297839425270602
action exploit:2
step: 2
current action: 2
N:[1 1 0 0]
Q_3:[-1.  2.  0.  0.]
==========
```

```
roll_dice: 0.7914951823916979
action exploit:2
step: 3
current action: 2
N:[1 2 0 0]
Q_4:[-1.  2.  0.  0.]
==========
roll_dice: 0.40242581791537846
action exploit:2
step: 4
current action: 2
N:[1 3 0 0]
Q_5:[-1.  2.  0.  0.]
==========
roll_dice: 0.3849103661520379
action exploit:2
step: 5
current action: 2
N:[1 4 0 0]
Q_6:[-1.  2.  0.  0.]
==========
roll_dice: 0.07361910965678975
action explore:3
step: 6
current action: 3
N:[1 4 1 0]
Q_7:[-1.  2.  0.  0.]
==========
N:[1 4 1 0]
Q:[-1.  2.  0.  0.]
```