Reinforcement Learning

Lesson 2: Multi-Armed Bandits

Edoardo Fazzari, 2022

Overview

- A *k*-armed Bandit Problem
- Action-value Methods
- Incremental Implementation
- Tracking a Non-stationary Problem
- Optimistic Initial Values
- Upper-Confidence-Bound Action Selection
- Gradient Bandit Algorithms
- Associative Search (Contextual Bandits)

A *k*-armed Bandit Problem

The problem

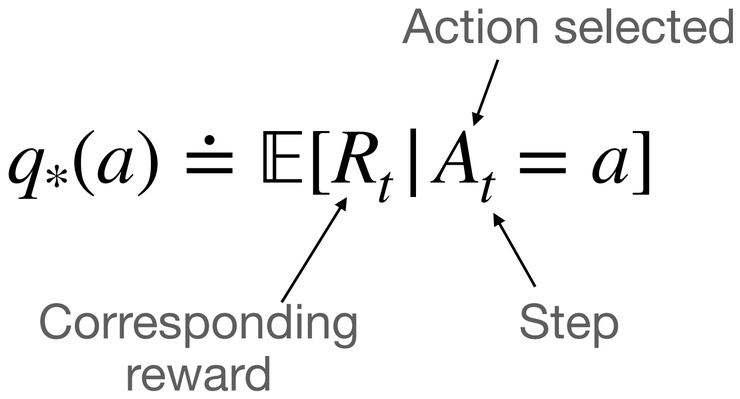
You are faced repeatedly with a choice among *k* different options, or actions. After each choice you receive a numerical reward chosen from a stationary probability distribution that depends on the action you selected

- Objective:
  - Maximize the expected total reward over some time period, e.g., over 1000 action selections, or *time steps*
- Slot machine analogy:
  - Each action selection is like a play of one of the slot-machine's levels, and the rewards are the payoffs for hitting the jackpot
- *Through repeated action selections you are to maximize your winnings by concentrating your actions on the best levels*

GOAL

Value of a particular action

- In *k*-armed bandit problem, each of the *k* actions has an expected or mean reward that that action is selected
    - Let's call this the ***value*** of that action
- The value of an arbitrary action *a* is the expected reward given that *a* is selected:



- We denote the estimated value of action *a* at the time step *t* as $Q_t(a)$
    - We would like $Q_t(a)$ to be close to $q_*(a)$

Greedy actions

If you maintain estimates of the action values, then there is at least one action whose estimates value is greatest

- When you select one of these actions, you are *exploiting your current knowledge of the values of the actions*
- If you select one of the non-greedy actions, you are *exploring* (this enable you to improve your estimate of the non-greedy action's value)

Keep in mind that

Exploitation is the right thing to do to maximize the expected reward on the one step, but exploration may produce the greater total reward in the long run *(Tradeoff needed)*

Action-value Methods

Action-value Methods

Methods for estimating the values of actions and for using the estimates to make action selection decisions

- (Recall) the true value of an action is the mean reward when that action is selected
- One way to estimate this is by averaging the reward actually received:

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}} \qquad (2.1)$$

Predicate

1 if predicate is true
0 if it is not

Considerations on (2.1)

- If the denominator is *zero*: we define $Q_t(a)$ as some default value (e.g., 0)
- If the denominator goes to *infinity*: $Q_t(a)$ converges to $q_*(a)$ *(law of large numbers)*
- We call this the *sample-average* method for estimating action values because estimate is an average of the sample of relevant rewards *(just one way, not necessarily the best one!!)*

Greedy action

- We write the *greedy action* selection method as:

$$A_t \doteq \arg\max_a Q_t(a) \qquad (2.2)$$

- With ties broken arbitrarily
- Greedy action selection always exploits current knowledge to maximize immediate reward
- A simple alternative is to behave greedily most of the time, but every once in a while, with small probability ϵ, select randomly from all the actions with equal probability (ϵ***-greedy methods***)

ϵ-Greedy Methods

The advantage

- In the limit as the number of steps increases, every action will be sample an infinite number of times, thus ensuring that all the $Q_t(a)$ converge to their respective $q_*(a)$
- This set a balance between exploration and exploitation

Incremental Implementation

How to compute the averages efficiently

Part 1

- Concentrate on a single action
- Let R_i denote the reward received after the *i*th selection of *this action*
- Let Q_n denote the estimate of its action value after selected *n*-1 times

$$Q_n \doteq \frac{R_1 + R_2 + \ldots + R_{n-1}}{n-1}$$

- *If this is done*: memory and computation requirements would grow over time as more rewards are seen
- Each additional reward would require additional memory to store it

How to compute the averages efficiently

Part 2

- We can use ***incremental formulas*** for updating averages with *small, constant, computation required* to process each reward
- Given Q_n and the *n*th reward, R_n, the new average of all *n* rewards can be computed by

Step size $\alpha_t(a)$

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i = \ldots = Q_n + \frac{1}{n}[R_n - Q_n] \qquad (2.3)$$

- $R_n - Q_n$ is an *error* in the estimate. It is reduced by taking a step toward R_n (the *target)*
- The target is presumed to indicate a desirable direction in which to move

A Simple Bandit Algorithm

- Initialize, for *a*=1 to *k*:
    - $Q(a) \leftarrow 0$
    - $N(a) \leftarrow 0$
- Loop forever:
    - A $\leftarrow$ ϵ-greedy action
    - R $\leftarrow$ *bandit(A)*
    - $N(A) \leftarrow N(A) + 1$
    - $Q(a) \leftarrow Q(a) + \frac{1}{N(A)}[R - Q(A)]$

Tracking a Non-stationary Problem

Non-stationary problem

Bandits problems in which the reward probabilities change over time

- Give more weight to recent rewards than to long-last rewards
- One of the most popular ways of doing this is to use a *constant step-size parameter*:

$$Q_{n+1} \doteq Q_n + \alpha[R_n - Q_n] \qquad (2.5)$$

- Where $\alpha \in (0,1]$ is constant
- Q_{n+1} is a weighted average of past rewards and the initial estimate Q_1:

$$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i \qquad (2.6)$$

Considerations on (2.6)

- The weight $\alpha(1-\alpha)^{n-i}$ given to the reward R_i depends on how many rewards ago, *n-i*, it was observed
- The quantity $1-\alpha$ is less than 1
    - Thus, the weight given to R_i decreases as the number of intervening rewards increases
    - The weight decays exponentially according to the exponent on $1-\alpha$
        - This is sometimes called *exponentially recency-weighted average*

Variable step-size

Sometimes it is convenient to vary the step-size form step to step

- Let $\alpha_n(a)$ denote the step-size parameter used to process the reward receive after the *n*th selection of action *a*
- Not all choices of the sequence $\{\alpha_n(a)\}$ guarantee convergence
- A result in stochastic approximation theory gives us the conditions required to assure convergence with probability 1:

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \qquad \text{and} \qquad \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty \tag{2.7}$$

Considerations on (2.7)

- For $\alpha_n(a) = 1/n$ both the conditions are met
- For $\alpha_n(a) = \alpha$ the second condition is not met
    - Indicating that the estimates never completely converge but continue to vary in response to the most recently received rewards *(this is actually desirable in a non-stationary environment)*
- Sequences of step-size parameters that meet the conditions (2.7) often converge very slowly or need considerable tuning in order to obtain a satisfactory convergence rate
    - *thus they are seldom used in applications and empirical research!*

Optimistic Initial Values

Initial action-value estimates bias

- All the methods we have discussed so far are dependent to some extent on the initial action-value estimates $Q_1(a)$
    - These methods are *biased* by their initial estimates
- For the sample-average methods → the bias disappears once all actions have been selected at least once
- For methods with constant α → the bias is permanent, though decreasing over time

Pro and con of the bias

- The *downside* is that the initial estimates become a set of parameters that must be picked by the user
- The *upside* is that they provide an easy way to supply some prior knowledge about what level of rewards can be expected
    - Initial action values can also be used as simple way to encourage exploration
        - e.g., an initial estimate of +5 (wildly optimistic)
    - This optimism encourages action-value methods to explore

Whichever actions are initially selected, the reward is less than the starting estimates; being disappointed, the learner switches to other actions

Optimistic Initial Values

This technique for encouraging exploration

- Initially, optimistic method performs worse because it explores more, but eventually it performs better because its exploration decreases with time
- This trick can be quite effective on stationary problems
- It is not well suited to non-stationary problems because its drive for exploration is inherently temporary
    - If the task changes, creating a renewed need for exploration, this method cannot help

Upper-Confidence-Bound Action Selection

Non-greedy selection (variant)

- It would be better to select among the non-greedy actions according to their potential for actually being optimal, taking into account both how close their estimates are to being maximal and the uncertainties in those estimates
- One way is to select actions according to

$$A_t \doteq \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right] \qquad (2.10)$$

Controls the degree of exploration (c>0)

Denotes the number of times that action *a* has been selected prior to time *t*

- If $N_t(a) = 0$, then *a* is considered to be a maximizing action

Upper-Confidence-Bound (UCB)

Idea

- The square-root term is a measure of the uncertainty or variance in the estimate of *a*'s value
- The quantity max'ed over is thus a sort of upper bound on the possible true value of action *a*, with *c* determining the confidence level
- Each time *a* is selected the uncertainty is presumably reduced:
    - $N_t(a)$ increments
- Each time action other than *a* is selected, *t* increases but $N_t(a)$ does not
    - The uncertainty estimate increases
    - The use of the natural logarithm means that the increase get smaller over time, but are unbounded
- All actions will eventually be selected, but actions with lower value estimates, or that have already been selected frequently, will be selected with decreasing frequency over time

Pros and Cons (UCB)

- UCB often performs well
- It is difficult to extend beyond bandits to the more general RL settings
- Difficult in dealing with non-stationary problems
- Difficult in dealing with large state spaces

In more advanced settings the idea of UCB action selection is usually not practical

Gradient Bandit Algorithms

Probability of taking an action

- Consider learning a numerical *preference* for each action *a*, $H_t(a) \in \mathbb{R}$
  - The larger the preference, the more often that action is taken
  - But, preference has no interpretation in terms of reward
  - Only the relative preference of one action over another is important
- The action probabilities are determine according to a *soft-max distribution:*

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} = \pi_t(a) \qquad (2.11)$$

Probability of taking action *a* at time *t*

- Initially all action preferences are the same (e.g., $H_1(a) = 0$, for all *a*) so that all actions have an equal probability of being selected

Stochastic gradient ascent idea

- On each step, after selecting action A_t and receiving the reward R_t, the action preferences are updated by:

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \qquad \text{and}$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t \tag{2.12}$$

- $\alpha > 0$ is a step-size parameter
- $\overline{R}_t \in \mathbb{R}$ is the average of the rewards up to but not including time *t* (with $\overline{R}_1 \doteq R_1$)
    - $\overline{R}_t$ serves as a *baseline* with which the reward is compared
        - If the reward is higher than the baseline, then the probability of taking A_t in the future is increased
        - If the reward is below baseline, then the probability decreased
        - The non-selected actions move in the opposite directions

Associative Search (Contextual Bandits)

Associative Search

- So far, we have considered only non-associative tasks:
    - Tasks in which there is no need to associate different actions with different situations
- In a general reinforcement learning task there is more than one situation, and the goal is to learn a *policy*:
        - a mapping from situations to the actions that are best in those situations.

Associative Search

An example

- Suppose there are several different *k*-armed bandit tasks, and that on each step you confront one of these chosen at random
    - Thus, the bandit task changes randomly from step to step
- If the probabilities with which each task is selected for you do not change over time, this would appear as a single stationary *k*-armed bandit task
- When a bandit task is selected for you, you are given some distinctive clue about its identity (but not its action values!)
- Now you can learn a policy associating each task with the best action to take when facing that task

Associative search

Final considerations

- It involves both trial-and-error learning to *search* for best actions, and *association* of these actions with the situations in which they are best
- Called in literature *contextual bandits*
    - They are intermediate between the *k*-armed bandit problem and the full reinforcement learning problem since
        - they involve learning a policy
        - but each action affects only the immediate reward

Bibliography:

Reinforcement Learning An Introduction (Second Edition), R. S. Sutton & A. G. Barto