Table of Contents

Base Notation

k: number of bandit arms, or size of the bandit environment
t: discrete time step
a: denotes an action. $a \in k$
A_t: denotes the action taken at time point t
$R_{t,a}$: realized (ex–post) reward at time point t, given that $A_t = a$
$Q_t(a)$: reward estimate for action a at time point t
$q_*(a) = E(R_t \mid A_t = a)$: expected reward of action a at time point t
$N_t(a)$: number of times action a has been selected until time point t

Base Terminology

Agent: The instance/robot to solve the problem.
Environment: The problem to solve.
Exploitation (greedy action): Choosing the action with the highest estimated value $\underset{a}{\operatorname{argmax}} Q_t(a)$, for the purpose of maximising the (expected) total reward.
Exploration: Choosing non–greedy actions for optimising/updating the estimates $Q_t(a)$ for each a which is *not* $\underset{a}{\operatorname{argmax}} Q_t(a)$. Exploration has the purpose of calculating better estimates for non–greedy actions, with the aim to minimize $|q_*(a)| - |Q_t(a)|$.
Valuation methods: Valuation methods define how the reward estimates $Q_t(a)$ are calculated.
`valuation_mehtod = average`: Considering a single action: $Q_{n+1} = \frac{R_1 + R_2 + \ldots + R_n}{n}$, where n denotes the n–th selection of the respective action. This can be re–written as $Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$
`valuation_mehtod = weighted`: Considering a single action: $Q_{n+1} = Q_n + \alpha[R_n - Q_n]$

Environments

At the moment, the repository contains 3 bandit environments.

1. **Stochastic Bandit Environment `stochastic_bandit`**
   The environment `stochastic_bandit` is a stationary bandit problem, where the reward of each arm is a random variable coming from either a gaussian normal or a log–normal distribution.
   When initialising the environment, the expected rewards for each action $q_{(a)}$ * from a normal distribution, where μ and σ are argument inputs of the environment class (`stochastic_moments = [0,1]` per default.)
   Arguments
   `size`: the number of bandit arms k
   `reward_distribution`: defines the theoretical distribution of rewards. Options: `["gaussian", "lognormal"]`.
   `stochastic_moments`: Defines the first and second moment of the normal

distribution the expected rewards $q_*(1), \ldots, q_*(k)$ are sampled from ($\mu = 0$ and $\sigma = 1$ by default).
Explanation, 5–armed bandit case: If the environment is initialised with `stochastic_moments = [1,2]`, the expected returns for each of the 5 reward distributions $q_*(1), \ldots, q_*(5)$ are random variables from a normal distribution with $\mu = 1$ and $\sigma = 2$ ($N(1, 2)$). After the initialisation, let $q_*(5) = 0.2$. Subsequently, $R_{t,5} \sim N(0.2, 1)$ in case of normally distributed rewards ($\sim logN(0.2, 1)$ applies if the user chose the log–normal distribution).

2. **Custom Stochastic Bandit Environment `stochastic_bandit_custom`**
The environment `stochastic_bandit_custom` is a stationary bandit problem. This environments grants more flexibility for defining the underlying problem, compared to `stochastic_bandit`.
Through the input argument `mu` and `sigma`, the user can customise the theoretical distributions which are used for reward sampling.
For `stochastic_bandit_custom`, the reward distribution are completely customisable, while `stochastic_bandit` *samples* the parameters of the rewards distributions during initialisation (i.e., the distribution parameters are stochastic itself).
Arguments
`size`: the number of bandit arms k
`mu`: an array of central values for the reward distributions, which are similar to the expected returns for each action $q_*(a)$.
`sigma`: an array of second stochastic moments for each reward distribution.
`reward_distribution`: defines the theoretical distribution of rewards. Options: `["gaussian", "lognormal"]`.
`stochastic_moments`: Defines the first and second moment of the normal distribution the expected rewards $q_*(1), \ldots, q_*(k)$ are sampled from ($\mu = 0$ and $\sigma = 1$ by default).
Explanation, 2–armed bandit case: If the environment is initialised with `mu = [0,0.5]` and `sigma = [2,1]`, the rewards for both actions follow $R_{t,1} \sim N(0, 2)$ and $R_{t,2} \sim N(0.5, 1)$ if `reward_distribution = "gaussian"`.

3. **Adversarial (non–stationary) Bandit `adversarial_bandit`**
The environment `adversarial_bandit` is a non–stationary version of the `stochastic_bandit` environment.
Arguments
`size`: the number of bandit arms k
`reward_distribution`: defines the theoretical distribution of rewards. Options: `["gaussian", "lognormal"]`.
`stochastic_moments`: Defines the first and second moment of a normal distribution which is used for sampling *the stationary component* of the expected rewards $q_*(1), \ldots, q_*(k)$ ($\mu = 0$ and $\sigma = 1$ by default).
Explanation, 3–armed bandit case: Given is an initiation with `stochastic_moments = [1,2]`. The expected rewards are then defined as $(R_t | A_t = a) (1, 2) $ for every action $a \in [1, .., k]$, implying that the expected reward $q_*(a)$ is constant over time and not dependent on t(note that $k=3 for this example). Let $q_*(3) = 1.2$. We can write $R_{t,3} \sim N(1.2, 0.1t)$, indicating that rewards will fluctuate stronger with increasing t.

Agents

There are currently 3 different agents available in the repository. Generally, agents differ on how to select the next action a.

1. **Random agent `RandomAgent`**
   The `RandomAgent` class selects action a randomly ($a \sim U(1,k)$).
   Arguments
   `env`: a bandit environment.
   `q_initialisation`: an array of length k with initial q–estimates. `None` indicates initiation with zeros.
   `valualtion_method`: Defines how q–estimates are calculated. Must be either `av–erage` or `weighted`.
   `alpha`: Scale parameter, weights importance of last reward observation when calculating q–estimates. Required if `valuation_method = weighted`.
   `name`: Custom name for your agent (optional).

2. **Epsilon–greedy agent `EpsilonGreedyAgent`**
   This agent selects a greedy action with probability a given probability, which is defined by a constant (ε). The agent uses exploitation with a probability of $1-\varepsilon$ and subsequently choose a (random) non–greedy action with probability ε. The probability for exploitation can be written as

   $$P\left(A_t = \underset{a}{\operatorname{argmax}} Q_t(a)\right) = 1-\varepsilon$$

   .
   Arguments
   `env`: a bandit environment.
   `epsilon`: Constant, input for ε (see above)
   `q_initialisation`: an array of length k with initial q–estimates. `None` indicates initiation with zeros.
   `valualtion_method`: Defines how q–estimates are calculated. Must be either `av–erage` or `weighted`.
   `alpha`: Scale parameter, weights importance of last reward observation when calculating q–estimates. Required if `valuation_method = weighted`.
   `name`: Custom name for your agent (optional).

3. **Upper–Confidence–Bound Agent `UCBAgent`**
   The UCB–Agent takes into account how "confident" it is in his Q–estimates by taking into account how ofen the respective action has been selected. The selection method can be written as:

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

   , where $N_t(a)$ is the number of times action a has been selected until time point $t $.
   Thus, the term $\sqrt{\frac{\ln t}{N_t(a)}}$ (the uncertainty term) puts the number of times an action a has been selected in relation to the total number of steps t, resulting in an addition to its estimate $Q_t(a)$ for actions which are less frequently selected. The constant c is a fixed parameters for scaling the uncertainty term.
   Arguments
   `env`: a bandit environment.
   `c`: Constant, input for c (see above)
   `q_initialisation`: an array of length k with initial q–estimates. `None` indicates initiation with zeros.
   `valualtion_method`: Defines how q–estimates are calculated. Must be either `av–erage` or `weighted`.

`alpha`: Scale parameter, weights importance of last reward observation when calculating q–estimates. Required if `valuation_method = weighted`.
`name`: Custom name for your agent (optional).