K-Armed Bandits

Release 1.0

Jonathan Kirkpatrick

Jun 11, 2024

CONTENTS:

1 k_armed_bandits module 3

2 non_stationary module 7

3 Indices and tables 9

Python Module Index 11

master_doc = ‘index’

CHAPTER

ONE

K_ARMED_BANDITS MODULE

class k_armed_bandits.**BanditsGame**(*K*, *T*, *seed=None*, *non_stationary_type='stationary'*, *verbose=True*)

Bases: `object`

Creates an instance of the bandits testbed.

Generates K NormalBandit objects to populate the BanditsGame

… .. attribute:: K

the number of NormalBandits to populate

type
int

verbose

optional parameter used to provide greater detail to stdout

Type
boolean

run_greedy(T):

runs an instance of the BanditsGame using the greedy algorithm

run_epsilon_greedy(T, epsilon):

runs an instance of the BanditsGame using the epsilon-greedy algorithm

run_optimistic_initial_values(T, initial_value):

runs an instance of the BanditsGame using the optimistic-greedy algorithm

run_gradient_bandit(T, alpha):

runs an instance of the BanditsGame using the gradient-bandit algorithm

run_epsilon_greedy(*T*, *epsilon*)

Runs an instance of the BanditsGame using the epsilon-greedy algorithm.

Parameters

- **T** (*int*) – The number of steps to run the algorithm.
- **epsilon** (*float*) – The probability of exploring a random action.

Returns

- *numpy.ndarray* – An array of rewards obtained at each step.
- *numpy.ndarray* – An array indicating whether the optimal action was chosen at each step.

run_gradient_bandit(*T*, *alpha*)

Runs an instance of the BanditsGame using the gradient-bandit algorithm.

Parameters

- **T** (*int*) – The number of steps to run the algorithm.

- **alpha** (*float*) – The learning rate for updating preferences.

Returns

- *numpy.ndarray* – An array of rewards obtained at each step.
- *numpy.ndarray* – An array indicating whether the optimal action was chosen at each step.

run_greedy(*T*)

Runs an instance of the BanditsGame using the greedy algorithm.

Parameters

T (*int*) – The number of steps to run the algorithm.

Returns

- *numpy.ndarray* – An array of rewards obtained at each step.
- *numpy.ndarray* – An array indicating whether the optimal action was chosen at each step.

run_optimistic_initial_values(*T*, *initial_value*)

Runs an instance of the BanditsGame using the optimistic-greedy algorithm.

Parameters

- **T** (*int*) – The number of steps to run the algorithm.
- **initial_value** (*float*) – The initial optimistic value for all actions.

Returns

- *numpy.ndarray* – An array of rewards obtained at each step.
- *numpy.ndarray* – An array indicating whether the optimal action was chosen at each step.

class k_armed_bandits.**NormalBandit**(*means*, *variance=1*, *verbose=True*)

Bases: object

A class to represent an individual bandit with a normal reward distribution.

...

mean

the average reward to be expected from this bandit

Type
float

variance

the variance associated with the normal reward distribution

Type
float

verbose

optional parameter used to provide greater detail to stdout

Type
boolean

pull():

outputs a randomly generated reward based on the bandit's distribution

pull(*t*)

Operates this instance of NormalBandit.

A random reward is generated from its defined normal reward distribution, which may vary depending on the time step t.

Parameters
t (*int*) – The current time step.

Returns
A float value representing the reward.

Return type
float

k_armed_bandits.**plot_results**(*avg_rewards_greedy*, *avg_rewards_epsilon_greedy*, *avg_rewards_optimistic*, *avg_rewards_gradient*, *opt_action_props_greedy*, *opt_action_props_epsilon_greedy*, *opt_action_props_optimistic*, *opt_action_props_gradient*, *best_epsilon*, *best_alpha*, *avg_rewards_per_trial_greedy*, *avg_rewards_per_trial_optimistic*, *avg_rewards_per_trial_epsilon_greedy*, *avg_rewards_per_trial_gradient*)

Plots the results of different algorithms.

Parameters

- **avg_rewards_greedy** (*numpy.ndarray*) – Average rewards obtained using the greedy algorithm.
- **avg_rewards_epsilon_greedy** (*numpy.ndarray*) – Average rewards obtained using the epsilon-greedy algorithm.
- **avg_rewards_optimistic** (*numpy.ndarray*) – Average rewards obtained using the optimistic initial values algorithm.
- **avg_rewards_gradient** (*numpy.ndarray*) – Average rewards obtained using the gradient bandit algorithm.
- **opt_action_props_greedy** (*numpy.ndarray*) – Proportion of times the optimal action was chosen using the greedy algorithm.
- **opt_action_props_epsilon_greedy** (*numpy.ndarray*) – Proportion of times the optimal action was chosen using the epsilon-greedy algorithm.
- **opt_action_props_optimistic** (*numpy.ndarray*) – Proportion of times the optimal action was chosen using the optimistic initial values algorithm.
- **opt_action_props_gradient** (*numpy.ndarray*) – Proportion of times the optimal action was chosen using the gradient bandit algorithm.
- **best_epsilon** (*float*) – The best epsilon value found during parameter tuning.
- **best_alpha** (*float*) – The best alpha value found during parameter tuning.

k_armed_bandits.**run_simulation**(*K*, *T*, *n_problems*, *algorithm*, *seed=None*, *non_stationary_type='stationary'*, ***kwargs*)

Runs multiple instances of the BanditsGame and computes average rewards and optimal action proportions.

Parameters

- **K** (*int*) – The number of bandits.
- **T** (*int*) – The number of steps to run each game.
- **n_problems** (*int*) – The number of independent problems to simulate.
- **algorithm** (*str*) – The algorithm to use ('greedy', 'epsilon_greedy', 'optimistic_initial_values', 'gradient_bandit').
- **seed** (*int, optional*) – Random seed for reproducibility (default is None).

- **non_stationary_type** (*str, optional*) – Type of non-stationary behavior ('stationary' or specified in non_stationary_types).
- ****kwargs** (*dict*) – Additional parameters specific to the algorithm.

Returns

- *numpy.ndarray* – Average rewards across all problems at each step.
- *numpy.ndarray* – Proportion of times the optimal action was chosen across all problems at each step.

CHAPTER

TWO

NON_STATIONARY MODULE

non_stationary.**abrupt_change**(*bandits*)

Permutes the means corresponding to each bandit.

At each time step, with probability 0.005, permutes the means corresponding to each of the reward distributions.

Parameters

bandits (*list of* NormalBandit) – List of bandits whose means need to be permuted.

non_stationary.**drift_change**(*previous_mean*, *initial_mean*)

Applies a small random drift to the mean.

Parameters

- **previous_mean** (*float*) – The mean at the previous time step.
- **initial_mean** (*float*) – The initial mean of the bandit.

Returns

The updated mean after applying the drift.

Return type

float

non_stationary.**generate_means**(*initial_mean*, *T*, *non_stationary_type*)

Generates a sequence of means for the given non-stationary type.

Parameters

- **initial_mean** (*float*) – The initial mean of the bandit.
- **T** (*int*) – The number of time steps.
- **non_stationary_type** (*str*) – The type of non-stationary behavior ('drift', 'mean_reverting', 'abrupt').

Returns

An array of means for each time step.

Return type

numpy.ndarray

non_stationary.**mean_reverting_change**(*previous_mean*, *initial_mean*)

Applies a mean-reverting change to the mean.

Parameters

- **previous_mean** (*float*) – The mean at the previous time step.
- **initial_mean** (*float*) – The initial mean of the bandit.

Returns

The updated mean after applying the mean-reverting change.

Return type
float

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

k

k_armed_bandits, 3

n

non_stationary, 7