striatum Documentation

Release 0.0.1

Y.-Y. Yang, Y.-A. Lin

September 14, 2016

Contents

1 API Reference **3**
- 1.1 striatum.bandit package . . . 3
- 1.2 striatum.storage package . . . 17

2 Simulations of bandits **21**
- 2.1 Exp3 . . . 21
- 2.2 Exp4.P . . . 22
- 2.3 Thompson sampling . . . 22
- 2.4 LinUCB . . . 22
- 2.5 UCB1 . . . 22

Bibliography **27**

Python Module Index **29**

Contents:

CHAPTER 1

API Reference

1.1 striatum.bandit package

1.1.1 Submodules

1.1.2 striatum.bandit.exp3 module

Exp3: Exponential-weight algorithm for Exploration and Exploitation This module contains a class that implements EXP3, a bandit algorithm that randomly choose an action according to a learned probability distribution.

class striatum.bandit.exp3.**Exp3**(*actions*, *historystorage*, *modelstorage*, *gamma*)

Bases: striatum.bandit.bandit.BaseBandit

Exp3 algorithm.

Parameters **actions** : list of Action objects

List of actions to be chosen from.

historystorage: a HistoryStorage object

The place where we store the histories of contexts and rewards.

modelstorage: a ModelStorage object

The place where we store the model parameters.

gamma: float, 0 < gamma <= 1

The parameter used to control the minimum chosen probability for each action.

References

[R1]

Attributes

exp3_	('exp3' object instance) The contextual bandit algorithm instances

Methods

add_action(actions)	Add new actions (if needed).
calculate_avg_reward()	Calculate average reward with respect to time.
calculate_cum_reward()	Calculate cumulative reward with respect to time.
exp3()	The generator which implements the main part of Exp3.
get_action(context[, n_actions])	Return the action to perform
get_action_with_id(action_id)	
plot_avg_regret()	Plot average regret with respect to time.
plot_avg_reward()	Plot average reward with respect to time.
reward(history_id, rewards)	Reward the previous action with reward.

add_action(*actions*)
Add new actions (if needed).

Parameters **actions** : iterable

A list of Action objects for recommendation

exp3()
The generator which implements the main part of Exp3.

get_action(*context*, *n_actions=1*)
Return the action to perform

Parameters **context** : {array-like, None}

The context of current state, None if no context available.

n_actions: int

Number of actions wanted to recommend users.

Returns **history_id** : int

The history id of the action.

action_recommendation : list of dictionaries

In each dictionary, it will contains {Action object, estimated_reward, uncertainty}

reward(*history_id*, *rewards*)
Reward the previous action with reward.

Parameters **history_id** : int

The history id of the action to reward.

rewards : dictionary

The dictionary {action_id, reward}, where reward is a float.

1.1.3 striatum.bandit.exp4p module

EXP4.P: An extention to exponential-weight algorithm for exploration and exploitation. This module contains a class that implements EXP4.P, a contextual bandit algorithm with expert advice.

class striatum.bandit.exp4p.**Exp4P**(*actions*, *historystorage*, *modelstorage*, *delta=0.1*, *p_min=None*, *max_rounds=10000*)
Bases: striatum.bandit.bandit.BaseBandit

Exp4.P with pre-trained supervised learning algorithm.

Parameters **actions** : list of Action objects

List of actions to be chosen from.

historystorage: a HistoryStorage object

The place where we store the histories of contexts and rewards.

modelstorage: a ModelStorage object

The place where we store the model parameters.

delta: float, 0 < delta <= 1

With probability 1 - delta, LinThompSamp satisfies the theoretical regret bound.

p_min: float, 0 < p_min < 1/k

The minimum probability to choose each action.

References

[R2]

Attributes

exp4p_	('exp4p' object instance) The contextual bandit algorithm instances

Methods

add_action(actions)	Add new actions (if needed).
calculate_avg_reward()	Calculate average reward with respect to time.
calculate_cum_reward()	Calculate cumulative reward with respect to time.
get_action([context, n_actions])	Return the action to perform
get_action_with_id(action_id)	
plot_avg_regret()	Plot average regret with respect to time.
plot_avg_reward()	Plot average reward with respect to time.
reward(history_id, rewards)	Reward the previous action with reward.

add_action(*actions*)

Add new actions (if needed).

Parameters **actions** : iterable

A list of Action objects for recommendation

get_action(*context=None*, *n_actions=1*)

Return the action to perform

Parameters **context** : dictionary

Contexts {expert_id: {action_id: expert_prediction}} of different actions.

n_actions: int

Number of actions wanted to recommend users.

Returns **history_id** : int

The history id of the action.

action_recommendation : list of dictionaries

In each dictionary, it will contains {Action object, estimated_reward, uncertainty}.

reward(*history_id*, *rewards*)

Reward the previous action with reward.

Parameters **history_id** : int

The history id of the action to reward.

rewards : dictionary

The dictionary {action_id, reward}, where reward is a float.

1.1.4 striatum.bandit.linthompsamp module

Thompson Sampling with Linear Payoff In This module contains a class that implements Thompson Sampling with Linear Payoff. Thompson Sampling with linear payoff is a contexutal multi-armed bandit algorithm which assume the underlying relationship between rewards and contexts is linear. The sampling method is used to balance the exploration and exploitation. Please check the reference for more details.

class striatum.bandit.linthompsamp.**LinThompSamp**(*actions*, *historystorage*, *modelstorage*, *context_dimension*, *delta=0.5*, *R=0.5*, *epsilon=0.1*, *random_state=None*)

Bases: striatum.bandit.bandit.BaseBandit

Thompson sampling with linear payoff.

Parameters **actions** : list of Action objects

List of actions to be chosen from.

historystorage: a HistoryStorage object

The place where we store the histories of contexts and rewards.

modelstorage: a ModelStorage object

The place where we store the model parameters.

delta: float, 0 < delta < 1

With probability 1 - delta, LinThompSamp satisfies the theoretical regret bound.

R: float, R >= 0

Assume that the residual $ri(t) - bi(t)^T \hat{\mu}$ is R-sub-gaussian. In this case, R^2 represents the variance for residuals of the linear model $bi(t)^T$.

epsilon: float, 0 < epsilon < 1

A parameter used by the Thompson Sampling algorithm. If the total trials T is known, we can choose epsilon = 1/ln(T).

References

[R3]

Attributes

linthomp_	('linthomp' object instance) The contextual bandit algorithm instances

Methods

add_action(actions)	Add new actions (if needed).
calculate_avg_reward()	Calculate average reward with respect to time.
calculate_cum_reward()	Calculate cumulative reward with respect to time.
get_action(context[, n_actions])	Return the action to perform
get_action_with_id(action_id)	
plot_avg_regret()	Plot average regret with respect to time.
plot_avg_reward()	Plot average reward with respect to time.
reward(history_id, rewards)	Reward the previous action with reward.

add_action(*actions*)
Add new actions (if needed).

Parameters **actions** : iterable

A list of Action oBjects for recommendation

get_action(*context*, *n_actions=1*)
Return the action to perform

Parameters **context** : dictionary

Contexts {action_id: context} of different actions.

n_actions: int

Number of actions wanted to recommend users.

Returns **history_id** : int

The history id of the action.

action_recommendation : list of dictionaries

In each dictionary, it contains {Action object, estimated_reward, uncertainty}.

reward(*history_id*, *rewards*)
Reward the previous action with reward.

Parameters **history_id** : int

The history id of the action to reward.

rewards : dictionary

The dictionary {action_id, reward}, where reward is a float.

1.1.5 striatum.bandit.linucb module

LinUCB with Disjoint Linear Models

This module contains a class that implements LinUCB with disjoint linear model, a contextual bandit algorithm assuming the reward function is a linear function of the context.

class striatum.bandit.linucb.**LinUCB**(*actions*, *historystorage*, *modelstorage*, *alpha*, *context_dimension=1*)

Bases: striatum.bandit.bandit.BaseBandit

LinUCB with Disjoint Linear Models

Parameters **actions** : list of Action objects

List of actions to be chosen from.

historystorage: a HistoryStorage object

The place where we store the histories of contexts and rewards.

modelstorage: a ModelStorage object

The place where we store the model parameters.

alpha: float

The constant determines the width of the upper confidence bound.

context_dimension: int

The dimension of the context.

References

[R4]

Attributes

linucb_	('linucb' object instance) The contextual bandit algorithm instances.

Methods

add_action(actions)	Add new actions (if needed).
calculate_avg_reward()	Calculate average reward with respect to time.
calculate_cum_reward()	Calculate cumulative reward with respect to time.
get_action(context[, n_actions])	Return the action to perform
get_action_with_id(action_id)	
plot_avg_regret()	Plot average regret with respect to time.
plot_avg_reward()	Plot average reward with respect to time.
reward(history_id, rewards)	Reward the previous action with reward.

add_action(*actions*)

Add new actions (if needed).

Parameters **actions** : iterable

A list of Action objects for recommendation

get_action(*context*, *n_actions=1*)

Return the action to perform

Parameters **context** : dict

Contexts {action_id: context} of different actions.

n_actions: int

Number of actions wanted to recommend users.

Returns history_id : int

The history id of the action.

action_recommendation : list of dict

Each dict contains {Action object, estimated_reward, uncertainty}

reward(*history_id*, *rewards*)

Reward the previous action with reward.

Parameters history_id : int

The history id of the action to reward.

rewards : dictionary

The dictionary {action_id, reward}, where reward is a float.

1.1.6 striatum.bandit.ucb1 module

Upper Confidence Bound 1 This module contains a class that implements UCB1 algorithm, a famous multi-armed bandit algorithm without context.

class striatum.bandit.ucb1.**UCB1**(*actions*, *historystorage*, *modelstorage*)

Bases: striatum.bandit.bandit.BaseBandit

Upper Confidence Bound 1

Parameters actions : {array-like, None}

Actions (arms) for recommendation

historystorage: a :py:mod:'striatum.storage.HistoryStorage' object

The object where we store the histories of contexts and rewards.

modelstorage: a :py:mod:'straitum.storage.ModelStorage' object

The object where we store the model parameters.

References

[R5]

Attributes

ucb1_	('ucb1' object instance) The multi-armed bandit algorithm instances.

Methods

add_action(actions)	Add new actions (if needed).
calculate_avg_reward()	Calculate average reward with respect to time.

Continued on next page

Table 1.5 – continued from previous page

calculate_cum_reward()	Calculate cumulative reward with respect to time.
get_action(context[, n_actions])	Return the action to perform
get_action_with_id(action_id)	
plot_avg_regret()	Plot average regret with respect to time.
plot_avg_reward()	Plot average reward with respect to time.
reward(history_id, rewards)	Reward the previous action with reward.
ucb1()	

add_action(*actions*)
Add new actions (if needed).

Parameters **actions** : iterable

A list of Action objects for recommendation

get_action(*context*, *n_actions=1*)
Return the action to perform

Parameters **context** : {array-like, None}

The context of current state, None if no context available.

n_actions: int

Number of actions wanted to recommend users.

Returns **history_id** : int

The history id of the action.

action : list of dictionaries

In each dictionary, it will contains {Action object, estimated_reward, uncertainty}

reward(*history_id*, *rewards*)
Reward the previous action with reward.

Parameters **history_id** : int

The history id of the action to reward.

rewards : dictionary

The dictionary {action_id, reward}, where reward is a float.

ucb1()

1.1.7 Module contents

Bandit algorithm classes

class striatum.bandit.**Exp3**(*actions*, *historystorage*, *modelstorage*, *gamma*)
Bases: striatum.bandit.bandit.BaseBandit

Exp3 algorithm.

Parameters **actions** : list of Action objects

List of actions to be chosen from.

historystorage: a HistoryStorage object

The place where we store the histories of contexts and rewards.

modelstorage: a ModelStorage object

The place where we store the model parameters.

gamma: float, 0 < gamma <= 1

The parameter used to control the minimum chosen probability for each action.

References

[R6]

Attributes

exp3_	('exp3' object instance) The contextual bandit algorithm instances

Methods

add_action(actions)	Add new actions (if needed).
calculate_avg_reward()	Calculate average reward with respect to time.
calculate_cum_reward()	Calculate cumulative reward with respect to time.
exp3()	The generator which implements the main part of Exp3.
get_action(context[, n_actions])	Return the action to perform
get_action_with_id(action_id)	
plot_avg_regret()	Plot average regret with respect to time.
plot_avg_reward()	Plot average reward with respect to time.
reward(history_id, rewards)	Reward the previous action with reward.

add_action(*actions*)

Add new actions (if needed).

Parameters **actions** : iterable

A list of Action objects for recommendation

exp3()

The generator which implements the main part of Exp3.

get_action(*context*, *n_actions=1*)

Return the action to perform

Parameters **context** : {array-like, None}

The context of current state, None if no context available.

n_actions: int

Number of actions wanted to recommend users.

Returns **history_id** : int

The history id of the action.

action_recommendation : list of dictionaries

In each dictionary, it will contains {Action object, estimated_reward, uncertainty}

reward(*history_id*, *rewards*)
Reward the previous action with reward.

Parameters **history_id** : int

The history id of the action to reward.

rewards : dictionary

The dictionary {action_id, reward}, where reward is a float.

class striatum.bandit.**Exp4P**(*actions*, *historystorage*, *modelstorage*, *delta=0.1*, *p_min=None*, *max_rounds=10000*)
Bases: striatum.bandit.bandit.BaseBandit

Exp4.P with pre-trained supervised learning algorithm.

Parameters **actions** : list of Action objects

List of actions to be chosen from.

historystorage: a HistoryStorage object

The place where we store the histories of contexts and rewards.

modelstorage: a ModelStorage object

The place where we store the model parameters.

delta: float, 0 < delta <= 1

With probability 1 - delta, LinThompSamp satisfies the theoretical regret bound.

p_min: float, 0 < p_min < 1/k

The minimum probability to choose each action.

References

[R7]

Attributes

exp4p_	('exp4p' object instance) The contextual bandit algorithm instances

Methods

add_action(actions)	Add new actions (if needed).
calculate_avg_reward()	Calculate average reward with respect to time.
calculate_cum_reward()	Calculate cumulative reward with respect to time.
get_action([context, n_actions])	Return the action to perform
get_action_with_id(action_id)	
plot_avg_regret()	Plot average regret with respect to time.
plot_avg_reward()	Plot average reward with respect to time.
reward(history_id, rewards)	Reward the previous action with reward.

add_action(*actions*)
Add new actions (if needed).

Parameters **actions** : iterable

A list of Action objects for recommendation

get_action(*context=None*, *n_actions=1*)

Return the action to perform

Parameters **context** : dictionary

Contexts {expert_id: {action_id: expert_prediction}} of different actions.

n_actions: int

Number of actions wanted to recommend users.

Returns **history_id** : int

The history id of the action.

action_recommendation : list of dictionaries

In each dictionary, it will contains {Action object, estimated_reward, uncertainty}.

reward(*history_id*, *rewards*)

Reward the previous action with reward.

Parameters **history_id** : int

The history id of the action to reward.

rewards : dictionary

The dictionary {action_id, reward}, where reward is a float.

class striatum.bandit.**LinThompSamp**(*actions*, *historystorage*, *modelstorage*, *context_dimension*, *delta=0.5*, *R=0.5*, *epsilon=0.1*, *random_state=None*)

Bases: striatum.bandit.bandit.BaseBandit

Thompson sampling with linear payoff.

Parameters **actions** : list of Action objects

List of actions to be chosen from.

historystorage: a HistoryStorage object

The place where we store the histories of contexts and rewards.

modelstorage: a ModelStorage object

The place where we store the model parameters.

delta: float, 0 < delta < 1

With probability 1 - delta, LinThompSamp satisfies the theoretical regret bound.

R: float, R >= 0

Assume that the residual $ri(t) - bi(t)^T \hat{\mu}$ is R-sub-gaussian. In this case, R^2 represents the variance for residuals of the linear model $bi(t)^T$.

epsilon: float, 0 < epsilon < 1

A parameter used by the Thompson Sampling algorithm. If the total trials T is known, we can choose epsilon = 1/ln(T).

References

[R8]

Attributes

linthomp_	('linthomp' object instance) The contextual bandit algorithm instances

Methods

add_action(actions)	Add new actions (if needed).
calculate_avg_reward()	Calculate average reward with respect to time.
calculate_cum_reward()	Calculate cumulative reward with respect to time.
get_action(context[, n_actions])	Return the action to perform
get_action_with_id(action_id)	
plot_avg_regret()	Plot average regret with respect to time.
plot_avg_reward()	Plot average reward with respect to time.
reward(history_id, rewards)	Reward the previous action with reward.

add_action(*actions*)
Add new actions (if needed).

Parameters **actions** : iterable

A list of Action oBjects for recommendation

get_action(*context*, *n_actions=1*)
Return the action to perform

Parameters **context** : dictionary

Contexts {action_id: context} of different actions.

n_actions: int

Number of actions wanted to recommend users.

Returns **history_id** : int

The history id of the action.

action_recommendation : list of dictionaries

In each dictionary, it contains {Action object, estimated_reward, uncertainty}.

reward(*history_id*, *rewards*)
Reward the previous action with reward.

Parameters **history_id** : int

The history id of the action to reward.

rewards : dictionary

The dictionary {action_id, reward}, where reward is a float.

class striatum.bandit.**LinUCB**(*actions*, *historystorage*, *modelstorage*, *alpha*, *context_dimension=1*)
Bases: striatum.bandit.bandit.BaseBandit

LinUCB with Disjoint Linear Models

Parameters **actions** : list of Action objects

List of actions to be chosen from.

historystorage: a HistoryStorage object

The place where we store the histories of contexts and rewards.

modelstorage: a ModelStorage object

The place where we store the model parameters.

alpha: float

The constant determines the width of the upper confidence bound.

context_dimension: int

The dimension of the context.

References

[R9]

Attributes

linucb_	('linucb' object instance) The contextual bandit algorithm instances.

Methods

add_action(actions)	Add new actions (if needed).
calculate_avg_reward()	Calculate average reward with respect to time.
calculate_cum_reward()	Calculate cumulative reward with respect to time.
get_action(context[, n_actions])	Return the action to perform
get_action_with_id(action_id)	
plot_avg_regret()	Plot average regret with respect to time.
plot_avg_reward()	Plot average reward with respect to time.
reward(history_id, rewards)	Reward the previous action with reward.

add_action(*actions*)

Add new actions (if needed).

Parameters **actions** : iterable

A list of Action objects for recommendation

get_action(*context*, *n_actions=1*)

Return the action to perform

Parameters **context** : dict

Contexts {action_id: context} of different actions.

n_actions: int

Number of actions wanted to recommend users.

Returns **history_id** : int

The history id of the action.

action_recommendation : list of dict

Each dict contains {Action object, estimated_reward, uncertainty}

reward(*history_id*, *rewards*)

Reward the previous action with reward.

Parameters **history_id** : int

The history id of the action to reward.

rewards : dictionary

The dictionary {action_id, reward}, where reward is a float.

class striatum.bandit.**UCB1**(*actions*, *historystorage*, *modelstorage*)

Bases: striatum.bandit.bandit.BaseBandit

Upper Confidence Bound 1

Parameters **actions** : {array-like, None}

Actions (arms) for recommendation

historystorage: a :py:mod:'striatum.storage.HistoryStorage' object

The object where we store the histories of contexts and rewards.

modelstorage: a :py:mod:'straitum.storage.ModelStorage' object

The object where we store the model parameters.

References

[R10]

Attributes

ucb1_	('ucb1' object instance) The multi-armed bandit algorithm instances.

Methods

add_action(actions)	Add new actions (if needed).
calculate_avg_reward()	Calculate average reward with respect to time.
calculate_cum_reward()	Calculate cumulative reward with respect to time.
get_action(context[, n_actions])	Return the action to perform
get_action_with_id(action_id)	
plot_avg_regret()	Plot average regret with respect to time.
plot_avg_reward()	Plot average reward with respect to time.
reward(history_id, rewards)	Reward the previous action with reward.
ucb1()	

add_action(*actions*)
Add new actions (if needed).

Parameters **actions** : iterable

A list of Action objects for recommendation

get_action(*context*, *n_actions=1*)
Return the action to perform

Parameters **context** : {array-like, None}

The context of current state, None if no context available.

n_actions: int

Number of actions wanted to recommend users.

Returns **history_id** : int

The history id of the action.

action : list of dictionaries

In each dictionary, it will contains {Action object, estimated_reward, uncertainty}

reward(*history_id*, *rewards*)
Reward the previous action with reward.

Parameters **history_id** : int

The history id of the action to reward.

rewards : dictionary

The dictionary {action_id, reward}, where reward is a float.

ucb1()

1.2 striatum.storage package

1.2.1 Submodules

1.2.2 striatum.storage.model module

Model storage

class striatum.storage.model.**MemoryModelStorage**
Bases: *striatum.storage.model.ModelStorage*

Store the model in memory.

Methods

get_model()
save_model(model)

get_model()

save_model(*model*)

class striatum.storage.model.**ModelStorage**
Bases: object

The object to store the model.

Methods

get_model()	Get model
save_model()	Save model

get_model()
Get model

save_model()
Save model

1.2.3 striatum.storage.history module

History storage

class striatum.storage.history.**History**(*history_id*, *action_time*, *context*, *action*, *reward_time=None*, *reward=None*)
Bases: object

action/reward history entry

Methods

update_reward(reward_time, reward)	update reward_time and reward

update_reward(*reward_time*, *reward*)
update reward_time and reward

class striatum.storage.history.**HistoryStorage**
Bases: object

The object to store the history of context, actions and rewards.

Methods

add_history(context, action[, reward])	Add a history record.
add_reward(history_id, reward)	Add reward to a history record.
get_history(history_id)	Get the preivous context, action and reward with history_id.
get_unrewarded_history(history_id)	Get the previous unrewarded context, action and reward with history_id.

add_history(*context*, *action*, *reward=None*)
Add a history record.

Parameters **context** : {array-like, None}

action : Action object

reward : {float, None}, optional (default: None)

add_reward(*history_id*, *reward*)
Add reward to a history record.

Parameters **history_id** : int

The history id of the history record to retrieve.

reward : float

get_history(*history_id*)
Get the preivous context, action and reward with history_id.

Parameters **history_id** : int

The history id of the history record to retrieve.

Returns history: History object

get_unrewarded_history(*history_id*)
Get the previous unrewarded context, action and reward with history_id.

Parameters **history_id** : int

The history id of the history record to retrieve.

Returns history: History object

class striatum.storage.history.**MemoryHistoryStorage**
Bases: *striatum.storage.history.HistoryStorage*

HistoryStorage that store all data in memory

Methods

add_history(context, action[, reward])	Add a history record.
add_reward(history_id, reward)	Add reward to a history record.
get_history(history_id)	Get the previous context, action and reward with history_id.
get_unrewarded_history(history_id)	Get the previous unrewarded context, action and reward with history_id.

add_history(*context*, *action*, *reward=None*)
Add a history record.

Parameters **context** : {array-like, None}

action : Action object

reward : {float, None}, optional (default: None)

add_reward(*history_id*, *reward*)
Add reward to a history record.

Parameters **history_id** : int

The history id of the history record to retrieve.

reward : float

get_history(*history_id*)
Get the previous context, action and reward with history_id.

Parameters **history_id** : int

The history id of the history record to retrieve.

Returns history: History object

get_unrewarded_history(*history_id*)
Get the previous unrewarded context, action and reward with history_id.

Parameters **history_id** : int

The history id of the history record to retrieve.

Returns history: History object

1.2.4 Module contents

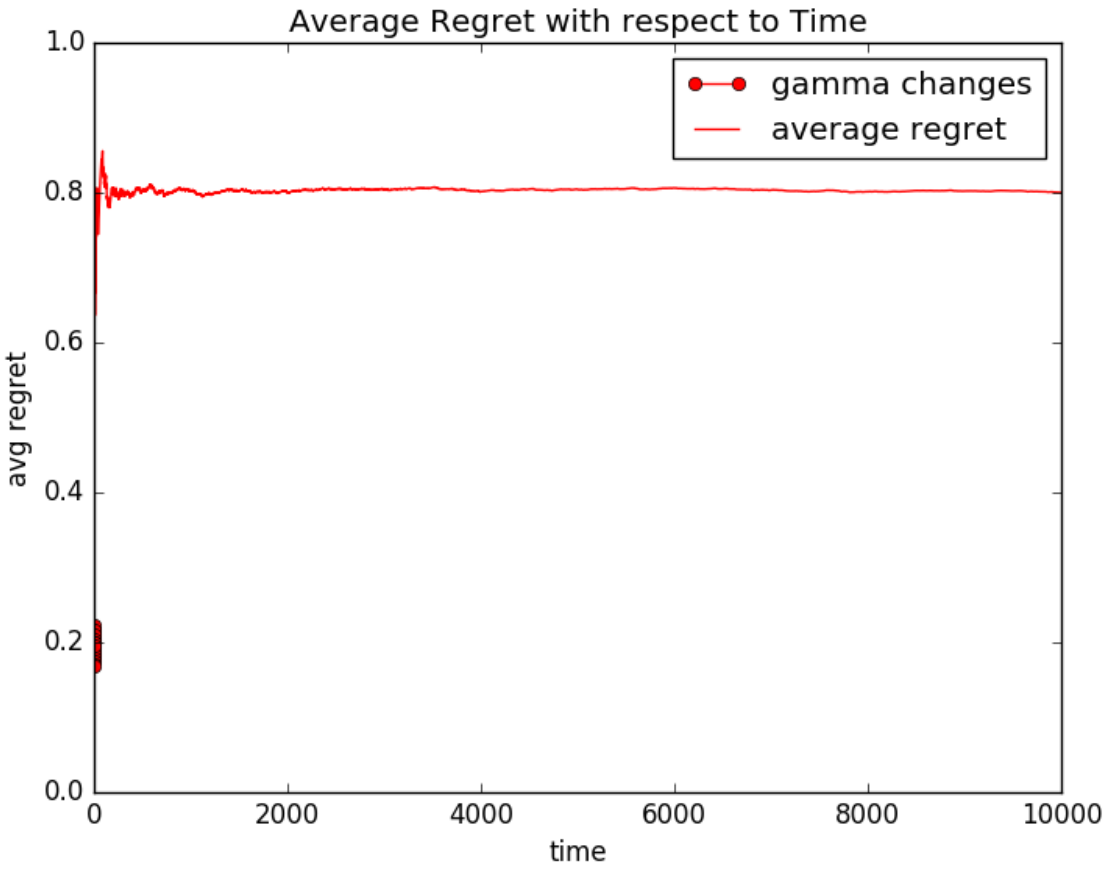
Storage classes

CHAPTER 2

Simulations of bandits

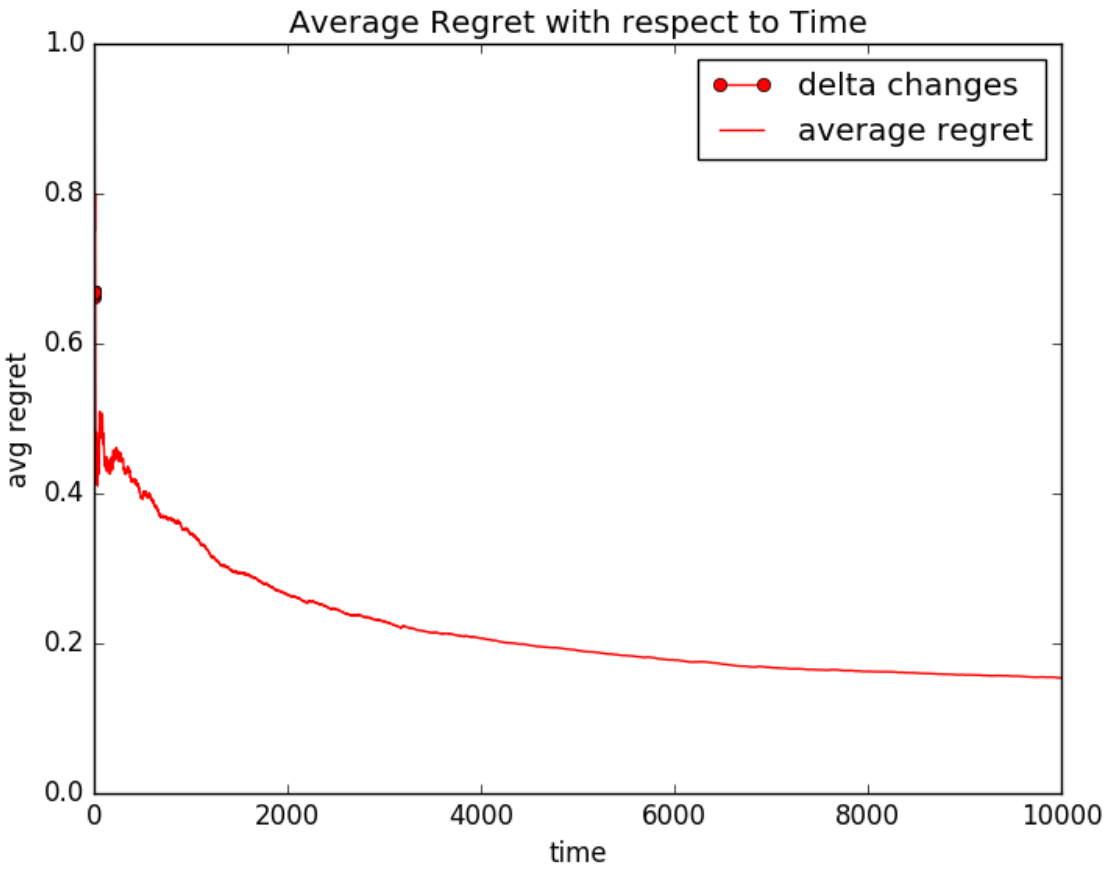
2.1 Exp3

Example file: `simulation/simulation_exp3.py`



2.2 Exp4.P

Example file: `simulation/simulation_exp4p.py`



2.3 Thompson sampling

Example file: `simulation/simulation_linthompsamp.py`

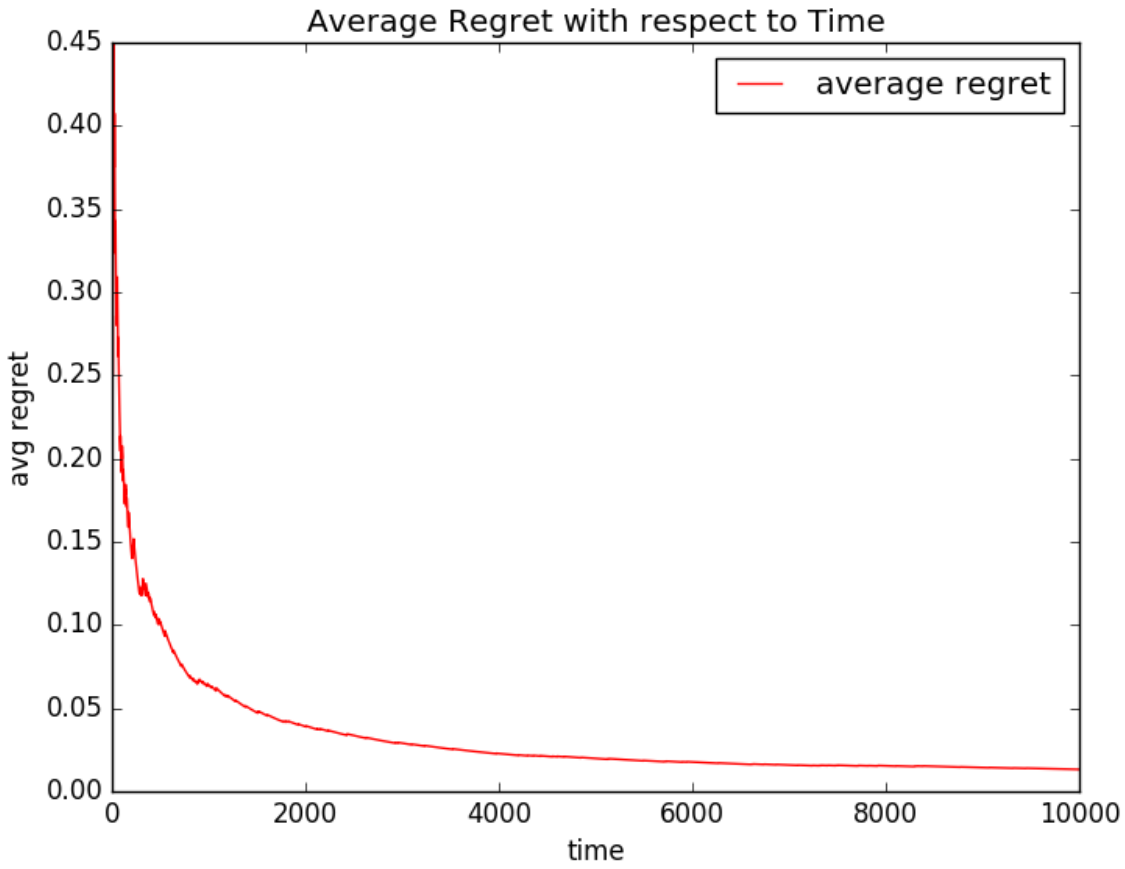
2.4 LinUCB

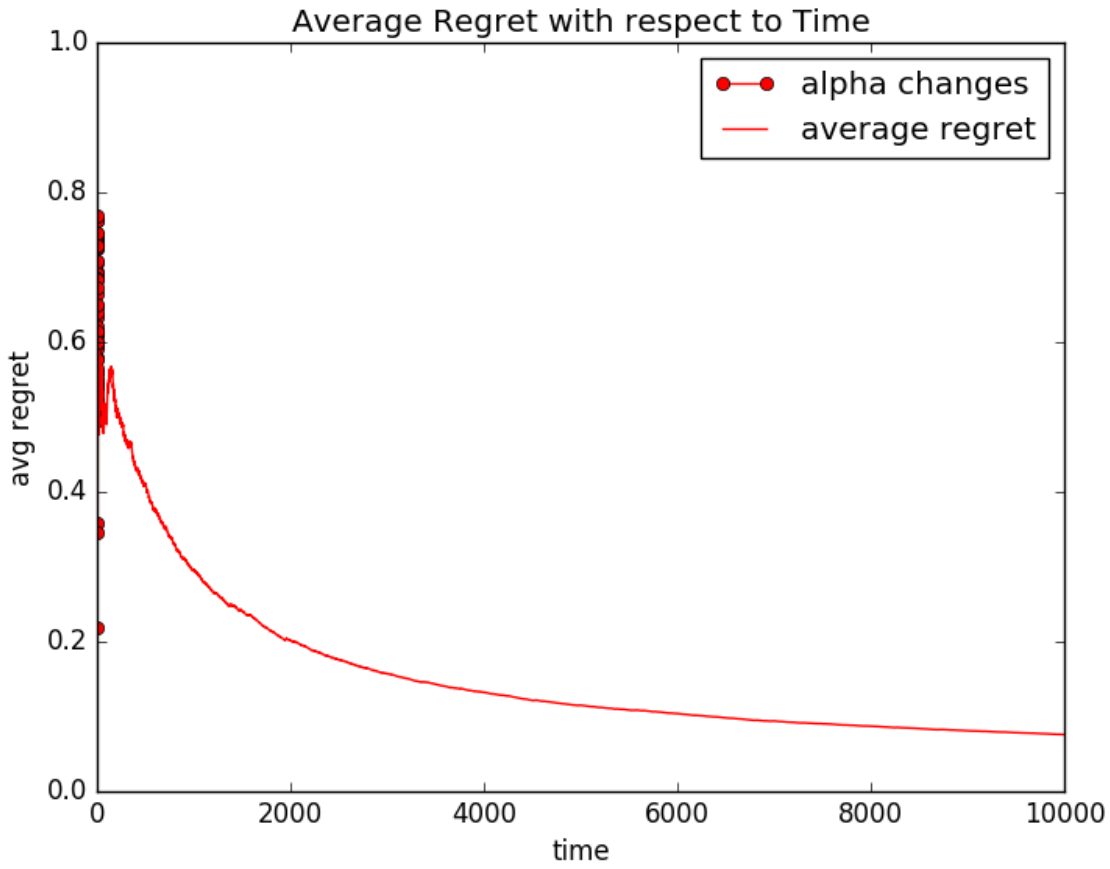
Example file: `simulation/simulation_linucb.py`

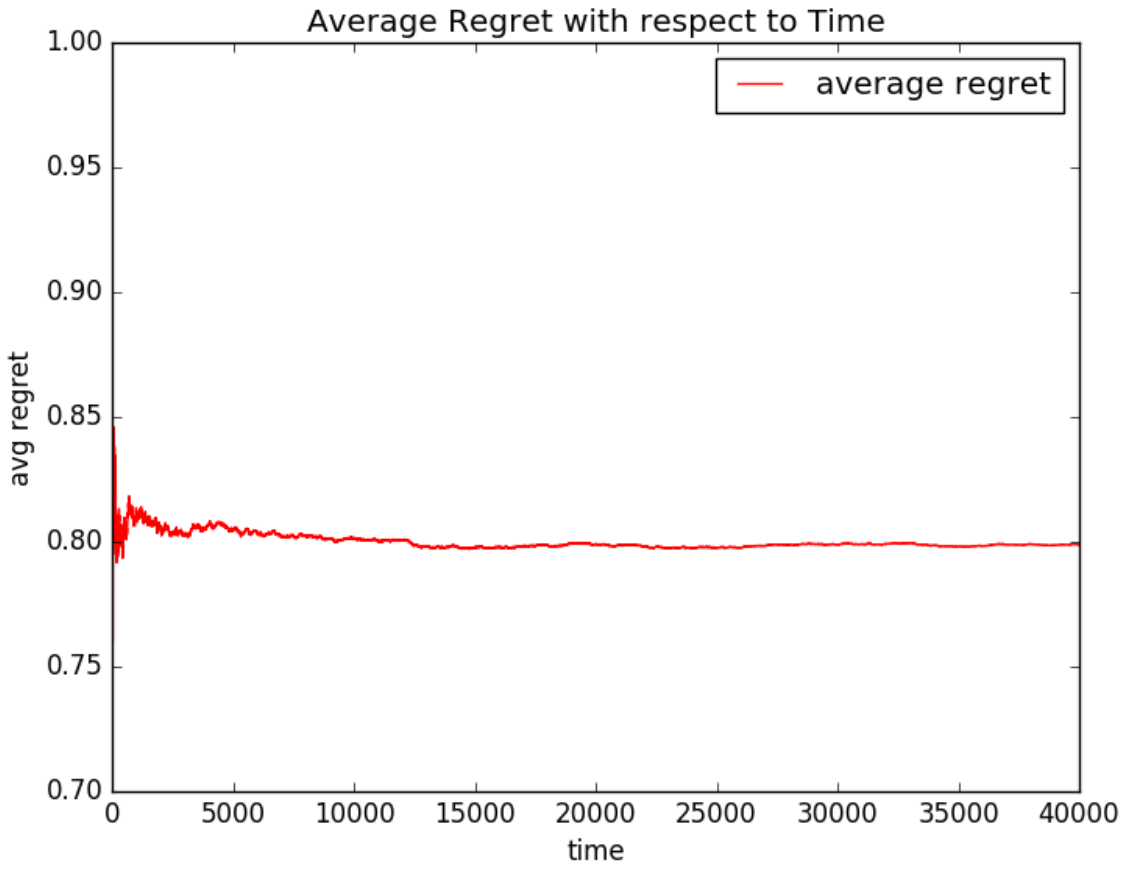
2.5 UCB1

Example file: `simulation/simulation_ucb1.py`

- genindex


Average Regret with respect to Time
average regret
avg regret
time
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
2000
4000
6000
8000
10000


Average Regret with respect to Time
alpha changes
average regret
avg regret
time
1.0
0.8
0.6
0.4
0.2
0.0
0
2000
4000
6000
8000
10000


Average Regret with respect to Time
average regret
avg regret
time
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
5000
10000
15000
20000
25000
30000
35000
40000

- modindex
- search

Bibliography

[R1] Peter Auer, Nicolo Cesa-Bianchi, et al. “The non-stochastic multi-armed bandit problem .” SIAM Journal of Computing. 2002.

[R2] Beygelzimer, Alina, et al. “Contextual bandit algorithms with supervised learning guarantees.” International Conference on Artificial Intelligence and Statistics (AISTATS). 2011u.

[R3] Shipra Agrawal, and Navin Goyal. “Thompson Sampling for Contextual Bandits with Linear Payoffs.” Advances in Neural Information Processing Systems 24. 2011.

[R4] Lihong Li, et al. “A Contextual-Bandit Approach to Personalized News Article Recommendation.” In Proceedings of the 19th International Conference on World Wide Web (WWW), 2010.

[R5] Peter Auer, et al. “Finite-time Analysis of the Multiarmed Bandit Problem.” Machine Learning, 47. 2002.

[R6] Peter Auer, Nicolo Cesa-Bianchi, et al. “The non-stochastic multi-armed bandit problem .” SIAM Journal of Computing. 2002.

[R7] Beygelzimer, Alina, et al. “Contextual bandit algorithms with supervised learning guarantees.” International Conference on Artificial Intelligence and Statistics (AISTATS). 2011u.

[R8] Shipra Agrawal, and Navin Goyal. “Thompson Sampling for Contextual Bandits with Linear Payoffs.” Advances in Neural Information Processing Systems 24. 2011.

[R9] Lihong Li, et al. “A Contextual-Bandit Approach to Personalized News Article Recommendation.” In Proceedings of the 19th International Conference on World Wide Web (WWW), 2010.

[R10] Peter Auer, et al. “Finite-time Analysis of the Multiarmed Bandit Problem.” Machine Learning, 47. 2002.

S

striatum.bandit, 10
striatum.bandit.exp3, 3
striatum.bandit.exp4p, 4
striatum.bandit.linthompsamp, 6
striatum.bandit.linucb, 7
striatum.bandit.ucb1, 9
striatum.storage, 20
striatum.storage.history, 18
striatum.storage.model, 17

A

add_action() (striatum.bandit.Exp3 method), 11
add_action() (striatum.bandit.exp3.Exp3 method), 4
add_action() (striatum.bandit.Exp4P method), 12
add_action() (striatum.bandit.exp4p.Exp4P method), 5
add_action() (striatum.bandit.LinThompSamp method), 14
add_action() (striatum.bandit.linthompsamp.LinThompSamp method), 7
add_action() (striatum.bandit.LinUCB method), 15
add_action() (striatum.bandit.linucb.LinUCB method), 8
add_action() (striatum.bandit.UCB1 method), 17
add_action() (striatum.bandit.ucb1.UCB1 method), 10
add_history() (striatum.storage.history.HistoryStorage method), 18
add_history() (striatum.storage.history.MemoryHistoryStorage method), 19
add_reward() (striatum.storage.history.HistoryStorage method), 19
add_reward() (striatum.storage.history.MemoryHistoryStorage method), 19

E

Exp3 (class in striatum.bandit), 10
Exp3 (class in striatum.bandit.exp3), 3
exp3() (striatum.bandit.Exp3 method), 11
exp3() (striatum.bandit.exp3.Exp3 method), 4
Exp4P (class in striatum.bandit), 12
Exp4P (class in striatum.bandit.exp4p), 4

G

get_action() (striatum.bandit.Exp3 method), 11
get_action() (striatum.bandit.exp3.Exp3 method), 4
get_action() (striatum.bandit.Exp4P method), 13
get_action() (striatum.bandit.exp4p.Exp4P method), 5
get_action() (striatum.bandit.LinThompSamp method), 14
get_action() (striatum.bandit.linthompsamp.LinThompSamp method), 7
get_action() (striatum.bandit.LinUCB method), 15
get_action() (striatum.bandit.linucb.LinUCB method), 8
get_action() (striatum.bandit.UCB1 method), 17
get_action() (striatum.bandit.ucb1.UCB1 method), 10
get_history() (striatum.storage.history.HistoryStorage method), 19
get_history() (striatum.storage.history.MemoryHistoryStorage method), 19
get_model() (striatum.storage.model.MemoryModelStorage method), 17
get_model() (striatum.storage.model.ModelStorage method), 18
get_unrewarded_history() (striatum.storage.history.HistoryStorage method), 19
get_unrewarded_history() (striatum.storage.history.MemoryHistoryStorage method), 20

H

History (class in striatum.storage.history), 18
HistoryStorage (class in striatum.storage.history), 18

L

LinThompSamp (class in striatum.bandit), 13
LinThompSamp (class in striatum.bandit.linthompsamp), 6
LinUCB (class in striatum.bandit), 14
LinUCB (class in striatum.bandit.linucb), 7

M

MemoryHistoryStorage (class in striatum.storage.history), 19
MemoryModelStorage (class in striatum.storage.model), 17
ModelStorage (class in striatum.storage.model), 17

R

reward() (striatum.bandit.Exp3 method), 11
reward() (striatum.bandit.exp3.Exp3 method), 4
reward() (striatum.bandit.Exp4P method), 13

reward() (striatum.bandit.exp4p.Exp4P method), 6
reward() (striatum.bandit.LinThompSamp method), 14
reward() (striatum.bandit.linthompsamp.LinThompSamp method), 7
reward() (striatum.bandit.LinUCB method), 16
reward() (striatum.bandit.linucb.LinUCB method), 9
reward() (striatum.bandit.UCB1 method), 17
reward() (striatum.bandit.ucb1.UCB1 method), 10

S

save_model() (striatum.storage.model.MemoryModelStorage method), 17
save_model() (striatum.storage.model.ModelStorage method), 18
striatum.bandit (module), 10
striatum.bandit.exp3 (module), 3
striatum.bandit.exp4p (module), 4
striatum.bandit.linthompsamp (module), 6
striatum.bandit.linucb (module), 7
striatum.bandit.ucb1 (module), 9
striatum.storage (module), 20
striatum.storage.history (module), 18
striatum.storage.model (module), 17

U

UCB1 (class in striatum.bandit), 16
UCB1 (class in striatum.bandit.ucb1), 9
ucb1() (striatum.bandit.UCB1 method), 17
ucb1() (striatum.bandit.ucb1.UCB1 method), 10
update_reward() (striatum.storage.history.History method), 18