
TicTacToe

Release 1.0.0

Kranthi Sedamaki

Apr 24, 2023

CONTENTS:

1	rl_tictactoe	1
1.1	policy_iteration module	1
1.2	q_learning module	2
1.3	tictactoe module	2
1.4	value_iteration module	5
2	Indices and tables	7
	Python Module Index	9
	Index	11

RL_TICTACTOE

1.1 policy_iteration module

class policy_iteration.PolicyIteration(*board_size: int = 3, win_condition: int = 3, gamma: float16 = 0.9*)

Bases: TicTacToe

Policy Iteration algorithm for Tic-Tac-Toe

static action_to_int(*action: ndarray[Any, dtype[int8]]*) → int
converts an action to an integer

Parameters

action (*npt.NDArray[np.int8]*) – the action

Returns

the integer representation of the action

Return type

int

get_policy(*epsilon: float = 0.0001, max_iterations: int = 100*) → Tuple[ndarray[Any, dtype[ScalarType]], ndarray[Any, dtype[ScalarType]]]

Generates a policy based on value iteration

Returns

A tuple of policy and values

Return type

Tuple[npt.NDArray, npt.NDArray]

update_values(*policy: ndarray[Any, dtype[int8]], prev_values: ndarray[Any, dtype[float16]]*) → ndarray[Any, dtype[float16]]

Updates the values of the states based on the policy

Parameters

- **policy** (*npt.NDArray[np.int8]*) – The policy to use
- **prev_values** (*npt.NDArray[np.float16]*) – The current values for each state

Returns

The updated values for each state

Return type

npt.NDArray[np.float16]

1.2 q_learning module

class q_learning.QLearning(*board_size: int = 3, win_condition: int = 3, gamma=0.9, learning_rate=0.1*)

Bases: PolicyIteration

Q Learning algorithm for Tic Tac Toe

get_policy() → Tuple[ndarray[Any, dtype[int8]], ndarray[Any, dtype[float16]]]

Returns

policy based on the q table npt.NDArray[np.float16]: the q table

Return type

npt.NDArray[np.int8]

max_expected_value(*state: int*) → float16

Returns the max expected value

Parameters

state (*int*) – the index of the current state

Returns

the max expected value

Return type

np.float16

q_learning(*num_episodes=1000000*) → None

Runs the Q-learning algorithm

update_q_table(*state: int, action: int*) → None

Updates the q table

Parameters

- **state** (*int*) – the index of the current state
- **action** (*int*) – the integer representation of the action
- **reward** (*int*) – the reward for the action
- **next_state** (*int*) – the index of the next state

1.3 tictactoe module

class tictactoe.TicTacToe(*board_size: int = 3, win_condition: int = 3, gamma: float16 = 0.9*)

Bases: object

A Tic-Tac-Toe game

static action_to_int(*action: ndarray[Any, dtype[int8]]*) → int

A utility function to convert the action to action int

Parameters

action (*npt.NDArray[np.int8]*) – the action

Returns

the action int

Return type

int

available_actions(*state: ndarray*) → ndarray[Any, dtype[int8]]

A utility function to get the available actions :param state: the current state of the board :type state: np.ndarray

Returns

the available actions

Return type

np.ndarray

static check_win(*state: ndarray[Any, dtype[int8]]*, *win_condition: int*) → Tuple[bool, int]

checks if there is a winner

Parameters

- **state** (*npt.NDArray[np.int8]*) – the board
- **win_condition** (*int*) – the number of markers in a row to win

Returns

(game_over, winner)

Return type

Tuple[bool, int]

draw_board()

draw the board on the screen

draw_game_over(*winner: int*)

draws the game over screen

draw_markers()

draw markers on the screen

static get_action(*state: ndarray*, *next_state: ndarray*) → Tuple[int, int]

A utility function to get the action from the state and next state :param state: the current state of the board :type state: np.ndarray :param next_state: the next state of the board :type next_state: np.ndarray

Returns

the action

Return type

Tuple[int, int]

index_to_state(*index: int*) → ndarray

A utility function to convert index to state :param index: index of the state from the state space :type index: int

Returns

the current state of the board

Return type

np.ndarray

static int_to_action(*action_int: int*) → ndarray[Any, dtype[int8]]

A utility function to convert the action int to action

Parameters**action_int** (*int*) – the action int

Returns

the action

Return type

npt.NDArray[np.int8]

static is_legal_action(*state: ndarray, x: int, y: int*) → bool

A utility function to check if the action is legal

is_terminal_state(*state: ndarray[Any, dtype[int8]]*) → bool

A utility function to check if the state is terminal

Parameters**state** (*npt.NDArray[np.int8]*) – the state**Returns**

returns true if the state is terminal else false

Return type

bool

is_valid_state(*state: ndarray[Any, dtype[int8]]*) → bool

Checks if the state is valid

Parameters**state** (*npt.NDArray[np.int8]*) – the input state**Returns**

returns true if the state is valid else false

Return type

bool

place_marker(*x: int, y: int*) → None

Places a marker on the board and updates the board inplace. Also updates the current player :param x: column :type x: int :param y: row :type y: int :param player: current player [1 -> player 1 (X), -1 -> player 2(O)] :type player: int

reset()

resets the game

reward_function(*state: ndarray[Any, dtype[int8]]*) → float

A utility function to get the reward of the a state

Returns

the reward of the current state

Return type

float

static state_to_index(*state: ndarray*) → int

A utility function to convert state to index :param state: the current state of the board :type state: np.ndarray

Returns

index of the state from the state space

Return type

int

which_players_turn(*state: ndarray[Any, dtype[int8]]*) → int

A utility function to get the current player

Returns

the current player

Return type

int

1.4 value_iteration module

class value_iteration.**ValueIteration**(*board_size: int = 3, win_condition: int = 3, gamma=0.9*)

Bases: TicTacToe

Value Iteration algorithm for Tic-Tac-Toe

static expected_value(*probabilities: ndarray[Any, dtype[float16]], value: ndarray[Any, dtype[float16]]*) → float16

A utility function to calculate the expected value

Parameters

- **probabilities** (*np.ndarray*) – the probabilities of the actions
- **rewards** (*np.ndarray*) – the rewards of the actions

Returns

the expected value

Return type

float

get_policy() → Tuple[ndarray[Any, dtype[ScalarType]], ndarray[Any, dtype[ScalarType]]]

Generates a policy based on value iteration

Returns

A tuple of policy and values

Return type

Tuple[npt.NDArray, npt.NDArray]

update_values(*epsilon: float = 0.0001, max_iterations: int = 100*) → ndarray[Any, dtype[float16]]

Performs value update

Parameters

- **epsilon** (*float, optional*) – the limiting difference. Defaults to 1e-4.
- **max_iterations** (*int, optional*) – max number of iterations the algorithm is allowed to run. Defaults to 100.

Returns

an array of values for each state

Return type

npt.NDArray[np.float16]

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

`policy_iteration`, [1](#)

q

`q_learning`, [2](#)

t

`tictactoe`, [2](#)

v

`value_iteration`, [5](#)

INDEX

A

`action_to_int()` (*policy_iteration.PolicyIteration static method*), 1
`action_to_int()` (*tictactoe.TicTacToe static method*), 2
`available_actions()` (*tictactoe.TicTacToe method*), 3

C

`check_win()` (*tictactoe.TicTacToe static method*), 3

D

`draw_board()` (*tictactoe.TicTacToe method*), 3
`draw_game_over()` (*tictactoe.TicTacToe method*), 3
`draw_markers()` (*tictactoe.TicTacToe method*), 3

E

`expected_value()` (*value_iteration.ValueIteration static method*), 5

G

`get_action()` (*tictactoe.TicTacToe static method*), 3
`get_policy()` (*policy_iteration.PolicyIteration method*), 1
`get_policy()` (*q_learning.QLearning method*), 2
`get_policy()` (*value_iteration.ValueIteration method*), 5

I

`index_to_state()` (*tictactoe.TicTacToe method*), 3
`int_to_action()` (*tictactoe.TicTacToe static method*), 3
`is_legal_action()` (*tictactoe.TicTacToe static method*), 4
`is_terminal_state()` (*tictactoe.TicTacToe method*), 4
`is_valid_state()` (*tictactoe.TicTacToe method*), 4

M

`max_expected_value()` (*q_learning.QLearning method*), 2
module
 policy_iteration, 1
 q_learning, 2
 tictactoe, 2

value_iteration, 5

P

`place_marker()` (*tictactoe.TicTacToe method*), 4
policy_iteration
 module, 1
PolicyIteration (*class in policy_iteration*), 1

Q

q_learning
 module, 2
`q_learning()` (*q_learning.QLearning method*), 2
QLearning (*class in q_learning*), 2

R

`reset()` (*tictactoe.TicTacToe method*), 4
`reward_function()` (*tictactoe.TicTacToe method*), 4

S

`state_to_index()` (*tictactoe.TicTacToe static method*), 4

T

tictactoe
 module, 2
TicTacToe (*class in tictactoe*), 2

U

`update_q_table()` (*q_learning.QLearning method*), 2
`update_values()` (*policy_iteration.PolicyIteration method*), 1
`update_values()` (*value_iteration.ValueIteration method*), 5

V

value_iteration
 module, 5
ValueIteration (*class in value_iteration*), 5

W

`which_players_turn()` (*tictactoe.TicTacToe method*), 4