cartpole-tf-dqn

Tianhao Zhou

Aug 24, 2020

CONTENTS

1 Replay Buffer 3

2 Visualizer 5

3 CLI Entrypoint 7

4 Configuration 9

5 Train 11

6 Utilities 13

7 Tests 15

Python Module Index 17

Index 19

class dqn_agent.**DqnAgent**(*state_space*, *action_space*, *gamma*, *lr*, *verbose*, *checkpoint_location*, *model_location*, *persist_progress_option*, *mode*, *epsilon*)

DQN agent with production policy and benchmark

collect_policy(*state*)

The policy for collecting data points which can contain some randomness to encourage exploration.

Returns action

load_checkpoint()

Loads training checkpoint into the underlying model

Returns None

load_model()

Loads previously saved model :return: None

policy(*state*)

Outputs a action based on model

Parameters **state** – current state

Returns action

random_policy(*state*)

Outputs a random action

Parameters **state** – current state

Returns action

save_checkpoint()

Saves training checkpoint

Returns None

save_model()

Saves model to file system

Returns None

train(*state_batch*, *next_state_batch*, *action_batch*, *reward_batch*, *done_batch*, *batch_size*)

Train the model on a batch

Parameters

- **state_batch** – batch of states
- **next_state_batch** – batch of next states
- **action_batch** – batch of actions
- **reward_batch** – batch of rewards
- **done_batch** – batch of done status
- **batch_size** – the size of the batch

Returns loss history

update_target_network()

Updates the target Q network with the parameters from the currently trained Q network.

Returns None

CHAPTER

ONE

REPLAY BUFFER

class replay_buffer.**DqnReplayBuffer**(*max_size*)

DQN replay buffer to keep track of game play records

can_sample_batch(*batch_size*)

Returns if a batch can be sampled

Parameters **batch_size** – the size of the batch to be sampled

Returns (bool) if can sample

get_volume()

Gets the current length of the records

Returns (int) the length of the records

record(*state*, *reward*, *next_state*, *action*, *done*)

Puts a game play state into records

Parameters

- **state** – current game state
- **reward** – reward after taking action
- **next_state** – state after taking action
- **action** – action taken
- **done** – if the episode is finished

Returns None

sample_batch(*batch_size*)

Samples a batch from the records

Parameters **batch_size** – the size of the batch to be sampled

Returns sample batch

CHAPTER

TWO

VISUALIZER

Training progress visualizer

class visualizer.**DummyTrainingVisualizer**
Used when no logging is required

get_ui_feedback()
A dummy logger that does nothing

Returns None

log_loss(*loss*)
A dummy logger that does nothing

Parameters **loss** – a list of loss history

Returns None

log_reward(*reward*)
A dummy logger that does nothing

Parameters **reward** – a list of reward history

Returns None

class visualizer.**StreamlitTrainingVisualizer**
Used when runs with stream lit

get_ui_feedback()
Gets the user defined config from the UI

Returns config

log_loss(*loss*)
Adds a loss history to the chart

Parameters **loss** – a list of loss history

Returns None

log_reward(*reward*)
Adds a reward history to the chart

Parameters **reward** – a list of reward history

Returns

class visualizer.**TrainingVisualizer**
Base training visualizer

abstract get_ui_feedback()
Gets the configuration from UI

Returns None

abstract log_loss(*loss*)
Logs a loss history to the desired visualization

Parameters **loss** – a list of loss history

Returns None

abstract log_reward(*reward*)
Logs a reward history to the desired visualization

Parameters **reward** – a list of reward history

Returns None

visualizer.**get_training_visualizer**(*visualizer_type*)
A factory wrapper to generate training progress visualizers.

Parameters **visualizer_type** – (str) the type of the visualizer to create

Returns TrainingVisualizer

CHAPTER

THREE

CLI ENTRYPOINT

entrypoint.**main**()

The CLI entrypoint to the APIs

Returns None

CHAPTER

FOUR

CONFIGURATION

Config

config.**DEFAULT_BATCH_SIZE = 128**
The default batch size the model should be trained on

config.**DEFAULT_CHECKPOINT_LOCATION = './checkpoints'**
The default location to store the training checkpoints

config.**DEFAULT_ENV_NAME = 'CartPole-v0'**
The OpenAI environment name to be used

config.**DEFAULT_EPSILON = 0.05**
The default value for epsilon

config.**DEFAULT_EVAL_EPS = 10**
The default number of episode the model should be evaluated with

config.**DEFAULT_GAMMA = 0.95**
The default discount rate for the Q learning

config.**DEFAULT_LEARNING_RATE = 0.001**
The default learning rate

config.**DEFAULT_MAX_REPLAY_HISTORY = 1000000**
The default max length of the replay buffer

config.**DEFAULT_MIN_STEPS = 10**
The minimum number of steps the evaluation should run per episode so that the tester can better visualize how the agent is doing.

config.**DEFAULT_MODE = 'train'**
The default mode the program should run in

config.**DEFAULT_MODEL_LOCATION = './model'**
The default location to store the best performing models

config.**DEFAULT_NUM_ITERATIONS = 50000**
The default number of iteration to train the model

config.**DEFAULT_PAUSE_TIME = 0**
The default value for pausing before execution starts to make time for screen recording. It's only available in testing mode since it's pointless to do so while training.

config.**DEFAULT_RENDER_OPTION = 'none'**
The default value for rendering option

config.**DEFAULT_TARGET_NETWORK_UPDATE_FREQUENCY = 120**
How often the target Q network should get parameter update from the training Q network.

config.**DEFAULT_VERBOSITY_OPTION = 'progress'**
The default verbosity option

config.**DEFAULT_VISUALIZER_TYPE = 'none'**
The default visualizer type

config.**MODE_OPTIONS = ['train', 'test']**
The supported modes

config.**RENDER_OPTIONS = ['none', 'collect']**
The available render options:

- none: don't render anything
- collect: render the game play while collecting data

config.**VERBOSITY_OPTIONS = ['progress', 'loss', 'policy', 'init']**
The available verbosity options:

- progress: show the training progress
- loss: show the logging information from loss calculation
- policy: show the logging information from policy generation
- init: show the logging information from initialization

CHAPTER

FIVE

TRAIN

train.**train_model**(*num_iterations=50000*, *batch_size=128*, *max_replay_history=1000000*, *gamma=0.95*, *eval_eps=10*, *learning_rate=0.001*, *target_network_update_frequency=120*, *checkpoint_location='./checkpoints'*, *model_location='./model'*, *verbose='progress'*, *visualizer_type='none'*, *render_option='none'*, *persist_progress_option='all'*, *epsilon=0.05*)

Trains a DQN agent by playing episodes of the Cart Pole game

Parameters

- **epsilon** – epsilon is the probability that a random action is chosen
- **target_network_update_frequency** – how frequent target Q network gets updates
- **num_iterations** – the number of episodes the agent will play
- **batch_size** – the training batch size
- **max_replay_history** – the limit of the replay buffer length
- **gamma** – discount rate
- **eval_eps** – the number of episode per evaluation
- **learning_rate** – the learning rate of the back propagation
- **checkpoint_location** – the location to save the training checkpoints
- **model_location** – the location to save the pre-trained models
- **verbose** – the verbosity level which can be progress, loss, policy and init
- **visualizer_type** – the type of visualization to be used
- **render_option** – if the game play should be rendered
- **persist_progress_option** – if the training progress should be saved

Returns (maximum average reward, baseline average reward)

CHAPTER

SIX

UTILITIES

Utilities

utils.**collect_episode**(*env*, *policy*, *buffer*, *render_option*)
Collect steps from a single episode play and record with replay buffer

Parameters

- **env** – OpenAI gym environment
- **policy** – DQN agent policy
- **buffer** – reinforcement learning replay buffer
- **render_option** – (bool) if should render the game play

Returns None

utils.**collect_steps**(*env*, *policy*, *buffer*, *render_option*, *current_state*, *n_steps*)
Collects a single step from the game environment with policy specified. It is currently not used in favor of collect_episode API.

Parameters

- **n_steps** – the number of steps to collect
- **current_state** – the current state of the environment
- **env** – OpenAI gym environment
- **policy** – DQN agent policy
- **buffer** – reinforcement learning replay buffer
- **render_option** – (bool) if should render the game play

Returns None

utils.**compute_avg_reward**(*env*, *policy*, *num_episodes*)
Compute the average reward across num_episodes under policy

Parameters

- **env** – OpenAI gym environment
- **policy** – DQN agent policy
- **num_episodes** – the number of episode to take average from

Returns (int) average reward

utils.**play_episode**(*env*, *policy*, *render_option*, *min_steps*)
Play an episode with the given policy.

Parameters

- **min_steps** – the minimum steps the game should be played
- **env** – the OpenAI gym environment
- **policy** – the policy that should be used to generate actions
- **render_option** – how the game play should be rendered

Returns episode reward

utils.**play_episodes**(*env*, *policy*, *render_option*, *num_eps*, *pause_time*, *min_steps*)
Play episodes with the given policy

Parameters

- **min_steps** – the minimum steps the game should be played
- **pause_time** – the time that should pause to prepare for screen recording
- **env** – the OpenAI gym environment
- **policy** – the policy that should be used to generate actions
- **render_option** – how the game play should be rendered
- **num_eps** – how many episodes should be played

Returns average episode reward

CHAPTER

SEVEN

TESTS

Tests for model training

class train_test.**TestTrain**(*methodName='runTest'*)

Test suite for model training

test_sanity_check()

Tests if the model training finishes without crashing

Returns None

test_training_effectiveness()

Test if the model training can achieve a performance better than a random policy

Returns None

PYTHON MODULE INDEX

c

config, 9

t

train_test, 15

u

utils, 13

v

visualizer, 5

INDEX

C

can_sample_batch() (*replay_buffer.DqnReplayBuffer method*), 3
collect_episode() (*in module utils*), 13
collect_policy() (*dqn_agent.DqnAgent method*), 1
collect_steps() (*in module utils*), 13
compute_avg_reward() (*in module utils*), 13
config
    module, 9

D

DEFAULT_BATCH_SIZE (*in module config*), 9
DEFAULT_CHECKPOINT_LOCATION (*in module config*), 9
DEFAULT_ENV_NAME (*in module config*), 9
DEFAULT_EPSILON (*in module config*), 9
DEFAULT_EVAL_EPS (*in module config*), 9
DEFAULT_GAMMA (*in module config*), 9
DEFAULT_LEARNING_RATE (*in module config*), 9
DEFAULT_MAX_REPLAY_HISTORY (*in module config*), 9
DEFAULT_MIN_STEPS (*in module config*), 9
DEFAULT_MODE (*in module config*), 9
DEFAULT_MODEL_LOCATION (*in module config*), 9
DEFAULT_NUM_ITERATIONS (*in module config*), 9
DEFAULT_PAUSE_TIME (*in module config*), 9
DEFAULT_RENDER_OPTION (*in module config*), 9
DEFAULT_TARGET_NETWORK_UPDATE_FREQUENCY (*in module config*), 9
DEFAULT_VERBOSITY_OPTION (*in module config*), 9
DEFAULT_VISUALIZER_TYPE (*in module config*), 10
DqnAgent (*class in dqn_agent*), 1
DqnReplayBuffer (*class in replay_buffer*), 3
DummyTrainingVisualizer (*class in visualizer*), 5

G

get_training_visualizer() (*in module visualizer*), 6
get_ui_feedback() (*visualizer.DummyTrainingVisualizer method*), 5
get_ui_feedback() (*visualizer.StreamlitTrainingVisualizer method*), 5
get_ui_feedback() (*visualizer.TrainingVisualizer method*), 5
get_volume() (*replay_buffer.DqnReplayBuffer method*), 3

L

load_checkpoint() (*dqn_agent.DqnAgent method*), 1
load_model() (*dqn_agent.DqnAgent method*), 1
log_loss() (*visualizer.DummyTrainingVisualizer method*), 5
log_loss() (*visualizer.StreamlitTrainingVisualizer method*), 5
log_loss() (*visualizer.TrainingVisualizer method*), 6
log_reward() (*visualizer.DummyTrainingVisualizer method*), 5
log_reward() (*visualizer.StreamlitTrainingVisualizer method*), 5
log_reward() (*visualizer.TrainingVisualizer method*), 6

M

main() (*in module entrypoint*), 7
MODE_OPTIONS (*in module config*), 10
module
    config, 9
    train_test, 15
    utils, 13
    visualizer, 5

P

play_episode() (*in module utils*), 13
play_episodes() (*in module utils*), 14
policy() (*dqn_agent.DqnAgent method*), 1

R

random_policy() (*dqn_agent.DqnAgent method*), 1
record() (*replay_buffer.DqnReplayBuffer method*), 3
RENDER_OPTIONS (*in module config*), 10

S

sample_batch() (*replay_buffer.DqnReplayBuffer method*), 3
save_checkpoint() (*dqn_agent.DqnAgent method*), 1
save_model() (*dqn_agent.DqnAgent method*), 1
StreamlitTrainingVisualizer (*class in visualizer*), 5

T

test_sanity_check() (*train_test.TestTrain method*), 15
test_training_effectiveness() (*train_test.TestTrain method*), 15
TestTrain (*class in train_test*), 15
train() (*dqn_agent.DqnAgent method*), 1
train_model() (*in module train*), 11
train_test
    module, 15
TrainingVisualizer (*class in visualizer*), 5

U

update_target_network() (*dqn_agent.DqnAgent method*), 1
utils
    module, 13

V

VERBOSITY_OPTIONS (*in module config*), 10
visualizer
    module, 5