api

API Documentation

November 4, 2019

Contents

1 Package deepwifi

1.1 Modules

- **DQL** *(Section 2, p. 4)*
    - **clone**: This module allows the cloning of a Keras model
      (Section 3, p. 5)
    - **ddql**: This module implements Double Deep QL
      (Section 4, p. 6)
    - **deepQL**: This module implements DeepQL - version 1
      (Section 5, p. 8)
    - **dql**: This module implements Deep QL with two networks: Q-network and target-network
      (Section 6, p. 13)
    - **test** *(Section 7, p. 16)*
- **Environment** *(Section 8, p. 17)*
    - **common** *(Section 9, p. 18)*
    - **env** *(Section 10, p. 25)*
    - **fairness**: The result ranges from 1/n (worst case) to 1 (best case), and it is maximum when all users receive the same allocation.
      (Section 11, p. 27)
    - **generic_ap**: Environment implementation (abstract class) that represents the experiment using Video This class implements the basic functions to control the APs, but it does not implement the QoE
      (Section 12, p. 28)
    - **gini**: Calculate the global reward penalized using the gini_coeficient coefficient of a numpy array of data.
      (Section 13, p. 33)
    - **grid_world**: Create a grid world
      (Section 14, p. 34)
    - **hossfeld**: The hossfeld index ranges from 0 (worst case) to 1 (best case), and it is maximum when all users receive the same allocation (homogeneity).
      (Section 15, p. 37)
    - **interface_env**: this defines the interface of the environment class environment all methods here should be implemented
      (Section 16, p. 38)
    - **qoe_ap**: Environment implementation (concrete class) that represents the experiment using Video and QoE, where QoE is calculated using only AP parameters
      (Section 17, p. 40)
    - **qoe_client**: this module calculates the MOS using only data from the client
      (Section 18, p. 43)
    - **qoe_hybrid**: this module uses two sources to calculate the MOS: from AP and Client
      (Section 19, p. 48)
    - **qoe_psnr**: this module calculates the MOS using only data from the client
      (Section 20, p. 51)
    - **testEnv** *(Section 21, p. 53)*
- **MAB** *(Section 22, p. 56)*
    - **mab**: This is a module that runs RL method to control wifi devices.
      (Section 23, p. 57)
- **Memory** *(Section 24, p. 58)*
    - **memory**: This module defines the interface for the replay memory buffer
      (Section 25, p. 59)

1.2 Variables

Name	**Description**
__package__	**Value:** `None`

2 Package deepwifi.DQL

2.1 Modules

- **clone**: This module allows the cloning of a Keras model
  (Section 3, p. 5)
- **ddql**: This module implements Double Deep QL
  (Section 4, p. 6)
- **deepQL**: This module implements DeepQL - version 1
  (Section 5, p. 8)
- **dql**: This module implements Deep QL with two networks: Q-network and target-network
  (Section 6, p. 13)
- **test** *(Section 7, p. 16)*

2.2 Variables

Name	Description
__package__	**Value:** `None`

3 Module deepwifi.DQL.clone

This module allows the cloning of a Keras model

3.1 Functions

clone_model(*model*)

4 Module deepwifi.DQL.ddql

This module implements Double Deep QL

4.1 Variables

Name	Description
LOG	**Value:** `logging.getLogger('DDQL')`

4.2 Class DDQL

object
deepwifi.DQL.deepQL.DeepQL
deepwifi.DQL.dql.DQL
deepwifi.DQL.ddql.DDQL

ref.

4.2.1 Methods

get_q_max(*self*, *sprime*)

```
the Q_max is calculated using the target network
@param sprime: the sequence of next states (s')

@return: the Qmax value used in the TD-error. to avoid overestimation used Q-function to predict t
         uses this value to obtain value of the Q(s', a') using the target network

Q_max = Q_target(s', arg max Q(s', a'))
                         a'
```

Overrides: deepwifi.DQL.deepQL.DeepQL.get_q_max

Inherited from deepwifi.DQL.dql.DQL(Section 6.1)

__init__(), copy_to_target(), copy_weights(), replay(), save_model()

Inherited from deepwifi.DQL.deepQL.DeepQL(Section 5.3)

format_state_to_predict(), get_action(), get_action_boltzmann(), get_action_eps_greedy(), predict_action_output(), remember(), run(), stop(), stop_running(), update_epsilon()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

4.2.2 Properties

Name	Description
Inherited from deepwifi.DQL.deepQL.DeepQL (Section 5.3) number_of_runs	
Inherited from object ___class___	

4.2.3 Class Variables

Name	Description
Inherited from deepwifi.DQL.deepQL.DeepQL (Section 5.3) TO_MUCH_ERROR_IN_A_ROW	

5 Module deepwifi.DQL.deepQL

This module implements DeepQL - version 1

the DeepQL uses an MLP Q-network it is retrained only after 'episodes' iterations the training uses replay memory

bibliographic references: ==========================

VAN HASSELT, Hado; GUEZ, Arthur; SILVER, David. Deep reinforcement learning with double q-learning. In: Thirtieth AAAI conference on artificial intelligence. 2016.

HASSELT, Hado V. Double Q-learning. In: Advances in Neural Information Processing Systems. 2010. p. 2613-2621.

WANG, Ziyu et al. Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581, 2015.

CLEMENTE, Alfredo V.; CASTEJÓN, Humberto N.; CHANDRA, Arjun. Efficient parallel methods for deep reinforcement learning. arXiv preprint arXiv:1705.04862, 2017.

MNIH, Volodymyr et al. Asynchronous methods for deep reinforcement learning. In: International conference on machine learning. 2016. p. 1928-1937.

5.1 Functions

softmax(*z*)

returns the softmax function (probabilities) given an array z

Parameters

`z`: an 1D array of float

(type=np.array)

Return Value

softmax(x)

(type=np.array)

softmax_2d(*z*)

Parameters

`z`: an array (2, n)

(type=np.array)

5.2 Variables

Name	Description
__package__	**Value:** `'deepwifi.DQL'`

5.3 Class DeepQL

object ─┐
deepwifi.DQL.deepQL.DeepQL

```
ref. https://keon.io/deep-q-learning/
     https://github.com/simoninithomas/deep_q_learning/blob/master/DeepQL%20Cartpole.ipy
     https://medium.com/@gtnjuvin/my-journey-into-deep-q-learning-with-keras-and-gym-3e7
     https://medium.com/@awjuliani/simple-reinforcement-learning-with-tensorflow-part-4-
```

5.3.1 Methods

__init__(*self*, *env*, *model*, *memory*, *timesteps*=`1`, *epsilon*=`0.01`, *epsilon_min*=`0.1`, *epsilon_decay*=`0.995`, *learning_rate*=`0.001`, *gamma*=`0.95`, *batch_size*=`32`, *episodes*=`30`, *epochs*=`1`, *interaction_interval*=`30`, *log_level*=`10`, ***kwargs*)

x.__init__(...) initializes x; see help(type(x)) for signature

Parameters

- `env`: the environment class
- `model`: the Keras model used to approximate the Q-function
- `memory`: the replay memory implementation

Overrides: object.__init__

save_model(*self*, *model_filename*=`'model.json'`)

save the model to a json file and the weights to a h5 file

Parameters

- `model_filename`: the filename with '.json' extension

remember(*self*, *states*, *actions*, *next_states*, *rewards*)

pushes s, a, s', r

Parameters

`states:`	list of initial state, one for each AP
`actions:`	list of actions taken, one for each AP
`next_states:`	list of next state, one for each AP
`rewards:`	list of rewards, one for each AP

format_state_to_predict(*self*, *values*, *batch_size*=1)

formats to use in predict, because predict needs first dimension ==> entries followed by the other dimension in values

Parameters

`values:`	list of values to convert to a numpy array
`batch_size:`	defines the size of the batch (first dimension size)

Return Value

a numpu array with self.timesteps, composed by the self.prev_states values (saved from previous runs) and the value passed as parameter

get_q_max(*self*, *sprime*)

```

the Q_max is calculated using the model network
notice that you don't need to call self.format_state_to_predict() for sprime
sprime format depends on the number of time steps, thus
dim(s') = (1, timesteps, num_features)
... num_features = self.state_dim

@param next_state: the next state s'
@return: the Q_max for the state s'

Q_max = max Q(s', a')
          a'
```

update_epsilon(*self*)

perform epsilon decay. To prevent the decay to occur, just set epsilon_decay to None. If epsilon_min is None, then decays forever. Otherwise decays while epsilon > epsilon_min

replay(*self*)

decides if the replay will occur, if not just returns uses the memory to recover a mini-batch that will be used to train the model network

Return Value

nothing

predict_action_output(*self*, *curr_state*)

Predict the reward value based on the given state this method formats 'curr_state' using self.format_state_to_predict() in order to call the keras predict()

Parameters

`curr_state`: the current state (one for each device)

Return Value

values for all the actions

(type=list)

get_action_eps_greedy(*self*, *curr_state*)

select the action (one for each ap), epsilon greedy way

Parameters

`curr_state`: a list of current states, one for each AP

Return Value

list[int]: each entry is a number that represents the action for that state

get_action_boltzmann(*self*, *curr_state*)

select the action (one for each ap), using boltzmann

Parameters

`curr_state`: a list of current state, one for each AP

Return Value

list[int]: each entry is a number that represents the action for that state

get_action(*self*, *states*)

overwrite this method to call self.get_action_eps_greedy() or self.get_action_boltzmann() to implement the search policy

Parameters

`states`: a list of states, one for each AP

stop(*self*)

change the stopping flag in the run(), so the program will stop at the end of the iteration

stop__running(*self*)

change the flag that controls the while loop in run() so the agent stop at the end of that execution

run(*self*, *run__id*=1, *wait__for__states*=10, *save__iterations*=20)

executes the control loop

Parameters

`wait_for_states`: how much time should sleep between get__states request

(type=int)

`save_iterations`: every 'save__iterations' iterations, save the model and the weights

Return Value

if the agents detected to much errors in a row

(type=bool)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

5.3.2 Properties

Name	Description
number__of__runs	number of times the program iteracted and acted upon the environment *(type=int)*
Inherited from object ___class___	

5.3.3 Class Variables

Name	Description
TO__MUCH__ERROR__I-N__A__ROW	**Value:** 20

6 Module deepwifi.DQL.dql

This module implements Deep QL with two networks: Q-network and target-network

the DQL uses an MLP Q-network it is retrained only after 'episodes' iterations the training uses replay memory

6.1 Class DQL

object
deepwifi.DQL.deepQL.DeepQL
deepwifi.DQL.dql.DQL

```
ref. https://keon.io/deep-q-learning/
     https://github.com/simoninithomas/deep_q_learning/blob/master/DQL%20Cartpole.ipynb
     https://medium.com/@gtnjuvin/my-journey-into-deep-q-learning-with-keras-and-gym-3e7
```

6.1.1 Methods

__init__(*self*, *env*, *model*, *memory*, *timesteps*=`1`, *epsilon*=`0.1`, *epsilon_min*=`0.1`, *epsilon_decay*=`0.995`, *learning_rate*=`0.001`, *gamma*=`0.95`, *batch_size*=`32`, *episodes*=`30`, *epochs*=`1`, *log_level*=`logging.DEBUG`, *interaction_interval*=`30`, ***kwargs*)

x.__init__(...) initializes x; see help(type(x)) for signature

Parameters

`env`: the environment class

`model`: the Keras model used to approximate the Q-function

`memory`: the replay memory implementation

Overrides: object.__init__ extit(inherited documentation)

copy_to_target(*self*)

copy_weights(*self*)

save_model(*self*, *model_filename*=`'model.json'`)

save the model and target networks to a json file and the weights to a h5 file overwritten method to save both networks

Parameters

`model_filename`: the filename with '.json' extension

Overrides: deepwifi.DQL.deepQL.DeepQL.save_model

get_q_max(*self*, *sprime*)

```

the Q_max is calculated using the target network
@param sprime: the sequence of next states (s')

@return: the Qmax value used in the TD-error, defined as the greedy move
Q_max = max Q_target(s', a')
         a'
```

Overrides: deepwifi.DQL.deepQL.DeepQL.get_q_max

replay(*self*, *C*=`10`)

produces the replay, that trains the model's parameters and if C replays occur then update target's parameters

Return Value

nothing

Overrides: deepwifi.DQL.deepQL.DeepQL.replay

Inherited from deepwifi.DQL.deepQL.DeepQL(Section 5.3)

format_state_to_predict(), get_action(), get_action_boltzmann(), get_action_eps_greedy(), predict_action_output(), remember(), run(), stop(), stop_running(), update_epsilon()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

6.1.2 Properties

Name	Description
Inherited from deepwifi.DQL.deepQL.DeepQL (Section 5.3)	
number_of_runs	
Inherited from object	

continued on next page

Name	Description
__class__	

6.1.3 Class Variables

Name	Description
Inherited from deepwifi.DQL.deepQL.DeepQL (Section 5.3) TO_MUCH_ERROR_IN_A_ROW	

7 Module deepwifi.DQL.test

7.1 Variables

Name	Description
LOG	**Value:** `logging.getLogger('Test DeepQL')`

8 Package deepwifi.Environment

8.1 Modules

- **common** *(Section 9, p. 18)*
- **env** *(Section 10, p. 25)*
- **fairness**: The result ranges from 1/n (worst case) to 1 (best case), and it is maximum when all users receive the same allocation.
  (Section 11, p. 27)
- **generic_ap**: Environment implementation (abstract class) that represents the experiment using Video This class implements the basic functions to control the APs, but it does not implement the QoE
  (Section 12, p. 28)
- **gini**: Calculate the global reward penalized using the gini_coeficient coefficient of a numpy array of data.
  (Section 13, p. 33)
- **grid_world**: Create a grid world
  (Section 14, p. 34)
- **hossfeld**: The hossfeld index ranges from 0 (worst case) to 1 (best case), and it is maximum when all users receive the same allocation (homogeneity).
  (Section 15, p. 37)
- **interface_env**: this defines the interface of the environment class environment all methods here should be implemented
  (Section 16, p. 38)
- **qoe_ap**: Environment implementation (concrete class) that represents the experiment using Video and QoE, where QoE is calculated using only AP parameters
  (Section 17, p. 40)
- **qoe_client**: this module calculates the MOS using only data from the client
  (Section 18, p. 43)
- **qoe_hybrid**: this module uses two sources to calculate the MOS: from AP and Client
  (Section 19, p. 48)
- **qoe_psnr**: this module calculates the MOS using only data from the client
  (Section 20, p. 51)
- **testEnv** *(Section 21, p. 53)*

8.2 Variables

Name	Description
__package__	**Value:** `'deepwifi.Environment'`

9 Module deepwifi.Environment.common

9.1 Functions

exec_cmd(*cmd*)

execute a shell command in the local computer

Parameters

`cmd`: command to be executed

exec_ssh(*host*, *cmd*)

kill_aps(*aps*, *kill_file*=`'kill.sh'`)

kill_stas(*stas*, *kill_file*=`'kill_sta.sh'`)

change_channel_hostapd(*aps*, *channels*)

start_hostapd(*aps*, *ids*, *conf_file*=`'hostapd.conf'`)

save_hostapd_config(*ap*, *run_file*=`'run.sh'`, *conf_file*=`'hostapd.conf'`, *kill_file*=`'kill.sh'`, *passphrase*=`'winet3014atm'`, *activate_get_set_server*=`False`)

create hostapd.conf

Parameters

`ap`: list[ap_config] contains a list of the aps' configuration parameters

`run_file`: the run.sh script filename

`conf_file`: the hostapd.conf configuration file for the ap's SSID

`kill_file`: the kill.sh script that stops all applications in the APs

save_wpa_config(*sta*, *ap*, *run_file*=`'run_sta.sh'`, *config_file*=`'wpa_supplicant.conf'`, *kill_file*=`'kill_sta.sh'`, *restart_file*=`'restart.sh'`, *ffox_file*=`'ffox.sh'`, *restart_ffox*=`5`, *browser*=`'opera'`, *passphrase*=`'winet3014atm'`)

create the wpa_supplicant.conf file for the designated sta

Parameters

`ap:` list[sta_config] contains a list of each station's configuration parameters

`ap:` list[ap_config] contains a list of each ap's configuration parameters

`run_file:` the run.sh script filename

`conf_file:` the wpa_supplicant.conf the create the connection to the correct AP

`kill_file:` the kill.sh script that stops all applications in the stations

Return Value

the wpa_supplicant.conf name

run_station(*sta*, *_id*=`''`, *run_file*=`'run_sta.sh'`)

call the run.sh script to run the applications in the STA

run_hostapd(*ap*, *_id*=`''`, *run_file*=`'run.sh'`)

calls the AP, and starts the hostapd

ap_is_running(*ap*)

calls the AP, and verifies if hostapd is running

sta_is_running(*sta*, *browser*=`'opera'`)

calls the STA, and verifies if wpa_supplicant is running

conf_stas(*aps*, *stas*, *restart_ffox*, *browser*)

conf_aps(*aps*)

start_devices(*aps*, *stas*, *max_retries*=`3`, *sleep_interval*=`10`, *_id*=`''`, *kill_ap*=`'kill.sh'`, *kill_sta*=`'kill_sta.sh'`, *browser*=`'opera'`)

Parameters

`max_retries`: number of retries, before reseting the device

`sleep_interval`: number of seconds the execution is suspended before retrying

reboot_devices(*devices*)

reboot the devices to get a clean slate

Parameters

`devices`: contains the hostname of each device

(type=list(namedtuple))

run_nodejs(*dir_*=`'/home/h3dema/Devel/server.js'`)

create server to collect browser data

Parameters

`dir_`: directory where the nodejs program is installed (from https://github.com/h3dema/server.js.git)

is_runnning_get_set_server()

run_get_set_server(*_id*, *dir_*=`'/home/h3dema/Devel/command_ap'`, *log_dir*=`'/home/h3dema/Devel/deepwifi/logs'`)

kill_get_set_server()

9.2 Variables

Name	Description
LOG	**Value:** `<logging.Logger object>`
aps	**Value:** `[AP(id=1, name='gnu-nb3', port=8080, iface='wlan0', mac='...`
stas	**Value:** `[Sta(id=11, name='cloud', iface='wlan0', mac='00:18:e7:7c...`
TEMPLATE_AP_START	**Value:** `'echo "Starting hostapd"\nT="‘hostname‘-{id}"\nLOG="$OUTP...`
HOSTAPD_FILE	**Value:** `'#This configuration file goes to {host}\ninterface={ifac...`

continued on next page

Name	Description
TEMPLATE_AP	**Value:** `’#!/bin/bash\n#\n# This scripts should run in {host}\n#\n...`
TEMPLATE_KILL_AP	**Value:** `’#!/bin/bash\nsudo pkill hostapd\nprocs=‘ps axf | grep no...`
WPA_FILE	**Value:** `’# This configuration file run in {host}\nctrl_interface=...`
TEMPLATE_STATION	**Value:** `’#!/bin/bash\n#\n# This configuration file belongs to {ho...`
TEMPLATE_FFOX	**Value:** `’#!/bin/bash\nBROWSER="{browser}"\nif [ "$#" -ne 1 ]; the...`
RESTART_FFOX	**Value:** `’#!/bin/bash\n#\nif [ "$#" -ne 1 ]; then\n echo "using...`
SITE_DASH	**Value:** `’http://150.164.10.51’`
TEMPLATE_KILL_ST-A	**Value:** `’#!/bin/bash\nsudo pkill wpa_supplicant\nsudo pkill Xvfb\...`
__package__	**Value:** `’deepwifi.Environment’`

9.3 Class AP_Config

object
    tuple
        deepwifi.Environment.common.AP_Config

AP(id, name, port, iface, mac, SSID, IP, initial_channel, initial_txpower)

9.3.1 Methods

__getnewargs__(*self*)

Return self as a plain tuple. Used by copy and pickle.

Overrides: tuple.__getnewargs__

__getstate__(*self*)

Exclude the OrderedDict from pickling

__new__(*_cls*, *id*, *name*, *port*, *iface*, *mac*, *SSID*, *IP*, *initial_channel*, *initial_txpower*)

Create new instance of AP(id, name, port, iface, mac, SSID, IP, initial_channel, initial_txpower)

Return Value

a new object with type S, a subtype of T

Overrides: object.__new__

__repr__(*self*)

Return a nicely formatted representation string

Overrides: object.__repr__

Inherited from tuple

__add__(), __contains__(), __eq__(), __ge__(), __getattribute__(), __getitem__(), __getslice__(), __gt__(), __hash__(), __iter__(), __le__(), __len__(), __lt__(), __mul__(), __ne__(), __rmul__(), count(), index()

Inherited from object

__delattr__(), __format__(), __init__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

9.3.2 Properties

Name	Description
IP	Alias for field number 6
SSID	Alias for field number 5
id	Alias for field number 0
iface	Alias for field number 3
initial_channel	Alias for field number 7
initial_txpower	Alias for field number 8
mac	Alias for field number 4
name	Alias for field number 1
port	Alias for field number 2
Inherited from object __class__	

9.4 Class ClientsConfig

object
  tuple
    deepwifi.Environment.common.ClientsConfig

Sta(id, name, iface, mac, AP, SSID, IP, webpage)

9.4.1 Methods

__getnewargs__(*self*)

Return self as a plain tuple. Used by copy and pickle.

Overrides: tuple.__getnewargs__

__getstate__(*self*)

Exclude the OrderedDict from pickling

__new__(*_cls*, *id*, *name*, *iface*, *mac*, *AP*, *SSID*, *IP*, *webpage*)

Create new instance of Sta(id, name, iface, mac, AP, SSID, IP, webpage)

Return Value

a new object with type S, a subtype of T

Overrides: object.__new__

__repr__(*self*)

Return a nicely formatted representation string

Overrides: object.__repr__

Inherited from tuple

__add__(), __contains__(), __eq__(), __ge__(), __getattribute__(), __getitem__(), __getslice__(), __gt__(), __hash__(), __iter__(), __le__(), __len__(), __lt__(), __mul__(), __ne__(), __rmul__(), count(), index()

Inherited from object

__delattr__(), __format__(), __init__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

9.4.2 Properties

Name	Description
AP	Alias for field number 4
IP	Alias for field number 6
SSID	Alias for field number 5
id	Alias for field number 0
iface	Alias for field number 2
mac	Alias for field number 3
name	Alias for field number 1
webpage	Alias for field number 7
Inherited from object ___class___	

10 Module deepwifi.Environment.env

10.1 Class environment

object

deepwifi.Environment.interface_env.Interface_Env

deepwifi.Environment.env.environment

10.1.1 Methods

__init__(*self*, *aps*)

x.__init__(...) initializes x; see help(type(x)) for signature

Parameters

`aps`: list of dictionary {id: int, ssh_user: str, shh_ip: string}

Overrides: object.__init__

ready(*self*)

rewards(*self*)

act(*self*, *actions*)

done(*self*)

by defaut don't finish overwrite if necessary

Overrides: deepwifi.Environment.interface_env.Interface_Env.done

reward(*self*, *curr_state*, ***kwargs*)

receives the current state, probes the environment and returns the reward

Parameters

`curr_state`: the current state

Return Value

a float number representing the reward in this state

(type=float)

Overrides: deepwifi.Environment.interface_env.Interface_Env.reward

valid_actions(*self*, *state*=`None`)

must be implemented in descendent

Return Value

a list of all valid actions

(type=list(int))

Overrides: deepwifi.Environment.interface_env.Interface_Env.valid_actions extit(inherited documentation)

get_states(*self*)

return a list of values that represents the state of each AP

Overrides: deepwifi.Environment.interface_env.Interface_Env.get_states

make_step(*self*, *action*)

must be implemented in descendent

Parameters

`action`: is a (list of) number (int) that represents the action to be taken

Return Value

next_state: a (list of) number (int) that represents the next state (one for each AP)

Overrides: deepwifi.Environment.interface_env.Interface_Env.make_step

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

10.1.2 Properties

Name	Description
Inherited from deepwifi.Environment.interface_env.Interface_Env (Section 16.2) action_size, state_dim, state_size	
Inherited from object ___class___	

11 Module deepwifi.Environment.fairness

The result ranges from 1/n (worst case) to 1 (best case), and it is maximum when all users receive the same allocation.

References: * https://en.wikipedia.org/wiki/Fairness_measure

11.1 Functions

fairness_index(*data*, *epsilon*=`1e-18`)

Return Value

the jain fairness index, bounded between 0 and 1 0 means the data is homogeneous (all values are equal) and 1 means the data is different

reward_jain(*data*)

11.2 Variables

Name	Description
___package___	**Value:** `'deepwifi.Environment'`

12 Module deepwifi.Environment.generic_ap

Environment implementation (abstract class) that represents the experiment using Video This class implements the basic functions to control the APs, but it does not implement the QoE

12.1 Functions

decode_txpower(*t*)

convert the data in info['txpower'] which is, for example, '15.00 dBm' into 15.0

Return Value

the value of the tx power

(type=float)

12.2 Class Generic_AP

object

deepwifi.Environment.interface_env.Interface_Env

deepwifi.Environment.generic_ap.Generic_A

12.2.1 Methods

__init__(*self*, *aps*, *model_filename*, *mac_mapping*=`{}`, *log_level*=`logging.DEBUG`, *log_name*=`'AP Controller'`, *wait_for_states*=`10`, *execute_action*=`False`)

initialize the environment

Parameters

`aps:`	list of aps controlled in the experiment
`model_filename:`	name of the file that contains the trained model *(type=str)*
`mac_mapping:`	a dictionary that maps the hostname to its mac address
`execute_action:`	if True send the selected actions to the devices

Overrides: object.__init__

command_ap(*self*, *server*, *port*, *iface*, *cmd*, *extra_params*=`None`)

```


@return: returns true if receive the response,
         also returns the data or an empty dict (if error)
@rtype bool, dict
```

restart_aps(*self*, *run_id*)

this is done because our ap sometimes crashes. the hostapd continues to run, but does not provide a channel

valid_actions(*self*, *state*=`None`)

```

return a list with all valid actions for a specific state,
    if state == None, return all possible states
@param state: current state
@return: list(int)
```

Return Value

a list of all valid actions

(type=list(int))

Overrides: deepwifi.Environment.interface_env.Interface_Env.valid_actions

one_hot(*self*, *channel*)

code the channel using one-hot encoding

Parameters

`channel`: *(type=int)*

Return Value

the channel hot encoded

(type=list(int))

get_states(*self*)

```
get the states, one for each AP
    the state contains:
    - ( #stations, ch1, ch2, ch3, ch4, ch5, ch6, ch7, ch8, ch9, ch10, ch11,
        tx_power, #num_neighbors, ch_noise_max, perc_phy_busy_time,
        sta_signal_avg,
        rec_bitrate_min, tx_byte_avg, rx_byte_avg )
@return: return the value that represent the state of all APs. Returns None if an e
```

Overrides: deepwifi.Environment.interface_env.Interface_Env.get_states

encode_action(*self*, *txpower*, *channel*)

Parameters

`action`: an integer that represents the action

Return Value

decoded values of txpower (1 to 15 dBm) and channel (1 to 11)

decode_action(*self*, *action*)

Parameters

`action`: an integer that represents the action

Return Value

decoded values of txpower (1 to 15 dBm) and channel (1 to 11)

setup_device(*self*, *ap*, *txpower*, *channel*)

change the tx power and the ap's channel

Parameters

`ap`: the ap

`txpower`: tx power (from 1 to 15 dBm)

`channel`: the 2.4GHz channel number (1 to 11)

make_step(*self*, *actions*, *retries*=5)

send commands to aps

Parameters

`actions`: is a list of number (int) that represents the action to be taken for each AP

(type=list(int))

`retries`: number of times this function tries to get the next_state from the devices, if unsuccessful then return None in next_state

Return Value

next_state: a (list of) number (int) that represents the next state

(type=list(int), float)

Overrides: deepwifi.Environment.interface_env.Interface_Env.make_step

get_model(*self*, *model_filename*)

called in the init() code to read the model from a file

Parameters

`model_filename`: name of the file that contains the trained model

(type=str)

Return Value

the model

Inherited from deepwifi.Environment.interface_env.Interface_Env(Section 16.2)

reward()

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

12.2.2 Properties

Name	Description
Inherited from deepwifi.Environment.interface_env.Interface_Env (Section 16.2) action_size, done, state_dim, state_size	
Inherited from object ___class___	

12.2.3 Class Variables

Name	Description
NUM_CHANNELS	**Value:** `11`
NUM_TXPOWER_LEVELS	**Value:** `15`
DEFAULT_C	**Value:** `0.4`

13 Module deepwifi.Environment.gini

```
Calculate the global reward penalized using the gini_coeficient coefficient of a numpy a
We use this value to provide a reward that account for the better distribution of MOS am
It substitutes the baseline that uses the average of MOS

based on the statsdirect equation shown in
    http://www.statsdirect.com/help/default.htm#nonparametric_methods/gini_coeficient.ht

Other references:
* https://en.wikipedia.org/wiki/Gini_coefficient
* https://towardsdatascience.com/gini_coeficient-coefficient-and-lorenz-curve-f19bb8f46d
* DORFMAN, Robert. A formula for the gini_coeficient coefficient. The review of economic
```

13.1 Functions

scale_minmax(*data*)

gini_coeficient(*user_data*, *epsilon*=`1e-18`)

Calculate the gini_coeficient coefficient of a numpy data. All values are treated equally, the values are first placed in ascending order, such that each x has rank i,

Return Value

0 if the data is homogeneous or 1 if the data

reward_gini(*data*)

13.2 Variables

Name	Description
___package___	**Value:** `'deepwifi.Environment'`

14 Module deepwifi.Environment.grid_world

```
Create a grid world

       x  0     1  ... dim_x-1
 y       +----+----+----+----+
 0       |    |    |    |    |
         +----+----+----+----+
 1       |    |    |    |    |
         +----+----+----+----+
 ..      |    |    |    |    |
         +----+----+----+----+
dim_y-1  |    |    |    |    |
         +----+----+----+----+
```

14.1 Variables

Name	Description
LOG	**Value:** `<logging.Logger object>`
UP	**Value:** `0`
DOWN	**Value:** `1`
LEFT	**Value:** `2`
RIGHT	**Value:** `3`
__package__	**Value:** `'deepwifi.Environment'`

14.2 Class grid_world

```
object ─┐
        │
Environment.interface_env.Interface_Env ─┐
                                         │
                                         deepwifi.Environment.grid_world.grid_world
```

14.2.1 Methods

___init___(*self*, *dim_x*, *dim_y*)

x.___init___(...) initializes x; see help(type(x)) for signature

Parameters

`aps:` list of dictionary {id: int, ssh_user: str, shh_ip: string}

Overrides: object.___init___

reward(*self*, *curr_state*, ***kwargs*)

minus the number of steps to the objective

Return Value

the reward

(type=float)

Overrides: Environment.interface_env.Interface_Env.reward

valid_actions(*self*, *state*=`None`)

must be implemented in descendent

Return Value

a list of all valid actions

(type=list(int))

Overrides: Environment.interface_env.Interface_Env.valid_actions
extit(inherited documentation)

get_states(*self*)

must be implemented in descendent should return a (list of) number (int) that represents the current state

Overrides: Environment.interface_env.Interface_Env.get_states
extit(inherited documentation)

make_step(*self*, *action*)

must be implemented in descendent

Parameters

`action`: is a (list of) number (int) that represents the action to be taken

Return Value

next_state: a (list of) number (int) that represents the next state

(type=list(int), list(int), float)

Overrides: Environment.interface_env.Interface_Env.make_step

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

14.2.2 Properties

Name	Description
done	returns true if the objective is achieved
Inherited from Environment.interface_env.Interface_Env action_size, state_dim, state_size	
Inherited from object ___class___	

15 Module deepwifi.Environment.hossfeld

The hossfeld index ranges from 0 (worst case) to 1 (best case), and it is maximum when all users receive the same allocation (homogeneity).

References: * https://en.wikipedia.org/wiki/Fairness_measure

15.1 Functions

hossfeld_index(*data*, *L*=`1`, *H*=`5`)

1 indicating perfect QoE fairness - all users experience the same quality. 0 indicates total unfairness, e.g. 50% of users experience highest QoE H and 50% experience lowest QoE L.

Parameters

`L`: lower bound in data, for MOS = 1

`H`: upper bound in data, for MOS = 5

Return Value

the Hostfeld fairness index, bounded between 0 and 1

reward_hossfeld(*data*, *C*=`0.4`)

gets a compromise between the average of the reward and the Hossfeld Index

Parameters

`data`: array with MOS values

Return Value

the reward for each entry

15.2 Variables

Name	Description
___package___	**Value:** `'deepwifi.Environment'`

16 Module deepwifi.Environment.interface_env

this defines the interface of the environment class environment all methods here should be implemented

16.1 Variables

Name	Description
__package__	**Value:** `'deepwifi.Environment'`

16.2 Class Interface_Env

object ─┐
deepwifi.Environment.interface_env.Interface_Env

16.2.1 Methods

__init__(*self*, *LOG_NAME*=`'environment'`, *log_level*=`10`)

x.__init__(...) initializes x; see help(type(x)) for signature

Parameters

`LOG_NAME`: the name assigned to the logger

Overrides: object.__init__

reward(*self*, ***kwargs*)

should return a real number

Return Value

the reward

(type=float)

valid_actions(*self*, *state*=`None`)

must be implemented in descendent

Return Value

a list of all valid actions

(type=list(int))

get_states(*self*)

must be implemented in descendent should return a (list of) number (int) that represents the current state

make_step(*self*, *action*)

must be implemented in descendent

Parameters

`action`: is a (list of) number (int) that represents the action to be taken

Return Value

next_state: a (list of) number (int) that represents the next state

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

16.2.2 Properties

Name	Description
done	returns true if the objective is achieved
state_size	this method is valid for discrete state space, where you can enumerate the total number of states *(type=int)*

state_dim	`the number of dimensions.`
	`    For example, a discrete 1-D space can have state_size`
	`    state_dim = 1 (because is 1D)`
	
	`@return: the number of dimensions in the state space`
	`@rtype: int`

action_size	number of actions *(type=int)*
Inherited from object __class__	

17 Module deepwifi.Environment.qoe_ap

```
Environment implementation (concrete class) that represents the experiment using Video a
QoE is calculated using only AP parameters


Uses a pre-trained RNN model to estimate the MOS, which consists of:
* Bit Error Rate (BER): variation of the Bit Error Rate (BER) that can cause the MAC fra
* frame aggregation: A-MPDU (MAC Protocol Data Unit) aggregation, allows many MAC frames
* number of competing stations: performance of the wireless network degrades withincreas
* traffic load: percentage of traffic over the maximum throughput of the interface
data needed: 'TX-Failed_*', 'TX-Pkts-All_*', 'AMPDUs Completed_*' --> xmit
             'tx_bytes' --> ifconfig
             'num_stations' --> iw station dump

definitions:
'FER' = 'txf_detrend' / ('txf_detrend' + 'txp_detrend')
'AMPDU' = np.sum('AMPDUs Completed_*')
'traffic_load' = 'tx_bytes_detrend' / 'tx_bytes'.max(iface)
```

17.1 Class QoE_AP

object

deepwifi.Environment.interface_env.Interface_Env

deepwifi.Environment.generic_ap.Generic_AP

deepwifi.Environment.qoe_ap.QoE_AP

defines the QoE as MOS_AP

17.1.1 Methods

reward(*self*, ***kwargs*)

check the MOS of each station

Parameters

`curr_state`: current state

Return Value

the reward

(type=float)

Overrides: deepwifi.Environment.interface_env.Interface_Env.reward

get_model(*self*, *model_filename*=`'model-ap.p'`)

get the trained model from a file

Parameters

`model_filename`: the name of the file containing the trained model

(type=str)

Return Value

the RNN model trained

Overrides: deepwifi.Environment.generic_ap.Generic_AP.get_model

Inherited from deepwifi.Environment.generic_ap.Generic_AP(Section 12.2)

__init__(), command_ap(), decode_action(), encode_action(), get_states(), make_step(), one_hot(), restart_aps(), setup_device(), valid_actions()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

17.1.2 Properties

Name	Description
Inherited from deepwifi.Environment.interface_env.Interface_Env (Section 16.2) action_size, done, state_dim, state_size	
Inherited from object __class__	

17.1.3 Class Variables

Name	**Description**
Inherited from deepwifi.Environment.generic_ap.Generic_AP (Section 12.2) DEFAULT_C, NUM_CHANNELS, NUM_TXPOWER_LEVELS	

18 Module deepwifi.Environment.qoe__client

```
this module calculates the MOS using only data from the client

* rt_1, rt
    * r[t] = reportedBitrate in time [t] / max_bitrate

* srt = not_running_time / (not_running_time + execution_time)
```

18.1 Class mos__client__abstract

object ─┐
deepwifi.Environment.qoe__client.mos__client__abstract

18.1.1 Methods

predict(*self*, *X*)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___init___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

18.1.2 Properties

Name	Description
Inherited from object ___class___	

18.2 Class mos__client__local

object ─┐
deepwifi.Environment.qoe__client.mos__client__abstract ─┐
deepwifi.Environment.qoe__client.mos__cli

codes the best regression obtained see MOS_CLIENT/Generate QoE Metric -Log.ipynb for the results

# data R_t = Selected bitrate for t-th chunk / Maximum bitrate R_t = Selected bitrate for (t-1)-th chunk / Maximum bitrate SR_t = Stalling length to play out the t-th chunk / (Stalling length to play out the t-th chunk + Time length of the t-th chunk)

# Equation QoE(R_{t-1}, R_{t}, SR_t) = a0 + a1 [log(R_t) + log(R_t1)] + a2 * SR_t + a3 | log(R_t) - log(R_t1) |

18.2.1 Methods

__init__(*self*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

predict(*self*, *X*)

finds the MOS for each entry (line) in X

Parameters

`X`: np.array[:, 3]. Contains three columns: R_t, R_t1, SR

Return Value

a list of rewards, one for each line in X

Overrides: deepwifi.Environment.qoe_client.mos_client_abstract.predict

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

18.2.2 Properties

Name	Description
Inherited from object	
__class__	

18.3 Class mos_client

object

deepwifi.Environment.qoe_client.mos_client_abstract

deepwifi.Environment.qoe_client.mos_cli

# X R_t = Selected bitrate for t-th chunk / Maximum bitrate R_t = Selected bitrate for (t-1)-th chunk / Maximum bitrate SR_t = Stalling length to play out the t-th chunk / (Stalling length to play out the t-th chunk + Time length of the t-th chunk)

18.3.1 Methods

__init__(*self*, *model*, *kernel*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

predict(*self*, *X*)

finds the MOS for each entry (line) in X

Parameters

`X`: np.array[:, 3]. Contains three columns: R_t, R_t1, SR

Return Value

a list of rewards, one for each line in X

Overrides: deepwifi.Environment.qoe_client.mos_client_abstract.predict

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

18.3.2 Properties

Name	Description
Inherited from object __class__	

18.4 Class QoE_Client

object

deepwifi.Environment.interface_env.Interface_Env

deepwifi.Environment.generic_ap.Generic_AP

deepwifi.Environment.qoe_client.QoE_C

defines the QoE as MOS_CLIENT

18.4.1 Methods

get_rs(*self*, *data*)

get_mos_from_aps(*self*)

it considers that each AP collects from the stations their data

get_mos_from_localhost(*self*)

it considers that the controller collects data from all the stations

reward(*self*, ***kwargs*)

check the MOS of each station

Parameters

`curr_state`: current state

Return Value

the reward

(type=float)

Overrides: deepwifi.Environment.interface_env.Interface_Env.reward

get_model(*self*, ***kwargs*)

The model is hard-coded in mos_client()

Parameters

`model_filename`: name of the file that contains the trained model

Return Value

the model object

Overrides: deepwifi.Environment.generic_ap.Generic_AP.get_model

Inherited from deepwifi.Environment.generic_ap.Generic_AP(Section 12.2)

__init__(), command_ap(), decode_action(), encode_action(), get_states(), make_step(), one_hot(), restart_aps(), setup_device(), valid_actions()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

18.4.2 Properties

Name	Description
Inherited from deepwifi.Environment.interface_env.Interface_Env (Section 16.2) action_size, done, state_dim, state_size	
Inherited from object __class__	

18.4.3 Class Variables

Name	Description
Inherited from deepwifi.Environment.generic_ap.Generic_AP (Section 12.2) DEFAULT_C, NUM_CHANNELS, NUM_TXPOWER_LEVELS	

19 Module deepwifi.Environment.qoe_hybrid

```
this module uses two sources to calculate the MOS: from AP and Client

* From client:
    * FR = reportedBitrate
    * Frame Loss = droppedFPS

    effectiveBitrate = (reportedBitrate * execution_time) / (execution_time + not_runnin
    effectiveBitrate = effectiveBitrate / reportedBitrate']

* From AP:
    * loss rate (PLR)
        packets = | rx_packets[t] - rx_packets[t-1] |
        PLR = rxdrop / (packets + rxdrop)

    * send bit rate (SBR)
        SBR = tx_bitrate / maximum tx_bitrate
```

19.1 Class mos_hybrid

object ─┐

deepwifi.Environment.qoe_hybrid.mos_hybrid

codes the best regression obtained

19.1.1 Methods

predict(*self*, *X*)

finds the MOS for each entry (line) in X

Parameters

`X`: np.array[:, 3]. Contains three columns: fr, sbr, plr

Return Value

a list of rewards, one for each line in X

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

19.1.2 Properties

Name	Description
Inherited from object	
___class___	

19.2 Class QoE_Hybrid

object

deepwifi.Environment.interface_env.Interface_Env

deepwifi.Environment.generic_ap.Generic_AP

deepwifi.Environment.qoe_hybrid.QoE_

defines the QoE as MOS_HYBRID

19.2.1 Methods

reward(*self*, ***kwargs*)

check the MOS of each station using command_ap module

Parameters

`curr_state`: current state

Return Value

the reward

(type=float)

Overrides: deepwifi.Environment.interface_env.Interface_Env.reward

get_model(*self*, ***kwargs*)

get the module from the file

Parameters

`model_filename`: name of the file that contains the trained model

Return Value

the model object that constains .fit() and .predict()

Overrides: deepwifi.Environment.generic_ap.Generic_AP.get_model

Inherited from deepwifi.Environment.generic_ap.Generic_AP(Section 12.2)

___init___(), command_ap(), decode_action(), encode_action(), get_states(), make_step(), one_hot(), restart_aps(), setup_device(), valid_actions()

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

19.2.2 Properties

Name	Description
Inherited from deepwifi.Environment.interface_env.Interface_Env (Section 16.2) action_size, done, state_dim, state_size	
Inherited from object ___class___	

19.2.3 Class Variables

Name	Description
Inherited from deepwifi.Environment.generic_ap.Generic_AP (Section 12.2) DEFAULT_C, NUM_CHANNELS, NUM_TXPOWER_LEVELS	

20 Module deepwifi.Environment.qoe_psnr

```
this module calculates the MOS using only data from the client

* rt_1, rt
    * r[t] = reportedBitrate in time [t] / max_bitrate

* srt = not_running_time / (not_running_time + execution_time)
```

20.1 Class QoE_PSNR

object

deepwifi.Environment.interface_env.Interface_Env

deepwifi.Environment.generic_ap.Generic_AP

deepwifi.Environment.qoe_client.QoE_Client

deepwifi.Environment.qoe_psnr.QoE

defines the QoE using PSNR (MOS) received from the client

20.1.1 Methods

get_mos_from_aps(*self*)

it considers that each AP collects from the stations their data

Overrides: deepwifi.Environment.qoe_client.QoE_Client.get_mos_from_aps

get_mos_from_localhost(*self*)

it considers that the controller collects data from all the stations

Overrides: deepwifi.Environment.qoe_client.QoE_Client.get_mos_from_localhost

get_model(*self*, ***kwargs*)

Uses the MOS from the client, thus there is no model.

Parameters

`model_filename`: name of the file that contains the trained model

Return Value

the model object

Overrides: deepwifi.Environment.generic_ap.Generic_AP.get_model

Inherited from deepwifi.Environment.qoe_client.QoE_Client(Section 18.4)

get_rs(), reward()

Inherited from deepwifi.Environment.generic_ap.Generic_AP(Section 12.2)

__init__(), command_ap(), decode_action(), encode_action(), get_states(), make_step(), one_hot(), restart_aps(), setup_device(), valid_actions()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

20.1.2 Properties

Name	Description
Inherited from deepwifi.Environment.interface_env.Interface_Env (Section 16.2) action_size, done, state_dim, state_size	
Inherited from object __class__	

20.1.3 Class Variables

Name	Description
Inherited from deepwifi.Environment.generic_ap.Generic_AP (Section 12.2) DEFAULT_C, NUM_CHANNELS, NUM_TXPOWER_LEVELS	

21 Module deepwifi.Environment.testEnv

21.1 Variables

Name	Description
STATES	**Value:** `[[[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15.0, 1, 81.0, 1.0...`
REWARD	**Value:** `[[1.0, 3.0], [2.0, 1.0], [1.0, 1.0], [1.0, 1.0], [2.0, 1....`
MAX_ITERATIONS	**Value:** `len(REWARD)-1`

21.2 Class test_qoe

object

deepwifi.Environment.interface_env.Interface_Env

deepwifi.Environment.generic_ap.Generic_AP

deepwifi.Environment.testEnv.test_qoe

21.2.1 Methods

__init__(*self*, *aps*, *model_filename*, *mac_mapping*=`{}`, *log_level*=`logging.DEBUG`)

initialize the environment

Parameters

`aps:`	list of aps controlled in the experiment
`model_filename:`	name of the file that contains the trained model *(type=str)*

Overrides: object.__init__

get_states(*self*)

```
get the states, one for each AP
    the state contains:
    - ( #stations, ch1, ch2, ch3, ch4, ch5, ch6, ch7, ch8, ch9, ch10, ch11,
        tx_power, #num_neighbors, ch_noise_max, perc_phy_busy_time,
        sta_signal_avg,
        rec_bitrate_min, tx_byte_avg, rx_byte_avg )
@return: return the value that represent the state of all APs. Returns None if an e
```

Overrides: deepwifi.Environment.interface_env.Interface_Env.get_states extit(inherited documentation)

make_step(*self*, *actions*)

send commands to aps

Parameters

`actions`: is a list of number (int) that represents the action to be taken for each AP

`retries`: number of times this function tries to get the next_state from the devices, if unsuccessful then return None in next_state

Return Value

next_state: a (list of) number (int) that represents the next state

(type=list(int), float)

Overrides: deepwifi.Environment.interface_env.Interface_Env.make_step extit(inherited documentation)

get_model(*self*, *model_filename*)

called in the init() code to read the model from a file

Parameters

`model_filename`: name of the file that contains the trained model

Return Value

the model

Overrides: deepwifi.Environment.generic_ap.Generic_AP.get_model extit(inherited documentation)

done(*self*)

returns true if the objective is achieved

Overrides: deepwifi.Environment.interface_env.Interface_Env.done

Inherited from deepwifi.Environment.generic_ap.Generic_AP(Section 12.2)

command_ap(), decode_action(), encode_action(), one_hot(), restart_aps(), setup_device(), valid_actions()

Inherited from deepwifi.Environment.interface_env.Interface_Env(Section 16.2)

reward()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

21.2.2 Properties

Name	Description
Inherited from deepwifi.Environment.interface_env.Interface_Env (Section 16.2) action_size, state_dim, state_size	
Inherited from object __class__	

21.2.3 Class Variables

Name	Description
Inherited from deepwifi.Environment.generic_ap.Generic_AP (Section 12.2) DEFAULT_C, NUM_CHANNELS, NUM_TXPOWER_LEVELS	

22 Package deepwifi.MAB

22.1 Modules

- **mab**: This is a module that runs RL method to control wifi devices.
  (Section 23, p. 57)

22.2 Variables

Name	Description
__package__	**Value:** `None`

23 Module deepwifi.MAB.mab

This is a module that runs RL method to control wifi devices.

To run this module use: python mab.py

23.1 Variables

Name	Description
__package__	**Value:** `None`

24 Package deepwifi.Memory

24.1 Modules

- **memory**: This module defines the interface for the replay memory buffer
  (Section 25, p. 59)
- **replay**: This module implements the replay memory buffer used in DQL
  (Section 26, p. 62)
- **replay_tuple**: This module implements the replay memory buffer used in DQL with multiple timesteps and multiple APs
  (Section 27, p. 64)

24.2 Variables

Name	Description
__package__	**Value:** `None`

25 Module deepwifi.Memory.memory

This module defines the interface for the replay memory buffer

25.1 Variables

Name	Description
__package__	**Value:** `'deepwifi.Memory'`

25.2 Class Transition

object
tuple
deepwifi.Memory.memory.Transition

Transition(state, action, next_state, reward)

25.2.1 Methods

__getnewargs__(*self*)

Return self as a plain tuple. Used by copy and pickle.

Overrides: tuple.__getnewargs__

__getstate__(*self*)

Exclude the OrderedDict from pickling

__new__(*_cls*, *state*, *action*, *next_state*, *reward*)

Create new instance of Transition(state, action, next_state, reward)

Return Value

a new object with type S, a subtype of T

Overrides: object.__new__

__repr__(*self*)

Return a nicely formatted representation string

Overrides: object.__repr__

Inherited from tuple

__add__(), __contains__(), __eq__(), __ge__(), __getattribute__(), __getitem__(), __getslice__(), __gt__(), __hash__(), __iter__(), __le__(), __len__(), __lt__(), __mul__(), __ne__(), __rmul__(), count(), index()

Inherited from object

__delattr__(), __format__(), __init__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

25.2.2 Properties

Name	Description
action	Alias for field number 1
next_state	Alias for field number 2
reward	Alias for field number 3
state	Alias for field number 0
Inherited from object __class__	

25.3 Class Memory

object ─┐
deepwifi.Memory.memory.Memory

25.3.1 Methods

__init__(*self*, *log_level*=10)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

push(*self*, **args*)

sample(*self*, *batch_size*)

__len__(*self*)

return the current number of elements stored in the memory

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

25.3.2 Properties

Name	Description
Inherited from object __class__	

26 Module deepwifi.Memory.replay

This module implements the replay memory buffer used in DQL

26.1 Variables

Name	Description
__package__	**Value:** `'deepwifi.Memory'`

26.2 Class ReplayMemory

object
Memory.memory.Memory
deepwifi.Memory.replay.ReplayMemory

26.2.1 Methods

__init__(*self*, *capacity*)

creates the memory

Parameters

`capacity`: size of the memory

Overrides: object.__init__

push(*self*, **args*)

Saves a transition

Parameters

`args`: contain the data that should be saved in the memory: state, action, next_state, reward

Overrides: Memory.memory.Memory.push

sample(*self*, *batch_size*)

Parameters

`batch_size`: number of elements that should be returned from the memory, if the memory does not contains this many elements, returns the whole memory

Return Value

a batch sample. this is a list[[Transition], [Transition]]

Overrides: Memory.memory.Memory.sample

__len__(*self*)

return the current number of elements stored in the memory

Overrides: Memory.memory.Memory.__len__

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

26.2.2 Properties

Name	Description
Inherited from object __class__	

27 Module deepwifi.Memory.replay_tuple

This module implements the replay memory buffer used in DQL with multiple timesteps and multiple APs

27.1 Variables

Name	Description
__package__	**Value:** `'deepwifi.Memory'`

27.2 Class ReplayMemoryTuple

object

Memory.memory.Memory

deepwifi.Memory.replay_tuple.ReplayMemoryTuple

27.2.1 Methods

__init__(*self*, *capacity*, *timesteps*=`1`, *num_devices*=`1`, *log_level*=`10`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

push(*self*, **args*)

```
Saves a transition for each controlled device
eg. ReplayMemoryTuple.push(states, actions, next_states, rewards)
@param args: contain a tuple that should be saved in the memory
             the lines in args should contain: state, action, next_state, reward
             notice then that len(args) ==  4
             e.g.
             args = ([[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15.0, 1, 81.0, 1.00514839
                      [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15.0, 1, 82.0, 0.94031465
                      ],
                      [71, 75],
                      [[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15.0, 1, 81.0, 1.0051625
                      [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15.0, 1, 82.0, 0.94031755
                      ],
                      [1.0, 3.0])
```

Overrides: Memory.memory.Memory.push

sample(*self*, *batch_size*)

Parameters

`batch_size`: number of elements that should be returned from the memory, if the memory does not contains this many elements, returns the whole memory

Return Value

a batch sample

Overrides: Memory.memory.Memory.sample

__len__(*self*)

return the current number of elements stored in the memory

Return Value

the number of elements and devices

(type=(int, int))

Overrides: Memory.memory.Memory.__len__

save(*self*, *filename*)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

27.2.2 Properties

Name	Description
Inherited from object __class__	

28 Package deepwifi.TCN

28.1 Modules

- **tcnn**: This module implements Temporal Convolutional Network *(Section 29, p. 68)*
- **weightnorm** *(Section 30, p. 72)*

28.2 Variables

Name	Description
___package___	**Value:** `None`

29 Module deepwifi.TCN.tcnn

```
This module implements Temporal Convolutional Network


Making the TCN architecture non-causal allows it to take the future into consideration t
However, it is not anymore suitable for real-time applications.
To use a non-causal TCN, specify padding='valid' or padding='same' when initializing the


code based on:
    * https://github.com/philipperemy/keras-tcn
    * https://github.com/locuslab/TCN/
ref.:
    * BAI, Shaojie; KOLTER, J. Zico; KOLTUN, Vladlen.
      An empirical evaluation of generic convolutional and recurrent networks for sequen
      arXiv preprint arXiv:1803.01271, 2018.
      https://arxiv.org/pdf/1803.01271

    * OORD, Aaron van den et al.
      Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499, 2016.
      https://arxiv.org/pdf/1609.03499.pdf
```

29.1 Functions

residual_block(*x*, *dilation_rate*, *nb_filters*, *kernel_size*, *padding*, *dropout_rate*=0, *activation*='relu', *kernel_initializer*='he_normal', *use_batch_norm*=False)

```
Defines the residual block for the WaveNet TCN

:param x: The previous layer in the model
:param dilation_rate: The dilation power of 2 we are using for this residual block
:param nb_filters: The number of convolutional filters to use in this block
:param kernel_size: The size of the convolutional kernel
:param padding: The padding used in the convolutional layers, 'same' or 'causal'.
:param activation: The final activation used in o = Activation(x + F(x))
:param dropout_rate: Float between 0 and 1. Fraction of the input units to drop.
:param kernel_initializer: Initializer for the kernel weights matrix (Conv1D).
:param use_batch_norm: Whether to use batch normalization in the residual layers or

:return A tuple where the first element is the residual model layer, and the second
    is the skip connection.
```

process_dilations(*dilations*)

get_opt(*opt*, *lr*, *decay*=0.0)

```
Args:
    opt: Optimizer name.
    lr: Learning rate.
    decay: Learning rate decay over each update.
```

accuracy(*y_true*, *y_pred*)

compiled_tcn(*num_feat*, *num_classes*, *nb_filters*, *kernel_size*, *dilations*, *nb_stacks*, *max_len*, *padding*=`'causal'`, *use_skip_connections*=`True`, *return_sequences*=`True`, *regression*=`False`, *dropout_rate*=`0.05`, *name*=`'tcn'`, *kernel_initializer*=`'he_normal'`, *activation*=`'linear'`, *opt*=`'adam'`, *lr*=`0.002`, *decay*=`0.0`, *use_batch_norm*=`False`)

```
Creates a compiled TCN model for a given task (i.e. regression or classification).
Classification uses a sparse categorical loss. Please input class ids and not one-ho

Args:
    num_feat: The number of features of your input, i.e. the last dimension of: (bat
    num_classes: The size of the final dense layer, how many classes (or values) we
    nb_filters: The number of filters to use in the convolutional layers.
    kernel_size: The size of the kernel to use in each convolutional layer.
    dilations: The list of the dilations. Example is: [1, 2, 4, 8, 16, 32, 64].
    nb_stacks : The number of stacks of residual blocks to use.
    max_len: The maximum sequence length, use None if the sequence length is dynamic
    padding: The padding to use in the convolutional layers.
    use_skip_connections: Boolean. If we want to add skip connections from input to
    return_sequences: Boolean. Whether to return the last output in the output seque
    regression: Whether the output should be continuous or discrete.
    dropout_rate: Float between 0 and 1. Fraction of the input units to drop.
    activation: The activation used in the residual blocks o = Activation(x + F(x)).
    name: Name of the model. Useful when having multiple TCN.
    kernel_initializer: Initializer for the kernel weights matrix (Conv1D).
    opt: Optimizer name.
    lr: Learning rate.
    decay: Learning rate decay over each update.
    use_batch_norm: Whether to use batch normalization in the residual layers or not
Returns:
    A compiled keras TCN.
```

29.2 Variables

Name	Description
LOG	**Value:** `logging.getLogger('TCNN')`

29.3 Class TCN

```
Creates a TCN layer.
```

```
Input shape:
    A tensor of shape (batch_size, timesteps, input_dim).

Args:
    nb_filters: The number of filters to use in the convolutional layers.
    kernel_size: The size of the kernel to use in each convolutional layer.
    dilations: The list of the dilations. Example is: [1, 2, 4, 8, 16, 32, 64].
    nb_stacks : The number of stacks of residual blocks to use.
    padding: The padding to use in the convolutional layers, 'causal' or 'same'.
    use_skip_connections: Boolean. If we want to add skip connections from input to each
    return_sequences: Boolean. Whether to return the last output in the output sequence,
    activation: The activation used in the residual blocks o = Activation(x + F(x)).
    dropout_rate: Float between 0 and 1. Fraction of the input units to drop.
    name: Name of the model. Useful when having multiple TCN.
    kernel_initializer: Initializer for the kernel weights matrix (Conv1D).
    use_batch_norm: Whether to use batch normalization in the residual layers or not.

Returns:
    A TCN layer.
```

29.3.1 Methods

__init__(*self*, *nb_filters*=64, *kernel_size*=2, *nb_stacks*=1, *dilations*=[1,2,4,8,16,32], *padding*='causal', *use_skip_connections*=True, *dropout_rate*=0.0, *return_sequences*=False, *activation*='linear', *name*='tcn', *kernel_initializer*='he_normal', *use_batch_norm*=False)

__call__(*self*, *inputs*)

30 Module deepwifi.TCN.weightnorm

30.1 Functions

get_weightnorm_params_and_grads(*p*, *g*)

add_weightnorm_param_updates(*updates*, *new_V_param*, *new_g_param*, *W*, *V_scaler*)

data_based_init(*model*, *input*)

30.2 Class SGDWithWeightnorm

keras.optimizers.SGD

deepwifi.TCN.weightnorm.SGDWithWeightnorm

30.2.1 Methods

get_updates(*self*, *loss*, *params*)

30.3 Class AdamWithWeightnorm

keras.optimizers.Adam

deepwifi.TCN.weightnorm.AdamWithWeightnorm

30.3.1 Methods

get_updates(*self*, *loss*, *params*)

31 Module deepwifi.run_acs

```
Running
=======
There are two options:

a) creates the configuration files (hostapd.conf, and wpa_supplicant.conf), create the s
   copy the files to the APs and STAs
   and then runs the experiment
python3 run_acs.py --save-wpa-conf --save-hostapd-conf

b) just run it. This pressuposes that the configuration files are copied to the devices
python3 run_acs.py
```

31.1 Functions

reboot(*aps*, *stas*)

get_best_channel(*data*, *valid_channels*=`[2412,2417,2422,2427,2432,2437,2442,2447,2452,2457,2462]`)

1) calculate the average interference factor for each channel 'c' IF[c] = 10^(chan_nf/5) + (busy time - tx time) / (active time - tx time) * 2^(10^(chan_nf/10) + 10^(band_min_nf/10)) 2) return the channel with the lowest average

Parameters

`data`: list of survey data for one AP

Return Value

the best channel

get_action(*env*, *aps*, *interval*=`1`, *num_min_surveys*=`5`, *tx_power*=`10`)

decide based on the ACS index

run_acs(*aps*, *stas*, *env*, *interaction_interval*, *run_id*=`'1'`)

31.2 Variables

Name	Description
LOG	**Value:** `logging.getLogger('RunClient')`

continued on next page

Name	Description
can_run	**Value:** `True`
TO_MUCH_ERROR_I-N_A_ROW	**Value:** `20`

32 Module deepwifi.run_acs2

```
Running
=======
There are two options:

a) creates the configuration files (hostapd.conf, and wpa_supplicant.conf), create the s
   copy the files to the APs and STAs
   and then runs the experiment
python3 run_acs.py --save-wpa-conf --save-hostapd-conf

b) just run it. This pressuposes that the configuration files are copied to the devices
python3 run_acs.py
```

32.1 Functions

reboot(*aps*, *stas*)

get_best_channel(*data*, *aps*, *valid_channels*=[2412,2417,2422,2427,2432,2437,2442,2447,2452,2457,2462])

```
1) calculate the average interference factor for each channel 'c'
IF[c] = 10^(chan_nf/5) + (busy time - tx time) / (active time - tx time) * 2^(10^(ch
2) return:
    a) the channel with the lowest average for each AP if they are different
    b) the second channel considering idle time

*********** NOTICE ***********
This method is specific for out experiment, because the algorithm that considers the
*********** NOTICE ***********

@param data: list of survey data for all AP
@return: the best channel for each AP
```

get_action(*env*, *aps*, *interval*=1, *num_min_surveys*=5, *tx_power*=10)

decide based on the ACS index

run_acs(*aps*, *stas*, *env*, *interaction_interval*, *run_id*='1')

32.2 Variables

Name	Description
LOG	**Value:** `logging.getLogger('RunClient')`
can_run	**Value:** `True`
TO_MUCH_ERROR_IN_A_ROW	**Value:** `20`

33 Module deepwifi.run_experiment

```
Running
=======
There are two options:

a) creates the configuration files (hostapd.conf, and wpa_supplicant.conf), create the s
   copy the files to the APs and STAs
   and then runs the experiment
python3 run_client.py --save-wpa-conf --save-hostapd-conf

b) just runs. This pressuposes that the configuration files are copied to the devices
python3 run_client.py  --qoe-model [client | ap | hybrid  | psnr ]
```

33.1 Functions

reboot(*aps*, *stas*)

33.2 Variables

Name	Description
LOG	**Value:** `logging.getLogger('RunClient')`

34 Module deepwifi.to_md

34.1 Functions

skip(*line*)

34.2 Variables

Name	Description
OUTPUT_DIR	**Value:** `'deepwifi.wiki'`
files	**Value:** `glob.glob('doc/*.html')`

35 Script script-LICENSE

36 Script script-monitor_ap_sh

37 Script script-monitor_sh

38 Script script-run_acs_old_py

```
Running
=======
There are two options:

a) creates the configuration files (hostapd.conf, and wpa_supplicant.conf), create the s
   copy the files to the APs and STAs
   and then runs the experiment
python3 run_acs.py --save-wpa-conf --save-hostapd-conf

b) just runs. This pressuposes that the configuration files are copied to the devices
python3 run_acs.py
```

38.1 Functions

reboot(*aps*, *stas*)

get_best_channel(*data*, *valid_channels*=`[2412,2417,2422,2427,2432,2437,2442,2447,2452,2457,2462]`)

1) calculate the average interference factor for each channel 'c' IF[c] = 10^(chan_nf/5) + (busy time - tx time) / (active time - tx time) * 2^(10^(chan_nf/10) + 10^(band_min_nf/10)) 2) return the channel with the lowest average

Parameters

`data`: list of survey data for one AP

Return Value

the best channel

get_action(*env*, *aps*, *interval*=`1`, *num_min_surveys*=`5`, *tx_power*=`10`)

decide based on the ACS index

run_acs(*aps*, *stas*, *env*, *interaction_interval*, *run_id*=`'1'`)

38.2 Variables

Name	Description
LOG	**Value:** `logging.getLogger('RunClient')`

continued on next page

Name	Description
can_run	**Value:** `True`
TO_MUCH_ERROR_IN_A_ROW	**Value:** `20`

39 Script script-teste_model_ipynb

40 Script script-teste_tcn_ipynb

Index