synergy_analysis detailed tutorials

24/7/2021

3 papers mentioned in this tutorials:

Paper 1: J. Chai and M. Hayashibe, ”Motor Synergy Development in High-Performing Deep Reinforcement Learning Algorithms,” in IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 1271-1278, April 2020.

Paper 2: J. Chai and M. Hayashibe, ”Quantification of Joint Redundancy considering Dynamic Feasibility using Deep Reinforcement Learning,” in ICRA 2021.

Paper 3: J. Chai and M. Hayashibe, ”Deep Reinforcement Learning with Gait Mode Specification for Quadrupedal Trot-Gallop Energetic Analysis,” in EMBC 2021.

0) Setup

Follow the installation instructions written in the README.md file in the synergy_analysis codebase.

If there are some problems, fix them case by case, except for the tricky libraries listed in the next page, which you must follow.

Libraries' version that must be followed:

1) install serializable by: (you must uninstall it first)
   pip install git+https://github.com/hartikainen/serializable.git@76516385a3a716ed4a2a9ad877e2d5cbcf18d4e6
2) tensorflow==2.2.0
3) tensorflow-probability==0.10.1

These are the tricky ones, others can be changed/ adapted easily.

All 3 papers results reproduction

Most of the commands to reproduce the results of all three papers can be found in:

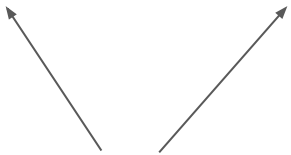
1) Paper1_commands.sh
2) Paper2_Arm2D_commands.sh
3) Paper2_Arm3D_commands.sh
4) Paper3_commands.sh

It is impossible to understand the commands now. You must follow all the explanations given later.

1) Training

Command lines to run a training (with the virtual environment synergy_analysis activated):
a)

softlearning run_example_local



Keywords to run any experiments

Command lines to run a training (with the virtual environment synergy_analysis activated):
b)

softlearning run_example_local examples.development

Command lines to run a training (with the virtual environment synergy_analysis activated):
c)

softlearning run_example_local examples.development --universe=gym
--domain=HalfCheetah --task=Energy0-v0

The agent. Can be:

1) HalfCheetah
2) HalfCheetahHeavy
3) FullCheetah
4) etc… (Check the codes, explained later)

The cost function variation. In this example, Energy0 means no energy consideration in the cost function. The coefficient for energy is 0. Always ends with -v0 .

universe is always gym

Command lines to run a training (with the virtual environment synergy_analysis activated):
d)

softlearning run_example_local examples.development --universe=gym
--domain=HalfCheetah --task=Energy0-v0 --exp-name=HC_E0_r1

The name of the folders for the experiments. Please follow STRICTLY the naming system: agent_energy_trial
Examples: HC_E0_r2
HCheavy_E0_r1
FC_E0_r3

Command lines to run a training (with the virtual environment synergy_analysis activated):
e)

softlearning run_example_local examples.development --universe=gym --domain=HalfCheetah --task=Energy0-v0 --exp-name=HC_E0_r1 --checkpoint-frequency=100 --trial-gpus 1 --algorithm SAC

Saving checkpoints per 100 epochs. The total number of epochs are 3000 which can be changed in the codes.

Number of GPU to be used.

Type of algorithms. Can be:

a) SAC
b) TD3

Available arguments

All the essential arguments can be found in utils.py

utils.py

```
234
235     parser.add_argument(
236         '--debug', action = 'store_true')
237     parser.add_argument(
238         '--gpu_choice', type=int, default=None)
239     parser.add_argument(
240         '--actor_size', type=int, default=256)
241     parser.add_argument(
242         '--n_layer', type=int, default=2)
243     parser.add_argument(
244         '--critic_size', type=int, default=256)
245     parser.add_argument(
246         '--epoch_length', type=int, default=1000)
247     parser.add_argument(
248         '--total_epoch', type=int, default=None)
249
```

For debugging in PyCharm

Choose the number of GPU to be used if available. In the lab server, from 0 to 3.

The results of the training/ training progresses csv/ training checkpoint can be found in:

synergy_analysis/experiments_results/gym

2) Test/Visualize the trained agents

In the folder
synergy_analysis/examples/development

python simulate_policy.py path_to_last_checkpoint/number_of_checkpoint
--max-path-length=1000 --num-rollouts=10

Example:

python simulate_policy.py
/home/jzchai/ray_results/gym/HalfCheetahHeavy/Energy0-v0/2019-11-18T13-52-17-HCheavy_E0_r10/ExperimentRunner_0_max_size=1000000,seed=9640_2019-11-18_13-52-18ders_cz3/checkpoint_2000 --max-path-length=1000
--num-rollouts=10

List of main trainable agents

Paper 1

HalfCheetah

1) HalfCheetah-Energy0-v0

HalfCheetah Heavy

2) HalfCheetahHeavy-Energy0-v0

FullCheetah

3) FullCheetah-Energy0-v0

Paper2

Arm2D

1) VA-Energy0-v0, VA4dof-Energy0-v0, VA6dof-Energy0-v0, VA8dof-Energy0-v0

Arm3D

2) RealArm3dof-Energy0-v0,RealArm4dof-Energy0-v0,RealArm5dof-Energy0-v0 ,RealArm6dof-Energy0-v0,RealArm7dof-Energy0-v0

Arm3D kinematic feasibility

3) RealArm7dof-Energy0-v1,RealArm7dof-Energy0-v2,RealArm7dof-Energy0-v9 ,RealArm7dof-Energy0-v3

Paper2

Arm3D Dynamic Feasibility

1) RealArm7dof-Energy0-v5,RealArm7dof-Energy0-v4, RealArm7dof-Energy0-v0

HalfCheetah Squat

2) HalfCheetahSquat2dof-EnergyAlt-v0,HalfCheetahSquat4dof-EnergyAlt-v0,HalfCheetahSquat6dof-EnergyAlt-v0

HalfCheetah Run

3) HalfCheetah2dof-Energy0-v0,HalfCheetah4dof-Energy0-v0,HalfCheetah-Energy0-v0

Paper2

Ant Run/ Squat, Redundant-Ant Squat

1) AntRun-Energy0-v0,AntSquaT-Energy0-v0,AntSquaTRedundant-Energy0-v0

Paper3

Quadruped with and without gait specification

1) FullCheetahHeavy-Energy0-v4,FullCheetahHeavy-SymlossG-v4,FullCheetah Heavy-SymlossT-v4

Quadruped gallop/ trot comparison at different speeds

2) FullCheetahHeavy-SymlossT-v4,FullCheetahHeavy-SymlossT-v6,FullCheeta hHeavy-SymlossG-v4,FullCheetahHeavy-SymlossG-v6

Quadruped gallop gait comparison at different passive joint spring effect

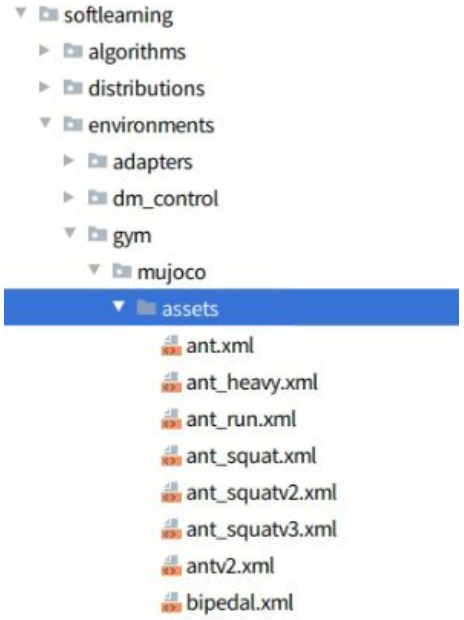
3) FullCheetahHeavy-MinSpringG-v4,FullCheetahHeavy-SymlossG-v4,FullChee tahHeavy-ExSpringG-v4

3) Procedure to add a new agent

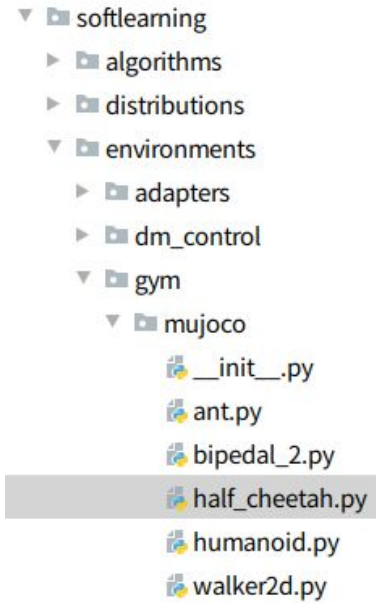
a) I expect you have the xml file of your new agent.

Put that xml file to

synergy_analysis/softlearning/environments/gym/mujoco/assets



b) In synergy_analysis/softlearning/environments/gym/mujoco, create the corresponding python file describing the reward function, etc. You can take example of other existing python files for other agents.




```python
class HalfCheetahEnv(mujoco_env.MujocoEnv, utils.EzPickle):
    def __init__(self,
                 xml_file='half_cheetah.xml',
                 forward_reward_weight=1.0,
                 ctrl_cost_weight=0.1,
                 reset_noise_scale=0.1,
                 exclude_current_positions_from_observation=True,
                 energy_weights=0.):...

    def control_cost(self, action):...

    def step(self, action):...

    def _get_obs(self):...

    def reset_model(self):...

    def viewer_setup(self):...
```

c) In synergy_analysis/softlearning/environments/gym/__init__.py, add:

```
{
    'id': 'HalfCheetah-Energy0-v0',
    'entry_point': (f'{MUJOCO_ENVIRONMENTS_PATH}'
                    '.half_cheetah:HalfCheetahEnv'),
},
{
    'id': 'Giraffe-Energy0-v0',
    'entry_point': (f'{MUJOCO_ENVIRONMENTS_PATH}'
                    '.giraffe:GiraffeEnv'),
},
{
    'id': 'HalfCheetahHeavy-Energy0-v0',
    'entry_point': (f'{MUJOCO_ENVIRONMENTS_PATH}'
```

You fix a name for your agent in the format: AgentName-yourchoice-v0

You import the environment name from the python file you just created.

d) In the file synergy_analysis/examples/development/variants.py:

```
ENV_PARAMS = {
    'Bipedal2d': {  # 6 DoF
        'Energy0-v0': {
            'target_energy':3
        },

    },
    'HalfCheetahHeavy': {  # 6 DoF
        'Energy0-v0': {
            'forward_reward_weight':1.0,
            'ctrl_cost_weight':0.1,
            'energy_weights':0,
        },

    },

    'HalfCheetah': {  # 6 DoF
        'EnergySix-v0': {
            'forward_reward_weight':1.0,
            'ctrl_cost_weight':0.1,
            'energy_weights':6.0,

        },
```

Add your new agent and its environment in this manner.

You can also change various parameters of your environment here in this manner.

e) In the file synergy_analysis/examples/development/variants.py :

```
DEFAULT_NUM_EPOCHS = 200

NUM_EPOCHS_PER_DOMAIN = {
    'Swimmer': int(3e2),
    'Hopper': int(1e3),
    'HalfCheetah': int(3e3),
    'Giraffe': int(2e3),
    'HalfCheetahHeavy':int(3e3),
    'FullCheetah':int(3e3),
    'Centripede':int(2e3),
    'Walker2d': int(1e3),
    'Bipedal2d':int(300),
```

Specify also the number of epochs you want to train your agents.

You can now use your new agent for training.

4) Change the number of training epochs

To change properties such as the number of training epochs (3000 for HalfCheetah)
or to see the list of agents available:

Check the variants.py file in synergyDRL/examples/development

```
DEFAULT_NUM_EPOCHS = 200

NUM_EPOCHS_PER_DOMAIN = {
    'Swimmer': int(3e2),
    'Hopper': int(1e3),
    'HalfCheetah': int(3e3),
    'Giraffe': int(2e3),
    'HalfCheetahHeavy':int(3e3),
    'FullCheetah':int(3e3),
    'Centripede':int(2e3),
    'Walker2d': int(1e3),
    'Bipedal2d':int(300),
```

5) Procedure to change the energy coefficient in the reward

a) For an existing agent, in synergy_analysis/softlearning/environments/gym/__init__.py, add:

```
{
    'id': 'HalfCheetah-EnergyPoint1-v0',
    'entry_point': (f'{MUJOCO_ENVIRONMENTS_PATH}'
                    '.half_cheetah:HalfCheetahEnv'),
},
{
    'id': 'HalfCheetah-Energy0-v0',
    'entry_point': (f'{MUJOCO_ENVIRONMENTS_PATH}'
                    '.half_cheetah:HalfCheetahEnv'),
}
```

Create a name that helps you identify the energy consideration weights. Here, I have decided a name for weight= 0.1

The case of 0 energy consideration.

b) In the file synergy_analysis/examples/development/variants.py :

```
'HalfCheetah': {  # 6 DoF
    'EnergySix-v0': {
        'forward_reward_weight':1.0,
        'ctrl_cost_weight':0.1,
        'energy_weights':6.0,

    },
    'EnergyFour-v0': {
        'forward_reward_weight':1.0,
        'ctrl_cost_weight':0.1,
        'energy_weights':4.0,

    },
    'EnergyTwo-v0': {
        'forward_reward_weight':1.0,
        'ctrl_cost_weight':0.1,
        'energy_weights':2.0,
    },
```

For your specified agent

Add the name/ identifier that you have created just now.

Change the corresponding parameters accordingly , for example, energy_weights is 4 here.

You can now train this agent with energy consideration.

6) To collect action data/ reward/ states information from all trained checkpoint

a) In synergy_analysis, run the command:

python examples/development/collect_actions_SAC.py --path path_to_folder

Example:

For TD3, it's development_TD3/collect_action_TD3.py

python examples/development/collect_actions_SAC.py --path /home/jzchai/PycharmProjects/synergy_analysis/experiments_results/gym/FullCheetah/Energy0-v0/2019-11-17T15-54-52-FC_E0_r14/ExperimentRunner_0_max_size=1000000,seed=2906_2019-11-17_15-54-5359rh3bj2

python examples/development/collect_actions_SAC.py --agent HC --tr _r1 --start 100 --final 3000 --step 100

python examples/development_TD3/collect_actions_TD3.py --agent HC --tr _r1 --start 100 --final 3000 --step 100

b) Your collected data will be found in synergy_analysis/experiments_results/collected_actions

Repeat for all the paths that you want to extract synergy for.

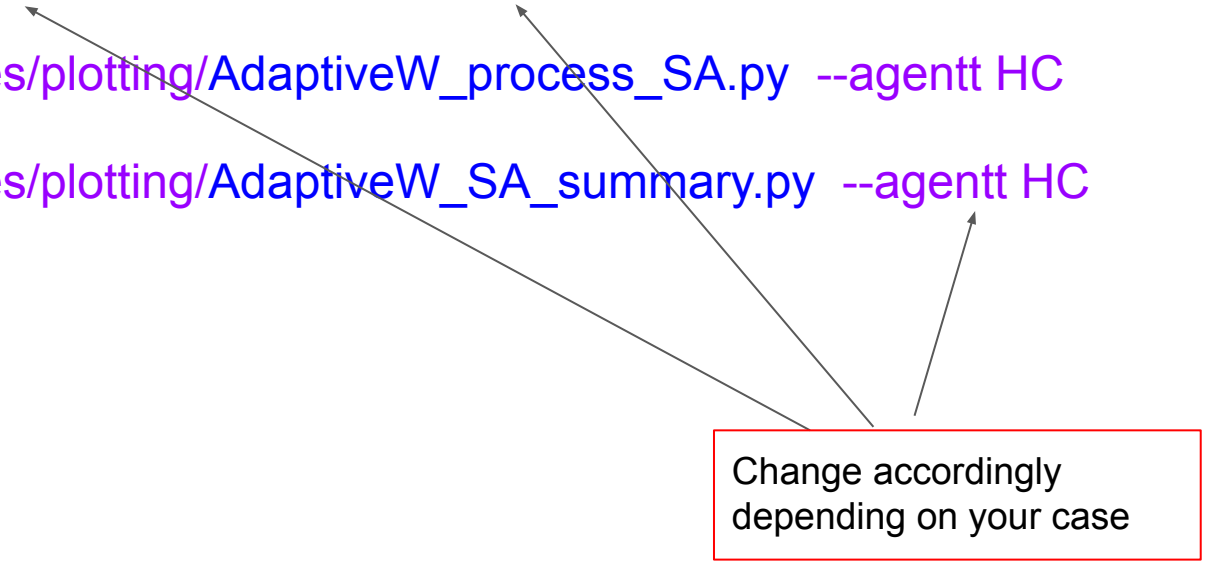
NOTE: You may need to modify the collect_action_SAC.py and collect_action_TD3.py to extract actions signals for your own experiments!

c) After collecting data, in synergy_analysis, run the command to preprocess synergy information:

```
python examples/plotting/AdaptiveW_Extract_SA_P_PI_corr_each_trial_SVD.py
--tr    _r1 _r2 _r3 _r4 _r5   --ee E0 --agentt HC

python examples/plotting/AdaptiveW_process_SA.py  --agentt HC

python examples/plotting/AdaptiveW_SA_summary.py  --agentt HC
```



7) Plot various figures

NOTES

To produce figures using the python files in synergy_analysis/development/plotting, you must have:

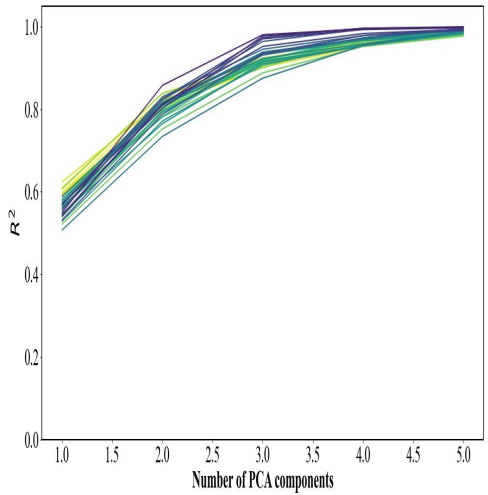
1) Run the training
2) Collected actions
3) Preprocessed the actions for synergy information

Commands for these steps are explained previously.

Output of all these figures will be in the path:
synergy_analysis/experiments_results/Synergy/

AdaptiveW_Extract_synergy_HC_compare_PI_spatial_evolution.py

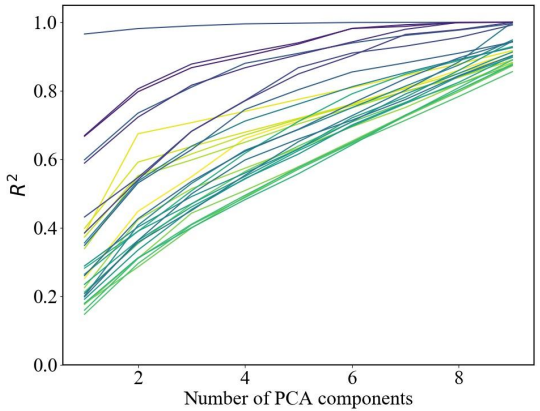
Plot spatial synergy development, output folder: synergy_development_{}, eg. synergy_development_HC



python examples/plotting/AdaptiveW_Extract_synergy_HC_compare_PI_spatial_evolution.py --agentt HC --ee E0 --tr _r1

AdaptiveW_Extract_synergy_HC_compare_PI_spatiotemporal_evolution_SVD.py

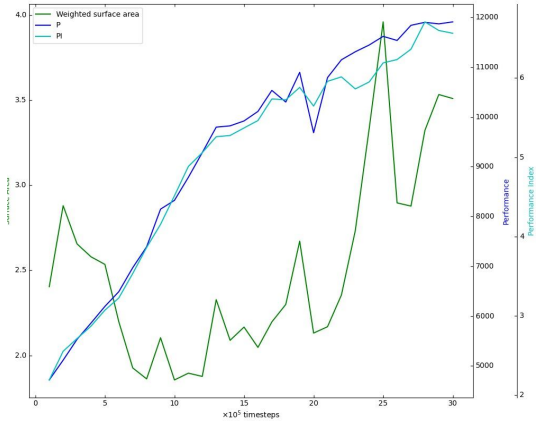
Plot spatiotemporal synergy development, output folder: synergy_development_{}, eg. synergy_development_HC



python examples/plotting/AdaptiveW_Extract_synergy_HC_compare_PI_spatiotemporal_evolution_SVD.py --agentt HC --ee E0 --tr _r1

AdaptiveW_surface_area_spatiotemporal_evolution_SVD.py

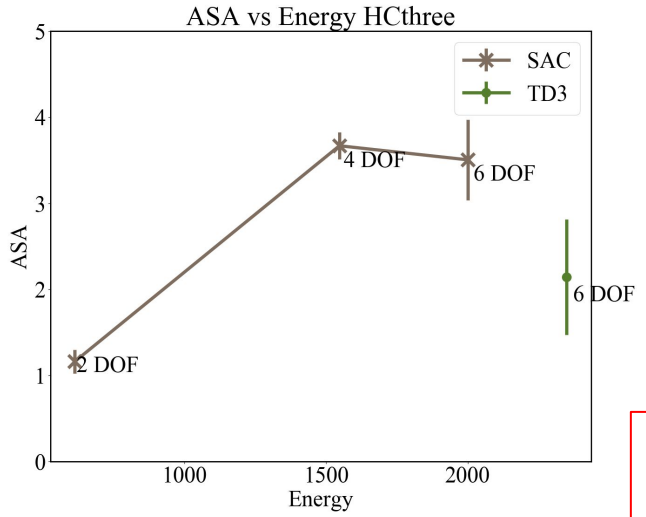
Plot spatiotemporal synergy surface area development, output folder: synergy_development_{}, eg. synergy_development_HC



python examples/plotting/AdaptiveW_surface_area_spatiotemporal_evolution_SVD.py --agentt HC --ee E0 --tr _r1

ASA_vs_P_lineplot.py

Plot ASA vs Performance, output folder: ASA_vs_P_line_plot



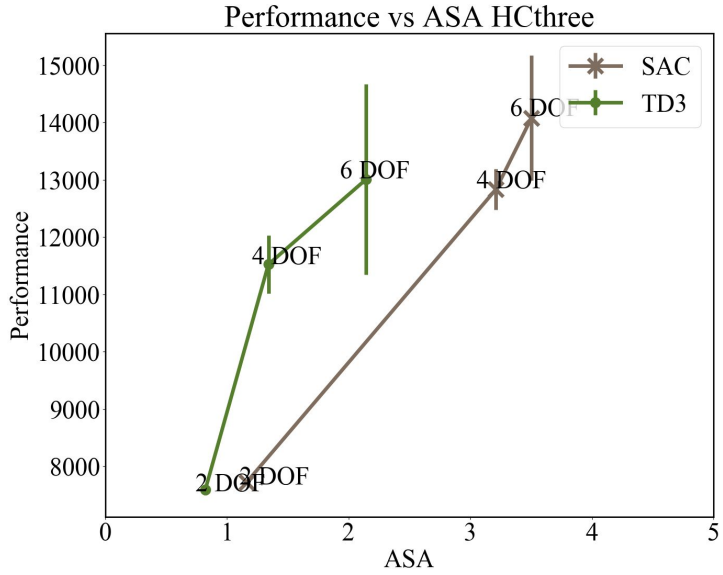
python examples/plotting/ASA_vs_P_lineplot.py HCthree

Need to open the ASA_vs_P_lineplot.py and maybe commons.py and add your configuration.

P_vs_ASA_lineplot.py

Plot ASA vs Performance, output folder: ASA_vs_P_inv_line_plot

Figures similar to second paper

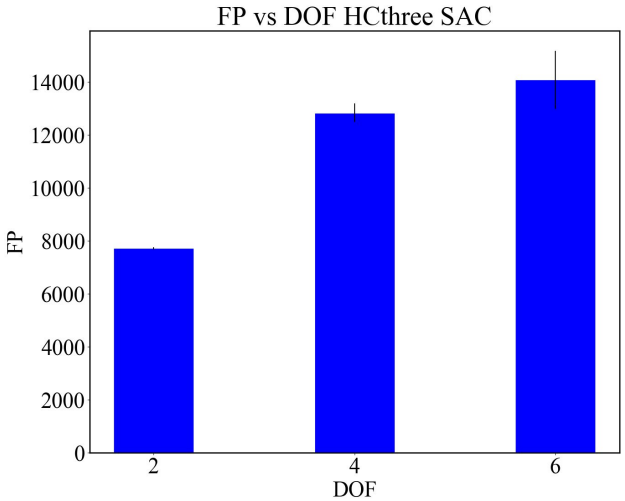


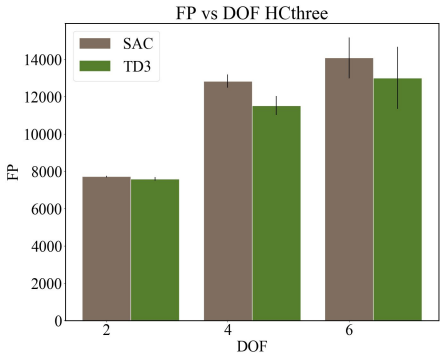
python examples/plotting/P_vs_ASA_lineplot.py HCthree

Need to open the P_vs_ASA_lineplot.py and maybe commons.py and add your configuration.

compare_dof_P_lineplot.py

Plot performance between different agents available in process_SA_final_summary, output folder: dof_P_bar_plot or dof_P_double_bars_plot



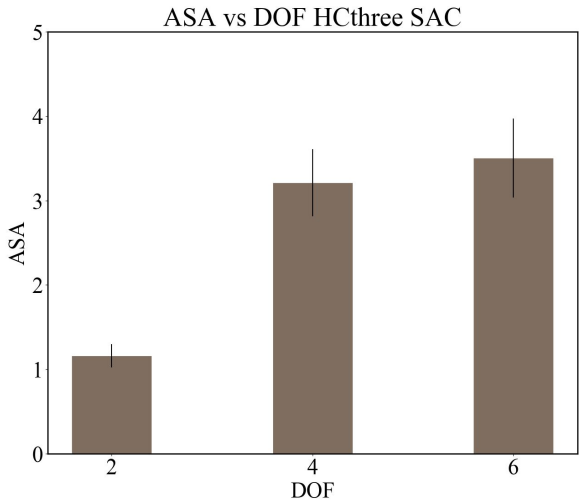


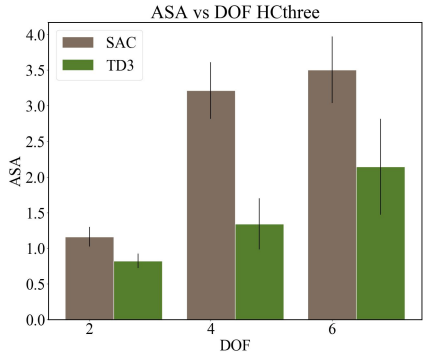
Need to open the compare_dof_P_lineplot.py and maybe commons.py and add your configuration.

python examples/plotting/compare_dof_P_lineplot.py HCthree
python examples/plotting/compare_dof_P_lineplot.py HCthree --double_bars

compare_dof_synergy_lineplot.py

Plot synergy between different agents available in process_SA_final_summary, output folder: dof_bar_plot or dof_double_bar_plot



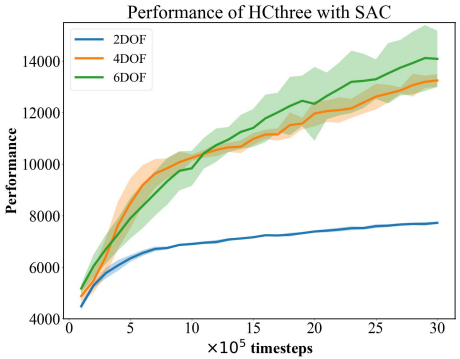


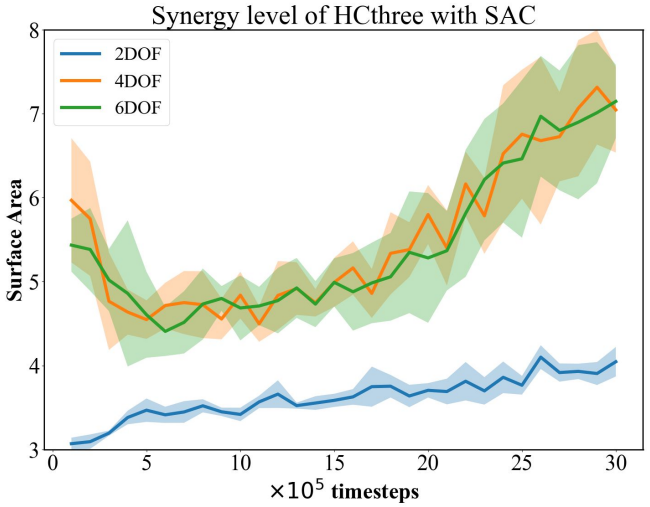
Need to open the compare_dof_synergy_lineplot.py and maybe commons.py and add your configuration.

python examples/plotting/compare_dof_synergy_lineplot.py HCthree
python examples/plotting/compare_dof_synergy_lineplot.py HCthree --double_bars

learning_progress_compare_synergy.py

Plot synergy, performance between different agents available in raw_csv (need to update in commons.py), output folder: compare_synergy_graphs





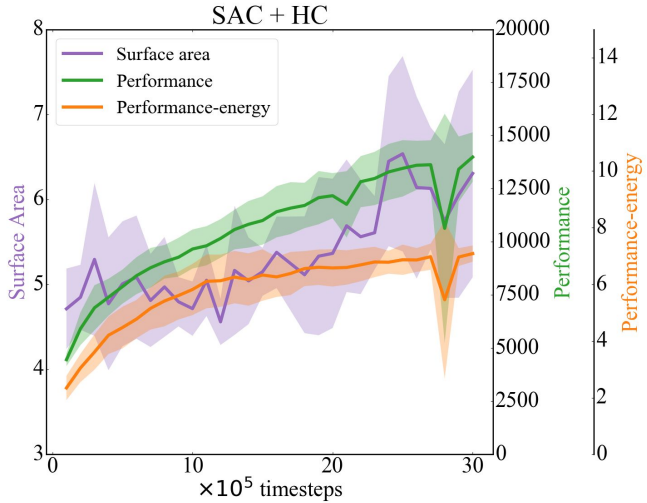
python examples/plotting/learning_progress_compare_synergy.py HCthree

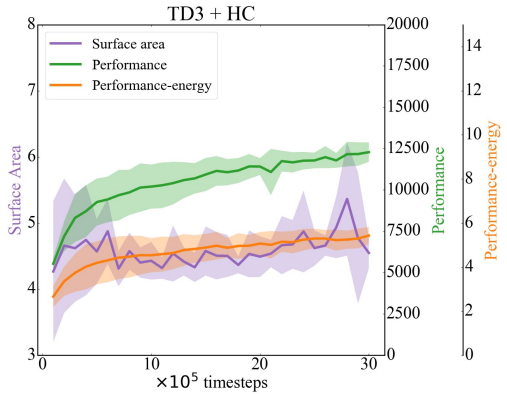
Need to open the learning_progress_compare_synergy.py and maybe commons.py and add your configuration.

learning_progress_synergy.py

Plot synergy, performance, performance-energy between different agents available in raw_csv (need to update in commons.py), output folder: learning_progress_graphs

Figures similar to the first paper





python examples/plotting/learning_progress_synergy.py

Need to open the learning_progress_synergy.py and maybe commons.py and add your configuration.

7) Bash files

Finally, in synergyDRL, there are a few bash files. They are files end with the format .sh

In linux, to execute these files, you need to do:

```
chmod +x name_of_the_files.sh

./name_of_the_files.sh
```

If you open these bash files, you will find the commands that I have just introduced earlier.

You can make use of these bash files and create your own bash files to automate the command lines that you want to run.

For example, when you have 10 command lines to run one after another, it is a good idea to use a bash file.

8) Add new algorithm

In softlearning/algorithm , create the new algorithm python file and its corresponding training loop. For example:

sac.py and rl_algorithm.py

Then in softlearning/algorithms/utils.py, add the corresponding keyword in the ALGORITHM_CLASSES dictionary:

```
ALGORITHM_CLASSES = {

    'SAC': create_SAC_algorithm,
```

Add this key.

Create this function by looking at other existing example in the dictionary.

In softlearning/algorithms/__init__.py, import the corresponding algorithm:

```
from .sql import SQL
from .sac import SAC
from .rsac import rSAC
from .sac_dpl import dplSAC
```

Add one line to import your new algorithm from the corresponding file

In the variants.py file in your development folder, add in the ALGORITHM_PARAMS_ADDITIONAL the key and the corresponding parameters for your algorithm. It is not necessary to modify ALGORITHM_PARAMS_BASE.

```
ALGORITHM_PARAMS_ADDITIONAL = {
    'SAC': {
        'type': 'SAC',
        'kwargs': {
            'reparameterize': REPARAMETERIZE,
            'lr': 3e-4,
            'target_update_interval': 1,
            'tau': 5e-3,
            'target_entropy': 'auto',
            'store_extra_policy_info': False,
            'action_prior': 'uniform',
            'n_initial_exploration_steps': int(1e3),
        }
    },
    'rSAC': {
        'type': 'rSAC',
        'kwargs': {
            'reparameterize': REPARAMETERIZE,
            'lr': 3e-4,
            'target_update_interval': 1,
            'tau': 5e-3,
            'target_entropy': 'auto',
            'store_extra_policy_info': False,
            'action_prior': 'uniform',
            'n_initial_exploration_steps': int(1e3),
        }
    },
```

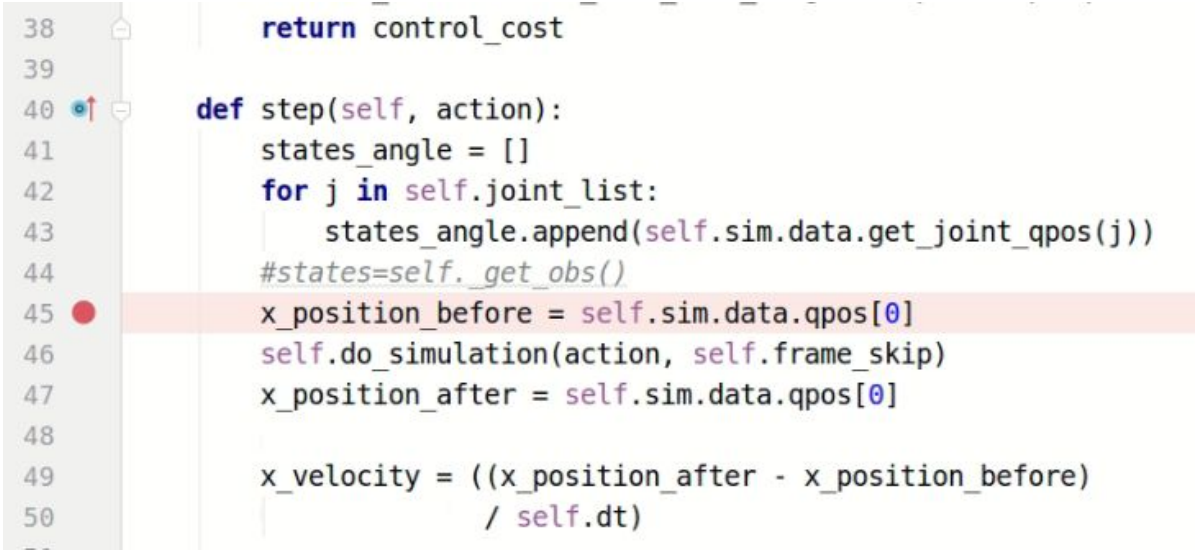
Hyperparameters in your algorithm

You can now use this algorithm by adding --algorithm new_algo_name when running the command line of training.

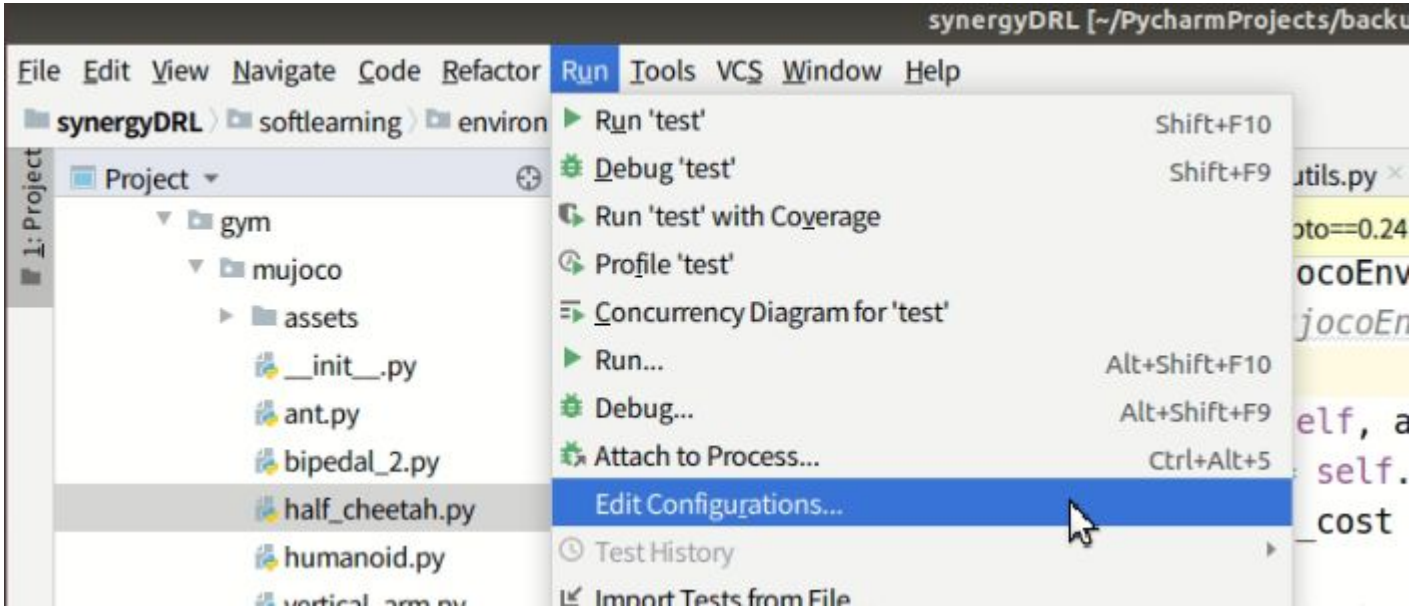
9) Debug in PyCharm

We can now set breakpoint in the program and debug normally.

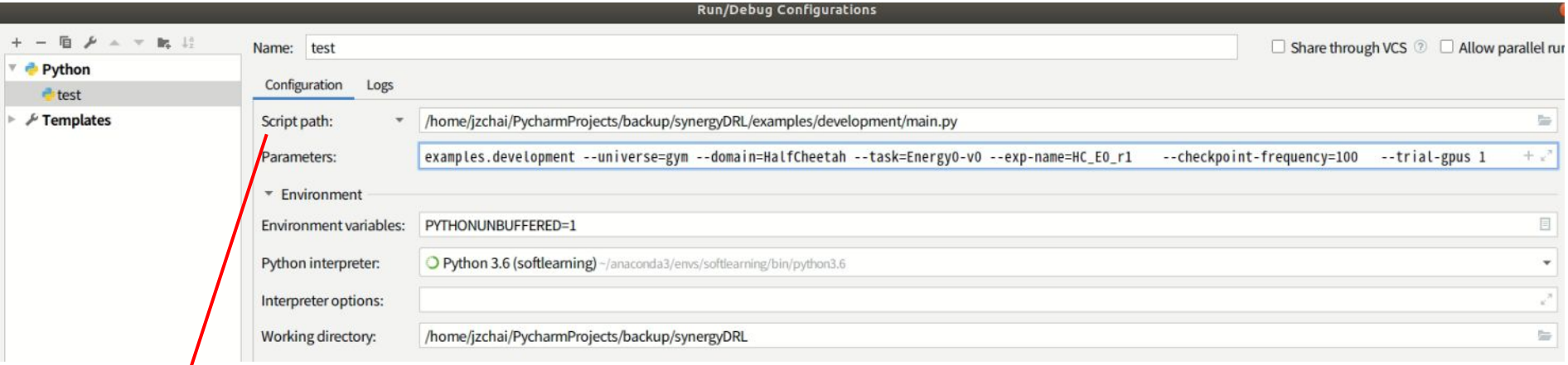
1) Set breakpoints in your programs. For example, in synergy_analysis/softlearning/environments/gym/mujoco/HalfCheetah.py



2) Edit configurations in Pycharm.

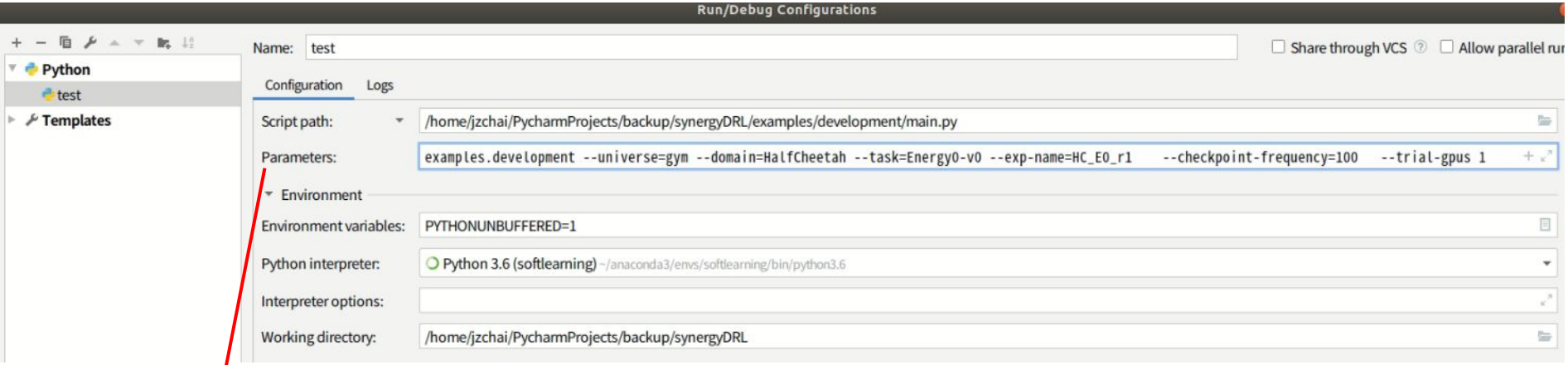


2.1) Edit configurations in Pycharm.



main.py in the development folder.

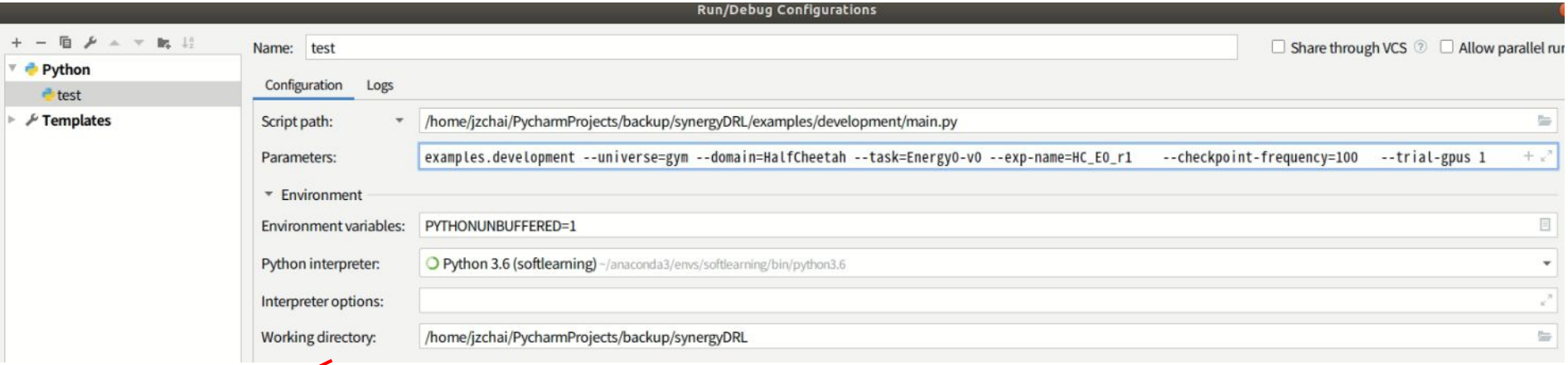
2.2) Edit configurations in Pycharm.



examples.development --universe=gym --domain=HalfCheetah
--task=Energy0-v0 --exp-name=HC_E0_r1 --checkpoint-frequency=100
--trial-gpus 1 --algorithm SAC --debug

Write your experiment configurations normally and add this --debug for debugging function.

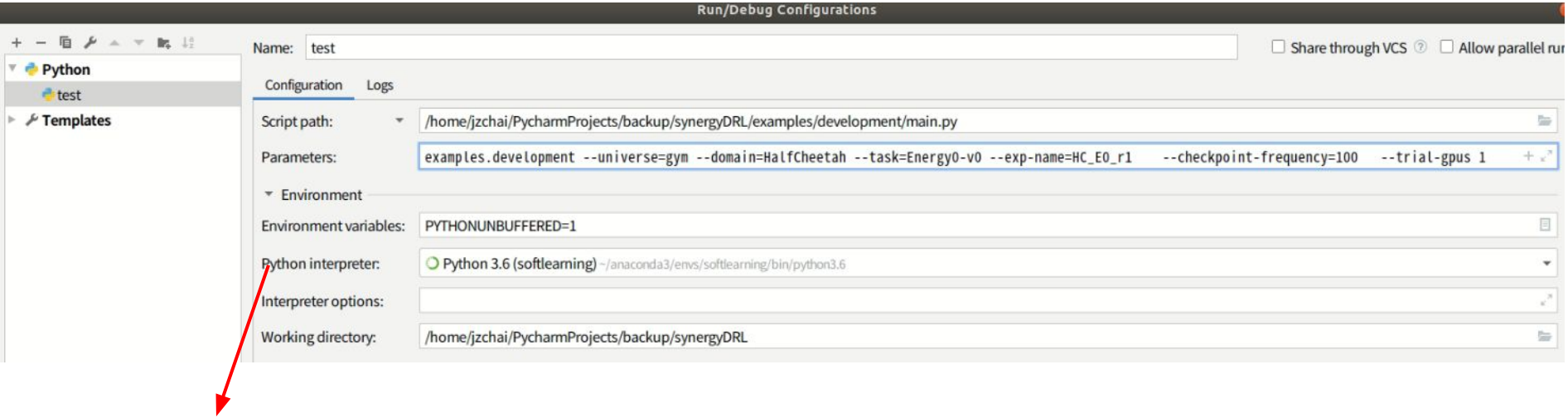
2.3) Edit configurations in Pycharm.



For working directory, I would suggest you to put synergy_analysis as the base directory (as shown in the figure).

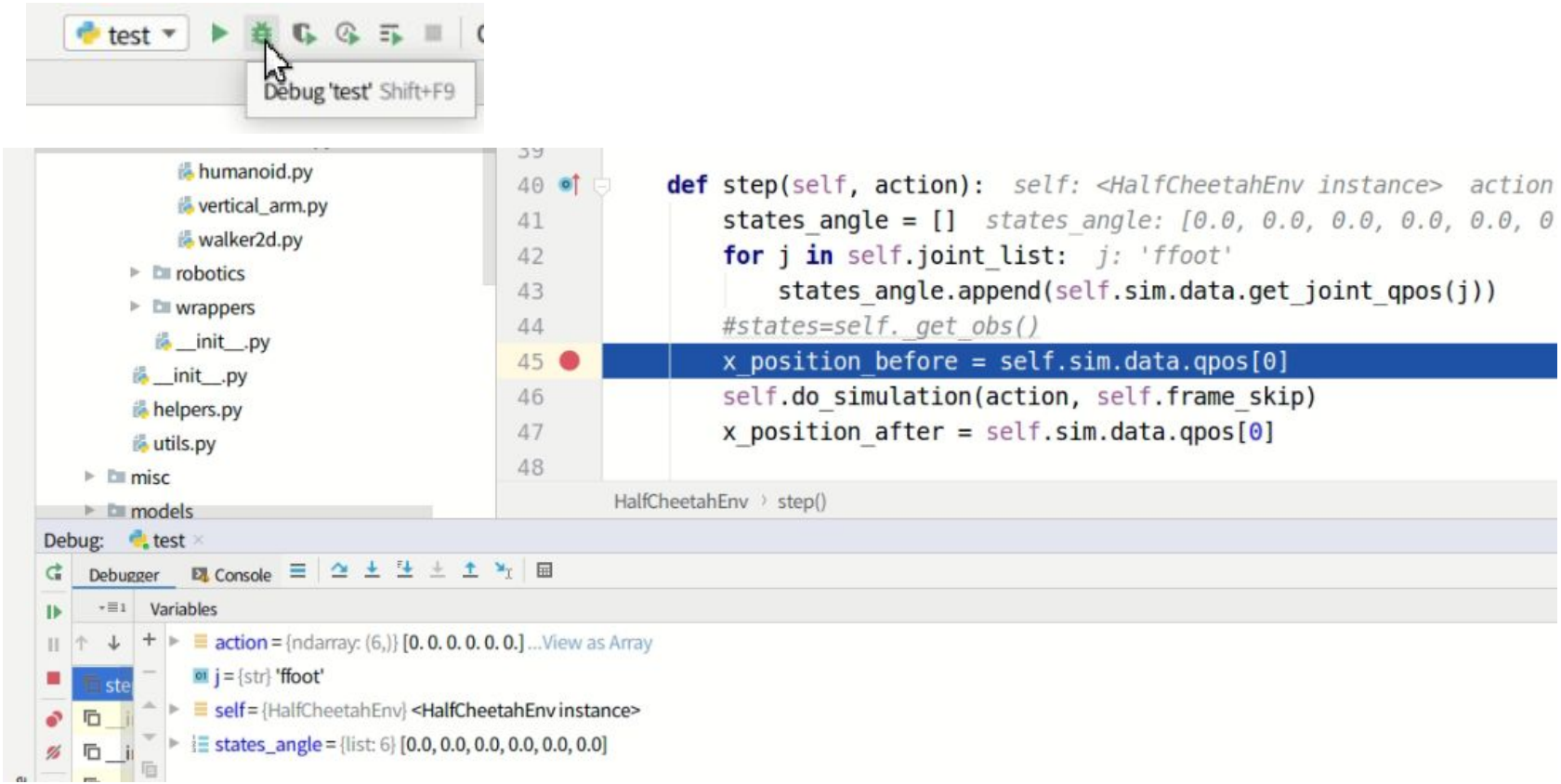
It is also possible to run elsewhere, but maybe your saved files/ outputs will be elsewhere as well. You just need to look for it.

2.4) Edit configurations in Pycharm.



Make sure you choose your correct Python interpreter.

3) Finally, debug:



10) Important Notes about debugging

1) In case of errors, make sure you updated the correct ray version. Tested version: ray 1.4.1
2) When --debug is written in the configurations, this will make your experiments run locally without GPU. So, don't write this when you are not debugging and when you wish to use GPU.