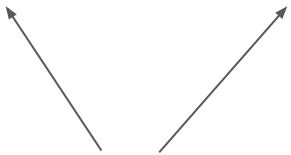
# SynergyDRL useful commands

7/1/2020

# 1) Training

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

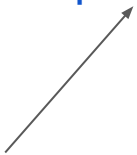
softlearning run_example_local



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



Modules of the codes to be run. It is either:

a) examples.development
b) examples.development_TD3

# 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 in -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

# 2) Test/Visualize the trained agents

In the folder
synergyDRL/examples/development or
synergyDRL/examples/development_TD3 :

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

# 3) 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
synergyDRL/examples/development_TD3

```
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),
```

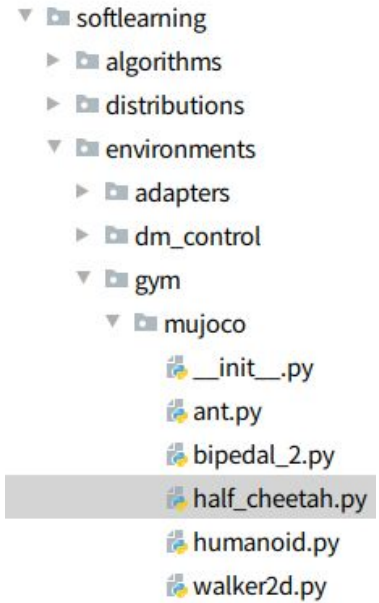
# 4) Procedure to add a new agent

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

Put that xml file to

anaconda3/envs/synergy_analysis/lib/python3.6/site-packages/gym/envs/mujoco/assets

b) In synergyDRL/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 synergyDRL/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 synergyDRL/examples/development/variants.py and synergyDRL/examples/development_TD3/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 synergyDRL/examples/development/variants.py and synergyDRL/examples/development_TD3/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.**

# 5) Procedure to change the energy coefficient in the reward

a) For an existing agent, in synergyDRL/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 synergyDRL/examples/development/variants.py and synergyDRL/examples/development_TD3/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
and
Extract synergy (produce the same figures as in the paper)

a) In synergyDRL, run the command:

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

or

python examples/development_TD3/collect_actions_TD3.py --path path_to_folder

**Example:**

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

Must give the path until this ExperimentRunner folder

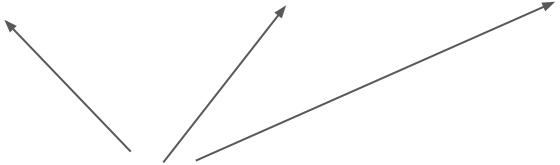
b) Your collected data will be found in synergyDRL/experiments_results/collected_actions

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

c) After collecting data, to extract synergy development graph, in synergyDRL, run the command:

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

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



Change accordingly based on your experiments, e.g. E0_TD3, FC, _r8, etc.

**Saved figures can be found in synergyDRL/experiments_results/Synergy**

# d) After collecting data, to bar plots, in synergyDRL, run the command:

```
python examples/plotting/AdaptiveW_Extract_SA_P_PI_corr_each_trial.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

python examples/plotting/AdaptiveW_plot_summary_histogram.py

python examples/plotting/AdaptiveW_plot_summary_histogram_performance.py
```

**This will give you bar plots to compare results between TD3 and SAC (if you have run the previous steps for both SAC and TD3).**

**Saved figures can be found in synergyDRL/experiments_results/Synergy**

e) Finally, to summarize all the results, in synergyDRL, run the command:

```
python examples/plotting/AdaptiveW_plot_summary_three_histograms.py

python examples/plotting/AdaptiveW_plot_summary_three_histograms_performance.py

python examples/plotting/learning_progress_synergy.py
```

**This will produce figures that can be found in the paper.**

**Saved figures can be found in synergyDRL/experiments_results/Synergy**

# 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.