

Codes README

Highly Efficient Self-Adaptive Reward Shaping for Reinforcement Learning

Contents

1 Requirements	1
2 Run SASR Algorithm	1
3 Run Experimental Results	3

1 Requirements

The SASR is implemented based on PyTorch which has been tested on:

```
pytorch==2.0.1+cu117
```

Install all dependent packages:

```
pip3 install -r requirements.txt
```

2 Run SASR Algorithm

Run the following command to train SASR algorithm on the task specified by <Task ID>:

```
python run-SASR.py --env-id <Task ID>
```

All available environments with sparse rewards evaluated in our paper are listed below:

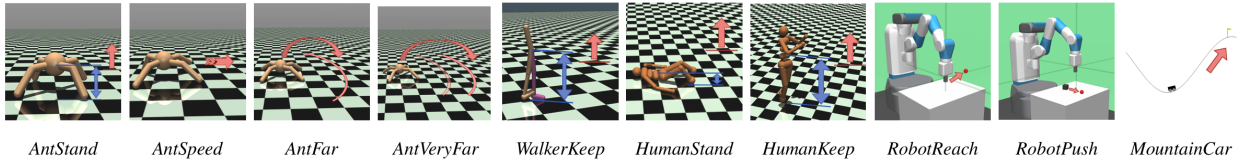


Figure 1: All available environments with sparse rewards

- Mujoco-Sparse tasks:

- MyMujoco/Ant-Height-Sparse: the *AntStand* task.
- MyMujoco/Ant-Speed-Sparse: the *AntSpeed* task.
- MyMujoco/Ant-Far-Sparse: the *AntFar* task.
- MyMujoco/Ant-Very-Far-Sparse: the *AntVeryFar* task.

- MyMujoco/Walker2d-Keep-Sparse: the *WalkerKeep* task.
- MyMujoco/HumanoidStandup-Sparse: the *HumanStand* task.
- MyMujoco/Humanoid-Keep-Sparse: the *HumanKeep* task.
- Robotics-Sparse tasks:
 - MyFetchRobot/Reach-Jnt-Sparse-v0: the *RobotReach* task.
 - MyFetchRobot/Push-Jnt-Sparse-v0: the *RobotPush* task.
- physical simulation tasks:
 - MountainCarContinuous-v0: the *MountainCar* task.

All hyper-parameters are set as default values in the code. You can change them by adding arguments to the command line. All available arguments are listed below:

```
--exp-name: the name of the experiment, to record the tensorboard and save the model.
--env-id: the task id
--seed: the random seed.
--cuda: the cuda device, default is 0, indicating to use cuda.
--gamma: the discount factor.

--pa-buffer-size: the buffer size to replay experiences.
--rb-optimize-memory: whether to optimize the memory
--batch-size: the batch size

--actor-lr: the learning rate of the actor
--critic-lr: the learning rate of the critic
--alpha: the alpha to balance the maximum entropy term
--alpha-autotune: whether to autotune the alpha, default is True
--alpha-lr: the learning rate of the alpha

--target-frequency: the target network update frequency
--tau: the tau for the soft update of the target network
--policy-frequency: the policy network update frequency

--total-timesteps: the total timesteps to train the model
--learning-starts: the burn-in period to start learning

--reward-weight: the weight factor of the shaped reward
--kde-bandwidth: the bandwidth of the kernel density estimation
--kde-sample-burnin: the burn-in period to sample the KDE
--rff-dim: the dimension of the random Fourier features
--retention-rate: the retention rate

--write-frequency: the frequency to write the tensorboard
--save-folder: the folder to save the model
```

3 Run Experimental Results

All experimental data are stored in `./experiments/exp-data.zip`. Before plotting the figures, you need to unzip the file to `./experiments/exp-data/` folder.

1. To evaluate the learning performance in comparison with baselines:

```
python ./experiments/comparison.py
```

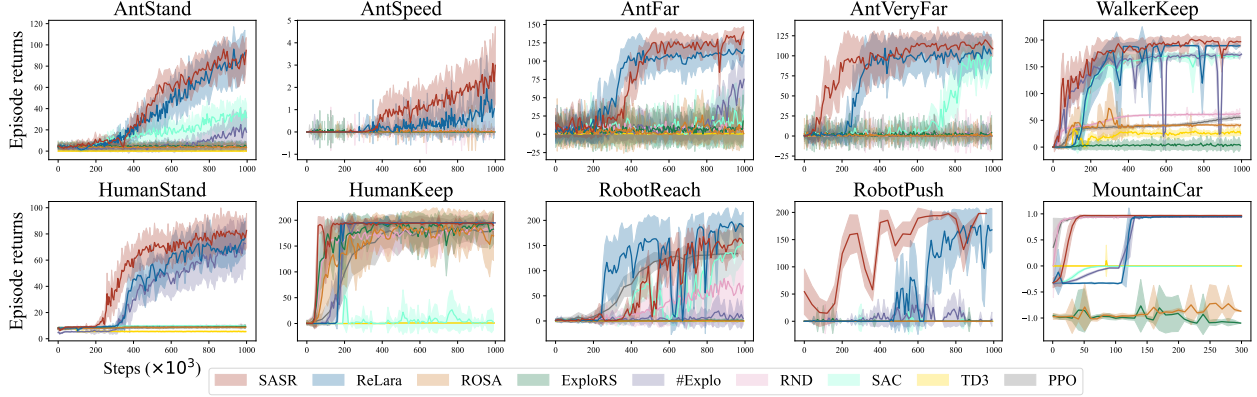


Figure 2: Comparison of the learning performance of SASR with the baselines.

2. **Ablation study #1:** To compare SASR with or without the sampling process:

```
python ./experiments/abla-without-sampling.py
```

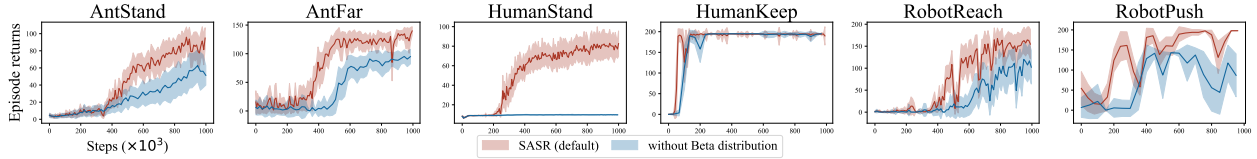


Figure 3: Comparison of the SASR with or without the sampling process.

3. **Ablation study #2:** To compare SASR with different retention rates:

```
python ./experiments/abla-retention-rate.py
```

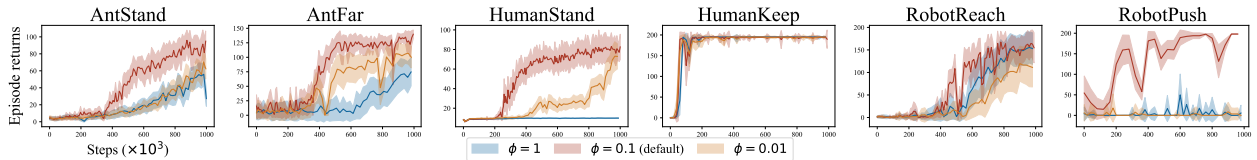


Figure 4: Comparison of SASR with different retention rates.

4. **Ablation study #3:** To compare SASR with different bandwidths h of Gaussian kernels:

```
python ./experiments/abla-bandwidth.py
```

5. **Ablation study #4:** To compare SASR with different RFF feature dimensions M :

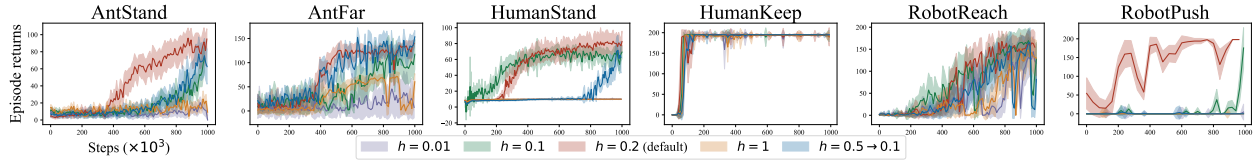


Figure 5: Comparison of SASR with different bandwidths of Gaussian kernels.

```
python ./experiments/abla-rff-dim.py
```

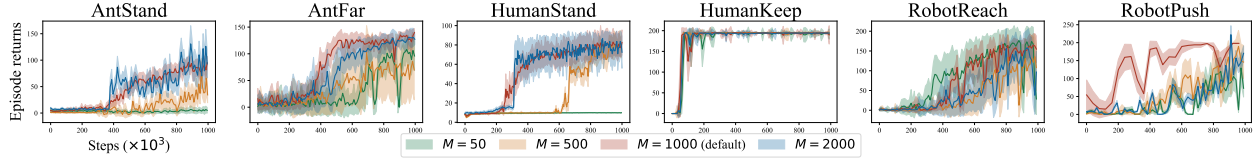


Figure 6: Comparison of SASR with different RFF feature dimensions.

6. **Ablation study #5:** To compare SASR with different scales of the shaped reward:

```
python ./experiments/abla-reward-weight.py
```

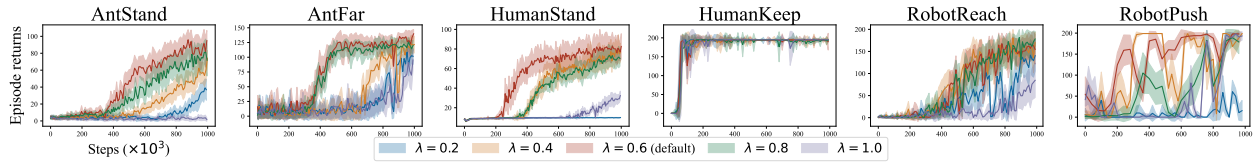


Figure 7: Comparison of different weight factors for the shaped reward.