pipeline passed | docs passing | coverage 95.00% | code style black

Stable Baselines3

Stable Baselines3 (SB3) is a set of reliable implementations of reinforcement learning algorithms in PyTorch. It is the next major version of Stable Baselines.

You can read a detailed presentation of Stable Baselines3 in the v1.0 blog post.

These algorithms will make it easier for the research community and industry to replicate, refine, and identify new ideas, and will create good baselines to build projects on top of. We expect these tools will be used as a base around which new ideas can be added, and as a tool for comparing a new approach against existing ones. We also hope that the simplicity of these tools will allow beginners to experiment with a more advanced toolset, without being buried in implementation details.

Note: despite its simplicity of use, Stable Baselines3 (SB3) assumes you have some knowledge about Reinforcement Learning (RL). You should not utilize this library without some practice. To that extent, we provide good resources in the documentation to get started with RL.

Main Features

The performance of each algorithm was tested (see *Results* section in their respective page), you can take a look at the issues #48 and #49 for more details.

Features	Stable-Baselines3
State of the art RL methods	:heavy_check_mark:
Documentation	:heavy_check_mark:
Custom environments	:heavy_check_mark:
Custom policies	:heavy_check_mark:
Common interface	:heavy_check_mark:
Ipython / Notebook friendly	:heavy_check_mark:
Tensorboard support	:heavy_check_mark:
PEP8 code style	:heavy_check_mark:
Custom callback	:heavy_check_mark:
High code coverage	:heavy_check_mark:
Type hints	:heavy_check_mark:

Planned features (v1.1+)

Please take a look at the Roadmap and Milestones.

Migration guide: from Stable-Baselines (SB2) to Stable-Baselines3 (SB3)

A migration guide from SB2 to SB3 can be found in the documentation.

Documentation

Documentation is available online: https://stable-baselines3.readthedocs.io/

RL Baselines3 Zoo: A Collection of Trained RL Agents

RL Baselines3 Zoo. is a collection of pre-trained Reinforcement Learning agents using Stable-Baselines3.

It also provides basic scripts for training, evaluating agents, tuning hyperparameters, plotting results and recording videos.

Goals of this repository:

1. Provide a simple interface to train and enjoy RL agents
2. Benchmark the different Reinforcement Learning algorithms
3. Provide tuned hyperparameters for each environment and RL algorithm
4. Have fun with the trained agents!

Github repo: https://github.com/DLR-RM/rl-baselines3-zoo

Documentation: https://stable-baselines3.readthedocs.io/en/master/guide/rl_zoo.html

SB3-Contrib: Experimental RL Features

We implement experimental features in a separate contrib repository: SB3-Contrib

This allows SB3 to maintain a stable and compact core, while still providing the latest features, like Truncated Quantile Critics (TQC) or Quantile Regression DQN (QR-DQN).

Documentation is available online: https://sb3-contrib.readthedocs.io/

Installation

Note: Stable-Baselines3 supports PyTorch 1.4+.

Prerequisites

Stable Baselines3 requires python 3.6+.

Windows 10

To install stable-baselines on Windows, please look at the documentation.

Install using pip

Install the Stable Baselines3 package:

```
pip install stable-baselines3[extra]
```

This includes an optional dependencies like Tensorboard, OpenCV or `atari-py` to train on atari games. If you do not need those, you can use:

```
pip install stable-baselines3
```

Please read the documentation for more details and alternatives (from source, using docker).

Example

Most of the library tries to follow a sklearn-like syntax for the Reinforcement Learning algorithms.

Here is a quick example of how to train and run PPO on a cartpole environment:

```
import gym

from stable_baselines3 import PPO

env = gym.make('CartPole-v1')

model = PPO('MlpPolicy', env, verbose=1)
model.learn(total_timesteps=10000)

obs = env.reset()
for i in range(1000):
    action, _states = model.predict(obs, deterministic=True)
    obs, reward, done, info = env.step(action)
    env.render()
    if done:
      obs = env.reset()

env.close()
```

Or just train a model with a one liner if the environment is registered in Gym and if the policy is registered:

```
from stable_baselines3 import PPO

model = PPO('MlpPolicy', 'CartPole-v1').learn(10000)
```

Please read the documentation for more examples.

Try it online with Colab Notebooks !

All the following examples can be executed online using Google colab notebooks:

- Full Tutorial
- All Notebooks
- Getting Started
- Training, Saving, Loading
- Multiprocessing
- Monitor Training and Plotting
- Atari Games
- RL Baselines Zoo
- PyBullet

Implemented Algorithms

Name	Recurrent	`Box`	`Discrete`	`MultiDiscrete`	`MultiBinary`	Multi Processing
A2C	:x:	:heavy_check_mark:	:heavy_check_mark:	:heavy_check_mark:	:heavy_check_mark:	:heavy_check_mark:
DDPG	:x:	:heavy_check_mark:	:x:	:x:	:x:	:x:
DQN	:x:	:x:	:heavy_check_mark:	:x:	:x:	:x:
HER	:x:	:heavy_check_mark:	:heavy_check_mark:	:x:	:x:	:x:
PPO	:x:	:heavy_check_mark:	:heavy_check_mark:	:heavy_check_mark:	:heavy_check_mark:	:heavy_check_mark:
SAC	:x:	:heavy_check_mark:	:x:	:x:	:x:	:x:
TD3	:x:	:heavy_check_mark:	:x:	:x:	:x:	:x:

Actions `gym.spaces` : * `Box` : A N-dimensional box that containes every point in the action space. * `Discrete` : A list of possible actions, where each timestep only one of the actions can be used. * `MultiDiscrete` : A list of possible actions, where each timestep only one action of each discrete set can be used. * `MultiBinary` : A list of possible actions, where each timestep any of the actions can be used in any combination.

Testing the installation

All unit tests in stable baselines3 can be run using `pytest` runner:

```
pip install pytest pytest-cov
make pytest
```

You can also do a static type check using `pytype` :

```
pip install pytype
make type
```

Codestyle check with `flake8` :

```
pip install flake8
make lint
```

Projects Using Stable-Baselines3

We try to maintain a list of project using stable-baselines3 in the documentation, please tell us when if you want your project to appear on this page ;)

Citing the Project

To cite this repository in publications:

```
@misc{stable-baselines3,
  author = {Raffin, Antonin and Hill, Ashley and Ernestus, Maximilian and Gleave, Adam and Kanervisto, Anssi and Dormann, Noah},
  title = {Stable Baselines3},
  year = {2019},
  publisher = {GitHub},
  journal = {GitHub repository},
  howpublished = {\url{https://github.com/DLR-RM/stable-baselines3}},
}
```

Maintainers

Stable-Baselines3 is currently maintained by Ashley Hill (aka @hill-a), Antonin Raffin (aka @araffin), Maximilian Ernestus (aka @ernestum), Adam Gleave (@AdamGleave) and Anssi Kanervisto (@Miffyli).

Important Note: We do not do technical support, nor consulting and don't answer personal questions per email. Please post your question on the RL Discord, Reddit or Stack Overflow in that case.

How To Contribute

To any interested in making the baselines better, there is still some documentation that needs to be done. If you want to contribute, please read **CONTRIBUTING.md** guide first.

Acknowledgments

The initial work to develop Stable Baselines3 was partially funded by the project *Reduced Complexity Models* from the *Helmholtz-Gemeinschaft Deutscher Forschungszentren*.

The original version, Stable Baselines, was created in the robotics lab U2IS (INRIA Flowers team) at ENSTA ParisTech.

Logo credits: L.M. Tenkes