tensorflow / agents

Efficient Batched Reinforcement Learning in TensorFlow

#reinforcement-learning #tensorflow #multi-processing #artificial-intelligence #python #vectorized-computation #control

5 commits | **1** branch | **1** release | **3** contributors | Apache-2.0

Branch: **master** | **New pull request** | **Find file** | **Clone or download**

danijar Update link to the paper. Latest commit `19e5e55` 4 days ago

agents	Fix typos in network comments. (#2)	5 days ago
.gitignore	Initial release.	8 days ago
AUTHORS	Initial release.	8 days ago
CONTRIBUTING.md	Initial release.	8 days ago
LICENSE	Initial release.	8 days ago
README.md	Update link to the paper.	4 days ago

setup.py	Update setup.py and upload to PyPi.	5 days ago

README.md

TensorFlow Agents

This project provides optimized infrastructure for reinforcement learning. It extends the OpenAI gym interface to multiple parallel environments and allows agents to be implemented in TensorFlow and perform batched computation. As a starting point, we provide BatchPPO, an optimized implementation of Proximal Policy Optimization.

Please cite the TensorFlow Agents paper if you use code from this project in your research:

```
@article{hafner2017agents,
  title={TensorFlow Agents: Efficient Batched Reinforcement Learning in TensorFlow},
  author={Hafner, Danijar and Davidson, James and Vanhoucke, Vincent},
  journal={arXiv preprint arXiv:1709.02878},
  year={2017}
}
```

Dependencies: Python 2/3, TensorFlow 1.3+, Gym, rumamel.yaml

Instructions

Clone the repository and run the PPO algorithm by typing:

```
python3 -m agents.scripts.train --logdir=/path/to/logdir --config=pendulum
```

The algorithm to use is defined in the configuration and `pendulum` started here uses the included PPO implementation. Check out more pre-defined configurations in `agents/scripts/configs.py`.

If you want to resume a previously started run, add the `--timestamp=<time>` flag to the last command and provide the timestamp in the directory name of your run.

To visualize metrics start TensorBoard from another terminal, then point your browser to `http://localhost:2222`:

```
tensorboard --logdir=/path/to/logdir --port=2222
```

To render videos and gather OpenAI Gym statistics to upload to the scoreboard, type:

```
python3 -m agents.scripts.visualize --logdir=/path/to/logdir/<time>-<config> --outdir=/path/to/outdir/
```

Modifications

We release this project as a starting point that makes it easy to implement new reinforcement learning ideas. These files are good places to start when modifying the code:

File	Content
`scripts/configs.py`	Experiment configurations specifying the tasks and algorithms.
`scripts/networks.py`	Neural network models defined as TensorFlow RNNCells.
`scripts/train.py`	The executable file containing the training setup.
`ppo/algorithm.py`	The TensorFlow graph for the PPO algorithm.

To run all unit tests, type:

```
python3 -m unittest discover -p "*_test.py"
```

For further questions, please open an issue on Github.

Implementation

We include a batched interface for OpenAI Gym environments that fully integrates with TensorFlow for efficient algorithm implementations. This is achieved through these core components:

- **`agents.tools.wrappers.ExternalProcess`** is an environment wrapper that constructs an OpenAI Gym environment inside of an external process. Calls to `step()` and `reset()`, as well as attribute access, are forwarded to the process and wait for the result. This allows to run multiple environments in parallel without being restricted by Python's global interpreter lock.
- **`agents.tools.BatchEnv`** extends the OpenAI Gym interface to batches of environments. It combines multiple OpenAI Gym environments, with `step()` accepting a batch of actions and returning a batch of observations, rewards, done flags, and info objects. If the individual environments live in external processes, they will be stepped in parallel.
- **`agents.tools.InGraphBatchEnv`** integrates a batch environment into the TensorFlow graph and makes its `step()` and `reset()` functions accessible as operations. The current batch of observations, last actions, rewards, and done flags is stored in variables and made available as tensors.
- **`agents.tools.simulate()`** fuses the step of an in-graph batch environment and a reinforcement learning algorithm together into a single operation to be called inside the training loop. This reduces the number of session calls and provieds a simple way to train future algorithms.

To understand all the code, please make yourself familiar with TensorFlow's control flow operations, especially `tf.cond()`, `tf.scan()`, and `tf.control_dependencies()`.

Disclaimer

This is not an official Google product.