Deep Eligibility Traces

Introduction

This repository consists of implementations of Eligiblity Traces and corresponding algorithms in the deep learning setting. Algorithms are implemented in `PyTorch` and `Tensorflow 2.0` on a range of problems. Custom toy problems are provided in the `MDPs` folder.

Baseline Algorithms

Following are the baseline algorithms combined with trace-based updates-

Algorithm	Link	Implementation	Status
Sarsa	Sutton & Barto	`sarsa.py`	✓
Double Sarsa	Sutton & Barto	`doublesarsa.py`	✓
Q-Learning	Sutton & Barto	`qlearning.py`	✓
Double Q-Learning	Sutton & Barto	`doubleqlearning.py`	✓
Expected Sarsa	Sutton & Barto	`expectedsarsa.py`	✓
Double Expected Sarsa	Sutton & Barto	`doubleexpectedsarsa.py`	✓

Trace Algorithms

Following algorithms are available in the current version-

PyTorch

Trace	Baseline Algorithms	Link	Implementation	Status
Q(λ)	Q(1)	Sutton & Barto	`watkinsq.py`	✓
QET(λ)	Q(1)	Expected Eligibility Traces	`qet.py`	✓
Replacing Trace	• ☑ Sarsa • ☑ Q-learning • ☑ Expected Sarsa • ☑ Double Sarsa • ☑ Double Q-learning • ☑ Double Expected Sarsa	Sutton & Barto	`torch_traces.py`	✓

Trace	Baseline Algorithms	Link	Implementation	Status
Accumulating Trace	• ☑ Sarsa • ☑ Q-learning • ☑ Expected Sarsa • ☑ Double Sarsa • ☑ Double Q-learning • ☑ Double Expected Sarsa	Sutton & Barto	`torch_traces.py`	✓
Dutch Trace	• ☑ Sarsa • ☑ Q-learning • ☑ Expected Sarsa • ☑ Double Sarsa • ☑ Double Q-learning • ☑ Double Expected Sarsa	Sutton & Barto	`torch_traces.py`	✓

Tensorflow 2.0

Trace	Baseline Algorithms	Link	Implementation	Status
Q(λ)	Q(1)	Sutton & Barto	`watkinsq.py`	✓
QET(λ)	Q(1)	Expected Eligibility Traces	`qet.py`	✓
Replacing Trace	• ☑ Sarsa • ☑ Q-learning • ☑ Expected Sarsa • ☑ Double Sarsa • ☑ Double Q-learning • ☑ Double Expected Sarsa	Sutton & Barto	`tf_traces.py`	✓
Accumulating Trace	• ☑ Sarsa • ☑ Q-learning • ☑ Expected Sarsa • ☑ Double Sarsa • ☑ Double Q-learning • ☑ Double Expected Sarsa	Sutton & Barto	`tf_traces.py`	✓

Trace	Baseline Algorithms	Link	Implementation	Status
Dutch Trace	• ☑ Sarsa • ☑ Q-learning • ☑ Expected Sarsa • ☑ Double Sarsa • ☑ Double Q-learning • ☑ Double Expected Sarsa	Sutton & Barto	`tf_traces.py`	✓

Custom Environments

Following is the list of custom toy environments-

Environment Name	Link	Implementation
CyclicMDP	ESAC	link
OneStateMDP	Sutton & Barto	link
OneStateGaussianMDP	Sutton & Barto	link
GeneralizedCyclicMDP	motivated by ESAC	link
StochasticMDP	hDQN	link
MultiChainMDP	ET(λ)	link

Usage

To run an implementation, use the following command-

```
python main.py --configs configs/configs.yaml --log_dir log/ --env <ENVIRONMENT>
```

For example, to run Q-Learning on the CyclicMDP environment using PyTorch library, use the following-

```
python main.py --configs configs/configs.yaml --log_dir log/ --env CyclicMDP
```

This will train the agent with default arguments listed in `configs.yaml` file.

Following is an example to enter custom arguments for Q-Learning on the CartPole-v0 environment using PyTorch library with replacing trace and lambda=0.5-

```
python main.py --configs configs/configs.yaml --log_dir log/ --alg QLearning --env
CartPole-v0 --lib torch --trace replacing --lamb 0.5 --num_steps 10000
```

Citation

If you find these implementations helpful then please cite the following-

```
@misc{karush17eligibilitytraces,
  author = {Karush Suri},
  title = {Deep Eligibility Traces},
  year = {2021},
  howpublished = {\url{https://github.com/karush17/Deep-Eligibility-Traces}},
  note = {commit xxxxxxx}
}
```