

# Deep Eligibility Traces

---

## Introduction

This repository consists of implementations of Eligibility 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	<a href="#">Sutton &amp; Barto</a>	<a href="#">sarsa.py</a>	✓
Double Sarsa	<a href="#">Sutton &amp; Barto</a>	<a href="#">doublesarsa.py</a>	✓
Q-Learning	<a href="#">Sutton &amp; Barto</a>	<a href="#">qlearning.py</a>	✓
Double Q-Learning	<a href="#">Sutton &amp; Barto</a>	<a href="#">doubleqlearning.py</a>	✓
Expected Sarsa	<a href="#">Sutton &amp; Barto</a>	<a href="#">expectedsarsa.py</a>	✓
Double Expected Sarsa	<a href="#">Sutton &amp; Barto</a>	<a href="#">doubleexpectedsarsa.py</a>	✓

## Trace Algorithms

Following algorithms are available in the current version-

### PyTorch

Trace	Baseline Algorithms	Link	Implementation	Status
$Q(\lambda)$	$Q(1)$	<a href="#">Sutton &amp; Barto</a>	<a href="#">watkinsq.py</a>	✓
Replacing Trace	<ul style="list-style-type: none"> <li>✓ Sarsa</li> <li>✓ Q-learning</li> <li>✓ Expected Sarsa</li> <li>✓ Double Sarsa</li> <li>✓ Double Q-learning</li> <li>✓ Double Expected Sarsa</li> </ul>	<a href="#">Sutton &amp; Barto</a>	<a href="#">traces.py</a>	✓

---

Trace	Baseline Algorithms	Link	Implementation	Status
Accumulating Trace	<ul style="list-style-type: none"> <li>✓ Sarsa</li> <li>✓ Q-learning</li> <li>✓ Expected Sarsa</li> <li>✓ Double Sarsa</li> <li>✓ Double Q-learning</li> <li>✓ Double Expected Sarsa</li> </ul>	<a href="#">Sutton &amp; Barto</a>	<a href="#">traces.py</a>	✓
Dutch Trace	<ul style="list-style-type: none"> <li>✓ Sarsa</li> <li>✓ Q-learning</li> <li>✓ Expected Sarsa</li> <li>✓ Double Sarsa</li> <li>✓ Double Q-learning</li> <li>✓ Double Expected Sarsa</li> </ul>	<a href="#">Sutton &amp; Barto</a>	<a href="#">traces.py</a>	✓

## Custom Environments

Following is the list of custom toy environments-

Environment Name	Link	Implementation
CyclicMDP	<a href="#">ESAC</a>	<a href="#">link</a>
OneStateMDP	<a href="#">Sutton &amp; Barto</a>	<a href="#">link</a>
OneStateGaussianMDP	<a href="#">Sutton &amp; Barto</a>	<a href="#">link</a>
GeneralizedCyclicMDP	motivated by <a href="#">ESAC</a>	<a href="#">link</a>
StochasticMDP	<a href="#">hDQN</a>	<a href="#">link</a>
MultiChainMDP	<a href="#">ET(<math>\lambda</math>)</a>	<a href="#">link</a>

## 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 xxxxxxxx}
}
```