dennybritz / **reinforcement-learning**

Branch: **master** ▾ **reinforcement-learning** / **DQN** /

Create new file **Upload files** **Find file** **History**

praveen-palanisamy Fixed typo — Latest commit `10ce5dc` 13 days ago

..		
.gitignore	Added DQN script to run outside of notebook	a year ago
Breakout Playground.ipynb	Add AtariEnvWrapper	a year ago
Deep Q Learning Solution.ipynb	DQN copy_model_parameters memory leak fixed, tensorboard summaries up…	4 months ago
Deep Q Learning.ipynb	Fixed issues with the DQN in the exercise notebook	13 days ago
Double DQN Solution.ipynb	Fix parameter in Double DQN	a year ago
README.md	Fixed the typo	a year ago
dqn.py	Fixed typo	13 days ago

README.md

Deep Q-Learning

Learning Goals

- Understand the Deep Q-Learning (DQN) algorithm
- Understand why Experience Replay and a Target Network are necessary to make Deep Q-Learning work in practice
- (Optional) Understand Double Deep Q-Learning
- (Optional) Understand Prioritized Experience Replay

Summary

- DQN: Q-Learning but with a Deep Neural Network as a function approximator.
- Using a non-linear Deep Neural Network is powerful, but training is unstable if we apply it naively.
- Trick 1 - Experience Replay: Store experience `(S, A, R, S_next)` in a replay buffer and sample minibatches from it to train the network. This decorrelates the data and leads to better data efficiency. In the beginning, the replay buffer is filled with random experience.
- Trick 2 - Target Network: Use a separate network to estimate the TD target. This target network has the same architecture as the function approximator but with frozen parameters. Every T steps (a hyperparameter) the parameters from the Q network are copied to the target network. This leads to more stable training because it keeps the target function fixed (for a while).
- By using a Convolutional Neural Network as the function approximator on raw pixels of Atari games where the score is the reward we can learn to play many of those games at human-like performance.
- Double DQN: Just like regular Q-Learning, DQN tends to overestimate values due to its max operation applied to both selecting and estimating actions. We get around this by using the Q network for selection and the target network for estimation when making updates.

Lectures & Readings

Required:

- Human-Level Control through Deep Reinforcement Learning
- Demystifying Deep Reinforcement Learning

- David Silver's RL Course Lecture 6 - Value Function Approximation (video, slides)

Optional:

- Using Keras and Deep Q-Network to Play FlappyBird
- Deep Reinforcement Learning with Double Q-learning
- Prioritized Experience Replay

Deep Learning:

- Tensorflow
- Deep Learning Books

Exercises

- [OpenAI Gym Atari Environment Playground](Breakout Playground.ipynb)
- Deep-Q Learning for Atari Games
  - [Exercise](Deep Q Learning.ipynb)
  - [Solution](Deep Q Learning Solution.ipynb)
- Double-Q Learning
  - This is a minimal change to Q-Learning so use the same exercise as above
  - [Solution](Double DQN Solution.ipynb)
- Prioritized Experience Replay (WIP)