Isaiah Zeilman

CURRENT TASK 11/10/22: DEVELOPING SYMMETRY IN R.L. NETWORKS

Source code: https://github.com/izeilman/rl_work

Notes for all programs:

- Trained models are stored in .pkl files, loaded with ;

```
from pickle import load

df = load(open("q_agent.pkl", 'rb'))
print(df.Q) # loads the Q table for a trained agent
```

- Ran with play.py
  - Should be clickable
  - ran from the command line with “python play.py“ with command line options

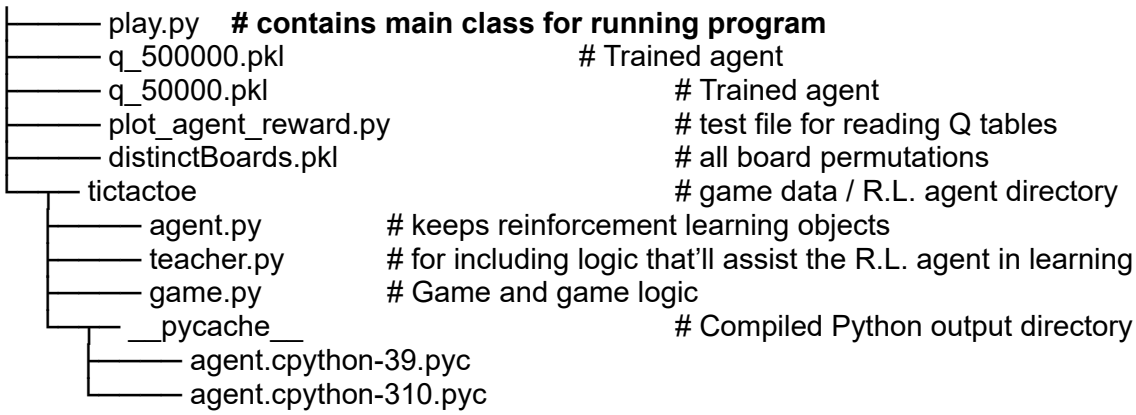
```
options:
  -h, --help            show this help message and exit
  -a {q,s}, --agent_type {q,s}
                        Specify the computer agent learning algorithm. AGENT_TYPE='q' for Q-learning and
                        AGENT_TYPE='s' for Sarsa-learning.
  -p PATH, --path PATH  Specify the path for the agent pickle file. Defaults to q_agent.pkl for AGENT_TYPE='q' and
                        sarsa_agent.pkl for AGENT_TYPE='s'.
  -l, --load            whether to load trained agent
  -t TEACHER_EPISODES, --teacher_episodes TEACHER_EPISODES
                        employ teacher agent who knows the optimal strategy and will play for TEACHER_EPISODES games
```

- Reward is supplied in game.py
- Learning function in agent.py
- teacher.py has game logic for training the agent
- agent.py keeps the R.L. objects

#BUGS/ISSUES:

- Getting a proper set of permutations for a connect-4 board
- Lack of way to compare symmetry in rotations
- Agents don’t always make the best moves while they will still act intelligent

```
Directory Tree:
└── rl_work                                                    # root folder for project
    ├── connect_4_rl                                           # root folder for connect-4
    │   ├── rotations.py                                       # Prototyping for rotating Q_tables
    │   ├── q_agent.pkl                                        # Trained agent
    │   ├── 10k.pkl                                            # Trained agent
    │   ├── merpk.pkl      # attempt at saving distinct board in game for use in plotting data
    │   ├── play.py   # contains main class for running program
    │   ├── test.py                                            # test file for reading Q tables
    │   ├── plot_agent_reward.py                               # test file for plotting agent reward
    │   └── connect_4                                          # game data / R.L. agent directory
    │       ├── agent.py          # keeps reinforcement learning objects
    │       ├── t.py              # a testing file I was using for prototyping a reward function
    │       ├── Capture.PNG       # screenshot of a bug, file will be cleared in release
    │       ├── teacher.py        # for including logic that'll assist the R.L. agent in learning
    │       ├── game.py           # Game and game logic
    │       └── __pycache__                                    # Compiled Python output directory
    │           ├── game.cpython-310.pyc
    │           ├── game.cpython-39.pyc
    │           ├── teacher.cpython-39.pyc
    │           ├── agent.cpython-39.pyc
    │           ├── agent.cpython-310.pyc
    │           └── teacher.cpython-310.pyc
    │
    ├── peg_solitaire_rl                                       # root folder for peg_solitaire
    │   ├── rotations.py                                       # Prototyping for rotating Q_tables
    │   ├── q_agent.pkl                                        # Trained agent
    │   ├── 10k.pkl                                            # Trained agent
    │   ├── possible_actions_test.py                           # file used for prototyping functions
    │   ├── q_agent_solitaire.pkl                              # Trained agent
    │   ├── play.py    # contains main class for running program
    │   ├── 100.pkl                                            # Trained agent
    │   ├── test.py               # test file currently containing code to read Q_tables
    │   ├── plot_agent_reward.py                               # test file for plotting agent reward
    │   └── peg_solitaire                                      # game data / R.L. agent directory
    │       ├── agent.py          # keeps reinforcement learning objects
    │       ├── teacher.py        # for including logic that'll assist the R.L. agent in learning
    │       ├── game.py           # Game and game logic
    │       └── __pycache__                                    # Compiled Python output directory
    │           ├── game.cpython-310.pyc
    │           ├── game.cpython-39.pyc
    │           ├── teacher.cpython-39.pyc
    │           ├── agent.cpython-39.pyc
    │           ├── agent.cpython-310.pyc
    │           └── teacher.cpython-310.pyc
    │
    └── ticktactoe_rl                                          # root folder for ticktactoe_rl
        ├── q_5M_reversed.pkl                                  # Trained agent
        ├── q_5000.pkl                                         # Trained agent
        ├── rotations.py                                       # Prototyping for rotating Q_tables
        └── q_5000000.pkl                                      # Trained agent
```


play.py **# contains main class for running program**
q_500000.pkl # Trained agent
q_50000.pkl # Trained agent
plot_agent_reward.py # test file for reading Q tables
distinctBoards.pkl # all board permutations
tictactoe # game data / R.L. agent directory
agent.py # keeps reinforcement learning objects
teacher.py # for including logic that'll assist the R.L. agent in learning
game.py # Game and game logic
__pycache__ # Compiled Python output directory
agent.cpython-39.pyc
agent.cpython-310.pyc