Assignment 3: Game Playing

Artificial Intelligence
WS 2023
Due: 2023-11-27, 12:00 noon

Katharina Hoedt
Verena Praher
Florian Schmid

1 Environment Activation / Framework re-installation

You will use the **same** `conda` **environment** that you set up for assignment 1.

Please note, that you will need to install the `graphviz` package via `conda`! It contains useful graph layouting routines. They produced the nice trees you could see in the lecture. If you don't do this, you'll have to contend with textual output alone.

However, we need to **download** a **new version** of the Python framework from MOODLE, and **re-install** the package it contains in this virtual `conda` environment.

- Activate the virtual environment you previously created:
  ```
  $ conda activate py311_ai_assignments
  ```
- For most shells, your command prompt should now have changed, to indicate that the virtual conda environment named `py311_ai_assignments` is now **active**.
- Download the file `ai_assignment3.zip` from the MOODLE course page.
- Unzip the `ai_assignment3.zip` into an empty directory.
- We will refer to this directory as your **base** directory.
- In your shell, navigate to the **base** directory.
- Issue the following command:
  ```
  $ pip install -e .
  $ conda install graphviz
  ```
  (It will install the new `ai_assignments` package and its dependencies into your active `conda` environment. Additionally, it will install the `graphviz` package and its dependencies into your active `conda` environment. It is a package with precompiled binaries, so that is why you have to use `conda install` here.)
- You are now ready to tackle the practical part of assignment 3! (With nice trees.)

2 Theoretical Questions (7 pts)

There is a **separate quiz** on MOODLE for theoretical questions, covering the algorithms that you will implement for this assignment sheet. For the **theoretical questions** you have an **unlimited number** of attempts, **but no feedback** whether or not your answers are correct.

Hint: For the given game trees, assume that nodes are visited from left to right.

3 Practical Part (Quiz + Code = 3 + 14 pts)

Note: 4 points for Minimax code, 5 points for AlphaBeta code, 5 points for Q-Learning code

Here is what to remember in general for the practical part:

- Each **problem instance** is encoded as a **JSON file**.
- This file is **attached** to a question in the MOODLE **quiz**.
- You will need to **download** the game instance and run your code to **play** it.
- You will then **copy** the **answer** into the **answer field** of the **question** in the **quiz**.
- In your shell, navigate to the **base** directory.
- Make sure your updated `conda` environment is **active**: (and you followed all instructions in section 1)
  ```
  $ conda activate py311_ai_assignments
  ```
- After you are done, **upload** the implementations of the 3 algorithms (`minimax.py`, `alphabeta.py` and `q_learning.py`) in a **zip** file to MOODLE!

Minimax and AlphaBeta

For this **practical** part of the assignment, you will be asked to implement the following adversarial search algorithms that you already heard about in the lecture:

- Minimax (`ai_assignments/search/adversarial/minimax.py`)
- AlphaBeta (`ai_assignments/search/adversarial/alphabeta.py`)

All practical assignments have a corresponding MOODLE quiz. Those quizzes consist of questions, which contain games with a particular starting position. You will have to apply your adversarial search algorithms ("game playing algorithms") to find a sequence of moves.

Note: If multiple paths lead to the same outcome for these algorithms, you may choose the first expanded / leftmost path.

Debugging Hints

Graphical Debugging your Game Trees: if you take a look at the file

```
ai_assignments/search/adversarial/drunk_player.py
```

and its contents, you'll see a pretty random game playing algorithm. You'll also find instructions on how you can render a bunch of nodes as trees, and maybe even annotate them. You can call the `drunk_player` like so:

```
$ python play.py <game-json-from-quiz> drunk_player
```

If you want to look at the solutions that an algorithm has produced, you can use

```
$ python replay.py <game-json-from-quiz> <player.path> [--show_possible]
```

which shows you the sequence of states and moves that your algorithms chose. The optional flag `--show_possible` shows you all the possible moves in each state.

Here is what to do for Minimax:

- Download the `Minimax` game starting position from MOODLE
- Implement the Minimax algorithm in
  `ai_assignments/search/adversarial/minimax.py`
- In a shell, try:
  `$ python play.py <game-json-from-quiz> minimax`
- Copy and paste the solution hash to the respective Minimax question in the quiz.

Here is what to do for AlphaBeta:

- Download the `AlphaBeta` game starting position from MOODLE
- Implement the AlphaBeta algorithm in
  `ai_assignments/search/adversarial/alphabeta.py`
  Hint: Be careful with the pruning condition, you might want to check out the pseudo-code shown in the lecture!
- In a shell, try:
  `$ python play.py <game-json-from-quiz> alphabeta`
- Copy and paste the solution hash to the respective AlphaBeta question in the quiz.

Q-Learning

For this **practical** part of the assignment, you will be asked to implement the following reinforcement learning algorithms that you already heard about in the lecture:

- Q-Learning (`ai_assignments/reinforcement_learning/q_learning.py`)

All practical assignments have a corresponding MOODLE quiz. Those quizzes contain questions, which contain training instances of a particular gridworld environment. You will have to train your agents with Q-Learning to find a policy.

General Hints for debugging

In case you are stuck, and your algorithm behaves in weird ways, or you cannot find the correct policy:

- generate a small training instance (size $= 3$)
  ```
  $ python generate.py gridworld 3 test/env.json
  ```
- run 1 episode only, compute the value function $Q(s, a)$ by hand
  and compare to the one your algorithm computed
- you can use
  ```
  $ python observe.py test/env.json test/q_learning.outcome
  ```
  to view a training instance, the environment, the immediate rewards, the policy, and value functions learned.

Here is what to do for Q-Learning:

- Download the training instance from MOODLE
- Implement the Q-Learning algorithm in
  `ai_assignments/reinforcement_learning/q_learning.py`
- In a shell, try:
  ```
  $ python train.py <training-json-from-quiz> q_learning
  ```
- Copy and paste the solution hash to the respective Q-Learning question in the quiz.