Planning Algorithms in AI
PS3: Value Iteration

Gonzalo Ferrer
Skoltech

24 November 2021

This problem set has two tasks and is individual work. You are encouraged to talk at the conceptual level with other students, discuss on the equations and even on results, but you may not show/share/copy any non-trivial code.

Submission Instructions

Your assignment must be submitted by 11:59pm on **December 5, Sunday**. You are to upload your assignment directly to Canvas as **two** attachments:

1. A `.tar.gz` or `.zip` file *containing a directory* named after your name with the structure shown below.

   ```
   yourname_ps3.zip:
   yourname_ps3/run.py
   yourname_ps3/utils.py
   yourname_ps3/vi.py
   yourname_ps3/plan_vi.mp4
   ```

2. A PDF with the answers to the questions below. Printed notebooks or scanned versions of hand-written documents, converted to PDFs, are perfectly acceptable (reduced size). No other formats (e.g., .doc) are acceptable. Your PDF file should adhere to the following naming convention: `yourname_ps3.pdf`.

Homework received after 11:59pm is considered late and will be penalized as per the course policy. The ultimate timestamp authority is the one assigned to your upload by Canvas. No exceptions to this policy will be made.

Problem Description

The environment is a simple 2d grid world with some obstacles. You can find the environment in the file `data_ps3.npz` and visualize it by simply executing the `run.py` script.

To support you and your algorithm, we provide a set of functions found in the `utils.py` file:

- `action_space`: This list contains all available actions.
- `plot_enviroment`: This function is for visualization.
- `transition_function`: This is an implementation of $x' = f(x, u)$. The action u is an element of `action_space`. It returns the next propagated state plus the results of the propagation: False, if it failed to propagate du to obstacles or end of bounds.

- `state_consistency_check`: This function checks whether or not the state is valid, i.e., is an obstacle or out of bounds.

Look for more details in help and in the code of these functions.

Task 1: Value Iteration G^*

In this task, you will calculate the optimal *cost-to-go* by using the Value Iteration algorithm explained in class.

A. (10 pts) Enumerate the action space. The coordinates of actions are $u = (\text{row}, \text{column})$.

B. (10 pts) Formulate the optimal cost-to-go G^* in recursive form.

C. (30 pts) Implement the VI algorithm for infinite length sequences. To show this, you are asked to include a picture of the final G^*

   The cost of traversing each node $l(x, u) = 1$ only if propagation is possible (there is not obstacle or out of bounds).

   Hint: You can implement a convergence criterion or simply run for a large number of iteration, say 100. Both options will be correct.

D. (10 pts) Experiment with different number of iterations. Start with a 1 iteration VI, describe the results obtained and reason why.

Task 2: Calculate a plan with VI

A. (10 pts) Formulate how to obtain the optimal policy u^* from G^*.

B. (20 pts) Implement an algorithm to obtain the optimal policy u^* from G^*. This policy can be a table. To test this, start at an initial position and execute the result of your policy and the transition function until you reach the goal. You will upload the video, that should be automatically generated if using the code in `run.py`.

C. (10 pts) Experiment with different parameters, such as starting points, the order of the states you use in VI (for loops) and the order of the actions. Explain your observations.