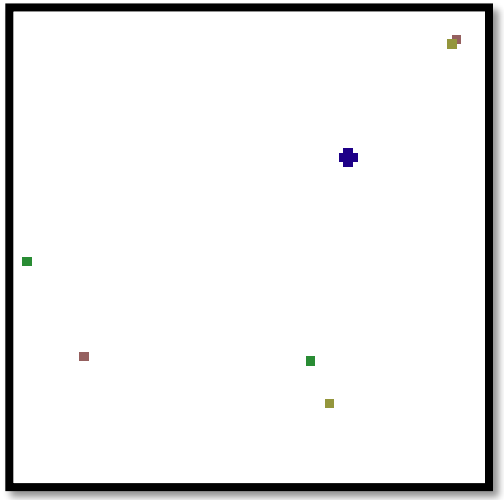
Introduction to Artificial Intelligence and Machine Learning

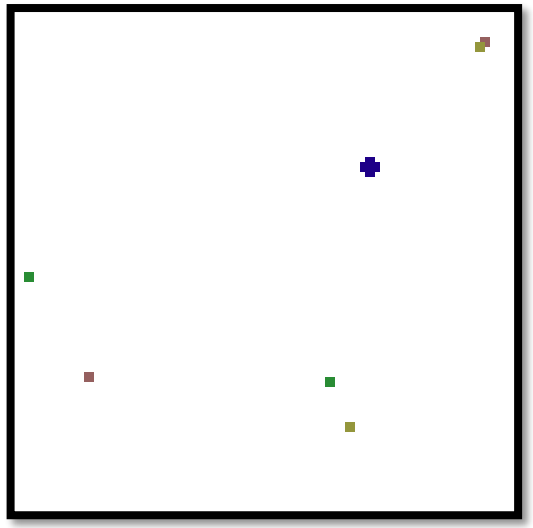
Final Project : Water World

Introduction



- A food-eating game!
- Three different kinds(colors) of food.
- Get points if you eat any one of the food.
- Observe the scores you get and design your own agent.

Problem Setup



- Map size: 200 * 200.
- Agent to control : 1 Blue dot .
  - Initialization: Center of the map.
- Actions : up, down, left, right, stay .
- Foods to eat : 3 red dots, 3 yellow dots, 3 green dots.
  - Initialization : random location of the map.
  - Breeding : Will breed a same-color dot at random location if you eat one.
  - Moving :
    - init: random dx, dy, velocity .
    - Rebound when hit a wall.
- Termination : after 10,000 steps.

Problem Setup(cont’d)

- Setup the game environment using the default settings :
  - game = WaterWorld() #no need to input any parameters.

- You can change the settings of the game environment to observe the changings, but TAs will run your code under default settings.

Files

- Key files:
  - ple/ple.py
  - ple/games/WaterWorld.py
- Files you can ignore:
  - ple/games/utils/*
  - ple/games/base/*
  - ple/games/premetives.py

Useful functions

- ple.py
  - act(action)
  - score()
  - getScreenRGB()
  - getScreenGrayscale() #[0.21, 0.72, 0.07]
  - getScreenDims() #(200,200)
  - getGameStateDims()
  - getGameState()

Useful settings

```
p = ple(game,force_fps = False, display_screen = True,
        state_preprocessor = function)
# force_fps : slower speed if False, will speed up if True.
# display_screen : display screen or not.(still open a window if False)
# state_preprocessor : function using to preprocess the state.
```

・You can add these two lines of code on the RHS in your code to hide the pygame window.

```
os.putenv('SDL_VIDEODRIVER', 'fbcon')
os.environ["SDL_VIDEODRIVER"] = "dummy"
```

Report

- 1. How to execute your code? (important)
- 2. Briefly describe your idea and what you've tried.
- 3. Results.
- 4. Any other things you want to share with TAs.
- 5. References.

Evaluation

- TAs will run your code for 10 times and get the average score as your final score. That is, your code should execute one round of the game.
- Print only the score your agent get.
  - E.g print p.score()
- To be fair, codes will be run at the same random state.

Environment and packages

- Python 2.7.14
- Packages in need:
  - numpy
  - pygame
  - pillow
- Allowed packages:
  - PyTorch 1.0.0+
  - keras 2.2.4+
  - Tensorflow 1.14.0+
  - pandas, scipy, scikit-learn …
- Feel free to contact TAs if you have difficulty in installing packages in need or if you want to use other packages.

Submission

- Your submission should include the following files:
  - A folder named your StudentID which contains
    - report_StudentID.pdf
    - MyAgent.py
    - Other python files and model files for this project.
- DO NOT upload ./ple.
- Submit to CEIBA.
- Compress the folder in a zip file.
- Deadline: 2020/1/15, 27:00

```
StudentID.zip
 StudentID
   ├ report_StudendID.pdf
   ├ MyAgent.py
   ├ model
   ├   …
```

Submission (cont’d)

- If your model file cannot be uploaded to CEIBA due to the size limit (15 MB), you may upload it to other cloud service platforms (e.g., Dropbox). However, your python program should be able to download the model automatically.

- No plagiarism is allowed.

11/19/2019 update

- Baselines :
  - Simple : score 70 points (5/10)
  - Medium : score 100 points (7/10)
  - Strong : score 150 points (8.5/10)
- Your rank will be taken into consideration !