IS 448 Mini project

Time to play some Pacman but with a *twist*!!!

You are not playing for the Pacman but against it. Your goal is to design 2 ghosts in the Pacman environment. The ghost agents should try to catch the Pacman as soon as possible. If the Pacman eats all the capsules before your ghost agents capture it then you will lose. The ghost agents also receive a time penalty, so the more time you take to capture the Pacman, the less score you will get.

Pacman agent learns its policy by training against ghosts which are moving randomly. You will be using this learned Pacman agent to train your ghost agents. Your learned ghost agents will then be tested against the same Pacman agent for a given number of test games.

The training games (for both Pacman and ghosts) are run in quiet mode by default, with no GUI (or console) display. Once training is complete, the game will enter testing mode. When testing, Ghosts's learning rate and epsilon parameter will be set to 0.0, effectively stopping Q-learning and disabling exploration, in order to allow Ghosts to exploit their learned policy.

Test games are shown in the GUI by default.

Setup :

In the game environment we have, Pacman Agent, and 2 RL Ghost agents.

python pacman.py -x 3 -y 2 -n 10 -g QLearningGhostAgent -a extractor=GhostAdvancedExtractor -p RandomPacmanAgent

NOTE : For running without GUI use –t
e.g
python pacman.py -x 3 -y 2 -n 10 -g QLearningGhostAgent -a extractor=GhostAdvancedExtractor -p RandomPacmanAgent –t

Output of text display is provided in file textDisplay.txt

For running in GPU Cluster, refer below.

This command will train the Smart Pacman agent for 50 Episodes, then 3 episode for ghost agents with RandomPacmanAgent, followed by 2 episode of training with Smart Pacman Agent and 5 episode for testing. Testing is done with Smart Pacman Agent. Here ghost agent will use learning agent described in QLearningGhostAgent class in qlearningGhostAgents.py file and use feature extractor defined in class GhostAdvancedExtractor in ghostfeatureExtractors.py file.

-x : Number of episodes for training ghost agents with Random/Simple Pacman Agent
-y : Number of episodes for training ghost agents with Smart Pacman Agent
-n : Total number of episodes.
So, n-x-y episodes will be for testing episodes.

The default learning parameters are epsilon=0.05,gamma(discount factor) = 0.8, alpha(learning rate) = 0.2

If you want to experiment with learning parameters, you can use the option -a,
For example run the following command to change the value of alpha, gamma and epsilon.

python pacman.py -x 3 -y 2 -n 10 -g QLearningGhostAgent -a extractor=GhostAdvancedExtractor,alpha=0.7,gamma=0.9,epsilon=0.2 -p RandomPacmanAgent

If you don't specify -p option a Pacman agent which uses simple features (described in simplefeatureextractor.py) and is trained using a single episode will be used.

For instance,

python pacman.py -x 3 -y 2 -n 10 -g QLearningGhostAgent -a extractor=GhostAdvancedExtractor,alpha=0.7,gamma=0.9,epsilon=0.2

will train the Smart Pacman agent for 50 Episodes, then 3 episode for ghost agents with a **Simple Pacman Agent**, followed by 2 episode of training with Smart Pacman Agent and 5 episode for testing. Testing is done with Smart Pacman Agent.

If you don't specify -g and -a options then randomGhost agent will be used.

For instance,

python pacman.py -x 3 -y 2 -n 10

Apart from this two files : qlearningGhostAgents.py and ghostfeatureExtractors.py, you should not change any other files.

Episodes:

The current episode ends and a new episode starts under following conditions:

- If the ghosts are scared and pacman eats any of it.
- If the ghosts are not scared and any of them eats pacman.
- If pacman eats all the capsules before ghosts eats the pacman.
- Maximum 500 time steps are reached. i.e, after 500 time steps current episode ends and a new episode starts.

Reward structure for Ghost :

You will receive -1 score at each time step and if any of the ghost catches the pacman then you get +500 score.

Evaluation Criteria :

- Smart Pacman Training Episodes : 50
- Ghost Training Episodes : 100
- Test Episodes : 500
- Average score over Test Episodes (500)

We will train your Ghost agent(s) for 100 episodes, and test them for the next 500 episodes. If you think 100 episodes are not enough for training your ghost agent(s) from scratch, you can provide us ghost agents which have pre-trained components (such as pre-trained neural networks etc). The code for training your ghost(s) should also be part of the final deliverable.

Hints/Tips:

- QLearningGhostAgent class has agentIndex parameter which can be used to take ghost specific decisions. Suppose if you want your first ghost agent to be a random agent and second ghost agent to use the learned values, you can write following logic in getAction function in qlearningGhostAgent class.

  ```
  if self.agentIndex == 1:
      return random.choice(self.getLegalActions(state))
  else:
      #Use learning
  ```

- **Directional Ghost Agent :**
  We have added a class DirectionalGhost in ghostAgents.py, this ghost is not a learning agent, at each time step, it simply calculates distance to pacman and rush towards it, or flee when scared.

  To run DirectionalGhostAgent use command:
  python pacman.py -x 3 -y 2 -n 10 -g DirectionalGhost

Log file (Directional_Vs_SmartPacman_log1000.txt) is provided for settings 1000 testing episodes, Directional Ghost is winning 610 times (out of 1000) with an average score of 238.2215.

Note: When you run it, you may not get the same score as ours due to randomness in the environment. Anyways this is for demonstration. The main takeaway from this is using basic features can help in winning against smart pacman.

You can use the feature directional ghost agent is using, but you need to come with more sophisticated features and use it in your learning ghost agent.

Benchmark score : Win rate : 0 and Average Score : -25

The benchmark score is obtained by running trained pacman against random ghosts for 500 testing episodes. Your trained ghost agents should be able to perform better than the random ghosts. You can compare against this score to evaluate the performance of your ghost agents.

Deliverables:

1. **Code file** - qlearningGhostAgents.py and ghostfeatureExtractors.py. If you want to use neural network based approaches, you can use same files and modify accordingly. **Only submit these 2 files.**

2. **Project report** – Maximum 6 pages providing the details of your algorithm. The report should mention each person's contribution.

For Running Pacman Project in GPU Cluster :

As our server is only command line, we need to disable GUI.

We are providing an additional submit script "submitPacman.py" which should be in your home directory "/home/<username>/".

Use the command to run :

$python submitPacman.py –n 1 –g 1 –p Pacman_MiniProject –s "pacman.py –x 1 –y 1 –n 4 -t"

Here,

Options after submitPacman.py are your job options

-n : Node ID
-g : GPU ID
-p : Project Name
-s : Script name

Options within quotes are your pacman project options,
"pacman.py –x 1 –y 1 –n 4"

-x : Number of episodes for training ghost agents with Random/Simple Pacman Agent
-y : Number of episodes for training ghost agents with Smart Pacman Agent
-n : Total number of episodes.
-t : Disables GUI