Guide to CE811 Pacman Engine

M. Fairbank. November 2021
Please email me with errors / requests for clarifications. m.fairbank@essex.ac.uk

1. Getting the Python Game Engine

Obtain the ce811 python pacman engine, ce811-pacman-engine-main.zip and unzip it.

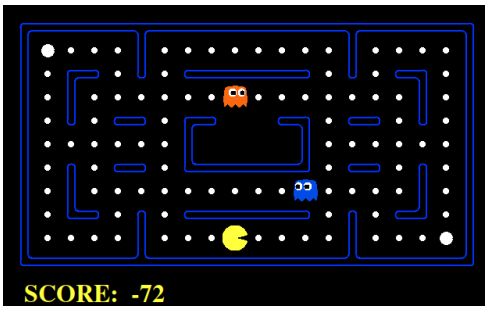
From the command line, go into the main folder, and run *pacman.py* to play an interactive game (keyboard controlled).

To run an interactive game:

python3 pacman.py

Note: you may need to replace the command python3 by your own command to run python3, e.g. "python pacman.py"

This should launch the game and allow you to play with keyboard control:



Notes:

1. It is possible to change the game map with the --layout option but we will not do this. Please always use the default layout ("mediumClassic"), which is the layout shown above.
2. This Pacman code base was originally created by Berkeley University of California, and we are using it in CE811 with permission.

1.1. Creating an Agent

To play a game where pacman is controlled by an AI agent:
python3 pacman.py -p GoWestAgent

This code is defined in searchAgents.py. Have a look at it.

To define our own agent, add a class to searchAgents.py which extends Agent class, like GoWestAgent, but give it a different class name it.

Exercise: Create an agent GoEastAgent. It is just like the GoWestAgent but always go east.

Run the game, using your new agent? Does it work? Don't forget to only go east if east is a valid action, otherwise the engine will raise an exception.

Note: You can create new agents in any file you like as long as its filename ends in gents.py, and your Python class is inherited from Agents.

1.2. Playing many games without graphics

To make the game non visual, use -q or --quietTextGraphics. This just displays the score. E.g. *python3 pacman.py -p GoWestAgent -q*

To make the game play 20 games, use -n 20 or --numGames 20. E.g. *python3 pacman.py -p GoWestAgent -q -n 20*

Run your GoEastAgent for 100 games, what is its win rate/average score?

Answer: Win Rate 0, average score: something around -531.057

Note that this version of pacman is a *stochastic game.* Hence you may get a slightly different average game score from my answer above. You can always reduce the randomness (reduce the sampling error) by playing a higher number of games.

2. Creating a Genetic Algorithm Agent

This page contains some advice about creating a genetic algorithm agent.

Fitness Evaluation

Whenever your genetic algorithm runs 1 iteration, it needs to evaluate the fitness of each member of its population. So there needs to be a fitness function defined. Of course, the simplest fitness function is the simply the "game score" (the number of points the agent receives in the game).

However, since Pacman is a stochastic game, you probably want to evaluate the game score several times for any one chromosome fitness evaluation. So to evaluate the fitness of one chromosome (one individual), we should probably play the game 10 times or so, and take the average game score as the fitness of that particular chromosome.

To play the game several times, you can use the *runGames(...)* function of *pacman.py*, which returns a list of games. Then you can evaluate the average game score from each of the games in that list. Once you have the fitness of a chromosome evaluated (or the fitness of each individual in a population), you can then implement an iteration of your genetic algorithm, and repeat.

Chromosome Interpreter

You also need to decide how a chromosome should get translated into an agent's playing strategy. Remember, a GA requires that different chromosomes have different fitnesses!

A good suggestion for this could come from Section 2.4.2.1 of the course textbook "Artificial Intelligence and Games" by G. Yannakakis and J. Togelius (available on line at university library and also at http://gameaibook.org/). This method maps nicely onto the *multiAgents.py*'s class *ReflexAgent*. This relies upon a helper method *evaluationFunction* which would need rewriting following Yannakakis and Togelius' method suggestion. The *evaluationFunction*'s behaviour must depend upon your chromosome's values. Using this method would likely mean the chromosome image above should be changed to show a list of floats as opposed to a list of binary bits.

Confusingly there are two "evaluations" nested here - the genetic algorithm's fitness evaluation (i.e. play 10 games and see what the average score is), and also the agent's "evaluationFunction" method (which needs calling several times, every single step of the main game loop).

Chromosome Mutation Operator

As the chromosome is likely to be a vector of floating point numbers (unlike those binary single digits shown in the image above), to implement occasional mutations, you probably want to add a small random floating point number to one or more of the genes in the chromosome.