Outline project specification

Overview

Ms Pac-Man is an interesting game from an AI perspective as it is a stochastic, dynamic, multi-agent environment, and there has been a reasonable amount of work done in the subject. Recent work has shown that Monte-Carlo tree search (MCTS) can be utilised by game-playing agents and, further, that it exhibits a considerable improvement on prior rule-based or simple tree search agents. MCTS works by running a number of simulated games from the current position and effectively builds a tree of the average consequences of selecting different actions. While it is running simulated games, the ghosts' behaviour must be simulated using a model; intuitively, and indeed empirically, when the model of the ghost behaviour closely matches the behaviour of the actual component, the algorithm performs well and high scores are achieved. However, when the model and actual behaviour differ, the score achieved can be as little as a tenth of that seen when they match.

The aim of this project is to investigate using machine learning algorithms to attempt to learn a model of the ghost behaviour in-game, so that the base MCTS algorithm can make more accurate simulations. Initially I will have to gain a deeper understanding of machine learning and the algorithms available: to this end I have enrolled in a Stanford University Coursera course in Machine Learning. Once I have the basic understanding I will also consult relevant papers on the subject.

Several approaches can be considered in this problem. Initially, a "generic" ghost model can be built with various parameters such as "aggressiveness" and whether or not it runs away from Pac-Man when the ghosts are edible. Clearly, there may be other useful parameters which will be discovered during the course of the project. This generic model can then be constructed with different parameters to evaluate how its behaviour compares to the various existing models of ghost behaviour. Then, data recorded from games will be analysed to attempt to fit appropriate parameters to get behaviour matching that observed. Eventually, the machine learning algorithm can be combined with the existing MCTS agent to construct an agent which performs this analysis during a game. This type of algorithm is an example of supervised machine learning.

Another method to consider will be to take a large body of game play data from various ghost opponents and attempt to find patterns in it. These patterns may correspond to different agents, and it may be sufficient to develop a few agents with different behaviours and attempt to classify the behaviour of the current opponent by matching it to the agent with the closest behaviour.

Objectives and project plan

To be completed before January:

1. Learn about machine learning using Stanford course
2. Evaluate other applications of machine learning to game-playing agents in the literature

January:

3. Examine the code of various existing ghost agents and attempt to analyse how these might all be rewritten as a single agent with different parameters
4. Prototype a classification algorithm and train it with hundreds of recorded games with different ghost agents
5. Design poster

February:

6. Attempt to write the classification algorithm into the existing Monte Carlo tree search agent
7. Run lots of games to evaluate performance compared to plain MCTS version and generate graphs

March:

8. Write report
9. If there is time, look at unsupervised machine learning – this will involve prototyping an algorithm to look at hundreds of recorded games and attempt to cluster them into different ghost behaviours

Previous work

The agent will be written using the framework provided by the University of Essex, at www.pacman-vs-ghosts.net. Most other work on Ms Pac-Man has used this framework or its predecessors, and a competition is held annually which sees various Ms Pac-Man and ghost agents pitted against each other.

Historically the winning agents have been simple rule-based systems, however recently more elegant solutions have started to appear. Robles and Lucas [1] first suggested applying a kind of tree search to the problem, which on each game tick builds a tree of possible paths from the current position, containing nodes representing items such as pills and edible or non-edible ghosts. The agent did not use full adversarial search, and can be seen as an extension of a rule-based system, however it performed reasonably well, beating the other rule-based agents of the time.

Such systems are prone to so-called 'pincer attacks' due to their lack of foresight; in such an attack, the ghosts surround Ms Pac-Man, forcing it to make a move which results in it being trapped and for which losing a life becomes inevitable. In an attempt to mitigate this, Ikehato and Ito [2] presented perhaps the first

application of Monte-Carlo tree search to Ms Pac-Man. Their algorithm runs many simulated games from the current position and builds a tree which stores the fraction survivability for each choice; it then picks moves which maximise this survivability.

Another MCTS agent was presented by Samothrakis *et al.* [3], modelling the game as a 5 player game with a max^n game tree [4]. This proved to be a very strong player.

MacKenzie-Leigh *et al*. [5] demonstrated that an MCTS agent can be improved upon using an ensemble method which combines short- and long-range planning techniques with the medium-range MCTS algorithm. The agent developed for that work will form the basis of the learning agent; thus the focus can be on the actual machine learning objectives rather than the base MCTS agent which will merely use the output from the learning.

Methodology and evaluation

This project involves a significant research aspect, as well as learning the concepts to be employed. It is expected that various prototypes will be developed to aid learning. Then, lessons learnt from the prototypes can be used in a more formal plan for the final agent, along with UML diagrams and sequence diagrams / flow charts. Furthermore, being research oriented, the specification is reasonably loose and and the project's success will be evaluated on how much the machine learning component improves on the average score obtained by the MCTS algorithm acting alone. The agent will also be submitted to a competition next September to be compared against other agents.

Machine learning prototypes will be developed in Octave, an open source Matlab clone. This language is particularly suitable for very quickly creating prototypes which process large amounts of data. Other tooling may be developed in C# due to its performance, elegance, and existence of multi-paradigm constructs which make data processing easier. If the expressiveness of a functional language is required, it can also interface quite seamlessly with F#. The actual agent will necessarily be developed in Java as that is the language of the framework and competition. Should any offline data persistence be required, MongoDB will be employed due to its speed and ease of use.

Parts amenable to it will be unit tested using either jUnit or the in-built Visual Studio test tools, whichever applies.

Marking scheme

The marking scheme chosen is the “experimentation-based project” scheme; the majority of the project consists of research and prototyping with some development of tooling. It is the algorithm that is important rather than there being a single piece of software produced which can be evaluated at the end.

References

[1] David Robles and Simon Lucas, “A simple tree search method for playing Ms. Pac-Man,” in *Proceedings of the 5th International Conference on Computational Intelligence in Games*, 2009, pp. 249–255.

[2] Nozomu Ikehata and Takeshi Ito, “Monte-Carlo tree search in Ms. Pac-Man,” in *CIG*, Sung-Bae Cho, Simon M. Lucas, and Philip Hingston, Eds. 2011, pp. 39–46, IEEE.

[3] Spyridon Samothrakis, David Robles, and Simon Lucas, “Fast approximate max-n Monte Carlo tree search for Ms. Pac-Man,” *IEEE Transactions on Computation Intelligence and AI in Games*, vol. 3, no. 2, pp. 142–154, June 2011.

[4] Carol Luckhart and Keki B. Irani, “An algorithmic solution of n-person games,” in *AAAI’86*, 1986, pp. 158–162.

[5] Stewart MacKenzie-Leigh, John Levine and Matthias Lenz, “*An ensemble agent for Ms Pac-Man,*” Manuscript submitted for publication.