University of Groningen - FMNS
Machine Learning: Project Proposal

Learning to play Pac-Man with Reinforcement Learning

David Langbroek (S2226707), Davide Aurucci (S3253767)
Albert Segarra Roca (S3255050), Fthi Arefayne (S3074641)

November 28, 2016

This project will consist in the implementation of an artificial intelligence agent that will learn to play the **Pac-Man** arcade game using **Neural Networks with Reinforcement Learning**. The chosen programming language for the project is **Python 3**.

1 Application

The environment for the agent will be a simulation of the Pac-Man game. This simulation will be implemented using the rules of the official game, although some simplifications will be made:

- There will be a **single level** and a **single life**
- The behaviour of the **ghosts will be stochastic**, in the sense that when a ghost reaches a tile intersection, a new moving direction will be chosen uniformly at random. More complex strategies for the ghost movement like chasing and scattering might be considered as possible improvements if we have enough time.
- There will be **no fruit rewards**, only pellets

This simulation will be implemented by ourselves. We will use two (or more) maze layouts to be able to analyze how the learned strategies in each layout perform in the other and how the layouts affect the performance.

The inputs of the neural network will resemble the ones described in [2], using a **single action neural network** and considering the possibility to add/remove/tweak the described higher-order features to try to improve the learning process. This means we will not use the whole game state as an input to the neural network but instead we will extract only **relevant features**. Added to the state of the game, the action

(move left, right, up, down) will also be an input to the neural network. The single output of the neural network will be the **desirability of the action** given the feature-described game state.

We also plan to add a new reward factor in the game, which will reward faster game completion by the agent. This can be implemented by decreasing the game score by a certain amount every T game steps.

2 Methods

We will use **Q-learning** [1] as a model-free reinforcement learning technique. We will also use **BFS** and **A*** [3] algorithms to compute some of the game state features, when pathfinding is required.

3 Setup of experiments

We will evaluate the performance of the agent based on the number of wins and the total score of the games, which will depend on the number of pellets collected and the time spent collecting them. This will be averaged on a reasonable number of game plays in order to obtain significant results. Performance will be compared for different parameter values of the learning algorithm (number of hidden layer neurons, learning rate value, different input feature sets, etc.) and different maze layouts.

4 Planning

1. **1st December**: Implementation of the simulation model
2. **8 December**: Definition of the specific input features that will be used and implementation of the respective algorithms to compute them
3. **22 December**: Implementation of the full Q-learning algorithm
4. **12 January**: Improvements and pilot experiments
5. **19 January**: Finish experiments and initial version of the report
6. **24 January**: Final version of the report and presentation

References

[1] Wikipedia Q-Learning https://en.wikipedia.org/wiki/Q-learning

[2] Bom, Luuk, Ruud Henken, and Marco Wiering. "Reinforcement learning to train ms. pac-man using higher-order action-relative inputs." 2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL). IEEE, 2013.

[3] Introduction to A* http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html