08.05.2019

RL Course Project Proposal

Group Members: Julius Taylor, Jonas Mahn

Working Title: ReinforcementMan vs. The GhostMaster

The Idea (poetic) (resemblance to the classic PacMan scenario must be a coincidence)

A lone wanderer in a maze in the never-ending search for mysterious power-pills. The pills are plenty and the wanderer is fast and enduring so he would be safe & happy in his search, if it wasn't for a horrible enemy villain: The ghostmaster. The ghostmaster thrives for only one goal: To catch the wanderer and consume his life force. Controlling his ghosts, he acts without mercy to find and consume his prey. But luckily for the wanderer, his enemies' vessels aren't the fastest runners in the maze. The wanderer can outrun them easily, so the ghost master has to rely on his oversight knowledge and strategy, to achieve his unholy goal.

The Scenario

Two RL Agents playing against each other in a 2D maze like environment. The first agent, the wanderer, controls the movement direction of one entity and receives reinforcement by consumed power-ups that are distributed in the maze.

The second agent, the ghostmaster, controls the moving direction of multiple "ghost"-entities and receives a negative reward as long as it hasn't caught the wanderer.

State-Space Options

What the Agents see

a) The graphics output of the game environment / a binary state matrix representation of the game screen to reduce computation complexity (complete knowledge of the scene)
b) "Eyes" – Obstacle Proximity information in the moving direction of a controlled body:
    a. Could be a single number that is zero when nothing is in the way for the next t steps and increases when an obstacle (wall, opponent, power up?) gets closer.
    b. A vector that includes a. but also information about the type of obstacle (wall, opponent, power up)

    (partial knowledge of the scene)

We would be interested in an asymmetric scenario where the wanderer-agent uses a version of b) and the ghostmaster-agent uses a). A scenario where the ghostmaster-agent always knows the position the maze and the position of its ghost, but knows the position of the wanderer only when it is close to one of the ghosts, might also be interesting.

The Scene

- The wanderer, and the ghosts all have a fixed starting position.
- The maze could be either fixed or randomized after each episode.
- The speed of the moving entities could be varied.
- The maze is filled by power-ups, that can be consumed by the wanderer.

Action-Space

The Wanderer

An action is a four element one-hot binary vector, representing the possible movement directions: north, south, west, east. Thereby four actions are possible: {1,0,0,0}, {0,1,0,0}, {0,0,1,0}, {0,0,0,1}.

The GhostMaster

The ghostmaster shall control n ghosts. An action is an n*4 element n-hot binary vector, representing all 4^n possible combination of movement directions of the n ghosts.

Reinforcement Signals

The Wanderer

The wanderer-agent receives a reward for every consumed power-up and now reward if no power-up was consumed in a timestep.

The GhostMaster

The ghostmaster-agent receives a negative reward as long as the wanderer is not caught.

An Episode

1. The scene is reset, all moving entities are moved to their respective starting position, the scene is filled with power-ups.
2. In every timestep t:
    a. The agents receive the current state information and select their action
    b. The ghosts are moved according to the action selected by the ghostmaster-agent, the wanderer is moved according to the action selected by the wanderer-agent
    c. Rewards are calculated
    d. Agents are updated using the calculated reward
3. The episode ends when either the wanderer is caught or the maze is cleared of all power-ups.

Training

Simultaneous training both agents might prove inefficient and success might be hard to measure, therefore the agents will be training on fixed versions of their opponents. The first opponents could be just random or based on simple movement rules.

Goals

We hope to observe interesting tactical behavior of the agents like the ghostmaster using its ghosts to surround the wanderer. To achieve that we want to implement a Deep Q-Learning algorithm for the agents and setup the scene (maze, entity-speed) in a way that supports such behavior.

Tools

- Python (+ Tensorflow) for implementing the Agents
- Open AI Gym for the game environment