CS533-Assignment1-Report

Chunxiao Wang

The assignment is to model video game Pac-Man as a Markov Decision Process(MDP) under some simpler assumptions of ghost movements. Since the initial direction of the ghost movement is assumed to be random, the Pac-Man can be treated as a stochastic MDP. The state space S, the action space A, the transition space T and the reward function R are specified as follows.

1. **State Space S**
   Since we assume only eating a dot, eating a ghost and clearing a board get reward, the bonus items, e.g. cherry in the real game will be ignored here. The state space can be defined as

   $$S = (board\_state, Pac\_cur\_d, Pac\_pending\_d, ghost\_cur\_d, ghost\_eatable, ghost\_v)$$

   - *board_state*: it is the state of the board array with each array cell taking the possible value from 0(the cell is wall), 1(the cell is empty), 2(the cell is dot), 3(the cell is Pac-man), 4(the cell is ghost1), 5(the cell is ghost2), ... , k(the cell is ghostl)...
   - *Pac_cur_d*: it is the current movement direction of Pac-man with possible values *(mouth up, mouth down, mouth right, mouth left).*
   - *Pac_pending_d*: it is the pending movement direction of Pac-man with possible values *(mouth up, mouth down, mouth right, mouth left).* If the Pac-man is currently heading in a direction and the player press the controller, the Pac-man will try to head in a new direction from the pending directions. But the change could be successful or unsuccessful. If successful, the heading direction will change, otherwise, it will stay at the current direction waiting for the new decision of player.
   - *ghost_cur_d*: it is the current movement direction of Pac-man with possible values *(mouth up, mouth down, mouth right, mouth left).* Since we have simplified assumption about the movements of ghost, there is no necessary to include a *ghost_pending_d*.
   - *ghost_eatable*: a k column vector with boolean element indicating if the ith ghost is eatable.
   - *ghost_eatable_time*: a k column vector with nonnegative number indicating the remaining time the ith ghost it eatble.
   - *ghost_v*: the velocity of the ghosts.

2. **Action Space *A***
   The set of controller action with possible values $A =$ *(up, down, right, left)*. There could also be a *no-action* action which means the player doing nothing, but to simplify the situation, I choose to ignore it.

3. **Transition Function *T***
   The transition function specifies the probability of going to state $s' \in S$ after action $a \in A$ from starting state $s \in S$, which is $T(s, a, s')$. We first specify the starting state which is the state when the game begins. The *board_state* is how the walls, dots, Pac-man and so on are located in the beginning. Usually Pac-man starts at the bottom of the board and the ghost are in the center of the board. Assume PacMan begins facing in an arbitrary direction $d \in$ *Pac_cur_h*. The *Pac_pending_h* could also be set to d because the controller has not moved. The *ghost_eatable_time* can be set to 0 and *ghost_eatable* is initialized all elements being false since PacMan has not eaten a large dot yet. The *ghost_v* can be set as a positive number. When the controller is pressed, an action a is taken. The updates are specified as follows.

   - If PacMan reaches the boundary of the board and there is no wall there, it will came back to the board from the opposite side of board boundary.
   - When PacMan comes across a small dot, PacMan eats it. Then the dot disappears and the state of the current cell is going to update. When comes across a large dot, PacMan eats it. Then the dot disappears and the current cell is going to update. Also all the elements in *ghost_eatable* are set to be true, *ghost_eatable_time* is set to be a positive number and the *ghost_v* become a smaller numer. The *ghost_eatable_time* will decrease until it goes to 0 indicating ghosts are no longer eatable, *ghost_eatable* has to be set as false again and the *ghost_v* is set to be the initial velocity again.
   - When PacMan comes across with a ghost and *ghosts_eatable* is true indicating that the ghost is eatble, PacMan eats the ghost. The ghost will go back to the board center and the corresponding element in *ghosts_eatable* to set to be false.When PacMan comes across a ghost and *ghosts_eatable* is false for the particular ghost, PacMan dies.
   - The movement updates of ghost is given in the problem description. All ghosts will continue moving in its current direction as indicated in *ghost_cur_h*. When a ghost reaches a wall, it will randomly pick a possible direction, then we need to update the corresponding element in *ghost_cur_h*.
   - When there is no dots and large dots, PacMan wins the level, gets reward points and moves to the next level. The *ghost_v* will be larger than the velocity in current level.

4. **Reward Function *R***
   The R is the change of scores between consecutive states. It is 1, 100, 1000 if the Pac-man eats a dot, the Pac-man eats a ghost and all the dots are cleared respectively. The goal is to survive and get through more levels to accumulate more points. This reward function encourages PacMan to survive and get through more levels.