Obligatorio
Inteligencia Artificial

Ximena Moure – 175660

Ismael Puricelli – 187086

Índice

1. Minimax

1.1 Jusitificaciones

Para mejorar la *evaluationFunction* decidimos utilizar los siguientes elementos del juego:

- La distancia a la comida más cercana.
- La distancia al fantasma más cercano.
- La distancia al fantasma asustado más cercano.
- La cantidad de capsulas disponibles.

Con estos elementos hacemos una combinación lineal con coeficientes que son los que ponderan la importancia de cada uno para el pacman.

La distancia a la comida más cercana la multiplicamos por una variable llamada FOOD=-3. Por lo que cuanto más cerca este la comida mayor va a ser su valor.

A la distancia al fantasma más cercano le hacemos la inversa ya que cuanto más cercano este el mismo su valor deberá ser menor para priorizar los estados en los que el fantasma más cercano se encuentre más lejos.
Luego de hacer la inversa lo multiplicamos por la variable GHOST=-5. Esta variable es mayor a la variable FOOD ya que queremos lograr que el pacman se aleje de los fantasmas para que no pierda.

La distancia al fantasma asustado más cercano la multiplicamos por una variable llamada SCARED_GHOST=-5. Por lo que cuanto más cerca este el fantasma mayor va a ser su valor. Esto implica que el pacman va a priorizar el “comerse” los fantasmas asustados, es por esto también que el valor de SCARED_GHOST es mayor al valor de FOOD.

La cantidad de capsulas la multiplicamos por una variable llamada CAPSULE=-15. Con esto queremos lograr que el fantasma se coma las capsulas al pasar cerca de una. No se busca que se dirija directamente a las capsulas, pero sí que al pasar cerca de una se la coma.

Las variables mencionadas anteriormente se encuentran definidas al comienzo del archivo *miniMax.py*.

1.2 Respuestas a las preguntas

I) El ambiente se clasifica de la siguiente manera:

-Completamente observable: El agente tiene acceso total al estado del ambiente durante todo el juego.

-Determinista: El estado siguiente del ambiente se encuentra determinado por el estado actual y las acciones elegidas por el agente.

-Secuencial: La decisión actual tomada por el agente puede afectar las decisiones futuras.
Por ejemplo, si el pacman come una capsula, en el próximo estado podrá comer un fantasma, si no lo hubiese hecho no podría hacerlo.

-Estático: En este caso el ambiente no cambia mientras que el agente toma una acción, sino que los movimientos de los agentes (pacman y fantasmas) son secuenciales.

-Discreto: El agente tiene una cantidad finita de acciones a ejecutar sobre el ambiente.

-Multiagente: Existen varios agentes que interactúan con el ambiente durante el juego, el pacman y los fantasmas.

-Conocido: Se conocen todas las reglas del ambiente.

II) Asumiendo que el agente siempre juega minimax, entonces su comportamiento será mejor o peor dependiendo de la estrategia utilizada por los otros agentes (fantasmas).
Si el oponente utiliza una estrategia min, entonces el agente jugara de manera óptima ya que minimax se basa en asumir que el oponente juega una estrategia min.
Sin embargo, si el oponente utiliza una estrategia random, el agente juega de manera menos optima, porque asume que el mismo siempre elige la mejor jugada, pero no es el caso. Por lo que pueden existir jugadas más óptimas.

2. QLearning

2.1 Justificaciones

En el archivo *qlearning.py* en el constructor (*_init_*) se definen las siguientes variables:

- epsilon=0.01. Es el coeficiente para calcular la distribución de probabilidad.
- tau=1
- alpha=0.8. Es la constante de entrenamiento.
- MIN_Q_STATE_NOT_EXPLORED=0. Esta variable representa el valor de Q por defecto en caso de que no exista un valor para el par estado acción (Por ejemplo, para actualizar el valor de Q cuando queremos obtener el Q máximo del estado sucesor si este no existe).

2.2 Respuestas a las preguntas

I) El ambiente se clasifica de la siguiente manera:

-Parcialmente observable: El agente no tiene acceso total al estado del ambiente durante todo el juego ya que únicamente ve lo que se encuentra dentro de su *view_distance*.

-Determinista: El estado siguiente del ambiente se encuentra determinado por el estado actual y las acciones elegidas por el agente.

-Secuencial: La decisión actual tomada por el agente puede afectar las decisiones futuras.

-Estático: En este caso el ambiente no cambia mientras que el agente toma una acción, sino que los movimientos de los agentes (pacman y fantasmas) son secuenciales.

-Discreto: El agente tiene una cantidad finita de acciones a ejecutar sobre el ambiente.

-Multiagente: Existen varios agentes que interactúan con el ambiente durante el juego.

-Conocido: Se conocen todas las reglas del ambiente.

II) A medida que aumenta la *view_distance* del pacman, el mismo se comporta de manera más eficiente, y cuando es menor de manera contraria.

III) Aprende más rápido ya que en QLearning el agente puede explorar o elegir la mejor acción basándose en conocimiento previo.

Es por esto que cuando el agente recibe una observación, el mismo actualiza su tabla de Q con estado acción con la recompensa obtenida. Luego elegirá la acción que le otorgue el mayor Q. Entonces si la *view_distance* es mayor tiene mayor acceso al tablero de juego lo que le permite calcular una mejor recompensa basándose en los elementos visibles.
Cuanto más distintas las recompensas para los estados mejor. Por ejemplo: si tenemos dos estados se aprende más rápido si las recompensas son distintas, ya que prioriza los distintos elementos del juego.

3. Optimizaciones

Elegimos hacer la optimización C, *“Mejorar la función de reward, consiguiendo mejores resultados.”*.
Para mejorarla decidimos utilizar los siguientes elementos del juego a partir de la observación según su *view_distance*:

- La cantidad de comida que se encuentra a una distancia = 1 del pacman (con distancia = 1 nos referimos a que en el próximo movimiento la puede comer).
- La cantidad de capsulas que se encuentran a una distancia = 1 del pacman.
- La cantidad de fantasmas asustados que se encuentran a una distancia = 1 del pacman.

Luego sumamos todos los elementos descritos anteriormente y se lo sumamos al score del juego.
Con esto lo que queremos lograr es que el pacman priorice aquellos estados en los cuales tiene a su alrededor elementos que puede ingerir (comida, capsulas y fantasmas asustados).
Si no hay elementos que pueda comer a su alrededor, entonces a lo que le sumamos al score le asignamos un valor negativo. De esta manera priorizamos aquellos estados en los que el pacman tenga al menos un elemento para comer.