2019

Práctica 2

APRENDIZAJE AUTOMÁTICO

CLAUDIA CASTELO SAGNOTTI –100363633
VÍCTOR GÓMEZ DE LA CÁMARA - 100363723

Contenido

Introducción

En este documento se recogen todas las respuestas a las preguntas que nos hacen en las fases de la práctica además de explicar y describir todas las decisiones tomadas a lo largo de toda la practica 2 de aprendizaje por refuerzo.

Fase 1

Teniendo en cuenta que es aconsejable utilizar un menor número de atributos mas relevantes para que Pacman cumpla su función y obtenga la mayor puntuación posible hemos decidido utilizar los siguientes:

- La distancia entre Pacman y los distintos fantasmas del layout.
  Hemos elegido esto como parte del estado debido a que pretendemos que pacman se guie por la distancia para ir a por los distintos fantasmas del mapa
- La distancia entre Pacman y la comida más cercana del Layout (si hay)
  Ya que la comida puede dar puntos y aunque no sea relevante para ganar el juego, siempre es bueno tener más puntuación. (Hemos elegido la más cercana solamente por simplicidad ya que el método para coger la distancia de la comida más cercana nos lo dan hecho)
- Los movimientos legales que puede realizar el Pacman (es decir, los movimientos que no están impedidos por las paredes.
  Los movimientos que puede hacer el Pacman son importantes ya que si no se puede mover en alguna dirección debido a un muro al aprender hay que tener en cuenta las acciones que puede o no puede realizar.

Hemos decidido discretizar los valores nominales necesarios para el desarrollo de esta práctica con las distancias entre Pacman, los fantasmas y la comida, ordenándolos dentro de unos rangos creados en función del tamaño de los mapas en los que se va a probar el agente.

El rango de valores que hemos elegido para discretizar es el siguiente:

- Para los valores menores o iguales que cinco, hemos puesto como denominación que la distancia es “Muy_cerca”
- Para los valores comprendidos entre seis y diez decimos que dicha distancia es “Cerca”
- Para los valores mayores de diez, pero menores o iguales que quince hemos llamado a la distancia de la siguiente manera: “Lejos”
- Y por último, para los valores mayores que quince le damos a la distancia en nombre de “Muy_lejos”

Teniendo este rango de valores tenemos como estado:

- Las acciones que puede hacer el Pacman que dependen de si tiene muros alrededor o no: [South,North,East,West,Stop]
- Las distintas distancias entre el Pacman y los fantasmas: [Muy_cerca,Cerca,Lejos,Muy_lejos]
- La distancia entre Pacman y la comida más cercana: [Muy_cerca,Cerca,Lejos,Muy_lejos]

Creemos que este estado tiene la suficiente información y es lo bastante pequeño como para ser lo suficientemente eficiente como para que Pacman aprenda rápida y eficazmente.

Como función de refuerzo hemos elegido la puntuación 'score' que normalmente ejerce un refuerzo negativo con cualquier movimiento exceptuando cuando come comida o se come algún fantasma que es un refuerzo positivo. De esta manera logramos que Pacman se centre sobre todo en maximizar la puntuación comiendo fantasmas ya que estos tienen un refuerzo mucho mayor que la comida.

Fase 2

Para esta fase se nos pide crear el agente y cambiar el código a medida que vamos probándolo para que obtengamos mejores resultados.

Para ello hemos cambiado en concreto dos de los cuatro métodos que se nos señala en el código (El profesor nos ha indicado que esos dos métodos son suficientes).

- def computeQLearningState(self, gameState):
  En este método se debe calcular el q-estado a partir del gameState del Pacman y retornar un String que contendrá el estado del Pacman para la tabla-q.
  Para ello primero creamos un String llamado key_state que será el String en el que guardaremos todos los atributos del estado.
  Para ayudarnos también creamos dos arrays, en el primero introduciremos el estado de los fantasmas del layout (si están vivos o muertos) y en el segundo se introduciran las distancias de todos los fantasmas a Pacman.
  Después lo primero que hacemos es meter con un bucle en el String key_state todos los movimientos legales del Pacman usando la función getLegalPacmanActions.
  Ahora para las distancias hacemos un bucle en el que comprobamos la distancia y según el rango que hemos explicado en el punto anterior metemos en el String el atributo correspondiente, siempre y cuando nos hayamos asegurado de que los fantasmas sigan vivos antes.
  Después hacemos lo mismo con la comida mas cercana usando la función getDistanceNearestFood.
  Una vez tenemos todo dentro del String lo devolvemos con return.
- def update(self, state, action, nextState, reward):
  En esta función se debe actualizar la tabla-q según los estados del Pacman.
  Primero creamos las variables en las que guardamos los estados correspondientes al estado anterior y el estado actual.
  Después creamos una variable en la que metemos el valor-q de el estado y la acción anterior y después hacemos otra en la que metemos el valor del estado actual.
  Ahora una vez tenemos los valores de la tabla-q codificamos la siguiente ecuación:
  self.alpha*(reward + value_actual*self.discount – value_anterior)
  Una vez tenemos el resultado de dicha ecuación usamos la función setQvalue para actualizar el valor de el estado y la acción correspondientes en la tabla-q.

Debido a que en un principio nuestro Pacman era algo malo aprendiendo decidimos que en lugar de poner las distancias de todos los fantasmas pondríamos solo la distancia al mas cercano. De esta manera el Pacman se fijaría más en los fantasmas que estuvieran cerca de el en lugar de en todos y dejaría de intentar ir a por varios a la ve, problema que sufría anteriormente.

Para ello lo que hicimos fue modificar el código de la función def computeQLearningState(self, gameState) de manera que con unos condicionales en el bucle que se usa para introducir las

distancias de los fantasmas nos aseguramos de que se pone prioridad en aquellos que estaban Muy cerca o cerca y después vienen los que tienen lejos y muy lejos.

Como, aunque el rendimiento del pacman había subido y mejorado, algunas veces los resultados seguían siendo bastante malos decidimos ver si podíamos mejorar aun mas la eficiencia de nuestro agente. Por ello tomamos la decisión de incluir en los estados un atributo más: la dirección en la que se encuentra el fantasma más cercano.

Elegimos la dirección en la que se encuentra el fantasma mas cercano porque muchas veces en las pruebas el pacman parecía que no tenía una dirección exacta a la que ir y por ello decidimos dársela de manera que fuera dirigiéndose al fantasma mas cercano.

Para colocar en la tabla-q el nuevo estado cambiamos ligeramente el método def computeQLearningState(self, gameState) para que en el bucle de las distancias a los fantasmas, una vez tenemos el fantasma mas cercano, conseguimos tanto la posición de pacman y la posición del fantasma para que comparando ambas con condicionales se metiera en el string key_state un nuevo valor nominal.

Dicho valor nominal es el siguiente según las posiciones:

- Si pacman está a la derecha del fantasma se llama: “Oeste”
- Si pacman esta a la izquierda del fantasma se denomina: “Este”
- Si pacman está a la arriba del fantasma se llama: “Sur”
- Si pacman está debajo del fantasma se denomina: “Norte”

Así de esta manera el rendimiento de nuestro agente mejoro bastante respecto a los agentes anteriores ya que ha logrado ganar bastantes partidas y mantener una media de puntuación relativamente mas alta que la de los anteriores.

Fase 3

Podemos afirmar sin mucha duda que el mejor Pacman que hemos desarrollado ha sido el tercero ya que ha obtenido una puntuación media superior, además ha superado muchas más pruebas que el anterior.

Dicho agente recibe el nombre de Fantasma_Agents ya que a pesar de que el resto también se llamaron Fantasma decidimos diferenciarlos de manera que el que hemos elegido sea Fantasma_Agents y el primero que creamos sea Fantasma1_Agents y el segundo Fantasma2_Agents.

Los resultados en general han sido mejores en los primeros mapas siendo estos labAA1 y labAA2, pero tampoco han sido muy bajos en el resto de los mapas.

Los mejores resultados se han obtenido concretamente en al hacer que los fantasmas se movieran aleatoriamente en todos los mapas en general.

Creemos que es debido principalmente a que si los fantasmas se mueven es más fácil para Pacman conseguir un objetivo que este cerca de él. Aunque también si están estáticos cuando aprende a comerlos todos lo consigue casi el 100% de las veces sobre todo en los mapas anteriormente mencionados.

Conclusiones

Esta practica ha sido mucho más corta, fácil y divertida que la anterior, también la hemos considerado mucho más provechosa e interesante.

Además, también nos hemos dado cuenta de que aunque en un principio dijimos que no íbamos a usar muchos atributos en los estados hemos tenido que meter uno mas de los que teníamos pensado para mejorar el rendimiento, retractándonos de nuestra decisión y aprendiendo de ella.

Y aunque en general estamos satisfechos con los resultados tenemos que admitir que podríamos haber mejorado aún más el Pacman de lo que hemos conseguido y que debido a una pequeña negligencia de nuestra parte no hemos podido aprovechar todo el potencial de esta práctica.