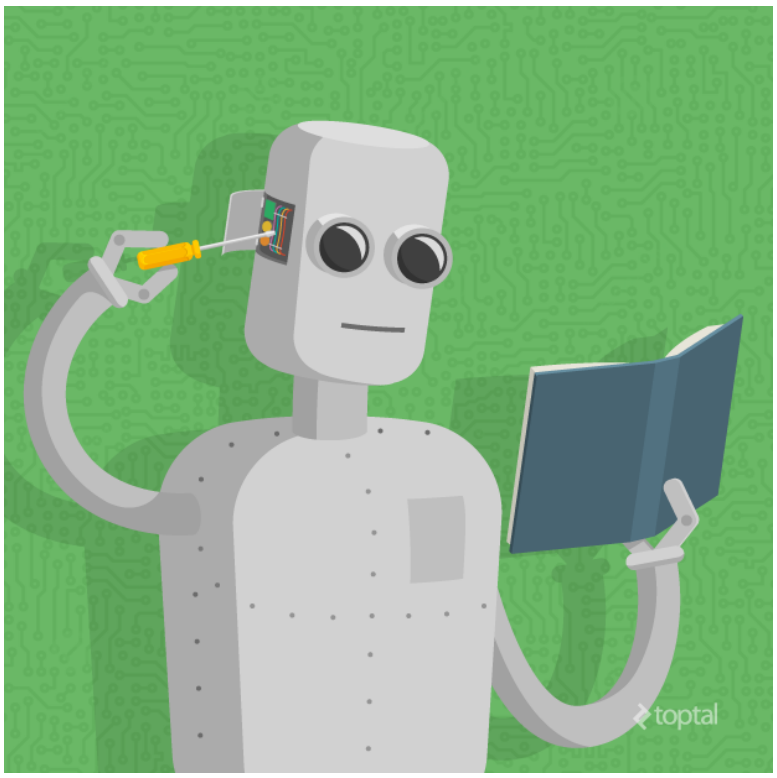
uc3m | Universidad Carlos III de Madrid

Tutorial 1



Aprendizaje Automático
23/02/2022

Marco Merola 100413665
Miguel Santana 100432505

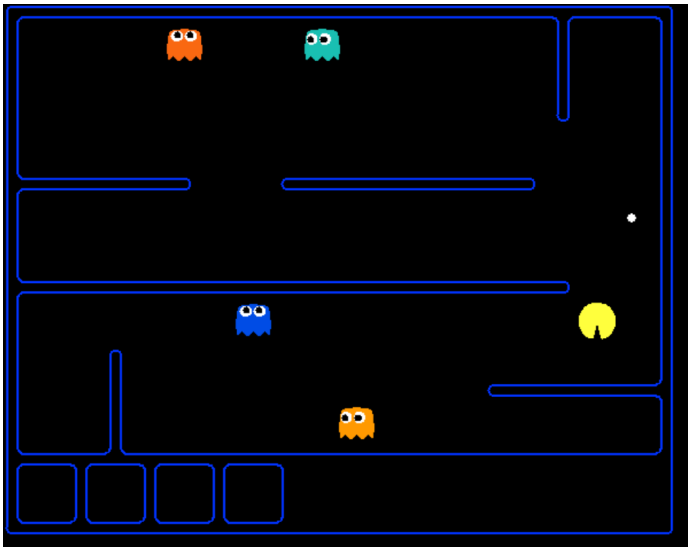
1 - En la interfaz se muestran varias opciones. En la parte de abajo se muestra la puntuación y las distancias de los distintos fantasmas(distancia Manhattan) , también muestra al pacman y a los fantasmas. En la terminal no se muestra la información detallada.

2- Los datos que creemos que son útiles para decidir lo que tiene que hacer el pacman son los siguientes: la posición del pacman, las acciones que puede realizar, los fantasmas que quedan vivos, la posición de estos, y la distancia hasta ellos.

3- Los laberintos están definidos por:

- “%”: Son las paredes del programa.
- “G”: Son las posiciones de los fantasmas.
- “P”: es la posición del pacman,
- “.”: Son las posiciones de la comida del pacman,

Hemos creado nuestro propio laberinto para realizar distintas pruebas, este laberinto se llama me.lay y es el siguiente:



4- En la terminal se muestra la información detallada pero solo cuando ejecutas el programa con una configuración del agente que no sea por defecto, en esta información de la terminal se encuentra la siguiente por cada tick(cada turno), el tamaño de mapa(width, height), la posición del pacman (horizontal, vertical), las acciones legales del pacman (hacia donde se puede mover), la dirección hacia donde está el pacman, el número de fantasmas, los fantasmas vivos, las posiciones de los fantasmas, las direcciones hacia que se mueven los fantasmas, la distancia de los fantasmas con respecto al pacman, el número de comida (pac dots) que se puede comer el pacman, la distancia hacia la comida más cerca y por último nos muestran el mapa donde la T es donde hay una pared y la F es donde no hay pared. Los datos que creemos que son útiles para decidir lo que tiene que hacer el pacman son los siguientes: la posición del pacman, las acciones que puede realizar, los fantasmas que quedan vivos, la posición de estos, la distancia hasta ellos y el tamaño del mapa.

5- La función printLineData() escribe en un fichero la siguiente información: la posición del pacman, los movimientos que puede realizar, los fantasmas vivos, la posición de los fantasmas, la distancia de los fantasmas con respecto al pacman. En primer lugar, se crea

un fichero si no existe, se abre y con la opción a+ que permite leer desde el principio y escribir al final del archivo. Por último se cierra el fichero

6- En nuestra implementación del comportamiento del pacman modificamos la función chooseAction, donde quitamos la opción de random y agregamos una función(chooseMove) que nos devuelve la dirección a la que tiene que ir. Esta función primero busca al fantasma que está más cerca del pacman, después utiliza el algoritmo a* utilizando sus coordenadas con las del fantasma y el mapa con las paredes, esto nos devuelve el camino que sigue el pacman, con lo que ya tenemos la dirección a la que se tiene que mover el pacman en cada turno hasta que llegue a la dirección donde está o estaba el fantasmas. Al hacer varias ejecuciones del programa con nuestra implementación del BasicAgentAA en el mapa por defecto, obtenemos la siguiente tabla, que indica que el número medio de rondas en el caso de que los fantasmas sean estáticos es de 25 y en caso de ser dinámicos se obtiene una media de 41,4.

Estáticos	25	25	25	25	25	25	25	25	25	25
Movimiento	50	35	37	50	40	32	37	49	45	39

7-Una de las principales ventajas del aprendizaje automático es que en términos generales viene muy bien al tipo de problema en el que los datos cambian, los escenarios varían o es posible que en algún caso sea realmente inviable una codificación de un problema. Estas características son justamente las que presenta el Pac-man, pues pudiera existir una infinidad de mapas con diseños distintos, fantasmas que se mueven dinámicamente y obstáculos muy diversos entre escenarios, por lo que ciertamente programar el comportamiento “adecuado” del Pac-man sin el uso de aprendizaje automático no siempre resultará en una solución óptima, e incluso según la forma en la que se programe podrían existir situaciones imposibles de resolver para el Pac-man o exageradamente costosas computacionalmente. Además, con el uso de aprendizaje automático aumentaría la capacidad del Pac-man para tomar decisiones que faciliten las decisiones. En conclusión, usar aprendizaje automático podría garantizar que la efectividad del Pac-man comiendo fantasmas sea la máxima y que además el coste computacional no sea tan elevado.

Conclusiones

Al analizar el objetivo, el equipo planteó posibles soluciones para poder codificarlas luego. Una de estas soluciones fue la de crear atributos que indiquen en todo momento si el Pac-man está rodeado de paredes que le impidan comer fantasmas. Según esta solución se plantearon todas las condiciones posibles para que el Pac-man se mueva según las posibles alternativas, e incluso si el movimiento más indicado no era legal, se decide por elegir aleatoriamente entre las posibles acciones legales. Sin embargo, se encontró que en algunos casos el Pac-man se congela si el fantasma más próximo se mueve hacia una posición y vuelve a la anterior.

Para controlar los problemas ya mencionados, se decide crear un vector con las posiciones más recientes para intentar visitarlas de nuevo. Este vector elimina la posición visitada menos recientemente luego de una cierta cantidad de ejecuciones. Se debe mencionar que este vector se eliminaba cada ronda, por lo que debió ser creado en el constructor de GameState y gestionarlo mediante un nuevo método dentro del fichero bustersAgents.py que controla la entrada y salida de elementos dentro del vector, que realmente funciona como una especie de cola, sin embargo esta implementación da problemas dependiendo del tipo de obstáculo que encuentra el Pac-man. Para resolver este problema, se ha decidido implementar A*, obteniendo un resultado satisfactorio.