Departamento de Electrónica y Computación
Universidad de Santiago de Compostela

Grado en Robótica

Aprendizaje Automático

Curso 2021-2022

Tutorial 1: Plataforma Pac-Man

Extracción de características y programación manual de un agente

26 de enero de 2022

- El objetivo de este tutorial es familiarizarse con la plataforma de juego Pac-Man que se utilizará en las prácticas de la asignatura y con el lenguaje de programación Python.
- Se puede realizar en Linux, Windows o Mac.
- Es importante ir realizando los ejercicios en orden.

1. Introducción

Las prácticas de esta asignatura van a apoyarse en una plataforma que recrea el videojuego clásico Pac-Man[1] (Figura 1). Se utilizará una versión muy simplificada del juego para poder desarrollar sistemas de control automáticos y explorar algunas capacidades del aprendizaje automático.

Figura 1: Captura de la interfaz básica de Pac-Man.

En esta versión del juego, en la pantalla aparecen tanto fantasmas como puntos de comida (*pac dot*). Pac-Man y los fantasmas se mueven una casilla por cada ciclo de ejecución, al que llamaremos turno o *tick*. En cada *tick*

[1] https://en.wikipedia.org/wiki/Pac-Man

en el que Pac-Man no se come a un fantasma o un punto de comida se resta -1 a la puntuación. Se premia con 200 puntos por cada fantasma comido y 100 puntos por cada punto de comida. El objetivo es comerse a todos los fantasmas que aparecen en la pantalla lo antes posible, maximizando de esta manera la puntuación. En la versión que utilizaremos los fantasmas se podrán comer siempre.

Durante el transcurso del juego se ofrece diferente información sobre el estado del mismo. Esta información será la que utilicemos para desarrollar nuestros comportamientos automáticos.

2. Ejecución del código

1. Descargar el código fuente del Pac-Man de Campus Virtual (fichero **pacman.zip**).
2. Descomprimir la carpeta *pacman*.
3. Abrir un terminal o consola de comandos y entrar dentro de la carpeta *pacman*.
4. Ejecutar el juego mediante el siguiente comando de consola:

   ```
   python busters.py
   ```

   Se mostrará una ventana con una interfaz básica del juego controlable por teclado.

   Esta interfaz inicializa una configuración por omisión: el Pac-Man se controla con el teclado, muestra los fantasmas completamente parados y el laberinto por omisión es *oneHunt*. Para poder ver todas las opciones de inicio disponibles hay que introducir el siguiente comando:

   ```
   python busters.py --help
   ```

   Los principales argumentos que se pueden cambiar son:

   - **-n GAMES** Número de juegos. Por omisión es 1.
   - **-l LAYOUT_FILE** El tablero del juego. Por omisión es *oneHunt*.
   - **-p TYPE** El tipo de agente Pac-Man. Por omisión es *BustersKeyboardAgent* (control por teclado).
   - **-g TYPE** El tipo de agente de fantasma. Para que los fantasmas se muevan de forma aleatoria: `-g RandomGhost`.
   - **-k NUMGHOSTS** El número máximo de fantasmas. El valor por omisión es 4. Todos los mapas predefinidos permiten de 1 hasta 4 fantasmas.
   - **-t** Tiempo de *delay* entre *frames*.

3. Ejercicios

Es necesario contestar claramente a cada una de las preguntas que se formulan en estos ejercicios. Para realizarlos se recomienda que primero se explore el comportamiento de Pac-Man probando diferentes argumentos y mirando los ficheros y el código del juego (consultar descripción de los ficheros importantes en el Anexo).

1. ¿Qué información se muestra en la interfaz? ¿Y en la terminal? ¿Cuál es la posición que ocupa Pac-Man inicialmente?
2. Según tu opinión, ¿qué datos podrían ser útiles para decidir lo que tiene que hacer Pac-Man en cada momento?
3. Revisa la carpeta *layouts*. ¿Cómo están definidos los mapas en estos ficheros? Diseña un mapa nuevo, guárdalo y ejecútalo en el juego.
4. Ejecuta el agente `BasicAgentAA` tal y como se indica a continuación:

   ```
   python busters.py -p BasicAgentAA
   ```

   Describe qué información se muestra por pantalla sobre el estado del juego en cada turno. De esta información,¿cuál crees que podría ser más útil para decidir automáticamente la siguiente acción de Pac-Man?

5. Programa una función de nombre `printLineData()` dentro del agente `BasicAgentAA` del fichero `busterAgents.py`. Esta función debe devolver una cadena de caracteres con la información que se considere relevante del estado del Pac-Man. Esta cadena de caracteres tendrá el siguiente formato:

   ```
   dato1,dato2,...,datoN
   ```

   donde `dato1`, `dato2`, ..., `datoN` son datos del estado actual del juego, por ejemplo, la distancia a los fantasmas, si están vivos o no, etc. Cada uno de los datos de la cadena anterior pueden ser valores numéricos (e.g., 0,1,2.3, etc) o valores nominales (e.g., *true*, *false*, *lejos*, *cerca*, *media_distancia*, etc), pero no estructuras complejas (por ejemplo, un dato no puede ser una matriz entera).

   En cada turno, dicha función será llamada desde el bucle principal del juego en `game.py` para que se escriba la cadena de caracteres que devuelve en un fichero. Cada llamada generará una línea diferente del fichero. Este paso es de gran importancia ya que servirá como primera versión de la fase de extracción de características y será imprescindible para las siguientes prácticas.

   Además, cada vez que se inicie una nueva partida o se abra el juego de nuevo, las nuevas líneas deben guardarse debajo de las antiguas. Es decir, que no se debe reiniciar el fichero de texto al empezar una nueva partida. Por tanto, el fichero de texto resultante tendrá que tener tantas líneas como turnos se hayan jugado en todas las partidas.

6. Programa un comportamiento *inteligente* para Pac-Man. Para ello, en la clase `BasicAgentAA`, se pide modificar el método `chooseAction()` que hasta ahora presentaba un comportamiento aleatorio. Pac-Man debe perseguir y comerse a todos los fantasmas de la pantalla. Compara los resultados ejecutando el juego con los fantasmas estáticos y en movimiento aleatorio (`-g RandomGhost`). Haz varias ejecuciones y determina cuántos turnos de media suele tardar en finalizar.

7. El agente programado en el ejercicio anterior no utiliza ninguna técnica de aprendizaje automático. ¿Qué ventajas crees que puede tener el aprendizaje automático para controlar a Pac-Man?

4. Documentación a entregar

El tutorial se debe realizar **obligatoriamente** en grupos de 2 personas y se entregará a través del entregador que se publicará en Campus Virtual **hasta las 13:00 horas del martes 25 de febrero de 2022**. El nombre del archivo comprimido debe contener los primeros apellidos de los dos alumnos, `ej. tutorial1-garcia-ranon.zip`.

El archivo comprimido debe incluir lo siguiente:

1. Un documento en formato **PDF** que debe contener:
   - Portada con los nombres de los alumnos.
   - Las respuestas a todas las preguntas y subpreguntas planteadas en los ejercicios.
   - Descripción de la función implementada para imprimir en fichero la información del estado del juego en cada turno.
   - Descripción de la implementación del comportamiento del agente `BasicAgentAA`.
2. Incluir en una carpeta llamada "material":
   - El nuevo mapa creado.
   - El fichero de texto que genera la función de extracción de caracteristicas que se ha programado.
   - El archivo/archivos de código fuente modificados por los alumnos con la extracción de características y el agente programado.

El peso de este tutorial sobre la nota final de la asignatura es de 0.25 puntos.

ANEXO I: Ficheros de código de la plataforma Pac-Man

- **bustersAgents.py**: Agentes con diferentes comportamientos de Pac-Man. Cada agente desarrollado se incluye en este fichero como una nueva clase. Los que hay disponibles son:

  1. **RandomPAgent**
  2. **BustersKeyboardAgent**
  3. ...

- **inference.py**: Código para calcular la distancia de los fantasmas.
- **busters.py**: La clase principal del juego.
- **bustersGhostAgents.py**: Clase donde se define el comportamiento de los agentes fantasma.
- **distanceCalculator.py**: Clase que computa las distancias.
- **game.py**: clase donde se ejecuta el bucle principal del juego (`def run:loop`). En este fichero se define el tablero en la clase **Grid**, las acciones de Pac-Man en la clase **Actions** y la clase **GameStateData**, donde almacena información del estado del juego.
- **ghostAgents.py**: Los agentes que controlan los fantasmas:

  1. **RandomGhost**
  2. **DirectionalGhost**

- **graphicsDisplay.py**: Interfaz gráfica del juego.
- **graphicsUtils.py**: Clase auxiliar dedicada a utilidades de la interfaz gráfica.
- **keyboardAgents.py**: Control de Pac-Man por teclado.
- **layout.py**: Código para leer los ficheros de tablero y almacenar su contenido.
- **util.py**: Clase auxiliar con funciones de apoyo para el juego.

ANEXO II: Consideraciones sobre la ejecución del código

El software de este tutorial está probado tanto en Linux como en Windows, con las versiones de python 2.7.2, 2.7.12 y 2.7.13. Si se dispone de Ubuntu o alguna otra distribución de Linux, y no se dispone de alguna versión de python 2.7.X, se puede probar a instalar cualquiera de esas versiones utilizando un gestor de versiones de python como pyenv (`https://amaral.northwestern.edu/resources/guides/pyenv-tutorial`). Entonces, para la ejecución del código:

1. Tratad de ejecutar con la versión de python que se tenga instalada ya sea en Linux o Windows.
2. Si aparece algún error, instalad una versión de python 2.7.X, por ejemplo 2.7.12.
3. Instalad el paquete numpy para esa versión de python que es necesario para poder ejecutar el código.
4. Ejecutad con esa versión de python.