Departamento de Informática
Universidad Carlos III de Madrid

Grado en Ingeniería Informática

Aprendizaje Automático

Curso 2018-2019

Tutorial 1: Plataforma Pac-Man

Extracción de características y programación manual de un agente.

4 de febrero de 2019

- 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]. Se utilizará una versión muy simplificada del juego para poder desarrollar sistemas de control automáticos y explorar algunas capacidades del aprendizaje automático.

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 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, maximizando 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. Lenguaje de programación Python

El Pac-Man que se va a utilizar está programado en Python, por lo que este tutorial servirá también para familiarizarse con este lenguaje de programación.

Python es un lenguaje **interpretado**, lo que implica que no será necesario compilar nada para ejecutar los programas. Éstos se ejecutan igual que un script o un archivo de procesamiento por lotes (.sh, .bat, etc.).

Para ejecutarlos es necesario tener instalado Python 2.7 en el ordenador (a menudo viene ya instalado por omisión). Existen dos versiones de Python en desarrollo: la 2.7 y la 3.5. Los códigos fuente que ejecutan cambian ligeramente, por lo que no son completamente compatibles. La versión 2.7 está mucho más extendida. Se puede descargar de aquí:

`https://www.python.org/download/releases/2.7/`

La diferencia más llamativa con respecto a otros lenguajes como Java o C es que no se usan llaves para determinar bloques de código, sino que se utilizan las propias sangrías de las líneas. Por ejemplo, si se tiene un if, las líneas que le pertenecerían serían las que están por debajo y con más sangría (más a la derecha).

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

Por eso Python no permite mezclar tabulaciones y espacios para establecer sangrías. Lo más recomendable es usar siempre espacios (4 espacios por cada sangría, por ejemplo).

En internet hay muchísimos ejemplos que se pueden consultar para resolver problemas concretos. El siguiente enlace oficial de Python es una buena referencia de consulta cuando surjan dudas y ver ejemplos de código:

```
https://docs.python.org/2/tutorial/index.html
```

3. Ejecución del código

1. Descargar el código fuente del Pac-Man de Aula Global (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
   ```

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

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*.

4. Ejercicios

Se pide contestar razonadamente 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 laberintos en estos ficheros? Diseña un laberinto nuevo, guárdalo y ejecútalo en el juego. Incluye una captura de pantalla del mismo en la memoria.

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. Dicha función será llamada desde el bucle principal del juego en *Game.py* para que se escriba en un 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.

   Por cada turno de juego, se debe guardar una línea con todos los datos concatenados del estado del juego que se calculan por omisión, separados por el carácter coma (,).

   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. No se debe reiniciar el fichero de texto al empezar una nueva partida, por lo que el fichero de texto resultante tendrá tantas líneas como turnos se hayan jugado en todas las partidas.

6. Programa un comportamiento "inteligente" para el 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?

5. Documentación a entregar

El tutorial se debe realizar **obligatoriamente** en grupos de 2 personas y la entrega del mismo se realizará a través del entregador que se publicará en Aula Global **hasta las 23:55 horas del viernes 22 de Febrero de 2019**.

El nombre del archivo comprimido debe contener los últimos 6 dígitos del NIA de los dos alumnos, `ej. tutorial1-123456-234567.zip` El archivo comprimido debe incluir lo siguiente:

1. Una memoria en formato **PDF** que debe contener:

   - Portada con los nombres y NIA de los dos alumnos.
   - Las respuestas a todas las preguntas planteadas en los ejercicios.
   - Descripción de la función implementada que imprime en fichero la información del estado del juego en cada turno.
   - Descripción de la implementación del comportamiento del agente `BasicAgentAA`.
   - Conclusiones y dificultades encontradas

2. Incluir en una carpeta llamada "material":

   - El nuevo mapa creado.
   - El fichero generado por la función de extracción de características que se ha programado.

- El archivo/archivos de código fuente modificados por los alumnos con la extracción de características y el agente programado.

Se pide prestar especial **atención a las normas de entrega**.

ANEXO: Ficheros de código de la plataforma Pac-Man

- **bustersAgents.py**: agentes con diferentes comportamientos de Pac-Man. Cada agente desarrollado se incluye como una nueva clase. Los que hay disponibles son **RandomPAgent**, **BustersKeyboardAgent** y la plantilla modificable de un agente sin programar llamado **GreedyBustersAgent**.
- **inference.py**: código para calcular la distancia de los fantasmas (se puede modificar).
- **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**: agentes que controlan los fantasmas (**RandomGhost** y **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.