Departamento de Informática
Universidad Carlos III de Madrid

Grado en Ingeniería Informática

Aprendizaje Automático

Curso 2020-2021

Práctica 2

Aprendizaje por refuerzo

21 de abril de 2021

1. Introducción

En esta práctica se aplicará el algoritmo *Q-learning* en el dominio del juego Pac-Man para construir un agente que funcione de forma automática en la mayor variedad de laberintos posibles. El objetivo del agente será maximizar la puntuación obtenida. Para ello se utilizarán técnicas de aprendizaje por refuerzo.

2. Tareas a realizar en la práctica

Las tareas a realizar en esta práctica se dividen en distintas fases (selección de la información del estado y función de refuerzo, construcción del agente, y evaluación) que se detallan a continuación:

Fase 1. Selección de la información del estado y función de refuerzo

Para el aprendizaje del agente se van a emplear tuplas de experiencia del tipo (*estado_tick*, *acción*, *estado_tick_siguiente*, *refuerzo*), donde *estado_tick* contiene información del estado del juego, *acción* es la acción que ejecuta el agente en ese estado, *estado_tick_siguiente* es el estado al que se llega tras ejecutar la acción *acción* en el estado *estado_tick*, y *refuerzo* es el refuerzo que se recibe al realizar la transición. Para ello, se deben tener en cuenta los siguientes aspectos:

1. Seleccionar los atributos que se van a emplear para representar un estado del juego. Esta información debe ser genérica y no ligada a laberintos en particular, puesto que se pretende que Pac-man juegue bien en la mayor variedad de laberintos posibles. Además, cuantos menos atributos se seleccionen mejor, y si son nominales con pocos valores aún mejor, ya que el tamaño de la tabla Q dependerá de esta selección.

2. Al contrario que en otras prácticas, en estas tuplas *estado_tick* y *estado_tick_siguiente* tienen los mismos atributos, sólo que referidos a diferentes instantes de juego.

3. Diseñar una función de refuerzo que permita a Pac-man lograr su objetivo de maximizar la puntuación obtenida en una partida.

Fase 2. Construcción del agente

1. Construir una nueva clase en *busterAgents.py* llamada *QLearningAgent*.

2. Implementar la funcionalidad del agente del tutorial 4 en esta nueva clase de forma que se construya y actualize una tabla Q de acuerdo al algoritmo Q-Learning. Esta tabla tendrá tantas filas como número de estados se haya decidido emplear, y tantas columnas como acciones pueda ejecutar Pac-man. Por ejemplo, si se seleccionan dos

atributos para representar el estado: la posición relativa del fantasma más cercano con respecto a Pac-man, y su distancia, tenemos un atributo nominal con 8 posibles valores (*arriba,izquierda,derecha,...,arriba_izquierda*) y tenemos un atributo numérico cuyo rango dependerá del tamaño del mayor laberinto donde juguemos. Supongamos que este rango va de 1 a 10, donde 10 es la distancia máxima a la que puede estar un fantasma. Entonces, el número de filas de nuestra tabla Q será $8 \times 10 = 80$ filas, que son todas las posibles combinaciones de valores de esos dos atributos. Por lo tanto, se puede ver que cuanto mayor es el número de atributos y el rango de éstos, mayor es el tamaño de la tabla Q. Por eso, se recomienda tener pocos atributos y tener un tamaño manejable de la tabla Q que favorezca la depuración del código y la velocidad de aprendizaje.

3. Seleccionar los valores que se consideren correctos para α, ϵ y γ.
4. El funcionamiento del agente debe ser el siguiente:
   - Al arrancar el agente leer la tabla Q existente mediante un método llamado *readQTable* y cargarla en memoria.
   - En cada tick de juego, hay que construir una tupla de experiencia del tipo (*estado_tick*, *acción*, *estado_tick_siguiente*, *refuerzo*) con la información seleccionada en la fase anterior. Esta tupla de experiencia se pasa como parámetro a un método llamado *update* para realizar una actualización de la tabla Q.
   - En cada tick de juego, a la hora de seleccionar una acción se va a emplear la estrategia ϵ-*greedy* como en el tutorial 4, lo que significa que se va a ejecutar una acción aleatoria con probabilidad ϵ y la acción $\arg\max Q(s, a)$ con probabilidad $1 - \epsilon$.
   - Al finalizar la partida, escribir la tabla Q mediante el método *writeQTable*. De esta forma, al arrancar nuevamente el agente, se va a partir de lo aprendido anteriormente.

NOTAS:

1. Para la construcción del agente se recomienda seguir un proceso de desarrollo incremental, en el que primero se aborden laberintos sencillos, para después poco a poco ir incrementando la dificultad. Para ello, en esta práctica se disponen de 5 laberintos: *labAA1.lay, labAA2.lay, labAA3.lay, labAA4.lay, labAA5.lay*. Se recomienda centrarse en primer lugar en *labAA1.lay*, y construir un agente que funcione perfectamente en este laberinto. Una vez obtenido, construir un agente que funcione bien en *labAA1.lay, labAA2.lay*. Y después construir un agente que funcione bien en los tres primeros laberintos. Y así sucesivamente, hasta obtener un agente que funcione bien en todos los laberintos.

   Como estos laberintos tienen un número de fantasmas diferente al número por defecto, es necesario que se modifique el parámetro k de los argumentos. Por ejemplo, la ejecución del primer laberinto quedaría así:

   ```
   python busters.py -k 1 -l labAA1 -p QLearningAgent
   ```

2. Durante el proceso de aprendizaje, puede ser necesario evaluar lo aprendido por el agente hasta el momento eliminando la exploración aleatoria de acciones. Para ello, entre partida y partida, se pueden establecer los valores de ϵ y α a 0 y lanzar una partida con estos parámetros. Si el agente aún no funciona bien, se pueden restablecer los valores ϵ y α a sus valores anteriores para continuar con el aprendizaje.
3. También durante el aprendizaje, se puede comprobar si las actualizaciones en la tabla Q se están haciendo correctamente. Para ello, se puede abrir el fichero que almacena la tabla y comprobar que realmente una tupla de experiencia lleva a actualizar la celda que le corresponde.

Fase 3. Evaluación del agente

1. Evaluar el/los agente/s resultante/s en distintos mapas y con configuraciones distintas de fantasmas, comparando los resultados obtenidos con los de otros agentes.
2. Reportar y analizar los resultados obtenidos.

3. Directrices para la documentación

Uno de los miembros del grupo deberá entregar una memoria en formato **PDF**, **que no debe superar las 20 páginas de extensión incluyendo portada e índice**, que deberá contener:

- Breve descripción explicando los contenidos del documento.

- Justificación del conjunto de atributos final elegido y su rango utilizado para la definición de los estados.
- Descripción de la función de refuerzo final empleada.
- Descripción de cualquier tratamiento sobre los datos que se lleve a cabo y de todos los pasos realizados.
- Descripción del código generado para llevar a cabo el aprendizaje de la política de comportamiento.
- Descripción del agente final implementado. Se debe indicar la evolución histórica de los agentes implementados hasta llegar al agente final, donde cada uno de estos agentes puede tener un conjunto de atributos diferente y destacar las diferencias entre unos y otros.
- Descripción y análisis de los resultados producidos por el agente final implementado tras la evaluación. En este apartado es importante describir por qué se cree que ha funcionado bien el agente seleccionado (si es éste el caso). En cualquier caso, se deberán describir posibles mejoras que se pueden hacer para aumentar su rendimiento.
- Conclusiones:
  - Conclusiones sobre la tarea a realizar.
  - Apreciaciones más generales sobre las prácticas de la asignatura como: para qué pueden ser útiles los modelos obtenidos y si se os ocurren otros dominios en los que aplicar aprendizaje por refuerzo, etc.
  - Descripción de los problemas encontrados a la hora de realizar esta práctica.
  - Comentarios personales. Opinión acerca de la práctica. Dificultades encontradas, críticas, etc.

Se valorará la claridad de la memoria, la exposición de las distintas alternativas y algoritmos utilizados, la justificación de los algoritmos elegidos, la presentación y el análisis de los resultados obtenidos y las conclusiones aportadas.

4. Normas de entrega

La práctica se deberá realizar en grupos de dos personas y podrá ser entregada a través del enlace que se publicará en Aula Global hasta las 23:55 horas del día **21 de Mayo** de 2021. El nombre del fichero debe contener los 6 últimos dígitos del NIA de los alumnos (ej practica2-387633-209339.zip). El fichero deberá incluir:

- Una memoria en formato **PDF**, que deberá contener al menos los contenidos descritos en la sección 3.
- Un fichero .zip que contenta *todo* el código necesario para ejecutar vuestro agente. Si se extrae ese fichero y se ejecuta el comando `python busters.py -p QLearningAgent`, tiene que ejecutarse vuestro agente desarrollado en esta práctica.