Universidad Carlos III de Madrid

Aprendizaje Automático

Grado en Ingeniería Informática

Grupo 83

Práctica 3: Aprendizaje por refuerzo

Autores:
Daniel Medina García
alejandro rodríguez Salamanca

10 de mayo de 2016

Índice

Introducción

1. Diagrama con los diferentes pasos de la tarea de aprendizaje realizada en la práctica

Ni puta idea de esto, pero viene en el guión. Entiendo que será algo como Diseñar el espacio de estados - programar el agente - aprendizaje online - generar el agente final

2. Generación del espacio de estados

Hablar de nuestro estado (dirección en la que se encuentra el fantasma más cercano y última direccion en la que se ha movido Pacman). Justificación del conjunto de atributos final elegido y su rango para la definicion de los estados. Se debe indicar la evolucion historica 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 con el conjunto final.

Espacios usados - 1. Dirección del fantasma más cercano, si hay muros alrededor. 2. El usado (y que ha perdido) Dani 3. Dirección del fantasma mas cercano y última dirección en la que se ha movido Pacman

---

TODO: Intro de la sección

Primero, probamos un espacio de estados formado por la dirección en la que se encuentra el fantasma más cercano, y si existen muros alrededor de PacMan, esto es, si el fantasma más cercano se encuentra por encima de PacMan y a la izquierda de éste, y hay un muro justo debajo, el estado sería `North,West,False,False,True,False`. Este espacio de estados nos daba un total de 256 estados (cuatro posibles valores de los dos primeros atributos, y dos posibles valores de los cuatro siguientes). Tras ejecutar un agente con este espacio de estados, nos dimos cuenta de que la información relativa a los muros no aportaba información relevante puesto que en nuestro agente sólo elegimos una acción de entre las legales, y aquellas direcciones en las que se encuentra un muro no están incluídas en dicho conjunto.

Después, probamos a eliminar dicha información sobrante y nos qudamos con un conjunto de estados compuesto por DANI, AQUÍ TE TOCA EXPLICAR EL QUE HAS ELEGIDO TÚ.

Finalmente, el conjunto de estados elegido fue el formado por la dirección del fantasma más cercano y la dirección del último movimiento realizado por PacMan. Esto nos da un total de 64 posibles estados. Estos atributos fueron elegidos por dos razones:

- Simplifica mucho el conjunto de estados - Al tener en cuenta únicamente tres atributos, el conjunto de estados queda muy reducido, lo que era uno de los objetivos que tratábamos de cumplir, puesto que un conjunto de estados muy grande puede contener estados que no se lleguen a alcanzar nunca.

- Aporta información necesaria y genérica - Otro de los requisitos que hemos intentado cumplir ha sido obtener un agente genérico que no esté *atado* a ningún mapa, y que sea capaz de desenvolverse con soltura en diversos escenarios. Por ello, los atributos escogidos son independientes del mapa, pero aportan la información que necesita PacMan para determinar hacia qué dirección debe moverse.

3. Generación de la tabla Q

Aprendizaje online con epsilon-greedy. Rewards: 10 si se acerca al fantasma más cercano, 199 si se come un fantasma, -1 si no hace algo —-

Para generar la tabla Q hemos hecho uso de la técnica conocida como *onlie learning*. Se trata de un método mediante el cual los datos se encuentran disponibles en orden secuencial (se generan mientras PacMan está jugando partidas) y son usados para actualizar la mejor predicción hasta el momento de la tabla Q.

En cuanto a los refuerzos, hemos decidido dividirlos en tres tipos:

- PacMan se come un fantasma - Cuando PacMan se come un fantasma, recibe un refuerzo positivo con un valor de 199 (coincide con la diferencia de puntuación entre el estado actual y el próximo estado al que se transiciona)-
- PacMan se acerca al fantasma más cercano - En este caso, entendemos que PacMan está realizando un movimiento beneficioso, por el cual recibe un refuerzo positivo con valor 10.
- PacMan se aleja del fantasma más cercano - Si PacMan está *persiguiendo* a un fantasma, y su distancia a éste aumenta, el refuerzo que se da es negativo, cuyo valor es de -1.

Además, se ha usado la técnica ϵ-greedy, por la cual nuestro agente toma la acción devuelta por la política obtenida hasta el momento un ϵ % de las veces, y (1-ϵ) % una acción aleatoria, para que el mapa sea explorado y se prueben diversas acciones para un mismo estado, con el objetivo de quedarnos con la que se considere mejor.

4. Agente automático

Hablar de cómo se ha implementado el agente

— Nuestro agente ha sido implementado sobre la clase PacmanQAgent, la cual extiende de QLearningAgent, por lo que trae cierto comportamiento predefinido para que resulte más sencillo implementar un agente utilizando la técnica de Q-Learning.

Primero, se inicializan los valores de `alpha, epsilon` y `discount` en el constructor de la clase, para ser usados posteriormente en la actualización del valor en la tabla Q. Además, se carga la tabla Q, que se encuentra en un fichero llamado `qtable.txt`. Esta tabla Q es almacenada en un objeto de tipo `Counter`, que se encuentra en `utils.py`. Dicho objeto actúa como un diccionario, en el que las claves serán tuplas estado-acción, y el valor, el correspondiente al que retorne la función Q.

Nuestra política es retornada por el método `getPolicy`. Este método recibe un estado, e itera sobre todas las acciones legales, devolviendo aquella acción cuyo valor en la tabla es el máximo para el estado dado.

Si una partida supera los 800 turnos, ésta acaba, pues entendemos que nuestra tabla no se está actualizando con valores que ayuden a nuestro agente a jugar mejor.

5. Evaluación

Para nuestro agente, hemos decidido probar dos configuraciones distintas a partir de las cuales hemos generados dos q-table que nos dispondremos a evaluar.

Agente 1:	α	0.3
	γ	0.8
	ϵ	0.8
Agente 2:	α	0.6
	γ	0.4
	ϵ	0.7

Para realizar la evaluación de ambas configuraciones hemos jugado 10 partidas en el mapa por defecto y en `finalMap` para comprobar cómo se comportaba el agente en dos escenarios completamente diferentes.

Primero, evaluaremos los datos obtenidos con los parámetros previamente mostrados, y después usaremos siempre la acción que indique nuestra política, sin actualizar los valores de la tabla Q durante las distintas partidas para observar las diferencias.

*Timeout = 2000 turnos - casillas sin puntuación.
MAPA DEFAULT Agente 1: score: 607 584 657 551 626 662 610 520 668 603
Agente 2: score: 710 610 609 608 596 496 635 577 537 674
FINALMAP Agente 1: score: 468 361 4 388 330 384 490 113 432 448
Agente 2: score: 372 461 524 591 317 284 454 364 386 394
MAPA DEFAULT Agente 1: score: 563 -523 -98 625 626 699 325 -75 -125 624
Agente 2: score: 551 267 484 - - - 367 -115 64 634
FINALMAP Timeout siempre, para ambos agentes.

Como podemos observar, en aquellas partidas jugadas con un porcentaje de movimientos aleatorios y en las cuales la tabla Q se actualiza, obtenemos unos resultados mucho mejores que cuando seguimos siempre la acción que devuelve nuestra política y mantenemos la tabla Q invariable. Esto puede ser debido a:

- Nuestro agente no ha aprendido una política buena - Pese a haber jugado un elevado número de partidas con nuestro agente para conseguir una tabla Q fiable cuyos valores se correspondiesen con la acción más adecuada para cada instante, no hemos conseguido que el algoritmo obtenga los valores adecuados, o la forma en la que expresamos los estados no es la más adecuada.

- La definición de estado que usamos es muy general - Al intentar usar una definición de estado lo más genérica posible y haber entrenado en diversos mapas, existe la posibilidad de que nuestro agente no sepa desenvolverse con soltura en ninguno.

Tras valorar los resultados, hemos llegado a la conclusión de que la gran diferencia obtenida es debido al segundo caso, puesto que cuando el agente realiza ciertas acciones aleatorias que le ayudan a explorar un mapa y salir de aquellas situaciones en las que no tiene claro qué debe hacer, y es capaz de actualizar la tabla Q para `recordar` esas decisiones en futuras partidas, es capaz de obtener unas puntuaciones muy buenas en las partidas jugadas, y más teniendo en cuenta los datos con los que cuenta para tomar las decisiones.

TODO: Decidir con cuál nos quedamos

6. Agente final

(El que mejores resultados obtenga de lo anterior) Descripción y análisis de los resultados producidos por el agente final implementado tras la evaluación. En este apartado es importante describir por que se cree que ha funcionado bien el agente seleccionado (si es este el caso). En cualquier caso, se deberán describir posibles mejoras que se pueden hacer para aumentar su rendimiento. Ademas, es importante responder a la siguiente cuestion:

¿El agente final desarrollado es capaz de superar en rendimiento a los agentes utilizados en la generacion de las tuplas de experiencia? Justificar la respuesta tanto en caso afirmativo como en caso contrario.

7. Conclusiones

Conclusiones sobre la tarea a realizar. Apreciaciones mas generales sobre las practicas de la asignatura como: para que pueden ser utiles los modelos obtenidos y si se os ocurren otros dominios en los que aplicar aprendizaje automatico, etc. • Descripcion de los problemas encontrados a la hora de realizar esta practica. • Comentarios personales. Opinion acerca de la practica. Dificultades encontradas, criticas, etc