Aprendizaje por refuerzo en robótica móvil

Propuesta de trabajo para la asignatura Inteligencia Artificial

Profesor: Ignacio Pérez Hurtado de Mendoza

Grado en Ingeniería Informática - Ingeniería del Software
Curso 2023/2024

Introducción y objetivo

El aprendizaje por refuerzo [1, 2, 3, 4] es un paradigma de aprendizaje automático en el que un agente interactúa con un entorno, observa las señales de recompensa o penalización resultantes de sus acciones y ajusta su comportamiento para maximizar la recompensa acumulada a lo largo del tiempo. En esencia, el agente aprende a través de la experiencia, tomando decisiones secuenciales para alcanzar sus objetivos basándose en la retroalimentación que recibe del entorno.

Dicho paradigma está estrechamente relacionado con los procesos de decisión de Markov (MDP por sus siglas en inglés), ya que muchos algoritmos de aprendizaje por refuerzo se basan en la teoría y los principios subyacentes de los MDP para tomar decisiones secuenciales en entornos dinámicos.

Los MDP son un marco matemático que modeliza las interacciones entre un agente que toma decisiones y un entorno estocástico en el que esas decisiones tienen consecuencias. En un MDP, el agente toma decisiones en cada paso de tiempo para maximizar una medida de rendimiento (generalmente la recompensa acumulada a lo largo del tiempo) en presencia de la incertidumbre en el entorno.

Los algoritmos de aprendizaje por refuerzo, como Q-Learning, SARSA, y algoritmos basados en políticas, utilizan los MDP para modelizar el entorno y calcular estrategias óptimas para tomar decisiones secuenciales. Estos algoritmos aprenden iterativamente a partir de la experiencia, ajustando sus políticas de acción para maximizar la recompensa esperada, utilizando técnicas como la exploración y la explotación para equilibrar el aprendizaje y la toma de decisiones.

En este trabajo nos centraremos en una sencilla aplicación del aprendizaje por refuerzo a la robótica móvil, en dónde un robot con ruedas debe planificar una ruta en un entorno con obstáculos. Para simplificar el problema, consideraremos discreto tanto el espacio de estados como el espacio de acciones. No obstante, el efecto de las acciones será estocástico. El objetivo será, por tanto, encontrar una política para navegar hacia el destino minimizando la posibilidad de colisionar con los obstáculos. Para ello, se utilizarán y compararán diferentes algoritmos de aprendizaje por refuerzo.

Lecturas recomendadas

Se recomiendan las siguientes lecturas antes de comenzar con el trabajo:

- Stuart Russell & Peter Norvig. Inteligencia Artificial, un enfoque moderno (2010), capítulo 22.
- Sebastian Thrun & Wolfram Bulgard. Probabilistic Robotics (2006), capítulo 14.

También se recomienda repasar el tema de la asignatura dedicado a los procesos de decisión de Markov y al aprendizaje por refuerzo, tanto la parte teórica como práctica.

Definición del problema

Suponemos un mapa de obstáculos definido por una matriz de ocupación, es decir, una matriz binaria en donde un 0 significa un espacio libre de obstáculos y un 1 significa la presencia de un obstáculo.

Cada celda de la matriz se corresponderá con una superficie de 0.5 x 0.5 metros en el entorno. Por tanto, un mapa de 10 x 10 metros vendrá definido por una matriz de 20 filas x 20 columnas.

El robot podrá localizarse en cualquier celda libre de obstáculos y podrá desplazarse a cualquiera de las 8 celdas adyacentes, siempre que esté libre de obstáculos.
[1]

Consideraremos también una celda como posición inicial del robot y una celda como posición final. Así pues, la definición del problema viene dada por:

- Matriz de ocupación O de m filas por n columnas. En donde una celda de la matriz contiene el valor 1 si la correspondiente posición está ocupada por obstáculo o 0 si es libre de obstáculo. Para evitar que el robot se salga del mapa, todo el contorno del mapa serán obstáculos, es decir tanto la primera como la última fila de la matriz y tanto la primera como la última columna de la matriz contendrán el valor 1.
- Posición inicial $\mathrm{P_{init}} = (\mathrm{x_a}, \mathrm{y_a})$ y posición final $\mathrm{P_{end}} = (\mathrm{x_b}, \mathrm{y_b})$ del robot, con $0 \leq \mathrm{x_a}, \mathrm{x_b} < \mathrm{n}$ y $0 \leq \mathrm{y_a}, \mathrm{y_b} < \mathrm{m}$. Ambas posiciones deben ser libres de obstáculos.
- Conjunto de estados $\mathrm{S} = (\mathrm{x}, \mathrm{y})$ con $\leq \mathrm{x} < \mathrm{n}$ y $0 \leq \mathrm{y} < \mathrm{m}$. Serían todas las posibles posiciones en las que se encuentra el robot. El conjunto de estados incluye tanto las celdas con obstáculos como las celdas sin obstáculos. Es decir, no se impide por definición que el robot se coloque en una celda con obstáculos, pero estará fuertemente penalizado por la función de recompensa.
- Conjunto de acciones $\mathrm{A} = \{\mathrm{N}, \mathrm{NE}, \mathrm{E}, \mathrm{SE}, \mathrm{S}, \mathrm{SW}, \mathrm{W}, \mathrm{NW}, \mathrm{wait}\}$ que representan los movimientos en las 8 posibles direcciones. La acción wait consiste en no moverse y quedarse en la misma posición.

[1] En problemas reales de robótica se suele utilizar una aproximación similar, aunque la resolución del mapa es mucho mayor (habitualmente 5 x 5 cm) y tanto el espacio de estados como de acciones es continuo, en lugar de discreto. Pero en la definición que aquí usaremos se asumen ciertas simplificaciones para facilitar la aplicación de algoritmos sencillos.

- Función de transición $T: SxA \rightarrow S$ que determina el estado resultante al aplicar una acción. Esta función será estocástica, pues modelizará un cierto error en la ejecución del movimiento y vendrá dada por la siguiente definición: $T(s, a) = s'$, en donde:
  - s es el estado de partida.
  - a es la acción a ejecutar.
  - s' es el estado tras ejecutar la acción a sobre s.
  - Si s es contiene obstáculos, entonces $s' = s$ independientemente de la acción a, es decir, si el robot está sobre un obstáculo, no podrá moverse.
  - Si $a = \text{wait}$, entonces $s' = s$, el efecto de dicha acción es determinista.
  - Para el resto de acciones, si s es libre de obstáculos, con una probabilidad $1 - P_{error}$ se producirá el movimiento deseado. Dado $s = (x, y)$, vendrá definido por:
    - Si $a = N$, $s' = (x, y - 1)$
    - Si $a = NE$, $s' = (x + 1, y - 1)$
    - Si $a = E$, $s' = (x + 1, y)$
    - Si $a = SE$, $s' = (x + 1, y + 1)$
    - Si $a = S$, $s' = (x, y + 1)$
    - Si $a = SW$, $s' = (x - 1, y + 1)$
    - Si $a = W$, $s' = (x - 1, y)$
    - Si $a = NW$, $s' = (x - 1, y - 1)$
  - Por otra parte, con una probabilidad P_{error} se producirá un movimiento de error a', que vendrá dado por una desviación a la izquierda o a la derecha sobre la dirección deseada. Es decir:
    - Si $a = N$, a' podrá ser NE o NW, de forma equiprobable.
    - Si $a = NE$, a' podrá ser N o E, de forma equiprobable.
    - Si $a = E$, a' podrá ser NE o SE, de forma equiprobable.
    - Si $a = SE$, a' podrá ser E o S de forma equiprobable.
    - Si $a = S$, a' podrá ser SE o SW de forma equiprobable.
    - Si $a = SW$, a' podrá ser E o W de forma equiprobable.
    - Si $a = W$, a' podrá ser SW o NW de forma equiprobable.
    - Si $a = NW$, a' podrá ser N o W de forma equiprobable.
  - Función de recompensa $R(s, a): \mathbb{R}$, que tendrá el siguiente valor:
    - $R(s, a) = K1$ si s es un estado distinto del destino y a es wait
    - $R(s, a) = R(s)$ en cualquier otro caso
    - $R(s) = K2$ si el estado s es un obstáculo.
    - $R(s) = -d$ si el estado s no es un obstáculo, siendo d la distancia Euclídea desde la posición s al destino.

    Los valores $K1$ y $K2$ deberán ser negativos. Se recibirá una penalización alta ($K2$) si el robot está en un obstáculo, y una penalización $-d$ que será menor cuanto más cerca esté el robot del destino. La penalización $K1$ es para que el

robot no se quede esperando en una posición diferente al destino. Se recomienda $\mathrm{K1} = -100$ y $\mathrm{K2} = -1000$ para el mapa de ejemplo.

El problema consiste en encontrar una política que asocie una acción a cada posible estado.

Guión de ejemplo

El notebook *RoboticsRL* adjunto a este enunciado contiene código explicativo para entender mejor la definición del problema. En dicho notebook no sólo se define el problema, sino que lo resuelve mediante el método de iteración de políticas. También se incluyen métodos para realizar una representación gráfica utilizando la biblioteca matplotlib. Se incluye un mapa de ejemplo que deberá ser utilizado para la comparativa de algoritmos. Este notebook es, por tanto, la base para la realización del trabajo. No obstante, no es obligatorio su uso, el alumno es libre de adaptarlo y reutilizarlo (o no) a su conveniencia.

Algoritmos de aprendizaje por refuerzo

Se utilizarán los siguientes algoritmos de aprendizaje por refuerzo:

- Q-Learning y Monte-Carlo, algoritmos descritos en el tema de aprendizaje por refuerzo de la asignatura.
- Algoritmo SARSA (state-action-reward-state-action). Este es un algoritmo muy parecido a Q-Learning, aunque con diferencias significativas en su enfoque y forma de actualización de los valores de la función de valor. En la referencia bibliográfica [2] se puede encontrar una detallada descripción del algoritmo.

Trabajo a realizar

El trabajo consistirá en aplicar algoritmos de aprendizaje por refuerzo para la resolución del problema descrito, es decir, supondremos desconocida tanto la función de transición como la función de recompensa.

- Primera convocatoria (junio):
  - Resolución del problema utilizando Q-Learning, para ello se puede usar la biblioteca mdptoolbox
  - Implementación propia del algoritmo Monte-Carlo.
  - Implementación propia del algoritmo SARSA
  - Comparativa de algoritmos y valores de hiperparámetros sobre el mapa de ejemplo.
  - Diseño de dos nuevos mapas y comparativa de algoritmos y valores de hiperparámetros.
- Segunda convocatoria (julio):

    - Resolución del problema utilizando Q-Learning, para ello se debe usar una implementación propia.
    - Implementación propia del algoritmo Monte-Carlo.
    - Implementación propia del algoritmo SARSA
    - Comparativa de algoritmos y valores de hiperparámetros sobre el mapa de ejemplo.
    - Diseño de dos nuevos mapas y comparativa de algoritmos y valores de hiperparámetros.
- Tercera convocatoria (noviembre):
    - Resolución del problema utilizando Q-Learning, de dos formas diferentes:
        - Utilización de la biblioteca mdptoolbox
        - Implementación propia.
    - Implementación propia del algoritmo Monte-Carlo.
    - Implementación propia del algoritmo SARSA
    - Comparativa de algoritmos y valores de hiperparámetros sobre el mapa de ejemplo.
    - Diseño de dos nuevos mapas y comparativa de algoritmos y valores de hiperparámetros.

Si el trabajo se entrega antes de junio, según el procedimiento de evaluación por curso, entonces se deberán seguir los criterios de entrega de la convocatoria de junio.

Documentos a entregar y descripción de la evaluación

La entrega del trabajo contendrá dos partes:

- Código fuente en Python
- Documentación

Adicionalmente, será obligatorio realizar una presentación y defensa ante el profesor, una vez hecha la entrega. Para ello, el profesor convocará a cada grupo de alumnos por email. Para aprobar el trabajo es obligatorio aprobar la parte de la presentación y defensa del mismo.

Documentación

La documentación debe tener un mínimo de 6 páginas y un máximo de 12 páginas. Se valorará la utilización del sistema LaTeX [5] y se deberá seguir un esquema de artículo científico, de acuerdo al formato *IEEE Conference Proceedings* [6]. La documentación deberá contener las siguientes secciones:

- Resumen o *Abstract*
- Introducción

- Preliminares
- Implementación
- Pruebas y experimentación
- Conclusiones
- Bibliografía

Presentación y defensa

Como parte de la evaluación del trabajo se deberá realizar una presentación y defensa del mismo, para lo que se citarán a los alumnos de manera conveniente. En esa reunión se deberá llevar preparada y realizar una presentación (PDF, PowerPoint o similar) de 20 minutos en la que deberán participar activamente todos los miembros del grupo. Cada miembro debe hablar por igual. Esta presentación deberá seguir a grandes rasgos la misma estructura de la documentación entregada, haciendo especial mención a los resultados obtenidos y al análisis crítico de los mismos. En los siguientes 20-40 minutos, el profesor procederá a hacer preguntas a cada uno de los miembros del grupo, que podrán ser preguntas sobre la documentación, el código realizado o cuestiones técnicas, teóricas o algorítmicas. Las preguntas podrán estar dirigidas a los dos miembros del grupo o sólo a uno de ellos. Adicionalmente, el profesor podrá solicitar la realización sobre la marcha de pequeñas modificaciones en el código entregado y/o pruebas adicionales. Cada uno de los alumnos del grupo deberá demostrar un completo dominio sobre todas las partes del código y la documentación entregada, independientemente de la manera en la cual se hayan distribuido el trabajo. Para aprobar este trabajo es obligatorio aprobar la parte de la presentación y defensa, siendo incluso posible que para un grupo de dos alumnos, uno de ellos supere la presentación y defensa y el otro no.

Criterios de evaluación

Para la evaluación se tendrá en cuenta el siguiente criterio de valoración, considerando una nota máxima de 4 en total para el trabajo:

- **Código fuente y experimentación (hasta 1.5 puntos)**: se valorará la claridad y buen estilo de programación, corrección, eficiencia y usabilidad de la implementación, así como calidad de los comentarios. En ningún caso se evaluará un trabajo con código copiado directamente de Internet o de otros compañeros.
- **Documentación (hasta 1.5 puntos)**: se valorará la claridad de las explicaciones, el razonamiento de las decisiones, el análisis y presentación de resultados y el correcto uso del lenguaje. La elaboración de la memoria debe ser original, por lo que no se evaluará el trabajo si se detecta cualquier copia del contenido.
- **Presentación y defensa (hasta 1 punto)**: se valorará la claridad de la presentación y la buena explicación de los contenidos del trabajo, así como las respuestas a las preguntas realizadas por el profesor. Para aprobar el trabajo es obligatorio sacar al menos 0.5 puntos en la presentación y defensa del mismo, pudiendo obtener una nota diferente cada uno de los integrantes de un grupo.

IMPORTANTE: Cualquier plagio, compartición de código o uso de material que no sea original y del que no se cite convenientemente la fuente, significará automáticamente la calificación de cero en la asignatura para todos los alumnos involucrados. Por tanto, a estos alumnos no se les conserva, ni para la actual ni para futuras convocatorias, ninguna nota que hubiesen obtenido hasta el momento. Todo ello sin perjuicio de las correspondientes medidas disciplinarias que se pudieran tomar.

Bibliografía

[1] Russell, Stuart and Norvig, Peter. Artificial Intelligence: A Modern Approach (2010). Chapter 22.

[2] Richard S. Sutton y Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[3] Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers, 2010.

[4] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra y Martin Riedmiller. *Deterministic Policy Gradient Algorithms*. 2015.

[5] *https://es.wikibooks.org/wiki/Manual_de_LaTeX*

[6] *https://www.ieee.org/conferences/publishing/templates.html*