Get started Open in app

Markel Sanz Ausin

15 Followers · About Follow

You have **1** free member-only story left this month. Sign up for Medium and get an extra one

Introducción al aprendizaje por refuerzo. Parte 1: el problema del bandido multibrazo.

Markel Sanz Ausin Mar 22 · 5 min read ★

Durante estos artículos, haré referencia a la terminología utilizada en inglés, ya que es la más común y la mejor forma de encontrar información en la bibliografía.

Hablemos del aprendizaje por refuerzo (**Reinforcement Learning**, en inglés). Ésta es una técnica de inteligencia artificial (IA) en la que un **agente inteligente (agent)** tiene que interactuar con un **entorno (environment)**, escogiendo una de las **acciones (action)** que el entorno ofrece en cada uno de los posibles **estados (state)**, e intentar conseguir la mayor **recompensa (reward)** posible a través de esas acciones.

Al principio, el agente no conoce nada sobre el entorno, por lo que tomará acciones de forma aleatoria. Si una acción trae una recompensa positiva, el agente deberá aprender a escoger esa acción más frecuentemente, mientras que si una acción trae una recompensa negativa, el agente deberá aprender a escoger esa acción menos frecuentemente. Así, **el agente aprenderá a escoger las acciones que maximicen la suma de recompensas recibidas,** también conocida como el **retorno (return)**.

El problema del bandido multibrazo (multi-armed bandit problem)

El problema del bandido multibrazo se puede ver como un problema de máquinas tragaperras, como en un casino. Si tenemos un número N de tragaperras, y cada una nos da una recompensa positiva con probabilidad **p**, y ninguna recompensa con probabilidad **(1-p)**, ¿podemos crear un agente que maximice las recompensas

escogiendo jugar siempre en la tragaperras que más beneficio nos vaya a proporcionar? Pues la idea es la misma; **tenemos un bandido con N brazos, y cada brazo tiene una probabilidad distinta de darnos una recompensa positiva. El objetivo es crear un agente que maximice esas recompensas.**

Para este artículo, usaremos un bandido de **N=5 brazos (5-armed bandit)**. Éstas serán las probabilidades de cada brazo de dar una recompensa positiva: **[0.1, 0.3, 0.05, 0.55, 0.4]**. Como podemos ver, la mejor acción entre estas cinco es tirar del cuarto brazo. Sin embargo, el agente no dispone de esta información. Por lo tanto, **deberá probar a tirar de todos los brazos varias veces, e ir aprendiendo cuál de todos es el mejor. Cuando vaya acumulando más información, empezará a tomar mejores decisiones, y a recibir mejores recompensas más frecuentemente.**

Política ε-voraz (ε-greedy policy)

Ésta será la **política (policy)** que decidirá qué acciones toma nuestro agente. La política ε-voraz consiste en que el agente casi siempre tomará la mejor acción posible dada la información que posee. Sin embargo, de vez en cuando, **con una probabilidad de ε, el agente tomará una acción completamente al azar.** De esta forma, si tras la primera acción el agente ha obtenido una recompensa positiva, no se quedará atascado escogiendo esa misma acción todo el rato. Con probabilidad ε el agente explorará otras opciones. Este valor ε lo decidiremos nosotros, y será la forma que tengamos de equilibrar el problema de **exploración y explotación (exploration vs. exploitation)**. La exploración consiste en explorar todas las acciones posibles varias veces para ver cuál es la mejor, a pesar de que durante esa exploración no obtengamos recompensas muy buenas. La explotación consiste en maximizar las recompensas, por lo que el agente escogerá la mejor de las acciones cada vez. Por ello, es importante equilibrar la exploración y la explotación: si solo exploramos dos de las cinco acciones posibles, no sabremos si las acciones que nunca hemos probado nos traerán mayores recompensas, por lo que la exploración es necesaria; y sin embargo, si nos pasamos todo el rato explorando todas las opciones una y otra vez, nunca utilizaremos ese conocimiento para poder escoger la mejor acción y conseguir la mayor recompensa posible.

El código

Crearemos un agente que resuelva el problema del bandido multibrazo. Ejecuta tú mismo el código paso a paso **en este enlace** o sigue leyendo para ver el código sin

ejecutarlo.

Empecemos con el código. Vamos a escribir una función que nos permita escoger uno de los cinco brazos, y nos devuelva una recompensa, dependiendo de la probabilidad de dicho brazo.

Algoritmo 1. Función para tirar de un brazo.

Ahora, crearemos la función que decide qué acción debe tomar el agente. Con probabilidad epsilon tomará una acción aleatoria; y si no, tomará la acción con mejor media de recompensas.

Algoritmo 2. Función para la política ε-voraz.

Por último, definamos las probabilidades de los brazos como hemos mencionado anteriormente, y definamos los parámetros epsilon y la cantidad de iteraciones o acciones que vamos a tomar. Definamos también tres listas donde guardaremos información sobre las acciones ejecutadas hasta este momento y las recompensas conseguidas para cada brazo.

Ejecutamos el algoritmo llamando a las funciones, y guardamos las recompensas.

Algoritmo 3. Ejecutando el bucle principal e imprimiendo la recompensa.

Al final, vemos que el algoritmo escribe esto:

```
Average reward for bandits is [0.19, 0.31, 0.04, 0.55, 0.40]
Best bandit is 3 with an average observed reward of 0.55
Total observed reward in the 1000 episodes has been 493
```

Tenemos la lista con la media de recompensas producidas por cada brazo, la cual se parece bastante a las probabilidades que hemos definido para cada uno de ellos. Es decir, con 1000 iteraciones hemos encontrado la probabilidad de éxito de cada brazo, y ahora sabemos que el cuarto brazo es el mejor (índice 3 en la lista). En este proceso, hemos obtenido 493 recompensas positivas. Es un número muy alto, teniendo en cuenta que escogiendo el mejor brazo desde el principio hubiéramos obtenido 550 recompensas positivas (porque tiene una probabilidad de 0.55). Podemos cambiar el

valor de epsilon y la cantidad de iteraciones, para ver cómo estos valores afectan a la recompensa total recibida.

Si deseas ejecutar y ver el código completo paso a paso, **hazlo en este enlace**.

Referencias:

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Multi-armed bandit problem (Wikipedia)

La serie completa de introducción al aprendizaje por refuerzo:

1. **Parte 1: el problema del bandido multibrazo**
2. Parte 2: Q-Learning
3. Parte 3: Q-Learning con redes neuronales, algoritmo DQN
4. Parte 4: Double DQN y Dueling DQN

Some rights reserved

Aprendizaje Por Refuerzo Machine Learning TensorFlow Reinforcement Learning

Redes Neuronales

About Help Legal

Get the Medium app


Download on the
App Store


GET IT ON
Google Play