Bandidos estocásticos: introducción, algoritmos y experimentos

Stochastic bandits: introduction, algorithms and experiments

Autor

Marcos Herrero Agustín

Director

Miguel Palomino Tarjuelo

Trabajo de Fin de Grado del Doble Grado en Ingeniería Informática - Matemáticas

Facultad de Informática, Universidad Complutense de Madrid

Curso 2021-2022

Resumen

Encuadrados dentro del aprendizaje por refuerzo, los bandidos constituyen una solución efectiva a las instancias más simples del dilema de la exploración-explotación. Un problema de bandidos consiste en un juego secuencial entre un agente y un entorno en el que en cada ronda el agente dispone de una serie de acciones disponibles y debe escoger una de ellas y recibir, como consecuencia, una cierta recompensa. Basándose en las recompensas recibidas anteriormente, el agente deberá ir perfeccionando sus decisiones de forma que consiga una recompensa acumulada lo mayor posible al final del juego. Para ello, tendrá que mantener un equilibrio entre la exploración de acciones poco probadas y la explotación de la mejor acción según la información de que dispone.

Desde el año 1933, en el que Thompson planteó la necesidad de aplicar un modelo como el de los bandidos a las pruebas clínicas, se han encontrado numerosas aplicaciones de los mismos. El A/B testing y los sistemas recomendadores son algunos de los campos en los que hoy en día el uso de bandidos resulta esencial, mientras que otros como el encaminamiento en redes o la inteligencia artificial en videojuegos son muy prometedores.

El modelo de los bandidos cuenta con numerosas y muy diversas variantes, pero aquí nos centraremos en los bandidos estocásticos. Son aquellos en los que cada acción disponible está asociada a una distribución de probabilidad, desconocida a priori por el agente, de la que se extrae la recompensa correspondiente cuando la acción es seleccionada. El agente tendrá, por tanto, que tratar de estimar la media de estas distribuciones a partir de las recompensas que obtenga de ellas.

En este trabajo se realiza un análisis de los algoritmos de bandidos, principalmente de aquellos pensados para bandidos estocásticos, utilizando una métrica conocida como remordimiento, que evalúa cómo de buenos son los resultados obtenidos en relación con cómo podrían ser. Se estudian, entre otros aspectos, los algoritmos de Explora- Primero y de la Cota Superior de Confianza (UCB), que obtienen buenos resultados al enfrentarse con bandidos estocásticos.

Palabras clave

Exploración-explotación. Bandidos. Remordimiento. Bandidos estocásticos. Explora-Primero. UCB.

Abstract

Framed in reinforcement learning, bandits are an effective solution to the most simple instances of the exploration-exploitation dilemma. A bandit problem consists in a sequential game between an agent and an environment in which, in every round, the agent has a set of available actions and must choose one of them and receive, as a result, a certain reward. Based on rewards received previously, the agent should improve its decisions in order to get a cumulative reward as big as possible at the end of the game. For this purpose, it will have to keep a balance between the exploration of actions that have been chosen few times and the exploitation of the best action according to the information it has.

Since 1933, year in which Thompson proposed the need to apply a bandit-like model to clinical trials, there have been found a lot of applications for them. A/B testing and recommender systems are just some of the fields where the use of bandits is essential nowadays. Other uses, like routing in networks or artificial intelligence in videogames, are very promising.

The bandit model has numerous and very diverse variants, but here we will focus on stochastic bandits. They are those in which every available action is associated to a probability distribution, unknown by the agent a priori, from which the corresponding reward is sampled when the action is selected. The agent must, then, try to estimate the mean of these distributions based on the rewards obtained from them.

In this dissertation we make an analysis of bandit algoritms, especially those thought for stochastic bandits, using a metric known as regret, that evaluates how good the outcomes are in relation to how good they could be. We study, among other aspects, the algorithms Explore-First and Upper Confidence Bound (UCB), which get good outcomes against stochastic bandits.

Keywords

Exploration-exploitation. Bandits. Regret. Stochastic bandits. Explore-First. UCB.

Índice

Agradecimientos

Aunque sin duda hay muchísimas personas a las que tengo que agradecer el haber llegado hasta aquí, si las nombrara a todas probablemente me daría para rellenar otro TFG. Por ello, creo que lo apropiado es que me quede en lo esencial.

En primer lugar, quiero agradecer a mi tutor Miguel su dedicación y su excelente labor supervisando mi trabajo. Agradezco también las numerosas sugerencias que ha hecho y su interés e implicación en que consiguiera acabar este trabajo a tiempo y de la mejor manera posible.

Por encima de todo quiero agradecer a mis amigos el apoyo que me han dado durante estos años de universidad (aunque a alguno lo haya conocido más recientemente), sin el cual seguramente habría abandonado la carrera hace mucho tiempo. Gracias por vuestra paciencia y comprensión y por ser capaces de sacarme una sonrisa en los mejores y en los peores momentos. En definitiva, gracias por quererme, pese a que no siempre es fácil y a menudo no haya estado a la altura. Sabed que os quiero y que sois parte fundamental de la persona en la que me he convertido. Vaya a donde vaya, os llevo conmigo. Hasta siempre, hermanos *altrajadores*.

1. Introducción

Motivación

Uno de los problemas más importantes que se tratan en el aprendizaje automático es el de la toma de decisiones secuenciales en condiciones de incertidumbre. La disyuntiva entre explotar la mejor alternativa conocida o explorar para mejorar el conocimiento que uno tiene sobre el entorno plantea el dilema de la exploración-explotación. El modelo de los bandidos afronta la versión más simple de este problema. A grandes rasgos, un problema de bandidos consiste en una serie de rondas en las que un agente ha de escoger una entre un conjunto prefijado de acciones y recibir, tras su decisión, una recompensa asociada. En nuestro modelo las mismas acciones están disponibles en todos los pasos y la elección de una en un momento dado no penaliza los resultados que se obtendrán en rondas sucesivas.

William R. Thompson introdujo los algoritmos de bandidos en un artículo publicado en 1933 en *Biometrika* [1]. Thompson estaba interesado en las pruebas clínicas y criticó que se esperara a disponer de un gran volumen de datos para empezar a usarlos en la toma de decisiones. Para Thompson, las pruebas debían ir adaptándose sobre la marcha conforme se iban obteniendo resultados. De hecho, dado que nunca se tienen suficientes observaciones para poder decir sin ninguna duda que un tratamiento es mejor que otro, el modelo propuesto debía prolongarse en el tiempo y sustituir a la forma que hasta entonces se tenía de tomar la decisión. Básicamente, Thompson proponía que conforme un determinado tratamiento demostraba ser mejor que los otros se aumentara sobre la marcha la fracción de individuos tratados con él. En contrapartida, la fracción de pruebas realizadas con el tratamiento que fuera mostrándose menos efectivo debía irse reduciendo. Según este modelo, nunca se llegaría a tomar la decisión definitiva de adoptar un tratamiento y descartar el otro sino que todos los tratamientos que demostraran alguna efectividad se seguirían utilizando simplemente actualizándose su frecuencia de uso en función de su efectividad manifestada.

Aunque los algoritmos de bandidos nunca han llegado a utilizarse para pruebas clínicas, sí que se han utilizado extensivamente en aplicaciones web: A/B testing, posicionamiento de anuncios, sistemas recomendadores... También se han utilizado para, por ejemplo, la selección de ruta en redes de datagramas y para la inteligencia artificial en videojuegos.

Existen distintas variantes del modelo de los bandidos en función de la forma en que se generan las recompensas o de la información adicional (contexto) a la que el agente tiene acceso en cada decisión. En este trabajo nos centraremos fundamentalmente en los bandidos estocásticos, que son aquellos en los que el agente no tiene acceso a ningún contexto y la recompensa de cada ronda se extrae de una distribución de probabilidad asociada a la acción escogida por el agente en dicha ronda. Se trata de la variante más básica y, probablemente, la que tiene un menor número de aplicaciones prácticas, pero su comprensión resulta esencial para poder entender otros tipos de bandidos más complicados, pues variaciones de muchas de las ideas y algoritmos que veremos aquí se reutilizan en estos otros entornos.

Objetivos y plan de trabajo

El objetivo principal de este trabajo es el estudio de los bandidos estocásticos y de algoritmos que resuelven problemas en los que aparecen este tipo de bandidos. Se hará hincapié en la explicación de las ideas e intuiciones de los mismos y en la realización de experimentos que las avalen. Enfatizaremos también el análisis de estos experimentos por medio de la métrica conocida como remordimiento. No nos desligamos por completo del estudio téorico de los bandidos y sus algoritmos, pues este es en parte necesario para efectuar las pruebas y entender y comentar los resultados obtenidos, lo que hará necesario estudiar algunos de los fundamentos matemáticos de los bandidos. Sin embargo, sí que se omitirán todas las demostraciones que se considere que no aportan mucho a la comprensión intuitiva del tema. Estas demostraciones se incluirán en mi TFG de Matemáticas, que complementará lo expuesto en este tratando aproximadamente los mismos temas pero desde una perspectiva formal y rigurosa en lugar de empírica.

Además, se plantea que este trabajo pueda servir de base o referencia inicial para otros TFGs que se realicen en años venideros y necesiten apoyarse en conceptos relacionados con bandidos estocásticos.

La estructura de la memoria es la siguiente:

- Las secciones 1 y 2 constituyen la introducción, que reúne la motivación, los objetivos y el plan de trabajo y el material utilizado. El contenido de ambas secciones es el mismo, salvo que la sección 1 está en español y la sección 2 en inglés.
- La sección 3 introduce el dilema de la exploración-explotación como motivación de lo que viene después.
- La sección 4 sirve de introducción a los bandidos en general, describiendo sus primeras intuiciones y su modelo matemático general y distinguiendo algunos de sus tipos.
- La sección 5 presenta los fundamentos matemáticos necesarios para el estudio de los bandidos estocásticos en las secciones siguientes.
- La sección 6 comienza el estudio de los bandidos estocásticos, concretando para estos el modelo general de bandidos visto en la sección 4.
- Las secciones 7 y 8 explican en detalle sendos algoritmos que funcionan bien con bandidos estocásticos: el algoritmo Explora-Primero y el algoritmo de la Cota Superior de confianza (UCB).
- Las secciones 9 y 10 constituyen la conclusión del trabajo. El contenido de ambas secciones es el mismo, salvo que la sección 9 está en español y la sección 10 en inglés.

Material utilizado

Bibliografía principal

La parte teórica sobre bandidos incluida en este trabajo está basada fundamentalmente en los capítulos 1, 4, 6 y 7 del libro [2]. Asimismo, algunos de los experimentos realizados parten de ejercicios planteados en [2], aunque en la mayoría de los casos no he seguido exactamente la línea planteada por estos. Los libros [3, 4] han servido como material de consulta complementario. Para los fundamentos matemáticos se ha recurrido principalmente a los libros [5, 6, 7, 8].

Lenguajes de programación y bibliotecas

Para la realización de los experimentos, que constituyen parte fundamental de este trabajo, se ha utilizado el lenguaje de programación Python junto con las bibliotecas matplotlib (para la realización de gráficas) y scipy (para todo lo relacionado con distribuciones de probabilidad). He escogido este entorno debido a su facilidad su uso y a que se adapta perfectamente a mis necesidades para este trabajo.

Al comenzar la elaboración del TFG ya contaba con un conocimiento superficial del lenguaje Python y de la biblioteca matplotlib, que solo ha sido necesario refrescar, ya que ninguna de las pruebas realizadas tiene una dificultad elevada. Por el contrario, la biblioteca scipy no la conocía antes y la busqué específicamente para este trabajo, precisamente por la necesidad de contar con una biblioteca que permitiera manejar variables aleatorias cómodamente. En un principio, de hecho, utilicé otra biblioteca, simpy, que también proporcionaba las funcionalidad que necesitaba, pero acabé sustituyéndola por ser excesivamente lenta. En cualquier caso, ambas bibliotecas cuentan con una documentación clara y no ha habido apenas ninguna dificultad en comenzar a utilizarlas.

Todo el código utilizado para los experimentos realizados se encuentra, organizado por secciones, en mi repositorio de GitHub [9]. De todas formas, no es necesario consultarlo para seguir el hilo del trabajo, ya que todos los scripts realizados generan figuras que se incluyen y comentan en algún punto de esta memoria.

2. Introduction

Motivation

One of the most important problems addressed in machine learning is sequential decision making under uncertainty. The quandary between exploiting the best known alternative or exploring to improve knowledge about the environment poses the exploration-exploitation dilemma. The bandit model confronts the simplest version of this problem. In broad terms, a bandit problem consists of a series of rounds in which an agent has to choose between a prefixed set of actions and receive, after its decision, and associated reward. In our model the same actions are available at every step and choosing one of them at a certain time doesn't penalize the outcomes that may be obtained in following rounds.

William R. Thompson introduced bandit algorithms in an article published in 1933 in Biometrika journal [1]. Thompson was interested in clinical trials and criticized that professionals waited until having a great volume of data to begin using them in decision making. According to Thompson, trials should be adapted on the fly as results were obtained. In fact, as we never have enough samples to tell without any doubt that one treatment is better than another, his suggested model should extend in time and substitute the previous way to make the decision. Basically, Thompson proposed that, as a particular treatment proved being better than the other, the fraction of individuals treated with it had to be increased. In contrast, the fraction of tests performed with the treatment that seemed less effective should be reduced. Following this model, we would never make the ultimate decision of adopting a treatment or rejecting other. All treatments that show some effectivity would be used, simply updating its frequency of use based on its exhibited effectivity.

Although bandit algorithms have never been applied to clinical trials, they have been extensively used in web applications: A/B testing, advert placement, recommender systems... They have also been applied, for example, to routing within a datagram network and to artificial intelligence in videogames.

There exist several variants of the bandit model according to the way rewards are generated or additional information (context) to which the agent has access on every decision. In this dissertation we will basically be focusing on stochastic bandits: those in which the agent doesn't have access to any context and the reward at each round is extracted from a probability distribution associated to the action chosen by the agent at that round. This is the most basic and, probably, the less applicable in practice variant, but it is important to understand this type of bandits if we want to be able to understand others which are more complex. That is because variations of a lot of the ideas and algorithms that we will be seeing here are reused in those other environments.

Objectives and work plan

The main objective of this dissertation is the study of stochastic bandits and of algorithms that solve problems in which this kind of bandit appears. We will emphasize the explanation of ideas and intuitions and the conducting of experiments that support these explanations. We will also stress the analysis of these experiments using the metric known as regret. We won't deviate entirely from the theoretical study of bandits and its algorithms, as part of it is necessary to carry out the tests and understand and comment its outcomes. This will make necessary to study some of the mathematical fundamentals of bandits. However, we will omit all proofs that we consider contribute little to the intuitive understanding of the subject. These proofs will be included in my Bachelor's Mathematics Dissertation, which will complement this one. The Mathemathics Dissertation will address roughly the same topics but from a formal and rigorous approach instead of empirical.

We also think this dissertation may serve as a basis or initial reference for others to be carried out in the following years and which need to rest on concepts related to stochastic bandits.

This dissertation's structure is the following:

- Sections 1 and 2 constitute the introduction, which presents the motivation, the objectives and work plan and the materials used. The content of both sections is the same, except that Section 1 is written

in Spanish and Section 2 in English.

- Section 3 introduces the exploration-exploitation dilemma as a motivation of what comes after.
- Section 4 serves as an introduction to bandits in general, describing its first intuitions and its general mathematical model and distinguishing some of its types.
- Section 5 presents the mathematical fundamentals required for the study of stochastic bandits in following sections.
- Section 6 begins the study of stochastic bandits, adapting to them the general model presented in Section 4.
- Sections 7 and 8 each explain in detail an algorithm that works well in the stochastic setting. Section 7 presents the Explore-First algorithm and Section 8 presents the Upper Confidence Bound algorithm (UCB).
- Sections 9 and 10 present conclusion. The content of both sections is the same, except that Section 9 is written in Spanish and Section 10 in English.

Materials used

Main bibliography

The theorethical part about bandits included in this dissertation is essentially based on chapters 1, 4, 6 and 7 from the book [2]. Likewise, some of the experiments conducted start from exercises proposed in [2], although in most cases I haven't followed exactly the line suggested. The books [3, 4] have served as complementary consulting material. To present the mathematical fundamentals we have mainly used the books [5, 7, 8].

Programming languages and libraries

The experiments, which constitute a fundamental part of this dissertation, have been conducted using the Python programming language along with libraries matplotlib (for graph plotting) and scipy (for everything related to probability distributions). I have chosen this environment because it is easy to use and it suits my needs for this dissertation.

When I began elaborating this dissertation I already had some superficial knowledge about Python and the matplotlib library, which was only necessary to refresh as none of the tests carried out has high difficulty. On the contrary, I didn't know the scipy library and I searched for it specifically for this dissertation, precisely because of the necessity of having a library that allowed me to manage random variables conveniently. At the beginning, in fact, I used another library, simpy, that also provided the functionality I needed, but I ended up replacing it because it was excessively slow. In any case, both libraries have a clear documentation and I have found almost no difficulty in using them.

All code used in the experiments conducted can be found, organized by sections, in my GitHub repository [9]. Anyway, it is not necessary to check the code to follow the work thread, as all scripts I made generate figures that are included and commented at some point of this dissertation.

3. El dilema de la exploración-explotación

En esta sección se introduce el concepto de exploración-explotación, el problema general al que los algoritmos de bandidos y el aprendizaje por refuerzo pretenden dar solución.

3.1. ¿Qué es el dilema de la exploración-explotación?

Muchas de las decisiones que tomamos a lo largo del día se escogen con un conocimiento muy limitado de los resultados que puede reportar cada una de las alternativas. En algunas ocasiones, lo conveniente resulta ser escoger la opción que ha dado mejores resultados en el pasado. Sin embargo, en muchas otras se requiere una cierta exploración periódica de las alternativas aparentemente subóptimas para cerciorarse de que realmente lo son. Esto es porque la información de que se dispone puede ser imprecisa debido a que:

- Los resultados asociados a algunas acciones presentan una cierta aleatoriedad, de manera que podemos tener la "mala suerte" de que todas las veces que probemos una determinada opción obtengamos un resultado muy alejado del resultado típico que obtendríamos si ejecutáramos esa misma alternativa un número mucho mayor de veces.
- Los resultados asociados a algunas acciones pueden depender o verse afectados por condiciones que hayan variado desde el momento en que recabamos la información.

Considera las siguientes situaciones:

- **Situación 1:** Abren un nuevo restaurante italiano en tu barrio, en el que ya conoces otro italiano en el que sabes que se come bien. La siguiente vez que decides comer en un italiano, ¿vuelves al restaurante de siempre o pruebas el nuevo?
- **Situación 2:** Eres un youtuber que lleva años subiendo vídeos de géneros A y B. En base a los datos de los que dispones, calculas que los vídeos del género A tienen una media de visualizaciones de x y los del género B una media de y, donde $x > y$. Si tu objetivo es maximizar el número total de visualizaciones de tus vídeos y tiene el mismo coste hacer un vídeo del género A o del género B, ¿es recomendable que de ahora en adelante subas exclusivamente vídeos del primer género?
- **Situación 3:** Eres un médico que tiene que tratar a una serie de pacientes aquejados de una misma dolencia, para la cual dispones de una serie de tratamientos de los que conoces la efectividad que han tenido en ocasiones anteriores frente a esta dolencia. ¿Deberías tratar a todos los pacientes con el tratamiento que ha mostrado una eficacia mayor en el pasado?

¿Qué tienen en común los dilemas anteriores? En todos ellos existe un agente que ha de elegir entre varias alternativas en base a una información limitada sobre las mismas. No se trata, además, de un marco puntual, sino que previsiblemente las mismas alternativas se presentarán al agente en situaciones sucesivas. La única forma en que el agente puede obtener información sobre una alternativa de cara al futuro es escogiéndola, perdiendo con ello la oportunidad de elegir otra en su lugar. Por último, la elección de una u otra de las alternativas no afectará a las opciones de que se dispondrá en ocasiones futuras ni a la bondad de los resultados obtenidos al elegirlas.

En una situación de este tipo el agente podría decidir maximizar su beneficio a corto plazo ejecutando la acción más favorable en función de la información de que dispone, lo que se conoce como **explotación**. Alternativamente, podría decidir probar alguna de las alternativas de la que tiene menos datos para obtener información sobre ellas, lo que se conoce como **exploración**. Nótese que, si bien esta última decisión no es útil para maximizar el beneficio a corto plazo (se está ejecutando una acción para la que a priori se tiene información de que no es la mejor) sí que puede serlo para maximizar el beneficio a largo plazo, al permitir que nos demos cuenta de que hay opciones más favorables que la que hasta el momento considerábamos la mejor. Por este motivo, en la toma de decisiones secuenciales se habla del **dilema de la exploración-explotación**.

En cualquiera de los ejemplos anteriores, u otro que se ajuste a este patrón, no se puede ignorar la explotación de la mejor opción conocida pero tampoco suele ser aconsejable renunciar por completo a la exploración. Si no ejecutamos con más frecuencia las acciones de las que disponemos de información favorable no estaremos haciendo ningún uso de la retroalimentación obtenida de estas alternativas en el pasado. Sin embargo, si solo aplicamos las acciones con información favorable corremos el riesgo de no detectar nunca la opción óptima, ya sea porque la juzgamos mal inicialmente o porque las condiciones que la hacían inferior se modifican con el tiempo. Por tanto, en un sistema complejo o cambiante es fundamental aplicar una estrategia adaptativa que mantenga un equilibrio entre explotación y exploración.

Con la automatización de la toma de decisiones, se hace necesario disponer de algoritmos sistemáticos para tratar el dilema de la exploración-explotación.

3.2. Exploración-explotación en el aprendizaje automático

En el ámbito del aprendizaje automático, el dilema de la exploración-explotación surge cuando existe un bucle de realimentación entre recopilación de datos y toma de decisiones. Esto no ocurre en todas las aplicaciones del aprendizaje automático. Cuando se dispone de datos suficientes para entrenar un modelo fiable desde el principio y no se espera que el modelo se adapte a futuros cambios en el entorno en que opera, se suele optar por separar el entrenamiento y el uso del modelo. Antes de poder utilizarse el modelo ha de entrenarse con los datos de que se dispone. Durante el uso del mismo no se recopilan nuevos datos y puede seguir utilizándose indefinidamente sin realizar una nueva exploración. Este tipo de aprendizaje automático se conoce como **aprendizaje offline**. Un ejemplo de este tipo de aplicación sería la detección facial, ya que se dispone de gran cantidad de ejemplos con los que realizar el entrenamiento previo y no se espera que los rasgos que permiten detectar caras humanas cambien a lo largo del tiempo. En el extremo contrario tenemos los modelos basados en **aprendizaje online**, que se caracterizan por requerir de una adaptación constante. Tras un entrenamiento inicial mínimo o incluso nulo, se espera que el modelo aprenda conforme va tomando decisiones para las diferentes instancias y observando el resultado de las mismas. En general, los sistemas recomendadores son modelos de este tipo.

Una de las tres ramas en que se subdivide el aprendizaje automático es el aprendizaje por refuerzo. Se caracteriza por la presencia de un agente inteligente que toma decisiones secuencialmente, recibiendo una recompensa tras cada una de ellas. En cada paso, el agente se encuentra en un cierto estado y elige una acción de entre las disponibles en ese estado. Tanto la recompensa obtenida como el estado del agente en la siguiente ronda dependen (aunque no están unívocamente determinados) del estado actual y de la acción escogida. El marco matemático para modelar esta clase de situaciones lo proporcionan los llamados procesos de decisión de Markov (*Markov Decision Processes*, MDP).

Sin duda, el dilema de la exploración-explotación en los problemas descritos en las situaciones 1, 2 y 3 de la página 11, al igual que en muchos otros con condiciones similares, puede modelarse mediante un MDP. Sin embargo, recordemos que en estas situaciones la decisión tomada en un momento dado no condiciona las decisiones que se puedan tomar en el futuro ni las recompensas que se puedan obtener de ellas. Merece la pena, pues, simplificar el modelo típico del aprendizaje por refuerzo cuando tratemos problemas de esta índole, prescindiendo del estado. Este MDP simplificado, con un único estado, es lo que conoceremos como bandido.

4. Introducción a los problemas de bandidos

En esta sección se describen los problemas de bandidos, se presenta su modelo matemático general y se discuten sus principales tipos y aplicaciones.

4.1. Los problemas de bandidos

Las situaciones discutidas en 3.1 se resumen en el siguiente ejemplo, paradigmático de los problemas de bandidos:

> Imaginemos que acudimos a un casino en el que hay un cierto número k de máquinas tragaperras y disponemos de n fichas. Por ahora, supondremos que a priori no disponemos de información que distinga las k máquinas. En cada paso podemos elegir una máquina en la que gastar una ficha, con lo que la máquina emitirá una cierta cantidad de dinero, que cobraremos. ¿En qué máquinas deberíamos gastar las n fichas para maximizar la recompensa total acumulada al final?

Un problema de bandidos no es más que una asignación secuencial de n recursos (fichas) entre un conjunto fijo de k acciones (máquinas tragaperras), de manera que entre la asignación de un recurso y el siguiente se recibe una respuesta que se puede utilizar para perfeccionar la decisión en rondas sucesivas. La entidad que procesa cada asignación realizada y genera una respuesta es lo que denominamos **bandido**. Las acciones disponibles, que son las mismas en todos los pasos, se denominan **brazos** del bandido.

> El nombre bandido multibrazo viene de la expresión coloquial "bandido de un solo brazo" (*one-armed bandit*) con que se llamaba a las máquinas tragaperras clásicas en EEUU. Se decían "bandidos" porque te roban el dinero y "de un solo brazo" en referencia la palanca con que se accionaban.

Se modeliza como un juego secuencial de n rondas, donde el número n se denomina **horizonte**. En cada ronda, una unidad de recurso se asigna a un brazo y, a continuación, se observa una cierta recompensa o pérdida. En otras palabras, el agente en cuestión se enfrentará un número n de veces a la decisión de elegir entre k brazos (siempre los mismos) recibiendo, después de cada elección, una realimentación asociada al brazo elegido.

El objetivo del agente habitualmente es maximizar la recompensa total acumulada al final de las n rondas, es decir, la ganancia a largo plazo. Sin embargo, hay excepciones a esta norma general:

- Puede ocurrir que el agente solo esté interesado en conocer una estrategia cercana a la óptima al final de las n rondas, pero no le importen las recompensas acumuladas a lo largo de estas. Los problemas de bandidos de este tipo se denominan problemas de exploración pura, en contraposición a los problemas más habituales que buscan un equilibrio entre exploración y explotación. Puede consultarse más información sobre problemas de exploración pura en el capítulo 33 de [2].
- En cuanto nos salimos del marco teórico y tratamos de aplicar bandidos en la práctica, el objetivo rara vez es tan simple como maximizar la recompensa total. Por ejemplo, si se utiliza un bandido para determinar qué anuncio mostrar a cada usuario que accede a una web normalmente se tomará como recompensa el número de clics, pero es claro que no interesa maximizar el número de clics a toda costa sin tener en cuenta otros factores, como la satisfacción del usuario.

Para los propósitos de este trabajo, nos bastará con considerar que los bandidos que trataremos tienen como objetivo único maximizar la recompensa total acumulada a largo plazo. Habitualmente, las recompensas que pueden proporcionar los brazos del bandido tendrán cotas superior e inferior conocidas, por lo que, si conocemos el horizonte n, también dispondremos de cotas superior e inferior de la recompensa total acumulada.

Un **algoritmo de bandidos** es un conjunto de pasos o instrucciones preestipulados seguidos por el agente durante el juego secuencial para alcanzar su objetivo de maximizar la recompensa total acumulada.

En este trabajo nos centraremos en introducir, explicar y analizar algunos de los algoritmos de bandidos más habituales.

Los bandidos, por ser MDPs de un solo estado, son las instancias más básicas del problema de la toma de decisiones secuencial con información limitada. Tratan de manera natural el dilema de la exploración-explotación, permitiendo afrontar cualquier problema que pueda expresarse en los términos del ejemplo de las máquinas tragaperras anterior. Pese a su simplicidad, realmente proporcionan un marco muy rico, aplicable a una enorme diversidad de circunstancias (entre las que se encuentran las situaciones 1, 2 y 3 de 3.1) y es por este motivo que han sido tan ampliamente estudiados y, pese a ello, siguen despertando gran interés en la comunidad informática y matemática.

Resolver un problema de bandidos no consiste en dar una secuencia de acciones que maximice la recompensa final, dado que a priori se carece de la información necesaria para dar con tal secuencia con una probabilidad alta. Más bien, consiste en dar una estrategia o **política** que, en función las acciones realizadas y recompensas recibidas (que llamaremos historia), determine la siguiente acción a realizar por el agente. El agente, por supuesto, no puede observar el futuro a la hora de tomar su decisión: esta solo puede depender de las acciones y recompensas anteriores a la ronda en que se toma la decisión. De forma similar, las recompensas son determinadas en función de la historia por un **entorno**, del cual el agente solo sabe que se encuentra en una cierta **clase de entornos**. Tanto la política como el entorno pueden estar aleatorizadas, lo que significa que ni las acciones ni las recompensas están totalmente determinadas por la historia previa.

Reflexionemos sobre cómo podríamos dar con una buena solución. Para ello, es preciso primero definir qué entendemos por "buena solución", para lo cual introducimos un concepto central en los problemas de bandidos: el **remordimiento** (*regret* en inglés), que se denota R_n. Que la recompensa que obtengamos siguiendo una determinada estrategia sea considerada aceptable dependerá de cuál sería la recompensa que hubiéramos podido obtener por otra vía. Dada una política alternativa a la del agente, podemos definir el remordimiento de nuestro agente relativo a esta estrategia como la diferencia entre la recompensa total que se espera obtener utilizando esta nueva estrategia y la recompensa total que se espera obtener con la estrategia actual. En este sentido, el remordimiento es una medida para comparar entre sí dos estrategias: será positivo si (de media) la nueva estrategia es mejor que la actual y negativo si (de media) es peor. Nos interesa, sin embargo, medir la bondad de la estrategia elegida en general y no solo respecto a estrategias concretas. Podemos fácilmente extender la noción de remordimiento para comparar con conjuntos de políticas en lugar de con políticas individuales: tomamos como remordimiento la diferencia entre la recompensa esperada por la mejor de las estrategias de la clase (la máxima de las recompensas esperadas) y la recompensa esperada por la estrategia de nuestro agente. Sin embargo, aun con esta extensión seguimos comparando el agente con unas estrategias determinadas. ¿Cómo debemos escoger la familia de políticas con la que comparar? Intuitivamente, para que el remordimiento nos diera una medida absoluta de cómo de bien funciona la estrategia que hemos diseñado, tendríamos que incluir en esta familia, que llamaremos **clase competidora**, a todas las posibles estrategias a seguir por el agente o, al menos, a un subconjunto suficientemente grande para que tengamos garantía de que contiene a una estrategia óptima. No obstante, dependiendo del tipo de bandido al que nos enfrentemos esto no siempre será conveniente. Si el agente no recibe información útil que le permita estimar la recompensa futura emitida por cada brazo (por ejemplo, porque las recompensas de los brazos no se ajusten a una distribución medianamente fija), no tiene mucho sentido exigirle que obtenga un resultado próximo al óptimo. Esto ocurre, por ejemplo, cuando el bandido tratado es de tipo antagonista (veremos luego lo que es).

Por ahora, supongamos que elegimos siempre como clase competidora una que contenga una estrategia óptima, con lo que el remordimiento quedará como la diferencia entre la mayor recompensa esperada y la recompensa esperada por la política que sigue el agente. En este contexto, minimizar el remordimiento equivale a maximizar la recompensa total acumulada. Sin embargo, ambas medidas cuantifican aspectos diferentes: la recompensa total mide el beneficio en términos absolutos, mientras que el remordimiento lo mide en relación con lo que se espera que habría obtenido si se hubieran elegido los brazos de la mejor manera posible.

Si para cada $\alpha \in \mathbb{R}^+$ se puede encontrar una estrategia en la clase competidora cuya recompensa esperada es mayor que α, cualquier estrategia incurrirá en un remordimiento infinito, de manera que esta métrica dejará de resultar útil para el estudio del problema. Una forma sencilla de evitar la aparición este tipo de problemas

es exigir que todas las recompensas estén acotadas (y, normalizándolas, se puede suponer que están en el intervalo $[0, 1]$). En efecto, si existe $C \in \mathbb{R}^+$ tal que el valor absoluto de cualquier recompensa que se pueda obtener en una ronda está acotado superiormente por C, se verifica:

$$R_n \leq Cn\,. \tag{1}$$

Es decir, no solo se garantiza que el remordimiento está siempre acotado, sino también que está en $O(n)$. Siempre habrá que imponer las restricciones necesarias sobre el problema que tratemos para que el remordimiento esté acotado, pues de otro modo será infinito y no podremos utilizarlo para analizar el comportamiento de los algoritmos que diseñemos. No obstante, la exigencia de que todas las recompensas estén acotadas no es necesaria en todos los casos para garantizar que $R_n = O(n)$. Por ejemplo, supongamos que para cada brazo a si se selecciona a la recompensa se extrae de una distribución $N(\mu_a, 1)$. En este caso, ninguna de las recompensas están acotadas: dado que la distribución Gaussiana tiene soporte no acotado, podrán tomar valores arbitrariamente altos con probabilidad no nula. Sin embargo, las recompensas esperadas μ_a sí que están acotadas, lo que nos da la finitud del remordimiento.

Suponiendo que se tiene una cota de la forma (1), cualquier estrategia utilizada para resolver un problema de bandidos, incluso una completamente aleatoria, incurre en un remordimiento en $O(n)$. Nuestro objetivo es producir algoritmos que, bajo ciertas condiciones, mejoren esta cota. Estas restricciones que se impondrán sobre la instancia a tratar serán normalmente más fuertes que las que hemos tratado hasta ahora, aunque no tanto como para no poder aplicarse a una amplia variedad de problemas.

En general, consideraremos una estrategia como buena cuando el remordimiento, visto como función del número n de recursos a asignar, crezca más lento que n cuando n se aproxima a infinito:

$$\lim_{n \to +\infty} \frac{R_n}{n} = 0\,.$$

No obstante, también podríamos ser más exigentes y pedir que el remordimiento esté en $O(\sqrt{n})$ u $O(\log(n))$.

4.2. Un primer experimento

Una vez de acuerdo en que el remordimiento constituye una medida adecuada del rendimiento de un algoritmo de bandidos, podemos utilizarlo para evaluar si una determinada estrategia es acertada. Vamos a utilizarlo para entender por qué es importante mantener un equilibrio apropiado entre exploración y explotación.

Consideremos que en nuestro casino imaginario hay $k = 2$ máquinas tragaperras y disponemos de n fichas para gastar en ellas. El juego constará, pues, de n rondas, en cada una de las cuales introduciremos una ficha en una de las máquinas y recibiremos una recompensa.

En la primera ronda tendremos que escoger una máquina arbitrariamente, dado que no disponemos de información de ninguna de ellas. A partir de la segunda, ya podremos elegir entre continuar explorando o explotar la mejor máquina encontrada hasta el momento. No obstante, no parece buena idea descartar la exploración hasta haber probado, al menos, ambas máquinas, ya que no podemos saber a priori si estamos dejando sin probar una máquina que dé recompensas mucho más altas. Pero, una vez probadas las dos, ¿podemos dedicarnos exclusivamente a la máquina que haya dado el mejor resultado o hemos de seguir explorando? Aunque para maximizar el beneficio a corto plazo puede ser buena idea la exclusividad, si nos interesa el largo plazo y disponemos de un elevado número de fichas esto no es conveniente. Podría ocurrir que simplemente hubiéramos tenido muy mala suerte con alguna de las máquinas y que, por tanto, la primera recompensa que nos dé sea un pésimo criterio para evaluarla. Pero, ¿cuántas veces hemos de probar cada una de las máquinas para tener confianza de que nuestro juicio es acertado y poder olvidarnos de la exploración?

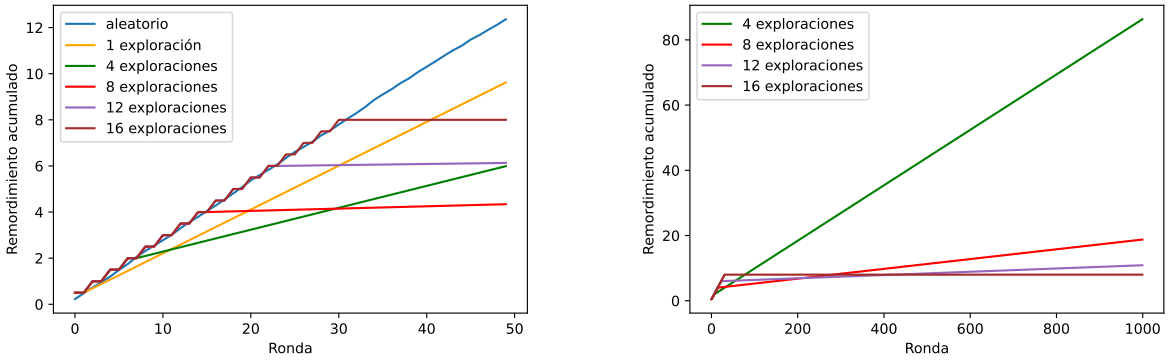
Consideremos que la primera máquina genera las recompensas a partir de una distribución Bernoulli(0.2) y la segunda a partir de una Bernoulli(0.7) (recordemos que una distribución Bernoulli(p) devuelve 1 con probabilidad p y 0 en caso contrario). Claramente, la estrategia óptima *con información completa* sería utilizar siempre la segunda máquina. Pero, precisamente, esta situación se aborda como un problema de bandidos porque no se dispone de información completa. El agente tendrá inicialmente que repartir sus

fichas entre ambas máquinas hasta conseguir determinar con una confianza suficiente la máquina que cree que es la mejor. A partir de ese momento se podrá dedicar exclusivamente a esa máquina.

Vamos a realizar un pequeño experimento para ver cuánto tiempo debería el agente invertir en la exploración en este caso concreto. En particular, comparamos el remordimiento acumulado por el agente para las siguientes estrategias:

- Elegir aleatoriamente en todas las rondas.
- Realizar un número determinado (1, 4, 8, 12 o 16) de exploraciones iniciales de cada uno de los brazos y, una vez transcurridas estas, dedicar todas las rondas restantes a explotar la máquina que ha dado mejores resultados en la exploración.

Al tratarse de un experimento estocástico, es necesario repetir el experimento un cierto número de veces para que los resultados arrojados sean significativos. En este caso se han realizado 100 simulaciones y las gráficas mostradas en la figura 1 son el resultado de promediar los resultados de estas. Hablaremos posteriormente sobre las distintas formas que hay de calcular el remordimiento en la práctica. Por ahora, considerar simplemente que se ha calculado como la media de una muestra aleatoria cuya esperanza es el citado remordimiento.



(a) $n = 50$.

(b) $n = 1000$.

Figura 1: Remordimiento acumulado frente a un bandido Bernoulli estimado con 100 simulaciones.

En la figura 1a se observa el remordimiento acumulado a lo largo de $n = 50$ rondas. Como cabía esperar, la estrategia de elegir aleatoriamente entre los dos brazos obtenidos (es decir, no tener en absoluto en cuenta la información que se obtiene de las máquinas a partir de las recompensas que proporcionan) da los resultados peores. Se observa que realizar una única exploración (probar una sola vez cada una de las máquinas) dista mucho de ser suficiente para determinar cuál es la mejor: incluso con un número de rondas tan reducido como este se observa un remordimiento acumulado muy elevado al final. El resto de estrategias funcionan bastante bien en este caso. Realizar 4 exploraciones por brazo no determina en todos los casos la mejor acción, pero sí en los suficientes para que el crecimiento del remordimiento sea lento. Realizar 8 o 12 exploraciones por brazo permite determinar, casi con toda probabilidad, el brazo correcto, como se observa en que el remordimiento apenas crece después de la fase de exploración. Sin embargo, lo pagan con tener que dedicar un número de rondas mayor a la exploración y durante esta fase el remordimiento crece muy rápidamente. Para este número reducido de rondas, la opción que consigue un menor remordimiento es la de realizar unas 8 exploraciones por brazo al principio. Si se realizan menos, hay numerosas ocasiones en que no se localiza el brazo correcto y por tanto el remordimiento aumenta más rápido. Si se realizan más, el remordimiento se incrementa algo más lentamente después de la fase de exploración pero no lo suficiente para que esto le permita compensar el coste de oportunidad de todo el tiempo destinado a explorar.

Nos gustaría que un número fijo de exploraciones por brazo, por ejemplo 8, bastara para tener una cierta seguridad de cuál es la acción que da mejores resultados, aunque fuera solamente para problemas similares al

propuesto. Sin embargo, nada más lejos de la realidad. Basta con incrementar el número n de fichas iniciales en el problema anterior para que 8 exploraciones por brazo dejen de ser suficientes. En la figura 1b se tiene el remordimiento acumulado a lo largo de $n = 1000$ rondas. Se ha omitido la estrategia aleatoria y la de una única exploración porque daban un remordimiento muy elevado.

En este caso se observa que el mínimo remordimiento al final de las n rondas no se obtiene con 8 exploraciones por brazo sino con 16. Esto es porque, al disponer de muchas más rondas, merece mucho más la pena dedicar algunas más al inicio a explorar para encontrar la acción óptima. Por tanto, se pone de manifiesto que ni en los problemas de bandidos más sencillos el número de rondas que conviene destinar a la exploración es fijo, sino que siempre va a depender del número total de rondas (en concreto, será una función creciente de este).

4.3. El modelo matemático de los bandidos

En esta sección se describe el modelo matemático general de los algoritmos de bandidos y se distinguen los principales casos particulares. Recapitulemos, formalizando un poco más, algunos de los conceptos que hemos presentado intuitivamente en 4.1.

- Un problema de bandidos es un juego secuencial entre un **agente** y un **entorno**. El juego transcurre en n rondas, donde n es un número natural positivo (quizás infinito) llamado **horizonte**. El horizonte n de un problema de bandidos puede ser conocido o desconocido de antemano por el agente.
- En cada ronda $t \in \{1, \dots, n\}$, el agente elije una acción A_t de un conjunto fijo $\mathcal{A}$ y, a continuación, el entorno revela una recompensa $X_t \in \mathbb{R}$. Tanto A_t como X_t son, en general, variables aleatorias que dependen de todas las acciones y recompensas anteriores. Hablaremos luego sobre la forma en que se genera esta recompensa.
- En muchas ocasiones, las recompensas generadas tienen una cota superior e inferior conocida por el agente. Cuando se dispone de esta información, digamos $X_t \in [a_t, b_t]$ para todo $t \in \{1, \dots, n\}$, puede interesarnos trabajar con las recompensas normalizadas:

$$X_t^* := \frac{X_t - a_t}{b_t - a_t} \in [0, 1]\,.$$

- El objetivo del agente será maximizar la recompensa acumulada $\sum_{t=1}^{n} X_t$ al final de las n rondas.
- Las acciones $a \in \mathcal{A}$ posibles en cada ronda se llaman habitualmente **brazos** del bandido. Salvo que se diga lo contrario, el conjunto $\mathcal{A}$ será finito. El número de brazos del bandido que estamos tratando se denotará por k.
- Una **política** π es un proceso estocástico que determina la acción a realizar por un agente en cada ronda t en función de las acciones realizadas y recompensas obtenidas en las rondas anteriores:

$$A_t = \pi(A_1, X_1, \dots, A_{t-1}, X_{t-1})\,.$$

  Decimos que es un proceso estocástico porque, aun fijados los valores concretos de A_1, X_1, …, A_{t-1} y X_{t-1}, el valor de A_t puede no quedar perfectamente determinado si la política establece que se tome una decisión aleatoria. Si los valores de las acciones y recompensas determinan (con probabilidad 1) la siguiente acción, diremos que la política π es determinista.
- Un **entorno** ε es un proceso estocástico que determina la recompensa a recibir por un agente en cada ronda t en función de las acciones realizadas y recompensas obtenidas en las rondas anteriores y de la acción realizada en la ronda t:

$$X_t = \varepsilon(A_1, X_1, \dots, A_{t-1}, X_{t-1}, A_t)\,.$$

  Igual que en el caso de la política, decimos que es un proceso estocástico porque el entorno ε no tiene por qué ser determinista, es decir, fijados los valores concretos de A_1, X_1, …, A_{t-1}, X_{t-1} y A_t el valor

de X_t puede no quedar perfectamente determinado. Incluso aunque la política sea determinista, si el entorno no lo es A_t será una variable aleatoria al depender de los valores de las recompensas anteriores, que son estocásticas. De la misma manera, aunque el entorno ε sea determinista, si la política no lo es X_t será una variable aleatoria ya que dependerá de los valores de las acciones anteriores, que son estocásticas.

El reto fundamental en los problemas de bandidos es que el entorno es desconocido para el agente. Todo lo que el agente sabe sobre el entorno es que pertenece a una cierta familia $\mathcal{E}$ llamada **clase del entorno**.

- El **remordimiento** (o remordimiento esperado) de un agente relativo a una política π es la diferencia entre la recompensa total esperada usando la política π y la recompensa total esperada usando la política del agente. El remordimiento del agente relativo a una clase competidora Π, que es un conjunto de políticas, es el máximo de los remordimientos relativos a cualquier política $\pi \in \Pi$.

  Cuando hablamos de recompensa esperada nos referimos a la esperanza matemática de la variable aleatoria recompensa total:
  $$\sum_{t=1}^{n} X_t \,.$$
  La anterior definición del remordimiento es genérica y se concreta de formas distintas en función del modelo de recompensas que se asuma para el bandido tratado. Algunos modelos de recompensas se introducen en la sección 4.6. En concreto, el que se trata principalmente en este trabajo es el de los bandidos estocásticos y una definición del remordimiento más concreta y acorde con este modelo se da en 6.3.

  Minimizar el remordimiento es maximizar la recompensa acumulada y, por tanto, equivale al objetivo final del agente. De hecho, es habitual en el estudio de los bandidos utilizar el remordimiento en lugar de la recompensa obtenida para medir cómo de buena ha sido una determinada ejecución. Esto es debido a que el remordimiento proporciona la comparativa de la estrategia empleada con la mejor estrategia de entre aquellas con las que nos interesa compararnos, mostrando cuánto margen de mejora existe. En cambio, por sí misma la recompensa obtenida no nos da la información de si se habrían podido obtener resultados mejores.

  El remordimiento se denota por R_n para destacar su dependencia del número total de rondas n ejecutadas. Visto como función de n, el remordimiento es una función creciente, pues cada nueva ronda aporta una cantidad no negativa (máxima recompensa posible menos recompensa obtenida) a la suma total.

4.4. Los algoritmos de bandidos

Recordemos que un **algoritmo de bandidos** es un conjunto de pasos o instrucciones preestipulados seguidos por el agente durante el juego secuencial para alcanzar su objetivo de maximizar la recompensa total acumulada.

En las siguientes secciones propondremos y explicaremos algunos algoritmos de bandidos que proporcionan buenos resultados en la resolución de problemas de este tipo. Seguirán un esquema similar al del algoritmo 1. La sección *parameters* listará los parámetros de entrada del algoritmo (por ejemplo, en el caso del algoritmo utilizado en 4.2 habría un único parámetro de entrada que sería el número m de exploraciones iniciales a realizar). La sección *initialize* mostrará el valor inicial de las variables locales del algoritmo. Finalmente, en el bucle *for* se encontrarán las instrucciones que expresan las acciones realizadas por el agente en cada una de las rondas.

Cada iteración del bucle se corresponderá con una ronda del juego secuencial, de manera que en cada una de ellas deberá aparecer una y solo una instrucción de la forma

$$x \leftarrow \text{ejecutar}(a),$$

Algoritmo 1 Esquema algoritmo de bandidos

parameters

parámetro1

parámetro2

...

end parameters

initialize

var1←valor1

var2←valor2

...

end initialize

for t in $1, \ldots, n$ **do**

Acciones del algoritmo en la ronda t

end for

que representa que el agente selecciona el brazo $a \in \mathcal{A}$ y obtiene una recompensa cuyo valor almacena en la variable x. Además de esta sentencia, el cuerpo del bucle contendrá las computaciones necesarias para que el agente determine cuál es el brazo que más le conviene elegir según la información de la que dispone. Todas estas instrucciones podrán emplear el valor de las variables locales y de los parámetros de entrada del algoritmo, pero ni el horizonte n ni ninguna información sobre la instancia del problema a la que se enfrenta el agente.

Los valores de los parámetros de entrada del algoritmo varían de una ejecución a otra, pero deben fijarse al comienzo de la ejecución con la información de que se dispone en ese momento. El problema radica en que, como se ha visto en 4.2, una elección de los parámetros de entrada que no tenga en cuenta el valor n conducirá, en ocasiones, a un mal rendimiento. En mucho casos no podemos, pues, permitirnos elegir el valor inicial de forma independiente de n.

Un algoritmo de bandidos se dice **atemporal** (*anytime* en inglés) si se pueden elegir adecuadamente los valores de todos sus parámetros antes de la ejecución sin conocer el número n de rondas. En caso contrario, se dice **no atemporal**. Por ejemplo, el algoritmo Explora-Primero, cuyas intuiciones esbozamos en 4.2 y que veremos con más detalle en la sección 7, es no atemporal.

Si el algoritmo que buscamos aplicar sobre un problema es atemporal, no habrá problema a la hora de elegir los valores iniciales de los parámetros (de hecho, habitualmente esta clase de algoritmos carece de parámetros). Sin embargo, si lo que buscamos es aplicar un algoritmo no atemporal necesitaremos o bien conocer a priori el valor n, lo cual ocurrirá solo en algunas ocasiones, o bien buscar el modo de sortear esta carencia. En concreto, existe una técnica denominada **truco de la duplicación** que permite transformar cualquier algoritmo no atemporal en uno atemporal. Este mecanismo se estudia en el apéndice A de este trabajo.

Adicionalmente al de n, el conocimiento de cierta información de la instancia tratada (por ejemplo, los saltos de suboptimalidad, que veremos cuando tratemos los bandidos estocásticos) también permitiría mejorar, y mucho, el valor dado a los parámetros inicialmente. En general, normalmente se tendrá un cierto conocimiento sobre el bandido en cuestión (por ejemplo, el modelo de recompensas), ya que sin él es casi imposible dar ninguna cota significativa sobre el remordimiento. Sin embargo hay otro tipo de información sobre la instancia que es artificial suponer que el agente pueda conocer a priori, como es el caso de los saltos de suboptimalidad. En la práctica, nos conformamos con una elección "subóptima" de los valores de

los parámetros (en el sentido de que si se dispusiera de toda la información se podrían dar valores más apropiados) a cambio de que sea aplicable en un caso real.

> Veremos en cada caso concreto si el conocimiento de la instancia enfrentada que se requiere es asumible o no en un caso real.

4.5. Aplicaciones de los algoritmos de bandidos

Aunque la motivación original de Thompson [1] para estudiar los problemas de bandidos era su aplicación a las pruebas clínicas, lo cierto es que las nuevas tecnologías han creado nichos mucho más adecuados para ellos. Los problemas de bandidos hoy en día juegan un papel importante en distintas industrias.

Los servicios en línea son objetivos naturales de los algoritmos bandidos: procesan secuencialmente una cantidad ingente de peticiones individuales, por lo que la ganancia de aprender de cada una de ellas e ir adaptando sus decisiones puede ser muy grande. Veamos algunos ejemplos de este tipo de aplicaciones:

- **A/B testing.** El A/B testing es utilizado para decidir entre dos o más posibles diseños de una página web enseñando cada diseño a una parte de los usuarios y viendo cuál da, en general, mejores resultados (por ejemplo, cuál maximiza el tiempo que los usuarios pasan en la web).

  Tradicionalmente, todos los diseños se enseñaban al mismo número de usuarios, fijado de antemano. Sin embargo, a priori no se dispone de información suficiente para elegir adecuadamente este número: si se elige demasiado bajo el testing no habrá servido para nada, mientras que si se elige demasiado alto se mostrará un diseño subóptimo a un número excesivo de usuarios. Además, tampoco parece lo más conveniente mostrar cada diseño al mismo número de usuarios si desde el principio los diseños reportan resultados diferentes: deberíamos centrar nuestros esfuerzos en diferenciar entre sí aquellos diseños que vayan demostrando ser mejores.

  Por estas razones, el A/B testing es una de las aplicaciones más claras de los algoritmos de bandidos. Tomando como conjunto $\mathcal{A}$ de brazos el conjunto de diseños disponibles, cada acceso de un usuario a la web constituye una ronda en la que el agente decide cuál de los posibles diseños mostrar al usuario y mide los resultados producidos.

- **Posicionamiento de anuncios.** Las páginas web disponen de conjuntos de anuncios para mostrar a los usuarios de la misma, pero solo pueden mostrar a cada uno un número limitado. Para simplificar, supongamos que solo pueden mostrar un anuncio a cada usuario. Uno de sus objetivos básicos es seleccionar qué anuncio mostrar a cada usuario (conforme van llegando) con el fin de maximizar el número de usuarios que hacen clic en el anuncio mostrado. Este problema puede modelizarse como un problema de bandidos en el que el conjunto $\mathcal{A}$ de acciones es el conjunto de anuncios disponibles y cada ronda consiste en mostrar un anuncio a un usuario que acaba de acceder y comprobar si hace clic en él o no.

- **Sistemas recomendadores.** De forma similar al ejemplo anterior, servicios como Netflix cuentan con largas listas de series o películas que pueden mostrar a sus usuarios y solo pueden mostrar a cada uno unas pocas de ellas. Puede modelizarse de forma similar al posicionamiento de anuncios. La recompensa, en este caso, dependerá, por ejemplo, de si el usuario ve la película recomendada o de qué puntuación le da tras haberla visto.

- **Búsquedas web.** Cuando un usuario introduce una consulta en un motor de búsqueda, este debe generar a partir de ella una lista de resultados que maximice la probabilidad de que alguno de ellos se ajuste a lo esperado por el usuario.

  Lo más habitual es comparar los documentos con la consulta (utilizando, por ejemplo, TF-IDF y la similitud del coseno) de forma independiente unos de otros. Esto conlleva que documentos similares recibirán puntuaciones de similitud parecidas, con lo que hay una probabilidad alta de que la lista que finalmente se muestra al usuario contenga muchas entradas parecidas.

Sin embargo, estudios empíricos demuestran que a menudo es preferible una lista diversa que una que contenga resultados redundantes. Por ejemplo, si se buscan palabras como "jaguar" o "bandido", que presentan varios significados muy dispares, será mucho más probable que el usuario encuentre la entrada que busca si la lista de resultados es diversa. Un algoritmo que emplea bandidos para conseguir esto se encuentra en [10].

Otro campo en el que pueden utilizarse bandidos es el problema del **encaminamiento en redes** [11]. En cada ronda, el agente recibe el origen y destino de un paquete a enviar y elige un camino en la red entre todos los que tienen ese origen y destino. En general, lo que se busca será minimizar la suma de los tiempos que tardan todos los paquetes en llegar a su destino.

Finalmente, otra aplicación de los algoritmos de bandidos se da en la **inteligencia artificial en videojuegos**. Los algoritmos de bandidos pueden utilizarse para explorar más eficientemente el árbol de posibles continuaciones centrándose en los subárboles más prometedores. Esta idea ha sido implementada con éxito en el diseño de un sistema que juega al go al nivel de los mejores jugadores del mundo [12].

4.6. El modelo de recompensas: bandidos estocásticos frente a antagonistas

Hasta ahora no hemos hecho ninguna asunción sobre la forma en que el entorno genera la recompensa X_t en función de las acciones escogidas y de las recompensas anteriores. Sin embargo, en muchas de las aplicaciones reales de los bandidos tiene sentido asumir ciertas condiciones que nos facilitarán el diseño de algoritmos apropiados para la resolución de los problemas

En nuestro ejemplo de las máquinas tragaperras parece razonable pensar que las recompensas proporcionadas por cada una de las máquinas seguirán una cierta distribución de probabilidad y se generarán con independencia de qué máquinas se hayan elegido o qué recompensas se hayan generado en otras rondas. Por examinar un caso más real, cuando los bandidos se utilizan para realizar A/B testing, que el usuario haga clic en un enlace o compre un producto está influido por el diseño de la web que le estemos mostrando, pero no por los diseños que hayamos mostrado en ocasiones anteriores o las decisiones de otros usuarios.

Los casos anteriores ponen de manifiesto que, en ocasiones, conviene concretar nuestro modelo de los bandidos imponiendo que las recompensas generadas por cada brazo sean independientes e idénticamente distribuidas (IID). Es decir, que cada vez que se seleccione un brazo la recompensa se extraiga de forma independiente de una distribución fija asociada al brazo y que no depende de la ronda. Los bandidos con este modelo de recompensas se denominan **bandidos estocásticos estacionarios** (*stochastic stationary bandits* en inglés). Salvo que se diga lo contrario, cuando hablemos de bandidos estocásticos nos referiremos a bandidos estocásticos estacionarios. Este tipo de bandidos son el tema principal de este trabajo y los analizaremos con detalle en las secciones siguientes.

En contrapartida, los bandidos en los que se no se puede hacer ninguna asunción sobre la forma en que se generan las recompensas, salvo que estas están acotadas, se llaman **bandidos antagonistas** (*adversarial bandits* en inglés). Más información sobre este tipo de bandidos puede consultarse en los capítulos 11 y 12 de [2].

Entre ambos extremos se encuentran los bandidos estocásticos no estacionarios, que son aquellos en que las recompensas se extraen de forma independiente de una distribución asociada al brazo seleccionado, pero esta distribución varía en función de la ronda. Estos bandidos pueden tratarse con las técnicas de bandidos antagonistas pero si se conoce, por ejemplo, que el número de veces que la distribución de los brazos cambia es relativamente pequeño, compensa utilizar otras estrategias más cercanas a las de los bandidos estocásticos. Más información sobre este tipo de bandidos puede consultarse en el capítulo 31 de [2].

4.7. Otros aspectos a considerar

4.7.1. Contexto

En muchos problemas de bandidos, en cada decisión el agente tiene acceso a información adicional, llamada **contexto**, que puede ayudar a predecir la calidad de las acciones. Por ejemplo, en el caso de un sistema recomendador, antes de decidir qué película recomendar el agente observa cierta información sobre el usuario que ha entrado en la página (y al que va dirigida la recomendación). No suele ser conveniente diseñar un sistema recomendador que solo tenga en cuenta las recomendaciones acertadas y fallidas previas en general, sino que tendrá que valorar especialmente aquellas con un contexto similar al actual.

Los bandidos que tienen en cuenta el contexto a la hora de seleccionar cada brazo se denominan bandidos contextuales, pero no los trataremos en este trabajo. Más información sobre ellos puede consultarse en el capítulo 18 de [2].

4.7.2. Retroalimentación auxiliar

En todo este trabajo se asume que en cada ronda, tras seleccionar un brazo, el agente observa exclusivamente la recompensa recibida, es decir, la asociada al brazo escogido. No se percata, pues, de la recompensa que habría obtenido si hubiera elegido cualquiera de los otros brazos. Esto es lo que se conoce como retroalimentación de bandido (*bandit feedback* en inglés).

Existen aplicaciones prácticas que plantean problemas similares a los de bandidos pero en los que se observa la recompensa asociada a varios de los brazos (aparte del escogido) o, incluso, a todos ellos. Considérese, por ejemplo, el problema de plantear un precio personalizado a cada cliente que desea comprar un producto. La recompensa será el precio al que se haya vendido si el cliente elige comprar y 0 en caso contrario. Si modelizamos el problema como uno de bandidos observamos que en cada ronda el agente obtiene más información que la recompensa asociada a la acción escogida. Supongamos que se ofrece un producto por un precio $x > 0$:

- Si la recompensa es x, es decir, el cliente ha comprado el producto, se puede asumir que también lo habría comprado por cualquier otro precio inferior a x. Esto nos revela la recompensa para todas las acciones consistentes en ofrecer un precio menor o igual que x. No se tiene información sobre la recompensa que se habría obtenido al ofrecer un precio mayor que x.
- Si la recompensa es 0, es decir, el cliente no ha comprado el producto, se puede asumir que tampoco lo habría comprado por cualquier otro precio superior a x. Esto nos revela la recompensa para todas las acciones consistentes en ofrecer un precio mayor o igual que x. No se tiene información sobre la recompensa que se habría obtenido al ofrecer un precio menor que x.

Este tipo de realimentación se denomina retroalimentación parcial (*partial feedback* en inglés). Por otra parte, aquella en la que el agente observa tras cada ronda la recompensa asociada a todos los brazos se denomina retroalimentación total (*full feedback* en inglés).

Hay autores que consideran que los tres tipos de retroalimentación dan lugar a problemas de bandidos, mientras que otros llaman problemas de bandidos solo a los que tienen retroalimentación de bandido. Para este trabajo nos hemos decantado por esta segunda opción, en línea con [2], por lo que en lo que sigue en todos los bandidos que introduzcamos el agente percibirá al acabar la ronda únicamente la recompensa asociada al brazo escogido.

5. Fundamentos matemáticos

El estudio de los bandidos requiere la introducción de ciertos fundamentos matemáticos que nos permitan estimar razonablemente las recompensas que se obtendrán con cada uno de los brazos. En concreto, si una recompensa se extrae de una distribución con media μ y desviación típica σ nos gustaría conocer cuál es la probabilidad de que la recompensa extraída caiga en un intervalo de la forma $(\mu - \sigma k, \mu + \sigma k)$ para cierto $k \in \mathbb{R}$. En particular, los bandidos estocásticos estacionarios plantean la cuestión, típica en estadística, de cómo estimar la media de una distribución a partir de muestras independientes extraídas de la misma.

En esta sección se introducen los fundamentos matemáticos requeridos para el estudio de los bandidos estocásticos que realizaremos en las secciones siguientes. Es necesario contar lo aquí expuesto para poder entender algunos de los planteamientos de las secciones posteriores pero, dado que la temática de esta sección está bastante alejado de los objetivos del TFG, se ha optado por reducirla lo más posible, omitiendo todas las demostraciones y gran parte de las intuiciones de la misma. Para algunos de los resultados, los más elementales y aquellos que menos tienen que ver con los algoritmos de bandidos, se proporciona una referencia a la bibliografía en la que se puede encontrar una demostración de los mismos. El resto de demostraciones, junto con algunas de las intuiciones aquí omitidas, se incluirán en mi TFG de Matemáticas.

5.1. Conceptos básicos de probabilidad y estadística

Espacio de probabilidad

Todos los desarrollos matemáticos que realicemos a lo largo del TFG se apoyarán, en última instancia, en un **espacio de probabilidad** $(\Omega, \mathcal{F}, P)$, donde Ω será un espacio muestral, $\mathcal{F}$ será una σ-álgebra de subconjuntos de Ω y P una medida de probabilidad sobre $\mathcal{F}$. No obstante, no trabajaremos directamente con él, sino que siempre mediará entre nosotros y el espacio muestral alguna variable aleatoria que nos permita abstraer la información estocástica que nos interese en cada momento. Esto nos ahorrará, en general, tener que retrotraernos al espacio muestral Ω o la σ-álgebra $\mathcal{F}$. La medida de probabilidad P sí la mencionaremos asiduamente, pero siempre acompañada de variables aleatorias para hacer referencia en realidad a la medida de probabilidad inducida por estas. En concreto, dado un predicado p y variables aleatorias $U, V, \dots$, se escribirá:

$$P(p(U, V, \dots)) := P(\{\omega \in \Omega : p(U(\omega), V(\omega), \dots) \text{ es cierto}\}).$$

Variables aleatorias unidimensionales

Una variable aleatoria es una función $X : \Omega \to \mathbb{R}$ tal que $X^{-1}(B) \in \mathcal{F}$ para todo conjunto B perteneciente a la σ-álgebra de Borel $\mathcal{B}(\mathbb{R})$. Toda variable aleatoria induce una medida de probabilidad en la σ-álgebra de Borel dada por:

$$P(X \in A) := P_X(A) := P(X^{-1}(A))$$

para todo $A \in \mathcal{B}(\mathbb{R})$. Esta medida de probabilidad se conoce como **distribución** de X.

Un ejemplo sencillo de variable aleatoria es la función indicador asociada a un conjunto. Dado $A \in \mathcal{F}$ se define la **función indicador** de A como $\mathbb{I}_A : \Omega \to \{0, 1\}$ tal que

$$\mathbb{I}_A(\omega) = \begin{cases} 1\,, & \text{si } \omega \in A \\ 0\,, & \text{en caso contrario} \end{cases}$$

Igual que con la medida de probabilidad, dado un predicado p y variables aleatorias $U, V, \dots$ se escribirá:

$$\mathbb{I}\{p(U, V, \dots)\} := \mathbb{I}_A$$

donde $A := \{\omega \in \Omega : p(U(\omega), V(\omega), \dots) \text{ es cierto}\})$.

Dado un conjunto $A \in \mathcal{B}(\mathbb{R})$ se dice que $X \in A$ **casi seguro** (abreviado c.s.) si $P_X(A) = 1$, de manera que no puede existir ningún conjunto $B \in \mathcal{B}(\mathbb{R})$ disjunto de A con $P_X(B) \neq 0$. Se denomina **soporte** de

X al menor conjunto $\text{sop}(X) \in \mathcal{B}(\mathbb{R})$ tal que $X \in \text{sop}(X)$ casi seguro. Las variables aleatorias **de soporte acotado** son aquellas cuyo soporte es un conjunto acotado en $\mathcal{B}(\mathbb{R})$. Por ejemplo, toda función indicador tiene soporte $\{0, 1\}$ y, por ello, es una variable aleatoria de soporte acotado.

La distribución de una variable aleatoria X siempre puede describirse mediante una función de distribución $F_X : \mathbb{R} \to [0, 1]$ dada por:

$$F_X(x) = P(X \leq x)\,.$$

Aquí trabajaremos únicamente con variables aleatorias discretas y absolutamente continuas, definidas como sigue:

- Una variable aleatoria X se dice **discreta** si su función de distribución es escalonada, es decir, se puede definir como una función a trozos que es constante en cada uno de los trozos. Esto es equivalente a que su soporte $\text{sop}(X)$ sea un conjunto discreto de puntos, que son los puntos de salto de la función de distribución. La distribución de una variable aleatoria discreta X habitualmente se caracteriza mediante su función de masa $p_X(x) = P(X = x)$
- Una variable aleatoria X se dice **absolutamente continua** si existe una función $f_X : \mathbb{R} \to \mathbb{R}$, denominada función de densidad de X, tal que:

  $$F_X(x) = \int_{-\infty}^{x} f_X(t)\ dt \quad \forall x \in \mathbb{R}\,.$$

  La función de densidad f_X es la forma habitual de caracterizar la distribución de una variable aleatoria absolutamente continua. Dado que no trataremos otro tipo de variables aleatorias continuas en este trabajo, podemos hacer abuso del lenguaje y llamar, en lo que sigue, variables aleatorias continuas a este tipo de variables.

Para el análisis matemático de los bandidos, lo más importante del estudio de una variable aleatoria será la estimación de su media. Recordemos que la **esperanza** o media de una variable aleatoria X se define como la integral de Lebesgue

$$E[X] = \int_{\Omega} X\ dP\,,$$

que, por el teorema del cambio de espacio de integración, se escribe como

$$E[X] = \int_{\mathbb{R}} X\ dP_X\,.$$

En los casos que trataremos la integral anterior toma la siguiente forma, que nos permite computarla:

- Si X es discreta:

  $$E[X] = \sum_{x \in \text{sop}(X)} xP(X = x)\,.$$

- Si X es continua:

  $$E[X] = \int_{-\infty}^{+\infty} x f_X(x)\ dx\,.$$

Una de las propiedades principales de la esperanza es la linealidad:

Teorema 5.1.1. *Dadas variables aleatorias X, Y y $c \in \mathbb{R}$:*

$$E[X + Y] = E[X] + E[Y]\,,$$

$$E[cX] = cE[X]\,.$$

Demostración. Teorema 2.2.5 de [5]. □

Aunque no es difícil dar con variables aleatorias cuya esperanza no está bien definida o es infinita, estas distribuciones rara vez aparecen en las aplicaciones. Todas las variables que tratemos en este trabajo tendrán esperanza bien definida y finita.

Además de estimar la esperanza de una variable aleatoria, nos será útil conocer cuánto se concentran sus valores en torno a este valor medio. Para ello es primario contar con medidas de la dispersión de X tales como su **varianza**:

$$V[X] := E[(X - E[X])^2] = E[X^2] - E[X]^2 .$$

y su **desviación típica**:

$$SD[X] = \sqrt{V[X]} .$$

Más en general, para cada $k \in \mathbb{N}$ denominamos **momento respecto del origen de orden** k de X a $E[X^k]$ y **momento centrado de orden** k de X a $E[(X - E[X])^k]$. Los momentos centrados pueden calcularse respecto a los momentos respecto del origen mediante:

$$E[(X - E[X])^n] = \sum_{i=0}^{n} \binom{n}{i} (-1)^{n-i} E[X^i] E[X]^{n-i} .$$

Para ciertos aspectos técnicos nos convendrá caracterizar la distribución de una variable aleatoria X de forma alternativa mediante la **función generatriz de momentos** $M_X : \mathbb{R} \to \mathbb{R}^+$ definida como

$$M_X(t) = E[\exp(tX)] .$$

Cuando $M_X(t)$ es finita para todo $t \in \mathbb{R}$, se cumple que los momentos de X de cualquier orden son todos finitos (una demostración de este hecho puede consultarse en [13]). En este caso, los momentos centrados pueden calcularse a partir de las derivadas de M_X como

$$E[X^k] = M_X^{(k}(0) .$$

para cualquier $k \in \mathbb{N}$.

La esperanza de una variable aleatoria (y, por extensión, sus momentos y su función generatriz) pueden no existir sin más que la suma/integral correspondiente diverja. No haremos mucho énfasis en esto porque se aleja demasiado del objetivo del trabajo y, en general, trabajaremos con distribuciones usuales cuya esperanza es conocida. No obstante, dado que trabajaremos principalmente con variables aleatorias de soporte acotado, es conveniente introducir el siguiente resultado:

Teorema 5.1.2. *Sean $a, b \in \mathbb{R}, a < b$, sea X una variable aleatoria de soporte acotado y sea $f : \mathbb{R} \to \mathbb{R}$ una función continua. Entonces, $E[f(X)]$ está bien definida y es finita. En particular, están bien definidos y son finitos todos sus momentos respecto del origen y centrados.*

Demostración. Combinar [14] con el hecho de que toda función continua en un conjunto acotado es acotada. □

Además, en el caso de variables de soporte acotado también se cuenta con una cota para la varianza que nos será útil en algunos contextos:

Teorema 5.1.3 (Desigualdad de Popoviciu)**.** *Sean $a, b \in \mathbb{R}, a < b$, y sea X una variable aleatoria tal que $X \in [a, b]$ casi seguro. Entonces:*

$$V[X] \leq \frac{(b-a)^2}{4} .$$

Demostración. [15]. □

Distribuciones de probabilidad

En las siguientes tablas se recogen las distribuciones de probabilidad que se usarán en los ejemplos a lo largo del trabajo, junto con sus principales propiedades.

Distribuciones discretas

Nombre	Símbolo	Función de masa	Media	Varianza
Bernoulli	Bernoulli(p)	$\begin{cases} P(X=1)=p \\ P(X=0)=1-p \end{cases}$	p	$p(1-p)$
Binomial	$B(n,p)$	$P(X=k)=\binom{n}{k}p^k(1-p)^{n-k}$	np	$np(1-p)$
Poisson	Pois(λ)	$P(X=k)=\dfrac{\lambda^k e^{-\lambda}}{k!}$	λ	λ

Distribuciones continuas

Nombre	Símbolo	Función de densidad	Media	Varianza
Normal o gaussiana	$N(\mu,\sigma)$	$f_X(x)=\dfrac{1}{\sigma\sqrt{2\pi}}\exp\left(-\dfrac{1}{2}\left(\dfrac{x-\mu}{\sigma}\right)^2\right)$	μ	σ^2
Exponencial	Exp(λ)	$f_X(x)=\lambda e^{-\lambda x}$	$\dfrac{1}{\lambda}$	$\dfrac{1}{\lambda^2}$
Uniforme	U(a,b)	$f_X(x)=\begin{cases} f_X(x)=\frac{1}{b-a} & \text{si } x\in[a,b] \\ f_X(x)=0 & \text{en caso contrario} \end{cases}$	$\dfrac{1}{2}(a+b)$	$\dfrac{1}{12}(b-a)^2$

Distribuciones conjuntas

Gran parte de las definiciones vistas para variables unidimensionales se generalizan de forma directa al caso en que $X=(X_1,\ldots,X_n)$ es una variable aleatoria multidimensional. Se denomina **distribución conjunta** de $X=(X_1,\ldots,X_n)$ a la medida de probabilidad $P_{X_1\ldots X_n}$ sobre la σ-álgebra de Borel n-dimensional $\mathcal{B}(\mathbb{R}^n)$ dada por:

$$P((X_1,\ldots,X_n)\in A):=P_{X_1\ldots X_n}(A):=P(\omega\in\Omega:(X_1(\omega),\ldots,X_n(\omega))\in A)$$

para todo $A\in\mathcal{B}(\mathbb{R}^n)$. Equivalentemente, dados $A_1,\ldots,A_n\in\mathcal{B}(\mathbb{R})$ se escribe:

$$P(X_1\in A_1,\ldots,X_n\in A_n):=P_{X_1\ldots X_n}(A_1\times\cdots\times A_n):=P(\omega\in\Omega:X_1(\omega)\in A_1,\ldots,X_n(\omega)\in A_n)$$

El soporte de $X=(X_1,\ldots,X_n)$ vuelve a ser menor conjunto $\operatorname{sop}(X)$ tal que $X\in\operatorname{sop}(X)$ casi seguro, es decir,

$$\operatorname{sop}(X):=\operatorname{sop}(X_1,\ldots,X_n):=\operatorname{sop}(X_1)\times\operatorname{sop}(X_n)\,.$$

La distribución conjunta de $(X_1,\ldots,X_n)$ siempre puede describirse mediante una función de distribución $F_{X_1\ldots X_n}:\mathbb{R}^n\to[0,1]$ dada por:

$$F_{X_1\ldots X_n}(x_1,\ldots,x_n)=P(X_1\leq x_1,\ldots,X_n\leq x_n)\,.$$

Si bien no suele hablarse de la esperanza de un variable multidimensional, sí es habitual requerir el cálculo de la esperanza de una función que depende de múltiples variables aleatorias. Dada una función $g:\mathbb{R}^n\to\mathbb{R}$, se define la esperanza de $g(X_1,\ldots,X_n)$ como

$$E[g(X_1,\ldots,X_n)]:=\int_\Omega g(X_1,\ldots,X_n)\,dP=\int_{\mathbb{R}^n}g(X_1,\ldots,X_n)\,dP_{X_1\ldots X_n}\,.$$

En el caso particular en que $X_1,\ldots,X_n$ son todas discretas o todas absolutamente continuas, se obtiene la generalización esperable:

- Si $X_1, \ldots, X_n$ son discretas, la distribución conjunta puede caracterizarse por medio de la función de masa:
$$p_{X_1,\ldots,X_n}(x_1, \ldots, x_n) = P(X_1 = x_1, \ldots, X_n = x_n)\,.$$
Dada una función $g : \mathbb{R}^n \to \mathbb{R}$, se tiene:
$$E[g(X_1, \ldots, X_n)] = \sum_{(x_1,\ldots,x_n)\in \mathrm{sop}(X_1,\ldots,X_n)} g(x_1, \ldots, x_n) P(X_1 = x_1, \ldots, X_n = x_n)\,.$$

- Si $X_1, \ldots, X_n$ son absolutamente continuas, la distribución conjunta puede caracterizarse por medio de la función de densidad:
$$f_{X_1,\ldots,X_n}(x_1, \ldots, x_n) = \frac{\partial^n}{\partial x_1 \ldots \partial x_n} F_{X_1\ldots X_n}(x_1, \ldots, x_n)\,.$$
Dada una función $g : \mathbb{R}^n \to \mathbb{R}$, se tiene:
$$E[g(X_1, \ldots, X_n)] = \int_{\mathbb{R}^n} g(x_1, \ldots, x_n) f_{X_1\ldots X_n}(x_1, \ldots, x_n)\, dx_1 \ldots dx_n\,.$$

Independencia de variables aleatorias

Se dirá que las variables aleatorias $X_1, \ldots, X_n$ son **independientes** si:
$$F_{X_1\ldots X_n}(x_1, \ldots, x_n) = F_{X_1}(x_1) \cdots F_{X_n}(x_n)$$
para todo $x_1, \ldots, x_n \in \mathbb{R}$, lo cual es equivalente a:

- Si $X_1, \ldots, X_n$ son discretas:
$$P(X_1 = x_1, \ldots, X_n = x_n) = P(X_1 = x_1) \cdots P(X_n = x_n)$$
para todo $x_1, \ldots, x_n \in \mathbb{R}$.
- Si $X_1, \ldots, X_n$ son continuas:
$$f_{X_1\ldots X_n}(x_1, \ldots, x_n) = f_{X_1}(x_1) \cdots f_{X_n}(x_n)$$
para todo $x_1, \ldots, x_n \in \mathbb{R}$.

Las variables aleatorias independientes verifican las siguientes propiedades:

Teorema 5.1.4. *Dadas variables aleatorias $X_1, \ldots, X_n$ independientes, se cumple:*

a) $E[X_1 \cdots X_n] = E[X_1] \cdots E[X_n]$

b) $V[X_1 + \cdots + X_n] = V[X_1] + \cdots + V[X_n]$

Demostración. a) Teorema 4.2.10 de [5].

b) Lema 5.2.5 de [5].

□

Distribuciones condicionadas

Sean X e Y variables aleatorias (posiblemente multidimensionales) y sea B un conjunto tal que $P(Y \in B) \neq 0$. Se define la **distribución** de X **condicionada** a que Y tome valor en A por

$$P_{X|Y\in B}(A) := \frac{P(X \in A, Y \in B)}{P(Y \in B)} = \frac{P_{X,Y}(A \times B)}{P_Y(B)}$$

Habiendo introducido las distribuciones condicionadas, es automática la definición de la **esperanza condicionada**. Suponiendo que X toma valores en $\mathbb{R}^k$ para cierto $k \in \mathbb{N}$, dada una función $g : \mathbb{R}^k \to \mathbb{R}$ se define la esperanza de $g(X)$ condicionada por $Y \in B$ como

$$E[g(X)|Y \in B] := \int_{\mathbb{R}^k} g(X)\, dP_{X|Y\in B}\,.$$

En el caso en que la variable Y es discreta, se tiene el siguiente resultado:

Teorema 5.1.5 (Ley de la esperanza total)**.** *Dada una variable X y una variable aleatoria discreta Y, se cumple:*

$$E[X] = \sum_{y\in sop(Y)} E[X|Y = y]P(Y = y)\,.$$

Demostración. Teorema 34.4 de [6]. □

Estimadores

Sea X una variable aleatoria de media μ y desviación típica σ. En numerosas ocasiones nos interesará ser capaces de estimar los valores de los parámetros μ y σ desconocidos a partir de muestras extraídas de la distribución de X.

Si $X_1, \ldots, X_n$ son variables aleatorias independientes e idénticamente distribuidas (i.i.d) con la distribución de X, se dice que $(X_1, \ldots, X_n)$ constituye una **muestra aleatoria simple (m.a.s)** de X. Su importancia radica en que a partir de una m.a.s de X podemos construir estimadores de μ y σ. Los estimadores que utilizaremos aquí son la media, la varianza y la desviación típica muestrales.

La **media muestral** $\hat{\mu}$ es el estimador más natural de la media μ de X y se define como:

$$\hat{\mu} := \frac{1}{n}\sum_{i=1}^{n} X_i\,.$$

Se tiene que $E[\hat{\mu}] = \mu$, de manera que $\hat{\mu}$ constituye un estimador insesgado de μ. Por otro lado, se da $V[\hat{\mu}] = \frac{\sigma^2}{n}$, lo cual nos dice que la varianza de la media muestral es proporcional a la varianza σ^2 de la distribución de X pero disminuye linealmente al aumentar el número de muestras.

La **varianza muestral** $\hat{\sigma}^2$, definida como

$$\hat{\sigma}^2 := \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \hat{\mu})^2\,,$$

es el estimador que utilizaremos para la varianza σ^2 de X. Por su parte, la desviación típica muestral

$$\hat{\sigma} := \sqrt{\hat{\sigma}^2}$$

será el estimador que usaremos para σ. El hecho de dividir por $n-1$ en lugar de n en la definición de $\hat{\sigma}^2$ se conoce como corrección de Bessel, y es gracias a ello que $E[\hat{\sigma}^2] = \sigma^2$, es decir, que $\hat{\sigma}$ es un estimador insesgado de σ^2. Sin embargo, pese a esta corrección la desviación típica muestral sigue presentando sesgo en la estimación de σ.

Las demostraciones de:

- $E[\hat{\mu}] = \mu$.
- $V[\hat{\mu}] = \frac{\sigma^2}{n}$.
- $E[\hat{\sigma}^2] = \sigma^2$.

pueden consultarse en el teorema 5.2.6 de [5].

Reproductividad de las familias de distribuciones

Es importante el siguiente resultado relativo a la distribución de la suma de variables aleatorias i.i.d.

Teorema 5.1.6. *Sean $X_1, \ldots, X_n$ variables aleatorias i.i.d. y sea $S := \sum_{i=1}^n X_i$. Se tiene:*

- *Si $X_1, \ldots, X_n$ tienen distribución* Bernoulli(p)*, entonces S tiene distribución $B(n, p)$.*
- *Si X_i tiene distribución $B(n_i, p)$ para cada $i \in \{1, \ldots, n\}$, entonces S tiene distribución $B(\sum_{i=1}^n n_i, p)$.*
- *Si X_i tiene distribución* Pois(λ_i) *para cada $i \in \{1, \ldots, n\}$, entonces S tiene distribución* Pois$(\sum_{i=1}^n \lambda_i)$.
- *Si $X_1, \ldots, X_n$ tienen distribuciín $N(\mu_i, \sigma_i)$ para cada $i \in \{1, \ldots, n\}$, entonces S tiene distribución $N(\sum_{i=1}^n \mu_i, \sqrt{\sum_{i=1}^n \sigma_i^2})$.*

Demostración. Capítulo 3 de [5]. □

5.2. Subgaussianidad

Para el estudio de los bandidos estocásticos es necesario disponer de formas de acotar la probabilidad de las colas de las distribuciones asociadas a los brazos. En concreto, buscamos acotar la probabilidad de que una variable aleatoria X dada tome valores a una distancia de su media mayor que un $\varepsilon > 0$ dado. Buscaremos condiciones sobre X para las que se dan cotas de la probabilidad de las colas de la distribución, de la forma:

$$P(X \geq E[X] + \varepsilon) \leq M \qquad\qquad P(X \leq E[X] - \varepsilon) \leq M$$

$$P(|X - E[X]| \geq \varepsilon) \leq M$$

donde M dependerá de ε y de los momentos de X, que asumiremos bien definidos. Nos interesará especialmente el caso en que X es una suma de variables aleatorias i.i.d. y querremos, en este caso, que la cota pueda calcularse a partir de la distribución de los sumandos.

Existen numerosas desigualdades de este tipo (desigualdad de Markov, desigualdad de Chebyshev,...) que se pueden aplicar sobre familias muy grandes de variables aleatorias pero que, sin embargo, no nos resultan útiles por ser demasiado poco ajustadas en el caso general. Por otro lado, la distribución normal cuenta con su propia desigualdad de concentración (la desigualdad de Mills), que es lo suficientemente ajustada para nuestros propósitos pero se puede aplicar únicamente a distribuciones normales. Una exposición más detallada sobre estas desigualdades puede encontrarse en mi TFG de Matemáticas.

En esta sección, nosotros saltaremos directamente a presentar las variables aleatorias subgaussianas, que presentan desigualdades de concentración suficientemente ajustadas (similares a la desigualdad de Mills) pero abarcan muchas más distribuciones aparte de la normal. Nos centraremos especialmente en las distribuciones de soporte acotado ya que, debido a lo "rápido" que decaen sus colas (se hacen 0 una vez nos salimos del soporte), siempre pertenecen a la familia subgaussiana. La explicación detallada de la intuición que nos lleva a definnirlas de esta manera, así como las demostraciones de los distintos resultados aquí enunciados se incluirá también en mi TFG de Matemáticas.

Definición. *Sea $\sigma \geq 0$. Una variable aleatoria X se dice* ***σ-subgaussiana*** *si verifica*

$$E[\exp(\lambda(X - E[X]))] \leq \exp\left(\frac{\lambda^2\sigma^2}{2}\right)$$

para todo $\lambda \in \mathbb{R}$. Lo escribiremos como $X \in \mathrm{subG}(\sigma)$.

Por la monotonía de las funciones implicadas, si $0 \leq \sigma_1 \leq \sigma_2$ se verifica

$$\exp\left(\frac{\lambda^2\sigma^2}{2}\right) \leq \exp\left(\frac{\lambda^2\sigma^2}{2}\right).$$

De ello se deduce que si una variable aleatoria X es σ_1-subgaussiana también es σ_2-subgaussiana para cualquier $\sigma_2 \geq \sigma_1$.

Las desigualdades de concentración para las variables aleatorias de la familia subgaussiana quedan recogidas en el siguiente teorema.

Teorema 5.2.1. *Sea $\sigma \geq 0$ y sea $X \in \mathrm{subG}(\sigma)$ de media μ. Para todo $\varepsilon > 0$ se verifica:*

$$P(X \geq \mu + \varepsilon) \leq \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right) \qquad P(X \leq \mu - \varepsilon) \leq \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)$$

$$P(|X - \mu| \geq \varepsilon) \leq 2\exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)$$

Nos interesa ahora extender el resultado a la media muestral de variables aleatorias subgaussianas independientes. No obstante, hemos de ser cautos, pues por ahora no sabemos si la media muestral de variables subgaussianas sigue siendo subgaussiana ni si el valor de σ se mantiene. En los siguientes teoremas se muestra el buen comportamiento de las distribuciones subgaussianas respecto a la suma y la multiplicación por constantes.

Teorema 5.2.2. *Sea $\sigma \geq 0$, $X \in \mathrm{subG}(\sigma)$ y $c \in \mathbb{R}$. Entonces, $cX \in \mathrm{subG}(|c|\sigma)$.*

Teorema 5.2.3. *Sean $\sigma_1, \sigma_2 \geq 0$, $X_1 \in \mathrm{subG}(\sigma_1)$ y $X_2 \in \mathrm{subG}(\sigma_2)$. Entonces:*

a) $X_1 + X_2 \in \mathrm{subG}(\sigma_1 + \sigma_2)$.

b) Si X_1 y X_2 son independientes, entonces $X_1 + X_2 \in \mathrm{subG}\left(\sqrt{\sigma_1^2 + \sigma_2^2}\right)$.

Por supuesto, el resultado anterior se generaliza de manera inmediata para la suma de cualquier número finito de variables aleatorias subgaussianas.

Ya estamos en condiciones de proporcionar desigualdades de concentración para la media muestral de variables aleatorias subgaussianas. Presentamos directamente el caso en el que todas las variables tienen el mismo parámetro σ y la misma media, pues es el que se dará en la práctica. Sean $X_1, \ldots, X_n \in \mathrm{subG}(\sigma)$. De los teoremas anteriores se obtiene que:

$$\frac{1}{n}\sum_{t=1}^{n} X_t \in \mathrm{subG}(\sigma)$$

Además, si $X_1, \ldots, X_n$ son independientes se tiene una propiedad más fuerte:

$$\frac{1}{n}\sum_{t=1}^{n} X_t \in \mathrm{subG}\left(\frac{\sigma}{\sqrt{n}}\right)$$

Aplicando el teorema 5.2.1 en ambos casos, se obtiene el siguiente resultado.

Corolario 5.2.1. *Sean $X_1, \ldots, X_n \in \mathrm{subG}(\sigma)$ variables aleatorias con la misma media μ. Se tiene:*

$$P\left(\frac{1}{n}\sum_{t=1}^{n} X_t \geq \mu + \varepsilon\right) \leq \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right) \qquad P\left(\frac{1}{n}\sum_{t=1}^{n} X_t \leq \mu - \varepsilon\right) \leq \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)$$

$$P\left(\left|\frac{1}{n}\sum_{t=1}^{n} X_t - \mu\right| \geq \varepsilon\right) \leq 2\exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)$$

Además, si $X_1, \ldots, X_n$ son independientes se da:

$$P\left(\frac{1}{n}\sum_{t=1}^{n} X_t \geq \mu + \varepsilon\right) \leq \exp\left(-\frac{n\varepsilon^2}{2\sigma^2}\right) \qquad P\left(\frac{1}{n}\sum_{t=1}^{n} X_t \leq \mu - \varepsilon\right) \leq \exp\left(-\frac{n\varepsilon^2}{2\sigma^2}\right)$$

$$P\left(\left|\frac{1}{n}\sum_{t=1}^{n} X_t - \mu\right| \geq \varepsilon\right) \leq 2\exp\left(-\frac{n\varepsilon^2}{2\sigma^2}\right)$$

Hasta aquí hemos caracterizado la familia de variables aleatorias con las que trabajaremos más habitualmente y hemos mostrado por qué es importante. Sin embargo, aún no hemos probado que ninguna de las distribuciones que habitualmente aparecen en las aplicaciones verifique esto. Si ninguna de ellas resulta ser subgaussiana, de poco nos servirá todo lo enunciado. Por suerte, sí que hay distribuciones de las que aparecen con más frecuencia, aunque hay que admitir que no tantas como nos gustaría.

En primer lugar, dado que se han definido como σ-subgaussianas precisamente aquellas variables aleatorias X que verifican

$$M_{X-E[X]}(\lambda) \leq M_{N(0,\sigma)}(\lambda)\,,$$

es claro que las variables aleatorias con distribución $N(0,\sigma)$ cumplen lo anterior con igualdad, así que son σ-subgaussianas. Generalizando un poco más, una variable aleatoria X con distribución $N(\mu,\sigma)$ verifica que $X - E[X]$ es $N(0,\sigma)$, así que X también resulta ser σ-subgaussiana. Por tanto, toda variable aleatoria normal es subgaussiana de parámetro su desviación típica.

¿Qué otras distribuciones cabe esperar que pertenezcan a la familia subgaussiana? Hemos hablado previamente de que las colas de una variable aleatoria de soporte acotado intuitivamente decaen muy rápidamente, así que estas son buenas candidatas. Nuestra conjetura resulta ser cierta, como se ve en el siguiente teorema.

Teorema 5.2.4 (Lema de Hoeffding)**.** *Sea X una variable aleatoria tal que X toma valores en el intervalo $[a,b]$. Entonces, $X \in \mathrm{subG}\left(\frac{b-a}{2}\right)$.*

Se obtiene, pues, que toda variable aleatoria de soporte acotado es σ-subgaussiana para algún $\sigma > 0$. Veamos algunos ejemplos:

- Si X es degenerada, entonces es σ-subgaussiana para cualquier $\sigma > 0$.
- Si $X \equiv U(a,b)$, entonces es $\frac{b-a}{2}$-subgaussiana.
- Si $X \equiv \mathrm{Bin}(n,p)$, entonces X es $\frac{n}{2}$-subgaussiana. En particular, si $X \equiv \mathrm{Bernoulli}(p)$, entonces X es $\frac{1}{2}$-subgaussiana.

Sin embargo, hay también muchas distribuciones usuales que no son σ-subgaussianas para ningún $\sigma > 0$. Entre ellas destacamos la distribución de Poisson y la exponencial.

6. Introducción a los bandidos estocásticos

En lo que sigue siempre que hablemos de bandidos estocásticos nos referiremos a bandidos estocásticos estacionarios.

Como ya adelantamos en el capítulo 4, un bandido estocástico es aquel en el que las recompensas generadas tras seleccionar un brazo se extraen de forma independiente de una distribución fija asociada al brazo. Formalmente, entonces, un bandido estocástico es una familia de distribuciones

$$\nu = (P_a : a \in \mathcal{A}) \,,$$

donde $\mathcal{A}$ es el conjunto de brazos o acciones disponibles. Cualquier distribución será válida para un brazo del bandido siempre que tenga esperanza finita, lo cual es necesario para que el remordimiento sea finito.

La interacción entre el agente y el entorno a lo largo de n rondas es la siguiente:

---

for t in $1, \ldots, n$ **do**
  El agente elige un brazo $A_t \in \mathcal{A}$
  El entorno extrae una recompensa X_t de la distribución P_{A_t}
  El entorno revela la recompensa al agente
end for

La característica fundamental de los bandidos estocásticos es que en cada ronda t la recompensa obtenida X_t solo depende de la acción inmediatamente anterior A_t, y no de recompensas y acciones de rondas previas:

$$X_t = \varepsilon(A_t) \,.$$

Entonces, si en una cierta ronda t el agente selecciona la acción $a \in \mathcal{A}$, es decir, la variable aleatoria A_t toma valor a, la distribución de la variable aleatoria recompensa X_t será P_a. Más formalmente:

$$P_{X_t | A_t = a} = P_a \quad \forall a \in \mathcal{A}, t \in \{1, \ldots, n\} \,.$$

Un ejemplo de bandido estocástico es el considerado en el experimento de 4.2, que constaba de dos brazos que generaban sus recompensas a partir de sendas distribuciones Bernoulli. La tarea del agente consistía en estimar la media de estas distribuciones mediante la exploración para, a continuación, poder centrarse en el brazo que ofrecía de media mejores resultados.

6.1. Preliminares

Objetivo y dificultades

Consideraremos que el objetivo del agente siempre es maximizar la recompensa acumulada a lo largo de las n rondas:

$$S_n := \sum_{t=1}^{n} X_t \,.$$

Al tratarse S_n de una variable aleatoria, cabe preguntarse cómo comparar dos distribuciones distintas de S_n. Considérense dos posibles distribuciones A y B para S_n, representadas en la figura 2. La distribución B tiene una esperanza mayor, así que por defecto elegiríamos esta. Sin embargo, hay una probabilidad razonable de que B dé una recompensa inferior a la mínima que puede dar A, así que si quisiéramos "ir sobre seguro" quizás no sería tan buena idea elegir B. Estas consideraciones, por supuesto, van más allá del estudio de los bandidos. Por ello, en este trabajo consideraremos "mejor" la distribución de S_n que tenga una esperanza mayor, obviando todas las demás propiedades de esta.

Otro punto que surge al examinar la cantidad que el agente pretende maximizar es si siempre conoceremos a priori el número de rondas n para el que queremos maximizar. Hay aplicaciones prácticas en las que, como

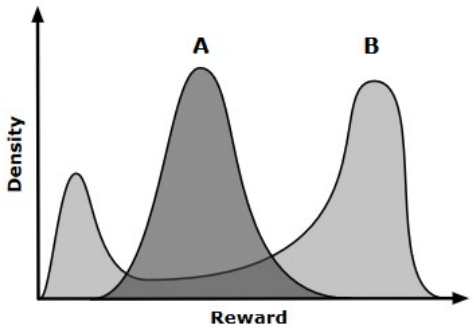


Figura 2: Dos distribuciones a comparar. Basada en la figura 4.1 de [2].

en nuestro ejemplo del casino, es habitual conocer de antemano el número de rondas que jugaremos, por ejemplo, en pruebas clínicas. Sin embargo, hay otras en las que a priori no se dispone de esta información, como en A/B testing o sistemas recomendadores. En la sección 4.2 se vio que una política concreta (en este caso, la política de explorar 8 veces cada brazo para decidir cuál es el mejor) puede funcionar bien, desde el punto de vista del remordimiento, para cierto valor n, pero mal para otro. Entre los algoritmos de bandidos habrá algunos que requieran conocer el valor de n y otros que no. Sin embargo, no nos preocupará este punto ya que el truco de la duplicación nos permite aplicar un algoritmo que utilice el valor de n a un problema en el que este es desconocido (véase apéndice A).

La dificultad principal que entraña un problema de bandidos estocásticos no es ninguna de las anteriores, sino el desconocimiento por parte del agente de las distribuciones P_a asociadas a cada uno de los brazos. Al fin y al cabo, una política que maximiza la esperanza de S_n para una cierta instancia ν del bandido puede minimizarla para otra. El agente puede tener más o menos información a priori sobre el bandido al que se enfrenta, pero nunca tendrá tanta como para no necesitar analizar las recompensas obtenidas para decidir cuál es la acción que más le conviene ejecutar (si la tuviera, no estaríamos ante un problema de bandidos).

Clase de entornos: bandidos estructurados y no estructurados

La información a priori que el agente tiene sobre el bandido se modeliza por medio de una clase de entornos $\mathcal{E}$ de la que tenemos garantía que contiene al bandido en cuestión. Habitualmente, el conjunto $\mathcal{A}$ de brazos será finito y la clase de entornos será de la forma

$$\mathcal{E} = \prod_{a \in \mathcal{A}} \mathcal{M}_a = \{\nu = (P_a : a \in \mathcal{A}) : P_a \in \mathcal{M}_a \ \forall a \in \mathcal{A}\},$$

donde, para cada $a \in \mathcal{A}$, $\mathcal{M}_a$ es una familia de distribuciones de probabilidad. Se dice, en este caso, que la clase de entornos $\mathcal{E}$ es **no estructurada**. Este tipo de clase de entornos representa un escenario en el que el agente solo conoce, para cada brazo, una familia de distribuciones que contiene a la distribución verdadera de ese brazo. No dispone, pues, de información que relacione entre sí las distribuciones de los brazos. Esto le impide, en particular, aprender nada sobre la distribución asociada a un brazo $a \in \mathcal{A}$ seleccionando otro brazo $b \in \mathcal{A} \setminus \{a\}$.

En caso de que no se verifique lo anterior, es decir, que $\mathcal{E}$ no sea un producto cartesiano finito de familias de distribuciones, diremos que $\mathcal{E}$ es una clase de entornos estructurada. Un ejemplo de este tipo de clase de entornos sería el siguiente bandido de dos brazos:

$$\mathcal{E} = \{(\text{Bernoulli}(\theta), \text{Bernoulli}(1-\theta)) : \theta \in [0,1]\}.$$

Como quizás pueda vislumbrarse en el ejemplo anterior, para aprovechar la información adicional que aporta una clase de entornos estructurada a menudo se requiere una estrategia *ad hoc*. Es por eso que los algoritmos de bandidos que veremos a lo largo del capítulo no serán demasiado útiles en estos casos. Volveremos sobre este ejemplo al final del capítulo.

Salvo que se diga lo contrario, todos los bandidos de este trabajo tendrán clase de entornos no estructurada y los algoritmos que explicaremos serán apropiados solo para este tipo de problemas.

6.2. Algunas notaciones

En los bandidos estocásticos que consideramos la distribución de cada brazo no varía a lo largo de todo el juego secuencial y, por ello, tampoco lo hace la bondad de cada brazo. En un número de rondas pequeño podría ser que tuviéramos mala suerte seleccionando el brazo óptimo pero, para n grande, las recompensas obtenidas al seleccionar sistemáticamente el brazo $a \in \mathcal{A}$ se concentrarán en torno a la esperanza de su distribución asociada P_a. Por tanto, resulta natural comparar los brazos entre sí por la media de sus distribuciones e identificar al brazo (o brazos) óptimo como aquel que maximiza esta media. Para poder tomar el máximo sin problemas requeriremos que el conjunto $\mathcal{A}$ sea finito.

Para cada brazo $a \in \mathcal{A}$ se denota con $\mu_a(\nu)$ la media de la distribución P_a de ν:

$$\mu_a(\nu) := \int_{-\infty}^{\infty} x \, dP_a \,,$$

que exigiremos siempre que sea finita para todo $a \in \mathcal{A}$. Dado que para toda ronda $t \in \{1, \dots, n\}$ se tiene $P_a = P_{X_t|A_t=a}$, se verifica que $\mu_a = E[X_t|A_t = a]$. Denotamos, asimismo, por $\mu^*(\nu)$ el máximo de las medias de las distribuciones de todos los brazos,

$$\mu^*(\nu) := \max_{a \in \mathcal{A}} \mu_a(\nu) \,,$$

y a^* al primer brazo con esta media,

$$a^* := \min\{a \in \mathcal{A} : \mu_a = \mu^*\} \,.$$

Nótese que puede haber más de un brazo óptimo y a^* solo denotará al primero de ellos. Asimismo, dado que existe una acción óptima, tiene sentido considerar cuánto de lejos está cada uno de los brazos del óptimo, es decir, cuál es la pérdida asociada a seleccionar cada brazo subóptimo. Se define el **salto de suboptimalidad** del brazo $a \in \mathcal{A}$ como

$$\Delta_a(\nu) := \mu^*(\nu) - \mu_a(\nu) \,.$$

Claramente, para todo brazo $a \in \mathcal{A}$ óptimo, $\Delta_a(\nu) = 0$. Cuando quede claro el bandido al que nos referimos, escribiremos simplemente μ_a, μ^* y Δ_a.

En la siguiente subsección presentaremos algunas estrategias con las que abordar los problemas de bandidos estocásticos. Todas tienen en común la siguiente consideración: buscan estimar la media μ_a de la distribución asociada a cada uno de los brazos $a \in \mathcal{A}$ para, con este conocimiento, poder destinar un mayor número de rondas a las acciones con μ_a mayor.

Asociada a una política π que resuelve un problema de bandidos estocásticos se tienen las siguientes cantidades.

- Para cada $a \in \mathcal{A}$ y cada $t \in \{1, \dots, n\}$

  $$T_a(t) := \sum_{s=1}^{t} \mathbb{I}\{A_s = a\}$$

  es el número de veces que la acción a habrá sido elegida por el agente al final de la ronda t. Como las acciones de agente y entorno no son, en general, deterministas, $T_a(t)$ es una variable aleatoria.

- Para cada $a \in \mathcal{A}$ y cada $t \in \{1, \dots, n\}$

  $$\hat{\mu}_a(t) := \frac{1}{T_a(t)} \sum_{s=1}^{t} \mathbb{I}\{A_s = a\} X_s$$

  es la media de las recompensas obtenidas por el agente provenientes del brazo a en las t primeras rondas. Se puede probar que $\hat{\mu}_a(t)$ tiene esperanza μ_a (se ve en mi TFG de Matemáticas).

6.3. Remordimiento en bandidos estocásticos

La política óptima de un bandido estocástico siempre es constante: es aquella que, independientemente de la ronda, elige siempre el brazo a cuya media μ_a es máxima. Entonces, con cualquier clase competidora Π que contenga las políticas constantes, el remordimiento de un bandido estocástico se escribe como:

$$R_n = n\mu^* - \mathbb{E}\left[\sum_{t=1}^{n} X_t\right] .$$

Nótese que aunque las recompensas X_t puedan no ser acotadas, sus medias lo son,

$$E[X_t] \leq \max_{a\in\mathcal{A}} \mu_a \,,$$

lo que nos da la acotación

$$R_n \leq \sum_{a\in\mathcal{A}} \Delta_a \leq n \max_{a\in\mathcal{A}} \Delta_a \,.$$

Es decir, en el caso de los bandidos estocásticos se obtiene que el remordimiento a lo largo de n rondas, R_n, está en $O(n)$ sin necesidad de exigir que las recompensas estén acotadas, sino tan solo que tengan media finita.

Variables aleatorias asociadas al remordimiento

Aunque resulta apropiado que el remordimiento sea una cantidad fija (no estocástica) para poder utilizarlo como medida comparativa entre distintos algoritmos, eliminar la estocasticidad del mismo nos puede hacer olvidar que, en la práctica, nunca lo podremos calcular de forma exacta. Cuando programemos un algoritmo de bandidos y midamos el remordimiento en que el bandido incurre en una determinada ejecución lo que estaremos haciendo será extraer una muestra de una de las siguientes variables aleatorias:

- **Remordimiento aleatorio** (*random regret* en inglés):

$$\hat{R}_n = n\mu^* - \sum_{t=1}^{n} X_t \,.$$

- **Pseudo-remordimiento** (*pseudo-regret* en inglés):

$$\bar{R}_n = n\mu^* - \sum_{t=1}^{n} \mu_{A_t} \,.$$

Es claro que

$$E[\hat{R}_n] = n\mu^* - E\left[\sum_{t=1}^{n} X_t\right] = R_n \,.$$

Para ver que también $E[\bar{R}_n] = E[R_n]$ basta con ver que $E[\mu_{A_t}] = E[X_t]$ para cualquier ronda t. Dado que se tiene $\mu_a = E[X_t|A_t = a]$, aplicando la ley de la esperanza total (teorema 5.1.5):

$$E[X_t] = \sum_{a\in\mathcal{A}} P(A_t = a)E[X_t|A_t = a] = \sum_{a\in\mathcal{A}} P(A_t = a)\mu_a = \sum_{a\in\mathcal{A}} P(A_t = a)E[\mu_{A_t}] \,,$$

de manera que se deduce $E[\bar{R}_n] = E[R_n]$.

Por tanto, la media muestral de una m.a.s tanto del remordimiento aleatorio $\hat{R}_n$ como del pseudo-remordimiento $\bar{R}_n$ puede servirnos para estimar el remordimiento esperado. Además, que sean variables aleatorias permite considerar sus medidas de dispersión (varianza, desviación típica, ...) y, a partir de ellas,

analizar con qué probabilidad el comportamiento de nuestro agente frente a un problema concreto puede alejarse del esperado, ya sea para mejor o para peor.

El remordimiento aleatorio $\hat{R}_n$ compara la mayor recompensa esperada $n\mu^*$, que es una cantidad fija determinista, con la recompensa total obtenida por el agente en la ejecución. Este segundo término es doblemente estocástico. En primer lugar, los brazos A_t que el agente escogerá en cada ronda t son, en general, variables aleatorias. En segundo lugar, incluso fijando el brazo escogido por el agente, digamos $A_t = a_t$, la recompensa $X_t|A_t = a_t$ se extrae de la distribución de probabilidad P_{a_t} y, por tanto, sigue siendo estocástica.

En cambio, el pseudo-remordimiento $\bar{R}_n$ compara la mayor recompensa esperada $n\mu^*$ con la suma de las medias de las distribuciones asociadas a los brazos escogidos por el agente. Este segundo término es estocástico en tanto que los brazos escogidos A_t lo son, pero fijadas las acciones de cada ronda deja de serlo.

Por tanto, el remordimiento aleatorio contempla toda la estocasticidad del juego secuencial, mientras que el pseudo-remordimiento filtra el ruido $X_t - \mu_{A_t}$ de las recompensas obtenidas. Esto hace que el pseudo-remordimiento sea una medida más "justa" del rendimiento del agente, ya que lo evalúa según si los brazos escogidos por este son buenos en lugar de por si se han comportado bien en la ejecución. Por este motivo, en la práctica para estimar el remordimiento preferiremos utilizar el pseudo-remordimiento siempre que sea posible. No obstante, el remordimiento aleatorio tiene la ventaja de que para obtener una m.a.s del mismo no se requiere conocer las medias de las distribuciones asociadas a los brazos no óptimos, sino que se calcula directamente a partir de las recompensas obtenidas en la ejecución.

Para extraer una muestra del remordimiento aleatorio $\hat{R}_n$ o del pseudo-remordimiento $\bar{R}_n$ es necesario conocer tanto las distribuciones de los brazos del bandido (entorno) como la estrategia seguida por el agente y realizar una simulación ronda a ronda de la interacción entre ambos. Si se extrae un elevado número de muestras puede esperarse que la media muestral de las mismas esté próxima al valor real R_n del remordimiento. Puesto que se calcula como una media de una serie de muestras extraídas de un proceso estocástico, la cantidad R_n también recibe el nombre de remordimiento esperado. La velocidad con que la media de las muestras se aproxime a R_n depende de la dispersión de $\hat{R}_n$ o $\bar{R}_n$, la cual dependerá del algoritmo utilizado.

Descomposición del remordimiento

El hecho de que se cumpla $E[\bar{R}_n] = R_n$ pone de manifiesto como el remordimiento asociado a la aplicación de cualquier algoritmo de bandidos sobre un bandido estocástico fijo no depende de las recompensas extraídas por el entorno, sino que queda determinado por las acciones del agente. Las realimentación dada por el entorno influye, naturalmente, en la recompensa acumulada que se obtenga en una ejecución concreta (o equivalentemente, en la muestra del remordimiento aleatorio que se extraiga en dicha ejecución), pero el remordimiento mide la penalización media y esta queda perfectamente determinada a partir de las medias μ_a de las distribuciones asociadas a cada brazo (que son constantes asociadas al bandido enfrentado, independientes del algoritmo que se utilice) y de los brazos A_t seleccionados en cada ronda por el agente. Podemos decir incluso más: dado que las recompensas X_t se extraen de manera independiente unas de otras, el remordimiento no puede variar por seleccionar los mismos brazos en un orden diferente. Por tanto, para un bandido estocástico fijo, el remordimiento dependerá únicamente del número veces que se seleccione cada brazo, es decir, del valor que tomen las variables aleatorias $T_a(n)$ para $a \in \mathcal{A}$.

En general, es claro que la política que determine qué brazo se selecciona tendrá en cuenta las recompensas que se han obtenido en rondas anteriores. Se podría argumentar, por tanto, que lo anterior no tiene ninguna importancia porque, si bien el remordimiento no depende de modo directo de las recompensas concretas extraídas, sí depende indirectamente a través de la política que las toma en consideración a la hora de elegir cuántas veces se selecciona cada brazo. Si bien esto es cierto para ejecuciones concretas (la muestra que se obtenga del pseudo-remordimiento variará en función de las recompensas que se extraigan en la ejecución), no lo es en media: $T_a(n)$ es una variable aleatoria que podrá tomar distintos valores en distintas ejecuciones, pero $E[T_a(n)]$ es una constante asociada al par bandido-política. La única influencia que tiene el entorno sobre el remordimiento esperado son, pues, las medias μ_a de las distribuciones.

Las definiciones

$$\bar{R}_n = n\mu^* - \sum_{t=1}^{n} \mu_{A_t}$$

y

$$R_n = n\mu^* - \mathbb{E}\left[\sum_{t=1}^{n} X_t\right]$$

expresan el pseudo-remordimiento y el remordimiento esperado como la suma de los remordimientos asociados a cada ronda $t \in \{1, \ldots, n\}$ y en cierto modo ocultan cómo lo único que influye en el remordimiento son los μ_a y los $T_a(n)$. Alternativamente, pueden escribirse como una suma de los remordimientos asociados a cada brazo $a \in \mathcal{A}$ del bandido, como se prueba en el siguiente teorema. Esto muestra de forma más clara los factores que influyen en el remordimiento.

Teorema 6.3.1 (Descomposición del pseudo-remordimiento)**.** *Para cualquier política π, bandido estocástico ν y horizonte $n \in \mathbb{N}$, el pseudo-remordimiento verifica:*

$$\bar{R}_n = \sum_{a \in \mathcal{A}} \Delta_a T_a(n)\,.$$

Tomando esperanzas en la expresión anterior se obtiene el resultado análogo para el remordimiento esperado.

Corolario 6.3.1 (Descomposición del remordimiento)**.** *Para cualquier política π, bandido estocástico ν y horizonte $n \in \mathbb{N}$, el remordimiento verifica*

$$R_n = \sum_{a \in \mathcal{A}} \Delta_a E[T_a(n)]\,.$$

Del resultado anterior se extraen las siguientes conclusiones:

- Si el objetivo es minimizar el remordimiento el agente debe usar cada brazo del bandido con frecuencia inversamente proporcional a su salto de suboptimalidad estimado.
- El remordimiento esperado de una ejecución de un algoritmo sobre un bandido estocástico solo depende de los saltos de suboptimalidad del bandido y del número de veces que (de media) se seleccione cada brazo.

No nos dice, pues, nada que no supiéramos, pero esta forma de expresar el pseudo-remordimiento nos será útil para explicar otras intuiciones en el futuro y, sobre todo, en la demostración de garantías teóricas de los algoritmos.

6.4. Asunción subgaussiana

Para poder garantizar un buen comportamiento de los algoritmos que veremos para resolver el problema de bandidos estocásticos, será necesario asumir una cierta regularidad en la velocidad en que las colas de las distribuciones asociadas a los brazos decaen. Nótese que, si la recompensa extraída de un brazo a pudiera tomar valores alejados de su media μ_a con probabilidad arbitrariamente alta, no habría forma de estimar en qué medida la media muestral de las recompensas extraídas de a son una buena estimación de μ_a. Resulta esencial, por tanto, poder aplicar una desigualdad de concentración que proporcione una cota más o menos ajustada de con qué probabilidad se alejan las recompensas de su media.

En el capítulo 5 vimos distintas desigualdades de concentración y llegamos a la conclusión de que, para que fueran medianamente ajustadas, era necesario realizar alguna asunción sobre las distribuciones que tratamos. En concreto, para este trabajo asumiremos siempre que las distribuciones de los brazos pertenecen a la familia subgaussiana, que vimos que proporciona fuertes desigualdades de concentración a la vez que incluye

a muchas de las distribuciones interesantes. En concreto, todas las de soporte acotado son subgaussianas y es bastante habitual encontrar distribuciones de este tipo.

Supondremos, entonces, que existen $\{\sigma_a\}_{a\in\mathcal{A}}$ positivos tales que para cada $a \in \mathcal{A}$

$$P_a \in \text{subG}(\sigma_a)\,.$$

Utilizaremos esta asunción para obtener cotas del remordimiento producido por los distintos algoritmos que veremos y para elegir, en función de datos conocidos, los parámetros de entrada del algoritmo de forma que dichas cotas sean lo menores posibles.

En las demostraciones de [2], los autores han decidido simplificar al máximo la intuición subgaussiana, de manera que se asume que las distribuciones asociadas a todos los brazos son 1-subgaussianas. Esta no es realmente una restricción sobre el tipo de distribuciones que se pueden considerar debido a los siguientes motivos:

1 Recordemos que toda distribución σ_1-subgaussiana es σ_2-subgaussiana para cualquier $\sigma_2 \geq \sigma_1$. Entonces, puesto que estamos considerando que hay un número finito de brazos, denotando $\sigma := \max_{a\in\mathcal{A}} \sigma_a$ tenemos que las distribuciones de todos los brazos son σ-subgaussianas.

2 Por el teorema 5.2.2 multiplicar una variable aleatoria σ-subgaussiana por $\frac{1}{\sigma}$ resulta en una variable aleatoria 1-subgaussiana. Aplicando esto sobre todas las distribuciones de los brazos, podemos obtener un nuevo bandido (diferenciándose del anterior solo en que las recompensas se escalan tras cada extracción) en el que las distribuciones de todos los brazos son 1-subgaussianas.

Asumir, pues, que el bandido es 1-subgaussiano no limita las distribuciones que se pueden tratar más que la asunción de que sean subgaussianas, y tiene la ventaja de simplificar las demostraciones y las fórmulas derivadas de estas. Además, en un caso real se requiere conocer el valor de σ para computar las cotas y es más fácil que se conozca un parámetro σ de la familia subgaussiana aplicable a todos los brazos del bandido que uno distinto para cada brazo. En bandidos estocásticos no estructurados, lo habitual es que de entrada todos los brazos sean iguales para el agente y contar con distintos σ_a para cada brazo atenta ligeramente contra esta filosofía.

Por otro lado, si se conocen los parámetros $\{\sigma_a\}_{a\in\mathcal{A}}$ de subgaussianidad de cada brazo el hecho de sustituirlos todos por su máximo supone dejar pasar una oportunidad de refinar la cota del remordimiento. Si las diferencias entre los σ_a son grandes, la variación entre tenerlos o no en cuenta puede ser importante. Además, el escalado para hacer que todas las distribuciones sean 1-subgaussianas oscurece el hecho de que el conocimiento del parámetro σ (o, al menos de un $\sigma \geq \sigma_a$ para todo $a \in \mathcal{A}$) siempre es necesario. Si se aplica un algoritmo de los que veremos con un bandido σ-subgaussiano con $\sigma > 1$ se deberá tener un cuenta que las recompensas están siendo escaladas un factor $\frac{1}{\sigma}$ para adaptar al caso la cota que veremos aquí.

En este trabajo se expondrán, se proporcionarán las intuiciones y se harán comprobaciones empíricas sobre los resultados teóricos proporcionados por [2], de manera que asumiremos, en adelante, que las distribuciones asociadas a todos los brazos de los bandidos estocásticos considerados son 1-subgaussianas. La consideración de distintos parámetros $\{\sigma_a\}_{a\in\mathcal{A}}$ y el correspondiente refinamiento de las cotas del remordimiento se tratarán en mi TFG de Matemáticas.

7. Explora-Primero

Como ya adelantamos en 4.2, la estrategia consistente en explorar todos los brazos un número fijo de veces al principio y, a continuación, dedicar el resto de rondas a explotar el mejor brazo encontrado en la exploración no funciona adecuadamente. Como vimos, variar el horizonte n hacía que el número de exploraciones iniciales necesarias para obtener un remordimiento reducido variara. Sin embargo, si hacemos depender el número m de exploraciones iniciales de cada brazo de n, obtenemos el algoritmo más sencillo de los que tienen un buen comportamiento frente a bandidos estocásticos: el algoritmo Explora-Primero (*Explore-First* en inglés), que también recibe otros nombre como Explorar-Y-Comprometerse (*Explore-Then-Commit* en inglés) y ε-Primero (ε-*First* en inglés).

7.1. El algoritmo

El algoritmo Explora-Primero (que en ocasiones abreviaremos como EP) consta de una fase de exploración, en la que prueba todos los brazos cierto número de veces, seguida de una fase de explotación, en la que el agente ejecuta únicamente la acción que ha proporcionado una mayor recompensa en la fase de exploración. Tiene como único parámetro el número $m \in \{0, \dots, \frac{n}{k}\}$ de veces que explora cada uno de los k brazos.

En las primeras mk rondas la política no tiene en cuenta las recompensas recibidas: ejecuta todas las acciones disponibles, una detrás de otra, y repite el proceso m veces. En las $n-mk$ rondas restantes, el agente ejecuta una única acción, que es aquella $\hat{a} \in \mathcal{A}$ para la que ha obtenido, de media, mayores recompensas en las mk primeras rondas, es decir, para la que $\hat{\mu}_a(mk)$ es máximo,

$$\hat{a} = \text{argmax}_{a \in \mathcal{A}}\{\hat{\mu}_a(mk)\}\,.$$

El esquema de Explora-Primero se muestra en el Algoritmo 2.

7.1.1. Desempate

En el caso de que haya varios brazos con $\hat{\mu}_a$ máximo, nos surge la duda de si fijar una regla para determinar cuál de ellos escogemos como $\hat{a}$ (por ejemplo, quedarnos siempre con el de menor índice de entre los que maximizan la media muestral), si deberíamos elegirlo aleatoriamente entre los candidatos o si deberíamos alternar entre ellos durante la fase de explotación. Además, en el caso de escogerlo aleatoriamente, cabría hacerlo una sola vez al finalizar la fase de exploración o escogerlo aleatoriamente en cada ronda de la fase de explotación.

En el caso en que las distribuciones de los brazos son continuas (y hay un número finito de brazos), es fácil ver que esta decisión no tiene importancia: el suceso consistente en que dos medias muestrales $\hat{\mu}_a$ y $\hat{\mu}_b$ correspondientes a brazos distintos coincidan tiene probabilidad cero. En efecto, dado que la suma de variables aleatorias continuas e independientes sigue siendo continua, $\hat{\mu}_a - \hat{\mu}_b$ es una variable aleatoria continua, así que toma el valor 0 con probabilidad 0. Entonces, el conjunto

$$\{\hat{\mu}_a(mk) = \hat{\mu}_b(mk)\} = \{\hat{\mu}_a(mk) - \hat{\mu}_b(mk) = 0\}$$

tiene probabilidad 0. Puesto que

$$\{\hat{\mu}_a(mk) = \hat{\mu}_b(mk) \text{ para algún } a \neq b\}$$

es la unión de un número finito de sucesos con probabilidad 0, necesariamente tiene probabilidad 0.

En el caso en que las distribuciones de los brazos son discretas, la probabilidad de que dos medias muestrales coincidan deja de ser nula, pero en general será pequeña.

Además, si las distribuciones asociadas a brazos distintos tienen media diferente, conforme incrementemos el valor de m disminuirá la probabilidad de que las medias muestrales de las recompensas de la fase de exploración coincidan. Vamos a comprobarlo empíricamente:

Algoritmo 2 Explora-Primero

parameters

m: número de veces que se explora cada brazo

end parameters

initialize

$s_a \leftarrow 0$ for a in $\mathcal{A}$

end initialize

▷ Fase de exploración

for t in $1, \ldots, m$ **do**

  for $a \in \mathcal{A}$ **do**

    $x \leftarrow$ ejecutar(a)

    $s_a \leftarrow s_a + x$

  end for

end for

$\hat{a} \leftarrow \text{argmax}_{a \in \mathcal{A}} s_a$

▷ Fase de explotación

for $t \in m+1, \ldots, n$ **do**

  ejecutar$(\hat{a})$

end for

- Intuitivamente, la probabilidad de que 2 medias muestrales coincidan es mayor cuanto más pequeño sea el soporte de las distribuciones de los brazos. Por este motivo, comenzamos considerando un bandido de 2 brazos con distribuciones Bernoulli(0.6) y Bernoulli(0.5) y un horizonte de $n = 1000$ rondas. Para cada valor de m entre 0 y 200 ejecutamos dos versiones del algoritmo Explora-Primero con parámetro m: una (línea azul) que en caso de empate selecciona el primer brazo (el óptimo) y otra (línea roja) que en caso de empate selecciona el segundo brazo (el subóptimo). No es necesario ni conveniente representar el remordimiento para las otras estrategias de desempate ya que, dado que escogerán en algunos casos el brazo 1 y en otros el brazo 2, el remordimiento que obtengan estará entre los dos mostrados. Realizamos 100 simulaciones de cada ejecución y promediamos el remordimiento obtenido. Los resultados se muestran en la figura 3.

  Se observa que la diferencia entre favorecer a un brazo o al otro es grande cuando el número de exploraciones es reducido. En el caso $m = 0$, en el que ambas medias muestrales siempre coinciden con valor 0 (dado que no se realiza exploración) es claro que el remordimiento es nulo cuando se favorece al brazo óptimo y máximo (0.1 del salto asociado al brazo subóptimo por 1000 rondas lleva a un remordimiento máximo de 100) cuando se favorece al brazo subóptimo. Conforme se incrementa el valor de m esta diferencia disminuye rápidamente, aunque hasta en torno a $m = 100$ se sigue apreciando la superioridad de la estrategia que favorece al brazo bueno. Notamos, asimismo, que este valor, $m = 100$, es próximo al m para el que remordimiento es mínimo.

- Nos interesa saber si lo anterior se repite cuando las distribuciones tienen un soporte mayor. Tomamos como distribuciones de los brazos Pois(1.2) y Pois(1) y repetimos el experimento (con la misma n y

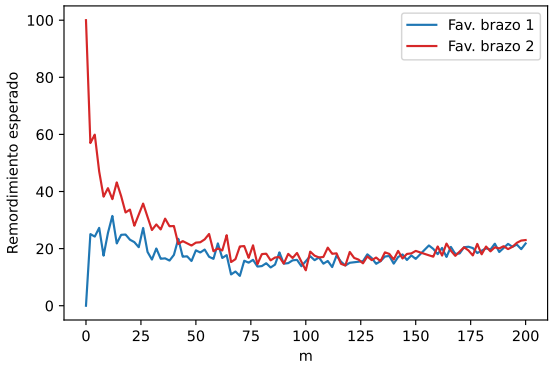


Figura 3: Remordimiento bandido Bernoulli de 2 brazos en función del brazo favorecido.

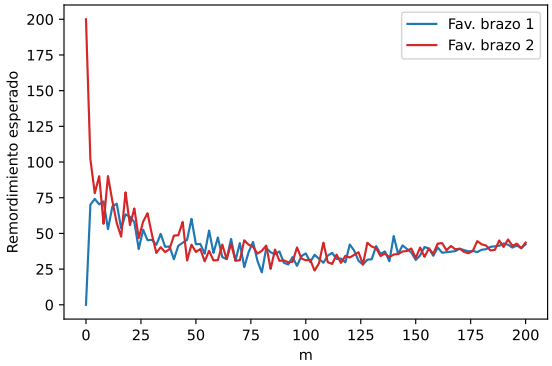


Figura 4: Remordimiento bandido Poisson de 2 brazos en función del brazo favorecido.

número de simulaciones). Los resultados se muestran en la figura 4.

En este caso, la diferencia empieza (para $m = 0$) siendo tan grande como en el caso Bernoulli, pero mengua muy rápidamente. Antes de $m = 50$ la superioridad de la estrategia que favorece al brazo óptimo ya ha desaparecido, y este es un valor de m considerablemente menor al valor en que se alcanza el mínimo del remordimiento (aproximadamente $m = 100$, nuevamente).

- Realizamos también el experimento para un caso en que las distribuciones de los brazos son continuas, aunque no esperamos encontrar diferencia entre ambas estrategias. Tomando como distribuciones de los brazos $N(0, 1)$ y $N(-0.1, 1)$, se obtiene la figura 5.

  Salvo en el caso especial $m = 0$, no se observa un remordimiento menor en la estrategia que favorece al brazo óptimo. Previsiblemente esto ocurre porque, al ser las distribuciones continuas, no se produce la igualdad de las medias muestrales en prácticamente ninguna ocasión, así que ambas estrategias resultan idénticas.

Concluimos que, si bien en distribuciones discretas y para valores pequeños de m se aprecia una cierta diferencia en el remordimiento obtenido en función de la estrategia de desempate, esta desaparece si se incrementa lo bastante el valor de m. De hecho, como mucho tiene que incrementar m hasta su valor óptimo para que la diferencia desaparezca. Intuitivamente, el valor óptimo de m es tal que el algoritmo ya ha recopilado la suficiente información para que los empates entre medias muestrales no se den con la bastante frecuencia como para que la estrategia de desempate tenga algún impacto en el remordimiento esperado.

En este trabajo se evitará, en lo posible, hacer ninguna asunción sobre la forma de desempate, de manera que los resultados teóricos sean válidos independientemente de cuál se implemente. En la práctica, no obstan-

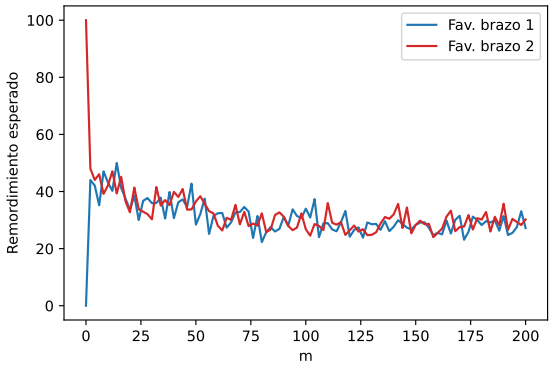


Figura 5: Remordimiento bandido Normal de 2 brazos en función del brazo favorecido.

te, hay que decantarse por alguna de las opciones y aquí seguiremos la aproximación de [3] y consideraremos que $\hat{a}$ se escoge una sola vez, aleatoriamente y con probabilidad uniforme entre los brazos con $\hat{\mu}_a$ máxima.

7.2. Análisis del remordimiento

En esta subsección estudiaremos el remordimiento en que incurre el algoritmo Explora-Primero en función de su parámetro m y al aplicarse sobre bandidos suficientemente regulares. El objetivo será, por un lado, obtener cotas del máximo remordimiento que se puede producir al utilizar este algoritmo, pero también dar con procedimientos que permitan seleccionar un valor adecuado para el parámetro m sin requerir de mucho conocimiento previo sobre el bandido enfrentado.

7.2.1. Idea general e intuiciones

Por el lema de descomposición del pseudo-remordimiento 6.3.1, el pseudo-remordimiento asociado a una ejecución de un algoritmo sobre un bandido estocástico puede escribirse como suma de las pérdidas asociadas a cada una de las acciones,

$$\bar{R}_n = \sum_{a \in \mathcal{A}} \Delta_a T_a(n)\,.$$

La aleatoriedad del mismo reside exclusivamente en las variables aleatorias $\{T_a(n)\}_{a\in\mathcal{A}}$, es decir, en cuántas veces se selecciona cada uno de los brazos. Ahora, en el caso del algoritmo Explora-Primero, sabemos que cada brazo será seleccionado exactamente m veces en la fase de exploración (no hay aleatoriedad aquí) y, en la fase de explotación, solo se seleccionará el brazo $\hat{a}$. De esta manera, se tiene

$$T_a(n) = m + (n - mk)\mathbb{I}\{\hat{a} = a\}\,,$$

con lo que

$$\bar{R}_n = m \sum_{a \in \mathcal{A}} \Delta_a + (n - mk)\Delta_{\hat{a}}\,.$$

Por tanto, el pseudo-remordimiento en Explora-Primero es una suma de una cantidad determinista, $m \sum_{a\in\mathcal{A}} \Delta_a$, y una cantidad estocástica. La clave para que el pseudo-remordimiento no se vea disparado por el segundo sumando es que $\Delta_{\hat{a}} = 0$ (es decir, $\hat{a}$ es un brazo óptimo) con probabilidad alta, lo cual ocurrirá cuando m sea lo bastante grande para que la confianza en que $\hat{a}$ sea óptimo sea elevada. Tomando esperanzas en la igualdad anterior se tiene la igualdad análoga para el remordimiento:

$$R_n = m \sum_{a \in \mathcal{A}} \Delta_a + (n - mk)E[\Delta_{\hat{a}}]\,,$$

Se verá que, si las distribuciones de los brazos tienen colas que decaen lo bastante rápido (en concreto, si son subgaussianas) el segundo sumando de la expresión anterior disminuye al menos exponencialmente con el valor de m. Nótese que el remordimiento se dispara tanto cuando m tiende a 0 (el segundo término crece mucho) como cuando tiende a $+\infty$ (el primer término crece mucho). Lo que queremos es encontrar un m intermedio (óptimo) tal que ambos sumandos tengan tamaño mediano y se minimice la suma total.

Por tanto, intuitivamente tenemos que la dependencia del remordimiento respecto del parámetro m, para bandidos subgaussianos, es la siguiente:

- Para m mucho menor que el óptimo, el remordimiento es elevado, pero disminuye rápidamente (exponencialmente) al aumentar m.
- Para m próximo al óptimo, el remordimiento es reducido y varía poco al variar m.
- Para m mucho mayor que el óptimo, el remordimiento es elevado y se incrementa linealmente al aumentar m, con pendiente aproximada $\sum_{a\in\mathcal{A}} \Delta_a$.

El siguiente ejemplo pondrá de manifiesto algunas de estas cuestiones.

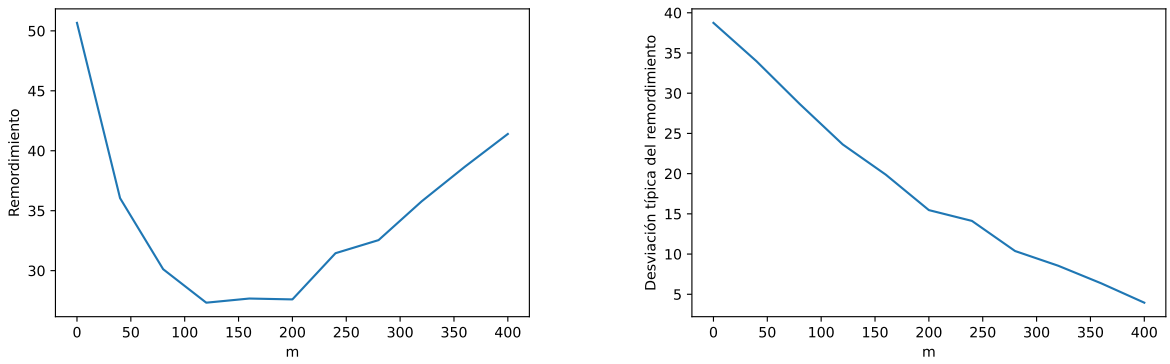
7.2.2. Medición del remordimiento

Incluye parte de la respuesta al ejercicio 6.9 de [2]

Antes de pasar a presentar las distintas cotas del remordimiento de Explora-Primero conviene analizar ligeramente cómo se produce el muestreo del pseudo-remordimiento para este algoritmo. En particular, nos interesa saber cuál es el impacto del parámetro m en la medición.

Para ello, consideramos un bandido de 2 brazos con distribuciones $N(0,1)$ y $N(-0.1,1)$ y un horizonte de $n = 1000$ rondas. Dado un valor de $m \in \{0,\dots,500\}$, para obtener una muestra del pseudo-remordimiento basta, como sabemos, con realizar una simulación del algoritmo y, en cada paso, sumar el salto de subóptimalidad del brazo escogido. Dado que a partir de la ronda $2m$ la elección de brazo de todas las rondas posteriores ya está perfectamente determinada, solo es necesario llevar a cabo la simulación hasta esta ronda, pues el remordimiento de las $n-2m$ últimas rondas puede calcularse de golpe sin más que conocer el brazo $\hat{a}$ elegido al final de la fase de exploración.

Para cada $m \in \{0,\dots,400\}$ realizamos 600 simulaciones del algoritmo Explora-Primero con parámetro m y calculamos la media y la desviación típica muestrales del pseudo-remordimiento. Los resultados se muestran en la figura 6.



(a) Esperanza. **(b)** Desviación típica.

Figura 6: Propiedades del pseudo-remordimiento de Explora-Primero en función del parámetro m. Esta figura, junto con la explicación de la misma, constituye parte de la respuesta al ejercicio 6.9 de [2].

En la figura 6a se muestra la media muestral del pseudo-remordimiento para distintos valores de m entre 0 y 400. Como el número de simulaciones es alto y la varianza de la desviación típica muestral disminuye linealmente con el número de simulaciones, esta media muestral será una buena aproximación del remordimiento esperado en la ejecución del algoritmo. Se observa como el remordimiento disminuye rápidamente para m inferior a 100, se mantiene más o menos constante y mínimo para m entre 100 y 200 y se incrementa de manera lineal a partir de $m = 200$. Se refleja, en definitiva, el fenómeno que mencionamos en el apartado anterior.

Se observa que el valor de m para el que se alcanza el remordimiento mínimo es $m_{Emp} = 120$. Nos gustaría contar con un método de estimar este valor de m a partir de los datos del problema, para que nos sea útil en un caso real. En general, no será factible aspirar a una precisión muy alta, pero, por la reducida variación del remordimiento para m en $(100, 200)$, nos daremos por satisfechos con una heurística que nos proporcione un valor de m en este intervalo.

En la figura 6b se muestra la desviación típica muestral del pseudo-remordimiento del experimento anterior para cada valor de m, que es una buena aproximación de la desviación típica real del pseudo-remordimiento. Se observa que esta disminuye linealmente conforme m aumenta hasta ser próximo a 0 cuando m es 400. Cuando $m \simeq 120$, la desviación típica es $\sigma \simeq 25$, que es casi tan grande como la media. Si bien seleccionar $m = 120$ proporciona un remordimiento en media de 27, una desviación típica de 25 nos da una probabilidad elevada de que el pseudo-remordimiento dé valores alejados de su media. Por tanto, existe mucha incertidumbre al extraer una sola muestra del remordimiento: no se puede esperar que si se repite la extracción se vaya a obtener un resultado similar. En este sentido, puede ser sabia una cierta preferencia por valores de m mayores, en los que la dispersión del pseudo-remordimiento es bastante menor. Por ejemplo, en este caso para que el pseudo-remordimiento sea mínimo en media habría que tomar m cercano a 120, pero esto conlleva una elevada incertidumbre: si se va a realizar una única ejecución hay una probabilidad no despreciable de que se obtenga un pseudo-remordimiento considerablemente mayor que el esperado. En cambio, si se opta por un m algo mayor, aunque el pseudo-remordimiento sea algo mayor en media será también más estable y se tendrá una confianza mayor de que al repetir la ejecución se vaya a obtener un valor similar.

Hay que destacar que la elevada desviación típica del pseudo-remordimiento deja de ser un problema cuando se toma un número elevado de muestras. En efecto, dado que el pseudo-remordimiento está restringido al intervalo $(0, n \max_{a \in \mathcal{A}} \Delta_a)$ podemos obtener una cota de su desviación típica σ, aunque sea poco ajustada, mediante la desigualdad de Popoviciu (teorema 5.1.3). A partir de ahí, el hecho de que la media muestral tenga para k muestras varianza $\frac{\sigma^2}{k}$ nos permite escoger k lo suficientemente grande para que la media muestral obtenida esté próxima a la media real del pseudo-remordimiento con probabilidad tan alta como queramos. Entonces, el problema de la desviación típica elevada no perjudica excesivamente a la medición del pseudo-remordimiento. Lo que sí conlleva es un comportamiento un tanto volátil del algoritmo, que se comportará bien en una ejecución y mal en la siguiente, lo cual podría ser un problema en un caso real.

Fijándonos un poco más nos damos cuenta de que la desviación típica muestral en realidad no aporta información adicional sobre la distribución del pseudo-remordimiento. Si se aplica el algoritmo de Explora-Primero con parámetro m sobre un bandido estocástico de 2 brazos en el cual el brazo subóptimo tiene salto de suboptimalidad Δ, solo hay dos posibles valores para el remordimiento:

- Si se detecta el brazo óptimo en la fase de exploración, el remordimiento en la fase de explotación será 0. Entonces, el remordimiento total será el obtenido en la fase de exploración: $m\Delta$.
- Si, por el contrario, se toma como óptimo el brazo subóptimo, la fase de explotación aumentará el remordimiento en $(n - 2m)\Delta$, pues el agente jugará sistemáticamente el brazo peor. Entonces, en este caso el remordimiento total será $m\Delta + (n - 2m)\Delta = (n - m)\Delta$.

En nuestro caso particular, el remordimiento puede tomar valor 12 o valor 88. La distribución del remordimiento queda perfectamente determinada solo con saber la probabilidad p con que toma valor 12, pero esto se puede calcular sin más a partir de la media:

$$E[\bar{R}_n] = 12p + 88(1 - p) \iff p = \frac{88 - E[\bar{R}_n]}{76}$$

En nuestro ejemplo, con una media de aproximadamente 27, se tiene que sobre el 80 % de las veces el remordimiento es 12 (se escoge el brazo "bueno") y el 20 % restante el remordimiento es 88 (se escoge el brazo "malo"). Por tanto, la desviación típica no aporta nueva información sobre la distribución. Aumentar el valor de m disminuye la desviación típica porque, aunque aumenta la proporción de muestras del remordimiento que toman el valor menor m, lo hace a costa de disminuir la distancia entre m y $n-m$ (es decir, el beneficio de averiguar el brazo bueno). En el caso extremo $m = \frac{n}{2}$, la desviación típica sería nula porque tendríamos una política de exploración uniforme y el remordimiento siempre sería m.

7.2.3. Cota principal del remordimiento

Recapitulando, tenemos entonces que el remordimiento asociado a una ejecución del algoritmo Explora-Primero con parámetro m y horizonte n verifica

$$R_n = \sum_{a \in \mathcal{A}} \Delta_a E[T_a(n)] \,,$$

donde

$$E[T_a(n)] = m + (n - mk)P(\hat{a} = a)$$

para cada $a \in \mathcal{A}$. Por tanto, para acotar el remordimiento del algoritmo, lo que necesitamos es acotar la cantidad

$$P(\hat{a} = a)$$

para cada $a \in \mathcal{A}$. Para ello, necesitaremos un resultado que acote la probabilidad de que las recompensas tomen valores alejados de la media de su distribución. A partir de aquí, por tanto, tendrá importancia esencial la desigualdad de concentración elegida para este propósito y los requerimientos necesarios para que esta se cumpla, que en nuestro caso será que las distribuciones asociadas a los brazos pertenezcan a la familia subgaussiana. En base a lo comentado en 6.4, asumiremos que el bandido a tratar es 1-subgaussiano, es decir,

$$P_a \in \text{subG}(1)$$

para cada $a \in \mathcal{A}$. Enunciamos a continuación la cota principal del remordimiento del algoritmo Explora-Primero según [2].

Teorema 7.2.1. *Cuando el algoritmo Explora-Primero con horizonte n y parámetro $m \in \{0, \ldots, \frac{n}{k}\}$ interactúa con un bandido estocástico 1-subgaussiano con saltos de suboptimalidad $\Delta_1, \ldots, \Delta_k$ se cumple que*

$$R_n \leq C_{EP}(n, m, \Delta_1, \ldots, \Delta_k) \,,$$

donde

$$C_{EP}(n, m, \Delta_1, \ldots, \Delta_k) := m \sum_{a \in \mathcal{A}} \Delta_a + (n - mk) \sum_{a \in \mathcal{A}} \Delta_a \exp\left(-\frac{m\Delta_a^2}{4}\right) .$$

La cota C_{EP} anterior ilustra el compromiso entre exploración y explotación. Si se destinan pocas rondas a la fase de exploración (m pequeño) el remordimiento en la fase de explotación (segundo término) es grande y, por tanto, el remordimiento total también lo es. Si, por el contrario, se destinan muchas rondas a la fase de exploración (m grande) es grande el remordimiento en la fase de exploración (primer término). Se ha de escoger, pues, un m intermedio para lograr un equilibrio entre el tamaño de ambos sumandos. La complejidad de la expresión nos impide dar una fórmula explícita para el valor de m que minimiza C_{EP}.

7.2.4. Fórmula explícita para m: m_{Teor}

Nos gustaría contar con una fórmula explícita para el valor de m que minimiza la cota C_{EP}. Sin embargo, derivando respecto de m e igualando a 0 la expresión resultante se obtiene una ecuación que, en general, no se puede resolver explícitamente. Una opción es aproximar el valor de m mediante técnicas de resolución de ecuaciones implícitas.

Alternativamente, si queremos una fórmula explícita para el m óptimo necesitamos realizar algunas asunciones y relajaciones que simplifican la expresión de C_{EP}. En primer lugar, nos ceñiremos al caso en el que el número k de brazos es 2. Se puede suponer, sin pérdida de generalidad, que el brazo óptimo es el primero, así que $\Delta_1 = 0$. Denotando $\Delta := \Delta_2$, la cota C_{EP} toma la forma:

$$C_{EP}(n, m, \Delta) = m\Delta + (n - 2m)\Delta \exp\left(-\frac{m\Delta^2}{4}\right)$$

En segundo lugar, relajamos la cota a una cuya derivada sea más sencilla:

$$C_{EP2}(n, m, \Delta) := m\Delta + n\Delta \exp\left(-\frac{m\Delta^2}{4}\right)$$

Como $n - 2m < n$, se tiene $R_n \leq C_{EP} < C_{EP2}$, así que C_{EP2} sigue siendo una cota del remordimiento, aunque menos ajustada.

Busquemos ahora el valor de m que minimiza C_{EP2}. Nótese que $\lim_{m\to+\infty} C_{EP2}(n, m, \Delta) = +\infty$. Se tiene:

$$\frac{\partial\, C_{EP2}}{\partial\, m} = \Delta - \frac{n\Delta^3}{4} \exp\left(-\frac{m\Delta^2}{4}\right)$$

Entonces:

$$\frac{\partial\, C_{EP2}}{\partial\, m} = 0 \iff \exp\left(\frac{m\Delta^2}{4}\right) = \frac{n\Delta^2}{4} \iff m = \frac{4}{\Delta^2} \log\left(\frac{n\Delta^2}{4}\right)$$

Como m está restringido a ser un entero no negativo, el valor de m que minimiza C_{EP2} será, salvo redondeo:

$$m_{Teor}(n, \Delta) = \max\left\{0, \left\lceil \frac{4}{\Delta^2} \log\left(\frac{n\Delta^2}{4}\right)\right\rceil\right\} \tag{2}$$

y el de la cota mínima:

$$C_{EP2\_min}(n, \Delta) := \min\left\{n\Delta, \Delta + \frac{4}{\Delta}\left(1 + \max\left\{0, \log\left(\frac{n\Delta^2}{4}\right)\right\}\right)\right\} \tag{3}$$

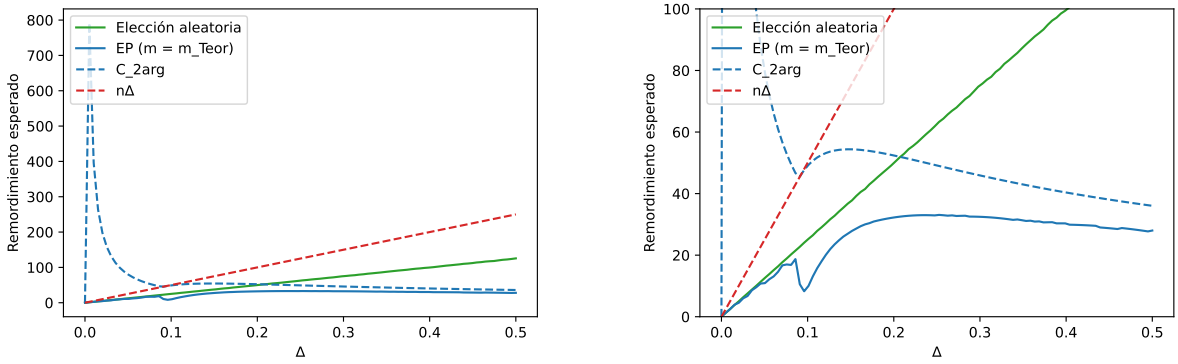
7.2.5. Remordimiento para $m = m_{Teor}$ reducido

Observamos que el primer argumento del mínimo anterior es una cota trivial: cualquier estrategia sobre un bandido de dos brazos obtendrá un remordimiento acotado superiormente por $n\Delta$. Pese a ello, si Δ es suficientemente pequeño el segundo argumento del mínimo (que depende de $\frac{1}{\Delta}$) será muy grande y la cota tomará el valor $n\Delta$. En este caso la cota obtenida no será significativa, es decir, utilizar el algoritmo Explora-Primero en esta situación no nos dará ninguna garantía adicional a cualquier otra estrategia (por ejemplo, seleccionar aleatoriamente el brazo en cada ronda). Nos interesa, entonces, comprobar dos aspectos:

- ¿Este hecho se da únicamente para valores extremadamente pequeños de Δ o tiene relevancia en casos reales?
- En el caso en que $C_{EP2\_min} = n\Delta$, ¿sigue siendo útil el algoritmo Explora-Primero?

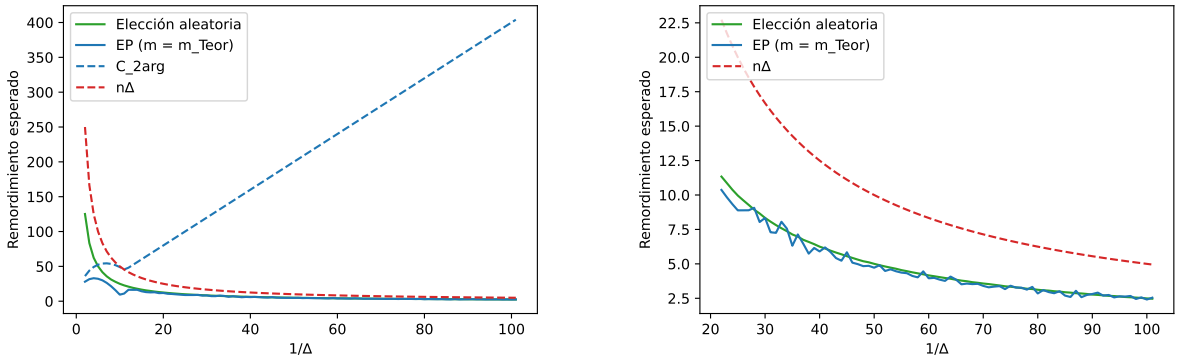
Vamos a tratar de dar respuesta a ambas preguntas, a la vez que evaluamos lo ajustado de la cota hallada y la bondad del algoritmo Explora-Primero, con el siguiente ejemplo. Dado $\Delta \in (0, 0.5)$, consideremos un bandido de dos brazos tal que las distribuciones de sus brazos sean $\mathrm{U}(0, 1 + 2\Delta)$ y $\mathrm{U}(0, 1)$, de manera que Δ es el salto de suboptimalidad del brazo subóptimo. Como las recompensas únicamente toman valores en el intervalo $(0, 2)$ el bandido es 1-subgaussiano en cualquier caso.

Para $\Delta \in (0, 0.5)$, ejecutamos el algoritmo Explora-Primero (con parámetro $m = m_{Teor}$) sobre la instancia del bandido con horizonte $n = 500$ y obtenemos el remordimiento resultado de promediar 500 repeticiones.



(a) **(b)**

Figura 7: Componentes de la cota C_{EP2_min} de Explora-Primero para valores grandes del salto de subóptimalidad Δ. El algoritmo se ejecuta sobre un bandido uniforme con horizonte $n = 500$.



(a) **(b)**

Figura 8: Componentes de la cota C_{EP2_min} de Explora-Primero para valores pequeños del salto de subóptimalidad Δ. El algoritmo se ejecuta sobre un bandido uniforme con horizonte $n = 500$.

Hacemos lo mismo para la estrategia consistente en seleccionar un brazo uniformemente al azar en cada una de las rondas. Calculamos también, para cada ronda, ambos argumentos del mínimo que da la cota C_{EP2_min}. Para abreviar, llamaremos simplemente $n\Delta$ al primer argumento del mínimo en la expresión de C_{EP2_min} y

$$C_{2arg} := \Delta + \frac{4}{\Delta}\left(1 + \max\left\{0, \log\left(\frac{n\Delta^2}{4}\right)\right\}\right)$$

al segundo. Representamos los resultados en las figuras 7 y 8.

En la figura 7 se observa el comportamiento para valores no muy pequeños de Δ. Observamos que en este caso el remordimiento del algoritmo Explora-Primero es mucho menor que el que se produce con una estrategia de elección aleatoria, como cabe esperar. Se ve también que la cota C_{2arg} es mucho menor que $n\Delta$, de manera que es la que siempre da valor C_{EP2_min}. En la figura 7b se observa que C_{EP2_min} resulta una cota bastante ajustada para el remordimiento para valores de $\Delta \in (0.2, 0.5)$, siendo más ajustada cuanto mayor es el valor de Δ. Por tanto, tenemos que para valores no muy pequeños de Δ el algoritmo Explora-Primero presenta un desempeño adecuado y la cota C_{EP2_min} es bastante cercana al valor real del remordimiento.

El "escalón" que se observa en la gráfica EP de la figura 7b corresponde al momento en que

$$\log\left(\frac{n\Delta^2}{4}\right)$$

pasa de ser negativo a positivo, de manera que m_{Teor} pasa de ser 1 a ser 16.

En contraste, en la figura 8 se muestra el resultado para valores cada vez más pequeños de Δ (en este caso, el eje X representa el inverso de Δ). En la parte izquierda de la figura 8a se observa el comportamiento adecuado (el que nos gustaría que se diera siempre) del algoritmo, hasta aproximadamente $1/\Delta = 10$. En este caso, C_{2arg} es muy inferior a $n\Delta$ (por lo que la cota del remordimiento no resulta vacua) y se observa que el remordimiento cometido por la estrategia Explora-Primero es muy inferior al que se da con una selección uniformemente aleatoria del brazo a ejecutar. Por tanto, para estos valores de Δ el algoritmo Explora-Primero sí resulta útil.

Sin embargo, si consideramos $\Delta < \frac{1}{10}$ se puede ver que la situación cambia. C_{2arg} se incrementa linealmente con $1/\Delta$ mientras que $n\Delta$, que se reduce mucho, pasa a convertirse en C_{EP2_min}, es decir, la cota deja de ser significativa. Pero, aunque ignoremos las garantías teóricas, el comportamiento empírico de las estrategias Explora-Primero y aleatoria se asemeja cada vez más conforme se reduce Δ. De hecho, en la figura 8b puede verse como, para $\Delta < \frac{1}{20}$, el remordimiento de ambas estrategias es prácticamente el mismo, haciéndose más y más parecido conforme disminuye Δ.

Por tanto, podemos concluir que el algoritmo Explora-Primero resulta del todo ineficaz en este caso para valores del salto del suboptimalidad inferiores a $\frac{1}{20}$. En general, este punto de inflexión tendrá una dependencia directa del valor del horizonte n. En efecto, si se repite el experimento para $n = 2000$ se obtiene la figura 9, en la que se observa que el Δ a partir del cual $n\Delta$ comienza a ser más pequeño que C_{2arg} es inferior.

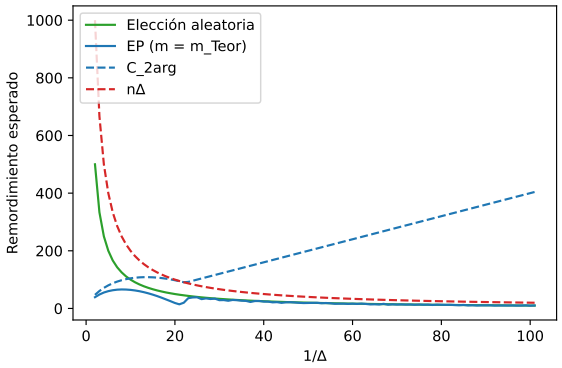


Figura 9: Componentes de la cota C_{EP2_min} de Explora-Primero para valores pequeños del salto de suboptimalidad Δ. El algoritmo se ejecuta sobre un bandido uniforme con horizonte $n = 2000$.

Nos preguntamos si es posible calcular este valor de Δ en función de n. Aunque la ecuación

$$n\Delta = C_{2arg}$$

no tiene solución explícita para Δ en términos de n, si tomamos $n = \frac{4}{\Delta^2}$ se da

$$n\Delta = \frac{4}{\Delta} < \Delta + \frac{4}{\Delta} = C_{2arg}\,.$$

Entonces el valor $\frac{2}{\sqrt{n}}$ puede servirnos como una estimación a la baja del Δ a partir del cual el algoritmo deja de ofrecer buenas garantías. Es decir, resulta inútil utilizar el algoritmo Explora-Primero para cualquier $\Delta < \frac{2}{\sqrt{n}}$. La expresión $\frac{2}{\sqrt{n}}$ disminuye de manera bastante lenta al incrementarse n, lo que explica que no hayamos ganado mucho margen al pasar de $n = 500$ en la figura 8 a $n = 2000$ en la figura 9.

7.2.6. Descomposición de la cota

Incluye la respuesta al ejercicio 6.2 de [2]

Como hemos visto en el apartado anterior, para cualquier n fijo la cota C_{2arg} puede ser arbitrariamente alta sin más que hacer tender a 0 el salto de suboptimalidad Δ. Por este motivo, es relevante el otro argumento $n\Delta$ de la cota C_{EP2_min}, que actúa como cota efectiva en los casos en que Δ es reducido (en concreto, $\Delta < \frac{2}{\sqrt{n}}$).

¿Cuál es el crecimiento asintótico del remordimiento R_n del algoritmo Explora-Primero cuando n va a infinito? A la vista de la cota

$$R_n \leq C_{EP2\_min}(n,\Delta) = \min\left\{n\Delta, \Delta + \frac{4}{\Delta}\left(1 + \max\left\{0, \log\left(\frac{n\Delta^2}{4}\right)\right\}\right)\right\}$$

nos vemos tentados de establecer

$$R_n = \min\{O(n), O(\log n)\} = O(\log n)$$

y darnos por satisfechos con el resultado. En efecto, existe una constante $K \in \mathbb{R}$ independiente de n tal que

$$R_n \leq K \log n\,.$$

pero dicha constante K es dependiente de Δ. Además se da

$$\lim_{\Delta \to 0} K = +\infty\,.$$

Por ello, incluso con Δ acotado, para cualquier n fijo la cantidad

$$C_{2arg} = \Delta + \frac{4}{\Delta}\left(1 + \max\left\{0, \log\left(\frac{n\Delta^2}{4}\right)\right\}\right)$$

puede ser arbitrariamente alta. Nos interesa, pues, dar con una cota de la forma

$$R_n \leq K f(n)$$

donde K sea una constante independiente de n y de Δ. Resulta que esto puede conseguirse de forma bastante directa a partir de $R_n \leq C_{EP2\_min}(n,\Delta)$ simplemente tomando como cota de R_n el primer argumento del mínimo cuando $\Delta \leq \frac{2}{\sqrt{n}}$ y el segundo en caso contrario. Nótese que C_{2arg} sí está acotada cuando $\Delta > \frac{2}{\sqrt{n}}$.

Vamos a obtener, pues, una nueva cota del remordimiento C_{Desc} de la forma

$$C_{Desc}(n,\Delta) = \Delta + C\sqrt{n}$$

para cierto $C > 0$.

Distinguimos dos casos en función de la relación entre n y Δ:

- Si $n \leq \frac{4}{\Delta^2}$, se tiene $\sqrt{n} \leq \frac{2}{\Delta}$ por lo que:

  $$\frac{n-1}{\sqrt{n}}\Delta \leq \frac{n}{\sqrt{n}}\Delta = \sqrt{n}\Delta \leq \frac{2}{\Delta}\Delta = 2$$

  Entonces:

  $$R_n \leq n\Delta = \Delta + (n-1)\Delta = \Delta + \frac{n-1}{\sqrt{n}}\Delta\sqrt{n} \leq \Delta + 2\sqrt{n}$$

- Si $n > \frac{4}{\Delta^2}$, se da $\log\left(\frac{n\Delta^2}{4}\right) > 0$ así que:

  $$R_n \leq \Delta + \frac{4}{\Delta}\left(1 + \log\left(\frac{n\Delta^2}{4}\right)\right)$$

Fijado $n \in \mathbb{N}$, definimos $f : \mathbb{R}^+ \to \mathbb{R}$ tal que:

$$f(x) = \frac{4}{x\sqrt{n}} \left(1 + \log\left(\frac{nx^2}{4}\right)\right)$$

Se tiene:

$$\lim_{x\to 0} f(x) = \lim_{x\to 0} \frac{4}{x\sqrt{n}} \cdot \lim_{x\to 0} \left(1 + \log\left(\frac{nx^2}{4}\right)\right) = (+\infty)\cdot(-\infty) = -\infty$$

$$\lim_{x\to +\infty} f(x) = \lim_{x\to +\infty} \frac{4}{x\sqrt{n}} + \lim_{x\to +\infty} \frac{4}{x\sqrt{n}} \log\left(\frac{nx^2}{4}\right) = 0 + 0 = 0$$

Calculemos el máximo de f en $\mathbb{R}^+$. Su derivada es

$$\begin{aligned} f'(x) &= \frac{-4\sqrt{n}}{x^2 n}\left(1 + \log\left(\frac{nx^2}{4}\right)\right) + \frac{4}{x\sqrt{n}} \frac{\frac{2nx}{4}}{\frac{nx^2}{4}} \\ &= \frac{-4}{x^2\sqrt{n}}\left(1 + \log\left(\frac{nx^2}{4}\right)\right) + \frac{8}{x^2\sqrt{n}} = \frac{4}{x^2\sqrt{n}}\left(1 - \log\left(\frac{nx^2}{4}\right)\right), \end{aligned}$$

de manera que:

$$f'(x) = 0 \iff \log\left(\frac{nx^2}{4}\right) = 1 \iff x^2 = \frac{4e}{n} \iff x = \frac{2e^{1/2}}{\sqrt{n}}$$

Tenemos, pues, que el máximo de f es:

$$f\left(\frac{2e^{1/2}}{\sqrt{n}}\right) = \frac{4}{\frac{2e^{1/2}}{\sqrt{n}}\sqrt{n}}\left(1 + \log\left(\frac{n\frac{4e}{n}}{4}\right)\right) = 4e^{-1/2}$$

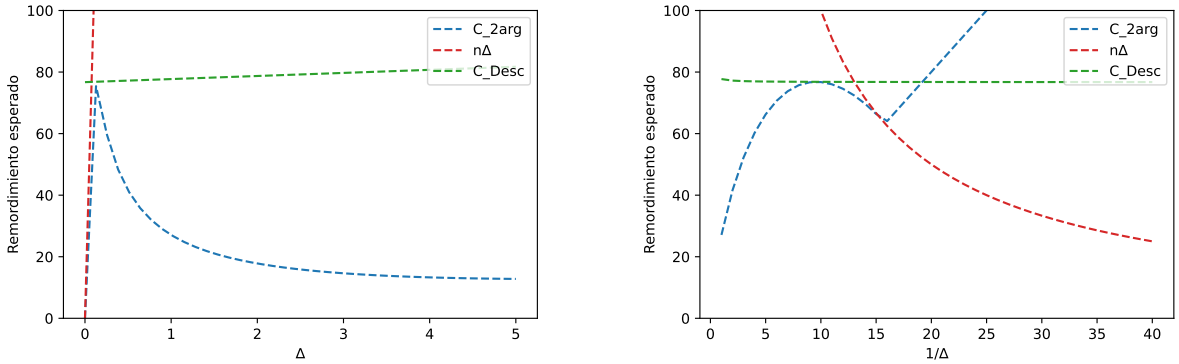
Por tanto:

$$R_n \leq \Delta + \frac{4}{\Delta}\left(1 + \log\left(\frac{n\Delta^2}{4}\right)\right) = \Delta + \frac{4}{\Delta\sqrt{n}}\left(1 + \log\left(\frac{n\Delta^2}{4}\right)\right)\sqrt{n} = \Delta + f(\Delta)\sqrt{n} \leq \Delta + 4e^{-1/2}\sqrt{n}$$

Por consiguiente, si tomamos $C := 4e^{-1/2} > 2$ se da

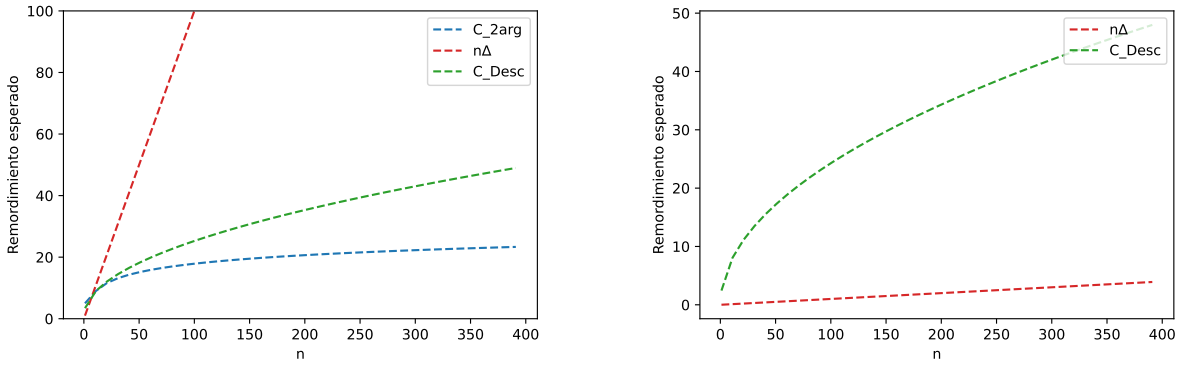
$$R_n \leq \Delta + C\sqrt{n}$$

y se tiene la cota $C_{Desc} = \Delta + 4e^{-1/2}\sqrt{n}$ para el remordimiento.



(a) Para valores grandes de Δ. **(b)** Para valores pequeños de Δ.

Figura 10: Comparación de C_{EP2_min} con la cota descompuesta para $n = 1000$ fijo.



(a) $\Delta = 1$. **(b)** $\Delta = 0.01$.

Figura 11: Comparación de C_{EP2_min} con la cota descompuesta para Δ fijo.

Como cabía pensar por haberse deducido de C_{EP2_min} y tener una dependencia mucho más limitada de Δ esta nueva cota es mucho menos ajustada. Este hecho puede observarse en las figuras 10 y 11. En todos los casos la cota C_{Desc} se sitúa en un punto intermedio entre C_{2arg} o bien $n\Delta$, de manera que siempre es menos ajustada que la mejor de ambas. En concreto, se da

$$C_{2arg} \leq C_{Desc} \leq n\Delta$$

para valores grandes de Δ ($\Delta > 0.1$) y

$$n\Delta \leq C_{Desc} \leq C_{2arg}$$

para valores pequeños de Δ. Sin embargo, en general está mucho más cerca que la mejor de ambas que de la peor, de manera que tiene un comportamiento aceptable.

La cota C_{Desc} sigue dependiendo de Δ, aunque en menor medida que C_{EP2_min}. Si tomamos en consideración todos los bandidos estocásticos posibles, que haya un sumando dependiente de Δ en la cota del remordimiento es inevitable, no solo para este algoritmo, sino para cualquiera que pruebe cada uno de los brazos al menos una vez. Por otro lado, en el ámbito de los bandidos estocásticos no estructurados es imposible garantizar un remordimiento acotado si no se prueban todos los brazos, ya que toda la información sobre las distribuciones de cada uno de los brazos proviene precisamente de escogerlo y evaluar la recompensa obtenida. Por ello, la única forma de obtener una cota independiente de Δ para el remordimiento de un algoritmo para bandidos estocásticos es conociendo una cota de Δ, algo que, afortunadamente, suele tenerse.

A diferencia de C_{EP2_min}, C_{Desc} sí que permite, en el caso de que Δ esté acotado, obtener una cota del remordimiento independiente de Δ. Así, si $\Delta \leq M$ se tiene

$$R_n \leq M + C\sqrt{n}$$

y ahora sí que podemos decir que $R_n = O(\sqrt{n})$, pues $R_n \leq (M+C)\sqrt{n}$ y la constante $M+C$ es independiente tanto de n como de Δ.

7.2.7. Elección óptima de m: m_{Opt}

Cabe preguntarse si el valor m_{Teor} calculado en (2) es realmente el mejor valor de m, m_{Opt}, con el que se puede ejecutar el algoritmo Explora-Primero o, por lo menos, si el remordimiento en que se incurre al elegir $m = m_{Teor}$ no es mucho mayor que el que se incurre al hacer $m = m_{Opt}$.

En general, el valor m_{Opt} que tiene una máxima probabilidad de minimizar el remordimiento no se podrá calcular, sino simplemente estimar ejecutando el algoritmo sobre el bandido en cuestión un número elevado de veces. El valor de m_{Opt} que se obtiene por este procedimiento es, pues, estocástico y puede que no sepamos

cuántas simulaciones realizar para que la aproximación sea buena. Sin embargo, hay casos en que, conociendo las distribuciones de los brazos y explotando sus propiedades, se puede calcular m_{Opt} de forma determinista.

Vamos a analizar a continuación algunos ejemplos de este tipo de bandidos. Es importante recalcar que el cálculo de m_{Opt} que realizaremos no resultaría de ninguna utilidad en la práctica, pues precisamente lo que se desconoce y se quiere estimar en un caso real son las distribuciones de los brazos. No obstante, nos servirá para comparar cómo de buena podemos esperar que sea la elección $m = m_{Teor}$.

a) En el caso trivial en el que las distribuciones de todos los brazos del bandido fueran degeneradas, es claro que $m_{Opt} = 1$. Esto es porque al extraer un valor de alguna de estas distribuciones se obtiene directamente como recompensa la media de la distribución en cuestión. Por tanto, basta probar una vez cada uno de los brazos para saber (con probabilidad 1) cuál es el brazo cuyo valor esperado es máximo.

b) Sean $p = 0.6$ y $q = 0.5$ y consideremos un bandido de dos brazos cuyo primer brazo tiene distribución Bernoulli(p) y el segundo tiene distribución degenerada con valor q. Nótese que la distribución degenerada es σ-subgaussiana para cualquier $\sigma > 0$, mientras que la Bernoulli, por estar restringida al intervalo $[0, 1]$, es $\frac{1}{2}$-subgaussiana. En consecuencia, el bandido es 1-subgaussiano y se pueden aplicar los resultados previos. Veamos qué valor de m debemos escoger para minimizar el remordimiento esperado después de $n = 1000$ rondas.

   Como $\Delta = 0.6 - 0.5 = 0.1$, la elección de m que minimiza la cota C_{EP2} es:

$$m_{Teor}(1000, 0.1) = \max\left\{0, \left\lceil \frac{4}{0.1^2} \log\left(\frac{1000 \cdot 0.1^2}{4}\right)\right\rceil\right\} = 367$$

   Calculemos ahora el valor m_{Opt} que minimiza el verdadero remordimiento. Asumimos que en caso de empate $\hat{\mu}_1(2m) = \hat{\mu}_2(2m)$ se escoge $\hat{a}$ al azar entre los 2 brazos. El remordimiento resultante de aplicar el algoritmo Explora-Primero sobre un bandido de dos brazos se escribe como:

$$R_n = m\Delta + (n-m)\Delta\left(P(\hat{\mu}_2(2m) > \hat{\mu}_1(2m)) + \frac{1}{2}P(\hat{\mu}_2(2m) = \hat{\mu}_1(2m))\right) \quad \forall m \in \left\{0, 1, \ldots, \frac{n}{2}\right\}$$

   Las recompensas $X_1, X_3, \ldots, X_{2m-1}$ obtenidas del primer brazo en la fase de exploración son variables aleatorias i.i.d. Bernoulli(p), así que su suma tendrá distribución binomial con m pruebas y la misma probabilidad de éxito:

$$m\hat{\mu}_1(2m) \sim B(m, p).$$

   En cambio, las recompensas obtenidas del segundo brazo son deterministas. El segundo brazo siempre da recompensa q al ser seleccionado, por lo que

$$m\hat{\mu}_2(2m) \sim qm\,.$$

   Entonces:

$$\begin{aligned} P(\hat{\mu}_2(2m) > \hat{\mu}_1(2m)) &= P(m\hat{\mu}_2(2m) > m\hat{\mu}_1(2m)) = P(B(m,p) < qm) = P(B(m,p) \leq \lceil qm - 1 \rceil) \\ P(\hat{\mu}_2(2m) = \hat{\mu}_1(2m)) &= P(m\hat{\mu}_2(2m) = m\hat{\mu}_1(2m)) = P(B(m,p) = qm) \\ &= P(B(m,p) \leq \lfloor qm \rfloor) - P(B(m,p) \leq \lceil qm - 1 \rceil) \end{aligned}$$

   y estas cantidades las podemos calcular, conocido el valor m, a través de la función de distribución de $B(m, p)$. Denotando F la función de distribución de $B(m, p)$, queda

$$R_n = m\Delta + (n-m)\Delta\left(F(\lceil qm - 1 \rceil) + \frac{1}{2}(F(\lfloor qm \rfloor) - F(\lceil qm - 1 \rceil))\right).$$

   Realizamos el cálculo de R_n para cada $m \in \{0, 1, \ldots, 500\}$ y representamos los resultados en la figura 12.

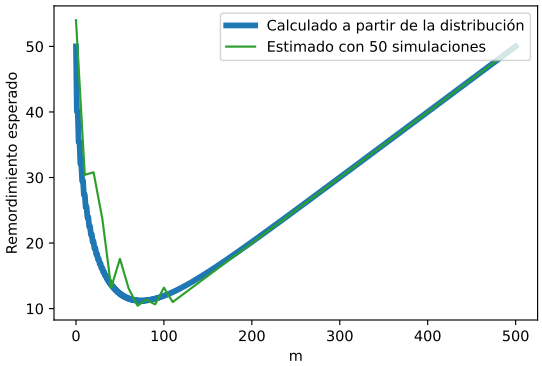


Figura 12: Remordimiento calculado y estimado del algoritmo Explora-Primero en función del parámetro m. El algoritmo se ejecuta sobre un bandido con un brazo Bernoulli(0.6) y otro brazo con distribución degenerada con valor 0.5.

Se obtiene que el valor de m que minimiza R_n es $m_{Opt} = 73$ y dicho valor mínimo es 11.16. En contraposición, $m_{Teor} = 367$ minimiza la cota C_{EP2} que tiene valor

$$C_{EP2\_min} = \min\left\{1000 \cdot 0.1\,,\, 0.1 + \frac{4}{0.1}\left(1 + \max\left\{0, \log\left(\frac{1000 \cdot 0.1^2}{4}\right)\right\}\right)\right\}$$
$$\simeq \min\{100, 0.1 + 40 \cdot 1.4\} = 56.1$$

Esto pone de manifiesto lo imperfecto de nuestro procedimiento para fijar m para según qué distribuciones.

Hemos representado también en la figura 12 la media muestral del pseudo-remordimiento calculada con 50 simulaciones. Pese a ser un número muy reducido de simulaciones, se observa como este remordimiento empírico está bastante próximo al calculado a partir de la distribución, en especial para valores altos de m.

c) Consideramos un bandido con dos brazos de distribuciones $N(0,1)$ y $N(-0.3,1)$ y buscamos minimizar el remordimiento esperado después de $n = 200$ rondas.

Como $\Delta = 0 - (-0.3) = 0.3$, la elección de m que minimiza la cota C_{EP2} es:

$$m_{Teor}(200, 0.3) = \max\left\{0, \left\lceil \frac{4}{0.3^2} \log\left(\frac{200 \cdot 0.3^2}{4}\right)\right\rceil\right\} = 67$$

Calculemos ahora el valor m_{Opt} que minimiza el verdadero remordimiento. En este caso, salvo cuando $m = 0$, las variables aleatorias $\hat{\mu}_1(2m)$ y $\hat{\mu}_2(2m)$ son continuas e independientes, así que su probabilidad de coincidir es nula. En cambio, cuando $m = 0$ coincidirán con probabilidad 1. Entonces, el remordimiento que resulta de aplicar el algoritmo Explora-Primero se escribe como:

$$R_n = \frac{1}{2}n\Delta \qquad \text{si } m = 0$$
$$R_n = m\Delta + (n-m)\Delta(P(\hat{\mu}_2(2m) > \hat{\mu}_1(2m)) \qquad \forall m \in \left\{1, \ldots, \frac{n}{2}\right\}$$

Las recompensas $X_1, X_3, \ldots, X_{2m-1}$ obtenidas del primer brazo en la fase de exploración son variables aleatorias i.i.d $N(0,1)$ así que su suma tiene distribución:

$$m\hat{\mu}_1(2m) \sim N(0, \sqrt{m})$$

Las recompensas $X_2, X_4, \ldots, X_{2m}$ obtenidas del segundo brazo en la fase de exploración son variables aleatorias i.i.d $N(-\Delta, 1)$ así que su suma tiene distribución:

$$m\hat{\mu}_2(2m) \sim N(-m\Delta, \sqrt{m})$$

Entonces, la resta $m\hat{\mu}_1(2m) - m\hat{\mu}_2(2m)$ tendrá distribución $N(m\Delta, \sqrt{2m})$. Por tanto:

$$P(\hat{\mu}_2(2m) > \hat{\mu}_1(2m)) = P(m\hat{\mu}_1(2m) - m\hat{\mu}_2(2m) < 0) = P(N(m\Delta, \sqrt{2m}) < 0)$$

Esta cantidad la podemos calcular, conocido el valor de m, a través de la función de distribución de $N(m\Delta, \sqrt{2m})$, o bien normalizarla:

$$1 - P(N(m\Delta, \sqrt{2m}) \leq 0) = 1 - P\left(\frac{N(m\Delta, \sqrt{2m}) - \Delta}{\sqrt{2m}} \leq -\frac{m\Delta}{\sqrt{2m}}\right) = 1 - P\left(N(0,1) \leq -\frac{m\Delta}{\sqrt{2m}}\right)$$

y utilizar la función de distribución de la $N(0,1)$. Denotando F la función de distribución de $N(0,1)$ queda:

$$R_n = \frac{1}{2}n\Delta \qquad \text{si } m = 0$$
$$R_n = m\Delta + (n-m)\Delta(P(\hat{\mu}_2(2m) > \hat{\mu}_1(2m)) \qquad \forall m \in \left\{1, \ldots, \frac{n}{2}\right\}$$

Realizamos el cálculo de R_n para cada $m \in \{0, 1, \ldots, 100\}$ y representamos los resultados en la figura 13.

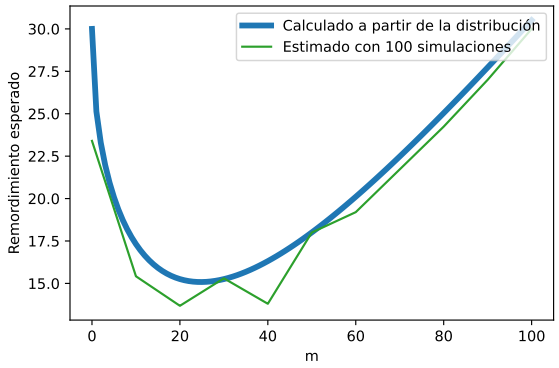


Figura 13: Remordimiento calculado y estimado del algoritmo Explora-Primero en función del parámetro m. El algoritmo se ejecuta sobre un bandido cuyos brazos tienen distribuciones $N(0,1)$ y $N(-0.3,1)$.

Se obtiene que el valor de m que minimiza R_n es $m_{Opt} = 25$ y dicho valor mínimo es 15.08. En contraposición, $m_{Teor} = 67$ minimiza la cota C_{EP2} que tiene valor:

$$C_{EP2\_min} = \min\left\{200 \cdot 0.3\,,\, 0.3 + \frac{4}{0.3}\left(1 + \max\left\{0, \log\left(\frac{200 \cdot 0.3^2}{4}\right)\right\}\right)\right\}$$
$$\simeq \min\{60, 0.3 + 13.33 \cdot 0.65\} \simeq 8.96$$

Aquí la diferencia entre el remordimiento esperado con m_{Opt} y la cota C_{EP2_min} es bastante más reducida que en el caso anterior. Es muy probable que esto se deba a que en el ejemplo b no se está teniendo en cuenta el parámetro σ real de las distribuciones empleadas, sino una cota superior suya (hay un brazo con distribución degenerada y otro $\frac{1}{2}$-subgaussiano y se interpretan ambos como 1-subgaussianos).

7.2.8. Elección de m independiente de Δ: m_{Expl}

Incluye parte de la respuesta al ejercicio 6.5 de [2]

La fórmula explícita que hemos dado para m en (2), para la cual el remordimiento queda acotado por C_{EP2_min}, tiene el inconveniente de que requiere conocer n y Δ para calcular m. Que se necesite el valor

de n no es un gran problema ya que, en el caso de que no se conozca, se puede adaptar la política mediante el truco de la duplicación (véase apéndice A). Sin embargo, es raro que se conozca el valor de Δ y sin él la fórmula explícita para m resulta inútil. Nos gustaría disponer de una fórmula explícita para m en términos únicamente de n. Vamos a ver que podemos obtenerla, pero a cambio de que la cota del remordimiento obtenida sea sustancialmente peor.

Partimos de la acotación

$$R_n \leq C_{EP2}(n, m, \Delta) = m\Delta + n\Delta \exp\left(-\frac{m\Delta^2}{4}\right)$$

Buscamos una cota para el segundo sumando que no dependa de Δ. Para ello, definimos $f : \mathbb{R}^+ \to \mathbb{R}$ tal que:

$$f(x) = nx \exp\left(-\frac{mx^2}{4}\right)$$

que verifica:

$$f(x) > 0 \quad \forall x \in \mathbb{R}^+$$

$$\lim_{x\to 0} f(x) = \lim_{x\to +\infty} f(x) = 0$$

Busquemos el valor máximo de f en $\mathbb{R}^+$:

$$f'(x) = n \exp\left(-\frac{mx^2}{4}\right) - \frac{nmx^2}{2} \exp\left(-\frac{mx^2}{4}\right) = \left(1 - \frac{mx^2}{2}\right) n \exp\left(-\frac{mx^2}{4}\right)$$

$$f'(x) = 0 \iff 1 - \frac{mx^2}{2} = 0 \iff x = \sqrt{\frac{2}{m}}$$

Entonces, el valor máximo de f es:

$$f\left(\sqrt{\frac{2}{m}}\right) = n\sqrt{\frac{2}{m}} \exp\left(-\frac{1}{2}\right)$$

Por tanto, tenemos:

$$R_n \leq m\Delta + n\Delta \exp\left(-\frac{m\Delta^2}{4}\right) = m\Delta + f(\Delta) \leq m\Delta + n\sqrt{\frac{2}{m}} \exp\left(-\frac{1}{2}\right) =: C_{EP3}(n, m, \Delta)$$

Buscamos ahora elegir m de la forma $m = n^\alpha$ para minimizar la complejidad asintótica de la cota. Se tiene:

$$C_{EP3}(n, n^\alpha, \Delta) = \Delta n^\alpha + \sqrt{2} \exp\left(-\frac{1}{2}\right) n^{1-\frac{\alpha}{2}}$$

Por ser $\alpha \mapsto \alpha$ y $\alpha \mapsto 1 - \frac{\alpha}{2}$ funciones lineales, una creciente y la otra decreciente, el valor de α que minimiza $\max(\alpha, 1 - \frac{\alpha}{2})$ es el que verifica $\alpha = 1 - \frac{\alpha}{2}$, es decir, $\alpha = \frac{2}{3}$. Por tanto, con la elección

$$m_{Expl}(n) := \lceil n^{2/3} \rceil$$

una cota con el menor orden asintótico posible partiendo de C_{EP3} es:

$$C_{EP3\_min}(n, \Delta) := C_{EP3}(n, n^{2/3}, \Delta) = \left(\Delta + \sqrt{2} \exp\left(-\frac{1}{2}\right)\right) \lceil n^{2/3} \rceil$$

Esta cota está en $\Theta(n^{2/3})$, sustancialmente peor que C_{EP2_min}, que estaba en $O(n^{1/2})$. Sin embargo, tiene la ventaja de que se ha obtenido con un valor de m que solo requiere el conocimiento de n y no el de Δ, así que resultará de mucha utilidad para aquellas situaciones (las más habituales) en que Δ es desconocido para el agente. Obsérvese, sin embargo, que el valor de la cota C_{EP3_min} sí que depende de Δ, así que en estas situaciones en que no se conoce Δ no se conocerá la cuantía de la cota.

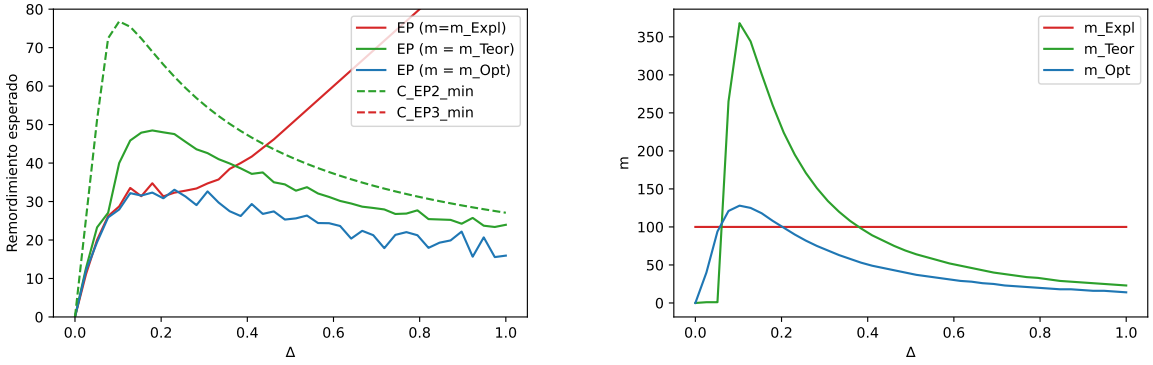
7.2.9. Comparación entre las cotas

Incluye parte de la respuesta al ejercicio 6.9 de [2]

En este apartado vamos a comparar entre sí las distintas cotas del remordimiento y elecciones del parámetro m para el algoritmo Explora-Primero que hemos visto a lo largo de la sección. Para ilustrarlo, utilizaremos como ejemplo un bandido de 2 brazos con distribuciones $N(0,1)$ y $N(-\Delta,1)$ para cierto $\Delta \in (0,1)$, de manera que el mejor brazo es siempre el primero, y como horizonte tomaremos $n = 1000$ rondas. Para distintos valores de $\Delta \in (0,1)$ calculamos:

- El valor m_{Opt} de m que tiene una probabilidad máxima de minimizar el remordimiento. En este caso, por ser la distribución de los brazos conocidas y pertenecientes a la familia normal, se puede calcular de forma determinista (en un caso real sería imposible).
- El valor m_{Teor} de m que minimiza la cota C_{EP2} y se calcula en términos de n y Δ, y la cota mínima C_{EP2_min}.
- El valor m_{Expl} de m que minimiza la cota C_{EP3} y se calcula únicamente en términos de n, y la cota mínima.
- Los remordimientos obtenidos al ejecutar el algoritmo Explora-Primero con $m = m_{Opt}$, m_{Teor} y m_{Expl}, estimados con 1000 simulaciones.

Los resultados se muestran en la figura 14.



(a) Remordimiento y cotas del mismo. **(b)** Valor escogido para m.

Figura 14: Comparativa entre las distintas formas de elegir m en el algoritmo Explora-Primero. El algoritmo se aplica sobre un bandido de dos brazos de distribución $N(0,1)$ y $N(-\Delta,1)$ para distintos valores de $\Delta \in (0,1)$.

Examinamos primero la figura 14a. Como cabía esperar, la ejecución con $m = m_{Opt}$ obtiene el menor remordimiento sea cual sea el valor de Δ. No obstante, se aprecia bastante fluctuación, pese al elevado número de simulaciones, lo cual nos hace pensar que quizás si utilizáramos este valor de m para un número reducido de ejecuciones y tuviéramos "mala suerte" el remordimiento obtenido distaría bastante de ser tan bueno como el que se ve aquí.

La ejecución con $m = m_{Teor}$ obtiene un remordimiento mayor que la anterior, pero la diferencia tampoco es muy grande, en especial para Δ próximo a 1. Además, habiéndose utilizado el mismo número de simulaciones que para el caso anterior, el remordimiento presenta muchas menos fluctuaciones, al menos para Δ reducido. La cota C_{EP2_min} proporciona un "techo" al remordimiento que puede alcanzar esta ejecución y, aunque es poco ajustada para Δ muy próximo a 0, mejora mucho para valores algo mayores de Δ, proporcionando una aproximación bastante certera del pseudo-remordimiento. Por consiguiente, no parece que haber hecho esta elección de m suponga una desventaja real frente a la elección óptima y, dado que la elección

óptima es imposible en la práctica, esta alternativa podría resultar prometedora. Sin embargo, no olvidemos que conocer los saltos de suboptimalidad tampoco es una exigencia muy factible en un caso real. Al fin y al cabo, disponer de los saltos de suboptimalidad asociados a cada brazo ya resuelve de antemano el problema que se pretende resolver utilizando algoritmos de bandidos, por lo que elegir m por este medio tampoco resulta útil.

La ejecución con $m = m_{Expl}$ es la que obtiene un remordimiento mayor, con diferencia, para valores altos de Δ, pero resulta sorprendentemente buena para valores pequeños de Δ. En concreto, prácticamente iguala el remordimiento óptimo para $\Delta \in (0, 0.3)$, además de tener pocas fluctuaciones. No obstante, a partir de $\Delta = 0.3$, el remordimiento comienza a crecer linealmente con Δ y se vuelve mucho peor que los otros dos. La cota C_{EP3_min} de este remordimiento, por su parte, no parece resultar nada útil: es demasiado poco ajustada como para darnos ninguna información. Pese a estos inconvenientes, el valor m_{Expl} resulta mucho más útil en la práctica que los anteriores. Por un lado, hay muchas ocasiones en que el valor de n será conocido y, por tanto, m_{Expl} se podrá calcular. Por otro lado, aunque n sea desconocido, si se emplea el truco de la duplicación se puede utilizar la fórmula de m_{Expl} para calcular m en cada una de las sucesivas aplicaciones del algoritmo, con lo que se podrá obtener un remordimiento asintóticamente equivalente al que se obtendría si se conociera n y se tomara $m = m_{Expl}$ (véase apéndice A).

La figura 14b complementa lo anteriormente expuesto y aclara algunos aspectos relativos a lo que se veía en la otra figura. Se observa que m_{Teor} resulta ser mucho mayor que m_{Opt} cuando Δ es pequeño. Esto explica, por un lado, que la distancia entre ambos remordimientos sea mayor en esta etapa y, por otro, que las fluctuaciones del remordimiento con $m = m_{Teor}$ sean mucho menores (vimos en 7.2.2 que un valor de m mayor reducía la varianza del pseudo-remordimiento y, con ello, las diferencias en muestras sucesivas extraídas del mismo). El valor de m_{Expl}, por su parte, al calcularse únicamente en términos de n no cambia al variar Δ. Su nula adaptación a Δ explica que se comporte mal para ciertos valores de este, pero hay otros, como $\Delta = 0.2$, en los que m_{Expl} coincide con m_{Opt}, proporcionando un valor óptimo del remordimiento (mucho mejor que el que proporciona $m = m_{Teor}$ para este mismo Δ). Sin embargo, para $\Delta > 0.4$, m_{Expl} es muy elevado en relación con m_{Teor} y m_{Opt}, ya que conserva su valor mientras estos últimos disminuyen notablemente cuando Δ tiende a 1. Por ende, en estos casos elegir $m = m_{Expl}$ hace que el algoritmo peque de una excesiva exploración, lo que le lleva al gran remordimiento que se observa en la otra figura.

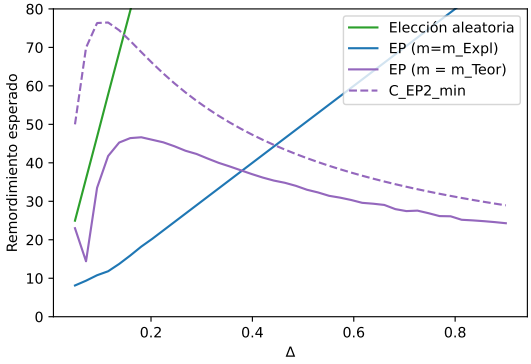
En suma, si bien las elecciones $m = m_{Opt}$ y $m = m_{Teor}$ proporcionan remordimientos aceptables para todos los valores de Δ, no son aplicables en la práctica porque requieren demasiada información de la instancia para su cómputo. En cambio, la opción $m = m_{Expl}$ sí es aplicable, pero solo funciona adecuadamente para algunos valores de Δ. Además, no tenemos forma de predecir para qué valores funciona bien de manera que no se sabe en qué casos es razonable aplicar esta elección del parámetro y en qué casos incurre en un remordimiento inasumible.

7.2.10. Remordimiento para $m = m_{Expl}$

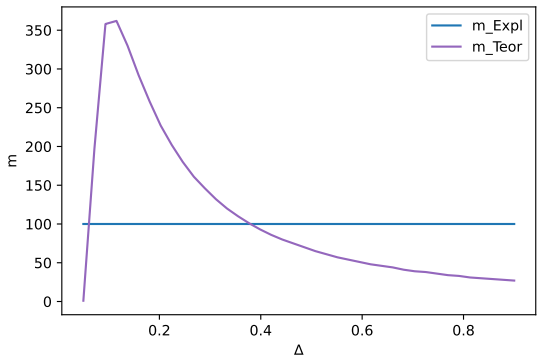
Una hipótesis razonable, a la vista de la figura 14, es que la elección $m = m_{Expl}$ dé buenos resultados en todos los casos en que el salto de suboptimalidad de Δ sea reducido.

Veámoslo con otro experimento. Consideremos un bandido de 2 brazos con distribuciones Bernoulli(0.95) y Bernoulli($0.95 - \Delta$) y un horizonte de $n = 1000$ rondas. Para distintos valores de $\Delta \in (0, 0.9)$ ejecutamos el algoritmo Explora-Primero con $m = m_{Expl}$ y calculamos el remordimiento en que se incurre promediando los resultados de 1000 simulaciones. Obtenemos, de la misma forma, también el remordimiento para la ejecución de Explora-Primero con $m = m_{Teor}$ y el remordimiento de la estrategia uniformemente aleatoria y calculamos la cota C_{EP2_min}, para tener otras métricas con las que comparar la primera. Representamos los resultados en las figuras 15 y 16.

En la figura 15 se observa un comportamiento muy similar al que ya vimos en la figura 14, de manera que cabe decir que lo comentado entonces sigue siendo válido aunque varíe la distribución de los brazos del bandido. En particular, la elección $m = m_{Expl}$ da muy buenos resultados (mejores que $m = m_{Teor}$) para $\Delta \in (0, 0.3)$ y considerablemente peores conforme aumenta el valor de Δ. Sobre esta figura también cabe comentar que la estrategia de selección uniformemente aleatoria del brazo a ejecutar comete un remordimiento

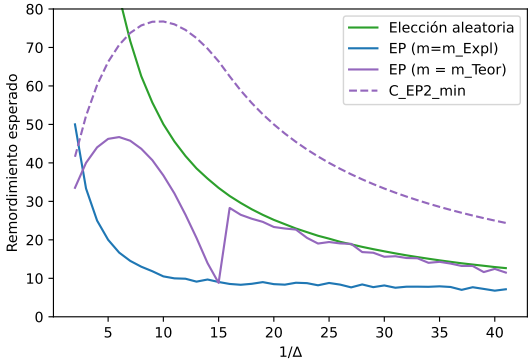


(a) Remordimiento y cotas del mismo.

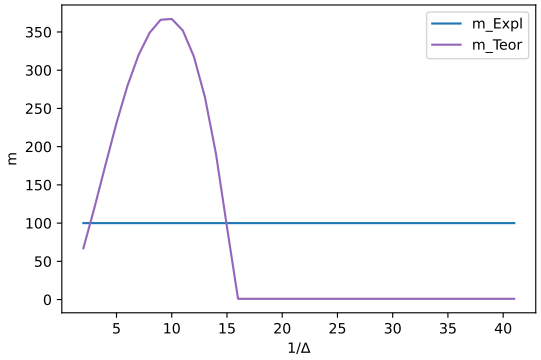


(b) Valor escogido para m.

Figura 15: Comportamiento de Explora-Primero con la elección $m = m_{Expl}$ del algoritmo Explora-Primero para valores grandes del salto de suboptimalidad Δ. El algoritmo se aplica sobre un bandido Bernoulli.



(a) Remordimiento y cotas del mismo.



(b) Valor escogido para m.

Figura 16: Comportamiento de Explora-Primero con la elección $m = m_{Expl}$ del algoritmo Explora-Primero para valores pequeños del salto de suboptimalidad Δ. El algoritmo se aplica sobre un bandido Bernoulli.

mucho más elevado que cualquiera de las otras dos.

La figura 16 refleja un panorama diferente. Como ya tuvimos ocasión de comprobar en 7.2.5 para la distribución uniforme (y, a la vista de estos resultados, es igualmente cierto para esta), el remordimiento de Explora-Primero para Δ muy pequeño es prácticamente el mismo que el de la política aleatoria. Esto ocurre porque cuando $\Delta < \frac{2}{\sqrt{n}}$ (que en este caso es 0.063) se da

$$\log\left(\frac{n\Delta^2}{4}\right) < 0$$

con lo que $m_{Teor} = 1$, que no es en general buena elección de m. Con ella, la exploración es casi nula, así que no es de extrañar que el remordimiento que se obtiene sea casi el de la estrategia aleatoria.

En cambio, independientemente de Δ siempre se tiene $m_{Expl} = \lceil n^{2/3} \rceil$ (que en este caso es $m_{Expl} = 100$). Gracias a esto, el remordimiento en que incurre el algoritmo Explora-Primero con $m = m_{Expl}$ resulta ser considerablemente inferior al de $m = m_{Teor}$ para los valores de pequeños de Δ. En concreto, se cumple para todo $\Delta < 0.3$, aunque la diferencia es especialmente elevada para $\Delta \simeq 0.07$. Reduciendo Δ a partir de 0.07, la diferencia entre los remordimientos de ambas estrategias disminuye progresivamente, con lo que cabe esperar que para Δ suficientemente pequeño todas las estrategias acaben desempeñando básicamente igual.

Este es un ejemplo de lo que ocurre en un caso concreto, pero dadas las similitudes entre estos resultados y

los obtenidos en los experimentos anteriores pensamos que tiene sentido concluir que la elección de $m = m_{Expl}$ es bastante apropiada para el caso en que Δ es pequeño, al menos en comparación con las alternativas disponibles.

7.2.11. Conclusiones

Concluimos que, si bien el algoritmo Explora-Primero tiene un remordimiento muy reducido si se escoge el parámetro m a partir de un conocimiento perfecto de la instancia, en la práctica esto no es útil porque la razón por la que requerimos un algoritmo de bandidos es precisamente la falta de conocimiento de la instancia. En cambio, si se elige m mediante alguna fórmula que no requiera tanto conocimiento de la instancia, como puede ser $m = m_{Expl}$, en muchas ocasiones no se obtienen buenos resultados. En concreto, Explora-Primero con $m = m_{Expl}$ solo nos es útil cuando el salto de suboptimalidad Δ es reducido.

Todo esto nos lleva a mirar más allá de esta simple estrategia en busca de un algoritmo más sofisticado. Veremos en la siguiente sección como UCB soluciona, al menos parcialmente, muchos de los problemas de Explora-Primero.

8. Cota superior de confianza

El algoritmo Exlora-Primero, que hemos visto en la sección anterior, proporciona buenos resultados para cualquier problema de bandidos estocásticos siempre que se escoja el valor de m adecuado para la instancia enfrentada. Sin embargo, si m no se escoge bien el resultado puede ser desastroso.

Dado que el problema es, en general, desconocido a priori para el agente, no es viable pretender elegir m de manera óptima: a lo más que podemos aspirar en un caso real es a tomar $m = m_{Expl} = \lceil n^{2/3} \rceil$ y ya hemos visto que esto funciona bien cuando el salto de suboptimalidad es reducido, pero muy mal cuando no lo es. Por tanto, conviene contar con un algoritmo cuyo buen funcionamiento no esté tan ligada a la elección de los parámetros del mismo o que, si lo está, baste el conocimiento de n para escogerlo adecuadamente. El algoritmo de la Cota Superior de Confianza (UCB), que veremos en esta sección, será la solución que buscamos.

8.1. Principio de optimismo ante la incertidumbre

La realización de predicciones sobre lo que ocurrirá en el futuro es esencial para la toma de decisiones. A priori, podríamos pensar que lo ideal es que las inferencias realizadas sean lo más libres de sesgo posibles. Esto, sin embargo, situaría las alternativas nunca probadas, de las que no se dispone de información concreta, en torno a la media de las alternativas ya probadas. Al fin y al cabo, si tenemos que estimar lo más objetivamente posible lo buena que es una opción de la que no disponemos de realimentación, es razonable pensar que no será mucho mejor ni peor que las otras opciones (precisamente si las estamos considerando como grupo es porque tienen algo en común y puede que este rasgo sea una de las pocas informaciones de las que dispongamos de la opción desconocida).

Por ejemplo, si tenemos que valorar un restaurante italiano al que ya hemos ido nos basaremos en las observaciones e impresiones que obtuvimos las anteriores veces que comimos en él. Pero si tenemos que valorar un restaurante italiano al que nunca hemos ido ni hemos recibido reseñas de él lo puntuaríamos como un restaurante italiano de calidad media: el hecho de que sea un restaurante italiano da un motivo para pensar que su calidad estará en torno a la del restaurante italiano medio y este es el único hecho que podemos tener en cuenta sobre el mismo. Sin embargo, ¿qué ocurriría si en estas circunstancias tuviéramos que escoger un restaurante italiano al que acudir? Si asignamos a los restaurantes desconocidos una valoración media, siempre perderán frente al mejor restaurante italiano que hemos probado. De esta manera, los restaurantes desconocidos seguirán siendo desconocidos y nunca sabremos si realmente tenían un calidad media entre los restaurantes italianos o eran mucho mejores.

Como ya hemos recalcado varias veces desde que comenzamos el estudio de los bandidos, dedicarse por entero a la opción que ha dado mejores resultados cuando hay opciones sin explorar no es en absoluto conveniente para el largo plazo. No obstante, una elección libre de sesgo parece empujarnos precisamente a esto, desincentivando la exploración cuando ya se dispone de una opción conocida positiva (el famoso "*más vale lo malo conocido que lo bueno por conocer*").

Llegamos, pues, a que una decisión, para ser acertada, ha de tener un cierto sesgo hacia las opciones desconocidas. Existen estudios psicológicos que sugieren que este sesgo está presente de manera natural en la mente humana y de otros animales. Lo que sabemos es que si queremos que nuestros algoritmos de bandidos se comporten adecuadamente, han de incorporar este sesgo en su justa medida.

Nos basaremos en el principio de optimismo ante la incertidumbre, que establece que debemos actuar como si el entorno fuera lo mejor que permite la información de que disponemos. Asignaremos a cada brazo una cota superior de confianza, que será con probabilidad alta una cota superior de la media de su distribución asociada y elegiremos, en cada ronda, el brazo que presente una cota mayor. Esta idea es la base del algoritmo de la Cota Superior de Confianza (en inglés *Upper Confidence Bound*, UCB). En particular, los brazos que aún no han sido explorados tendrán $+\infty$ como cota, lo que les dará prioridad frente a los ya probados.

8.2. El algoritmo

Para cada $t \in \{1, \ldots, n\}$ y cada $a \in \mathcal{A}$, $\mathrm{UCB}_a(t-1)$ denotará la **cota superior de confianza** del brazo a al comienzo de la ronda t.

En la ronda t el agente habrá observado $T_a(t-1)$ muestras del brazo a y la media de las recompensas obtenidas de este brazo será $\hat{\mu}_a(t-1)$. ¿Cómo deberíamos elegir $\mathrm{UCB}_a(t-1)$? Por supuesto, simplemente escogiendo $\mathrm{UCB}_a(t-1) = +\infty$ sabríamos que es una cota superior de la media μ_a de la distribución asociada al brazo con absoluta certeza. Sin embargo (y aunque parezca paradójico), necesitamos que la cota superior de confianza, además de ser optimista, sea lo bastante ajustada como para permitirnos decidir entre los brazos en cada ronda. Quién domina y en qué medida en esta pugna entre optimismo y realismo lo decide un parámetro $\delta \in [0, 1]$ del algoritmo.

El **nivel de confianza** δ es una cota superior de la probabilidad de que la cota superior de confianza asociada a un brazo cualquiera $a \in \mathcal{A}$ al comienzo de una ronda cualquiera t sea menor o igual que la media de la distribución asociada al brazo a. Es decir, en toda ronda $t \in \{1, \ldots, n\}$ y para todo brazo $a \in \mathcal{A}$ se ha de verificar:

$$P(\mathrm{UCB}_a(t-1) \leq \mu_a) \leq \delta\,.$$

Analicemos cómo elegir $\mathrm{UCB}_a(t-1)$ para $a \in \mathcal{A}$ y $t \in \{1, \ldots, n\}$ arbitrarios. Basándonos en la información de que disponemos en la ronda t sobre el brazo a, es decir, las recompensas obtenidas de este brazo en las $t-1$ primeras rondas, parece claro que se ha de escoger $\mathrm{UCB}_a(t-1)$ mayor que $\hat{\mu}_a(t-1)$, que es nuestra mejor estimación de μ_a. Será, pues,

$$\mathrm{UCB}_a(t-1) = \hat{\mu}_a(t-1) + \varepsilon$$

para cierto $\varepsilon > 0$. Por tanto, se tendrá

$$P(\hat{\mu}_a(t-1) \leq \mu_a - \varepsilon) \leq \delta\,.$$

Dado que $E[\hat{\mu}_a(t-1)] = \mu_a$, desde un punto de vista probabilístico δ no es más que una cota de la probabilidad de la cola izquierda de la variable aleatoria $\hat{\mu}_a(t-1)$. Podemos, pues, obtener una expresión de ε en función de δ utilizando alguna desigualdad de concentración que se aplique sobre esta variable. El hecho de que se pueda aplicar una desigualdad más o menos fuerte dependerá de las asunciones que hagamos sobre las distribuciones de los brazos.

Aquí supondremos, como acostumbramos, que las recompensas de todos los brazos siguen una distribución 1-subgaussiana. Recordemos de 5.2 que si $Y_1, \ldots, Y_n \in \mathrm{subG}(1)$ son i.i.d. de media μ se verifica

$$P\left(\frac{1}{n}\sum_{t=1}^{n} Y_t \leq \mu - \varepsilon\right) \leq \exp\left(-\frac{n\varepsilon^2}{2}\right)$$

para todo $\varepsilon > 0$. Dado $\delta \in (0, 1)$ y eligiendo ε como:

$$\varepsilon = \sqrt{\frac{2\log\left(\frac{1}{\delta}\right)}{n}}\,,$$

la desigualdad toma la forma:

$$P\left(\frac{1}{n}\sum_{t=1}^{n} Y_t \leq \mu - \sqrt{\frac{2\log\left(\frac{1}{\delta}\right)}{n}}\right) \leq \delta\,.$$

¿Podemos aplicar esto en el caso en que los Y_t son las recompensas asociadas a un brazo $a \in \mathcal{A}$? Puesto que $T_a(t-1)$ no es un escalar, sino una variable aleatoria, $\hat{\mu}_a(t-1)$ no es estrictamente una media muestral. La intuición nos dice, sin embargo, que se comportará de modo semejante, de manera que se tendrá

$$P\left(\hat{\mu}_a(t-1) \leq \mu_a - \sqrt{\frac{2\log\left(\frac{1}{\delta}\right)}{T_a(t-1)}}\right) \lessapprox \delta\,, \tag{4}$$

Algoritmo 3 UCB

parameters

δ: nivel de confianza

end parameters

initialize

$T_a \leftarrow 0$ for a in $\mathcal{A}$

$\hat{\mu}_a \leftarrow 0$ for a in $\mathcal{A}$

$\mathrm{UCB}_a \leftarrow +\infty$ for a in $\mathcal{A}$

end initialize

for t in $1, \ldots, n$ **do**

$a_{\mathrm{UCB}} \leftarrow \mathrm{argmax}_{a \in \mathcal{A}} \mathrm{UCB}_a$

$x \leftarrow \mathrm{ejecutar}(a_{\mathrm{UCB}})$

$\hat{\mu}_a \leftarrow \frac{1}{T_a+1}(T_a \hat{\mu}_a + x)$

$T_a \leftarrow T_a + 1$

$\mathrm{UCB}_{a_{\mathrm{UCB}}} \leftarrow \hat{\mu}_a + \sqrt{\frac{2\log(1/\delta)}{T_a}}$

end for

aunque esto no se sigue directamente de los resultados que hemos visto. En lo que a proporcionar intuiciones sobre por qué el algoritmo funciona se refiere podemos tomar (4) como suficientemente cierto. Una solución más rigurosa, aunque más compleja, se dará en mi TFG de Matemáticas cuando se demuestren formalmenete las garantías de este algoritmo.

Así pues, partiendo de que se tiene (4), o lo que es lo mismo

$$P\left(\hat{\mu}_a(t-1) + \sqrt{\frac{2\log\left(\frac{1}{\delta}\right)}{T_a(t-1)}} \leq \mu_a\right) \lessapprox \delta\,,$$

podemos escoger $\mathrm{UCB}_a(t-1)$ como $\hat{\mu}_a(t-1) + \sqrt{\frac{2\log\left(\frac{1}{\delta}\right)}{T_a(t-1)}}$ siempre que $T_a(t-1) \neq 0$. En el caso $T_a(t-1) = 0$ no se tiene ninguna información sobre el brazo a, dado que no ha sido elegido hasta el momento, así que el principio de optimismo ante la incertidumbre nos lleva a darle prioridad máxima, es decir, escoger $\mathrm{UCB}_a(t-1) = +\infty$. De esta manera, la cota superior de confianza del brazo a al comienzo de la ronda t, $\mathrm{UCB}_a(t-1)$, queda definida como:

$$\mathrm{UCB}_a(t-1) = \begin{cases} +\infty & \text{si } T_a(t-1) = 0 \\ \hat{\mu}_a(t-1) + \sqrt{\frac{2\log\left(\frac{1}{\delta}\right)}{T_a(t-1)}} & \text{en caso contrario} \end{cases}$$

Esta definición la aplicaremos para todo $a \in \mathcal{A}$ y todo $t \in \{1, \ldots, n\}$.

Entonces, en cada ronda $t \in \{1, \ldots, n\}$ el algoritmo UCB ejecutará la acción a_{UCB} que maximice las cotas superiores de confianza $\mathrm{UCB}_a(t-1)$ de cada brazo:

$$a_{\mathrm{UCB}} := \mathrm{argmax}_{a \in \mathcal{A}}\{\mathrm{UCB}_a(t-1)\}\,.$$

En el caso de que haya varios brazos $a \in \mathcal{A}$ con $\mathrm{UCB}_a(t-1)$ máximo, lo mismo que argumentamos en 7.1.1 muestra que es prácticamente irrelevante la forma de desempatar. De hecho, en este caso es incluso menos

relevante, ya que una elección mala será corregida para las rondas sucesivas en cuanto empiece a tener impacto en las recompensas. Al igual que con el algoritmo Explora-Primero, evitaremos hacer ninguna asunción sobre el modo de desempate de manera que los resultados sean válidos sin importar cuál se implemente. Para las implementaciones que realicemos nos decantaremos por una elección aleatoria de a_{UCB}, con probabilidad uniforme entre todos los brazos $a \in \mathcal{A}$ con $\text{UCB}_a(t-1)$ máximo. El esquema de UCB se muestra en el Algoritmo 3.

8.3. Elección del nivel de confianza

Elegir el nivel de confianza δ con el que ejecutar el algoritmo UCB es un asunto delicado. Cuanto menor sea δ, menor será la tolerancia a que la cota superior de confianza de un brazo a resulte ser, en un momento dado, menor que la media de la distribución asociada al brazo, de manera que el algoritmo será necesariamente más *optimista*. Pero no nos confundamos con la terminología: será más optimista porque estará siendo más fiel al principio de optimismo ante la incertidumbre, pero no por ello tendrá una baja tolerancia a fallos, sino al contrario. En este contexto, el *optimismo* va ligado a la exploración y el *realismo* va ligado a la explotación.

Una política *realista* (es decir, con un valor de δ elevado) se dedicará principalmente a los brazos que hayan dado buenos resultados en las primeras rondas, ya que en este caso la cota superior de confianza de cada brazo estará próxima a la media de recompensas obtenidas de este. Si determina correctamente cuál es el mejor brazo, es decir, si el brazo con $\hat{\mu}_a$ máximo coincide con el brazo con μ_a máximo, esta estrategia obtendrá, por tanto, un remordimiento muy bajo. Sin embargo, en el caso de no determinarlo correctamente, el riesgo de que el algoritmo se centre en un brazo subóptimo por el resto de rondas es muy alto. En el caso extremo en que $\delta = 1$ el algoritmo probará inicialmente una vez cada uno de los brazos y en cada una de las rondas restantes se decantará por el brazo cuya media muestral de recompensas sea mayor.

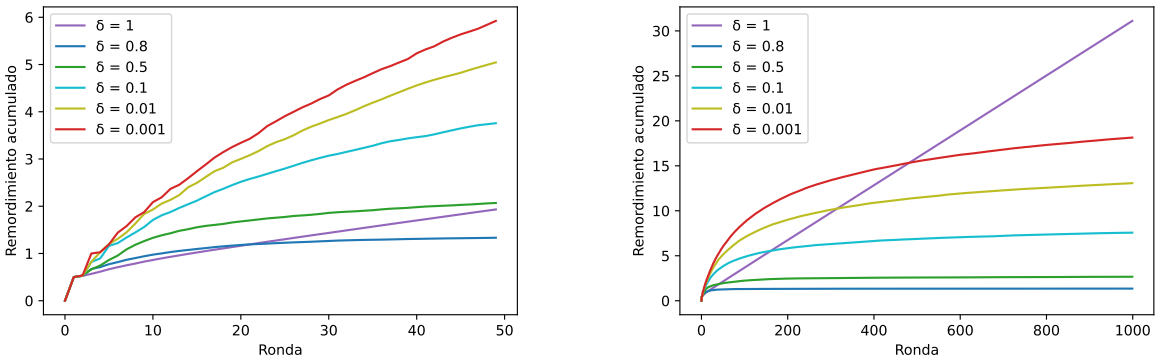
Una política *optimista* (es decir, con un valor de δ reducido), por su parte, estará mucho más centrada en la exploración. Al situar la cota superior de confianza de los brazos que han sido probados pocas veces muy por encima de su media, fomenta que sean elegidos. De esta manera, el riesgo de que el algoritmo se acabe centrando en brazos subóptimos será muy bajo: para que esto ocurriera la cota superior de confianza del brazo óptimo debería ser menor que su media real, lo cual ocurre probabilidad δ. No obstante, lo que sí puede ocurrir, especialmente si el número de n de rondas no es muy grande, es que el algoritmo emplee demasiadas en explorar y no termine de centrarse en el brazo óptimo.

Todo esto nos recuerda a algo que ya hablamos en la introducción a los bandidos, que para lograr el equilibrio entre exploración y explotación que maximice la recompensa total acumulada (o, equivalentemente, minimice el remordimiento total acumulado) siempre se ha de ajustar el esfuerzo destinado a exploración/explotación en función del número n de rondas. En concreto, en 4.2 vimos cómo ningún número de exploraciones fijo proporcionaba ni tan siquiera medianamente buenos resultados para cualquier número n de rondas. Por ello, todas las elecciones de m que propusimos para el algoritmo Explora-Primero dependían de n o requerían aplicar el truco de la duplicación (apéndice A)

Reformulamos ahora el experimento de 4.2 para comprobar cómo se acumula el remordimiento cuando se usa el algoritmo UCB para distintos valores de δ independientes de n. Tenemos, pues, un bandido de dos brazos con distribuciones Bernoulli(p_1) y Bernoulli(p_2). Veamos, en primer lugar, qué ocurre cuando p_1 y p_2 son bastante distintos, pongamos $p_1 = 0.7$ y $p_2 = 0.2$. Ejecutamos el algoritmo UCB para distintos valores de $\delta \in (0, 1]$. Realizamos esto para $n = 50$ rondas y para $n = 1000$ rondas. Los resultados se muestran en la figura 17.

En la figura 17a se puede observar como la política con $\delta = 0.8$ resulta clara vencedora en el caso $n = 50$. Esta estrategia se corresponde con una tendencia bastante fuerte a la explotación de la mejor opción encontrada pero sin descartar del todo la exploración. Destaca también el hecho de que, para $n = 50$ e inferior, la política trivial con $\delta = 1$ tiene un desempeño bastante bueno: es la mejor de todas las estrategias probadas para $n < 20$ y la segunda mejor para $n = 50$, a escasa distancia de la óptima. En cambio, otras estrategias, como la de tomar $\delta = 0.001$ obtienen un remordimiento 6 veces superior.

Sin embargo, el valor final del remordimiento no es lo único que merece la pena analizar en esta gráfica. Cabe ver también la variación de la pendiente de cada una de las curvas. Se observa que todas las estrategias,



(a) $n = 50$. **(b)** $n = 1000$.

Figura 17: Remordimiento acumulado en la ejecución del algoritmo UCB para distintos valores del parámetro δ. El algoritmo se aplica sobre un bandido de dos brazos cuyas distribuciones son Bernoulli(0.7) y Bernoulli(0.2).

salvo la de $\delta = 1$, reducen, progresivamente y de manera notable, el ritmo al que incrementan el remordimiento acumulado conforme avanzan las rondas. En otras palabras, el remordimiento acumulado asociado a estas estrategias se incrementa de forma sublineal respecto a las rondas. El hecho de que la política con $\delta = 1$ no lo haga anuncia que el comportamiento de la misma será mucho peor en cuanto el número total n de rondas aumente un poco. Esto se constata en la figura 17b.

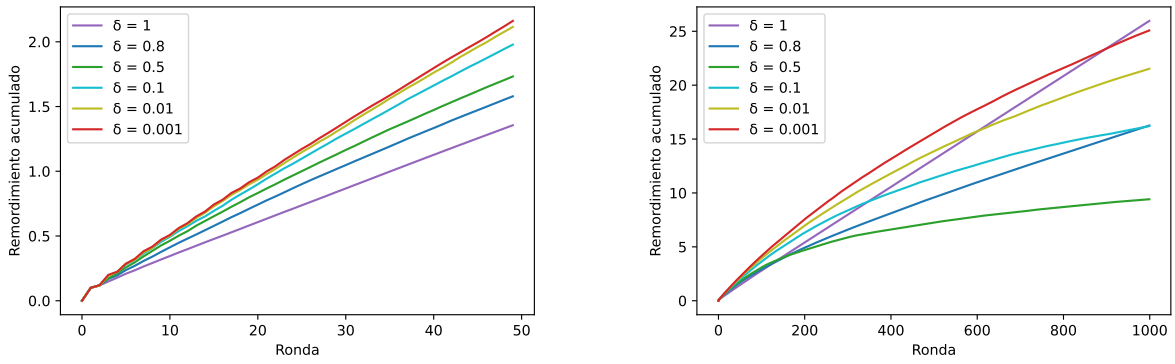
En efecto, en la figura 17b se muestra un panorama muy distinto para la estrategia con $\delta = 1$. Mientras que ninguna de las demás estrategias acaba alcanzando un remordimiento final superior a 20, el de esta supera 40. Esto es porque el resto de estrategias realizan un "esfuerzo" de aprendizaje que, a cambio de perjudicarlas en cierta medida en las primeras rondas, les permite ir puliendo paulatinamente la información con la que trabajan obteniendo mucho mejores resultados en el largo plazo. En cambio, con $\delta = 1$ el aprendizaje es prácticamente nulo, así que no le supone ninguna ventaja aumentar el número de rondas pues desempeña prácticamente igual en las primeras que en las últimas.

De la figura 17b también cabe señalar que la estrategia con $\delta = 0.8$ sigue siendo la dominante con $n = 1000$ rondas y no parece que fuera a ser superada por otra aunque se realizara un número mayor. Esto muestra como en ocasiones un valor elevado de δ puede obtener un comportamiento sorprendentemente bueno. Desgraciadamente, en general no seremos capaces de predecir esto de antemano y veremos que los resultados teóricos nos sugerirán un valor de δ mucho más reducido para este caso. En la práctica, nos acabaremos conformando con un valor subóptimo de δ, pero veremos que esto será más que suficiente para obtener resultados mejores que el algoritmo Explora-Primero que hemos usado hasta ahora.

Ahora repetimos el experimento con p_1 y p_2 más próximos, para comprobar cuánto afecta este hecho. Escogemos $p_1 = 0.5$ y $p_2 = 0.4$ y realizamos las pruebas, nuevamente, para $n = 50$ y $n = 1000$. Los resultados se muestran en la figura 18.

En la figura 18a vemos que, a diferencia de en el caso anterior, el crecimiento de los remordimientos acumulados de todas las estrategias parece lineal en lugar de sublineal. De hecho, en este caso la política con $\delta = 1$ es la mejor por un amplio margen y, dado que aparenta ser la recta con menor pendiente, parece que seguirá siendo la mejor sin importar cuántas rondas hagamos. (Hemos de admitir que cuando observamos este resultado pensamos que se había de deber a un error de programación, dado que no parecía coherente con la intuición que teníamos sobre el algoritmo el hecho de que la estrategia con $\delta = 1$ venciera a todas las demás para *cualquier* número de rondas).

Por fortuna, al aumentar el número n de rondas de 50 a 1000 (como se aprecia en la figura 18b) se desvanece la ilusión de linealidad: se observa la reducción con el tiempo de la pendiente de todas las políticas salvo aquella asociada a $\delta = 1$, que al ser la que más crece pierde su posición dominante. En este caso, la estrategia que muestra una mayor reducción de su pendiente es aquella con $\delta = 0.5$ y, por ello, es la que se



(a) $n = 50$. **(b)** $n = 1000$.

Figura 18: Remordimiento acumulado en la ejecución del algoritmo UCB para distintos valores del parámetro δ. El algoritmo se aplica sobre un bandido de dos brazos cuyas distribuciones son Bernoulli(0.5) y Bernoulli(0.4).

sitúa en una mejor posición y con una amplia diferencia al final de las n rondas.

Dado que se ha observado que para una distancia menor entre p_1 y p_2 se reduce el rendimiento de UCB (con parámetro $\delta < 1$) en las primeras rondas, cabe pensar que si esta distancia es aún menor este hecho se extienda a un número mayor de rondas y quizás el algoritmo empiece a flaquear en ciertas aplicaciones prácticas. Este hecho se constata repitiendo el experimento anterior con $p_1 = 0.5$ y $p_2 = 0.49$. El resultado puede verse en la figura 19.

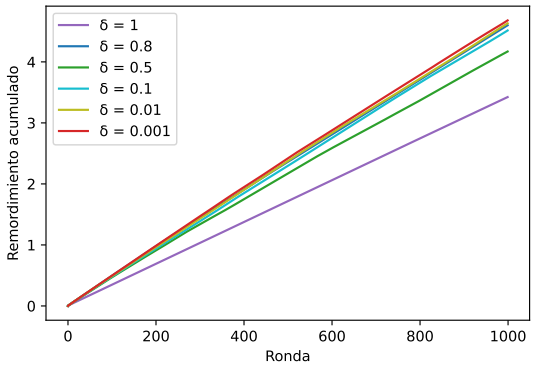


Figura 19: Remordimiento acumulado en la ejecución del algoritmo UCB para distintos valores del parámetro δ y horizonte $n = 1000$. El algoritmo se aplica sobre un bandido de dos brazos cuyas distribuciones son Bernoulli(0.5) y Bernoulli(0.49).

Se observa, como cabía esperar, que incluso con un número elevado de rondas todas las elecciones del parámetro δ dan un incremento del remordimiento acumulado aproximadamente lineal. Nuevamente, la estrategia con $\delta = 1$ vuelve a ser la mejor. Podemos adivinar que si ampliáramos lo suficiente el número de rondas observaríamos que, en las políticas con $\delta < 1$, la pendiente acaba por reducirse mucho . Por tanto, se puede concluir que cuanto más similares son las medias de dos brazos más difícil le resulta al algoritmo UCB (en especial con δ bajo) comenzar a sacar ventaja a otras estrategias más directas. Finalmente lo consigue, pero el número de rondas requerido puede ser muy alto. Entonces, en los casos en que trabajemos con brazos con medias muy próximas y un número de rondas reducido, preferiremos utilizar otras estrategias en lugar de UCB.

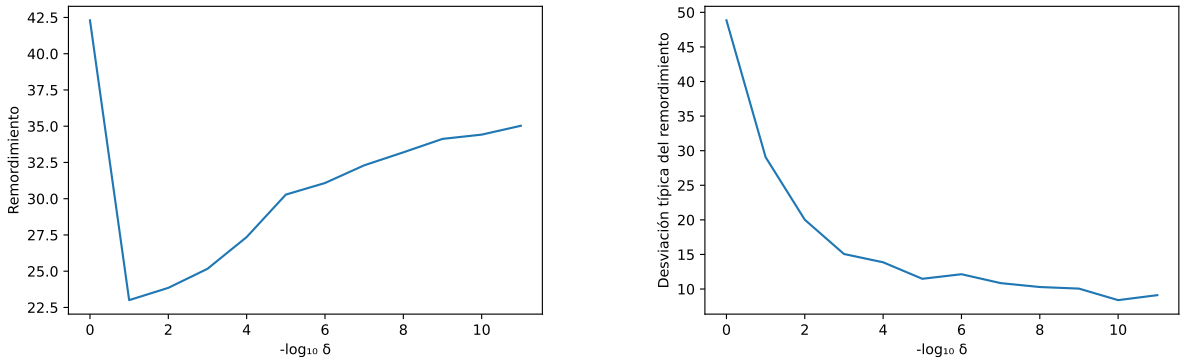
A la vista de los experimentos realizados, no hay una elección del parámetro δ adecuada para cualquier

número n de rondas por lo que δ ha de escogerse en función de n. Vemos a continuación una primera intuición sobre cómo hacer esto. En el caso de que nuestra heurística fallara y UCB_a acabara por debajo de μ_a, en el peor escenario el algoritmo dejaría de jugar ese brazo (dedicándose a otros que creería mejores) y generaría un pseudo-remordimiento lineal en n. Para compensar este caso peor, intuitivamente siempre se ha de escoger $\delta \leq \frac{1}{n}$, pues de esta manera la frecuencia δ con que se dan situaciones en las que la heurística falla compensará el remordimiento n generado en estas. Haciendo esto, la intuición nos dice que el remordimiento esperado será próximo al pseudo-remordimiento medio de los casos en que la heurística no falla. Por desgracia, hay más aspectos a tener en cuenta además de esta inocente idea (por ejemplo, el hecho de que los $\hat{\mu}_a$ no sean auténticas variables aleatorias), lo que lleva a que las elecciones concretas de δ que proporcionan buenos resultados teóricos tengan justificaciones mucho menos intuitivas. En general, elegiremos $\delta = \frac{1}{n^2}$ (y se verá empíricamente que no es una mala opción) pero no podremos justificar esta decisión mediante la intuición, sino que tendremos que esperar a la demostración de los resultados teóricos que se presentan en la siguiente sección. Estas demostraciones se realizarán en mi TFG de Matemáticas.

8.4. Análisis del remordimiento

8.4.1. Medición del remordimiento

Antes de pasar a presentar las cotas del remordimiento, comprobemos el impacto del parámetro δ de UCB sobre la media y la desviación típica muestrales del pseudo-remordimiento en un caso concreto. Igual que en 7.2.2, consideramos un bandido de 2 brazos con distribuciones $N(0,1)$ y $N(-0.1,1)$ y un horizonte de $n = 1000$ rondas. Para cada $\delta \in \{1, 10^{-1}, 10^{-2}, \ldots, 10^{-11}\}$ realizamos 600 simulaciones del algoritmo UCB con parámetro δ y calculamos la media y la desviación típica muestrales del pseudo-remordimiento. Los resultados se muestran en la figura 20.



(a) Esperanza. **(b)** Desviación típica.

Figura 20: Propiedades del pseudo-remordimiento de UCB en función del parámetro δ.

En la figura 20a se observa que hay un valor de δ, que en este caso parece ser $\delta = 0.1$, para el que la media muestral del pseudo-remordimiento (es decir, el remordimiento) es mínima. Conforme nos alejamos de este valor de δ el remordimiento se incrementa, aunque hay que señalar que lo hace mucho más rápido al aumentar δ que al reducirlo: para $\delta = 1$ se alcanza un remordimiento de 42.5 (casi $\frac{n\Delta}{2}$, el que se tendría con una estrategia aleatoria).

Por otro lado, en la figura 20b podemos apreciar como la desviación típica del pseudo-remordimiento disminuye al reducirse δ, aunque lo hace cada vez más lento. Esto está relacionado con algo que ya hemos comentado: que a menor valor de δ el algoritmo, al fomentar más la exploración, tiene un bajo riesgo de acabarse centrando en un brazo subóptimo. Si bien al realizar una exploración más exhaustiva de cada uno de los brazos hay un remordimiento "base" (resultado de la exploración más o menos equitativa de cada uno de los brazos hasta empezar a decantarse por algunos de ellos) mayor, como se reduce la probabilidad de que el algoritmo pase a centrarse principalmente en algún brazo no óptimo, hay muchas más posibilidades de

que el pseudo-remordimiento sea próximo al remordimiento base, por lo que no es extraño que la desviación típica del pseudo-remordimiento sea menor. Observamos además que la desviación típica muestral del pseudo-remordimiento para el δ óptimo ($\delta = 0.1$) es aproximadamente 25, muy alta en relación con el remordimiento medio, lo que quiere decir que la incertidumbre es alta y habrá que promediar muchas muestras del pseudo-remordimiento siempre que queramos obtener una estimación razonable del remordimiento.

Esto nos recuerda a lo que ocurría con Explora-Primero, en el cual el remordimiento (esperado) alcanzaba su valor mínimo para un cierto valor del parámetro m, pero incrementar m siempre reducía la desviación típica muestral del remordimiento. Salvo por el hecho de que UCB es más resiliente por no llegar a descartar nunca del todo la exploración, ambos algoritmos tienen en común que cuentan con un parámetro (m en el caso de Explora-Primero y δ en el caso de UCB) que regula el equilibrio exploración-explotación de manera que el remordimiento (medio) mínimo se consigue para un valor intermedio del parámetro, pero si se opta por una mayor exploración (aumentar m o disminuir δ) siempre se puede disminuir la desviación típica muestral del pseudo-remordimiento a costa de aumentar su media muestral.

El algoritmo UCB se distingue en que nunca llega a descartar del todo la exploración de un brazo que en un comienzo parece subóptimo, hecho que esperamos que haga a este algoritmo sustancialmente mejor que su predecesor. No obstante, comparando los resultados de este experimento con los obtenidos al realizarlo con Explora-Primero en 7.2.2, se aprecia que, si bien el mínimo remordimiento esperado es algo menor en el caso de UCB ($23 < 27$) el valor del parámetro δ a escoger para ello, $\delta = 0.1$, está muy alejado del valor de δ que al final de la sección anterior adelantamos que íbamos a escoger: $\delta = \frac{1}{n^2} = 10^{-6}$. De hecho, eligiendo $\delta = 10^{-6}$ vemos que se obtiene un remordimiento de en torno a 32, que es mayor de lo que se obtenía en Explora-Primero eligiendo cualquier m en $(100, 200)$.

¿Significa esto que el algoritmo UCB falla en su objetivo de ser mejor que Explora-Primero? No necesariamente. La razón de elegir $\delta = \frac{1}{n^2}$ es poder escoger el parámetro en función únicamente de n, lo que, gracias al truco de la duplicación, permite al agente ejecutar el algoritmo UCB sin ningún conocimiento previo sobre el problema salvo el número de brazos. En un caso real rara vez (por no decir nunca) dispondremos del valor del salto de suboptimalidad Δ para poder tomar $m = m_{Teor}$. Entonces, no resulta justo comparar el resultado obtenido por Explora-Primero con m escogido en función de tanto n como del salto de suboptimalidad Δ ($m = m_{Teor}$) con el obtenido por UCB con δ elegido únicamente a partir de n. Es decir, deberíamos comparar, en todo caso, Explora-Primero con $m = m_{Expl}$ con UCB con $\delta = \frac{1}{n^2}$.

De todas formas en este caso concreto

$$m_{Expl} = \lceil n^{2/3} \rceil = 100\,,$$

para el cual Explora-Primero sigue obteniendo un remordimiento en media mejor que el de UCB con $\delta = \frac{1}{n^2}$. Esto se debe a que se ha escogido un valor de Δ, $\Delta = 0.1$, bastante pequeño, en el rango en que la elección $m = m_{Expl}$ funciona adecuadamente. Como se vio en 7.2.10, el algoritmo Explora-Primero con parámetro $m = m_{Expl}$ tiene un comportamiento muy bueno cuando Δ es reducido y muy malo cuando Δ es elevado. Por tanto, si bien al enfrentarnos a un bandido con Δ reducido estamos obteniendo un resultado algo mejor con Explora-Primero (y $m = m_{Expl}$) que con UCB (y $\delta = \frac{1}{n^2}$), se espera que para instancias con un salto de suboptimalidad Δ mayor, UCB resulte vencedor con una diferencia mucho más grande.

8.4.2. Cotas del remordimiento

Al ejecutar el algoritmo UCB con nivel de confianza $\delta = \frac{1}{n^2}$ sobre un bandido tal que las distribuciones de sus brazos son 1-subgaussianas, el siguiente teorema da una cota superior del remordimiento:

Teorema 8.4.1. *Si el algoritmo UCB con nivel de confianza $\delta = \frac{1}{n^2}$ interactúa con un bandido estocástico 1-subgaussiano con saltos de suboptimalidad $\Delta_1, \ldots, \Delta_k$ se cumple que:*

$$R_n \leq C_{UCB}(n, \Delta_1, \ldots, \Delta_k)$$

donde:

$$C_{UCB}(n, \Delta_1, \ldots, \Delta_k) := 3 \sum_{a \in \mathcal{A}} \Delta_a + \sum_{a \in \mathcal{A}: \Delta_a > 0} \frac{16 \log(n)}{\Delta_a}$$

Como ocurría con la cota principal de Explora-Primero (sección 7.2), la dada por el teorema anterior tiene el inconveniente de depender de los inversos de los saltos de suboptimalidad Δ_a, lo que hace que si alguno de los Δ_a es muy pequeño el valor de la cota se dispare. Por suerte, disponemos de una cota alternativa que no depende de los inversos de los saltos de suboptimalidad.

Teorema 8.4.2. *Si el algoritmo UCB con nivel de confianza $\delta = \frac{1}{n^2}$ interactúa con un bandido estocástico 1-subgaussiano con saltos de suboptimalidad $\Delta_1, \ldots, \Delta_k$ se cumple que:*

$$R_n \leq C_{UCB2}(n, \Delta_1, \ldots, \Delta_k)$$

donde:

$$C_{UCB2}(n, \Delta_1, \ldots, \Delta_n) := 8\sqrt{nk\log(n)} + 3\sum_{a \in \mathcal{A}} \Delta_a$$

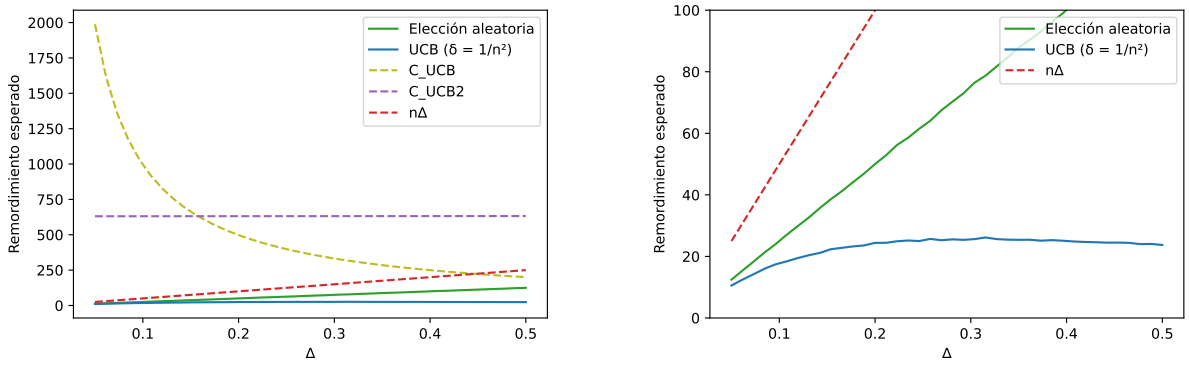
8.4.3. Comportamiento del algoritmo y bondad de las cotas

Recordemos que para Explora-Primero vimos que para valores de Δ no demasiado reducidos ($\Delta \leq \frac{2}{\sqrt{n}}$) el remordimiento cometido era muy parecido al de la política uniformemente aleatoria. Después de aquello cabe temerse lo peor respecto al comportamiento del algoritmo UCB en este caso.

Vamos a hacer algunos experimentos para poner a prueba el algoritmo UCB y observar cómo de ajustadas son las cotas C_{UCB} y C_{UCB2}. Para simplificar, nos limitaremos a probarlo para bandidos de 2 brazos, en los que las cotas toman la forma

$$C_{UCB}(n, \Delta) = 3\Delta + \frac{16\log(n)}{\Delta},$$
$$C_{UCB2}(n, \Delta) = 8\sqrt{2n\log(n)} + 3\Delta$$

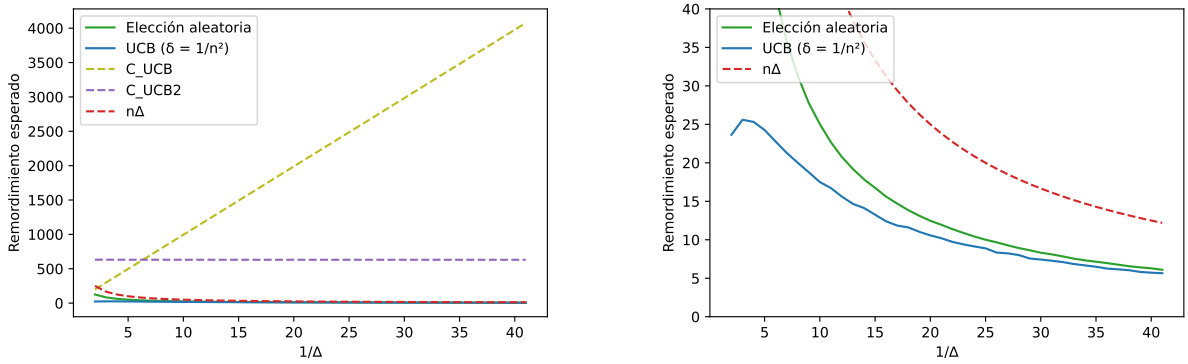
Comenzamos ejecutando el algoritmo UCB sobre un bandido de 2 brazos con distribuciones asociadas $U(0, 1+2\Delta)$ y $U(0, 1)$ y horizonte $n = 500$ (como en el experimento de 7.2.5). Para $\Delta \in (0, 0.5)$, ejecutamos el algoritmo UCB (con parámetro $\delta = \frac{1}{n^2}$) sobre la instancia del bandido y obtenemos el remordimiento resultado de promediar 200 repeticiones. Hacemos lo mismo para la estrategia uniformemente aleatoria. Calculamos, además, las cotas C_{UCB} y C_{UCB2}. Representamos los resultados en las figuras 21 y 22.



(a) **(b)**

Figura 21: Remordimiento de UCB y valores de las cotas para Δ grande. El algoritmo se ejecuta sobre un bandido uniforme con $n = 500$.

En la figura 21 se muestran los resultados para valores "grandes" de Δ ($\Delta = 0.5$ es lo máximo que puede ser sin que este bandido deje de ser 1-subgaussiano). Se puede observar que la estrategia UCB tiene un comportamiento muy bueno en relación con la de elección aleatoria y, si se compara con la figura 7, se ve que también es considerablemente mejor que Explora-Primero con $m = m_{Teor}$, especialmente cuando Δ es más alto. Sin embargo, aunque el remordimiento de UCB es reducido, las cotas obtenidas para el mismo (C_{UCB}



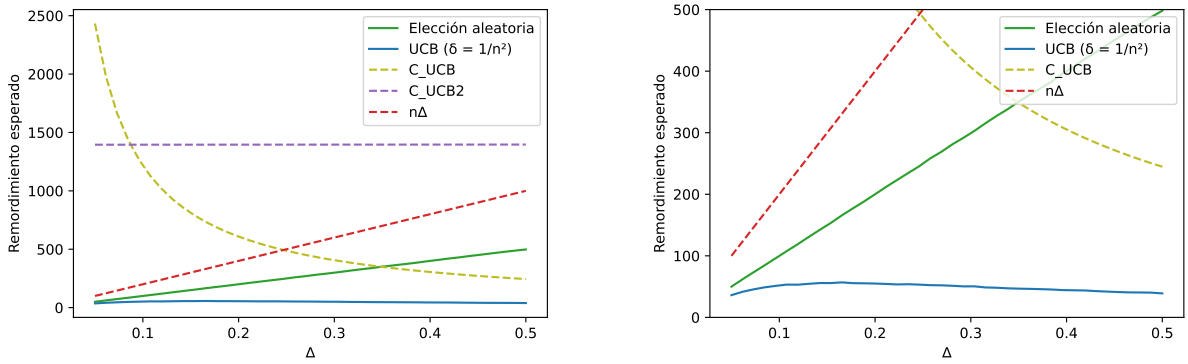
(a) **(b)**

Figura 22: Remordimiento de UCB y valores de las cotas para Δ pequeño. El algoritmo se ejecuta sobre un bandido uniforme con $n = 500$.

y C_{UCB2}) no son nada ajustadas, hasta el punto de que son superiores a la cota trivial $n\Delta$ para casi todo valor de Δ. La cota C_{UCB} experimenta una fuerte reducción al aumentar el valor de Δ pero, pese a ello, solo resulta algo más ajustada que $n\Delta$ para unos pocos valores de Δ cercanos a 0.5. La cota C_{UCB2}, por su parte, apenas cambia al variar Δ (la aportación a la cota del término que depende de Δ es mínima), siendo más ajustada que C_{UCB} para valores pequeños de Δ y menos para valores grandes pero, en cualquier caso, siempre menos ajustada que $n\Delta$.

En la figura 22 se muestran los resultados para valores pequeños de Δ. Como ocurría con Explora-Primero con $m = m_{Teor}$, el algoritmo UCB no obtiene un remordimiento aceptable cuando el salto de suboptimalidad es pequeño. De hecho, cuando Δ se reduce mucho el remordimiento es muy similar al de la estrategia de selección aleatoria. Respecto a las cotas, ni C_{UCB} ni C_{UCB2} resultan nada ajustadas para este caso, siendo $n\Delta$ una cota mucho más reducida y cercana al remordimiento que se obtiene.

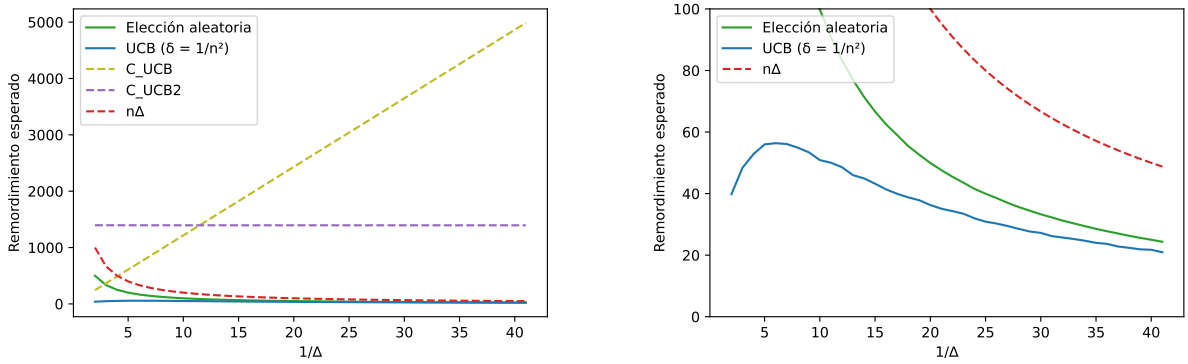
Dada la escasa utilidad que han mostrado las cotas C_{UCB} y C_{UCB2} en este ejemplo, nos preguntamos si esto se debe a que hemos escogido un horizonte $n = 500$ demasiado reducido. Para descartar este hecho repetimos el experimento con $n = 2000$. Los resultados se presentan en las figuras 23 y 24.



(a) **(b)**

Figura 23: Remordimiento de UCB y valores de las cotas para Δ grande. El algoritmo se ejecuta sobre un bandido uniforme con $n = 2000$.

Vemos que con esta modificación el comportamiento del algoritmo UCB mejora. En la figura 23 se observa que la diferencia entre el remordimiento de UCB y el de la estrategia aleatoria es mucho mayor de lo que era para $n = 500$. Además, la cota C_{UCB} resulta también ser mucho más ajustada para valores altos de Δ,



(a) **(b)**

Figura 24: Remordimiento de UCB y valores de las cotas para Δ pequeño. El algoritmo se ejecuta sobre un bandido uniforme con $n = 2000$.

siendo ahora considerablemente menor que $n\Delta$. En cambio, la cota C_{UCB2} no ha mejorado. En la figura 24 vemos que el remordimiento de UCB también es más reducido en relación con el de la política aleatoria, incluso para valores muy pequeños de Δ, aunque las cotas C_{UCB} y C_{UCB2} siguen sin resultar útiles en este caso.

Hemos observado que, si bien incrementar el valor de n hace que la cota C_{UCB} sea más ajustada, esta sigue sin estar próxima al remordimiento medio. Además, la cota C_{UCB2} sigue, pese a todo, sin ser útil en ningún caso. Notamos que:

$$C_{UCB} \leq n\Delta \iff \Delta > 4\sqrt{\frac{\log n}{n-3}}$$

$$C_{UCB2} \leq n\Delta \iff \Delta > \frac{8\sqrt{2n\log n}}{n-3}$$

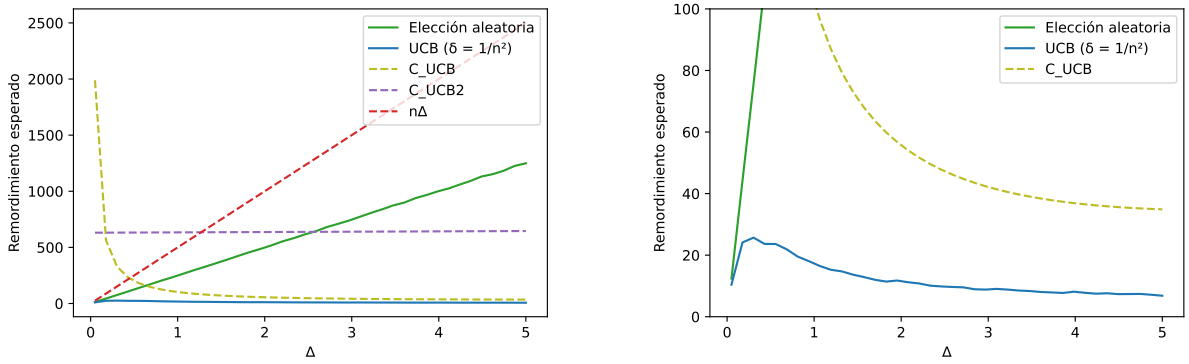
de manera que, independientemente de la distribución de los brazos, se tiene:

- Para $n = 500$, la cota C_{UCB} es mejor que la trivial $n\Delta$ si y solo si $\Delta > 0.447$, mientras que C_{UCB2} lo es solo si $\Delta > 1.27$.
- Para $n = 2000$, la cota C_{UCB} es mejor que $n\Delta$ si y solo si $\Delta > 0.27$, mientras que C_{UCB2} lo es solo si $\Delta > 0.7$.

Estos resultados se ajustan a lo que se observa en las gráficas anteriores. Entonces, para dar con casos en los que las cotas C_{UCB} y C_{UCB2} sean significativas con un valor de n reducido, deberíamos considerar Δ mayores. Sin embargo, la mayor parte de las distribuciones 1-subgaussianas que conocemos (Bernoulli, Uniforme,...) lo son precisamente por tomar valores en un intervalo de longitud 2 (por el lema de Hoeffding, teorema 5.2.4). Esto hace que no tenga mucho sentido considerar, para estos casos, saltos de suboptimalidad Δ mayores que 2.

Sí conocemos, no obstante, una distribución 1-subgaussiana cuyo soporte es no acotado y, por tanto, tal que Δ puede ser tan grande como queramos: la normal de varianza 1. Consideremos, pues, un bandido de 2 brazos con distribuciones $N(\Delta, 1)$ y $N(0, 1)$ para $\Delta \in (0, 5)$. Ejecutamos el algoritmo UCB con $\delta = \frac{1}{n^2}$ y $n = 500$ y representamos los resultados en la figura 25. En ella, podemos observar como para valores de Δ mayores que 2 la cota C_{UCB} pasa a acotar de forma bastante ajustada el remordimiento del algoritmo. Respecto a la cota C_{UCB2}, tenemos al menos que es más ajustada que la trivial $n\Delta$ para todo $\Delta > 1.27$, pero para estos valores es también mucho menos ajustada que C_{UCB}, de manera que seguimos sin encontrarle la utilidad.

Podemos, por tanto, concluir que el algoritmo UCB funciona muy bien (es decir, suficientemente mejor que la estrategia uniformemente aleatoria) para los valores de Δ no demasiado reducidos, $\Delta > 0.2$ en este



(a) **(b)**

Figura 25: Remordimiento de UCB para un bandido gaussiano con $n = 500$.

caso. Para valores de Δ mucho más pequeños, el algoritmo presenta las mismas carencias que Explora-Primero, con lo que es incapaz de obtener resultados mucho mejores que la política aleatoria. Respecto a las cotas obtenidas, ninguna de las dos resulta ser significativa en la mayoría de los casos habituales, aunque la cota C_{UCB} sí lo es cuando Δ es suficientemente grande ($\Delta > 2$ en este caso).

8.4.4. Comparación entre Explora-Primero y UCB

Incluye la respuesta al ejercicio 7.8 de [2]

Ya nos hemos hecho varias veces la pregunta de hasta que punto merece la pena utilizar UCB en lugar de Explora-Primero. En este apartado trataremos de finalmente responderla.

De entrada nos centraremos en la ejecución de UCB con parámetro $\delta = \frac{1}{n^2}$ y la compararemos con Explora-Primero para distintas elecciones del parámetro m y distintos saltos de suboptimalidad Δ. Consideramos un bandido de 2 brazos con distribuciones $N(0,1)$ y $N(-\Delta,1)$ para $\Delta \in (0,1)$ y un horizonte $n = 1000$. Para distintos valores de $\Delta \in (0,1)$, ejecutamos sobre el bandido el algoritmo Explora-Primero con $m = 25, 50, 75, 100, m_{Teor}$ y m_{Opt} (nótese que $m_{Expl} = 100$) y el algoritmo UCB con $\delta = \frac{1}{n^2} = 10^{-6}$. Para cada uno, extraemos 600 muestras del pseudo-remordimiento, promediamos los resultados y los representamos en la figura 26.

Naturalmente, el comportamiento es muy distinto si Δ es próximo a 0 o si es próximo a 1. Esto es especialmente cierto para Explora-Primero con m tomando un valor independiente de Δ ($m = 25, 50, 75$ o 100). En todos ellos, se observa un pico del remordimiento para Δ cercano a 0, seguido de un descenso más suave y finalmente un ascenso siguiendo una asíntota lineal. Recordemos que el remordimiento del algoritmo Explora-Primero para un bandido de 2 brazos se escribe como

$$R_n = m\Delta + (n - 2m)E[\Delta_{\hat{a}}]\,,$$

donde, por el teorema 7.2.1, el segundo sumando se acota por

$$(n - 2m)\Delta \exp\left(-\frac{m\Delta^2}{4}\right)\,,$$

es decir,

$$R_n \leq m\Delta + (n - 2m)\Delta \exp\left(-\frac{m\Delta^2}{4}\right)\,.$$

Para un valor de m fijo, el primer sumando es el relevante para valores altos de Δ y el segundo para valores bajos. En la región intermedia en la que ambos tienen un peso mediano es en la que se alcanza el mínimo remordimiento (para $\Delta \neq 0$). Esto explica, pues, el pico inicial (cuando el segundo término tiene mayor

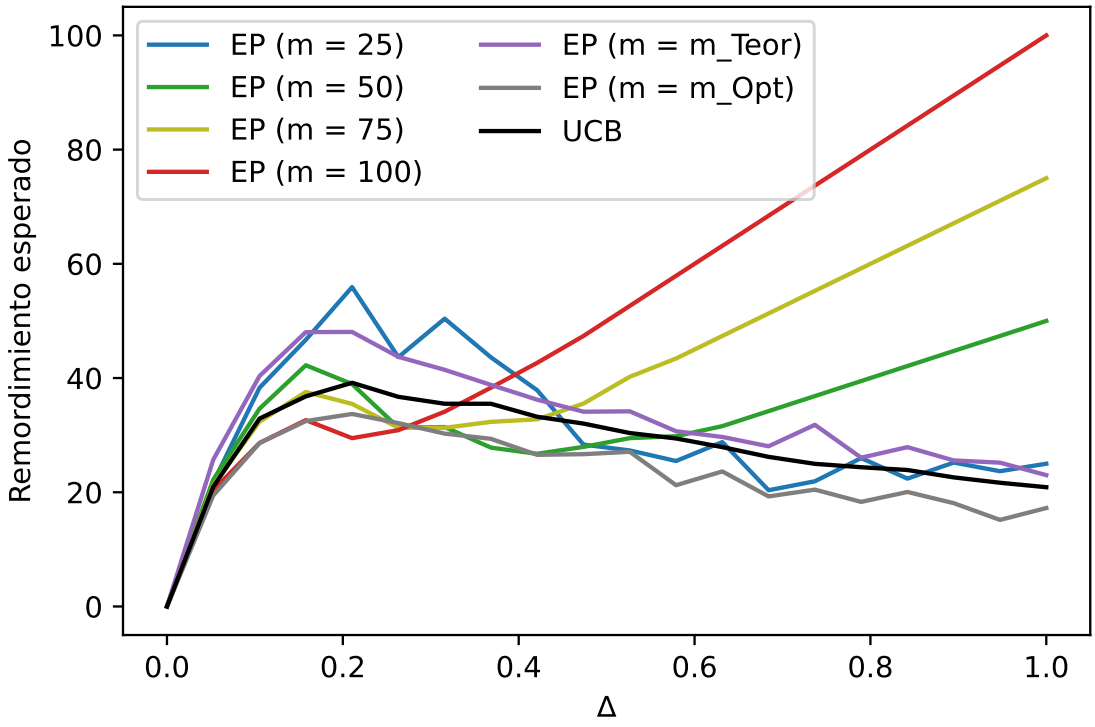


Figura 26: Comparativa entre UCB y Explora-Primero con distintas elecciones del parámetro m. Se ejecutan sobre un bandido gaussiano con horizonte $n = 1000$.

peso), el descenso suave (cuando ambos términos van igualándose) y la asíntota lineal (a partir de cierto punto, solo el primer término tiene peso y, por ello, la gráfica es aproximadamente la de $m\Delta$). Analicemos las diferencias en el remordimiento obtenido para las distintas elecciones de m en el algoritmo Explora-Primero:

- Las estrategias con $m = 100$ y $m = 75$ (es decir, m próximo a m_{Expl}) obtienen el remordimiento menor (rondando el óptimo) entre 0 y 0.25, pero son las que peor se comportan para Δ grande. El pico que se da en todas las estrategias para Δ pequeño es muy reducido en estas, pero el crecimiento lineal asintótico empieza en Δ bastante bajo ($\Delta = 0.4$) y tiene una pendiente elevada.
- Por su parte, la ejecución con $m = 25$ experimenta el peor remordimiento de todas para Δ reducido, pero ronda el remordimiento óptimo para Δ entre 0.4 y 0.8 y uno de los mejores para $\Delta = 1$. En esta ocasión, el pico en Δ reducido es pronunciado, pero desde él se produce un fuerte descenso y el crecimiento asintótico comienza tan tarde y tiene tan poca pendiente que apenas se aprecia en la gráfica.
- La ejecución con $m = 50$ es un término medio entre las dos anteriores, de manera que en ella se aprecia bien tanto el pico para Δ reducido como la reducción posterior y la asíntota para Δ elevado.
- Respecto a la estrategia con $m = m_{Teor}$, no olvidemos que utiliza un valor de m diferente para cada valor de Δ (el m_{Teor} para cada Δ se muestra en la figura 27). Sin embargo, pese a que escoge valores de m muy elevados para Δ pequeño, el remordimiento que produce es similar a la de la estrategia con $m = 25$, con un descenso algo menos pronunciado en la fase intermedia y, por ello, algo más alejada del remordimiento óptimo.
- La estrategia con $m = m_{Opt}$ (la elección de m para cada Δ se muestra en la figura 27) , como sabemos, escoge para cada Δ el m con el que el remordimiento producido es menor, de manera que es lógico que

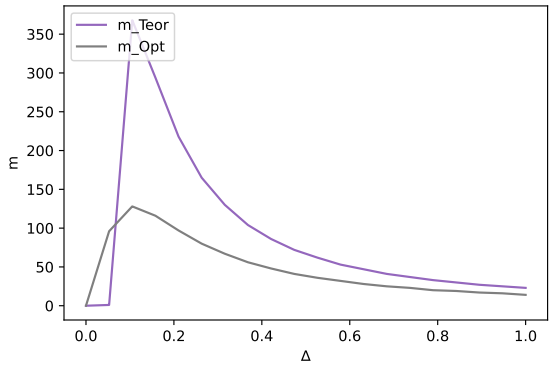


Figura 27: Valores de m escogidos al ejecutar Explora-Primero con $m = m_{Teor}$ o $m = m_{Opt}$.

esta estrategia obtenga el mínimo remordimiento de todas las que utilizan Explora-Primero (en la figura 26 se ve como en ciertos momentos algunas estrategias con m fijo parecen tener menor remordimiento que $m = m_{Opt}$, pero esto probablemente es un fallo en la estimación debido a la elevada varianza del pseudo-remordimiento cuando m es pequeño)

Pasamos ahora a comparar con UCB. El remordimiento en que incurre el algoritmo UCB es el más estable de todos los mostrados (apenas presenta fluctuaciones) y, además, no presenta ningún caso especialmente malo. Es la única estrategia, excluyendo la de $m = m_{Opt}$ cuyo remordimiento no excede 40 para ningún valor de $\Delta \in (0, 1)$. Ninguna de las otras elecciones de m de Explora-Primero (ni siquiera $m = m_{Teor}$ que no es aplicable en la práctica) consigue obtener un remordimiento menor que el de UCB para cualquier valor de Δ, pero también es cierto que las estrategias con $m = 100$ y $m = 75$ son mejores que ella para $\Delta \in (0, 0.3)$ y la estrategia con $m = 25$ es mejor para $\Delta \in (0.4, 0.8)$.

Se concluye, por tanto, que UCB con $\delta = \frac{1}{n^2}$ es una estrategia equilibrada, que desempeña en media mejor que Explora-Primero para cualquier elección de m salvo aquella en que m se escoge de forma óptima para cada Δ ($m = m_{Opt}$). Si bien, por tanto, no se puede decir que UCB sea mejor que Explora-Primero (eligiendo adecuadamente el número de exploraciones se obtienen mejores resultados con Explora-Primero que con UCB) desde un punto de vista práctico UCB resulta mucho más atractivo, ya que basta escoger $\delta = \frac{1}{n^2}$ para obtener buenos resultados, mientras que no disponemos de ningún procedimiento para estimar m_{Opt}. Solo merecerá la pena utilizar Explora-Primero con $m = m_{Expl}$ (elección también fácil de calcular) cuando se sepa de antemano que los saltos de suboptimalidad van a ser reducidos.

Nos preguntamos ahora cómo variarán los resultados obtenidos si se varía el valor de δ con que se ejecuta, pues en el experimento anterior únicamente hemos ejecutado UCB con parámetro $\delta = \frac{1}{n^2} = \frac{1}{10^6}$. Repetimos ahora el experimento para UCB con parámetro $\delta = 1$, $\frac{1}{10^4}$ y $\frac{1}{10^8}$. Representamos los resultados en la figura 28, en la que también representamos nuevamente el remordimiento obtenido para Explora-Primero con $m = 25$ y con $m = 100$ y para UCB con $\delta = \frac{1}{10^6}$.

Para $\delta = 1$ el remordimiento crece mucho con Δ, como no podía ser de otra manera dada la escasa exploración que se lleva a cabo con esta elección de parámetro. Exceptuando este caso, todos las gráficas del remordimiento asociado a UCB tienen la misma forma y no están muy separadas unas de otras. Esto nos dice que el valor de δ que se escoja para resolver este problema de bandidos no es muy crítico: hay un amplio rango de valores de δ (al menos $(10^{-8}, 10^{-2})$) para los que el algoritmo UCB tiene un remordimiento aceptable. Sí hay que admitir, no obstante, que hay valores de δ para los que el remordimiento de UCB es menor que si se toma $\delta = \frac{1}{n^2}$, por ejemplo, $\delta = 10^{-4}$ o $\delta = 0.01$. Para este último se observa que el remordimiento obtenido es claramente menor que el de Explora-Primero con $m = m_{Opt}$, por lo que una adecuada (que no necesarimante óptima) elección de δ permite a UCB superar a Explora-Primero incluso con la mejor elección posible del parámetro m.

Además, que las gráficas sean prácticamente paralelas nos dice que el salto de suboptimalidad Δ no influye

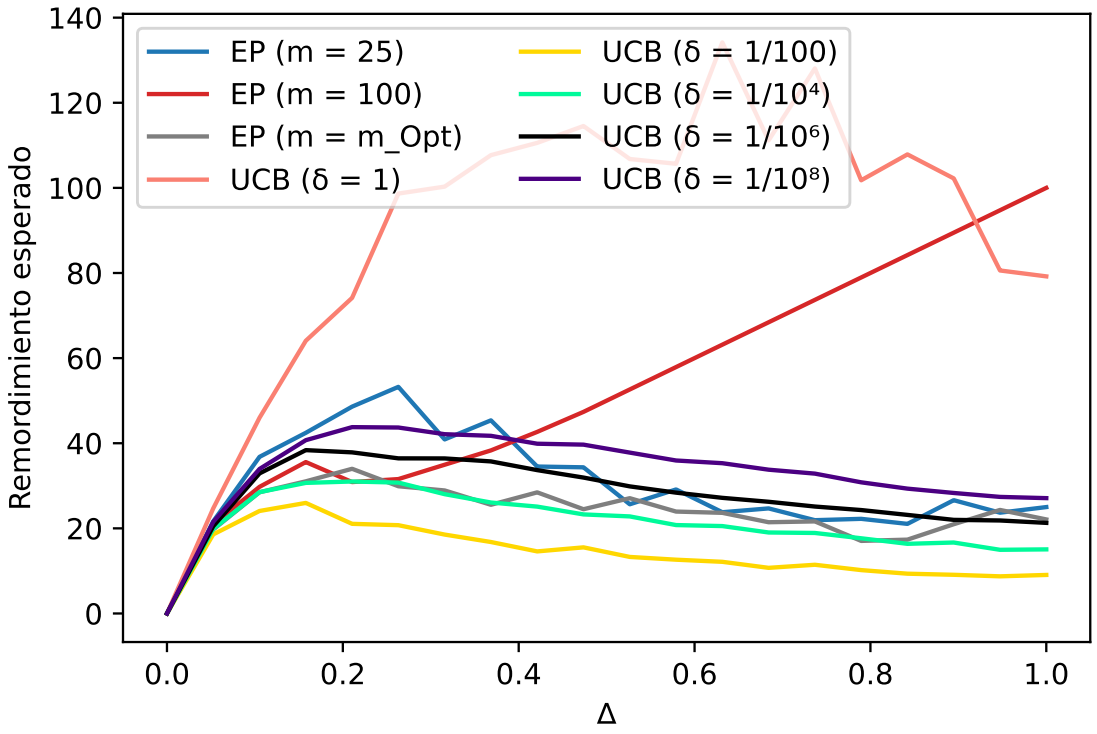


Figura 28: Comparativa entre UCB y Explora-Primero con distintas elecciones de los parámetros δ y m. Se ejecutan sobre un bandido gaussiano con horizonte $n = 1000$.

mucho sobre el valor de δ para el que el algoritmo desempeña adecuadamente. Esto resulta muy ventajoso de cara a diseñar métodos de elección de δ, pues, dada la escasa influencia de Δ, se podrá calcular δ en función de exclusivamente n. Entonces, empleando el truco de la duplicación si fuera necesario, se ejecutaría el algoritmo sobre un problema de bandidos estocásticos sin exigir al agente ningún conocimiento a priori sobre el mismo.

8.4.5. Conclusiones

Hemos visto que la correcta elección del parámetro δ hace al algoritmo UCB considerablemente mejor que su predecesor Explora-Primero. Sin embargo, lo verdaderamente importante es que, aun eligiendo δ "sin complicarnos mucho la vida" (por ejemplo, $\delta = \frac{1}{n^2}$), de forma que es muy probable que esté a considerable distancia del valor óptimo, el remordimiento obtenido sigue siendo similar al remordimiento que obtendría Explora-Primero eligiendo $m = m_{Opt}$ (elección que es inviable en la práctica).

9. Conclusión

Con lo expuesto, consideramos que quedan cubiertos los objetivos planteados para este trabajo. Se ha explicado de forma sencilla, autocontenida y bastante exhaustiva el área de los bandidos estocásticos haciendo hincapié, como dijimos en la introducción, en la parte de intuiciones y experimental de los mismos. Por otro lado, la parte más formal y rigurosa, que aquí se ha tratado solo superficialmente, se detallará en mi TFG de Matemáticas.

El campo de los algoritmos de bandidos, pese a ser un área muy concreta dentro del aprendizaje por refuerzo, es aún una disciplina muy amplia, pues distintos tipos de bandidos (estocásticos, antagonistas, contextuales...) requieren estrategias muy distintas para obtener resultados adecuados. En este trabajo se ha tratado de forma detallada uno de los modelos más básicos, el de los bandidos estocásticos, buscando que sirva de introducción a los bandidos para un lector sin ninguna base previa en el tema. Se espera, asimismo, que otros alumnos realicen otros TFGs sobre bandidos en cursos venideros, centrándose en alguno de los tipos que aquí no se han desarrollado. Con esto en consideración, pensamos que este trabajo puede ser de utilidad a estos alumnos para tomarse como referencia inicial en el caso de que requieran apoyarse en algunas de las ideas que se utilizan en bandidos estocásticos para el estudio de la variante de los bandidos que ellos elijan.

10. Conclusion

Taking into account what we presented, we consider every objective of the dissertation to be covered. The topic of stochastic bandits has been explained in a simple, self-contained and exhaustive way. As we said in the introduction, we focused on intuitions and experiments. On the other hand, the most formal and rigorous part, which here has been addressed superficially, will be detailed in my Mathemathics Dissertation.

The field of bandit algorithms, in spite of being a very specific area in reinforcement learning, is still a really wide subject, as different type of bandits (stochastic, adversarial, contextual ...) require very different strategies to obtain adequate outcomes. In this dissertation we have dealt with one of the most basic models, stochastic bandits, intending for it to serve as an introduction to bandits for a reader without any prior experience on the subject. We expect, as well, that other students will make dissertations about bandits in the following years, focusing on one of the types that we have not developed here. Considering this, we think this dissertation could be useful to these students: it could be taken as one of several initial references in case they need to build on some of the ideas we used in stochastic bandits for the study of the variant of their choice.

Apéndice A El truco de la duplicación

Incluye parte de la respuesta al ejercicio 6.6 de [2]

El truco de la duplicación (*doubling trick* en inglés) es un procedimiento general que convierte un algoritmo de bandidos (posiblemente no atemporal) en uno atemporal. Es decir, elimina la dependencia de n en la elección de los parámetros del algoritmo.

El algoritmo inicial contará, en general, con una serie de parámetros $p_1, \dots, p_r$ cuyo valor debe decidirse antes de la ejecución del mismo. Si este algoritmo es no atemporal será porque es necesario dar un cierto valor (dependiente de n) a alguno de estos parámetros para obtener unas determinadas garantías sobre el remordimiento. En general, se partirá de un resultado del tipo siguiente para el algoritmo inicial:

Teorema: Si se ejecuta este algoritmo sobre un bandido (de cierto tipo) con valores $p_1 = f_1(n), \dots, p_r = f_r(n)$ para los parámetros del mismo, se tiene la siguiente cota para el remordimiento:

$$R_n \leq C(n)$$

Además del valor de n, quizás se requieran otros datos sobre la instancia para elegir el valor de los parámetros (es decir, quizás $f_1, \dots, f_r$ no sean función exclusivamente de n). Este otro conocimiento requerido no puede sortearse por el procedimiento que veremos aquí. En caso de que no se dispusiera de la información adicional a n necesaria para inicializar $p_1, \dots, p_n$ no quedaría otra que renunciar a esta cota sobre el remordimiento y buscar una elección distinta de los parámetros que no utilice información de la que no se dispone. Aquí asumiremos que el resto de datos sobre la instancia que se requieren para escoger los parámetros son conocidos al inicio de la ejecución.

Para evitar seguir haciendo referencia a los parámetros $p_1, \dots, p_n$ podemos suponer, sin pérdida de generalidad, que el algoritmo únicamente depende de n como parámetro. En efecto, basta con incluir al inicio del algoritmo la inicialización de los parámetros del mismo como $p_1 = f_1(n), \dots, p_r = f_r(n)$. De esta manera, denotamos $\mathrm{ALG}(n)$ al nuevo algoritmo (posiblemente no atemporal), que solo requiere (adicionalmente a lo ya conocido) de un valor de n para poder ejecutarse, y llamaremos parámetro de ALG a este valor de n proporcionado. Nuestro objetivo será transformar el algoritmo $\mathrm{ALG}(n)$ en uno ALG' sin parámetros (y, por tanto, atemporal).

Por supuesto, siempre podríamos comenzar dando un valor cualquiera a n y llamar a $\mathrm{ALG}(n)$ con este valor, con lo que ya tendríamos un algoritmo atemporal que resuelve el problema. Pero, en tal caso, con toda probabilidad el algoritmo incurriría en un remordimiento muy elevado. Lo que nos interesa es que el algoritmo atemporal ALG' que se obtenga cuente con una cota para el remordimiento similar a la de ALG para, de esta manera, que los resultados que veamos para algoritmos no atemporales puedan aplicarse al contexto habitual en que no se conoce el valor del horizonte n.

Supondremos que el juego secuencial cuenta con un número n de rondas totales, pero el agente no conoce este número y, por tanto, lo único que puede hacer es ejecutar una ronda tras otra del juego secuencial hasta recibir el aviso (tras completar la ronda n-ésima) de que el juego ha finalizado. Esto no supondría ningún problema si no se requiriera el valor de n para la inicialización de los parámetros del algoritmo, pero en nuestro caso sí será necesario.

La idea del truco de la duplicación consiste en aplicar sucesivamente el algoritmo ALG para un número de rondas cada vez mayor, hasta que se alcance el número total de rondas n, desconocido a priori. La transformación va, pues, asociada a una sucesión $T := (T_i)_{i=0}^{+\infty}$ de números naturales y creciente. Es claro que esta sucesión verificará

$$\lim_{i \to +\infty} T_i = +\infty \,,$$

de manera que existirá un $i \in \mathbb{N}$ tal que $T_i \geq n$. Denotaremos

$$L := \min\{i \in \mathbb{N} \mid T_i \geq n\} \,.$$

Explicamos ahora los pasos que realizará el algoritmo atemporal ALG'. ALG' comenzará ejecutando ALG con parámetro T_0 hasta el final de la ronda T_0. Para $i = 0, 1, \dots, L-2$, al finalizar la ronda T_i se realizará

un reinicio de ALG con parámetro $T_{i+1} - T_i$ y se ejecutará hasta el final de la ronda T_{i+1}. Finalmente, al acabar la ronda T_{L-1} se volverá a realizar un reinicio de ALG, esta vez con parámetro $T_L - T_{L-1}$, que se ejecutará hasta el final de la ronda n. Los pasos que acabamos de describir se muestran como pseudocódigo en el algoritmo 4.

Algoritmo 4 Algoritmo atemporal obtenido por el truco de la duplicación

```
i ← 0
Inicializar ALG_0 := ALG(n = T_0)
for t in 1, ..., n do
    if t > T_i then
        Inicializar ALG_i := ALG(n = T_{i+1} − T_i)
        i ← i + 1
    end if
    Seleccionar brazo usando ALG_i
    Alimentar ALG_i con la recompensa obtenida
end for
```

Observamos que se realizan un total de $L + 1$ ejecuciones de ALG para cubrir las n rondas:

- Entre las rondas 1 y T_0 (T_0 rondas), con parámetro T_0.
- Entre las rondas $T_0 + 1$ y T_1 ($T_1 - T_0$ rondas), con parámetro $T_1 - T_0$.
- ...
- Entre las rondas $T_i + 1$ y T_{i+1} ($T_{i+1} - T_i$ rondas), con parámetro $T_{i+1} - T_i$.
- ...
- Entre las rondas $T_{L-2} + 1$ y T_{L-1} ($T_{L-1} - T_{L-2}$ rondas), con parámetro $T_{L-1} - T_{L-2}$.
- Entre las rondas $T_{L-1} + 1$ y n ($n - T_{L-1}$ rondas), con parámetro $T_L - T_{L-1}$.

Notemos que todas las ejecuciones de ALG, salvo la última, toman como parámetro (horizonte de ALG) el número de rondas que finalmente se ejecutan con esa instancia de ALG. En la última ejecución es imposible conseguir esto si $T_L > n$, pues el parámetro debe estar escogido al lanzar el algoritmo y el agente solo conoce el horizonte final n cuando ya ha realizado las n rondas. Sin embargo, esto no plantea ningún problema, ya que, por ser el remordimiento una función creciente en el número de rondas, se tendrá

$$R_{n-T_{L-1}} \leq R_{T_L - T_{L-1}} .$$

Es decir, el remordimiento en que incurre ALG al ejecutar solo $n - T_{L-1}$ rondas de las $T_L - T_{L-1}$ "planeadas" será a lo sumo el mismo que si hubiera ejecutado todas las rondas. De esta manera, el remordimiento de esta última etapa se acota de la misma forma que el de las demás.

Para cada número $z \in \{1, \ldots, n\}$, denotamos R_z el remordimiento de ALG para z rondas. El remordimiento de ALG$'$ para las n rondas, R'_n, verifica

$$R'_n \leq R_{T_0} + \sum_{i=1}^{L-1} R_{T_{i+1} - T_i} .$$

El objetivo será transformar una cota significativa, $R_z \leq C(z)$, del remordimiento de ALG en una similar para el remordimiento de ALG$'$. Para ello, la clave será elegir la sucesión $T = (T_i)_{i=0}^{+\infty}$ convenientemente. Claramente, no cualquier sucesión T de números naturales y creciente nos proporcionará una cota adecuada. Por ejemplo, eligiendo $T_i = i$ para cada $i \in \mathbb{N}$ se tendría $L = n$ de manera que

$$R'_n \leq R_{T_0} + \sum_{i=1}^{n-1} R_{T_{i+1} - T_i} .$$

En este caso, incluso aunque dispusiéramos de una cota constante para el remordimiento de ALG (es decir, $R_N \leq C$ para cualquier $N \in \mathbb{N}$, hecho que nunca se dará), la cota del remordimiento de ALG$'$ quedaría

$$R'_n \leq Cn\,,$$

es decir, $R'_n \in \Theta(n)$. Por tanto, claramente es necesaria una sucesión T que se incremente mucho más rápidamente que la anterior. La sucesión $T = (T_i)_{i=0}^{+\infty}$ que se escoge más habitualmente es la llamada **sucesión geométrica**, $T_i = 2^i$, para la que se tiene $L = \lceil \log_2 n \rceil$. En ella el número de rondas cubiertas por cada instancia de ALG se duplica en cada fase, lo que da su nombre al truco de la duplicación.

Estudiemos a continuación los dos tipos de cotas más habituales para el remordimiento de ALG y veamos si es posible preservarlas al convertir el algoritmo en atemporal:

- Supongamos que para cierto $\alpha \in (0,1)$ se tiene

  $$R_z \in \Theta(z^\alpha)\,,$$

  para cualquier número N de rondas. Entonces, existe un $c \in \mathbb{R}$ tal que

  $$R_z \leq cz^\alpha$$

  para cualquier $z \in \mathbb{N}$. Tenemos, pues,

  $$R'_n \leq cT_0^\alpha + c\sum_{i=1}^{L-1}(T_{i+1} - T_i)^\alpha\,.$$

  Veamos qué ocurre si tomamos como T la sucesión geométrica $T_i := 2^i$. En este caso se tiene $L = \lceil \log_2 n \rceil$ y

  $$R'_n \leq c + c\sum_{i=1}^{\lceil \log_2 n \rceil - 1} 2^{(i-1)\alpha}$$

  Aplicando la fórmula de la suma de una progresión geométrica obtenemos

  $$\sum_{i=1}^{\lceil \log_2 n \rceil - 1} 2^{(i-1)\alpha} = \frac{2^{(\lceil \log_2 n \rceil - 1)\alpha} - 1}{2^\alpha - 1} \leq \frac{n^\alpha - 1}{2^\alpha - 1}\,.$$

  Por tanto

  $$R'_n \leq c + c\frac{n^\alpha - 1}{2^\alpha - 1} \in \Theta(n^\alpha)\,.$$

  Es decir, utilizar la sucesión geométrica $T_i = 2^i$ con un algoritmo con remordimiento en $O(n^\alpha)$ permite obtener un algoritmo atemporal que conserva la naturaleza asintótica de la cota.

- Supongamos ahora que se verifica

  $$R_z \in \Theta(\log z)\,,$$

  para cualquier número z de rondas. Es decir, existe un $c \in \mathbb{R}$ tal que

  $$R_z \leq c \log z\,.$$

  En este caso, se tiene

  $$R'_n \leq c \log T_0 + c\sum_{i=1}^{L-1} \log(T_{i+1} - T_i)\,.$$

  Si ahora tomamos $T_i = 2^i$ (con lo que $L = \lceil \log_2 n \rceil$), lo que obtenemos es

  $$R'_n \leq c\sum_{i=1}^{\lceil \log_2 n \rceil - 1} \log(2^{i-1}) = c \log 2 \sum_{i=1}^{\lceil \log_2 n \rceil - 1} (i-1) = c \log 2 \sum_{i=1}^{\lceil \log_2 n \rceil - 2} i$$

Aplicando la fórmula de la suma de una progresión aritmética, se obtiene

$$R'_n \leq c \log 2 \frac{(\lceil \log_2 n \rceil - 1)(\lceil \log_2 n \rceil - 2)}{2} \in \Theta((\log n)^2)$$

Por tanto, en esta ocasión la sucesión T_i más típica no nos permite conservar del todo el orden asintótico de la cota del remordimiento.

Si bien la cota obtenida no es catastrófica, es buen momento para plantearnos otras elecciones de T que quizás puedan permitirnos mejorarla. Consideramos la **sucesión exponencial** $T_i = 2^{2^i}$, para la cual se da $L = \lceil \log_2(\log_2 n) \rceil$. Se tiene:

$$\begin{aligned} R'_n &\leq c \log 2 + c \sum_{i=1}^{\lceil \log_2(\log_2 n) \rceil - 1} \log(2^{2^{i+1}} - 2^{2^i}) \\ &\leq c \log 2 + c \sum_{i=1}^{\lceil \log_2(\log_2 n) \rceil - 1} \log(2^{2^{i+1}}) \\ &= c \log 2 + c \log 2 \sum_{i=1}^{\lceil \log_2(\log_2 n) \rceil - 1} 2^{i+1} \end{aligned}$$

Aplicando la fórmula de la suma de una progresión geométrica, se obtiene

$$\sum_{i=1}^{\lceil \log_2(\log_2 n) \rceil - 1} 2^{i+1} = \frac{2^{\lceil \log_2(\log_2 n) \rceil + 1} - 4}{2 - 1} = 4(2^{\lceil \log_2(\log_2 n) \rceil - 1} - 1) \leq 4(\log_2 n - 1)\,.$$

Por tanto

$$R'_n \leq c \log 2 + 4c \log 2(\log_2 n - 1) \in \Theta(\log n).$$

Es decir, utilizar la sucesión exponencial $T_i = 2^{2^i}$ con un algoritmo de remordimiento en $O(\log n)$ permite obtener un algoritmo atemporal que conserva la naturaleza asintótica de la cota.

Por tanto, los principales tipos de cotas para el remordimiento se conservan mediante el uso del truco de la duplicación. Si bien se podrían explorar otros tipos de cotas (por ejemplo, $\Theta((\log z)^\gamma)$) pensamos que lo visto aquí es suficiente para el carácter introductorio de este trabajo.

Referencias

[1] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, 1933.

[2] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge University Press, 2020.

[3] A. Slivkins, *Introduction to Multi-Armed Bandits*. Foundations and Trends in Machine Learning, 2019.

[4] S. Bubeck and N. Cesa-Bianchi, *Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems*. Foundations and Trends in Machine Learning, 2012.

[5] G. Casella and R. L.Berger, *Statistical Inference*. Thomson Learning, second ed., 2002.

[6] P. Billinsgley, *Probability and Measure*. John Wiley & Sons Inc., 1995.

[7] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. 2. John Wiley & Sons Inc., 1970.

[8] S. Boucheron, G. Lugosi, and P. Massart, *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

[9] M. Herrero Agustín, "Codigo TFG informatica." `https://github.com/marcosherreroa/Codigo-TFG-Informatica`, 2022.

[10] R. Kleinberg, F. Radlinski, and T. Joachims, "Learning diverse rankings with multi-armed bandits," *25th International Conference on Machine Learning*, 2008.

[11] M. S. Talebi, Z. Zou, R. Combes, A. Proutiere, and M. Johansson, "Stochastic online shortest path routing: The value of feedback," *IEEE Transactions on Automatic Control*, 2018.

[12] S. Gelly, Y. Wang, R. Munos, and O. Teytaud, "Modification of UCT with patterns in Monte-Carlo go," *INRIA*, 2006.

[13] cardinal (`https://stats.stackexchange.com/users/2970/cardinal`), "Existence of the moment generating function and variance." Cross Validated. URL:`https://stats.stackexchange.com/q/32787` (version: 2017-04-13).

[14] Matthew Drury (`https://stats.stackexchange.com/users/74500/matthew-drury`), "Do all bounded probability distributions have a definite mean?." Cross Validated. URL:`https://stats.stackexchange.com/q/442296` (version: 2019-12-26).

[15] Zen (`https://stats.stackexchange.com/users/9394/zen`), "Variance of a bounded random variable." Cross Validated. URL:`https://stats.stackexchange.com/q/50552` (version: 2019-07-26).