Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Bandidos multibrazo para selección de cartera

Estudante: Daniel Sergio Vega Rodríguez

Dirección: Javier Parapar López

Álvaro Barreiro García

A Coruña, junio de 2022.

Este trabajo esta dedicado a mi familia

Agradecimientos

Quisiera agradecer a mi familia, amigos, compañeros y demás allegados por su apoyo incondicional durante el desarrollo de este trabajo.

A mi madre, mi padre y mi hermana, por el apoyo constante a lo largo de todo el año. A mis amigos y a Andrea por darme las fuerzas para seguir trabajando. Y especialmente a mis compañeros de laboratorio por su paciencia, ayuda y por todos esos kilos de café.

He tenido una excelente experiencia desarrollando este trabajo junto al resto de los integrantes del IRLab. Gracias Alfonso por arreglar las desfeitas que hago en el ordenador. Gracias Manuel por tu aleatorio y explosivo sentido del humor. Gracias Anxo por ayudarme siempre que lo necesitaba. Gracias Gilberto por tu ayuda y guía que estuvo siempre ahí.

Es también especialmente digna de mención la contribución de mis directores, Javier y Álvaro, a la hora de asesorarme por el camino correcto y guiarme cuando perdía las pistas de mis objetivos.

Resumen

Este proyecto consiste en la implementación y validación de modelos para recomendación de cartera (*portfolio recommendation*). Para ello será necesario el estudio, diseño, desarrollo e implementación de un software capaz de asistir al usuario en la selección de valores bursátiles en los que invertir. Nos centraremos en un los modelos denominados como "Bandidos Bayesianos" para poder estimar qué valores ofrecen mejor rentabilidad al usuario combinando explotación y exploración. Dichos modelos seleccionarán valores en función de una estimación de probabilidades construida a partir de conjuntos de datos que describen el comportamiento de dichos valores en el pasado. Los valores más prometedores serán usualmente recomendados al usuario, con la esperanza de obtener un resultado positivo en la inversión futura. Las recomendaciones dadas por el algoritmo tenderán a variar a medida que nuevos datos son integrados en el modelo, ya que estos usualmente revelarán cambios en el comportamiento de los valores observados.

Abstract

This project revolves around the implementation and validation of models meant for portfolio recommendation. To accomplish this, the investigation, design, development and implementation of a demonstrative application will be necessary. The resulting tool will assist the user in making decisions over stock values. This project will focus on a series of models based on the Bayesian Bandit agent to estimate which values are likely to be promising. Said models will choose values based on a collection of likelihood random variables which are constructed from datasets that describe the historic behaviour of the considered values. The most promising values will be usually recommended to the user. That is, hoping to obtain a positive result in the future investment. Recommendations given by the algorithm will tend to change over new data additions to the model. Those additions will often reveal changes in the behaviour of the observed stock values.

Palabras clave:

- Bandido Bayesiano
- Aprendizaje por refuerzo
- Mercado de valores
- Sistema de recomendación
- Bandidos con contexto
- Tensorflow

Keywords:

- Bayesian Bandit
- Reinforcement Learning
- Stock market
- Recommendation system
- Context bandits
- Tensorflow

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introdución

El sector financiero es uno de los negocios más innovadores de los últimos años. Siguiendo el boom de las criptomonedas y la banca digitalizada, la población general está comenzando a mostrar interés por la inversión en renta variable y otros productos financieros disponibles en mercados públicos cuyo riesgo es significativo.

De manera paralela, nuevos y antiguos modelos de inteligencia artificial sub-simbólica están aplicándose con éxito a casos de uso cada vez más complejos. Muchas compañías ya han invertido cuantiosas cantidades de recursos en implementar modelos capaces de resolver, al menos aproximadamente, una de las incertidumbres más grandes de la humanidad en el presente, el futuro de los valores bursátiles.

Ya en el momento en el que se escribió este documento, hay una gran cantidad de "bots" operando en diferentes mercados financieros, sirviendo a los intereses de diferentes entidades o particulares. Algunos ofrecen más rentabilidad a costa de un riesgo proporcional, y es dicha proporción la gran distinción entre la calidad de los agentes.

1.1 Motivación

La principal idea que impulsa el desarrollo de este trabajo es evaluar el rendimiento de una serie de modelos basados en Bandidos Bayesianos. Recientemente se ha demostrado, tanto teóricamente como empíricamente, que su uso junto con *Thompson Sampling* ofrece resultados muy competitivos comparados con otros algoritmos de recomendación.

A parte de la investigación en la aplicación de estos algoritmos de aprendizaje reforzado, también se busca desarrollar una herramienta para ejecutar experimentos acerca de los mismos con menores tiempos de desarrollo.

1.2 Bases de las que parte el trabajo

El proyecto parte de un modelo de aprendizaje por refuerzo conocido como Bandido, específicamente una variante conocida como Bandido Bayesiano. Un Bandido es un algoritmo relativamente simple basado en iteraciones en las que, dada una observación, toma una acción y consecuentemente recibe una recompensa que sirve para entrenar al agente para la siguiente iteración. Tanto las observaciones, como las acciones y las recompensas están acotadas en un espacio vectorial determinado. Hay una gran variedad de políticas que pueden ser usadas para transformar la observación en acción.

El Bandido Bayesiano tiene esta denominación porque su aplica el teorema de Bayes para estimar la función de distribución de probabilidad del valor de recompensa para cada brazo. De manera que luego, se puede efectuar un muestreo sobre dicha distribución para tomar una decisión. El tipo de distribución depende en gran medida del entorno en el que se utiliza el agente. El resultado de haber tomado esta acción luego es usado para generar una distribución de probabilidad a posteriori, que se utilizará en las siguientes iteraciones.

El mercado de valores es un entorno abundante en datos que pueden ser utilizados para entrenar a esta familia de algoritmos. Sin embargo, la complejidad de su comportamiento traerá consigo problemáticas que será necesario solventar con diferentes técnicas, que darán lugar a agentes más complejos, con mayor rendimiento.

1.3 Objetivos

Este proyecto aspira a alcanzar dos objetivos. Siendo el primero el objetivo principal y prioritario, y el segundo un objetivo adicional.

- Estudiar y evaluar, de manera empírica, la viabilidad de la aplicación de modelos de Bandidos Bayesianos Multibrazo al entorno bursátil.
- Desarrollar una aplicación que contribuya a acelerar la experimentación y desarrollo de Bandidos Bayesianos Multibrazo.

A lo largo del desarrollo del trabajo se espera dar respuesta al primer objetivo, y proveer el producto software indicado por el segundo.

Capítulo 2

Fundamentos:

En este capítulo se va a introducir al lector a términos e ideas fundamentales para el desarrollo de este TFG. Los diferentes fundamentos se separan en dos grupos: el problema, es decir, el mercado de valores y la solución, el modelo de Bandido Bayesiano.

2.1 Bandidos multibrazo:

Antes de definir un Bandido, es conveniente introducir el concepto de aprendizaje reforzado, y por alusión, el aprendizaje automático (*machine learning*).

Definición 2.1.1: Aprendizaje automático

Se le denomina aprendizaje automático a una disciplina del campo de la inteligencia artificial que se basa el diseño de algoritmos que sirven un propósito específico, a base de buscar patrones y generar predicciones sobre entorno al que están expuestos.
De esta manera, los algoritmos se optimizan a base de entrenamientos. A medida que son entrenados estos tenderán a ser más competentes, resolviendo cualquiera que fuera el propósito para el que fueron concebidos.

Dicho de otra manera, con aprendizaje automático, la solución no se construye en el propio algoritmo, sino que el algoritmo reúne experiencia con la cual sintetizar la solución al problema.

El aprendizaje automático es una disciplina especialmente relevante en el presente, cada vez se le encuentran más aplicaciones a los algoritmos que siguen este paradigma. Esto se debe a la gran flexibilidad que ofrecen a la hora de integrarlos en nuevos problemas, irresolubles en la práctica por algoritmos de IA simbólica. Gracias a su espíritu declarativo, en el que el desarrollador no tiene porqué saber específicamente como solucionar el problema para poder implementar un algoritmo que lo solvente, su potencial es astronómico.

Definición 2.1.2: Algoritmos de aprendizaje reforzado

Se les denomina algoritmos de aprendizaje reforzado a los se basan en un ciclo durante su entrenamiento que consiste en los siguientes pasos:

- Observar el entorno.
- Llevar a cabo una acción, cuya forma puede variar significativamente dependiendo del algoritmo.
- Recibir retroalimentación (*feedback*) del resultado de haber efectuado la acción. Como consecuencia el algoritmo utilizará esta información para corregir su estado de manera que ofrecerá un mejor rendimiento en futuras acciones.

La idea con el aprendizaje reforzado es conseguir un rendimiento satisfactorio a largo plazo, evitando que el algoritmo se atasque en una tendencia local del entorno.

Qué es un Bandido?

Un Bandido Multibrazo es un algoritmo de aprendizaje reforzado diseñado para resolver problemas en los que hay que tomar decisiones limitadas en un espacio de posibilidades más grande. Como en el caso de los demás algoritmos de aprendizaje reforzado, el algoritmo es entrenado a lo largo de una serie de iteraciones en las que reacciona a una observación y utiliza un *feedback*, que de ahora en adelante denominaremos como ***reward*** (recompensa), para reforzarse y ofrecer mejor rendimiento en el futuro.

Anotación 2.1.1: Origen del nombre de Bandido

El origen del nombre proviene de las clásicas máquinas tragaperras estadounidenses. Estos dispositivos cuentan con un brazo o palanca que se utiliza para probar suerte con ellas. Cada vez que este brazo es utilizado hay una probabilidad desconocida de obtener una recompensa.
Generalmente, en los antros en los que se encuentran estas máquinas, se suelen encontrar una ristra de ellas juntas. La idea del algoritmo viene de considerar a la ristra de máquinas como un único bandido, un Bandido Multibrazo. De manera que se concibe un algoritmo cuyo objetivo es estimar que brazo es el más rentable, es decir el que más recompensa ofrece a largo plazo.

Un ejemplo muy simplificado de Bandido Multibrazo en acción sería el que se muestra en la figura 2.1. En este ejemplo el brazo se encuentra en medio de un período de aprendizaje. Este Bandido dispone de 3 brazos de los cuales ha de escoger uno como su acción a llevar

a cabo. Iteraciones previas le otorgan al bandido cierta capacidad de estimación del valor de *reward* asociado a cada brazo. Sin embargo, el valor real será siempre desconocido hasta el momento en el que se toma la decisión. La decisión que se toma depende de la política que se tome y la naturaleza de la estimación con la que se cuenta. Gran parte de este trabajo se enfoca en buscar políticas y estimaciones lo más adecuadas posibles al problema entre manos.

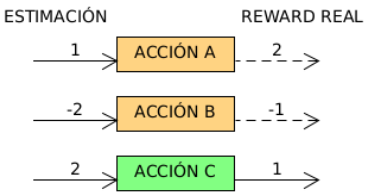

Figura 2.1: Ejemplo simplificado de bandido en una iteración de aprendizaje.

Volviendo al ejemplo, el algoritmo sigue la política de tomar el brazo cuya estimación sea máxima[1]. En consecuencia el brazo C es elegido. Sin embargo, la recompensa ha sido menor de lo estimado, de manera que el método de estimación ha de ser reforzado, para así ofrecer predicciones más fiables en las siguientes iteraciones.

En la versión canónica de este modelo, solo se descubre el valor del *reward* obtenido por el brazo que fue elegido, de manera que los otros brazos no reciben actualización alguna en su estimación.

Que problemas puede solucionar un Bandido Multibrazo?

Como se puede inferir de la naturaleza del algoritmo. Este sirve para distribuir un recurso escaso entre una serie de posibilidades, con la esperanza de obtener una "satisfacción" máxima. Esto puede ser repartir monedas en una ristra de tragaperras con la esperanza de maximizar ganancias, un procesador a una colección de procesos con la esperanza de máxima eficiencia, un cuadro de una página web a una lista de anuncios con la esperanza de maximizar clicks... Los casos son abundantes en muchas áreas en la actualidad.

El algoritmo se adapta al problema de la siguiente manera: el recurso escaso son el número de acciones que puede tomar el Bandido por iteración, las posibilidades son los brazos y la satisfacción el *reward*.

[1] Es decir, una política *greedy*.

Conceptos de relevancia:

Definición 2.1.3: Agente

En este documento se le denomina agente a una implementación funcional de un modelo de Bandido. Esto se debe a que esta implementación es capaz de percibir una observación y reaccionar a la misma con cierto sentido, que es como se define a un agente inteligente.

Definición 2.1.4: Oráculo

Se le denomina oráculo a un agente perfecto que siempre toma la mejor decisión posible en todas las iteraciones a las que se le expone.

Definición 2.1.5: Episodio

Se le denomina episodio a una colección de iteraciones que se usan para entrenar a un agente.

Definición 2.1.6: *Regret*

El *regret* o arrepentimiento es una métrica que se utiliza para evaluar el rendimiento de un Bandido Multibrazo. Para calcularlo se utiliza la siguiente fórmula:

$$regret = \sum_{t=1}^{T} oracle(t) - reward(t)$$

Donde $oracle(t)$ es el *reward* obtenido por un agente oráculo en la iteración t, $reward$ es el *reward* que consigue el Bandido en la iteración t y T es el número total de iteraciones del episodio.

Bandidos Bayesianos:

Como se ha explicado antes, un Bandido Multibrazo recurre a una estimación de *reward* y a una política de selección que utiliza esta estimación para escoger el brazo que será ejecutado en una iteración dada. El Bandido Bayesiano se basa en el teorema de Bayes y la estadística Bayesiana [1] para construir la estimación que será procesada por la política.

Teorema 2.1.1: Teorema de Bayes

$$\mathcal{P}(A|B) = \frac{\mathcal{P}(B|A)\mathcal{P}(A)}{\mathcal{P}(B)}$$

Donde:

- A y B son eventos y $\mathcal{P}(B) \neq 0$
- $\mathcal{P}(A|B)$ es una probabilidad condicional, es decir, es la probabilidad de ocurrir A dado B. Lo mismo aplica a $\mathcal{P}(B|A)$ pero de manera inversa.
- $\mathcal{P}(A)$ y $\mathcal{P}(B)$ son las probabilidades de ocurrencia de A y B respectivamente, sin condiciones previas, es lo que se conoce como probabilidad **a priori**.

El teorema de Bayes es el fundamento en el que se asientan los algoritmos que se van a desarrollar en este TFG. A continuación se va a exponer como se integra esta noción en el modelo de Bandido Multibrazo.

Sea:

- Una lista de brazos.
- *rewards* acotados en un intervalo numérico. Para este ejemplo se asumirá que es discreto.

En el caso del Bandido Bayesiano, la estimación no es un número, sino una función de distribución de probabilidad. Esta distribución, denominada de ahora en adelante como **distribución a priori**, representa las probabilidades del hipotético valor de *reward*. Hay una distribución a priori por brazo, y estas son utilizadas por la política del agente para determinar que acción se va a tomar a continuación.

Una vez se decide que brazo se activa, y se recibe la *reward* real, se utiliza este dato para generar una nueva función de distribución, a la que se le denominara distribución a posteriori o **distribución posterior** de ahora en adelante. Para construir esta distribución se siguen las pautas del teorema de Bayes de la siguiente manera.

Definición 2.1.7: Teorema de Bayes aplicado al Bandido Bayesiano

Sea:

- El *reward* hipotético denominado como r y un valor arbitrario dentro de su intervalo como x.
- Una distribución a priori, es decir una estimación previa al resultado, $Priori(x) = \mathcal{P}[r = x]$.
- Un evento R que representa la obtención de un *reward* real con un valor determinado.
- Una distribución posterior, construida aplicando el conocimiento después de obtener el resultado. $Posterior(x|R) = \mathcal{P}[r = x|R]$. Esta es una probabilidad condicional dado que busca la probabilidad del evento $r = x$ ya habiendo acontecido el evento R.

$$\mathcal{P}[r = x|R] = \frac{\mathcal{P}[R|r = x]\mathcal{P}[r = x]}{\mathcal{P}[R]}$$

Donde $\mathcal{P}[R|r = x]$ es la probabilidad condicional de que se de el evento R, dada la estimación de la distribución a priori. Este término es una distribución en sí, aunque no de probabilidad. A esta distribución se le denominará **función de verosimilitud** (likelyhood) de ahora en adelante $Likelyhood(x|R) = \mathcal{P}[R|r = x]$.

Utilizando esta metodología, es posible perfeccionar la estimación de *reward* a través de las iteraciones. Para ello, la distribución posterior de una iteración pasa a ser la distribución a priori de la siguiente, donde se aplicará la estadística bayesiana de nuevo para obtener una nueva distribución posterior, y así sucesivamente hasta el final del episodio.

Como es lógico, normalmente la distribución a priori y a posterior pertenecen a la misma familia de distribuciones de probabilidad (p. ej. distribución gaussiana). De manera que a ambas distribuciones se les denomina distribuciones conjugadas. A la distribución prior se le denomina como prior conjugada (*conjugate prior*) [2] si esta ofrece una forma cerrada[2] (*closed-form expression*) para el cálculo de la distribución posterior. Hay una serie de familias que se relacionan con otras de esta manera [3] (p. ej. verosimilitud - experimento de bernoulli / prior y posterior - distribución beta).

[2] Una expresión matemática que usa una cantidad finita de operaciones elementales.

Anotación 2.1.2: Distribución conjugada

De ahora en adelante se le denominará distribución conjugada a una familia de distribuciones que son prior conjugadas con una función de verosimilitud determinada.

Es posible llevar a cabo este cálculo usado distribuciones a priori no conjugadas, el hecho de que lo estén es una mera conveniencia que simplifica el algoritmo. Al no estar conjugadas el uso de integración numérica para resolver las iteraciones es una probable necesidad.

Exploración *versus* explotación:

Un gran dilema que siempre se presenta, explícita o implícitamente, en los problemas a los que se aplican Bandidos es la decisión entre la explotación y la exploración.

Se le denomina **explotación** cuando la política de un agente selecciona siempre los brazos cuya estimación de *reward* es máxima, con el objetivo de obtener el mejor rendimiento posible en el momento presente. Por otro lado, la **exploración** es un comportamiento de la política del agente en el que se buscan nuevos máximos en brazos cuya estimación no tiene porque ser máxima, de manera que se puede obtener conocimiento de brazos potencialmente más rentables que los que se consideran en el presente.

Un Bandido que se enfoque completamente en la **explotación** será sub-óptimo prácticamente en todos los episodios en los que se ponga en práctica, debido a que se enfocará en el primer máximo local que tenga al alcance. En caso contrario tampoco se obtendrán buenos resultados, ya que el algoritmo atraviesa el episodio sin buscar rentabilidad en ningún momento, priorizando el conocimiento a los resultados.

Todo Bandido ha de integrar una política que reparta la toma de decisiones entre estas dos prioridades, de manera que el algoritmo busque maximizar las *rewards* sin dejar de lado la búsqueda de brazos con mayor potencial. Esto hace del Bandido un algoritmo adaptable a entornos extensos y/o no estacionarios.

Políticas para elección de brazos:

En la anterior sección se explicó como se construyen las estimaciones utilizadas en un Bandido Bayesiano. En esta sección se expone cómo, a partir de estas estimaciones, previas a un resultado, se utilizan para decidir que brazo (o brazos) se van a activar en una iteración dada. Al proceso de elección se le denomina comúnmente política del Bandido o del agente. Hay una gran cantidad de políticas desarrolladas, entre las más utilizadas están: ϵ-greedy, una política que se basa en un hiperparámetro ϵ para determinar si se elegirán los brazos con mejores estimaciones de *reward* o se explorarán brazos al azar en busca de máximos encubiertos, **UCB** (Upper Confidence Bound), una política algo más compleja, cuya explicación

escapa del propósito de esta sección o **Thompson Sampling**, la política que se utilizará de manera exclusiva en todos los experimentos documentados en esta memoria.

Definición 2.1.8: Thompson Sampling

Thompson Sampling es uno de los métodos más arcaicos para resolver el dilema exploración-explotación. Aunque esto sea así, experimentos y artículos recientes han demostrado un rendimiento muy competitivo y estable comparado a otras políticas más modernas.

$$a(t) = argmax(\tilde{X})$$

Donde $\tilde{X}$ es un vector con los diferentes muestreos de cada distribución prior que representa cada brazo. $a(t)$ es el brazo elegido en la iteración t.
Es decir, dadas unas estimaciones, que han de ser distribuciones de probabilidad, correspondientes a una colección de brazos. Se procede a muestrear cada una de las distribuciones, la distribución cuyo muestreo sea máximo (asumiendo que se busca maximizar la *reward*) será considerada elegida, de manera que su brazo será activado en la iteración presente.

2.2 Valores bursátiles y mercado público:

El objetivo principal de este TFG es la aplicación de modelos de Bandidos Bayesianos Multibrazo al entorno bursátil a través de una serie de experimentos, con la esperanza de obtener un sistema de recomendación para la inversión en los valores observados.

Mercado público de valores:

El mundo de las finanzas está experimentando un auge de crecimiento de gran magnitud en el presente. Desde la integración de las finanzas en el mundo digital, la banca online y los métodos de pago alternativos, las finanzas personales y profesionales evolucionan a un ritmo comparable al de la tecnología que las soporta.

Algunos de los conceptos que se van a exponer en esta sección, aunque relevantes en el presente, tienen siglos, sino milenios de antigüedad.

Definición 2.2.1: Mercado público

Se le denomina **mercado público** a una plataforma en la cual usuarios[a] pueden proceder a ofertar o demandar un producto a un precio determinado. Estas ofertas o demandas se les denomina como órdenes de venta y de compra respectivamente.
Cuando una orden de compra y otra de venta coinciden en el precio determinado por el producto subyacente, se produce una transacción en la cual el que ofertó el producto recibe el valor acordado en una divisa acordada y el que demandó recibe el producto en sí. El mercado es público porque por norma general toda entidad capaz puede registrar órdenes en el mercado.

[a] Individuos o personas judiciales.

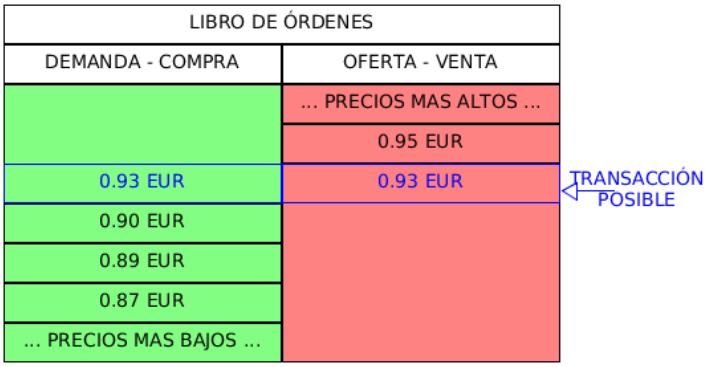

Figura 2.2: Ejemplo simplificado de mercado para un producto específico.

Anotación 2.2.1: Ejemplo simplificado de mercado público

Sea este un mercado, con un solo producto, limones. Una orden representa el valor que le da un usuario a un limón y el tipo de orden la intención del usuario, es decir, compra o venta. La figura 2.2 representa las órdenes en este mercado.
Cuando dos órdenes de compra y venta respectivamente coinciden en valor, se ejecuta la transacción, intercambiando así el producto por su valor acordado. Las dos órdenes se eliminan del libro y se actualiza el precio de mercado del limón al precio en el que se acordó la transacción.
Por lo tanto, el precio de un limón en el mercado público de limones es de **0.93** euros.

Un mercado como el que se ejemplifica es capaz de regular de manera natural el precio de sus productos. Por ejemplo, si hay gran demanda de limones, los compradores estarán dispuestos a elevar el precio de compra para competir con las órdenes de los otros compradores para así, obtener su preciado limón.

De este concepto de mercado surge el mercado bursátil en el cual, en lugar de comerciar con limones, se comercia con instrumentos financieros.

Instrumentos financieros:

Definición 2.2.2: Instrumento financiero

Un instrumento financiero es un un contrato entre 2 entidades al que se le puede dar valor y por tanto comprarlo y venderlo. Normalmente la adquisición de un instrumento de este tipo implica que el comprador toma la responsabilidad de una de las partes.

En este TFG se van a usar acciones como sujetos de experimentación.

Definición 2.2.3: Acción

Una acción en el contexto bursátil es un contrato por el cual una entidad es dueña de un porcentaje de otra entidad. En este contrato viene establecido cuál es el porcentaje de posesión. Generalmente, la entidad poseedora obtiene una serie de derechos sobre la poseída.

Las acciones se establecen y venden en primera instancia por la entidad cuya posesión va a ser ofertada. Una vez adquiridas de esta entidad, las acciones se pueden comerciar en un mercado público, denominado bolsa. Dependiendo del valor percibido de la entidad poseída, fluctuará el valor de sus acciones.

Al representar el histórico de precios de mercado en un período de tiempo, se puede observar la progresión del precio e incluso especular acerca del futuro precio de la acción. En la

figura 2.3 se muestra un ejemplo en el que se observan los precios de Apple Inc.

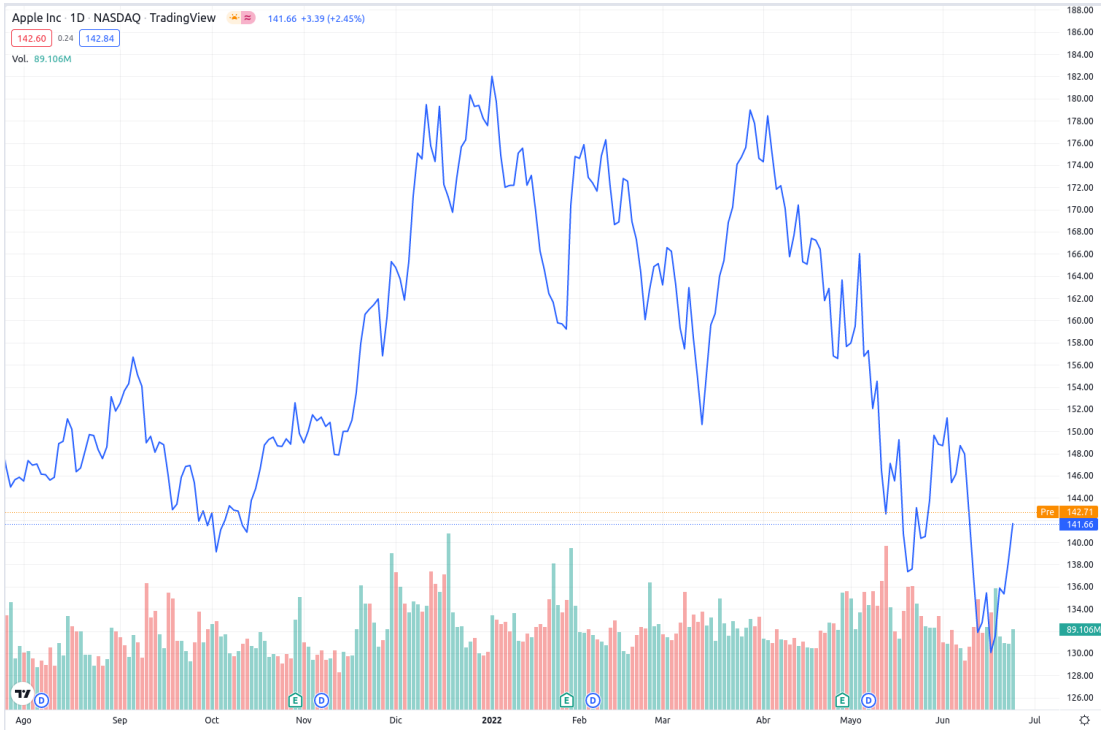

Figura 2.3: Histórico de valores de las acciones de Apple Inc. (línea azul), intervalos diarios. Gráfico obtenido de tradingview.com

Existe una gran variedad de productos financieros, algunos más arriesgados que otros, con su consecuente rentabilidad proporcional al riesgo que someten.

Tendencias y comportamiento de mercados públicos:

Como hemos visto en los apartados previos, la bolsa parte de principios relativamente simples. Es la interacción de miles de millones de entidades con el mercado la fuente de complejidad e incertidumbre, la cual que supone un imponente desafío para humanos y máquinas por igual.

El precio percibido de un instrumento en un día cualquiera puede verse afectado por una infinidad de factores, tanto relativos a precios pasados como a eventos externos o opiniones de influencia.

Por lo general, hay grupos de valores bursátiles que tienden a comportarse de manera similar a lo largo del tiempo. En la mayoría de los casos, los instrumentos financieros tienden a subir de precio a largo plazo, propulsados la inflación y el crecimiento macroeconómico. A corto plazo, los valores son más susceptibles a oscilar de maneras poco predecibles.

Aplicación de Bandidos al entorno bursátil:

En el la sección 2.1 se especifica que los modelos de Bandido Multibrazo se aplican a problemas en los que hay:

- Un recurso escaso.
- Una serie de posibilidades en las que depositar dicho recurso.
- Un objetivo que se satisface más o menos, dependiendo de las posibilidades que son escogidas.

La adaptación del modelo al entorno es de lo más simple. El recurso escaso es la inversión, las posibilidades son los diferentes valores bursátiles en los que se puede invertir, el objetivo es maximizar el retorno de la inversión a lo largo del tiempo.

En los experimentos documentados en esta memoria, cada brazo de un Bandido representa a un valor bursátil, concretamente a una acción del S&P500. El recurso escaso será una inversión indivisible, de manera que solo es posible depositarla en un brazo por iteración. El período de tiempo que transcurre entre iteraciones se determinará en cada experimento.

Capítulo 3

Modelos: Implementación, Descripción y Resultados

En en este capítulo se van a exponer los diferentes experimentos que se llevaron a cabo durante el desarrollo del trabajo. Cada experimento es una extensión en algún aspecto del experimento que le precede. Intercalado entre los experimentos se encuentran diferentes planteamientos que posibilitan la implementación de los consecuentes. Todo experimento se diserta en 4 partes.

- Motivación: la razón detrás de que este experimento se lleve a cabo.
- Condiciones del experimento: consideraciones previas acerca del entorno y los objetivos del agente.
- Metodología: descripción de cómo se llevó a cabo el experimento.
- Resultados: datos y figuras acerca de las conclusiones del experimento.

3.1 Experimento A: Bernoulli MAB con datos sintéticos

Motivación

Este es el primero de los experimentos y por tanto, sirve como una introducción al bandido multibrazo bayesiano. En este, se desarrollarán soluciones para entornos muy simples, con la finalidad de enfocarse entender el funcionamiento del modelo, sus capacidades y sus limitaciones. Ningún entorno bursátil se tendrá en cuenta en este experimento debido a su intrínseca complejidad, la cual alejaría al experimento de sus objetivos.

Condiciones del experimento

Se va a partir de un entorno compuesto de 4 brazos, cuyas *rewards* son binarias. Cada brazo tiene una probabilidad θ_i determinada, que no cambia a lo largo del episodio del experimento. Los diferentes valores que puede tener una *reward* son:

- **1**, si hay beneficio, es decir, el resultado es el deseado. La probabilidad de que un brazo retorne este valor al ser elegido es θ_i. Siendo i el brazo en cuestión.
- **0**, si no hay beneficio, el resultado es indeseado. La probabilidad de que un brazo retorne este valor al ser elegido es $1 - \theta_i$.

Dadas estas condiciones, la variable aleatoria de cada brazo es un experimento de Bernoulli cuyo valor p es θ_i. Por lo tanto, estimar la probabilidad de obtener x recompensas de un brazo en n iteraciones es el resultado de una distribución binomial (de probabilidad θ_i), ya que esta es una consecución de dichos experimentos de Bernoulli en los que los resultados son independientes. Esta independencia también es una condición inicial del experimento.

En un segundo apartado se va a probar un entorno similar al anterior, con la diferencia que los valores θ_i no son estáticos sino que varían en función del tiempo.

Metodología

Se va a implementar un Bandido Bayesiano Multibrazo en R. Se ha escogido este lenguaje debido a su colección de utilidades relevantes para este experimento y la rapidez de implementación.

La distribución a priori es un distribución beta ($alpha$ = 1, $beta$ = 1) dado que es la distribución conjugada a priori de la binomial. Es con la distribución binomial con la cual obtenemos la función de verosimilitud necesaria para calcular la función de distribución de probabilidad condicional posterior:

$$\mathcal{P}(Evidencia|Hecho) = \frac{\mathcal{P}(Hecho|Evidencia)\mathcal{P}(Evidencia)}{\mathcal{P}(Hecho)}$$

Se puede probar [3] que dadas:

$$fbeta_{prior}(\alpha, \beta), fbeta_{posterior}(\alpha', \beta')$$

$$\alpha' = \alpha + r, \beta' = \beta + 1 - r$$

Donde r es el *reward* de una iteración determinada.

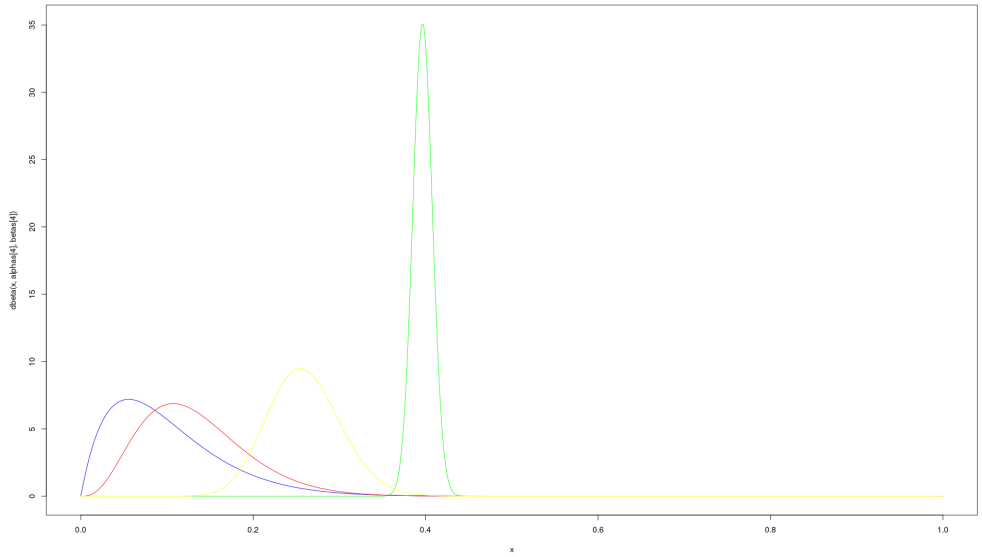

Figura 3.1: Distribuciones posteriores de los 4 brazos del bandido.

Resultados: Apartado 1, θ_i estáticos

Los valores θ_i son estáticos durante todas las iteraciones. El bandido elige un brazo por iteración. Hay 2000 iteraciones.

Los valores θ son:

θ_{Azul}	θ_{Rojo}	$\theta_{Amarillo}$	θ_{Verde}
0.1	0.2	0.3	0.4

Al finalizar el episodio de 2001 iteraciones, las distribuciones posteriores de los diferentes brazos tienen el aspecto mostrado en la figura 3.1. Como se puede apreciar, las medias de las distribuciones posteriores son relativamente próximas a los valores θ_i de cada brazo.

De 2001 iteraciones:	Azul	Rojo	Amarillo	Verde
Veces escogido	18	28	106	1849
Porcentaje	0,9%	1,4%	5,3%	92,4%
Veces rentable[1]	1	3	27	733
Porcentaje (sobre veces escogido)	5,5%	10,7%	25,4%	39,6%

En cuanto a *regret*, al ejecutar repetidos episodios de entrenamiento la media es mayor

que cero. Cabe destacar que hay un número considerable de casos en los que el *regret* es negativo. Esto se debe a que la simplicidad del experimento descubre claramente cualquier anomalía estadística de la distribución binomial usada para obtener la *reward*. Lo cual resulta en rentabilidades por encima de la estimación real.

Resultados: valores θ_i variables

En este experimento los valores θ_i varían en función del tiempo siguiendo las curvas de bezier mostradas en la figura 3.2

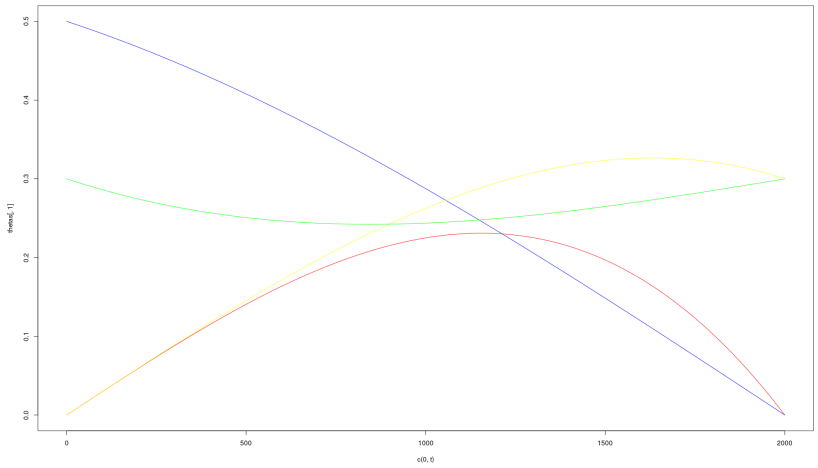

Figura 3.2: θs de los diferentes brazos en función del tiempo.

Como se puede apreciar, el brazo azul tiene más posibilidades de aportar beneficio que los otros al principio del período. Sin embargo, sus posibilidades se reducen progresivamente, acabando por ser el brazo menos beneficioso. Por otro lado, el brazo amarillo incrementa su rentabilidad llegando a ser el más beneficioso a finales del período. Después del entrenamiento, las distribuciones posteriores tienen al aspecto mostrado por la figura 3.3.

Aparentemente, el brazo azul ha sido el más explotado, esto se puede apreciar al ver lo concentrada que es su distribución posterior, lo que es proporcional a la asiduidad de su elección. El brazo amarillo, que ofrece más rentabilidad al final del período, supera en magnitud a la distribución azul en valores de abscisas. Los otros dos brazos, que son menos rentables en general, fueron menos explotados.

En cuanto a *regret*, se han obtenido las *regrets* de 2000 entrenamientos diferentes, de los cuales se destila un *regret* medio de **138.77**, un máximo de **236.3** y mínimo **16.3.** Los diferentes *regrets* se pueden apreciar en la figura 3.4

Como es natural, el Bandido Bayesiano utilizado en los experimentos es menos eficaz cuando los diferentes brazos tienen una probabilidad de recompensa variable. La recompensa

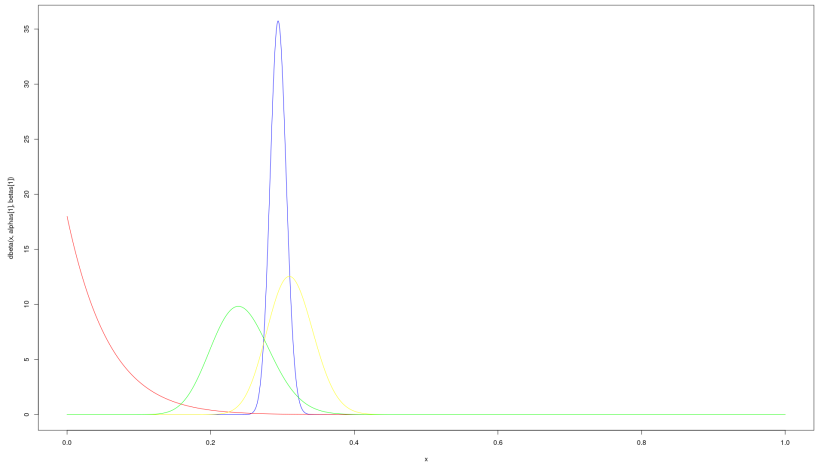

Figura 3.3: Distribuciones posteriores de los 4 brazos del bandido.

máxima esperada es **713.6** si se escoge siempre el brazo con mejor probabilidad[2], por lo tanto el bandido tiene un rendimiento [3] medio del **80,5%**.

Definición 3.1.1: Métrica de rendimiento

El rendimiento es igual a la recompensa obtenida por el bandido dividido la recompensa obtenida por el oráculo. Esta métrica se utilizará en experimentos con *rewards* discretas.

3.2 Experimento B: Bernoulli MAB con datos reales

Motivación

El anterior experimento establecía unas bases de la aplicación del modelo de Bandido Multibrazo Bayesiano en un entorno sintético, basado en una simplificación de valores bursátiles. En este experimento se va a aplicar el modelo a datos bursátiles reales. En concreto se va a utilizar un extenso *dataset* del S&P500. La idea es entender como afecta al rendimiento la incertidumbre de las acciones. También se evaluará el rendimiento para selecciones con diferentes períodos de retroalimentación. Cabe destacar que este rendimiento no se corresponde con los intereses del usuario final[4]. Para ello, hará falta introducir mejoras que se darán en los próximos experimentos.

[2] A un agente que alcance este rendimiento se le denominará oráculo en posteriores experimentos.
[3] Véase Definición 3.1.1
[4] Dado que las *rewards* son binarias, no se valora una subida del 10% y una del 2% de diferente manera. Un inversor sí lo hace.

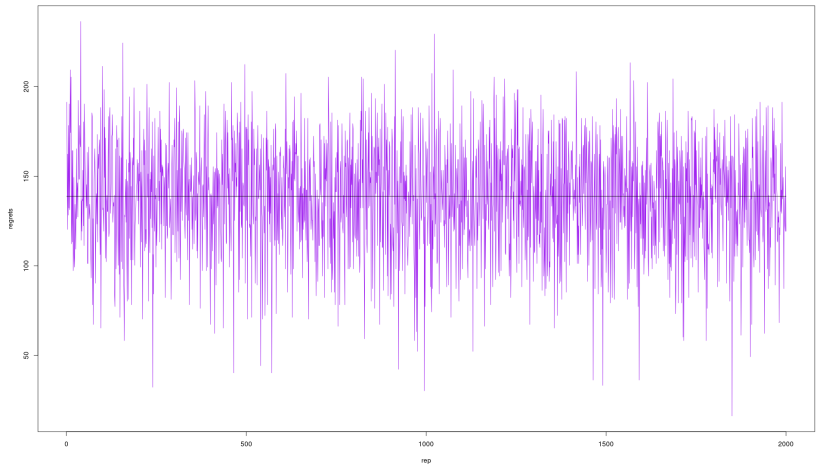

Figura 3.4: Regrets de 2000 entrenamientos diferentes en mismas condiciones

Condiciones del experimento

En los anteriores experimentos el *reward* de un brazo venía dado por el resultado de un experimento de Bernoulli cuya probabilidad podía ser estática o variar con el tiempo. Ahora que partimos de datos de valores bursátiles, necesitamos traducir los datos a una secuencia de *rewards* binarios. Otra complicación es que la definición del modelo de Bandido utilizado no contempla casos en los que el *reward* no se obtiene instantáneamente después de la elección de un brazo, es más, es una precondición que esto se produzca. Para solucionar ambos obstáculos se discretizará la línea temporal, colapsando elecciones y respuestas en un mismo punto, ignorando así el retraso entre la acción y la *reward*. Para efectuar el proceso, debemos considerar las siguientes sentencias:

- La *reward* debe ser obtenida en un período de tiempo determinado.
- Si en dicho período, el valor bursátil es mayor en el final que el el principio, la *reward* asociada a ese período es 1.
- Si el valor bursátil es menor o igual en el final que en el principio, el *reward* asociado a ese período es 0.

Por lo tanto, la función que dado un día i y una longitud de período (en días) p es igual a la *reward* en la iteración del día i es la siguiente:

$$reward(i,p) = \begin{cases} 1 & \text{si } (v_{i+p} - v_i) > 0 \\ 0 & \text{si } (v_{i+p} - v_i) \leq 0 \end{cases}$$

Hay una dificultad más que debe ser atendida, esta es la posibilidad de que algunos brazos estén cerrados, para ello será necesario omitir la posibilidad de elección de los mismos en la iteración en la que presentan esta condición. El caso más común de esta ocurrencia es los fines de semana, en los cuales la bolsa esta cerrada.

Metodología:

Se dividirá el experimento en dos apartados:

- Evaluar el funcionamiento en un solo brazo. Esto será una mera prueba de que el algoritmo es compatible con el entorno provisto.
- Evaluar el funcionamiento con múltiples brazos. El dataset utilizado alberga 7163 archivos CSV que representan información acerca de diferentes valores bursátiles pertenecientes o que pertenecieron al S&P500, en este experimento se usan dos parámetros presentes en todos los archivos: los valores de cierre y las fechas de los mismos. Con estos se sintetizarán unas listas de pares (una por brazo) que enlazarán valores de *reward* a tiempos dados. Estas listas serán consumidas por el agente.

  En este apartado se comprobará el rendimiento del modelo en un entorno real y se comparará con el rendimiento del entorno sintético. A partir de las conclusiones extraídas de este apartado se procederá a mejorar el modelo para, en consecuencia, mejorar el rendimiento.

En ambos casos se implementará un Bandido Bayesiano Multibrazo en R, escogido este lenguaje por las mismas razones que en el experimento anterior. Las distribuciones a priori y posteriori son Beta y la funciones de verosimilitud son experimento de Bernoulli (al igual que en el experimento anterior).

El entrenamiento consiste en iterar desde la fecha de fundación del S&P500 hasta la última fecha registrada por el dataset, el número de iteraciones depende de la longitud del período de recompensa. En cada iteración se evaluarán los brazos del bandido que son elegibles para el muestreo y se escogerá uno de los brazos siguiendo las mismas técnicas que en los otros experimentos. Un brazo es elegible si tiene un valor de cierre en la fecha de la iteración (de no cotizar ese día no hay valor de cierre).

Resultados: Apartado 1, un solo brazo, Alphabet Inc.

El resultado no se puede mostrar en términos de *regret* o *reward* debido a que el modelo utilizado es un Bandido Multibrazo. El brazo utilizado se puede apreciar en la figura 3.5

En cuanto a rentabilidad, el bandido parece encontrar mejores tasas de beneficio en proporción con el tamaño del período de tiempo considerado. Esto es porque la tendencia general

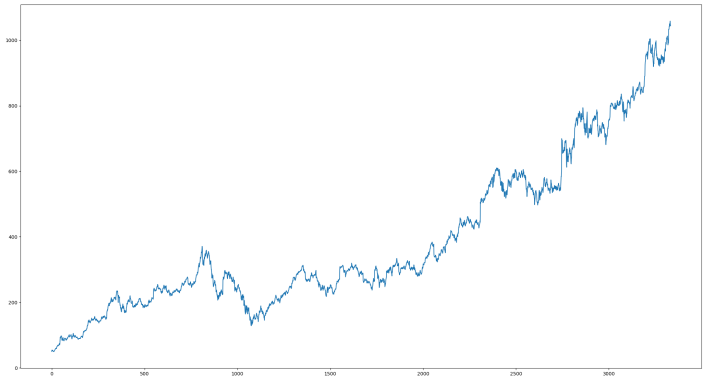

Figura 3.5: Valores diarios de Alphabet en el *dataset*

es creciente y al usar períodos grandes el bandido "esquiva" las tendencias bajistas locales. Esto es fácil de apreciar en la figura 3.5, es más probable encontrarse en alza cuando más tiempo haya entre inversión y retorno. Es importante destacar que aunque las tasas lleguen a ser altas, no hay información de la magnitud de beneficio hipotético ya que el bandido considera como un éxito rotundo tanto una subida del 0.1% como del 20% (recompensas binarias).

Resultados: Apartado 2, todos los brazos del dataset (S&P 500)

Los resultados, para diferentes períodos de tiempo entre iteraciones, son los siguientes:

Longitud del período	Regret	Rendimiento	Media de *reward*
1 Día.	7750	44.89%	0.46
1 Semana.	184	52.08%	0.508
1 Mes (30 días).	23	60.34%	0.543

Este experimento revela una serie de problemas del algoritmo. Comenzando con el rendimiento, se aprecia un rendimiento bajo en períodos diarios. Para mejorarlo el es necesario aplicar reformas y extensiones al modelo de Bandido. Sin embargo, antes de poder llevar esto a cabo es necesario cambiar la definición de *reward*, ya que esta no se alinea con el interés real del usuario. Este cambio se llevará a cabo en los experimentos D y E.

En la figura 3.7 muestra otra incidencia. A medida que la cantidad de brazos aumenta, la exploración de cada brazo se reduce. Esto se debe a que solo se obtienen evidencias nuevas en un brazo por iteración y a que la proporción de iteraciones por brazo en el ejercicio es baja. Al incrementar el período de recompensa la cantidad de iteraciones se reduce y las distribuciones

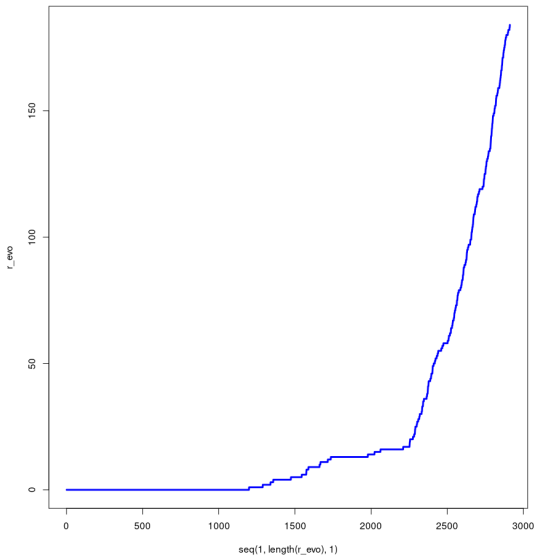

Figura 3.6: Evolución del *regret*, períodos semanales.

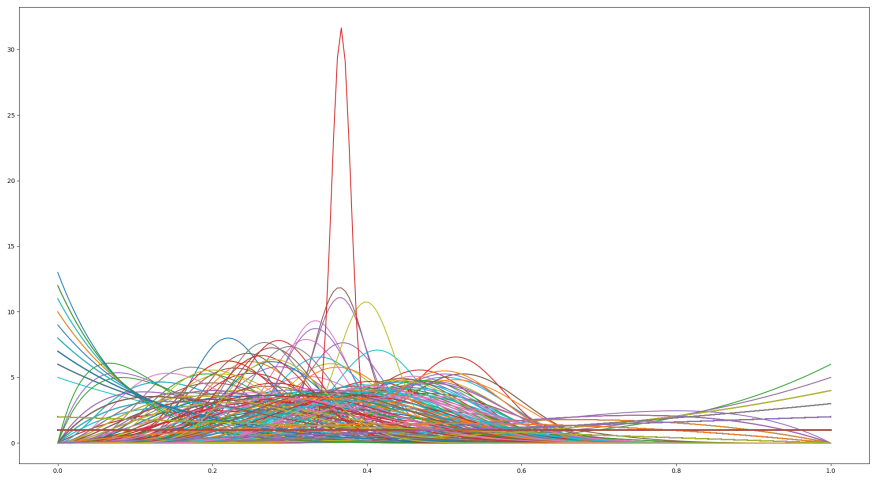

Figura 3.7: Distribuciones posteriores de todos los brazos.

posteriores de los brazos pierden precisión considerablemente. Una posible solución a este problema sería elegir varios brazos por iteración para aumentar el número de evidencias [4]. La otra opción es limitar el número de elecciones por ejercicio en función de la cantidad de iteraciones.

También cabe destacar el hecho de que muchos de los brazos son explotados sin utilidad aparente. Esto se debe a que hay valores bursátiles que no cotizan en el "presente" y por tanto la información obtenida no tiene uso posible. Además, consumen iteraciones que se podrían haber usado en valores útiles.

Otra limitación es la plataforma que ejecuta el algoritmo. Hasta ahora el modelo ha sido relativamente sencillo, sin embargo, ya se empieza a percibir un consumo de tiempo considerable cada vez que se lleva a cabo un ejercicio. En los siguientes experimentos se recurrirá a tecnologías más sofisticadas para llevar a cabo los experimentos con mayor eficiencia.

3.3 Experimento C: Bernoulli MAB implementado en Tensorflow

Motivación

El experimento que se va a presentar a continuación va a implementar el mismo modelo de Bandido Bayesiano Multibrazo que en el anterior. La principal adición del mismo va a radicar en cambios en el entorno con que se alimenta al agente y en las tecnologías subyacentes.

Para implementar el experimento se va a utilizar Tensorflow en Python. Las principales motivaciones que inducen este cambio son las siguientes:

- Tensorflow cuenta con un kit de operaciones y estructuras de datos cuya eficiencia es considerablemente superior a R. Implementar experimentos con esta biblioteca acelerará el curso del trabajo en general ya que las pruebas requerirán menos tiempo.
- La aplicación demostrativa requerirá de agentes implementados en Tensorflow para Python debido a su eficiencia y flexibilidad. Esta primera implementación servirá de base para familiarizarse con el paradigma de la biblioteca y sus métodos de desarrollo. A partir del diseño de este experimento, se construirán las mejoras de los siguientes, desembocando en modelos listos para ser integrados en la plataforma final.

Condiciones del experimento

En este experimento el entorno es un conjunto de valores bursátiles, los cuales pueden estar abiertos o cerrados dependiendo del día del año. Lo que implica esto es que un valor que no cotiza en un día determinado no puede ser elegido, ni tendrá ninguna evolución en

su valor (ese día) y por tanto esta cerrado. En caso contrario esta abierto. Los objetos que se transmiten entre el entorno y la política de elecciones (acciones)[5] son los siguientes:

- **Acción:** un número entre el -1 y el número total de brazos menos 1. Si el número no es negativo, la acción representa el brazo escogido por la política. Si es -1 implica que todos los brazos están cerrados en la iteración y por lo tanto, no se puede elegir ningún valor en el que invertir (ej. es domingo). Al ser la acción -1 se entiende que el agente toma la decisión de no hacer nada.
- **Observación:** alberga un vector de booleanos, cada elemento representa si el brazo es elegible o no en la siguiente iteración. El agente utilizará este vector como una máscara para filtrar acciónes deshabilitadas.
- ***Reward*:** puede ser 0 o 1 al igual que en los experimentos anteriores. Comunica si el valor bursátil del brazo ha subido en esta iteración, en cuyo caso será 1. En términos de arquitectura del experimento, la *reward* forma parte de la observación junto con el vector de booleanos.

Metodología

Para implementar el experimento se usará la biblioteca TF-Agents. Esta biblioteca perteneciente a TensorFlow divide la implementación de un Bandido (agente) en 3 elementos principales (figura 3.8).

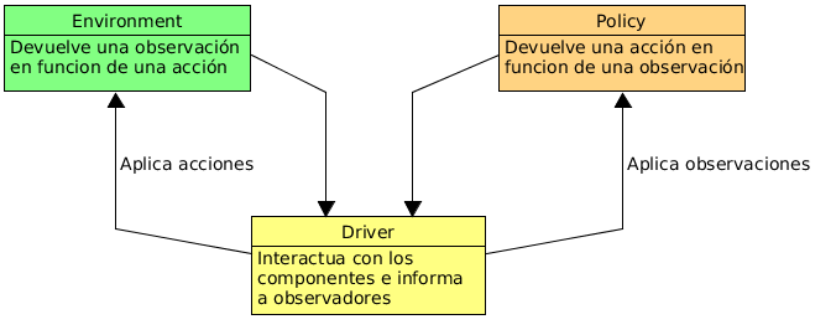

Figura 3.8: Elementos del agente

El entorno en este modelo concreto es una entidad que genera los datos con los que interactúa el agente. Estos datos consisten principalmente una tupla por iteración, la cual consta de una observación y una *reward* (correspondiente a la acción de la iteración anterior).

La *policy* o política, es la entidad que procesa la información, toma decisiones y aprende de las *rewards* de las mismas. Es el corazón del Bandido, cambiar la política supone cambiar

[5] Ambos términos significan lo mismo en este contexto

el modelo de elección del bandido, en este caso es **Thompson Sampling**. Hay otros modelos muy utilizados como **LinUCB** o ϵ**-greedy**. Más información acerca de la política del agente se puede consultar en la sección 2.1.

El driver es, a grandes rasgos, un bucle "inteligente" que lleva a cabo el experimento. Este comunica la observación a la *policy* y la acción al entorno de manera sucesiva a lo largo de la ejecución de un episodio. Esta entidad también se encarga de informar a los observadores con datos del experimento en tiempo real. El formato de estos datos suele denominarse trayectorias. Son tuplas de información del entorno y la política en una iteración concreta.

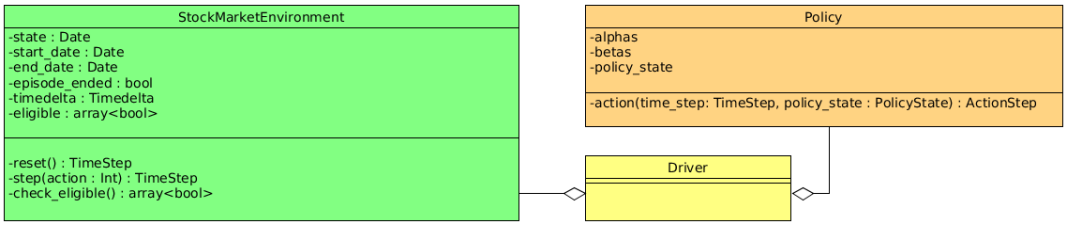

Figura 3.9: Atributos específicos de este modelo

En el caso concreto de este experimento los características especiales de las diferentes clases que conforman el modelo se pueden ver en la figura 3.9.

Una interesante consideración es que se podría tener en cuenta a este agente como un Bandido Bayesiano Multibrazo **con contexto**. Esto se debe a que se está usando una observación para alterar la política de selección, concretamente para prohibir algunas de las acciones. Los Bandidos sin contexto solamente requieren de *rewards* para funcionar como es debido. Aunque teóricamente lo expuesto tendría sentido, un Bandido con contexto suele utilizar la observación para alterar la percepción de *reward* potencial, bien a base de sesgos en el muestreo o antes de generar la distribución de verosimilitud. Por otro lado este Bandido descarta posibilidades de manera absoluta, por razones justificadas por supuesto.

Resultados

A continuación se mostrarán los resultados de un episodio de ejecución de la implementación resultante.

Hubo un total de **10978** iteraciones en las que se eligió un brazo como acción. La media de rendimiento durante el entrenamiento ha sido de un **67.5%**. El rendimiento en la última iteración es del **61.8%**. En la **figura 3.11** se aprecia la tendencia del mismo. El rendimiento se estabiliza por encima del 60% e incluso asciende en el final del entrenamiento. El principio de la gráfica es poco relevante debido a su inestabilidad entre entrenamientos.

Los datos obtenidos dan a entender que el crecimiento de la *regret* en relación al número de iteraciones es subpolinómico (véase 3.11). La media de medianas de los brazos es de **0.4737**.

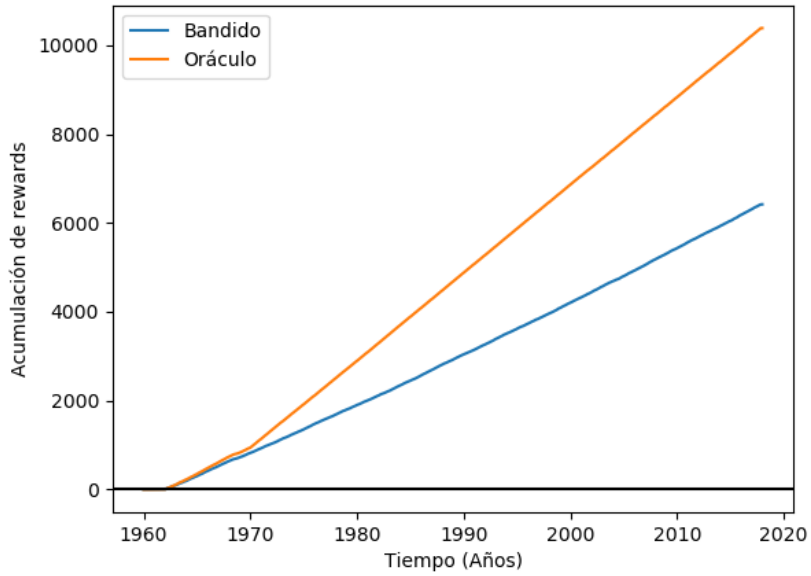

Figura 3.10: Evolución de rewards acumulados del oráculo y del bandido

El número de brazos que son elegibles en cada iteración puede verse en la figura 3.12. Y los 9 brazos más explotados (del 2º al 10º), exceptuando el primero (ya que debido a irregularidades en el conjunto de datos se elige un número anormal de veces), se pueden ver en la figura 3.13.

Los resultados del experimento C son mejores que el los del experimento B, aún utilizando la misma política de selección. La diferencia radica en el entorno, ya que en este último experimento la manera en la que se construyen los datos es diferente.

La velocidad en la oscilan las ejecuciones del experimento C es muy superior[6] a la del experimento anterior. Aunque esto no es de especial importancia en la experimentación actual, si lo será más adelante, cuando sea necesario ejecutar muchos episodios diferentes.

Aparte de los hechos previos, las limitaciones que padece este modelo son exactamente las mismas que el del experimento anterior, lo cual era lógico y predecible. Los siguientes experimentos aportarán mejoras al rendimiento del modelo.

[6] Alrededor de un orden de magnitud.

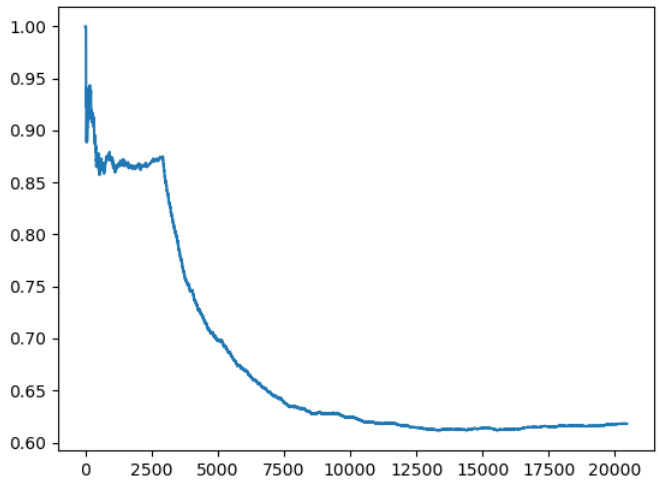

Figura 3.11: Evolución del rendimiento a lo largo del entrenamiento

3.4 Experimento principal (D): MAB con distribuciones Gaussianas

Motivación

Los modelos utilizados en los anteriores experimentos funcionan razonablemente bien en entornos más simples. Sin embargo, usar *rewards* binarias para entrenar a un bandido cuyo entorno son valores bursátiles supone una discordancia entre el objetivo del agente y el del usuario. Dicho de otra manera, el usuario desea obtener beneficios, a mayor magnitud mejor. El agente "busca"[7] elegir valores que acaben en positivo al terminar un período determinado, sin importar la magnitud de la subida (o de la bajada).

Condiciones del experimento

En los modelos implementados en los anteriores experimentos el bandido era teóricamente similar a la figura 3.14.

Donde:

$$\alpha_i' = \alpha_i + reward$$

$$\beta_i' = \beta_i + 1 - reward$$

$$Beta^{8} : \mathbb{N} \times \mathbb{N} \to ([0,1] \in \mathbb{R} \to [0,1] \in \mathbb{R})$$

[7] El agente es un algoritmo que trata de obtener el máximo *reward* acumulado posible.
[8] Aportar los parámetros de un brazo a Beta devuelve la función de distribución de ese brazo

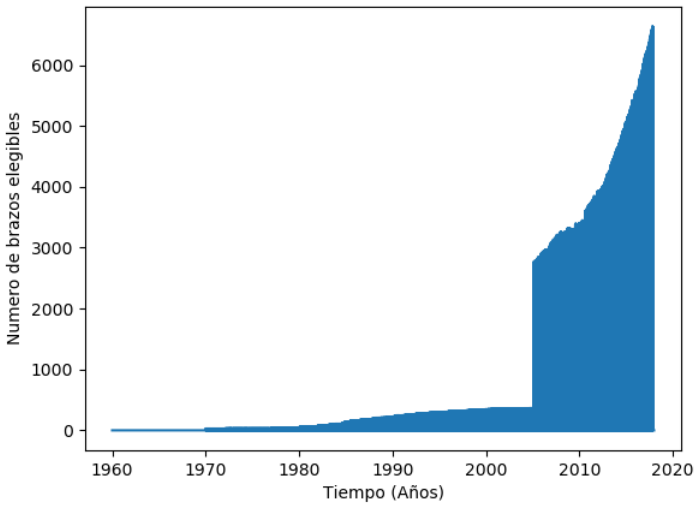

Figura 3.12: Brazos abiertos a lo largo del tiempo

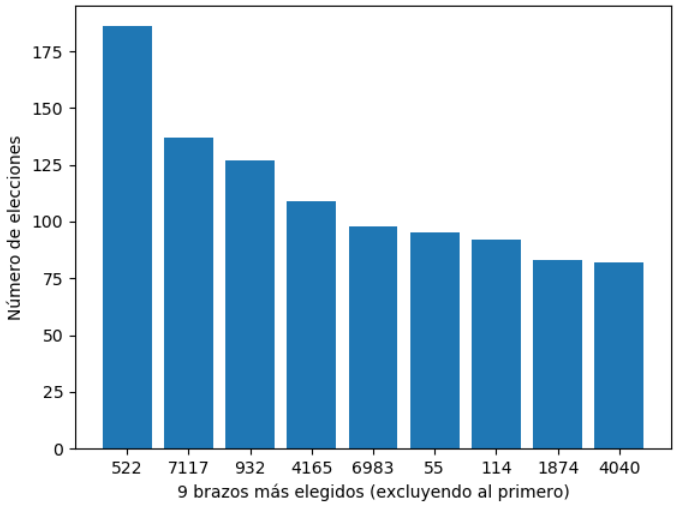

Figura 3.13: 9 brazos más elegidos (excluyendo el primero)

Las distribuciones de todos los brazos, tanto prior como posterior, son distribuciones beta cuyo parámetro varía según el comportamiento de cada brazo a lo largo de un episodio. La distribución a priori inicial de todos los brazos es $Beta(\alpha, \beta), \alpha = 1, \beta = 1$.

El uso de la distribución beta en este modelo se debe a que es la distribución conjugada de la binomial, la cual produce resultados discretos que están entre 0 y 1. Dado que las *rewards* han sido hasta ahora binarios, el valor de una *reward* a la hora de tomar una acción puede ser

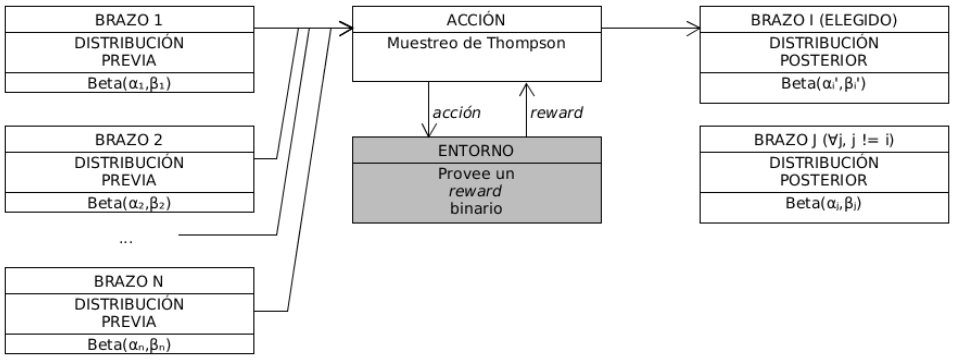

Figura 3.14: Funcionamiento del bandido en una iteración

considerado como el resultado de un experimento de Bernoulli.

En consecuencia a lo anteriormente expuesto en la motivación, es necesario utilizar *rewards* no binarias, concretamente, continuas. Para poder adaptar el modelo a esta modificación es necesario cambiar el tipo de distribución usada por los brazos, o bien alterar los parámetros para que las distribuciones beta sigan comportándose de la manera deseada. También es necesario considerar el uso de *rewards* no binarias en bandidos con diferentes políticas de elección.

Si llevamos el concepto a al caso específico de este proyecto, el *reward* de un brazo i en un tiempo[9] t podría definirse como una de las siguientes alternativas:

- $r(i,t) = \frac{v_i'(t) - v_i(t)}{v_i(t)} = \frac{\Delta v_i(t)}{v_i(t)} \in [-1, \infty)$
- $r(i,t) = \frac{v_i'(t)}{v_i(t)} \in [0, \infty)$

$$r : K \times T \to \mathbb{R}$$

Dados K como el conjunto de brazos y T como el de iteraciones. Donde $v_i(t)$ es el valor bursátil del brazo i en la iteración t al comienzo del período y $v_i'(t)$ es el mismo valor pero al final del período.

Ambas opciones representan el crecimiento de un valor bursátil en un período de tiempo determinado. Sin embargo la primera ofrece la ventaja de que para un crecimiento y una recesión opuestos, el valor absoluto de los mismos será el mismo. Por ejemplo, si la *reward* de una iteración es 0.2 y la del la siguiente es -0.2, usando la otra definición las *rewards* serían 1.2 y 0.8[10]. El uso de la primera alternativa podría ayudar a simplificar el modelo.

[9] En este caso el tiempo se refiere a una iteración la cual se corresponde con un período entre la acción y la observación, el cual es constante entre iteraciones.

[10] Como aclaración, en este ejemplo el valor acabaría siendo un 96% del valor original.

Una decisión especialmente relevante surge a partir de lo anteriormente expuesto. Para encontrar una función conjugada adecuada para el modelo es necesario determinar una función de verosimilitud cuyo comportamiento sea semejante a la función hipotética de *rewards* r.

Dado que el entorno exhibe un comportamiento complejo, es necesario simplificarlo. De manera que consideramos que un valor bursátil puede crecer o decrecer a valores que se encuentran cerca de su valor presente, sin cambios bruscos entre períodos de tiempo. Esto nos proporcionaría un comportamiento similar a una distribución normal o gaussiana cuya media sería cercana al cero en el intervalo positivo[11]. El detalle perdido por la simplificación puede ser reimplementado en el modelo con la ayuda de otras extensiones. Una distribución de verosimilitud más precisa sería una suma de tres o más distribuciones normales, esto daría representación a comportamientos volátiles que ocurren ocasionalmente en este tipo de productos financieros. Sin embargo, encontrar una conjugación compatible para esta distribución es mucho más complicado y escapa los objetivos actuales del experimento.

Es importante destacar que el valor de *reward* esta acotado en $[0, \infty)$ lo cual es diferente al dominio de la distribución normal $(-\infty, \infty)$. Podría usarse una distribución normal acotada pero, dada la complejidad que eso aportaría al modelo[12], se ignorará esta diferencia de dominio.

Ahora que se ha podido identificar una familia de distribuciones que encaja con a r, se puede dar con la función de distribución conjugada. Según el siguiente artículo [5], la función de distribución normal se corresponde con diferentes funciones conjugadas dependiendo de si se conoce alguno de los parámetros de la función de verosimilitud. En este caso no se sabe ni la media ni la desviación típica, por lo tanto la función conjugada sería la función de distribución gaussiana-inversa-gamma.

En este experimento se van a llevar a cabo entrenamientos de dos agentes cuyas distribuciones conjugadas son diferentes. En el primero se va a asumir que la desviación típica de la función de verosimilitud es conocida, de manera que la única incógnita del modelo va a será la media. La familia de distribuciones conjugadas en este caso es, al igual que en el caso de la verosimilitud, la gaussiana. En el segundo agente se considerarán desconocidos tanto la media como la desviación típica, de manera que la distribución conjugada es la gaussiana-inversa-gamma o, como se explicará más adelante, la gaussiana-gamma.

Metodología: distribución gaussiana conjugada

Dada una distribución gaussiana a priori $\mathbf{N}(\mu_{i,k}, \sigma_{i,k})$, los párametros de la distribución posterior $\mathbf{N}(\mu_{i+1,k}, \sigma'_{i+1,k})$ serían:

[11] Los valores bursátiles, concretamente los de acciones tienden a subir a muy largo plazo.
[12] Renormalizar la distribución y buscar una conjugada menos conocida y más compleja.

$$\mu_{i+1,k} = \frac{n_{i,k}\mu + r_{i,k}}{n_{i,k} + 1}$$

$$\sigma_{i+1,k} = \frac{1}{n_{i,k} + 1}$$

Donde i es el número de la secuencia de iteraciones del ejercicio y k es el brazo que es elegido en la iteración i. El desarrollo de esta conclusión es conocido y publicado [3].

El episodio comienza con una distribución a priori $\mathbf{N}(0, 1)$. A partir de las *rewards* obtenidas en las iteraciones, aplicadas utilizando las expresiones mostradas previamente, obtenemos un agente funcional. Cabe destacar que la primera distribución a priori puede ser aguzada a base de conocimiento previo del entorno. En este experimento se partirá del desconocimiento.

Metodología: agente con refuerzo completo

En este experimento también se va a entrenar a una variante de este agente. Debido a la baja proporción de iteraciones por número de brazos, es probable que agentes sujetos al dilema de explotación/exploración obtengan resultados deficientes. Para tratar de mejorar el rendimiento, se va a desacoplar la exploración de la explotación, de manera que el agente tendrá la capacidad de explorar todo el entorno mientras explota una sección del mismo en todas las iteraciones. Lógicamente esto implica que ciertas garantías teóricas dejen de aplicar a agentes con esta modificación. Sin embargo, en este proyecto se trata de observar los resultados empíricos de los agentes, de manera que los efectos teóricos serán desestimados.

A efectos de implementación, esta modificación consiste en ofrecer información de las *rewards* de todos los brazos en toda iteración al agente. Es importante destacar que esta fuente de información existe en escenarios reales, de manera que agentes que implementan esta funcionalidad se pueden beneficiar de ella en cualquier otra situación en cuyos brazos sean productos financieros. Al aplicar este refuerzo a todos los brazos, se puede conseguir un aprendizaje mas rápido que sin el. Por otro lado, este método tiene la potencial desventaja de sobreentrenar brazos en tendencias locales, de manera que su rendimiento se reduzca en cambios de tendencia.

A lo largo de este documento se le denominara a Bandidos con esta modificación Bandidos con refuerzo completo, para abreviar.

Metodología: distribución gaussiana-gamma

En esta casuística, los valores de la media y la varianza de la distribución de verosimilitud son desconocidos. Otra opción es usar una parametrización alternativa de la función de distribución normal, dichos parámetros son: μ como la mediana y τ como la precisión, de manera que $\tau = \sigma^{-2}$. Los efectos en el algoritmos son idénticos, pero a la hora de implementarlo

se prefirió la segunda parametrización. Lo cual cambia la distribución conjugada de **distribución normal-inversa-gamma** a la de **distribución normal-gamma**, cuya fórmula es la siguiente 3.15.

$$f(x, \tau \mid \mu, \lambda, \alpha, \beta) = \frac{\beta^{\alpha}\sqrt{\lambda}}{\Gamma(\alpha)\sqrt{2\pi}}\,\tau^{\alpha-\frac{1}{2}}\;e^{-\beta\tau}\;e^{-\frac{\lambda\tau(x-\mu)^2}{2}}$$

Figura 3.15: Función de distribución normal-gamma

La función de distribución normal-gamma requiere de los siguientes hiper-parámetros:

- μ' es la estimación de la mediana. Es decir, la mediana de la distribución ”prior”.
- λ' es el número de observaciones que han conformado la distribución “prior” en una iteración dada.
- α' es el número total de muestras.
- β' es una medida de la varianza.

Cuando el brazo es muestreado en consecuencia de haberlo elegido como acción, las siguientes funciones [3] determinan como se actualizan los hiper-parámetros:

$$\mu_0 \leftarrow \frac{\lambda\mu_0 + r_i}{\lambda + 1}$$
$$\lambda \leftarrow \lambda + 1$$
$$\alpha \leftarrow \alpha + \frac{1}{2}$$
$$\beta \leftarrow \beta + \frac{\lambda}{\lambda + 1}\frac{(r_i - \mu_0)^2}{2}$$

Donde r_i es la *reward* obtenida en la iteración i.

Esta arquitectura daría lugar a un modelo teóricamente compatible con en dominio de las *rewards*. Dado que Tensorflow no implementa esta distribución, es necesario buscar una manera eficiente de utilizarla. Afortunadamente un muestreo de la función de distribución de probabilidad se puede generar a partir de:

1. Un muestreo de una distribución gamma con parámetros α y β con el que se obtiene la precisión τ.
2. Un muestreo de una distribución gaussiana con parámetros μ y τ (obtenido en el paso anterior) con el que se consigue el muestreo equivalente.

Los agentes con esta distribución posterior también incluirán refuerzo completo de los brazos. Esto se debe a que, previo a ejecutar episodios con estos agentes, los resultados de los anteriores ya indicaron una considerable mejora de rendimiento usando esta técnica.

Resultados: Descripción y contexto

Los siguientes apartados van a mostrar los resultados de un total de 4 algoritmos diferentes en 50 episodios independientes. El conjunto de datos de entrenamiento es un subconjunto de 1000 valores bursátiles en un período de 4000 iteraciones. Cada iteración esta separada de la siguiente por 24h (precios de cierre). La extensión de tiempo del episodio es por tanto 10,95 años. A la hora de calcular el beneficio anualizado, se considerarán 11 años como la duración del período de inversión. Los valores bursátiles fueron escogidos aleatoriamente de entre los más de 7000 de los que dispone el set de datos íntegro. La sección de 4000 días fue extraída buscando la mayor cantidad posible de valores abiertos a lo largo del período.

Las métricas expuestas son las siguientes:

- **Ejecuciones en positivo**: este número entero indica cuántos de los episodios en los que se utilizó el algoritmo han acabado con un resultado en el que el inversor no perdería dinero de seguir el comportamiento del agente.
- **Ejecuciones que superan al rendimiento del índice**: este número entero indica cuántos de los episodios en los que se utilizó el algoritmo han acabado con un resultado en el que la rentabilidad de la inversión supera a invertir en todos los brazos una fracción idéntica. Es decir, que considerando a todos los brazos como un solo producto financiero, lo que se denomina de ahora en adelante como un índice, usar el agente trae mejores resultados.
- **Rendimiento del índice**: este porcentaje indica el crecimiento medio que experimentan todos los brazos desde el principio hasta el final del período del experimento. Lógicamente es el mismo en todos los resultados, ya que usan el mismo entorno. Este valor es 306.06%.
- **Rendimiento de la media de episodios**: este porcentaje indica el crecimiento medio de la inversión hipotética[13] de los 50 episodios en los que se ejecutó el algoritmo.
- **Beneficio anualizado de la mediana de episodios**: la mediana de los episodios es el episodio cuyo rendimiento es la mediana de los rendimientos de los episodios. El beneficio anualizado de la misma es el crecimiento anual medio que experimentó la inversión hipotética a lo largo del ejercicio, de invertir en la mediana de los episodios.

[13] Es decir, la inversión en la que se depositaría capital al inicio del experimento y se recuperaría al final.

- **Beneficio anualizado de la media de episodios**: la media de los episodios es la media de los rendimientos de los episodios. El beneficio anualizado de la misma es el crecimiento anual medio que experimentó la inversión hipotética a lo largo del ejercicio, de invertir en cantidades iguales en los 50 experimentos simultáneamente.

Resultados: Distribución gaussiana

Ejecuciones en positivo	66% (33/50)
Ejecuciones que superan al rendimiento del índice	26% (13/50)
Rendimiento del índice	306.08%
Rendimiento de la media de episodios	276.08%
Beneficio anualizado de la mediana de episodios	4.46%
Beneficio anualizado de la media de episodios	9.71%

Tabla 3.1: Resultados del bandido con distribución posterior gaussiana.

El crecimiento de las inversiones de los 50 episodios se puede observar en la figura 3.16.

Resultados: Distribución gaussiana con refuerzo completo

Ejecuciones en positivo	78% (39/50)
Ejecuciones que superan al rendimiento del índice	44% (22/50)
Rendimiento del índice	306.08%
Rendimiento de la media de episodios	360.02%
Beneficio anualizado de la mediana de episodios	8.73%
Beneficio anualizado de la media de episodios	12.4%

Tabla 3.2: Resultados del bandido con distribución posterior gaussiana con refuerzo completo.

El crecimiento de las inversiones de los 50 episodios se puede observar en la figura 3.17.

Resultados: Distribución gaussiana-gamma con refuerzo completo

Ejecuciones en positivo	44% (22/50)
Ejecuciones que superan al rendimiento del índice	40% (20/50)
Rendimiento del índice	306.08%
Rendimiento de la media de episodios	408.57%
Beneficio anualizado de la mediana de episodios	8.677%
Beneficio anualizado de la media de episodios	28,052%

Tabla 3.3: Resultados del bandido con distribución posterior gaussiana-gamma con refuerzo completo.

El crecimiento de las inversiones de los 50 episodios se puede observar en la figura 3.18.

Comparación de resultados

Agentes	CONTROL	G	FRG	FRGG
En positivo	65%	66%	78%	44%
Superan al índice	20%	26%	44%	40%
Rendimiento de la media	303.91%	276.08%	360.02%	408.57%
B.A. de la mediana	4.51%	4.46%	8.73%	8.677%
B.A. de la media	10.68%	9.71%	12.4%	28,052%

Tabla 3.4: Tabla de resultados del experimento D.

Donde:

- CONTROL: resultados de un agente cuya política es escoger un brazo al azar en cada iteración.
- G: distribución conjugada gaussiana.

- FRG: distribución conjugada gaussiana con "refuerzo completo".
- FRGG: distribución conjugada gaussiana-gamma con "refuerzo completo".

Los resultados obtenidos en este experimento son un tanto inesperados, probablemente debido a la ininteligibilidad del entorno y del agente. El algoritmo de control que consiste en ejecutar elecciones aleatorias tiene un rendimiento bastante bueno, especialmente para ser aleatorio. Esto es, de todas maneras, perfectamente lógico debido a que en 11 años que recorre el experimento prácticamente todos los agentes experimentan un crecimiento más, o menos significativo.

Una de las conclusiones más relevantes que se puede extraer de los resultados es que utilizar un agente cuyas distribuciones posteriores son gaussianas, es decir, con la asunción da una varianza conocida, da lugar a resultados deficientes. En concreto, el agente gaussiano ha obtenido rentabilidades peores que el agente aleatorio, con una muy leve subida en la estabilidad del mismo. Estos resultados parecen indicar que realmente usar este agente, en este entorno, supone un comportamiento similar sino equivalente a invertir al azar. Al ejecutar más instancias del este experimento se puede comprobar que en efecto esta consideración se confirma. La razón para este rendimiento posiblemente radique en la poca oportunidad que da este entorno para explorar los brazos, sin embargo esto se hace intencionalmente, ya que los entornos reales padecerán de inconvenientes similares.

En cuanto a los agentes que usan refuerzo en todos los brazos durante sus iteraciones, los resultados son buenos, especialmente teniendo en cuenta que son Bandidos de elección única, lo cual fomenta significativamente la inestabilidad. Ambos agentes han conseguido superar a la inversión simultánea en todos los brazos. El agente gaussiano con refuerzo completo ha consegido superar a la media con una diferencia de un 53.94%. El agente gaussiano-gamma sobrepasa al anterior agente con una diferencia de un 48.55%, de manera que supera la media por un 102.49%.

Cabe destacar la diferencia entre los agentes con refuerzo completo, aunque el gaussiano ofrece menor rentabilidad, este es mucho más estable que el gaussiano-gamma. De hecho, menos de la mitad de inversiones usando la distribución posterior gaussiana-gamma acaban en positivo. Por otro lado, en los episodios en los que este agente es rentable, lo es por mucho más que cualquier otro agente del experimento. De hecho, el beneficio anualizado que ofrece de media este agente es de un 28.052%, más dos de veces mayor que el agente gaussiano.

Es importante tener en cuenta que el set de datos utilizado y la duración del experimento fomentan significativas magnitudes de rentabilidad, por tanto no hay garantía de que los agentes tengan un comportamiento similar en condiciones diferentes. En experimentos con menos iteraciones, es probable que el rendimiento de los agentes se vea perjudicado. En cuanto a los brazos utilizados, estos parten de acciones de empresas estadounidenses, donde la tendencia a largo plazo es siempre positiva.

El experimento da lugar a las siguientes conclusiones:

- Como es lógico los agentes implementados son muy sensibles a los complejos patrones del entorno al que se exponen, dando lugar a un gran porcentaje de bandidos que no consiguen siquiera mantener la inversión inicial. Para poder solventar esta inestabilidad será necesario integrar extensiones mas complejas al algoritmo, como es el caso del contexto. Planteamientos acerca de esta extensión se expondrán en la sección 3.5.

- Ofrecer información acerca de las *rewards* de todos los brazos en todas las iteraciones parece tener buenos resultados empíricos. Por otro lado, también es posible que la estabilidad de los agentes se vea mermada por ello. La explotación intensiva de los brazos acaba reduciendo la varianza de sus distribuciones, provocando una dilatada susceptibilidad a pérdidas por cambios repentinos en el estado del mercado. Para paliar este efecto colateral se puede considerar utilizar alguna técnica para hacer al agente olvidar los datos del pasado. Este tema se expone con mayor deliberación en la sección 3.5.

- Una manera simple de obtener un sistema funcional estable es ejecutar una ristra de agentes simultáneamente, de manera que la inversión se reparte entre ellos en partes iguales. Los agentes seleccionados pueden ser gaussianos, gaussianos-gamma o una mezcla de ambos. Esta selección dependería de la magnitud de riesgo que el usuario está dispuesto a asumir. En principio, los resultados del experimento indican que la inversión en este conglomerado de agentes tendría un beneficio anualizado superior al de simplemente invertir en todo.

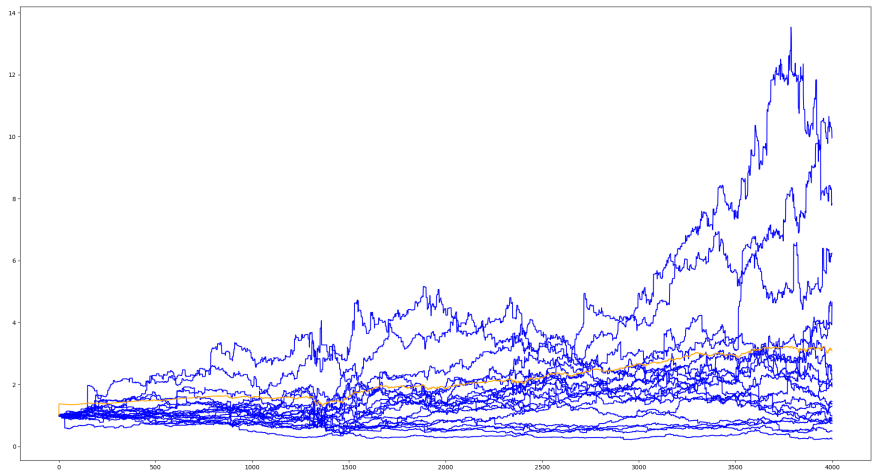

Figura 3.16: Crecimiento de las diferentes inversiones de los 50 episodios (azul) vs crecimiento de la inversión simultánea en todos los brazos (naranja). Agente gaussiano.

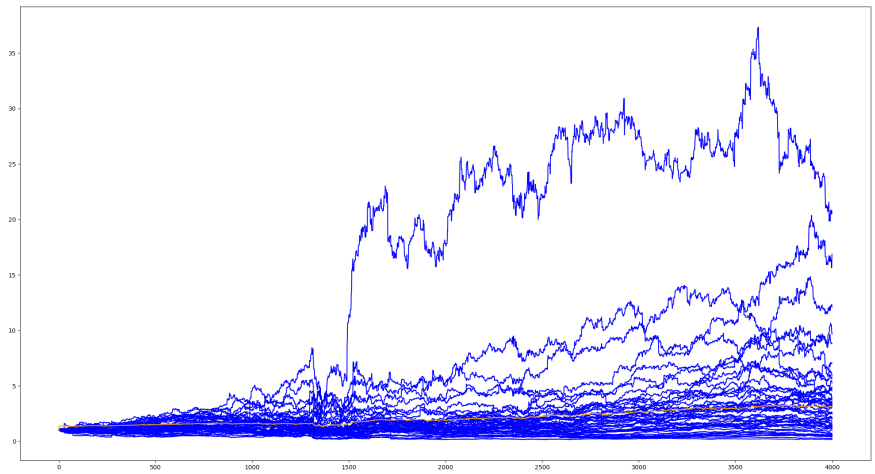

Figura 3.17: Crecimiento de las diferentes inversiones de los 50 episodios (azul) vs crecimiento de la inversión simultánea en todos los brazos (naranja). Agente gaussiano con refuerzo completo.

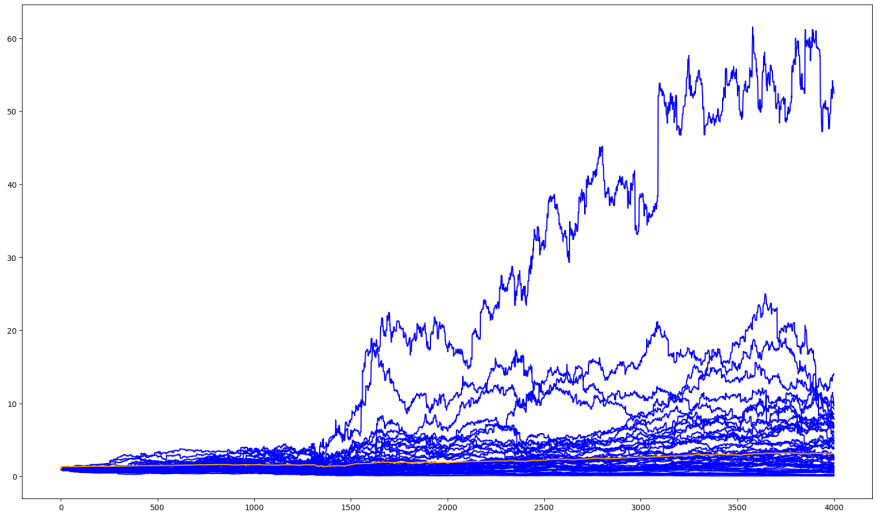

Figura 3.18: Crecimiento de las diferentes inversiones de los 50 episodios (azul) vs crecimiento de la inversión simultánea en todos los brazos (naranja). Agente gaussiano-gamma con refuerzo completo

3.5 Planteamientos acerca de la extensión del modelo

Previo al último experimento se presentan una serie de consideraciones acerca de la mejoras y extensiones a modelos de Bandidos que pueden ofrecer un aumento en el rendimiento de los mismos. Muchos de los métodos aquí planteados no serán aplicados en ningún experimento, sino que son proposiciones a aplicar en trabajos futuros. Esto se debe a que implementarlos escapa de los objetivos de este trabajo.

3.5.1 Bandidos con contexto:

Hasta ahora, toda la información de la que dependía el agente, a la hora de estimar la rentabilidad de los brazos, era el *reward* proporcionado por la observación. Esto da lugar a un modelo muy limitado, que no se puede aprovechar de otros datos que pueden ser muy relevantes a la hora de determinar que brazo elegir en la siguiente iteración, como puede ser el volumen de mercado. Otro problema es el hecho de que normalmente los diferentes valores bursátiles no son variables independientes, es más, suele haber grupos que tienden a un mismo comportamiento. Estas relaciones de influencia grupal no se tienen en cuenta en experimentos anteriores. En general, podemos agrupar las idiosincrasias de este entorno en los siguientes grupos:

- Dependencia entre resultados de diferentes brazos.
- Cambios bruscos e inesperados[14] en la distribución de resultados de uno o más brazos.
- Tendencias embebidas en tendencias[15].

Lo que todo esto sugiere es que hay una serie de factores subyacentes los cuales influyen en el comportamiento de los brazos. Sin embargo, no es posible determinar con precisión que factores afectan a que brazos, en que proporción o si la proporción es directa o inversa. Esta consideración se puede ver ejemplarizada en la figura 3.19.

Integrar información acerca de los eventos que influyen en los productos financieros utilizados como brazos escapa del dominio de este trabajo. Esto se debe a las astronómicas cantidades de tiempo que implicaría investigar como adaptar esa información a algoritmos como los que se muestran en este documento. Sin embargo, es posible aprovechar la existencia de estos factores ocultos para relacionar brazos que resultan tener comportamientos similares o similarmente opuestos con asiduidad. Es decir, agrupar brazos en grupos que reaccionan de manera similar ante factores desconocidos. Una vez generados estos grupos se puede usar el

[14] No se dispone de la totalidad del contexto que lleva a ese cambio.

[15] Un valor bursátil puede estar en alza en un período mensual y en descenso en período cuatrimestral y en alza en período anual.

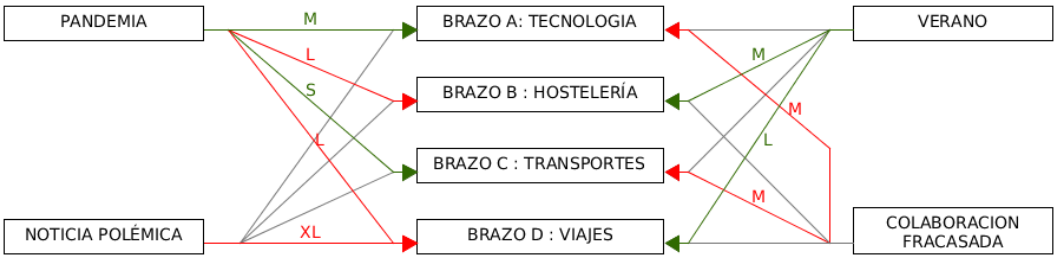

Figura 3.19: Ejemplo hipotético de como factores fuera del alcance del agente tienen efectos comunes en los brazos. Los colores representan si el efecto es positivo o negativo. Las tallas representan la magnitud.

rendimiento de unos brazos como contexto para sesgar la perspectiva de rentabilidad de otros brazos.

Una técnica común y eficaz para aplicar contexto a la elección de un brazo es el uso de vectores de características asociados iteraciones y brazos, que almacenan datos acerca de los mismos. Los modelos con contexto se denominan como "disjuntos" si a cada brazo se le puede asignar un vector de característica diferente en una misma iteración. La idea sería desarrollar una correspondencia entre los valores del vector y la potencial rentabilidad de los brazos, de manera que nos pueda ayudar a tomar una decisión más informada. Por ejemplo: si desarrollamos un bandido multi-brazo para mostrar publicidad con mejor tasa de clicks, el rendimiento mejoraría al generar una correspondencia entre los datos de la persona que va a ver el anuncio con los anuncios, de manera que se elige el más atractivo para esa persona.

Contexto lineal para Thompson Sampling:

Actualmente se han publicado variedad de adaptaciones de diferentes políticas para aceptar contexto [6]. Uno de los algoritmos de Bandido con contexto más utilizados por la comunidad es el algoritmo de Thompson Sampling para Bandidos con contexto con retornos lineales [7]. Este algoritmo se basa en que, dado un vector de características que representa el contexto de un brazo, hay otro vector, denominado predictor, tal que el producto escalar del uno por el otro es la *reward* que se estima que retornara el brazo. A partir del error a la hora de hacer esta predicción el vector predictor se va reforzando para imitar el comportamiento del brazo que corresponde.

Este algoritmo requiere de una importante precondición para ofrecer garantías teóricas de su rendimiento. Esta precondición es la linealidad de las *rewards* respecto al vector de características. Dicho de otra manera, la recompensa es, o tiende a ser una combinación lineal de los elementos del contexto. En realidad, el entorno bursátil no tiene porqué cumplir con esta precondición, de hecho, no lo hace. Sin embargo, esta condición será relajada en el siguiente

experimento en el que se va a implementar este algoritmo.

Para calcular la estimación de reward y reforzar el modelo, el algoritmo consiste en los elementos y pasos que se muestran a continuación:

- Elementos:

$$B_i \in \mathcal{M}_{n\times n}(\mathbb{R}) \qquad \forall i \in \mathbb{N} \cup \{0\}$$

$$\vec{v}_i \in \mathbb{R}^n \qquad \vec{w}_i \in \mathbb{R}^n$$

$$\hat{r}_i \in \mathbb{R}^n \qquad r_j \in \mathbb{R} \qquad \forall j \in \mathbb{N}$$

$$\vec{x}_i \in \mathbb{R}^n$$

- Inicialización:

$$B_o \coloneqq I_n \qquad \vec{v}_0 \coloneqq \vec{\mathbf{0}}_n \qquad \vec{w}_0 \coloneqq \vec{\mathbf{0}}_n$$

- Iteración j:

$$B_j \coloneqq B_{j-1} + \vec{x}_j^T \cdot \vec{x}_j$$

$$\vec{v}_j \coloneqq \vec{v}_{j-1} + \vec{x}_{j-1} \cdot r_j$$

$$\vec{w}_j \coloneqq B_j^{-1} \cdot \vec{v}_j$$

$$\hat{r}_j \coloneqq \vec{w}_j^T \cdot \vec{x}_j$$

Para simplificar las anteriores expresiones se han omitido los detalles enumerados a continuación:

1. Hay una réplica de todas de las variables expuestas por cada brazo que posee el Bandido.
2. Cada brazo utiliza una j diferente (j_k) que representa la cantidad de iteraciones en el que el brazo es elegido.
3. En una iteración de un episodio solo se computan las operaciones en el brazo que fue escogido en esa iteración. De manera que su valor de j se incrementa en 1.
4. En el caso de los brazos no elegidos, sus variables de la siguiente iteración son iguales a las de la anterior.

La distribución de verosimilitud es gaussiana, lo mismo aplica a la distribución posterior. Para obtener las distribuciones posteriores en una iteración determinada se lleva acabo la siguiente computación:

$$\mu_{i,k} \coloneqq \vec{x}_{i,k}^T \cdot \vec{w}_{i,k}$$

$$\sigma_{i,k} \coloneqq \nu^2 \cdot \vec{x}_{i,k}^T \cdot B_{i,k}^{-}1 \cdot \vec{x}_{i,k}$$

$$\mathbf{N}(\mu_{i,k}, \sigma_{i,k})$$

Un método alternativo para obtener el vector de características sin implementación:

Hay muchas maneras de implementar esta extensión, empezando por la dimensión del vector de contexto y su contenido hasta el método que condensa el contenido y contribuye a la elección de la siguiente acción.

A continuación se va a mostrar una posible configuración del vector de características para los brazos de una agente con contexto lineal. En el experimento E se utiliza otra configuración, esto se debe al gran consumo de memoria que esta configuración supone. Para acelerar el proceso del agente se optó por un vector de características más ligero y menos descriptivo.

Del conjunto de información que se podría extraer del conjunto de datos y del historial del entrenamiento, podrían considerarse los siguientes parámetros para incorporar al vector de contexto:

Sea $\vec{v}_{i,t} = [v_{1,i,t}, ..., v_{|K|,i,t}] \in \mathbb{R}^{|K|}$ un vector de características bruto[16] para un brazo $i \in K$ en un tiempo $t \in T$, el vector consta de los siguientes elementos:

- Aumento de volumen de mercado entre las dos últimas iteraciones. Acotado en $[0, \infty)$ (1 elemento al principio del vector).
- La covarianza actual del brazo i con todos los demás brazos ($|K| - 1$ elementos).

Aplicando una combinación lineal de los diferentes elementos podemos obtener un vector con el cual sesgar el resultado del muestreo de las distribuciones posteriores. Como en los anteriores modelos, el brazo cuyo muestreo es máximo será el aplicado como acción en la iteración presente.

$$\vec{x}_{i,t} = \sum_{j}^{|K|} v_{j,i,t} * \beta_j$$

Donde $x_{i,t}$ es el vector de características procesado del brazo i en la iteración t que será integrado en la política del agente y β_j el coeficiente correspondiente al elemento j del vector $\vec{v}_{i,t}$. Los coeficientes para cada elemento se definen como:

- β_1 es un valor a especificar que mide la relevancia de este dato.
- $\beta_a = d_a \cdot r'_a, a \in \{2, ..., |K|\}$

Donde r'_a es el último *reward* obtenido de cada uno de los diferentes brazos, acotado en $[0, \infty)$ y d_a es un valor de decadencia temporal que se explicará a continuación.

[16] Los valores serán alterados antes de entregar el vector al agente.

Calculo de covarianzas: A la hora de obtener las covarianzas de cada brazo con los demás se pueden obtener de dos maneras:

- A partir de la totalidad de los datos del conjunto de entrenamiento. La ventaja de esta opción es que la covarianza será precisa desde el principio y el rendimiento sera considerablemente superior en valores que mantienen la misma correlación durante todo el episodio. Al aplicar nuevos conjuntos de datos habría que modificar las covarianzas de manera similar a la segunda opción.

- A partir del historial de observaciones que el bandido tiene en un momento dado. La ventaja de esta opción es que la covarianza de valores cuya correlación verdadera varía con el tiempo será mas precisa. Otra ventaja es que la manera de computar la covarianza durante el entrenamiento y el aprendizaje posterior es exactamente la misma. El mayor inconveniente es que al principio del entrenamiento estos valores serán muy imprecisos.

El segundo método parece una mejor opción ya que nos interesa que el agente funcione mejor[17] una vez entrenado, y no nos importa que el rendimiento sea bajo al comienzo del entrenamiento. Una manera de calcular la covarianza [8] de dos brazos $i, j \in K$ en un tiempo t sería la siguiente:

$$cov_t(i,j) = \sum_{x=0}^{t-1} (r(i,x) - \mu_{i,t}) \cdot (r(j,x) - \mu_{j,t})$$

donde t es la iteración presente y $\mu_{x,t}$ la mediana de la función de probabilidad del brazo $x \in K$. En otras palabras, definimos la covarianza como la suma de la multiplicación de todos los rewards pasados de los brazos i y j a los cuales les restamos su mediana.

La **complejidad computacional** teórica del cálculo de las covarianzas sera de $O(t^2k)$ en el peor de los casos. Sin embargo, dado que la covarianza en t para cualquier par de brazos se puede calcular a partir de $t-1$ la complejidad se reduce a $O(tk)$. Y dado que en una iteración solo un brazo es actualizado[18] la complejidad se reduce aún mas a $O(t)$ o $O(tl)$ donde $l <= |K|$ es el número de brazos elegidos por iteración.

Calculo del factor de decadencia: determinar la magnitud de este parámetro en función de la antigüedad de los rewards es vital para el correcto funcionamiento del contexto. Antes de determinar valores específicos, conviene analizar como debería de cambiar según el estado subyacente del entorno.

El mercado bursátil pasa por épocas estables e inestables en las cuales el conocimiento previo del comportamiento de un brazo es más o menos relevante respectivamente. En el

[17] El agente será más adaptable a cambios
[18] De no usarse refuerzo completo.

caso de que el agente pueda saber lo estable que es el mercado, el factor de decadencia debería adaptarse al mismo.

La magnitud de los períodos de tiempo que transcurren entre iteraciones del agente también es relevante a la hora de determinar el valor del factor de decadencia.

La fórmula de este parámetro podría ser la siguiente:

$$d_a = \frac{x}{\Delta_t^y}$$

Donde x, y son una coeficientes de correción y Δ_t la diferencia temporal entre el presente y la iteración del *reward* medido en iteraciones.

Limitaciones del modelo descrito y mejoras necesarias: El modelo de contexto que se ha expuesto hasta ahora tiene una serie de limitaciones prácticas que lo hacen inviable para producir agentes útiles. Los problemas más severos son:

1. **El tamaño de los vectores de características.** Debido a que se pueden manejar grandes cantidades de brazos en un ejercicio, que el contexto crezca proporcionalmente a estos da lugar a operar con matrices gigantescas, del orden de giga-bytes. Las CPUs tardarán bastante en procesar las operaciones, y las GPUs pueden no tener memoria suficiente para alojarlas.

2. **Ineficacia a la hora de lidiar con muchos brazos cerrados.** Aunque solo unos pocos brazos estén disponibles , el contexto va a ejecutar la misma cantidad de operaciones, produciendo ineficientes matrices dispersas.

3. **Imposibilidad de extender el numero de brazos durante un ejercicio.** El hecho de que las dimensiones del contexto se correspondan con el número de brazos dificulta enormemente adaptar el modelo para incorporar brazos nuevos o eliminar brazos que ya no son útiles.

Para poder sortear dichas limitaciones se ha concebido un formato de vectores de característica con menos atributos, cuyo tamaño puede ser independiente del número de brazos. Esto permite ser más flexible en cuanto al número de brazos activos durante un ejercicio, a costa de cierta precisión perdida al reducir los datos a campos de menor tamaño.

El formato en cuestión se utiliza y expone en el experimento E (véase 3.6).

3.5.2 Olvido intencional del aprendizaje:

Si nos detenemos a razonar un momento acerca de los datos que constituyen las distribuciones de cada brazo, las cuales representan una aproximación del comportamiento del brazo;

no tiene mucho sentido tener en cuenta datos del brazo con años de antigüedad respecto a la iteración presente, o al menos que tengan el mismo peso que los más recientes. En un entorno inestable resulta lógico que un brazo que olvide parte del pasado funcione mejor que un bandido que considera cada resultado de su historia con la misma importancia. Dicho esto, también hay que ser cuidadoso con la magnitud de decadencia en la importancia de los resultados, ya que excederse resultará en un modelo que apenas puede ser entrenado y sus distribuciones serán altamente imprecisas. Una vez entendido por que es una buena idea aplicar un factor de decadencia o descuento a resultados más antiguos también se puede extender la funcionalidad del factor adaptándolo a las características del entorno. De manera que para períodos relativamente estables los brazos considerarán más información acerca del entorno, mientras que en períodos convulsos lo contrario ocurrirá.

En el modelo para la construcción de un vector de características que se explica en el apartado anterior contiene un modificador denominado factor de decadencia. Su propósito es evitar un sobre-entrenamiento de un brazo mientras esta atravesando una tendencia. Esto es una técnica para olvidar deliberadamente datos aprendidos por el agente en el pasado, ya que estos son perjudiciales para el rendimiento del agente.

Generalmente a la hora de hacer al agente olvidar lo que aprendido es necesario discriminar que datos han de ser eliminados. Ya que es común que haya datos que siguen siendo relevantes y necesarios para un buen funcionamiento junto a datos perjudiciales.

Una idea que surge naturalmente al indagar este concepto es ir menguando el impacto de eventos más antiguos a favor de los más recientes. En el caso del entorno que se maneja, esto protegería al agente de cambios de tendencia en períodos determinados.

Esto saca a flote una nueva incógnita, cual ha de ser el valor del hiperparámetro que descuenta a estos valores. En el apartado anterior se ve un ejemplo de para determinar este valor. Este ejemplo depende de dos valores, x e y. Estos han de ser determinados por el usuario en función del entorno al que se exponga el agente.

Una planteamiento interesante es determinar los hiperparámetros de manera dinámica, de manera que a partir de la estabilidad del mercado se obtendrán unos valores óptimos para el mismo. Una idea de aplicación de este concepto sería elevar el valor de los hiperparámetros a partir del histórico de *rewards* de los brazos, en función de la varianza de estos.

3.5.3 Diferentes adaptaciones a entornos de difícil aprendizaje:

Hay una inmensidad de diferentes extensiones la cuales no encajan en los anteriores apartados. En este apartado se presentaran mejoras que facilitan la adaptación del modelo a un entorno inestable. Estas ideas serán descritas de manera breve y puede que más adelante, en trabajos futuros sean puestas en práctica y, en consecuencia, documentadas con mayor detalle.

Moderación del comportamiento usando aprendizaje no supervisado: en la anterior idea se expuso una manera de adaptarse según la inestabilidad del mercado, de manera que el agente olvida o menosprecia información antigua. Sin embargo es común y lógico que cuando un entorno retorne a la estabilidad, parte del comportamiento antiguo vuelva a ser relevante.

Clasificación de brazos a través de aprendizaje no supervisado: podría usarse una red neuronal o alguna tecnología similar para agrupar brazos en diferentes *clusters* que indiquen algún tipo de relación entre ellos que deba ser considerada como contexto.

Agente auxiliar de reconocimiento de patrones (Analisis técnico): en un valor bursátil pueden darse diferentes patrones que incitan a los inversores a actuar de una manera determinada. Detectar estos patrones y aplicarlos al contexto podría aumentar el rendimiento del agente. Podría usarse una red neuronal o tecnologías similares para obtener valores de contexto relevantes.

Uso de medias móviles y otros instrumentos como parte del contexto: hay instrumentos financieros ampliamente utilizados para sintetizar información acerca del comportamiento de un valor bursátil. Aplicar estos instrumentos como parte del contexto podría incrementar el rendimiento del agente. Un ejemplo de instrumento son las medias móviles, una media móvil es básicamente la media del valor de los n últimos períodos de cotización.

3.6 Experimento adicional (E): MAB con contexto lineal y distribuciones Gaussianas

Motivación

En los anteriores experimentos se llevó a cabo la evaluación de modelos de Bandidos Bayesianos sin contexto. Los resultados, aunque en general satisfactorios, muestran las inconveniencias de estos modelos al lidiar con el entorno bursátil, principalmente su inestabilidad. Esto, aunque inevitable debido al riesgo que supone invertir todo a un brazo por iteración, puede ser mejorado.

Para poder obtener modelos más estables es posible integrar el uso de observaciones acerca del presente para así tener una predicción más informada del futuro.

El principal objetivo de este experimento es comprobar los resultados de aplicar un Bandido Bayesiano con Thompson Sampling y contexto lineal al entorno con el que se trabaja en este proyecto. Dependiendo de los resultados del experimento se podrían aplicar extensiones y correcciones al modelo en trabajos futuros.

Condiciones del experimento

En este experimento se parte de exactamente del mismo entorno que en el anterior. Un total de 1000 brazos que representan valores bursátiles durante 4000 iteraciones con un día de separación.

Para aplicar el modelo de agente se van a asumir que existen unas relaciones **lineales** desconocidas entre los valores del vector de características de una iteración y las *rewards* obtenidas en la siguiente iteración.

Metodología

Se va a implementar un Bandido Bayesiano con Thompson Sampling y contexto lineal no disjunto. El funcionamiento del agente se expone a continuación:

Sea:

1. $\vec{x}_i(t) \in \mathbb{R}^m$ un vector de características de un brazo i en un tiempo t.
2. $i \in [0, k)$ siendo k el número de brazos.
3. $m \in (0, k)$ un parámetro a determinar generalmente del orden de 1 o 10.
4. $t \in [t, T]$ siendo T las iteraciones totales del episodio.

Para obtener $\vec{x}_i$ se lleva a cabo una agrupación de brazos utilizando el algoritmo de *clustering* k-means, donde la m es el numero de clústeres. Los brazos son agrupados utilizando su historial, es decir:

$$\vec{h}_{i,t} = (r_{i,0}, r_{i,1}, \ldots, r_{i,t})$$

Siendo $\vec{r}_{i,t}$ la *reward* del brazo i en la iteración t.

Una vez obtenidos los grupos, se calcula la media de *rewards* de la última iteración en cada uno de los grupos. El vector con estas medias será el vector de características que se proveerá como observación a la política de selección del agente.

Una vez recibida la observación se procederá con el algoritmo expuesto en la sección 3.5.1.

Resultados

Los resultados percibidos por este agente en las mismas condiciones que los del experimento anterior, han sido significativamente deficientes en comparación. La tabla de resultados se muestra a continuación.

Ejecuciones en positivo	6.67% (1/50)
Ejecuciones que superan al rendimiento del índice	6.67% (1/50)
Rendimiento del índice	306.08%
Rendimiento de la media de episodios	50.04%
Beneficio anualizado de la mediana de episodios	Negativo
Beneficio anualizado de la media de episodios	Negativo

Tabla 3.5: Resultados del bandido con contexto lineal.

La media del crecimiento de las inversiones acaba con un 50% del capital, es decir, se pierde la mitad de la inversión. La razón detrás de estos resultados puede ser:

- La relajación de la precondición de contexto lineal puede haber derivado en un agente que no es capaz de gestionar las observaciones que se le proporcionan de manera útil.
- El generador del vector de características no es lo suficientemente elástico para este entorno, de manera que los cambios bruscos en el entorno no se dan manejado con éxito.
- Puede que el número de grupos utilizado sea insuficiente o incluso excesivo. Sería necesario ejecutar más experimentos para comprobarlo.

La conclusión que se extrae de este experimento adicional, de cara a trabajo futuro, es que el algoritmo probado no es válido en este entorno, y por lo tanto será necesario experimentar con otros hiperparámetros y recurrir a técnicas más elaboradas.

Capítulo 4

Aplicación demostrativa

En este capítulo se disertará acerca del diseño de una aplicación demostrativa con la cual poder experimentar con los agentes que fueron implementados en los experimentos D y E. Este producto software consiste en una aplicación web con la que el usuario puede interactuar con una serie de agentes.

4.1 Concepto inicial:

Como ya se ha explicado anteriormente, el producto software resultante de este proyecto va a ser una aplicación web. A continuación se definirán los cimientos de lo que será el diseño de esta aplicación.

Que se busca alcanzar con este diseño:

Los objetivos a los que se apunta a la hora de perfilar la estructura del programa son:

- **Rapidez:** dada la gran cantidad de operaciones que un agente tiene que efectuar, nos interesa un producto optimizado y preparado para lidiar con estas operaciones con un alto grado de rendimiento; tanto durante el preprocesado como durante el cálculo de acciones.
- **Estabilidad:** de poco sirve la aplicación si exhibe comportamientos irregulares, o los agentes tienden a fallar ante modificaciones superficiales.
- **Extensibilidad:** es muy posible que al terminar el proyecto el producto no ofrezca unas tasas de rentabilidad atractivas al usuario. Sin embargo, aunque esto ocurra, se busca que el producto sea una plataforma sobre la cual mejorar lo desarrollado para eventualmente desarrollar agentes con mejor rendimiento.

- **Reusabilidad:** aparte de la extensión del producto para el caso de uso bursátil, sería interesante abstraer y desacoplar ciertos elementos del diseño para hacer posible la producción de agentes para otras finalidades. Esto podría ahorrar tiempo a otros investigadores con ideas similares a las expuestas en el proyecto.

Es importante destacar que ante la situación de tener que optar por eficiencia computacional o de memoria, se **priorizará la eficiencia computacional sobre la de memoria**.

Formato de la aplicación:

La idea es que este software sea una aplicación web, razones para escoger este formato son:

- **Facilidad de acceso para el usuario** al no requerir instalación de ningún cliente (salvo el navegador web en si mismo). Esto permite acceder a las funcionalidades del producto desde cualquier lugar siempre y cuando se pueda en-rutar el cliente con el servidor del producto.
- **Abundancia de librerías y documentación.** El ecosistema web posee una gran cantidad de utilidades ya desarrolladas y documentadas que pueden ser fácilmente integradas en el producto.
- **Descentralización de la computación.** El agente que recomienda valores bursátiles está siendo ejecutado en un servidor el cual puede ser optimizado para su ejecución. El usuario solamente necesita un dispositivo con capacidades reducidas para interactuar con el agente.
- **Posibilidad de acceso concurrente y centralización de datos y agentes.** Esta ventaja probablemente no sea explotada por este proyecto ya que requiere una inversión de tiempo mayor. Sin embargo, es interesante plantearse que como trabajo futuro se puede extender el producto para que sirva a varios usuarios al mismo tiempo y que estos puedan compartir agentes y datos.

Una buena opción para implementar este software sería el uso del *framework* de **Django** el cual se basa en Python como lenguaje de programación. La ventaja principal de este *framework* sobre los otros es su enfoque en incluir de base una gran cantidad de herramientas que facilitaran la implementación de la aplicación y dejarán más tiempo para desarrollar y extender los agentes en sí. Y es que el tiempo de desarrollo es un factor de gran importancia en este proyecto. Otras alternativas en Python son **Flask** y **Pyramid**, aunque son mas que aceptables para desarrollar este producto, **Django** parece ofrecer una mayor velocidad de desarrollo.

Arquitectura básica y dependencias:

Se espera que la aplicación resultante tome una forma semejante a la que se muestra en el diagrama de la figura 4.1.

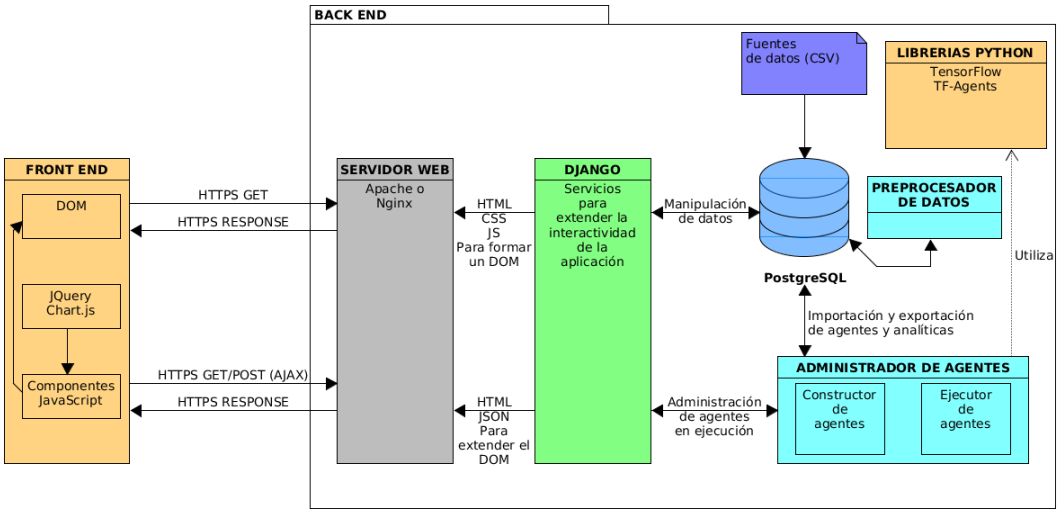

Figura 4.1: Arquitectura básica

Descripción del diagrama:

- **Administrador de agentes (AgMg):** es el núcleo del producto, la parte que lleva a cabo los entrenamientos de los agentes y, por ende, la parte mas importante tanto para el producto como para el proyecto. Se prevé dividir este módulo en dos componentes con distintas finalidades. Primero, un "constructor" que se dedica a generar las estructuras que constituyen un agente basándose en instrucciones provistas por el usuario. Segundo, un "ejecutor" o *driver* que lleva a cabo el entrenamiento del agente. Los agentes y los datos resultado de su entrenamiento pueden ser guardados en la base de datos de la aplicación.

- **Bases de datos:** se barajan distintos posibles DBMS[1] para suplir las necesidades de almacenamiento de la aplicación. Dada la naturaleza de los datos a almacenar, un modelo relacional es de lo más adecuado. **PostgreSQL** es una elección idónea debido a su eficiencia, estabilidad, código abierto y estricto modelo relacional. La función de la base de datos en esta aplicación es crucial ya que no solo almacena los datos que se van a procesar sino que almacena también los datos procesados y las propias estructuras que procesan y se entrenan con estos datos. En resumen, se busca que la base de datos le aporte la totalidad de la persistencia a las funcionalidades de la aplicación.

[1] DataBase Management System

- **Django:** este *web framework* se usa para interconectar el cliente, con la base de datos y con el administrador de agentes. Django ofrece una API que proporciona un acceso simple y seguro a la base de datos. Aparte de lo anterior, este *framework* también permite aportar otras funcionalidades como la autenticación de usuarios a la aplicación con una inversión de tiempo mínima. Django también resulta de gran utilidad durante el desarrollo debido a su metodología de diseño minimalista y el sistema de modelos que permite efectuar migraciones con facilidad.

- **Servidor web:** el software específico para efectuar las tareas de un servidor web será determinado más adelante. Su relevancia en el desarrollo es menor pues un servidor orientado a producción no es estrictamente necesario para el funcionamiento de la aplicación. Es posible que el propio usuario sea el que decida en que servidor implantar la aplicación.

- **Front end:** las bibliotecas utilizadas para implementar la interfaz web van a ser:

  - **JQuery**: esta famosa biblioteca dispensa un gran abanico de utilidades genéricas para el desarrollo web. El sistema de selectores con el cual se puede acceder a elementos del DOM es de especial utilidad, junto con el método para ejecutar solicitudes AJAX.
  - **JQuery UI**: en este módulo se van a fundamentar muchos de los elementos de la interfaz, como pueden ser ventanas, diálogos, acordeones o botones. También incluye una buena variedad de efectos de animación.
  - **Chart.js**: se utilizará a la hora de representar grandes cantidades de datos, acerca de los brazos o los resultados, a través de gráficas.

Estructuras de datos:

Como se ha expuesto anteriormente el estado (persistencia) de la aplicación se basa únicamente en las estructuras de datos almacenadas en el DBMS. Dado que se dará frecuentemente el caso de experimentos que comparten la misma información, o una parte de ella, como conjunto de datos de entrenamiento; es lógico desarrollar una estructura que facilite la reutilización de los datos preprocesados en diferentes contextos. Para ello se ha concebido la siguiente organización de los datos:

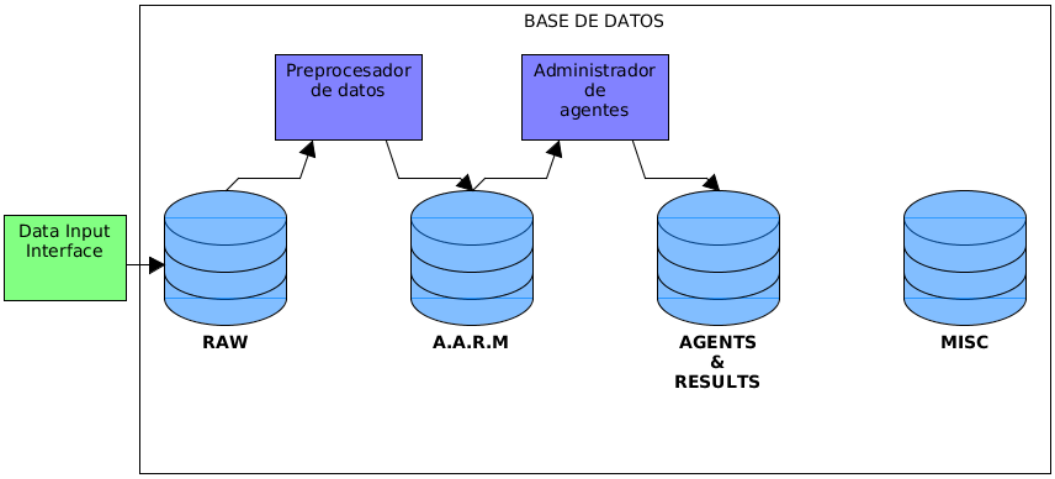

Figura 4.2: Almacenamiento de datos, resultados y agentes

Donde:

- **RAW:** es una agrupación de datos que corresponde a información bursátil sin procesar, posiblemente de diferentes fuentes y formatos ligeramente distintos. Estos datos son extraídos de una interfaces específicas para las fuentes de datos que use la aplicación. La idea sería poder dejar abierta la posibilidad de añadir más interfaces de datos en el futuro de ser necesario.

- **A.A.R.M:** estas siglas hacen referencia a (Abstract Arm Relational Model). Un A.A.R.M. es una estructura de datos que representa a un brazo perfectamente válido para ser utilizado en un entrenamiento por uno de los agentes implementados. Los A.A.R.M.s son el resultado de preprocesar y normalizar los datos bursátiles crudos de manera que todos puedan ser utilizados juntos. La existencia de esta estructura permite reusar los datos procesados de un experimento por otros y utilizar brazos que representan productos financieros de naturalezas diferentes en un mismo bandido. Esta estructura se explicará con más detalle más adelante.

- **Agents & Results:** el producto de los diferentes entrenamientos ejecutados por el administrador de agentes y las estadísticas resultantes se almacenan para futura consulta. También es conveniente almacenar configuraciones de agentes, o experimentos enteros en pausa, para uso posterior.

- **Misc:** en la base de datos también se ha de dar almacenamiento a la configuración de módulos auxiliares al administrador de agentes, como es el caso del propio *web framework*.

Concepto de interfaz con el usuario:

Este apartado del diseño no es de tanta importancia comparado con el administrador de agentes o el almacenamiento de datos. Sin embargo, es necesario proporcionar una interfaz que facilite el uso de las herramientas implementadas.

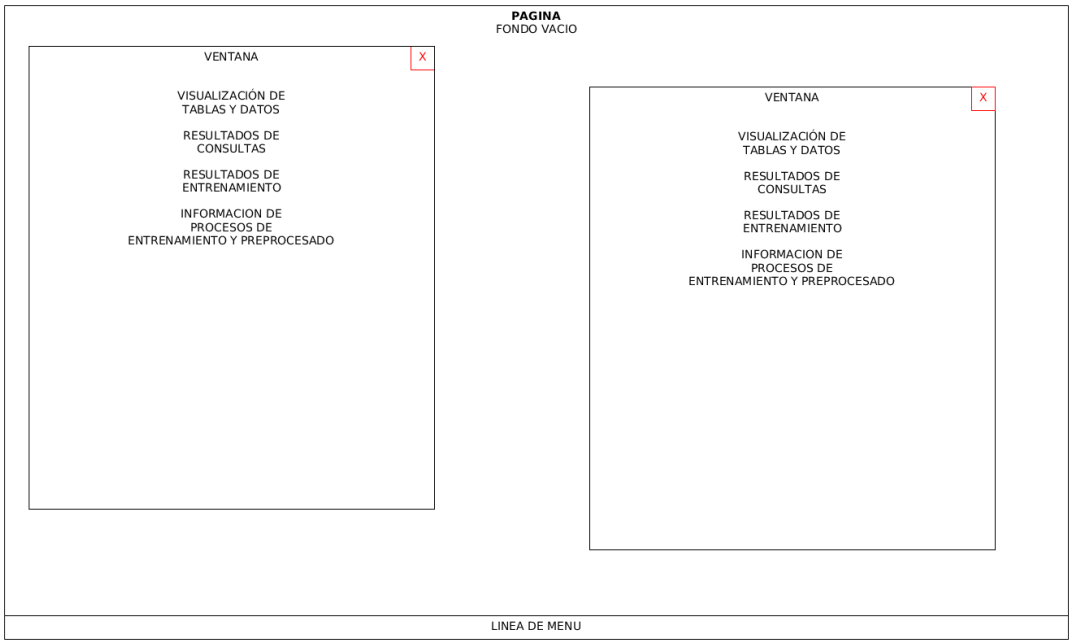

Figura 4.3: Concepto básico de una posible interfaz

La idea de este diseño sería un sistema de ventanas con una especie de menú en la parte inferior de la página. El usuario puede utilizar botones y formularios para interactuar con los datos y los agentes. El usuario visualiza los datos y esquemas de la aplicación a través de las ventanas. Las ventanas se podrían cerrar antes de efectuar acciones de manera que no se envía nada al servidor y se podría crear una jerarquía de ventanas en casos de uso más complejos.

Fuentes de datos de entrenamiento:

Dadas las limitaciones en recursos de este proyecto no se puede volcar demasiado esfuerzo en el desarrollo de este apartado. Aun siendo esto verdad, se ha de implementar algún sistema rudimentario con el cual expandir el cuerpo de datos de la aplicación.

Para poder llevar esto a cabo el usuario puede subir archivos CSV que contengan fechas y precios de un valor bursátil, de manera que luego la propia base de datos se encarge de procesar esta información cruda en un AARM nuevo.

Una posible funcionalidad de gran utilidad, que podría implementarse en el futuro, sería el uso de APIs de información financiera. Yahoo y Google entre muchas otras ofrecen este

servicio. Esto permitiría al sistema actualizar sus brazos automáticamente.

4.2 Implementación: Base de datos

Definición de un A.A.R.M. y estructuras auxiliares: La figura 4.4 muestra un modelo relacional para el almacenamiento de A.A.R.Ms, cada elemento sera explicado en detalle a continuación.

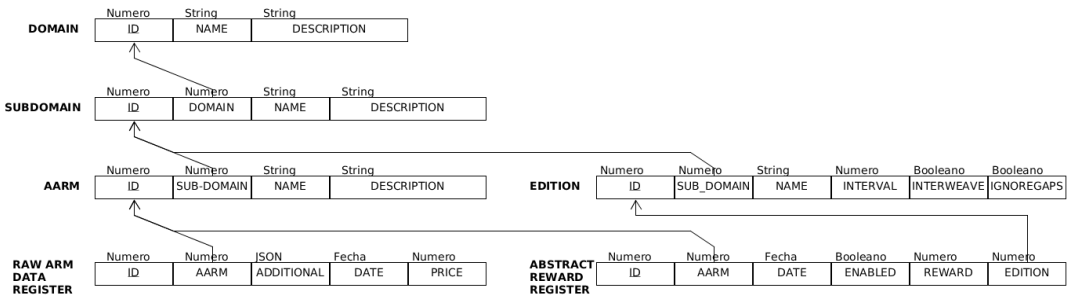

Figura 4.4: Estructura de los brazos abstractos

- **Domain:** es un dominio de A.A.R.Ms. Un ejemplo sería el mercado bursátil. En este proyecto solo hay un dominio que es el previamente mencionado. La razón de ser de esta relación es puramente organizativa. En el futuro otros usuarios podrían añadir dominios nuevos de cualquier tipo de brazo compatible con los agentes. Tienen dos campos de datos, el nombre y la descripción.
- **Subdomain:** hace referencia a un grupo de brazos semánticamente similares entre ellos y cuyo proceso de normalización dio lugar a valores de *reward* cuyo intervalo es compatible[2].
- **AARM:** representa a un brazo en el formato universal. Todos los AARMs comparten la misma estructura y han de ser compatibles con cualquier Bandido que pueda generar el administrador de agentes.
- **Edition:** es un caso de procesado de los datos crudos de un brazo en datos preparados para ser consumidos por un agente. Un AARM puede haber sido procesado en muchas ediciones diferentes, pero todas parten de los mismos datos. Las ediciones se suelen diferenciar en la distancia temporal entre sus iteraciones, pero también hay otras diferencias. Aparte de una referencia al subdominio al que pertenecen y a su nombre,

[2] Dos subdominios pueden tener un mismo proceso de normalización de manera que son compatibles también.

almacenan 3 campos que indican estas diferencias. **INTERVAL** es un número que representa la cantidad de días que transcurren entre una iteración y otra. **INTERWEAVE** es un booleano que indica si se entrelazan iteraciones (véase 4.5), **IGNOREGAPS** es un booleano que indica que las iteraciones pueden tener un período variable dependiendo de la disponibilidad de los brazos (véase 4.5).

- **Abstract Reward Register:** representa un único valor de *reward* normalizado según la política del sub-dominio. Se identifican a través del tiempo usando un *timestamp* que representa un punto de tiempo específico como un único número. El booleano **ENABLED** es verdadero cuando este brazo está habilitado en las iteraciones de entrenamiento cuyo valor temporal coincida con este registro o falso en caso contrario. De ser falso el brazo será ignorado por el agente ejecutor durante esa iteración. La tupla también incluye referencias a la iteración y subdominio que pertenece.

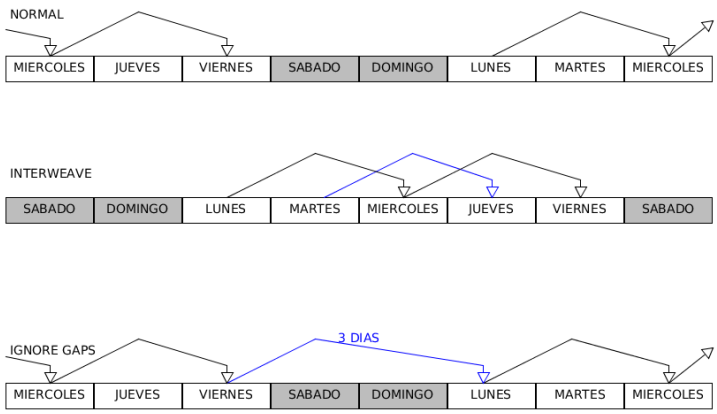

Figura 4.5: Modificaciones especiales en ediciones: normal, entrelazado e ignorar saltos. En todos los casos el intervalo es 2 días.

Definición de agentes y experimentos: La figura 4.6 muestra el modelo relacional para el almacenamiento de agentes, experimentos y sus ejecuciones en la bases de datos. Cada elemento sera explicado en detalle a continuación.

- **Agent**: tuplas de esta relación representan diferentes agentes con diferentes configuraciones posibles. A la hora de ser utilizados, el *agent runner* utiliza estos datos para construir un agente. Posee los siguientes parámetros:
  - GAUSSIAN: indica si el agente utiliza distribuciones posteriores gaussianas. En la versión actual del software, este valor en falso significa que el agente utiliza distribuciones gaussianas-gamma (véase 3.4).

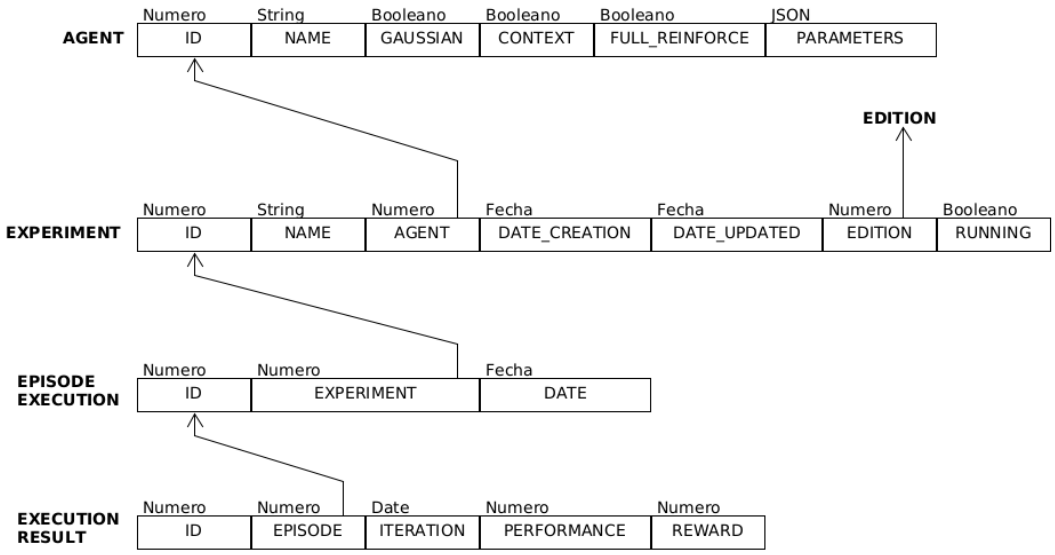

Figura 4.6: Estructura de registro de agentes y experimentos.

  - CONTEXT: indica si el agente utiliza contexto. En la versión actual del software, este valor en verdadero significa que el agente utiliza el contexto lineal del experimento E (véase 3.6).
  - FULL_REINFORCE: indica si el agente aplica aprendizaje en brazos que no son elegidos. Esta técnica se indica en el experimento D (véase 3.4).
  - PARAMETERS: un diccionario clave valor para guardar parámetros específicos de alguna configuración. También se prevé utilizar este campo para extender las funcionalidades del software en el futuro.

- **Experiment**: esta relación registra todos los experimentos que se pueden ejecutar o se han ejecutado. Para que un experimento sea posible es necesario un agente y una edición. De manera que la edición le será proporcionada al agente al momento de ser ejecutado el experimento. Aparte del id y el nombre, esta relación guarda fechas de creación y de última modificación, el id del agente que se utiliza y el id de la edición que se consume. También se guarda un parámetro booleano denominado **RUNNING** que informa de si el experimento está en curso en el presente.

- **Episode execution**: una tupla de esta relación representa a una ejecución específica del experimento al que referencia. Solo una ejecución por experimento se puede llevar a cabo simultáneamente. Sin embargo se pueden ejecutar ilimitados experimentos simultáneamente. Cada tupla almacena la fecha en la que la ejecución se llevó a cabo.

- **Execution result**: un episodio produce una gran cantidad de datos, estos son almacenados en esta relación. Cada tupla guarda los resultados de una iteración. Para ello registra: una referencia al episodio que la generó, la fecha de la iteración (**ITERATION**), el valor de rendimiento (**PERFORMANCE**) y la *reward* de la iteración. El valor de rendimiento representa el crecimiento de una hipotética inversión que comenzó en 100% al comienzo del episodio y va oscilando (normalmente en tendencia positiva) a lo largo de las iteraciones hasta el final del episodio.

4.3 Implementación: Interfaces

El diagrama de la figura 4.7 muestra a división de los diferentes componentes que conforman el software y las interfaces a través de las cuales se comunican.

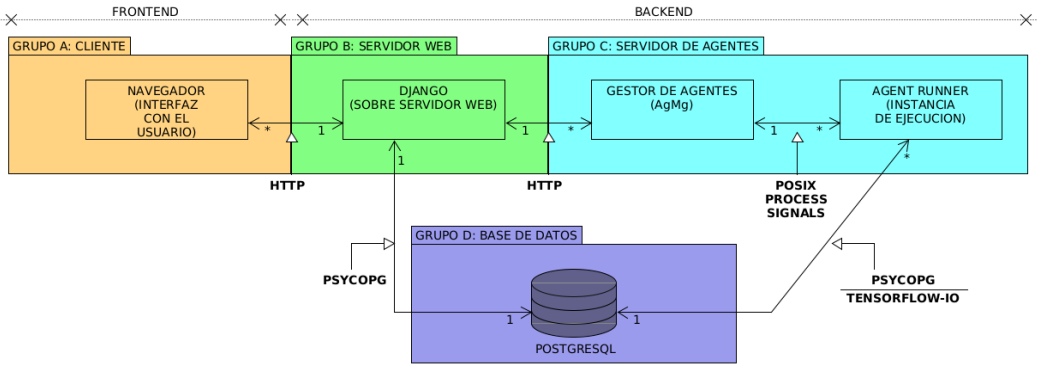

Figura 4.7: Interfaces y grupos

Los grupos que se muestran en la figura indican que esos módulos están pensados para poder funcionar en máquinas diferentes de ser necesario, para ello, las direcciones IP y puertos de los diferentes componentes han de ser conocidas en los grupos pertinentes.

La idea es aprovechar la arquitectura para ejecutar los grupos en máquinas especializadas para la tarea del grupo en cuestión. Un ejemplo de una instalación sería:

- El cliente es un portátil *lowcost*.
- El servidor web se encuentra en un ordenador con un procesador potente, detrás de capas de seguridad.
- La base de datos está instalada en una máquina con grandes capacidades de almacenamiento y computación.

- Hay dos servidores de agentes, ambos son máquinas con potentes tarjetas gráficas para acelerar las operaciones tensoriales.

Los diferentes *endpoints* de la interfaz entre el grupo A y el grupo B se describirán en detalle en la sección 4.4. Por otro lado la del grupo B con el C, y la del gestor de agentes con el *agent runner* se disertarán en la sección 4.5.

4.4 Implementación: Front-End

En este proyecto, el front-end consiste de una colección de archivos HTML, CSS y JavaScript cuyo propósito es la generación de un DOM[3] específico para los casos de uso del usuario en este sistema. Los casos de uso se muestran en la figura 4.10.

La interfaz de usuario consta, a grandes rasgos, de los siguientes elementos:

- Ventanas: son marcos cuadrados de tamaño variable y dinámico. En su interior se puede mostrar contenido específico del caso de uso al que pertenece la ventana. Todos tienen la posibilidad de cerrarse con una cruz en la esquina superior derecha. El contenido también puede integrar elementos interactivos. Un ejemplo de ventana se puede ver en la figura 4.8. Es posible mostrar el contenido de una ventana en una pestaña alterna.
- Menú principal: es una barra permanentemente ligada a la parte inferior del espacio en el que se "renderiza" la aplicación web. En el se encuentra información básica y global del sistema junto con accesos a diferentes casos de uso fundamentales. un ejemplo de menú se puede ver en la figura 4.9.

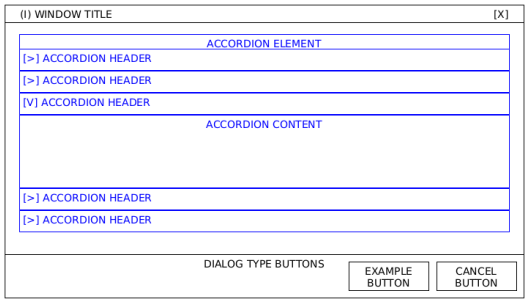

Figura 4.8: Ejemplo de ventana

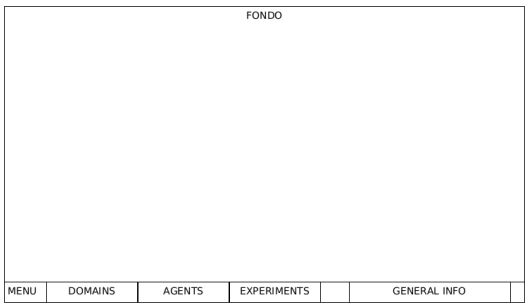

Figura 4.9: Estructura del menú

Para estructurar el contenido y personalizarlo a medida que el estado de la aplicación cambia, se utiliza el sistema de plantillas para construir los diferentes componentes de la aplicación. Cuando un elemento interactivo de la página provoca la generación de una ventana se producen los siguientes pasos:

[3] Document Object Model

1. El elemento llama a una función (*launchContent*) con la URL que corresponde a la ventana a generar (véase 24).
2. Esta función lanza una solicitud HTTP asíncrona para recuperar el contenido de dicha URL.
3. Cuando el contenido es recuperado este es pasado por parámetro a una función (*raiseContent*) que se encarga de los siguientes pasos.
4. Integrar el nuevo contenido en el DOM.
5. Aplicar las inicializaciones pertinentes a algunos elementos, como pueden ser la propia ventana o los "acordeones".
6. El contenido esta preparado para su consumición.

Ventanas: A continuación se muestra una lista con todas las ventanas que la aplicación implementa:

- **DOMAINS**: muestra un menú en formato acordeón en el que se muestran todos los dominios almacenados en la base de datos. Al expandir una de las cabeceras se mostrará un sub-menú de subdominios que son integrados en la interfaz en el momento que la cabecera es expandida. Al expandir una cabecera de subdominio se muestra una descripción del mismo junto con un conjunto de botones para acceder a las diferentes ediciones del mismo. Al abrir esta ventana en modo estándar, el botón de edición generará una nueva ventana EDITION. En modo selección enviará el id de la edición al formulario pertinente.
- **EDITION**: muestra información de una edición junto con todos los AARMs que la integran. Se puede filtrar que AARMs se muestran según su nombre.
- **AARM**: muestra información de un sólo brazo en una edición determinada.
- **AGENTS**: contiene un menú en formato acordeón en el que se muestran los agentes almacenados en la base de datos. Al lado del nombre del agente en la cabecera, se muestran iconos que informan del tipo de Bandido utilizado en su configuración. Al expandir el contenido se muestra más información del agente y opciones de gestión (como su eliminación). En la parte inferior de la ventana hay dos botones para crear tanto agentes (NEW_AGENT) como experimentos (NEW_EXPERIMENT).
- **NEW_AGENT**: esta ventana contiene un formulario para crear un agente. Una vez rellenados los campos se puede presionar un botón en la parte inferior para enviar los datos al servidor.

- **EXPERIMENTS**: contiene un menú en formato acordeón en el que se muestran los experimentos almacenados en la base de datos. Al lado del nombre de experimento en la cabecera, se muestra un icono que indica si el experimento está en ejecución. Al expandir el contenido se muestra más información junto con un elemento interactivo para acceder a la ventana del experimento en cuestión (EXPERIMENT).
- **NEW_EXPERIMENT**: esta ventana contiene un formulario para crear un experimento. Una vez rellenados los campos se puede presionar un botón en la parte inferior para enviar los datos al servidor. A la hora de escoger una edición y un agente, las ventanas DOMAINS y AGENTS se podrán abrir en modo selección.
- **EXPERIMENT**: en esta ventana se muestra información sobre un experimento y sus ejecuciones.
- **CHART WINDOW**: esta ventana sirve para mostrar un gráfico cuyo contenido depende del contexto que genere a la ventana.

Casos de uso: los casos de uso planteados para esta aplicación web están conceptualmente muy relacionados con las ventanas que se implementan. Véase el diagrama de la figura 4.10.

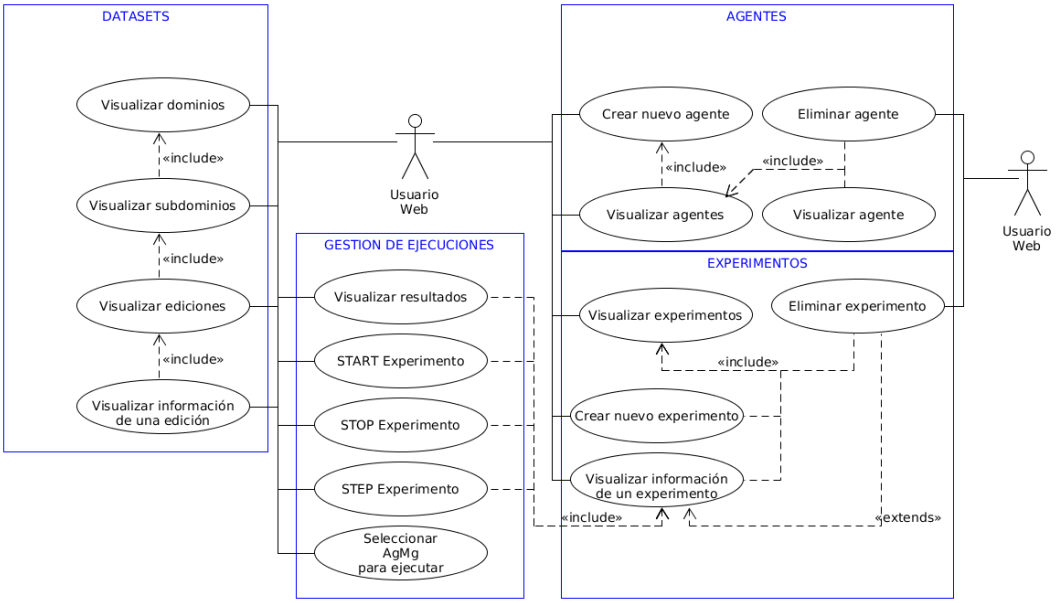

Figura 4.10: Diagrama de casos de uso del usuario web en la aplicación demostrativa

Interfaz http front-end/back-end: el servidor web acepta la siguiente colección de solicitudes **HTTP**:

URL	Tipo	Descripción
admin/	GET	Envía la página de administración de Django.
domains	GET	Envía el contenido de la ventana DOMAINS.
domain/<domain_id>/subdomains	GET	Completa el contenido de la ventana DOMAINS con datos de los subdominios de un dominio.
subdomain/<subdomain_id>/editions	GET	Completa el contenido de la ventana DOMAINS con datos de las ediciones de un subdominio.
edition/<edition_id>	GET	Envía el contenido de la ventana EDITION, es decir, una edición determinada.
aarm/<aarm_id>	GET	Envía el contenido de la ventana AARM, es decir, un AARM determinado.
agents	GET	Envía el contenido de la ventana AGENTS/.
experiments	GET	Envía el contenido de la ventana EXPERIMENTS.
new_agent	GET	Envía el contenido de la ventana NEW AGENT.
new/agent	POST	Recibe los datos para la creación de un agente.
new_experiment	GET	Envía el contenido de la ventana NEW EXPERIMENT
new/experiment	POST	Recibe los datos para la creación de un experimento.
experiment/<experiment_id>	GET	Envía el contenido de la ventana EXPERIMENT.
start/<exp_id>	GET	Envía una solicitud de START a un experimento determinado.
stop/<exp_id>	GET	Envía una solicitud de STOP a un experimento determinado.
step/<exp_id>	GET	Envía una solicitud de STEP a un experimento determinado.

Tabla 4.1: Lista de solicitudes HTTP legales en la aplicación web.

4.5 Implementación: Back-End

El back-end de la aplicación se basa fundamentalmente en 3 elementos:

- **Aplicación Web**: sólo hay una por instalación. Se implementa utilizando Django y se encarga de comunicar el estado de la aplicación al cliente HTTP que utiliza el usuario. También sirve de intermediaria entre el usuario y los *Agent Manager*s.
- **Agent Manager**: puede haber varios por instalación. Su propósito es recibir órdenes desde la app. web y en consecuencia administrar una colección de *Agent Runners*.
- **Agent Runner**: puede haber varios por *Agent Manager*. Su funcionalidad principal es llevar a cabo la ejecución de un experimento.

Interfaces entre los componentes:

En la figura 4.11, se pueden apreciar los protocolos que utilizan los diferentes elementos del back-end para sincronizar su funcionamiento. Los componentes en amarillo indican que son emisores o receptores de señales de proceso POSIX, los componentes en verde son receptores de HTTP que sirven a la aplicación web.

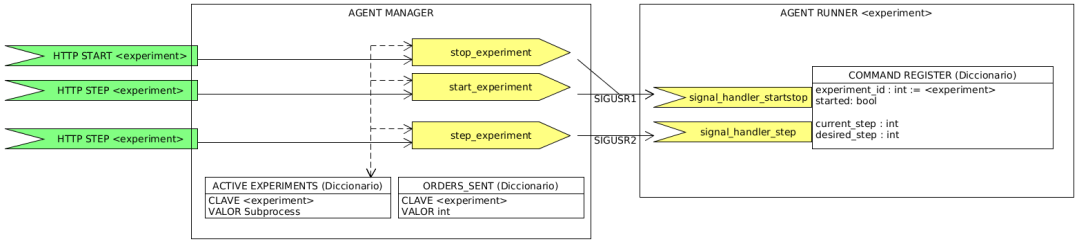

Figura 4.11: Comunicación entre los componentes del back-end

Como consecuencia de su protocolo de comunicación, un proceso *Agent Manager* y un proceso de *Agent Runner* al que este administra han de estar en el mismo entorno de ejecución. En la tabla 4.2 se muestra como evoluciona una orden del usuario a un agente en las diferentes interfaces.

Aplicación Web:

Este módulo sirve para proveer, al cliente utilizado por el usuario, los datos e interactividad con los que el usuario lleva a cabo el uso del la aplicación. Se basa en el *web framework* de Django [9] para llevar a cabo este cometido. En la figura 4.12 se expone como esta se comunica con el resto del sofware.

Instrucción del usuario (Solucitud HTTP)	Solicitud HTTP la App. Web al AgMg	Señal POSIX del AgMg al Agent Runner
START <ID experimento>? <ID AgMg>	http://<Direccion del AgMg>/start/<ID experimento>	SIGUSR1 >><PID runner>
STEP <ID experimento>? <ID AgMg>	http://<Direccion del AgMg>/step/<ID experimento>	SIGUSR1 >><PID runner>
STOP <ID experimento>? <ID AgMg>	http://<Direccion del AgMg>/stop/<ID experimento>	SIGUSR2 >><PID runner>

Tabla 4.2: Secuencia de comunicación entre el usuario y un *agent runner*.

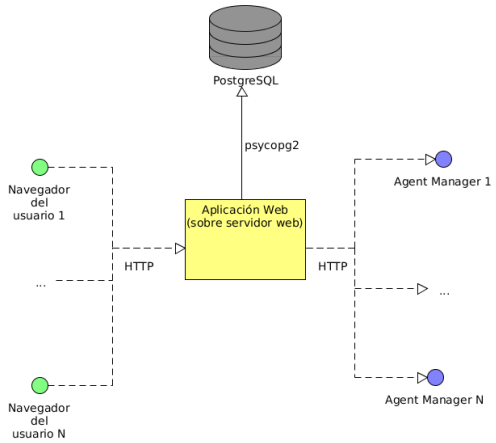

Figura 4.12: Relaciones de la aplicación web. La dirección de las flechas indica el sentido de las solicitudes.

En cuando al sistema de plantillas[4] utilizado por la aplicación:

- A la página inicial se le denomina **dashboard** y hereda de **frame** que contiene el fondo de la aplicación.
- Las plantillas que generan una ventana heredan de una plantilla (**window**) que proporciona los elementos necesarios para la construción de las mismas.
- Otros contenidos que son integrados a través de solicitudes AJAX son variados y no siguen una pauta específica de la que hereden.

Agent Manager:

Las instancias de *Agent Manager* son procesos bastante simples, su razón de ser se basa en desacoplar la administración de ejecuciones de las ejecuciones en sí, de manera que cada ejecución tiene un proceso propio que las aísla. Además, separar las ejecuciones en procesos

[4] Documentos que se utilizan, junto con datos dinámicos, para generar el HTML que se envía al navegador. El siguiente link ofrece más información acerca de las plantillas.

delega la distribución de los recursos del sistema al sistema operativo ahorrando trabajo de implementación.

Este módulo utiliza la biblioteca **FastAPI** para recibir solicitudes HTTP de la aplicación web y las bibliotecas **signal** y **subprocess** para generar y controlar instancias de *agent runner*. El servidor HTTP sobre el que opera el módulo es **Uvicorn**,

Dado que el protocolo que comunica al *Agent Manager* con los *Agent Runners* tiene un total de 2 señales para 3 instrucciones diferentes, hace falta mantener un estado en ambas entidades de manera que recibir una señal puede implicar significados diferentes dependiendo de la señal que se envió previamente.

Cuando un *Agent Manager* emite una señal SIGUSR1 puede implicar 3 cosas:

- START si la anterior señal SIGUSR1 significaba STOP
- STOP si la anterior señal SIGUSR1 significaba START
- START si no hay señal anterior.

Cada vez que un *Agent Manager* emite una señal SIGUSR2 a un *runner* significa que este ha de ejecutar exactamente una iteración más del episodio en el que se produce el experimento, es decir, STEP.

Las instancias de *Agent Manager* guardan los experimentos y las instrucciones que les fueron enviadas en los registros que se pueden apreciar en la figura 4.11.

Agent Runner:

Las instancias de *Agent Runner* llevan a cabo las ejecuciones de los experimentos. Estos experimentos utilizan la biblioteca **Tensorflow** como base de la implementación de los experimentos. El diagrama de clases de la figura 4.13 muestra, en perspectiva global, la estructura de un *Agent Runner*.

Este módulo utiliza un diccionario denominado como **command_register** cuyos valores dictaminan el comportamiento del *driver* que lleva a cabo el experimento. Este proceso cuenta con dos manejadores de señales POSIX que alteran el valor de este registro, de manera que a través de estas señales se puede controlar la ejecución, sin interrumpir indebidamente la iteración que se este llevando a cabo.

Cuando se lleva a cabo un experimento, es el propio *runner* es el que se encarga de interactuar con la base de datos, tanto para leer el material necesario para el experimento como para escribir los resultados del mismo. Para ello, una parte de la implementación encapsulada en un módulo denominado *db_interface* se encarga específicamente de esta tarea. Este módulo contiene dos clases denominadas *Reader* y *Writer* cuyas funcionalidades describe su nombre.

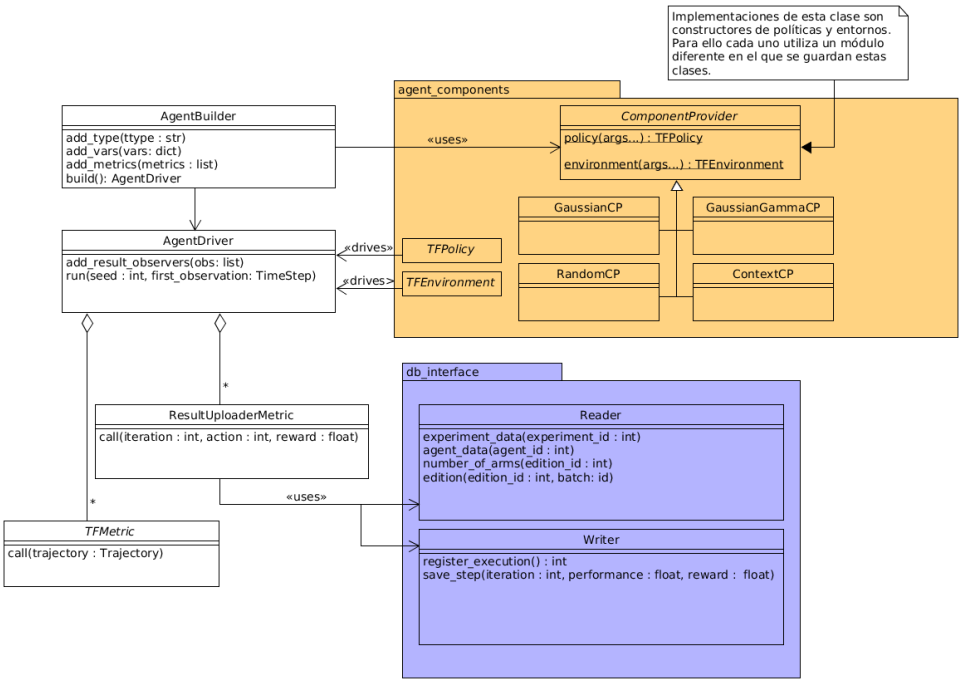

Figura 4.13: Estructura de un *Agent Runner*

A la hora iniciar un experimento, un *agent runner* es instanciado por el *agent manager*. Al comenzar la ejecución, se procede a construir el agente que se va a utilizar siguiendo las pautas de la base de datos y se importan los *datasets* pertinentes. Para construir un agente se utiliza una clase constructora denominada **AgentBuilder**. Durante la construcción se le proporciona a esta clase un proveedor de componentes de un tipo de agente determinado. En esta versión de la aplicación hay 4:

- Agente con distribuciones posteriores gaussianas (Experimento D 3.4).
- Agente con distribuciones posteriores gaussianas-gamma (Experimento D 3.4).
- Agente con política de decisión aleatoria (se utiliza como control).
- Agente con distribuciones posteriores gaussianas y contexto lineal (Experimento E 3.6).

Una vez se asigna este proveedor, este es utilizado para construir la política y el entorno que se usarán en el experimento. Finalmente estos dos componentes son ensamblados en una instancia de **AgentDriver** y el experimento esta listo para ser ejecutado.

El *driver* comunica los resultados que se van obteniendo a través de un observador, el cual es una instancia de **ResultUploader**, el cual se comunica con una instancia de **Writer** para subir los resultados a la base de datos.

Capítulo 5

Metodología y organización de desarrollo:

En este capítulo se va a exponer la metodología utilizada para administrar los recursos y llevar a cabo el proyecto global del TFG. En una primera parte se especifica el uso de SCRUM y porque se ha decidido utilizarlo. En la segunda parte se expone la consumición total de recursos por parte del proyecto. Y en una tercera, y última parte, se desgranan los diferentes *sprints* del proyecto.

5.1 Metodología:

Para este TFG se decidió llevar a cabo una versión de SCRUM adaptada a las necesidades y los recursos específicos del proyecto. En síntesis, SCRUM es un marco de desarrollo perteneciente a un paradigma conocido como Agile. Esta metodología se basa en segmentos temporales denominados *sprint*s en los que se desarrolla incrementalmente el producto final. A cada *sprint* se le asignan una serie de objetivos denominados historias de usuario, de manera que los contribuyentes del *sprint* los intentan alcanzar el el tiempo que este dura. Algunas de estas historias de usuario pueden requerir más *sprints*, a las tareas que parten de estas historias se les denomina *epics*. SCRUM también incluye diferentes roles y formatos de comunicación entre los contribuyentes del proyecto, pero estos son desestimados debido a que, en este proyecto, solamente se cuenta con un contribuyente [1]. En su lugar se definirán 2 roles a continuación.

Este proyecto se divide en *sprints* cuya duración es de 2 semanas. En cuanto a recursos humanos, hay dos roles implicados:

- **Desarrollador**: el principal actor del proyecto. Es el que genera, organiza y lleva a cabo las tareas de todos los *sprints*. El desarrollador es el alumno que cursa el TFG.

[1] Bajo guía y supervisión de los directores del proyecto

- ***Product Owners***: son 2. Su función fundamental es asistir, asesorar y auditar en pos del correcto desarrollo del proyecto. Este rol corresponde a los directores del proyecto.

5.2 Coste de recursos por parte del proyecto:

Recursos humanos:

El proyecto requirió de un total de **15 *sprints*** de 2 semanas de 5 días laborables a 3 horas por día. Lo que da una cantidad aproximada de unas **450 horas dedicadas al proyecto** por el desarrollador. El coste por hora del desarrollador es de 18 EUR.

Los *Product Owner*s dedicaron una cantidad conjunta aproximada de 2 horas por *sprint*, lo que resulta en **30 horas totales de supervisión**.

Recurso	Coste por hora	Horas uso	Coste total
Desarrollador	18 EUR	450	8.100 EUR
Product Owner	30 EUR	30	900 EUR

Tabla 5.1: Costes de los recursos humanos

Recursos materiales:

CPU	GPU	Memoria principal	Tiempo de uso	Gastos estimados de amortización
AMD Ryzen 7 2700	NVIDIA GeForce RTX 2080	64Gb	6048 Horas	468.75 EUR
AMD Ryzen Threadripper 3960X	NVIDIA RTX A6000	128Gb	336 Horas	145,83 EUR

Tabla 5.2: Consumo estimado de los ordenadores utilizados para desarrollar el proyecto

5.3 Registro de *sprints*:

Los diferentes sprints que forman parte del proyecto se pueden agrupar en base a las temáticas de las tareas que están asignadas a los mismos. Se distinguen y definen 5 grupos:

- **Investigación** (Inv): contienen tareas relacionadas con la recopilación y asimilación de la información necesaria para llevar a cabo el trabajo.
- **Desarrollo** (Dev): relativos a tareas que implican un diseño de una entidad. Sea esta un algoritmo, una estructura de datos o una aplicación web. La implementación de experimentos entra dentro de este apartado debido a que se conciben como prototipos para el desarrollo de la aplicación demostrativa.

- **Implementación** (Imp): como dice el nombre, incluyen principalmente tareas de implementación de la aplicación demostrativa.
- **Pruebas y correcciones** (Debug): contienen tareas destinadas a la validación de la aplicación demostrativa y sus agentes.

La distribución de sprints por agrupación se puede ver en la figura 5.1.

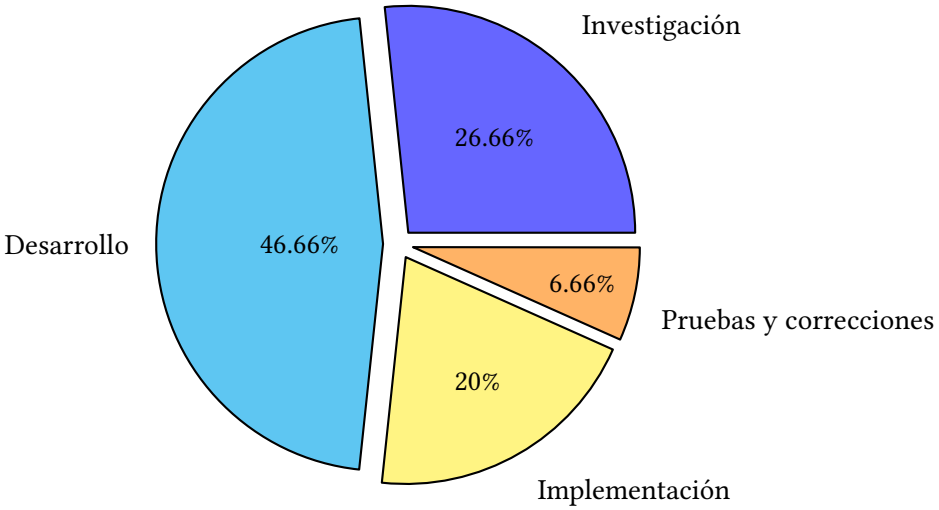

Figura 5.1: Distribución de los sprints

Los 15 *sprints* que componen el desarrollo del proyecto se exponen a continuación (figura 5.3):

Número	Fecha	Nombre
1	13 SEP / 26 SEP	Inv - Bandido Bayesiano Multibrazo.
2	27 SEP / 08 OCT	Inv - Métodos para la mejora del modelo inicial. Experimento A.
3	11 OCT / 24 OCT	Inv - Extensión del modelo de Bandido. Experimento B.
4	25 OCT / 08 NOV	Inv - Otros modelos e ideas anexas, miscelánea.
5	09 NOV / 21 NOV	Dev - Familiarización con herramientas e ideas básicas.
6	22 NOV / 05 DEC	Dev - Experimento C.
7	06 DEC / 23 DEC	Dev - Modelo de administrador de agentes.
8	31 ENE / 13 FEB	Dev - Refactorización, pruebas, APIs y diseños.
9	14 FEB / 27 FEB	Dev - Experimento D parte I/III.
10	28 FEB / 13 MAR	Dev - Desarrollo a partir de las ideas de contexto.
11	14 MAR / 27 MAR	Dev - Diseño de la aplicación demostrativa. Experimento D parte II/III.
12	18 ABR / 01 MAY	Imp - Modelos relacionales y RDBMS. Experimento D parte III/III.
13	16 MAY / 31 MAY	Imp - Front-end.
14	01 JUN / 15 JUN	Imp - Back-end. Memoria I/II.
15	15 JUN / 28 JUN	Debug - Pruebas de la aplicación demostrativa. Memoria II/II.

Tabla 5.3: Lista de sprints del proyecto.

Capítulo 6

Conclusiones y trabajo futuro

El principal objetivo de este trabajo ha sido investigar y evaluar la viabilidad de la aplicación de Bandidos Bayesianos como un sistema de recomendación a la hora de invertir en mercados bursátiles. En consecuencia se llevaron a cabo una serie de experimentos para resolver la incertidumbre planteada por el objetivo del proyecto.

Los diferentes experimentos aportaron, de manera incremental, la información necesaria para finalmente formular la respuesta a la pregunta. ¿Son los Bandidos Bayesianos un modelo de sistema de recomendación apto para el uso en el ámbito de los instrumentos financieros? La respuesta es, **técnicamente sí**, aunque aún queda mucho por hacer para así asegurar la estabilidad de su funcionamiento.

Los resultados de los experimentos indican que se necesitan más pasos intermedios y elementos observables, para así aportarle al agente las mejores posibilidades de predecir algo del calibre de la inexpugnable incertidumbre que domina el futuro de las finanzas.

Cabe destacar que los algoritmos implementados en este proyecto son solamente una base, un comienzo, del cual parte una senda de perfeccionamiento aún por ser tomada. De ahí surge el segundo objetivo del trabajo, el desarrollo de una herramienta que permita seguir trabajando con mayor facilidad en las bases construidas por el proyecto. Metafóricamente, los experimentos son los cimientos, la aplicación demostrativa las herramientas.

En cuanto a la perspectiva personal, este trabajo ha sido, en todas sus fases, intensivamente formativo para el desarrollador, que ha tenido la oportunidad poner en práctica variadas disciplinas de las ciencias de la computación.

Se han cumplido los siguientes **objetivos**:

- Estudiar y evaluar, de manera empírica, la viabilidad de la aplicación de modelos de Bandidos Bayesianos Multibrazo al entorno bursátil.
- Desarrollar una aplicación que contribuya a acelerar la experimentación y desarrollo de Bandidos Bayesianos Multibrazo.

A base de los siguientes **resultados**:

- Un total de 5 experimentos que aportan los resultados necesarios para alcanzar el primer objetivo.
- Una aplicación demostrativa, la cual cumple con los requisitos especificados por el segundo objetivo.

Trabajo futuro:

La experimentación y desarrollos teóricos, aunque abundantes desde la perspectiva del desarrollador, solo son el principio, una fundamentación sobre la que extender los planteamientos y diseños en gran medida.

Este trabajo esta rodeado por un océano de posibilidades de extensión. Y la aplicación desarrollada fomenta la exploración del mismo. Es de relevancia considerar los siguientes puntos, los cuales quedaron pendientes de revisión, en futuras contribuciones a esta temática:

- Probar los modelos del Experimento D en diferentes productos financieros de naturaleza distinta a acciones del S&P500.
- Implementar un bandido bayesiano de múltiple elección. Posiblemente uno que modere de manera automática las proporciones de inversión que van a cada elemento.
- Probar los modelos del Experimento D en entornos mixtos en los que se combinan brazos de distintas naturalezas[1].
- Desarrollar políticas alternativas a Thompson Sampling para Bandidos Bayesianos.
- Diseñar e implementar modelos de Bandidos contextuales con diferentes configuraciones de vectores de características.
- Evaluar si la técnica de refuerzo completo convierte al agente en una estructura equivalente a una red neuronal. Evaluar si esto puede implicar que son las redes neuronales mejores candidatos para tratar este problema.
- Extender las funcionalidades de la aplicación para aportarle más flexibilidad de uso.
- Generar un nuevo instrumento de análisis técnico a partir de las distribuciones a priori de un Bandido aplicado lo que se está midiendo[2].

[1] Lógicamente, conservando el objetivo común de maximizar rentabilidad.
[2] La información ya esta presente solo es necesario visualizarla

Apéndices

Apéndice A

Leyenda y notación

Teorema: Ejemplo de teorema

En estos cuadros figuran teoremas extraídos de referencias o conocimiento de dominio público.

Definición: Ejemplo de definición

En estos cuadros figuran definiciones que han sido sintetizadas a partir del desarrollo del trabajo y de teoremas expuestos previamente. Pueden incluir figuras auxiliares.

Anotación: Ejemplo de anotación

En estos cuadros figuran notas que son de importancia pero por alguna razón no encajan el el flujo del tema que se trata. Normalmente porque se ramifican sobre el tronco principal del contenido.

Bibliografía

[1] T. Bayes, "Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s," *Philosophical transactions of the Royal Society of London*, no. 53, pp. 370–418, 1763.

[2] H. Raiffa and R. Schlaifer, *Applied statistical decision theory / Howard Raiffa and Robert Schlaifer.*, ser. Studies in managerial economics. Boston Mass.: Division of Research, Graduate School of Business Administration, Harvard University, 1961.

[3] D. Fink, "A compendium of conjugate priors," 1997.

[4] J. Komiyama, J. Honda, and H. Nakagawa, "Optimal regret analysis of thompson sampling in stochastic multi-armed bandit problem with multiple plays," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1152–1161.

[5] K. P. Murphy, "Conjugate bayesian analysis of the gaussian distribution," *def*, vol. 1, no. $2\sigma 2$, p. 16, 2007.

[6] D. Cortes, "Adapting multi-armed bandits policies to contextual bandits scenarios," *arXiv preprint arXiv:1811.04383*, 2018.

[7] S. Agrawal and N. Goyal, "Thompson sampling for contextual bandits with linear payoffs," in *International conference on machine learning*. PMLR, 2013, pp. 127–135.

[8] B. Everitt, *The Cambridge dictionary of statistics*. Cambridge, UK; New York: Cambridge University Press, 2002. [En línea]. Disponible en: http://www.worldcat.org/search?qt=worldcat_org_all&q=052181099X

[9] "The web framework for perfectionists with deadlines." [En línea]. Disponible en: https://www.djangoproject.com/