Facultad de Ciencias Económicas y Empresariales

ICADE

Optimización bayesiana multiobjetivo de los hiperparámetros de un algoritmo de aprendizaje reforzado profundo en el ámbito financiero

Autor: Obdulia Freire Pasquau

Director: Eduardo César Garrido Merchán

MADRID, Junio 2024

Resumen

Los mercados financieros son cada vez más complejos, presentando desafíos significativos para los analistas y gestores de carteras. Sin embargo, con la aparición de herramientas avanzadas como la inteligencia artificial y el machine learning, se están desarrollando nuevas técnicas para tratar de predecir y optimizar el comportamiento del mercado. En este contexto, la optimización multiobjetivo, en particular a través de enfoques como la optimización bayesiana y el random search, ha cobrado relevancia. Estas técnicas permiten manejar múltiples objetivos simultáneamente, ofreciendo una forma más sofisticada de abordar la toma de decisiones en la gestión de carteras. La optimización bayesiana ha demostrado ser superior en otros sectores, como la robótica, gracias a su capacidad para manejar la incertidumbre, trabajar con grandes volúmenes de datos y mejorar el rendimiento en entornos complejos. Sin embargo, en el sector financiero, debido a la imprevisibilidad del mercado y la existencia de ruido, todavía hay muy poca investigación en esta área. Esta investigación se centra en comparar y demostrar la superioridad de la optimización bayesiana multiobjetivo frente al random search, evaluando su efectividad mediante la métrica del hipervolumen.

Palabras clave: optimización, multiobjetivo, hipervolumen, PPO, DRL.

Abstract

Financial markets are becoming increasingly complex, presenting significant challenges for analysts and portfolio managers. However, with the advent of advanced tools such as artificial intelligence and machine learning, new techniques are being developed to predict and optimize market behavior. In this context, multi-objective optimization, particularly through approaches such as Bayesian optimization and random search, has gained relevance. These techniques allow for the simultaneous handling of multiple objectives, offering a more sophisticated way to approach decision-making in portfolio management. Bayesian optimization has proven to be superior in other sectors, such as robotics, due to its ability to handle uncertainties, work with large volumes of data and improve performance in complex environments. However, in the financial sector, due to the unpredictability of the market and the existence of noise, there is still very little research in this area. This research focuses on comparing and demonstrating the superiority of multi-objective Bayesian optimization over random search, evaluating their effectiveness using the hypervolume metric.

Keywords: optimization, multi-objective, hypervolume, PPO, DRL.

Índices de contenidos

Índice de figuras

1. Introducción

1.1. Justificación del tema

Los mercados financieros ofrecen multitud de alternativas de inversión, que abarcan una amplia variedad de activos con diferentes niveles de rentabilidad, liquidez y volatilidad (CNMV, s.f.). En el ámbito de la gestión de carteras, el objetivo principal de los inversores reside en encontrar el equilibrio entre la rentabilidad y la volatilidad, sin embargo, los mercados financieros son complejos y tienen un componente de incertidumbre que dificulta encontrar una proporción adecuada (Chordia, T. et al., 2001). En respuesta a este desafío, surgen teorías que tratan de desarrollar estrategias y métodos para optimizar la asignación de recursos financieros, con el objetivo de encontrar un equilibrio óptimo entre rendimiento y riesgo (Chambers, R. G., et al., 2002).

En 1950 Harry Markowitz desarrolló la Teoría Moderna de la Cartera, una de las contribuciones más influyentes en este campo, que propuso la diversificación como clave para reducir el riesgo sin sacrificar los rendimientos (Kumar, V., 2018). Esto se consigue mediante la combinación de activos que no estén perfectamente correlacionados, ya que los movimientos de los precios de los activos individuales tienden a compensarse entre sí. Para implementar la teoría, se calcula una frontera de carteras eficiente, o frontera de Markowitz, que representa todas las posibles combinaciones de activos que ofrecen el máximo rendimiento esperado para un determinado nivel de riesgo (Jobson & Korkie, 2012). Es una teoría que está aceptada hoy en día en el mundo tanto académico como laboral, y que destaca principalmente por su facilidad en la ejecución. A pesar de su gran contribución al mundo financiero, esta teoría tiene varias limitaciones entre las que destacan asumir una distribución normal de los retornos, una alta dependencia en los datos históricos y la inadaptabilidad a las condiciones cambiantes del mercado (Grajales Bedoya, D. D., 2009).

A medida que la tecnología avanza y los mercados financieros se vuelven cada vez más complejos (Wieland, O. L., 2015)., los enfoques tradicionales para la gestión de carteras están mostrando sus limitaciones en términos de capacidad para abordar la multitud de variables y escenarios que surgen constantemente en el mercado (Garrido Merchán, E. C., 2023). El uso de infraestructuras tecnológicas de gestión del Big Data y

Machine Learning Operations (MLOps) son capaces de superar estas limitaciones e incluir mejoras en la toma de decisiones y optimizar la gestión de carteras (Xu, J., 2022). Emplean algoritmos sofisticados y técnicas de aprendizaje automático, que permiten trabajar con grandes volúmenes de datos financieros y explorar los complejos patrones presentes en ellos (Sohangir, S., et al., 2018). Esto incluye la capacidad de analizar una amplia gama de indicadores técnicos y sus combinaciones, lo que proporciona una visión más completa del riesgo y el rendimiento de una cartera. Mediante algoritmos de inteligencia artificial (IA) se permite una toma de decisiones más rápida y precisa al automatizar procesos analíticos e identificar oportunidades de inversión en tiempo real (Yu, P., et al., 2019). Además, estos algoritmos y nuevas tecnologías pueden adaptarse dinámicamente a cambios en el mercado y ajustar estrategias de inversión de manera proactiva para optimizar el rendimiento de la cartera (Garrido Merchán, E. C., 2023).

Entre los algoritmos que están revolucionando la manera en la que se analiza el mercado, se encuentra el aprendizaje profundo por refuerzo (DRL), que ofrece un enfoque más flexible y que permite una mayor adaptabilidad a las condiciones del mercado actual (Yu, P., 2019). Una de las ventajas de emplear estos algoritmos es su capacidad de adaptación a las condiciones del mercado en tiempo real, lo que permite al inversor ajustar sus estrategias de inversión de forma dinámica frente a condiciones volátiles del mercado (Zhang, Z. et al., 2019). Además, permite trabajar con relaciones no lineares entre las variables, lo que permite elaborar modelos financieros más complejos y con una mayor precisión (Huang, S. et al., 2024).

Por último, el uso de DRL (Deep Reinforcement Learning) contempla la posibilidad de considerar múltiples objetivos simultáneamente (Jyothi, R. et al., 2022). Mientras que la Teoría Moderna se centra exclusivamente en la maximización del rendimiento para un determinado nivel de riesgo, el aprendizaje reforzado (RL) permite considerar además otros factores, desde indicadores financieros como la pérdida máxima acumulada (máximum drawdown en inglés), hasta factores no financieros como los criterios ESG (Environmental, Social and Governance). Este último se refiere a aspectos medioambientales, sociales y de gobernanza, que están cobrando una creciente importancia en el ámbito financiero (Rumyantseva, A. et al., 2022).

1.2. Estado de la cuestión y metodología

El objetivo principal de este trabajo consiste en realizar un experimento para comprobar si existen indicios de que la optimización bayesiana multiobjetivo y el DRL son capaces de predecir una mejor rentabilidad en el ámbito de la gestión de carteras. Para ello se ha realizado un experimento en el que se compara la rentabilidad obtenida por un modelo de optimización bayesiana multiobjetivo con un modelo de búsqueda aleatoria.

La investigación se estructura en cinco secciones con el fin de abordar de manera el tema en cuestión. En la primera parte del estudio, se lleva a cabo una revisión de la literatura pertinente, proporcionando un marco conceptual y teórico para la comprensión del experimento realizado. A continuación, se define el alcance mediante la definición de los objetivos, la hipótesis, las asunciones y las restricciones del proyecto. En la siguiente sección se expone el marco teórico, donde se lleva a cabo una revisión de los estudios realizados en las dos áreas principales de la tesis: la optimización bayesiana multiobjetivo y el DRL en el ámbito financiero, para sentar una base de la investigación realizada con el tema del trabajo. Posteriormente, se explica el procedimiento que se ha seguido y se presentan los resultados. Por último, se elaboran unas conclusiones derivadas del análisis. Al final del informe se incluyen las referencias bibliográficas pertinentes y los anexos correspondientes.

2. Estado del Arte

El problema de la selección de carteras es un tema estándar en la ingeniería financiera y ha recibido mucha atención en las últimas décadas (Branke et al., 2009). Este campo es complejo y ha generado numerosas teorías y algoritmos para resolver u optimizar los métodos de selección de carteras. Desde enfoques tradicionales como el enfoque de optimización de cartera de Markowitz basado en la media-varianza o el modelo de asignación de activos de Black y Litterman (Garrido Merchán, E. C., et al., 2023).

El enfoque clásico de optimización de carteras, basado en el modelo de media-varianza, ha sido objeto de críticas debido a sus supuestos poco realistas (Dai, Z., et al., 2019). Como respuesta a estas críticas, en los últimos años se han desarrollado modelos de optimización de carteras que incorporan restricciones más realistas, como la cardinalidad y las cotas mínimas y máximas de participación de activos en la cartera, así como medidas alternativas de riesgo como VaR y lower partial momento (García, F., 2019). Sin embargo, la introducción de estas restricciones y medidas adicionales ha añadido complejidad a la resolución del problema de optimización de carteras (Metaxiotis, K., & Liagkouras, K., 2012).

Para abordar esta complejidad, los algoritmos evolutivos (EAs) han demostrado ser herramientas fundamentales en la resolución de problemas de optimización que son demasiado complejos para ser tratados con técnicas determinísticas (Bartz-Beielstein, T., 2014). Los EAs son especialmente adecuados para problemas de optimización multiobjetivo (MOP), ya que se inspiran en los principios de selección natural y reproducción genética de Darwin, los cuales son inherentes a la naturaleza multiobjetivo de estos problemas (Metaxiotis, K., & Liagkouras, K., 2012). Gracias a los algoritmos evolutivos multiobjetivo (MOEAs), el modelo clásico de media-varianza ha sido extendido para abordar objetivos en conflicto y restricciones diversas de manera más efectiva (Zhou, A., et al., 2011)

Existen varios estudios que aplican métodos multicriterio para optimizar carteras, se pueden mencionar, entre otros, los siguientes: un enfoque multicriterio en un marco clásico de teoría de utilidad bajo incertidumbre, en lugar de uno lineal (Ballestero et al., 2011); un marco de tres criterios para la optimización inversa de carteras ESG (Utz et al., 2013); una modificación del modelo de Markowitz a través de un nuevo modelo de tres

criterios que permite a los inversores personalizar sus asignaciones de activos e incorporar todas las preferencias personales con respecto a rentabilidad, riesgo y responsabilidad social (Gasser et al., 2017); un modelo híbrido de selección de carteras ESG que combina toma de decisiones multicriterio (MCDM) y problemas de optimización multiobjetivo (MOOP) (Wu et al., 2019).

De acuerdo con la literatura académica, los métodos tradicionales de optimización de carteras están empezando a quedarse obsoletos, lo que ha llevado al surgimiento de métodos que combinan el enfoque de aprendizaje automático con la optimización de carteras (Metawa, N., 2019). “La optimización bayesiana es una metodología para optimizar funciones objetivo complicadas que ha demostrado su éxito en las ciencias, la ingeniería y otros campos” (Garnett, R., 2023)., ya que permite abordar la complejidad e incertidumbre inherentes a la selección de activos. En 2011, investigadores de la Universidad de British Columbia en Vancouver, Canadá, publicaron un artículo seminal que propone estrategias de gestión de carteras de funciones de adquisición en la optimización bayesiana (Hoffman, M., et al., 2011). Demostraron que la combinación adaptativa de múltiples funciones de adquisición mejora consistentemente el rendimiento sobre las estrategias individuales en una variedad de escenarios de optimización. En el estudio, diseñaron diversas estrategias de "hedge" basadas en la optimización bayesiana y emplearon experimentos en entornos simulados y en problemas de optimización estándar para evaluar su desempeño. Los investigadores llegaron a la conclusión de que gestionar un conjunto de funciones de adquisición en lugar de confiar en una sola, muestran consistentemente un mejor rendimiento que las estrategias individuales de adquisición en una variedad de escenarios de optimización. El estudio demuestra que combinar adaptativamente múltiples funciones de adquisición puede mejorar eficazmente la optimización bayesiana, lo cual es relevante para mi investigación puesto que destaca la aplicabilidad de los métodos bayesianos en diversos contextos.

Un artículo más reciente también analiza el uso de la optimización bayesiana para introducir mejoras en la gestión de carteras, sin embargo, emplea un enfoque diferente al que se va a utilizar en esta investigación (Vasconcelos, T. de P., et al., 2021). En su estudio, los autores emplean un método denominado Self-Tuning Portfolio-based Bayesian Optimization (SETUP-BO), el cual se enfoca en mejorar las estrategias de cartera existentes en la optimización bayesiana, como GP-Hedge y No-PASt-BO,

mediante la incorporación de ajustes automáticos de hiperparámetros. Aunque el método propuesto en el artículo ofrece ventajas significativas al mejorar la eficiencia de la gestión de carteras en la optimización bayesiana mediante ajustes automáticos de hiperparámetros, el uso de las cajas negras, permiten una mayor flexibilidad al permitir que el investigador ajuste manualmente los hiperparámetros según las necesidades específicas del problema. Además, las cajas negras pueden ser más adecuadas en casos donde la interpretación y explicación de los resultados son críticas, ya que proporcionan una visión más transparente del proceso de optimización y permiten un mayor control sobre el modelo y sus parámetros.

Aunque se han realizado diversos estudios sobre métodos de optimización de carteras, son escasas las investigaciones que abordan el uso de la optimización bayesiana en el DRL. Esta investigación busca dar más luz sobre la utilidad de integrar ambos métodos.

3. Alcance

3.1. Hipótesis

A través de esta investigación se buscará proporcionar evidencia empírica que respalde la superioridad del DRL frente a los resultados obtenidos mediante búsqueda aleatoria en la optimización de carteras financieras ESG. Para ello voy a comprobar metodológicamente la eficacia de la optimización bayesiana multiobjetivo en comparación con la búsqueda aleatoria en la gestión de carteras financieras.

H0: no habrá una mejora significativa en la eficiencia del proceso de optimización al utilizar optimización bayesiana multiobjetivo en lugar de la búsqueda aleatoria tradicional.
H1: el uso de optimizació resultará en una mejora significativa, lo que se traducirá en carteras financieras más eficientes y estables en términos de rendimiento y riesgo.

3.2. Objetivos

O1: Redacción de una memoria completa donde documente de manera detallada todo el proceso de investigación y los resultados obtenidos.
O2: Implementación práctica de los algoritmos de optimización de carteras financieras utilizando DRL, optimización bayesiana y búsqueda aleatoria. Llevaré a cabo una prueba de concepto del código desarrollado, ejecutando los algoritmos en un entorno de prueba y analizando los resultados obtenidos.
O3: Comprensión del modelo, interpretación de los resultados obtenidos y extracción de conclusiones.

3.3. Asunciones

A1: Asumo que el enfoque temporal utilizado en este trabajo es escalable y aplicable a diferentes horizontes temporales. Esta premisa implica que el método de optimización de carteras puede ser adaptado y ajustado según las necesidades específicas de diferentes marcos temporales.
A2: Se parte del supuesto de que los resultados obtenidos al utilizar datos del índice Dow Jones Industrial Average pueden generalizarse y aplicarse con éxito a otros índices

bursátiles, es decir, que las tendencias y patrones observados en el mercado financiero global a través del Dow Jones Industrial Average son representativos y pueden extrapolarse de manera efectiva a otros índices financieros.

3.4.Restricciones

R1: Este tipo de investigación requiere de más tiempo para llevar a cabo un mayor número de experimentos y un análisis en mayor profundidad.

R2: Los recursos disponibles para llevar a cabo la investigación han sido limitados. Se requiere de un mayor presupuesto para adquirir datos de calidad, acceder a herramientas y software especializado.

R3: Se requiere de un ordenador con mayor potencia computacional para llevar a cabo esta investigación.

4. Marco Teórico

El objetivo de este trabajo es proporcionar evidencia sobre la eficacia del DRL y la optimización bayesiana para la gestión de carteras. Por un lado, el DRL es un método que implica el funcionamiento de unas entidades computacionales, los agentes inteligentes, que aprenden a tomar decisiones optimizadas mediante la interacción con el entorno y la retroalimentación recibida en forma de recompensas o penalizaciones (Naeem, M. et al., 2020). Por otro lado, la optimización bayesiana es una metodología que se basa en la teoría de la probabilidad y la estadística. Mediante un enfoque sistémico, explora el espacio de posibles soluciones considerando una función objetivo específica (Garnett, R., 2023).

4.1.DRL

El DRL es una disciplina de la IA resultado de la combinación de dos enfoques: el aprendizaje por refuerzo (RL) y el aprendizaje profundo (DL). Por un lado, el RL aprende a tomar decisiones secuenciales y maximizar recompensas a largo plazo en un entorno interactivo. Los agentes aprenden explorando diferentes acciones y recibiendo retroalimentación en forma de recompensas o penalizaciones, es decir, aprenden a través de la experiencia (Amazon Web Services, s.f.). Por otro lado, el DL emplea redes neuronales que le permiten manejar grandes volúmenes de datos y aprender características abstractas y complejas de forma automática (LeCun, Y. et al., 2015). La combinación de estos enfoques permite aprovechar las fortalezas de ambos para mejorar el rendimiento de los agentes inteligentes en entornos complejos como son los mercados financieros, donde hay un alto grado de incertidumbre y hay muchas variables desconocidas y que son difíciles de explicar (Singh, V. et al., 2022).

El DL facilita decisiones más informadas y mejora la capacidad de generalización del agente en entornos desconocido, y el RL guía al agente en la búsqueda de estrategias óptimas mediante la exploración de nuevas acciones y la explotación del conocimiento adquirido. En el caso de la optimización de carteras, el agente inteligente aprende a tomar decisiones óptimas a través de la interacción con el mercado financiero, recibiendo recompensas o penalizaciones en función de las acciones que maximizan la rentabilidad

y minimizan el riesgo de la cartera (Yu, P. et al., 2019). La política óptima de inversión se determina mediante la maximización de la función de recompensa para cada situación del mercado. La política tomada puede adaptarse a diferentes horizontes temporales y preferencias de inversión del usuario.

En este trabajo, en concreto se ha empleado el algoritmo de DRL: Proximal Policy Optimization (PPO). A continuación, se explicará brevemente su función y cómo opera.

4.1.1 PPO

El PPO, u Optimización de Política Próxima, "es un algoritmo de RL que optimiza directamente la función política. Pertenece a la familia de los métodos de gradiente de política y es conocido por su estabilidad y fiabilidad. PPO utiliza un enfoque de optimización de regiones de confianza para actualizar la política con el fin de garantizar cambios graduales, evitando grandes actualizaciones de la política, que pueden provocar inestabilidad" (Corecco, S., et al., 2023). Por esto mismo, es un algoritmo que entrena al agente de manera que maximice la recompensan en un entorno establecido, asegurándose que las actualizaciones de las políticas son proximales y no se alejan, más allá de lo determinado, de la política actual para evitar la inestabilidad.

El siguiente gráfico muestra el funcionamiento del algoritmo. Para ello se emplea una función (LCLIP) que limita la magnitud de las modificaciones de la política. Si la acción se vuelve más probable tras el último paso de gradiente, esto podría significar que la política está mejorando (Théberge, A., 2019). No obstante, el PPO limita este cambio para evitar que la política se desvíe demasiado, lo que podría empeorar el desempeño. Si la probabilidad de la acción disminuye, el PPO permite revertir el cambio. En definitiva, si la política está cambiando en la dirección incorrecta, se pueden realizar ajustes significativos para corregir estos errores sin restricciones severas. Lo mismo sucede a la inversa. Si la probabilidad de la acción disminuye, el PPO de nuevo limita este cambio para prevenir que se haga demasiado improbable, pudiendo provocar fallos en la política. Si la acción se vuelve más probable, lo cual es contraproducente, el PPO no pone restricciones significativas, permitiendo corregir estos errores.

Figura 1: Funcionamiento del algoritmo PPO.

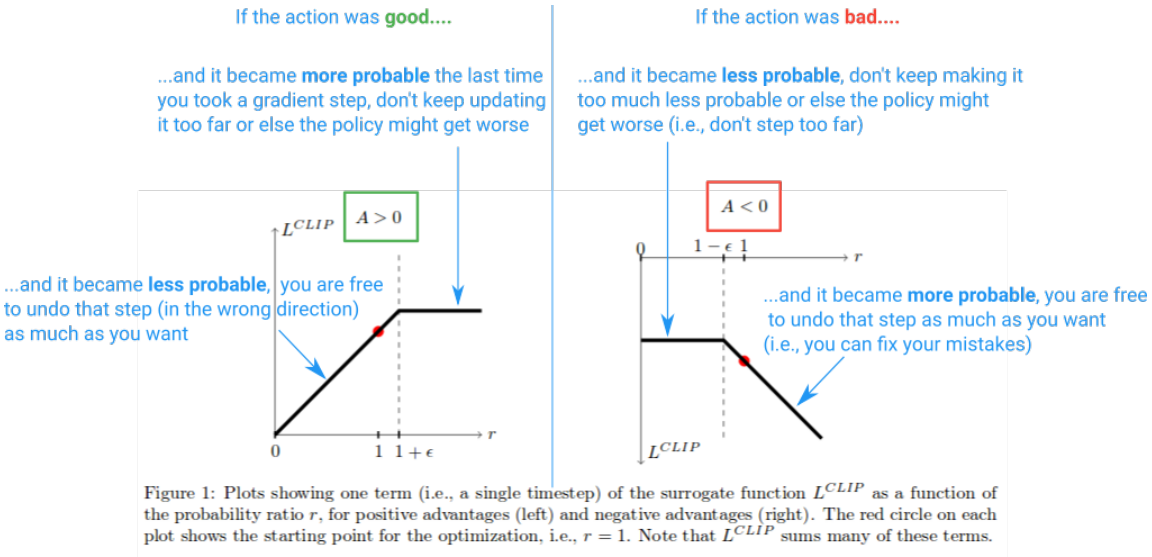
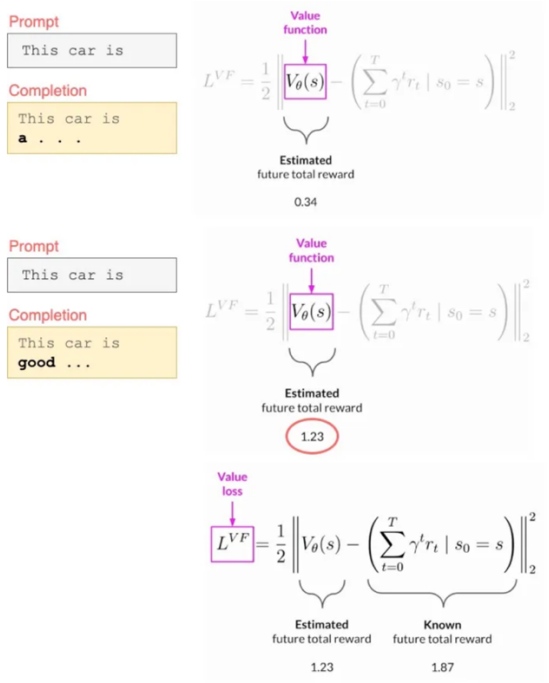


Figure 1: Plots showing one term (i.e., a single timestep) of the surrogate function L^{CLIP} as a function of the probability ratio r, for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization, i.e., $r = 1$. Note that L^{CLIP} sums many of these terms.

Fuente: Théberge, A. (2019).

En la gestión de carteras, el PPO guía gradualmente el proceso de mejora de las estrategias de optimización, mediante la iteración de varios ciclos. Cada ciclo tiene dos fases, la fase de exploración y la fase de explotación. En la primera fase, el algoritmo genera carteras potenciales y evalúa su calidad mediante el uso de una función de valor, que estima la recompensa esperada para cada cartera en función del ratio de Sharpe y la puntuación ESG. La función juzga la calidad de las carteras que se han generado en comparación con los objetivos de optimización establecidos. La finalidad de esta función consiste en minimizar la pérdida de valor, es decir, la diferencia entre la recompensa futura real y la aproximación de la función de valor, consiguiendo mejorar así las predicciones de recompensas futuras.

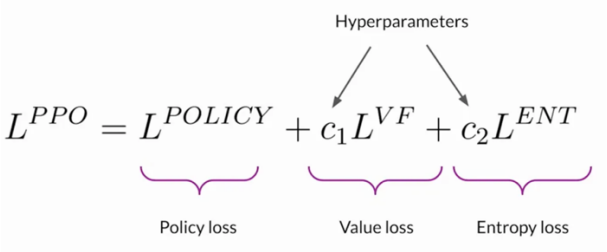
Figura 2: Función de pérdida de valor del PPO.



Fuente: Latypov, O. (2023)

Tras la fase de exploración, el algoritmo se basa en la retroalimentación recibida de la primera fase para realizar pequeños ajustes en las políticas de generación de carteras, ajustando así gradualmente el modelo. el algoritmo PPO también tiene en cuenta la entropía para preservar la diversidad y explorar diferentes posibilidades de carteras, lo que asegura que el algoritmo no se estanque en óptimos locales y tenga la capacidad de encontrar soluciones diversas y óptimas.

Figura 3: Función objetivo del PPO.



Fuente: Latypov, O. (2023).

4.2.Optimización Bayesiana

En el aprendizaje automático es preciso, a parte de los parámetros del modelo que se ajustan a medida que se entrena, el ajustar los hiperparámetros. “Los hiperparámetros son parámetros cuyos valores controlan el proceso de aprendizaje y determinan los valores de los parámetros del modelo que un algoritmo de aprendizaje acaba aprendiendo. El prefijo 'hyper' implica que son parámetros de 'alto nivel' que controlan el proceso de aprendizaje y los parámetros del modelo que resultan de él” (Nyuytiymbiy, K., 2020). Para mejorar obtener los datos con mejores rendimientos, se lleva a cabo un proceso antes de la fase de entrenamiento conocida como ajuste de hiperparámetros (Wu, J., et al., 2019). Existen numerosas técnicas para encontrar la mejor combinación de estos parámetros, desde la búsqueda manual hasta la búsqueda por cuadrícula (grid search), sin embargo, estas técnicas impiden el manejo de grandes dimensiones de datos. Para resolver este problema, se propuso el algoritmo de búsqueda aleatoria (J. Bergstra, et al., 2012), método mediante el cual se descubrió que solo algunos hiperparámetros son importantes para la mayoría de los conjuntos de datos, dando lugar a una mayor eficiencia. Al probar combinaciones aleatorias dentro de un rango de valores, mejora la eficacia en los espacios de alta dimensión (Wu, J., et al., 2019). Sin embargo, no es confiable para modelos complejos debido a que no cubre adecuadamente el espacio de hiperparámetros y, por tanto, puede no capturar algunas de las interacciones entre hiperparámetros. Esta limitación puede resultar en la omisión de combinaciones críticas de hiperparámetros y afectar la capacidad del modelo para alcanzar la configuración óptima (Turner, R., 2021).

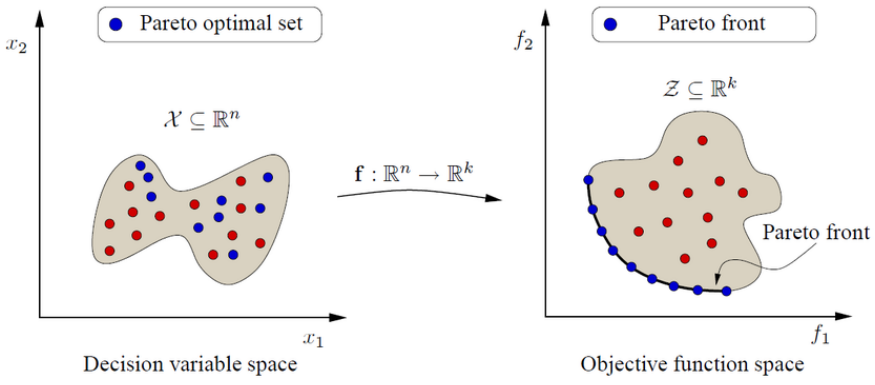
Ajustar hiperparámetros en modelos complejos presenta desafíos significativos debido a la naturaleza de "caja negra" de la función objetivo (Garrido Merchán, E. C., et al., 2020). La complejidad de esta función impide la aplicación directa de métodos tradicionales de optimización. La optimización bayesiana resuelve este problema, ya que puede gestionar de manera efectiva la incertidumbre y la variabilidad en el espacio de hiperparámetros, adaptándose iterativamente conforme se acumula más información. Este método permite encontrar la combinación óptima de pesos de la cartera en función de objetivos, como el ratio de Sharpe y la calificación ESG. Lo hace mediante la construcción de un modelo probabilístico del espacio con todas las posibles carteras. La función objetivo es una caja negra, donde se conocen las entradas y salidas, pero el funcionamiento interno es desconocido, permitiendo hacer así una exploración eficiente del espacio de búsqueda. Además, la función objetivo puede incluir ruido puesto que no

se modela un valor único, sino que se modela una distribución de probabilidad del valor. Esto es muy importante en la gestión de carteras debido a que es un entorno en el que las condiciones son cambiantes y muy volátiles. Debido a que en este trabajo se consideran múltiples objetivos, la optimización bayesiana debe formular y optimizar múltiples funciones objetivo, cada una representando un objetivo diferente.

4.2.1 Conjunto de Pareto en la Optimización Multiobjetivo

A través de la construcción de modelos probabilísticos para cada función objetivo, la optimización bayesiana explora el espacio de búsqueda para encontrar soluciones que representen un equilibrio óptimo entre estos objetivos. El conjunto de soluciones óptimas se conoce como conjunto de Pareto. Como señala la literatura, las soluciones de Pareto representan un equilibrio óptimo entre estos objetivos, de tal manera que no es posible mejorar uno de ellos sin empeorar al menos uno de los otros (Lotov, A.V., 2008). En otras palabras, las soluciones de Pareto son aquellas en las que se alcanza un óptimo trade-off: mejorar un objetivo implica inevitablemente sacrificar otro. La optimización bayesiana multiobjetivo es útil puesto que permite seleccionar y encontrar la combinación óptima de objetivos relevantes. Esto proporciona flexibilidad al inversor, al permitirle seleccionar la solución que mejor se adapte a sus necesidades y objetivos individuales.

Figura 4: Representación gráfica del conjunto de Pareto.



Fuente: Gomes, G. F., et al. (2019).

La optimización bayesiana emplea procesos gaussianos se utilizan para modelar funciones objetivo, donde se busca encontrar el conjunto óptimo de hiperparámetros. En este tipo de optimización, las pequeñas modificaciones causan cambios leves en el error. Esta suposición de que el error varía suavemente con respecto a los hiperparámetros se observa en la literatura, aunque no se ha confirmado específicamente en el contexto del DRL (Garrido Merchán, E. C., et al., 2020). En DRL, los resultados son altamente variables, los procesos pueden ser lentos e inestables, y a veces no funcionan correctamente, especialmente en escenarios con mucho ruidosas como en las finanzas. Es un método novedoso y complejo y por ello, se está empezando a implementar en el sector financiero, sin embargo, se han realizado experimentos en otras áreas como la robótica o en Amazon, y se ha visto que la optimización bayesiana puede ser efectiva en estos casos (Perrone, V., et al., 2021).

4.3. Integración de ambos enfoques: DRL y optimización bayesiana

Ambos métodos aportan beneficios al objetivo de este trabajo. La optimización bayesiana permite una modelización precisa del problema mientras que el DRL ofrece una capacidad de adaptación y aprendizaje continuo del mercado. El proceso de funcionamiento es el siguiente. En primer lugar, el inversor elige los valores iniciales para los parámetros de la cartera en función de la estrategia de inversión que se quiera conseguir. Un agente de aprendizaje profundo interactúa con el entorno del mercado, tomando decisiones sobre la asignación de activos según su política óptima aprendida. El agente va recibiendo retroalimentación en forma de recompensas o penalizaciones. Por último, el modelo bayesiano estima la relación entre las configuraciones de la cartera y la función objetivo. Para ello, el modelo selecciona varias configuraciones de cartera para evaluar teniendo en cuenta el rendimiento óptimo. Este proceso de repite reiteradamente, a medida que se va ejecutando, el agente va ajustando su política basándose en la evaluación recibida y la optimización bayesiana refina su modelo con cada evaluación. El proceso se detiene cuando se alcanza un criterio de convergencia predefinido, la mejor configuración de la cartera se selecciona como la solución óptima. Esta combinación permite que el aprendizaje profundo por refuerzo guíe la toma de decisiones en tiempo real.

5. Descripción del modelo

Para lograr el objetivo de mi trabajo, he creado un modelo en Python que ilustra la resolución de un problema multiobjetivo de gestión de carteras. Este modelo emplea una combinación de optimización bayesiana multiobjetivo y aprendizaje por refuerzo profundo (DRL). Su propósito es optimizar varios indicadores financieros basados en la política financiera determinada por un agente entrenado con un algoritmo de DRL. El objetivo principal es demostrar cómo resolver un problema multiobjetivo de gestión de carteras mediante la optimización bayesiana multiobjetivo de múltiples indicadores financieros derivados de la política financiera estimada por un agente entrenado con DRL. Además, busco determinar si hay evidencia suficiente para afirmar que un modelo de optimización bayesiana multiobjetivo ofrece mejores resultados que un modelo de optimización bayesiana interpolada de objetivo único o de búsqueda aleatoria.

Trabajaré con el algoritmo de DRL, el PPO, para la realización de operaciones en el mercado bursátil. El PPO se utilizará para entrenar un agente que tomará decisiones de inversión en el mercado bursátil, es decir, buscará optimizar los hiperparámetros específicos para maximizar el rendimiento financiero mientras minimiza el riesgo asociado a las operaciones.

Cuando se entrena un modelo de aprendizaje automático, cada conjunto de datos y modelo requiere un conjunto único de hiperparámetros. Los hiperparámetros son "variables de configuración externa que los científicos de datos utilizan para administrar el entrenamiento de modelos de machine learning" (Amazon Web Services, s.f.). La única manera de determinar los hiperparámetros adecuados es llevando a cabo múltiples experimentos, donde se selecciona un conjunto de hiperparámetros y se ejecuta a través del modelo. Básicamente, implica entrenar su modelo de manera secuencial con diferentes conjuntos de hiperparámetros. Este proceso puede realizarse manualmente o utilizando diversos métodos automatizados de ajuste de hiperparámetros. Mediante la optimización bayesiana multiobjetivo, el modelo busca devolver la combinación óptima de hiperparámetros para los algoritmos de DRL.

El modelo emplea un conjunto de datos que va desde 2008 hasta 2022, y para comprobar el desempeño, se usan datos del año 2023. En el modelo comprueba cómo funciona la optimización bayesiana multiobjetivo con respecto a otros métodos, como la

optimización bayesiana de un solo objetivo y la búsqueda aleatoria. Para hacer esto, realizaremos varias iteraciones de experimentos, cada una con 10 ejecuciones para asegurarnos de obtener resultados confiables. Un factor importante para tener en cuenta es que la optimización multiobjetivo, como se señala, produce un conjunto de soluciones óptimas conocido como el conjunto de Pareto, que satisfacen los objetivos conflictivos. Para realizar una comparación justa, es esencial seleccionar una solución representativa de este conjunto para contrastarla con la mejor solución encontrada por métodos de optimización de objetivo único.

Para simplificar el análisis inicial y centrarse en comprender cómo afectan los hiperparámetros a estos objetivos conflictivos de rendimiento y riesgo, se elimina el objetivo ESG (Environmental, Social, and Governance) en la primera ejecución. Esto reduce la complejidad del problema al centrarse únicamente en la estabilidad o volatilidad anual como medida de riesgo, lo que facilita la comprensión y el análisis de los resultados. Una vez que se ha establecido una comprensión sólida de cómo afectan los hiperparámetros a los objetivos de rendimiento y riesgo principales, se reintroduce el objetivo ESG en ejecuciones posteriores para realizar un análisis más completo y holístico.

El modelo se trabaja en el marco de trabajo, FinRL, que se basa en la biblioteca Stable Baselines3. Es un marco diseñado específicamente para aplicaciones financieras, como el trading de acciones. Proporciona una serie de herramientas y utilidades que facilitan la implementación y evaluación de algoritmos de DRL en entornos financieros. Además, se hará uso del paquete de Python, Dragonfly, se utilizará para llevar a cabo la optimización bayesiana multiobjetivo, lo que implica optimizar múltiples objetivos de manera simultánea. Este paquete proporciona herramientas para manejar objetivos conflictivos y puede ser útil en problemas donde se busca un equilibrio entre diferentes criterios de rendimiento o riesgo.

El objetivo principal es evaluar y comparar el rendimiento de tres métodos diferentes: la optimización bayesiana multiobjetivo, la optimización bayesiana de objetivo único interpolada y random search. El experimento se llevará a cabo a lo largo de 50 iteraciones para refinar los hiperparámetros de cada método. Se ha decidido emplear un número relativamente bajo de pasos de tiempo para hacer los experimentos más factibles en términos de tiempo de ejecución. Se sugiere la posibilidad de mejorar la eficiencia del

experimento utilizando GPUs si se busca una mayor precisión en los resultados. Cada método será ejecutado en 10 ocasiones diferentes para tener en cuenta el ruido inherente en las variables objetivo y así obtener una evaluación más confiable del rendimiento de cada uno.

Como se menciona en la sección de alcance del trabajo, una de las restricciones es la falta de potencia computacional. Lo ideal es poder incorporar algoritmos genéticos, explorar más índices financieros, como NASDAQ, y evaluar una variedad más amplia de algoritmos de aprendizaje por refuerzo profundo, como A3C. El modelo que se presenta en este trabajo sirve como base para poder profundizar en esta investigación en el futuro.

La ampliación de los experimentos permitirá una evaluación más completa y detallada de los métodos de optimización en una variedad de escenarios y aplicaciones. Al llevar a cabo estos experimentos adicionales, se espera obtener una comprensión más profunda de la eficacia y las limitaciones de cada método, lo que enriquecerá significativamente los hallazgos y conclusiones de este estudio.

El contenido del modelo se organiza en 7 secciones, que son las siguientes:

1. Definición del problema
2. Paquetes de Python
    - *2.1 Instalación de Paquetes*
    - *2.2 Verificación de Paquetes Adicionales*
    - *2.3 Importación de Paquetes*
    - *2.4 Creación de Carpetas*
3. Descarga de los datos que se van a emplear en el modelo
4. Preprocesar Datos
    - *4.1 Indicadores Técnicos*
    - *4.2 Realización de Ingeniería de Características*
5. Construcción del entorno
    - *5.1 División de Datos de Entrenamiento*
    - *5.2 Definición de Entorno*
    - *5.3 Inicialización de Entorno*
6. Implementación de los algoritmos de DRL
7. Ejecución del experimento

A continuación, se desarrollarán en mayor detalle cada una de las secciones.

5.1.Definición del problema

El problema que se aborda en este trabajo consiste en diseñar una solución de trading automatizado para la asignación de carteras. Para ello, modelamos el proceso de trading de acciones como un Proceso de Decisión de Markov (MDP), un marco conceptual utilizado para modelar el proceso de toma de decisiones secuenciales en un entorno estocástico. El MDP describe un sistema en el que un agente toma decisiones en un entorno en función de su estado actual y recibe recompensas o penalizaciones como resultado de esas decisiones.

Se formula el objetivo de trading como un problema de maximización para buscar la asignación óptima de activos que maximice métricas de rendimiento, es decir, de Sharpe, lo que permite utilizar técnicas de optimización para identificar la cartera que genere el mejor rendimiento posible bajo ciertas condiciones y restricciones del mercado.

El código también describe los componentes del entorno de aprendizaje por refuerzo, que son:

- Acción: conjunto de acciones permitidas que el agente puede tomar en el entorno. Una acción típica puede ser la asignación de pesos a diferentes activos en la cartera ($a \in A$: $a \in (-1,1)$).
- Función de recompensa: mecanismo de incentivo que guía al agente hacia la toma de decisiones que maximizan ciertas métricas de rendimiento. En este caso, la función evalúa el cambio de valor de la cartera cuando se toma una determinada acción en un estado dado y se alcanza un estado nuevo.

  Figura 5: Función de recompensa del RL.

  $$r(s, a, s') = v' - v$$

  - Donde $r(s, a, s')$ es la función de recompensa y, donde v' y v representan los valores de la cartera en el estado s' y s, respectivamente.
- Estado: describe las observaciones que el agente recibe del entorno, como las diversas características del estado, los precios históricos, volúmenes de transacción, y otros indicadores financieros relevantes. Esta información se emplea para guiar las decisiones de inversión del agente de trading.

- Entorno: conjunto de acciones y estados que definen el mercado en el que opera el agente. En este modelo se compone de las acciones que integran el índice Dow 30, el cual está compuesto por un conjunto selecto de acciones representativas del mercado bursátil estadounidense.

Los datos de las acciones individuales que utilizaremos en este estudio se obtienen a través de la API de Yahoo Finance. Estos datos incluyen precios de apertura, cierre, máximo, mínimo y volumen de transacciones, que son fundamentales para el análisis y la toma de decisiones en el trading algorítmico.

5.2. Paquetes de Python

En esta sección del modelo se descargan, verifican e importan los paquetes necesarios para la ejecución del código. Para ello, se obtienen las definiciones y su funcionamiento de PyPI (Python Package Index., s.f.). Estos son:

- Paquetes de la librería FinRL: permiten descargar datos financieros, realizar el preprocesamiento de estos, construir entornos de simulación, implementar algoritmos de aprendizaje profundo por refuerzo (ej.: PPO), evaluar estrategias de trading y visualizar los resultados del backtesting. Ofrecen una infraestructura completa para el desarrollo y análisis de estrategias de trading automatizado.
- Yahoo Finance API: proporciona acceso a datos financieros de las acciones.
- Pandas: biblioteca que proporciona herramientas de análisis de datos, necesarios en este modelo para la manipulación de los datos financieros.
- Numpy: biblioteca que proporciona el soporte para las matrices y operaciones matemáticas de gran complejidad.
- Matplotlib: biblioteca de visualización de datos en Python que permite crear gráficos.
- Stockstats: extensión de pandas que proporciona estadísticas financieras y técnicas de análisis para datos de series temporales financieras.
- OpenAI gym: proporciona un entorno estandarizado para desarrollar y evaluar algoritmos de aprendizaje por refuerzo, facilitando la creación de agentes de trading.

- stable-baselines: biblioteca que implementa algoritmos de aprendizaje por refuerzo en Python, específicamente adaptados para aplicaciones en finanzas.
- Tensorflow: biblioteca desarrollada por Google, que proporciona herramientas para construir y entrenar modelos de aprendizaje profundo, incluidos los utilizados en aplicaciones de trading algorítmico.
- Pyfolio: biblioteca que se usa en la evaluación de carteras de inversión y análisis de estrategias de trading puesto que ofrece herramientas de medición del rendimiento, análisis de los riesgos y es capaz de realizar el análisis de atribución de rendimiento.

5.3.Descarga de datos

En esta sección del modelo, se obtienen los datos del mercado bursátil desde Yahoo Finance. La biblioteca FinRL utiliza una herramienta llamada YahooDownloader para interactuar con la API de Yahoo Finance y acceder a los datos necesarios. También se incorporan los datos de ESG, extraídos de Bloomberg.

5.4.Reprocesar datos

En esta parte del código, se realiza el preprocesamiento de los datos a partir de la función de Python FeatureEngineer. Es una etapa muy importante, puesto que es la fase en la que se preparan, limpian y transforman los datos originales en un formato adecuado para su análisis o para alimentar un modelo de aprendizaje automático. En esta fase se llevan a cabo dos acciones principales. Por un lado, se añaden indicadores técnicos. En la práctica, a la hora de hacer trading de acciones se tienen en cuenta varios factores: los precios históricos, las acciones que están en posesión, y los indicadores técnicos. Los indicadores técnicos son herramientas utilizadas por los traders y analistas financieros para analizar y pronosticar movimientos futuros en los precios de los activos financieros (Chen, J., 2021). En este caso, se emplean dos indicadores técnicos de seguimiento de tendencias: RSI y MACD, que ayudan a identificar oportunidades de compra o venta en el mercado. Mientras que el RSI permite identificar si un activo financiero está siendo sobrecomprado o sobrevendido, el MACD detecta cambios en la dirección de la tendencia

y posibles puntos de entrada y salida, es decir, momentos en los que un operador decide abrir o cerrar una posición (Maverick, J. B., 2024).

En entornos financieros, se incorpora el índice de turbulencia como un marcador que se activa en eventos inesperados o con alta volatilidad, tal como conflictos geopolíticos o crisis sanitarias como la guerra en Ucrania o la pandemia de COVID-19. En tales escenarios, un agente previamente entrenado enfrenta un dilema al no haber sido expuesto a estos datos, lo que conlleva a que tome decisiones de forma aleatoria. El propósito del índice de turbulencia radica en interrumpir el comportamiento habitual del agente en estas situaciones críticas, permitiéndole ejecutar maniobras defensivas como la venta completa de activos, la búsqueda de estabilidad con mínima variabilidad o la diversificación de la cartera como medida de precaución.

Dentro del código, este índice se encuentra en estado "FALSE", es decir, desactivado, dado que se emplea en contextos reales. En este estudio, se lleva a cabo una simulación que busca respaldar de forma empírica la superioridad del DRL frente a los resultados obtenidos mediante estrategias de búsqueda aleatoria en la optimización de carteras financieras centradas en criterios ESG.

Una vez se procesan los datos, se ordenan los datos por fecha y por símbolo y se crean dos listas vacías (cov_list, return_list), que se emplearán para almacenar las matrices de covarianza y rendimientos respectivamente. Se itera sobre las filas del DataFrame creado en el paso anterior después de un período de retroceso de un año (lookback). Observar los datos retrospectivamente durante un período de tiempo, permite capturar las tendencias y patrones históricos que pueden influir en los precios de los activos financieros. Además, utilizar datos históricos permite que el modelo tenga una visión más amplia y establezca relaciones más sólidas entre las variables. Esto puede mejorar la capacidad del modelo para predecir el comportamiento futuro.

La matriz de covarianza se calcula a partir de los rendimientos históricos de las acciones y describe cómo se han relacionado entre sí durante el período de tiempo. Esta matriz informa sobre la relación y la variabilidad conjunta de los rendimientos de los diferentes activos, lo que ayuda a los inversores a comprender la diversificación, el riesgo y la estructura de correlación de sus carteras, lo que es fundamental para la gestión de riesgos y la construcción de carteras eficientes. Una vez calculada, se agrega la matriz de

covarianza de los rendimientos a cov_list y los rendimientos, a return_list. Por último, se crea un nuevo DataFrame llamado df_cov para almacenar las covarianzas calculadas y las fechas correspondientes. Este DataFrame se fusiona con el DataFrame original df, agregando las matrices de covarianza como nuevas columnas. Finalmente, se reordena el DataFrame resultante por fecha y símbolo, y se restablece el índice para garantizar un orden adecuado y eliminar los índices anteriores.

5.5.Construcción del entorno

En esta fase del modelo, se diseña el entorno de simulación para el trading automatizado de acciones. Este entorno de simulación se construye sobre el marco de OpenAI Gym, que emula las condiciones del mercado bursátil y proporciona la plataforma sobre la cual se entrena y evalúa los algoritmos de trading, para desarrollar y validar las estrategias de trading. Dado que las tareas financieras son inherentemente impredecibles y requieren interacción continua, optamos por modelar el problema como un MDP. Durante el proceso de entrenamiento, el agente observa la fluctuación de los precios de las acciones, realiza acciones (como comprar, vender o mantener) y evalúa las recompensas asociadas con esas acciones. Este proceso continuo permite que el agente ajuste y perfeccione su estrategia de trading a medida que interactúa con el entorno. La meta es desarrollar una estrategia que maximice las recompensas a lo largo del tiempo, lo que implica una adaptación continua a las condiciones cambiantes del mercado.

En esta sección se emplean los paquetes de (Python Package Index, s.f.):

- Numpy y panda para el análisis de datos. Numpy se emplea para trabajar con operaciones matemáticas eficientes en matrices y arrays multidimensionales, mientras que Pandas se especializa en la manipulación y análisis de datos tabulares, en este caso, los precios de las acciones.
- Gym es una biblioteca de código abierto en Python, es decir, un conjunto de código que está disponible públicamente y permite a los usuarios acceder, modificar y redistribuir el código fuente de un software de manera libre y abierta. La biblioteca Gym ofrece un marco de trabajo sobre el que se crea y desarrolla el entorno que simula el trading automatizado de acciones. Gym incluye varios

módulos, entre ellos gym.utils y gym.spaces, que ofrecen herramientas y permiten definir espacios de observación y acción para los agentes de DRL.

- Matplotlib y pyplot para la creación y visualización de gráficos.
- Stable_baselines3

La construcción del modelo consta de dos fases. En la primera fase, denominada etapa de entrenamiento, se toma una porción de datos que abarca desde el 1 de enero de 2008 hasta el 31 de diciembre de 2022. La segunda fase corresponde a la etapa de test o validación, donde se utiliza la porción restante de los datos, específicamente el año 2023. Durante la etapa de entrenamiento, el conjunto de datos se emplea para ajustar los parámetros del modelo, permitiendo que los agentes del algoritmo aprendan las políticas óptimas. Una vez completado el entrenamiento, se emplea la porción restante de datos, conocida como el conjunto de validación, para validar que las políticas se hayan aprendido de forma correcta y que el modelo pueda generalizar efectivamente, es decir, pueda tener un buen desempeño con datos nuevos no vistos anteriormente. Esta fase es muy importante debido a que, si se utilizara el mismo conjunto de datos para el entrenamiento y la evaluación, existe el riesgo de sobreajuste, donde el modelo simplemente memoriza los datos en lugar de aprender patrones generalizados, que es el objetivo fundamental del modelo (scikit learn, s.f.).

También se introduce un diccionario llamado “metadadata” que se emplea para almacenar información sobre cómo mostrar la información en el entorno. En este caso específico, se establece la clave 'render.modes' con el valor ['human'], lo que indica que se está configurando un modo de renderizado para el entorno de forma que la información visualizada sea comprensible y adecuada para un usuario humano, es decir, que la representación sea clara y fácil de entender para el usuario final.

En la sección inicial del código, se definen una serie de funciones y métodos diseñados para ser utilizados en la estructura 'StockPortfolioEnv'. Estas funciones facilitan las acciones básicas como la compra, venta, cálculo de recompensas y seguimiento del desempeño de la cartera, es decir, especifican cómo se comporta y se relaciona el entorno de simulación de trading de acciones, permitiendo gestionar las operaciones esenciales ejecutadas en dicho entorno. A continuación, describiré brevemente cada una de las acciones y su utilidad:

- Función '_init_' es una función de especial relevancia en las fases iniciales de un entorno. Su función es la de preparar las variables y las configuraciones iniciales que se necesitan para simular las operaciones de trading en un entorno controlado. Para ello, emplea indicadores técnicos y otros parámetros que usa como instrucción para tomar decisiones sobre las acciones que debe realizar en cada paso de la simulación. Entre los parámetros que utiliza la función se encuentra 'stock_dim', que especifica el número de acciones que se incluyen en la simulación, o 'lookback', que indica el número de periodos retrospectivos que se considerarán en la toma de decisiones.
- Función 'step' es relevante por varios motivos. Por un lado, es la responsable de desencadenar acciones secuenciales en la simulación de trading, es decir, de guiar el avance y evolución de las inversiones a lo largo del tiempo. También participa en la toma de decisiones, influyendo en como asignar recursos para la inversión o que acciones tomar para ajustar la cartera. Es una función que facilita la evaluación del desempeño de la cartera en cada paso puesto que calcula el rendimiento, actualiza el valor de la cartera y registra métricas como el Sharpe ratio. Por último, es la responsable de la finalización de la simulación, es decir, determina cuándo finaliza la simulación.
- Función 'reset' se emplea para el restablecimiento de la simulación a su estado original. Cuando se activa esta función se desencadenan una serie de acciones. En primer lugar, se reinician variables clave del entorno como el historial del valor de activos, esto implica que se establece de nuevo el registro del valor de los activos en la cartera al valor inicial. Además, se inicializan estados para la simulación, como las matrices de covarianza y el estado actual basado en indicadores técnicos. El objetivo de esta función es reposicionar la simulación en el punto de partida.
- Función 'render'
- Función 'softmax_normalization' normaliza las acciones de la cartera con el objetivo de equilibrar y ajustar las ponderaciones de las acciones de forma proporcional. Para ello, la función lleva a cabo una serie de acciones. Por un lado, calcula el numerador a través de la exponenciación de los valores de las acciones, lo que significa que las acciones con valores más altos tendrán una influencia mayor en la distribución final de ponderaciones. Más tarde, obtiene el

denominador mediante la suma de las exponenciales de todos los valores de las acciones, lo cual crea un factor de normalización esencial para asegurar que la distribución resultante esté dentro de un rango válido. Finalmente, la salida de la función softmax se genera al dividir el numerador por el denominador, garantizando que las acciones estén ponderadas adecuadamente y en proporción unas con otras.

- Función 'save_asset_memory' documenta y guarda el historial de rendimiento de la cartera en un Dataframe estructurado en dos columnas: en una de ellas guarda las fechas y en la otra, los rendimientos diarios adjudicados a dicha fecha.
- Función 'save_action_memory' documenta y almacena el historial de las acciones tomadas en la cartera por fecha.
- Función '_seed' establece la semilla aleatoria en el modelo para garantizar que la secuencia de números aleatorios sea consistente en cada ejecución. Esto asegura que el modelo aprenda a generalizar patrones en lugar de depender de información específica memorizada, lo que promueve la consistencia en su comportamiento y la adaptabilidad en diversas situaciones.
- Función 'get_sb_env' inicializa el entorno de OpenAI

Una vez se han definido las funciones del código, se configura y prepara el entorno de simulación de cartera de acciones. Para ello se empieza calculando la dimensión de la cartera, es decir, la cantidad de acciones que contiene el modelo y se establece el espacio de estados del entorno de simulación, que es equivalente a la dimensión. Posteriormente se configura el entorno de simulación, definiendo los parámetros del entorno e incluyendo elementos como el horizonte temporal máximo, la cantidad inicial de dinero o el coste porcentual por transacción. Por último, se obtiene el entorno de entrenamiento que servirá como un campo de práctica virtual para que un modelo pueda aprender a invertir y mejorar sus estrategias en un ambiente simulado y controlado.

5.6. Implementación de los algoritmos de DRL

En esta sección del código se construyen y compara el rendimiento de tres modelos diferentes: modelo de optimización bayesiana multiobjetivo (MOBO), modelo de optimización bayesiana uni-objetivo interpolada (baseline) y la búsqueda aleatoria.

El modelo de optimización bayesiana con un único objetivo finaliza con una única caja negra que combina dos objetivos conflictivos (Sharpe y Max DD o Sharpe y ESG). Esta situación puede resultar problemática debido a que la solución obtenida puede no ser óptima para cada objetivo de forma independiente. Por lo tanto, se espera que el enfoque de optimización estándar tenga un rendimiento peor en comparación con la optimización bayesiana multiobjetivo (MOBO).

A continuación, se describe el proceso de construcción de los modelos.

5.6.1 Modelo BASELINE

En primer lugar, se comienza construyendo el modelo BASELINE, el cual no implementa directamente la optimización bayesiana; en lugar de ello, emplea una aproximación de "optimización estándar". El modelo tiene como objetivo el generar una única solución óptima, que equilibre los dos objetivos, mediante la interpolación de ambos. Este modelo emplea un enfoque determinista puesto que se buscan soluciones utilizando reglas de optimización predefinidas, sin tener en cuenta la incertidumbre en los parámetros.

Para su construcción, se comienza importando el módulo “timeseries” de la librería “pyfolio”, previamente explicada en la sección “5.2 Paquetes de Python”. Timeseries contiene funciones importantes para el análisis de las series temporales financieras.

Una vez importadas las librerías necesarias, se construye la función de optimización, bajo el nombre “interpolated_financial_portfolio_optimization_baseline” que tomará como entrada el vector “x” que contiene los siguientes parámetros:

- Ent_coef: “(float) Coeficiente de entropía para el cálculo de pérdidas” (Stable Baselines, 2021). La entropía se utiliza para evitar que el modelo se ajuste a óptimos locales. Un mayor valor de este coeficiente implica que el agente debe explorar un mayor número de acciones.
- Learning_rate: hiperparámetro que indica el tamaño de los ajustes que se deben aplicar a los pesos del modelo durante el proceso de entrenamiento, es decir, cuánto se modifican los pesos del modelo en cada iteración del algoritmo de optimización. Un valor de learning_rate más alto implica que se realizan

actualizaciones más significativas en cada iteración, lo que a su vez tiene el riesgo de que el modelo no converja correctamente y no encuentre la solución óptima. Por otro lado, un valor bajo de este hyperparámetro supone actualizaciones menos significativas por cada iteración. Un valor excesivamente bajo hace que el modelo se ejecute de forma más lenta y también tenga dificultad en alcanzar el óptimo global (Brownlee, J., 2019).

- Gamma: hiperparámetro que se emplea como factor de descuento de las recompensas futuras (Sutton, R. S., 2015) y ayuda al modelo a equilibrar la importancia de las recompensas a corto y largo plazo en el proceso de toma de decisiones del agente permitiéndole maximizar el retorno esperado a lo largo de su interacción con el entorno. Un valor más alto de gamma hace que el agente considere recompensas a largo plazo, mientras que un valor más bajo prioriza recompensas inmediatas (Sood, S., 2023).
- Clip_range: parámetro que restringe la magnitud de la actualización de las políticas para mejorar la estabilidad y evitar divergencias durante el entrenamiento en algoritmos como PPO.
- Gae_lambda: es un algoritmo de ventaja generalizada (GAE) que controla la ponderación de las estimaciones de ventajas en el horizonte temporal, corto y largo plazo, durante el entrenamiento de un modelo de aprendizaje por refuerzo (Intellabs, s.f.).
- Timesteps: cantidad de pasos o iteraciones que se utilizaran en la optimización (Pardo, F., et al., 2022).

Estos hiperparámetros influyen en como el modelo toma decisiones para obtener la combinación de factores óptima, por ello considera factores como la actualización de pesos o la importancia de las recompensas futuras, puesto que el objetivo de emplear y ajustar estos hiperparámetros es el de optimizar el algoritmo RL.

La notación "x [0], x [1], …" asignada a cada hiperparámetro indica el valor que se le asigna a "x" y su correspondiente posición "[0]" en el vector. Aunque el valor que se asigna a timesteps es de 10, para realizar un experimento real, debería de ser mucho más alto, en torno a 80.000.

Tras haber importado las librerías necesarias y construido la función de optimización con la definición de los hiperparámetros, se crea un agente de aprendizaje por refuerzo (DRLAgent) y se indica: el entorno de entrenamiento en el que va a trabajar (env_train) y un rango de valores a los hiperparámetros previamente explicados en esta sección. Al proporcionar intervalos de valores para cada hiperparámetro, se amplía el espacio de búsqueda y se mejora la capacidad de encontrar la configuración de hiperparámetros más adecuada para el modelo.

Por último, se construye el modelo de tipo PPO dentro del agente de aprendizaje con los parámetros que se han definido. Se divide el dataset y se selecciona el periodo comprendido entre el 1 de enero de 2023 y el 31 de diciembre del mismo año para entrenar al modelo. También se crea un entorno de gym personalizado (StockPortfolioEnv) para el trading de cartera de acciones del conjunto de entrenamiento, al que se le proporciona el conjunto de datos seleccionados y otras configuraciones específicas del entorno, como parámetros de observación y acción (env_kwargs). Durante el proceso de entrenamiento, el modelo interacciona con el entorno y va aprendiendo, mejorando así su desempeño en la tarea de optimización.

Cuando el modelo ya está creado, se realizan predicciones en el entorno de trading y se almacenan los resultados. Los rendimientos se convierten a un formato compatible con la biblioteca Pyfolio, que emplea una función que calcula estadísticas de rendimiento de la estrategia de trading, entre ellas el Maximum Drawdown y el Sharpe Ratio. Por último, se calcula una métrica que pondera el máximo drawdown multiplicándolo por 30 y suma el coeficiente de Sharpe. De esta forma, se busca tener una métrica equilibrada entre la maximización del retorno y la minimización de las pérdidas extremas, lo que puede ofrecer una medida óptima entre el rendimiento y el riesgo en la evaluación global de la estrategia de trading.

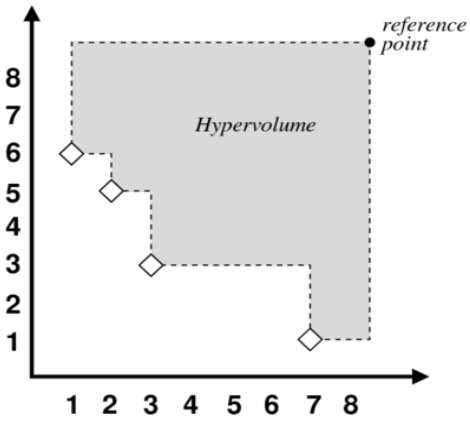
5.6.2 Modelo MOBO

Mientras que el modelo BASELINE calcula una media ponderada de los objetivos y devuelve una única solución, el modelo MOBO considera dos funciones de forma interrelacionada. Cada una de estas funciones está modelada a través de un proceso

gaussiano, lo que permite al modelo MOBO proporcionar un conjunto de soluciones conocidas como el frente de Pareto, que ofrece un equilibrio óptimo de soluciones entre los diferentes objetivos. En contraste, el modelo BASELINE es más limitado ya que, al combinar las funciones, puede comprometer la integridad de los datos, especialmente si los objetivos contienen valores atípicos. Esta limitación puede impactar negativamente en la precisión de la media ponderada resultante.

Para evaluar la calidad de un frente de Pareto se utiliza la métrica del hipervolumen, que es el volumen del espacio objetivo o área que encierra el frente de Pareto con respecto a un punto de referencia en el espacio de valores. Estos valores son los objetivos múltiples del problema de optimización, en el caso de este trabajo serían el espacio formado por el ratio de sharpe y el maximum drawdown (o ESG). Para una mayor comprensión, a continuación, se muestra gráficamente el frente de Pareto y el área que representa el hipervolumen:

Figura 6: Representación gráfica del hipervolumen.

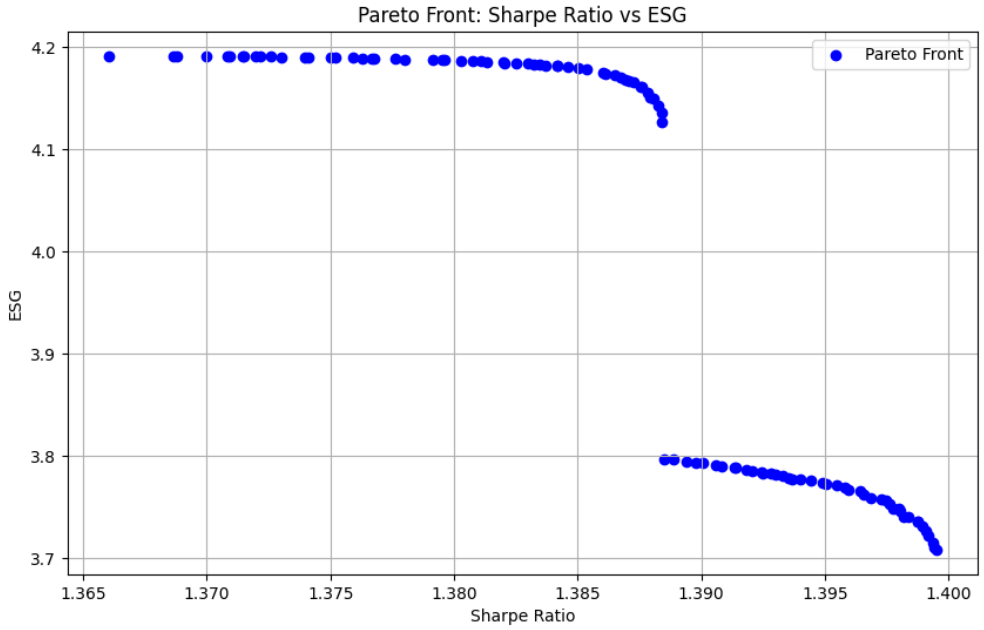


Un mayor hipervolumen es mejor puesto que indica que el frente de Pareto cubre una mayor parte del espacio objetivo, lo que significa que las soluciones en el frente de Pareto son más diversas y se extienden más en el espacio de los objetivos, proporcionando una mejor representación del trade-off entre los objetivos. Por lo tanto, en la optimización multiobjetivo, maximizar el hipervolumen es un objetivo deseable, ya que refleja tanto la calidad como la diversidad de las soluciones encontradas.

Otro de los factores que se deben tener en cuenta es que cuando se habla de optimización multiobjetivo, a menudo se debe lidiar con objetivos que se desean maximizar y otros que se desean minimizar. En este trabajo en concreto, lo óptimo es encontrar un ratio de Sharpe lo más alto posible y un máximum drawdown lo más bajo posible, es decir, son objetivos opuestos. Al tratarse de un problema de maximización, puesto que se está buscando una mayor rentabilidad, se pueden ajustar los ejes multiplicando por -1 el maximum drawdown, para convertir la minimización del drawdown en la maximización de su inverso.

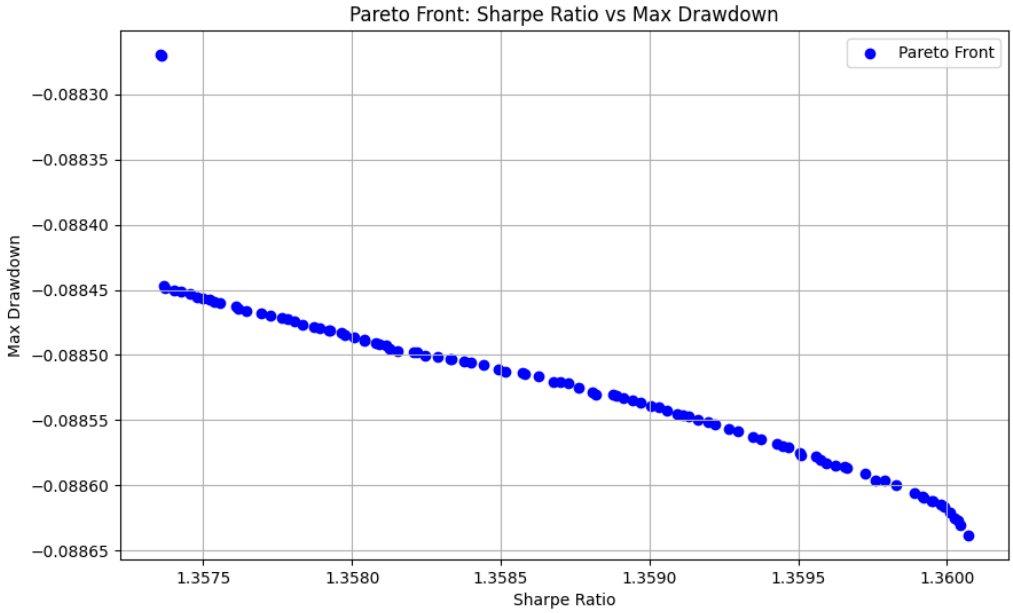
A continuación, se muestran dos gráficas con el frente de Pareto para ambos experimentos: los objetivos ESG y sharpe ratio; y los objetivos sharpe ratio y máximum drawdown.

Figura 7: Frente de Pareto: Sharpe vs ESG.



Fuente: elaboración propia a partir de los datos del código.

Figura 8: Frente de Pareto: Sharpe Ratio vs Maximum Drawdown.



Fuente: elaboración propia a partir de los datos del código.

En ambos gráficos se puede observar claramente la relación entre los objetivos. En el caso del frente de Pareto obtenido entre el máximo drawdown y el Sharpe Ratio, se muestra que estos objetivos son contradictorios. Es decir, a mayor riesgo, se obtiene una mayor rentabilidad, pero también un mayor máximo drawdown. Por otro lado, si se asume menos riesgo, se consigue un menor Sharpe Ratio, pero también un menor máximo drawdown. La figura ilustra perfectamente este trade-off entre ambos objetivos, mostrando cómo la mejora en uno conlleva inevitablemente una disminución en el otro.

5.6.3 Lanzamiento del experimento: comparación de los modelos de optimización

En esta fase se prepara, ejecuta y registra los resultados de un experimento de optimización comparativa entre los dos modelos previamente explicados: el modelo MOBO y el modelo BASELINE. Para ello se importan funciones de la librería Dragonfly, "una biblioteca de Python de código abierto para optimización bayesiana escalable. La optimización bayesiana se utiliza para optimizar funciones de caja negra cuyas evaluaciones suelen ser costosas. Más allá de las técnicas de optimización básicas,

Dragonfly proporciona una variedad de herramientas para ampliar la optimización bayesiana a problemas a gran escala" (Dragonfly Contributors, s.f.).

Para realizar la comparación se siguen las siguientes fases:

1) Fase 1: se crea un diccionario que asigna claves (nombres de modelos) a tuplas que contienen la función (o funciones) objetivo y un archivo de configuración asociados a cada modelo.
2) Fase 2: se definen variables como el número de repeticiones o el número de iteraciones y se inicializa una matriz vacía que va a almacenar los resultados de los diferentes métodos en cada repetición y sobre los objetivos.
3) Fase 3: se establecen los límites y parámetros para la optimización, como los límites de los hiperparámetros para el algoritmo de optimización multiobjetivo y un límite de restricción.
4) Fase 4: se realiza la iteración sobre las semillas aleatorias definidas en seeds. Para cada semilla, se realiza la optimización multiobjetivo con el modelo MOBO y se registra el mejor valor encontrado en la matriz vacía. Se repite el mismo proceso para el modelo BASELINE.

5.6.4 Random Search

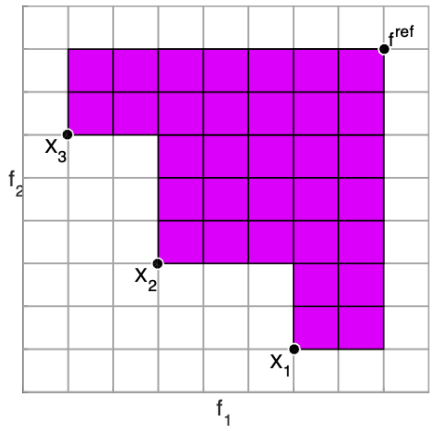
En este fragmento de código se está realizando un proceso de optimización mediante la técnica de búsqueda aleatoria (Random Search), para poder integrarlo en el experimento y comparar el rendimiento de la técnica Random Search con la optimización bayesiana. Para ello, se realiza el mismo número de iteraciones que se tomaron para el modelo MOBO. En cada iteración, se generan valores aleatorios para los hiperparámetros: entropía, learning-rate, gamma, clip-range y lambda o factor de descuento, con unos rangos especificados. Estos valores aleatorios se utilizan como argumentos para la función que representa el modelo BASELINE bajo la estrategia de optimización de búsqueda aleatoria. Se ejecuta la función con los valores aleatorios de los hiperparámetros en cada iteración y el resultado se almacena en la matriz.

5.7. Resultados

Una vez construidos los modelos, se procede a evaluar y comparar los modelos empleando dos técnicas. Por un lado, se calculan las medias de los rendimientos para entender cómo se comporta cada uno de los modelos en promedio en relación con cada uno de los objetivos, sin embargo, es una métrica limitada por que no captura la distribución completa de los resultados. Por esto mismo, se emplea una segunda métrica: el hipervolumen. Este indicador solo se emplea para el modelo MOBO y random search puesto que ambos devuelven un conjunto de datos y, además, el modelo BASELINE era un modelo puramente explicativo.

El hipervolumen es una métrica que cuantifica el volumen total del espacio objetivo cubierto por un conjunto de soluciones no dominadas, conocido como el frente de Pareto (Auger, A., et al., 2012). Este valor se evalúa respecto a un punto de referencia definido en el espacio de objetivos.

Figura 9: Representación gráfica del hipervolumen.



En el presente trabajo se han realizado dos experimentos: comparando el ESG y el Sharpe Ratio; comparando el Maximum Drawdown y el Sharpe Ratio. Debido a que los valores obtenidos del primer experimento, ESG y Sharpe Ratio, son más evidentes y ayudan a entender mejor la comparativa entre el modelo MOBO y random search, usaré este experimento para explicar los resultados obtenidos del hipervolumen.

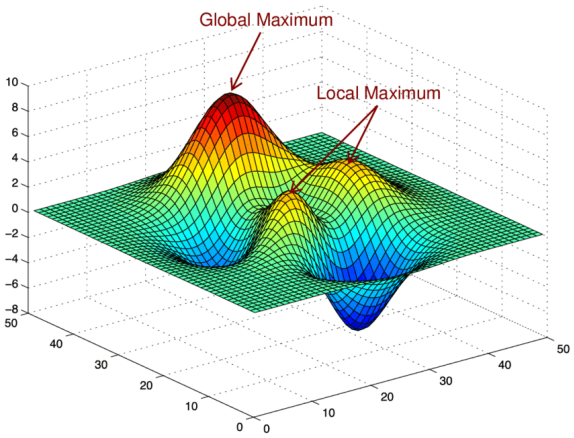
Para el cálculo del hipervolumen, es importante recalcar que el sharpe ratio se ha colocado en el eje “x” y la métrica ESG como eje “y”. El objetivo principal es comparar la capacidad de dos métodos de optimización, MOBO y random search, para encontrar soluciones no dominadas en este espacio de objetivos. Los resultados obtenidos son los siguientes:

- Hipervolumen del frente de Pareto en MOBO: 161.76864680712305
- Hipervolumen del frente de Pareto en random search: 6.1110692817318

Un hipervolumen más alto indica que método ha logrado encontrar un conjunto extenso y diverso de soluciones no dominadas en el espacio de objetivos definido por el sharpe ratio y el ESG. Por otro lado, un valor bajo de hipervolumen implica que el método ha encontrado un número limitado de soluciones. Un hipervolumen más alto es indicativo de un mejor desempeño en problemas de optimización multiobjetivo puesto que demuestra que el modelo cubre un espacio objetivo mayor y, por tanto, una mayor probabilidad de ofrecer una óptimo global. La siguiente gráfica muestra un ejemplo de ello, pero teniendo en cuenta tres dimensiones en lugar de dos, que son las que tenía el experimento realizado en este trabajo al tener en cuenta dos objetivos únicamente.

La optimización bayesiana emplea modelos probabilísticos para guiar la búsqueda hacia regiones prometedoras del espacio objetivo, lo cual facilita la exploración efectiva de soluciones óptimas en términos de Pareto. La búsqueda aleatoria, al generar soluciones de manera aleatoria sin una dirección específica hacia regiones óptimas del espacio objetivo, no logra explorar de manera eficiente el espacio de soluciones no dominadas. Un hipervolumen mayor supone una mayor diversidad y calidad de las soluciones, y asegura que las soluciones encontradas son óptimas. Por tanto, en el contexto de este experimento, la optimización bayesiana ha demostrado ser superior a la búsqueda aleatoria en términos de los objetivos evaluados.

Figura 10: Representación gráfica de los beneficios de obtener un mayor hipervolumen.



Fuente: Jin, C. (2015).

6. Conclusiones

El objetivo de la presente investigación era demostrar la superioridad de la optimización bayesiana multiobjetivo frente al random search. Para ello, se realizaron dos experimentos ilustrativos: uno tomando como objetivos el ESG y el Sharpe Ratio, y otro tomando como objetivos el Sharpe Ratio y el máximo drawdown. Los experimentos muestran que el hipervolumen, métrica empleada para comparar ambas técnicas, es superior en la optimización bayesiana frente al random search, lo que implica que existen indicios de que la optimización bayesiana produce resultados más óptimos en la gestión de carteras.

Sin embargo, el problema que se está abordando en esta investigación tiene un grado de complejidad muy elevado y requiere recursos significativos, como una gran potencia computacional, que no se han podido emplear en este estudio (una de las principales limitaciones de este trabajo). Debido a esto, se ha tenido que reducir tanto el tamaño como la complejidad del experimento. Experimentos futuros deberían comparar estos métodos con algoritmos genéticos, incluir más índices (como NASDAQ), explorar más algoritmos de DRL (como A3C) y realizar un mayor número de repeticiones.

No obstante, aunque los resultados de esta investigación no sean concluyentes para afirmar rotundamente que la optimización bayesiana da mejores resultados que el random search, se han extraído dos conclusiones clave para futuras investigaciones. Primero, este trabajo proporciona un ejemplo de cómo se debería realizar el experimento, destacando la necesidad de añadir una mayor complejidad al modelo, aunque la base ya está establecida. Segundo, el experimento muestra indicios de la mejora de la optimización bayesiana con respecto al random search, lo que podría indicar que, en escenarios con recursos computacionales adecuados y modelos más complejos, la optimización bayesiana podría ser significativamente más efectiva.

Código

El código de este trabajo se encuentra almacenado en un repositorio en GitHub. Para acceder a él, pinchar el siguiente enlace:
https://github.com/liafreirep24/TFG-Analytics.git

Declaración de Uso de Herramientas de IA

ADVERTENCIA: Desde la Universidad consideramos que ChatGPT u otras herramientas similares son herramientas muy útiles en la vida académica, aunque su uso queda siempre bajo la responsabilidad del alumno, puesto que las respuestas que proporciona pueden no ser veraces. En este sentido, NO está permitido su uso en la elaboración del Trabajo fin de Grado para generar código porque estas herramientas no son fiables en esa tarea. Aunque el código funcione, no hay garantías de que metodológicamente sea correcto, y es altamente probable que no lo sea.

Por la presente, yo, Obdulia Freire Pasquau, estudiante de E2 Business Analytics de la Universidad Pontificia Comillas al presentar mi Trabajo Fin de Grado titulado " Optimización bayesiana multiobjetivo de los hiperparámetros de un algoritmo de aprendizaje reforzado profundo en el ámbito financiero", declaro que he utilizado la herramienta de Inteligencia Artificial Generativa ChatGPT u otras similares de IAG de código sólo en el contexto de las actividades descritas a continuación:

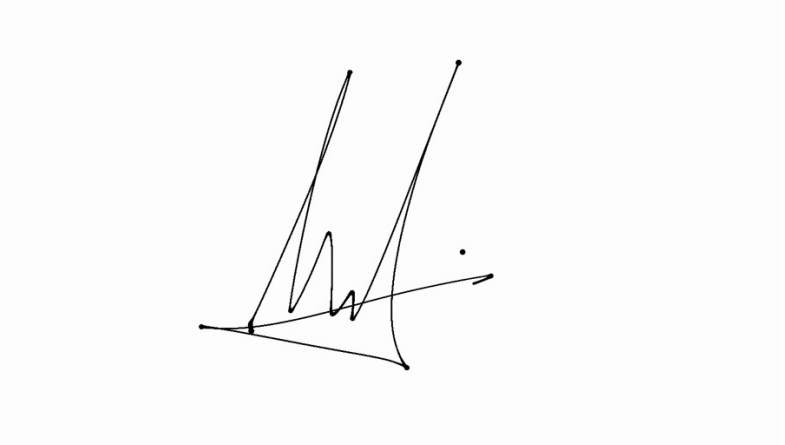
1. Brainstorming de ideas de investigación: Utilizado para idear y esbozar posibles áreas de investigación.
2. Crítico: Para encontrar contra-argumentos a una tesis específica que pretendo defender.
3. Referencias: Usado conjuntamente con otras herramientas, como Science, para identificar referencias preliminares que luego he contrastado y validado.
4. Metodólogo: Para descubrir métodos aplicables a problemas específicos de investigación.
5. Interpretador de código: Para realizar análisis de datos preliminares.
6. Estudios multidisciplinares: Para comprender perspectivas de otras comunidades sobre temas de naturaleza multidisciplinar.
7. Corrector de estilo literario y de lenguaje: Para mejorar la calidad lingüística y estilística del texto.
8. Generador previo de diagramas de flujo y contenido: Para esbozar diagramas iniciales.
9. Sintetizador y divulgador de libros complicados: Para resumir y comprender literatura compleja.

10. Revisor: Para recibir sugerencias sobre cómo mejorar y perfeccionar el trabajo con diferentes niveles de exigencia.
11. Traductor: Para traducir textos de un lenguaje a otro.

Afirmo que toda la información y contenido presentados en este trabajo son producto de mi investigación y esfuerzo individual, excepto donde se ha indicado lo contrario y se han dado los créditos correspondientes (he incluido las referencias adecuadas en el TFG y he explicitado para que se ha usado ChatGPT u otras herramientas similares). Soy consciente de las implicaciones académicas y éticas de presentar un trabajo no original y acepto las consecuencias de cualquier violación a esta declaración.

Fecha: 21 de junio de 2024.

Firma:



Referencias

Amazon Web Services. (s.f.). What is reinforcement learning? Recuperado de https://aws.amazon.com/es/what-is/reinforcement-learning/

Auger, A., Bader, J., Brockhoff, D., & Zitzler, E. (2012). Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications. Theoretical Computer Science. Recuperado de https://pdf.sciencedirectassets.com/271538/1-s2.0-S0304397511002428/main.pdf

Bartz-Beielstein, T., Branke, J., Mehnen, J., & Mersmann, O. (2014). Evolutionary algorithms. First published: April 24, 2014. https://doi.org/10.1002/widm.1124

Branke, J., Scheckenbach, B., Stein, M., Deb, K., & Schmeck, H. (2009). Portfolio optimization with an envelope-based multi-objective evolutionary algorithm. European Journal of Operational Research, 199(3), 684-693. https://doi.org/10.1016/j.ejor.2008.01.054

Brownlee, J. (2019, 6 de agosto). How to Configure the Learning Rate When Training Deep Learning Neural Networks. Recuperado de: https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/

Chambers, R. G., & Quiggin, J. (2002). Resource allocation and asset pricing (WP 02-03). Department of Agricultural and Resource Economics, The University of Maryland, College Park. Recuperado de https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=theories+of+optimization+of+financial+resource+allocation&btnG=#:~:text=Resource%20allocation%20and%20asset%20pricing

Chen, J. (2021). Technical indicator: Definition, analyst uses, types and examples. Investopedia. Recuperado de https://www.investopedia.com/terms/t/technicalindicator.asp

Chordia, T., Subrahmanyam, A., Anshuman, V.R. (2001). Trading activity and expected stock returns. Journal of Financial Economics, 82(2), 227-259. https://doi.org/10.1016/S0304-405X(00)00080-5

CNMV. (s.f.). El mercado de valores y los productos de inversión. https://www.cnmv.es/DocPortal/Publicaciones/Guias/ManualUniversitarios.pdf

Corecco, S., Adorni, G., & Gambardella, L. M. (2023). Proximal Policy Optimization-Based Reinforcement Learning and Hybrid Approaches to Explore the Cross Array Task Optimal Solution. Machine Learning and Knowledge Extraction, 5(4), 1660-1679. https://doi.org/10.3390/make5040082

Dai, Z., & Wang, F. (2019). Sparse and robust mean–variance portfolio optimization problems. Physica A: Statistical Mechanics and its Applications, 523, 1371-1378. https://doi.org/10.1016/j.physa.2019.04.151

Dragonfly Contributors. (s.f.). Dragonfly documentation. Recuperado de https://dragonfly-opt.readthedocs.io/en/master/

Garnett, R. (2023). Bayesian Optimization. Washington University in St Louis. Recuperado de https://bayesoptbook.com/book/bayesoptbook.pdf

García, F., González-Bueno, J., Oliver, J., & Rueda-Barrios, G. (2019). Medidas de riesgo en la selección de carteras [Risk measures in portfolio selection]. Revista Espacios, 40(38), 18. Recuperado de https://www.revistaespacios.com/a19v40n38/19403818.html

Garrido Merchán, E. C. (2023). La evolución de la gestión de carteras: Del modelo de Markowitz al Deep Reinforcement Learning y la Optimización Bayesiana. Recuperado de

https://expost.comillas.edu/la-evolucion-de-la-gestion-de-carteras-del-modelo-de-markowitz-al-deep-reinforcement-learning-y-la-optimizacion-bayesiana/

Garrido-Merchán, E. C., & Hernández-Lobato, D. (2020). Dealing with categorical and integer-valued variables in Bayesian Optimization with Gaussian processes. Neurocomputing, 380, 20-35. https://doi.org/10.1016/j.neucom.2019.11.004

Gasser, Stephan M. & Rammerstorfer, Margarethe & Weinmayer, Karl, 2017. "Markowitz revisited: Social portfolio engineering," European Journal of Operational Research, Elsevier, vol. 258(3), pages 1181-1190. https://ideas.repec.org/a/eee/ejores/v258y2017i3p1181-1190.html

Grajales Bedoya, D. D. (2009). Gestión de portafolios: Una mirada crítica más allá de Markowitz. Portafolio management: A critical view beyond Markowitz. Universidad EAFIT Medellín, (Número 15), julio-diciembre. Recuperado de https://www.redalyc.org/pdf/3223/322327246008.pdf

Gomes, G. F., Almeida, F. A. de, Alexandrino, P. S. L., & Cunha Jr, S. S. (2019). A multiobjective sensor placement optimization for SHM systems considering Fisher information matrix and mode shape interpolation. Engineering with Computers, 35(8). https://www.researchgate.net/publication/325404197_A_multiobjective_sensor_placement_optimization_for_SHM_systems_considering_Fisher_information_matrix_and_mode_shape_interpolation

Hoffman, M., Brochu, E., & de Freitas, N. (2011). Portfolio Allocation for Bayesian Optimization. Department of Computer Science, University of British Columbia. Recuperado de https://www.cs.ubc.ca/~nando/papers/uaiBayesOpt.pdf

Huang, S., Yang, K., Qi, S., & Wang, R. (2024). When Large Language Model Meets Optimization. Recuperado de https://arxiv.org/html/2405.10098v1

Intellabs. (s.f.). Continuous Proximal Policy Optimization. Recuperado de: https://intellabs.github.io/coach/components/agents/policy_optimization/cppo.html

J. Bergstra, Y. Bengio. (2012). Random search for hyper-parameter optimization
Journal of Machine Learning Research, 13 (1), pp. 281-305.
https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf?ref=broutonlab.com

Jin, C. (2015, July). A sequential process monitoring approach using Hidden Markov Model for unobservable process drift. https://doi.org/10.13140/RG.2.2.32537.83041.

Jobson, J. D., & Korkie, B. (2012). Estimation for Markowitz efficient portfolios. Journal of the American Statistical Association, 75(371), 544-554. https://doi.org/10.1080/01621459.1980.10477507

Jyothi, R., & Krishnamurthy, G. N. (2022). Deep-Reinforcement Learning-Based Architecture for Multi-Objective Optimization of Stock Prediction. EJECE, European Journal of Electrical Engineering and Computer Science, 6(4). Recuperado de https://www.ejece.org/index.php/ejece/article/view/436/273

Kumar, V. (2018). A simplified perspective of the Markowitz portfolio theory. International Journal of Research and Analytical Reviews, 5(3), 2348-1269. https://www.researchgate.net/publication/256034486_A_Simplified_Perspective_of_the_Markowitz_Portfolio_Theory

Latypov, O. (2023). A comprehensive guide to Proximal Policy Optimization (PPO) in AI. Medium. Recuperado de https://medium.com/@oleglatypov/a-comprehensive-guide-to-proximal-policy-optimization-ppo-in-ai-82edab5db200

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444. https://doi.org/10.1038/nature14539

Li, C., Chen, Y., Yang, X., Wang, Z., Lu, Z., & Chi, X. (2022). Intelligent Black–Litterman portfolio optimization using a decomposition-based multi-objective DIRECT algorithm. Applied Sciences, 12(14), 7089. https://doi.org/10.3390/app12147089

Lotov, A.V., Miettinen, K. (2008). Visualizing the Pareto Frontier. In: Branke, J., Deb, K., Miettinen, K., Słowiński, R. (eds) Multiobjective Optimization. Lecture Notes in Computer Science, vol 5252. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-88908-3_9

Ma, T., & McGroarty, F. (2017). Social Machines: how recent technological advances have aided financialisation. Research Article, Volume 32, páginas 234–250. Recuperado de https://link.springer.com/article/10.1057/s41265-017-0037-7

Maverick, J. B. (2024). How do the MACD and RSI indicators differ? Investopedia. Recuperado de https://www.investopedia.com/ask/answers/122214/what-are-main-differences-between-moving-average-convergence-divergence-macd-relative-strength-index.asp

Metaxiotis, K., & Liagkouras, K. (2012). Multiobjective evolutionary algorithms for portfolio management: A comprehensive literature review. Expert Systems with Applications, 39(14), 11685–11698. https://doi.org/10.1016/j.eswa.2012.04.053

Metawa, N., Elhoseny, M., Hassanien, A. E., & Hassan, M. K. (Eds.). (2019). Artificial intelligence in finance: Expert systems in finance. Smart financial applications in big data environments. Routledge.

Naeem, M., Rizvi, S. T. H., & Coronato, A. (2020). A Gentle Introduction to Reinforcement Learning and its Application in Different Fields. IEEE Access, 8. DOI: 10.1109/ACCESS.2020.3038605. Recuperado de https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9261348

Nyuytiymbiy, K. (2020). Parameters and Hyperparameters. Towards Data Science. Medium. https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac

Pardo, F., Tavakoli, A., Levdik, V., & Kormushev, P. (27 de enero de 2022). Time Limits in Reinforcement Learning. Recuperado de: https://arxiv.org/pdf/1712.00378

Perrone, V., Shen, H., Zolic, A., Shcherbatyi, I., Ahmed, A., Bansal, T., Donini, M., Winkelmolen, F., Jenatton, R., Faddoul, J. B., & Pogorzelska, B. (2021). Amazon SageMaker Automatic Model Tuning: Scalable Gradient-Free Optimization. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '21). https://doi.org/10.1145/3447548.3467098

Python Package Index. (s.f.). PyPI - the Python Package Index. Recuperado de https://pypi.org/

Rumyantseva, A., & Tarutko, O. (2022). Impact of the ESG Principles on the Corporate Financial Strategy. En Springer Proceedings in Business and Economics. Recuperado de https://link.springer.com/chapter/10.1007/978-3-031-14410-3_32

scikit-learn. (s.f.). Cross-validation: Evaluating estimator performance. Recuperado de https://scikit-learn.org/stable/modules/cross_validation.html

Singh, V., Chen, S.-S., Singhania, M., Nanavati, B., Kar, A. K., & Gupta, A. (2022). How are reinforcement learning and deep learning algorithms used for big data based decision making in financial industries – A review and research agenda. International Journal of Information Management Data Insights. Recuperado de https://pdf.sciencedirectassets.com/778772/1-s2.0-S2667096822X00025/1-s2.0-S2667096822000374/main.pdf

Sohangir, S., Wang, D., Pomeranets, A. et al. (2018). Big Data: Deep Learning for financial sentiment analysis. J Big Data 5, 3. https://doi.org/10.1186/s40537-017-0111-6

Sood, S., Papasotiriou, K., Vaiciulis, M., & Balch, T. (2023). Deep Reinforcement Learning for Optimal Portfolio Allocation: A Comparative Study with Mean-Variance Optimization. J.P. Morgan AI

Research, J.P. Morgan Global Equities; Oxford-Man Institute of Quantitative Finance. Recuperado de: https://icaps23.icaps-conference.org/papers/finplan/FinPlan23_paper_4.pdf

Stable Baselines. (2021). Proximal Policy Optimization (PPO) - Stable Baselines 2. Disponible en: https://stable-baselines.readthedocs.io/es/master/modules/ppo2.html

Sutton, R. S. & Barto, A. G. (2015). Reinforcement Learning: An Introduction. Recuperado de: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

Théberge, A. (2019). Clipped Proximal Policy Optimization Algorithm. VITALab. https://vitalab.github.io/article/2019/05/09/PPO.html

Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., & Guyon, I. (2021). Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020. arXiv preprint. Rceuperado de from https://arxiv.org/abs/2104.10201

Vasconcelos, T. de P., de Souza, D. A. R. M. A., Virgolino, G. C. de M., Mattos, C. L. C., & Gomes, J. P. P. (2022). Self-tuning portfolio-based Bayesian optimization. Expert Systems with Applications, 188, Article 113619. https://www.sciencedirect.com/science/article/abs/pii/S0957417421012082

Wieland, O. L. (2015). Modern financial markets and the complexity of financial innovation. Department of Business, University of Minnesota Crookston, United States. Recuperado de https://www.researchgate.net/publication/283185927_Modern_Financial_Markets_and_the_Complexity_of_Financial_Innovation

Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H., & Deng, S.-H. (2019). Hyperparameter optimization for machine learning models based on Bayesian optimization. Journal of Electronic Science and Technology, 17(1), 26-40. https://www.sciencedirect.com/science/article/pii/S1674862X19300047

Xu, J. (2022). Future and Fintech: ABCDI and Beyond. Springer. Recuperado de https://books.google.es/books?hl=en&lr=&id=mWF2EAAAQBAJ&oi=fnd&pg=PA451&dq=MLOps+financial&ots=lzN3zuCiAD&sig=HMyZx4XcshrzXJOez4EO0vhRzl8&redir_esc=y#v=onepage&q=MLOps%20financial&f=false

Yu, P., Lee, J. S., Kulyatin, I., Shi, Z., & Dasgupta, S. (2019). Model-based Deep Reinforcement Learning for Dynamic Portfolio Optimization. Recuperado de https://arxiv.org/pdf/1901.08740v1

Zhang, Z., Zohren, S., & Roberts, S. (2019). Deep Reinforcement Learning for Trading. Department of Engineering Science, Oxford-Man Institute of Quantitative Finance, University of Oxford. Recuperado de https://arxiv.org/pdf/1911.10107

Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., & Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. https://i2pc.es/coss/Docencia/SignalProcessingReviews/Zhou2011.pdf