Universidad de San Andrés

Escuela de Negocios

Licenciatura en Finanzas

Aprendizaje de refuerzo aplicado a cobertura de derivados

Autor: Santiago Martín Kestler

Legajo: 31064

Director: Pablo Macri

Co-Director: Federico Glancszpigel

Buenos Aires, Argentina, Junio 2024

Índice

Índice de figuras

Índice de cuadros

Siglas

DDPG *deep deterministic policy gradient* o política de gradiente determinística profunda en inglés

MDP *Markov decision process* o proceso de decisión de Markov en inglés

P&L *profit and loss* o ganancia y pérdida en inglés

BSM Black Scholes Merton

Resumen

En el presente trabajo se estudia el uso de un algoritmo de aprendizaje de refuerzo o *reinforcement learning* en inglés, para realizar cobertura de riesgo en derivados. En especial, se analiza la factibilidad de este algoritmo para su aplicación en casos de cobertura de opciones de compra y venta europeas. Para ello, se comparan las posiciones de cobertura del algoritmo con aquellos de la resolución por el modelo tradicional de cobertura utilizando la fórmula de Black Scholes Merton (BSM). Se concluye tras el análisis de las posiciones y del error de estimación del modelo que es posible la aplicación de este algoritmo para cubrir el riesgo de derivados proveniente del subyacente. Además, se estima que para el modelo aplicado es necesario repetir el entrenamiento para un nuevo derivado. Por último, es necesario contar con una elevada cantidad de datos para ejercer un entrenamiento efectivo del programa.

1. Introducción

Los derivados son contratos financieros que adquieren valor a partir de uno o varios activos. Los precios de estos derivados fluctúan a partir de los subyacentes y son usados principalmente para acceder mercados particulares y como forma de cubrirse ante el riesgo. En la administración de portafolios, estos instrumentos son utilizados para reducir el riesgo cubriendo posiciones, y así agregar valor mediante el acceso o el ajuste de exposiciones a clases de activos y administrar flujos de dinero. Si bien los derivados son útiles de diversas formas, no son tan fáciles de valuar en comparación a un bono o una acción dada la complejidad del pago al vencimiento que estos poseen ya que dependen del valor de otro activo. En el caso de las opciones europeas se cuenta con una fórmula cerrada para calcular su valor. Sin embargo, hay otros casos en los que esto no necesariamente existe por lo que se debe recurrir a modelos matemáticos más complejos. Cabe destacar que el valor no es lo único que puede aportar una fórmula cerrada para los derivados; ésta da a conocer los factores que afectan al precio del activo y por lo tanto, permite que el agente pueda cubrirse ante estos factores. En particular, en este trabajo se hace hincapié en el efecto que tienen los movimientos del precio del subyacente en los derivados. Una forma de estudiar este impacto es a través de la sensibilidad que tiene cada derivado al precio del subyacente. Esto se conoce como la delta de la opción.

Para las opciones europeas, el método que permite la obtención de esta sensibilidad es la fórmula propuesta por BSM, que por medio de sus derivadas parciales da el respectivo efecto de los factores de riesgo. Sin embargo, para aplicar esta fórmula es necesario asumir supuestos. Por ejemplo, se considera una volatilidad constante o la ausencia de costos de transacción en el mercado. Por otro lado, este método no es aplicable a otro tipo de opciones con diferentes características en sus pagos al vencimiento como las opciones americanas u opciones exóticas como las asiáticas depednientes del promedio del activo subyacente.

A partir de este escenario, se busca encontrar un modelo con un alcance más universal para encontrar la sensibilidad al precio del subyacente y lograr aplicarlo a un caso de cobertura dinámica. Es decir, se intentará lograr que el modelo sea capaz de rebalancear las posiciones en el momento que se produzca un movimiento en el precio del subyacente. En función a esto, se propone utilizar un método de inteligencia artificial denominado aprendizaje de refuerzo (*reinforcement learning* en inglés). Este método consiste en un agente autónomo que toma decisiones en pos de maximizar una función de

recompensa. En este caso, el agente tiene como objetivo determinar una política de decisión óptima para maximizar la ganancia (o minimizar el costo) de realizar la cobertura de una cartera de derivados financieros.

2. Revisión literaria

El aprendizaje de refuerzo fue propuesto por Bellman (1957). Este trabajo establece los pilares fundamentales del modelo. En primer lugar, el autor formula los procesos de decisión de Markov. Dichos procesos se basan en las cadenas de Markov, que a grandes rasgos son definidos como procesos estocásticos sin memoria. La principal diferencia entre procesos de decisión y las cadenas de Markov, es que en los procesos de decisión hay un agente cuyo objetivo es encontrar una política de decisión que maximice una función de recompensa dada. En este sentido, Watkins (1989) a partir de la ecuación de Bellman, plantea un algoritmo para encontrar las acciones que maximicen la función de recompensa dada, y así encontrar la política de decisión óptima dentro de un proceso de decisión de Markov. Este algoritmo se denomina como aprendizaje Q (*Q-Learning* en inglés). Consiste en tener en cuenta la media móvil de las recompensas que el agente obtiene en un estado, más la suma de las recompensas esperadas actualizadas al estado actual, estimando la suma como el máximo valor Q calculado para el próximo estado.

Sin embargo, el aprendizaje Q no funciona óptimamente con procesos que contengan gran cantidad de estados. A razón de ello, Mnih et al. (2013) propuso como solución la implementación de redes neuronales profundas como una forma de aproximar el valor Q futuro para cada acción posible mediante las redes neurales. Se elije el mayor valor y se lo descuenta, dando como resultado la suma estimada de las recompensas futuras descontadas. De esta forma, con la suma estimada de recompensas y la recompensa actual, se puede obtener un valor Q objetivo que puede ser implementado en un entrenamiento utilizando un algoritmo de descenso de gradiente. Algunos algoritmos que utilizan este método son el *deep Q-learning* o aprendizaje Q profundo en inglés, que utiliza redes neuronales para generar valores Q, *double deep Q-learning* o aprendizaje Q profundo doble en inglés, que consiste en incorporar un modelo de entrenamiento para seleccionar los valores Q para los estados futuros. Por último, se puede mencionar el *dueling deep Q-learning* o aprendizaje Q profundo de duelo en inglés, a partir de la división de el valor Q en el valor de un estado y la ventaja de tomar una acción, estima estos componentes a través de redes neuronales.

En el plano de las finanzas, se ha aplicado el aprendizaje de refuerzo a distintas áreas. En este trabajo, nos enfocaremos en los trabajos relacionados a la cobertura de derivados. Hull (2020) explica como se puede implementar el aprendizaje de refuerzo en un caso de cobertura delta de opciones europeas en un mercado con costos de transacción, teniendo como objetivo

minimizar la varianza del costo de cobertura. Por otro lado, en el artículo Cao et al. (2021) se implementó un algoritmo que calcula las recompensas a partir de un modelo híbrido entre *profit and loss* o ganancia y pérdida en inglés (P&L) y formulación de flujos de caja de una cartera de derivados. Utiliza un algoritmo de *deep deterministic policy gradient* o política de gradiente determinística profunda en inglés (DDPG), que consiste en utilizar redes neuronales para aproximar tanto la política que sigue el agente como las funciones de acción-valor. Por otro lado, Buehler, Gonon, Teichmann, Wood et al. (2019) estudian un algoritmo de aprendizaje Q para cobertura de portafolio en mercados en donde se rige el modelo de BSM y otro bajo un modelo de volatilidad estocástica que propone Heston. Por otro lado, con el trabajo que llevan a cabo Buehler, Gonon, Teichmann y Wood (2019) se expande el estudio anterior de modelos que incorporan costos de transacción y volatilidad estocástica con una aplicación a opciones de compra europeas con datos de mercado del índice S&P 500. En último lugar, el estudio presentado por Kolm y Ritter (2019) presenta un caso de cobertura de derivados bajo un entorno con costos de transacción con un algoritmo de aprendizaje Q, pero modificando el objetivo bajo una aproximación denominada *one-step sarsa target* o objetiva sarsa de un paso en inglés, en donde la función objetivo no es necesariamente la que resulta en el mayor valor.

En relación con lo presentado, este trabajo se enfoca en replicar los resultados de los estudios mencionados al aplicar algoritmos de aprendizaje Q en la cobertura de opciones que posean o no, una fórmula para el cálculo de su valor, analizando si es viable la aplicación de este modelo para realizar una cobertura dinámica de los derivados que se le provee al algoritmo.

3. Metodología

3.1. *Hedging*

Para este trabajo se toma en consideración el estudio de opciones. Este derivado se ve afectado por los movimientos de su subyacente, siendo este una acción del mercado, se puede plantear un portafolio capaz de cubrir el riesgo del derivado que proviene del subyacente. Este portafolio de cobertura para cada momento t hasta el vencimiento del derivado se ve representado como $H_t = m_t(S_{t+1} - S_t) - N(V_{t+1} - V_t)$ en donde S_t es el precio del subyacente y V_t es el precio del derivado para cada momento t del tiempo, N es la posición total en el derivado y m_t es la posición en el subyacente con la que se ejerce la cobertura en cada t. Para el caso tradicional aplicando BSM se define que m_t es representado como la variación del valor del derivado en el subyacente para cada momento del tiempo representado como la delta del activo $\Delta = \frac{\partial V}{\partial S}$. A su vez, es posible extender este portafolio a los casos en los que se encuentren múltiples activos en el mercado de forma que $H_t = m_t^1(S_{t+1}^1 - S_t^1) + m_t^2(S_{t+1}^2 - S_t^2) + \cdots + m_t^i(S_{t+1}^i - S_t^i) - N(V_{t+1} - V_t)$ con $i = 1, \ldots, n$ siendo n la cantidad total de activos. De esta manera, la posición m_t^i con BSM se define como la derivada parcial del valor de la opción en el subyacente i, tal que $m_t^i = \frac{\partial V}{\partial S^i}$.

3.2. Aprendizaje de refuerzo

En general, se puede plantear un algoritmo de aprendizaje de refuerzo por medio de la relación entre un agente y su entorno. El agente es un programa que se encarga de interactuar con el ambiente simulado o real en el que se encuentra por medio de la toma de decisiones. Una vez que el agente realiza una acción, el entorno responde por medio de una recompensa o penalización y cambia el estado en el que se encuentra el agente, se denomina estado a la situación en la que se encuentra el agente en cada momento del tiempo. El desafío se presenta con los componentes del ambiente que el agente no puede controlar, por ejemplo, el modelo puede saber como son calculadas sus recompensas, pero no tiene forma de cambiar ese cálculo para obtener más valor, en consecuencia a la relación entre el estado y las acciones que se toman con la recompensa resultante de la interacción. Por lo tanto, el único medio por el que el agente puede maximizar las recompensas es aprendiendo a tomar la decisión que pueda aportar mayor valor para el estado actual.

3.3. Proceso de decisión de Markov

La representación del ambiente en los algoritmos de aprendizaje de refuerzo se realiza a través de procesos, en especial, se define a través del MDP. Para explicar este proceso es necesario partir de un nivel más simple llamado cadenas de Markov. Las cadenas de Markov son procesos aleatorios sin memoria, en el contexto del aprendizaje de refuerzo, una cadena de Markov es una serie de estados aleatorios que siguen la denominada propiedad de Markov. Esta propiedad establece que el futuro es independiente del pasado dado el presente y se puede expresar en la siguiente ecuación:

$$P\left[S_{t+1}|S_t\right] = P\left[S_{t+1}|S_1, \ldots, S_t\right]$$

En donde S_t representa el estado actual en el momento t en el que se encuentra el agente, mientras que S_{t+1} es el estado futuro. A partir de esta ecuación se puede interpretar que el estado actual en que se encuentra el agente percibe toda la información de los estados pasados. Esto se debe a que bajo la propiedad de Markov la probabilidad condicional de pasar a un estado futuro dado el estado actual del agente es igual a la probabilidad de pasar a un próximo estado dado todos los estados anteriores.
En base a esto, las cadenas de Markov utilizan las probabilidades de transición entre estados que puede ser definidas bajo la siguiente fórmula:

$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s'|S_t = s\right]$$

A su vez, hay que tener en cuenta que para cada par de estados se le asigna una probabilidad de transición diferente, esto deriva en la matriz de probabilidad de transición de estado como se representa a continuación:

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix}$$

Concluyendo con las cadenas de Markov, se puede examinar la figura 1 en donde se representa una estructura común de este proceso, en este caso, representado con el conjunto de estados $s = S_0, S_1, S_2, S_3$ en donde en cada estado hay una probabilidad de transición representado por cada flecha, por ejemplo, estando en el estado S_0 el agente tiene una probabilidad de 20 % de pasar al estado S_1 o del 10 % de pasar al estado S_3 o finalmente, una probabilidad del 70 % de quedarse en ese mismo estado.

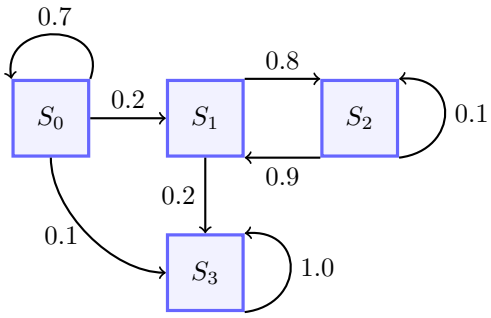

Figura 1: Diagrama de cadena de Markov

Dando paso a los procesos de decisión de Markov se definen algunos componentes que no están presentes en las cadenas de Markov, las recompensas y las acciones que el agente puede realizar. Las recompensas en un MDP se representan en función del estado que se encuentra el agente y la acción que toma. De esta manera, se definen las recompensas mediante la seguiente fórmula:

$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1}|S_t = s, A_t = a]$$

En resumen, esto expresa que la función de recompensa es el valor esperado de la recompensa futura R_{t+1} dado el estado en el que se encuentra el agente S_t y la acción que toma el agente en el presente A_t.
A su vez, la incorporación de acciones condiciona la matriz de probabilidades de transición. En este caso, la probabilidad no depende solamente del estado en el que se encontraba el agente, sino también que se ve modificada por la decisión del agente. Dado que se relaciona con el estado futuro en donde se posicionará el agente, como muestra la siguiente ecuación:

$$\mathcal{P}_{ss'}^a = \mathbb{P}\left[S_{t+1} = s'|S_t = s, A_t = a\right]$$

Por otro lado, para realizar una acción el algoritmo opera bajo una función que se denomina política. Esta función define la distribución de probabilidad de las acciones para cada estado como se ve expresado en la ecuación 1.

$$\pi\left(a|s\right) = \mathbb{P}\left[A_t = a|S_t = s\right] \tag{1}$$

Bajo esta política, podemos encontrar que el valor de un estado es igual al rendimiento esperado obtenido partiendo del estado actual y siguiendo la misma política por todos los estados restantes, hasta el estado terminal como se representa en la ecuación 2.

$$v_\pi \doteq \mathbb{E}\left[G_t|S_t = s\right], para\ todo\ s \in S \tag{2}$$

Siendo G_t la sumatoria de recompensas ponderadas por un factor de descuento relacionado a la importancia de la recompensa para la tarea que se realiza, siendo expresada de la siguiente manera:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

El objetivo del algoritmo es obtener el mayor valor posible. Para ello el agente debe ser capaz de maximizar la política. Una política es mejor que otra cuando la función de valor alcanzada en todos los estados posibles con la política π es mayor que la alcanzada con una política π'.
En conjunto con todo lo desarrollado, el proceso de decisión de Markov se define como cadenas de Markov a las que se le incorpora un proceso de recompensas con un agente capaz de tomar decisiones frente a un conjunto de acciones para tomar.

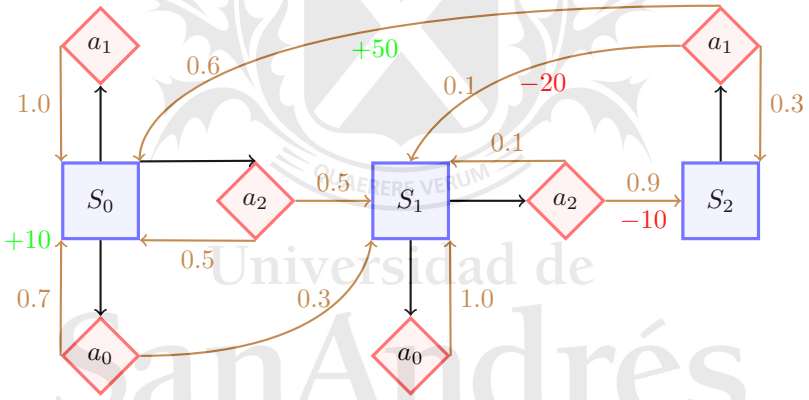

Figura 2: Diagrama de MDP

En contraste, la figura 2 muestra la estructura de un ambiente definido por un MDP, los cuadrados representan los estados que hay en el ambiente, mientras que los rombos son las acciones que el agente puede tomar, en cada estado las flechas negras nos muestran las acciones a las que el agente tiene acceso y las flechas marrones muestran los posibles caminos en los que puede terminar el agente en base a la acción que tomó en donde tiene las posibilidad de obtener recompensas o recibir una penalización. A modo de ejemplo, si el agente comienza en S_0 puede realizar tres acciones a_0, a_1 o a_2 a partir de esto en el caso de tomar a_0 tiene 70 % de probabilidad de obtener diez puntos de recompensa y mantenerse en el estado actual o un 30 % de probabilidad de terminar en el estado S_1 y no obtener niguna recompensa.

3.3.1. Ecuación de Bellman, su valor esperado y la función Q

Para encontrar políticas óptimas y funciones de valor (como la detallada anteriormente en la ecuación 2), se utiliza la ecuación de Bellman definida en la expresión 3. Esta fórmula expresa el valor de estar en un nuevo estado como la expectativa de recompensa que tiene el agente al dejar el estado en el que se encontraba junto con el valor del estado futuro.

$$v(s) = \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) | S_t = s\right] \tag{3}$$

Sin embargo, la ecuación de Bellman no incorpora una política de decisiones, para ello se define la ecuación esperada de Bellman donde el valor del estado depende también de la política, como se puede observar en la siguiente fórmula:

$$v_\pi(s) = \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s\right]$$

Se considera que la ecuación de valor es útil para evaluar la efectividad de una política pero no permite la búsqueda de una política óptima. Para este cálculo se utiliza la función del valor estado-acción o también llamada función Q, que se expresa de la siguiente manera:

$$q_\pi(s,a) \doteq \mathbb{E}_\pi\left[G_t | S_t = s, A_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$$

Esta función presenta el valor de un par estado-acción reflejado en la suma de recompensas futuras descontadas que el agente espera obtener en promedio, tras alcanzar un estado siguiendo una política.
A su vez, la función Q se puede reformular en base a la recompensa inmediata y el valor Q de un estado futuro:

$$q_\pi(s,a) \doteq \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a\right]$$

Suponiendo que al estar en un estado s el agente se enfrenta a dos acciones, cada una tiene un valor Q diferente, por lo tanto, para definir el valor de encontrarse en un estado se puede promediar estos valores Q como señala la ecuación 4.

$$v_\pi(s) = \sum_{a \in A} \pi\left(a|s\right) q_\pi\left(s,a\right) \tag{4}$$

De esta forma, como se observa en la ecuación 5, el valor de tomar una acción es el resultado del promedio de las funciones de valor-estado para cada estado del entorno tras la toma de esa acción, más la recompensa obtenida.

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \tag{5}$$

Para saber que tanto valor aporta pasar de un estado a otro bajo una política se debe promediar el valor de los estados futuros con una probabilidad de transición ponderada por la política que sigue el agente, como se puede observar en la siguiente fórmula:

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$

A su vez, en base a que el agente toma una acción que lo deja en un estado s' y a partir de allí es capaz de tomar otra acción bajo una política. Se puede representar el valor que tiene la toma de esa acción como:

$$q_\pi(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'|s') q_\pi(s', a')$$

3.3.2. Ecuación óptima de Bellman y política óptima

Para resolver un MDP se debe hallar la función óptima de valor: aquella que alcance el mayor valor comparada con las demás funciones, que se define de la siguiente manera:

$$v_*(s) = \max_\pi \ v_\pi(s)$$

Notese que en este caso la función óptima de valor se obtiene al maximizar la política, es decir, al encontrar la función que tome las decisiones que mayor valor aportan para todos los estados. Sin embargo, como se discutió anteriormente, no es conveniente usar la función de valor para la búsqueda de una política óptima por lo que es necesario centrarse primero en la optimización de la función Q.

$$q_*(s, a) = \max_\pi \ q_\pi(s, a)$$

La ecuación óptima de Bellman se basa en que el agente, ante un estado en el que se enfrenta con múltiples acciones, tome aquella con mayor valor Q bajo la política óptima, como se detalla a continuación:

$$v_*(s) = \max_a q_*(s, a)$$

Se siguen teniendo en cuenta los valores de los estados futuros, pero en este caso, se conocen los valores óptimos de cada estado, como en la ecuación 6.

$$q_*(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s') \tag{6}$$

Como se puede observar en la ecuación 7, para el valor del estado se promedian los valores óptimos de los posibles estados.

$$v_*(s) = \max_a R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s') \tag{7}$$

Dada la acción que toma el agente, el ambiente lo deja en determinado estado donde busca maximimizar el valor de la acción que toma, como se ve a continuación:

$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} q_*(s', a')$$

Finalmente, como se puede observar en la 8 la política óptima se puede obtener al hallar $q_*(s,a)$ y tomar la acción que permite al agente alcanzar ese valor Q.

$$\pi_*(a|s) = \begin{cases} 1 & si\ a = \text{argmax}_{a \in A}\, q_*(s,a) \\ 0 & en\ otro\ caso \end{cases} \tag{8}$$

Para aproximar la política óptima se utilizan algoritmos como la iteración de política o la iteración de valores. El algoritmo de iteración de política funciona a través de dos pasos, primero evalúa una política y luego, actúa *greedy* o ambiciosamente en inglés con esa política. El primer paso, consiste en usar la ecuación esperada de Bellman para evaluar una política a través de todos los estados. Una vez finalizado el primer proceso, el agente actúa ambiciosamente, es decir, realiza las acciones que lo llevan a los estados que mayor valor arrojaron en el paso anterior con la política evaluada. De esta forma, el algoritmo comienza tomando decisiones aleatorias como política, la evalúa, actúa ambiciosamente y vuelve al primer paso con esta nueva política, así iterativamente hasta converger a la política óptima.

Por otro lado, la iteración de valores, como indica su nombre, se centra en estimar la función de valor óptima en vez tratar directamente con la política. Esto se debe a que la función de valor óptima es aquella que sigue la política óptima. Por lo tanto, el modelo consiste en iterar la función de valor que se observo en la ecuación 7, es decir, actualizando el valor de un estado en base al valor de los estados del anteriores hasta converger a la función de valor óptima para todos los estados. Sin embargo, aunque este algoritmo proporcione la función de valor que sigue la política óptima, no da a conocer cuál es esa política. Para abordar este problema surge una variante del modelo que se centra en la iteración de valores Q. Con este algoritmo se itera la ecuación 6 para estimar los valores Q óptimos en función de cada par de estado y acción. Una vez obtenidos estos valores, la política óptima será aquella alcanzada al

tomar las decisiones que resultan en el mayor valor Q para cada estado del entorno.

3.4. Aprendizaje Q

Las soluciones que se abordaron para un MDP consideran que el agente conoce las probabilidades de transición entre estados y la recompensas que obtendrá. Sin embargo, en una gran cantidad de casos, como en la cobertura de derivados, el agente desconoce estos componentes. Para estos casos se utilizan algoritmos como el aprendizaje Q. Este programa, al igual que el algoritmo iterativo de valores Q, se centra en encontrar la política óptima a través de la maximización de los valores Q.

$$Q(s,a) \leftarrow \alpha r + \gamma \max_{a'} Q(s',a') \tag{9}$$

La idea principal de este algoritmo es estimar los valores Q a medida que avanza en el entorno y explora las recompensas que obtiene al realizar acciones. Como se observa en la ecuación 9, el algoritmo aproxima los valores Q basado en dos términos, el primer término mantiene un registro de las recompensas obtenidas y el segundo término, toma en cuenta la suma de recompensas esperadas descontadas al estado actual, actuando de manera óptima. En particular, r es una media móvil de las recompensas obtenidas y α es la tasa de aprendizaje, es decir, a una tasa mayor, más importancia tendrá lo ya explorado por el agente. Luego, para estimar la suma de recompensas se toma el máximo valor Q posible para el próximo estado del entorno, dado que se asume que el agente esta actuando óptimamente y γ es la tasa de descuento que trae al valor del estado actual la suma de recompensas futuras.

A continuación se detallan los pasos para la implementación del programa de aprendizaje Q:

- En primer lugar, cada vez que el agente realiza una acción, termina en otro estado con una recompensa. Para este cambio de estado se define una función de la siguiente manera:

```
def cambio(estado,accion):

    P= probabilidad_transicion[estado][accion]
    prox_estado= np.random.choice([0,1,2],p= P)
    recompensa= recompensas[estado][accion][prox_estado]
```

```
        return prox_estado,recompensa
```

En general, una vez el agente toma una decisión, en base a las probabilidades de transición que el agente no conoce, termina en otro estado con la recompensa correspondiente de ese camino,

- Luego, se define una política de exploración con la que el agente toma decisiones aleatorias para explorar el entorno.

```
def pol_exploracion(estado):
    accion= np.random.choice(acciones[estado])
return accion
```

- Por último, con la tasa de aprendizaje, una tasa de decrecimiento, la tasa de descuento y el estado inicial se inician las iteraciones. En cada instancia el agente, por medio del accionar, avanza en el entorno pasando de un estado a otro y redefine los valores Q como expresa la ecuación 9,

```
for i in range(N):

    accion= pol_exploracion(estado)
    estado_futuro, recompensa= cambio(estado,accion)

    valorQ_futuro= max(valor_Q[estado_futuro])
    alfa= alfa*factor_decrecimiento

    valor_Q[estado,accion]*= 1-alfa
    valor_Q[estado,accion]+= alfa*(recompensa+
                             gama*valorQ_futuro)

    estado= proximo_estado
```

Una aplicación de este algoritmo, a modo de ejemplo, es un caso de *trading* o operación en inglés, de una acción. En este caso, los estados del entorno son los precios, las acciones son dos, comprar o vender y las recompensas, la ganancia o pérdida que hay entre estados. Entonces, se inicializan los valores Q en cero con los demás parámetros necesarios y el agente comienza por tomar decisiones aleatorias. A medida que toma decisiones, incurre en pérdidas o ganancias que lo llevan a diferentes valores Q, hasta llegar hasta el último dato de la serie de tiempo. Luego, se vuelve a repetir este entrenamiento una cantidad de N veces mientras se actualizan los valores Q a medida que se

toman decisiones que alcanzan un valor mayor adoptando una política ambiciosa. De esta forma, el algoritmo termina convergiendo hacia la política óptima con valores Q óptimos para el agente.

4. Modelo

A continuación, se describe la serie de pasos por los que se construye el algoritmo de aprendizaje de refuerzo para la cobertura de opciones europeas:

(1) Se inicializan los hiperparámetros para el entrenamiento y testeo del modelo como la cantidad de acciones que puede realizar el agente, los estados del ambiente, la tasa de política ε-greedy, la tasa de actualización α y los parámetros para el cálculo de precios del subyacente y el derivado,

```
muestra_entrenamiento= # datos_entrenamiento
muestra_testeo= # datos_testeo

N_position= posicion_derivado
N_action= cantidad_decisiones

epsilon_minimo= tolerancia_epsilon
epsilon= probabilidad_exploracion
tasa_decrecimiento= decrecimiento_epsilon

estados_precio_stock= cantidad_total_estados
alfa= alfa_valorQ

mu= retorno_anual_stock
volatilidad= volatilidad_anual
valor_stock= valor_inicial_stock
rf= tasa_libre_riesgo

ejercicio= precio_ejercicio
Maturity= vencimiento_derivado
```

(2) Se crea la tabla de precios para el derivado y el subyacente, con este último se definen la cantidad de estados en función a un límite inferior y superior. Además, se crea la tabla de valores Q,

```
precio_stock= tabla_precios_a_vencimiento
precio_derivado= tabla_precios_derivado

valor_Q= np.zeros((Maturity+1,
        estados_precio_stock
        N_position,N_action))
```

```
tabla_estados= np.round(precio_stock)
tabla_estados= np.where(
            tabla_estados<limite_inferior,
            limite_inferior,
            tabla_estados)
tabla_estados= np.where(
            tabla_estados>limite_superior,
            limite_superior,
            tabla_estados)

tabla_estados= tabla_estados-limite_inferior
```

(3) El agente realiza una acción bajo la política ε-*greedy* o ε-ambiciosa en inglés, en donde decide si comprar o vender una cantidad de subyacente "a" para cubrir el derivado. En este caso, si el agente decide ser ambicioso, realizará la acción que menor valor Q genera buscando disminuir su costo de cobertura,

```
for t in range(Maturity):
    if np.random.rand()<= epsilon:
        accion= random.randrange(0,N_action)
    else:
        try:
          accion= min(valor_Q[t][s][a])
        except:
          accion= random.randrange(0,N_action)
```

(4) Se define la recompensa que obtiene el agente como la varianza del costo de cobertura H_t^2 compuesto por la variación del precio del derivado $N(V_{t+1} - V_t)$, con N como la posición total en el derivado y la variación del precio del subyacente $m_t(S_{t+1} - S_t)$ con m_t como la posición en el subyacente en cada momento t,

```
if operacion=='short' and opcion=='call':
    coef_1= 1
    coef_2= -1
if operacion=='long' and opcion=='put':
    coef_1= -1
    coef_2= -1
if operacion=='long' and opcion=='call':
    coef_1= -1
    coef_2= 1
```

```
    if operacion=='short' and opcion=='put':
        coef_1= 1
        coef_2= 1

recompensa= (coef_1*N*(opcion[t+1]-opcion[t])+
            coef_2*m*(stock[t+1]-stock[t]))**2
```

(5) Luego, se definen los valores Q y se actualizan por medio del aprendizaje Q como se detalla en las secciones anteriores. Luego, se actualiza la probabilidad ε con una tasa de decrecimiento para que el agente sea más propenso a actuar ambiciosamente,

```
    for t in range(Maturity):

        nuevo_valorQ= recompensa +
            f_descuento*valor_Q[t+1][s+1][a+1]

        valor_Q[t][s][a]= valor_Q[t][s][a]+
            alfa*(nuevo_valorQ-valor_Q[t][s][a])

    if epsilon>epsilon_minimo:
        epsilon= epsilon*tasa_decrecimiento
```

(6) Finaliza el entrenamiento y comienza el testeo. Los valores Q utilizados son los resultantes de la fase de entrenamiento. Se definen nuevamente las tablas de precios y se crea la tabla de valores para la delta del derivado en cada momento t,

(7) Se definen las posiciones en el subyacente a modo de cobertura para el análisis del modelo. Para la cobertura delta se utilizan los valores de la tabla creada anteriormente resultantes de la derivación parcial de la fórmula de valor. Luego, para el modelo de aprendizaje de refuerzo se utilizan las acciones que toma el agente correspondiente a los valores Q obtenidos del entrenamiento,

(8) Se registra los resultados de las variaciones de precio en cada momento con su respectiva posición definida por el algoritmo de aprendizaje de refuerzo, en base al método P&L. De esta manera, el resultado para el derivado es $P\&L = N(V_{t+1} - V_t)$ y para el subyacente $P\&L = m_t(S_{t+1} - S_t)$,

(9) Se definen las recompensas que obtiene el agente al igual que en el entrenamiento, tanto teniendo en cuenta la delta y como la posición que

toma el agente para los mínimos valores Q,

(10) En función a los datos, se calcula la desviación absoluta (DA) entre las posiciones de cobertura que se toman de los dos métodos $DA = \frac{\sum_{t=0}^{T} |\Delta_t - P_t|}{N}$. Además, se calcula la recompensa total que arroja cada modelo como $\sum_{t=0}^{T} R_t$. En base a esto, se calcula el costo de cobertura promedio resultante para cada modelo junto con su respectiva varianza y desvío estándar.

5. Datos

Como se vio anteriormente, los algoritmos de aprendizaje de refuerzo requieren una gran cantidad de datos dado que se busca que el agente sea capaz de explorar la mayor cantidad de estados del ambiente. Además, se busca que el algoritmo sea capaz de replicar la cobertura que se logra con el modelo de BSM. Debido a esto, se decide realizar el entrenamiento del algoritmo a través de una generación de precios con un movimiento browniano geométrico para el subyacente y con un modelo de pricing de BSM para las opciones europeas. Ya que, por un lado, permite la generación de datos que sea necesaria para el desarrollo del modelo y por otro, sirve como forma de testear en primera instancia que el algoritmo sea válido, asegurando que sea posible obtener los mismos resultados que con la cobertura tradicional que toma el supuesto que el movimiento del subyacente es browniano geométrico. De esta manera, sea $\mu \in \mathbb{R}$, $\sigma > 0$ y $S_0 > 0$, en tiempo continuo $\{S_t : t \geqslant 0\}$, los precios de las acciones subyacentes seguirán el movimiento,

$$S_{t+1} = S_t e^{\left(\mu - \frac{\sigma^2}{2}\right)\Delta t + \sigma\sqrt{\Delta t}\varepsilon}$$

donde S es el precio del subyacente, μ es la tasa de *drift* que bajo probabilidades de riesgo neutrales es igual a la tasa libre de riesgo, σ la volatilidad del activo, Δt el salto temporal y $\varepsilon\sqrt{\Delta t}$ un proceso de Wiener dado $\varepsilon \to N(0,1)$. En base a esto, se puede simular el precio de opciones europeas de compra y venta por medio de las ecuaciones que proponen BSM. En caso de una opción de compra se establece la siguiente expresión,

$$C = Se^{-qT}N(d1) - Ke^{-rT}N(d2)$$

y para una opción de venta se formula como:

$$P = Ke^{-rT}N(-d2) - Se^{-qT}N(-d1)$$

con $d1 = \frac{ln(\frac{S}{K}) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}$ y $d2 = d1 - \sigma\sqrt{T}$.

6. Resultados numéricos

Se busca analizar la convergencia de un algoritmo de aprendizaje de refuerzo a la cobertura de derivados utilizando la delta. Para este propósito, se realizan pruebas para opciones europeas de compra con vencimiento $T = 10$, precio de ejercicio $K = 100$ y manteniendo una posición larga. Además, se define un subyacente que sigue un movimiento browniano geométrico con $S_0 = 100$, volatilidad $\sigma = 20\,\%$, tasa libre de riesgo $r = 0$ y sin repartición de dividendos. Por otro lado, también se testea para el caso de opciones de venta con posición larga, cambiando el vencimiento a $T = 20$, manteniendo el precio de ejercicio a $K = 100$ y con una volatilidad de $\sigma = 25\,\%$ a modo de explorar los resultados del modelo para un derivado con otras características. Por parte de la parametrización del algoritmo, se asigna un conjunto de datos para el entrenamiento de 3.000.000 de simulaciones de precios, mientras que para el testeo se destina una cantidad de 100.000 simulaciones. Luego, se define un máximo de 15 estados posibles en función del precio del subyacente con una cota inferior de \$93 y una cota superior de \$107. El agente será capaz de tomar 11 decisiones de cobertura entre comprar y vender hasta 5 acciones o mantener su posición. Esta elección se debe a que, por un lado, la cantidad de acciones permite al agente explorar diversos estados dentro de una una muestra de datos grande y por otro lado, se expone el modelo a un menor costo computacional. En relación a las demás variables, la tasa de exploración del entorno ε que tiene el algoritmo se fija en $100\,\%$ con un decrecimiento de 0,99 y una tasa mínima de exploración de $5\,\%$. Además, los datos de los estados más actuales tendrán una importancia de $\alpha = 1\,\%$.

En primer lugar, se presentan los resultados con diez opciones de compra europeas con una posición larga. En el siguiente cuadro, se presentan las métricas del costo de cubrirse calculado por el método de P&L, junto con la diferencia entre las posiciones del algoritmo y la cobertura por delta.

	Aprendizaje de refuerzo	**Cobertura delta**
Costo esperado (\$)	0,00567	0,00824
Varianza del costo (\$)	22,37151	18,95236
Desvío estándar del costo (\$)	4,72985	4,35343
Diferencia absoluta promedio de la posición	0,30539	

Cuadro 1: Resultados con opciones de compra

De esta forma, en el cuadro 1 se observa que la cobertura con el aprendizaje de refuerzo tiene en promedio un menor costo esperado, pero posee una mayor variación que con la delta de la opción. Sin embargo, se prueba que hay una convergencia en las posiciones de cobertura que posee el agente en ambos casos con un promedio de diferencia en la posición del subyacente de 0,30539.

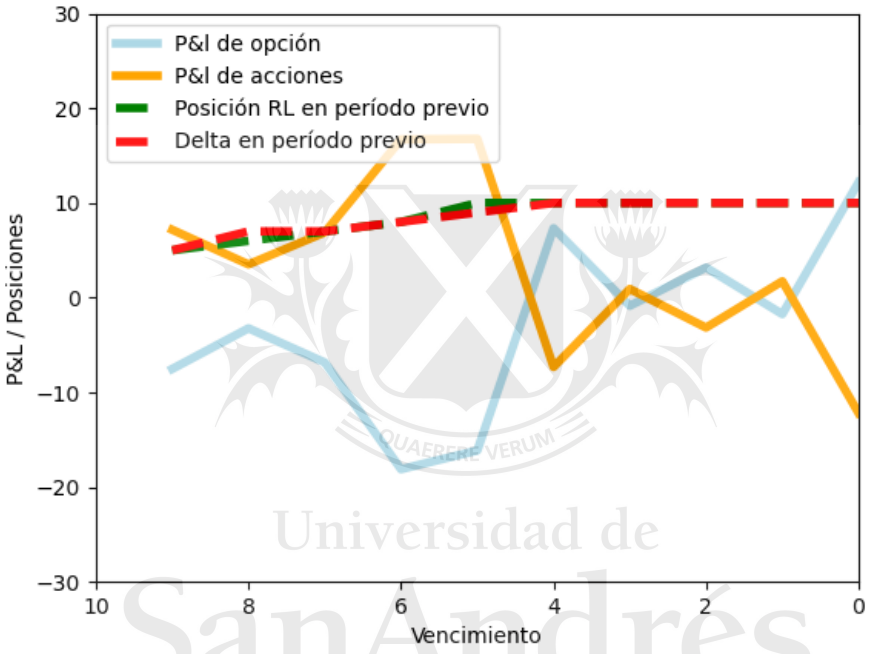

Figura 3: Resultados de última simulación con opciones de compra

A su vez, en la figura 3 se visualizan los resultados de las opciones y acciones de la última simulación del testeo desde el momento $t = 0$ hasta el vencimiento del derivado, junto con las posiciones de cobertura en el subyacente de los dos métodos. De esta manera, se observa que el algoritmo sigue los movimientos de cobertura necesarios ante los cambios de precios que tiene el subyacente. Por otro lado, se visualiza que las posiciones que toma el agente se muestran en convergencia con las obtenidas por el metódo de cobertura con la delta del derivado.

A modo de expandir el análisis de los resultados, se realizan diez testeos del modelo para este caso de estudio. Para estos resultados se calcula el error entre la posición que resulta del aprendizaje de refuerzo con la del modelo

tradicional a forma de analizar la serie de errores.

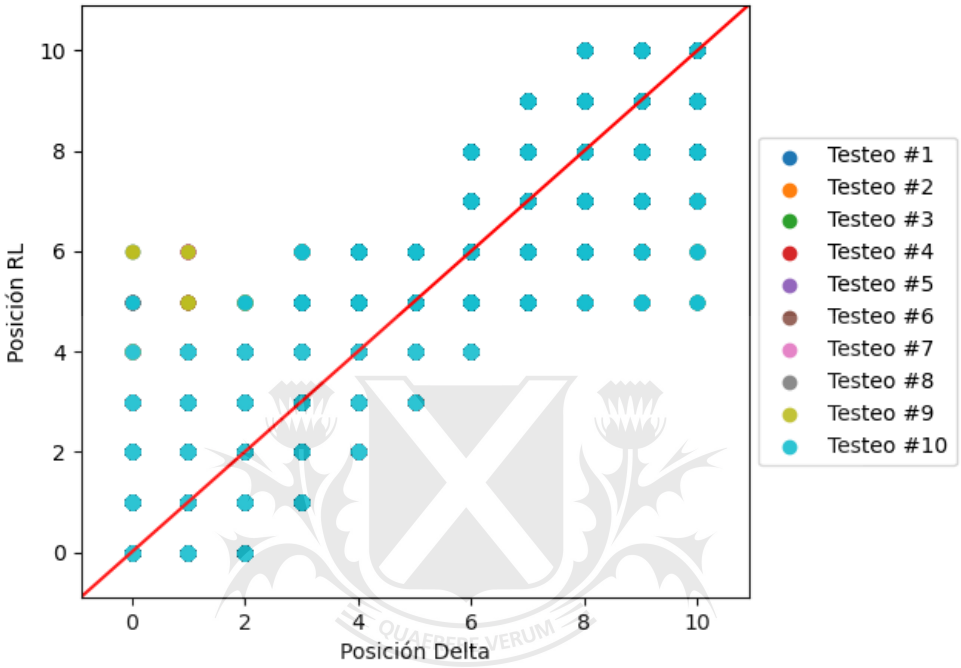

Figura 4: Regresión entre modelos con opciones de compra

Regresión Lineal	
Coeficiente	1,0059
Error cuadrático medio	0,33
Coeficiente de determinación	0,96

Cuadro 2: Resultados regresión con opciones de compra

En la figura 4 se observa el gráfico de regresión entre las posiciones de delta y el modelo para cada testeo, es importante destacar que las posiciones se expresan en enteros representando los resultados que se obtendrían en el mercado, sin tener en cuenta las partes decimales que resultan por BSM. Enfatizando en estos resultados, en el cuadro 2 se presentan los resultados de la regresion lineal entre las observaciones. En estos resultados, en promedio, con un coeficiente de determinación de $0,96$ y un error cuadrático promedio de $0,33$ las posiciones de delta ajustan aproximadamente en un $1,0059$ con

respecto a la posición de aprendizaje de refuerzo.

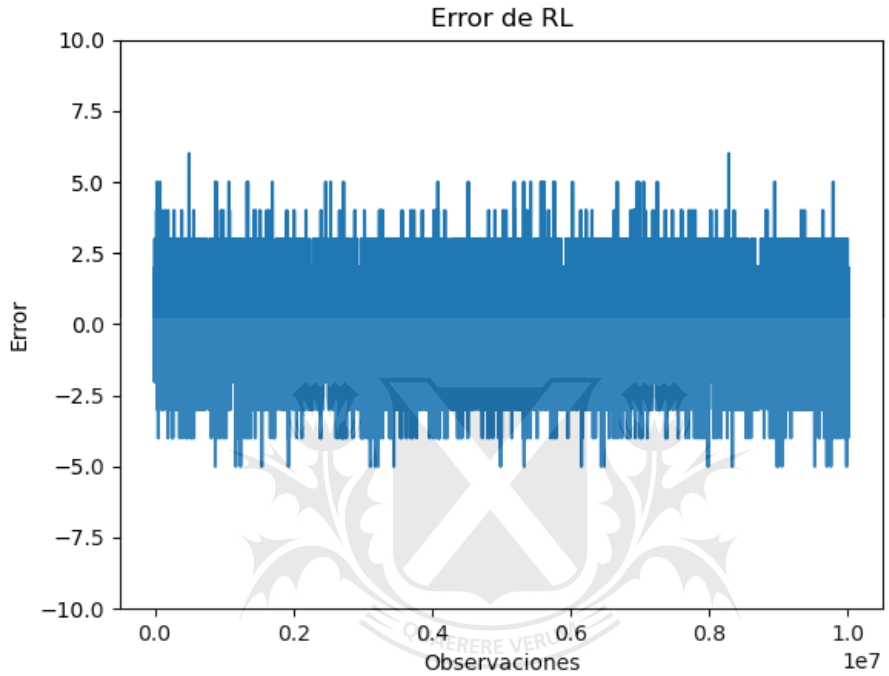

Figura 5: Error con opciones de compra

Dickey Fuller Aumentado			
Estadístico	-939,7376		
P-valor	0		
Valores críticos	1 %	5 %	0 %
	-3,9587	-3,4104	-3,1270

Cuadro 3: Test de raíz unitaria con opciones de compra

En especial, observando el error entre las posiciones en la figura 5 se observa que la diferencia se mantiene cercana a valores entre cero, con un promedio de $0,0021$. Para un análisis más amplio se realiza el test de Dickey Fuller a manera de identificar que la serie es estacionaria, como se puede observar en el cuadro 3 el p-valor resultante es menor a un $0,05$ y $0,01$ expresando que la serie no posee tendencia determinística.

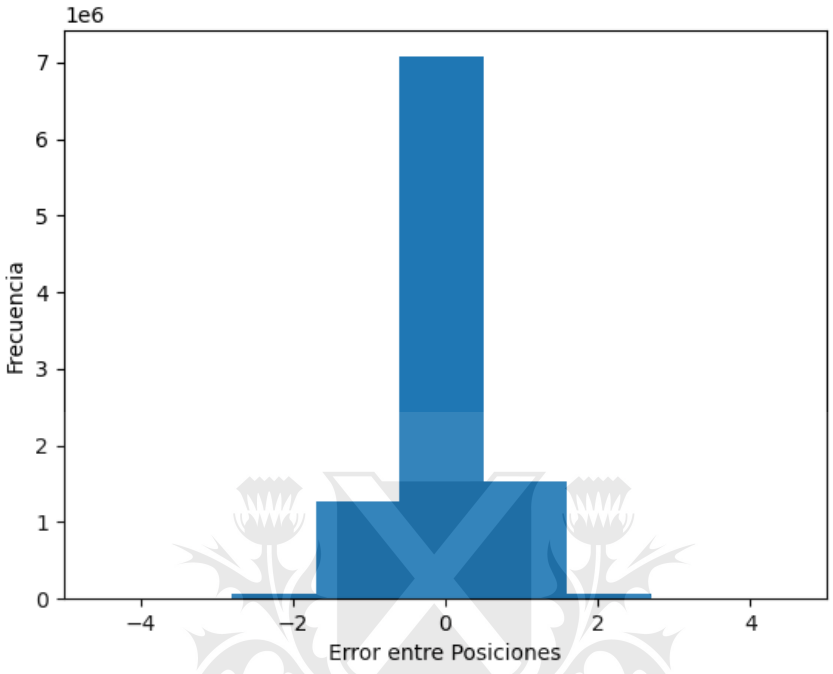

Figura 6: Histograma del error con opciones de compra

Por último, se analiza el histograma del error entre posiciones en la figura 6, se puede destacar que la frecuencia más grande para todos los errores obtenidos para los testeos se encuentran entre un valor de cero, interpretando que en promedio el error llega a ser nulo para una gran cantidad de casos.

En segunda instancia, se presentan los resultados para la cobertura de diez opciones de venta europea con posición larga. Al igual que en el caso anterior, se presentan los resultados del valor monetario del costo de cobertura junto con la diferencia de las posiciones en promedio de ambos modelos.

	Aprendizaje de refuerzo	**Cobertura delta**
Costo esperado ($)	0,0024	-0,0080
Varianza del costo($)	46,0450	32,3312
Desvío estándar del costo ($)	6,7856	5,6860
Diferencia absoluta promedio de la posición	0,736285	

Cuadro 4: Resultados con opciones de venta

De igual manera, se puede comprobar en el cuadro 4 que la varianza es mayor en comparación a la cobertura delta y para ambos casos, el costo esperado se acerca a cero. Además, se observa que la diferencia de posiciones en promedio es 0,736285 mostrando un error promedio absoluto mayor que en el caso anterior.

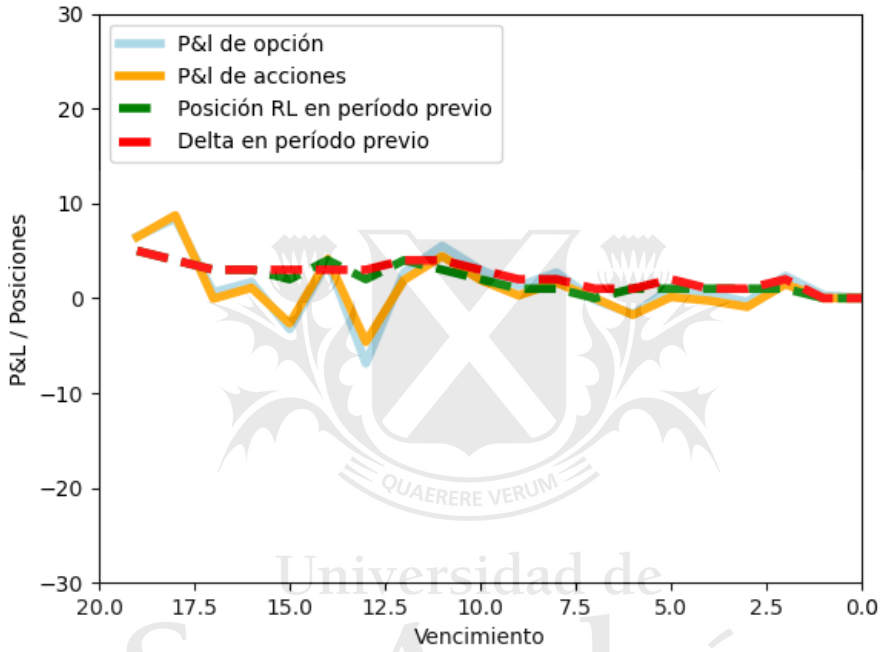

Figura 7: Resultados de última simulación con opciones de venta

Por otro lado, en la figura 7 se grafica las posiciones que mantiene el agente en el subyacente por medio de los dos métodos en la última simulación. Junto con el cuadro 4, se observa la diferencia de posiciones, de forma que la posición que adquiere el agente a lo largo del tiempo entre los dos modelos mantiene similutes. Además, se puede encontrar que la relación de cobertura entre las opciones y subyacente se mantiene incluso utilizando la posición del algoritmo y logra cumplir con la cobertura de la opción.

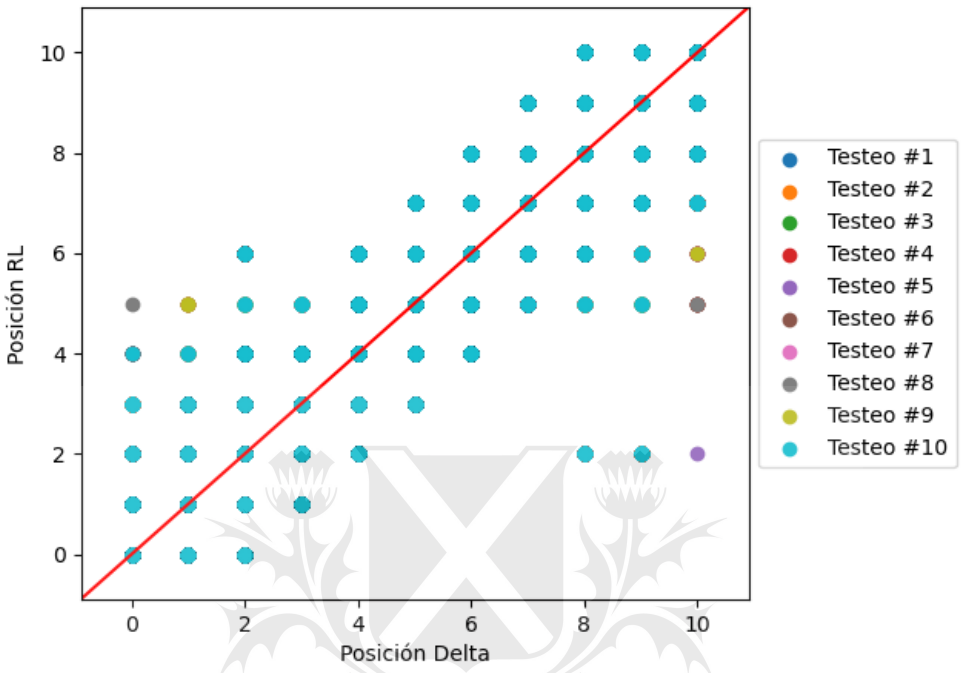

Figura 8: Regresión entre modelos con opciones de venta

Regresión Lineal	
Coeficiente	1,0669
Error cuadrático medio	0,37
Coeficiente de determinación	0,96

Cuadro 5: Resultados regresión con opciones de venta

Ampliando nuevamente el análisis de los resultados, se observan las posiciones del modelo y la cobertura delta en la figura 8, se puede destacar que a lo largo de los testeos las posiciones se mantiene cercanas a la línea de 45 grados. Dada la regresión lineal entre las variables se puede interpretar, con un coeficiente de determinación de $0,96$, que delta explica en $1,0669$ la posición del modelo.

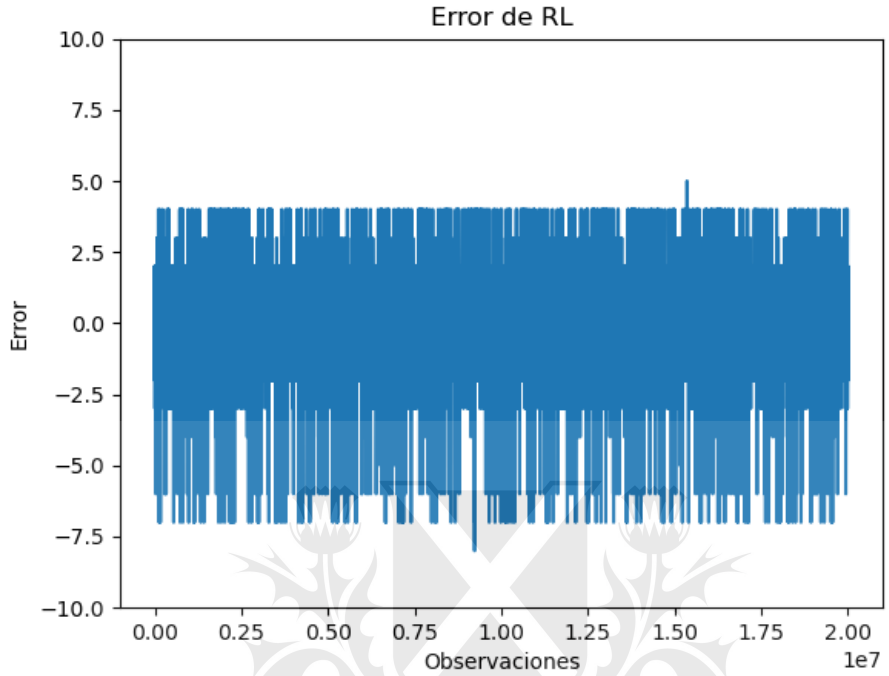

Figura 9: Error con opciones de venta

Dickey Fuller Aumentado			
Estadístico	-1187,3882		
P-valor	0		
Valores críticos	1 %	5 %	0 %
	-3,9587	-3,4104	-3,1270

Cuadro 6: Test de raíz unitaria con opciones de venta

A continuación, se analiza el error entre las posiciones al igual que con los resultados anteriores, en la figura 9 se visualiza una serie estacionaria para los testeos que en relación con el test de Dickey Fuller del cuadro 6 se indica que los errores no siguen una tendencia determinística con un p-valor de cero.

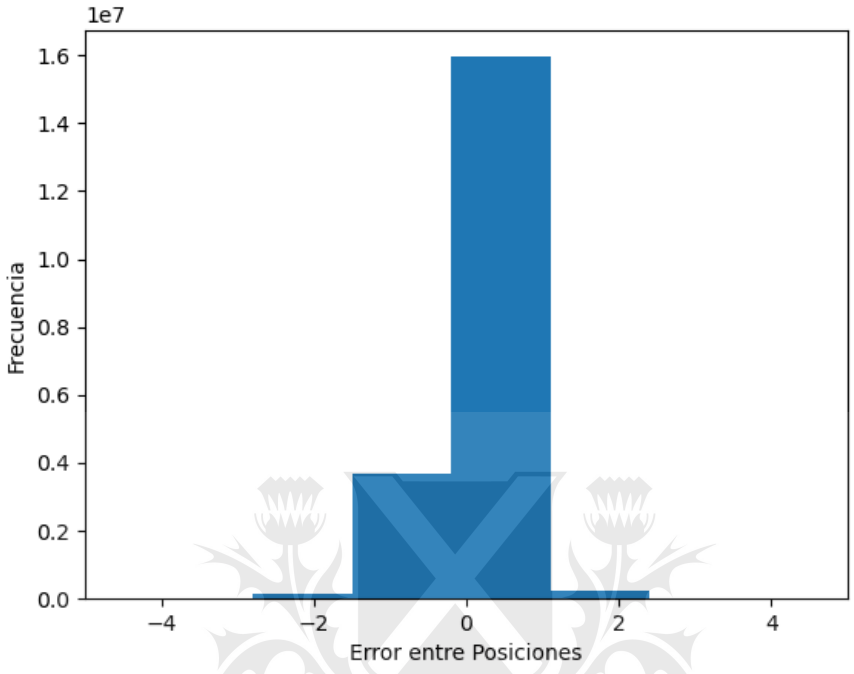

Figura 10: Histograma del error con opciones de venta

Expandiendo con el histograma de frecuencias del error, en la figura 10 se observa que la mayor densidad de observaciones se encuentran cercanas a cero, por lo que las posiciones de los modelos, en promedio, son similares con un error promedio de $-0{,}031$.

7. Conclusión

Al finalizar la descripción del modelo y presentar los resultados obtenidos por este algoritmo de aprendizaje de refuerzo se puede llegar a diferentes conclusiones. Centrándose en los resultados de las posiciones, se observa en las regresiones para ambos casos de derivados que las posiciones entre la delta y la cantidad resultante del algoritmo son similares por lo que podrían replicar el modelo tradicional. Esto puede comprobarse en los resultados con un coeficiente de aproximadamente $1,006$ y $1,06$ para cada caso de la regresión, por lo que se puede intuir que las posiciones del modelo, en promedio, sobreestiman las cantidades de cobertura. Analizando este resultado, esta estimación se puede deber al alcance que tiene el entrenamiento con los estados más extremos del ambiente, al contener menos información de estos valores el algoritmo tiende a sobreestimar la posición óptima.
Se tiene en cuenta el gráfico de las posiciones y recompensas por P&L para cada escenario en la figura 3 y 7, en particular, se puede observar que utilizando las posiciones de aprendizaje de refuerzo es posible realizar una cobertura de los derivados. A su vez, se observa que a lo largo del tiempo hasta el vencimiento, la posición del modelo es capaz de mantenerse cercana a la delta de las opciones.
Observando los errores de los resultados, se pueden analizar las figuras 5 y 9 junto a los tests de raíces unitarias de los cuadros 3 y 6. Se concluye que el error no posee tendencia determinística y se mantiene en errores cercanos a 0, por lo que en promedio las posiciones convergen a los mismos valores. Al visualizar con las figuras 6 y 10 las frecuencias de los datos para cada caso, se logra reconocer que la mayor densidad de datos se encuentra en valores de cero, interpretando que para la mayoría de testeos el algoritmo logra encontrar la posición de cobertura semejante a la delta de las opciones.
Al centrarse en los resultados pertinentes al costo de cobertura, con los cuadros 1 y 4 se observa que para el testeo, en general, se logra alcanzar un costo de cobertura esperado cercano a cero. En especial, en el caso de las opciones de compra incluso se muestra un mejor resultado que el tradicional. Sin embargo, también se evidencia en ambos casos una mayor variabilidad del costo del modelo en relación con su contraparte.
Por otro lado, hay que destacar que este modelo tiene ciertas debilidades, en un principio se debe tener en cuenta la necesidad de repetir el entrenamiento y luego la muestra de datos que se precisa. Como se menciona, en primer lugar, es necesario un entrenamiento previo para cada derivado que se busque cubrir, es decir, los valores Q generados para un derivado en un principio no son útiles para otro que posea diferentes características. Luego, en segundo lugar, para aplicar este modelo a un caso real se debe tener en

cuenta que requiere una gran muestra de datos para ser entrenado de forma que se pueda asegurar que el agente es capaz de explorar diferentes estados del ambiente para llegar al valor más apropiado para cada situación.
En resumen, se puede concluir que el algoritmo es capaz de replicar las soluciones para cubrir opciones europeas del modelo tradicional presentando algunas variaciones en el costo de cubrirse. Por lo tanto, el modelo es candidato para ser aplicado a otros casos de derivados. Por último, cabe destacar que la muestra de datos para el entrenamiento y la necesidad de repetir el entrenamiento para diferentes derivados son desventajas que presentan este algoritmo de aprendizaje de refuerzo.

Referencias

Bellman, R. (1957). A Markovian decision process. *Journal of mathematics and mechanics*, 679-684.

Buehler, H., Gonon, L., Teichmann, J., & Wood, B. (2019). Deep hedging. *Quantitative Finance*, *19*(8), 1271-1291.

Buehler, H., Gonon, L., Teichmann, J., Wood, B., Mohan, B., & Kochems, J. (2019). Deep Hedging: Hedging Derivatives Under Generic Market Frictions Using Reinforcement Learning. SSRN Scholarly Paper ID 3355706. *Social Science Research Network, Rochester, NY*.

Cao, J., Chen, J., Hull, J., & Poulos, Z. (2021). Deep hedging of derivatives using reinforcement learning. *arXiv preprint arXiv:2103.16409*.

Hull, J. C. (2020). Machine learning in business: An introduction to the world of data science. *(No Title)*.

Kolm, P. N., & Ritter, G. (2019). Dynamic replication and hedging: A reinforcement learning approach. *The Journal of Financial Data Science*, *1*(1), 159-171.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Watkins, C. J. C. H. (1989). Learning from delayed rewards.