Universidades de Burgos, León y Valladolid

Máster universitario

Inteligencia de Negocio y Big Data en Entornos Seguros

TFM del Máster Inteligencia de Negocio y Big Data en Entornos Seguros

Optimización de Ejecución de Órdenes en Mercados Financieros con Deep Learning

Presentado por Raúl Gallardo Pérez
en Universidad de Burgos — 4 de septiembre de 2020
Tutor: Bruno Baruque Zanón y
Juan José Rodríguez Díez

Resumen

El problema que se va a tratar es la optimización de la ejecución de órdenes en los diferentes mercados financieros. Esto se traduce en qué tiempo y forma se comprará o venderá un activo financiero.

Esta optimización se realiza aproximando mediante aprendizaje por refuerzo de forma tabular y posteriormente realizando una aproximación con Deep Learning.

Los activos que se van a usar son acciones de empresas europeas, aunque este mismo enfoque se pueda realizar en cualquier activo financiero (acciones, futuros, opciones) y cualquiera que sea el subyacente (empresas, índices bursátiles, renta fija, divisas o materias primas).

Descriptores

Optimización de ejecución de órdenes, Volume Weighted Average Price, VWAP, Percent On Volume, POV, Aprendizaje Profundo, Aprendizaje por Refuerzo, Aprendizaje Automático, activos financieros, libro de órdenes limitadas.

Abstract

The problem to be dealt with is the optimization of order execution in the different financial markets. This translates into what time and how a financial asset will be bought or sold.

This optimization has been carried out using Reinforcement Learning with a tabular way and later making an approach with Deep Learning.

The assets to be used are equities from European companies, although this same approach can be carried out on any financial asset (stocks, futures, options) and whatever the underlying asset (equities, stock indices, fixed income, currencies or commodities).

Keywords

Order execution optimization, Volume Weighted Average Price, VWAP, Percent On Volume, POV, Deep Learning, Reinforcement Learning, Machine Learning, financial assets, limit order book.

Índice general

Índice de imágenes

Índice de tablas

Parte I

Introducción y descripción del proyecto

Introducción

Tanto los gestores de fondos de inversor, los gestores de planes de pensiones como, de forma general, cualquier inversor en los mercados financieros se presenta ante tres cuestiones principalmente:

- Qué activo financiero comprar o vender
- Cuánto comprar o vender de ese activo
- Cómo minimizar el coste de la transacción

Este Trabajo Fin de Máster se centrará en este tercer punto, por tanto, en ningún caso se tomarán decisiones de qué comprar o cuánto comprar, sino sólo la ejecución y transmisión de estas decisiones en pro de reducir los costes asociados.

Para llevar a cabo esta tarea, lo que haremos será diseñar un algoritmo que decidirá cómo de pacientes seremos en la transmisión de las órdenes (segundos, minutos o días). En el análisis realizado por la gestora AQR [10], se establecía que el tiempo medio de ejecución era inferior a un día, lo que analizando el tamaño de la gestora es realmente poco tiempo.

AQR, la gestora de fondos liderada por Clifford Asness, a fecha de 30 de julio de 2020, tiene unos 143.000 millones de dólares bajo gestión [3], denominado comúnmente *Assets Under Management* o simplemente *AUM*. Teniendo en cuenta estos números, desde el estudio realizado por Frazzini et al. [10] nos está diciendo que una gestora de este tamaño está tardando de media un día en hacer o deshacer una posición, entendiendo por posición la compra o venta de un activo financiero. Para llevarlo a un mundo más cercano a todos nosotros, es como si quisiéramos comprar o vender una

vivienda y tardáramos de media un día encontrar vendedor o comprador respectivamente y realizar la transacción.

Para llevar a cabo esta tarea se han obtenido cotizaciones en tiempo real de los principales activos financieros que a la vez nos sirvan para generalizar las conclusiones a las que se llegarán.

Según cómo clasifiquemos un activo financiero puede caer dentro de diferentes clasificaciones de categorías. Las clasificaciones que se pueden encontrar son *clase de activo*, *localización geográfica*, *subyacente* y si los datos son referidos a empresas podemos clasificarlas también base a su *capitalización bursátil*.

Para explicar estas clasificaciones, podemos tener el ejemplo de la empresa *Apple* y del denominado *Bund*. El análisis sobre Apple serían *acciones*, *americanas*, *renta variable* de *alta capitalización* ó *large cap*. Por otra parte, si se analiza el Bund[1] se tratarían de *futuros*, *europeos (alemanes)* y *renta fija*.

1.1. Motivación y alcance del proyecto

Es preciso, en pro de comprender el desarrollo del presente documento, entender perfectamente de lo que estamos hablando y que en ocasiones no es tan sencillo como parece, para ello, se hará uso de ciertos ejemplos que nos guiarán el camino a abstracciones mayores en los diferentes activos financieros que serán sujetos de estudio.

Imaginemos que trabajamos para una empresa que se dedica a la compra y venta de viviendas, la cual está deseando adquirir 100 viviendas teniendo la posibilidad de comprarlas en los siguientes dos lugares:

1. Pueblo de 500 habitantes en la provincia de Zamora
2. Una gran ciudad, véase, Londres

Pensemos inicialmente en la localidad de la provincia de Zamora. Seguramente haya alguna casa vacía, digamos que se encuentren 5 viviendas en venta, ¿qué ocurre con las 95 restantes? Lo más probable es que el precio

[1] El Bund es la renta fija emitida por el gobierno alemán con una duración de 10 años. La rentabilidad de este producto es la referencia sobre la que se calculan las primas de riesgo (diferencia entre el Bund y la renta fija local). A su vez, sirve de subyacente para derivados financieros como los futuros y las opciones.

medio de compra comience a subir debido a que estaremos comprando viviendas que en un principio no estaban en venta.

Si ahora pensamos en la compra de esas 100 viviendas en una gran ciudad, a pesar de que haya porcentualmente un menor número de viviendas disponibles, nos será muy sencillo encontrar estas viviendas a la venta y no modificaremos el precio medio de la compra en la localidad.

Más formalmente, lo que estamos atendiendo es a la presencia de lo que en microeconomía se denominada *curva de oferta-demanda*. Ésta, como se puede ver en la imagen 1.1, lo que relaciona en el supuesto de una economía con un único producto es el precio P de un producto al que estarán dispuestos los oferentes a vendernos una cantidad Q de dicho producto. En este primer ejemplo se trataría de la *curva de oferta*, mientras que la relación entre el precio P que estarían dispuestos los compradores por una cantidad Q del producto es lo que se denomina *curva de demanda*. En el punto donde se cruzan ambas curvas es el llamado *punto de equilibrio* y es dónde se cruzan las compras y las ventas.

Si seguimos observando la imagen 1.1, podemos analizar cómo la cantidad Q_s que los oferentes están dispuestos a poner en el mercado es sustancialmente menor que la cantidad Q_d que los demandantes estarían dispuestos a comprar para un mismo precio P_q.

Esta es la base teórica por la que según necesitamos una mayor cantidad de un activo de forma general, o de una compañía de forma particular el precio medio de la compra irá aumentando. Si analizamos de nuevo la curva de la oferta tiene una pendiente positiva, generando un coste marginal ascendente por cada nueva acción que necesitemos en cartera.

Hemos hecho este análisis desde el punto de vista de un comprador de acciones, pero, lo mismo sucede si nuestra intención es deshacer una posición y vender total o parcialmente nuestra cartera de acciones de una empresa determinada. Por esta razón, resulta evidente que si tenemos fijos los demandantes de una empresa el precio medio ponderado de una venta de 100 acciones será igual o superior a si quisiéramos vender 1000 acciones.

Esta es la idea más importante sobre la que nos apoyaremos. Hay que reincidir en que el objetivo será minimizar el precio medio, minimizar el impacto en el mercado o adaptarnos y aprender una función que determine la adquisición o liquidación de posiciones, de una compra de un activo financiero. Lo contrario aplica si lo que se pretende es liquidar una posición.

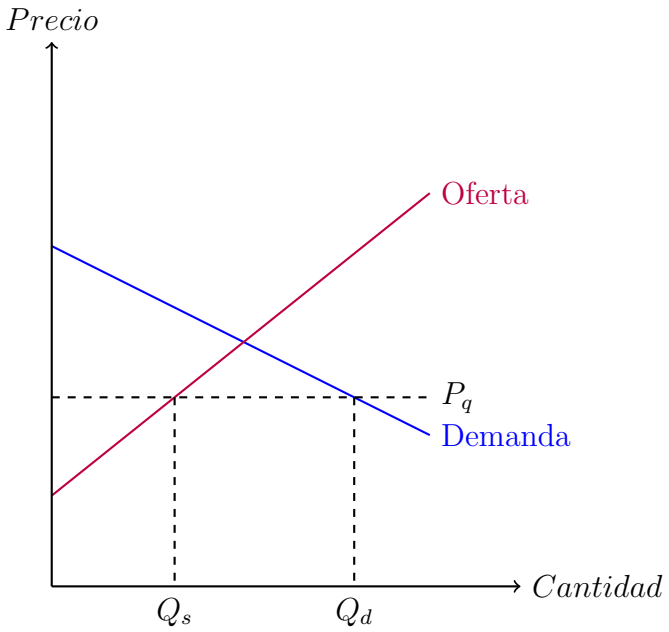


Imagen 1.1: Oferta y demanda en función de la cantidad demandada y ofertada

En otras palabras, minimizar el coste de la operación, en lo que se denomina *impacto en el mercado*[2].

[2]Este es el coste que queremos minimizar, aunque hay otros costes asociados en la compra-venta y mantenimiento de una serie de activos en cartera. Estos costes pueden resumirse en brokerage (lo que nos cobra el intermediario por procesar nuestra orden), mercado (lo que cobra la bolsa dónde cotiza el activo, por ejemplo, BME (Bolsas y Mercados Españoles) sería el que cobrara esta comisión de querer comprar acciones del Banco Santander en la bolsa de Madrid), liquidación (coste por realizar los apuntes contables poniendo a las partes compradora y vendedora de acuerdo), la custodia (coste asociado a mantener una posición en cartera) y por último los impuestos donde podríamos incluir la Tasa Tobin que en España no es de aplicación, pero, en Francia sí, gravándose cada compra-venta en un porcentaje en base al nominal o cantidad de dinero que mueve la operación.

Objetivos del proyecto

Como ya se ha visto en la Introducción, el objetivo del proyecto es minimizar una función de coste. Esta función puede ser el precio medio de ejecución al que ha sido adquirido el activo financiero, el impacto en el mercado (no recorriendo la curva de oferta en nuestra contra) o un algoritmo de ejecución. En este proyecto no se trabajará con la decisión de compra o de venta desde un punto de vista de *Portfolio Manager*, sino que se hará sólo la *optimización de las ejecuciones*.

Para llegar a esta optimización, el proyecto se basará en las técnicas usadas históricamente como el *POV* (Percent On Volume), *VWAP* (Volume Weighted Average Price o Precio Medio Ponderado por Volumen).

Cartea et al. en [6] exponen el funcionamiento de las bolsas las cuales se encargan de poner en contacto a inversores pacientes y proveedores de liquidez que transmiten órdenes pasivas al Limit Order Book (LOB en adelante), como compensación reciben el spread bid-ask. Para entender esta compensación nos vamos a basar en la imagen 2.2.

Imaginemos que tenemos un mercado con solo dos agentes donde el primero quiere adquirir de manera agresiva 1 acción del banco Santander para en ese mismo instante venderla, mientras que el segundo agente lo único que hace es cumplir sus deseos proveyendo de la acción del Santander cuando compra y adquiriendola posteriormente.

Si nos fijamos de nuevo en la imagen 2.2 observamos que el primer nivel de precios al que se pueden adquirir acciones del banco es a 2.485 € (primer nivel del ask) y el primer precio al que se pueden vender es a 2.483 €. La diferencia entre estos dos precios es lo que se denomina spread bid-ask (0.002 €), es lo que le cuesta al primer agente realizar esta operación, y es por ende, lo que gana el segundo agente. Este segundo agente es lo que se denomina

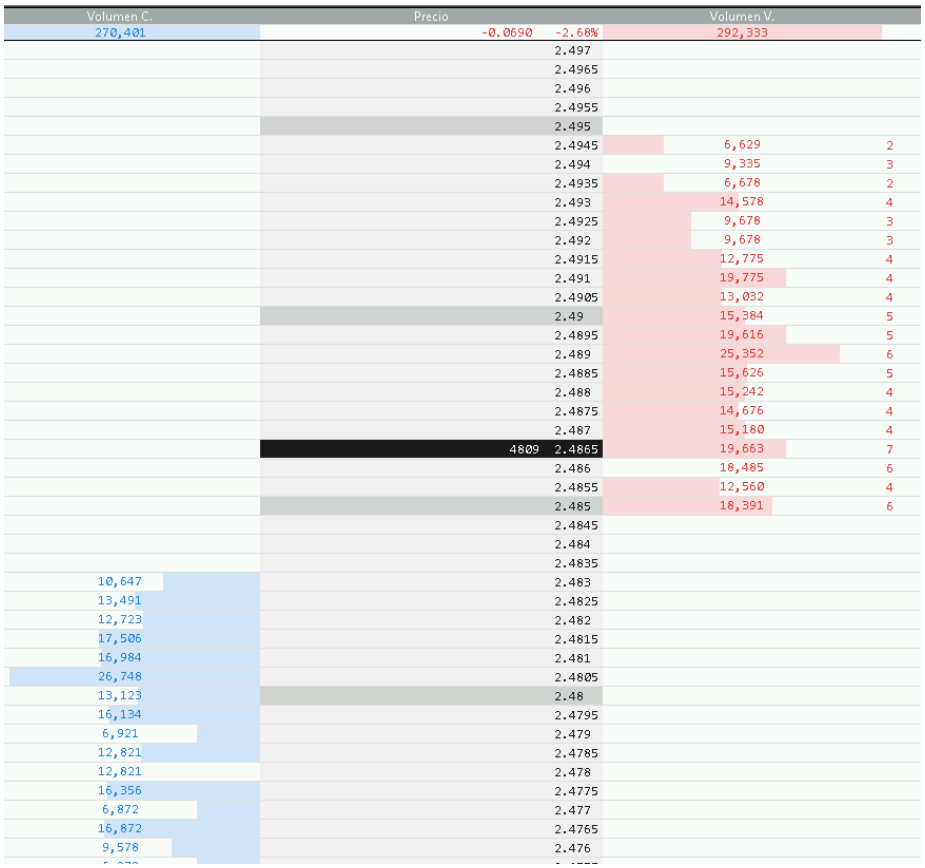


Imagen 2.2: Libro de órdenes del Banco Santander

creador de mercado o market maker, que volviendo al ejemplo inmobiliario, es como si las inmobiliarias no sólo se dedicaran a la comercialización de casas, sino que pudieran tener en su propio stock viviendas, facilitando tanto la compra como la venta de inmuebles.

Lo que se puede concluir de lo anterior es que un mayor spread se traduce en mayores costes de ejecutar órdenes a mercado, mientras que incentiva proveer liquidez mediante órdenes límite[3]. Por esta razón, la amplitud del spread es un buen indicador, mas no el único, de los costes de transacción. Otro de los componentes más importante es la profundidad en cada nivel de

[3]Cuando se habla de proveer de liquidez a un mercado consiste en colocar órdenes límite en el LOB. Se habla de proveer liquidez ya que cuando un partícipe quiere adquirir (o deshacerse) de un activo financiero cuantas más posiciones haya en el LOB más fácil, rápido y barato le será a éste. La liquidez de un activo financiero es la facilidad (velocidad y coste) de convertirlo por dinero, por ejemplo, una vivienda es mucho menos líquida que una acción de Apple o que una cuenta bancaria.

precios en el LOB, que nos muestra el número de participantes y el tamaño de las órdenes aquí colocadas por éstos.

Como se ha explicado con el ejemplo de las viviendas en una pequeña localidad de Zamora o en una gran ciudad, trabajar con grandes órdenes que son transmitidas a mercado de una vez se traduce en *comerse* todo el volumen de uno o más niveles, incrementándose el coste medio ponderado de la operación.

Cuando contamos con inversores impacientes, que necesitan adquirir o liquidar su inventario de manera más o menos inmediata, la primera estrategia que se puede tomar (ya que no quieren o no pueden esperar que se ejecuten sus órdenes a precios limitados), es dividir la posición en diferentes momentos del tiempo. Esto significa que si quisieran comprar 1000 acciones, podrían comprar 100 cada minuto durante 10 minutos para no arrastrar el precio y además dar tiempo al resto de participantes a colocar órdenes límites en el LOB.

Una vez entendida la lógica, es importante tener un Benchmark como referencia. Los Benchmark más usados son:

- Precio de llegada: precio al que está cotizando el activo cuando se lanza la orden padre[4]
- Time-Weighted-Average-Price (TWAP): precio medio del activo durante el intervalo de tiempo en el que se realiza la adquisición o liquidación de la posición
- Volume-Weighted-Average-Price (VWAP): precio medio ponderado por volumen durante un tiempo un intervalo de tiempo[5]

Es preciso diferenciar el TWAP del VWAP ya que son conceptos muy parecidos, mas, no iguales. Supongamos que en un franja de tiempo tenemos dos operaciones cruzadas, una de 9 acciones a 5 € y otra de 1 acción a 10

[4]Se habla de orden padre a la orden de compra o de venta que se va a tramitar. A su vez, esta orden padre puede dividirse en órdenes hijas que tramitarán la decisión tomada en la padre.

[5]Con el VWAP se atienden a dos acepciones distintas que pueden llevar a confusión. A nivel de cálculo, el VWAP es el precio medio ponderado por volumen. Se traduciría en el precio medio de todos los agentes que han intervenido en una franja de tiempo. Por otra parte, se le refiere también a un algoritmo de ejecución de órdenes, en cuyo caso lo que se hace es estimarse el precio medio ponderado por volumen que debería ejecutarse en base a un histórico.

€. El TWAP sería $(5 + 10)/2 = 7{,}5$€, mientras que en el caso del VWAP el resultado sería $(5 * 9 + 10 * 1)/10 = 5{,}50$€.

Tanto la adquisición como la liquidación de las posiciones se realizan con órdenes con tamaño la unidad. En el caso de los futuros, esto puede ser de aplicación directa en la operativa ya que el coste es lineal, sin embargo, las acciones de empresas puede estar sujetas a mínimos. Este mínimo puede ser, por ejemplo, 0,10 % sobre el nominal con un mínimo de 5 €. Esto implicaría que la compra de una acción que esté cotizando a 5 € duplicaría su coste de adquisición directamente. Por simplicidad estos costes no han sido incluidos, pero, de cara a ponerlo en producción es necesario tenerlo en cuenta.

Teniendo en cuenta este marco base, los objetivos que tendrá el trabajo fin de máster son:

Objetivo	**Descripción**
OBJ-01	Minimizar el precio medio de adquisición mediante un algoritmo de aprendizaje por refuerzo de un activo financiero
OBJ-02	Aprender el comportamiento de algoritmos de ejecución, más concretamente el Percent On Volume (POV) y el Volume Weighted Average Price
OBJ-03	Implementar los objetivos OBJ-01 y OBJ-02 de forma tradicional (tabular) y mediante el uso de redes neuronales
OBJ-04	Que los modelos sean capaces de funcionar en activos en los que no han sido diseñados

Tabla 2.1: Objetivos del proyecto

Conceptos teóricos

Este Trabajo Fin de Máster aúna dos mundos, por un lado, la ciencia de datos y por otro los movimientos y características propias de los mercados financieros, siendo ambas partes igual de importantes. Primero veamos ciertas características de estos mercados financieros.

3.1. Identificación del problema

Una vez entendido el problema de optimización que se quiere llevar a cabo, la primera pregunta que surge es, ¿cómo se puede minimizar este coste? De nuevo volvamos a la intuición primero, para ser desarrollada después. Si pensamos en el problema anterior de la compra de las 100 viviendas, ¿es lo mismo comprar esa cantidad de viviendas según vayan saliendo al mercado, por ejemplo, durante 2 años, que si pretendemos comprarlas todas de golpe? Evidentemente, no. En el mercado inmobiliario puede suceder lo siguiente, resulta que estamos en ese pueblo de 500 habitantes de Zamora donde hay 5 viviendas vacías y que podemos comprar directamente, dentro de dos meses uno de los habitantes puede encontrar trabajo en Valladolid y poner su casa a la venta (y nosotros comprarla) y a los 6 meses que fallezca otro de los habitantes y que sus herederos pongan la vivienda a la venta, que nosotros nuevamente compraremos. Así sucesivamente hasta que adquiramos el total de las viviendas deseadas.

¿Ha cambiado el mercado inmobiliario del pueblo como ocurriría si se construyeran nuevas viviendas? No. ¿Ha cambiado el precio de la vivienda durante este periodo de tiempo? Si aplicamos el ceteris paribus, tampoco. Con las acciones de una empresa en un mercado financiero ocurre exactamente lo mismo, si nosotros queremos comprar 1000 acciones, el impacto

en el mercado (con su coste asociado) es menor si compramos 100 acciones, esperamos unos minutos, volvemos a comprar otras 100 acciones y así sucesivamente hasta terminar de adquirir las acciones deseadas, que si hacemos esta misma transacción de golpe. Esta será nuestra primera forma de optimizar el coste, ir entrando paulatinamente en el activo financiero deseado dividiendo la orden a lo largo del tiempo, dando posibilidad a que nuevos vendedores acudan al mercado a satisfacer sus necesidades.

Por tanto, existen dos técnicas para minimizar el coste:

- Esperar a nuevos participantes
- Colocar órdenes a un precio más ventajoso y esperar a que la orden se ejecute a nuestro precio mejorado

Para ilustrar la segunda de las técnicas, pongamos en un suponer que la vivienda promedio (asumiendo que todas son iguales y no tenemos preferencia por ninguna), está valorada en 100.000 €. Como sabemos, si compráramos las 100 viviendas arrastraríamos el precio, obteniendo un precio medio superior a esos 100.000 €, pero, ¿qué ocurriría si nos ponemos un *límite* de compra de 95.000 €?

Si el comprador pusiera ese límite en 95.000 € lo que ocurriría es que inicialmente no tendría ninguna vivienda en su posesión y que tardaría en cruzar su orden. Una posibilidad, es que hasta que no bajara el precio de las viviendas, alcanzando los 95.000 € no se ejecutaría la compra. El resultado, por tanto, es que por un lado se ha tardado más en realizar la compra y por otra parte, que de haber habido alguna subida de precios no solo no tendría la vivienda al precio original, sino que de demandarla ahora pagaría aún más. Este último punto es lo que se denomina *riesgo de mercado* y es el riesgo inherente a los cambios de precio[6].

Como es evidente, cuanto más lejos coloquemos la orden límite menor probabilidad tendremos de que nuestra orden se cruce, pero, mayor será la ventaja ya que tendremos un precio medio de ejecución más favorable. Esta operativa además cuenta con otra cuestión muy importante.

Volvamos a nuestro mundo inmobiliario y hagámonos la siguiente pregunta: ¿Qué ocurre si ponemos una orden de compra a 95.000 € y hay

[6]El riesgo en los mercados financieros no solo tratan de posibles pérdidas, sino, que también incluyen la incertidumbre. En el caso del riesgo de mercado hablaríamos del riesgo asociado a los cambios de precios, los cuales pueden ser tanto positivos (generen beneficios), como negativos (generen pérdidas). Evidentemente, a estos movimientos negativos son a los que hay que prestar una mayor atención

otro comprador previo a nosotros a ese mismo precio? Lo que ocurre es que la primera venta la asume el primer comprador y nosotros quedamos a la espera de que haya más vendedores.

Esto significa que solo podremos tener certeza de que una orden se ha ejecutado cuando nuestro precio ha sido traspasado. El corolario ante la anterior afirmación es que en función del número de compradores o vendedores por cada nivel de precio, nos será más sencillo que se ejecute nuestra orden. El porcentaje de órdenes que se ejecutan cuando el precio ha tocado nuestro precio se denomina *fill rate* estando acotado inferiormente por 0 (no se cruza ninguna orden) y el 100 % (se cruzan todas). Este fill rate es inversamente proporcional al número de posiciones en cada nivel de precios.

Si analizamos este comportamiento nos encontramos con una función de beneficio con los siguientes términos:

- Como término **negativo**
  - El riesgo de que la orden no se ejecute al ir el precio en contra
  - El tamaño de la cola nos supone un coste ya que tenemos que tener en cuenta el *fill rate*
- Como término **positivo**
  - La mejora en el precio medio de ejecución hace que nuestros gastos operativos disminuyan. Este valor va a depender de qué porcentaje del inventario se ejecute con órdenes límite (y a qué precio) y qué porcentaje se ejecute con órdenes a mercado

Si lo analizamos matemáticamente, tenemos la distancia al valor al que esté cotizando el activo a la que se va a lanzar la orden D_n, la probabilidad de que la cotización sobrepase el nivel de precios al que hemos colocado la orden P_N (denotado con N al ser superado el precio), la probabilidad de que la cotización iguale (sin ser rebasadado) el nivel de precios al que se ha colocado la orden P_n y por último, el ya citado fill rate F_n del nivel de precios. Con estos elementos podemos observar como la probabilidad de ejecución es:

$$P_e = P_N + P_n F_n \tag{3.1}$$

Como se ha visto, si un agente adquiere o liquida de su inventario una cantidad Q de un activo financiero al precio al que cotiza, este se ve impactado. Por otra parte, existe la posibilidad de ejecutar a precios más beneficiosos, a los que hay que incluir los riesgos de mercado. Esta función de optimización podría resumirse como:

$$B_e = D_n P_e \tag{3.2}$$

Donde B_e es el beneficio de la ejecución.

Tipos de órdenes

En los mercados financieros vamos a encontrar principalmente tres tipos de órdenes, donde por simplicidad se explicarán solo desde el punto de vista de la compra, aunque lo mismo aplica para la venta. Estas órdenes son las siguientes:

- **Órdenes agresivas**
  - Órdenes a mercado: se trata de órdenes que compran la cantidad deseada del activo financiero al precio al que estén.
  - Órdenes a stop: se trata de un tipo de órdenes que compra la cantidad deseada del activo financiero a partir de un precio dado que está a un precio menos favorable (más alto que al precio al que esté cotizando el activo si se habla de una compra, o más bajo en el caso de las ventas)
- **Órdenes pasivas**
  - Órdenes a límite: se trata de órdenes, que al revés que las órdenes stop, tienen un precio de ejecución más favorable que el precio de cotización (más bajo que el precio de cotización para las compras y más alto para las ventas)

Para los algoritmos de ejecución se utilizan bien sean las órdenes a mercado o bien las órdenes límites. En este trabajo nos centraremos en las órdenes a mercado.

Recapitulando, tenemos como incógnita, dato a minimizar, o al menos como punto de referencia el impacto de mercado. Este impacto puede llegar a ser estimado en las órdenes agresivas, pero, con las órdenes limitadas hay

que añadir el fill rate y la probabilidad de alcanzar la orden en función a la volatilidad del activo y la distancia a la que se haya colocado estar orden. Por esta razón, sólo se van a utilizar órdenes a mercado para realizar la investigación.

Limitaciones y particularidades

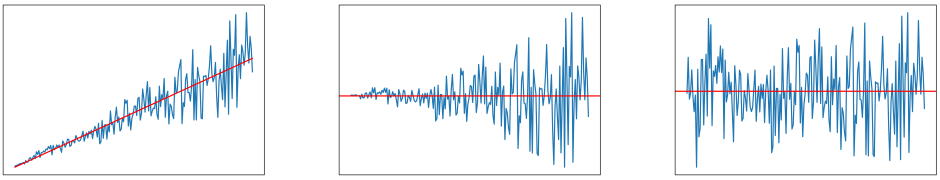
Hasta aquí se ha estudiado toda la información de la que disponemos y sus tremendas bondades, pero, al menos igual de importantes son éstas que las limitaciones inherentes de los modelos que vamos a construir.

Por contra a las ciencias exactas, las ciencias sociales tienen un comportamiento que no es siempre el mismo, dificultando los modelos que podemos construir para las diferentes tareas que podemos desarrollar. Los principales problemas que vamos a encontrarnos a la hora de intentar realizar estos modelos son:

- Datos no estacionarios
- Distribuciones no normales
- Rellenado y reacción de los diferentes niveles de precios ante nuevos agentes

Una extensa bibliografía y estudios soportan la idea de que los datos financieros son no estacionarios. Según [30] una serie de datos r_t es *estrictamente estacionaria* cuando la distribución conjunta de (r_{t_1}, r_{t_k}) es idéntica a (r_{t_1+t}, r_{t_k+t}) para todo t siendo k la media y la varianza de los datos es constante en una población. En otras palabras, que una serie de datos sea estrictamente estacionaria se traduce en que tanto medias como varianzas sean invariantes ante un desplazamiento temporal, lo que es lo mismo, que sean constantes a lo largo de dicha serie temporal.

Al ser esta condición muy difícil de encontrar, también se pueden encontrar series *débilmente estacionarias* a las cuales símplemente se las va a exigir que su media sea constante para cualquier momento temporal. En finanzas es típico asumir datos débilmente estacionarios y será lo que se haga en este trabajo. Por el contrario, una serie no estacionaria no mantiene ni medias ni varianzas constantes. Para visualizar mejor el concepto, podemos observar los tres tipos de series de datos en la imagen 3.3.



(a) No estacionaria (b) Débilmente estacionaria (c) Estacionaria

Imagen 3.3: Series de datos

3.2. Mercados Financieros

Dada la naturaleza de los mercados financieros, y su evidente posibilidad de generar beneficios, éstos han sido objeto de estudio durante décadas en búsqueda de la posibilidad de ser predichos. Estas predicciones tienen como principales objetivos la adivinación de la dirección del movimiento (si subirá o bajará la cotización de un activo), el tamaño del movimiento, la volatilidad (entendida como desviación estándar de los retornos) o el momento en el que se realizarán estos movimientos. A pesar de los recursos asignados a esta tarea adivinatoria, los resultados no siempre han sido los esperados.

Dos de las principales ideas que subyacen bajo el movimientos de estos activos son los conocidos como martingalas y el paseo aleatorio o random walk. Los mercados financieros están marcados por un gran componente aleatorio, siendo asemejados a lo que se conoce como *random walk*, como bien se explica en [5].

Movimiento Browniano

Si atendemos a la hipótesis del camino aleatorio, podemos organizar la dependencia que puede existir entre los retornos de un activo r_t y r_{t+k} siendo éstos los retornos en los momentos t y $t+k$. Si definimos $f(r_t)$ y $g(r_{t+k})$ considerando que:

$$Cov[f(r_t), g(r_{t+k})] = 0 \tag{3.3}$$

Para cualquier t y $k \neq 0$. Esta propiedad nos dice que no hay correlación serial entre los retornos en ningún momento del tiempo y por tanto, los movimientos futuros no pueden ser explicados por retornos pasados. Lo que podemos entender, es que como no hay una correlación serial el próximo movimiento será aleatorio y por tanto impredecible. Podemos asimilar, al

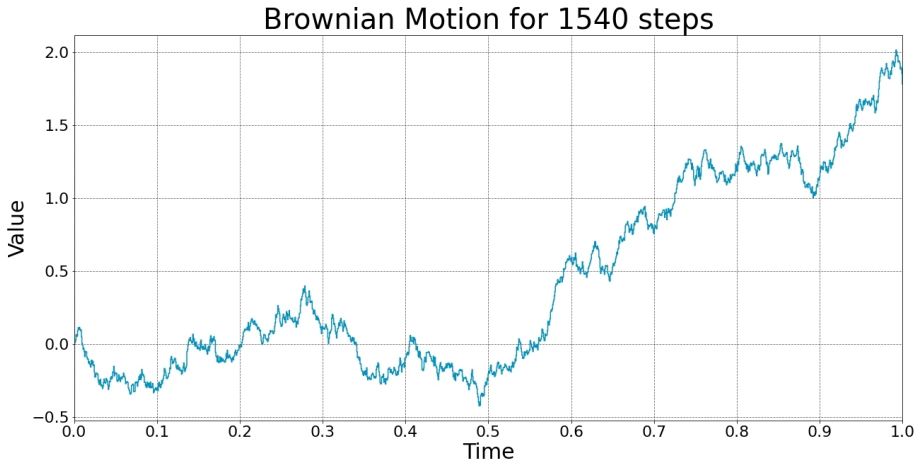


Imagen 3.4: Ejemplo de movimiento browniano

menos en cierta medida, que la evolución de la cotización de un activo es equiprobable y por tanto, no podremos predecir cuál será el mejor momento para introducir las órdenes en base a las cotizaciones o los retornos pasados.

Si el número total de momentos temporales T lo hacemos arbitrariamente grande y aplicamos el teorema central del límite podemos tratar la distribución de los retornos como una normal, a pesar de que en intervalos del histórico de retornos $(t, t+k)$ sean no normales.

El movimiento browniano o *Brownian Motion* fue descubierto por el botánico James Brown por el que cierto tipo de partículas, como es el caso del polen, en un fluido el cual hace que este movimiento sea continuo. Un activo financiero se asemeja a esta partícula moviéndose en un fluido que sería el conjunto de agentes intervinientes. En el caso concreto del precio, el movimiento browniano afecta en un único eje de ordenadas (en el eje de abscisas tenemos el tiempo y es invariante al movimiento), mientras que el caso del polen tiene un movimiento en tres dimensiones.

Para ilustrar el movimiento browniano podemos ver en la imagen 3.4 una serie temporal generada aleatoriamente gracias al artículo [17], la cual se asemeja bastante al posible comportamiento de un activo financiero.

Acercando esta equiprobabilidad del movimiento a la ejecución de órdenes significa que no podemos prever si el activo subirá o bajará y por tanto si es más conveniente acelerar o frenar el lanzamiento de órdenes para tener un mejor precio medio ponderado.

Martingalas

Como se comenta en [5], el primer modelo financiero es posible que sea el conocido como *martingala*, proveniente del siglo XVI. La martingala establece que estamos en un juego justo en el que no encontramos nada a favor ni de nosotros, ni de nuestros oponentes, lo cual se puede definir como:

$$E[P_{t+1}|P_t, P_{t-1}, ...] = P_t \tag{3.4}$$

Donde P_t es el precio de cotización de hoy, que será equivalente al precio de cotización de mañana P_{t+1} teniendo en cuenta toda la serie histórica. De aquí se deduce que la mejor estimación de un precio futuro, minimizando el error obtenido, es el precio actual y por tanto cualquier aberración del precio (desviación sobre el precio actual) debe corregirse, y por ende, podemos invertir en ello. Esta es la base de la archiconocida estrategia de inversión en el casino apostando al rojo o al negro el doble en cada tirada en caso de que no salga lo que deseamos.

En la práctica, y en lo que atañe al problema de la optimización de ejecución de órdenes, que un activo se comporte de modo que se cumpla la condición de la martingala supone que podremos fluir insuflando órdenes como un fuelle. Según el activo cae podemos adquirir más y más rápido el stock deseado, mientras que si el activo comienza una tendencia alcista compraríamos menos o más lento a esperas de que el activo retornara a su cotización inicial P_t. En el caso de que queramos vender o liquidar un activo, aplicaría lo mismo, pero, en sentido contrario.

Si hiciéramos un túnel de 50 metros (como el que tienen muchas carreteras) podremos diseñarlo como si la Tierra fuera una superficie plana, sin embargo, si nuestro deseo fuera el de realizar el Eurotúnel que conecta Francia y Gran Bretaña y con una longitud de 50 km, tendremos que tener en cuenta la curvatura de la Tierra.

En el caso de los mercados financieros, ocurre algo bastante parecido. Si analizamos tanto el movimiento browniano, como la martingala, a pesar de parecer contradictorias se pueden cumplir de una manera bastante coherente. El paseo aleatorio nos dice que el precio evolucionará de manera aleatoria, pudiendo subir, bajar o mantenerse el precio. Si calculamos la probabilidad para cada nivel de precio mantenerse en el precio actual será lo más probable, como en la imagen 3.5. Por otra parte, tenemos la martingala que nos dice que el precio justo es el actual y que una aberración en el precio debe de corregirse.

Imagen 3.5: Cono de probabilidad en el QQQ (fondo de inversión cotizado como una empresa que invierte en el sector tecnológico en EEUU) con la plataforma Thinkorswim

De aplicarse la martingala en el sentido estricto la martingala, esto es lo que encontramos en el corto plazo, evidentemente en el largo plazo no aplica ya que de ser así ningún tipo de inversión daría frutos, cosa que no es cierta, aunque todos los activos no se comportan de la misma manera. De aplicarse estrictamente significaría que toda caída de precios vendría acompañado por una subida y toda subida por una bajada de igual tamaño siempre.

Ejecución de órdenes

En anteriores secciones y apartados se han introducido ciertos conceptos financieros que no solo sirven para el objeto del presente trabajo, sino, como una base suficientemente sólida para entender los conceptos sobre los que nos apoyaremos.

La idea de la optimización en la ejecución de órdenes surge en [1] donde se plantea no sólo minimizar los costes, lo cual está bien, sino, que introducía un binomio riesgo-rentabilidad donde se crea una *frontera eficiente* análogo al existente en las carteras de media-varianza creadas en la Teoría Moderna de Porfolios (Modern Portfolio Theory) por Harry Markowitz[7].

[7]Harry Markowitz es el ganador del Nobel de economía en el año 1990. La Teoría Moderna de Porfolios, creada en 1952, combina n activos activos a los que asigna un peso w_i siendo $\sum w_i = 1$, donde 0 implica la no presencia del activo en la cartera y 1 que el 100 % de la cartera. Lo que hace Markowitz en este momento es optimizar probando todas las posibles combinaciones de modo que tenemos como resultado una dupla de rentabilidad (retornos medios de cada cartera) y de volatilidad (desviación estándar de los resultados). Esta dupla crea una *frontera eficiente* donde ninguna combinación nos

Lo que se hace en [1] es crear una estrategia que se dedique a liquidar un bloque de acciones de una empresa de la cartera de manera constante. Para ello se plantean dos componentes:

1. **Impacto de mercado:** si una posición grande se adquiere o liquida rápidamente, los costes crecen ya que movemos el mercado en nuestra contra

2. **Riesgo de volatilidad:** como todo activo financiero tiene una volatilidad inherente, corremos un riesgo de mercado que será tanto mayor según la volatilidad del activo y el tiempo que estemos expuestos a este riesgo

Con estos dos elementos se puede clasificar una estrategia de liquidación constante durante n periodos los cuales pueden ser temporales, de cambios de precio, de volumen, etc., aunque los autores lo determinan de manera temporal (para poder escalarlo después a un modelo continuo desde el enfoque discreto). Esta estrategia será tal que la lista $x_0, ..., x_n$ representan el conjunto de acciones que quedan por liquidar (o adquirir). De aquí se deduce las siguientes dos ecuaciones:

$$n_k = x_{k+1} - x_k \tag{3.5}$$

$$v_k = \frac{n_k}{t} \tag{3.6}$$

Donde t es la unidad de tiempo, cambio de precios o volumen acumulado citado anteriormente, n. n_k son las acciones que se adquieren en entre dos instantes temporales y v_k es la velocidad en cada unidad de tiempo. Teniendo tanto la velocidad cómo la estrategia de liquidación (adquisición) definida, se nos plantean claros tres posibles escenarios:

1. **Minimizar la varianza:** esto se consigue liquidando toda la posición en un único paso

2. **Minimizar el impacto:** liquidando la mínima cantidad posible de acciones en cada momento temporal (minimizando v_k)

proporciona mayor rentabilidad para una volatilidad dada, ni una menor volatilidad para una rentabilidad dada.

3. **Punto intermedio:** teniendo en cuenta tanto la varianza como el impacto con un parámetro λ que simboliza la aversión al riesgo

De este modo, se pueden probar todas las estrategias x, las cuales pueden ser definidas por el número de pasos n para realizar la liquidación, y nos quedamos con aquellas que minimizan:

$$\min_x(E(x) + \lambda V(x)) \tag{3.7}$$

Donde $E(x)$ es el coste de la estrategia x medido como la diferencia media entre el punto de inicio de la liquidación y el precio medio ponderado obtenido por la estrategia, y donde $V(x)$ es la varianza de estas diferencias. Así lo que conseguimos, según vamos dando valores a λ, es obtener una frontera eficiente donde ninguna estrategia tiene menor coste para una volatilidad dada, o menos volatilidad para un coste dado, al igual que nos ocurría con el modelo de media-varianza de Markowitz. Como se puede entender, el resultado es una función convexa.

Estamos ante un proceso aleatorio, por lo que la última cuestión que se plantea en [1] a la hora de evaluar una estrategia con media $E(x)$ y varianza $V(x)$, no es sólo como se ve en la ecuación 3.7, sino, teniendo en cuenta este componente aleatorio mediando lo que se denomina *Liquidity adjusted Value at Risk* o *L-VaR*.

Para entender el L-VaR, hay que estudiar primero el *Value at Risk* o *VaR*. El VaR es una medida de volatilidad que tiene en cuenta un intervalo de confianza p, generalmente al 95 % o al 99 %. La fórmula es la siguiente:

$$VaR = z\sigma\sqrt{t} \tag{3.8}$$

Donde z determina el intervalo de confianza, σ es el valor de la desviación estándar y t es el horizonte temporal sobre el que se hace el cálculo. Imaginemos que analizamos la volatilidad diaria de un activo y nos da como resultado un 1 % y queremos saber cuál es su volatilidad anualizada al 95 % contando que el año tiene 252 días hábiles, substituyendo en 3.8:

$$VaR = 1{,}64 * 1\,\% * \sqrt{252} = 26{,}11\,\% \tag{3.9}$$

Como suele ocurrir en finanzas, y este caso no es menos, hablamos de una técnica más que de una métrica definida. Este VaR se puede utilizar para calcular una posible caída en un marco temporal (diario, mensual,

anual), para analizar la volatilidad, la correlación entre diversos activos, o como en el caso que nos atañe, para estudiar la liquidez de un activo y una estrategia para adquirir o liquidar posiciones.

Utilizando la ecuación 3.8 como base, podemos llegar al L-VaR como:

$$L{-}VaR_p(x) = \lambda\sqrt{V(x)} + E(x) \tag{3.10}$$

La conclusión, sería, analizar y quedarse con aquellas estrategias que minimicen el L-Var para un nivel λ, ya esta estrategia seleccionada será no ya la que minimice el coste, sino, la que tendremos menor coste con un nivel de confianza.

Impacto temporal y permanente

En la imagen 1.1 se analizó cómo se recorre la curva de oferta según vamos adquiriendo una mayor cantidad del bien producido, en nuestro caso activos financieros. Lo que se entiende es que en el momento que disminuimos esa demanda por estos activos, se corrige el precio al que se adquieren los mismos.

Implícitamente, lo que estamos asumiendo es un impacto temporal. Con impacto temporal lo que se observa es simplemente el desplazamiento por el libro de órdenes en función del tamaño x_k. Se habla de desplazamiento temporal porque anteriores o nuevos agentes volverán a colocar posiciones en el ask o en el bid de nuevo, volviendo al estado inicial antes de lanzar la orden, como si de un muelle se tratara.

Por otra parte, existe también la posibilidad de que este impacto sea permanente, creando no ya un movimiento a lo largo de la curva de oferta (o de demanda si estamos liquidando), sino que provoque un *desplazamiento* de la propia curva de oferta (demanda). Esto se puede ver en la imagen 3.6 donde P_0 es el precio en equilibrio inicial, P_1 es el precio de equilibrio una vez hemos impactado en el mercado de manera permanente y P_2 es el precio que pagaríamos por una cantidad Q_2 con impacto temporal. Cuando se habla de impacto permanente lo que se entiende es un desplazamiento de la curva de de oferta o de demanda durante todo lo que resta de la orden.

Entenderemos como impacto total a la suma del impacto temporal más el impacto permanente. Podemos ver cómo funcionaría gráficamente en la imagen 3.7, donde se analiza un ejemplo en el que un impacto en el mercado muy grande pueda provocar que parte de ese impacto sea de manera permanente.

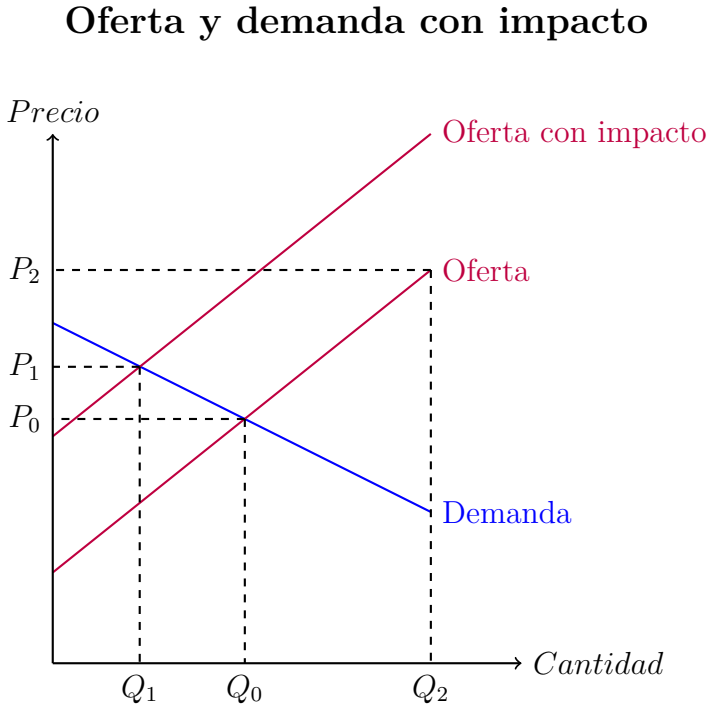


Imagen 3.6: Oferta-demanda con impacto de mercado y sus puntos de equilibrio

El impacto en el mercado que estudiaremos e intentaremos controlar es el que generaremos nosotros. Podemos entender que las cotizaciones históricas y futuras de los activos financieros son consecuencia de los impactos de mercado del conjunto de los agentes que intervienen, tanto comprando como vendiendo los diferentes activos.

Al estar realizando simulaciones y no con dinero real no podemos estimar el impacto permanente. Incluso aunque trabajáramos con compras y ventas reales nos sería complejo hacer este cálculo ya que no se puede realizar dos muestras exactamente iguales con la misma compra en el mismo activo y en el mismo momento temporal.

Técnicas de ejecución de órdenes

Como estamos estudiando, no hay una única técnica de ejecución de órdenes y mucho menos, tampoco hay una única solución óptima inmejorable. En [1] se trata de buscar aquella estrategia que minimice el coste esperado en función del L-VaR, en [6] implementa un algoritmo que siguiendo como

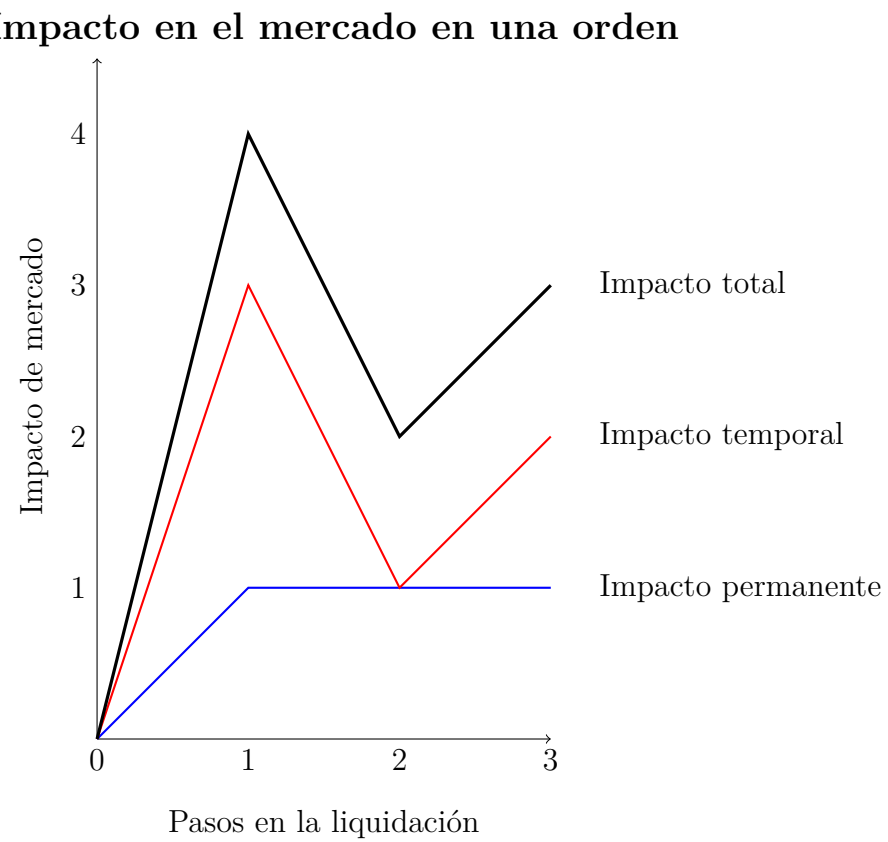


Imagen 3.7: Influencia del impacto temporal y permanente en el impacto total de una orden

referencia a [1] lo mejora mediante el uso de órdenes limitadas las cuales no tienen impacto en mercado.

Por otra parte, también podemos encontrar trabajos en otra dirección como es la de [9] que trata de dividir la orden padre en órdenes hijas óptimas, consiguiendo este óptimo en función del desequilibrio que hay entre las posiciones del ask y del bid.

También se pueden hallar otros trabajos que hayan traspasado la barrera de la econometría hacia el machine learning como puede ser lo publicado en [23] donde se trata de mejorar el VWAP gracias al uso de órdenes a mercado y de una arquitectura en Double Q-Learning, que se verá más adelante.

Estas técnicas han sido todas pensadas con la contraparte, el mercado que nos va a vender o a comprar a nosotros y a cómo impactarles lo mínimo posible. Aunque este es el enfoque predominante académicamente, no es

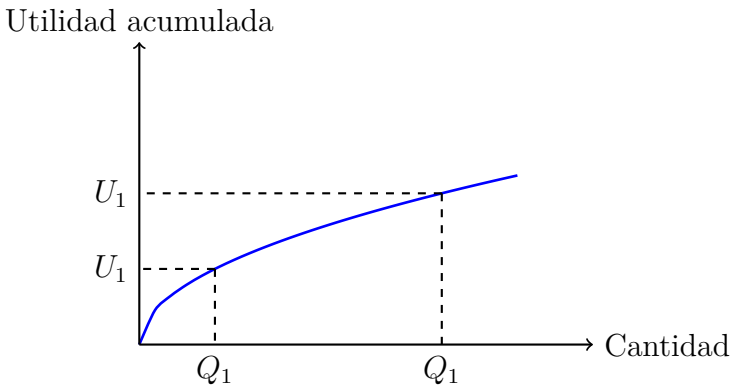


Imagen 3.8: Función de utilidad

descabellado pensar la opción inversa en la que es el agente (nosotros) quien supone el cimiento del análisis. Si nos centramos en el agente, y no en el mercado, tenemos que hablar del concepto de *utilidad*.

Ya en el año 1738, Daniel Bernoulli [4] nos explicaba tanto el concepto de utilidad como su aplicación en finanzas. Bernoulli nos dice que el *valor* de un producto no está basado en su *precio*, sino, en la *utilidad* que nos brinda. El precio de este producto es el mismo para todo el mundo, mientras que su utilidad depende de las circunstancias personales de cada persona. El ejemplo que pone para ilustrar su tesis es la de un boleto de lotería que tiene un 50 % de probabilidad de caer agraciado con un premio de 20.000 ducados o de 0 ducados en caso contrario. Lo que se expone es que una persona pobre aceptaría 9.000 ducados por ese boleto, que tiene 10.000 ducados de esperanza matemática, ya que preferirá la seguridad del dinero en la mano, mientras que una persona acaudalada preferirá maximizar su beneficio ya que se lo puede permitir.

Continuando con [4], nos dice que cualquier incremento de riqueza, independientemente del tamaño de este incremento, da como resultado un aumento de utilidad, la cual será inversamente proporcional a la cantidad de bienes que ya se posea. Gráficamente, se puede resumir en la imagen 3.8.

En relación a lo que nos atañe, esta misma idea podría ser implementada a la adquisición o liquidación de posiciones. Si un agente desea adquirir una cantidad Q de un activo es entendible que la utilidad aportada por las Q_0 iniciales es mayor que las Q_n posteriores y por tanto, sería entendible que la cantidad de adquieren (o liquidan) sea decreciente a lo largo de la operación.

Una vez se han visto las estas diferentes posibilidades podemos resumir las diferentes técnicas en relación a la ejecución de órdenes como:

- Volume Weighted Average Price (VWAP)
- Percent on Volume (POV)
- Time Weighted Average Price (TWAP)
- Desequilibrio del volumen
- L-VaR
- Función de utilidad

Estos son sólo un ejemplo de las diferentes posibilidades que podemos encontrar, encontrándose este campo de estudio totalmente abierto.

3.3. Ciencia de Datos

En la sección 3.2 se trabajaba el primero de los pilares, siendo, la *ciencia de datos* el otro pilar. La definición de ésta es, a pesar de lo extendido de uso, bastante difuso. Como se define en [26], podemos decir que la ciencia de datos es un conjunto de principios que apoyan y guían la extracción de información y de conocimiento desde los datos.

Muy cercano a esta ciencia de datos encontramos la *minería de datos*, que engloba cientos de algoritmos para obtener información de los datos. Posiblemente, el campo con las mayores aplicaciones pueda ser el márquetin, donde podemos encontrar desde anuncios a ventas cruzadas.

La ciencia de datos también tiene un uso muy extendido en cuanto a los denominados CRM[8], donde se estudia el comportamiento de los clientes para poder obtener el máximo beneficio. Netflix nos recomienda las series y películas que más se nos adaptan [13], los bancos lo usan para la detección de fraudes[25] o Amazon nos recomienda ventas relacionadas con los productos que estamos viendo gracias a estas técnicas[14].

Pero, la ciencia de datos no se queda en estos algoritmos propios de la minería de datos, sino que es un campo interdisciplinar que junta los procesos y los sistemas para extraer información con grandes volúmenes

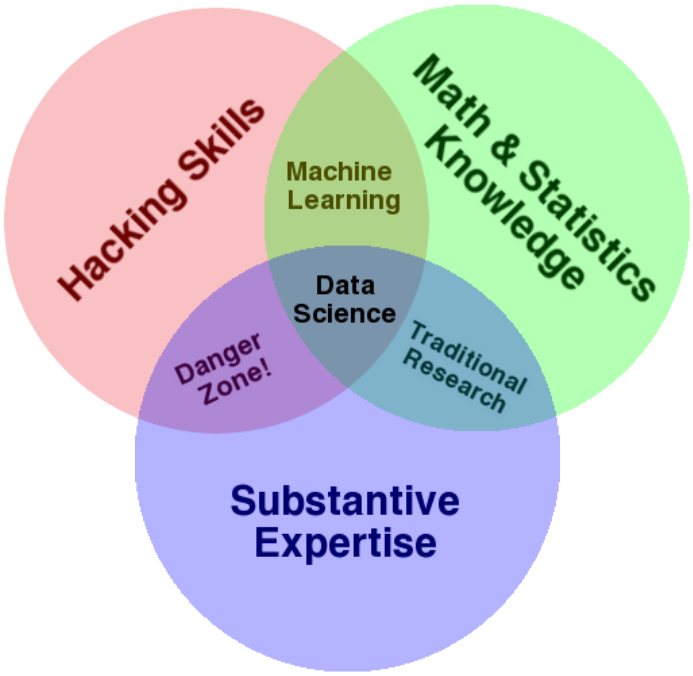


Imagen 3.9: Diagrama de Venn que define la ciencia de Datos[7]

de datos como fuente, independientemente de que esta información esté estructurada o no.

La mejor forma de hacernos una imagen es ver el diagrama de Venn realizado por Andrew Conway[7]. Lo que hace Conway es definir la ciencia de datos como la intersección entre las matemáticas y estadísticas con el conocimiento en un campo concreto (en este caso el financiero) junto con las habilidades técnicas propias de la informática.

Con estos tres puntales, su implementación incumbe principios, técnicas y procesos para entender de una forma automatizada el análisis de los datos con los que trabaja. Uno de los principales objetivos que tiene, más allá del propio estudio de la ciencia, es poder llegar a la toma de decisiones basadas en los datos.

La toma de decisiones basadas en los datos se refiere a abandonar la toma de decisiones basada en la intuición o en la experiencia del directivo para realizarlo a través de los datos, siendo estas decisiones tomadas en gran medida de manera automatizada.

[8]Customer Relationship Management

Aprendizaje automático

Con la ciencia de datos lo que vamos a poder conseguir es *aprender* a tomar las decisiones correctas en relación a la ejecución de órdenes. Estas decisiones se deben de tomar basándonos en los datos y de una manera automatizada.

Si pensamos en la manera más naíf de aprendizaje, lo primero que nos puede venir a la mente es la manera en la que se comporta un niño. Imaginemos a ese niño que quiere hacer su primer puzzle. Lo que hace es ir cogiendo las piezas de una manera aleatoria hasta que van encajando las piezas. Poco a poco, este niño va aprendiendo que la forma de una pieza tiene que coincidir con el resto que tiene alrededor (además de encajar el dibujo). Con el tiempo, el niño será capaz de identificar, mediante ensayo y error, dónde va cada pieza. Ésto es el aprendizaje.

Como bien se extrae de [2], nos encontramos en la era del *Big Data*. Tanto empresas como usuarios generamos una cantidad ingente de información, sea consciente o inconscientemente. Pensemos en una tienda de ropa online con presencia en todo el territorio. Según los clientes vayan adquiriendo nuevos artículos se pueden quedar registrados todos los detalles de la compra como el usuario, la fecha, artículos comprados, artículos visitados, total de dinero gastado. Como es evidente, esto marca un registro que se puede estudiar y analizar. La tienda de ropa, por su parte, deseará maximizar su beneficio, bien sea vendiendo más, vendiendo con más margen, aumentando el ticket medio por cliente o mejorando la logística de la compañía (por ejemplo, anticipando olas de frío o calor o la disponibilidad de productos en fechas señaladas como navidades o el día del padre). En resumen, provocando una mayor satisfacción al cliente, que es precisamente lo que desea el propio cliente.

Los hábitos de consumo de los clientes pueden variar notable en función de muchas variables como son las geográficas (diferentes países pueden tener diferentes costumbres), lingüísticamente, por edad, sexo o nivel de estudios, generándose patrones dignos de ser estudiados y que reducen el marasmo de datos, aparentemente aleatorios, a información útil y valiosa para las empresas. Continuando el caso de la tienda de ropa, es entendible un cliente que tras un bikini se compre unas chanclas o un pareo, y otro cliente tras la compra de un traje adquiera una corbata.

Esto que a nosotros nos puede parecer muy intuitivo hay que modelarlo gracias a algún tipo de algoritmo que nos permita estudiar y sacar provecho de los datos. Al igual que ocurría en el caso del niño con el puzzle, aquí, la

empresa de ropa puede estudiar los datos de las compras de los clientes de modo que un algoritmo sea capaz de *aprender* cómo mejorar el proceso de venta, a pesar de estar trabajando con un problema estocástico.

Esta es la base sobre la que se construye el *aprendizaje automático* o *machine learning*, tan de boga en nuestros días, y que será sobre lo que nos apoyemos de cara a solucionar nuestro problema de la ejecución de órdenes.

Si analizamos, al menos mentalmente, el enfoque que se ha dado a este aprendizaje automático, dista sutilmente con respecto a la investigación clásica. Históricamente, los investigadores han tratado de extraer las leyes o normas que rigen nuestro entorno, ya sean leyes físicas como las que se consiguieron tras los estudios de Newton, o incluso saliéndonos de las ciencias exactas, las que se encuentran en las ciencias sociales como fue la ley de oferta y demanda como se vio en la imagen 1.1. En el caso del aprendizaje automático lo que se intenta, en lenguaje coloquial, es llegar a una *aproximación* lo más fidedigna posible. Este es el punto clave. No se intenta acertar, sino, aproximar.

El resultado de este nuevo paradigma es la aplicación en multitud de campos, además de lo analizado en la tienda de ropa, se puede aplicar a la detección de transferencias bancarias fraudulentas, predicciones en bolsa, spam[9] en los correos electrónicos. También se usan en la industria para la detección de errores, la optimización de procesos o en la medicina para la diagnosis clínica. Actualmente, una de las puntas de lanza, junto con los ejemplos anteriores, es el reconocimiento tanto de imagen, como de vídeo, lo cual abre un campo inmenso de estudio y de aplicaciones prácticas.

Una de las principales características que se suele incluir en el aprendizaje automático, y de la que no disfrutan los modelos clásicos, es su capacidad de adaptación. Históricamente lo que se ha tratado es de encontrar esa *fórmula mágica* capaz de explicar un proceso o un comportamiento, estableciéndose esta fórmula mágica de forma estática. Con el aprendizaje automático, lo que se puede conseguir es también la adaptación de manera *continua* a los diferentes procesos. Si continuamos con nuestro ejemplo de la tienda de ropa, puede que junto a ese traje, la moda nos diga que los clientes suelen comprar corbatas lisas, y por tanto esas son las corbatas que se deben de recomendar. Un modelo estático, lo que haría es recomendar siempre ese tipo de corbatas, mientras que si disponemos de un modelo dinámico podemos adaptarnos a las *modas* consiguiendo que si al año siguiente lo que se compran son corbatas de cuadros, sean éstas las recomendadas.

[9]El spam son correos electrónicos no deseados enviados, generalmente, de manera masiva con objeto publicitario.

Entrenamiento y test

Otra de las características propias del aprendizaje, especialmente cuando hablamos de aprendizaje automático, es la validación del aprendizaje. Tal es su importancia que merece un apartado en este trabajo.

Cuando nosotros realizamos un aprendizaje uno de los principales objetivos es ser capaces de poder replicar esta nueva habilidad adquirida. Si continuamos el ejemplo del niño con el puzzle, estaríamos hablando de que una vez el niño haya aprendido a hacer el puzzle, éste sea capaz de solventar con éxito nuevos puzzles que se le presenten.

Una vez hemos realizado este aprendizaje es necesario *evaluar* el rendimiento del mismo, existiendo, por norma general, un sesgo en dicho rendimiento. Intuitivamente es muy fácil de entender, aunque detrás tenga su desarrollo formal, si se ha aprendido una tarea lo normal es que tengamos un mejor resultado repitiendo esta tarea (el puzzle del niño) que aplicándola en un conjunto que no hemos entrenado (un puzzle nuevo y diferente).

Según [18], si suponemos que vamos a estimar el error asociado con la optimización de un modelo de aprendizaje en un conjunto de datos, podemos dividir la muestra que tenemos disponible de manera aleatoria. De esta forma podemos, con unos datos de partida, entrenar el modelo en un subconjunto de datos y validar su comportamiento en otro subconjunto de datos que no han sido usados para el entrenamiento.

En base a lo que se deduce de [18], observamos que tenemos unos datos de entrenamiento, unos de validación (donde se optimizan los parámetros) y unos de test que serán usados para medir el rendimiento del modelo. En la imagen 3.10 se puede observar esta idea, que puede ser desarrollada a una validación cruzada como se puede observar en la imagen 3.11.

En el caso que nos atañe, la ejecución de órdenes, lo que se usará es una modificación o adaptación de esta validación cruzada. Lo que tendremos es el histórico de un activo financiero, véase Inditex, el cual será recorrido desde un punto inicial aleatorio hasta que se hayan adquirido las acciones deseadas. Este proceso se repetirá en sucesivas ocasiones (llamadas episodios) mientras la información de las recompensas van *enseñando* al modelo. Como vemos, es una técnica de remuestreo muy parecida a la validación cruzada.

Clases de aprendizaje automático

Una vez explicado grosso modo el concepto de aprendizaje, así cómo el funcionamiento la división de una muestra en datos para el entrenamiento,

Imagen 3.10: Datos de entrenamiento, validación y test usados en un modelo de aprendizaje

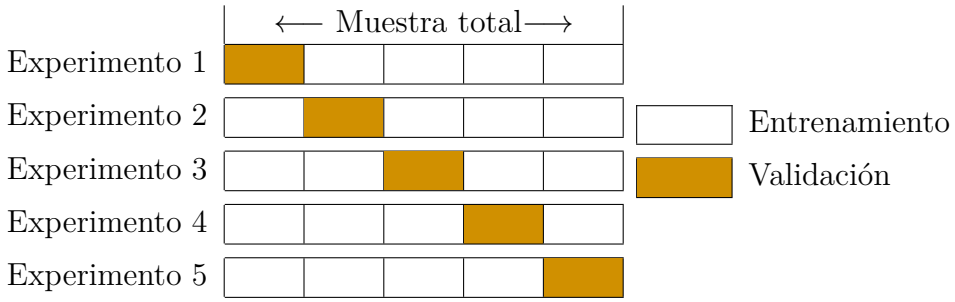


Imagen 3.11: Validación cruzada en 5 partes

para el test y la validación no podíamos detenernos ahí, sino que, se van a detallar las principales clases de aprendizaje automático que han sido usadas en este Trabajo Fin de Máster.

El aprendizaje automático se divide en tres clases las cuales son el aprendizaje supervisado, el aprendizaje no supervisado y el aprendizaje por refuerzo. El aprendizaje no supervisado, al no ser usado, no va a ser detallado.

Aprendizaje supervisado

El aprendizaje supervisado es la clase más extendida y en la práctica usada del aprendizaje automático. El aprendizaje supervisado, como se explica en [2], es una tarea por la cual se busca aprender a transformar una entrada en una salida y cuyos valores provienen de un supervisor.

El aprendizaje supervisado funciona de modo que por cada observación de la muestra x_i, con $i = 1, ..., n$ hay una etiqueta y_i asociada. A su vez, cada observación x_i puede estar compuesta por $k \geq 1$ atributos. Lo que se busca es ajustar un modelo g con una serie de parámetros θ tal que sea capaz de devolvernos la etiqueta y_i para cada observación o predictor, como se puede analizar en la ecuación 3.11. La intención de que sea a su vez capaz de replicar este comportamiento para futuras observaciones, cuando hablamos

de predicción, o entender mejor la relación que hay ente los atributos y la etiqueta, cuando hablamos de inferencia.

$$y = g(\chi|\theta) \tag{3.11}$$

A su vez, el aprendizaje puede ser dividido principalmente en:

- **Clasificación**
- **Regresión**

Es muy común que las etiquetas a las que intentamos aproximarnos en el aprendizaje supervisado sean *cualitativas*. Dentro de estas variables cualitativas podemos encontrar la nacionalidad del comprador de una tienda online o su profesión. Cuando se intentan predecir etiquetas cualitativas estamos hablando de *clasificación*, ya que conlleva clasificar una observación asignándola a una categoría o clase.

Para ilustrarlo mejor, veamos un sencillo ejemplo. Imaginemos que un banco quiere categorizar a sus clientes por solvencia crediticia de cara a concederles un préstamo mediante dos atributos que son el nivel de ingresos y el nivel de ahorro de cliente. Aquellos clientes de bajo riesgo serán aquellos que ganen mucho y que ahorren mucho, mientras que si el cliente o bien ahorra poco o bien gana poco sería tratado como un cliente de riesgo ya que podría incumplir los pagos si le bajan el sueldo o con cualquier gasto imprevisto. En la imagen 3.12 se puede observar gráficamente cómo se podría comparar este clasificador, el cual se podría implementar mediante un método conocido como árbol.

Estos métodos de clasificación pueden no sólo devolvernos una clase o categoría, sino, también la probabilidad de pertenencia a la misma. En [18] nos adelantan que los tres clasificadores más usados son la regresión logística, el análisis lineal discriminante y el k vecinos más próximos, aunque también se pueden encontrar otros métodos como los árboles o las máquinas de vectores de soporte.

El otro tipo de aprendizaje supervisado es la regresión. Mientras que en la clasificación se buscaban conseguir etiquetas cualitativas, en las regresiones lo que se hace es hallar etiquetas *cuantitativas*. De nuevo, veamos un ejemplo de su funcionamiento. Es bien sabido que los coches con mayor caballaje consumen más gasolina y es eso precisamente lo que se puede observar en la

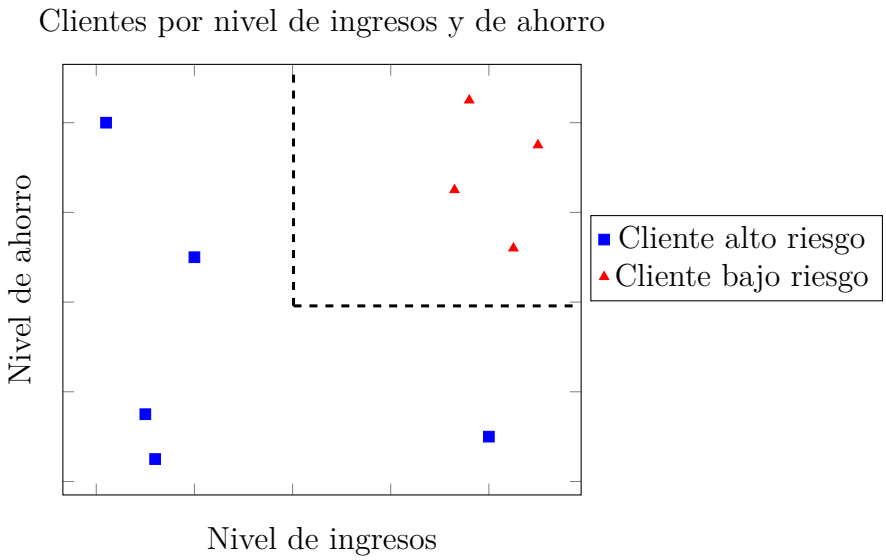


Imagen 3.12: Ejemplo de aprendizaje supervisado identificando clientes de alto y bajo riesgo de crédito

imagen 3.13. Lo que se ha generado, con el conjunto de datos Auto MPG[10], es una regresión lineal a través de la cual si le proporcionamos el caballaje de un coche nos devuelve un valor estimado de las millas que puede recorrer por cada galón de gasolina consumido.

Al igual que nos ocurre con la clasificación, la regresión no tiene una única función f para llegar a la predicción. A pesar de ello, seguirán la forma de la ecuación 3.12 donde tenemos el valor de la regresión r es el resultado de la aplicación de una función f a la que se le ha pasado la muestra x y a la que hay que añadir el error ϵ. Cabe destacar que esta función f en la imagen 3.13 es una función lineal, aunque se le puede aproximar cualquier otro tipo de función ya sea polinomial, multivariada, etc.

$$r = f(x) + \epsilon \tag{3.12}$$

En el ámbito de la ejecución de órdenes, la decisión de cual es acción de compra más conveniente en cada momento puede asemejarse a la clasificación ya que podemos tener como resultado un valor cualitativo que sea el que más se aproxime a la minimización de costes. Por otra parte, el cálculo del coste asociado a una compra puede llegar a ser relacionado con una regresión.

[10]El conjunto de datos se puede conseguir de http://archive.ics.uci.edu/ml/datasets/Auto+MPG

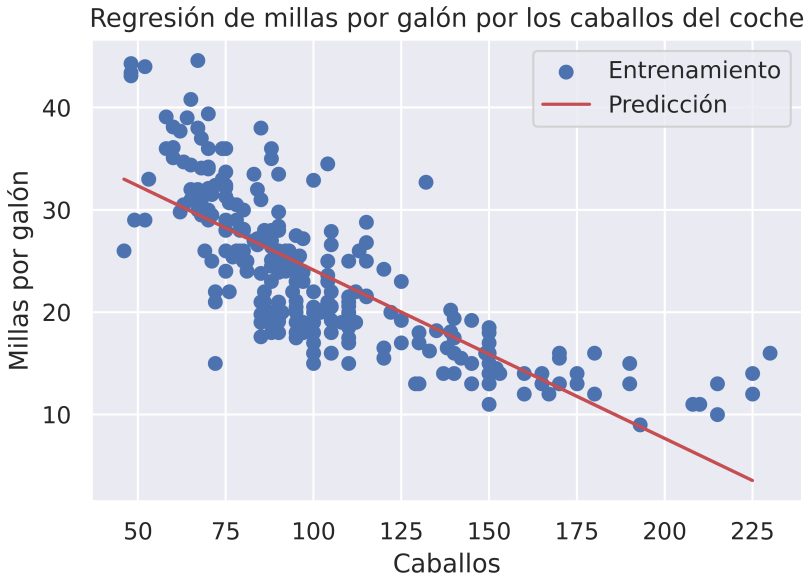


Imagen 3.13: Ejemplo de una regresión para predecir el consumo de un coche en millas por galón en función del caballaje del coche

Más adelante se verá como se ha usado la regresión para la implementación de uno de los modelos de ejecuciones de órdenes.

Aprendizaje por refuerzo

El aprendizaje por refuerzo, según se extrae de [29] es aprender, traduciendo situaciones a acciones, para maximizar una recompensa. Al aprendiz no se le dice qué acciones debe de tomar sino que debe *descubrirlas* por sí mismo averiguando qué acciones son las que mayor recompensa le ofrecen.

En el aprendizaje por refuerzo, un agente (el aprendiz) busca maximizar estas recompensas por medio de sus acciones y para hacerlo debe haberlas probado antes y haber comprobado que eran la mejor opción de todas las disponibles. Para descubrir estas acciones el agente debe probar acciones que no haya probado en el pasado y posteriormente explotar aquellas decisiones correctas que ha aprendido. Esto significa que el agente debe tanto hacer lo que sabe, como investigar por si se puede hacer mejor. En un proceso estocástico, el agente debe realizar esta operación sucesivas veces y progresivamente, según va a aprendiendo y aumentando la relevancia de su aprendizaje, ir basándose cada vez más en su experiencia.

Otro punto clave es que el aprendizaje por refuerzo considera el problema completo de la relación del agente interactuando con un entorno desconocido. Si nos fijamos, este aprendizaje es el más parecido al aprendizaje de un ser humano. Un niño va aprendiendo poco a poco que no hay que tocar las cosas calientes o que darse un golpe duele, mientras que una golosina puede alegrarles la mañana. Es por ello que hay cientos de ejemplos donde este tipo de aprendizaje puede dar lo mejor de sí mismo.

Conclusiones del aprendizaje automático

Varias de las técnicas vistas nos pueden ser de ayuda, por ejemplo, con la regresión propia del aprendizaje supervisado se puede calcular cual es volumen esperado en un momento específico del tiempo.

Si lo analizamos detenidamente, y sin profundizar todavía en sus características, el aprendizaje por refuerzo es lo que hemos estamos buscando desde el principio. Significa ir aprendiendo cómo influyen nuestras decisiones a la adquisición (o la liquidación) de posiciones en el mercado financiero. Si nuestra acción es no actuar lo que tendremos es una mayor volatilidad de los resultados, pero, no tendremos coste alguno como se deduce de la ecuación 3.7. Por el otro lado, si tomamos acciones que adquieran muchas unidades del activo financiero tendremos un mayor coste debido a su deslizamiento a lo largo de libro de órdenes. Este es el punto donde se juntan los mercados financieros y la ciencia de datos que se han desarrollado hasta ahora.

El modelo: Aprendizaje por refuerzo

En el anterior apartado se han definido y ejemplificado los diferentes modos de aprendizaje automático, incluyendo ejemplos para su fácil comprensión y su vinculación o posible vinculación a la ejecución de órdenes los mercados financieros. Para finalizarlo, se ha concluido que el faro que va a guiar el desarrollo es el aprendizaje por refuerzo.

Elementos propios del aprendizaje por refuerzo

Más allá de los ya citados *agente*, que es quién toma las decisiones, y el *entorno*, que es quién establece las recompensas, [29] diferencia cuatro elementos en un sistema de aprendizaje por refuerzo que son:

1. La política
2. La señal de recompensa

3. La función de valor

4. El modelo del entorno (este es opcional)

La *política* define el comportamiento del agente, mapeando el estado en el entorno con las acciones que se deben de tomar. Imaginemos que tenemos un robot aspirador que ha chocado contra una pared, este sería el estado, qué debería hacer en ese estado sería lo que define la política. En el caso más sencillo sería consultar una tabla que relacione estado y acciones. Por lo general, las políticas deben ser estocásticas, asignando probabilidades a cada acción.

La *señal de recompensa* establece cuál es el objetivo del problema que queremos solventar mediante el aprendizaje. A cada paso el entorno le devuelve la recompensa, mientras que el agente intenta maximizar estas recompensas mediante la política. Esta es la base de la modificación de la política ya que si una de la acciones tiene poca recompensa elegirá otra. Hay que señalar que la recompensa puede estar diferida en el tiempo, recibiéndose varios pasos después.

La *función de valor* establece qué es bueno en el largo plazo. El valor del estado es la cantidad total de recompensa que tendrá partiendo desde ese estado. Decimos que establece lo bueno en el largo plazo ya que lo estamos buscando son aquellas acciones que maximicen el valor y no las recompensas instantáneas. El entorno nos dará la recompensa, generalmente al final, mientras que tenemos que *estimar el valor* por el camino. Es importante remarcar que este componente es el más importante de todos ya que es el que dice qué es bueno y qué es malo.

El *modelo del entorno* estima el comportamiento futuro del entorno. Dado un estado y una acción el modelo debe predecir el siguiente estado y la siguiente recompensa.

Procesos de Decisión de Markov

Los *Procesos de Decisión de Markov*, o *MDP* por sus siglas en inglés, son un marco para el problema de aprender a través de la interacción a conseguir una meta, la recompensa. El aprendiz, que es quien toma las decisiones, es el llamado *agente*, el cual va interactuando con el *entorno* mediante la selección de las acciones y las respuestas del entorno que presenta nuevas situaciones o *estados* y recompensas. Esto se puede ver perfectamente en la imagen 3.14

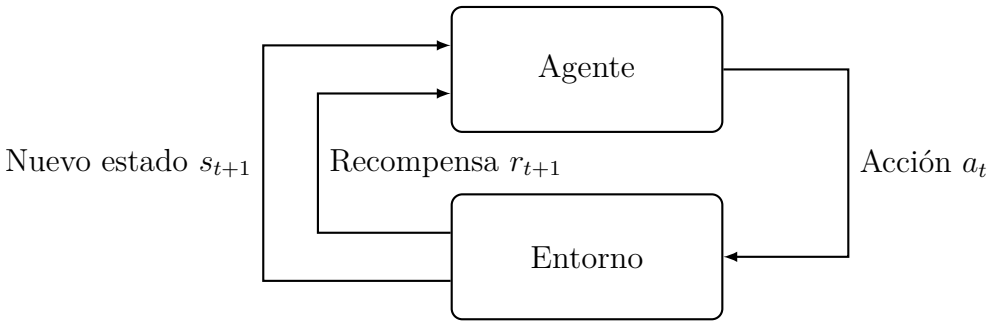


Imagen 3.14: Relación entre el agente y el entorno en un MDP

Según se explica en [24], en un MDP tenemos un juego finito de estados S donde uno de ellos, s_0 es el estado inicial. En cada instante temporal $t = 0, 1, ...$ su estado correspondiente es $s_t \in S$. Para cada estado $s_t \in S$ tenemos un conjunto finito A_s de acciones. Cuando se toma una acción $a \in A_{s_t}$ en el momento t, se incurre en un coste $c(s, a, t) \in C$, y su siguiente estado s' tiene una distribución de probabilidad $p(s, s', a, t)$. Se dice que el proceso es *estacionario* si el coste c y la distribución de probabilidad p son independientes del instante temporal t. Una *política* π asigna para cada momento temporal $t \in \{0, 1, 2, ...\}$ y estado $s \in S$ una acción $A(s, t)$.

Una vez que se tiene la distribución probabilidad p, se puede estimar el coste esperado en función de cada acción a en un momento t del tiempo y un estado s_t como se puede ver en la ecuación 3.13.

$$c(s, a) \doteq \mathbb{E}[C_t | S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{c \in C} c \sum_{s' \in S} p(s, s', a, t) \quad (3.13)$$

En la literatura es bastante común encontrar el resultado de una acción en un momento t y un estado s_t como una recompensa, pero, se va a continuar con el concepto de coste porque se aproxima más a lo que nos vamos a encontrar en la adquisición o liquidación de posiciones. La única diferencia estriba en que los costes se minimizan y las recompensas se maximizan.

Si nos retrotraemos a nuestro problema, el estado inicial s_0 se corresponde con nuestro estado inicial, cuando se toma la decisión de la adquisición o liquidación del activo financiero, el estado final s_T cuando se ha terminado la adquisición y el estado intermedio s_k cualquiera de los que podremos encontrar por el camino.

Metas, costes, retornos y episodios

En el aprendizaje por refuerzo la *meta* está formalizada en términos de una señal llamada coste (o recompensa) que es pasada al agente por parte del entorno. A cada paso, el coste es un simple número $c_t \in \mathbb{R}$, y lo que se busca es la minimización del valor esperado de la suma de los costes recibidos. El *retorno* es la suma de los costes $G_t \doteq c_{t+1} + c_{t+2} + c_{t+3} + ... + c_T$.

Cada subsecuencia de interacciones con un estado inicial y un estado final es conocido por *episodio*. Este enfoque tiene sentido cuando hay un paso final bien definido, que en nuestro caso será cuando se acabe la compra de las posiciones, además, cabe destacar que el número de pasos que se deben de realizar sigue una variable estocástica ya que si se adquiere muy rápido se tardará menos en completar la tarea que si se dilata en el tiempo.

Falta definir el concepto de *descuento*, como nos señala [29]. El agente, según va avanzando a lo largo de cada episodio, los cuales se sucederán como si de un videojuego se tratara, va probando las diferentes acciones en función del estado en el que se encuentre minimizando los costes descontados a lo largo de lo que queda episodio. Hay que descontar los costes futuros ya que tienen que ser estudiados en el momento actual. Imaginemos que tenemos dos episodios distintos con el mismo coste total, sin embargo, el primero de los episodios lo hace en t pasos, mientras que el segundo de los episodios lo hace en $2t$ pasos. Es de entender que es preferible el primero y por tanto, para que sean comparables no solo cuando tienen el mismo coste, sino, cuando tienen costes distintos, debemos traernos al momento presente cual es el coste total y como se va decidiendo a lo largo de cada episodio. Este análisis se realiza en cada paso entendiendo el coste total como el coste en lo que resta de episodio. Para modelar esta idea aparece el ya citado descuento que se puede estudiar según la ecuación 3.14.

$$G_T \doteq c_{t+1} + \gamma c_{t+2} + \gamma^2 c_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k c_{t+k+1} \tag{3.14}$$

Donde γ es un parámetro que mide el descuento, con $0 \leq \gamma \leq 1$. Si $\gamma = 0$, solo tendríamos se tendría en cuenta el siguiente paso, si $\gamma = 1$ todos los elementos pesan lo mismo dando igual que el coste sea en el siguiente paso o dentro de 10.

Políticas y función de valor

Una función de valor nos dice cómo de bueno es para el agente el estar en un estado determinado, en términos de futuros costes que pueden ser esperados.

Formalmente, una *política* transforma los estados en probabilidades de seleccionar cada una de las acciones disponibles. Como el aprendizaje por refuerzo se cimienta en la modificación del comportamiento en base a su experiencia, si tenemos muchas acciones tendremos una menor probabilidad de acceder a cada una de ellas y por ello, se tardará mucho en aprender y/o se llegará a decisiones subóptimas. La ecuación que define la función de valor es la 3.15.

$$v_\pi \doteq \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k c_{t+k+} \middle| S_t = s\right] \qquad \forall s \in S \tag{3.15}$$

El valor de un estado final es 0 por definición ya que no va a tener ningún coste futuro. Muy parecida a la idea de la función de valor para cada estado es la de el valor de una acción a en un estado s con una política π que se definirá como $q_\pi(s, a)$ como se puede observar en la ecuación 3.16.

$$q_\pi \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k c_{t+k+} \middle| S_t = s, A_t = a\right] \tag{3.16}$$

El agente debe mantener el v_π y q_π y ajustar los parámetros que mejor encajen con los retornos que se vayan observando.

Además el agente tiene como dilema el peso que va a dar al coste actual o al coste futuro, como nos explica [29] esto es lo que se llama *ecuación de Bellman*, la cual está definida en 3.17. Dada la heterocedasticidad de los datos financieros, junto con la varianza con respecto a las medias que se presentan, tanto intuitivamente, como empíricamente se verá que la tasa de descuento a usar será muy cercana a 1. Que estuviera cercana a 0 significaría que no tendría muy en cuenta futuros pasos, comportándose de manera *egoísta*, con desastrosos resultados.

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s] = \sum_a \pi(a|s) \sum_{s',c} p(s', c|s, a)\left[r + \gamma v_\pi(s')\right], \qquad \forall s \in S \tag{3.17}$$

Ante un estado inicial y las acciones disponibles, el entorno nos puede llevar a diferentes estados s', así como a un coste c, lo que hace la ecuación de Bellman 3.17 es promediar todas las posibilidades, asignándole pesos según la probabilidad de que ocurran. La ecuación establece que el valor de un estado inicial (o cualquier estado intermedio) es el valor descontado del siguiente estado más el coste del recorrido entre s y s'.

Para cada estado s tiene que haber una o más de una acción que sea la mejor en términos de costes según la ecuación de Bellman. Una política que asigne probabilidades distintas de 0 a estas acciones óptimas se le considerará una política óptima ya que será capaz de elegir siempre la mejor acción.

Hasta este punto se conoce ya cómo funciona matemáticamente este aprendizaje por refuerzo, pero, falta explicar cómo se produce trasvase entre la *exploración* que nos lleva a aprender y la *explotación* que nos lleva a usar lo aprendido. Aquí es donde surge la *avidez* o *greedy*, por su término en inglés que será el usado por ser el estándar de uso. El greedy se usa con un parámetro ϵ o ϵ-*greedy* el cual se encuentra $0 \leq \epsilon\text{-}greedy \leq 1$ y será responsable de decirnos con qué probabilidad usaremos una acción aleatoria (exploración) o la opción óptima según el modelo (explotación). A su vez, esta ϵ-*greedy* viene acompañada de un *declive* o *decay*, provocando que al inicio del aprendizaje todas las acciones sean de exploración, al desconocerse el entorno, y poco a poco según se vaya aprendiendo, se vayan explotando más las acciones que nos indique el modelo. Esta evolución se realiza actualizando (restando) al ϵ-*greedy* el decay a cada nuevo episodio. Matemáticamente, si una política se comporta de modo ϵ-*greedy* conseguirá aprender las acciones óptimas convirtiéndose a su vez en una política óptima.

Diferencia Temporal

Uno de los mayores avances que se encuentran en el aprendizaje por refuerzo es la *diferencia temporal (DT)* o *temporal-difference (TD)*. DT es capaz de aprender directamente de la experiencia sin la existencia de un modelo que explique las dinámicas del entorno. También los métodos de DT actualizan sus estimaciones basadas en otras estimaciones, sin esperar al resultado final. Esta técnica se llama *boostrap* y ayuda a aprender más rápido al realizar muchas actualizaciones en la política ante una única respuesta del entorno.

La DT puede dividirse, a su vez, en dos clases principales denominadas *on-policy* y *off-policy*. Comencemos por la on-policy. El primer paso es aprender una función acción-valor en vez de una función estado-valor. Lo

que hay que estimar es $q_\pi(s, a)$ para el comportamiento actual de la política π y para todos los estados s y acciones a.

Lo que se consideran son transiciones, dentro de un mismo episodio, entre pares estado-acción y aprender los valores de estos estado-acción, siguiendo la forma de la ecuación 3.18.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[C_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \quad (3.18)$$

Donde α es la velocidad a la que se aprende. Esta actualización se realiza después de cada transición desde un estado no final S_t. Si S_{t+1} es el estado terminal entonces $Q(S_{t+1})$ se define como cero. Este algoritmo que usa la quíntupla $(S_t, A_t, C_{t+1}, S_{t+1}, A_{t+1})$, que como nos enseña [29], es conocido como *Sarsa* que viene del acrónimo (State, Action, Reward, State, Action). Sarsa tiene la capacidad, con probabilidad 1, de alcanzar una política y a una función acción-valor óptima al visitar el espacio estado-acción un número infinito de veces.

En 1989, [32] desarrolló un algoritmo off-policy conocido como *Q-Learning* que se define como se ve en la ecuación 3.19.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[C_{t+1} + \gamma \min_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (3.19)$$

Recalcar que en el trabajo original [32] se maximizaba la recompensa, pero, al trabajar con costes tenemos que minimizarlo. A diferencia de lo que ocurría en Sarsa, en Q-Learning la función acción-valor Q aproxima el valor óptimo q_*, independientemente de la política seguida. También se consigue de esta forma converger de una manera más rápida al punto óptimo, aunque se requiera para que todos los pares acción-valor se actualicen que éstos continúen siendo visitados.

La diferencia es muy sutil entre ambos casos. Por un lado, Sarsa consigue la política óptima, pero, por otro lado Q-Learning obtiene el camino óptimo guiado por aquellas acciones óptimas. En la imagen 3.15 hay un implementación gracias a [29] de esta diferencia. En esta imagen se presenta un agente que tiene que aprender el camino entre el punto de partida S y el punto final G. Si por el camino se sale tiene una recompensa de -1, pero, si cae por el acantilado (the cliff), tiene una recompensa de -100. En este entorno, al contar ambos ejemplos con un porcentaje de movimientos aleatorios (para

hacer la exploración), Sarsa aprendería el camino más seguro ya que es el que ofrece la esperanza matemática más alta. En cambio, el algoritmo Q-Learning aprendería el camino óptimo ya que es el que maximiza las decisiones.

Como el QLearning alcanza decisiones óptimas será uno de los dos algoritmos que se van a usar para modelar la ejecución de órdenes. Para que quede más claro, se va a poder ver su pseudocódigo en el algoritmo 1.

Algoritmo 1 Q-learning

Inicialización $Q(s,a), \forall s \in S, a \in A(s)$ *y* $Q(estadofinal, \cdot) = 0$
for each episodio **do**
  Inicialización de S
  for each paso del episodio **do**
    Se elige una acción A usando la política de Q (ϵ-*greedy*)
    Se toma la acción A obteniéndose R y S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \, \text{mín}_a \, Q(S',a) - Q(S,A)]$
    $S \leftarrow S'$
    $\epsilon \leftarrow \epsilon - decay$
  end for
end for

Una vez vistos estos dos algoritmos, cabe diferenciar los algoritmos on-policy de los off-policy. Los primeros actualizan los valores Q usando el valor del siguiente estado s' y la política actual con la acción a. Los algoritmos off-policy actualizan sus valores Q usando el valor del siguiente estado s' y la siguiente acción a'. Llevándolo a un lenguaje más cercano, en on-policy se estima el retorno para el estado-acción asumiendo que la política continúa, mientras que en off-policy se estima el coste descontado futuro para un estado-acción.

En los algoritmos vistos hasta ahora se trata la minimización del coste en la construcción de las políticas objetivo. Esta minimización provoca una sobreestimación del valor Q ya que se utilizan los mismos datos para seleccionar la acción y para el cálculo del valor Q. No solo [29], sino otros autores nos documentan este suceso, como es el caso de [15].

Para solucionar este problema surge el *Double Q-Learning* que utiliza dos estimaciones distintas $Q_1(a)$ y $Q_2(a)$ del valor real $q(a)$. Se puede usar una de las estimaciones, $Q_1(a)$ para seleccionar la mejor acción y la otra estimación, $Q_2(a)$, para calcular el valor real $q(a)$. A pesar de tener dos estimaciones sólo una de ellas será usada en cada actualización duplicándose la memoria

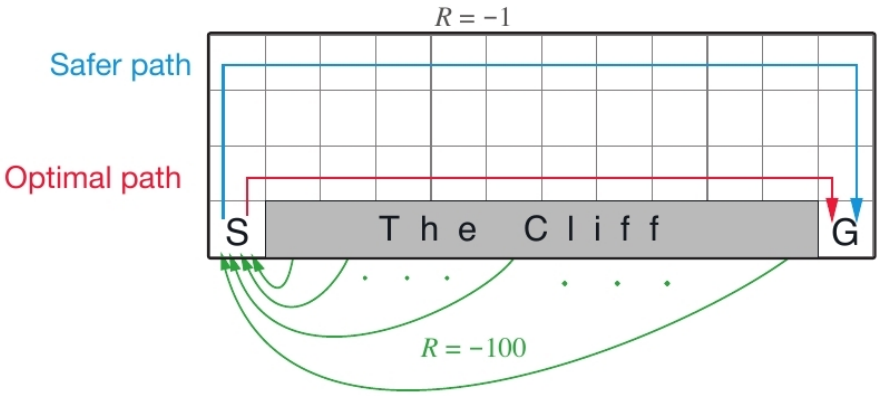


Imagen 3.15: Comparación del comportamiento de Sarsa (camino más seguro) contra el comportamiento de Q-Learning (camino óptimo)

necesaria para mantener el algoritmo, pero, sin alterar los cálculos necesarios a cada paso. La ecuación que gobierna el Double Q-Learning es 3.20 y el pseudocódigo es el que se encuentra en el algoritmo 2

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha[C_{t+1} + \gamma Q_2(S_{t+1}, \underset{a}{\operatorname{argmin}}\, Q_1(S_{t+1})) - Q_1(S_t, A_t)] \tag{3.20}$$

El enfoque inicial de implementación, máxime cuando el número de estados y de acciones no es muy grande, sigue una forma tabular, pero, este enfoque plantea dos problemas:

1. **Memoria:** muchos de los estados son posibles, mas, no siempre probables. Tener estos estados definidos y almacenados por si ocurren es muy costoso en términos de memoria. Puede haber entornos que utilicen realmente muy pocos de los estados posibles

2. **Actualización:** por definición para converger o bien a la política óptima o al camino óptimo, se deben de acceder infinitas veces a todos los estados

Veamos un ejemplo muy simple. Las tres en raya se calcula que tienen $3^9 = 19683$ estados diferentes[11], ahora solo habría que imaginar problemas más complejos como el que estamos trabajando.

[11]Habría que descartar aquellas posiciones imposibles como que ganen ambos jugadores o que estén ocupadas todas las casillas, pero, nos sirve para ilustrar la idea.

Algoritmo 2 Double Q-learning

Inicialización $Q_1(s,a), Q_2(s,a), \forall s \in S, a \in A(s)$ *y* $Q(estadofinal, \cdot) = 0$
for each episodio **do**
  Inicialización de S
  for each paso del episodio **do**
    Se elige una acción A usando la política ϵ-*greedy* en $Q_1 + Q_2$
    Se toma la acción A obteniéndose R y S'
    if Con probabilidad $= 50\,\%$ **then**
      $Q_1(S,A) \leftarrow Q_1(S,A) + \alpha[R + \gamma Q_2(S', \text{mín}_a\, Q_1(S',a)) - Q_1(S,A)]$
    else
      $Q_2(S,A) \leftarrow Q_2(S,A) + \alpha[R + \gamma Q_1(S', \text{mín}_a\, Q_2(S',a)) - Q_2(S,A)]$
    end if
    $S \leftarrow S'$
    $\epsilon \leftarrow \epsilon - decay$
  end for
end for

Redes Neuronales

Como se puede ver en [19], la forma más básica de pensar en una red neuronal es la de entenderla como una aproximadora de una función. Esta función puede ser tan sencilla como una función lineal de la forma $f(x) = 4x$, o algo mucho más complejo como puede ser transformar una imagen en una etiqueta que defina lo que aparecen en ésta.

A pesar de estar cada vez más en boga, las redes neuronales o redes neuronales artificiales no son un concepto novedoso, sino que surgieron en 1943 por McCulloch y Pitts [21]. Han pasado muchos años, pero, pueden que hayan vuelto para quedarse, como se explica en [12], se encuentran las siguientes razones:

- La cantidad de datos disponibles para entrenar las redes neuronales
- En problemas complejos es frecuente que el rendimiento de las redes neuronales sea superior a la de otras técnicas de aprendizaje automático
- La capacidad de cálculo hace posible entrenar redes neuronales en un tiempo asumible
- Las redes neuronales han entrado en un círculo virtuoso de financiación y desarrollo

- Una red neuronal puede aproximar cualquier función si es suficientemente grande

El *perceptron* es una de las arquitecturas más sencillas de redes neuronales, siendo inventadas en 1957 por Frank Rosenblatt. El perceptrón está basado en un tipo de neurona denominada *unidad lógica con umbral* o *threshold logic unit (TLU)*, la cual tiene tanto la entrada como la salida valores numéricos y cada conexión con las entradas están asociadas a un peso w_i, al igual que se puede ver en la imagen 3.16. La TLU calcula la suma ponderada de las entradas como $z = w_0x_0 + w_1x_1 + w_2x_2 + w_nx_n$.

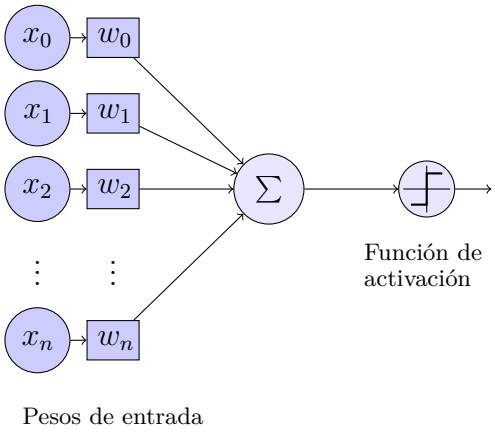


Imagen 3.16: Ejemplo de la arquitectura de un perceptrón

Tras esta suma ponderada se aplica lo que se llama *función de activación* pudiendo discretizar la suma ponderada. [19] nos da el ejemplo de dos posibles funciones de activación 3.21 usando un umbral de 0.

$$heaviside(z) = \begin{cases} 0 & si \quad z < 0 \\ 1 & si \quad z \geq 0 \end{cases} \qquad signo(z) = \begin{cases} -1 & si \quad z < 0 \\ 0 & si \quad z = 0 \\ 1 & si \quad z > 0 \end{cases} \tag{3.21}$$

Cuando se entrena este perceptrón, o en general cualquier red neuronal, lo que se consigue es encontrar los valores correctos para los diferentes pesos $w_0, ..., w_n$. Un perceptrón se compone de una capa de k TLUs, encontrando el caso especial en el que todas las neuronas de la capa están conectadas la cual se conoce como *capa totalmente conectada* o *capa densa*.

Atendiendo a las funciones de activación, pueden servir tanto para hacer clasificaciones propias del aprendizaje supervisado, como tareas de aprendizaje por refuerzo si se le proporciona la retroalimentación necesaria. Como se verá más adelante, es bastante parecido a lo que necesitamos, introducir una serie de valores numéricos y que aplicando una serie de pesos obtengamos un valor discreto indicándonos la acción a tomar.

El diseño de las redes neuronales no podía quedar aquí, siendo la continuación lógica apilar y conectar diferentes capas de perceptrones consiguiendo lo que se llama *Perceptrón Multi Capa* o *Multilayer Perceptron (MLP)*. Este MLP ya sí es capaz de realizar ciertas tareas de las que adolece el perceptrón, como es el caso de no ser capaz de aproximar un or excluso o XOR. Un MLP se compone de una capa de entrada, una capa capa de salida y una o más capas ocultas entre la capa de entrada y la de salida. En la imagen 3.17 se puede ver un ejemplo de MLP.

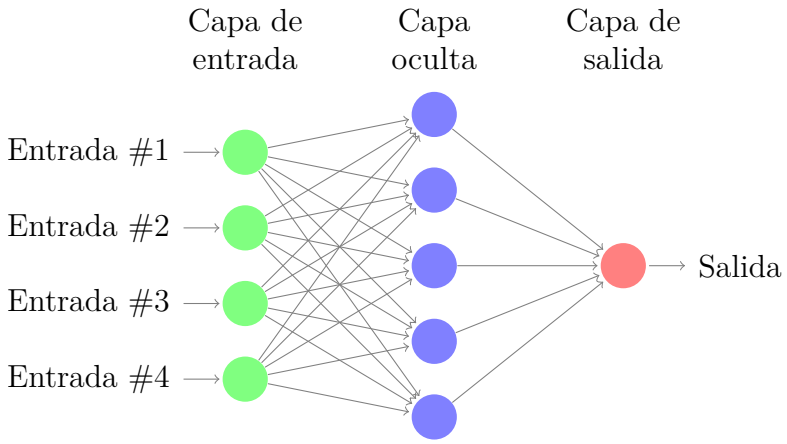


Imagen 3.17: Ejemplo de un MLP con una capa oculta

Cuando se apila una gran cantidad de capas ocultas es cuando se comienza a llamar *Red Neuronal Profunda* o *Deep Neuronal Network*, aunque éste es un concepto difuso ya que en inicios de los años 90 serían clasificados como una DNN contaba con dos capas ocultas y hoy en día se pueden encontrar perfectamente decenas o cientos de capas ocultas.

Aprendizaje por refuerzo con Aprendizaje Profundo

A pesar de la gran investigación en redes neuronales que se ha producido en las últimas décadas, especialmente al albor de las mejoras en la capacidad de computación, de almacenamiento y la transmisión de la información. Sin embargo, no fue hasta 2013 cuando una pequeña startup británica llamada

DeepMind realizó una adaptación del aprendizaje por refuerzo usando redes capaz de jugar a juegos de Atari[12] usando únicamente los píxeles de la pantalla como se puede ver en [22].

Al igual que lo que se vio en los elementos propios del aprendizaje por refuerzo 3.3, aquí se va a contar igualmente con la política, la señal de recompensa, la función de valor y el modelo del entorno, siendo lo que cambia diametralmente la implementación. Se va a pasar de una implementación realizada de un modo tabular (cada estado y cada acción se consultan en una tabla) a que sea una red neuronal profunda quien tome la decisión.

Uno de los problemas que plantea el Q-Learning es que no escala muy bien cuando se presenta a muchos estados y acciones. Si nos pusiéramos a intentar solventar el famoso juego Pac-Man, estaríamos hablando de unas 10^{45} estados distintos sin haber contado las posibles posiciones de los fantasmas, convirtiendo este problema en algo inmanejable.

Durante años, la solución pasó por realizar una aproximación vía reducción dimensional, como la introducida en el aprendizaje no supervisado. En 2013 DeepMind demostró que las redes neuronales profundas podían hacerlo mucho mejor, especialmente en problemas complejos, y sin tener que hacer esa reducción dimensional de los atributos. Cuando se usa una red neuronal profunda para aproximar los valores Q, se le llama *Red Profunda Q* o *Deep Q-Network (DQN)* y cuando se usar una DQN para aproximar un problema en Q-Learning se le llama *Aprendizaje Q Profundo* o *Deep Q-Learning*.

Para representar la diferencia entre la aplicación del modelo tabular o el de Deep Learning es sólo de implementación, ya que tanto la entrada como la salida de los datos ha de seguir el mismo formato. En otras palabras, en ambos casos se utilizan los estados como entradas y como salida el conjunto de las acciones posibles junto con un valor numérico que determina el valor Q, siendo aquella acción con un menor valor el que ha de ser elegido como la correcta y por ende, la que será tomada. De modo gráfico, se puede observar 3.18.

En el formato tabular simplemente hay que buscar en una tabla, siendo en el ejemplo con 2 posibles estados y 5 acciones. Para un estado determinado se tendría que tomar la acción que tuviera un valor menor. Este formato tabular tendría como dimensión el número variables que se contemplen en los estados más uno para la acción, que por ejemplo para analizar el Percent

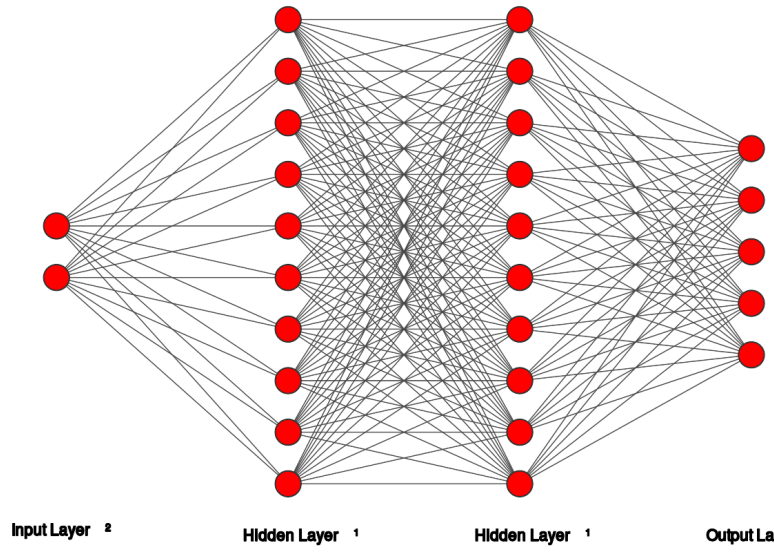
[12]Atari es una marca de juegos y consolas arcade creada en 1972 y entre cuyos juegos destacan el pong, asteroids o el Pac-Man

On Volume serán dos variables (liquidación del algoritmo y del modelo de aprendizaje) más uno de las acciones, sumando un total de 3 dimensiones.

Para la red neuronal tendremos tantos nodos de entrada como variables integren cada estado, después se encontrarían las capas ocultas y como salida tendremos una capa con tantos nodos como posibles acciones, quedándonos de nuevo con el nodo con un valor menor.

	A_0	A_1	A_2	A_3	A_4
S_0					
S_1					

(a) Q-Learning tabular



(b) Deep Q-Learning

Imagen 3.18: Comparación Q-Learning tabular y con Deep Learning

Conclusiones

Tras haber estudiado las clases de aprendizaje, a pesar de encontrar ciertas similitudes no terminaban de encajar con el problema de la ejecución de órdenes que tenemos entre manos.

En el momento que encontramos el aprendizaje por refuerzo empezamos a ver la luz. A pesar de esto, no es oro todo lo que reluce y aparecieron ciertos problemas de escalabilidad que fueron solventados por las redes neuronales. No hemos sido los primeros en tener esta idea, ya que [23] implementó satisfactoriamente este problema con Double Deep Q-Learning. Basándonos en este trabajo donde se utilizan capas de redes neuronales densas, donde todas las neuronas de cada capa están conectadas con todas las neuronas de la siguiente capa (al igual que en la imagen 3.18), se ha simplificado usando solo 2 capas ocultas de 32 neuronas cada una que serán conectadas con la entrada (los estados) y con la capa de salida (las acciones).

Parte II

Desarrollo del proyecto

Aspectos relevantes del desarrollo del proyecto

En este capítulo se va a estudiar todo lo necesario, a nivel técnico, que ha sido imprescindible para llevar a cabo el proyecto. Se ha estructurado en dos partes, una más orientada a la infraestructura, tanto a nivel de hardware como del software necesario para gestionar esta recogida de información, y por otra parte los elementos necesarios para realizar la investigación.

5.1. Infraestructura y Proceso ETL

La infraestructura del proyecto consta de los elementos usados para la recogida de información, su distribución, su posterior procesamiento y por último su consumo y uso tanto para la exploración de los datos como para la explotación en el entrenamiento del modelo de aprendizaje por refuerzo. Una de las fases que requieren un mayor nivel de detallismo y trabajo, aunque a veces no se le da la importancia que merece es la *ETL* que proviene de *Extracción*, *Transformación* y *Limpieza* de los datos.

A diferencia de lo que ocurre en trabajos puros de desarrollo donde la arquitectura de software, las metodologías de desarrollo o incluso la arquitectura de hardware y su conectividad tienen una vital importancia, en esta ocasión donde nos encontramos ante un trabajo más relacionado con la investigación, la flexibilidad y la adaptación a las pruebas, los errores y las mejoras que se han ido encontrando por el camino tienen una mayor relevancia, así como, la presencia de una fuerte base científica tanto económico-financiera como tecnológica.

Gran parte de la infraestructura con la que se contaba se ha diseñado por y para el proceso ETL que se detallará a continuación.

Datos disponibles

Como en cualquier trabajo de data science o de Big data, los datos son una de las partes más importantes, si no la más importante, ya que si estos carecen de calidad o no representan bien el problema de estudio todo el trabajo carece de sentido pudiendo llegar a conclusiones erróneas.

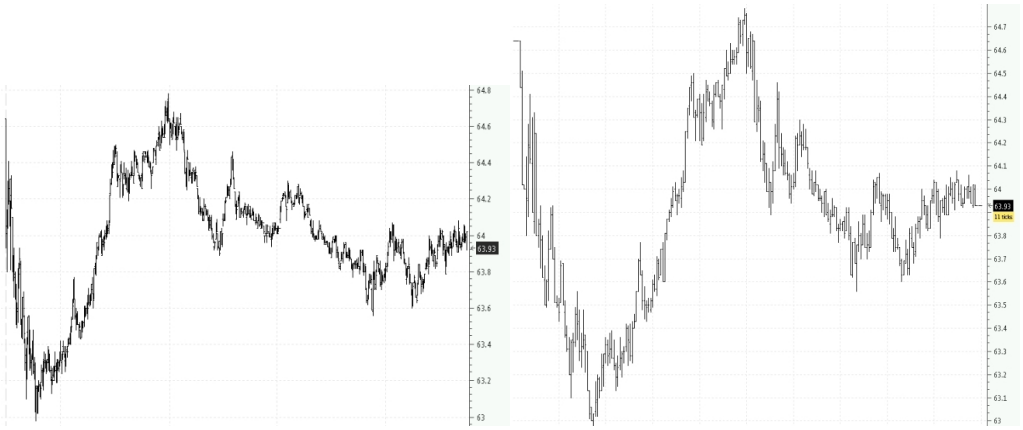
En este caso los datos con los que se contarán serán:

- Precios de cotización histórica por unidad de tiempo
- Datos de cotizaciones por cada cambio de precio
- Por cada nivel de precios dentro del libro de órdenes[13]:
  - Volumen de compradores (para el Bid) o de vendedores (para el Ask)
  - Número de órdenes
  - Precio del nivel

Cuando en finanzas se trabaja con cotizaciones históricas, el estándar es trabajar mediciones sobre un espacio de tiempo como puede ser un día, una hora o un minuto. Dentro de estas mediciones se guardan los llamados *Apertura, Máximo, Mínimo* y *Cierre*. Esto significa que si graficamos la evolución de un activo financiero a lo largo de un día, se crea una abstracción en la que no se obtienen todos los datos del día, sino, a qué precio inició el día (*Apertura*), cuál el precio más alto que alcanzó a lo largo del día (*Máximo*), el precio más bajo al que cotizó durante el día (*Mínimo*) y por último a qué precio acabó el día (*Cierre*), sin tener en cuenta el recorrido por el camino. Como buena serie de datos, vienen asociado a un valor temporal y es muy común que a estos cuatro datos se le añada el *Volumen*. El volumen es el número de acciones o contratos que se han negociado durante un periodo de tiempo (entre la Apertura y el Cierre). El volumen, como se ha dicho, no se mide en dinero sino en acciones cruzadas.

[13]El libro de órdenes (o Limit Order Book) es un registro donde se encuentran las órdrnes límites de todos los participantes para un activo financiero y cada uno de los niveles de precio

Esta forma de almacenar los datos con la Apertura, Máximo, Mínimo y Cierre se suelen representar gráficamente a través de lo que se denominan *gráficos de barras* o con *gráficos de velas japonesas*, también conocidos como *gráficos de velas*.



(a) Airbus en minutos (b) Airbus en ticks

Imagen 5.19: Series de datos de Airbus

El segundo tipo de datos con los que podemos trabajar es con las cotizaciones obtenidas por cambios de precio. Cuando un activo está cotizando y se produce una compra-venta pueden ocurrir dos situaciones; o bien se realiza al mismo precio que la útima transacción, o bien, lo hace a un nuevo precio. Este tipo de velas tiene la misma información y representación gráfica que la anterior, lo que cambia es el timing. En la imagen 5.19 se pueden ver las cotizaciones de un día en la empresa Airbus graficado en gráficos de barras de 1 minuto y de 100 ticks respectivamente. Mientras que en el primer caso la duración temporal de cada representación es de un minuto, en el segundo la representación es de 100 cambios de precios independientemente del tiempo que tarde en ello.

El último tipo de datos que vamos a referenciar es el que tenemos en el *Libro de Órdenes*, *Limit Order Book* o simplemente *LOB*. Lo que obtenemos con el LOB es una fotografía donde encontraremos cuántos compradores y vendedores hay por cada nivel precios y cuánto quieren comprar y vender éstos en un momento del tiempo. Esta información será almacenada por cada cambio de precio de modo que se creará un *vídeo* con la evolución de compradores y vendedores a lo largo del tiempo mediante las fotografías que se irán almacenando.

Imagen 5.20: Libro de órdenes del futuro de la deuda alemana a 10 años (Bund)

En la imagen 5.20 se puede ver un ejemplo de los datos que tendremos accesibles, en este caso sobre los futuros sobre la deuda pública alemana a 10 años. Si atendemos a las V's del Big Data (Volumen, Velocidad, Variabilidad y Valor), afectarían a todas menos a la variabilidad (ya que están estructuradas). Como se deduce de la ecuación (3.2), la probabilidad de la ejecución de una orden es inversamente proporcional a la distancia al precio de cotización, por ello, el Valor marginal de cada nuevo dato según nos alejamos en el LOB es cada vez menor. A pesar de ello, en su conjunto pueden ofrecer un mayor conocimiento del problema.

Extracción de datos

Uno de los puntos más críticos en este trabajo, y sobre lo que se sustentará todo lo demás será su materia prima, los datos. Éstos han sido recogidos desde abril de 2019 hasta agosto de 2020. En las imágenes 5.21 y 5.22 se puede observar cómo se recogen los datos y los campos que se pueden obtener, aunque a continuación se verá más en detalle.

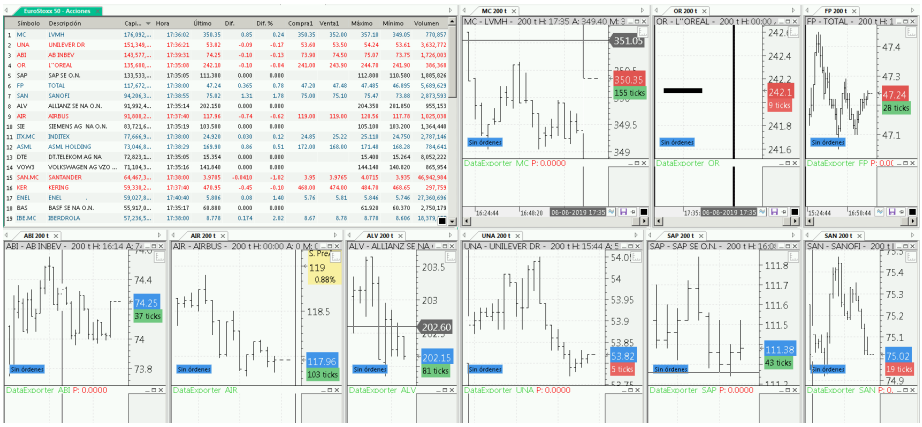


Imagen 5.21: Recogida de datos con VisualChart

En el caso de 5.22 lo que podemos observar es el denominado futuro sobre el Dax 30 alemán, que representa a las 30 mayores empresas alemanas. Sobre este índice, podemos encontrar a la izquierda el histórico de cotizaciones cada 5 minutos. A la derecha lo que obtenemos son los compradores (en azul) y los vendedores (en rojo) por cada nivel de precios. Así mismo, también se pueden ver cuántos compradores y vendedores hay, a la vez que el volumen que desean comprar y vender en su conjunto por cada nivel de precios. Para entenderlo mejor, esto significa que en el nivel de 12006,5 hay un total de 5 vendedores que desean vender 9 contratos del Dax.

Imagen 5.22: Cotización del futuro del índice bursátil alemán, el Dax 30

Esta recogida de información, que como se ha visto, se refiere a los datos de cotización y la información relativa al libro de órdenes en tiempo real para cada uno de los activos. En la imagen 5.22 se puede observar la cotización en tiempo real para el futuro del índice bursátil alemán, el Dax 30, mientras

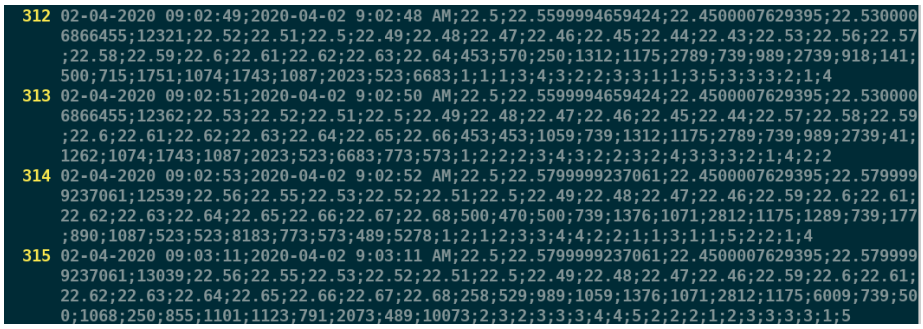


Imagen 5.23: Fichero con la cotización de Inditex almacenada

que en la imagen 5.23 se puede ver cómo se traducen estas cotizaciones en un fichero en formato *csv* donde cada línea es una medición. De este modo se puede observar cómo se traduce lo que teníamos en la pantalla en algo que pueda ser procesado.

Para recoger esta información se ha hecho uso de la plataforma VisualChart. Esta plataforma nos permite tanto el manejo y análisis de los datos en la propia aplicación como la ampliación de su usabilidad mediante código en *.NET* y más concretamente en *C# y en VB .NET*. Para la exportación de la información ha sido usado C#.[14]. A diferencia de lo que ocurre en muchos proyectos de Big Data, donde la fuente de datos es accesible mediante una API que mediante una autenticación puede ser consumida casi con cualquier lenguaje y sistema operativo, el acceso a los datos de cotizaciones se realiza mediante esta aplicación, VisualChart, la cual sólo funciona con Windows como sistema operativo. De esta forma se convierte en un requisito.

Data Lake

En 2010, James Dixon CTO de Pentaho[15], escribió un artículo[8] en el que tras un evento discutía el uso de Hadoop por parte de las diferentes empresas con las siguientes conclusiones:

- El 80-90 % de las empresas trabajan con datos estructurados o semi estructurados
- La fuente de datos es típicamente una única aplicación o sistema
- Los datos no suelen conllevar transacciones comerciales

[14] Se puede encontrar más información de la plataforma en https://www.visualchart.com

[15] Pentaho es un software de inteligencia de negocio.

- Hay preguntas que hacerle a los datos
- Puede haber muchas preguntas que puedan surgir en el futuro
- Hay muchas comunidades de usuario que tienen preguntas sobre los datos
- Los datos tienen una escala o volumen diario que no encaja con el uso de una base de datos

Habiendo encontrado todos estos puntos en común y las limitaciones técnicas del procesamiento de información sin haber establecido primero las preguntas a los datos, concibió la idea de *data lake* como una solución óptima. Inventó el data lake como un gran cuerpo de agua en un estado natural, sin procesar, siendo este lago alimentado por las aguas de las diferentes fuentes y consumido por los diferentes usuarios y aplicaciones.

Esto es precisamente lo que nos ha ocurrido. Parafraseando los puntos que establecía Dixon en [8] se pueden concluir los siguientes puntos:

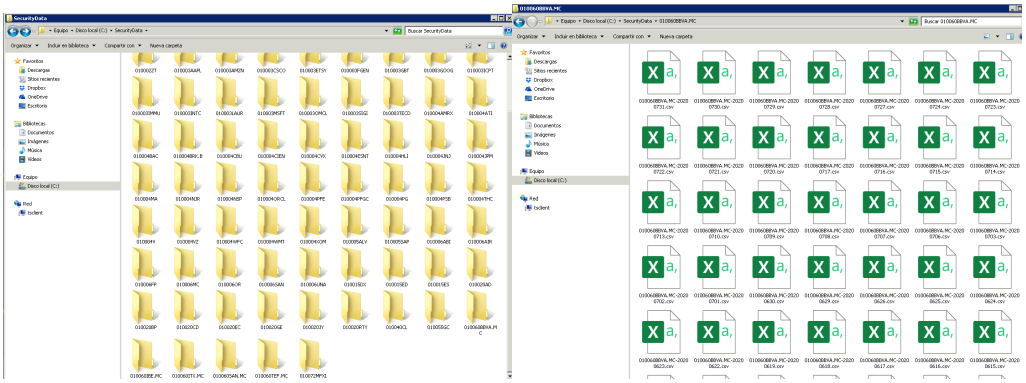
- Los datos que recibimos son semiestructurados
- Tenemos una única fuente de datos que nos alimenta[16]
- Nuestra pregunta es cómo optimizar la ejecución de órdenes
- No todas las preguntas son resueltas, por ejemplo, con esta información se pueden crear porfolios de acciones, estudiar la relación entre los diferentes activos, crear algoritmos de market making, etc.
- No somos los únicos ni en plantearse el problema que tenemos entre manos ni el resto de preguntas. Hay millones de personas que se plantean estas cuestiones
- Se ha utilizado un subconjunto tremendamente pequeño de activos. Solo la bolsa de Nueva York (New York Stock Exchange o NYSE) cuenta con unas 2400 compañías

[16] En realidad tenemos más de una fuente de datos. Cada uno de los activos puede cotizar en una Bolsa o Exchange distinto siendo éstos la fuente original de los datos, sin embargo, VisualChart se encarga de agregar toda esta información y servirla desde un único punto de acceso.

El universo que se ha utilizado han sido los datos de cotización de Inditex, BBVA, el Banco Santander, Telefónica, SAP, Allianz y Airbus.

Como se ha visto a lo largo del trabajo, las diferentes transacciones bursátiles se producen de forma asíncrona, pudiendo llegar en cualquier momento. Los códigos que se ejecutan en VisualChart lo hacen de forma asíncrona, por lo que no presenta un problema a mayores de sincronización, convirtiéndose en una cuestión trivial. Como estos datos pueden llegar de manera asíncrona y ser procesados de esta forma, lo que se ha optado, para minimizar las operaciones de escritura en el disco duro, es guardar en una lista todas cotizaciones e imprimirlas cada vez que se alcancen los 1000 elementos en la misma.

Los datos extraídos formarán una carpeta por cada uno de los activos financieros y a su vez, un fichero por cada día como se puede ver en la imagen 5.24.



(a) Carpetas por activo (b) Ficheros generados en BBVA

Imagen 5.24: Datos almacenados en el servidor

Servidores y decisiones de diseño

Como para la extracción de datos se tenía como requisito el uso de Windows como sistema operativo, se ha optado por seguir los siguientes pasos:

- Contratar un servidor dedicado
- Instalar y configurar VMWare ESXi para virtualizar servidores
- Crear un servidor privado virtual con Windows Server 2008

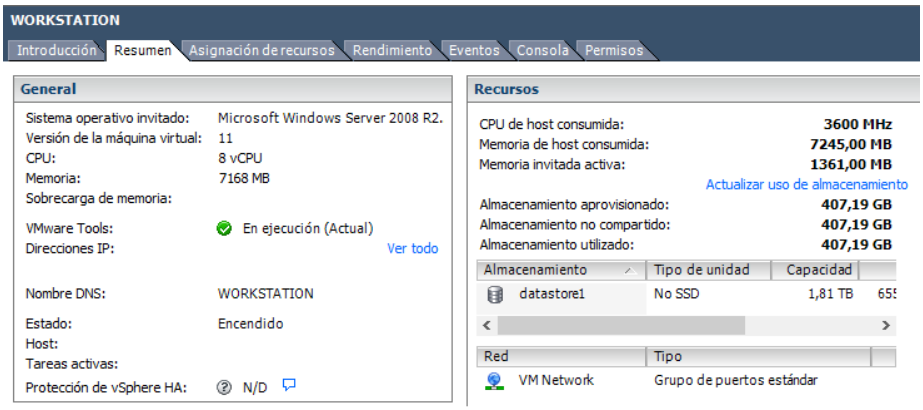


Imagen 5.25: Datos del Servidor Privado Virtual

- Instalación de VisualChart
- Instalación de Microsoft Visual Studio
- Creación del código que fuera capaz de exportar a un fichero los datos de las cotizaciones obtenidos de manera asíncrona
- Aplicación del código a todos los activos de los que se van a obtener los datos
- Creación de un servidor FTP para poder acceder al data lake

En base a la buena experiencia presentada, el servidor se contrató en https://www.wholesaleinternet.net/. Esta compañía ofrece los servidores de manera muy sencilla de configurar con ESXi. ESXi es un sistema operativo especialmente basado en una distribución de Linux llamada Red Hat y desarrollado por la empresa VMWare, el cual está diseñado para la virtualización de servidores. Por esta razón, es la mejor opción.

Pese a su antigüedad, se ha elegido Windows Server 2008 por la contrastada estabilidad que presentaba con VisualChart. En la imagen 5.25 se puede ver el servidor, así como sus características de hardware, donde se le han asignado 8 cores, 7168 MB de memoria RAM y 400 GB de disco duro.

Tras la instalación de VisualChart y de Microsoft Visual Studio se desarrolló el código necesario para poder extraer los datos de los activos como se ve en la imagen 5.26. En base al tamaño de los ficheros generados los primeros días en aquellos activos que más transacciones tienen y por tanto, mayor número de muestras habría que almacenar la memoria asignada a

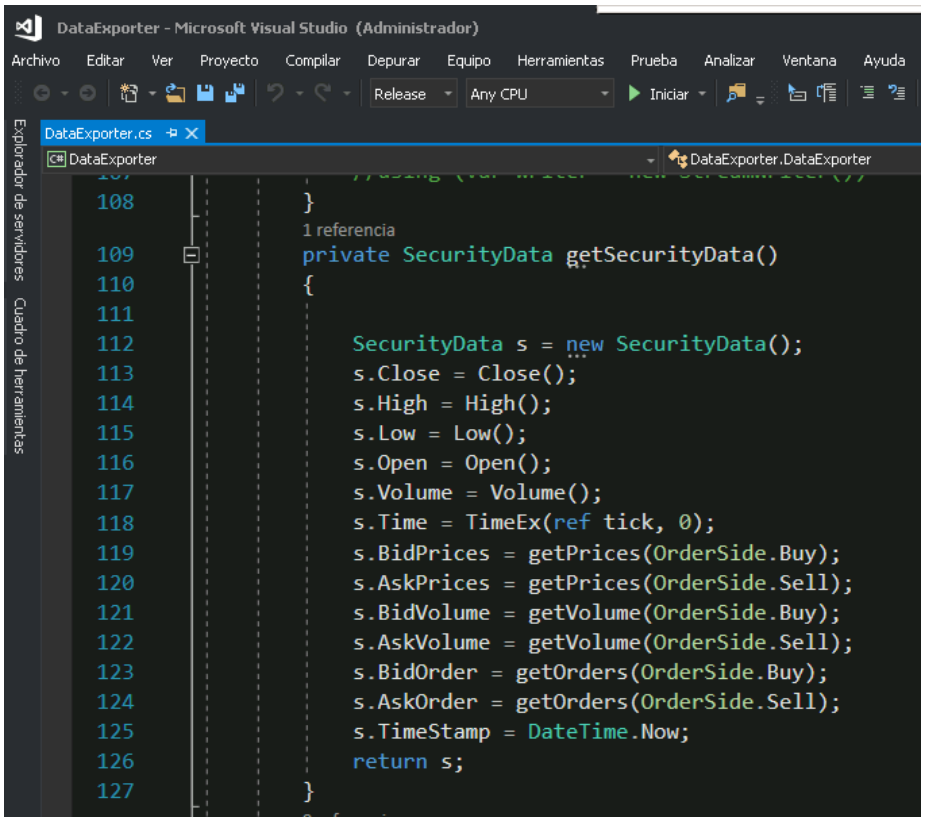


Imagen 5.26: Fragmento del código para exportar las cotizaciones en tiempo real

la máquina virtual es suficiente. El almacenamiento se ha realizado almacenando en una carpeta por activo, siendo el nombre de la carpeta el del código del activo en VisualChart, ya que sirve a modo de identificador único. Por ejemplo, en el caso de Inditex el código es 010060ITX.MC. Dentro de estas carpetas los datos de cada día, con horario de Madrid, se han ido almacenando de manera independiente.

De querer escalar el procedimiento tanto a lo largo del tiempo como del número de activos de los que se quisiera recoger la información, habría que pensar en implementar un almacenamiento en la nube, o implementarlo en algún otro tipo de arquitectura. Por ejemplo, se podría reservar otra máquina en el servidor dedicado que mediante una tarea en batch moviera todos los ficheros generados en el día y actualizara de ese modo el data lake, pero, en este caso no ha sido necesario.

Para la distribución de los datos para su consumo se ha incorporado, dentro del servidor privado virtual que aloja los datos originales, un servidor

FTP de modo que se pudiera acceder desde cualquier lugar. Ante la posibilidad de caída del servidor privado virtual se han hecho copias de seguridad tanto en Dropbox como en el propio host, realizándose siempre estas copias durante el fin de semana ya que es el momento en el que ningún activo cotiza y la máquina tiene menor carga de trabajo.

Transformación de los datos

Dentro de todo el proceso de ETL la transformación de los datos ha sido la parte menos compleja. En la extracción hay que montar toda la infraestructura, el mantenimiento de la misma, así como, el propio desarrollo de exportador de los datos o las copias de seguridad. En la limpieza, como se verá, se pueden encontrar diversos fallos siendo alguno de ellos no tan evidente de encontrar.

La transformación de los datos, que en cierta medida se ha realizado de manera conjunta con la limpieza de los datos, ha tenido un punto clave y es la distancia temporal entre las muestras.

Veamos la imagen con una muestra del libro de órdenes del Banco Santander 5.27. Si en el mismo segundo (puede haber más de una transacción por segundo que quede registrada), hay un cambio en el nivel de precios al que se ha realizado la transacción, por pequeña que sea, ni los oferentes ni los demandantes de acciones del Banco Santander han cambiado. Volviendo al ejemplo con el inmobiliario por el que se comenzó, es como si en un momento del tiempo hay 2 casas vacías, simulamos una compra y al día siguiente tenemos en cuenta que sigue habiendo esas 2 casas vacías. Para minimizar este problema, lo que se ha hecho son los siguientes pasos:

- Acumular el volumen cruzado de aquellas muestras que se separen en menos de 10 segundos
- Eliminar las muestras que que se separen en menos de 10 segundos
- Crear un nuevo campo con el volumen acumulado por el camino.

Este nuevo campo será el utilizado tanto para modelar el algoritmo Percent On Volume (POV) y el Volume Weighted Average Price (VWAP). Este es un problema inherente a las simulaciones que estamos realizando y que sólo puede solventarse haciendo el entrenamiento y su puesta en producción con dinero real.

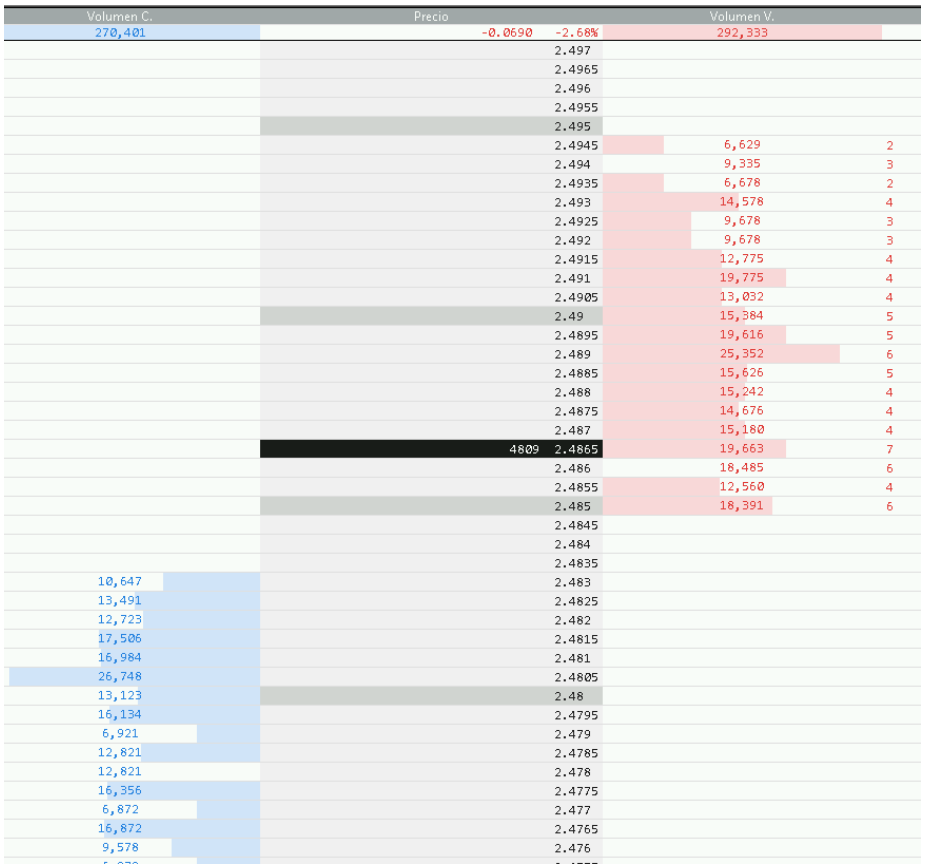


Imagen 5.27: Libro de órdenes del Banco Santander

Hay que recordar que estamos modelando operaciones cruzadas en un momento temporal, las cuales tienen unas características determinadas, pero, que no tienen por qué repetirse en tiempo y en forma. La conclusión de esta aseveración es que estamos intentando representar el posible comportamiento que tendrían los algoritmos de aprendizaje en producción y aunque se tomen todas las medidas posibles para minimizar la discrepancia entre lo modelado y la realidad, siempre habrá diferencias. Por esta razón, hasta que no se ponga en producción con compras y ventas reales, con el consecuente aprendizaje que tienen los algoritmos de aprendizaje reforzado, no se tendrás resultados definitivos.

Limpieza de los datos

Gracias a la exploración de los datos es como se consigue, en muchos casos y este no iba a ser menos, la detección de posibles errores en los datos. A pesar de ser tratados por una plataforma, que se traten de datos semiestructurados

(mucho más fáciles de gestionar que datos no estructurados) y que vengan bastante limpios en resumen, no quita que también haya errores. Entre los principales que se han encontrado están:

- Horarios de cotización falsa
- Valores de cotizaciones no válidos
- Valores erróneos en el volumen

Los diferentes activos tienen un horario de cotización y por tanto, están definidos. Si hay un horario fuera este marco temporal, se debe indudablemente a un error. En el caso español, la bolsa se encuentra abierta entre las 9:00 y las 17:30. Toda muestra fuera de este horario debe ser borrado. Lo mismo debe ocurrir con los festivos y los fines de semana, aunque no se han encontrado errores con respecto a la fecha.

El segundo error encontrado son los valores de cotización no válidos. Estos son más complejos de encontrar que los anteriores, ya que sólo necesitaban filtrar por horario. En este caso, lo que se intenta evitar es que dada una cotización de 10 € al siguiente instante temporal no pueda valer 100 €. Estos errores se pueden solventar dibujando los datos recolectados y para solucionarlo es casi un método quirúrgico, al ser datos puntuales que hay que eliminar.

Este tercer tipo de errores puede ser el más difícil de detectar y solucionar. El volumen negociado pertenece al conjunto de los números naturales $\mathbb{N}$, por lo que cualquier cifra arbitrariamente grande es válida. Los dos métodos implementados, como se verá más adelante, con respecto a la ejecución de órdenes son el POV y el VWAP, siendo necesario para el VWAP la estimación del volumen esperado a una hora y minuto concreto. Para esta estimación se han eliminado los valores atípicos por encima del percentil 99.9, siendo los valores superiores no eliminados sino truncados a ese percentil 99.9. De esta forma, se disminuyen distorsiones sin bajar la relevancia estadística de los datos.

Toda esta limpieza se ha realizado a mano.

5.2. Desarrollo

Esta es la sección del trabajo de mayor importancia. Aquí se estudian e implementan todos los conceptos vistos en los conceptos teóricos. La clave

son las experiencias y el aprendizaje que se ha obtenido por el camino. Durante todo el desarrollo tiene importancia la exploración de los datos

Exploración

La exploración de los datos, a pesar de lo que se puede pensar inicialmente, no se trata de una fase estanca e inicial de la que hay que olvidarse después, sino que en muchos casos es recomendable volver en el afán de dar un paso más en la investigación Es natural comenzar por aquí, aunque se conozca a priori el ámbito de trabajo, ya que hay que conocer fehacientemente lo que se tiene entre manos.

Primero vamos a analizar dos de los campos más importantes que podemos encontrar: el volumen negociado y el retorno. Para rememorar, se entiende como volumen negociado la cantidad de acciones o contratos intercambiados en un momento dado. Si una muestra nos da como volumen 100 y el activo cotiza a 20 € significa que se han intercambiado 100 acciones por una cantidad total de 2.000 €. La otra variable que se muestra es el retorno, definido como:

$$Retorno_n = \frac{Cotización_{n+1} - Cotización_n}{Cotización_n} * 100 \tag{5.22}$$

Este análisis se puede observar en la tabla 5.2 y en las imágenes 5.28 y 5.29. Como se puede apreciar tanto en la tabla como en las imágenes, no estamos hablando de distribuciones normales, ni cercanas a ello.

La curtosis es una medida de forma que determina la concentración en torno a la media de la distribución. Cuando los valores son muy altos, significa que la distribución es *leptocúrtica* y los valores están muy concentrados, en cambio, si los valores son muy bajos se habla de una distribución *mesocúrtica*. En el caso del volumen y de los retornos de Inditex tenemos un exceso con respecto a la distribución normal de 1003.64 y 64.05 respectivamente. Como ya se adelantó, las distribuciones financieras no suelen ser normales, ni cerca, y aquí se ha visto cómo tienen distribuciones mucho más concentradas. Solo hace falta ver que se ha necesitado utilizar la escala logarítmica para representar los histogramas para que se pudieran distinguir las distribuciones.

El sesgo lo que mide es cómo de asimétricos con respecto a la media son los datos. Si el valor del sesgo es 0 nos dice que los datos están centrados en la media, los valores negativos nos dicen que hay una asimetría hacia la izquierda, mientras que cuando vemos un valor positivo nos dice que

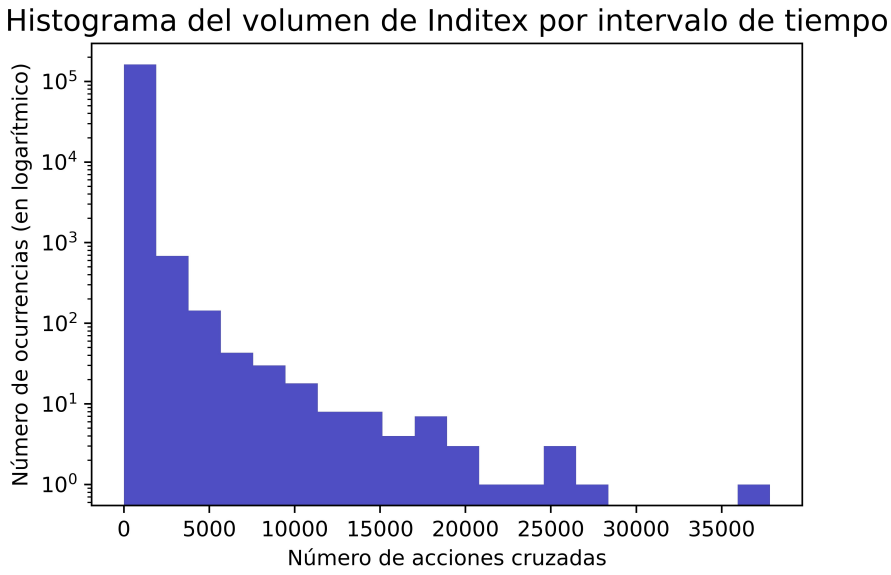


Imagen 5.28: Histograma del volumen en Inditex

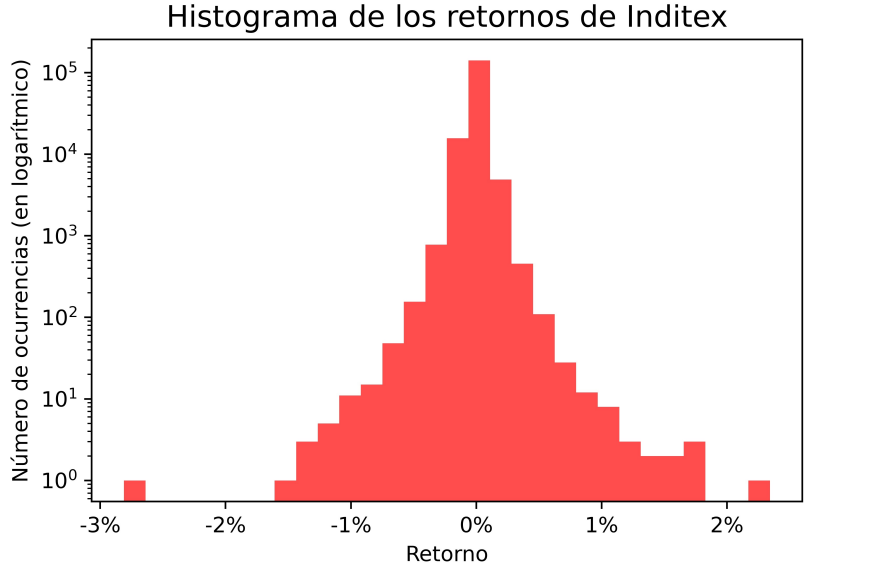


Imagen 5.29: Histograma de los retornos en Inditex

Métrica	Volumen	Retornos
Muestras	163290	163120
Media	279.96	-0.0002 %
Desviación estándar	446.08	0.0697 %
Valor mínimo	1	-2.8103 %
Percentil 25	95	-0.0362 %
Percentil 50	200	0 %
Percentil 75	389	0.03623 %
Valor máximo	37829	2.3426 %
Curtosis	1003.64	64.05
Sesgo de asimetría	22.18	0.04

Tabla 5.2: Estadística descriptiva del volumen negociado y los retornos porcentuales

estamos ante una asimetría hacia la derecha. El sesgo que encontramos en la distribución del volumen y de los retornos en Inditex es de 22.18 y de 0.04 respectivamente. Esto nos dice que el volumen está tremendamente sesgado a la derecha y que, por tanto, es muy común encontrar que en un momento temporal haya mucho volumen negociado. En el caso de los retornos lo que nos explica es que hay cierto sesgo hacia los valores positivos, pero, muy pequeño. Esto también se deduce de que el percentil 50 de la distribución se sitúe en el 0 % como se ve en la tabla 5.2. Este análisis es imprescindible por dos razones, el primero, para entender los datos de los activos sobre los que se está trabajando y el segundo para encontrar posibles errores. Como un posible error, se podría encontrar caídas o subidas de valoración arbitrariamente grandes o volúmenes negativos que es imposible que se produzcan.

Explotación

En esta sección se estudiarán la explotación de los datos, o lo que es lo mismo el uso de los datos para crear, optimizar y adaptar los algoritmos de aprendizaje de cara a la ejecución de órdenes.

Aproximación de funciones

Como ya se adelantó anteriormente, el aprendizaje por refuerzo es capaz aprender tareas gracias a los costes o recompensas que se le vayan propor-

cionando según los estados y las acciones. Esta tarea puede ser encontrar el camino para alcanzar la máxima puntuación en un juego, como demostraron en [22], o de modo general una función.

Cuando alguien se inicia en la inversión lo primero que viene a la mente es intentar maximizar las ganancias, consiguiendo una ventaja inherente al mercado, ya sea por la lógica (mejores decisiones) o por la implementación. Este caso no iba a ser menos.

El primer intento, con [23] como base, se implementó el aprendizaje por refuerzo con Double Q-Learning de forma tabular inicialmente. Los atributos que generaban los estados con los que se contaban eran:

- Volatilidad
- Segundos hasta el final del día
- Desequilibrio de posiciones en el bid y en el ask

La *volatilidad* se medía como el cambio porcentual entre el precio de cotización y el del inicio del día. El resultado se dividía en 20 partes por percentiles, quedando un 5 % de los datos en cada una de las partes. Los *segundos hasta el final del día* se discretizaban, al igual que caso anterior en 20 partes iguales por percentiles. Por último, el desequilibrio se medía como el volumen total de las 3 primeras posiciones del bid contra las 3 primeras posiciones del ask. Como la primera posición es más importante que la segunda, y así sucesivamente, se ponderaron de la forma que el desequilibrio D en un instante t del tiempo quedaba medido como:

$$D_t = \frac{3ask_1 + 2ask_2 + ask_3}{3bid_1 + 2bid_2 + bid_3} \tag{5.23}$$

Como en los otros casos, también se dividió el resultado en 20 partes por percentiles. Con respecto a las acciones se determinaron 10 posibilidades, partiendo de no hacer nada (no adquirir ninguna acción), hasta el 10 (adquirir todas las restantes). Esto nos da como resultado una tabla de $20*20*20*10 = 80{,}000$ posibilidades y 8,000 estados distintos. La tabla era bastante grande y la mayoría de los casos no eran visitados, como se puede ver en la imagen 5.30, con el coste de almacenamiento que provoca.

El otro punto clave, no es solo ya la configuración de los diferentes estados, sino la función que se quiere aproximar. Como se introducía anteriormente, la

Imagen 5.30: Estados sin visitar

Hiperparámetro	Inicio	Fin	Salto
Descuento	0	1	0.1
Ratio de Aprendizaje	0	1	0.1
Decay	1	9	2

Tabla 5.3: Optimización de hiperparámetros

primera tentación es y fue intentar minimizar el precio medio de adquisición. En teoría es una buena idea, pero, puede que en la práctica no tanto.

Tras haber montado todo lo necesario y un inicio no demasiado prometedor, se optimizaron los hiperparámetros del ratio de aprendizaje, a través del cual el algoritmo actualiza los valores Q más o menos rápido, y el descuento por el que se pondera más o menos los futuros pasos dentro del episodio. Esta optimización se llevó a cabo de manera iterativamente con la configuración que se ve en la tabla 5.3. Que el decay tenga valor 1 significa que alcanza el valor de 0 y por tanto todas las acciones las toma algoritmo, sin ninguna acción aleatoria, en el último episodio, mientras que el valor de 9 lo alcanzaría al llegar al primer noveno de los episodios proyectados.

A pesar de que toda la optimización tenía mejores precios medios al final del aprendizaje, no eran nada significativos, como se puede apreciar en la imagen 5.31 donde se puede ver el entrenamiento con la menor pendiente en una regresión lineal.

La pendiente de la regresión lineal significa que el coste medio ha variado a lo largo de los episodios, comparando el precio por acción al que se iniciado la compra y el precio medio ponderado obtenido al final del episodio. El

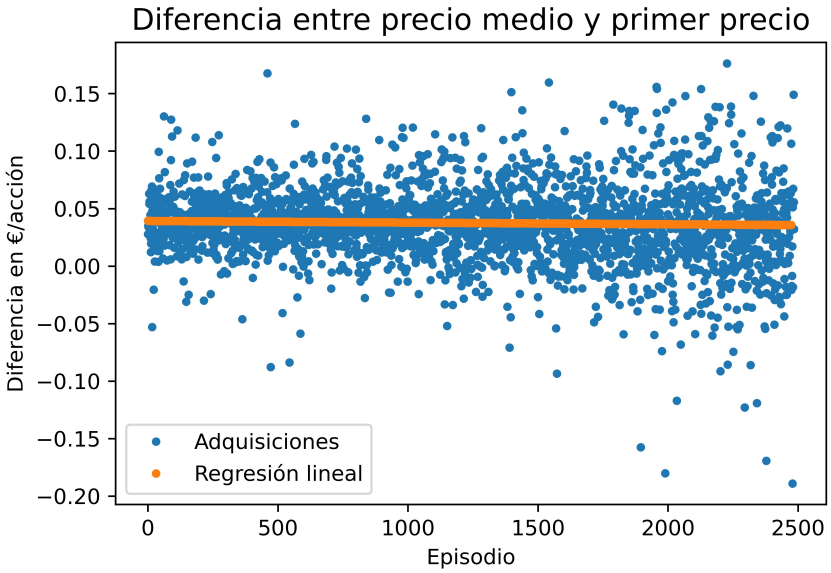


Imagen 5.31: Diferencia del precio medio de la adquisición contra el primer precio encontrado

resultado es que la menor pendiente encontrada es de $-1,42 * 10^{-8}$, lo que es virtualmente igual 0.

Los diferentes algoritmos de aprendizaje son como un edificio, cada uno de los pasos que se dan deben ser en la dirección correcta. Si los datos no son correctos, o los atributos, el algoritmo de aprendizaje, los hiperparámetros del modelo o la propia función que se quiere aproximar no va en la buena dirección, se nos cae. En el caso de intentar mejorar el precio medio de compra el problema es claro.

Si nos fijamos en la tabla 5.2 la media de los retornos es tremendamente pequeña, $-0{,}0002\,\%$, con respecto a la desviación estándar, $0{,}0697\,\%$, o lo que es lo mismo en este caso, una gran cantidad de ruido para una señal determinada. Estamos hablando, en esta ocasión, de que la desviación estándar es 348,50 veces superior a la media.

Tras un intento infructuoso y volver a construir desde cero todo el proyecto, se decidió simplificarlo todo. Este cambio hizo comenzar con la idea de [1] de la liquidación constante para posteriormente pasar al Percent on Volume (POV) y al Volume Weighted Average Price (VWAP).

En las imágenes 5.32 y 5.33 se analiza el comportamiento que se propone en [1] donde se vende de manera constante siempre el mismo número de

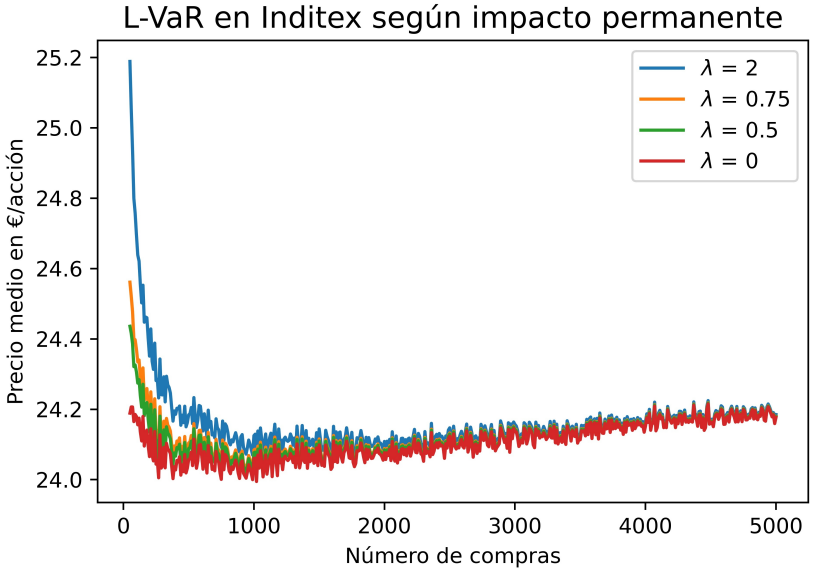


Imagen 5.32: Precio medio de adquisición

acciones. Con esto en mente, se puede ver el precio medio de adquisición (es análogo a las ventas, pero, en este trabajo se hace todo a través de compras) a través del *L-VaR*. El L-VaR, como se analiza en la ecuación 3.8 el coste de implementar una estrategia a lo que se le suma un número de desviaciones típicas del valor de la cartera restante de modo que se puede decir el precio de venta con un intervalo de confianza.

En el caso de esta primera implementación, se han adquirido 2.000.000 de acciones que han sido divididas en un número de compras (todas constantes) comenzando en 50 compras de 40.000 acciones y se llega a las 5000 compras de 400 acciones.

En la imagen 5.33 se puede comprobar cómo al retrasar la adquisición del total de la cartera la volatilidad de la misma es mayor, además, este incremento es continuo una vez nos despegamos de las aberraciones del inicio donde el impacto crea una gran varianza. Este análisis se realiza en función del coste permanente λ. Cuando $\lambda = 0$ todo el impacto en el mercado es temporal, el cual será un caso ideal, mas, poco realista, mientras que cuando $\lambda = 2$ tendríamos que el impacto permanente es equivalente a 2 veces el impacto temporal. Este segunda caso seguramente tampoco sea realista por exceso de impacto, pero, para demostrarlo solo puede hacerse de manera empírica y con dinero real.

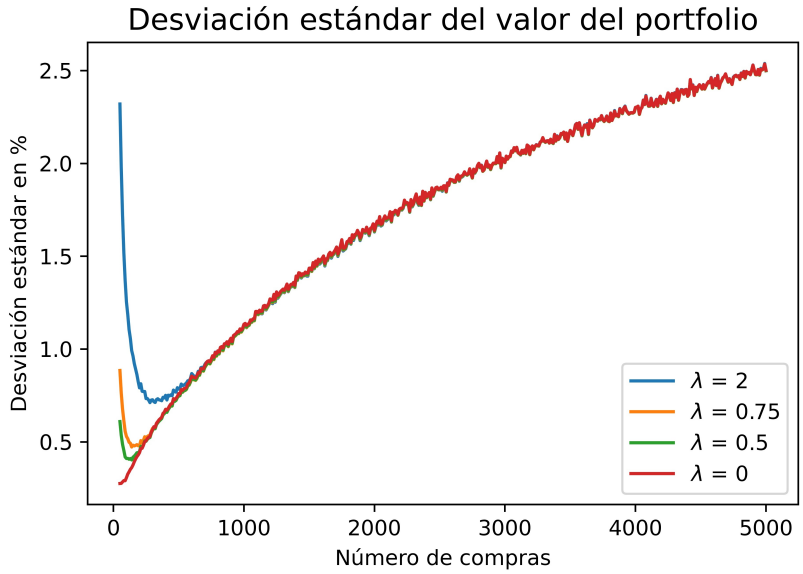


Imagen 5.33: Desviación estándar del porfolio restante

Por otra parte, en 5.32 se puede observar el precio medio de adquisición el cual se puede asimilar al comportamiento de un palo de hockey, siendo este análisis realizado también en función del impacto temporal parametrizado por λ.

De cualquier modo, lo que es evidente a la vista de la imagen 5.32, tanto de una forma intuitiva y de manera analítica, si se compra muchas acciones de manera constante, el precio medio subirá y si dilatamos mucho en el tiempo la adquisición se encontrará una gran volatilidad en los resultados. En ambas casos, el L-VaR tomará valores muy grandes, por lo que debe de haber un punto intermedio óptimo para cada uno de los valores λ.

Como todas las muestras se han hecho con la misma semilla, lo que encontramos es un desplazamiento vertical causado por el impacto, pero, que genera el mismo punto óptimo, el cual supondría realizar 1020 compras con 1960.78 acciones por compra.

Funciones objetivo

Ya conocemos que el aprendizaje por refuerzo puede aproximar el comportamiento de una función, bien sea aplicando un enfoque tabular como haciendo uso de redes neuronales, cabe pensar en *cómo* hacerlo.

El aprendizaje por refuerzo hace uso del concepto de recompensa (o coste según el ámbito del problema) para conseguir aproximar una función. Los ejemplos más comunes que se pueden encontrar son:

- **Maximización de la recompensa:** este enfoque es el usado cuando se simulan juegos como se hace en [22]
- **Minimización del coste:** este enfoque es el usado para el problema del vendedor de viaje (Traveling Salesman Problem en inglés) [11] donde se intenta minimizar el coste de transporte en las visitas de un vendedor de viaje. También fue la aproximación usada al principio intentando minimizar el coste medio de adquisición
- **Aproximación a una función:** este es el enfoque que se usará en adelante donde se va a trabajar en *parecerse* al POV y al VWAP.

Como clasificación inicial puede valer, pero, sigue sin estar definido cómo se puede aproximar a esa función. La opción más intuitiva es usar la diferencia de los resultados obtenidos por el algoritmo de aprendizaje por refuerzo y los de la función aproximada. Matemáticamente estaríamos hablando de las ecuaciones 5.24, 5.25 y 5.26 donde $A(i)$ es la adquisición en un momento i, I_Q es la cantidad inicial que queremos comprar, F es la función que queremos aproximar, P es la proporción sobre la función en la que se va a intervenir y D es la distancia entre el resultado obtenido y la función aproximada.

$$A_r(t) = \frac{\sum_{i=0}^{t} A(i)}{I_Q} \tag{5.24}$$

$$F_r(t) = \frac{\sum_{i=0}^{t} F(i)P}{I_Q} \tag{5.25}$$

$$D(t) = |A_r(t) - F_r(t)| \tag{5.26}$$

Es importante apreciar tanto la presencia del porcentaje en la intervención de la función donde 0 equivale a no realizar ninguna adquisición y 1 supone adquirir las mismas acciones o contratos que se han negociado en el mercado, como del sumatorio en 5.24 y en 5.25. Incluir este sumatorio implica aproximar la evolución de las compras desde el inicio hasta tener la cartera con la posición deseada. De no existir este sumatorio lo que se haría es intentar adivinar cual será el siguiente volumen negociado cayendo de nuevo en un error.

POV

Tras el infructuoso intento con la mejora del precio de adquisición en vez de la aproximación de una función como es el POV o el VWAP, es importante replantear las características que generarán los diferentes estados.

Recordando el caso ya estudiado se contó con la *volatilidad*, *segundos hasta el final del día* y el *desequilibrio de posiciones en el bid y en el ask*. Recapitulando y para explicar las características necesarias, el POV consiste en adquirir (o liquidar) un porcentaje constante del volumen negociado en un intervalo temporal determinado, siendo lo más común no ser superado el 20 o 30 % del total. Este porcentaje viene determinado por la prisa que tenga el agente en adquirir o liquidar la posición, que en todos los casos será de 100.000 acciones.

Por esta razón, tendremos un parámetro que caracteriza el algoritmo, el porcentaje, mientras que para representar el estado concreto en el que nos encontremos habrá que contar con la diferencia entre cuánto hemos adquirido y cuánto dice el algoritmo que tenemos que adquirir, como bien está explicado a través de las ecuaciones 5.24, 5.25 y 5.26.

Como el ratio de adquisición del POV y de la cartera están acotadas entre 0 (al inicio del proceso) y 1 (cuando está todo adquirido), 5.26 tomará valores en el intervalo [-1,1] que se discretizarán en 20 posibles posiciones cuyos tamaños de rango vendrán determinados por una distribución normal. De este forma todos los intervalos tienen la misma probabilidad de ser visitados, al revés de lo que ocurriría si el tamaño de los intervalos fueran siempre los mismos. Esta transformación es la que se ve en la imagen

El otro estado con el que se ha contado es la situación de la adquisición estando acotado en el intervalo [0,1], siendo 0 cuando se inicia el proceso de adquisición y 1 cuando se haya terminado el proceso. En esta ocasión, a pesar de haber discretizado también este intervalo en 20 posibles posiciones, estas posiciones son del mismo tamaño ya que el proceso de adquisición debe de pasar del 0 % al 100 % de manera lineal y por ello, mantenemos la equiprobabilidad de caer en cada una.

A partir de aquí, se han comenzado a realizar las pruebas tanto para comprobar el funcionamiento del aprendizaje por refuerzo de cara a aproximar la función del POV y del VWAP, así como, si el uso del aprendizaje profundo mejora los resultados.

Es evidente que con los estados elegidos tanto la versión tabular como la que hace uso del aprendizaje profundo son capaces apender el comportamiento del POV. Esto se desprende de la evolución de la comparación entre

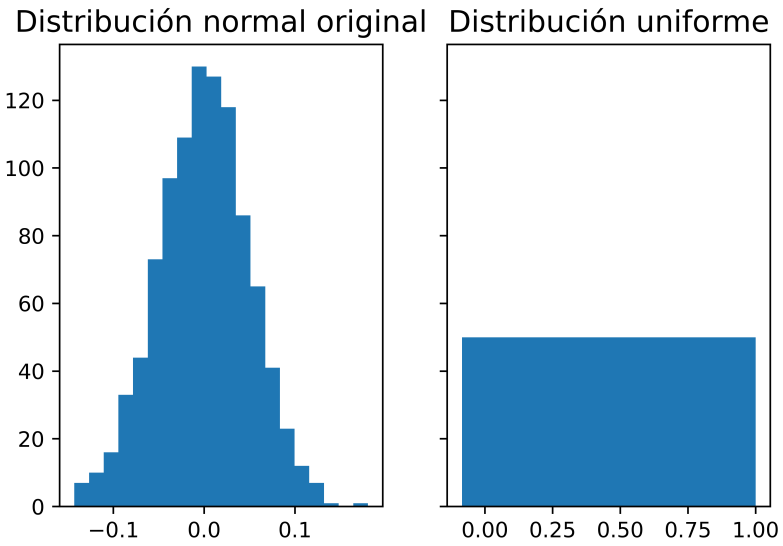


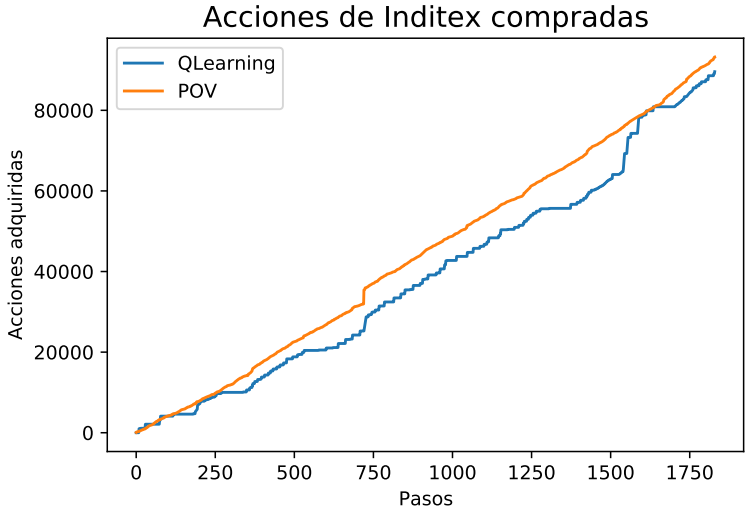
Imagen 5.34: Transformación en la diferencia entre el algoritmo de ejecución y el modelo entrenado

las acciones que dicta el POV y las adquiridas por el QLearning (versión tabular) y el Double Deep QLearning (versión con aprendizaje profundo) como se puede ver en la imagen 5.35.

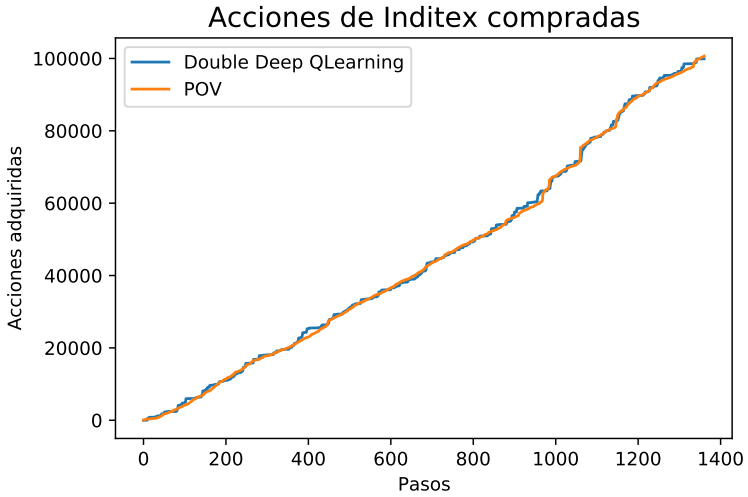
Por su parte, en la imagen 5.36 se analiza la diferencia que hay en valor absoluto entre el porcentaje que debe ser adquirido según el POV y el que ha sido adquirido por el algoritmo de aprendizaje usado. La ecuación exacta es la que se puede ver en 5.26.

Si se estudian la media y las desviaciones del ejemplo anterior siguiendo la ecuación 5.26 encontramos que en QLearning tenemos una media de 5,26 % y una desviación estándar de 3,12 %, mientras que en Double Deep QLearning la media es de 0,56 % y la desviación es de 0,52 %.

El último estudio, aunque no por ello menos importante, que se va a realizar sobre el POV es el de comparar la evolución del precio medio de un episodio según se van adquiriendo las acciones, en este caso de Inditex. Es por esta razón por la que al principio del gráfico se encuentra una mayor variabilidad en la diferencia de precios ya que hay menos datos sobre los que hacer el análisis. Por otra parte, hay que entender el funcionamiento, a nivel operativo, del algoritmo de aprendizaje. Según el algoritmo de aprendizaje se va distanciando, éste *acelerará* el ritmo comprando de manera

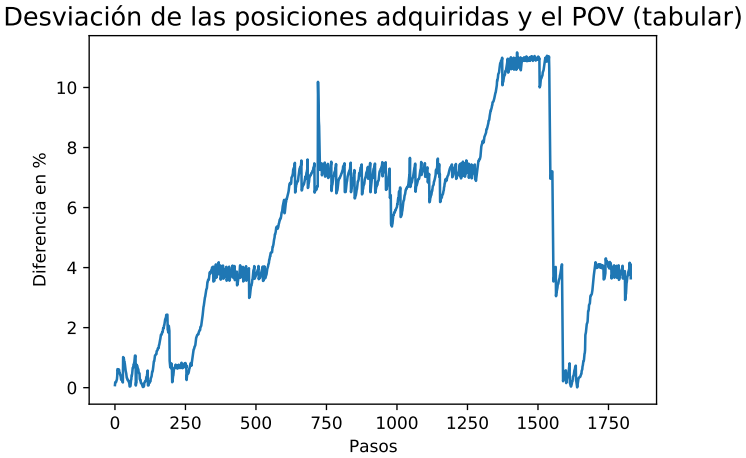


(a) Tabular

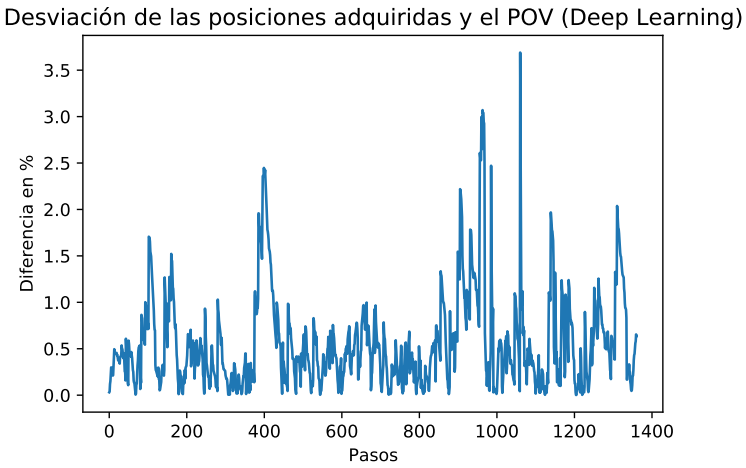


(b) Deep Learning

Imagen 5.35: Acciones acumuladas según el POV y el algoritmo de aprendizaje



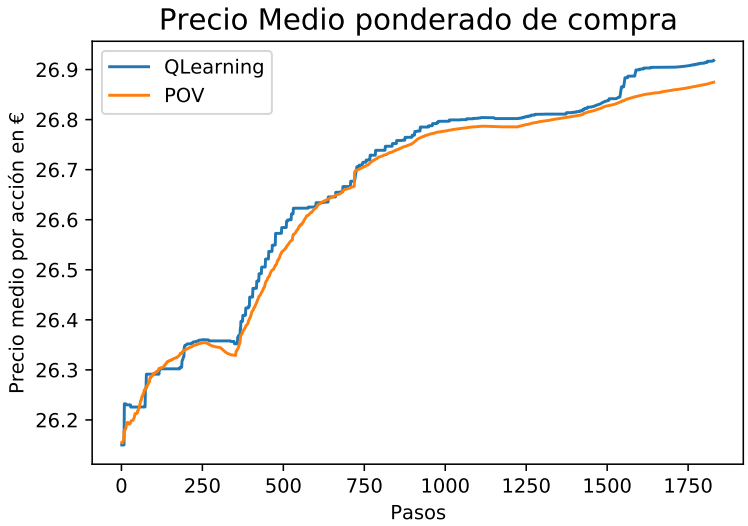
(a) Tabular



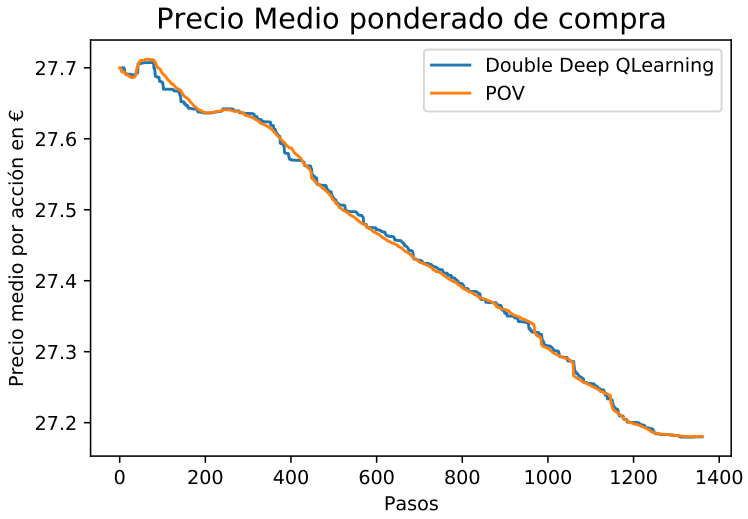
(b) Deep Learning

Imagen 5.36: Diferencia entre el porcentaje de acciones adquiridas según el POV y el algoritmo de aprendizaje

más agresiva. Si por el contrario, el algoritmo se pasa en las compras, *frenará* para adecuarse a la función que se quiere aproximar.



(a) Tabular



(b) Deep Learning

Imagen 5.37: Precio medio de adquisición a lo largo de los pasos en el POV

En el caso del POV, resulta evidente la mejora del uso del aprendizaje profundo en lo que a error se refiere.

VWAP

Al igual que ha ocurrido con el POV, se va hacer este análisis usando un método tabular de QLearning y un método de aprendizaje profundo. A pesar de tener esta misma estructura, encontramos diferencias sustanciales. La principal es la diferencia del número de estados ya que hay que agregar una nueva dimensión: el volumen esperado a una hora determinada.

Recapitulando, el VWAP, nos dice cuál es el volumen esperado en un momento determinado del tiempo. Algo tan sencillo puede hacerse con una simple media, pero, hacerlo así puede tener ciertos problemas. Si los datos son asíncronos, no llegando por cada intervalo temporal, sino que depende de los intercambios de los diferentes agentes, ¿sobre qué se hace la media? En caso de que un intervalo temporal no haya tenido ninguna transacción, ¿significa que el volumen esperado es cero? De ninguna manera, lo que significaría es que no ha habido ninguna muestra. Por estas cuestiones, intentar hacer medias para calcular el VWAP no parece que sea la mejor de las decisiones.

Por estas razones, la siguiente idea que surge es si puede hacerse a través de una regresión. De esta manera evitamos tanto los problemas anteriores, como la presencia de datos atípicos, o incluso la uniformidad del valor del VWAP a lo largo de la sesión bursátil. Matemáticamente este último punto podría entenderse como la ausencia de derivadas en la función del VWAP.

Para realizar esta regresión es necesario agrupar por minutos los datos de las velas de ticks. Así conseguimos tener un único dato por cada minuto de cada día, en el caso de que haya habido transacciones. En este momento estamos dispuestos a hacer un gráfico de dispersión y hacer la ya citada regresión. En la imagen 5.38 se puede ver el resultado del cálculo del VWAP, siendo esa línea morada la función que se va a aproximar. Estamos hablando de una regresión lineal de segundo orden, como se explica en [31], ya que las de primer grado tienen mucho error y regresiones de órdenes superiores no mejoran el modelo. Para ilustrarlo, el MAE (Mean Absolute Error) es de 49,58, lo cual es bastante teniendo en cuenta la gran variabilidad que se puede ver en el volumen registrado en cada unidad de tiempo. Teniendo ya definida la función que se va a aproximar ya estamos preparados para hacer las pruebas pertinentes y sus resultados.

Al igual que ocurría en el caso del POV, el VWAP también puede ser aproximado por el aprendizaje por refuerzo, aunque con resultados un poco distintos. Si nos fijamos en la evolución de las acciones adquiridas por el VWAP y por el algoritmo de aprendizaje (5.39) vemos que a simple vista (al

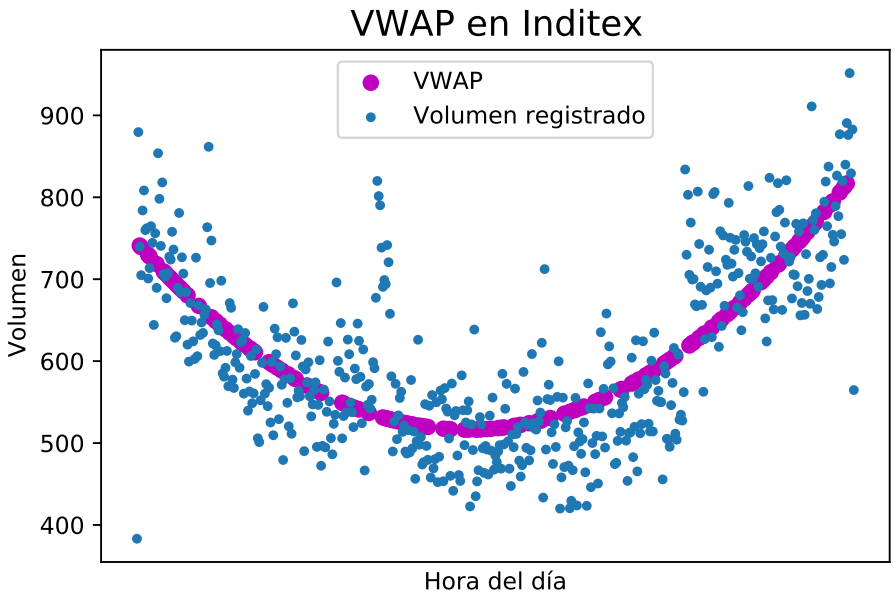
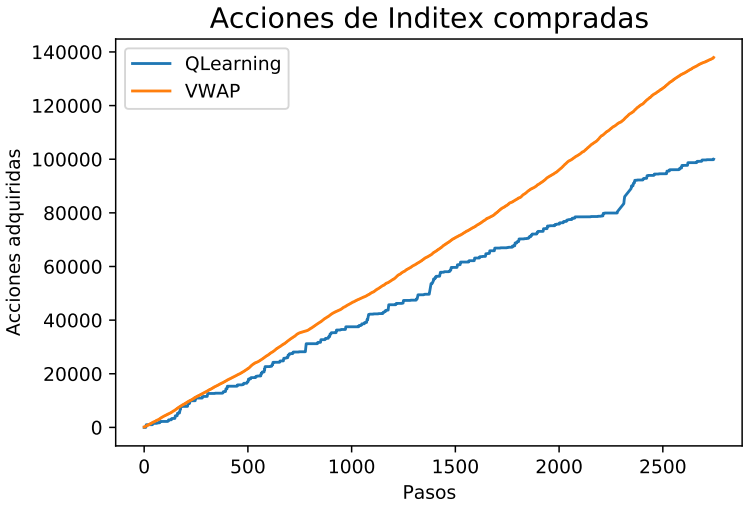


Imagen 5.38: Regresión de segundo grado del VWAP en Inditex

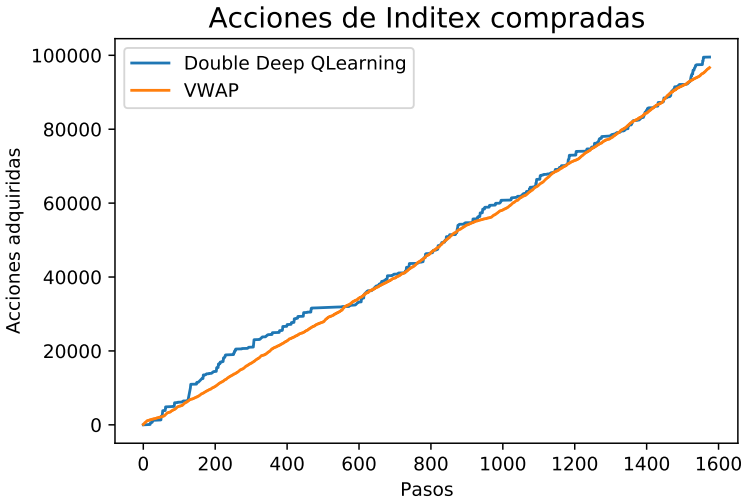
menos en el caso tabular), tiene una mayor separación, indicando un mayor error en el aprendizaje.

Esta cuestión del error por la distancia entre el VWAP y la aproximación se puede ver de manera indudable en la imagen 5.40 observamos el valor absoluto de la diferencia entre las posiciones adquiridas y las que teóricamente deberíamos haber adquirido. Una vez más, la aproximación con el aprendizaje profundo y el Double Deep QLearning nos arroja mejores resultados. Muestra de estos mejores resultados es comprobar como, en este ejemplo, las media son 14,32 % y 1,90 %, mientras que las desviaciones estándar son 10,80 % y 1,80 % para el modelo tabular y el que usa aprendizaje profundo respectivamente.

Por último, analizamos la evolución del precio medio de adquisición que tiene, una vez más, una mayor variabilidad al principio del proceso ya que dispone de menos datos sobre los que hacer la media. Hay que recalcar el impacto que puede tener tomar decisiones de compra muy grandes que provocan un mayor impacto en el mercado. Eventualmente, aunque no sea lo más común, puede haber ocasiones en las que el precio medio de adquisición sea menor siguiendo el algoritmo de aprendizaje que mediante el VWAP, dependiendo tanto de las acciones concretas tomadas a lo largo del episodio y de la evolución de las cotizaciones.



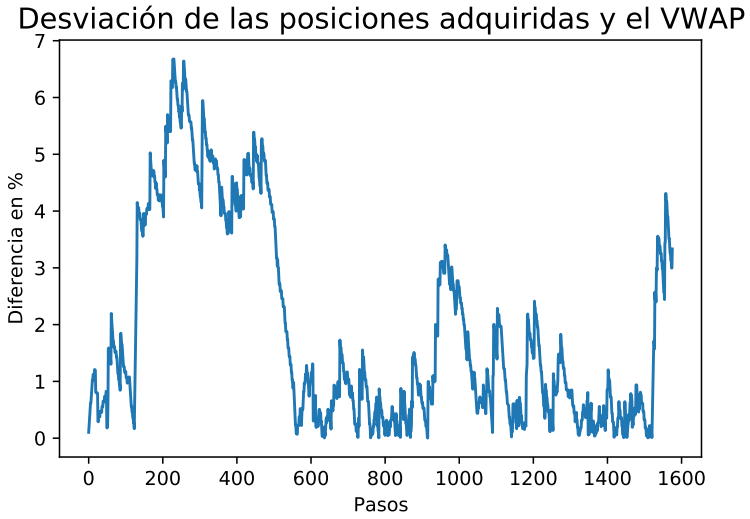
(a) Tabular



(b) Deep Learning

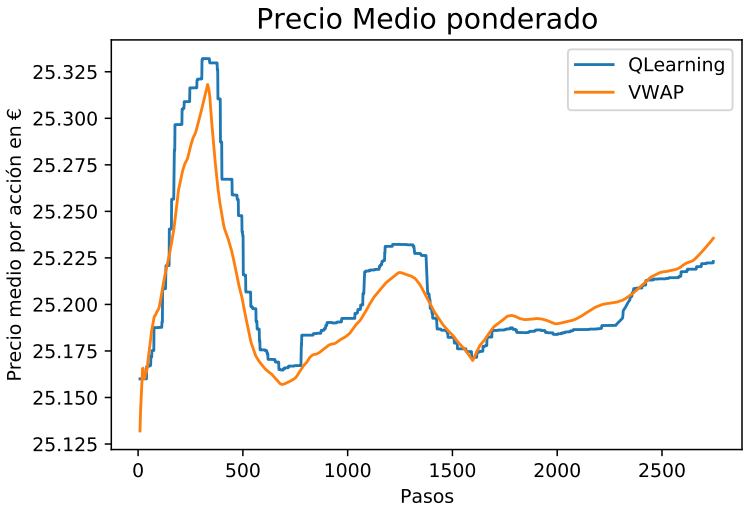
Imagen 5.39: Acciones acumuladas según el VWAP y el algoritmo de aprendizaje

(a) Tabular

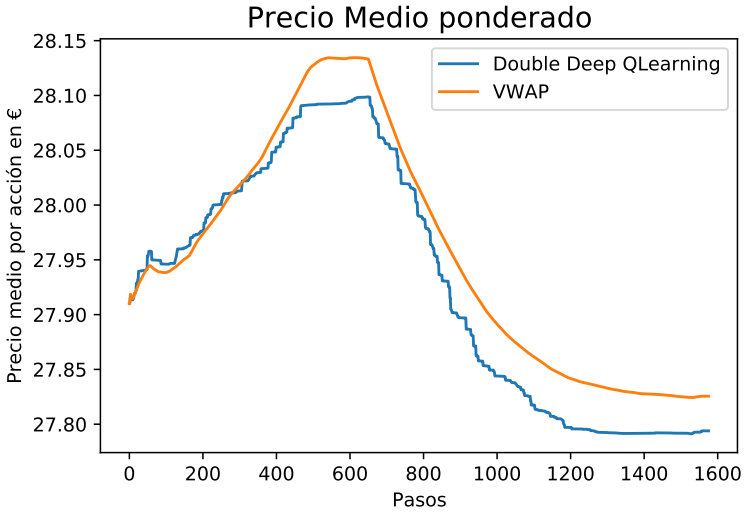


(b) Deep Learning

Imagen 5.40: Diferencia entre el porcentaje de acciones adquiridas según el VWAP y el algoritmo de aprendizaje



(a) Tabular



(b) Deep Learning

Imagen 5.41: Precio medio de adquisición a lo largo de los pasos en el VWAP

5.3. Resultados

Los resultados anteriores además de ser ilustrativos, muestran la evolución del algoritmo, de la diferencia con el POV y el VWAP y el precio medio de adquisición, pero, no incluyen impacto permanente. En la siguiente tabla, se podrán ver los resultados incluyendo el impacto permanente y diferentes λ en diferentes activos.

Además, los resultados del POV y el VWAP estudiados muestran los resultados aplicándose sólo a las acciones de Inditex, mientras que puede ser interesante probar en otros activos. Para esta tarea nos hemos hecho uso de las principales empresas del selectivo español y alguna de las empresas europeas más grandes o conocidas.

En los apartado anteriores se entrenaron los 4 modelos distintos (POV y VWAP en formato tabular y con aprendizaje profundo) para las acciones de Inditex con un resultado bastante satisfactorio. Partiendo con esa base se han usado las tablas y los pesos en las redes neuronales para con un entrenamiento más pequeño (con menos episodios) adaptarnos a las diferentes acciones de otras empresas. Los resultados se pueden ver en las tablas 5.7, 5.8, 5.9 y 5.10.

Características e hiperparámetros usados

Uno de los puntos clave de toda investigación son los características usadas en las pruebas y los hiperparámetros, bien sea para entender la investigación o bien sea para poder replicar.

La primera es la que contiene los hiperparámetros que han gobernado todas las pruebas, como se ve en la tabla **??**. Vamos a comentar cada uno de los puntos para entender su origen y su valor. El número de acciones se eligió arbitrariamente para que fuera lo suficientemente grande como para dejar al algoritmo tanto aprender como tener un gran número de pasos. En el caso de los atributos nos encontramos 2 para el POV que suponen diferencia de adquisición entre el algoritmo y el modelo por un lado y por otro cuanto queda por adquirir por el modelo. Para el VWAP se le añade una tercera característica que es el horario del día que determina las acciones que se le espera comprar. Para los atributos se han usado 20 bins, al igual que se usa un 20 % para el porcentaje del volumen negociado (en el POV) o esperado (en el VWAP) que se desea adquirir. En el caso de ϵ hay que mencionar que se ha usado un 0,15 en los entrenamientos de los modelos mientras que para presentar los resultados se ha usado un 0,10 como se explica un poco más adelante.

Hiperparámetro	Valor
Acciones a comprar	100.000
# de atributos del estado (POV)	2
# de atributos del estado (VWAP)	3
Número de bins de los atributos	20
Porcentaje del POV y el VWAP	20 %
Ratio de aprendizaje	0,1
Descuento	0,95
ϵ inicial	1
ϵ mínimo (entrenamiento)	0,15
ϵ mínimo (explotación)	0,10
Episodios (tabular)	10.000
Episodios (Deep Learning)	1500

Tabla 5.4: Hiperparámetros

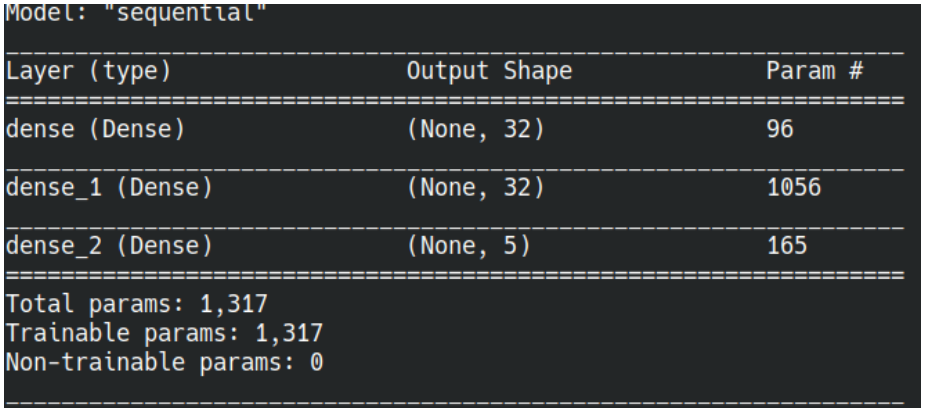


Imagen 5.42: Definición del modelo de la red neuronal

Otro de los puntos más importantes es la definición de las redes neuronales usadas y sus características. La red neuronal que se va a usar se compone de una entrada 2 y 3 atributos para el POV y el VWAP respectivamente que se conectan densamente con dos capas densas de 32 neuronas (como se puede ver en la imagen 5.42) con una salida con 5 posibilidades que se corresponden con cada una de las posibles acciones disponibles. Estas acciones, que serán las mismas para los modelos tabulares y de Deep Learning, nos permiten adquirir desde 0 acciones hasta un total de 1000 en cada uno de los pasos. Evidentemente si la acción tomada supera al número de acciones restantes por adquirir, se adquieren las acciones restantes.

Acción	Número de acciones a comprar
0	0
1	5
2	100
3	500
4	1000

Tabla 5.5: Acciones disponibles

Continuando con las características de las redes neuronales, encontramos un mínimo de experiencias de 200, que se corresponden con los primeros 200 pasos, a partir de los cuales comenzarán a ser entrenados los diferentes pesos de las neuronas. Esto se hará hasta haber recorrido las 20.000 experiencias, que será establecido como límite, que una vez alcanzado se irá borrando la experiencia más antigua. Cómo se van borrando estas experiencias, esta es la razón más fehaciente de la adaptación de las redes neuronales a entornos cambiantes. Si por ejemplo, se necesitan 1000 pasos para realizar la compra completa de las 100.000 acciones, significaría que en 20 episodios tenemos una memoria sobre la que entrenar la red totalmente nueva. A cada paso que se da en cada uno de los episodios se produce un entrenamiento, donde de las 20.000 experiencias (o las que hubiera en el momento del entrenamiento), se toma un total de 16 experiencias aleatorias que son usadas para actualizar los pesos de la red. Se ha probado manualmente con otros valores del batch, pero, 16 ha dado buenos resultados con menos recursos consumidos.

Hay que recordar que en el modelo con Deep Learning se usan 2 redes ya que son necesarias para el Double Q-Learning. Por último el valor del ratio de aprendizaje se ha usado ese por convención.

Procedimiento de los resultados

Con los parámetros y configuraciones explicadas en el punto anterior, el procedimiento que se ha seguido ha sido, primero hacer un pequeño entrenamiento para poder terminar de adaptar el algoritmo de aprendizaje por refuerzo y después realizar 100 ejecuciones con $\epsilon = 0{,}10$ de modo que el 90 % de las decisiones las toma el algoritmo. De esta forma se consigue tener lo modelos de aprendizaje bien *orientados* ya que son conscientes de que si el modelo ha adquirido menos posiciones de las que marca el POV o el VWAP se debe acelerar el ritmo y viceversa, mientras que diferentes

Característica	Valor
Tipo de las capas	densas
Copia de los pesos)	cada 25 pasos
Máximo de experiencias	10.000
Mínimo de experiencias	200 %
Tamaño del batch	16
Ratio de aprendizaje de la red	0,01
Número de redes	2

Tabla 5.6: Parámetros de las redes neuronales

activos pueden tener una *sensibilidad* distinta sobre cuánto acelerar o frenar en un momento determinado.

Es importante diferenciar este binomio entrenamiento-cálculo en lo que al valor de ϵ se refiere. En el primero se produce un descenso desde 1, en el primer episodio, hasta 0,15 que se tiene que alcanzar y mantener desde la mitad de los episodios en adelante. De esta forma se consigue una labor de *exploración* ya que lo interesante no es el valor concreto en producción, sino, aprender las diferentes posibilidades aunque no sea necesario hacerlo tan a fondo como en el aprendizaje inicial del modelo. En el segundo caso, nuestra ambición es tener una labor de *explotación* usando como $\epsilon = 0{,}10$ la razón es muy sencilla, si ϵ es muy grande el resultado final dependerá mucho del componente aleatorio pudiendo tanto aumentar sustancialmente los costes como la desviación con la función a aproximar. Si por el contrario tuviéramos un ϵ demasiado bajo lo que puede ocurrir es que el modelo puede quedar en un mínimo local. Imaginemos que en un estado donde la cantidad restante de adquirir sea pequeña y la acción que implica un menor coste fuera no adquirir ninguna posición, lo que se puede provocar es que el POV o el VWAP fuera distanciándose sin que el modelo pudiera comprar más acciones y no llegando al estado final. Empíricamente se ha llegado a ese 15 % para la exploración y al 10 % para la explotación.

Dentro de la explotación no sólo se realizan los cálculos de las diferentes compras, sino, que además se ha incluido diferentes valor para el coste permanente, que son los siguientes: 0 %, 50 %, 75 % y 200 %. Estos valores se traducen en que el 0 %, el 50 %, etc. del valor del impacto que sufrimos se acumula a modo de impacto permanente. El valor correcto no se puede aproximar más allá de la mera intuición, teniendo que conocer estos valores de manera empírica y con dinero real. Según la tabla 5.7, el valor medio de

adquisición de Inditex en el POV es de 25,63 € y contando que todas la pruebas se han hecho con 100.000 acciones, la prueba necesitar compras y ventas de 2.563.000 €. Continuando con las tablas, con estos 100 episodios lo que se ha hecho es conseguir los valores medios de las siguientes métricas:

1. Diferencia de los porcentajes entre las compras algoritmos y de los modelos de aprendizaje
2. Desviaciones estándar del punto anterior
3. Precio medio de adquisición de las acciones en el episodio conseguido por el POV o el VWAP
4. Precio medio de adquisición del modelo sin impacto permanente
5. Precio medio de adquisición del modelo con un 50 % de impacto permanente
6. Precio medio de adquisición del modelo con un 75 % de impacto permanente
7. Precio medio de adquisición del modelo con un 100 % de impacto permanente
8. Sobrecoste en porcentaje del precio medio de adquisición con un 50 % de coste permanente en relación al del POV o el VWAP

Activo	Media	Desviación	Precio POV	0 %	50 %	75 %	200 %	50 % sobre POV
Inditex	11.83 %	6.28 %	25.63	25.64	25.76	25.81	26.10	0.47 %
BBVA	35.80 %	20.98 %	4.24	4.24	4.29	4.31	4.44	**1.21 %**
Santander	**371.15 %**	**236.18 %**	3.50	3.50	3.54	3.55	3.64	1.09 %
Telefónica	38.48 %	20.33 %	5.84	5.84	5.89	5.92	6.07	0.98 %
SAP	11.79 %	8.34 %	111.20	111.21	111.49	111.63	112.33	**0.26 %**
Allianz	**8.33 %**	**5.12 %**	199.01	199.15	199.85	200.20	201.94	0.42 %
Airbus	11.79 %	7.43 %	122.67	122.80	124.00	124.61	127.62	1.09 %

Tabla 5.7: Análisis de los resultado del POV de forma tabular. El 0 %, 50 %, 75 % y 200 % representan el precio medio de ejecución con ese porcentaje de impacto permanente

Activo	Media	Desviación	Precio VWAP	0 %	50 %	75 %	200 %	50 % sobre VWAP
Inditex	6.95 %	3.68 %	26.16	26.17	26.28	26.34	26.62	0.47 %
BBVA	20.31 %	9.04 %	4.30	4.30	4.35	4.37	4.49	**1.11 %**
Iberdrola	16.45 %	7.92 %	9.24	9.24	9.28	9.30	9.39	0.39 %
Santander	**70.98 %**	**51.68 %**	3.27	3.27	3.30	3.32	3.41	0.92 %
Telefónica	14.12 %	6.54 %	6.01	6.01	6.07	6.09	6.23	0.90 %
SAP	**5.08 %**	**3.13 %**	110.76	110.75	111.05	111.20	111.95	**0.27 %**
Allianz	14.54 %	8.68 %	199.97	199.96	200.59	200.90	202.45	0.31 %
Airbus	29.16 %	16.96 %	116.36	116.37	117.53	118.11	121.02	1.00 %

Tabla 5.8: Análisis de los resultado del VWAP de forma tabular. El 0 %, 50 %, 75 % y 200 % representan el precio medio de ejecución con ese porcentaje de impacto permanente

Activo	Media	Desviación	Precio POV	0 %	50 %	75 %	200 %	50 % sobre VWAP
Inditex	**5.64 %**	**3.82 %**	26.62	26.62	26.73	26.79	27.06	0.40 %
BBVA	41.51 %	21.75 %	4.26	4.26	4.32	4.34	4.49	**1.39 %**
Iberdrola	23.89 %	14.60 %	9.36	9.36	9.40	9.43	9.53	0.45 %
Santander	**50.72 %**	**28.55 %**	3.35	3.35	3.39	3.41	3.51	1.20 %
Telefónica	19.65 %	12.00 %	5.78	5.78	5.84	5.88	6.05	1.16 %
SAP	18.60 %	12.10 %	112.15	112.14	112.41	112.54	113.22	**0.23 %**
Allianz	27.11 %	16.57 %	198.68	198.63	199.29	199.62	201.26	0.30 %
Airbus	34.45 %	21.32 %	120.13	120.11	120.97	121.40	123.55	0.70 %

Tabla 5.9: Análisis de los resultado del POV con aprendizaje profundo. El 0 %, 50 %, 75 % y 200 % representan el precio medio de ejecución con ese porcentaje de impacto permanente

Activo	Media	Desviación	Precio VWAP	0 %	50 %	75 %	200 %	50 % sobre VWAP
Inditex	10.42 %	8.13 %	26.71	26.71	26.82	26.88	27.16	0.41 %
BBVA	12.42 %	5.24 %	4.39	4.39	4.44	4.47	4.61	**1.25 %**
Iberdrola	4.42 %	2.34 %	9.06	9.06	9.10	9.12	9.21	0.41 %
Santander	21.16 %	10.29 %	3.36	3.36	3.39	3.41	3.50	1.08 %
Telefónica	**1.99 %**	**1.14 %**	5.85	5.85	5.91	5.95	6.11	1.13 %
SAP	19.02 %	11.32 %	114.31	114.31	114.58	114.72	115.40	**0.24 %**
Allianz	30.36 %	18.35 %	196.91	196.89	197.57	197.91	199.59	0.34 %
Airbus	**36.35 %**	**21.75 %**	119.78	119.78	120.65	121.09	123.28	0.73 %

Tabla 5.10: Análisis de los resultado del VWAP con aprendizaje profundo. El 0 %, 50 %, 75 % y 200 % representan el precio medio de ejecución con ese porcentaje de impacto permanente

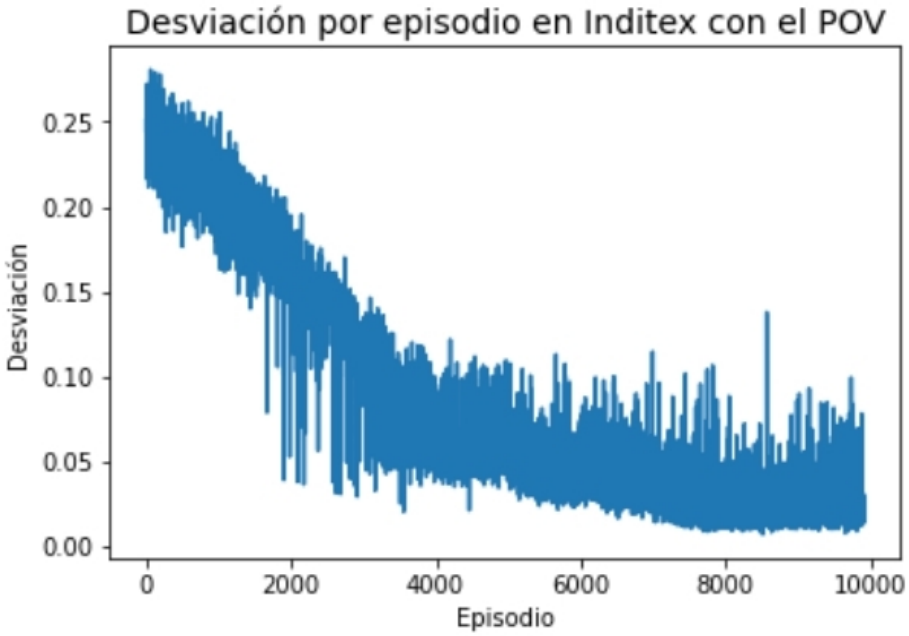


Imagen 5.43: Evolución del aprendizaje en Inditex con el POV tabular

El aprendizaje es claro, aunque resulta conveniente revisar la capacidad de aprender que han tenido los diferentes organismos. En las imágenes 5.43 y 5.44 tenemos el aprendizaje en Inditex del POV y del VWAP. Esto no solo sucede en el modelo tabular, como es el caso anterior, sino que en el formato con Double Deep Q-Learning ocurre lo mismo como se puede observar con el ejemplo del POV en Inditex de la imagen 5.45 y con el VWAP en SAP podemos ver la imagen 5.46.

A pesar de que esta forma de implementación es coherente y de hecho funciona, cabe destacar la diferencia entre algunas de las acciones que ahora se revisarán. En las tablas 5.7, 5.8, 5.9 y 5.10, a pesar de ser diferentes siguen ciertos patrones. Las compras en el Banco Santander son las que peor funcionan, mientras que, no sólo Inditex (donde han sido entrenados los modelos) sino que Iberdrola, Telefónica o SAP obtienen valores en las medias de las diferencias bastante competentes.

En el caso del sobrecoste lo que se aprecia es que SAP y Allianz son en general las compañías mejor paradas, mientras que el BBVA, Santander o Telefónica son de las más afectadas.

Ambas métricas tienen mucha razón de ser. Primero, en relación a aquellas acciones que les cuesta más aproximar el algoritmo viene por el

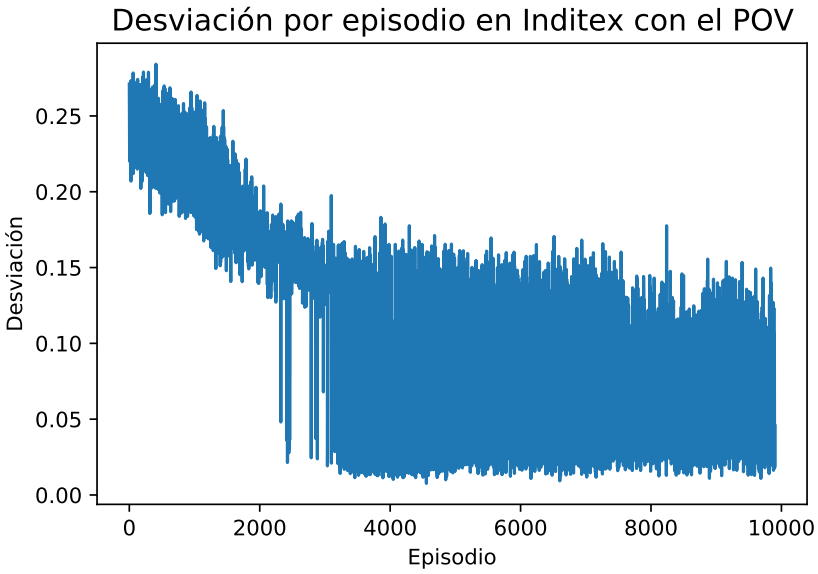
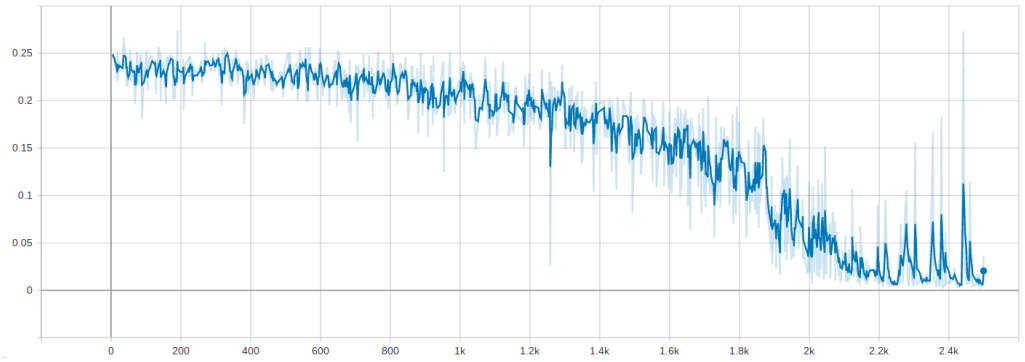


Imagen 5.44: Evolución del aprendizaje en Inditex con el POV tabular

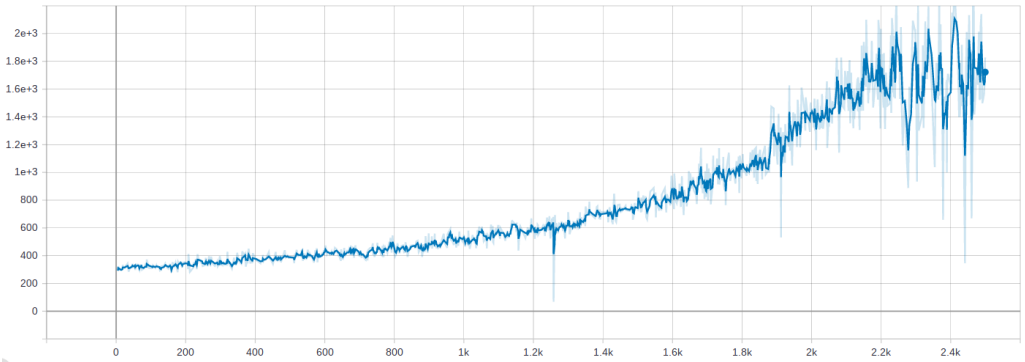
volumen cruzado en cada unidad de tiempo. Mientras que las acciones que se pueden tomar tienen una cantidad fija de acciones[17] que se van a comprar, cada una de las compañías tienen un volumen negociado diferente. Para ilustrarlo, la media de acciones negociadas en cada vela de ticks en el Banco Santander es de 1871 y en Inditex (donde se ha desarrollado todo) 280. Esto provoca que las empresas, de modo general, que tienen un volumen negociado parecido por vela de ticks a Inditex registrarán un mejor rendimiento. De una manera más concreta, serán aquellas empresas que se adapten mejor a esta estructura de acciones las que se revelarán vencedoras en esta métrica.

El segundo de los problemas, que en cierta medida puede ser más importante, es la métrica del sobrecoste. Este sobrecoste puede venir tanto de la desviación del algoritmo, siendo máximo este problema cuando el algoritmo avanza más rápido que la función aproximada creando un mayor impacto en el mercado. La segunda razón por la que se crea este mayor impacto o sobrecoste es por la liquidez de la compañía. La liquidez de un activo se define como la capacidad de adquirir o liquidar una posición en un

[17]Es importante diferenciar, a pesar de compartir denominación, las acciones que puede tomar el algoritmo de aprendizaje por refuerzo y las acciones en el sentido económico que se refieren a cada una de las partes alícuotas en las que se divide el capital social de una empresa. En inglés es mucho más sencillo ya que en el primer caso se alude al concepto de *action* y en el segundo al de *share*.



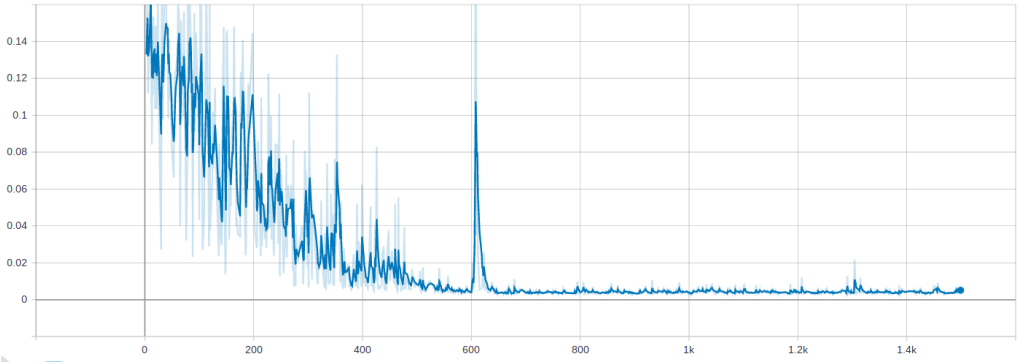
(a) Error medio en cada episodio



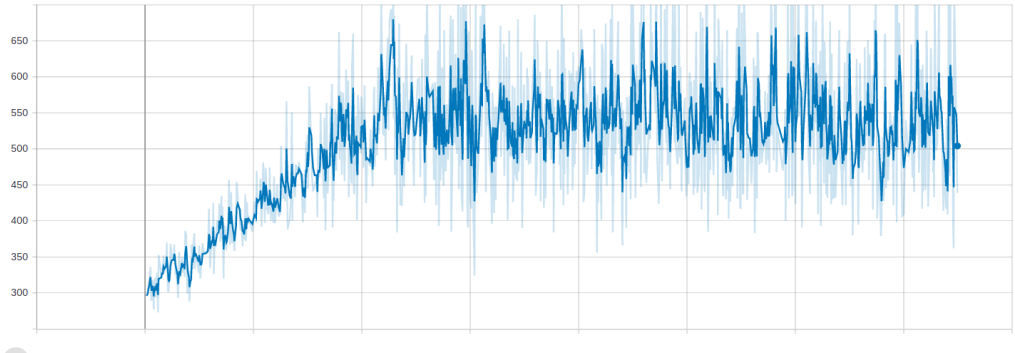
(b) Número de pasos en cada episodio

Imagen 5.45: Aprendizaje en Inditex con Double Deep Q-Learning y el POV

momento determinado teniendo en cuenta su coste asociado. Si retomamos el inmobiliario que se vio al comienzo, es mucho más sencillo y tiene un menor coste (en porcentaje) la adquisición o venta de un inmueble en una gran ciudad que en un pueblo pequeño. Aquí ocurre igual, si la empresa es más pequeña, tiene un menor volumen, o más concretamente, si tiene pocas posiciones en el libro de órdenes se arrastrarán más niveles y por ello habrá un mayor coste sin que por ello se haya hecho algo mal en el algoritmo de aprendizaje. Esta es la razón por la que Allianz o SAP tiene un menor sobreprecio que el BBVA.



(a) Error medio en cada episodio



(b) Número de pasos en cada episodio

Imagen 5.46: Aprendizaje en Inditex con Double Deep Q-Learning y el VWAP

Parte III

Conclusiones finales

Conclusiones y Líneas de trabajo futuras

Tras la experiencia ganada a lo largo del desarrollo tanto técnico, teórico y a la propia redacción del presente trabajo queda claro que ha habido un proceso de conocimiento que nos ha llevado desde el desconocimiento práctico en la materia al manejo de cierta teoría financiera y el uso de las técnicas más avanzadas dentro de la ciencia de datos para abordar la ejecución de órdenes.

7.1. Conclusiones

Continuamente nos bombardean desde los medios de comunicación y redes sociales con conceptos, en gran medida con un afán de bombo y márquetin, como los de ciencia de datos, inteligencia artificial, aprendizaje automático o big data. A tenor de estos mensajes parece que estos conceptos son la quintaesencia de la tecnología y que por ellos mismos serán capaces de solventar cualquiera de nuestros problemas golpe de click.

Nada más lejos de la realidad. El conocimiento del ámbito cobra una especial importancia en la ciencia informática y de no ser así lo más probable es que ese afán de fuerza bruta por fuerza bruta nos lleve a lugares y conclusiones no deseadas.

Muestra de este problema ante el que es fácil caer fue cuando se intentó optimizar el precio medio de ejecución directamente, lo que nos trajo infructuosos resultados. La construcción de los diferentes modelos, como si de edificios se trataran, tuvieron que ser construidos con una fuerte base

primero teórica y después técnica a modo de cemento que imbricara las diferentes partes de las que se compone.

Ventajas y desventajas de la aproximación con aprendizaje reforzado

Tras todo el desarrollo realizado queda claro que el aprendizaje reforzado es la herramienta idónea para abordar el problema de la ejecución de órdenes, incluso tratándose de una cuestión tan compleja que incluye la heterocedasticidad o la distribución no normal de los datos.

Incluso el modelo más sencillo tabular de QLearning cuando ha ido en la dirección correcta (aproximando las funciones de POV y VWAP) ha sido capaz de alcanzar el éxito. Una vez terminado esta visión tabular y en cuanto hemos tocados las redes neuronales en el aprendizaje profundo los resultados han llegado bastante fluidos y rápidos.

A pesar de estas buenas noticias, el aprendizaje por refuerzo no es el bálsamo de Fierabrás que todo lo cura, por lo que se van a exponer una serie de desventajas o problemas que plantea para después terminar este punto con las ventajas encontradas y disfrutadas.

La principal desventajas que se han hallado pueden categorizarse del siguiente modo:

- Recursos: el aprendizaje por refuerzo puede hacer uso de grandes recursos tanto en memoria (solo hay que recordar el tamaño de las tablas Q) o de las redes neuronales. También se tratan de procesos que consumen muchos recursos a nivel de potencia de cálculo, de modo que mal diseñados pueden exponenciar los tiempos

- Hiperparámetros: la optimización de los hiperparámetros puede no ser sencillas y una mala configuración no llegar a ningún punto llegando a posibles conclusiones erróneas o descartando áreas de investigación completas

- Evaluación y depuración: a pesar de que los procesos de aprendizaje automático están guiados la depuración de los problemas y la evaluación no es siempre sencilla. Aquí podemos ver como de no conocer bien el problema podríamos haber pensado en un error en los datos cuando vimos la imprecisión del aprendizaje de los modelos en el Banco Santander

Ahora vayamos con la parte buena, las ventajas del aprendizaje por refuerzo:

- Adaptabilidad: esta es una de las grandes ventajas del aprendizaje automático en general y del aprendizaje por refuerzo en particular. Las empresas y los mercados financieros son entes vivos que evolucionan y la que otrora era una empresa pequeña (grande) puede pasar a ser una empresa grande (pequeña). Este cambio que puede parecer trivial cambia el volumen de gran manera y por tanto el comportamiento de los algoritmos de ejecución. Estos procesos generalmente no suelen ser algo repentino[18], por lo que aumentos (disminuciones) de volumen negociado pueden llevar paulatinamente (gracias a esa $\epsilon > 0$) a compras más agresivas (tranquilas)
- Mejora del error: especialmente cuando hemos aplicado aprendizaje profundo la similitud entre los precios de ejecución de los algoritmos y los modelos de aprendizaje eran sorprendentemente parecidos. Difícilmente podría haberse conseguido estos resultados de otra forma

A nivel teórico estas han sido las principales ventajas e inconvenientes. Ahora se va a ver una ventaja que hemos encontrado de forma colateral. El VWAP se calcula de manera síncrona por unidad de tiempo siendo la escala de a minuto la que hemos elegido, mientras que los cambios de precios llegan de forma asíncrona. De hacerlo de la manera tradicional, como si de un árbol de decisión se tratara, puede conllevar mayores costes. Un ejemplo del algoritmo ejecutado de esta manera tradicional podría ilustrarse de la siguiente forma:

1. Determinación de la hora $\Rightarrow$ 11:50 AM
2. Determinación del porcentaje sobre el VWAP en el que se quiere participar $\Rightarrow$ 10 %
3. Cálculo del VWAP a esa hora determinada $\Rightarrow$ 10.000 acciones cruzadas
4. Cálculo de las acciones que se deben adquirir $\Rightarrow$ $10.000 * 10\,\% = 1000$

El problema de este enfoque es que en una escala menor, reproduce la cuestión que nos había traído hasta aquí, dividir las órdenes para tener

[18]Salvo el caso de los splits y contrasplits que multiplicar o dividir el precio y el volumen de una acción por 10 de un día para otro.

menor impacto. Con esta forma tradicional lo que se hace es lanzar esas 1000 acciones a mercado (con el impacto que eso puede traer), independientemente de la velocidad a la que el activo genere el volumen.

Este volumen puede llegar de manera totalmente discreta en momentos muy claros y discretos (propio de activos menos líquidos como puede ser una pequeña empresa) o de una forma totalmente continua. El problema que quiero representar es que el enfoque tradicional lanza las órdenes de manera síncrona y discreta pudiendo impactar más al mercado tanto por tamaño como por el aumento de la probabilidad de encontrar *seco* el mercado eligiendo incorrectamente el momento en el que se lanza las órdenes. Hacerlo asíncronamente nos permite lanzar órdenes de manera no discreta, sino, continua y fluida.

Resultado de los objetivos

En base a los 4 objetivos planteados al inicio, no todos han sido satisfechos, mas, han los resultados han sido bastante satisfactorios. Rememorando los cuatro objetivos:

1. **OBJ-01:** Minimizar el precio medio de adquisición
2. **OBJ-02:** Aprender el POV y el VWAP
3. **OBJ-03:** Implementar el OBJ-01 y el OBJ-02 de forma tabular y con redes neuronales
4. **OBJ-04:** Funcionamiento de los modelos en activos donde no fueron entrenados

El primero de los objetivos, como se ha visto, no ha podido ser satisfecho, mientras que los otros sí que se han podido (con la salvedad de la referencia del tercer objetivo para con mejorar el precio medio de adquisición). Por último, el cuarto objetivo también ha sido satisfecho, funcionando en otros activos donde no ha sido entrenado, aunque con algunos haga falta ciertas modificaciones que se verán en los trabajos futuros.

7.2. Trabajos futuros

A pesar de los resultados obtenidos este debe de ser un final, sino un comienzo. Hay muchas cuestiones por plantear o mejorar entre las que se deben señalar:

- Adaptabilidad a las diferentes empresas y activos
- Acotación de las posibles acciones erróneas por parte del modelo
- Adaptación a los costes de ejecución por parte del resto de agentes
- Uso de órdenes limitadas
- Ponerlo en producción

Como se ha apreciado en los resultados, los modelos diseñados adolecen ante cambios en el tamaño de las empresas y de su volumen negociado. La acción que conllevaba una mayor compra era de 1000 acciones. Si por un casual un activo financiero, una vez aplicado el porcentaje que debemos adquirir del volumen negociado, nos dijera que debemos adquirir más de esas 1000 acciones por unidad de tiempo, el modelo irremediablemente se iría distanciando sistemáticamente. Por el lado inverso también aplica. Si el activo fuera demasiado pequeño, bien sea el tamaño o por la definición de las transacciones, como ocurre con los futuros donde cada futuro puede mover nominales de 10.000-300.000 € se podría tomar una acción de adquisición de 300 millones de € en un único instante temporal.

Por esta razón, el primer trabajo futuro debería ser ajustar estas acciones al activo financiero y que puede ser de dos formas: aumentando las posibles acciones del modelo de aprendizaje automático (con el consecuente aumento de los tiempos de entrenamiento) o categorizar al activo financiero. Uno de los primeros intentos fue usar una gran cantidad de posibles acciones y que la red neuronal adivinara y se adaptara mucho mejor a las necesidades de volumen a ejecutar. El resultado fue que la red se perdía no convergiendo en buenos valores del error con respecto a la función aproximada o convergiendo en valores significativamente peores que usando menos acciones. Por esta experiencia empírica es mejor el segundo caso, el cual se puede realizar teniendo un caso base en las acciones a tomar y encontrar un multiplicador que sea el que mejor se ajuste al activo en particular.

Esta adaptación es interesante tanto para el cambio de activo, como para cambios dentro del propio activo. En las empresas, por ejemplo, son muy típicos el uso de splits y contrasplits. Un split se produce cuando el precio de una acción es muy alto y se decide dividirlo para que sea más fácil de adquirir. Imaginemos que una empresa cotiza a 1000 € y quiere dividir su cotización por 10 pasando a valer 100 €. El volumen negociado se multiplicará casi automáticamente por 10 ya que el tamaño de la empresa (el valor total de la misma) no ha variado. El contrasplit ocurre cuando

pasa lo contrario, se pasa de cotizar a 100 a los 1000 €. Estas decisiones empresariales implican, por tanto, un cambio de facto en el activo y en su volumen.

El segundo problema a mejorar sería el de las acciones erróneas. Imaginemos una empresa muy pequeña a la que aplicamos nuestros modelos de aprendizaje y debido al componente aleatorio adquirimos o vendemos una cantidad muy grande en relación al volumen negociado. El resultado, además del impacto que supone en el mercado, puede llegar provocar la suspensión de cotización del activo. Los activos financieros suelen tener un límite de cotización y una transacción así puede llevarnos a esa situación a la que no queremos llegar. En esencia, habría que acotar en función del activo subyacente un tope para las acciones o contratos comprados o vendidos en un instante temporal determinado. En caso contrario se puede llegar al caso del llamado *fat finger*[19]

El tercero de los puntos a desarrollar es parecido al anterior, pero, en sentido contrario. Mientras en el segundo problema se habla de un tope, aquí se trataría de un mínimo. Las consecuencias de este tope y del mínimo no son simétricas, pero, de todas formas hay que estudiarlo. Lo que puede causar no tener un mínimo es un aumento de costes en las transacciones, aunque no en el impacto de mercado. Las comisiones que se suelen pagar a la hora de tramitar una orden incluyen:

- Brokerage: puede incluir un porcentaje sobre el nominal como ocurre con el mercado bursátil europeo, un coste por unidad como en los derivados o el mercado bursátil americano
- Cánones de mercado: los establece cada bolsa independientemente
- Organismos reguladores: en EEUU ciertos activos son autoregulados por la industria, como es el caso de la NFA, y cobran por contrato tramitado
- Liquidación: la entidad que custodia el valor carga gastos por los apuntes contables
- Tasas Tobin: algunos países aplican un coste por nominal, en el caso de las acciones, o por contrato, en el caso de los derivados

[19]Se refiere como fat finger al error de introducir una cantidad errónea y tramitar esa orden. Es fácil imaginar que entre tramitar una orden de compra por 100.000 y una de 1 millón de euros apenas hay diferencia, salvo en las consecuencias.

Teniendo en cuenta estos valores, pongamos un ejemplo para entender el problema:

- Activo → empresa española cotizando a 1 €
- Brokerage → 0,10 % con un mínimo de 5 €
- Cánones de mercado → 5 €
- Organismos reguladores → 0 €
- Liquidación → 10 €
- Tasas Tobin → 0 €

En total, si el algoritmo nos dijera de comprar 1 única acción a un 1 € habríamos pagado un total de $1 + 5 + 5 + 10 = 21$ €. No tiene sentido ninguno. Por ello, habría que saber diferenciar por clase de activo, mercado y país los costes asociados a la operativa y reoptimizar los algoritmos para que se adapten al POV y al VWAP.

La cuarta de las cuestiones que deberían ser desarrolladas es incluir el uso de órdenes limitadas para la ejecución de órdenes. El uso de órdenes limitadas plantea diferentes cuestiones y de cierta complejidad. La primera sería el tamaño de las posiciones (igual que ocurría con las acciones actuales), la segunda sería implementar el algoritmo que gestione el fill rate y la tercera sería determinar la distancia a la que se colocan las órdenes.

Este algoritmo debe de tomar por ejecutadas aquellas órdenes limitadas que sean cruzadas por el precio, si por un casual un activo está cotizando a 1 € y se coloca una orden limitada de compra a 0,99 antes de que el valor del activo caiga a 0,98 se tendrá certeza de haber sido ejecutada. Si por el contrario, se coloca a ese 0,99 mientras que el valor del activo llega a ese precio pero no se rebasa, tendríamos que aplicar un porcentaje (que desconocemos pero que irá en función del volumen cruzado en el nivel de precios y del número de posiciones en dicho nivel).

Este cuarto problema, posiblemente el que mayor reto técnico suponga puede ser enfrentado desde el punto de vista del aprendizaje, con infinidad de posibles soluciones, siendo las primordiales usar un mayor número de acciones significando de manera discreta el número de acciones, el tipo de orden (a mercado o límite) y la distancia a la que se coloca la orden. Posiblemente, en base a la experiencia cuando a la red se le dan demasiadas opciones, no alcance soluciones óptimas. La segunda posibilidad podría ser

plantear 3 redes distintas. La primera que elija el tipo de orden, la segunda el tamaño de la posición y la tercera (en caso de haber elegido el uso de órdenes limitadas) que decida la distancia a la que se colocan.

El quinto y último problema que debería ser desarrollado puede que sea el más obvio de todos. Poder llevar este estudio a producción con dinero real y con una plataforma concreta que nos permita el desarrollo necesario. Para ello haría falta:

- Determinar la plataforma destino
- Implementar la recogida de los datos en tiempo real y su procesamiento
- Implementar los algoritmos capaces de tomar las decisiones
- Conectar estos algoritmos con la cuenta operativa que será la que asuma las posiciones compradas o vendidas
- Conectar los algoritmos y la cuenta con la toma de decisiones de inversión que sean las que determinen que activo, qué cantidad y en qué momento temporal comprar o vender

Apéndices

Apéndice A

Herramientas utilizadas

En este apéndice se van a exponer las herramientas y bibliotecas utilizadas para la consecución del trabajo. Los desarrollos se han realizado en python y en la tabla A.1 se han detallado tanto estas herramientas y bibliotecas como la versión utilizada y que se ha demostrado funcional.

- *Dash:* es una biblioteca creado por plotly principalmente orientada a la creación de dashboards mediante python y el uso de un navegador web. Ha sido usada para estudiar el comportamiento en tiempo real de los estados visitados.
- *Dash-core-components:* en esta biblioteca se encuentran todos los componentes del dashboard, tales como las tablas o los gráficos de dispersión
- *Dash-html-components:* dash se utiliza para la presentación de dashboards en un navegador web y mediante esta biblioteca se consigue generar el html necesario para alimentar el navegador
- *Datetime:* es una biblioteca nativa de python que nos proporciona el manejo de fechas y horas. Esta biblioteca es usada tanto para limpiar los datos para minimizar la repetición de muestras que fueran virtualmente iguales, ya que apenas hay distancia temporal entre ellas. Con estos datos limpios lo que se hace es agregar el volumen negociado mientras se quita aquellas muestras con menos de 10 segundos entre ellas. También es usada para generar el volumen esperado en el VWAP.
- *Glob:* es una biblioteca que se usa para obtener, mediante un patrón, los ficheros csv con los datos de las cotizaciones.

Herramienta	Versión instalada
Dash	1.10.0
Dash-core-components	1.9.0
Dash-html-components	1.0.3
Datetime	3.7.5
Glob	3.7.5
Jupyter Notebook	5.7.8
Matplotlib	3.2.1
Multiprocessing	3.7.5
Numpy	1.18.2
Os	3.7.5
Pandas	1.0.3
Pickleshare	0.7.5
Python	3.7.5
Scikit-learn	0.22.2.post1
Sys	3.7.5
Tensorflow	2.1.0
Tqdm	4.45.0
VisualChart	6.3.7.2

Tabla A.1: Herramientas y bibliotecas utilizadas

- *Jupyter Notebook:* es una aplicación web que permite crear documentos con código, visualizaciones o incluso text de forma interactiva. Siendo casi más conocido que el python, se utiliza para generar casi todas las visualizaciones.

- *Matplotlib:* es una biblioteca diseñada para crear visualizaciones en python, siendo la responsable de generar la mayoría de las imágenes del proyecto.

- *Multiprocessing:* python, por defecto, funciona con un único núcleo de los procesadores. Para mejorar el rendimiento nos cabe la posibilidad de crear más instancias, mejorar el hardware o usar el resto de los núcleos disponibles. Esta biblioteca está diseñada para gestionar este problema, gestionando los procesos para ser usados todos los recursos disponibles. Es usado para estudiar la liquidación constante y para la limpieza de los datos.

- *Numpy:* es una biblioteca diseñada para trabajar con vectores y matrices en python de una forma eficiente. Es una de las más utilizadas en el proyecto para trabajar con los datos de las cotizaciones y otros cálculos.

- *Os:* es una biblioteca diseñada para hacer uso de cierta funcionalidad propia del sistema operativo tales como la creación de carpetas. Por esta razón, se usa para la limpieza de los datos.

- *Pandas:* es una biblioteca creada para el manejo de datos como una extensión de numpy. Es capaz de crear estructuras de datos como tablas o series de datos. Es ampliamente usada para todos los cálculos y el manejo de datos a lo largo de todo el proyecto.

- *Pickleshare:* es una biblioteca diseñada para la serialización y deserialización de objetos. Se usa para almacenar y reproducir los objetos del entorno o las tablas de los algoritmos de aprendizaje.

- *Python:* es un lenguaje de programación sobre el que se ha diseñado casi todo el proyecto (a excepción de la recogida de datos) lanzado en 1991.

- *Scikit-learn:* es una biblioteca diseñada para trabajar con aprendizaje automático. Ha sido usada para generar la regresión al calcular el VWAP a cada una de las horas del día.

- *Sys:* es una biblioteca creada para interactuar con el intérprete, y más en concreto, para hacer uso de los parámetros con los que se lanza un código en python. De esta forma se pueden concatenar ejecuciones con diferentes parámetros y configuraciones.

- *Tensorflow:* es una biblioteca desarrollada para por Google orientada a solucionar problemas de aprendizaje automático y entrenar redes neuronales. Como resulta evidente, ha sido usada para implementar las redes neuronales tanto en el POV como en el VWAP.

- *Tqdm:* es una biblioteca que nos permite crear barras de progreso. Es muy útil para ver la evolución de los diferentes procesos.

- *VisualChart:* es el software que se ha encargado de toda la recogida de datos y que mediante un código en C# ha sido capaz de almacenar en ficheros CSV toda esta información.

Apéndice B

Documentación técnica de programación

B.1. Introducción

A pesar de ser un trabajo con un alto componente de investigación, esto no quita que los experimentos, códigos y procesos no sean accesibles. Este anexo va a incluir dos secciones, por una parte la recogida de datos con Visual Chart y por otro el acceso al repositorio con el código necesario para realizar la investigación.

B.2. Configuración y uso de la recogida de datos

VisualChart es un software privado que se encarga del procesamiento de datos de cotizaciones de sucesivos activos financieros, que si bien son datos públicos, esto no los hace gratuitos. Las diferentes bolsas cobran por ello.

Para reproducir el comportamiento de este Trabajo Fin de Máster, primero, tenemos que descargar la plataforma desde la página web https://www.visualchart.com. La descarga es gratuita, pero, nos requiere que nos registremos con nuestros datos. Si bien tanto plataforma como el código que se ha diseñado para la extracción de los datos funcionarán correctamente, por defecto se tendrán datos en *delay* no pudiendo observar la información en tiempo real. Dado que estamos en un documento académico, si alguien se

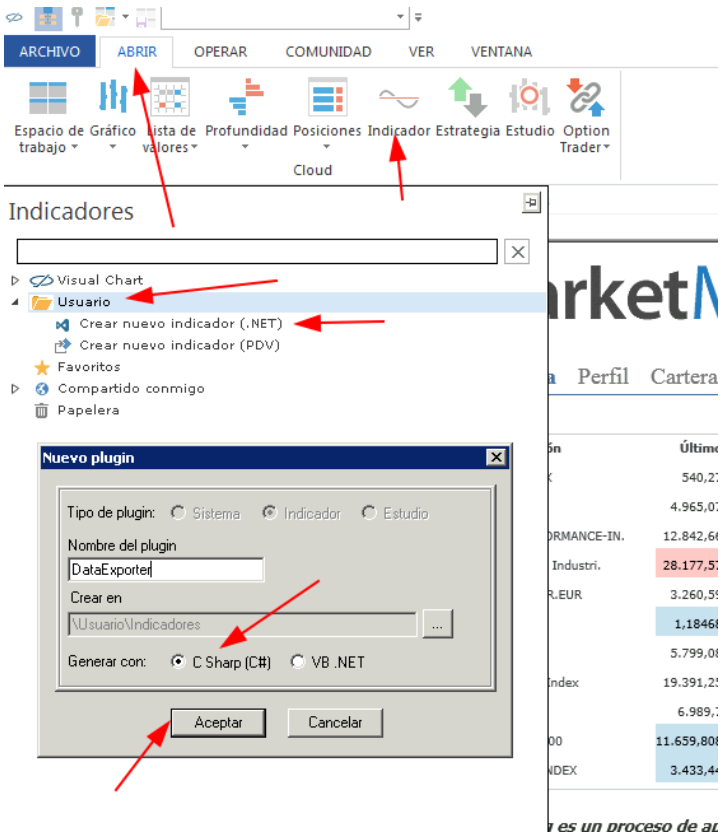


Imagen B.1: Creación de un indicador en Visual Chart

pone en contacto con la empresa desde una institución educativa seguramente le proporcionen el acceso con datos en tiempo real.

Una vez que tenemos descargada la plataforma y creado el usuario, podemos proceder a la instalación, siempre, recordando que sólo funciona con sistemas operativos de Windows. Cabe destacar que es necesario tener instalado Visual Studio y las plantillas de programación que proporciona VisualChart durante la instalación. Cuando abramos VisualChart y se introduzca el usuario y la clave, podremos empezar a extraer los datos. Para realizar esta extracción se tendremos que crear un *Indicador* como se ve en la imagen B.1.

Tras la creación del indicador debe de aparecernos una ventana de Visual Studio donde copiaremos y pegaremos el código fuente del extractor de datos. Tras esta creación, se debe de compilar el código, que será almacenado en los servidores de VisualChart en nombre del usuario, no quedando el código fuente en local en nuestro ordenador.

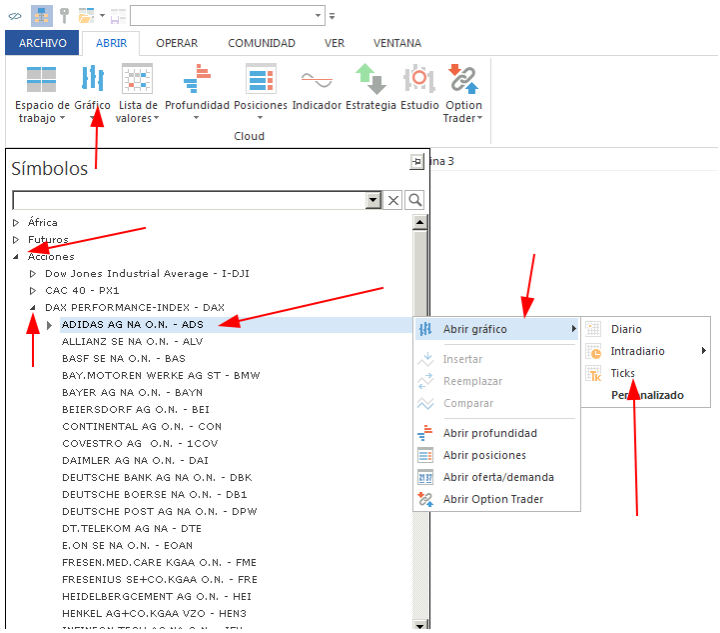


Imagen B.2: Abrir gráfico de Adidas

Una vez se haya compilado correctamente el código, debemos abrir un gráfico por cada uno de los activos sobre el que se quieran extraer los datos de cotizaciones. En la imagen B.2 se puede ver cómo se abre un gráfico de cotización de Adidas.

Para terminar sólo nos queda insertar el indicador que hemos creado sobre el gráfico que acabamos de abrir. Con esto ya estaríamos exportando los datos del activo financiero. Es importante notar que sólo se estarán obteniendo los datos mientras se tengan la plataforma conectada, el gráfico abierto y el indicador insertado para cada activo que queramos.

B.3. Descarga y ejecución del código

Para la descarga de todo el código usado en el Trabajo se hace mediante el enlace a bitbucket `https://https://bitbucket.org/raulgallardop/rg-tfm/`. Una vez descargado el funcionamiento consistiría en instalar los paquetes que contiene el fichero *requirements.txt*. Tras esa instalación el funcionamiento de todos los scripts es el mismo:

1. python3 QLearning-POV.py
2. python3 QLearning-VWAP.py

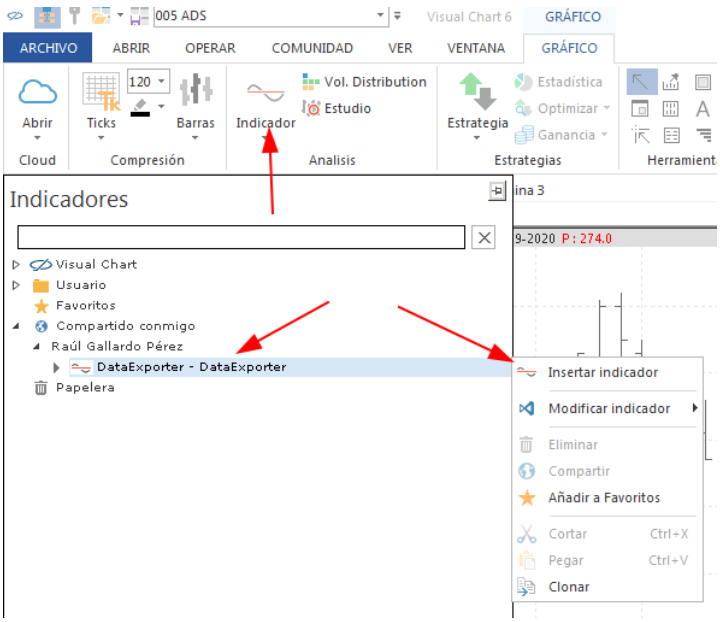


Imagen B.3: Insertar un indicador en un gráfico de cotizaciones

3. python3 QLearning-POV-Keras.py
4. python3 QLearning-VWAP-Keras.py
5. python3 GenerateVWAPData-POV.py
6. python3 POV-Study.py
7. python3 POV-Keras-Study.py
8. python3 VWAP-Study.py
9. python3 VWAP-Keras-Study.py

Hay que destacar que se han separado todos los ficheros para que cada estudio fuera autónomo, ganando en flexibilidad para futuras investigaciones.

Con respecto a los datos usados, se ha incorporado en el repositorio datos de SAP para que se pueda experimentar con ellos. Si por el contrario se quisiera procesar datos de otra empresa, a las ejecuciones antes mostradas sólo habría que añadir el código de la empresa generado por VisualChart. Por ejemplo, para SAP estaríamos hablando de 010005SAP o de 010060ITX.MC si quisiéramos trabajar con Inditex.

Bibliografía

[1] Robert Almgren and Neil A Chriss. Optimal liquidation. *Available at SSRN 53501*, 1997.

[2] Ethem Alpaydin. *Introduction to machine learning*. 2020.

[3] AQR, jun 2019.

[4] Daniel Bernoulli. Exposition of a new theory on the measurement of risk. In *The Kelly capital growth investment criterion: Theory and practice*, pages 11–24. World Scientific, 2011.

[5] John Y Campbell, John J Champbell, John W Campbell, Andrew W Lo, Andrew W Lo, and A Craig MacKinlay. *The econometrics of financial markets*. princeton University press, 1997.

[6] Álvaro Cartea and Sebastian Jaimungal. Optimal execution with limit and market orders. *Quantitative Finance*, 15(8):1279–1291, 2015.

[7] Andrew Conway. http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram, 2013.

[8] James Dixon. https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/, 2010.

[9] David Easley, Marcos Lopez de Prado, and Maureen O'Hara. Optimal execution horizon. *Mathematical Finance*, 25(3):640–672, 2015.

[10] Andrea Frazzini, Ronen Israel, and Tobias J Moskowitz. Trading costs. 2018.

[11] Luca M Gambardella and Marco Dorigo. Ant-q: A reinforcement learning approach to the traveling salesman problem. In *Machine Learning Proceedings 1995*, pages 252–260. Elsevier, 1995.

[12] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[13] Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19, 2015.

[14] Larry Hadersty. https://www.amazon.science/the-history-of-amazons-recommendation-algorithm, 2019.

[15] Hado V. Hasselt. Double q-learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2613–2621. Curran Associates, Inc., 2010.

[16] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.

[17] Vladimir Illievski. https://towardsdatascience.com/animated-visualization-of-brownian-motion-in-python-3518ecf28533, 2016.

[18] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[19] Jannes Klaas. *Machine learning for finance: principles and practice for financial insiders*. Packt Publishing Ltd, 2019.

[20] Max Kuhn, Kjell Johnson, et al. *Applied predictive modeling*, volume 26. Springer, 2013.

[21] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[23] Brian Ning, Franco Ho Ting Ling, and Sebastian Jaimungal. Double deep q-learning for optimal execution. 12 2018.

[24] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[25] Clifton Phua, Damminda Alahakoon, and Vincent Lee. Minority report in fraud detection: classification of skewed data. *Acm sigkdd explorations newsletter*, 6(1):50–59, 2004.

[26] Foster Provost and Tom Fawcett. Data science and its relationship to big data and data-driven decision making. *Big data*, 1(1):51–59, 2013.

[27] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[28] Stuart Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2002.

[29] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2011.

[30] R.S. Tsay. *Analysis of Financial Time Series*. CourseSmart. Wiley, 2010.

[31] Jake VanderPlas. *Python data science handbook: Essential tools for working with data*. .º'Reilly Media, Inc.", 2016.

[32] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.