UNIVERSIDAD DE SEVILLA

DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS

Aplicación de

Técnicas de aprendizaje automático para la predicción de tráfico marítimo en el contexto de toma de decisiones de ámbito empresarial

---

Memoria de Tesis Doctoral para la obtención del grado de Doctor en Informática presentada por

Isidro Lloret Galiana

co-dirigida por los doctores

D. José Antonio Troyano Jiménez

D. Fernando Enríquez de Salamanca Ros

Sevilla, julio de 2021.

A mi madre.

Índice general

Índice de figuras

Índice de cuadros

Agradecimientos

Agradecemos a la Autoridad Portuaria de la Bahía de Algeciras (APBA) la cesión de los datos utilizados en esta investigación. Además, la APBA también se ha prestado amablemente a resolver las dudas planteadas acerca de la comprensión de estos.

Capítulo 1

Motivación e introducción

Existen infinidad de oportunidades de aplicación de técnicas de aprendizaje automático al ámbito empresarial. A pesar de ello, los casos de éxito de este tipo de proyectos no son tan abundantes como se podría esperar. Solo las grandes empresas, y en especial las relacionadas con el ámbito de internet, tienen totalmente asumido el potencial de estas técnicas y las ventajas competitivas que pueden reportarles.

Una de las principales razones por las que estas técnicas no se han extendido de forma generalizada al ámbito empresarial, reside en la dificultad de comunicación entre dos perfiles profesionales muy distantes: el experto del dominio (que bien puede ser un directivo de la empresa o un técnico) y el experto en la tecnología (que debe manejar tanto herramientas informáticas como conceptos matemáticos y estadísticos). Para que un proyecto de esta naturaleza prospere, el experto en tecnología debe aprender a interpretar los datos generados desde la empresa y, al mismo tiempo, el experto en el dominio debe comprender qué tipo de soluciones pueden ofrecerle las técnicas de aprendizaje automático. Solo de esta forma, puede entablarse un diálogo efectivo desde el que construir una solución eficaz, y a medida, que ayude a la toma de decisiones a partir de los datos.

Los puertos marítimos son elementos de gran importancia en las redes de transporte de las cadenas de suministro. Las autoridades portuarias son responsables de tomar decisiones sobre estas infraestructuras y sus instalaciones a la luz de las predicciones de demanda de transporte de mercancías [67, 13]. El flujo de mercancías desagregado permite un análisis segmentado de la futura demanda de transporte de mercancías para ayudar a priorizar las inversiones en transporte y el crecimiento de la industria logística [27]. Es posible analizar los tráficos a potenciar por una autoridad portuaria o la comunidad portuaria en su conjunto, así como la detección de nuevos nichos de mercado, mediante un análisis minucioso por tipología de producto acabado,

semi-acabado y materias primas, siguiendo el código arancelario, tanto de mercancías en régimen de tránsito como de importación-exportación. También se podría analizar los puertos de origen y destino de las mercancías, tanto a nivel nacional como internacional, intentando detectar las posibilidades de incremento de la competitividad y los volúmenes de tráfico.

El objetivo de este trabajo es evaluar la utilidad de las técnicas de aprendizaje automático en la predicción del flujo de mercancías a través de los puertos marítimos. Las técnicas de aprendizaje automático basan sus predicciones en modelos inferidos a partir de datos. Existen múltiples aplicaciones del aprendizaje automático en muchos dominios, como la sanidad, la educación o el deporte, pero sólo recientemente se ha aplicado a las transacciones comerciales, por lo que su aplicación al estudio de las tendencias del comercio internacional es limitada [4].

Nos interesan especialmente los modelos de aprendizaje profundo. El aprendizaje profundo es un tipo específico de aprendizaje automático que permite a una máquina aprender conceptos complicados construyéndolos a partir de conceptos más simples. Los diferentes componentes de este tipo de modelos se organizan en una estructura profunda y por capas [23].

1.1. Competiciones de predicción M

Las Competiciones M[1] son un conjunto de competiciones de predicción de series temporales lideradas por Spyros Makridakis, uno de los principales expertos mundiales en predicción.

Según S. Makridakis et al. (2018) [46], los métodos de predicción basados en aprendizaje automático (*ML*; *machine learning*) necesitan obtener mejores resultados, requiriendo un menor tiempo de procesamiento, y no funcionar como una caja negra. También comentan que la capacidad de los modelos de ML en la determinación efectiva de la estacionalidad y la tendencia es un tema controvertido. Tras la celebración de la Competición M4 en 2018, los mismos autores [47] confirman que los resultados de los métodos presentados, basados puramente en ML, no llegan a superar ni siquiera todos los métodos de referencia. Por otra parte, estos autores critican los buenos resultados publicados sobre modelos basados en ML, donde encuentran una serie de limitaciones que han servido de motivación en la realización de este trabajo de investigación, y que son:

- Conclusiones basadas en muy pocas o incluso una serie temporal.

[1] `https://mofc.unic.ac.cy/`

- Se realizan predicciones a corto plazo, y a menudo de un solo paso, no considerando medio y largo plazo.
- No se comparan los resultados con otros métodos de referencia.

Por estos motivos, desde un primer momento hemos planteado nuestro trabajo de investigación como un reto, el de superar los métodos de referencia de la Competición M4. Con este propósito, hemos creado varios modelos puramente basados en ML para la predicción de un conjunto de series temporales seleccionadas, con diferentes niveles de granularidad y horizontes de predicción para corto, medio y largo plazo. Además, hemos utilizado las mismas métricas de evaluación de los resultados de predicción definidas en esta competición.

R. J. Hyndman (2020) [33] hace un resumen de las cinco competiciones celebradas hasta la fecha actual:

- M1 (1982). Se utilizaron 1001 series. Generó mucha controversia una de las conclusiones publicadas acerca de que los métodos estadísticos sofisticados o complejos no necesariamente proporcionaban mejores resultados, motivando las siguientes dos competiciones M.
- M2 (1993). Se utilizaron solo 29 series temporales. Los resultados fueron estadísticamente los mismos que en la competición anterior.
- M3 (2000). Se utilizaron un total de 3003 series disponibles públicamente, obtenidas de diferentes dominios y con granularidad anual, mensual, semanal y diaria. Las conclusiones alcanzadas fueron similares a las de las competiciones anteriores.
- M4 (2018). Replicó los mismos resultados de competiciones anteriores, utilizando un conjunto extendido y diverso de 100.000 series. Se incorporaron como métodos de referencia los principales métodos de predicción estadísticos, incluyendo dos basados en ML. Las series temporales y el código fuente de algunos de los métodos participantes son públicos.
- M5 (2020). Es una competición diferente a las anteriores en algunos aspectos, que fueron sugeridos por los participantes de la M4, entre los que destacamos:
  - Uso de datos de ventas jerarquizados.
  - Junto con las series se incluyen variables explicativas el precio, promociones, día de la semana y eventos especiales.

- La mayoría de las 42.840 series utilizadas mostraban intermitencia en los datos (valores a cero).

Todas las competiciones M han dado lugar a publicaciones que son de referencia en el contexto de la predicción de series temporales. La Competición M4 ha derivado en una serie de trabajos publicados en un número especial de la revista *International Journal of Forecasting*[2]. Algunas de estas publicaciones y otras anteriores relacionadas han sido de gran utilidad en nuestro trabajo de investigación.

1.2. Organización de este trabajo

Comenzamos en el capítulo 2 con una introducción al entorno portuario en cuanto a sus tecnologías y sistemas de información, así como a la generación de datos estadísticos, y a partir de estos, la obtención del conjunto de series temporales utilizadas en nuestra investigación. En el capítulo 3 tratamos los principales conceptos relacionados con la predicción de series temporales junto con las métricas de evaluación de la exactitud de las predicciones. Continuamos en el capítulo 4 con la explicación de los métodos de predicción de referencia de la Competición M4 y los estándares de predicción ETS y ARIMA. En el capítulo 5 proponemos un algoritmo de predicción por búsqueda de coincidencia (*matching pursuit*), donde se aproxima la serie temporal a predecir mediante la suma de un conjunto de series sinusoidales extraídas de un diccionario. Seguimos en el capítulo 6 con una introducción al aprendizaje profundo y a las redes neuronales, y en el capítulo 7 con la descripción de nuestra tarea concreta de predicción, las series temporales utilizadas y el enfoque basado en aprendizaje automático. Posteriormente, explicamos los dos modelos de redes neuronales profundas que presentamos como principal aportación de este trabajo en cuanto a los resultados obtenidos: el modelo convolucional, basado en una pila de capas convolucionales causales dilatadas, y el modelo recurrente con arquitectura codificador-decodificador, basado en capas GRU, que se detallan en los capítulos 8 y 9 respectivamente. La metodología utilizada en nuestra experimentación con estos dos modelos la exponemos en el capítulo 10. Los resultados obtenidos junto con gráficas que muestran la calidad de los resultados se presentan en el capítulo 11. Finalmente, en el capítulo 12 incluimos las conclusiones de nuestro trabajo de investigación.

[2] `https://www.sciencedirect.com/search/advanced?pub=International%20Journal%20of%20Forecasting&cid=271676&volume=36&issue=1/`

Capítulo 2

El entorno marítimo portuario

Una vez definido el tipo de trabajo de investigación a realizar en esta tesis, en la que nuestro interés está en aplicar técnicas de aprendizaje automático a la toma de decisiones de ámbito empresarial, debíamos encontrar una entidad que pusiera a nuestra disposición los datos para llevarla a cabo. Particularmente, en la zona del Campo de Gibraltar disponemos de buenas relaciones con la Autoridad Portuaria de la Bahía de Algeciras (APBA), siempre abierta a la colaboración con universidades, y que amablemente nos ha cedido estos datos además de dedicar una parte de su tiempo a explicarnos su significado, ya que no estamos acostumbrados a la terminología marítimo portuaria.

Merece la pena y además nos sentimos obligados a empezar este capítulo con una sección en la que describimos el entorno portuario en cuanto a las tecnologías y sistemas de información que originan los datos y permiten su almacenamiento para el posterior procesamiento por las autoridades portuarias. En concreto, los datos que hemos utilizado en este trabajo son los relativos a las estadísticas de tráfico marítimo de importación de mercancías de Marruecos a España. En la segunda sección explicaremos brevemente el proceso de realización de estas estadísticas y la simplificación que hemos hecho de sus datos, obteniendo un modelo conceptual simple con la información únicamente necesaria para la predicción del tráfico de mercancías.

2.1. Tecnologías y sistemas de información

Hoy en día nos encontramos inmersos en una constante innovación tecnológica que es necesario aprovechar desde el punto de vista de la competitividad a nivel global. Esto obliga a que la gestión de los puertos se enfoque hacia su integración como nodos logísticos intermodales de las cadenas de transporte. Para prestar servicios intermodales, los actores portuarios de-

sarrollan diferentes formas de relaciones inter-organizacionales destinadas a gestionar los recursos que conduzcan a la satisfacción del cliente, como en el transporte por carretera o por ferrocarril [16]. El uso de nuevas tecnologías busca una logística más eficiente, satisfaciendo de la mejor manera posible las necesidades de los usuarios, además de cuidar la protección medioambiental de forma complementaria. De ahí surge el concepto de Puertos Inteligentes (*Smart Ports*). Los factores clave que contribuyen al éxito de las tecnologías y sistemas de información de los *Smart Ports* son la capacidad de satisfacer las demandas cambiantes de los usuarios y la capacidad de adaptarse a los nuevos avances, sin necesitar reestructurarse constantemente [3].

La adquisición de datos operacionales depende en gran medida de las tecnologías de la información avanzadas [29]. A continuación, presentamos un resumen de estas en el ámbito portuario:

- Los sistemas de navegación por satélite (*GPS*; *global positioning system*) permiten el posicionamiento y seguimiento de contenedores, buques, vehículos, etc.

- El intercambio electrónico de datos (*EDI*; *electronic data interchange*) es utilizado por los principales puertos. En Europa el estándar *UN/EDIFACT* (*United Nations/EDI for administration, commerce and transport*), define varios tipos de mensajes EDI que dan soporte a las operaciones portuarias.

- La tecnología de identificación por radiofrecuencia (*RFDI*; *radio frequency identification*) permite la identificación e intercambio de información de objetos etiquetados.

- El reconocimiento óptico de caracteres (*OCR*; *optical character recognition*) se usa para la identificación de contenedores y vehículos. Tiene la ventaja respecto a *RFDI* de que se evita que el objeto a identificar vaya equipado con la misma tecnología.

- Las redes de sensores inalámbricos (*WSN*; *wireless sensor networks*) están compuestas por sensores interconectados de forma que cubren un determinado espacio y monitorizan condiciones físicas o ambientales de forma cooperativa.

- Los dispositivos móviles disponen de grandes capacidades a nivel de computación, comunicación y sensores, que pueden ser aprovechadas en infinidad de aplicaciones, como la desarrollada por nosotros mismos

para la APBA[1], que integra el uso de la cámara y el envío/recepción de mensajes *SMS* para la vigilancia del correcto funcionamiento de las linternas de señalización marítima cada cierto tiempo, comunicándose con una estación base.

- Las tecnologías de la comunicación deben dotar a los puertos de redes inalámbricas fiables (equipadas para entornos difíciles) que permitan conectar dispositivos móviles, sensores y actuadores, a través de *routers* equipados con diferentes opciones de conectividad.

Los sistemas de información se han vuelto indispensables para la competitividad de los puertos, facilitando la comunicación y la toma de decisiones para mejorar la visibilidad, la eficiencia, la fiabilidad y la seguridad en las operaciones portuarias [29]. Describimos a continuación los principales sistemas de información portuaria:

- La ventanilla única nacional (*NSW*; *national single window*). Un problema que viene de lejos en el sector del transporte marítimo es la complejidad y el tiempo que conlleva la presentación de documentos a la entrada y salida de los puertos. Los consignatarios de los buques siguen soportando el peso de tener que rellenar documentos, en algunas ocasiones en papel, que incluyen información similar y distribuirlos a las diferentes autoridades. Esto aumenta los costes y provoca retrasos, reduciendo la competitividad del transporte marítimo. La ventanilla única nacional se concibe como el único punto en el que se comunica por una sola vez la información a declarar con ocasión de la escala de un buque en un puerto. En el caso europeo (Directiva 2010/65/UE[2]) cada país dispone de una ventanilla única, que debe compartir con todos los países miembros la información reglamentaria aduanera relacionada con las operaciones de importación, exportación y tránsito de mercancías. En España la ventanilla única la ofrece el organismo Puertos del Estado, que a efectos de que los sistemas sean interoperables y compatibles, ha establecido y publicado los formatos de puesta a disposición de la información requerida a los buques, junto con la sintaxis y la estructura de los mensajes de intercambio electrónico, y las reglas de negocio y procedimientos necesarios. Las autoridades portuarias son los puntos de acceso local a la ventanilla nacional.

[1]Desarrollo de software Android para vigilancia de linternas de señalización marítima. I. Lloret. Universidad de Cádiz. Contrato con APBA, art. 11/45 LRU - 68/83 LOU, 2013.

[2]Directiva 2010/65/UE del Parlamento Europeo y del Consejo, de 20 de octubre de 2010, sobre las formalidades informativas exigibles a los buques a su llegada o salida de los puertos de los Estados miembros.

- Los sistemas de comunidad portuaria (*PCS*; *port community systems*) son plataformas de software inter-organizacionales que interconectan a los actores de la comunidad portuaria, permitiendo los servicios comerciales y el intercambio de información entre el puerto y sus clientes, y una diversidad de partes interesadas [53]. Estas plataformas se crearon comúnmente sobre la base de los estándares EDI. Por ejemplo, en Europa cada PCS debe facilitar una ventanilla única. Actualmente, el Puerto Bahía de Algeciras dispone del PCS denominado Teleport 1.0, que se engloba en dos grandes bloques: servicios dirigidos al buque y servicios dirigidos al tráfico pesado. Se tiene planeado introducir nuevas funcionalidades etapa por etapa, hasta llegar a la nueva versión Teleport 2.0, que se centrará en servicios, procesos y colaboración comunitaria.

- El servicio de tráfico de buques (*VTS*; *vessel traffic system*) es un sistema de información marítimo crítico en la mar, tanto en términos de seguridad como de eficiencia, que permite controlar los movimientos de un buque en un área geográfica limitada, especialmente en las zonas de mayor confluencia como las cercanas a lo puertos, a los que informan de forma precisa sobre la hora de llegada de los buques, lo que permite planificar eficientemente los operaciones a llevar a cabo. Pueden proporcionar servicios de información, asistencia a la navegación y organización del tráfico. Se pretende que el capitán de un buque pueda tomar en todo momento la decisión náutica más adecuada para realizar un tránsito seguro por estas aguas. Particularmente, la zona del Estrecho de Gibraltar soporta una enorme densidad de tráfico marítimo al ser punto obligado de paso para todos los buques cuyas líneas unen los puertos del Atlántico y Norte de Europa con los del Mediterráneo e incluso los más importantes puertos de las costas de Asía y Golfo Pérsico, a través del Canal de Suez, unido al tráfico rutinario de buques entre España y Ceuta, Melilla y Marruecos. El servicio de tráfico marítimo de esta zona está ubicado en Tarifa y dispone de radares, sistema de identificación automática de buques (*AIS*; *automatic identification system*) basado en GPS y sistemas de comunicaciones. En esta zona la Organización Marítima Internacional (*IMO*; *international maritime organization*) estableció un dispositivo de separación de tráfico para encauzar los flujos de tráfico en las direcciones este y oeste, e incrementar la seguridad de la navegación en la zona. Es muy importante que el personal operativo en este servicio de tráfico esté altamente cualificado para el puesto y sea capaz de tomar las decisiones correctas en cada situación. En este sentido cabe destacar el reciente

estudio realizado en la Universidad de Cádiz en el año 2014 [15], que resalta la necesidad de contar con un test fiable para la selección de futuros operadores de control del tráfico marítimo basado en la medida de la conciencia situacional. Logrando mantener la conciencia situacional se potencia la adquisición, la representación, la interpretación y la utilización de cualquier información relevante para comprender los eventos que ocurren, de forma que el futuro operador pueda anticiparse a los acontecimientos futuros, tomando decisiones inteligentes y manteniendo el control.

- Un sistema operativo de terminal (*TOS*; *terminal operating system*), tiene como objetivo principal controlar el movimiento y el almacenamiento de la carga de una terminal de contenedores. Integra diferentes módulos y sistemas expertos que dan soporte a la asignación de amarres, la planificación de la estiba y la asignación de otros recursos.

- Los sistemas de reserva de puerta (*GAS*; *gate appointment systems*) permiten racionalizar y planificar la llegada de camiones en las terminales mediante la asignación de ventanas de tiempo específicas para la entrega de contenedores, ofreciendo un trato preferencial a los camiones que eligen programar esa cita.

- Los sistemas de apertura automática de puertas posibilitan una rápida identificación de contenedores, camiones y conductores, integrándose dicha información en el TOS.

- Los sistemas de patio automatizados conducen horizontalmente los contenedores entre el muelle y el patio, además de facilitar el apilamiento del patio. La automatización puede incluir otras áreas como las puertas de terminal y las grúas de muelle.

- Los sistemas de información para el control de vías y tráfico portuario están orientados a minimizar la congestión, los accidentes y la reducción de emisiones contaminantes, con el fin de incrementar la productividad en el puerto y el ahorro energético.

- Los sistemas de transporte inteligente (*ITS*; *intelligent transportation systems*) engloban un conjunto de soluciones tecnológicas de las telecomunicaciones y sistemas informáticos aplicados a los vehículos e infraestructuras de transporte por carretera, para mejorar el rendimiento de los sistemas de transporte. En Europa, se diseñó en 2010 un mar-

co legal (Directiva 2010/40/UE[3]) y un plan de acción correspondiente para acelerar su despliegue.

- Los sistemas intermodales de información puerto-interior hacen más eficiente la integración de los sistemas portuarios con las redes logísticas interiores. En Europa el proyecto FutureMed desarrollado con fondos europeos investiga este tipo de opciones para tráfico de mercancías, de pasajeros y turístico [69].

2.2. Datos estadísticos de tráfico marítimo portuario

La Directiva 2009/42/CE[4] obliga a los estados miembros de la Unión Europea a recoger y facilitar estadísticas comparables, fidedignas, sincronizadas y regulares sobre el volumen y la evolución del transporte marítimo de mercancías y de pasajeros efectuado por buques que hagan escala en los puertos situados en su territorio.

La elaboración de las estadísticas de tráfico en los puertos de titularidad estatal le corresponde a Puertos del Estado según se indica en la Ley de Puertos[5]. Estas recogerán los movimientos de mercancías de cualquier tipo, pesca y avituallamientos entre los buques en puerto y tierra u otros medios de transporte; escalas de buques mercantes; así como el embarque y desembarque de pasajeros y vehículos. Las autoridades portuarias ponen a disposición de los usuarios el suelo y las infraestructuras del dominio público portuario para realizar sus actividades, y también prestan servicios directamente a su cargo, recaudando mediante tasas lo establecido en la Ley. Las tasas se calculan, para operaciones de carga y descarga, en función de las toneladas de mercancías, número de contenedores, kilos de pesca fresca capturada, etc. La información viene declarada en una serie de documentos que son los denominados manifiestos de carga, declaraciones sumarias de depósito temporal, manifiestos de pesca, declaraciones únicas de escala y otros utilizados para el cálculo de consumo de suministros [55]. Las bases de datos de las autoridades portuarias registran cada movimiento declarado de carga o descarga de

[3]Directiva 2010/40/UE del Parlamento Europeo y del Consejo de 7 de julio de 2010 por la que se establece el marco para la implantación de los sistemas de transporte inteligentes en el sector del transporte por carretera y para las interfaces con otros modos de transporte.

[4]Directiva 2009/42/CE del Parlamento Europeo y del Consejo, de 6 de mayo de 2009, sobre la relación estadística del transporte marítimo de mercancías y pasajeros

[5]Real Decreto Legislativo 2/2011, de 5 de septiembre, por el que se aprueba el Texto Refundido de la Ley de Puertos del Estado y de la Marina Mercante.

mercancía, almacenándose toda la información necesaria, como fecha, peso o unidades, lugar de origen y destino, código arancelario, etc. Aparte de su uso en estadísticas y para el cobro de tasas, esta información puede servir de ayuda en la toma de decisiones en el ámbito de la gestión portuaria. En concreto, las estadísticas de comercio exterior son cruciales [17], entre otros, para los siguientes fines:

- Análisis de los patrones comerciales: origen y destino geográfico de los flujos de mercancías.
- Estudio del patrón geográfico del comercio y su evolución como clave para diseñar acuerdos comerciales con países específicos.
- Análisis de los efectos de creación y desviación del comercio tras la celebración de determinados acontecimientos o la firma de acuerdos.
- Impacto en los flujos comerciales de la reducción de las barreras arancelarias y no arancelarias.

2.2.1. Obtención de series temporales de tráfico marítimo

Actualmente, el estándar europeo UN/EDIFACT es el más utilizado mundialmente en la transmisión de documentos electrónicos de la cadena de suministros, y en particular en facturación electrónica. Los mensajes EDIFACT están estructurados jerárquicamente para contener toda la información del documento original. Cuando un buque mercante desea entrar en un puerto envía un mensaje de *solititud de escala* a través de la ventanilla única que ofrecen las autoridades portuarias. Se inicia entonces un proceso en el que el buque debe informar sobre las mercancías que desea cargar y descargar, para lo cual puede enviar diversos mensajes de *manifiesto de carga* y de *declaración sumaria* respectivamente. La autoridad portuaria comunica los datos aduaneros de forma inmediata a nivel nacional y europeo (figura 2.1), y los datos estadísticos de forma periódica cuando le sean requeridos.

En nuestro trabajo de investigación hemos utilizado un conjunto de datos de tráfico de mercancías extraído de las bases de datos de la APBA. Para comprender el significado y la estructura de estos datos es necesario describir básicamente los documentos EDIFACT de solicitud de escala, manifiesto de carga y declaración sumaria. Utilizaremos las guías que Puertos del Estado tiene a disposición pública sobre estos mensajes electrónicos para la interacción con las autoridades portuarias (ventanilla única nacional):

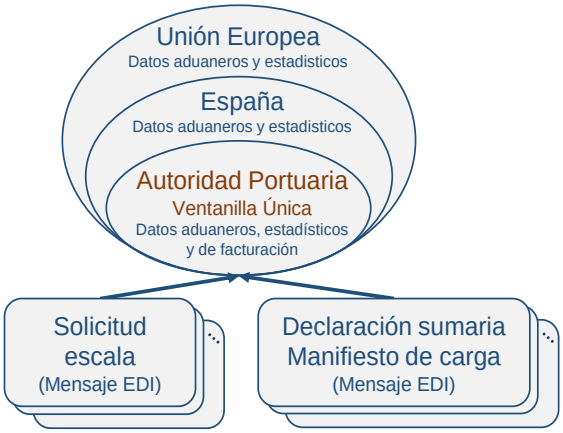

Figura 2.1: La autoridad portuaria como ventanilla única recibe información sobre peticiones de escalas de buques y movimientos de mercancías, que se utilizarán a efectos aduaneros, estadísticos y para el cobro de tasas portuarias. Los datos estadísticos y aduaneros son comunicados a nivel nacional y europeo.

- Mensaje de solicitud de escala [56]. Es un mensaje enviado por el consignatario del buque a la autoridad portuaria (AP) que contiene datos de identificación del buque, línea, y datos generales sobre la mercancía que transporta y los tipos de movimientos a efectuar. En respuesta a este mensaje la AP asigna un número de escala para el año en curso, que es referencia única para la transacción y para un viaje y medio de transporte determinado. Se almacena en las bases de datos la fecha de entrada y de salida de aguas en el momento que se produzcan. En resumen, los datos de la escala de interés para nuestro estudio son: año de la escala, número de escala, y las fechas de entrada y de salida de aguas.

- Mensaje de comunicación de declaración sumaria y manifiesto de carga [57]. La guía nos indica que se usa un mismo documento para las dos comunicaciones, denominándolo Declaración Sumaria, y que por tanto se utiliza para cualquier mercancía que un barco con escala en puerto vaya a cargar o descargar. De esta manera, la información contenida en una declaración sumaria se desglosa principalmente en los dos apartados siguientes:

  - Información relativa al buque, viaje y escala: en este apartado

nos interesan los datos de la escala, comentados anteriormente, además del tipo de movimiento a realizar (carga/descarga).

- Información relativa a la carga transportada: en este apartado nos interesan los datos relativos a cada conocimiento de embarque[6] y las partidas que contiene[7].

Los mensajes con esta información son enviados a la AP por el consignatario del buque y el de cada conocimiento de embarque. Se comienza con el envío de un mensaje del consignatario del buque indicando, entre otros datos, el tipo de movimiento a realizar (carga/descarga). A continuación, para esta operación de carga/descarga declarada, los consignatarios de mercancía envían mensajes con la información de cada conocimiento de embarque, identificado por su número. Por cada conocimiento de embarque los consignatarios deben detallar, entre otros, los datos de puerto de origen y destino, así como el modo de transporte previo (con el que las mercancías llegan al puerto) o posterior (con el que las mercancías salen del puerto), y de las diferentes partidas contenidas, el código de partida y el peso. La figura 2.2 muestra un modelo para estos datos con las diferentes entidades y su relación.

A partir de este modelo de datos es posible convertir la información transaccional de cada movimiento realizado en el puerto, donde se incluye información a diferentes niveles (fecha de entrada en aguas de la escala, tipo de movimiento, número de conocimiento de embarque, partida arancelaria y toneladas), en una información analítica seleccionada. En la figura 2.3 se muestra un ejemplo de transformación de estos datos transaccionales en un conjunto de series temporales de importación de mercancías de Marruecos a España, correspondientes a unos determinados códigos arancelarios. Entendemos por importación aquellas mercancías que parten de un puerto extranjero con descarga en el Puerto de Algeciras, continuando la mercancía su viaje por carretera o ferrocarril hasta su destino.

En este capítulo hemos pretendido dar una visión global del entorno marítimo portuario a nivel de los sistemas de información existentes, y particularmente de la información que fluye a través de la ventanilla única en relación con la declaración de mercancías por parte de los buques en sus operaciones de carga y descarga en puerto. A partir del análisis de estos datos

[6]El conocimiento de embarque (BL; *bill of lading*) es un documento contractual propio del transporte marítimo en el que se identifican los actores que intervienen, así como la carga transportada y el consignatario o receptor de la mercancía.

[7]Una partida de mercancía se caracteriza por tener un único código arancelario y pertenecer a un único conocimiento de embarque.

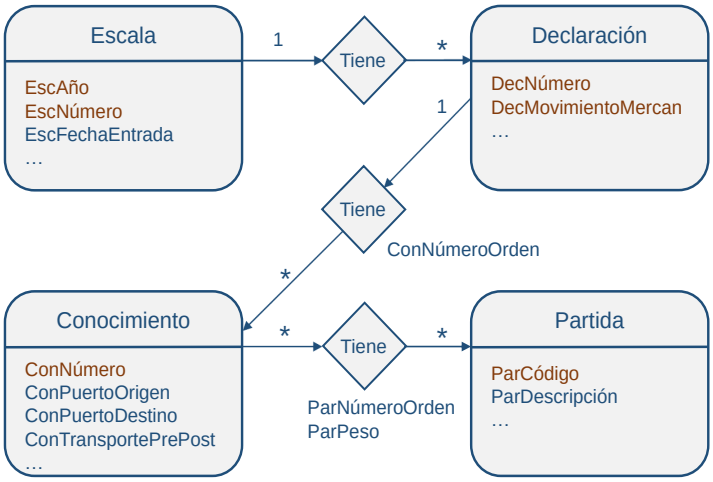

Figura 2.2: Modelo de datos entidad/relación del movimiento de mercancías en un puerto marítimo de titularidad estatal. En color naranja se muestran las claves principales.

hemos podido comprender cómo se obtiene el conjunto de datos utilizado en este trabajo de investigación, y que ha sido amablemente cedido por la Autoridad Portuaria de la Bahía de Algeciras.

Datos transaccionales

Esc Año	Esc Número	EscFecha Entrada	DecNúmero	Dec Mvto	Con Nº Orden	Con Transp. PrePost	Mc Puerto Origen	Con Puerto Destino	Par Nº Orden	Par Código	Par Peso
2000	20093	24/01/2000	11310520093	2	1	ZZ2	MATNG	ESALG	1	0302	7,871
2000	20093	24/01/2000	11310520093	2	2	ZZ2	MATNG	ESALG	2	0302	12,036
2000	20093	24/01/2000	11310520093	2	3	ZZ2	MATNG	ESALG	3	0302	12,386
2000	20093	24/01/2000	11310520093	2	4	ZZ2	MATNG	ESALG	4	0302	10,296
2000	20093	24/01/2000	11310520093	2	5	ZZ2	MATNG	ESALG	5	0302	7,428
2000	20093	24/01/2000	11310520093	2	6	ZZ2	MATNG	ESALG	6	0302	9,776
2000	20093	24/01/2000	11310520093	2	7	ZZ2	MATNG	ESALG	7	0302	24,271

Semana	Capítulo 03	Capítulo 07	Capítulo 10	Capítulo 16	Capítulo 62	Capítulo 69	Capítulo 85
08/01/2000	791,979	8.412,843	0.000	73,433	0.000	23,358	138,988
15/01/2000	304,595	6.444,314	0.000	115,848	0.000	0.000	103,661
22/01/2000	739,750	7.094,340	0.000	158,931	6,380	36,249	155,589
29/01/2000	971,840	7.474,971	0.000	93,323	103,183	8,153	100,262
05/02/2000	1.006,829	9.448,060	0.000	106,626	19,386	1,350	125,611
12/02/2000	1.023,876	10.379,844	0.000	206,781	17,627	70,299	130,997
19/02/2000	988,155	11.175,882	0.000	184,026	0.000	71,489	192,386
26/02/2000	1.134,311	7.973,192	15,050	279,024	0.000	23,120	167,283

Datos analíticos

Figura 2.3: Generación de datos analíticos a partir de datos transaccionales para predicción de tráfico marítimo. La mercancía importada semanalmente de Marruecos a través del Puerto de Algeciras para diferentes capítulos arancelarios se obtiene a partir de los datos transaccionales de cada movimiento de descarga de mercancías (*DecMovimientoMercan='2'*), con transporte previo/posterior por carretera o tren (*ConTransportePrePost='ZZ2' or ConTransportePrePost='ZZ6'*), puerto de origen en Marruecos (*ConPuertoOrigen='MA*'*), puerto de destino Algeciras (*ConPuertoDestino='ESALG'*) y los capítulos arancelarios seleccionados mediante los dos primeros dígitos del código arancelario.

Capítulo 3

Introducción a la predicción de series temporales

La predicción del futuro es algo apasionante, para lo cual es necesario conocer el pasado y la forma en que condiciona los próximos sucesos, especialmente cuando se rompe la tendencia o estacionalidad/periodicidad. En este capítulo trataremos de presentar los aspectos más relevantes de las series temporales en relación con la tarea de predicción, así como los métodos ingenuos de predicción y las métricas de evaluación de la exactitud de las predicciones.

En el desarrollo de este trabajo de investigación, siempre utilizaremos el termino *serie temporal* para hacer referencia a series temporales univariantes, por tanto, cada predicción estará basada en los valores previos de una sola variable endógena y ninguna exógena.

3.1. Introducción

Una serie temporal es una secuencia de observaciones $\mathfrak{o}_t$ tomadas de una variable $\mathfrak{O}$ en instantes ordenados de tiempo t, que para nuestros fines consideraremos equiespaciados, y que será representada como

$$\mathfrak{O} = \{\mathfrak{o}_1, \mathfrak{o}_2, \ldots, \mathfrak{o}_T\}, \tag{3.1}$$

donde $\mathfrak{o}_T$ es la última observación.

La figura 3.1 muestra una serie temporal correspondiente a la actividad de una termita, capturada mediante un sensor acelerómetro. En esta serie resultaría muy interesante caracterizar los eventos producidos por los insectos, de forma que posteriormente se pudiera aplicar este conocimiento a otras series temporales (incluso con mayor ruido) para conseguir la detección precoz de

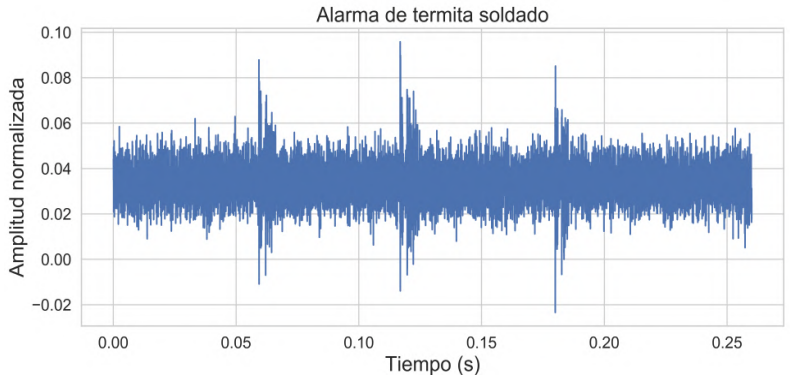

Figura 3.1: Serie temporal con una ráfaga de tres eventos acústicos por la alarma de una termita soldado.

plagas, como en el caso de algunas de nuestras investigaciones anteriores [20], [18], [22] y [19]. A raíz de estas publicaciones la multinacional Syngenta Crop Protection AG se interesó en nuestro trabajo y tuvo lugar una colaboración en forma de contrato[1]. También aplicamos una de estas técnicas, basada en estadística de orden superior, a la detección de eventos en series temporales en relación a la calidad del suministro eléctrico [21]. Particularmente esta técnica dio lugar a una patente de detección de termitas[2].

La figura 3.2 es otro ejemplo de serie temporal que utilizaremos frecuentemente en este trabajo, y que contiene datos mensuales de importaciones de Marruecos a España a través del Puerto de Algeciras en el periodo 2000-2016, específicamente para los productos del Capítulo 3 arancelario (pescado y crustáceos, moluscos y otros invertebrados acuáticos). Esta serie nos insinúa el desafío de predecir su comportamiento en el futuro. Observamos, que existe una tendencia positiva acompañada de una marcada estacionalidad de periodo anual. Además, en el año 2008 se produce una bajada significativa de las importaciones, debida a la crisis económica, que quizás podría haberse predicho por su relación con el PIB de ambos países [28]. En general, en la predicción de series temporales se debe tener en cuenta la tendencia y estacionalidad, tema que trataremos en la sección 3.3 de descomposición

[1] Análisis de emisiones de insectos con espectros de tercer orden. J. J. González de la Rosa e I. Lloret. Universidad de Cádiz. Contrato con Syngenta Crop Protection AG, art. 11/45 LRU - 68/83 LOU, 2005.

[2] Method of detecting termites using electronic and computational techniques employing multidimensional spectra. J. J. González de la Rosa e I. Lloret. Patente WO2006128932 2006.

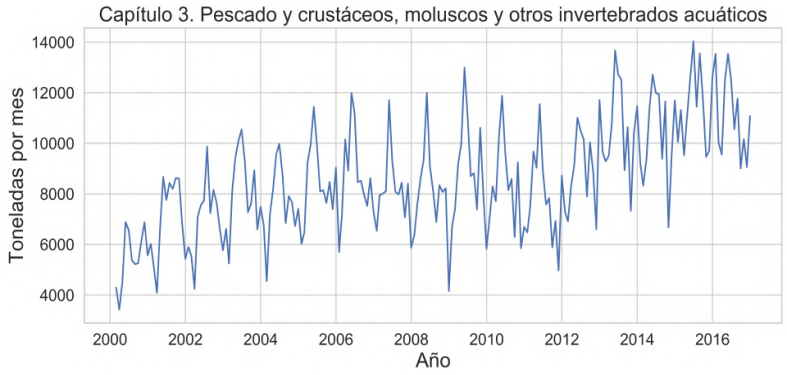

Figura 3.2: Serie temporal de importaciones de Marruecos a España a través del Puerto de Algeciras (Capítulo 3 arancelario).

de series. Fruto de nuestro trabajo con este tipo de series temporales hemos podido publicar un articulo [44] con parte de la investigación realizada.

Para el propósito de la predicción debemos ampliar la definición de serie temporal de la forma

$$\mathfrak{O} = \{\mathfrak{o}_1, \mathfrak{o}_2, \ldots, \mathfrak{o}_T, \mathfrak{o}_{T+1}, \ldots, \mathfrak{o}_{T+h}, \ldots\}, \tag{3.2}$$

donde $\mathfrak{o}_T$ es la última observación previa a la predicción de los valores $\{\mathfrak{o}_{T+1}, ..., \mathfrak{o}_{T+h}\}$, siendo h el horizonte de predicción. Se trata de una predicción múltiple de h valores. Una vez obtenida la predicción, podemos evaluar su exactitud utilizando diferentes métricas, que serán analizadas en detalle en la sección 3.5. Los métodos de predicción pueden ser muy sencillos, como utilizar la observación más reciente (lo que se denomina como método ingenuo), o muy complejos, como los basados en redes neuronales profundas.

3.2. Autocorrelación

En estadística, la correlación de dos variables aleatorias nos da una medida del grado de relación lineal entre ambas, o sea, dos variables están correlacionadas si a una variación de la primera le corresponde una variación, en la misma proporción, a la segunda. Se puede utilizar esta definición para el caso de una serie temporal que se desea comparar consigo misma, pero desplazada un determinado número de observaciones o desfase (*lag*). La definición de función de autocorrelación (*ACF; autocorrelation function*) de una serie $\mathfrak{O}$ respecto a ella misma con un lag k sería:

$$\text{ACF}(k) = \frac{\sum_{t=k+1}^{T}(\mathfrak{o}_t - \overline{\mathfrak{o}})(\mathfrak{o}_{t-k} - \overline{\mathfrak{o}})}{\sum_{t=1}^{T}(\mathfrak{o}_t - \overline{\mathfrak{o}})^2}, \tag{3.3}$$

donde T es la longitud de la serie temporal y $\overline{\mathfrak{o}}$ denota la media de sus valores.

La función de autocorrelación resulta de gran utilidad para encontrar patrones repetitivos dentro de una serie temporal. La figura 3.3 nos muestra la ACF de una serie temporal con una marcada estacionalidad anual, que en el caso de datos mensuales quedaría reflejada en el desfase 12, o sea, ACF(12).

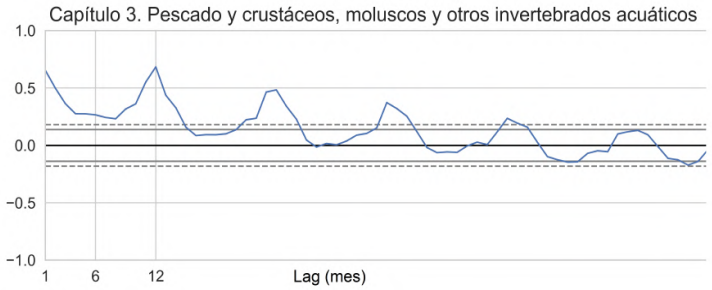

Figura 3.3: Función de autocorrelación de una serie temporal de importación de mercancías de Marruecos a través del Puerto de Algeciras. En esta serie temporal mensual, el máximo local en el lag 12 indica que existe autocorrelación anual.

Para determinar si una serie presenta estacionalidad podemos utilizar el test de V. Assimakopoulos y K. Nikolopoulos (2000) [2] con un 90 % de nivel de confianza, que está basado en la significancia de ACF(m), siendo m el número de observaciones del periodo estacional.

Cuando los valores de la AFC tienden a disminuir suavemente desde el primero, a medida que aumenta el lag, nos encontramos ante una serie en la que existe una tendencia, debido a que las observaciones cercanas en tiempo tienen valores parecidos. Si en la serie existe tendencia y estacionalidad, tendríamos una combinación de ambos efectos como se muestra en la misma figura 3.3.

3.3. Descomposición de series

Las series temporales pueden exhibir diferentes comportamientos y, con el propósito de su predicción, a menudo conviene descomponerlas en varias

componentes: tendencia, estacionalidad y residuo. La tendencia puede incluir ciclos de más de un año que suelen estar relacionados con factores económicos. El modelo de descomposición clásico se sigue utilizando mucho en la actualidad, y en él se consideran dos formas para determinar el valor de cada observación $\mathfrak{o}_k$ dependiendo de cada componente según si la descomposición es aditiva o multiplicativa:

- Aditiva: $\mathfrak{o}_k = t_k + s_k + r_k$
- Multilicativa. $\mathfrak{o}_k = t_k \cdot s_k \cdot r_k$

donde t_k, s_k y r_k representan las componentes de tendencia, estacionalidad y residuo, respectivamente, de la observación k.

Si la componente estacional permanece constante a lo largo de la serie se utiliza el modelo aditivo, en cambio, si es proporcional a la tendencia se aplica el multiplicativo. Esto último es lo habitual en las series temporales económicas, como son las utilizadas en este trabajo. También es posible convertir una serie con componentes multiplicativas en aditivas usando la función logaritmo o una transformación de tipo Box-Cox [6].

En la descomposición de una serie, lo primero que debemos obtener es la tendencia, para lo cual se calcula una la media móvil que elimine el efecto de la estacionalidad, promediando sobre un conjunto de las m observaciones que constituyen el periodo. Hay dos casos:

- Si m es impar, aplicamos una media móvil de orden m (m-**MA**). Por ejemplo, para una serie con datos diarios y una posible estacionalidad semanal ($m = 7$):

$$t_k = \frac{1}{7}(\mathfrak{o}_{k-3} + \mathfrak{o}_{k-2} + \mathfrak{o}_{k-1} + \mathfrak{o}_k + \mathfrak{o}_{k+1} + \mathfrak{o}_{k+2} + \mathfrak{o}_{k+3}).$$

- Si m es par, aplicamos una media móvil de orden m y después otra de orden 2 ($2 \times m$-**MA**). De esta forma conseguimos una media ponderada que respecto a las observaciones es simétrica. Por ejemplo, para una serie con datos trimestrales y una posible estacionalidad anual ($m = 4$):

$$t_k = \frac{1}{8}\mathfrak{o}_{k-2} + \frac{1}{4}\mathfrak{o}_{k-1} + \frac{1}{4}\mathfrak{o}_k + \frac{1}{4}\mathfrak{o}_{k+1} + \frac{1}{8}\mathfrak{o}_{k+2}.$$

Una vez obtenida la tendencia, esta se elimina, obteniendo cada valor de la nueva serie según el tipo de descomposición:

- Aditiva: $\mathfrak{o}_k - t_k$

- Multiplicativa: $\mathfrak{o}_k/t_k$

La estacionalidad se extrae partiendo la nueva serie en trozos de longitud m, y haciendo el promedio de estos. Así obtenemos el periodo estacional donde cada valor suele denominarse índice estacional. La serie s_k se construye por concatenación del periodo estacional hasta que su longitud coincida con la de la serie inicial. En cada tipo de descomposición obtenemos el residuo mediante:

- Aditiva: $r_k = \mathfrak{o}_k - t_k - s_k$
- Multiplicativa: $r_k = \mathfrak{o}_k/t_k/e_k$

Independientemente de si se ha eliminado la tendencia, se dice que una serie está ajustada estacionalmente si se le ha extraído la estacionalidad.

La figura 3.4 nos muestra un ejemplo de descomposición multiplicativa utilizando la función `seasonal_decompose` del módulo Statsmodels [64] de Python.

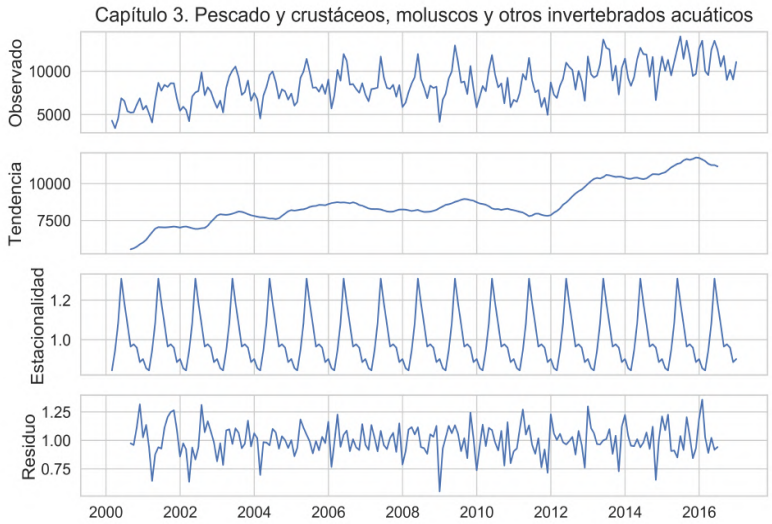

Figura 3.4: Descomposición multiplicativa de una serie temporal de importaciones de Marruecos a España a través del Puerto de Algeciras.

En la descomposición multiplicativa puede darse un problema de división por cero cuando algún valor t_k sea cero. Esto puede resolverse asegurando que la serie sea positiva mediante la adición de una cantidad antes de la descomposición, que una vez realizada la predicción se restauraría.

Otro método sencillo para obtener la tendencia es una simple recta de regresión, como veremos más adelante cuando analicemos los métodos de predicción de referencia en la sección 10.1.

3.4. Métodos ingenuos de predicción

En esta sección pretendemos explicar tres métodos sencillos de predicción (Naïve 1, Naïve S y Naïve 2) que se suelen utilizar como referencia. Estos métodos tienen la característica de ser muy intuitivos y facilitar la detección de carencias básicas en otros métodos más sofisticados. Para la comparación de los tres métodos, se muestran las predicciones en diferentes condiciones.

3.4.1. Naïve 1

Este método ingenuo se basa en la simple suposición de en el futuro se va mantener constante el último valor observado. La ecuación de predicción es:

$$\widehat{\mathfrak{o}_{T+k}} = \mathfrak{o}_T \qquad k = 1, \ldots, h, \tag{3.4}$$

donde $\widehat{\cdot}$ denota predicción.

3.4.2. Naïve S

Este método Naïve S se basa en la suposición de en el futuro se va a repetir constantemente el último periodo estacional observado. Su ecuación de predicción es:

$$\widehat{\mathfrak{o}_{T+k}} = \mathfrak{o}_{T-m+\mathrm{rem}(k,m)} \qquad k = 1, \ldots, h, \tag{3.5}$$

donde $\mathrm{rem}(k, m)$ denota el resto de la división entera de k entre m, siendo m la longitud del periodo estacional.

3.4.3. Naïve 2

El método Naïve 2 [48] requiere un ajuste previo de la estacionalidad (sección 3.3) una vez superado un test con un 90 % de nivel de confianza. Después, se efectúa la predicción de la misma forma que en el método Naïve 1, y posteriormente se restaura la estacionalidad para obtener la predicción final.

Las ecuaciones de la predicción son:

$$\begin{aligned} d_k &= \mathfrak{o}_k / e_k \qquad && k = 1 \dots T, \\ \widehat{d_{T+k}} &= d_T \qquad && k = 1 \dots h, \\ \widehat{\mathfrak{o}_{T+k}} &= \widehat{d_{T+k}} \cdot e_{T+k} \qquad && k = 1 \dots h, \end{aligned} \tag{3.6}$$

donde d_k denota cada valor de la serie desestacionalizada.

Las figuras 3.5 y 3.6 muestran ejemplos de predicciones de los métodos Naïve 1, Naïve S y Naïve 2 para una serie estacional y no estacional respectivamente.

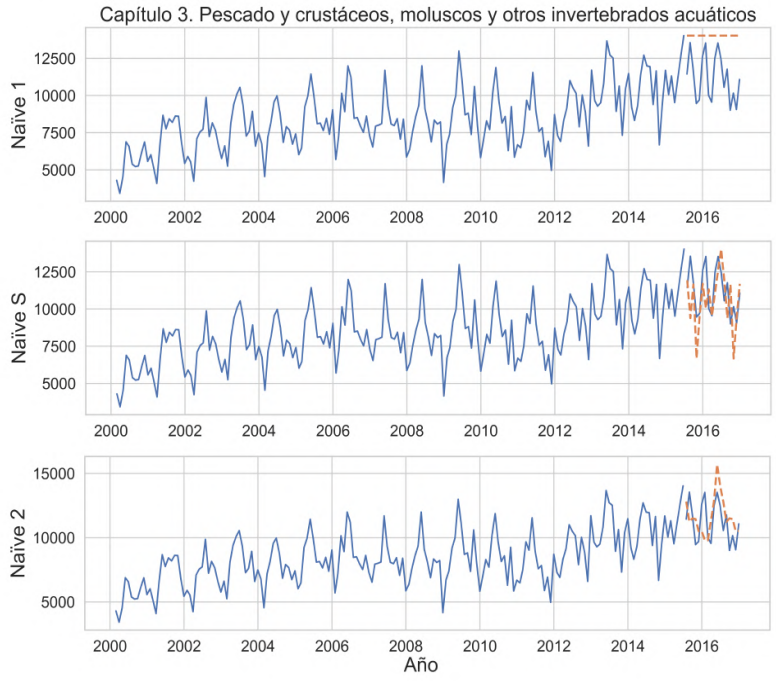

Figura 3.5: Predicción de los métodos de referencia Naïve 1, Naïve S y Naïve 2 para una serie estacional.

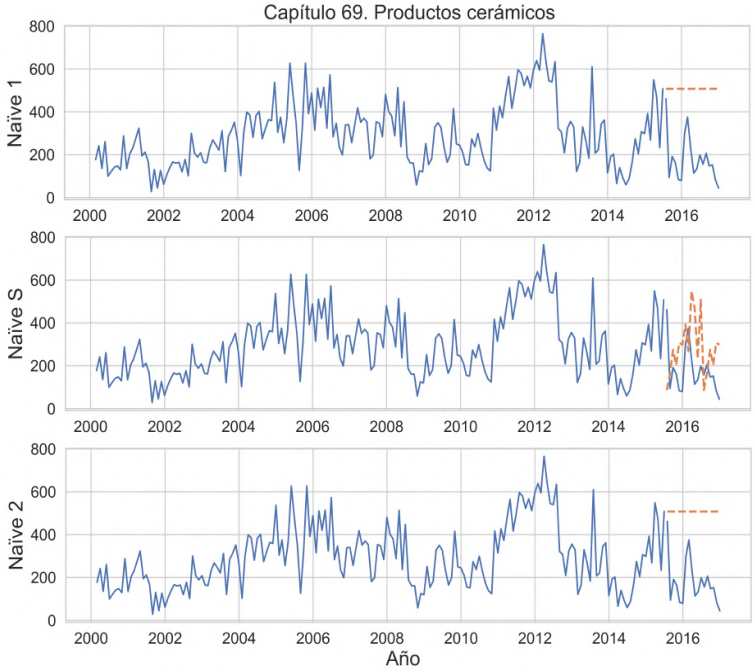

Figura 3.6: Predicción de los métodos de referencia Naïve 1, Naïve S y Naïve 2 para una serie no estacional.

3.5. Métricas de evaluación

Para comparar la exactitud de la predicción de diferentes modelos se suele dividir la serie temporal en dos partes, los datos de entrenamiento y los de test [34]. Los datos de entrenamiento se utilizan para ajustar los parámetros de cada modelo, que una vez optimizados podrán producir sus predicciones. Dado que los datos de test no intervienen en la realización de las predicciones, suponemos que cada modelo se comportará probablemente de forma similar prediciendo los datos de test que cuando se enfrente a datos nuevos. La figura 3.7 muestra esta división de los datos de la serie, donde hemos considerado un tamaño de los datos de test igual al horizonte de predicción.

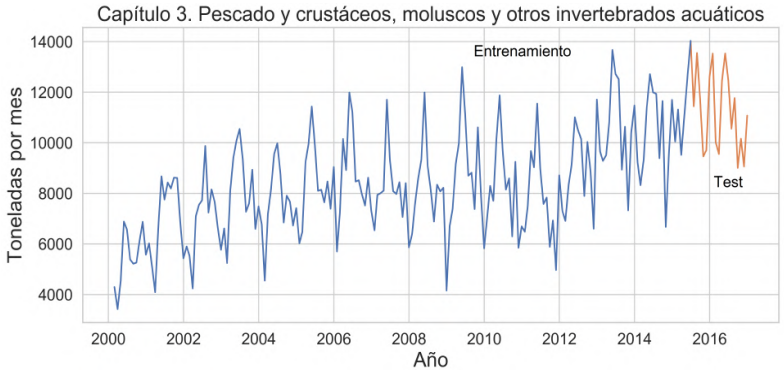

Figura 3.7: Datos de entrenamiento y test para la predicción de una serie temporal. La longitud de los datos de test es el horizonte de predicción.

Las métricas de evaluación de predicciones que vamos a describir en las siguientes subsecciones están expresadas en base a: datos de entrenamiento $\{\mathfrak{o}_1, \mathfrak{o}_2, \ldots, \mathfrak{o}_T\}$; datos de test $\{\mathfrak{o}_{T+1}, \mathfrak{o}_{T+2}, \ldots, \mathfrak{o}_{T+h}\}$, donde h es el horizonte de predicción; y datos de predicción $\{\widehat{\mathfrak{o}_{T+1}}, \widehat{\mathfrak{o}_{T+2}}, \ldots, \widehat{\mathfrak{o}_{T+h}}\}$. Estas se encuentran clasificadas según R. J. Hyndman y A. B. Koehler (2006) [36] en dependientes de la escala, basadas en errores porcentuales, relativas y escaladas.

3.5.1. Dependientes de la escala de los datos

Las métricas dependientes de la escala más comunes son:

- Error cuadrático medio (*MSE*; *mean squared error*):

$$\mathrm{MSE} = \frac{1}{h} \sum_{t=T+1}^{T+h} (\mathfrak{o}_t - \widehat{\mathfrak{o}_t})^2 \tag{3.7}$$

- Error absoluto medio (*MAE*; *mean absolute error*):

$$\text{MAE} = \frac{1}{h}\sum_{t=T+1}^{T+h} |\mathfrak{o}_t - \widehat{\mathfrak{o}}_t| \tag{3.8}$$

Aunque el MSE sea muy utilizado, como por ejemplo en la minimización por mínimos cuadrados para la obtención de la recta de regresión, el MAE es más robusto y soporta mejor los valores atípicos (*outliers*), por lo que algunos autores lo recomiendan para problemas de predicción. Por este motivo, hemos utilizado el MAE en la optimización de nuestros modelos. El único inconveniente que presenta el MAE es que, al depender de la escala, no se puede utilizar para comparar el error en la predicción de series diferentes, aunque en el caso de que todas tengan la misma escala sería muy buena elección por su sencillez y facilidad de interpretación.

3.5.2. Basadas en errores porcentuales

Para evitar la dependencia de la escala se pueden usar otro tipo de métricas, como las basadas en errores porcentuales, siendo las más utilizadas las siguientes:

- Error porcentual absoluto medio (*MAPE*; mean absolute percentage error):

$$\text{MAPE} = \frac{1}{h}\sum_{t=T+1}^{T+h} \left|\frac{\mathfrak{o}_t - \widehat{\mathfrak{o}_t}}{\mathfrak{o}_t}\right| \cdot 100\,\% \tag{3.9}$$

  Esta métrica, como otras de tipo porcentual, tiene el inconveniente de que si algún valor $\mathfrak{o}_t$ es cero da un resultado infinito o indefinido, y para valores cercanos a cero el error puede aumentar enormemente. También en series positivas se penaliza más a los errores negativos ($\mathfrak{o}_t < \widehat{\mathfrak{o}_t}$), que no tienen límite de error, en cambio los positivos tienen un límite del 100 %, lo que condujo a la utilización de las llamadas medidas simétricas (S. Makridakis [45]).

- Error porcentual absoluto medio simétrico (*sMAPE*; symmetric mean absolute percentage error):

$$\text{sMAPE} = \frac{2}{h}\sum_{t=T+1}^{T+h} \frac{|\mathfrak{o}_t - \widehat{\mathfrak{o}_t}|}{|\mathfrak{o}_t| + |\widehat{\mathfrak{o}_t}|} \cdot 100\,\% \tag{3.10}$$

Esta métrica porcentual proporciona un valor entre 0 % y 200 %. Tiene también un importante inconveniente si trabajamos con series temporales intermitentes, cuando un determinado valor observado $\mathfrak{o}_t$ y su correspondiente predicción $\widehat{\mathfrak{o}_t}$ son iguales a cero. También son problemáticos los valores de $\mathfrak{o}_t$ cercanos a cero. Una solución a estos problemas se encuentra sumando una constante al denominador [52]. Nosotros hemos adoptado una solución similar, sumando 1 a todas las series antes de la predicción, y restándolo después de calcular la medida del error. De esa forma también evitamos los problemas relacionados con el ajuste estacional multiplicativo de algunos métodos de predicción de referencia (sección 10.1).

Para comparar predicciones de series con valores positivos y mucho mayores que cero se suele utilizar la metrica MAPE por su simplicidad.

3.5.3. Relativas

Las métricas relativas, como las porcentuales, también eliminan la escala de los datos, y entre ellas se encuentran:

- Medidas del error basadas en errores relativos. Cada error relativo individual $r_t = e_t/e_t^*$ viene dado por el cociente de cada error e_t del método de predicción a evaluar, entre su correspondiente error e_t^* de un método de referencia estándar. Un ejemplo de este tipo de métrica puede ser el error absoluto relativo medio (*MRAE; mean relative absolute error*):

$$\mathrm{MRAE} = \frac{1}{h}\sum_{t=T+1}^{T+h} |r_t| = \frac{1}{h}\sum_{t=T+1}^{T+h} \left|\frac{\mathfrak{o}_t - \widehat{\mathfrak{o}_t}}{e_t^*}\right|. \tag{3.11}$$

- Medidas de error relativas a otras medidas de referencia. Son parecidas las anteriores pero dividiendo la medida del error del método a evaluar entre la de un método de referencia estándar. Un ejemplo de este tipo de métrica puede ser el MAE relativo al MAE del método Naïve 1 ($Rel_{Naïve1}MAE$):

$$\mathrm{Rel}_{\mathrm{Naïve1}}\mathrm{MAE} = \frac{\mathrm{MAE}}{\mathrm{MAE}_{\mathrm{Naïve1}}}. \tag{3.12}$$

Este tipo de métricas busca eliminar la escala de los datos comparando el error en las predicciones con el de algún método simple de referencia, pero los dos tipos sufren problemas. Las medidas de error basadas en errores relativos los sufren cuando algún error e_t^* del método de referencia es cercano a cero; y

en las medidas de error relativas a otras lo sufren cuando se pretende evaluar la precisión para un horizonte de predicción simple, no de múltiples pasos (*multi-step*), en cuyo caso el MAE relativo no tendría sentido calcularlo si se intentan comparar las predicciones de varias series, debido a la diferencia de escala para cada predicción simple.

3.5.4. Escaladas

La métricas escaladas, también eliminan la escala de los datos, pero evitando todos los problemas anteriores:

- El error escalado absoluto medio (*MASE*; *mean absolute scaled error*) fue propuesto por R. J. Hyndman [36] con el fin de hacer posible la comparación de la exactitud de las predicciones para todos los tipos de series. El MASE estacional es una métrica escalada que divide cada error absoluto individual de la predicción entre el error absoluto medio (MAE) del método Naïve S (sección 3.4.2) sobre los datos de entrenamiento para predicciones de un paso; estando siempre definido excepto en el caso de que todos los datos de entrenamiento sean iguales a cero. Su interpretación es muy sencilla, valores del MASE mayores que uno indican que las predicciones son peores, en promedio, que las predicciones de un paso del método Naïve S sobre los datos de entrenamiento. Se define formalmente como:

$$\begin{aligned} \text{MASE} &= \frac{1}{h}\frac{\sum_{t=T+1}^{T+h} |\mathfrak{o}_t - \widehat{\mathfrak{o}}_t|}{\text{MAE}(\text{NaïveS}_{train})} \\ &= \frac{1}{h}\frac{\sum_{t=T+1}^{T+h} |\mathfrak{o}_t - \widehat{\mathfrak{o}}_t|}{\frac{1}{T-m}\sum_{t=m+1}^{T} |\mathfrak{o}_t - \mathfrak{o}_{t-m}|} \end{aligned} \tag{3.13}$$

  donde m es la longitud del periodo estacional considerado.

  Debemos tener en cuenta que el método Naïve S efectúa predicciones de un solo paso (*one-step*), lo cual significa que cada predicción $\widehat{\mathfrak{o}}_t$ es igual a $\mathfrak{o}_{t-m}$. En el caso de tratar con series no estacionales, simplemente consideraríamos $m = 1$, convirtiendo el MASE estacional en MASE.

3.5.5. Media ponderada global

La media ponderada global (*OWA*; *overall weighted average*) se ha utilizado recientemente en la Competición M4[3] de predicción para obtener, a

[3] `https://mofc.unic.ac.cy/m4/`

partir de las métricas sMAPE y MASE, un único valor que resuma estos dos errores y facilite la clasificación de los diferentes modelos. Un valor menor que 1 en esta media ponderada significa que el error del modelo evaluado es menor que el error del método de referencia Naïve 2 (sección 3.4.3). El OWA se calcula para cada modelo de predicción como se indica a continuación:

1. Se computa el sMAPE y MASE para cada serie individualmente.
2. La media de los sMAPE y la media de los MASE son obtenidas sobre todas las series.
3. La media del sMAPE y la media del MASE se hacen relativas a sus correspondientes, media del sMAPE y media del MASE, de la predicción del método Naïve 2.
4. Finalmente, se calcula la media aritmética del sMAPE relativo y el MASE relativo.

$$\text{OWA} = \left(\frac{\overline{\text{sMAPE}}}{\overline{\text{sMAPE}}_{\text{Naïve2}}} + \frac{\overline{\text{MASE}}}{\overline{\text{MASE}}_{\text{Naïve2})}} \right) /2 \tag{3.14}$$

Capítulo 4

Métodos de suavizado y estándares de predicción

Comenzaremos este capítulo describiendo tres métodos de suavizado que tienen propiedades bien conocidas y pueden ayudarnos a mejorar otros más complejos. Seguidamente, describiremos el método Theta, ganador de la Competición M3[1]. Finalmente, introduciremos los métodos basados en modelos estadísticos ETS y ARIMA, muy utilizados en la actualidad, en los que es posible seleccionar automáticamente el mejor modelo mediante un criterio de información.

4.1. Métodos de suavizado exponencial

El suavizado exponencial fue propuesto inicialmente por R. G. Brown (1956) [8] y posteriormente expandido por C. C. Holt (1957) [31] y P. R. Winters (1960) [71]. Los tres métodos que describimos en esta sección pueden utilizarse opcionalmente con ajuste previo de estacionalidad, por lo que no es necesario considerarla en el propio método.

4.1.1. Simple (SES)

El método de suavizado exponencial simple (*SES*; *simple exponential smoothing*) basa las predicciones en una media ponderada de las observaciones previas, dándole un mayor peso a las más cercanas a la predicción, y decayendo exponencialmente a medida que están más alejadas temporalmen-

[1] `https://mofc.unic.ac.cy/m3/`

te:

$$\widehat{\mathfrak{o}_{t+1}} = \alpha\mathfrak{o}_t + \alpha(1-\alpha)\mathfrak{o}_{t-1} + \alpha(1-\alpha)^2\mathfrak{o}_{t-2} + \ldots, \tag{4.1}$$

donde $0 \leq \alpha \leq 1$ es el parámetro que controla el suavizado. Para un valor alto de α el decaimiento es más rápido, dándole menos peso a las observaciones mas lejanas; en cambio si es pequeño, el decaimiento es lento y las observaciones más alejadas influyen en mayor medida en la predicción.

Otra forma de expresar la ec. 4.1 de predicción es considerando las distintas componentes que influyen en ella:

$$\begin{aligned} \text{Predicción} \quad & \widehat{\mathfrak{o}_{t+1}} = l_t \\ \text{Componente de nivel} \quad & l_t = \alpha\mathfrak{o}_t + (1-\alpha)l_{t-1}, \end{aligned} \tag{4.2}$$

donde l_t denota el nivel (*level*) debido a las observaciones previas, que es la única componente en este tipo de suavizado. Existen además, las componentes de tendencia y de estacionalidad, que se utilizan en otros métodos.

El ajuste de los parámetros de este método se basa en encontrar los valores α y l_0, que minimicen el error en predicciones de un paso, considerando exclusivamente los datos de entrenamiento:

$$\sum_{k=1}^{T} (\mathfrak{o}_k - \widehat{\mathfrak{o}_k})^2. \tag{4.3}$$

La minimización de este error conduce a un problema de optimización no lineal que debe resolverse mediante el uso de algún método numérico.

Las predicciones a partir de la observación T se pueden obtener iterativamente con la ecuación 4.2 considerando cada predicción $\widehat{\mathfrak{o}_{t+1}}$ como futuro valor observado $\mathfrak{o}_t$, dando lugar a predicciones constantes:

$$\widehat{\mathfrak{o}_{T+k}} = l_T \qquad k = 1, 2, \ldots, h. \tag{4.4}$$

4.1.2. De tendencia lineal de Holt

Basado en el anterior SES, este método tiene en cuenta además la componente relativa a la tendencia de la serie, de forma que la predicción en el instante $t+1$ depende del nivel y tendencia en el instante t:

$$\begin{aligned} \text{Predicción} \quad & \widehat{\mathfrak{o}_{t+1}} = l_t + b_t \\ \text{Componente de nivel} \quad & l_t = \alpha\mathfrak{o}_t + (1-\alpha)(l_{t-1} + b_{t-1}) \\ \text{Componente de tendencia} \quad & b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}, \end{aligned} \tag{4.5}$$

donde b_t denota la tendencia (pendiente) debida a las observaciones previas, siendo $0 \leq \beta \leq 1$.

Una vez ajustados los parámetros que minimizan el error de predicción de un salto, se pueden generar las predicciones mediante la ecuación 4.5 considerando cada predicción como una nueva observación, dando lugar a una recta de pendiente b_T:

$$\widehat{\mathfrak{o}_{T+k}} = l_T + kb_T \qquad k = 1, 2, \ldots, h. \tag{4.6}$$

4.1.3. De tendencia amortiguada (Damped)

Partiendo del método Holt, es posible usar la tendencia (pendiente) atenuada por un factor ϕ, donde $0 < \phi < 1$, lo cual conduce a las ecuaciones del método:

$$\begin{aligned} \text{Predicción} \quad & \widehat{\mathfrak{o}_{t+1}} = l_t + \phi b_t \\ \text{Componente de nivel} \quad & l_t = \alpha \mathfrak{o}_t + (1-\alpha)(l_{t-1} + \phi b_{t-1}) \\ \text{Componente de tendencia} \quad & b_t = \beta(l_t - l_{t-1}) + (1-\beta)\phi b_{t-1}. \end{aligned} \tag{4.7}$$

Tras el ajuste de los parámetros, se pueden generar las predicciones de forma similar al método Holt, dando lugar a una curva en la que la pendiente se va amortiguando a medida que se producen las nuevas predicciones:

$$\widehat{\mathfrak{o}_{T+k}} = l_T + (\phi + \phi^2 + \cdots + \phi^k) b_T \qquad k = 1, 2, \ldots, h. \tag{4.8}$$

Las figuras 4.1 y 4.2 muestran ejemplos de predicción de los métodos SES, Holt y Damped para una serie estacional (previamente ajustada) y otra no estacional, respectivamente. Visualmente es difícil apreciar el amortiguamiento en la predicción del modelo Damped, pero numéricamente sí se puede comprobar.

4.2. Método Theta

El método Theta fue el ganador de la Competición M3 de predicción. Sus autores, V. Assimakopoulos y K. Nikolopoulos (2020) [2], aproximaron cada serie temporal con varias líneas que se extrapolaban de forma separada y combinaban para formar la predicción. En este método, cada línea de aproximación parte de la serie original $\mathfrak{O}$, modificando su curvatura mediante el

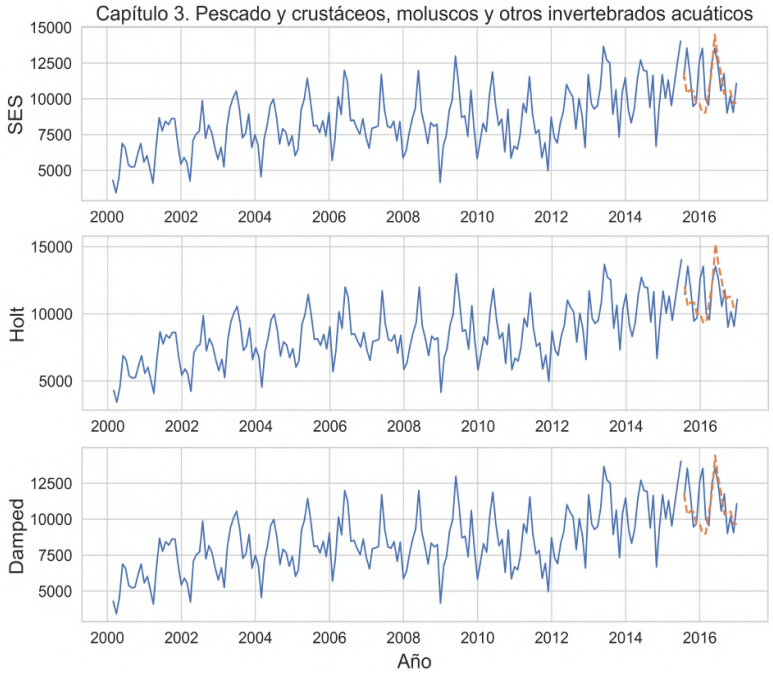

Figura 4.1: Predicción de los métodos de referencia SES, Holt y Damped para una serie estacional (con ajuste de estacionalidad).

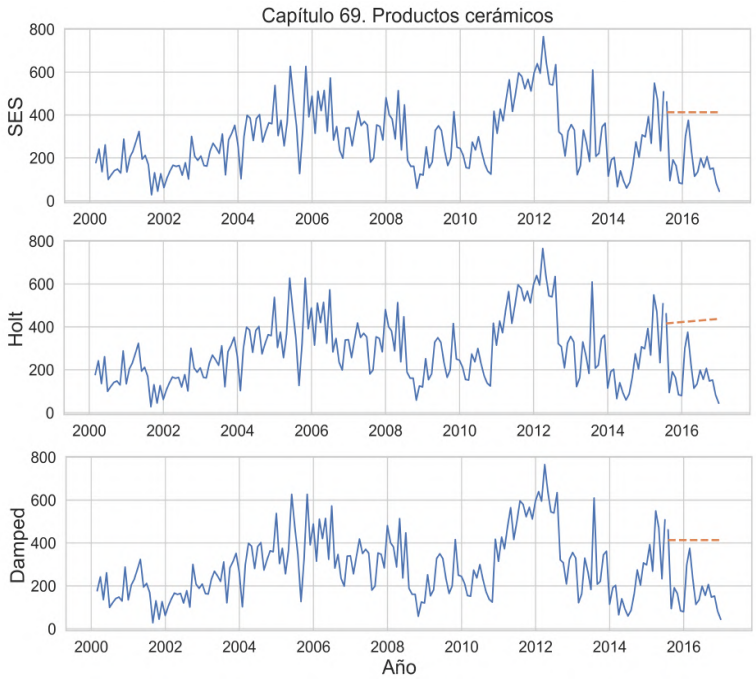

Figura 4.2: Predicción de los métodos de referencia SES, Holt y Damped para una serie no estacional.

factor Theta ($\Theta \geq 0$) aplicado a las segundas diferencias, para dar lugar a una serie aproximada N, tal que,

$$n''_t(\Theta) = \Theta \cdot (\mathfrak{o}_t - 2\mathfrak{o}_{t-1} + \mathfrak{o}_{t-2}). \tag{4.9}$$

Este tipo de transformación mantiene en la nueva serie el valor medio y la pendiente de la serie original.

Concretamente, para la Competición M3 se utilizaron dos líneas de aproximación, junto con el siguiente método de predicción:

1. Comprobación de la estacionalidad mediante un test estadístico propuesto por los autores (sección 3.2).
2. Ajuste de la estacionalidad multiplicativa.
3. Aproximación con dos líneas:
   - $\Theta = 0$; da lugar a una recta de regresión, para la predicción a largo plazo.
   - $\Theta = 2$; duplica las curvaturas locales, para predicción a corto plazo.
4. Extrapolación lineal de la recta de regresión, y mediante el método SES para la segunda línea.
5. Combinación por media aritmética de ambas predicciones.
6. Restablecimiento de la estacionalidad.

4.3. Modelos ETS

En los métodos de suavizado exponencial es posible considerar opcionalmente la componente estacional (aditiva o multiplicativa) y la tendencia (aditiva, aditiva amortiguada, multiplicativa o multiplicativa amortiguada), dando lugar a un método diferente en cada caso. La elección del método se hace normalmente *ad hoc*, en base a los componentes que se reconozcan en las series. En esta sección introducimos los modelos estadísticos subyacentes a estos métodos, que permiten generar las mismas predicciones, además de posibilitar la selección automática del mejor modelo según un criterio de información. Un modelo estadístico es un proceso estocástico generador de datos del que se puede obtener la distribución probabilidad de la predicción. El trabajo de R. J. Hyndman et al. [32] recoje los principales avances en este

sentido. Para cada método de suavizado exponencial se pueden definir dos modelos, según si el error se considera aditivo o multiplicativo. Cada modelo se define mediante varias ecuaciones derivadas de su correspondiente método: la ecuación que describe los datos observados en función de los componentes, y las que describen cómo cambian los componentes (estado) en cada paso de tiempo.

Un modelo ETS (*Error*, *Trend*, *Seasonal*) viene determinado por el tipo de error, la tendencia y la estacionalidad. En el caso del método SES, partiendo de las ecuaciones ya conocidas,

$$\begin{aligned}\widehat{\mathfrak{o}_{t+1}} &= l_t \\ l_t &= \alpha \mathfrak{o}_t + (1-\alpha) l_{t-1},\end{aligned}$$

si consideramos el error $e_t = \mathfrak{o}_t - \widehat{\mathfrak{o}_t} = \mathfrak{o}_t - l_{t-1}$, podemos reescribir las ecuaciones como:

$$\begin{aligned}\text{Observación} \quad & \mathfrak{o}_t = l_{t-1} + e_t \\ \text{Componente de nivel} \quad & l_t = l_{t-1} + \alpha e_t,\end{aligned} \tag{4.10}$$

donde los errores e_t son independientes y siguen una distribución normal $N(0, \sigma^2)$. De esta forma, queda completamente definido el modelo estadístico de error aditivo subyacente al método SES. Si considerásemos el error relativo,

$$e_t = \frac{\mathfrak{o}_t - \widehat{\mathfrak{o}_t}}{\widehat{\mathfrak{o}_t}} = \frac{\mathfrak{o}_t - l_{t-1}}{l_{t-1}},$$

entonces las ecuaciones del modelo SES con error multiplicativo se podrían escribir como,

$$\begin{aligned}\text{Observación} \quad & \mathfrak{o}_t = l_{t-1}(1 + e_t) \\ \text{Componente de nivel} \quad & l_t = l_{t-1}(1 + \alpha e_t).\end{aligned} \tag{4.11}$$

El entrenamiento de estos modelos se puede hacer minimizando la suma de los errores al cuadrado de las predicciones de un salto con los datos de entrenamiento, o ajustando los parámetros de forma que se maximice la verosimilitud de los datos generados por el modelo para la distribución de probabilidad de los errores.

La predicciones se obtienen usando las ecuaciones del modelo de forma iterativa, considerando que no existe error ($e_t = 0$) para siguientes predicciones.

Otros modelos ETS pueden obtenerse igualmente para los dos tipos de errores y cada método de suavizado exponencial.

4.3.1. Selección automática del modelo

Una de las grandes ventajas de este modelado estadístico es que, una vez optimizados los modelos mediante máxima verosimulitud, se puede utilizar un criterio de información para la selección del mejor, lo que permite hacerlo automáticamente. Los tres criterios utilizados por R. J. Hyndman et al. (2008) [35] son:

- Criterio de información de Akaike (*AIC*; *Akaike information criterion*). Se define como:

$$\text{AIC} = -2log(L) + 2k, \tag{4.12}$$

  donde L es la verosimilitud del modelo y k el número de parámetros a ajustar. El criterio penaliza los modelos con mayor número de parámetros. El valor más bajo esta asociado al mejor modelo.

- AIC corregido (AICc). Evita la tendencia a la selección de modelos con mayor número de parámetros para *datasets* de entrenamiento pequeños que provoca un sobreajuste (*overfitting*):

$$\text{AICc} = \text{AIC}\frac{2k(k+1)}{T-k-1}, \tag{4.13}$$

  donde T es la longitud de la serie temporal previa a la predicción.

- Criterio de información bayesiano de Schwarz (*BIC*; *bayesian information criterion*). Es parecido a AIC pero penaliza un poco más el mayor número de parámetros a ajustar:

$$\text{BIC} = \text{AIC} + k[log(T) - 2]. \tag{4.14}$$

4.4. Modelos ARIMA

ARIMA es un modelo estadístico utilizado para predicción de series temporales que combina la autorregresión (*AR*; *autoregressive*), con la diferenciación/integración (*I*; *integrated*) y media móvil (*MA*; *moving average*). Este tipo de modelo admite infinitas posibilidades, por lo que G. E. P. Box and G. M. Jenkins (1970) [7] crearon un método de modelado iterativo que constaba de tres etapas: selección del modelo, estimación de sus parámetros y comprobación del modelo. Este método ha sido modificado a lo largo de los años siendo ampliamente utilizado hoy en día el algoritmo Hyndman-Khandakar (2008) [32] que automatiza este trabajo.

4.4.1. Diferenciación

La diferenciación se utiliza para convertir una serie temporal en estacionaria, o sea, que sus propiedades estadísticas no dependan del tiempo en que se observen. Por ejemplo, una serie con tendencia o estacionalidad es no estacionaria.

Para convertir una serie con tendencia en una serie estacionaria, debemos obtener las diferencias entre cada par de valores consecutivos, dando lugar una nueva serie donde cada observación seria:

$$\mathfrak{o}'_t = \mathfrak{o}_t - \mathfrak{o}_{t-1}. \tag{4.15}$$

Si la serie aún continuara siendo estacionaria, se puede tomar una segunda diferencia, pero no se recomiendan más de dos, con el fin de que el modelo sea interpretable.

Cuando la no estacionariedad se debe a que la serie es estacional, entonces los primero que se debe hacer es aplicar una diferencia estacional de la forma

$$\mathfrak{o}'_t = \mathfrak{o}_t - \mathfrak{o}_{t-m}, \tag{4.16}$$

donde m es la longitud en observaciones del periodo estacional. Si la serie continua siendo no estacionaria, entonces habría que decidir entre aplicar una nueva diferenciación estacional u ordinaria.

Existen diferentes tests para comprobar si una serie es no estacionaria, incluso para series estacionales. También se puede comprobar esta condición de forma visual, mediante la gráfica de la serie y de las funciones de autocorrelación y autocorrelación parcial, aunque esto se escapa fuera del contexto de nuestro trabajo.

Para una mayor sencillez en la explicación de los modelos ARIMA vamos a considerar las series no estacionales, aunque en realidad, tanto la ecuación del modelo como los algoritmos de selección automática la tienen en cuenta.

4.4.2. Autorregresión

Un modelo autorregresivo AR(p) relaciona futuros valores de una serie temporal con valores previos, y se define como

$$\mathfrak{o}_t = \phi_0 + \phi_1\mathfrak{o}_{t-1} + \phi_2\mathfrak{o}_{t-2} + \cdots + \phi_p\mathfrak{o}_{t-p} + e_t, \tag{4.17}$$

donde p es el orden del modelo, $\phi_0 \dots \phi_p$ son sus parámetros y e_t es el término de error, no correlacionado con los anteriores (ruido blanco).

4.4.3. Media móvil

Un modelo de media móvil MA(q) relaciona futuros valores de una serie temporal con los errores de las predicciones anteriores, y se define como

$$\mathfrak{o}_t = \theta_0 + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q} + e_t, \tag{4.18}$$

donde q es el orden del modelo, $\theta_0 \dots \theta_p$ son sus parámetros y e_t es el término de error (ruido blanco).

4.4.4. ARIMA

Un modelo ARIMA(p, d, q) de predicción de una serie temporal combina la diferenciación de orden d con la autoregresión de las p diferencias anteriores y la media móvil de los q errores anteriores de predicción, quedando definido mediante la ecuación

$$\mathfrak{o}'_t = c + \sum_{i=1}^{p} \phi_i \mathfrak{o}'_{t-i} + \sum_{i=1}^{q} \theta_i e_{t-i} + e_t, \tag{4.19}$$

donde c, ϕ_i y θ_i son los parámetros del modelo, que deben estimarse con los datos de entrenamiento. La estimación puede hacerse minimizando los errores de predicción de un paso, o mediante el método de máxima verosimilitud, que busca los valores de los parámetros que maximizan la probabilidad de que el modelo prediga los datos observados.

Para la obtención de predicciones, con el modelo previamente ajustado, se deben deshacer las diferencias y dejar el término $\mathfrak{o}_t$ a la izquierda del igual. A continuación, para $t = T + 1$ se obtiene la primera predicción, y así sucesivamente los siguientes valores de t, siempre teniendo en cuenta que las predicciones se consideran futuras observaciones y los errores de predicción se asumen cero para futuros términos de error.

Para una serie con periodo estacional m se utilizaría un modelo ampliado ARIMA$(p, d, q)(P, D, Q)_m$, que tendría además la diferenciación propia de la estacionalidad, así como su componente autorregresiva y de media móvil. Se omiten los detalles por simplicidad.

4.4.5. Selección automática del modelo

Para la selección automática del modelo ARIMA(q, d, q) se utiliza el algoritmo iterativo de Hyndman-Khandakar (2008) [35] que consta de los siguientes pasos:

1. Determinar el número d de diferencias a aplicar sobre la serie original, para lo cual se utiliza el test Kwiatkowski–Phillips–Schmidt–Shin (KPSS) [41] hasta obtener una serie estacionaria.

2. A partir de la serie diferenciada, se prueban diferentes valores de p y q. Se elige el mejor modelo, según el valor más bajo del AICc (ecuación 4.13) previa optimización de cada modelo.

3. Se consideran pequeñas variaciones en (p, q) del modelo seleccionado y también se prueba con un valor del parámetro $c = 0$. De estas pruebas, se selecciona el modelo con menor AICc.

Este algoritmo también puede opcionalmente tratar la estacionalidad, aunque la hemos omitido por simplicidad. En cualquier caso, siempre es posible ajustar la estacionalidad previamente a la utilización de este tipo de algoritmos.

Capítulo 5

Predicción basada en la búsqueda de coincidencia

5.1. Introducción

Una serie temporal puede ser considerada como una secuencia de valores obtenidos mediante el muestreo de una función en el dominio del tiempo. La transformada discreta de Fourier transforma una secuencia en dominio del tiempo a otra equivalente en el dominio de la frecuencia, mediante la cual obtenemos las diferentes frecuencias que componen la serie temporal. Esto puede resultar muy interesante desde el punto de vista de la predicción, al poderse detectar las frecuencias principales y reproducir su comportamiento en el futuro. El problema reside en que esta descomposición parte de la suposición de que la serie inicial completa se repite periódicamente en el tiempo, suponiendo un grave inconveniente en la predicción. Cuando partimos de una serie temporal para realizar la predicción de futuros valores, no podemos considerar que las frecuencias contenidas lo hagan en un número de ciclos exactos, porque entonces no tendría sentido predecir; la predicción sería la repetición de la misma serie.

La series temporales pueden contener varias fuentes periódicas, debidas a los periodos estacionales, y otros de longitud superior a un año como pueden ser los de tipo económico. En ese caso, una descomposición de la serie en sus diferentes sinusoidales a distintas frecuencias sería ideal para la predicción. Para predecir el futuro de una serie basándonos en la periodicidad de las series contenidas debemos, en primer lugar, encontrar la sinusoidal que mejor se adapte a la serie (mayor similitud o coincidencia), con la idea de que la extensión temporal de esta se corresponda con los futuros valores de la serie. La diferencia entre la serie original y la sinusoidal con mayor coincidencia

daría lugar a una serie residuo, a la cual se le podría repetir este proceso de forma iterativa. A este algoritmo voraz, podemos establecerle como condición de terminación alguna basada en el residuo, o simplemente repetir el bucle un número de veces con el fin de encontrar las sinusoidales más importantes. El residuo final se podría tratar como otra serie de entrada para un algoritmo de predicción, de forma que al final se puedan sumar, por un parte la predicción de la parte periódica, más la de la parte residual.

5.2. Algoritmo *Matching Pursuit*

La idea de la búsqueda de coincidencia (*matching pursuit*) fue estudiada formalmente por S. G. Mallat y Z. Zhang (1993) [49]. Según estos autores, puede existir una gran cantidad de patrones contenidos en una serie temporal y la precisión en su caracterización motiva la utilización de diccionarios redundantes de series temporales atómicas con los que descomponerlas como una combinación líneal de estos átomos, evitando que esta información se diluya por combinación de series menos parecidas. Como ya hemos comentado, la transformada de Fourier omite formas ondulatorias que podrían detectarse si las incluimos en un diccionario suficientemente amplio.

Considerando una serie temporal como un vector, donde cada componente del vector es un valor de la serie, el algoritmo propuesto por Mallat y Zhang parte de un espacio de Hilbert H, y un diccionario de vectores $D = (g_\gamma)_{\gamma\in\Gamma}$, tales que $\|g_\gamma\| = 1$, que además es completo, o sea, que el espacio vectorial generado por el diccionario es el propio H. Siendo $f \in H$ el vector a aproximar, se desea calcular una expansión de f sobre los vectores seleccionados de D con la idea de extraer la estructura interna de la función. Esto se consigue con aproximaciones sucesivas de f mediante proyecciones ortogonales sobre elementos de D. Inicialmente se escoge $g_{\gamma_1} \in D$ que descompone el vector f en

$$f = \langle f, g_{\gamma_1}\rangle g_{\gamma_1} + R_{f_1}, \tag{5.1}$$

donde R_{f_1} es el vector residual después de aproximar f en la dirección de g_{γ_1}. Para minimizar $\|R_{f_1}\|$ se debe encontrar $g_{\gamma_1} \in D$ tal que $|\langle f, g_{\gamma_1}\rangle|$ sea máximo. Si se repite n veces estos pasos, tomando cada residuo como el nuevo vector (serie) a descomponer, se obtiene una descomposición final de f de la forma:

$$\begin{aligned} f &= \langle f_1, g_{\gamma_1}\rangle g_{\gamma_1} + \langle f_2, g_{\gamma_2}\rangle g_{\gamma_2} + \cdots + \langle f_n, g_{\gamma_n}\rangle g_{\gamma_n} + R_{f_n} \\ &= \sum_{i=1}^{n} a_i g_{\gamma_i} + R_{f_n}, \end{aligned} \tag{5.2}$$

donde a_i denota $\langle f, g_{\gamma_i} \rangle$. Este algoritmo es convergente, con la norma de los residuos decayendo en cada iteración. Como hemos indicado anteriormente, la condición de terminación del algoritmo la podemos fijar de forma arbitraria.

```
serie_f = train.reshape(-1, 1) # Serie a aproximar (long.'serie_lon')
serie_integral = np.zeros(serie_lon) # Serie aproximaciones acumuladas

# BUCLE principal: se repite un número determinado de iteraciones,
#   o en función de otra condición basada, p.ej., en el residuo.
for k in range(0, n_iteraciones):

    # MAPA de CALOR ('mapa_calor[i, j]') de productos escalares
    #  <'serie_f', 'serie_g'>, donde cada 'serie_g', perteneciente al
    #  diccionario, es una sinusoidal con frecuencia 'i' y desfase 'j'
    for j in range(n_desfases): # Se repite para cada desfase 'j'
        # Cada fila de la matriz 'w_t' contiene los valores de w*t
        # de una frecuencia angular w para diferentes valores de t.
        matriz_series_g = np.cos(w_t + desfases[j])
        # Se normaliza cada serie contenida en la 'matriz_series_g'
        matriz_series_g = normalizar(matriz_series_g)
        # La columna 'j' contiene cada valor <'serie_f', 'serie_g'>
        mapa_calor[:, j] = np.matmul(matriz_series_g, serie_f)

    # Búsqueda de la 'serie_g' del diccionario más parecida a 'serie':
    #   donde 'pos[0]' es la frecuencia y 'pos[1]' el desfase
    pos = np.unravel_index(abs(mapa_calor).argmax(), mapa_calor.shape)
    serie_g = normalizar(np.cos(w_t[pos[0],:]+desfases[pos[1]]))

    # Producto escalar <'serie_f', 'serie_g'>
    producto_escalar = mapa_calor[pos[0], pos[1]]

    # Acumulación de la aproximación a la 'serie_integral'
    serie_integral = serie_integral + producto_escalar * serie_g

    # Actualización de 'serie_f' con el residuo
    serie_f = serie_f - producto_escalar * serie_g.reshape(-1, 1)
```

Figura 5.1: Núcleo del algoritmo *Matching Pursuit*. Ver figura 5.2.

El núcleo de nuestra implementación en Phyton de este algoritmo voraz se muestra en las figuras 5.1 y 5.2. Para la generación del diccionario hemos considerado experimentalmente $16T$ series sinusoidales con las frecuencias espaciadas uniformemente en el intervalo de cero a la frecuencia de Nyquist, siendo T la longitud de la serie anterior a la predicción; y para cada una de estas serie hemos considerado 180 desfases de un grado (desde 0^{o} a 179^{o}). También hemos fijado experimentalmente en 15 el número de iteraciones de nuestro algoritmo Matching Pursuit.

El conjunto de productos escalares entre cada serie g del diccionario y la

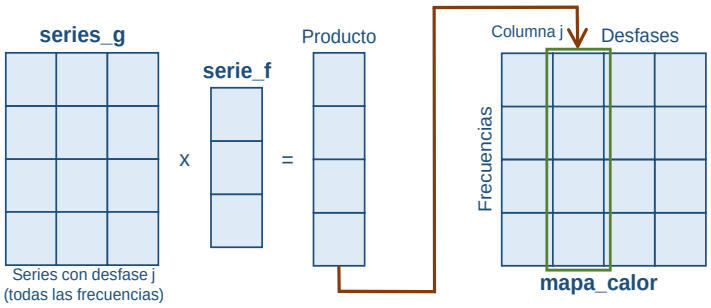

Figura 5.2: Obtención del mapa de calor en el algoritmo *Matching Pursuit*. Para un desfase 'j', la columna 'j' del mapa contiene los productos escalares de cada 'serie_g', correspondiente a cada frecuencia, y la 'serie_f'.

serie f se ha almacenado utilizado un mapa de calor, donde el número de fila corresponde a una frecuencia y el de columna al desfase de la serie g. En cada iteración del algoritmo, se escoge la serie g con mayor coincidencia con la serie f a partir del máximo en valor absoluto del mapa de calor. La posición de este valor nos determina la frecuencia y desfase de la serie g.

5.3. Ejemplos de predicción

En las figuras 5.3 y 5.4 se muestran dos ejemplos con los mapas de calor y las predicciones en las tres primeras iteraciones, así como la predicción final tras el total de iteraciones. En los mapas de calor se hace constar la frecuencia y desfase de la serie seleccionada mediante una leyenda indicativa.

En general las predicciones son aceptables y capturan perfectamente las periodicidades, aunque en la tendencia a veces se cometen grandes errores por la impredecibilidad del futuro. En la sección 5.5 planteamos algunas posibles mejoras al método de predicción.

5.4. Resultados

Los resultados obtenidos, tanto con datos de validación como de test, se muestran en el capítulo 11, donde se comparan con los resultados de los métodos de referencia, estándares de predicción y otros modelos basados en redes neuronales profundas desarrollados en este trabajo. Globalmente no son buenos resultados, pero para datos diarios el comportamiento es excelente.

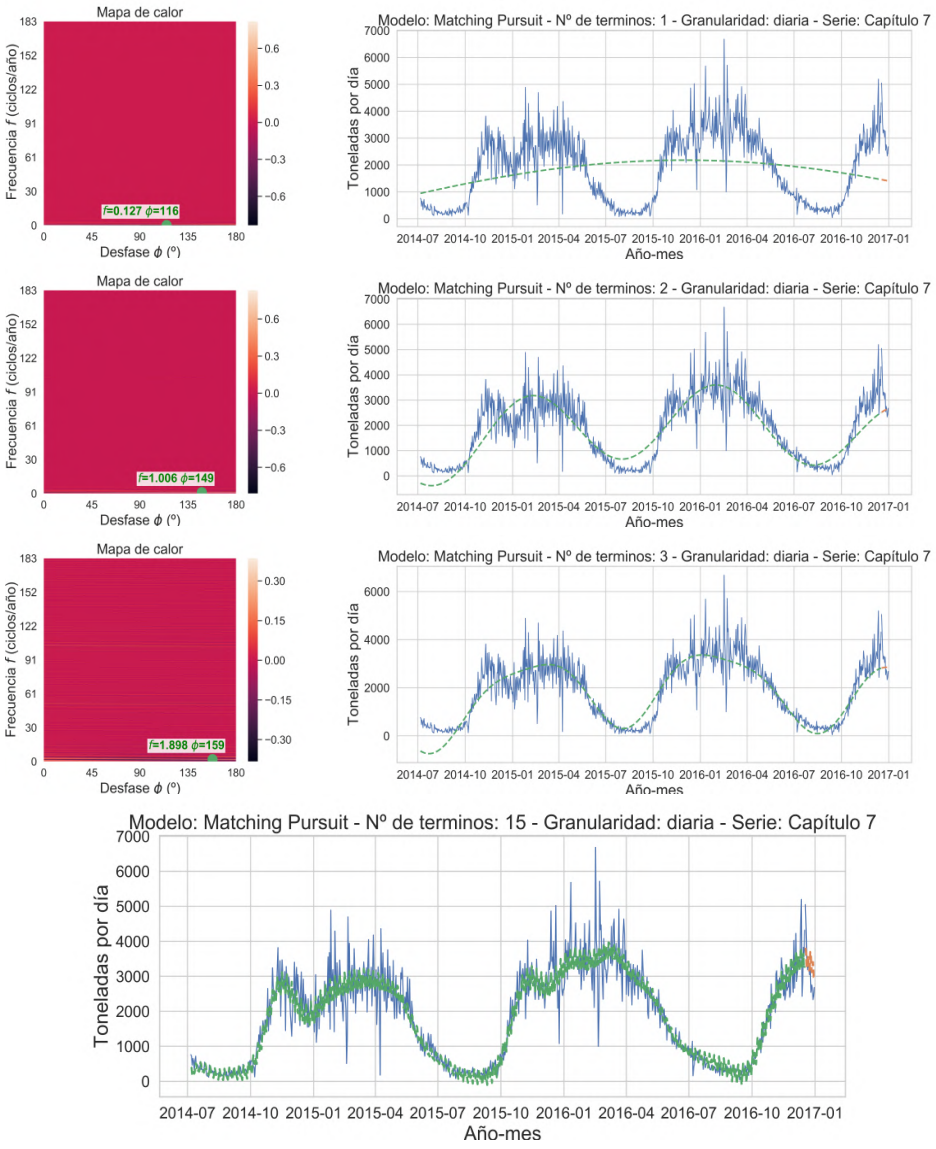

Figura 5.3: Predicción del algoritmo *Matching Pursuit* de una serie temporal de granularidad diaria. La primera serie g seleccionada del diccionario tiene frecuencia 0,127 ciclos/año y un desfase de 116º. Es una baja frecuencia que intenta capturar la característica principal de la serie f. Las siguientes series g seleccionadas tiene unas frecuencias aproximadas de 1 y 2 ciclos/año respectivamente, que sumadas a la anterior ya aproximan considerablemente bien la serie f. Después de las 15 iteraciones se muestra la aproximación y predicción finales. Esta es una serie con una gran periodicidad anual y semanal capturada en la predicción.

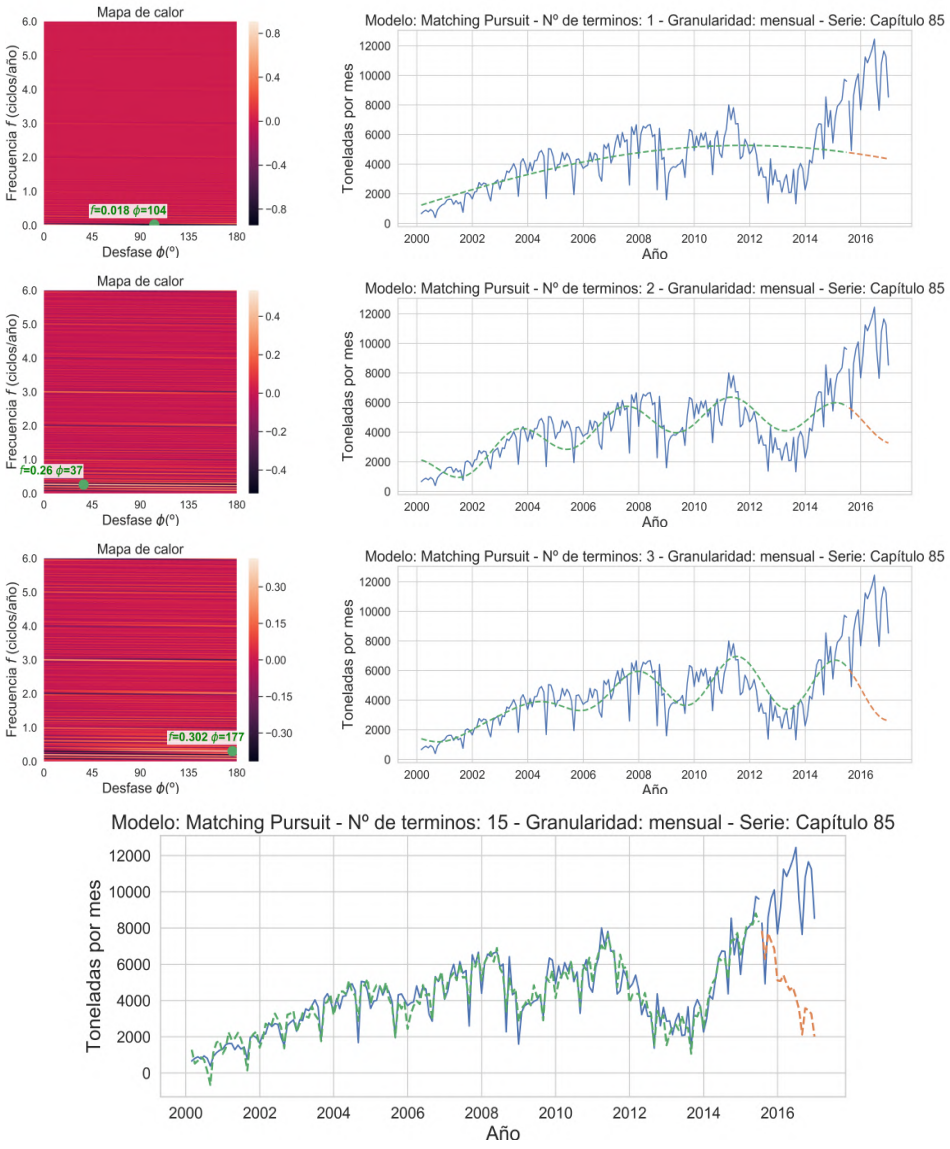

Figura 5.4: Predicción del algoritmo *Matching Pursuit* de una serie temporal de granularidad mensual. La primera serie g seleccionada del diccionario tiene frecuencia 0,018 ciclos/año y un desfase de 104º. Es una baja frecuencia que intenta capturar la tendencia de la serie f. Las siguientes series g seleccionadas tienen unas frecuencias de 0,26 y 0,302 ciclos/año respectivamente, que capturan el comportamiento periódico más relevante. Después de las 15 iteraciones se muestra la aproximación final y una predicción que captura el comportamiento periódico, pero no así la tendencia final que adquirirá la serie f, por lo que la predicción tiene un gran error.

Además, hay que tener en cuenta que todavía hay un amplio margen de mejora, como se indica en la sección 5.5.

5.5. Mejoras y otros posibles usos

Consideramos que las predicciones obtenidas todavía pueden ser mejoradas, quizás con un diccionario adaptado a extraer las características particulares de nuestras series que más puedan ayudar a la predicción. Mallat y Zhang le dan una gran importancia al aprendizaje del diccionario como una tarea equivalente a encontrar la estructura interna de las series. I. Tošić y P. Frossard (2011) [68] nos presentan una amplia revisión de los principales algoritmos para el aprendizaje de diccionarios. Trabajos recientes en esta línea son:

- M. Khodayar et al. (2020) [38] y L. Jun y L. Shi-chang (2020) [37] para predicción del consumo eléctrico.
- B. Chen (2020) [10] para predicción de la calidad del aire.

Como mejora, también sería posible utilizar un método híbrido en el que se combinase la predicción del algoritmo Matching Pursuit con la de un modelo basado en el residuo.

El algoritmo *Matching Pursuit* está relacionado con otros de nuestros trabajos anteriores ([20], [18], [22] y [19]) sobre detección de plagas de termitas. En este caso, consideramos que la caracterización mediante átomos fundamentales de los eventos producidos por las termitas, para un determinado sensor acelerómetro, sería un factor clave en su detección.

Capítulo 6

Introducción al aprendizaje profundo

Cada vez es más común oír los términos aprendizaje automático (*ML*; *machine laerning*) y aprendizaje profundo (*DL*; *deep learning*). Los algoritmos basados en ML y DL forman ya parte de nuestra vida cotidiana por la gran cantidad de datos que generamos, nosotros y nuestro entorno, pero realmente, ¿a qué nos referimos con estos términos? Podemos decir que un algoritmo ML extrae información a partir de unos datos, la cual se representa en un modelo que bien podría realizar inferencias sobre datos no observados o realizar otro tipo de tarea. Estos modelos pueden estar basados en redes neuronales artificiales, las cuales cada vez tienen mayores capacidades de entrenamiento y cálculo debido a los avances en los últimos años. Como consecuencia, el diseño de modelos es cada vez más complejo y con mayor profundidad de capas, lo cual posibilita un aprendizaje estructurado profundo (DL) de las características necesarias, a partir de los datos en bruto, para resolver un determinado problema. Este aprendizaje se consigue de forma automática utilizando un procedimiento de propósito general.

De forma jerárquica el DL forma parte del ML, que a su vez está dentro del concepto más amplio de inteligencia artificial (*AI*; *artificial intelligence*); hacer que las máquinas aprendan forma parte de la inteligencia artificial (figura 6.1).

6.1. Introducción al aprendizaje automático

Un algoritmo de aprendizaje automático se define como aquel que es capaz de aprender de la experiencia para realizar una tarea, combinando un procedimiento de optimización, una función de coste, un modelo y un conjun-

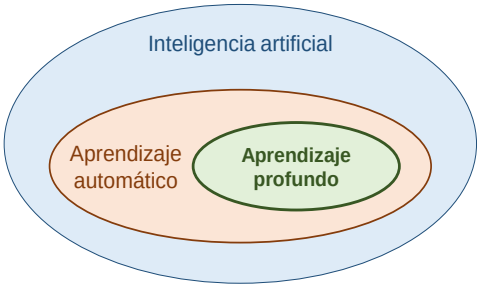

Figura 6.1: Relación entre la inteligencia artificial, el aprendizaje automático y el aprendizaje profundo.

to de datos (*dataset*). En las siguientes subsecciones vamos tratar cada uno de estos elementos. Además, describiremos algunos conceptos relacionados con la optimización, como el sobrentrenamiento, la capacidad y la regularización. Finalmente, veremos cómo establecer funciones de coste para redes probabilísticas mediante la estimación de máxima verosimilitud, así como para redes no probabilísticas de regresión justificaremos el uso de las funciones de coste error cuadrático medio (MSE) y error absoluto medio (MAE).

6.1.1. Tipos de tareas

Los dos tipos principales de problemas que pueden resolverse mediante ML son la regresión y la clasificación. Vamos a describirlos en función de los datos de entrada del algoritmo (o una serie de características extraídas de estos) y los datos de salida:

- Regresión. En este caso el algoritmo realizará un mapeo $\mathbb{R}^n \rightarrow \mathbb{R}^h$, donde una entrada $\mathbf{x}$ producirá una salida $\mathbf{y}$ con una dimensionalidad h. Un ejemplo para $h > 1$ puede ser la predicción de las futuras h temperaturas máximas diarias en función de otras variables meteorológicas de días anteriores. Normalmente consideraremos la salida $\mathbf{y}$ como una estimación de algún estadístico obtenido a partir de $\mathbf{x}$, como puede ser la media o la mediana (sección 6.1.9). También es posible plantear este tipo de problema probabilísticamente, como proponen A. V. D. Oord et al. para la generación de audio [54], donde para cada predicción que se desease realizar, la salida contendría las probabilidades de todos los posibles valores que pueda tomar.

- Clasificación. Esta tarea pretende determinar a qué categoría pertenece la entrada del algoritmo, que realizará un mapeo $y = f(\mathbf{x})$, donde $y \in$

$\{1, \ldots, c\}$, siendo c el número de categorías posibles. Se suele utilizar para el reconocimiento de objetos. Es posible plantear este problema probabilísticamente, donde la salida sea un conjunto de valores con la probabilidad de pertenecer a cada categoría.

Otros tipos de tareas comunes son:

- Agrupación (*clustering*). Para esta tarea es necesario disponer de una medida de distancia entre puntos, lo cual permite al algoritmo crear agrupamientos.
- Transcripción. A partir de una representación relativamente desestructurada de algún tipo de datos se debe transcribir a texto. Un ejemplo sería, el reconocimiento de voz o poner automáticamente títulos a fotografías.
- Traducción. La entrada al algoritmo sería una secuencia de símbolos en un lenguaje que se deberá convertir a una secuencia en otro.
- Estimación de una distribución de probabilidad. La tarea consiste en aprender una función $p_{modelo}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, que se interpretaría como una función densidad de probabilidad para entrada continua o una función masa de probabilidad si la entrada fuese discreta.

6.1.2. El conjunto de datos (*dataset*)

Un modelo basado en ML debe aprender a realizar una tarea a partir de un conjunto de datos compuesto por ejemplares, también denominados puntos, que contienen esa experiencia necesaria para conseguir el aprendizaje. Básicamente existen dos formas de que este se produzca:

- Aprendizaje supervisado. Es la forma más común de aprendizaje. El *dataset* contiene tanto ejemplares de entrada al algoritmo como sus valores de salida correspondientes, y se encuentra dividido en datos de entrenamiento y test (sección 3.5) con el fin de poder evaluar el comportamiento real del algoritmo para ejemplos no vistos anteriormente. Problemas típicos que utilizan este tipo de aprendizaje son la clasificación y regresión.
- Aprendizaje no supervisado. Se utiliza el *dataset* completo, sin división. El algoritmo podría aprender implícitamente la distribución de probabilidad generadora de los datos como una parte en la solución de problemas más complejos. Otro ejemplo de este tipo de aprendizaje es el agrupamiento (*clustering*) de datos.

6.1.3. Funciones de coste

En un marco de aprendizaje supervisado debemos distinguir entre una función de coste, utilizada por el optimizador para el ajuste de los parámetros del modelo, y una medida de la exactitud de la salida del modelo. Distinguiremos dos casos:

- Regresión: en este tipo de problema son habituales funciones de coste como el error absoluto medio (MAE), error cuadrático medio (MSE) y error porcentual absoluto medio (MAPE). La elección del error puede hacerse de forma arbitraria, o en función de la estimación que pretendamos obtener como salida del modelo, la media o mediana (sección 6.1.9). Estos errores también pueden usarse como métricas de evaluación de los resultados de los modelos, además de otras que fueron definidas en la sección 3.5.

- Clasificación: se pueden utilizar métricas de la exactitud de las predicciones basadas en el conteo de aciertos sobre el total, pero en cambio no sirven como funciones de coste, ya que no son de curva suave (*smoothed*) que permitan la obtención del gradiente (sección 6.1.4). Una opción sería usar la función de error *hinge* [24], que sirve para ambas medidas del error (exactitud y coste) y se define como:

$$máx(0, 1 - ty), \tag{6.1}$$

  donde $t \in \{-1, 1\}$ es el valor correcto para una salida y del modelo. Este error es 0 cuando t e y tienen el mismo signo e $|y| \leq 1$; si no, el error es $1 - ty$; una función derivable.

  Considerando un modelo de aprendizaje automático probabilístico, cuya salida contenga la probabilidad de pertenencia a cada posible clase, entonces se pueden usar métricas basadas en la entropía cruzada (sección 6.1.8), para la exactitud y función de coste.

6.1.4. El procedimiento de optimización

La mayoría de los algoritmos de ML requieren algún tipo de optimización. Debido la complejidad y no linealidad de la función de coste en modelos de aprendizaje profundo, esta suele minimizarse mediante un optimizador basado en el gradiente descendente (*GD*; *gradient descent*). La función de coste se denota mediante $J(\boldsymbol{\theta})$, donde $\boldsymbol{\theta}$ es un vector con los parámetros del modelo a ajustar. La optimización por GD es un método iterativo en el que en cada iteración (*epoch*) se modifican los parámetros $\boldsymbol{\theta}$ del modelo a partir

del gradiente promedio $\Delta J(\boldsymbol{\theta})$, obtenido para todo el conjunto de datos de entrenamiento, de la forma

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \, \Delta J(\boldsymbol{\theta}), \tag{6.2}$$

donde α es el ratio de aprendizaje. La función de coste puede verse como una especie de superficie con montañas y valles en el espacio multidimensional de los parámetros. El gradiente negativo indica la dirección de descenso en esta superficie que conduce a un valle donde el error encuentra un valor mínimo en promedio.

Cuando el número de ejemplos de entrenamiento es muy grande, se utiliza el método del gradiente descendente estocástico (*SGD*; *stochastic gradient descent*), en el que en cada iteración se selecciona aleatoriamente un conjunto (*minibatch*) de ejemplos de entrenamiento con la doble finalidad de obtener una buena estimación del gradiente y conseguir una buena eficiencia en tiempo; sorprendentemente mejores que en algunas técnicas de optimización más elaboradas.

Existen otras variantes del procedimiento SGD, como la Adam (*adaptive moment estimation*) [39], utilizado en este trabajo para nuestros modelos de aprendizaje profundo. Este método computa de forma individual para cada parámetro del modelo un ratio de aprendizaje adaptativo a partir de la estimación del primer y segundo momento de los gradientes anteriores del parámetro.

6.1.5. Sobrentrenamiento (*overfitting*)

El principal objetivo de un algoritmo ML es que realice correctamente su tarea para datos no utilizados previamente en el entrenamiento. Mediante la optimización conseguimos minimizar el error de entrenamiento, pero lo realmente interesante es el error con los datos de test, donde verdaderamente vamos a comprobar el comportamiento del modelo para nuevos datos.

Debemos considerar que nuestro dataset está constituido por un conjunto de ejemplares generados independientemente por una distribución de probabilidad p_{datos}, y que los datos de entrenamiento y test están idénticamente distribuidos. Por tanto, no tiene sentido centrarnos en minimizar exclusivamente el error de entrenamiento y no considerar errores propios de la aleatoriedad de los datos que después vamos a encontrar en los datos de test. O sea, un sobreentrenamiento del modelo conduce a una mala generalización para nuevos datos. Es habitual observar como el error con datos de entrenamiento va decayendo progresivamente durante el entrenamiento, pero en cambio, con datos de test comienza a crecer después de haber alcanzado

un mínimo.

Otro concepto importante es el de capacidad de un modelo. Con ello nos referimos a la capacidad que tiene de adaptarse a una variedad de funciones $\mathbf{y} = f(\mathbf{x})$. Podemos controlar el sobreentrenamiento buscando el equilibrio entre un modelo con poca capacidad, que va a tener dificultades en ajustarse a los datos de entrenamiento, y otro modelo con alta capacidad, que puede sobreajustarse memorizando propiedades de los datos de entrenamiento que no sirven en el conjunto de test.

6.1.6. Regularización

La regularización es cualquier medida que se tome respecto al algoritmo de aprendizaje que evite el sobreentrenamiento. Esto conlleva limitar la familia de funciones que el modelo puede aproximar, o sea, la capacidad del modelo. Hay varias formas de regularización entre las que están:

- Penalización en la función de coste de determinados valores de los parámetros θ a ajustar, que en el caso de un problema de regresión quedaría definida como:

$$J(\boldsymbol{\theta}) = Error_{entrenamiento} + R(\boldsymbol{\theta}), \tag{6.3}$$

  donde $R(\boldsymbol{\theta})$ es la función de regularización, que en el caso de la regularización *Elastic Net* sería: $l_1 \sum_i |\theta_i| + l_2 \sum_i \theta_i^2$, donde l_1 y l_2 son los factores que permiten una combinación de las regularizaciones conocidas como $L1$ y $L2$ respectivamente. Esta regularización busca minimizar el error de entrenamiento sin olvidar que los valores de los parámetros deben ser pequeños. La minimización del término $L1$ conduce a parámetros pequeños o cero, y la $L2$ a parámetros pequeños pero no cero. En resumen, se buscan soluciones $\mathbf{y} = f(\mathbf{x})$ con pendientes pequeñas o con menos parámetros.

- Interrupción temprana del optimizador (*early stopping*). Probablemente sea la forma de regularización más utilizada en aprendizaje profundo. Se puede entender que es una regularización en la duración de la optimización, que se interrumpe cuando el error en un conjunto de datos separados del conjunto de entrenamiento no puede mejorar más. Los valores de los parámetros $\boldsymbol{\theta}$ son salvados cada vez que se mejora este error, y una vez que finaliza el entrenamiento se recuperan los últimos valores salvados. Un parámetro denominado *paciencia*, indica el número de iteraciones que esperamos a que mejore el error antes de interrumpir. Es una técnica muy efectiva, además de simple.

Existen otras soluciones de regularización, como el denominado descarte (*dropout*) en redes neuronales, que explicamos en la sección 6.2.5.

6.1.7. Hiperparámetros y validación

Un hiperparámetro es aquel que se puede seleccionar libremente y que pudiera tener algún efecto sobre los resultados del algoritmo de aprendizaje. Son parámetros que no se ajustan en el proceso de optimización. Por ejemplo, en un problema de regresión polinómica sería el grado del polinomio, y en un modelo de red neuronal con varias capas el tamaño de cada una. Dependiendo del tipo de modelo existen una multitud de hiperparámetros que pueden configurarse. De forma común a cualquier tipo de modelo podemos considerar algunos, como son los relacionados con:

- Optimizador
- Regularización
- Función de coste

Para cada combinación de los valores de los hiperparámetros podemos realizar el entrenamiento y comprobar el rendimiento de nuestro algoritmo de aprendizaje frente a los datos de test. La combinación de hiperparámetros que obtuviera mejor resultado sería la elegida, pero nos plantearíamos la duda de si para otros datos de test diferentes esa configuración sería realmente la mejor. Observamos que indirectamente hemos utilizado los datos de test para el ajuste de los hiperparámetros, lo cual puede provocar que el modelo no generalice bien con datos no disponibles hasta el momento y el resultado del modelo no se corresponda con el real. Por ello, para ajustar los hiperparámetros y posteriormente evaluar el modelo con nuevos datos, es necesario dividir el conjunto de datos disponibles en tres partes: entrenamiento, validación y test. El ajuste de hiperparámetros se haría entrenando y evaluando los resultados con los datos de validación, y la evaluación final del modelo se efectuaría entrenándolo con los datos unidos de entrenamiento y validación, para obtener el resultado final con los datos de test.

A veces puede darse el caso de que incluso evitando usar los datos de test para la configuración de hiperparámetros, estos influyan en el resultado. Ocurre cuando son datos públicos con los que la comunidad científica intenta probar una infinidad algoritmos hasta que se consigue batir los resultados de referencia publicados anteriormente. Esto motiva la necesidad de cambiar con frecuencia los datos de test publicados y aumentar el número de ejemplares contenidos.

6.1.8. Estimación de máxima verosimilitud (*MLE*)

La estimación de máxima verosimilitud (*MLE*; *maximum likelihood estimation*) es un método de estimación de los parámetros de una distribución de probabilidad con la que pretendemos aproximarnos lo máximo posible a la verdadera distribución de probabilidad cuando disponemos de un conjunto de datos generados por esta última.

Dado un conjunto de m observaciones $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m$ independientemente extraídas a partir de la distribución de probabilidad generadora de datos p_{datos}, deseamos estimar los parámetros $\boldsymbol{\theta}$ de una distribución de probabilidad p_{modelo} que mejor la aproxime a p_{datos}, suponiendo que existe una combinación de parámetros tal que p_{modelo} sea la propia p_{datos}. El estimador de máxima verosimilitud para $\boldsymbol{\theta}$ se define como

$$\begin{aligned} \boldsymbol{\theta}_{MLE} &= \arg\max_{\boldsymbol{\theta}} p_{modelo}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m; \boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{modelo}(\boldsymbol{x}_i; \boldsymbol{\theta}), \end{aligned} \tag{6.4}$$

que de forma equivalente se puede escribir como

$$\begin{aligned} \boldsymbol{\theta}_{MLE} &= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} log\, p_{modelo}(\boldsymbol{x}_i; \boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x} \sim p_{data}} log\, p_{modelo}(\boldsymbol{x}; \boldsymbol{\theta}), \end{aligned} \tag{6.5}$$

donde $\mathbb{E}_{\mathbf{x} \sim p_{data}}$ denota el valor esperado considerendo que $\mathbf{x}$ se ha generado mediante la distribución de probabilidad de los datos p_{data}. Esta última expresión equivale a minimizar la entropía cruzada de p_{datos} y p_{modelo}:

$$- \mathbb{E}_{\mathbf{x} \sim p_{data}} log\, p_{modelo}(\boldsymbol{x}; \boldsymbol{\theta}). \tag{6.6}$$

En un problema de aprendizaje supervisado, como una clasificación probabilística, el MLE puede ser generalizado para la estimación de la probabilidad condicional $P_{modelo}(\boldsymbol{y}|\boldsymbol{x})$ que predice $\boldsymbol{y}$ en función de $\boldsymbol{x}$. La optimización del modelo podríamos hacerla entonces con una función de coste mediante la entropía cruzada:

$$J(\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{x},\mathbf{y} \sim P_{data}} log\, P_{modelo}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}). \tag{6.7}$$

Cualquier función de coste consistente en el logaritmo negativo de la máxima verosimilitud es una entropía cruzada entre la distribución empírica definida por los datos de entrenamiento y la distribución de probabilidad definida por el modelo, independientemente de si la distribución de probabilidad es la Bernoulli u otra cualquiera.

6.1.9. Funciones de coste *MSE* y *MAE*

La elección entre el error cuadrático medio (MSE) y el error absoluto medio (MAE) no tiene por qué ser arbitraria. En un problema de regresión podemos hacer que la tarea consista en aprender un determinado estadístico $\mathbf{y}$ condicionado a la entrada $\mathbf{x}$ ($\mathbf{y} = f(\mathbf{x})$). Es decir, nuestra función de coste se minimizará cuando nuestro modelo represente una función f (a partir de los parámetros $\boldsymbol{\theta}$) que de lugar a ese estadístico. En este sentido se han obtenido dos resultados según el estadístico concreto que la función f deba representar [23]:

- La media de $\mathbf{y}$ condicionado a $\mathbf{x}$ se obtiene para la función de coste error cuadrático medio (MSE):

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{y} - f(\mathbf{x}, \boldsymbol{\theta}))^2, \tag{6.8}$$

- La mediana de $\mathbf{y}$ condicionado a $\mathbf{x}$ se obtiene para la función de coste error absoluto medio (MAE):

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\|\mathbf{y} - f(\mathbf{x}, \boldsymbol{\theta})\|_1, \tag{6.9}$$

A estos resultados se llega considerando que podemos entrenar el modelo con infinitos ejemplares obtenidos de la verdadera distribución de probabilidad generadora de los datos.

6.2. Introducción a las redes neuronales profundas

Las redes neuronales artificiales (*NN*; *neural networks*) son un tipo de modelo de aprendizaje automático inspirado en las redes neuronales biológicas del cerebro. En esta sección describiremos brevemente los inicios de las NN, desde el perceptrón, hasta llegar las redes neuronales profundas (DNN; *deep neural networks*) con la red de propagación hacia adelante, denominada perceptrón multicapa (*MLP*; *multilayer perceptron*). También describiremos las funciones de activación más habituales para cada tipo de capa. Introduciremos la propagación hacia atrás de errores para el calculo del ajuste de los parámetros de la red, dentro del proceso de optimización. Finalmente, trataremos una forma de regularización por descarte *dropout*. Otras arquitecturas

de DNN que han tenido un gran éxito son la convolucional y recurrente, que las expondremos en los capítulos 8 y 9 respectivamente, formando el grueso de nuestra aportación en esta investigación sobre predicción de series temporales.

El aprendizaje de representaciones es un reto que hoy en día ha sido alcanzado por las DNN permitiendo que las entradas de los modelos no necesiten ningún procesamiento previo, siendo obtenida esta representación de forma automática por el algoritmo de aprendizaje. Los distintos niveles en los que se compone una DNN permite que cada uno de ellos obtenga una abstracción cada vez mayor de los datos, resultando en funciones complejas de los datos en bruto.

6.2.1. Perceptrón

Desde la década de 1940 las técnicas de aproximación de funciones se utilizaron como motivación para la creación de modelos de aprendizaje automático. En ese contexto F. Rosenblatt en 1958 [60] creo el perceptrón, hoy en día la red neuronal más sencilla y consistente en un modelo lineal de capa simple usado para clasificación binaria. Un clasificador binario debe decidir si la entrada, representada por un vector $\mathbf{x}$ de números, corresponde a una u otra clase en $\{0, 1\}$, proporcionando la salida y:

$$y = \begin{cases} 1 & \text{si } \ \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{en otro caso} \end{cases} \tag{6.10}$$

donde $\mathbf{w} \cdot \mathbf{x}$ denota el producto escalar entre el vector de pesos $\mathbf{w}$ y la entrada $\mathbf{x}$, y b es un valor de sesgo (*bias*) o umbral de activación de neurona (función de activación escalón unitario). De forma similar a cómo la sinapsis pasa información a otras neuronas biológicas, podemos ver cómo el perceptrón toma la entrada desde las conexiones dándoles un peso a cada una.

Lamentablemente, las críticas de M. Minsky y S. Papert (1969) [51] sobre las limitaciones de los perceptrones llevó a una reacción contra todo el enfoque de las redes neuronales. Se señalaron varios de los defectos de la familia de modelos lineales, como su incapacidad para aprender la función XOR, aunque en la década de 1960 ya habían comenzado a aparecer aplicaciones eficientes de la regla de la cadena basadas en programación dinámica [63]. Sería necesario el desarrollo del perceptrón multicapa y los medios de cálculo del gradiente en estos modelos para que fuese posible la aproximación a funciones no lineales. La idea de que esto era factible y funcionaba fue descubierta por diferentes grupos de investigación en la década de 1970 y 1980 [42].

6.2.2. Perceptrón multicapa (MLP)

El perceptrón multicapa, también llamado red neuronal de propagación hacia adelante, constituye la esencia del aprendizaje profundo. Es un modelo de red neuronal con una capa de entrada, una o varias capas ocultas y una capa de salida. En cada capa puede haber una o varias unidades neuronales como el perceptrón, pero con una función de activación que depende de la capa. La ecuación de una neurona artificial de una capa con salida y queda definida mediante

$$y = \phi(\mathbf{w}\mathbf{x} + b), \tag{6.11}$$

donde ϕ es una función de activación lineal o no lineal, y b el bias. En una capa con múltiples unidades, cada unidad dará lugar a una salida que formará parte del vector de salida $\mathbf{y}$ definido como

$$\mathbf{y} = \phi(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{6.12}$$

donde $\mathbf{W}$ es la matriz de pesos en la que cada fila corresponde a una unidad, y $\mathbf{b}$ el vector con el sesgo de cada neurona. La función de activación ϕ se aplica elemento a elemento sobre $\mathbf{W}\mathbf{x} + \mathbf{b}$ dando como resultado el vector $\mathbf{y}$.

Una capa de entrada permite proporcionar los datos de entrada a la red neuronal; es el vector de entrada. Las capas ocultas toman como entrada la salida de la capa anterior. A medida que la información pasa por diferentes capas ocultas se va obteniendo una representación de la información más abstracta mediante la posibilidad del uso de funciones de activación no lineales que permiten al modelo entender la correcta interacción entre las variables de entrada, evitando los problemas del perceptrón. La capa de salida ofrece la predicción del modelo, que puede ser por ejemplo, un valor real en un problema típico de regresión, o un conjunto de probabilidades o valor discreto en una clasificación. Excepto la capa de entrada, cada una de las capas toma como entrada la salida completa de la capa anterior, por lo que se denominan capas totalmente conectadas o densas. El número de capas encadenadas en el modelo se llama profundidad. En una red neuronal hacia adelante no existen conexiones de realimentación de las salidas hacia las entradas. Cuando esto ocurre se denominan redes neuronales recurrentes (*RNN*; *recurrent neural networks*), que trataremos en el capítulo 9.

Una red neuronal en su conjunto realiza una aproximación a una función que mapea la entrada $\mathbf{x}$ a la salida $\mathbf{y}$, guiada durante el entrenamiento por la optimización de una función de coste. La no linealidad los modelos de redes neuronales conlleva la no convexidad de las funciones de coste más interesantes, por lo que se suele utilizar algún método iterativo de gradiente descendente para el ajuste de los parámetros (pesos y sesgos). La elección de

la función de coste está relacionada con el tipo de salida de la red. Como se explicó en las subsecciones 6.1.3, 6.1.8 y 6.1.9, para un modelo de clasificación probabilístico se puede utilizar la entropía cruzada, y para clasificación discreta el error *hinge*; en cambio, para una regresión se puede utilizar el MAE o MSE. Siempre que sea posible se utiliza la entropía cruzada, pues proporciona mejores resultados, ya que evita una posible saturación de las neuronas de salida con la función de activación sigmoidal que se produce con MAE y MSE.

6.2.3. Funciones de activación

La función de activación ϕ de una capa definida como $\mathbf{y} = \phi(\mathbf{W}\mathbf{x}+\mathbf{b})$ da lugar a su salida. Excepto la función identidad, todas introducen no linealidad en la salida de capa. Los tipos más comunes de salida y sus correspondientes funciones de activación son:

- Lineal. La activación es la función la identidad, o sea, la salida de la capa viene dada por $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$, por tanto cada neurona tiene una salida lineal. Se puede utilizar en cualquier capa, pero específicamente en la capa de salida para problemas de regresión. Tienen la ventaja de que no se saturan, por lo que no suponen ningún problema para el optimizador basado en el gradiente descendente.

- Sigmoidal. Una función de activación sigmoidal, como la función logística (σ) o la tangente hiperbólica (tanh), produce una salida con forma de S para cada neurona de la capa (figura 6.2). Presenta el inconveniente de que se satura con facilidad cuando no se opera en su zona central. Por este motivo, no es recomendable su uso para capas ocultas, pero en caso de que sea necesaria, tanh facilita más el entrenamiento al mantener valores pequeños en las activaciones. En cambio, como función de activación de una capa de salida es frecuente utilizarla siempre que la función de coste aplicada invierta la función de activación, y el conjunto se transforme en una función lineal, como ocurre cuando se utiliza como función de coste la entropía cruzada. Se suele utilizar en capas de salida en problemas de clasificación binaria entendiendo la salida de la red como una estimación de $P(y = 1 \mid \mathbf{x})$ para una distribución de probabilidad de Bernoulli condicionada a la entrada $\mathbf{x}$ de la red neuronal.

- Softmax. En problemas de clasificación con múltiples alternativas se suele utilizar en la capa de salida la función de activación softmax, que

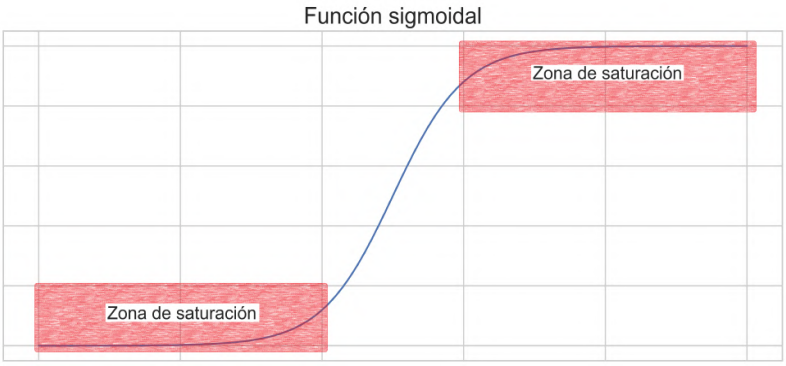

Figura 6.2: Función sigmoidal. Zonas de saturación.

se puede entender como una generalización de la función logística, y representa la distribución de probabilidad *multinoulli* sobre las diferentes posibles clases.

- Lineal rectificada. Este tipo de unidad (*ReLU*; *rectified linear unit*) tiene como función de activación el rectificador, dando lugar a un tipo de capa definida como $\mathbf{y} = máx(0, \mathbf{Wx})$. Podemos decir que cada unidad ReLU funciona como una de tipo lineal, pero sustituyendo la salida por el valor cero si fuese negativa. Esto aporta la necesaria no linealidad, además de que evita los problemas de saturación de las funciones de activación sigmoidales. Por los buenos resultados obtenidos se ha convertido en la función de activación por defecto para capas ocultas. Obviamente, la desventaja que tiene es que cuando la activación es cero, la unidad neuronal no puede aprender en la optimización por gradiente descendente. Este problema se atenúa inicializando el vector de sesgos a valores 0.1.

6.2.4. Propagación hacia atrás (*back-propagation*)

En un perceptrón multicapa (MLP) los datos de entrada son procesados por las diferentes capas ocultas hasta la capa de salida en una propagación hacia adelante de la información. Un algoritmo de aprendizaje debe ser capaz de modificar los parámetros del modelo a partir del errores en las salidas. La propagación hacia atrás del error total $J(\boldsymbol{\theta})$ cometido, desda la capa de salida hacia la capa de entrada, permite obtener en qué medida es afectado por cada

parámetro. Buscamos obtener el gradiente

$$\Delta J(\boldsymbol{\theta}) = (\frac{\partial J(\boldsymbol{\theta})}{\partial J(\theta_1)}, \ldots, \frac{\partial J(\boldsymbol{\theta})}{\partial J(\theta_m)}), \tag{6.13}$$

y por tanto necesitamos calcular la derivada parcial de la función de coste respecto a cada parámetro a optimizar. En la sección 6.1.4 explicamos el procedimiento general de optimización de estos parámetros para un conjunto de ejemplares de entrenamiento, pero ahora en lo que nos vamos a centrar es en la forma de obtener esas derivadas parciales para un solo ejemplar. La generalización a un conjunto completo se obtiene mediante la suma de los gradientes individuales.

Para el cálculo del gradiente se requiere una función de coste continua y derivable, que estará condicionada por las funciones de activación, como puede ser la función logística $\sigma(x)$, cuya derivada es $\sigma(x)(1-\sigma(x))$, y que utilizaremos en nuestra explicación. Nos apoyaremos en el modelo de la figura 6.3, que es básicamente una red MLP mínima, extendida con una unidad para el cálculo del error o función de coste MSE, que para un solo ejemplar de entrenamiento sería

$$J(\boldsymbol{\theta}) = \|\mathbf{y}^{(2)} - \mathbf{y}\|^2, \tag{6.14}$$

donde $\mathbf{y}$, $\mathbf{y}^{(2)}$ representan el valor objetivo y la salida de la red respectivamente; y $\|.\|^2$ la norma-2 al cuadrado. En nuestro modelo de dos neuronas por capa el error sería

$$J(\boldsymbol{\theta}) = \frac{(y_1^{(2)} - y_1)^2 + (y_2^{(2)} - y_2)^2}{2}. \tag{6.15}$$

El error depende de los parámetros $\boldsymbol{\theta}$, que son los pesos de nuestro modelo y que no aparecen directamente en la expresión, pero lógicamente cada valor de salida $y_k^{(2)}$ del modelo depende de estos; hemos obviado los sesgos por simplicidad en el modelo.

La propagación hacia atrás del error, desde la salida hasta cada uno de los pesos del modelo, para encontrar la forma en que estos deben modificarse se consigue básicamente mediante la aplicación de la regla de la cadena para derivadas. Esta tarea puede ser descompuesta en varios pasos [59]:

1. Computación hacia adelante. A partir de la entrada $\mathbf{x}$ la información fluye hacia adelante calculándose, en cada unidad, la salida y la derivada de la salida respecto a la entrada y respecto a cada peso; así como el error J. Los resultados quedan almacenados para un uso posterior.

2. Propagación del error hacia la capa de salida. Nuestro interés está encontrar en qué cantidad debemos incrementar los pesos (parámetros)

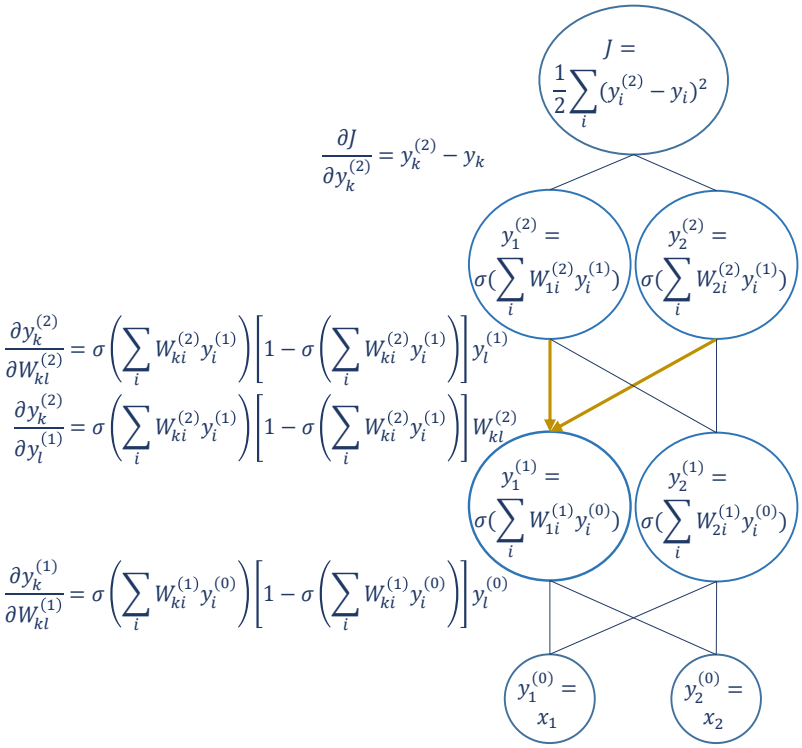

Figura 6.3: Propagación hacia adelante y hacia atrás en un red neuronal MLP. El modelo se ha ampliado con una unidad para el cálculo de la función de coste o error J a partir de la salida de la red neuronal ($y_1^{(2)}$,$y_2^{(2)}$) para un par de entrenamiento dado por la entrada (x_1, x_2) y su salida objetivo (y_1, y_2). Sin pérdida de generalidad, la red está compuesta por una capa de entrada, una oculta y una de salida, se han considerado dos unidades por capa y se han obviado los sesgos; la función de activación es la logística σ y la función de coste la MSE. En cada unidad aparece la función que realiza y a la izquierda de cada capa se muestra la derivada de la salida de cada unidad respecto a su entrada (si es necesaria para capas inferiores) y la derivada de la salida respecto a cada peso si se dispone de ellos. De abajo a arriba (hacia adelante) se obtiene la salida y el error de la red a partir de la entrada; y de arriba a abajo (hacia atrás) se obtiene el gradiente del error respecto de cada peso mediante la multiplicación de las derivadas correspondientes (explicación en la sección 6.2.4); como ejemplo, para obtener $\frac{\partial J(\boldsymbol{\theta})}{\partial W_{kl}^{(1)}}$necesitamos sumar la derivada obtenida por cada camino (resaltado en amarillo) de donde pueda provenir error. Utilizamos el superíndice $^{(\cdot)}$ para indicar el número de capa.

de la capa de salida. Entonces para la capa 2 y debemos obtener cada medida de error $\frac{\partial J}{\partial W_{kl}^{(2)}}$ correspondiente al peso l de una neurona k, que siguiendo la regla de la cadena sería

$$\frac{\partial J}{\partial W_{kl}^{(2)}} = \frac{\partial J}{\partial y_k^{(2)}} \frac{\partial y_k^{(2)}}{\partial W_{kl}^{(2)}}. \tag{6.16}$$

La primera derivada $\frac{\partial J}{\partial y_k^{(2)}}$ la obtenemos fácilmente a partir de la ecuación 6.15:

$$\frac{\partial J}{\partial y_k^{(2)}} = y_k^{(2)} - y_k, \tag{6.17}$$

y la segunda derivada se obtiene mediante la derivada de la salida de una neurona con función de activación logística σ:

$$\begin{aligned} \frac{\partial y_k^{(2)}}{\partial W_{kl}^{(2)}} &= \sigma(\sum_i W_{ki}^{(2)} y_i^{(1)})(1 - \sigma(\sum_i W_{ki}^{(2)} y_i^{(1)}))y_l^{(1)} \\ &= y_k^{(2)}(1 - y_k^{(2)})y_l^{(1)}. \end{aligned} \tag{6.18}$$

El resultado simplificado para poder modificar un peso de la capa de salida es

$$\frac{\partial J}{\partial W_{kl}^{(2)}} = (y_k^{(2)} - y_k)y_k^{(2)}(1 - y_k^{(2)})y_l^{(1)}. \tag{6.19}$$

Los valores necesarios para obtener esta derivada parcial se han calculado en el primer paso, de computación hacia adelante.

3. Propagación de errores hasta la capa oculta. Aquí debemos tener en cuenta que una neurona oculta tiene varias fuentes de error ya que su salida está conectada a las unidades de la capa siguiente y debemos considerar todos los caminos posibles por donde pueda fluir el error hacia atrás. De esta forma, para obtener la derivada de la fuente de error proveniente de la unidad 1 (J_1) de la capa 2 respecto del peso l de la neurona k de la capa oculta 1 usaremos la ecuación

$$\frac{\partial J_1}{\partial W_{kl}^{(1)}} = \frac{\partial J_1}{\partial y_k^{(1)}} \frac{\partial y_k^{(1)}}{\partial W_{kl}^{(1)}} \tag{6.20}$$

y la derivada de la fuente de error proveniente de la unidad 2 (J_1) de la capa 2 respecto del mismo peso de la capa oculta 1 sería

$$\frac{\partial J_2}{\partial W_{kl}^{(1)}} = \frac{\partial J_2}{\partial y_k^{(1)}} \frac{\partial y_k^{(1)}}{\partial W_{kl}^{(1)}}. \tag{6.21}$$

Si sumamos el efecto de todas las fuentes de error (J_i) posibles podemos escribir la ecuación que nos permitirá modificar un peso l de una neurona k de la capa oculta 1:

$$\frac{\partial J}{\partial W_{kl}^{(1)}} = \left(\sum_i \frac{\partial J_i}{\partial y_k^{(1)}} \right) \frac{\partial y_k^{(1)}}{\partial W_{kl}^{(1)}} \tag{6.22}$$

El primer factor, compuesto por un sumatorio, lo podemos calcular a partir de la regla de la cadena aplicada a cada término, conduciendo a la ecuación:

$$\sum_i \frac{\partial J_i}{\partial y_k^{(1)}} = \sum_i (y_i^{(2)} - y_i) y_i^{(2)} (1 - y_i^{(2)}) W_{kl}^{(1)}. \tag{6.23}$$

El segundo factor lo podemos obtener de forma similar a la ecuación 6.18:

$$\frac{\partial y_k^{(1)}}{\partial W_{kl}^{(1)}} = y_k^{(1)} (1 - y_k^{(1)}) y_l^{(0)}. \tag{6.24}$$

Finalmente, la ecuación que permite ajustar un peso l de una neurona k de la capa oculta 1 es

$$\frac{\partial J}{\partial W_{kl}^{(1)}} = \left[\sum_i (y_i^{(2)} - y_i) y_i^{(2)} (1 - y_i^{(2)}) W_{kl}^{(1)} \right] y_k^{(1)} (1 - y_k^{(1)}) y_l^{(0)}. \tag{6.25}$$

Todos los valores utilizados en esta expresión se han calculado en la propagación hacia adelante, y en la propagación hacia atrás son operados para obtener el resultado.

4. Modificación de pesos. En los pasos anteriores hemos obtenido la derivada parcial del error $J(\boldsymbol{\theta})$ respecto a cada parámetro, cuyo valor nos da una idea del error cometido por el parámetro. Por este motivo podemos ver la concatenación de derivadas como una propagación del error desde su salida hasta cada parámetro. Para la modificación de los parámetros del modelo se utiliza la ecuación 6.2, aunque esto es más bien parte del procedimiento de optimización.

Esta explicación es una introducción al cálculo del gradiente del error, que realmente se implementa en la práctica mediante el algoritmo *back-propagation* [61], diseñado eficientemente para este propósito.

6.2.5. Descarte (*dropout*)

La técnica de regularización por descarte (*dropout*) consiste en la eliminación de algunas unidades neuronales que no sean de la capa de salida. Esto se consigue simplemente multiplicando por cero la salida de estas unidades durante el entrenamiento. De forma más precisa, en un entrenamiento con el método del gradiente descendente estocástico, cada vez que se carga un ejemplar de entrenamiento, se elige aleatoriamente, con una determinada probabilidad (hiperparámetro), las unidades de entrada y ocultas que tendrán la salida habilitada, y el resto serán multiplicadas por cero. Normalmente para neuronas de entrada esta probabilidad es 0,8 y en ocultas 0,5. El proceso de actualización de pesos se realiza de forma habitual por el optimizador. Como vemos, la técnica *dropout* no tiene coste computacional, y además se puede aplicar a una gran variedad de modelos.

Lo interesante de esta técnica es su interpretación. Pensemos qué significa eliminar aleatoriamente varias unidades de un modelo para cada ejemplar de entrenamiento. En realidad, lo que estamos haciendo por cada ejemplar, es entrenar a una red neuronal diferente a la inicialmente diseñada (con todas las neuronas habilitadas) y sumando sus diferentes efectos del aprendizaje en la modificación de pesos durante la optimización. Es algo similar a como si entrenáramos una gran cantidad de modelos por separado y después tomásemos la media de las salidas de todos los modelos, pero realizado de forma eficiente computacionalmente. En la figura 6.4 se muestra un ejemplo de los diferentes modelos a los que daría lugar esta técnica, con la que conseguimos una efectiva regularización, que evita el sobreajuste y mejora el error de generalización.

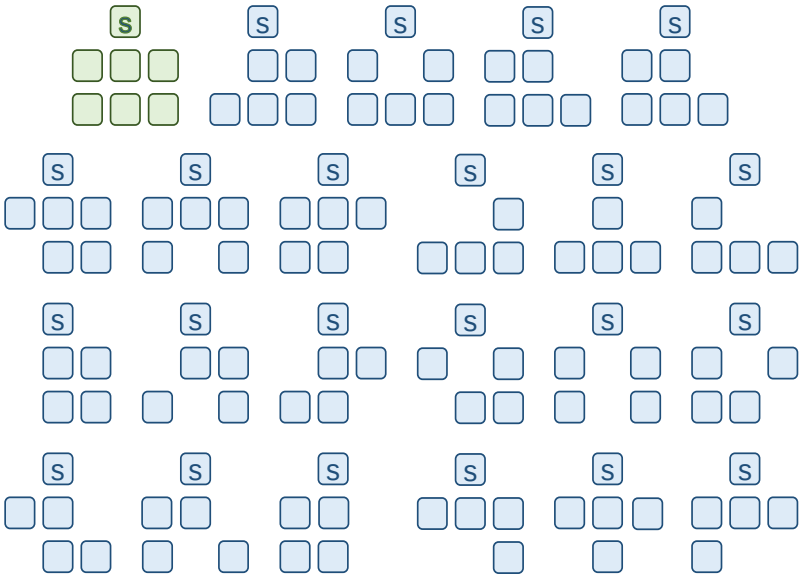

Figura 6.4: Efecto de la aplicación del descarte (*dropout*) de unidades neuronales. Se muestra una red neuronal simple de una capa de entrada de tres unidades, una oculta de 3 unidades y una de salida (S) de una unidad. En verde el modelo completo con todas las unidades habilitadas. En azul diferentes configuraciones resultado del descarte de una o dos unidades. Se observa el gran número de modelos diferentes que se pueden llegar a generar.

Capítulo 7

Tarea de predicción de tráfico marítimo

En este capítulo describimos la tarea de predicción de series temporales. En primer lugar, presentamos el *dataset* utilizado en nuestra experimentación, y posteriormente explicamos el enfoque del aprendizaje automático a la predicción de series.

7.1. Descripción de los datos y horizontes de predicción

Nuestro *dataset* consiste en siete series desagregadas de importación de mercancías de Marruecos a España a través del puerto de Algeciras en el periodo de 2000 a 2016, correspondiente a los capítulos arancelarios europeos mostrados en el cuadro 7.1. Las series han sido seleccionadas por su diversidad en cuanto a la estacionalidad y tendencia, lo cual nos permite evaluar el comportamiento de nuestras técnicas de predicción bajo diferentes condiciones.

Las series temporales de tráfico marítimo son actualizadas en las bases de datos con la fecha/hora para cada operación portuaria (carga o descarga). A partir de estas series, con granularidad a nivel de minutos, las hemos remuestreado a nivel de trimestre, mes, semana y día; y para cada una de ellas hemos considerado un horizonte de predicción de varios pasos, en la forma definida en la Competición M4 [47]. Con ello hemos conseguido 28 series que cubren la predicción a corto, medio y largo plazo, ayudando a la toma de decisiones estratégicas, de planificación y operación. El *dataset* se encuentra publicado en Mendely Data con autorización de la APBA [43].

Capítulo	Descripción
3	Pescados y crustáceos, moluscos y demás invertebrados acuáticos
7	Hortalizas, plantas, raíces y tubérculos
10	Cereales
16	Preparaciones de carne, pescado o crustáceos, moluscos o demás invertebrados acuáticos
62	Prendas y complementos (accesorios), de vestir, excepto los de punto
69	Productos cerámicos
85	Máquinas, aparatos y material eléctrico, y sus partes; aparatos de grabación o reproducción de sonido, aparatos de grabación o reproducción de imágenes y sonido en televisión, y las partes y accesorios de estos aparatos

Cuadro 7.1: Códigos arancelarios europeos de las series temporales seleccionadas para predicción.

En el cuadro 7.2 mostramos cada granularidad con su correspondiente horizonte de predicción, periodo de estacionalidad e intervalo temporal abarcado. Las series temporales diarias son las que presentan dos tipos de estacionalidad (anual y semanal). Para este tipo de series hemos acortado su intervalo temporal a los últimos dos años y medio, que es una longitud suficiente para nuestros propósitos.

Nivel de granularidad	Periodo estacional	Horizonte de predicción	Intervalo temporal
Trimestral	Año	8	2000 a 2016
Mensual	Año	18	2000 a 2016
Semanal	Año	13	2000 a 2016
Diario	Año/Semana	14	2014-07 a 2016-12

Cuadro 7.2: Características de las series temporales seleccionadas para predicción según su granularidad.

La figura 7.1 muestra las siete series temporales con granularidad mensual y los datos de validación y test para cada una de ellas. Hemos descrito el proceso de validación y test en la subsección 6.1.7, y en la sección 7.2

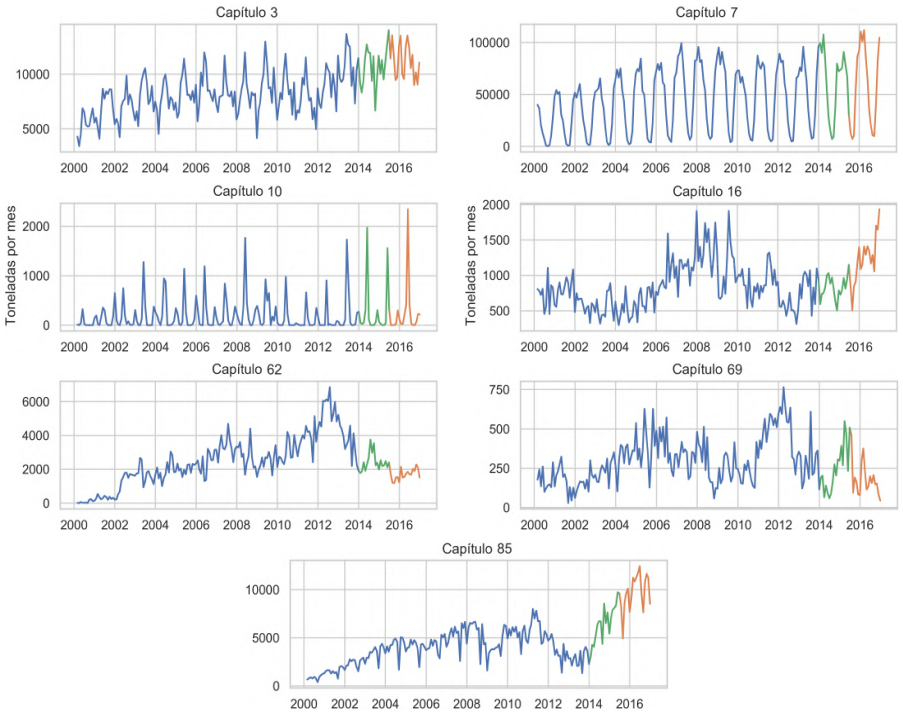

Figura 7.1: Representación gráfica de las series mensuales seleccionadas para predicción. El color verde representa los datos de validación y el naranja los datos de test.

describimos la forma en la que segmentaremos las series para plantear el problema de predicción. Como podemos observar en las gráficas, hay series con una fuerte componente estacional (como el capítulo 7), series intermitentes (como el capítulo 10) y series con una mezcla de tendencia, estacionalidad y ruido (como el capítulo 3).

Las series con valores cero o cercanos a cero pueden causar problemas en el cálculo de las medidas de error (subseccción 3.5.2). Para evitarlo, hemos tratado todas las series de la misma forma, sumando previamente la constante 1 a ellas.

La figura 7.2 nos muestra las gráficas de autocorrelación de las series diarias considerando un intervalo temporal 2000-2016. En ellas se observa qué series tienen estacionalidad semanal, anual o ambas. También, se aprecian los diferentes patrones de comportamiento de las siete series. La estacionalidad semanal parece estar presente en todas las series, incluso en los capítulos 7 y

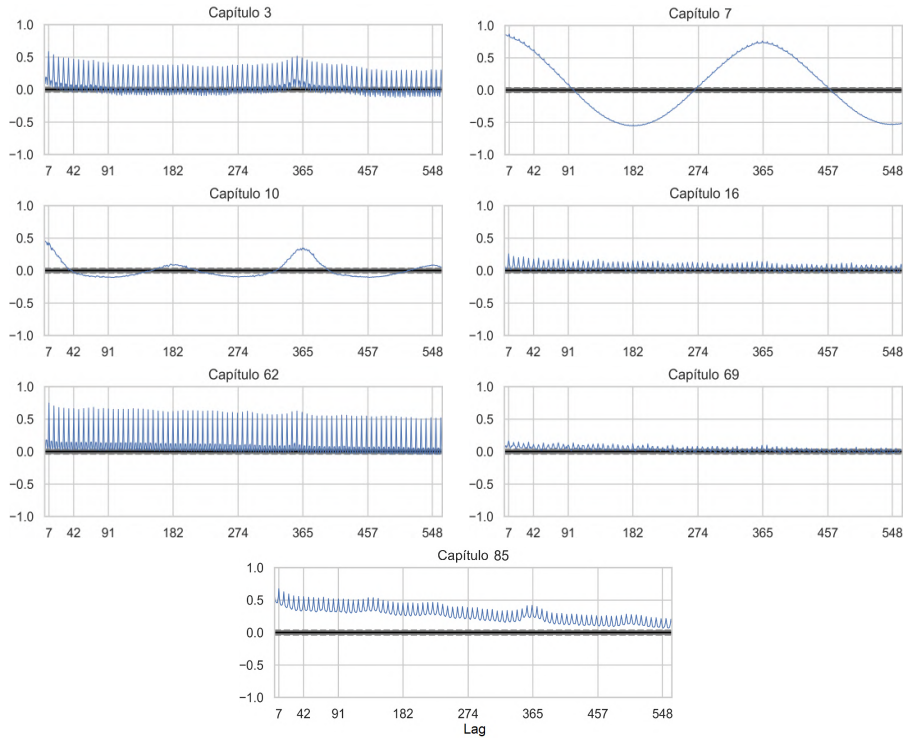

Figura 7.2: Gráficas de autocorrelación de las series diarias seleccionadas para predicción en el periodo 2000-2016.

10, donde aparece superpuesta a la estacionalidad anual. La serie del capítulo 7 presenta una estacionalidad anual muy bien definida. Y el capítulo 10 parece integrar la estacionalidad anual con la semestral.

7.2. Enfoque basado en aprendizaje automático

Recordemos que una serie temporal es una secuencia de observaciones $\mathfrak{o}_t$ tomadas de una variable en sucesivos e igualmente espaciados instantes de tiempo t. Para nuestra tarea de predicción de múltiples pasos la representaremos como

$$S = \{\mathfrak{o}_1, \mathfrak{o}_2, \ldots, \mathfrak{o}_T, \mathfrak{o}_{T+1}, \ldots, \mathfrak{o}_{T+h}, \ldots\}, \tag{7.1}$$

donde $\mathfrak{o}_T$ es la última observación previa a la predicción de los valores observados $\{\mathfrak{o}_{T+1}, ..., \mathfrak{o}_{T+h}\}$, siendo h el horizonte de predicción.

Como sabemos, un algoritmo de aprendizaje automático es capaz de aprender de la experiencia para realizar una tarea gracias a la combinación de un procedimiento de optimización, una función de coste, un modelo y un *dataset*. La predicción de series temporales puede realizarse como una tarea de regresión, donde se le pide a un algoritmo de aprendizaje automático que realice una predicción numérica $\mathbf{y}$ de los próximos h pasos, condicionada a las n observaciones previas $\mathbf{x}$. Para resolver este problema hemos adoptado dos estrategias (figura 7.3), denominadas *multisalida* y recursiva [5]. En la estrategia multisalida el modelo define una función $\mathbb{R}^n \rightarrow \mathbb{R}^h$ para obtener $\mathbf{y}$ directamente a partir de $\mathbf{x}$, en cambio, en la estrategia recursiva el modelo define una función $\mathbb{R}^n \rightarrow \mathbb{R}$ para obtener una predicción de un único valor escalar y a partir de $\mathbf{x}$. En la realización de la siguiente predicción, modificaremos el vector $\mathbf{x}$ eliminando la observación más antigua y añadiendo el valor escalar y de forma que quede temporalmente ordenado. Similarmente, repetiremos este proceso hasta que obtengamos las h predicciones de un solo paso que contendrá el vector $\mathbf{y}$.

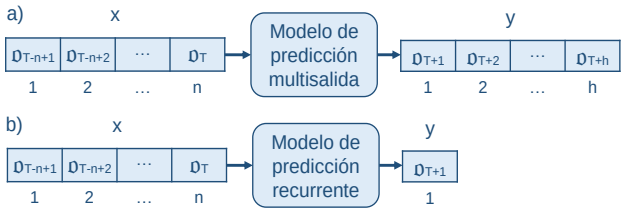

Figura 7.3: Modelos vector a vector de aprendizaje automático para predicción de series temporales. El vector de entrada $\mathbf{x}$ contiene las n observaciones previas a la predicción. El vector de salida $\mathbf{y}$ contiene los valores de la predicción. a) El modelo multisalida obtiene completamente los h (horizonte de predicción) valores de la predicción. b) El modelo recurrente obtiene solo un valor escalar de predicción, que se debe utilizar iterativamente hasta obtener los h valores.

Una vez definido el tipo de estrategia de nuestro modelo, necesitaremos un dataset de ejemplos de entrenamiento con los que nuestro algoritmo de aprendizaje pueda recibir la experiencia en un esquema de aprendizaje supervisado. Los conjuntos de entrenamiento para nuestras dos estrategias de predicción son creados a partir de los valores observados $\{\mathfrak{o}_1, ..., \mathfrak{o}_T\}$, y consisten en dos matrices $\mathbf{X}$ e $\mathbf{Y}$ (figura 7.4), donde cada par de entrenamiento viene dado por la fila i de $\mathbf{X}$ e $\mathbf{Y}$. El objetivo de nuestro modelo es aprender

$$X = \begin{bmatrix} \mathfrak{o}_1 & \mathfrak{o}_2 & \dots & \mathfrak{o}_n \\ \mathfrak{o}_2 & \mathfrak{o}_3 & \dots & \mathfrak{o}_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathfrak{o}_{T-h-n+1} & \mathfrak{o}_{T-h-n+2} & \dots & \mathfrak{o}_{T-h} \end{bmatrix} \quad Y = \begin{bmatrix} \mathfrak{o}_{n+1} & \mathfrak{o}_{n+2} & \dots & \mathfrak{o}_{n+h} \\ \mathfrak{o}_{n+2} & \mathfrak{o}_{n+3} & \dots & \mathfrak{o}_{n+1+h} \\ \vdots & \vdots & \ddots & \vdots \\ \mathfrak{o}_{T-h+1} & \mathfrak{o}_{T-h+2} & \dots & \mathfrak{o}_{T} \end{bmatrix}$$

(a) Multisalida

$$X = \begin{bmatrix} \mathfrak{o}_1 & \mathfrak{o}_2 & \dots & \mathfrak{o}_n \\ \mathfrak{o}_2 & \mathfrak{o}_3 & \dots & \mathfrak{o}_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathfrak{o}_{T-n} & \mathfrak{o}_{T-n+1} & \dots & \mathfrak{o}_{T-1} \end{bmatrix} \quad Y = \begin{bmatrix} \mathfrak{o}_{n+1} \\ \mathfrak{o}_{n+2} \\ \vdots \\ \mathfrak{o}_{T} \end{bmatrix}$$

(b) Recursiva

Figura 7.4: Conjuntos de datos de entrenamiento para las estrategias de predicción multisalida y recursiva.

de los pares cómo predecir $\mathbf{y}$ a partir de $\mathbf{x}$ cuando se use la estrategia multisalida, y como predecir una salida simple y a partir de $\mathbf{x}$ en el caso recursivo. La fortaleza de la estrategia multisalida está en su capacidad de capturar la dependencia entre los valores predichos. En la estrategia recursiva, cada valor predicho depende de la exactitud de los anteriores, acumulándose los errores, por lo que se pueden deteriorar. La ventaja de la estrategia recursiva reside en un conjunto de entrenamiento aumentado en $h - 1$ pares respecto al de la estrategia multisalida. Además, utiliza datos más cercanos a la predicción como entradas durante el entrenamiento. En nuestros experimentos, hemos obtenido mejores resultados utilizando la estrategia multisalida para series diarias y semanales, y la estrategia recursiva para mensuales y trimestrales. Esto tiene sentido debido al menor número de puntos de entrenamiento en series de granularidad trimestral y mensual.

Durante el proceso de aprendizaje, el procedimiento de optimización tiene como objetivo minimizar una función de coste, que proporciona una medida del error en las predicciones del modelo. Esto se consigue ajustando ciertos parámetros del modelo, representados por el vector $\boldsymbol{\theta}$, que definen la función $\mathbf{y} = f(\mathbf{x}, \boldsymbol{\theta})$ del modelo; a partir de ahora, cuando usemos la estrategia de predicción recursiva, consideraremos $\mathbf{y}$ como un vector unidimensional. Entre las posibles funciones de coste, hemos escogido el error absoluto medio (MAE), definido sobre el conjunto completo de pares de entrenamiento $(\mathbf{x}, \mathbf{y})$

del *dataset* de la forma

$$J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \|\mathbf{y}_i - \mathbf{f}(\mathbf{x}, \boldsymbol{\theta})_i\|_1, \tag{7.2}$$

donde $\|.\|_1$ denota la norma L_1. Minimizar la función de coste MAE conduce a una función $f(\mathbf{x}, \boldsymbol{\theta})$ que estima la mediana de $\mathbf{y}$ para cada $\mathbf{x}$ (sección 6.1.9); asumiendo que los pares de entrenamiento son un subconjunto de los infinitos ejemplos de entrenamiento observables de la verdadera distribución generadora de los datos [23]. Hemos utilizado otras funciones de coste específicas de regresión, pero los resultados obtenidos han sido peores. Esto es consistente con los recientes trabajos de L. Chai et al. (2019) [9], donde los autores concluyen que la función de coste MAE puede alcanzar mejores capacidades de robustez y generalización que el error cuadrático medio (MSE); y J. Qi et al. (2020) [58], sobre las ventajas del MAE en el contexto de las redes neuronales profundas basadas en regresión vector a vector.

Como sabemos, debido a la complejidad de los modelos de redes neuronales, estos se suelen optimizar mediante un método iterativo como el del gradiente descendente, que ofrece la posibilidad de controlar el sobreentrenamiento durante la optimización mediante la técnica *early stopping* (subsección 6.1.6). Con este fin hemos separado el 33 % de los pares de entrenamiento de forma que no influyan en la optimización. El valor de la función de coste $J(\boldsymbol{\theta})$ se calcula aparte para estos pares también, salvando los parámetros $\boldsymbol{\theta}$ cada vez que se alcanza un nuevo mínimo en la función de coste para ellos. Una vez que la optimización se ha completado, los valores salvados se utilizan para definir el modelo.

También hemos descompuesto las series temporales en las porciones de entrenamiento, validación y test, junto con sus respectivos conjuntos de entrenamiento (subsección 6.1.7). En nuestra tarea concreta de predicción de series temporales, los datos de test se usan para evaluar la exactitud del modelo entrenado a partir de los datos anteriores a la predicción (figura 7.5a). Pero previamente es necesario ajustar los hiperparámetros del modelo. Para ello, los datos de entrenamiento se dividen en validación y un nuevo conjunto de entrenamiento (figura 7.5b), con los que el modelo se entrena repetidamente hasta que los hiperparámetros son optimizados para la predicción de los datos de validación. Hemos considerado una longitud de los datos de test y validación igual al horizonte de predicción, que es lo que necesitamos para nuestra tarea.

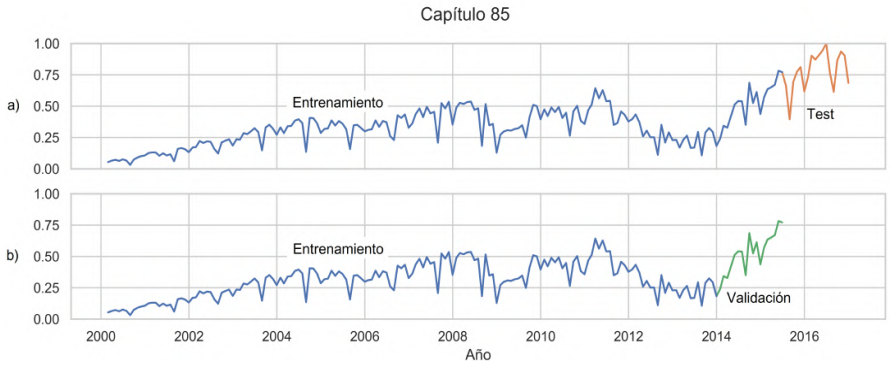

Figura 7.5: Datos de entrenamiento, validación y test durante: a) el proceso de test o prueba; b) el proceso de validación.

Capítulo 8

Predicción con redes convolucionales 1D

La competición de reconocimiento de imágenes Imagenet en 2012 supuso un hito en las redes neuronales convolucionales (*CNN*; *convolutional neural networks*) después de lograr la victoria el modelo convolucional AlexNet, diseñado por A. Krizhevsky et al. [40]. En años anteriores este tipo de redes fueron abandonadas en gran medida por las principales comunidades de visión artificial y aprendizaje automático, pero en Imagenet este modelo logró unos resultados espectaculares, reduciendo casi a la mitad las tasas de error de las mejores soluciones de la competencia. La profundidad del modelo fue esencial para la victoria, que se logró con ayuda de las unidades de procesamiento gráficas (*GPU*; *graphics processing unit*), con las que se aumentó notablemente el rendimiento durante el proceso de entrenamiento. Hoy en día son el tipo de redes utilizadas para casi todas las tareas de reconocimiento y detección.

Una red neuronal convolucional [1] es un tipo de red neuronal en la que las neuronas están organizadas por bloques, resultando en una especialización similar a la que se encuentra en las neuronas del cortex visual de un cerebro biológico. Aunque los conceptos originarios acerca de las CNN están inspirados en el reconocimiento de imágenes, este tipo de modelos se aplica con buenos resultados a otras tareas.

La estructura jerárquica de las CNN las hace menos densas y menos propensas al sobreentrenamiento que el modelo típico MLP de capas completamente conectadas (densas). Las CNN suelen incluir dos tipos de capas:

- Capa convolucional. Aplica filtros por medio de operaciones que involucran a una región de los datos de entrada. En el procesamiento de imágenes, por ejemplo, estos filtros permiten que ciertas características

se resalten, como bordes, líneas verticales, etc.

- Capa de agrupamiento (*pooling*). Usualmente se aplica después de una capa de convolución y resume, mediante la reducción de la dimensionalidad, las características extraídas por los filtros convolucionales.

Normalmente se completa la arquitectura de una CNN con una capa final densa, que realiza el razonamiento de alto nivel de la red a partir de las características obtenidas por las capas previas convolucionales y de agrupamiento.

Las CNN diseñadas para el procesamiento de imágenes tratan datos 2D, pero también pueden definirse para cualquier otra dimensión. Las CNN 1D, por ejemplo, procesan datos secuenciales como series temporales [26]. En cuanto al entrenamiento, la propagación de gradientes hacia atrás es tan simple como en una red MLP [42], permitiendo la optimización de todos los pesos de todos los filtros.

En las próximas secciones, vamos a exponer nuestro modelo de aprendizaje profundo para la predicción de series temporales, basado en capas convolucionales 1D. Para ello, comenzaremos explicando desde la convolución a nivel de una unidad neuronal, hasta llegar a una capa con las características necesarias para la predicción, denominada capa convolucional causal dilatada. También trataremos la capa de agrupamiento, y finalmente, detallaremos el diseño del modelo convolucional DCCNN que hemos usado en nuestros experimentos.

8.1. De la convolución a la capa convolucional causal dilatada

Las redes convolucionales realizan la operación de convolución a nivel de cada unidad neuronal, siendo habitual en la práctica sustituir esta operación por la correlación cruzada de un vector de entrada $\mathbf{x}$ y un vector $\mathbf{k}$, llamado kernel o filtro, que debe ser ajustado por el algoritmo de aprendizaje. Por tanto, podemos definir el vector de salida $\mathbf{y}$ de una unidad convolucional 1D como

$$y_t = (\mathbf{x} \star \mathbf{k})_t = \sum_i x_{t+i-1} k_i, \tag{8.1}$$

donde $\star$ denota el operador de correlación cruzada. Cada y_t se puede interpretar como una medida de similitud entre una región de la entrada $\mathbf{x}$ (comenzando en x_t) y el filtro $\mathbf{k}$, obtenida mediante el producto interno. La salida $\mathbf{y}$ suele llamarse mapa de características.

Es interesante destacar que a efectos prácticos es indiferente considerar la operación de convolución o correlación cruzada; esto es, voltear cualquiera de los dos vectores ($\mathbf{x}$ o $\mathbf{k}$), habida cuenta de que el orden en que se procesan no influye en el algoritmo de optimización, que al final ajustará los parámetros del filtro $\mathbf{k}$ adecuadamente para minimizar el error. En definitiva, la operación de la unidad neuronal para obtener cada y_t consiste en sumar los productos de todos pares de elementos diferentes formados por un elemento de una región de $\mathbf{x}$ y otro de $\mathbf{k}$.

En un problema de predicción de series temporales, cualquier valor de salida y_t no puede depender de los futuros valores x_{t+i}, por lo que se debe utilizar otra forma de convolución, denominada causal, donde cada región implicada de $\mathbf{x}$ termina en x_t, definiéndose como

$$y_t = \sum_{i=0}^{l-1} x_{t-i} k_{l-i}, \tag{8.2}$$

donde l denota la longitud del filtro k. Además, para que los vectores de salida y entrada tengan el mismo tamaño, o sea, que se pueda obtener un y_t por cada x_t, se consideran cero los elementos x_{t-i} no definidos. En la figura 8.1 se muestra la implementación de esta operación.

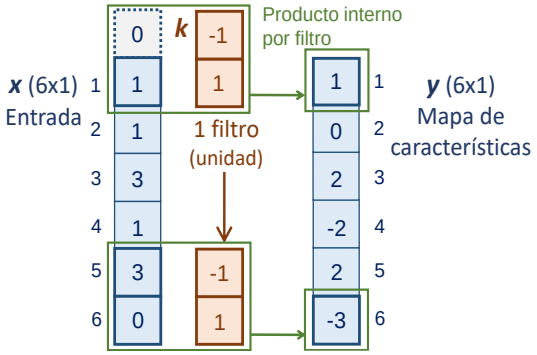

Figura 8.1: Operación de una unidad convolucional causal 1D. El producto interno de una región de la entrada (no futura) y el vector filtro se calcula para cada instante de tiempo (*zancada* = 1), rellenando con ceros las posiciones anteriores del vector de entrada no definidas. Cada característica es almacenada en el vector de salida.

Una capa convolucional 1D está compuesta para varias unidades que operan de la misma forma sobre el vector de entrada, produciendo una matriz de salida con una columna por cada unidad f:

$$Y_{t,f} = \sum_{i=0}^{l-1} x_{t-i} K_{l-i,f}. \tag{8.3}$$

La figura 8.2 muestra la operación de esta capa que procesa un vector de entrada $\mathbf{x}$.

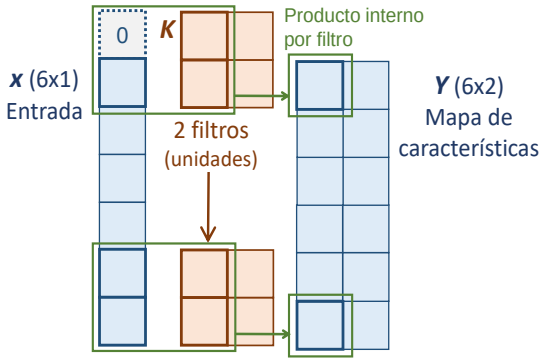

Figura 8.2: Operación de una capa convolucional causal 1D con un vector de entrada. Dos unidades generan una salida matricial de dos columnas.

Suponiendo que el vector $\mathbf{x}$ tenga una longitud n y que la capa convolucional contenga m unidades, la salida de esta capa sería una matriz $\mathbf{Y}$ de tamaño nxm. Como es muy común trabajar con una pila de capas convolucionales, la siguiente capa debería ser capaz de procesar una entrada matricial. Esta capa procesaría una matriz $\mathbf{X}$ de tamaño nxm para obtener una matriz de salida $\mathbf{Y}$ de tamaño nxm, en la cual cada característica $Y_{t,f}$ se calcularía usando una región de la matriz $\mathbf{X}$, terminada en la fila t, y la matriz $\mathsf{K}_{:,:,f}$[1] correspondiente al filtro f:

$$Y_{t,f} = \sum_{i=0}^{l-1} \sum_{j} X_{t-i,j} \mathsf{K}_{l-i,j,f}, \tag{8.4}$$

donde K denota el tensor de filtros. La figura 8.3 muestra la operación de esta capa.

Para tratar las dependencias largas en las series temporales, una solución es aumentar el tamaño de la región de entrada que influye en las características (campo receptivo). Esto puede lograrse mediante la *dilatación* de los

[1]La notación (:) se utiliza como sustituto de cualquier valor para un determinado índice. En este caso, al usarse para dos índices, denota una matriz.

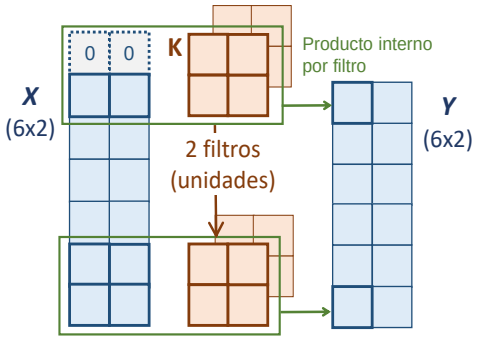

Figura 8.3: Operación de una capa convolucional causal 1D con una matriz de entrada. Dos unidades generan una salida matricial de dos columnas.

filtros, que consiste en que se apliquen sobre una región mayor que su propio tamaño saltándose valores de la entrada con una cierta tasa r:

$$Y_{t,f} = \sum_{i=0}^{l-1} \sum_{j} X_{t-ir,j} \mathsf{K}_{l-i,j,f}. \tag{8.5}$$

La figura 8.4 muestra el mapa de características obtenido por una capa convolucional causal dilatada, y cómo la tasa de dilatación ($r = 2$) permite aumentar el campo receptivo. Una pila de capas con una tasa de dilatación con crecimiento exponencial da lugar a un crecimiento del campo receptivo exponencial. Podemos apreciar este crecimiento en la figura 8.7 de la sección 8.3, donde expondremos nuestro modelo de predicción convolucional.

Para completar la capa convolucional causal dilatada, debemos añadir por cada filtro un sesgo b_f entrenable, y finalmente, una función de activación ϕ que usualmente es la función de activación rectificador:

$$Y_{t,f} = \phi(\sum_{i=0}^{l-1} \sum_{j} X_{t-ir,j} \mathsf{K}_{l-i,j,f} + b_f). \tag{8.6}$$

8.2. La capa de agrupamiento (*pooling*)

En la sección anterior hemos visto cómo cada unidad solo puede procesar a la vez una parte de los datos de entrada, con los que está localmente conectada. Con ello se consigue detectar pequeñas características, a la vez que se reduce el número de pesos de cada neurona, disminuyendo los requerimientos de memoria y evitando el sobrentrenamiento (modelos más simples).

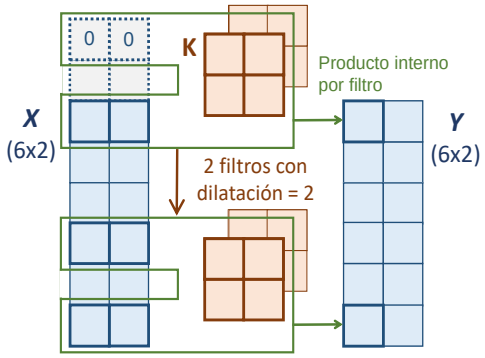

Figura 8.4: Operación de una capa convolucional causal dilatada 1D con tasa de dilatación $r = 2$.

Además, los pesos de cada unidad son compartidos (están atados) entre las distintas regiones involucradas, pudiéndose localizar una cierta característica en diferentes posiciones del vector de entrada. El uso de pesos compartidos también es una forma de evitar el sobrentrenamiento.

Después de una o varias capas convolucionales se suele añadir una capa de agrupamiento cuya función es unir un grupo de características semánticamente similares en una sola, por ejemplo, con la computación del máximo de estas características. De esta forma se logra algo muy interesante, que las características sean invariantes a pequeñas traslaciones de los datos de entrada, lo cual es muy útil cuando nuestro interés está en detectar una determinada característica y no su posición exacta en los datos de entrada. En el contexto de la predicción de series temporales quizás sea más importante conservar esta información sobre la posición de las características.

La capa de agrupamiento reduce el número de características, simplificando el modelo, y también aumentando el campo receptivo. En la figura 8.5 se muestra una red convolucional causal, en la que cada capa permite ampliar en uno el campo receptivo, lo cual puede suponer un problema por la gran cantidad de niveles y parámetros de la red a entrenar si deseamos que el campo receptivo abarque completamente los datos de entrada. En cambio, en la figura 8.6 se muestra el efecto de introducir una capa de agrupamiento que resume dos características en una sola, lográndose la reducción de parámetros y capas para alcanzar el mismo campo receptivo.

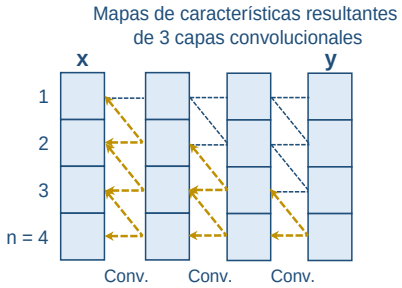

Figura 8.5: Red neuronal convolucional causal sin capa de agrupamiento. Cada capa contiene un solo filtro de longitud 2 y zancada 1. En amarillo se resalta la necesidad de 3 capas para que el campo receptivo abarque el vector de entrada.

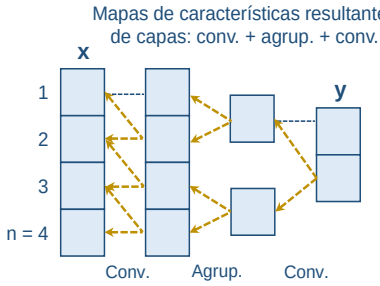

Figura 8.6: Red neuronal convolucional causal con capa de agrupamiento. Cada capa convolucional contiene un solo filtro de longitud 2 y zancada 1. La capa de agrupamiento reduce a la mitad el primer mapa de características, y también, los parámetros y el número de capas convolucionales necesarias para que el campo receptivo abarque el vector de entrada.

8.3. Red neuronal convolucional causal dilatada (*DCCNN*)

En esta sección presentamos nuestro modelo DCCNN (*dilated causal convolutional neural network*) de predicción de series temporales basado en redes convolucionales. Este modelo de aprendizaje profundo aprovecha las nuevas arquitecturas utilizadas por A. V. D. Oord et al. [54] en el diseño de *Wavenet*, una red neuronal profunda para la generación de audio que está basada en una pila de capas convolucionales causales dilatadas que permiten tratar adecuadamente las dependencias temporales largas en una secuencia de datos en bruto.

Nuestro modelo DCCNN define una función vector a vector, $\mathbf{y} = f(\mathbf{x})$, a partir de los n valores previos de una serie temporal para generar la predicción de los siguientes h valores. El modelo es entrenado para estimar la mediana de $\mathbf{y}$ condicionada a $\mathbf{x}$, como explicamos en la sección 6.1.9, y se implementa mediante una pila de capas convolucionales como las definidas en la ecuación (8.6). La pila contiene c capas, cada una compuesta de m unidades (filtros) con una tasa de dilatación r empezando en 1 y duplicándose en cada capa sucesiva. La primera capa mapea el vector de entrada $\mathbf{x}$ sobre un mapa de características nxm. Las siguientes capas toman como entrada el mapa de características de la capa anterior para producir el nuevo, en el que las características están influenciadas por un campo receptivo que es consecuentemente duplicado cada vez, empezando en una longitud 2 (figura 8.7). Hemos determinado el número de capas c para que la longitud del campo receptivo no exceda la longitud del vector de entrada $\mathbf{x}$:

$$c = \lfloor \log_2 n \rfloor, \tag{8.7}$$

donde $\lfloor . \rfloor$ denota la operación parte entera. Una vez que el vector de entrada $\mathbf{x}$ ha sido procesado por la pila de capas convolucionales, el mapa de características obtenido es transformado en un vector de longitud nxm. Entonces, una capa densa de h unidades lo procesa para generar la predicción de múltiples pasos, quedando almacenada en el vector de salida $\mathbf{y}$.

Es importante destacar que la dilatación reduce significativamente el número de capas convolucionales, no siendo necesario añadir capas de agrupamiento. El número de filtros m de las capas convolucionales es un hiperparámetro que hemos ajustado durante el proceso de validación del modelo.

La metodología utilizada para esta red neuronal se desarrolla en el capítulo 10, y los resultados experimentales así como su comparación con diferentes modelos de referencia se detallan en el capítulo 11.

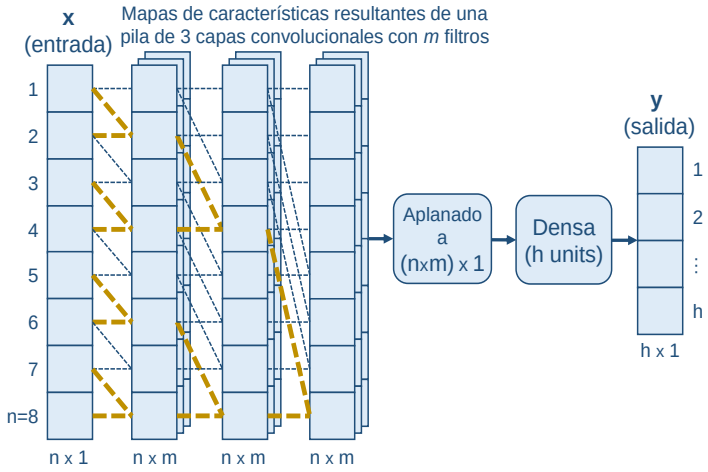

Figura 8.7: Modelo de predicción DCCNN. Sin pérdida de generalidad, hemos considerado un vector de entrada $\mathbf{x}$ de longitud $n = 8$. Cada capa convolucional (omitida por claridad) contiene m filtros de 2 filas que dan lugar a un nuevo mapa de características. Las líneas amarillas muestran como un aumento en potencias de 2 de la tasa de dilatación conlleva un aumento igual del campo receptivo. El último mapa de características es aplanado para su procesamiento por la capa densa de salida.

Capítulo 9

Predicción con redes recurrentes

Las redes neuronales recurrentes (*RNN*) [50] son otra de las principales arquitecturas en el catálogo actual de los modelos de redes neuronales profundas. Están básicamente caracterizadas por tener capas que, además de recibir información de la capa previa de la red, también la reciben de su propio estado en la iteración previa, o sea, tienen ciclos y por tanto memoria de estado. Esto las hace dependientes del tiempo y provoca que el orden en que los datos son proporcionados influya directamente sobre los resultados obtenidos, algo apropiado cuando se trabaja con series temporales. En la memoria de estado se mantiene información sobre la historia de todos los elementos de la secuencia procesados anteriormente, lo cual hace que sean un tipo de red muy potente.

En las próximas secciones, vamos a exponer nuestro modelo de aprendizaje profundo para la predicción de series temporales basado en una RNN. Comenzaremos explicando la evolución desde la capa densa hasta llegar a la unidad recurrente con puertas (*gated recurrent unit; GRU*). Después trataremos el aprendizaje en redes recurrentes, y más concretamente la enseñanza forzada. Finalmente, detallaremos el diseño del modelo recurrente EDRNN que hemos usado en nuestros experimentos, basado en un codificador-decodificador.

9.1. De la capa densa a la capa GRU

Una típica capa densa (totalmente conectada) contiene m unidades neuronales que mapean un vector de entrada $\mathbf{x}$ d-dimensional a un vector de salida $\mathbf{y}$ m-dimensional, quedando definida por

$$\mathbf{y} = \phi(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{9.1}$$

donde ϕ es la función de activación que posibilita la no linealidad, $\mathbf{W}$ es la matriz de pesos mxd entrenable, y $\mathbf{b}$ un vector de sesgos m-dimensional entrenable (figura 9.1).

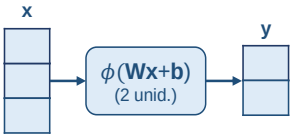

Figura 9.1: Capa densa.

Si realimentamos la salida hacia la entrada obtenemos una capa RNN que puede operar sobre una secuencia de entradas $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\tau$ que genera una secuencia de salidas *ocultas* $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_\tau$, en la que cada salida $\mathbf{h}_t$ depende de la entrada $\mathbf{x}_t$ (en el mismo instante de tiempo) y la salida $\mathbf{h}_{t-1}$ (del instante anterior):

$$\mathbf{h}_t = \phi(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}), \tag{9.2}$$

donde ϕ es usualmente la función de activación tanh y $\mathbf{U}$ es una matriz de pesos mxm entrenable (figura 9.2).

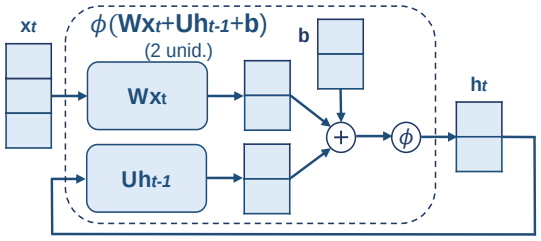

Figura 9.2: Capa recurrente.

Considerando la capa RNN como un sistema dinámico, $\mathbf{h}_t$ representaría el estado en el instante de tiempo t, que puede ser usado para almacenar información sobre las activaciones disparadas por los datos anteriormente procesados. El entrenamiento de una RNN usando un método basado en el gradiente descendente presenta el principal problema de que el error tiende a hacerse cero en su propagación hacia atrás. En 1997 S. Hochreiter y J. Schmidhuber [30] proporcionaron un remedio a este problema con la arquitectura *Long Short-Term Memory* (*LSTM*), consiguiendo un flujo constante del error mediante el uso de mecanismos de puertas para controlar paso de la información. Posteriormente, en 2014 K. Cho et al. [11] propusieron la

arquitectura *Gated Recurrent Unit* (*GRU*), basada en este mismo concepto, y estructuralmente similar, pero más simple y usualmente más rápida que una LSTM, alcanzando resultados similares [14].

En una capa GRU el estado de salida $\mathbf{h}_t$ es una mezcla del estado anterior $\mathbf{h}_{t-1}$ y el estado candidato $\tilde{\mathbf{h}}_t$, gobernada en cada dimensión por la puerta de actualización (*update gate*) $\mathbf{u}_t$,

$$\mathbf{h}_t = \mathbf{u}_t \odot \tilde{\mathbf{h}}_t + (1 - \mathbf{u}_t) \odot \mathbf{h}_{t-1}, \tag{9.3}$$

donde $\odot$ denota el producto Hadamard, que permite que un valor cercano a 1 en una dimensión de la puerta $\mathbf{u}_t$ cause una actualización de esta dimensión en el estado de salida $\mathbf{h}_t$ al valor de esa dimensión en el estado candidato $\tilde{\mathbf{h}}_t$. Similarmente, un valor cercano a 0 causa una actualización del estado de salida desde el estado previo, permitiendo recordar información útil para el futuro (figura 9.3).

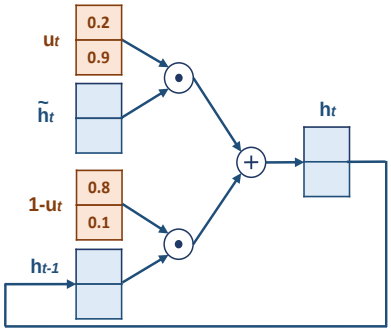

Figura 9.3: Salida de una capa GRU a partir del estado anterior y el estado candidato.

El valor de la puerta de actualización $\mathbf{u}_t$ se obtiene a partir de la entrada actual $\mathbf{x}_t$ y el estado de salida anterior $\mathbf{h}_{t-1}$:

$$\mathbf{u}_t = \sigma(\mathbf{W}_u \mathbf{x}_t + \mathbf{U}_u \mathbf{h}_{t-1} + \mathbf{b}_u), \tag{9.4}$$

donde σ es la función sigmoide logística, $\mathbf{W}_u$ y $\mathbf{U}_u$ son matrices de pesos entrenables, y $\mathbf{b}_u$ es un vector de sesgos entrenable.

La salida candidata $\tilde{\mathbf{h}}_t$ se calcula de forma similar al estado de salida en una capa RNN típica, pero usando la puerta de borrado $\mathbf{r}_t$ para descartar selectivamente parte la información contenida en el estado previo $\mathbf{h}_{t-1}$ (figura 9.4):

$$\tilde{\mathbf{h}}_t = \phi(\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}). \tag{9.5}$$

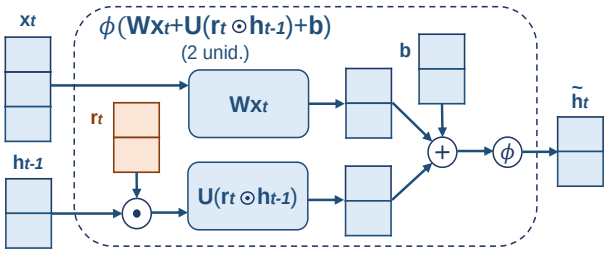

Figura 9.4: Salida candidata de una capa GRU.

El valor de la puerta de borrado $\mathbf{r}_t$ se obtiene de forma similar al de la puerta de actualización:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r). \tag{9.6}$$

Como un único bloque, la capa GRU es capaz de preservar información relevante para el futuro, olvidando lo que no se considera necesario.

9.2. Enseñanza forzada (*teacher forcing*)

Una capa recurrente puede desplegarse como una secuencia de capas hacia adelante, en la que se comparten los parámetros, y la salida de cada capa es la entrada de la siguiente. Por ejemplo, el desplegado de la capa recurrente de la ecuación 9.2 daría lugar a una red profunda como se muestra en la figura 9.5. Este tipo de gráfico proporciona una descripción explícita de los cálculos a realizar y cómo fluye la información en el tiempo.

Vista como una red hacia adelante, el entrenamiento de una RNN se puede hacer con el algoritmo de propagación hacia atrás (*BP*), que aplicado a una red desplegada se denomina de propagación hacia atrás en el tiempo (*BPTT*; *backpropagation through time*). Mediante los parámetros compartidos en cada capa de esta red se logra una mejor generalización con datos no vistos anteriormente, pudiéndose utilizar un *dataset* de entrenamiento de menor tamaño.

En las redes recurrentes donde la salida es realimentada hacia el modelo se puede realizar el entrenamiento con el procedimiento de *aprendizaje forzado* (*teacher forcing*) [70], en el cual se realimentan los valores verdaderos de las predicciones en lugar de las predicciones. Es decir, el modelo aprenderá a minimizar el error de la predicción en un instante t condicionada al valor actual de la entrada y el valor anterior verdadero de predicción. Un ejemplo

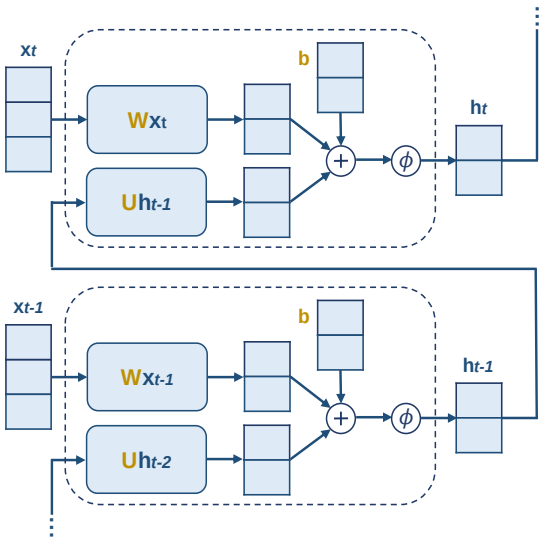

Figura 9.5: Desplegado de una capa recurrente en una secuencia de capas hacia adelante. En amarillo se muestran los parámetros compartidos por las capas.

del conjunto de datos de entrenamiento para aprendizaje forzado, específico para el modelo decodificador que exponemos en la siguiente sección, lo podemos ver en el cuadro 9.7b).

La ventaja del aprendizaje forzado está en que la optimización de parámetros en el entrenamiento converge más rápidamente. Esto se debe a que al principio del entrenamiento, cuando los pesos no están ajustados, se comenten grandes errores en la predicciones, que sin el uso de esta técnica serían realimentados hacia el modelo, acumulándose y por tanto dificultando el aprendizaje. El inconveniente que tiene es que en la prueba real de modelo las salidas sí son realimentadas, y esa diferencia entre entrenamiento y prueba puede llevar a un peor rendimiento e inestabilidad del modelo.

9.3. Red neuronal recurrente codificador-decodificador (*EDRNN*)

Este modelo de aprendizaje automático está basado en la arquitectura codificador-decodificador propuesta en 2014 de forma independiente por K. Cho [11] e I. Sutskever [66] para el mapeo secuencia a secuencia en la solu-

ción a problemas de traducción automática de textos. Utilizando esta misma arquitectura, A. Suilin consiguió ganar en 2017 [65] la competición de predicción de tráfico web organizada por Kaggle. También ha sido utilizada exitosamente, para la predicción de ventas en Amazon, por D. Salinas et al. en 2020, con el modelo probabilístico autoregresivo DeepAR [62].

Nuestro modelo EDRNN (*encoder-decoder recurrent neural network*) define un función vector a vector $\mathbf{y} = f(\mathbf{x})$, desde los n valores previos de una serie temporal hacia los h próximos valores de predicción. Este modelo está entrenado para estimar la mediana de $\mathbf{y}$ condicionada a $\mathbf{x}$, como explicamos en la section 6.1.9. La red neuronal está implementada usando capas GRU definidas según las ecuaciones 9.3-9.6 para nuestro caso particular de una secuencia de entradas x_t escalares. El modelo consiste en dos redes neuronales separadas (codificador y decodificador) entrenadas conjuntamente. Esta arquitectura (figura 9.6) permite el procesamiento de secuencias de entrada y salida de diferente longitud. Su diseño y operación se describen a continuación:

- Codificador. Una capa GRU con m unidades, previamente entrenada, procesa el vector de entrada $\mathbf{x} = (x_1, \ldots, x_n)$ en orden temporal, actualizando el estado $\mathbf{h}_t$ para cada valor de entrada. El estado final $\mathbf{c}$ codifica la historia completa de la secuencia de entrada.

- Decodificador. Consiste en una capa GRU con m unidades y una capa densa con 1 unidad. En cada instante de tiempo t la capa densa mapea el estado $\mathbf{h}_t$ de la capa GRU a la salida del decodificador y_t, y la capa GRU mapea el estado previo $\mathbf{h}_{t-1}$ y la salida previa del decodificador y_{t-1} al estado actual $\mathbf{h}_t$. Por tanto, el modelo del decodificador puede definirse como

$$\mathbf{y}_t = g(\mathbf{h}_{t-1}, y_{t-1}). \tag{9.7}$$

  El decodificador se entrena de acuerdo con esta función utilizando el procedimiento de aprendizaje forzado de la sección anterior, con un valor inicial $\mathbf{h}_{t-1}$ igual al estado final $\mathbf{c}$ del codificador, y el valor inicial de y_{t-1} igual al último valor de la entrada x_n del codificador (cuadro 9.7). Para generar la primera predicción y_1 del decodificador comenzamos inicializándolo de la misma forma que hemos visto para el entrenamiento. Entonces, iteramos sucesivamente realimentando cada salida del decodificador como una nueva entrada del decodificador, y cada estado de salida de la capa GRU como nuevo estado de entrada, hasta que se completan las h predicciones consecutivas.

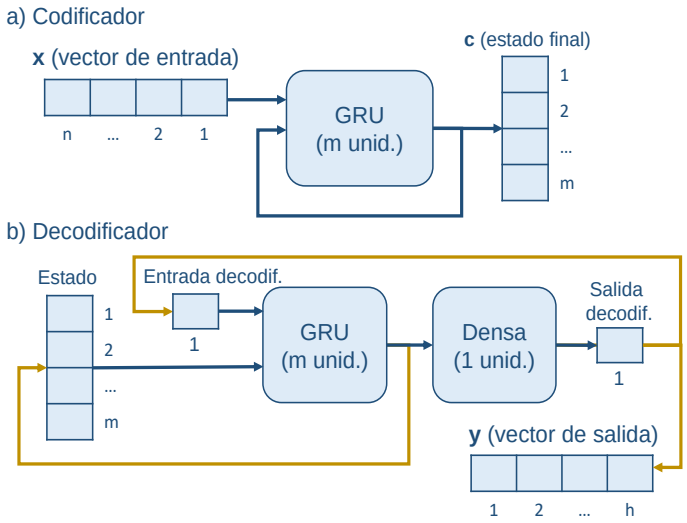

Figura 9.6: Modelo de predicción EDRNN. a) Codificador. El vector de entrada $\mathbf{x}$ es resumido en el vector de estado $\mathbf{c}$. b) Decodificador. El estado se inicializa con el estado final $\mathbf{c}$ del codificador, y la entrada del decodificador con la última entrada x_n del codificador. En amarillo, resaltamos el bucle de predicción donde, para cada iteración, el nuevo valor predicho se almacena en la siguiente componente del vector $\mathbf{y}$, y las salidas de las capas GRU y densa son realimentadas.

$$X_c = \begin{bmatrix} \mathfrak{o}_1 & \mathfrak{o}_2 & \dots & \mathfrak{o}_n \\ \mathfrak{o}_2 & \mathfrak{o}_3 & \dots & \mathfrak{o}_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathfrak{o}_{T-h-n+1} & \mathfrak{o}_{T-h-n+2} & \dots & \mathfrak{o}_{T-h} \end{bmatrix}$$

$$X_d = \begin{bmatrix} \mathfrak{o}_n & \mathfrak{o}_{n+1} & \dots & \mathfrak{o}_{n+h-1} \\ \mathfrak{o}_{n+1} & \mathfrak{o}_{n+2} & \dots & \mathfrak{o}_{n+h} \\ \vdots & \vdots & \ddots & \vdots \\ \mathfrak{o}_{T-h} & \mathfrak{o}_{T-h+1} & \dots & \mathfrak{o}_{T-1} \end{bmatrix} \quad Y_d = \begin{bmatrix} \mathfrak{o}_{n+1} & \mathfrak{o}_{n+2} & \dots & \mathfrak{o}_{n+h} \\ \mathfrak{o}_{n+2} & \mathfrak{o}_{n+3} & \dots & \mathfrak{o}_{n+1+h} \\ \vdots & \vdots & \ddots & \vdots \\ \mathfrak{o}_{T-h+1} & \mathfrak{o}_{T-h+2} & \dots & \mathfrak{o}_T \end{bmatrix}$$

(a) Multisalida

$$X_c = \begin{bmatrix} \mathfrak{o}_1 & \mathfrak{o}_2 & \dots & \mathfrak{o}_n \\ \mathfrak{o}_2 & \mathfrak{o}_3 & \dots & \mathfrak{o}_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathfrak{o}_{T-n} & \mathfrak{o}_{T-n+1} & \dots & \mathfrak{o}_{T-1} \end{bmatrix} \quad X_d = \begin{bmatrix} \mathfrak{o}_n \\ \mathfrak{o}_{n+1} \\ \vdots \\ \mathfrak{o}_{T-1} \end{bmatrix} \quad Y_d = \begin{bmatrix} \mathfrak{o}_{n+1} \\ \mathfrak{o}_{n+2} \\ \vdots \\ \mathfrak{o}_T \end{bmatrix}$$

(b) Recursiva

Figura 9.7: Conjuntos de datos de entrenamiento forzado para las estrategias de predicción multisalida y recursiva del modelo ECRNN. Cada fila de X_c, X_d e Y_d contiene una entrada para el codificador, una entrada para el decodificador y una salida para el decodificador, respectivamente.

El número de unidades m de las dos capas GRU es un hiperparámetro que hemos ajustado en el proceso de validación del modelo. La metodología utilizada para esta red neuronal se desarrolla en el capítulo 10, y los resultados experimentales así como su comparación con diferentes modelos de referencia se detallan en el capítulo 11.

Capítulo 10

Metodología para los modelos de aprendizaje profundo

En este capítulo presentamos nuestra metodología experimental. En primer lugar, describiremos brevemente los métodos de referencia utilizados para la comparación de nuestros resultados de predicción. Después trataremos las métricas de evaluación de los resultados, y finalmente especificaremos los principales aspectos de nuestros escenarios experimentales.

10.1. Métodos de referencia

En esta sección exponemos los métodos de referencia de la Competicion M4 de predicción, que nosotros también utilizaremos para comparar los resultados de esta investigación, y cuyo código fuente está disponible online[1].

En algunos de estos métodos, se realiza un ajuste de la estacionalidad (sección 3.3), eliminándola de las series temporales cuando se supera un test de estacionalidad con un 90 % de nivel de confianza, basado en la función de autocorrelación (ecuación 3.3). En otros métodos también se elimina la tendencia lineal mediante una regresión de mínimos cuadrados. En cualquier caso, una vez que las predicciones se han obtenido, y antes de que sean evaluados los resultados, la tendencia y estacionalidad son restauradas.

Estos métodos de referencia están compuestos por el conjunto de métodos ingenuos del capítulo 3 junto con los métodos de suavizado exponencial del capítulo 4. Además, se han incluido dos métodos sencillos basados exclusivamente en aprendizaje automático, más específicamente en redes neuronales, que son:

[1] https://github.com/M4Competition/M4-methods

- MLP. Perceptrón multicapa de arquitectura y parametrización muy básicas, desarrollado en Python usando la librería Scikit-learn (función `MLPRegressor`). Contiene una sola capa oculta de 6 unidades y una capa de salida con una sola unidad. Todas las capas usan la función de activación identidad. Produce la secuencia de predicciones mediante una estrategia recursiva. Se realizan 29 predicciones a las que se le calcula la mediana como predicción final.

- RNN. Red recurrente de arquitectura y parametrización sencilla desarrollada en Python usando Keras (función `SimpleRNN`) y TensorFlow. Se corresponde con una capa de 6 unidades, como la expuesta en la ecuación 9.2 y figura 9.2, seguida de una capa densa de una unidad con la función de activación identidad, que produce la secuencia de predicciones mediante una estrategia recursiva.

En ambos modelos, MLP y RNN, la longitud de la entrada es 3, y se realiza un ajuste previo de la estacionalidad y tendencia. El cuadro 10.1 muestra un resumen de todos estos métodos de referencia, organizándolos según el procesamiento previo de los datos.

En los resultados de la Competición M4, además de los métodos anteriores, se incluyeron los estándares de comparación ETS y ARIMA, ambos de selección automática del mejor modelo, y que ya tratamos en el capítulo 4. Estos estándares fueron incluidos debido a su utilización extensiva en estudios sobre predicción en los últimos años, así como en anteriores competiciones M. Los resultados reflejaron que los métodos Theta y Comb fueron superiores globalmente a ETS y ARIMA, ambos con selección del mejor modelo específico para cada serie; superando el modelo ARIMA al estándar ETS. Se comprobó, por tanto, que las diferentes fuentes de incertidumbre en los datos, modelos y parámetros hacen de la selección automática del modelo una misión difícil y desafiante. A lo largo de las diferentes competiciones M, los modelos ARIMA han ido mejorando sus resultados, acercándose más a los métodos de referencia. Esto se ha conseguido al identificarse sus carencias y mitigarse sus principales problemas, como el sobreajuste. La elección automática de modelos quedó cuestionada también debido a que estos se basan en minimizar los errores de las predicciones de un solo paso con datos de entrenamiento. Como resultado, es evidente que los modelos óptimos seleccionados estarán lejos de ser los mejores en predicciones de muchos más pasos hacia adelante, y más aún, si se evalúan con una métrica que difiera de la utilizada en el entrenamiento. Queda patente que este tipo de algoritmos de selección automática requieren de investigaciones adicionales que puedan hacerlos más competitivos en el futuro [47].

Métodos estadísticos sin ajustes previos en las series		
1	Naïve 1	Los valores de la predicción son iguales a la última observación antes de la predicción.
2	Naïve S	El último periodo estacional observado antes de la predicción se repite consecutivamente en la predicción.
Métodos estadísticos con ajuste de estacionalidad		
3	Naïve 2	Igual que Naïve 1 pero con ajuste de estacionalidad.
4	SES	Suavizado exponencial simple.
5	Holt	Suavizado exponencial de tendencia lineal de Holt.
6	Damp	Suavizado exponencial de tendencia amortiguada.
7	Comb	Media aritmética de los tres métodos SES, Holt y Damped.
8	Theta	Fue el método ganador de la Competición M3. Utiliza líneas de extrapolación.
Modelos de aprendizaje automático (ajuste de estacionalidad y tendencia)		
9	MLP	Perceptrón multicapa.
10	RNN	Red neuronal recurrente simple.

Cuadro 10.1: Resumen de los métodos de referencia de la Competición M4.

10.2. Métricas de evaluación

En la sección 3.5 hemos clasificado y expuesto las métricas más utilizadas según R. J. Hyndman (2006) [36], entre las que se encuentran las de la Competición M4, que son las mismas que vamos a utilizar en nuestro trabajo:

- sMAPE. Error porcentual absoluto medio simétrico. Debido a los problemas de esta métrica en relación con los valores de las series temporales cercanos o iguales a cero, hemos añadido la constante 1 a todas las series antes de realizar la predicción, y posteriormente se la hemos restado a la predicción antes de evaluar su exactitud. De esta forma también evitamos problemas en el ajuste estacional multiplicativo de las series.

- MASE. Error escalado absoluto medio. Esta métrica utiliza la longitud m del periodo estacional, la cual la hemos considerado igual a un año para las granularidades trimestral, mensual y semanal; y a una semana para la granularidad diaria. Estos valores son los más beneficiosos para los métodos de referencia M4. En series no estacionales no tiene sentido usar estos periodos, por lo cual, para estos casos hemos considerado $m = 1$.

- OWA. Media ponderada global de sMAPE y MASE. Se utiliza para la clasificación de los modelos de predicción. Es una medida muy intuitiva en la que un valor menor que 1 significa que el error del modelo evaluado es menor que el error del método de referencia Naïve 2 (subsección 3.4.3).

10.3. Marco de implementación y configuración

Los modelos propuestos, convolucional (DCCNN) y recurrente (EDRNN), han sido implementados usando la librería Keras [12] con TensorFlow de fondo, y ejecutados en un procesador Intel(R) Core(TM) i7-6700 CPU @3.4GHz de cuatro núcleos con 32GB y una tarjeta gráfica NVIDIA GeForce GTX 1070. El generador de números pseudoaleatorios ha sido inicializado para que los resultados puedan ser reproducibles.

Confiando en las capacidades del aprendizaje profundo hemos trabajado directamente con los datos en bruto de la series temporales, sin realizar ninguna transformación sobre estas, ni ajuste de estacionalidad o tendencia;

solamente se han escalado los datos al rango [0, 1] antes de la entrada a los modelos y posteriormente restaurado en las predicciones.

Hemos seleccionado los hiperparámetros de los modelos exclusivamente en base a los resultados del proceso de validación (sección 7.2), manteniendo aparte los datos de test exclusivamente con el fin de evaluar el rendimiento de los modelos. Debido al gran número de combinaciones posibles en la elección de la mejor configuración de cada modelo, y el consumo en tiempo en cada caso, hemos ajustado la mayoría de hiperparámetros manualmente. Estos son los siguientes:

- Optimizador: Adam
- Función de activación en capas de salida: identidad.
- Tamaño del lote (*batch*): el conjunto completo de datos de entrenamiento.
- Números de épocas (*epochs*): 125 y 525 para DCCNN y EDRNN respectivamente.
- Función de pérdida (*loss*): MAE.
- Interrupción temprana del optimizador (*early stopping*): se ha reservado un 33 % de los datos de entrenamiento para el control del sobreajuste (sección 7.2).
- Longitud de las entradas a los modelos: hemos seleccionado la correspondiente a un año, confirmando la propuesta de C. Hamzaçebi (2008) [25] para series temporales estacionales.
- Otros hiperparámetros del modelo DCCNN que hemos fijado son la zancada de la convolución a 1 y la longitud del filtro a 2.

Solo hemos seleccionado automáticamente un hiperparámetro en cada modelo, que es el siguiente:

- Número de filtros en las capas convolucionales del modelo DCCNN.
- Número de unidades en las capas recurrentes del modelo EDRNN.

El ajuste de estos hiperparámetros, considerando cada modelo y granularidad, lo hemos realizado a partir de las predicciones de todas las series convertidas a un mismo rango [0, 1], y promediando los resultados de su evaluación con la métrica MAE. El cálculo se realiza para cada valor del hiperparámetro, que varía según la granularidad:

- Series diarias: de 1 a 250.
- Otras granularidades: de 1 a 200.

Los resultados se almacenan en un vector, y entonces se calcula una media móvil de orden 7 a partir de este vector, y también una desviación estándar móvil de orden 7, almacenando los resultados en dos vectores denotados por $ma7$ y $std7$ respectivamente. De esta forma disponemos de información sobre la variabilidad de los valores promediados. Finalmente, el valor seleccionado del hiperparámetro es el asociado el valor mínimo de la función $ma7+std7/2$. La figura 10.1 muestra un ejemplo de la gráfica de esta función, obtenida para cada valor del hiperparámetro. El cuadro 10.2 muestra los valores de los hiperparámetros seleccionados automáticamente con este método para todos los modelos y granularidades.

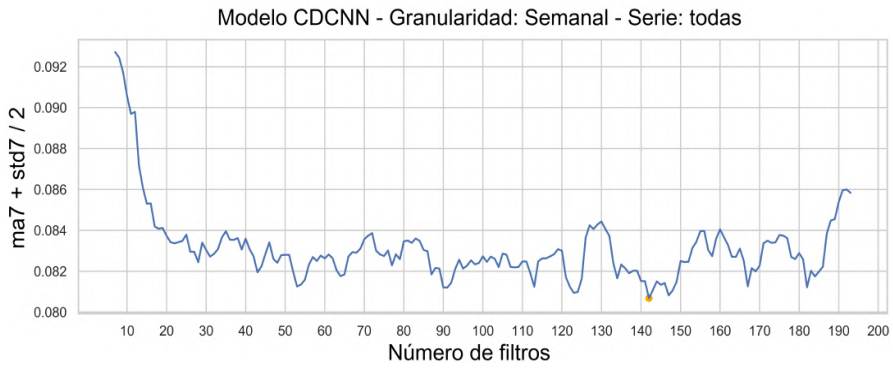

Figura 10.1: Selección automática de hiperparámetros. El número de filtros en las capas convolucionales del modelo DCCNN para series con granularidad semanal se selecciona al valor 142, que es el mínimo de la función $ma7 + std7/2$.

En cuanto a la estrategia de predicción, hemos elegido la recursiva para series con pocos datos de entrenamiento, como las trimestrales y mensuales, consiguiendo un mejor resultado. Entendemos que esto se debe a la ventaja de esta estrategia, respecto de la estrategia multisalida, de disponer de un conjunto de entrenamiento mayor, además de utilizar datos más cercanos a la predicción como entradas durante el entrenamiento (sección 7.2). Para datos con granularidad semanal y diaria, en cambio, hemos podido aprovechar completamente el potencial de la estrategia multisalida, capturando la dependencia entre los valores de la predicción.

Modelo DCCNN			
Número de filtros en las capas convolucionales			
Trimestral	Mensual	Semanal	Diaria
24	52	142	39

Modelo EDRNN			
Número de unidades de las capas recurrentes			
Trimestral	Mensual	Semanal	Diaria
20	137	29	211

Cuadro 10.2: Valores de los hiperparámetros seleccionados para los modelos propuestos de aprendizaje profundo.

Capítulo 11

Resultados experimentales

En este capítulo presentamos los principales resultados de nuestra experimentación. En primer lugar, incluimos una comparativa para mostrar que las dos arquitecturas que hemos propuesto de aprendizaje profundo mejoran los resultados de los métodos de referencia de la Competición M4.

En la segunda sección mostraremos ejemplos gráficos de predicción generados por los dos modelos de redes neuronales, convolucional y recurrente, para los cuatro niveles de granularidad usados en nuestra experimentación.

Con objeto de reducir la variabilidad de los resultados, cada predicción se ha repetido siete veces, y después se ha realizado su promedio antes de calcular la medida del error.

11.1. Resultados de validación y test

En los resultados que presentamos a continuación comparamos nuestros dos modelos, convolucional (DCCNN) y recurrente (EDRNN), frente a los métodos de referencia de la Competición M4 (Naïve 1, Naïve S, Naïve 2, SES, Holt, Damped, Theta, Comb, MLP y RNN). Nuestra principal meta fue batir a estos métodos, si era posible, y este objetivo ha sido cumplido satisfactoriamente.

El método basado en la contribución del capítulo 5, sobre predicción mediante búsqueda de coincidencia (*matching pursuit*), también lo hemos incluido en la comparación de resultados. Este método utiliza un algoritmo iterativo de aproximaciones sucesivas a la serie temporal a predecir, en el que en cada iteración se selecciona de un diccionario la función sinusoidal que mejor se aproxima a la serie. Este método lo hemos incluido con el nombre MP15, por las 15 iteraciones consideradas para el algoritmo.

Además, al igual que en los resultados de la Competición M4, hemos

incluido los estándares de predicción ETS y ARIMA, ambos con selección automática del mejor modelo, y que hemos denominado AETS y AARIMA respectivamente.

El cuadro 11.1 muestra el rendimiento global de nuestros dos modelos, DCCNN y EDRNN, y el algoritmo MP15, considerando todas las series y niveles de granularidad. Con los dos modelos de aprendizaje profundo hemos logrado mejorar el mejor OWA de los métodos de referencia, tanto con datos de validación como de test. El modelo DCCNN mejora entre un 0.74 % y 6.93 % y el modelo EDRNN mejora entre un 4.96 % y 8.93 %. El algoritmo MP15 obtiene resultados globales bastante alejados de los mejores métodos de referencia. El estándar AETS obtiene unos resultados pobres, pero en cambio AARIMA, aunque no consigue mejorar con datos de validación el mejor de los métodos de referencia, con datos de test sí supera globalmente a estos y también a nuestro modelo EDRNN.

Los resultados por cada nivel de granularidad ofrecen una visión más detallada, que podemos observar en los cuadros 11.2 y 11.3 para datos de validación y test respectivamente. Resaltamos principalmente el modelo EDRNN con datos diarios y el DCCNN con datos semanales. En ambos se ha utilizado una estrategia de predicción multisalida que mejora significativamente el mejor OWA de los métodos de referencia en más de un 12 %. También destaca el modelo DCCNN con datos mensuales y una estrategia de predicción recursiva que mejora el mejor OWA de los métodos de referencia. En todo caso excepto para granularidad trimestral, al menos uno de los modelos DCCNN o EDRNN mejora los métodos de referencia. El algoritmo MP15 obtiene los peores resultados con granularidad trimestral y mensual, aunque es destacable su resultado con datos diarios, donde mejora a los métodos de referencia más de un 5 %. Respecto a los estándares AETS y AARIMA, el primero tiene el inconveniente de que no es capaz de procesar periodos de longitud mayor que 24. El segundo obtiene unos buenos resultados solo con datos de test para granularidades trimestrales, mensuales y semanales, pero no tanto con datos diarios.

La estacionalidad ha sido capturada perfectamente por los modelos DCCNN y EDRNN, y también por el algoritmo MP15. Esto queda evidenciado en el modelo DCCNN con datos semanales, que supera ampliamente los mejores métodos de referencia Naïve 2 y Naïve S, con datos de validación y test respectivamente. Estos dos métodos hacen sus predicciones exclusivamente en base a la estacionalidad (Naïve 2) y el último periodo estacional observado (Naïve S). También el modelo EDRNN captura exitosamente la estacionalidad al superar a todos los métodos de referencia con datos diarios, donde todas las series superan el test de estacionalidad semanal. Las gráficas de algunos ejemplos de predicción de ambos modelos en la sección 11.2 muestran

	Validación				Test			
	sMAPE	MASE	OWA	%	sMAPE	MASE	OWA	%
Naïve 1	64.999	1.958	1.226	-41.51 %	62.331	2.109	1.211	-40.51 %
Naïve S	50.910	1.505	0.951	-9.75 %	46.554	1.434	**0.862**	**0.00 %**
Naïve 2	56.182	1.512	1.000	-15.42 %	54.360	1.654	1.000	-16.06 %
SES	51.056	1.248	0.867	-0.11 %	52.401	1.327	0.883	-2.48 %
Holt	52.706	1.363	0.920	-6.16 %	55.279	1.361	0.920	-6.73 %
Damped	51.502	1.274	0.880	-1.54 %	52.388	1.320	0.881	-2.22 %
Theta	50.916	1.249	**0.866**	**0.00 %**	52.372	1.312	0.878	-1.93 %
Comb	51.483	1.290	0.885	-2.12 %	52.616	1.322	0.883	-2.53 %
MLP	55.795	1.526	1.001	-15.58 %	61.680	1.728	1.090	-26.46 %
RNN	50.061	1.279	0.868	-0.23 %	56.112	1.451	0.955	-10.79 %
DCCNN	48.811	1.287	**0.860**	**0.74 %**	46.279	1.245	**0.802**	**6.93 %**
EDRNN	41.860	1.273	**0.794**	**8.39 %**	44.607	1.352	**0.819**	**4.96 %**
MP15	54.520	1.537	**0.994**	**-14.68 %**	50.969	1.542	**0.935**	**-8.49 %**
AETS	55.619	1.396	**0.957**	**-10.44 %**	56.276	1.428	**0.949**	**-10.14 %**
AARIMA	52.319	1.263	**0.883**	**-1.97 %**	48.870	1.200	**0.812**	**5.76 %**

Cuadro 11.1: Resultados globales de validación y test. Se muestra la reducción del error (%) sobre el mejor de los métodos de referencia.

este fenómeno. Como era de esperar, el algoritmo MP15 captura también correctamente la estacionalidad, obteniendo muy buenos resultados con datos diarios, donde es capaz de capturar diferentes tipos de periodicidad, siendo solo superado por el modelo EDRNN.

Con series temporales de datos intermitentes, las cuales se encuentran en todas las granularidades, pero especialmente en la diaria (capítulos arancelarios 10 y 69), los resultados nos llevan a resaltar el modelo EDRNN.

La granularidad trimestral ha sido la única que nuestros modelos no han podido superar en ningún momento. Pensamos que esto se debe al tamaño del *dataset* trimestral, penalizando los modelos de aprendizaje profundo, que necesitan una gran cantidad de datos para ser entrenados adecuadamente. El algoritmo MP15 ha obtenido las peores predicciones con los datos trimestrales. La ocultación de información periódica en datos trimestrales lo penaliza, además de la dificultad que tiene para un algoritmo que se basa en la periodicidad, la realización de predicciones a largo plazo.

11.2. Ejemplos de predicción

Esta sección esta dividida en diferentes subsecciones donde mostraremos algunas de las predicciones de nuestros modelos de redes neuronales DCCNN y EDRNN, y del algoritmo de predicción *matching pursuit* MP15. Las predicciones se han realizado con datos de test para una serie temporal seleccionada

	Trimestral			Mensual			Semanal			Diaria		
	sMAPE	MASE	OWA	sMAPE	MASE	OWA	sMAPE	MASE	OWA	sMAPE	MASE	OWA
Naïve 1	56.319	2.322	1.162	58.706	2.287	1.356	59.454	1.202	1.406	85.519	2.019	1.087
Naïve S	51.429	2.169	1.074	45.763	1.432	0.945	50.687	1.136	1.264	55.760	1.283	**0.699**
Naïve 2	52.278	1.862	1.000	46.727	1.571	1.000	42.438	0.853	**1.000**	83.285	1.760	1.000
SES	45.257	1.787	0.912	40.165	1.284	0.839	47.635	0.838	1.053	71.169	1.085	0.736
Holt	52.541	2.214	1.097	40.660	1.358	0.867	47.789	0.845	1.058	69.834	1.034	0.713
Damped	47.622	1.924	0.972	39.611	1.262	**0.825**	47.678	0.842	1.055	71.098	1.069	0.731
Theta	44.598	1.771	0.902	40.300	1.301	0.845	47.630	0.839	1.053	71.134	1.087	0.736
Comb	47.529	1.963	0.982	40.044	1.295	0.840	47.701	0.841	1.055	70.659	1.059	0.725
MLP	42.913	1.893	0.919	45.762	1.608	1.001	57.495	1.175	1.366	77.010	1.429	0.868
RNN	40.265	1.642	**0.826**	44.627	1.531	0.965	45.805	0.872	1.051	69.545	1.068	0.721
DCCNN	51.970	2.168	**1.079 -30.6%**	35.950	1.202	**0.767 7.1%**	39.194	0.698	**0.871 12.9%**	68.130	1.079	**0.716 -2.3%**
EDRNN	48.718	2.070	**1.022 -23.7%**	34.882	1.248	**0.771 6.6%**	41.763	0.779	**0.949 5.1%**	42.076	0.996	**0.536 23.4%**
MP15	68.764	2.768	**1.401 -55.3%**	48.344	1.585	**1.022 -23.8%**	43.134	0.689	**0.912 8.8%**	57.838	1.106	**0.661 5.4%**
AETS	51.309	2.090	**1.052 -16.6%**	42.355	1.310	**0.870 -5.4%**	52.744	0.952	**1.180 -18.0%**	76.069	1.233	**0.807 -15.4%**
AARIMA	47.209	1.821	**0.940 -4.3%**	44.945	1.355	**0.912 -10.5%**	44.220	0.749	**0.960 4.0%**	72.900	1.128	**0.758 -8.4%**

Cuadro 11.2: Resultados de validación por nivel de granularidad. Se muestra la reducción del error (%) sobre el mejor de los métodos de referencia.

	Trimestral			Mensual			Semanal			Diaria		
	sMAPE	MASE	OWA	sMAPE	MASE	OWA	sMAPE	MASE	OWA	sMAPE	MASE	OWA
Naïve 1	48.339	2.628	1.290	58.402	2.410	1.545	66.643	1.865	0.979	75.940	1.534	1.157
Naïve S	39.012	2.032	1.018	41.322	1.446	1.000	44.657	0.939	**0.581**	61.224	1.319	0.967
Naïve 2	39.361	1.943	1.000	43.536	1.378	1.000	62.988	2.073	1.000	71.556	1.224	1.000
SES	35.838	1.725	0.899	42.589	1.269	0.950	51.281	1.297	0.720	79.893	1.017	0.974
Holt	48.496	1.800	1.079	42.132	1.212	**0.924**	51.037	1.289	0.716	79.449	1.141	1.021
Damped	35.706	1.671	**0.884**	41.963	1.225	0.927	51.253	1.296	0.719	80.630	1.087	1.007
Theta	35.945	1.680	0.889	42.819	1.264	0.950	51.083	1.289	0.716	79.643	1.016	0.972
Comb	37.098	1.687	0.905	42.095	1.226	0.928	51.191	1.294	0.718	80.078	1.081	1.001
MLP	44.774	2.381	1.181	52.707	1.738	1.236	61.403	1.562	0.864	87.835	1.232	1.117
RNN	44.596	2.236	1.142	47.765	1.467	1.081	50.908	1.128	0.676	81.179	0.973	**0.965**
DCCNN	37.467	1.985	**0.987 -11.7%**	38.072	1.314	**0.914 1.1%**	40.644	0.779	**0.510 12.1%**	68.932	0.902	**0.850 11.9%**
EDRNN	41.651	2.289	**1.118 -26.5%**	41.690	1.407	**0.989 -7.1%**	51.410	1.009	**0.652 -12.1%**	43.679	0.703	**0.592 38.6%**
MP15	52.357	2.845	**1.397 -58.1%**	48.486	1.624	**1.146 -24.1%**	45.91	0.93	**0.589 -1.3%**	57.122	0.768	**0.713 26.3%**
AETS	34.392	1.494	**0.821 7.0%**	41.515	1.314	**0.954 -3.2%**	60.315	1.606	**0.866 -49.1%**	88.881	1.296	**1.150 -19.0%**
AARIMA	33.795	1.640	**0.851 3.7%**	38.271	1.265	**0.899 2.7%**	43.290	0.803	**0.537 7.5%**	80.122	1.090	**1.005 -4.0%**

Cuadro 11.3: Resultados de test por nivel de granularidad. Se muestra la reducción del error (%) sobre el mejor de los métodos de referencia.

por cada granularidad, comparándose visualmente con el mejor de los métodos de referencia M4 y el mejor de los estándares AETS y AARIMA de esa granularidad. Además, con objeto de comparar visualmente las predicciones de series intermitentes, hemos incluido específicamente dos claros ejemplos de series diarias de este tipo.

11.2.1. Ganularidad trimestral

Hemos escogido la serie del capítulo arancelario 7 (hortalizas, plantas, raíces y tubérculos), que presenta una estacionalidad puramente anual y tendencia ascendente antes y después de la crisis económica 2008-2014:

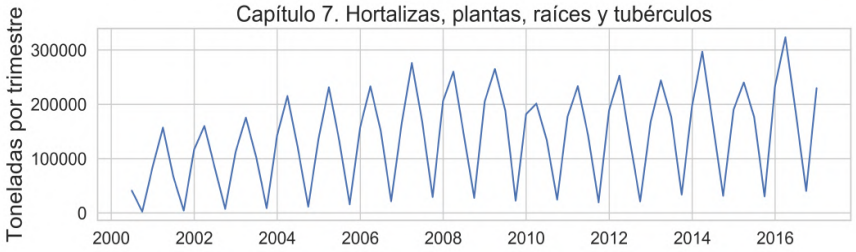

Figura 11.1: Serie trimestral capítulo 7.

La figura 11.2 muestra las predicciones de nuestros modelos en comparación con el mejor de los métodos de referencia (Damped) y el mejor de los métodos de selección automática de modelo (AETS). Tanto Damped como AETS obtienen muy buenas predicciones; y entre nuestros modelos, se observa que DCCNN obtiene la mejor predicción tal y como nos indicaron los resultados numéricos del cuadro 11.3.

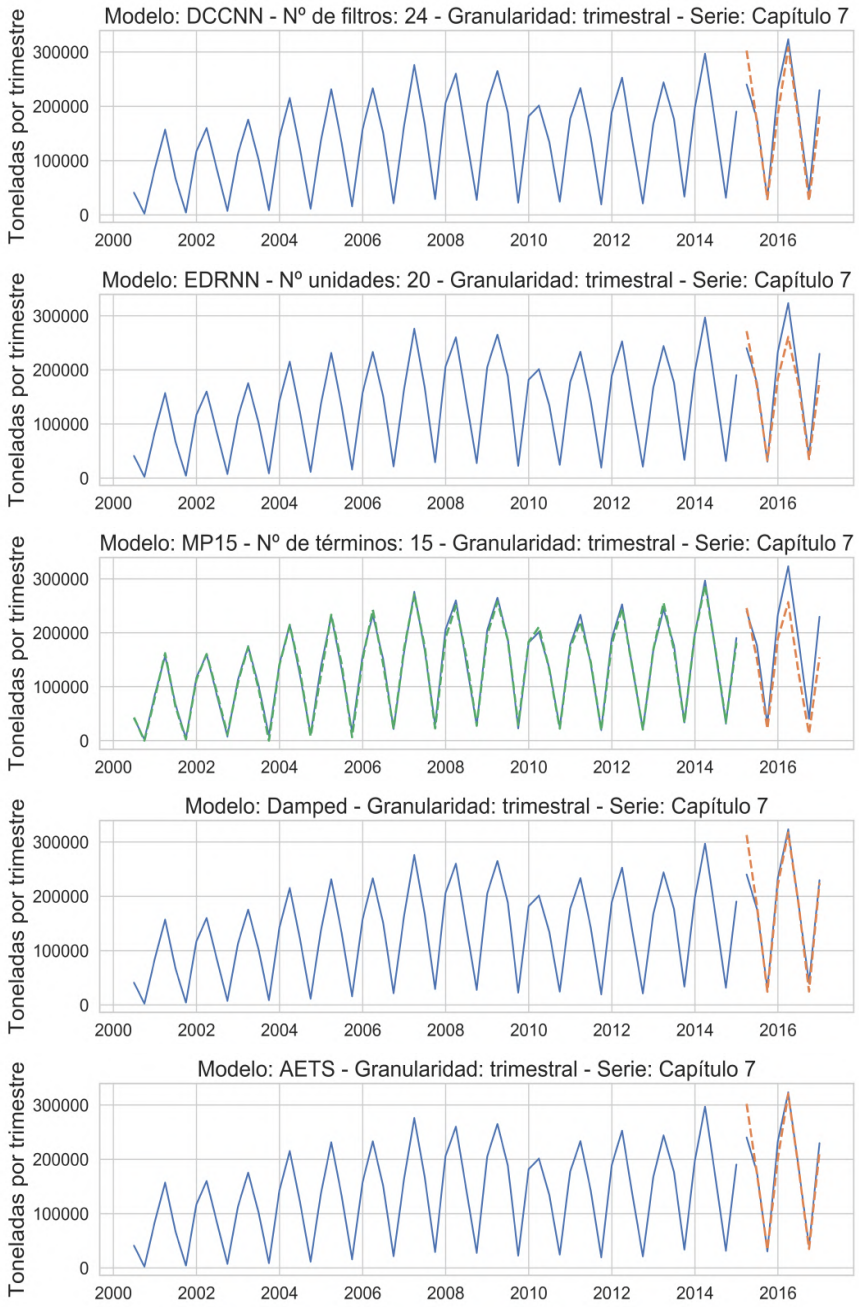

Figura 11.2: Ejemplos de predicción con datos de test. Serie trimestral correspondiente al capítulo 7.

11.2.2. Granularidad mensual

La serie seleccionada en este caso es el capítulo arancelario 10 (cereales), que tiene estacionalidad anual y semestral, además de una tendencia ascendente antes y después de la crisis 2008-2014:

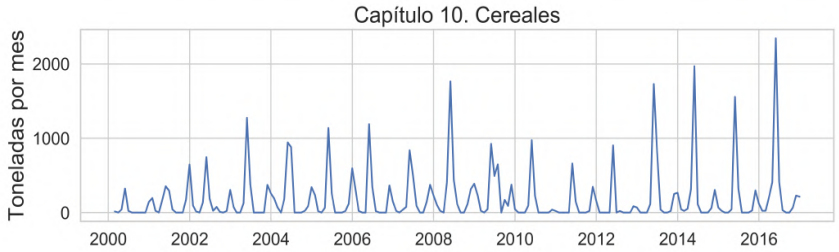

Figura 11.3: Serie mensual capítulo 10.

La figura 11.4 muestra las predicciones de nuestros modelos en comparación con el mejor de los métodos de referencia (Holt) y el mejor de los métodos de selección automática de modelo (AARIMA). Observamos que el modelo AARIMA obtiene la mejor predicción; y en cuanto nuestros modelos, DCCNN y EDRNN muestran un comportamiento superior a MP15. Esto está de acuerdo con los resultados numéricos del cuadro 11.3, si bien, para esta serie concreta, entre DCCNN y EDRNN no se aprecia apenas diferencia.

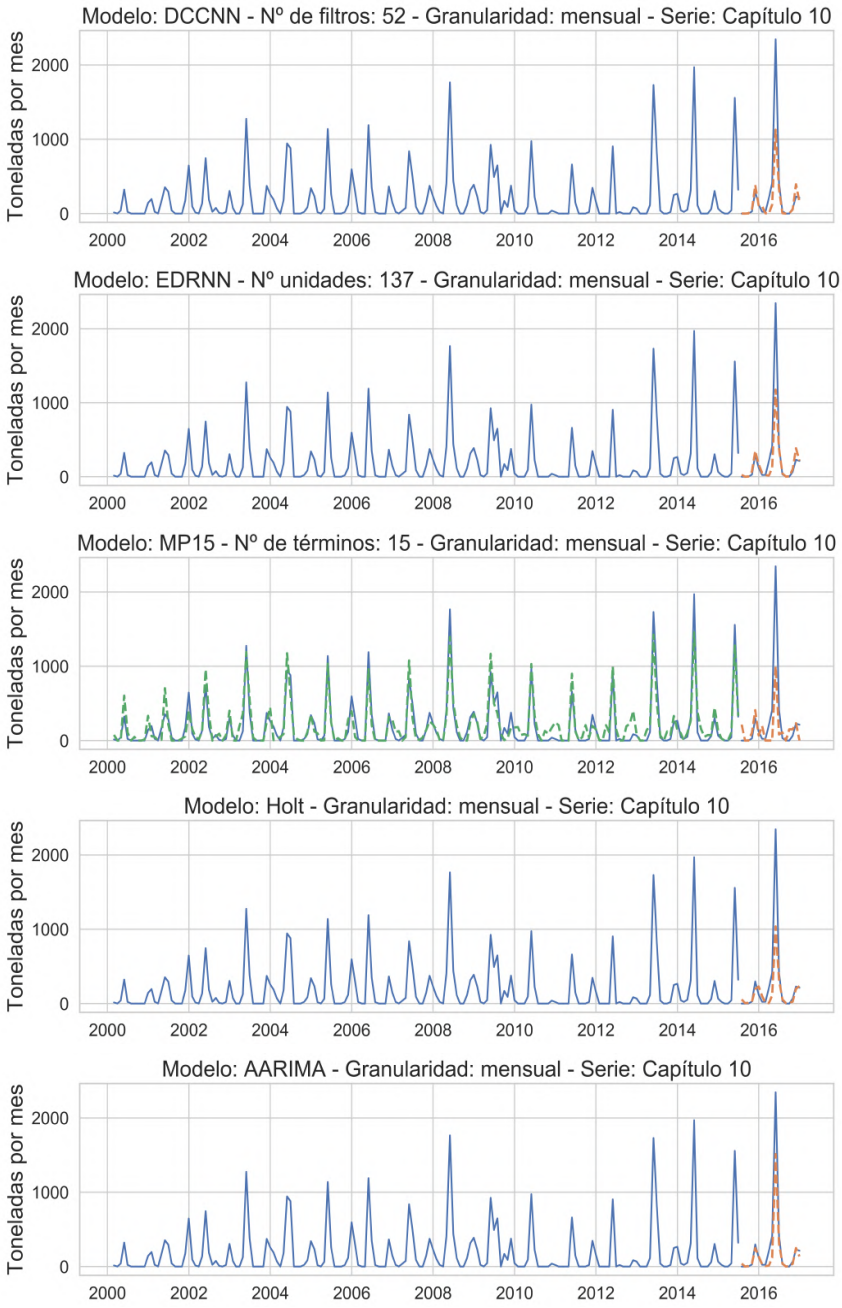

Figura 11.4: Ejemplos de predicción con datos de test. Serie mensual correspondiente al capítulo 10.

11.2.3. Granularidad semanal

Hemos seleccionado la serie del capítulo arancelario 85 (máquinas, aparatos y material eléctrico, y sus partes; aparatos de grabación o reproducción de sonido, aparatos de grabación o reproducción de imágenes y sonido en televisión, y las partes y accesorios de estos aparatos), que principalmente presenta una fuerte tendencia ascendente interrumpida en parte por la crisis económica del 2008-1014. También tiene una pequeña estacionalidad anual, que no se aprecia en la gráfica de la serie pero sí en la gráfica de autocorrelación de la figura 7.2. La gráfica de la serie seleccionada es la siguiente:

Figura 11.5: Serie semanal capítulo 85.

La figura 11.6 muestra las predicciones de nuestros modelos en comparación con el mejor de los métodos de referencia (Naïve S) y el mejor de los métodos de selección automática de modelo (AARIMA). En primer lugar, hay que resaltar que los modelos AETS no capturan la estacionalidad de periodos de más de 24 observaciones, no pudiendo ser competencia para los modelos AARIMA. En esta figura se ha recortado el intervalo temporal de las gráficas para lograr una mejor visualización de las predicciones. Observamos que nuestro modelo DCCNN obtiene la mejor predicción por delante del modelo AARIMA, de acuerdo con los resultados numéricos del cuadro 11.3.

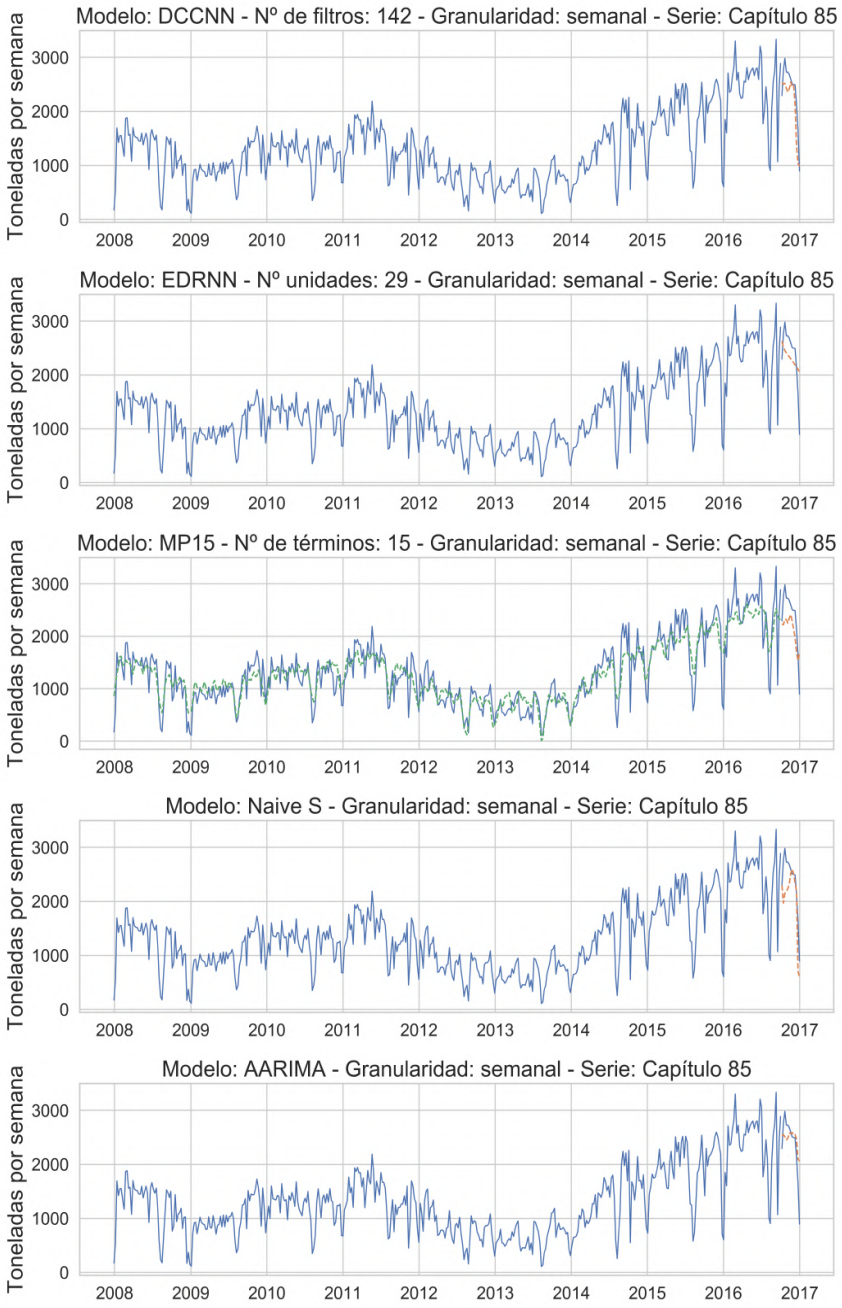

Figura 11.6: Ejemplos de predicción con datos de test. Serie semanal correspondiente al capítulo 85.

11.2.4. Granularidad diaria

Hemos escogido la serie del capítulo arancelario 3 (pescados y crustáceos, moluscos y demás invertebrados acuáticos). Tiene una marcada estacionalidad semanal sin una tendencia clara. Las series de granularidad diaria solo contienen datos de los últimos dos años y medio disponibles, por lo que en algunos casos, como el de nuestra serie seleccionada, es difícil apreciar una pequeña estacionalidad anual que sí queda más clara en la gráfica de autocorrelación de la figura 7.2. La gráfica de la serie capítulo 3 es la siguiente:

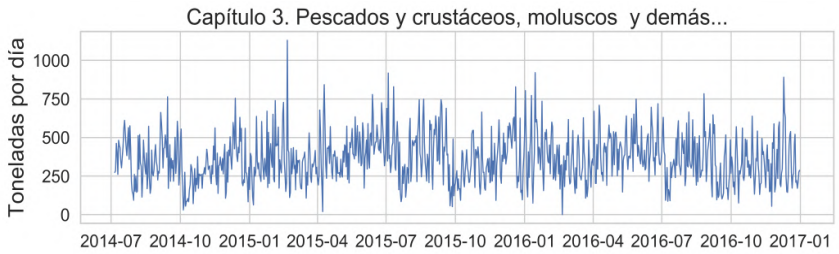

Figura 11.7: Serie diaria capítulo 3.

La figura 11.8 muestra las predicciones de nuestros modelos en comparación con el mejor de los métodos de referencia (RNN) y el mejor de los métodos de selección automática de modelo (AARIMA). Hemos recortado el intervalo temporal de las gráficas para lograr una mejor visualización de las predicciones. Tal y como indican los resultados numéricos del cuadro 11.3, nuestro modelo EDRNN tiene un comportamiento excelente. También destacamos la predicción de nuestro algoritmo MP15, que obtiene el segundo mejor resultado con series de granularidad diaria, donde todas presentan estacionalidad semanal en mayor o menor medida según refleja la figura 7.2.

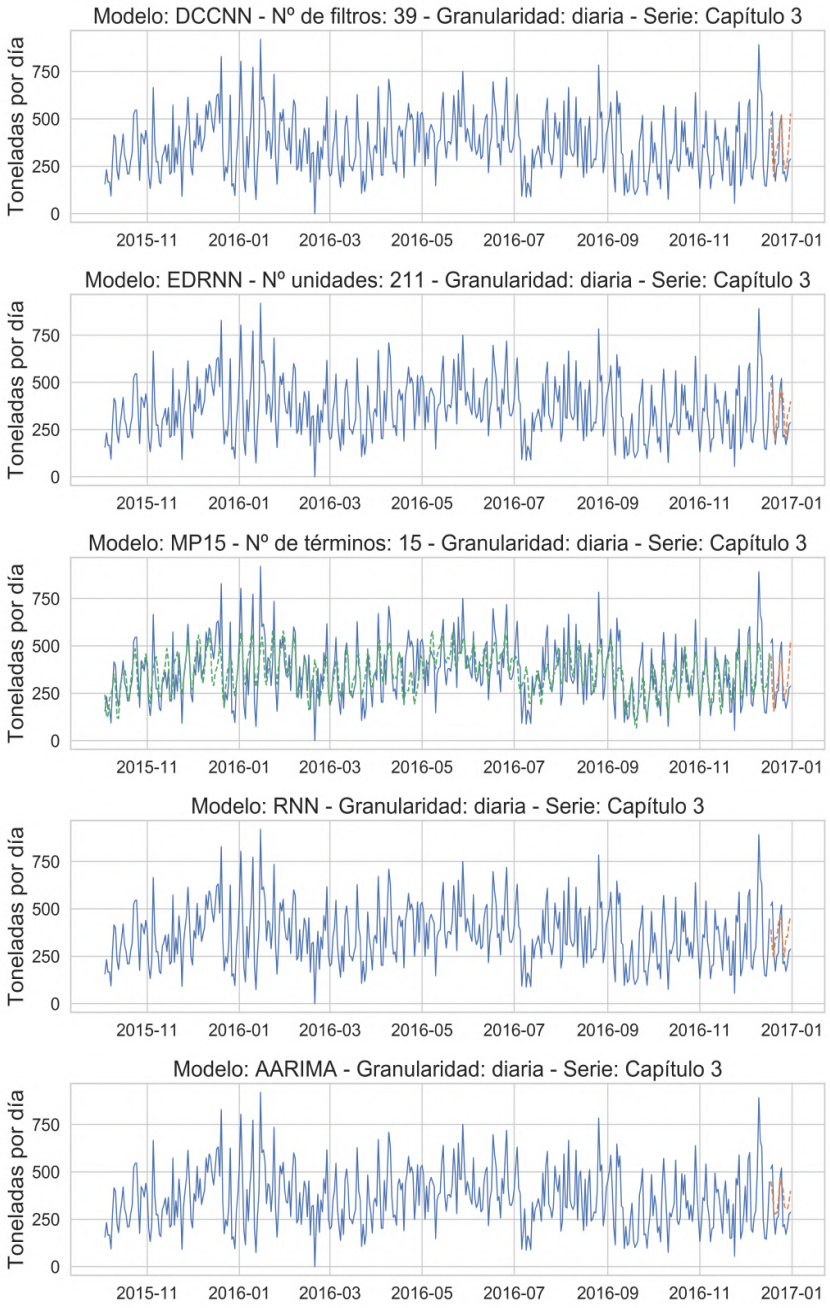

Figura 11.8: Ejemplos de predicción con datos de test. Serie diaria correspondiente al capítulo 3.

11.2.5. Series intermitentes de granularidad diaria

Con objeto de comparar visualmente las predicciones de series intermitentes, hemos incluido específicamente dos series diarias correspondientes a los capítulos arancelarios 10 (cereales) y 69 (productos cerámicos). Estas series, como todas las de granularidad diaria solo contienen datos de los últimos dos años y medio disponibles. Sus representaciones gráficas son las siguientes:

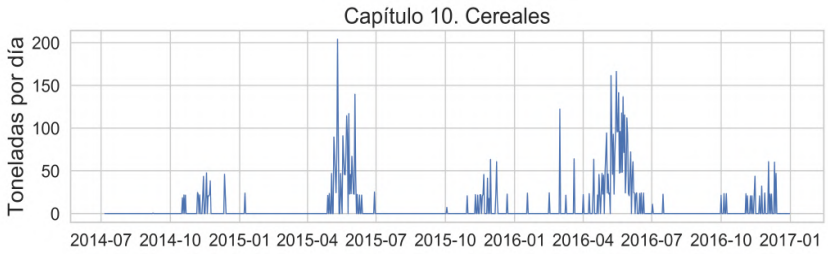

Figura 11.9: Serie diaria intermitente capítulo 10.

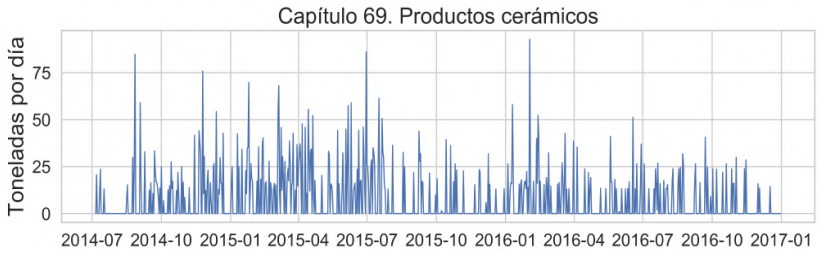

Figura 11.10: Serie diaria intermitente capítulo 69.

En las figuras 11.11 y 11.12 mostramos las predicciones, para las series de los capítulos 10 y 69, de nuestros modelos en comparación con el mejor de los métodos de referencia (Naïve S) y el mejor de los métodos de selección automática de modelo (AARIMA). Hemos recortado el intervalo temporal para un mejor detalle en la visualización de las predicciones. Podemos comprobar que el modelo EDRNN es superior a los demás para este tipo de series, que se encuentran principalmente en la granularidad diaria, lo cual está en concordancia con los excelentes resultados numéricos obtenidos por este modelo en el cuadro 11.3.

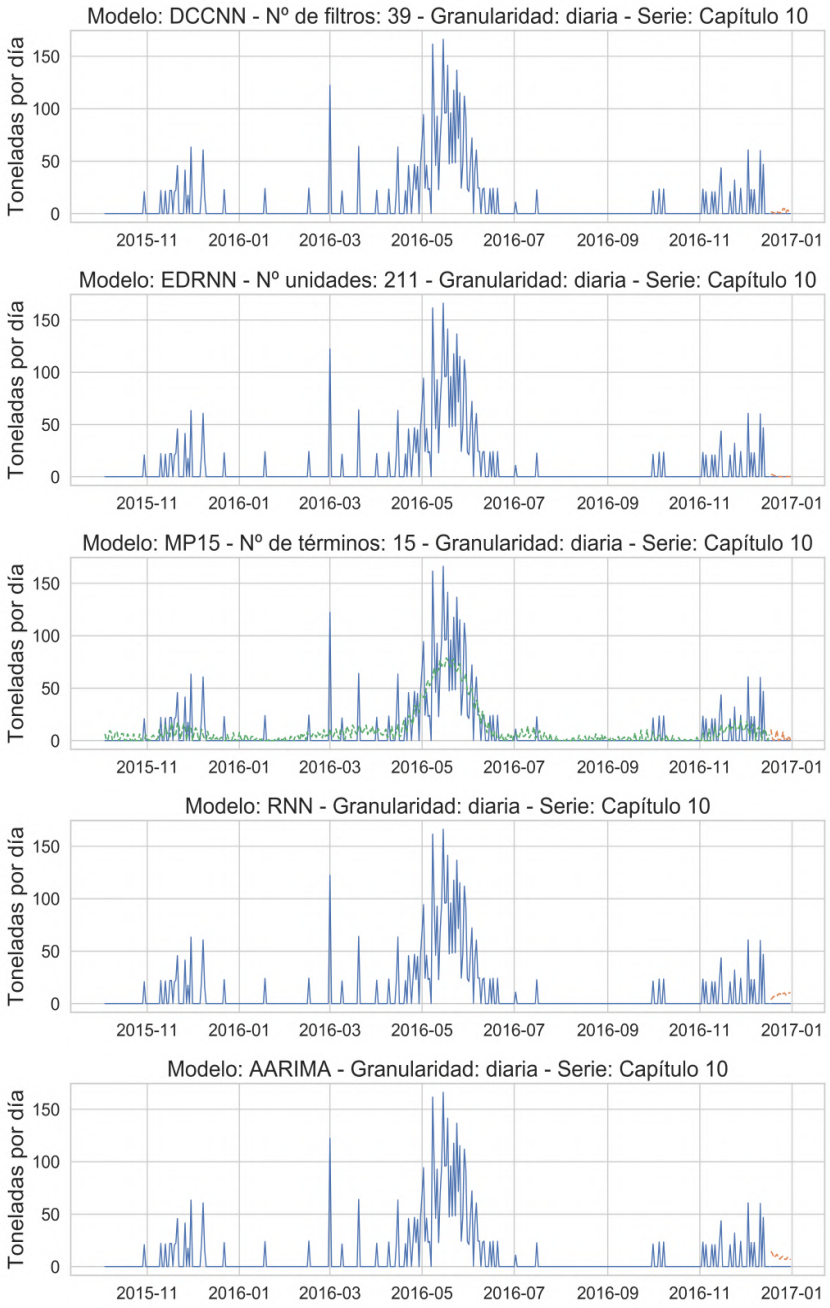

Figura 11.11: Ejemplos de predicción con la serie diaria intermitente correspondiente al capítulo 10.

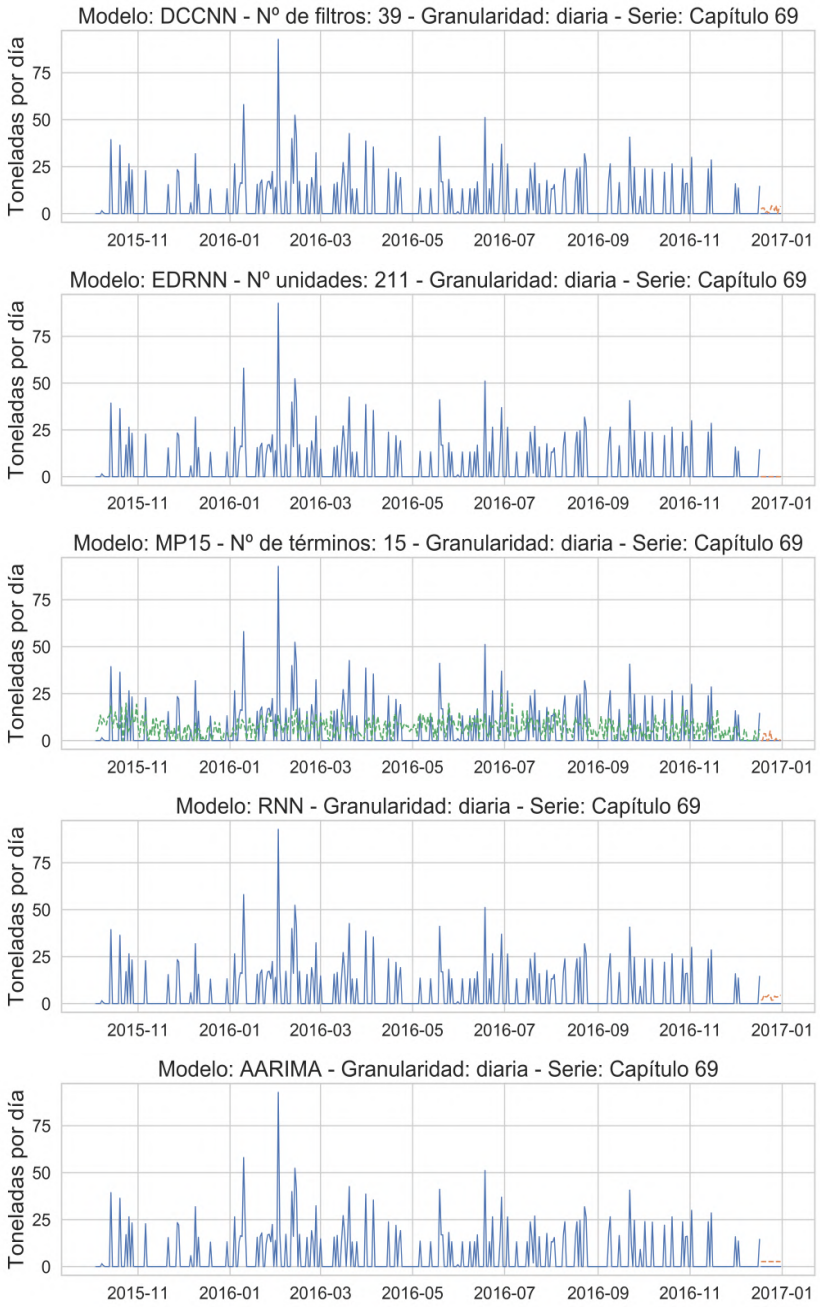

Figura 11.12: Ejemplos de predicción con la serie diaria intermitente correspondiente al capítulo 69.

Capítulo 12

Conclusiones

En este trabajo hemos evaluado la utilidad de dos modelos de aprendizaje profundo y un algoritmo de búsqueda de coincidencia para predecir el flujo de mercancías a través de los puertos marítimos. Hemos experimentado con siete series temporales de mercancías importadas de Marruecos a través del Puerto de Algeciras. Las series iniciales tenían una granularidad de un minuto, y fueron remuestreadas en cuatro granularidades (trimestral, mensual, semanal y diaria), cada una asociada a un horizonte de predicción específico (8 trimestres, 18 meses, 13 semanas y 14 días). El *dataset* de las series temporales utilizadas se encuentra disponible en Mendeley Data [43].

Los experimentos muestran que nuestros dos modelos, la red neuronal recurrente codificador-decodificador y la red neuronal convolucional causal dilatada, pueden gestionar estas series en bruto sin eliminar primero la estacionalidad o la tendencia. Ambos modelos han superado globalmente la medida de exactitud OWA de todos los modelos de referencia de la Competición M4. Por niveles de granularidad, destacamos en primer lugar el modelo recurrente con datos diarios, que ha mejorado el OWA del mejor método de referencia en un 38,6 % y del estándar AARIMA en un 42,6 % para el conjunto de datos de test. En segundo lugar destaca el modelo convolucional con datos semanales, mejorando el OWA del mejor método de referencia en un 12,1 % y del estándar AARIMA en un 4,6 % para el conjunto de datos de test; este modelo con series mensuales también consigue superar el OWA del mejor método de referencia en un 1,1 %. Para las series trimestrales los resultados han sido pobres debido a la limitación en el volumen de datos disponibles para un adecuado entrenamiento de los modelos. El modelo recurrente consigue mejores resultados que el convolucional con series más intermitentes, como las diarias. Además, queremos destacar el acierto en la elección del año como longitud de los datos de entrada a los modelos, avalado por el trabajo de C. Hamzaçebi [25], que propone un tamaño de entrada igual a la longitud

del periodo estacional para mejorar las predicciones con series estacionales. Los resultados y la visualización de las predicciones nos muestran que los dos modelos capturan correctamente la estacionalidad y la tendencia, incluso para series intermitentes, sin aplicar ninguna transformación previa sobre los datos y simplemente escalándolos al rango [0, 1]. En cuanto a las estrategias de predicción, la estrategia de múltiples salidas arroja resultados exitosos en las series semanales y diarias, capturando la dependencia entre los valores pronosticados. Sin embargo, para las series mensuales y trimestrales, en las que se dispone de menos datos de entrenamiento, hemos obtenido mejores resultados con la estrategia recursiva, que aumenta el tamaño de los datos de entrenamiento utilizando datos de entrada más cercanos a la predicción.

El algoritmo de búsqueda de coincidencia ha obtenido sus mejores resultados con datos diarios, mejorando el OWA del mejor método de referencia en un 26,3 % y del estándar AARIMA en un 30,3 % para el conjunto de datos de test. La existencia de información de diferentes periodicidades, como la semanal y anual, en datos de granularidad diaria permite que este algoritmo obtenga unos buenos resultados.

Por último, queremos destacar este trabajo como contribución a salvar la brecha existente entre la industria y el entorno académico, que parecen desarrollarse de forma más o menos independiente uno del otro. El éxito del estudio demuestra la importancia de este tipo de colaboración, que puede dar lugar a soluciones innovadoras, tal y como las que se presentan en esta tesis.

12.1. Resumen de nuestra aportación

A modo de resumen, mostramos lo más destacable de nuestra aportación:

- Acercar el mundo empresarial y el académico.
- Aplicación de técnicas de aprendizaje automático al ámbito empresarial.
- Utilización de dos modelos de aprendizaje profundo, y un algoritmo de búsqueda de coincidencia, en la predicción de series temporales económicas.
- Ayuda a las autoridades portuarias en la toma de decisiones sobre infraestructuras e instalaciones a la luz de las predicciones de demanda de transporte de mercancías desagregado.
- Creación y publicación en Mendeley Data del *dataset* utilizado en este trabajo.
- Uso de series temporales en bruto, sin aplicar ninguna transformación previa; simplemente se han escalado al rango [0, 1] para los modelos de redes neuronales.
- En los modelos de aprendizaje profundo:
  - Ambos superan globalmente los métodos de referencia M4.
  - El modelo recurrente destaca con series diarias y series intermitentes.
  - El modelo convolucional destaca con series semanales.
  - Uso común de un tamaño de entrada a los modelos de un año para todas las granularidades, que permite capturar la estacionalidad y tendencia.
  - Dos estrategias de predicción:
    - Recursiva, para series trimestrales y mensuales.
    - De múltiples salidas, para series semanales y diarias.
- El algoritmo de búsqueda de coincidencia destaca con series diarias.

Bibliografía

[1] Aloysius, N., and Geetha, M. A review on deep convolutional neural networks. In *2017 International Conference on Communication and Signal Processing (ICCSP)* (2017), IEEE, pp. 0588–0592.

[2] Assimakopoulos, V., and Nikolopoulos, K. The theta model: a decomposition approach to forecasting. *International Journal of Forecasting 16*, 4 (2000), 521–530.

[3] Attia, T. M. Importance of communication and information technology and its applications in the development and integration of performance in seaports. *Renewable Energy and Sustainable Development 2*, 2 (2016), 137–146.

[4] Batarseh, F., Gopinath, M., Nalluru, G., and Beckman, J. Application of machine learning in forecasting international trade trends. *arXiv preprint arXiv:1910.03112* (2019).

[5] Bontempi, G., Taieb, S. B., and Le Borgne, Y.-A. Machine learning strategies for time series forecasting. In *European Business Intelligence Summer School* (2012), Springer, pp. 62–77.

[6] Box, G. E. P., and Cox, D. R. An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological) 26*, 2 (1964), 211–243.

[7] Box, G. E. P., and Jenkins, G. M. *Time series analysis: forecasting and control*. Springer Science & Business Media, 1970.

[8] Brown, R. G. *Exponential smoothing for predicting demand*. Cambridge, Massachusetts: Arthur D. Little Inc., 1956.

[9] Chai, L., Du, J., Liu, Q.-F., and Lee, C.-H. Using generalized gaussian distributions to improve regression error modeling for deep learning-based speech enhancement. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 27*, 12 (2019), 1919–1931.

[10] Chen, B. Air quality index forecasting via deep dictionary learning. *IEICE Transactions on Information and Systems 103*, 5 (2020), 1118–1125.

[11] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[12] Chollet, F., et al. Keras, 2015. `https://keras.io`, Accedido el 26-04-2020.

[13] Chopra, S. *Supply chain management: strategy, planning, and operation*. Pearson, Boston, 2019.

[14] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[15] Cordón, J. R., Ramiro Olivier, P., García Sedeño, M. A., and Walliser Martín, J. Diseño y validación de una prueba de selección para controladores de tráfico marítimo basada en la medida de la conciencia situacional. *Revista de Psicología del Trabajo y de las Organizaciones 30*, 2 (2014), 83–93.

[16] De Martino, M., Errichiello, L., Marasco, A., and Morvillo, A. Logistics innovation in seaports: an inter-organizational perspective. *Research in Transportation Business & Management 8* (2013), 123–133.

[17] Escamilla-Navarro, L., García-Menéndez, L., and Pérez-García, E. Integration of foreign trade and maritime transport statistics in Spain. *Maritime Policy & Management 37*, 4 (2010), 347–375.

[18] González de la Rosa, J. J., Lloret, I., Moreno, A., Puntonet, C., and Górriz, J. Wavelets and wavelet packets applied to detect and characterize transient alarm signals from termites. *Measurement 39*, 6 (2006), 553–564.

[19] González de la Rosa, J. J., Lloret, I., Puntonet, C., and Górriz, J. Higher-order statistics to detect and characterise termite emissions. *Electronics Letters 40*, 20 (2004), 1316–1317.

[20] González de la Rosa, J. J., Lloret, I., Puntonet, C., Piotrkowski, R., and Moreno, A. Higher-order spectra measurement techniques of termite emissions. A characterization framework. *Measurement 41*, 1 (2008), 105–118.

[21] González de la Rosa, J. J., Munoz, A. M., Lloret, I., Puntonet, C. G., and Górriz, J.-M. Power transients characterization and classification using higher-order cumulants and competitive layers. In *International Conference on Adaptive and Natural Computing Algorithms* (2007), Springer, pp. 782–789.

[22] González de la Rosa, J. J., Puntonet, C., and Lloret, I. An application of the independent component analysis to monitor acoustic emission signals generated by termite activity in wood. *Measurement 37*, 1 (2005), 63–76.

[23] Goodfellow, I., Bengio, Y., and Courville, A. Deep learning, 2016. `http://www.deeplearningbook.org`. Accedido el 16-01-2021.

[24] Groenen, P. J., Nalbantov, G., and Bioch, J. C. SVM-Maj: a majorization approach to linear support vector machines with different hinge errors. *Advances in Data Analysis and Classification 2*, 1 (2008), 17–43.

[25] Hamzaçebi, C. Improving artificial neural networks' performance in seasonal time series forecasting. *Information Sciences 178*, 23 (2008), 4550–4559.

[26] Harbola, S., and Coors, V. One dimensional convolutional neural network architectures for wind prediction. *Energy Conversion and Management 195* (2019), 70–75.

[27] Havenga, J. H. The importance of disaggregated freight flow forecasts to inform transport infrastructure investments. *Journal of Transport and Supply Chain Management 7*, 1 (2013), 1–7.

[28] Havenga, J. H., and Pienaar, W. J. The creation and application of a national freight flow model for south africa. *Journal of the South African Institution of Civil Engineering 54*, 1 (2012), 2–13.

[29] Heilig, L., and Voss, S. Information systems in seaports: a categorization and overview. *Information Technology and Management 18*, 3 (2017), 179–201.

[30] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural Computation 9*, 8 (1997), 1735–1780.

[31] Holt, C. C. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting 20*, 1 (2004), 5–10.

[32] Hyndman, R., Koehler, A. B., Ord, J. K., and Snyder, R. D. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

[33] Hyndman, R. J. A brief history of forecasting competitions. *International Journal of Forecasting 36*, 1 (2020), 7–14.

[34] Hyndman, R. J., and Athanasopoulos, G. *Forecasting: principles and practice*. OTexts, 2018.

[35] Hyndman, R. J., Khandakar, Y., et al. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software 27*, 3 (2008), 1–22.

[36] Hyndman, R. J., and Koehler, A. B. Another look at measures of forecast accuracy. *International Journal of Forecasting 22*, 4 (2006), 679–688.

[37] Jun, L., and Shi-chang, L. Application of sparse representation method based on K-SVD-OMP in electricity load forecasting. *Electric Machines & Control/Dianji Yu Kongzhi Xuebao 24*, 9 (2020).

[38] Khodayar, M., Liu, G., Wang, J., Kaynak, O., and Khodayar, M. E. Spatiotemporal behind-the-meter load and PV power forecasting via deep graph dictionary learning. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[39] Kingma, D. P., and Ba, J. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[40] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25* (2012), 1097–1105.

[41] Kwiatkowski, D., Phillips, P. C., Schmidt, P., and Shin, Y. Testing the null hypothesis of stationarity against the alternative of a unit root: how sure are we that economic time series have a unit root? *Journal of Econometrics 54*, 1-3 (1992), 159–178.

[42] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature 521*, 7553 (2015), 436–444.

[43] Lloret, I. Imported goods from Morocco to Spain through the port of Algeciras (2000-2016). *Mendeley Data, V1* (2020).

[44] Lloret, I., Troyano, J. A., Enríquez, F., and González de la Rosa, J. J. Two deep learning approaches to forecasting disaggregated freight flows: convolutional and encoder–decoder recurrent. *Soft Computing 25*, 12 (2021), 7769–7784.

[45] Makridakis, S. Accuracy measures: theoretical and practical concerns. *International Journal of Forecasting 9*, 4 (1993), 527–529.

[46] Makridakis, S., Spiliotis, E., and Assimakopoulos, V. Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS One 13*, 3 (2018).

[47] Makridakis, S., Spiliotis, E., and Assimakopoulos, V. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting 36*, 1 (2020), 54–74.

[48] Makridakis, S. G., Wheelwright, S. C., and Hyndman, R. J. *Forecasting: Methods and Applications*, 3rd ed. New York: Wiley, 1998.

[49] Mallat, S. G., and Zhang, Z. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing 41*, 12 (1993), 3397–3415.

[50] Mandic, D. P., and Chambers, J. *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. John Wiley & Sons, Inc., USA, 2001.

[51] Minsky, M., and Papert, S. *Perceptrons: an introduction to computational geometry*. Cambridge. MA: MIT Press, 1969.

[52] Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., and Damas, L. Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems 14*, 3 (2013), 1393–1402.

[53] Moros-Daza, A., Amaya-Mier, R., and Paternina-Arboleda, C. Port community systems: A structured literature review. *Transportation Research Part A: Policy and Practice 133* (2020), 27–46.

[54] Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: a generative model for raw audio. *arXiv preprint arXiv:1609.03499* (2016).

[55] Pascual, Ó. S. Estadísticas de tráfico portuario. *Indice: Revista de Estadística y Sociedad*, 64 (2015), 13–18.

[56] Puertos del Estado. Guía de usuario: solicitud de escala, 2016. `http://www.puertos.es/Documents/BERMAN%204.1.2%20v1.0.pdf`. Accedido el 20-02-2021.

[57] Puertos del Estado. Guía de usuario: declaraciones sumarias, 2019. `http://www.puertos.es/es-es/Documents/IFCSUM%20v.3.4.1.pdf`. Accedido el 20-02-2021.

[58] Qi, J., Du, J., Siniscalchi, S. M., Ma, X., and Lee, C.-H. On mean absolute error for deep neural network based vector-to-vector regression. *IEEE Signal Processing Letters 27* (2020), 1485–1489.

[59] Rojas, R. The backpropagation algorithm. In *Neural networks*. Springer, 1996, pp. 149–182.

[60] Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review 65*, 6 (1958), 386.

[61] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature 323*, 6088 (1986), 533–536.

[62] Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting 36*, 3 (2020), 1181–1191.

[63] Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Networks 61* (2015), 85–117.

[64] Seabold, S., and Perktold, J. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference* (2010).

[65] Suilin, A. kaggle-web-traffic, 2017. `https://github.com/Arturus/kaggle-web-traffic`, Accedido el 17-05-2021.

[66] Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems* (2014), pp. 3104–3112.

[67] Systematics, C. Inc. NCHRP report 388: A guidebook for forecasting freight transportation demand. *Transportation Research Board, Washington DC* (1997).

[68] Tošić, I., and Frossard, P. Dictionary learning. *IEEE Signal Processing Magazine 28*, 2 (2011), 27–38.

[69] Val, S., and Roldán, F. Spanish port hinterland intermodal information–futuremed pilot. *International Journal of Advanced Logistics 2*, 2 (2013), 1–17.

[70] Williams, R. J., and Zipser, D. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation 1*, 2 (1989), 270–280.

[71] Winters, P. R. Forecasting sales by exponentially weighted moving averages. *Management Science 6*, 3 (1960), 324–342.

Apéndice A

Resumen del currículum investigador del doctorando

Este apéndice muestra el listado de publicaciones de carácter investigador del doctorando, junto con una descripción detallada de las más relevantes, y también las colaboraciones con empresas y patentes.

A.1. Publicaciones

Se muestran en orden temporal los datos de las publicaciones en revistas y congresos:

- Lloret, I., Troyano, J.A., Enríquez, F., González de la Rosa, J.J.. Two deep learning approaches to forecasting disaggregated freight flows: convolutional and encoder-decoder recurrent. *Soft Computing* 25 - 12 (2008), pp. 7769 - 7784.

- González de la Rosa, J.J, Lloret, I., Puntonet, C., Piotrkowski, R., Moreno, A.. Higher-order spectra measurement techniques of termite emissions. A characterization framework. *Measurement: Journal of the International Measurement Confederation.* 41 (2008), pp. 105 - 118.

- González de la Rosa, J.J, Moreno, A., Lloret, I., Puntonet, C., Gorriz, J.M.. Power transients characterization and classification using higher-order cumulants and competitive layers. En *Proceedings of the 8th International Conference on Adaptative and Natural Computing Algorithms* 4431 (2007), pp. 782 - 789.

- González de la Rosa, J.J, Lloret, I., Moreno, A., Puntonet, C., Gorriz, J.M.. Higher-order spectral characterization of termite emissions using

acoustic emission probes. En *Sensors Applications Symposium* (2007), pp. 230 - 238.

- Puntonet, C., González de la Rosa, J.J, Lloret, I., Gorriz, J.M.. Wavelets transforms applied to termite detection. En *International Conference on Enterprise Information Systems* 1 (2007), pp. 163 - 167.
- Puntonet, C., González de la Rosa, J.J, Lloret, I., Gorriz, J.M.. On the performance of a HOS-based ICA algorithm in BSS of acoustic emission signals. *Lecture Notes in Computer Science* 3889 (2006), pp. 400 - 405.
- González de la Rosa, J.J, Puntonet, C., Piotrkowski, R, Lloret, I., Gorriz, J.M.. Two ICA algorithms applied to BSS in non-destructive vibratory tests. *Lecture Notes in Computer Science* 4132 (2006), pp. 221 - 229.
- González de la Rosa, J.J, Lloret, I., Puntonet, C., Moreno, A., Gorriz, J.M.. Third-order spectral characterization of termite's emission track. *Lecture Notes in Computer Science* 3991 (2006), pp. 316 - 323.
- González de la Rosa, J.J, Moreno, A., Lloret, I., Pallarés, V, Liñán, M. Characterisation of frequency instability and frequency offset using instruments with incomplete data sheets. *Measurement: Journal of the International Measurement Confederation* 39 - 7 (2006), pp. 664 - 673.
- González de la Rosa, J.J, Lloret, I., Moreno, A., Puntonet, C., Gorriz, J.M.. Wavelets and wavelet packets applied to detect and characterize transient alarm signals from termites. *Measurement: Journal of the International Measurement Confederation.* 39 - 6 (2006), pp. 553 - 564.
- González de la Rosa, J.J, Lloret, I., Puntonet, C., Gorriz, J.M.. Wideband transducers and higher-order spectral characterization of low-level ultrasounds in wood. En *Proceedings of the Fourth IEEE Workshop on Sensor Array and Multichannel Processing* (2006), pp. 667 - 671.
- González de la Rosa, J.J, Lloret, I., Moreno, A., Piotrkowski, R, Ruzzante, J.E., Puntonet, C., Gorriz, J.M.. Higher-order spectra and ICA used for identification and SNR enhancement of acoustic emission signals. En *Mediterranean Electrotechnical Conference* 13 (2006).
- González de la Rosa, J.J, Lloret, I., Puntonet, C., Gorriz, J.M.. Frequency calibrations with conventional time interval counters via GPS traceability. En *International Conference on Enterprise Information Systems.* 1 (2006), pp. 20-26.

- Puntonet, C., González de la Rosa, J.J, Lloret, I., Gorriz, J.M.. Recognition of insect emissions applying the discrete wavelet transform. *Lecture Notes in Computer Science* 3686 (2005), pp. 505 - 513.

- González de la Rosa, J.J, Puntonet, C., Lloret, I., Gorriz, J.M.. Wavelets and wavelet packets applied to termite detection. *Lecture Notes in Computer Science*. 3514 - 1 (2005), pp. 900 - 907.

- González de la Rosa, J.J, Puntonet, C., Lloret, I.. An application of the independent component analysis to monitor acoustic emission signals generated by termite activity in wood. *Measurement: Journal of the International Measurement Confederation* 37 - 1 (2005), pp. 63 - 76.

- González de la Rosa, J.J, Lloret, I., Moreno, A., Liñán, M., Puntonet, C., Piotrkowski, R.. Characterization of termite emissions by using higher-order spectra. En *Congress 4TH E-GLEA Meeting* 1 (2005), pp. 1 - 23.

- Puntonet, C., González de la Rosa, J.J, Gorriz, J.M., Lloret, I.. Wavelet Packets for Insect Identification. En *Conferencia Iberoamericana en Sistemas, Cibernetica e Informatica* (2005), pp. 23 - 29.

- González de la Rosa, J.J, Lloret, I., Ruzzante, J.E., Piotrkowski, R., Armeite, M., López-Pumarega, M.I..
  Higher order analysis of acoustic emission signals. En *IEEE International Conference on Computational Intelligence for Measurement Systems and Applications* (2005), pp. 296 - 300.

- González de la Rosa, J.J, Lloret, I., Puntonet, C., Gorriz, J.M.. Wavelets transforms applied to termite detection. En *IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications* 1 (2005), pp. 23 - 29.

- González de la Rosa, J.J, Lloret, I., Puntonet, C.. A graphical review of noise-instability characterization in electronic systems. En *II International Conference on Informatics in Control, Automation and Robotics*. 1 (2005), pp. 12 - 16.

- González de la Rosa, J.J, Puntonet, C., Gorriz, J.M., Lloret, I.. An application of ICA to identify vibratory low-level signals generated by termites. *Lecture Notes in Computer Science* 3195 (2004), pp. 1126 - 1133.

- González de la Rosa, J.J, Lloret, I., Puntonet, C., Gorriz, J.M.. Higher-order statistics to detect and characterise termite emissions. *Electronics Letters*, 40 - 20 (2004), pp. 1316 - 1317.
- González de la Rosa, J.J, Lloret, I., Gorriz, J.M., Puntonet, C.. An application of the independent component analysis to monitor acoustic emission signals generated by termite activity in wood. En *First International Conference on Informatics in Control, Automation and Robotics* 3 (2004), pp. 11 - 18.
- González de la Rosa, J.J, Puntonet, C., Gorriz, J.M., Lloret, I.. An application of HOS and ICA to detect and characterize noisy signals generated by termites. En *IEEE 4th International Conference on Intelligent Systems Design and Application* (2004), pp. 163 - 168.
- Gallero, F.C., Lloret, I., Ariza, O.. NOX, SO2, and TSP concentration levels from monitoring network in the Campo de Gibraltar region and their contributing sources. En *First International Conference on Applied Sciences and the Environment* (1998), pp. 75 - 84.

A.2. Detalle de publicaciones relevantes

En esta sección incluimos en orden cronológico varias publicaciones del currículum investigador del doctorando relacionadas con series temporales, junto con sus indicios de calidad.

En la más reciente de todas describimos nuestra tarea de predicción con las series temporales de tráfico marítimo, los dos modelos de aprendizaje profundo (convolucional y recurrente), la metodología experimental y los resultados obtenidos. Ha sido publicada recientemente (junio de 2021), en un número especial de la revista *Soft Computing* sobre sus aplicaciones en la economía. Para los indicios de calidad hemos tomado el último año disponible (2020) en *Web of Science* (*WoS*):

Autores	Lloret Galiana, Isidro *(Orcid: 0000-0002-8628-247X)* Troyano Jiménez, José Antonio Enríquez de Salamanca Ros, Fernando González de la Rosa, Juan José
Título	Two deep learning approaches to forecasting disaggregated freight flows: convolutional and encoder–decoder recurrent

Revista	*Soft Computing, 25(12), 7769-7784*
Año	2021 (junio)
Índice de impacto	3,643 JCR
Categoría	*Computer Science, Interdisciplinary Applications*
Cuartil	Q2 (59/142)
Comentarios de revisores	Rev1. The topic is interesting and the authors make good results / Rev2. Results are interesting

También son muy interesantes los trabajos publicados en relación con el procesamiento de series temporales para la detección de plagas de termitas, y la caracterización de los eventos acústicos producidos, tanto en su actividad de alimentación y excavación, como en la emisión de señales de alarma. En la línea de trabajo de la aplicación de la estadística de orden superior, en 2008 publicamos un artículo en el que caracterizamos la actividad de las termitas con el fin de eliminar la subjetividad en la posible detección de una infestación en entornos ruidosos. Esto se consiguió por medio del análisis de cortes diagonales del biespectro y triespectro:

Autores	González de la Rosa, Juan José Lloret Galiana, Isidro García Puntonet, Carlos Piotrkowski, Rosa Moreno Munoz, Antonio
Título	Higher-order spectra measurement techniques of termite emissions. A characterization framework
Revista	*Measurement, 41(1), 105-118*
Año	2008
Índice de impacto	0,662 JCR
Categoría	*Engineering, Multidisciplinary*
Cuartil	Q3 (37/67)
Citas	19 (WoS) / 29 (Scopus)

En el año 2006 publicamos otro artículo relacionado con la posibilidad de usar *wavelets* y *wavelets packages* en la detección y caracterización de los transitorios provocados por las señales de alarma de termitas:

Autores	González de la Rosa, Juan José Lloret Galiana, Isidro Moreno Munoz, Antonio García Puntonet, Carlos
Título	Wavelets and wavelet packets applied to detect and characterize transient alarm signals from termites
Revista	*Measurement, 39(6), 553-564*
Año	2006
Índice de impacto	0.525 JCR
Categoría	*Engineering, Multidisciplinary*
Cuartil	Q3 (34/67)
Citas	15 (WoS) / 17 (Scopus)

Otro interesante trabajo fue publicado en el año 2005, en el que conseguimos exitosamente separar señales de alarma de termitas del ruido, mediante técnicas de separación ciega de señales por aplicación del análisis de componentes independientes (*ICA*; *independent component analysis*):

Autores	González de la Rosa, Juan José García Puntonet, Carlos Lloret Galiana, Isidro
Título	An application of the independent component analysis to monitor acoustic emission signals generated by termite activity in wood
Revista	*Measurement, 37(1), 63-76*
Año	2005
Índice de impacto	0,413 JCR
Categoría	*Engineering, Multidisciplinary*
Cuartil	Q2 (32/65)
Citas	39 (WoS) / 36 (Scopus)

Por último, es destacable también la primera publicación, que fue en el año 2004, donde propusimos el uso del biespectro como estadístico para la detección de eventos acústicos de termitas, incluso con series temporales de baja relación señal/ruido:

Autores	González de la Rosa, Juan José Lloret Galiana, Isidro García Puntonet, Carlos Gorriz Saez, Juan Manuel
Título	Higher-order statistics to detect and characterise termite emissions
Revista	*Electronics Letters, 40(20), 1316-1317*
Año	2004
Índice de impacto	0,968 JCR
Categoría	*Engineering, Electrical and Electronic*
Cuartil	Q2 (72/209)
Citas	32 (WoS) / 34 (Scopus)

A.3. Colaboraciones y transferencia

En relación con nuestro trabajo con series temporales es interesante una colaboración con empresa:

- González de la Rosa, J. J., y Lloret, I. Análisis de emisiones de insectos con espectros de tercer orden. Contrato con la multinacional Syngenta Crop Protection AG. Universidad de Cádiz (2005), contrato art. 11/45 LRU - 68/83 LOU.

Y finalmente, como resultado de parte de nuestra investigación, es destacable la patente licenciada:

- González de la Rosa, J. J., y Lloret, I. Method of detecting termites using electronic and computational techniques employing multidimensional spectra. Patente WO2006128932, 07/12/2006. Licenciada por Natural Conections and Consulting, S.L. (KONECTIA) en mayo de 2008.

16 de julio de 2021