

# **Predicciones a corto plazo en series temporales de alta frecuencia**

Trabajo Fin de Máster

Supervisores: Aritz Adin y Guzmán Santafé

Master en Modelización e Investigación Matemática, Estadística y Computación.

24 de septiembre de 2021

Harkaitz Goyena Baroja

*Departamento de Estadística, Informática y Matemáticas.*

*Institute for Advanced Materials and Mathematics (INAMAT2)*

*Universidad Pública de Navarra*

**E-mail:** harkaitz.goyena@unavarra.es

**upna**

Universidad Pública de Navarra  
Nafarroako Unibertsitate Publikoa



## Resumen

La necesidad de obtener predicciones a corto plazo precisas y rápidas se hace patente en distintas áreas. Por ejemplo en la industria se requiere predecir paradas o errores en los procesos productivos, o predecir la producción de energía eólica para una mejor atención de la demanda eléctrica.

El objetivo principal de este trabajo es estudiar la capacidad predictiva de algunos métodos clásicos de series temporales y su comparación con métodos basados en aprendizaje automático (redes neuronales) para el análisis de datos temporales de alta frecuencia.

El análisis de la capacidad predictiva de los distintos métodos se realizará mediante su aplicación a mediciones horarias en tiempo real de la velocidad del viento en algunas localizaciones de Arabia Saudí para el periodo 2013-2016.

## Abstract

Fast and accurate short-term forecasts are becoming crucial in different areas, such as industry, where we may use forecasts to predict standstills or errors in the production processes, or in planning the electric budget by predicting the amount of energy generated from wind power.

The main objective of this work is to study and compare the predictive performance of some classic time series methods and machine learning based methods (neural networks) when analyzing high frequency temporal data.

The analysis of the predictive power of these methods will be performed by applying them to real-time data from hourly observations of the wind speed in some locations from Saudi Arabia during the period 2013-2016.

# Índice general

<b>1. Introducción y objetivos</b>	<b>1</b>
<b>2. Formulación</b>	<b>2</b>
2.1. Introducción a las series temporales . . . . .	2
2.1.1. Modelos estacionarios y función de autocorrelación . . . . .	4
2.2. Procesos estacionarios . . . . .	6
2.2.1. Propiedades básicas . . . . .	6
2.2.2. Propiedades de las funciones de media y autocorrelación muestrales . . . . .	8
2.3. Modelos ARMA . . . . .	9
2.3.1. Procesos ARMA(p,q) . . . . .	9
2.4. Estimación y predicción de procesos ARMA con R . . . . .	10
2.5. Introducción a las redes neuronales . . . . .	11
2.5.1. Funciones del perceptrón multicapa . . . . .	12
2.5.2. Entrenamiento de la red . . . . .	15
2.5.3. Propagación hacia atrás del error . . . . .	17
2.5.4. Regularización en Redes Neuronales . . . . .	21
2.6. Entrenamiento y predicción con modelos de Redes Neuronales en R	21
2.7. Intervalos de predicción en Redes Neuronales . . . . .	23
2.8. Cálculo de los RMSE de las predicciones a una, dos y tres horas	23
<b>3. Datos eólicos</b>	<b>25</b>
3.1. Velocidad residual del viento . . . . .	25
3.1.1. Obtención de las velocidades residuales . . . . .	26
3.1.2. Representación de las velocidades residuales medias . . . . .	27
<b>4. Resolución y Resultados</b>	<b>29</b>
4.1. Predicción con modelos ARMA . . . . .	29
4.1.1. Selección del modelo para la localización 73 . . . . .	29
4.1.2. Modelo ARMA final para la localización 73 . . . . .	36
4.1.3. Selección del modelo para la localización 2137 . . . . .	37
4.1.4. Modelo ARMA final para la localización 2173 . . . . .	41
4.2. Predicción con Redes Neuronales . . . . .	43
4.2.1. Transformación utilizando las frecuencias significativas . .	43

4.2.2. Transformación añadiendo información a las frecuencias significativas . . . . .	45
4.2.3. Intervalos de predicción del modelo final para la localización 73 . . . . .	48
4.2.4. Intervalos de predicción del modelo final para la localización 2137 . . . . .	49
<b>5. Conclusiones</b>	<b>51</b>
5.1. Comparación de los modelos en las distintas localizaciones . . . . .	51
5.2. Conclusiones sobre la implementación . . . . .	52
5.3. Líneas futuras . . . . .	53
<b>Lista de Figuras</b>	<b>56</b>
<b>Lista de Tablas</b>	<b>58</b>



## Capítulo 1

# Introducción y objetivos

El amplio consenso en relación a los crecientes efectos negativos del calentamiento global generado por los gases de efecto invernadero (IPCC, 2014) ha incitado a la comunidad internacional a buscar fuentes energéticas alternativas libres de carbono. Debido a esto, el aumento en el uso de fuentes energéticas renovables ha sido constante durante las últimas décadas.

Una de las principales fuentes de energía renovables es la energía eólica, en la cual la predicción precisa y rápida de la velocidad del viento y la potencia generada es crucial para cuantificar y planificar la demanda de energía en el tendido eléctrico.

Para realizar dichas predicciones utilizaremos distintos modelos de series temporales: (i) métodos estadísticos clásicos como los modelos Autorregresivos de Media Móvil (ARMA) y sus casos particulares, y (ii) métodos basados en aprendizaje automático (redes neuronales). El principal objetivo de este trabajo fin de máster es comparar la capacidad de dichos métodos para realizar predicciones a corto plazo en series temporales de alta frecuencia. Concretamente, se analizarán mediciones horarias de la velocidad del viento en algunas localizaciones de Arabia Saudí durante el periodo 2013-2016 (datos originales extraídos de <https://repository.kaust.edu.sa/handle/10754/667127>). En este trabajo nos centraremos en la predicción temporal de la velocidad del viento.

La memoria se estructura en cinco capítulos. En el Capítulo 2 se introducen los conceptos básicos necesarios para el estudio de series temporales utilizando métodos estadísticos clásicos como los modelos ARMA, y métodos basados en aprendizaje automático (redes neuronales). En el Capítulo 3 se describe el conjunto de datos con las mediciones horarias de la velocidad del viento en ciertas localizaciones de Arabia Saudí. En el Capítulo 4 se muestran los resultados obtenidos con los diferentes métodos considerados en este trabajo. Finalmente, en el Capítulo 5 se describen las principales conclusiones obtenidas, analizando las fortalezas y debilidades de los distintos métodos a la hora de realizar predicciones a corto plazo en series temporales de alta frecuencia.

# Capítulo 2

## Formulación

El objetivo de este trabajo es el estudio de distintos métodos de series de temporales y la comparación de su capacidad predictiva a corto plazo al aplicarlas sobre series temporales de alta frecuencia. En este capítulo se introducirán algunos de los conceptos básicos de series temporales (Brockwell et al., 2016) y redes neuronales (Bishop, 2006).

### 2.1. Introducción a las series temporales

Una **serie temporal** es un conjunto de valores  $x_t$  observados en tiempos  $t$  específicos. Podemos distinguir dos tipos principales de series temporales:

- Una *serie temporal discreta* es aquella en la que el conjunto  $T_0$  de los instantes en los que se realizan las observaciones es un conjunto discreto. Los datos en tiempo real con mediciones horarias de la velocidad del viento que utilizaremos posteriormente son un ejemplo de series discretas.
- Una *serie temporal continua* es aquella en la que las observaciones se obtienen de manera continua durante un intervalo de tiempo.

Como los datos analizados en este trabajo corresponden a una serie temporal discreta, trataremos con estas a partir de ahora.

El principal objetivo del análisis de series temporales es definir técnicas que permitan realizar predicciones a partir de los datos observados. Por lo tanto, una parte importante del análisis de series temporales consiste en establecer un modelo de probabilidad que represente el comportamiento de las variables  $X_t$  de la serie temporal. Debido a la naturaleza impredecible de las observaciones futuras supondremos que cada observación  $x_t$  es la realización de una variable aleatoria  $X_t$  concreta.

Un **modelo de series temporales** para los datos observados  $\{x_t\}$  es la especificación de la distribución conjunta de una secuencia de variables aleatorias  $\{X_t\}$  partiendo de la suposición de que  $\{x_t\}$  es una realización de estas. Normalmente se utiliza el término serie temporal tanto para referirse al conjunto

de variables aleatorias  $\{X_t\}$  como para la realización o valores observados de la serie,  $\{x_t\}$ .

Con frecuencia, especificar la distribución conjunta de las variables aleatorias  $\{X_t\}$  es difícil y requiere demasiados parámetros como para ser estimados a partir de los datos, por lo que muchos modelos de series temporales sólo especifican los momentos de **primer y segundo orden**, es decir,  $\mu_t = E(X_t)$  y  $Cov(X_{t+h}, X_t)$ , para  $t = 1, 2, \dots$  y  $h = 0, 1, 2, \dots$ . En el caso en que  $\{X_t\}$  tenga una distribución normal multivariante, los momentos de primer y segundo orden determinan completamente la distribución conjunta. Vamos a introducir distintos tipos de modelos de series temporales.

- **Series de media nula.** Son series cuyos valores oscilan en torno al valor 0 con una variabilidad constante. Un ejemplo de este tipo de series es el ruido iid.

Possiblemente el modelo de serie temporal más simple es aquel en el que las observaciones son simplemente variables aleatorias independientes e idénticamente distribuidas (iid) con media cero. Nos referimos a dicha secuencia  $X_1, X_2, \dots$  como **ruido iid**. Por definición, se puede escribir, para cualesquiera entero  $n$  y números reales  $x_1, \dots, x_n$ ,

$$P[X_1 \leq x_1, \dots, X_n \leq x_n] = P[X_1 \leq x_1] \dots P[X_n \leq x_n] = F(x_1) \dots F(x_n)$$

donde  $F(\cdot)$  es la función de distribución acumulada de cada una de las variables aleatorias idénticamente distribuidas  $X_1, X_2, \dots$ . En este no hay ninguna dependencia entre las observaciones, por lo que el conocimiento de las variables  $X_1, \dots, X_n$  no aporta información sobre el comportamiento de  $X_{n+h}$ ,  $h \geq 1$ . Aunque en un principio pueda parecer que el ruido iid es un proceso poco interesante a la hora de realizar predicciones, juega un papel clave para construir modelos más complicados de series temporales.

- **Series con tendencia.** Son aquellas series en las que se aprecia una tendencia ascendente o descendente en las observaciones, en este caso, utilizar un modelo de media nula es claramente inapropiado. Un modelo para series con tendencia es

$$X_t = m_t + Y_t$$

donde  $m_t$  es una función que cambia lentamente y se conoce como **componente de tendencia** e  $Y_t$  es una serie de media nula. Dicho componente de tendencia puede estimarse mediante el método de mínimos cuadrados.

- **Series estacionales.** Son aquellas series cuyo comportamiento se repite periódicamente. Un modelo para series estacionales, asumiendo la ausencia de tendencia es

$$X_t = s_t + Y_t$$

donde  $s_t$  es una función periódica de  $t$  con periodo  $S$  ( $s_{t-S} = s_t = s_{t+S}$ ) e  $Y_t$  es una serie de media nula.

### 2.1.1. Modelos estacionarios y función de autocorrelación

Se debe introducir el concepto de **serie estacionaria**, ya que los modelos básicos de series temporales son modelos para series estacionarias. Informalmente, se dice que una serie temporal  $\{X_t, t = 0, \pm 1, \dots\}$  es estacionaria si tiene propiedades estadísticas similares a las de la serie “desplazada”  $\{X_{t+h}, t = 0, \pm 1, \dots\}$  para cada entero  $h$ . Reduciendo el estudio a aquellas propiedades que dependen únicamente en los momentos de primer y segundo orden de  $\{X_t\}$ , se puede expresar esto de manera más precisa mediante las siguientes definiciones.

**Definición 2.1.1.** Sea  $\{X_t\}$  una serie temporal tal que  $E(X_t^2) < \infty$ . La **función de medias** de  $\{X_t\}$  es

$$\mu_t = E(X_t)$$

La **función de covarianzas** de  $\{X_t\}$  es

$$\gamma_{t,s} = \gamma(t, s) = Cov(X_t, X_s) = E((X_t - \mu_t)(X_s - \mu_s))$$

para todos los enteros  $t$  y  $s$ .

**Definición 2.1.2.**  $\{X_t\}$  es una serie (**débilmente**) estacionaria si

- (I)  $\mu_t$  es independiente de  $t$ .
- (II)  $\gamma_{t+h,h}$  es independiente de  $t$  para cada  $h$ .

Una serie temporal  $\{X_t, t = 0, \pm 1, \dots\}$  es estrictamente estacionaria si  $(X_1, \dots, X_n)$  y  $(X_{1+h}, \dots, X_{n+h})$  tienen la misma distribución conjunta para todos los enteros  $h$  y  $n > 0$ . Si  $X_t$  es estrictamente estacionaria y  $E(X_t^2) < \infty$ , es fácil comprobar que  $\{X_t\}$  es débilmente estacionaria. Utilizaremos el término serie estacionaria para referirnos a las series débilmente estacionarias de la Definición 2.1.2.

**Definición 2.1.3.** Sea  $\{X_t\}$  una serie temporal estacionaria. La **función de autocovarianza** (ACVF) de  $\{X_t\}$  en el retardo  $h$  es

$$\gamma(h) = \gamma_h = Cov(X_{t+h}, X_t).$$

La **función de autocorrelación** (ACF) de  $\{X_t\}$  en el retardo  $h$  es

$$\rho(h) = \rho_h = \frac{\gamma_h}{\gamma_0} = Cor(X_{t+h}, X_t),$$

donde  $\gamma_0 = \gamma(0, 0) = Cov(X_t, X_t) = Var(X_t)$ .

Estos conceptos pueden ilustrarse con un ejemplo

**Ejemplo 1.** Ruido Blanco

Sea  $X_t$  una secuencia de variables aleatorias incorreladas, cada una con media cero y varianza  $\sigma^2$ , entonces, está claro que  $X_t$  es estacionaria, ya que su función de autocovarianza

$$\gamma_h = \gamma(t+h, t) = \begin{cases} \sigma^2, & \text{si } h = 0, \\ 0, & \text{si } h \neq 0 \end{cases}$$

no depende de  $t$ . Nos referiremos a dicha secuencia como **Ruido Blanco** (con media 0 y varianza  $\sigma^2$ ), y lo indicaremos mediante la notación

$$\{X_t\} \sim \text{WN}(0, \sigma^2)$$

### La función de autocorrelación muestral

Aunque se ha definido la función de autocorrelación para modelos de series temporales, en la práctica no se dispone del modelo, sino de los valores observados  $\{x_1, x_2, \dots, x_n\}$ . Una de las herramientas para evaluar el grado de dependencia en los datos y poder elegir un modelo para los datos que refleje dicha dependencia es la **función de autocorrelación muestral** (ACF muestral). Si se cree que los datos son la realización de una serie temporal estacionaria  $\{X_t\}$ , la ACF muestral nos dará una estimación de la ACF teórica de  $\{X_t\}$ , y permitirá elegir uno de los modelos de series estacionarias.

**Definición 2.1.4.** Sean  $x_1, \dots, x_n$  valores observados de una serie temporal. La **media muestral** de  $x_1, \dots, x_n$  es

$$\bar{x} = \frac{1}{n} \sum_{t=1}^n x_t.$$

La **función de autocovarianza muestral** es

$$\hat{\gamma}_h = n^{-1} \sum_{t=1}^{n-|h|} (x_{t+|h|} - \bar{x})(x_t - \bar{x}), \quad -n < h < n.$$

La **función de autocorrelación muestral** es

$$\hat{\rho}_h = \frac{\hat{\gamma}_h}{\hat{\gamma}_0}, \quad -n < h < n.$$

### La función de autocorrelación parcial

**Definición 2.1.5.** La **Función de Autocorrelación Parcial (PACF)** de un proceso ARMA  $\{X_t\}$  es la función  $\alpha(\cdot)$  definida por las ecuaciones

$$\alpha(0) = 1$$

y

$$\alpha(h) = \phi_{hh}, \quad h \geq 1$$

donde  $\phi_{hh}$  es la última componente de

$$\phi_h = \Gamma_h^{-1} \gamma_h, \quad (1.1.1)$$

con  $\Gamma_h = [\gamma(i-j)]_{i,j=1}^h$  y  $\gamma_h = [\gamma(1), \gamma(2), \dots, \gamma(h)]'$

Para un conjunto cualquiera de observaciones  $\{x_1, \dots, x_n\}$  con  $x_i \neq x_j$  para algunos  $i$  y  $j$ , la **PACF muestral**  $\hat{\alpha}(h)$  viene dada por

$$\hat{\alpha}(0) = 1$$

y

$$\hat{\alpha}(h) = \hat{\phi}_{hh}, \quad h \geq 1,$$

donde  $\hat{\phi}_{hh}$  es la última componente de

$$\hat{\phi}_h = \hat{\Gamma}_h^{-1} \hat{\gamma}_h, \quad (1.1.2)$$

La importancia de la PACF muestral radica en que al analizarla junto con la ACF permite hipotetizar sobre los parámetros  $p$  y  $q$  más adecuados para el modelo ARMA(p,q).

## 2.2. Procesos estacionarios

El objetivo del análisis de series temporales es predecir una serie que normalmente es no determinista ya que contiene un componente aleatorio. Si dicho componente aleatorio es estacionario es posible desarrollar técnicas que permitan predecir sus valores futuros.

### 2.2.1. Propiedades básicas

Las ACVF y ACF proporcionan una medida útil del grado de dependencia entre los valores de una serie temporal en distintos instantes de tiempo, y por esta razón juegan un papel importante al considerar la predicción de valores futuros de la serie en función de los valores pasados y presentes.

#### Propiedades básicas de $\gamma(\cdot)$

1.  $\gamma(0) \geq 0$ ,
2.  $|\gamma(h)| \leq \gamma(0)$  para todo  $h$ ,
3.  $\gamma(\cdot)$  es par, es decir,  $\gamma(h) = \gamma(-h)$ .

La primera propiedad se basa en el hecho de que  $Var(X_t) \geq 0$ . La segunda es la consecuencia inmediata del hecho de que  $|\rho(h)| = \frac{|\gamma(h)|}{\gamma(0)} \leq 1$ . La tercera se deduce inmediatamente de observar que

$$\gamma(h) = Cov(X_{n+h}, X_n) = Cov(X_n, X_{n+h}) = \gamma(-h)$$

**Definición 2.2.1.**  $\{X_t\}$  es una serie temporal **estRICTAMENTE ESTACIONARIA** si

$$(X_1, \dots, X_n)' \stackrel{d}{=} (X_{1+h}, \dots, X_{n+h})'$$

para todos los enteros  $h$  y  $n \geq 1$ . (Se utiliza  $\stackrel{d}{=}$  para indicar que ambos vectores aleatorios tienen la misma función de distribución conjunta)

**Propiedades de las series temporales estRICTAMENTE ESTACIONARIAS**

- a. Las variables aleatorias  $X_t$  son idénticamente distribuidas,
- b.  $(X_t, X_{t+h})' \stackrel{d}{=} (X_1, X_{1+h})'$  para todos los enteros  $t$  y  $h$ ,
- c.  $X_t$  es débilmente estacionaria si  $E(X_t) < \infty$  para todo  $t$ ,
- d. Que  $X_t$  sea débilmente estacionaria no implica que sea estrictamente estacionaria,
- e. Las secuencias iid son estrictamente estacionarias.

Uno de los modos más simples de construir una serie temporal  $\{X_t\}$  estacionaria (y por tanto débilmente estacionaria si  $E(X_t) < \infty$ ) es “filtrar” una secuencia de variables aleatorias independientes e idénticamente distribuidas (iid). Sea  $\{Z_t\}$  una secuencia iid, que es estrictamente estacionaria por la propiedad (e), se define

$$X_t = g(Z_t, Z_{t-1}, \dots, Z_{t-q}) \quad (2.1.3)$$

para alguna función  $g$  de variable real. Entonces,  $X_t$  es estrictamente estacionaria, ya que  $(Z_{t+h}, \dots, Z_{t+h-q}) \stackrel{d}{=} (Z_t, \dots, Z_{t-q})$  para todos los enteros  $h$ . Se sigue de la ecuación 2.1.3 que  $\{X_t\}$  es **q-dependiente**, es decir, que  $X_s$  y  $X_t$  son independientes siempre que  $|t-s| > q$  (una secuencia iid es 0-dependiente). De manera similar, centrándonos en los momentos de segundo orden, se dice que una serie es **q-correlada** si  $\gamma(h) = 0$  siempre que  $|h| > q$ . Una secuencia de ruido blanco es 0-correlada, mientras que el proceso MA(1) es 1-correlado. El proceso de Media Móvil de orden  $q$  que se define a continuación es  $q$ -correlado.

**Definición 2.2.2.** Proceso MA( $q$ ).  $\{X_t\}$  es un **proceso de Media Móvil de orden  $q$**  si

$$X_t = Z_t + \theta_1 Z_{t-1} + \theta_q Z_{t-q} \quad (2.1.4)$$

donde  $\{Z_t\} \sim WN(0, \sigma^2)$  y  $\theta_1, \dots, \theta_q$  son constantes.

Es fácil comprobar que (2.1.4) define una serie estacionaria que es estrictamente estacionaria si  $\{Z_t\}$  es ruido iid, en ese caso, (2.1.4) es el caso particular de (2.1.3) en que la función  $g$  es lineal.

La importancia de los procesos MA( $q$ ) radica en el hecho de que todo proceso  $q$ -correlado es un proceso MA( $q$ ) (Brockwell and Davis, 2009).

## 2.2.2. Propiedades de las funciones de media y autocorrelación muestrales

Un proceso estacionario  $\{X_t\}$  queda caracterizado, al menos desde una perspectiva de segundo orden, por su media  $\mu$  y su función de autocovarianza  $\gamma(\cdot)$ . Por lo que la estimación de  $\mu$ ,  $\gamma(\cdot)$  y la función de autocorrelación  $\rho(\cdot) = \gamma(\cdot)/\gamma(0)$  a partir de las observaciones  $x_1, \dots, x_n$  cumple un papel crucial en los problemas de inferencia, y en particular a la hora de construir un modelo adecuado para los datos observados.

### Estimación de $\mu$

El estimador de la media  $\mu$  de un proceso estacionario es la media muestral

$$\bar{X}_n = \frac{X_1 + \dots + X_n}{n} \quad (2.2.5)$$

Dicho estimador es insesgado, ya que

$$E(\bar{X}_n) = \frac{E(X_1) + \dots + E(X_n)}{n} = \mu$$

Para realizar inferencia sobre  $\mu$  a partir de la media muestral  $\bar{X}_n$ , es necesario conocer la distribución de  $\bar{X}_n$  o una aproximación de ésta. Para muchas series temporales, y en particular para los modelos lineales y ARMA,  $\bar{X}_n$  es aproximadamente normal con media  $\mu$  y varianza  $\frac{1}{n} \sum_{|h|<\infty} \gamma(h)$  para  $n$  grande. Por lo que un intervalo de confianza aproximado al 95 % es

$$(\bar{X}_n - 1.96v^{1/2}/\sqrt{n}, \bar{X}_n + 1.96v^{1/2}/\sqrt{n}) \quad (2.2.6)$$

donde  $v = \sum_{|h|<\infty} \gamma(h)$ . Generalmente no se conoce  $v$ , por lo que será necesario estimarlo a partir de los datos.

### Estimación de $\gamma(\cdot)$ y $\rho(\cdot)$

En la Sección 2.1.1 se han definido las funciones de autocovarianza y autocorrelación muestrales como

$$\hat{\gamma}(h) = \frac{1}{n} \sum_{t=1}^{n-|h|} (X_{t+|h|} - \bar{X}_n)(X_t - \bar{X}_n) \quad (2.2.7)$$

y

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)} \quad (2.2.8)$$

Ambos estimadores  $\hat{\gamma}(h)$  y  $\hat{\rho}(h)$  son sesgados aunque se sustituya el divisor  $n$  por  $n - |h|$ . A pesar de esto, por lo general los estimadores son asintóticamente

insesgados. La ACVF tiene la propiedad deseable de que para cada  $k \geq 1$  tanto la matriz de covarianzas muestral  $k$ -dimensional

$$\hat{\Gamma}_k = \begin{bmatrix} \hat{\gamma}(0) & \hat{\gamma}(1) & \cdots & \hat{\gamma}(k-1) \\ \hat{\gamma}(1) & \hat{\gamma}(0) & \cdots & \hat{\gamma}(k-2) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\gamma}(k-1) & \hat{\gamma}(k-2) & \cdots & \hat{\gamma}(0) \end{bmatrix} \quad (2.2.9)$$

como la matriz de correlaciones muestral  $\hat{R}_k = \hat{\Gamma}_k / \hat{\gamma}(0)$  son semidefinidas positivas.

La ACF muestral cumple un papel importante en la selección de modelos adecuados para los datos. Para realizar inferencia sobre  $\rho(h)$ , es necesario conocer la distribución muestral del estimador  $\hat{\rho}(h)$ . Aunque la distribución de  $\hat{\rho}(h)$  es intratable incluso para muestras de las series temporales más básicas, habitualmente se puede obtener una buena aproximación mediante una distribución normal para muestras grandes. Para modelos lineales y en particular para modelos ARMA, la distribución de  $\hat{\rho} = (\hat{\rho}(1), \dots, \hat{\rho}(k))'$  para muestras grandes es aproximadamente normal, esto es

$$\hat{\rho} \sim N(\rho, n^{-1}W) \quad (2.2.10)$$

donde  $\rho = (\rho(1), \dots, \rho(k))'$  y  $W$  es la matriz de covarianzas cuyo elemento  $(i, j)$  viene dado por la **fórmula de Bartlett** (Brockwell et al., 2016, p. 53).

## 2.3. Modelos ARMA

Se va a proceder a introducir una importante familia paramétrica de series temporales estacionarias, los procesos Autorregresivos de Media Móvil o ARMA. Para una gran cantidad de funciones de autocovarianza  $\gamma(\cdot)$  es posible encontrar un proceso ARMA  $\{X_t\}$  cuya ACVF  $\gamma_X(\cdot)$  sea una buena aproximación de  $\gamma(\cdot)$ . En particular, para cualquier entero positivo  $K$  existe un proceso ARMA  $\{X_t\}$  tal que  $\gamma_X(h) = \gamma(h)$  para  $h = 0, 1, \dots, K$ . Por esta y otras razones, los procesos ARMA cumplen un papel crucial en el modelado de datos de series temporales.

### 2.3.1. Procesos ARMA(p,q)

**Definición 2.3.1.**  $\{X_t\}$  es un **proceso ARMA(p,q)** si  $\{X_t\}$  es estacionario y para todo  $t$

$$X_t - \phi_1 X_{t-1} - \dots - \phi_p X_{t-p} = Z_t + \theta_1 Z_{t-1} + \dots + \theta_q Z_{t-q}, \quad (3.1.11)$$

donde  $Z_t \sim WN(0, \sigma^2)$  y los polinomios  $(1 - \phi_1 z - \dots - \phi_p z^p)$  y  $(1 + \theta_1 z + \dots + \theta_q z^q)$  no tienen factores en común.

El proceso  $\{X_t\}$  se dice **procesos ARMA(p,q) con media  $\mu$**  si  $\{X_t - \mu\}$  es un proceso ARMA(p,q).

Suele resultar más cómodo reescribir la ecuación (3.1.11) como

$$\phi(B)X_t = \theta(B)Z_t, \quad (3.1.12)$$

donde  $\phi(\cdot)$  y  $\theta(\cdot)$  son los polinomios de grado  $p$  y  $q$

$$\phi(z) = 1 - \phi_1z - \dots - \phi_pz^p,$$

$$\theta(z) = 1 + \theta_1z + \dots + \theta_qz^q$$

y  $B$  es el operador retardo ( $B^j X_t = X_{t-j}$ ,  $B^j Z_t = Z_{t-j}$ ,  $j = 0, 1, \dots$ ).

La serie  $\{X_t\}$  se dice **proceso Autorregresivo de orden p** (AR(p)) si  $\theta(z) \equiv 1$ , y se dice **proceso de Medias Móviles de orden q** (MA(q)) si  $\phi(z) \equiv 1$ .

A continuación se definen algunas de sus principales características.

#### Existencia y unicidad

La solución estacionaria  $\{X_t\}$  de (3.1.11) existe (y es la única solución estacionaria) si y sólo si

$$\phi(z) = 1 - \phi_1z - \dots - \phi_pz^p \neq 0 \text{ para todo } |z| = 1 \quad (3.1.13)$$

## 2.4. Estimación y predicción de procesos AR-MA con R

Vamos a realizar la estimación y predicción de los distintos procesos ARMA ayudándonos de las librería `tseries` (Trapletti and Hornik, 2020) y `forecast` (Hyndman et al., 2021) del software R.

Ambas librerías requieren del paquete `stats`, que define el tipo de objeto `ts` utilizado tanto por `tseries` como por `forecast`. Además este paquete define la función `arima(x,c(p,d,q))`, que nos permite ajustar un modelo ARMA con los parámetros  $(p, q)$  que fijemos a la serie temporal  $x$  (fijando  $d = 0$ ).

El paquete `tseries` proporciona funcionalidades básicas para tratar con los objetos de tipo `ts`, que representan a las series temporales en el lenguaje R. En nuestro caso utilizaremos dos funciones que nos permiten contrastar la hipótesis de que la serie temporal a tratar sea estacionaria.

- La función `adf.test(x)` realiza un contraste aumentado de Dickey-Fuller (Dickey and Fuller, 1979) para la hipótesis nula de que la serie temporal  $x$  tiene una raíz unitaria, siendo su hipótesis alternativa que la serie temporal sea estacionaria.
- La función `kpsstest(x)` realiza un contraste de Kwiatkowski–Phillips–Schmidt–Shin (KPSS) (Kwiatkowski et al., 1992) para la hipótesis nula de que la serie  $x$  sea estacionaria.

El paquete `forecast` nos proporcionará el resto de las funcionalidades que utilizaremos para estimar y predecir procesos ARMA.

- Las funciones `Acf(x)` y `Pacf(x)`, que calculan (y por defecto muestran) las ACF y PACF estimadas de la serie temporal `x`, y permiten hipotetizar sobre los modelos que pueden resultar adecuados para modelizar la serie temporal estudiada.
- La función `auto.arima` realiza la selección del mejor modelo ARIMA (por defecto de acuerdo al criterio AIC) entre las combinaciones de parámetros que definamos como argumentos. Como en nuestro caso la serie temporal de entrada es estacionaria la función realizará la selección del mejor modelo ARMA.
- La función `checkresiduals(m)` también juega un papel clave en la estimación de modelos ARMA, realizando un test de Ljung-Box (Ljung and Box, 1978) para la hipótesis nula de independencia de los residuos del modelo `m`. Por defecto muestra la serie temporal de los residuales del modelo que ajustamos junto con su ACF muestral e histograma, de modo que se pueda analizar gráficamente el comportamiento de los residuales del modelo.
- Este paquete también introduce la función `forecast(x,h,m)`, que permite obtener predicciones a futuro para un número de periodos `h` seleccionando junto con sus intervalos de predicción correspondientes para una serie temporal `x` utilizando el modelo `m` de series temporales.

## 2.5. Introducción a las redes neuronales

El modelo lineal de regresión más simple consiste en una combinación lineal de las variables de entrada

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \dots + w_D x_D, \quad (5.0.14)$$

donde  $x = (X_1, \dots, X_D)'$ . Este modelo se conoce habitualmente como **regresión lineal**. La propiedad clave de este modelo es que es una función lineal de los parámetros  $w_0, \dots, w_D$ . Además, también es una función lineal de las variables de entrada  $x_i$ , lo cual impone limitaciones notables al modelo. Por lo que se extiende la clase de modelos considerando combinaciones lineales de funciones no lineales fijadas de las variables de entrada, de la forma

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) \quad (5.0.15)$$

donde  $\phi_j(\mathbf{x})$  se dicen **funciones base**. Tomando el máximo valor del índice  $j$  como  $M - 1$  se tiene un modelo con un total de  $M$  parámetros. El parámetro  $w_0$  se dice **parámetro de sesgo**. Por comodidad se suele definir la “función base” dummy  $\phi_0(\mathbf{x}) = 1$  de modo que

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}' \boldsymbol{\phi}(\mathbf{x}) \quad (5.0.16)$$

donde  $\mathbf{w} = (w_1, \dots, w_{M-1})'$  y  $\phi = (\phi_1, \dots, \phi_{M-1})'$ .

Un enfoque para adaptar los modelos lineales de regresión a problemas a mayor escala consiste en fijar de antemano el número de funciones base pero permitir que estas sean adaptativas, o en otras palabras, utilizar formas paramétricas de las funciones base cuyos parámetros se ajusten durante el entrenamiento. El más exitoso de los modelos de este tipo en el contexto del reconocimiento de patrones es la red neuronal conocida como **perceptrón multicapa** o red neuronal *feed-forward*. Uno de los principales problemas de esta adaptación es que la función de verosimilitud, que es la base del entrenamiento de la red neuronal, no es convexa. Sin embargo, en la práctica suele merecer la pena invertir más recursos computacionales en la fase de entrenamiento para obtener un modelo compacto que sea rápido a la hora de procesar nuevos datos.

Aunque el término “red neuronal” tiene su origen en intentos de encontrar representaciones matemáticas del procesado de información en sistemas biológicos (Rumelhart et al., 1986), desde la perspectiva de las aplicaciones prácticas en el reconocimiento de patrones, el realismo biológico impone restricciones completamente innecesarias. Por lo que resulta práctico centrarse en las redes neuronales como modelos eficientes para el reconocimiento de patrones estadísticos.

Se comenzará considerando la forma funcional del modelo de red neuronal, incluyendo la parametrización específica de las funciones base, para posteriormente tratar el problema de determinar los parámetros de la red en un contexto de máxima verosimilitud, que involucra la resolución de problemas de optimización no lineales. Esto requiere evaluar las derivadas de la log verosimilitud respecto a los parámetros de la red, que pueden obtenerse de manera eficiente utilizando la técnica de la propagación hacia atrás de los errores.

### 2.5.1. Funciones del perceptrón multicapa

Los modelos lineales de regresión se basan en combinaciones lineales de funciones base no lineales  $\phi_j(\mathbf{x})$  fijadas previamente, y toman la forma

$$y(\mathbf{x}, \mathbf{w}) = f \left( \sum_{j=1}^M w_j \phi_j(\mathbf{x}) \right) \quad (5.1.17)$$

La idea es extender el modelo para hacer que las funciones base  $\phi_j(\mathbf{x})$  dependan de parámetros que se ajusten, junto con los coeficientes  $\{w_j\}$ , durante el entrenamiento. Las redes neuronales utilizan funciones base que siguen la misma forma que (5.1.17).

Esto conduce al modelo básico de red neuronal que puede ser descrito como una serie de transformaciones funcionales. Se comienza construyendo  $M$  combinaciones lineales de las variables de entrada  $x_1, \dots, x_D$  de la forma

$$a_j = \sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)} \quad (5.1.18)$$

donde  $j = 1, \dots, M$  y el superíndice (1) indica que los parámetros correspondientes pertenecen a la primera “capa” de la red. Se utiliza el término **pesos** para referirse a los parámetros  $w_{ji}^{(1)}$  y **términos de sesgo** para referirse a los parámetros  $w_{j0}^{(1)}$ , repitiendo la nomenclatura utilizada para los modelos lineales de regresión. Los valores  $a_j$  se conocen como **activaciones**. Cada una de estas se transforma utilizando una **función de activación** diferenciable y no lineal  $h(\cdot)$  para obtener

$$z_j = h(a_j) \quad (5.1.19)$$

Dichas cantidades corresponden a las salidas de las funciones base en (5.1.17) que en el contexto de las redes neuronales se llaman **unidades ocultas**. Las funciones no lineales  $h(\cdot)$  que se consideran habitualmente suelen ser funciones sigmoides como la logística o la tangente hiperbólica. Siguiendo (5.1.17), estos valores vuelven a combinarse linealmente para obtener las activaciones de la unidad de salida

$$a_k = \sum_{j=1}^M w_{kj}^{(2)} z_j + w_{k0}^{(2)} \quad (5.1.20)$$

donde  $k = 1, \dots, K$ , y  $K$  es el número total de salidas. Esta transformación corresponde a la segunda capa del perceptrón, y de nuevo, los  $w_{k0}^{(2)}$  son parámetros de sesgo. Finalmente, las activaciones de las unidades de salida se transforman mediante una función de activación adecuada para obtener el conjunto de salidas  $y_k$  del perceptrón. La elección de la función de activación se realiza en función de la distribución de las variables objetivo. Para problemas clásicos de regresión, la función de activación es la identidad, es decir,  $y_k = a_k$ . Mientras que para problemas de clasificación binarios, la activación de cada unidad de salida,  $a_k$ , se transforma utilizando la función sigmoide logística de modo que

$$y_k = \sigma(a_k) \quad (5.1.21)$$

donde

$$\sigma(a) = \frac{1}{1 + e^{-a}} \quad (5.1.22)$$

Posteriormente se pueden combinar las distintas etapas para obtener la función global de la red, la cual, para funciones de activación sigmoides de las unidades de salida, toma la forma

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left( \sum_{j=1}^K w_{kj}^{(2)} h \left( \sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right) \quad (5.1.23)$$

donde el conjunto de todos los pesos y parámetros de sesgo se agrupa en un vector  $\mathbf{w}$ . Por lo que el modelo de red neuronal es una función no lineal de un conjunto de variables de entrada  $\{x_i\}$  a un conjunto de variables de salida  $\{y_k\}$  controlada por un vector  $\mathbf{w}$  de parámetros ajustables.

El proceso de evaluación de (5.1.23) puede interpretarse como la propagación hacia adelante de la información a través de la red.

Los parámetros de sesgo pueden incluirse en el conjunto de pesos definiendo una variable de entrada adicional  $x_0$  con valor fijo  $x_0 = 1$ , de modo que (5.1.18) toma la forma

$$a_j = \sum_{i=0}^D w_{ji}^{(1)} x_i \quad (5.1.24)$$

Realizando un proceso similar podemos añadir los parámetros de sesgo de segunda capa a los pesos, obteniendo la siguiente expresión de la función global de la red

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left( \sum_{j=0}^K w_{kj}^{(2)} h \left( \sum_{i=0}^D w_{ji}^{(1)} x_i \right) \right) \quad (5.1.25)$$

El modelo de red neuronal se compone de dos etapas de procesamiento, cada uno de los cuales recuerda a un modelo de perceptrón. Esta es la razón por la que la red neuronal también se conoce como Perceptrón Multicapa (MLP).

Si las funciones de activación de todas las unidades ocultas se toman lineales, entonces, como la composición de transformaciones lineales es una transformación lineal, se puede encontrar una red equivalente sin unidades ocultas. Sin embargo, si el número de unidades ocultas es menor que el número de unidades de entrada o salida, las transformaciones generadas por la red no son todo lo generales que pueden llegar a ser debido a la pérdida de información generada por la reducción de dimensionalidad al pasar a la capa oculta.

La arquitectura de red descrita hasta ahora puede generalizarse fácilmente considerando capas de procesamiento adicionales. Cada una de ellas consta de una combinación lineal ponderada de la forma (5.1.20) seguida de una transformación elemento a elemento utilizando una función de activación no lineal. Cabe notar que existe una cierta confusión en la literatura en lo referente a la terminología para contar el número de capas en las redes neuronales. Por lo que la red descrita hasta ahora podría describirse como una red de 3 capas (considerando el número de capas de unidades, y considerando la capa de entrada) o como una red de una capa oculta (considerando el número de capas de unidades ocultas). Se utilizará la terminología en la que la arquitectura descrita se dice red de dos capas, ya que el número de capas de pesos adaptativos es importante a la hora de determinar las propiedades de la red.

Como existe una correspondencia directa entre el diagrama de la red y su función global, pueden desarrollarse redes más generales considerando diagramas de red más complejos. Sin embargo, dichos diagramas deben restringirse a la arquitectura *feed-forward* (a la cual pertenece el perceptrón multicapa), dicho de otra manera, a aquella arquitectura que no tiene ciclos, para asegurar que las salidas son funciones deterministas de las entradas. La Figura 2.1 ilustra un ejemplo de esto. Cada unidad (oculta o de salida) computa una función dada por

$$z_k = h \left( \sum_j w_{kj} z_j \right) \quad (5.1.26)$$

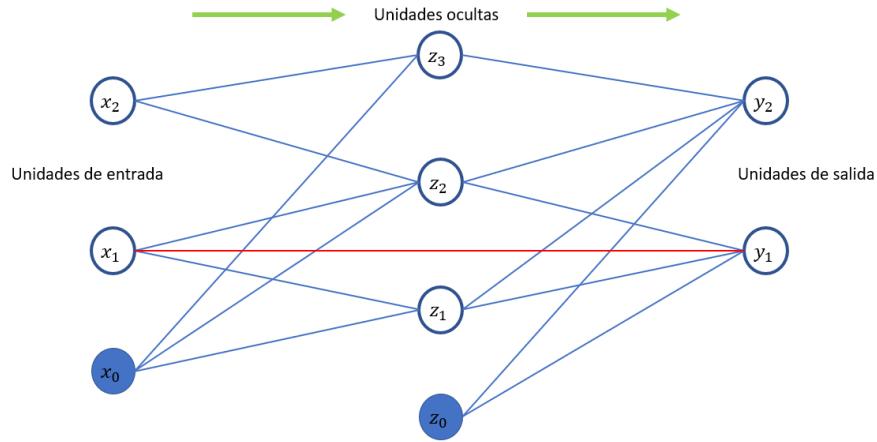


Figura 2.1: Ejemplo de un perceptrón multicapa general

donde la suma recorre todas las unidades con conexiones hacia la unidad  $k$  (incluyendo el término de sesgo). Para un conjunto dado de valores aplicado sobre las entradas de la red, la aplicación sucesiva de (5.1.26) permite evaluar las activaciones de todas las unidades de la red, incluyendo las de las unidades de salida.

### 2.5.2. Entrenamiento de la red

Hasta ahora, se han descrito las redes neuronales como una clase general de funciones paramétricas no lineales de un vector  $\mathbf{x}$  de variables de entrada a un vector  $\mathbf{y}$  de variables de salida. Un enfoque simple para determinar los parámetros de la red consiste en minimizar una suma de cuadrados del error. Dado un conjunto de entrenamiento compuesto por los vectores de entrada  $\{\mathbf{x}_n\}$ , donde  $n = 1, \dots, N$ , junto con sus vectores objetivo  $\mathbf{t}_n$ , se minimiza la función de error

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n\| \quad (5.2.27)$$

#### Optimización de los parámetros

La siguiente tarea a realizar consiste en encontrar un vector de pesos  $\mathbf{w}$  que minimice la función de error  $E(\mathbf{w})$  elegida. Para esto, resulta útil disponer de una imagen geométrica de la función de error, que puede verse como una superficie sobre el espacio de los pesos como se ilustra en la Figura 2.2. Nótese que si se toma un pequeño paso en el espacio de estados desde  $\mathbf{w}$  a  $\mathbf{w} + \delta\mathbf{w}$ , entonces el cambio en la función de error es  $\delta E \simeq \delta\mathbf{w}' \nabla E(\mathbf{w})$ , donde el vector  $\nabla E(\mathbf{w})$  apunta a la dirección de mayor crecimiento de la función de error. Como

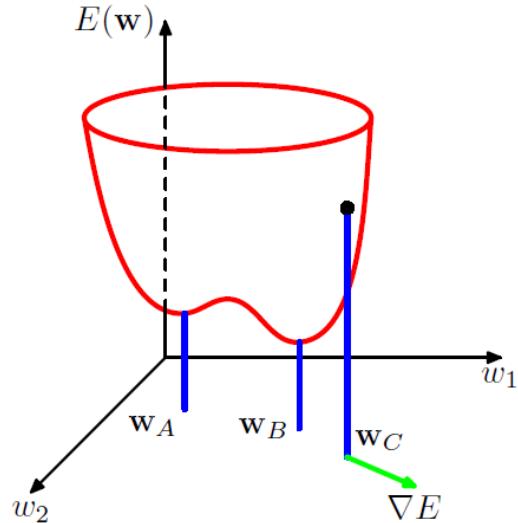


Figura 2.2: Representación de la función de error  $E(\mathbf{w})$  como una superficie sobre el espacio de los pesos. Fuente: Bishop (2006)

el error  $E(\mathbf{w})$  es una función continua suave de  $\mathbf{w}$ , su menor valor tendrá lugar en un punto del espacio de pesos en el que el gradiente de la función de error se desvanezca, de modo que

$$\nabla E(\mathbf{w}) = 0 \quad (5.2.28)$$

ya que en otro caso se puede tomar un pequeño paso en la dirección de  $-\nabla E(\mathbf{w})$ , reduciendo el error. Se pretende encontrar un vector  $\mathbf{w}$  tal que  $E(\mathbf{w})$  toma su menor valor. Sin embargo, la función de error suele tener una gran dependencia no lineal de los pesos y los términos de sesgo, por lo que suelen existir varios puntos del espacio de pesos en los que el gradiente se desvanece (o es numéricamente muy pequeño). Es más, normalmente existen múltiples mínimos no equivalentes. Un mínimo que corresponde al menor valor de la función de error para cualquier vector de pesos se dice **mínimo global**. Cualquier mínimo con un valor correspondiente de la función de error mayor se dice **mínimo local**. Para aplicar satisfactoriamente una red neuronal puede no ser necesario encontrar el mínimo global (en general no se conoce si se ha encontrado el mínimo global) pero puede ser necesario comparar distintos mínimos locales para encontrar una solución suficientemente buena.

Como no es posible encontrar una solución analítica general de la ecuación  $\nabla E(\mathbf{w})$  se recurre a procedimientos numéricos iterativos. La mayoría de dichas técnicas implican seleccionar un valor inicial  $\mathbf{w}(0)$  del vector de pesos para posteriormente desplazarse a través del espacio de pesos en una sucesión de pasos de la forma

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta \mathbf{w}^{(\tau)} \quad (5.2.29)$$

donde  $\tau$  representa el paso de la iteración. La diferencia entre los diferentes algoritmos suele consistir en seleccionar diferentes actualizaciones del vector de pesos  $\Delta\mathbf{w}^{(\tau)}$

### Optimización de Descenso por Gradiente

El enfoque más simple para utilizar la información del gradiente es tomar la actualización de los pesos en (5.2.29) para que tome un pequeño paso en la dirección del gradiente negativo, de modo que

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)}) \quad (5.2.30)$$

donde el parámetro  $\eta > 0$  se conoce como **ratio de aprendizaje**. Tras cada actualización, el gradiente es reevaluado para el nuevo vector de pesos y se repite el proceso. Nótese que la función de error se define respecto al conjunto de entrenamiento, por lo que cada paso requiere procesar todo el conjunto de entrenamiento para evaluar  $\nabla E$ . Los métodos que utilizan todo el conjunto de datos al mismo tiempo se conocen como **métodos por lotes**. En cada paso el vector de pesos se desplaza en la dirección de mayor decrecimiento de la función de error, por lo que este enfoque se conoce como **descenso por gradiente**.

Para encontrar un mínimo suficientemente bueno, puede ser necesario ejecutar un algoritmo basado en el gradiente múltiples veces, comenzando desde distintos puntos iniciales aleatorios, y comparar su eficacia en un conjunto de validación independiente.

### 2.5.3. Propagación hacia atrás del error

Una vez definido el método iterativo para optimizar los pesos, se requiere una técnica eficiente para evaluar el gradiente de la función de error  $E(\mathbf{w})$  para un perceptrón multicapa. Este objetivo puede alcanzarse utilizando una estrategia de transferencia local de mensajes en la que la información se transmite alternativamente hacia adelante y hacia atrás a través de la red y se conoce como **propagación hacia atrás del error** (*error backpropagation* o simplemente *backprop*).

Se debe aclarar que el término propagación hacia atrás se utiliza en la literatura de redes neuronales para referirse a varios conceptos. Por ejemplo, la arquitectura de perceptrón multicapa a veces es llamada red de propagación hacia atrás. El término backpropagation también se utiliza para describir el entrenamiento de un perceptrón multicapa utilizando el descenso por gradiente aplicado a una función de error de suma de cuadrados. Para aclarar dicha terminología resulta útil analizar el proceso de entrenamiento más cuidadosamente. La mayoría de los algoritmos de entrenamiento implican un proceso iterativo para minimizar una función de error, ajustando los pesos en una secuencia de pasos. En cada paso se pueden distinguir dos etapas. En la primera se deben evaluar las derivadas de la función de error respecto a los pesos. La contribución de la propagación hacia atrás consiste en aportar un método computacionalmente

eficiente para evaluar dichas derivadas. Como es en esta etapa en la que se realiza la propagación hacia atrás de los errores, cobra sentido utilizar el término backpropagation para describir la evaluación de las derivadas. En la segunda etapa, se utilizan las derivadas para computar los ajustes a realizar en los pesos. La más simple de estas, y la considerada originalmente por Rumelhart et al. (1986), implica el uso del descenso por gradiente. Es importante reconocer la diferencia entre las dos etapas.

### Evaluación de las derivadas de la función de error

Considérese un modelo lineal simple en el que las salidas  $y_k$  son combinaciones lineales de las variables de entrada  $x_i$  tal que

$$y_k = \sum_i w_{ki} x_i \quad (5.3.31)$$

junto con una función de error que, para un patrón de entrada  $\mathbf{x}_n$  toma la forma

$$E_n = \frac{1}{2} \sum_k (y_{nk} - t_{nk})^2 \quad (5.3.32)$$

donde  $y_{nk} = y_k(\mathbf{x}_n, \mathbf{w})$ . El gradiente de dicha función de error respecto a un peso  $w_{ji}$  es

$$\frac{\partial E_n}{\partial w_{ji}} = (y_{nj} - t_{nj}) x_{ni} \quad (5.3.33)$$

el cual puede interpretarse como un cálculo “local” que involucra el producto de una “señal de error” asociada al extremo de salida del enlace  $w_{ji}$  y la variable  $x_{ni}$  asociada al extremo de entrada del enlace.

En una red *feed-forward* general, cada unidad computa una suma ponderada de sus entradas de la forma

$$a_j = \sum_i w_{ji} z_i, \quad (5.3.34)$$

donde  $z_i$  es la activación de una unidad, o entrada, que envía una conexión a la unidad  $j$ , y  $w_{ji}$  es el peso asociado a dicha conexión. En la Sección 2.5.1 se ha mostrado que pueden incluirse términos de sesgo introduciendo una unidad o entrada extra con una activación fijada a 1. Esto permite no tratar explícitamente con los términos de sesgo. La suma en (5.3.34) se transforma mediante una función de activación no lineal  $h(\cdot)$  para obtener la activación  $z_j$  de la unidad  $j$  de la forma

$$z_j = h(a_j), \quad (5.3.35)$$

donde una o más de las variables  $z_i$  en la suma (5.3.34) pueden ser entradas, y de manera similar, la unidad  $j$  de (5.3.35) puede ser una salida.

Para cada patrón del conjunto de entrada, se supone que se ha provisto a la red del vector de entrada correspondiente y se han calculado las activaciones de todas las unidades ocultas y de salida de la red aplicando repetidamente (5.3.34) y (5.3.35). Este proceso suele denominarse propagación hacia adelante, ya

que puede entenderse como el flujo de información hacia adelante a través de la red.

Considérese ahora la evaluación de la derivada de  $E_n$  respecto a un peso  $w_{ji}$ . Las salidas de las distintas unidades dependen del patrón de entrada ( $\mathbf{x}_n$ ) particular. Sin embargo, para mantener despejada la notación, se suele omitir el subíndice  $n$  de las variables de la red. Comiéncese notando que  $E_n$  depende de los pesos  $w_{ji}$  únicamente a partir de la suma de las entradas  $a_j$  a la unidad  $j$ . Por lo que aplicando la regla de la cadena para derivadas parciales se obtiene

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}. \quad (5.3.36)$$

Se introduce la notación

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j}, \quad (5.3.37)$$

donde los  $\delta$  se conocen habitualmente como errores. Utilizando (5.3.34), se puede escribir

$$\frac{\partial a_j}{\partial w_{ji}} = z_i. \quad (5.3.38)$$

Sustituyendo (5.3.37) y (5.3.38) en (5.3.36) se obtiene

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i. \quad (5.3.39)$$

La ecuación (5.3.39) expresa que la derivada requerida se obtiene simplemente multiplicando el valor de  $\delta$  de la unidad en el extremo de salida del peso por el valor de  $z$  para la unidad en el extremo de entrada del peso (donde  $z = 1$  en el caso de los términos de sesgo). Nótese que esta expresión toma la misma forma que el modelo lineal simple considerado al comienzo de esta sección. Así, para evaluar las derivadas basta con calcular el valor  $\delta_j$  para cada unidad oculta y de salida de la red y aplicar (5.3.39).

Como ya se ha mostrado, para las unidades de salida, se tiene

$$\delta_k = y_k - t_k, \quad (5.3.40)$$

dado que se utiliza el enlace canónico como función de activación de las unidades de salida. Para evaluar los  $\delta$  de las unidades ocultas se aplica de nuevo la regla de la cadena para derivadas parciales,

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j}, \quad (5.3.41)$$

donde la suma recorre todas las unidades  $k$  a las que  $j$  envía conexiones. La disposición de las unidades y los pesos se ilustra en la Figura 2.3. Nótese que las unidades etiquetadas con  $k$  pueden incluir otras unidades ocultas o unidades de salidas. Al escribir la ecuación 5.3.41, se utiliza el hecho de que las variaciones en  $a_j$  afectan a la función de error únicamente a partir de variaciones en las

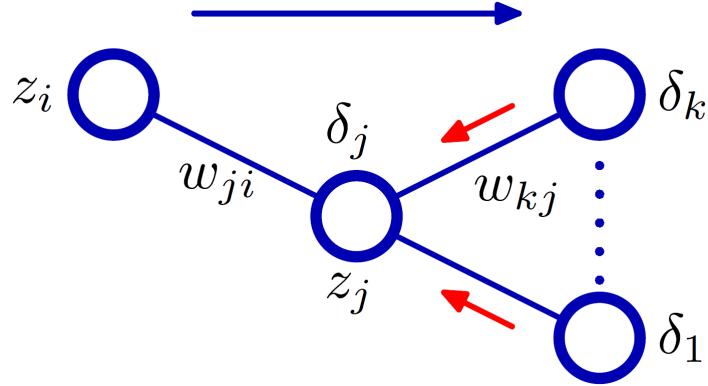


Figura 2.3: Representación del cálculo de  $\delta_j$  para las unidades ocultas  $j$  por propagación hacia atrás de los  $\delta$  de las unidades  $k$  a las que la unidad  $j$  envía conexiones. Fuente: Bishop (2006)

variables  $a_k$ . Si se sustituye la definición de *delta* dada por (5.3.37) en (5.3.41) y se aplican (5.3.34) y (5.3.35) se obtiene la siguiente **fórmula de propagación hacia atrás**

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k, \quad (5.3.42)$$

que indica que el valor de  $\delta$  de una unidad oculta en particular se puede obtener propagando los  $\delta$  hacia atrás desde unidades situadas más adelante en la red, tal y como se ilustra en la Figura 2.3.

Por consiguiente, el método de propagación hacia atrás puede resumirse de la siguiente manera

#### Propagación hacia atrás del error

1. Se aplica el vector de entrada  $\mathbf{x}_n$  a la red y se propaga hacia adelante a través de la red utilizando (5.3.34) y (5.3.35) para obtener las activaciones de todas las unidades ocultas y de salida.
2. Se evalua  $\delta_k$  para todas las unidades de salida utilizando (5.3.40).
3. Se propagan los *delta* hacia atrás utilizando (5.3.42) para obtener  $\delta_j$  para cada unidad en la red neuronal.
4. Se utiliza (5.3.39) para evaluar las derivadas.

Para los métodos por lotes, la derivada del error total  $E$  puede obtenerse repitiendo los pasos anteriores para cada patrón ( $\mathbf{x}$ ) del conjunto de entrenamiento y realizando la suma:

$$\frac{\partial E}{\partial w_{ji}} = \sum_n \frac{\partial E_n}{\partial w_{ji}}. \quad (5.3.43)$$

En la derivación anterior se supone implícitamente que cada unidad oculta o de salida de la red tiene la misma función de activación  $h(\cdot)$ .

Puede encontrarse más información sobre la eficiencia del método de propagación hacia atrás, su aplicación a otros tipos de derivadas y su aplicación para el cálculo de la matriz hessiana (segunda derivada) en Bishop (2006)(Secciones 5.3.3, 5.3.4 y 5.4, respectivamente) , aunque para comprender nuestro análisis bastará con lo introducido hasta ahora.

#### 2.5.4. Regularización en Redes Neuronales

El número de unidades de entrada y salida de una red neuronal está determinado generalmente por la dimensionalidad del conjunto de datos, mientras que el número  $M$  de unidades ocultas es un parámetro libre que puede ajustarse para obtener el mejor rendimiento predictivo. Nótese que  $M$  controla el número de parámetros (pesos y términos de sesgo) de la red, por lo que es de esperar que en un contexto de máxima verosimilitud exista un valor óptimo de  $M$  que produzca el mejor rendimiento predictivo y que se corresponde con el equilibrio óptimo entre sub-aprendizaje (*under-fitting*) y sobre-aprendizaje (*over-fitting*).

Sin embargo, el error de generalización no es una función simple de  $M$  debido a la presencia de mínimos locales en la función de error. Un enfoque para controlar la complejidad de una red neuronal para evitar el sobre-aprendizaje consiste en tomar un valor de  $M$  relativamente grande y controlar la complejidad mediante la adición de un término de regularización a la función de error. El regularizador más simple es el cuadrático, con el que se obtiene un error regularizado de la forma

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda}{2} \mathbf{w}' \mathbf{w}. \quad (5.4.44)$$

Este regularizador también se conoce como caída del peso (*weight decay*). La complejidad efectiva del modelo se determina tomando un coeficiente de regularización  $\lambda$ . Dicho regularizador puede interpretarse como el logaritmo negativo de una distribución a-priori gaussiana de media cero sobre el vector de pesos  $\mathbf{w}$ .

### 2.6. Entrenamiento y predicción con modelos de Redes Neuronales en R

Vamos a realizar el entrenamiento y predicción de los distintos modelos ayudándonos de la librería **caret** (Kuhn, 2021) para realizar el entrenamiento de modelos de redes neuronales de una capa de la librería **nnet** (Venables and Ripley, 2002) y las redes multicapa de la librería **RSNNS** (Bergmeir and Benítez, 2012).

El paquete **caret** (Classification And REgresion Training) proporciona las funcionalidades básicas para entrenar y predecir con distintos modelos de aprendizaje automático. En nuestro caso lo utilizaremos para entrenar modelos de redes neuronales para problemas de regresión.

- La función más importante de la librería **caret** para nuestros objetivos es la función **train**, que nos permitirá realizar la selección y entrenamientos de modelos con una gran capacidad de personalización. Podemos seleccionar el método que se utilizará para realizar la selección de los hiperparámetros del modelo (definiendo incluso las particiones a utilizar) mediante un objeto de clase **trainControl**. También podemos elegir el espacio de búsqueda de los hiperparámetros a través del argumento **tuneGrid**.
- Además de la función **train**, la librería **caret** nos proporciona la función **predict**, que realiza las predicciones correspondientes al mejor modelo obtenido en el entrenamiento pasándole como argumento el objeto obtenido como resultado de la función **train**.

La función **nnet** de la librería **nnet** proporciona las funcionalidades para entrenar redes neuronales de una única capa oculta, tomando como argumentos el número de neuronas de la capa y el parámetro de regularización por caída del peso.

La función **mlpML** de la librería **RSNNS** define funcionalidades para entrenar redes neuronales con hasta tres capas ocultas, tomando como argumentos el número de neuronas en cada una de las capas.

Como el método **train** de **caret** “envuelve” a los dos métodos definidos anteriormente, podremos utilizarlos directamente desde la función **train**, definiendo en **tuneGrid** las distintas combinaciones de hiperparámetros a utilizar y pudiendo pasarle como argumento parámetros propios del entrenamiento de cada uno de los modelos (como por ejemplo el número de iteraciones en el proceso de selección de los pesos). A lo largo del proceso de entrenamiento hemos utilizado el método **'cv'** (validación cruzada) con 10 *folds*, cuyo funcionamiento puede resumirse como:

1. Se partitiona el conjunto en 10 subconjuntos (*folds*).
2. Para cada combinación de parámetros de **tuneGrid** se obtiene la media de las raíces del error cuadrático medio (RMSE) obtenidas en el conjunto de validación para cada uno de los *folds* entrenando el modelo con los 9 *folds* restantes.

El RMSE para cada uno de los *folds* se calcula como

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_{(i)})^2}{n}}, \quad (6.0.45)$$

donde el índice *i* recorre los elementos del *fold* que se utiliza como validación.

3. Se selecciona la combinación de parámetros con menor RMSE medio y se entrena un modelo con dichos parámetros y el conjunto completo de entrenamiento.

## 2.7. Intervalos de predicción en Redes Neuronales

Para poder comparar los intervalos de predicción de las redes neuronales y los modelos ARMA utilizados, debemos construir intervalos de predicción para las redes neuronales. Vamos a construirlos utilizando el método DPC (*Dual Perturb and Combine*) según la definición de Almeida and Gama (2015), que consiste en perturbar cada muestra del conjunto test  $k$  veces, añadiendo ruido blanco a la observación de cada uno de los atributos. Para posteriormente predecir cada una de las muestras perturbadas y obtener la predicción final como una agregación de las predicciones obtenidas para las muestras perturbadas.

En nuestro caso lo hemos implementado de la siguiente manera. Para cada variable de entrada  $x_i$ , se realizan  $k = 100$  perturbaciones de la forma

$$x_j = x_i + \delta_j,$$

donde  $\delta_j$  es ruido blanco  $N(0, \sigma^2)$ .

Se obtienen  $k = 100$  predicciones  $\hat{y}_j$ , y se toma como predicción final el valor

$$\hat{y}_i = \frac{\sum_{j=1}^k \hat{y}_j}{k}$$

y se definen los límites inferior y superior del intervalo como

$$[\min(\hat{y}_j), \max(\hat{y}_j)] = [L(i), U(i)]$$

Para obtener los intervalos de predicción al 95 % seleccionamos el parámetro  $\sigma \in \{0.1n \mid n \in \mathbb{N}\}$  de modo que sea el mínimo valor para el que se obtenga un PICP (Probabilidad de Cobertura del Intervalo de Predicción) en el conjunto de entrenamiento mayor o igual que 0.95. El cual se define como

$$PICP = \frac{1}{N} \sum_{i=1}^N c(i), \quad (7.0.46)$$

donde  $N$  es el número de muestras en el conjunto de entrenamiento y

$$c(i) = \begin{cases} 1, & \text{si } \hat{y}_i \in [L(i), U(i)], \\ 0, & \text{en otro caso.} \end{cases}$$

## 2.8. Cálculo de los RMSE de las predicciones a una, dos y tres horas

Utilizaremos las raíces de los errores cuadráticos medios (RMSE) para evaluar la capacidad predictiva de los modelos. Se obtienen del siguiente modo:

- Para las predicciones a una hora:

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_{(i)})^2}{n}},$$

donde la suma recorre las observaciones de los residuos de la velocidad del viento en el conjunto test y  $\hat{y}_{(i)}$  es la predicción a un paso obtenida utilizando las observaciones anteriores de la serie temporal.

- Para las predicciones a dos horas:

$$RMSE = \sqrt{\frac{\sum_{i=2}^n (y_i - \hat{y}_{(i)}^{(2)})^2}{n}},$$

donde la suma recorre desde la segunda a la última observación de los residuos de la velocidad del viento en el conjunto test y  $\hat{y}_{(i)}^{(2)}$  es la predicción a dos pasos (tomando  $\hat{y}_{(i-1)}$  en lugar de  $y_{i-1}$  como último valor observado).

- Para las predicciones a tres horas:

$$RMSE = \sqrt{\frac{\sum_{i=2}^n (y_i - \hat{y}_{(i)}^{(3)})^2}{n}},$$

donde la suma recorre desde la segunda a la última observación de los residuos de la velocidad del viento en el conjunto test y  $\hat{y}_{(i)}^{(3)}$  es la predicción a tres pasos (tomando  $\hat{y}_{(i-2)}$  y  $\hat{y}_{(i-1)}^{(2)}$  en lugar de los valores observados  $y_{i-1}$  e  $y_{i-2}$ ).

# Capítulo 3

## Datos eólicos

Mientras en los países desarrollados pueden realizarse predicciones a partir de simulaciones de alta resolución actualizadas continuamente y disponibles públicamente, los países en vías de desarrollo no cuentan con dichas facilidades.

Para investigar la velocidad del viento y la consecuente potencia eólica en Arabia Saudí, Giani et al. (2020) realizaron simulaciones horarias de alta resolución del modelo de Investigación y Predicción del Clima (WRF, Skamarock et al. (2008)) motivado por el modelo *European Centre for Medium-Range Weather Forecast*. Dichas simulaciones del modelo WRF fueron realizadas con cuatro configuraciones diferentes, obteniendo así diferentes conjuntos de datos correspondientes a los años comprendidos desde 2013 hasta 2016. En nuestro caso utilizaremos los mismos datos que en Huang et al. (2021), que se corresponden con la simulación que más se aproxima a las observaciones. Consideraremos la velocidad superficial horaria residual del viento en Arabia Saudí que describiremos en la siguiente sección.

### 3.1. Velocidad residual del viento

Como se ha mencionado en la sección anterior, realizaremos la predicción sobre los residuos obtenidos a partir de los datos de la velocidad del viento  $Y_t(\mathbf{s})$  (Huang et al., 2021). A continuación, se describe como se han obtenido estos residuos.

Se denota por  $Z_t(\mathbf{s})$  la velocidad superficial del viento en la localización  $\mathbf{s} \in \mathbf{S}$  en el instante  $t \in \{0, \dots, T\}$ , donde  $t$  indica el número de horas a partir de las 00:00 del 1 de enero de 2013 (eliminando el 29 de febrero de 2016 por simplicidad) y  $\mathbf{s} \in \mathbf{S}$  hace referencia a una localización geográfica de Arabia Saudí. Del total de 3173 localizaciones disponibles en los datos originales, en este trabajo vamos a modelizar de forma independiente las series temporales correspondientes a dos localizaciones, una en el interior y otra cerca de la costa de Arabia Saudí (ver Figura 3.1). Dichas localizaciones corresponden a los identificadores 73 y 2173 del fichero de datos `wind_residual.nc` disponible en <https://repository>.

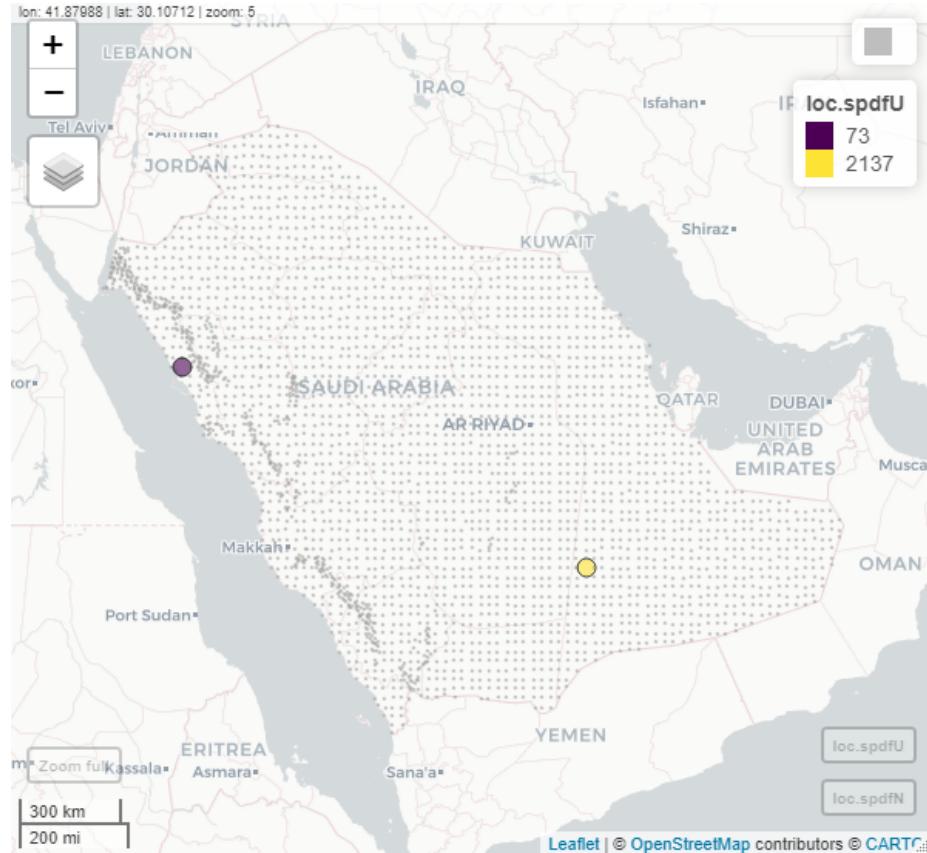


Figura 3.1: Posición de las localizaciones con cuyos datos vamos a trabajar (en gris el resto de localizaciones del conjunto de datos)

[kaust.edu.sa/handle/10754/667127](http://kaust.edu.sa/handle/10754/667127).

### 3.1.1. Obtención de las velocidades residuales

Tal y como se describe en Huang et al. (2021), en los datos originales se observan patrones inter-diarios y multi-diarios en los datos de velocidad del viento, que pueden explicarse mediante los flujos de calor tierra/océano y los patrones de mesoscala de vientos recurrentes en la península arábiga. Para cada localización  $\mathbf{s} \in S$ , se asume que  $\sqrt{Z_t(\mathbf{s})}$  depende de armónicos de la siguiente forma

$$\sqrt{Z_t(\mathbf{s})} = \beta_0(\mathbf{s}) + \sum_{i=1}^F \left\{ \beta_{i,1}(\mathbf{s}) \cos \left( \frac{2\pi t}{T_i} \right) + \beta_{i,2}(\mathbf{s}) \sin \left( \frac{2\pi t}{T_i} \right) \right\} + \gamma(\mathbf{s}) Y_t(\mathbf{s}), \quad (1.1.1)$$

donde  $F$  es el número de armónicos,  $\beta_0(\mathbf{s})$  es la intersección (*intercept*),  $\beta_{i,1}(\mathbf{s})$ ,  $\beta_{i,2}(\mathbf{s})$ ,  $i = 1, \dots, F$ , son los coeficientes para cada armónico de periodo  $T_i$ , y  $\gamma(\mathbf{s})$  es el parámetro de escalado de cada localización tal que el residual  $Y_t(\mathbf{s})$  tenga varianza unitaria en cada localización. En el diagnóstico de la transformada de Fourier realizado por Huang et al. (2021), se observaron  $F = 5$  armónicos significativos con periodos de un año, medio año, un día, doce horas y ocho horas. La transformación de los datos utilizando la raíz cuadrada se realiza para una mejor aproximación a la normalidad en datos que generalmente presentan sesgos hacia la derecha debido a ráfagas de viento ocasionales. La elección de la transformación de la raíz cuadrada está justificada por la literatura en modelado del viento (Gneiting, 2002).

Para obtener la serie temporal de residuales con la que vamos a trabajar, Huang et al. (2021) estimaron los distintos parámetros siguiendo los siguientes pasos. En primer lugar, los parámetros  $\beta_0(\mathbf{s})$  y  $\beta_{i,1}(\mathbf{s})$ ,  $\beta_{i,2}(\mathbf{s})$ ,  $i = 1, \dots, F$  fueron estimados mediante regresión lineal en cada localización utilizando las observaciones del periodo comprendido entre 2013 y 2015 (datos de entrenamiento). Dichos parámetros se utilizaron para estimar el proceso de media cero  $Y_t(\mathbf{s})$  (tomando  $\gamma(\mathbf{s})$  para que  $Y_t(\mathbf{s})$  sea estandarizada) en la ecuación (1.1.1).

### 3.1.2. Representación de las velocidades residuales medias

En las Figuras 3.2 y 3.3 se han representado las medias y desviaciones estándar diarias por año para las velocidades del viento residual correspondientes a las localizaciones 73 y 2137, cuyas coordenadas geográficas (longitud,latitud) son (36.8419, 26.0991) y (48.3264, 20.8656) respectivamente.

En ambas localizaciones, podemos observar que el comportamiento medio diario es bastante similar para todos los años, con desviaciones relativamente pequeñas, por lo que es de esperar que si encontramos un modelo que ajuste correctamente los datos de entrada este no tenga problemas en predecir nuevas observaciones de la velocidad residual.

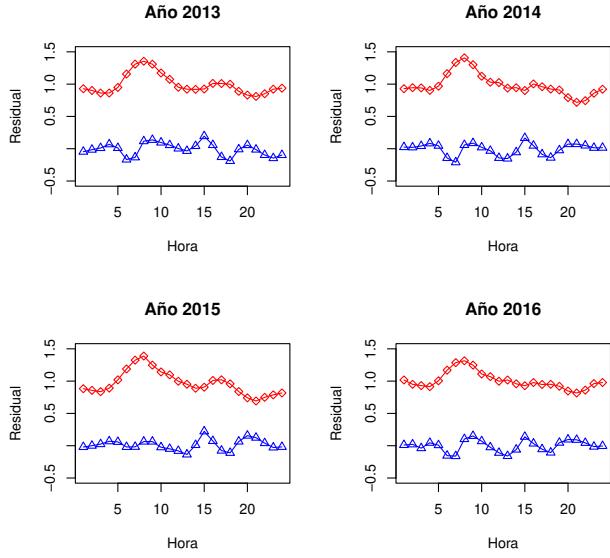


Figura 3.2: Velocidades residuales medias del viento (en azul) junto con sus desviaciones estándar (en rojo) en la localización 73

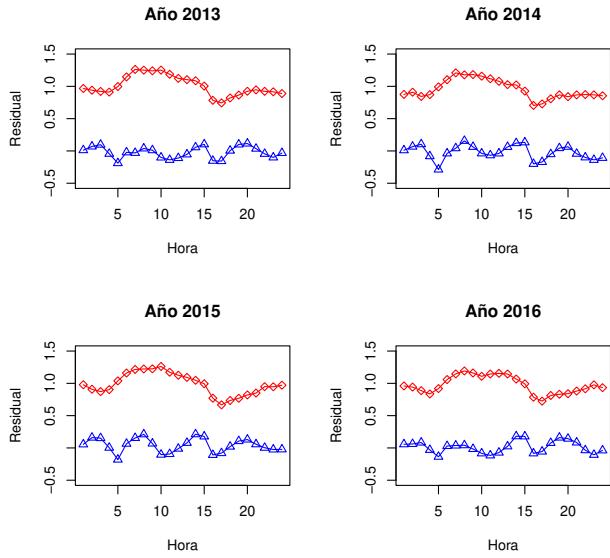


Figura 3.3: Velocidades residuales medias del viento (en azul) junto con sus desviaciones estándar (en rojo) en la localización 2137

# Capítulo 4

## Resolución y Resultados

### 4.1. Predicción con modelos ARMA

A continuación se describen los resultados obtenidos al utilizar los métodos de series temporales (modelos ARMA) descritos en el Capítulo 2 para realizar predicciones a corto plazo en las series temporales de velocidad residual del viento en ambas localizaciones.

#### 4.1.1. Selección del modelo para la localización 73

En primer lugar realizaremos un análisis preliminar de la serie de residuales, comenzando por analizar si la serie es estacionaria y analizando las ACF y PACF muestrales para obtener información sobre los modelos que puedan resultar adecuados para modelizar el problema.

Procedemos a realizar distintos tests estadísticos para comprobar si la serie temporal de las velocidades residuales  $\{Y_t\}$ ,  $t = 1, \dots, T$ , con  $T = 24 \frac{h}{día} \cdot 365 \frac{días}{año} \cdot 4 años = 35040$  es estacionaria. En primer lugar aplicamos el test de raíz unitaria de Dickey-Fuller, obteniendo un  $p$ -valor menor que 0.01. Posteriormente aplicamos el test KPSS, obteniendo un  $p$ -valor mayor que 0.1. Ambos tests sugieren que la serie puede ser estacionaria (Sección 2.4), utilizaremos modelos ARMA para modelizarla.

Utilizaremos las observaciones de los dos primeros años (2013 y 2014) para entrenar los distintos modelos de series temporales, y los datos correspondientes al tercer año (2015) para seleccionar el modelo que mejor se ajusta en términos de error cuadrático medio (RMSE).

Procedemos a visualizar las ACF y PACF muestrales de las observaciones de la serie temporal que utilizaremos para entrenar el modelo, es decir, las correspondientes a los años 2013 y 2014 [Figura 4.1]. El comportamiento “sinusoidal” de la ACF junto con una PACF que prácticamente “desaparece” a partir del retraso 24 indica que quizás podamos modelar la serie mediante un modelo AR(24). Vamos a entrenar dicho modelo y a calcular el RMSE (error cuadrático medio)

de las predicciones a uno, dos y tres días respecto a los valores observados para el año 2015.

Al ajustar un modelo AR(24) a las observaciones de los dos primeros años y analizar los residuales se rechaza la hipótesis de independencia de los residuos, ya que obtenemos un  $p$ -valor menor que  $2.2 \cdot 10^{-16}$  para el test de Ljung-Box. Sin embargo, si observamos la gráfica de distribución de los residuales observamos que tiene un comportamiento similar a una distribución normal con menor desviación a la estimada. En la Tabla 4.1 se ha calculado el RMSE de las predicciones, mientras que en la Figura 4.3 se han representado las estimaciones junto a sus intervalos de predicción.

Tabla 4.1: RMSE de las predicciones del modelo AR(24)

nº horas	1	2	3
RMSE	0.5044542	0.7181819	0.8186915

En la Figura 4.3 podemos observar que las predicciones se ajustan bastante bien a los datos reales para el periodo seleccionado. Por lo tanto, se concluye que el test de Ljung-Box podría estar afectado por la alta dimensionalidad de los datos.

Para intentar encontrar un modelo con un buen balance entre bondad de ajuste (en el sentido de los residuales) y capacidad predictiva (en el sentido del RMSE respecto a los datos de validación) vamos a entrenar diferentes modelos ARMA(p,q) seleccionando series temporales más cortas y utilizando la selección automática de modelos en función del AIC con valores máximos de 24 para los parámetros p y q. Una vez seleccionados los modelos, analizaremos los residuales y evaluaremos el RMSE en la serie de validación.

Comenzamos aplicando la selección automática con la serie de entrenamiento al completo, obteniendo un modelo ARMA(24,13). Para este modelo, volvemos a obtener un comportamiento de los residuales similar al de los del modelo AR(24), es decir, un  $p$ -valor para el test de Ljung-Box menor que  $2.2 \cdot 10^{-16}$  y una distribución de los residuales similar a una normal con menor varianza a la estimada. Al analizar los errores cuadráticos medios para las predicciones a uno, dos y tres días [Tabla 4.2] comprobamos que estos muestran resultados ligeramente mejores con respecto al modelo AR(24), aunque el modelo ARMA(24,13) seleccionado sigue sin tener un buen ajuste en el sentido de la normalidad de los residuales.

Tabla 4.2: RMSE de las predicciones del modelo ARMA(24,13) obtenido utilizando la serie de entrenamiento completa

nº horas	1	2	3
RMSE	0.5025062	0.7138486	0.8130757

Del mismo modo que en el caso del modelo AR(24), los intervalos de predicción para el modelo ARMA(24,13) [Figura 4.4] contienen en su interior a la

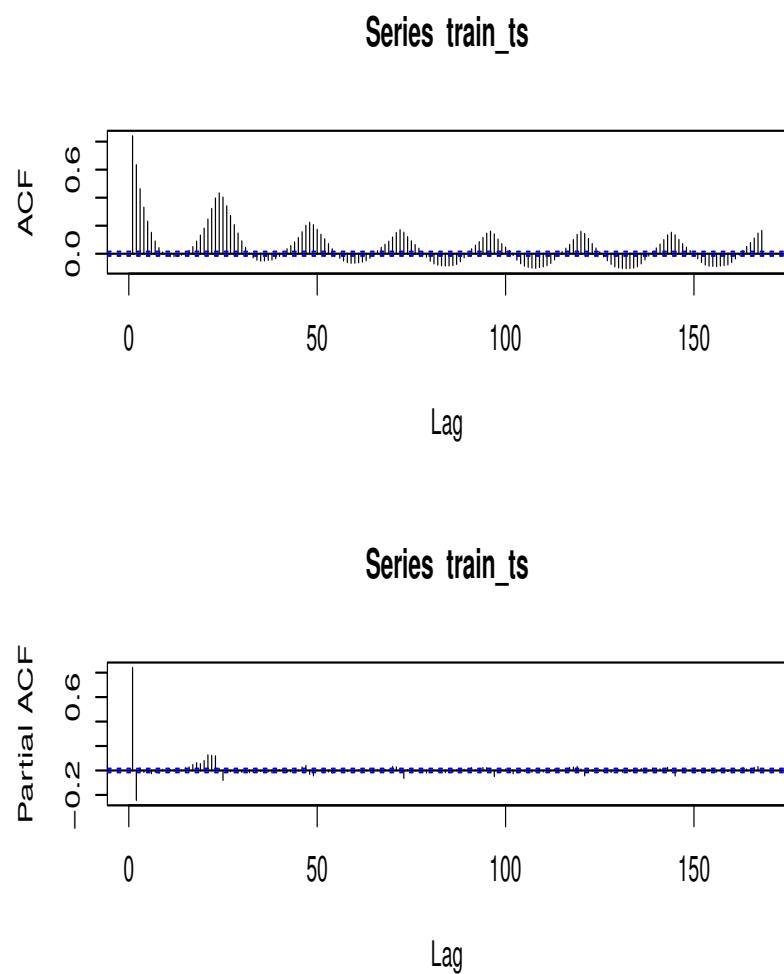


Figura 4.1: ACF y PACF muestrales de la serie de entrenamiento

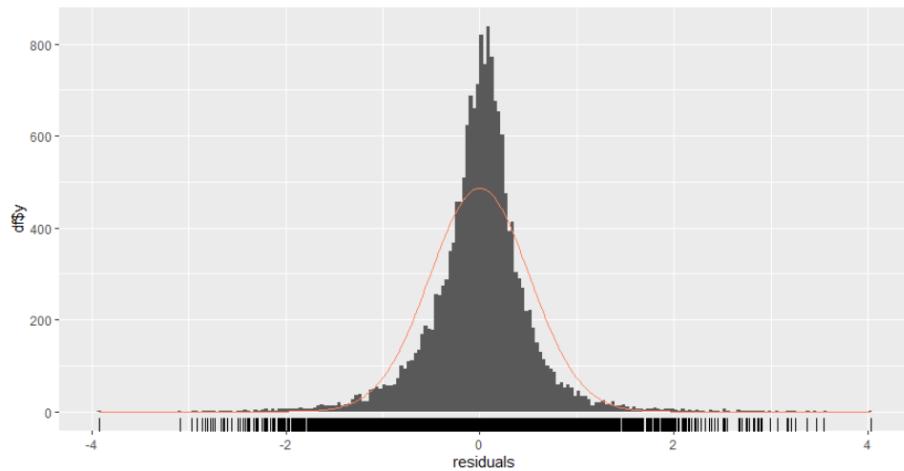


Figura 4.2: Distribución de los residuales del modelo AR(24)

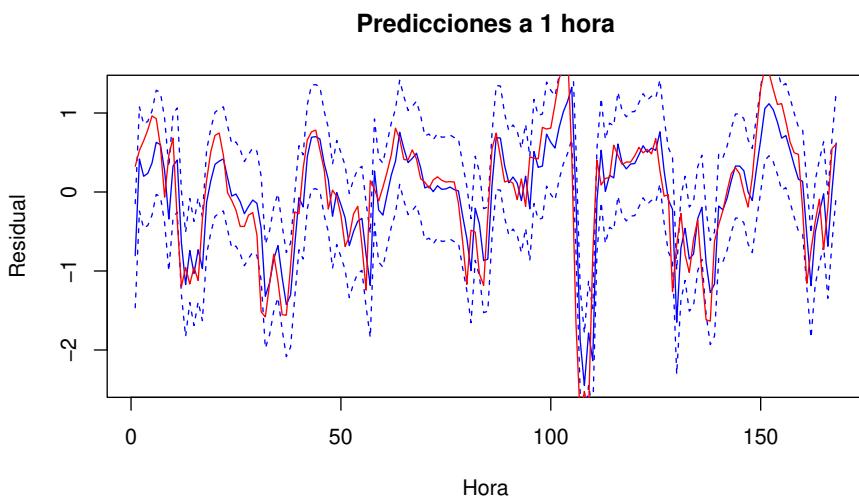


Figura 4.3: Intervalos de predicción a una hora para el modelo AR(24) (Zoom a la primera semana) Las líneas azules representan la predicción junto con sus intervalos, mientras que la roja representa las observaciones

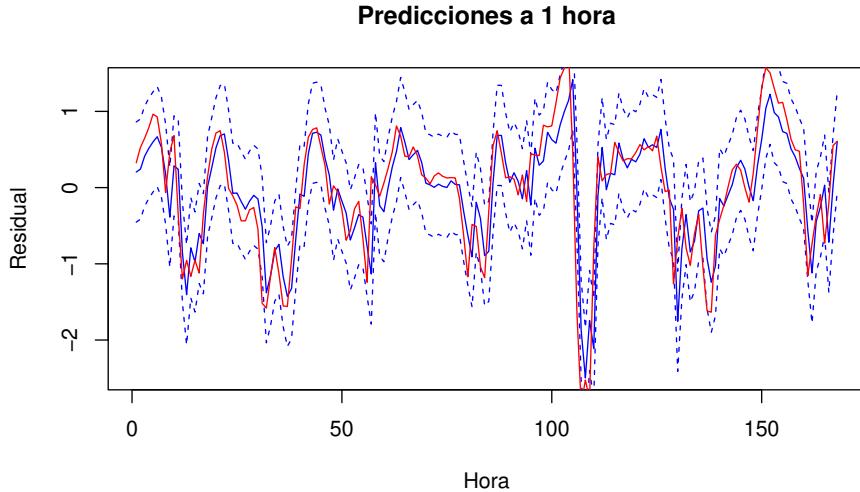


Figura 4.4: Intervalos de predicción a una hora para el modelo ARMA(24,13)

serie de validación (salvo quizás en algunos puntos extremos), y aunque no la ajustan perfectamente en el sentido del RMSE, permiten hacerse una idea del futuro comportamiento de los residuos de la velocidad del viento.

Posteriormente procedemos a repetir el proceso realizando la selección automática y predicción con la mitad de la serie de entrenamiento más cercana al último valor (datos correspondientes únicamente al año 2014). En este caso obtenemos un modelo ARMA(24,11) que sigue sin tener un buen ajuste en el sentido de la normalidad de los residuales y cuyo ajuste en el sentido del RMSE [Tabla 4.3] no mejora las predicciones obtenidas con el modelo ARMA(24,13) entrenado con la serie completa. Analizando los intervalos de predicción [Figura 4.5] también parecen peores que los del modelo entrenado y seleccionado con todos los datos.

Tabla 4.3: RMSE de las predicciones del modelo ARMA(24,11) obtenido utilizando la mitad de la serie de entrenamiento correspondiente al año 2014

nº horas	1	2	3
RMSE	0.503946	0.7153927	0.8145664

Continuando con el análisis repetimos el proceso realizando el entrenamiento con los datos correspondientes a los últimos  $12 \cdot 365 = 4380$  valores (la segunda mitad del año 2014), obteniendo un modelo ARMA(24,3). Si analizamos las gráficas de residuales [Figura 4.6] de este modelo observamos que se acercan más a las esperables en caso de independencia en los residuales. En lo que respecta al RMSE [Tabla 4.4] tenemos un peor ajuste que para el modelo ARMA(24,13)

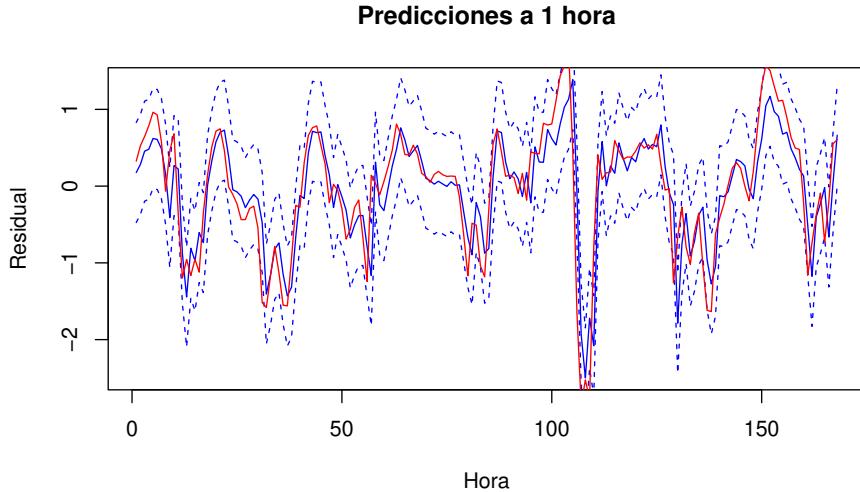


Figura 4.5: Intervalos de predicción a una hora para el modelo ARMA(24,11)

entrenado con las observaciones de los años 2013 y 2014 al completo. Sin embargo, en lo que respecta a los intervalos de predicción podemos observar un comportamiento similar a el de los modelos ARMA anteriores.

Tabla 4.4: RMSE de las predicciones del modelo ARMA(24,3) obtenido utilizando las observaciones de la segunda mitad de 2014

nº horas	1	2	3
RMSE	0.5094853	0.7257241	0.8281664

Finalmente, se realiza el proceso entrenando y seleccionando el modelo únicamente con  $6 \cdot 365 = 2190$  observaciones de la serie temporal, es decir, el último trimestre del año 2014, obteniendo un modelo ARMA(24,3) cuyos residuales se muestran en la Figura 4.8. Al igual que con los modelos anteriores, se rechaza la hipótesis nula de independencia en los residuales ( $p$ -valor de  $7.609 \times 10^{-6}$ ), aunque sus representaciones gráficas se parecen más a lo esperado. En lo que respecta a el ajuste en la serie de validación observamos que el RMSE [Tabla 4.5] mejora respecto al modelo entrenado con 6 meses a la hora de realizar predicciones a 2 y 3 horas.

Tabla 4.5: RMSE de las predicciones del modelo ARMA(24,3) obtenido utilizando las observaciones de los últimos tres meses de 2014

nº horas	1	2	3
RMSE	0.5112428	0.724563	0.8261759

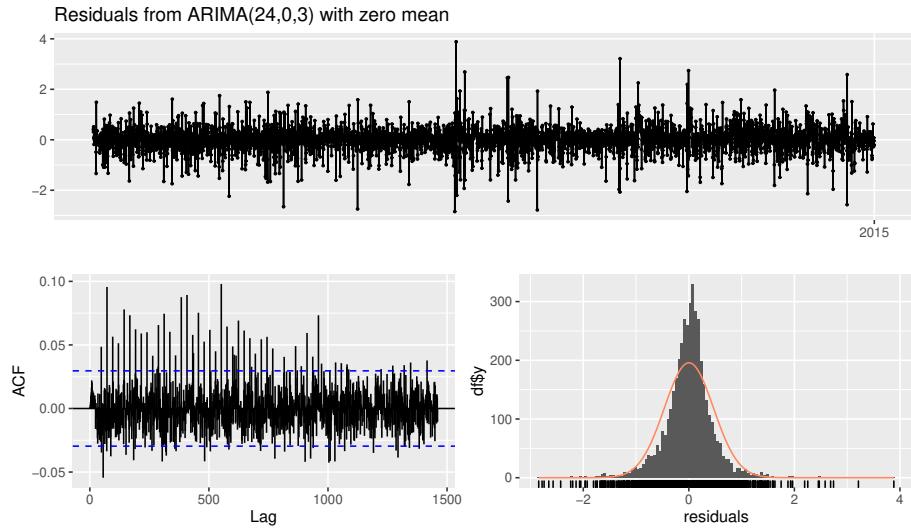


Figura 4.6: Gráficas de análisis de residuales del modelo ARMA(24,3) entrenado con 6 meses

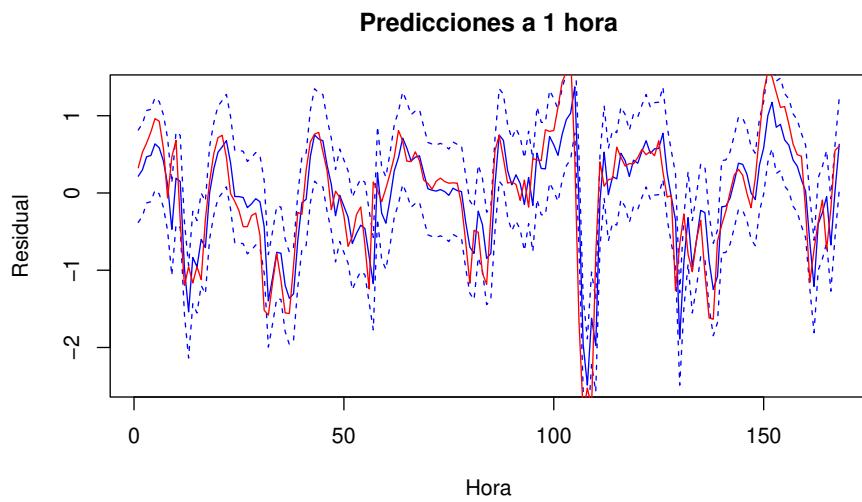


Figura 4.7: Intervalos de predicción a una hora para el modelo ARMA(24,3) entrenado con 6 meses

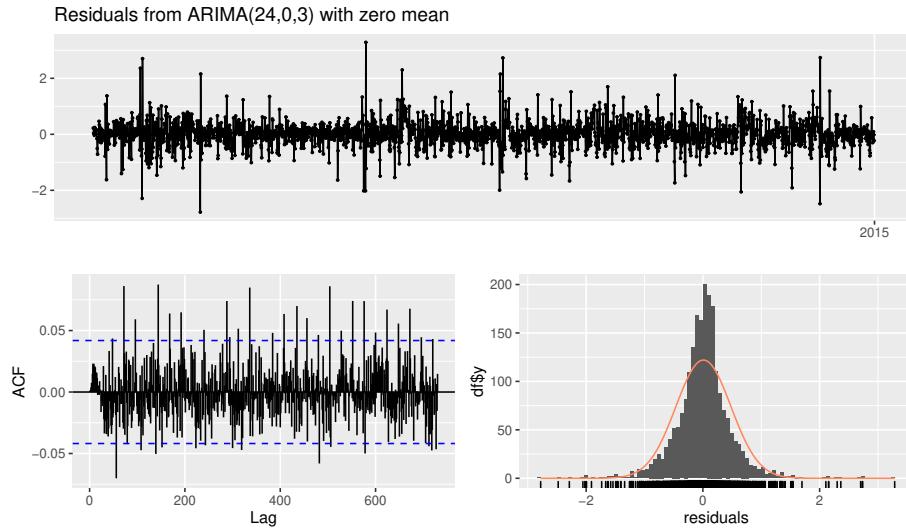


Figura 4.8: Gráficas de análisis de residuales del modelo ARMA(24,3) entrenado con 3 meses

Por lo tanto, debido a que en ningún de los casos se ha encontrado un modelo cuyos residuales puedan considerarse independientes, vamos a seleccionar el modelo que mejor se ajusta a los datos de validación, es decir, el modelo entrenado con la serie de entrenamiento al completo, ya que hemos observado que la tendencia del RMSE es empeorar conforme se reduce el tamaño de la subserie de entrenamiento.

#### 4.1.2. Modelo ARMA final para la localización 73

Una vez analizada la influencia de la dimensionalidad de la serie de entrada en los residuales y la capacidad predictiva y tras observar que se obtienen mejores resultados en términos de predicción al utilizar el conjunto de datos de entrenamiento al completo, procedemos a realizar la selección y entrenamiento de un modelo ARMA con las observaciones correspondientes a los años 2013, 2014 y 2015.

Al realizar la selección y entrenamiento con los tres primeros años de la serie de los residuales de la velocidad del viento  $\{Y_t\}$  obtenemos un modelo ARMA(24,5) que vuelve a tener un mal ajuste en el sentido de los residuales [Figura 4.9]. En lo que respecta a la capacidad predictiva, analizando los RMSE de las predicciones [Tabla 4.6] observamos un comportamiento similar al de los diferentes modelos que hemos probado, con un ajuste ligeramente peor, seguramente debido a que el año 2015 [Figura 3.2] tiene un comportamiento ligeramente diferente en lo que se refiere a las medias diarias, y en este caso lo estamos teniendo en cuenta a la hora de entrenar el modelo.

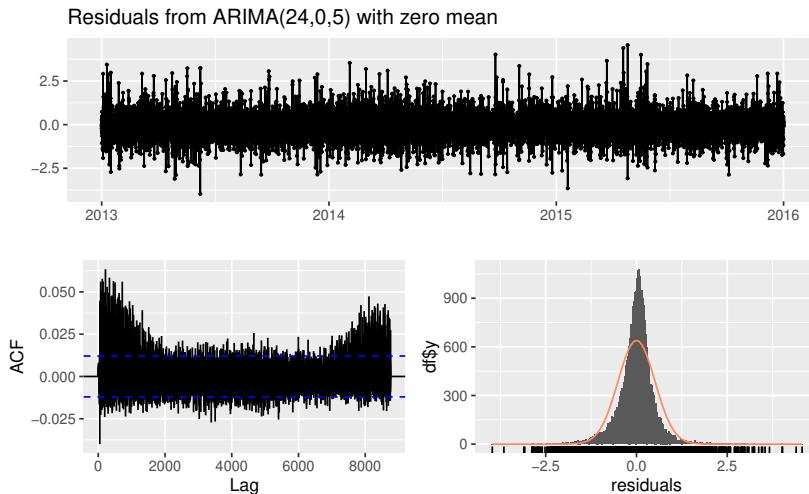


Figura 4.9: Gráficas de residuales del modelo ARMA(24,5) seleccionado para la localización 73

Tabla 4.6: RMSE de las predicciones del modelo ARMA(24,5) seleccionado para la localización 73

nº horas	1	2	3
RMSE	0.5077213	0.7181873	0.8156235

En lo que respecta a las predicciones, si analizamos los intervalos de predicción a una hora [Figura 4.10] observamos que se ajustan bastante bien a los datos reales.

#### 4.1.3. Selección del modelo para la localización 2137

De manera similar a como hemos hecho para la localización 73, en primer lugar realizaremos un análisis preliminar de la serie de residuales. Comenzaremos por analizar si la serie es estacionaria y analizar las ACF y PACF muestrales para obtener información sobre los modelos que puedan resultar adecuados para modelizar el problema.

Procedemos a realizar distintos tests estadísticos para comprobar si la serie temporal de las velocidades residuales  $\{Y_t\}$ ,  $t = 1, \dots, T$ , con  $T = 24 \frac{\text{h}}{\text{día}} \cdot 365 \frac{\text{días}}{\text{año}} \cdot 4 \text{ años} = 35040$  es estacionaria. En primer lugar aplicamos el test de de raíz unitaria de Dickey-Fuller, obteniendo un  $p$ -valor menor que 0.01. Posteriormente aplicamos el test KPSS, obteniendo un  $p$ -valor de 0.088, podríamos asumir estacionariedad. Como el resultado de ninguno de los dos tests nos da razones para rechazar la hipótesis de que la serie es estacionaria, utilizaremos modelos ARMA para modelizarla.

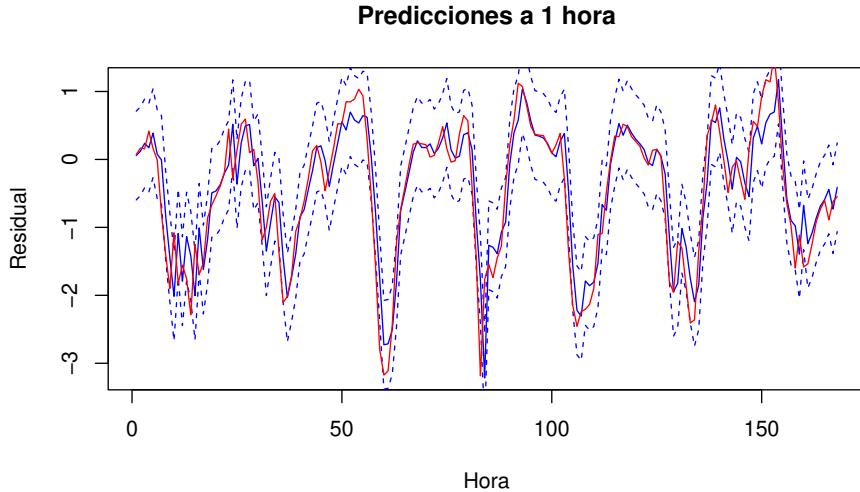


Figura 4.10: Intervalos de predicción a una hora para el modelo ARMA(24,5) seleccionado para la localización 73

Utilizaremos las observaciones de la serie temporal durante los dos primeros años (2013 y 2014) para entrenar los distintos modelos de series temporales, para finalmente seleccionar el que mejor ajuste (en el sentido del error cuadrático medio (RMSE)) las observaciones correspondientes al tercer año (2015).

Procedemos a visualizar las ACF y PACF muestrales de las observaciones de la serie temporal que utilizaremos para entrenar el modelo, es decir, las correspondientes a los años 2013 y 2014 [Figura 4.11].

El decrecimiento “sinusoidal” de la ACF junto con una PACF que prácticamente “desaparece” a partir del retardo 24 indica que quizá podamos modelar la serie mediante un modelo AR(24).

Al ajustar un modelo AR(24) a las observaciones de los dos primeros años y analizar los residuales observamos un comportamiento similar al que tenían los residuales del modelo AR(24) entrenado para la localización anterior, es decir, nos vemos obligados a rechazar la hipótesis de residuos independientes por obtener un p-valor ínfimo para el test de Ljung-Box, aunque la distribución de los residuales parece ser similar a una normal. Si calculamos el RMSE de las predicciones [Tabla 4.1] comprobamos que mejoran respecto a los del modelo AR(24) de la localización anterior, a pesar de tener un ajuste en el sentido de los residuales muy mejorable. En lo que respecta a los intervalos de predicción [Figura 4.12], observamos que estos tienen un comportamiento muy similar al que observábamos en la localización 73.

Como para el caso de esta localización tampoco obtenemos un buen ajuste con el modelo AR(24), vamos a realizar un proceso similar al realizado para la

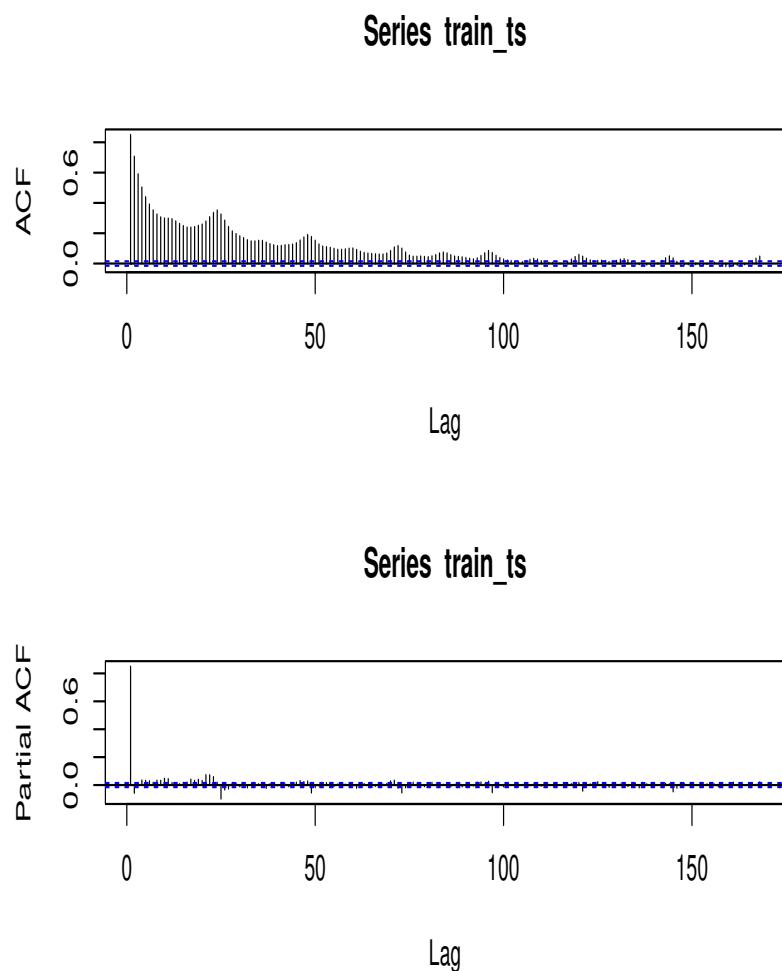


Figura 4.11: ACF y PACF muestrales de la serie de entrenamiento de la localización 2137

Tabla 4.7: RMSE de las predicciones del modelo AR(24)

nº horas	1	2	3
RMSE	0.5210577	0.7055411	0.7996031

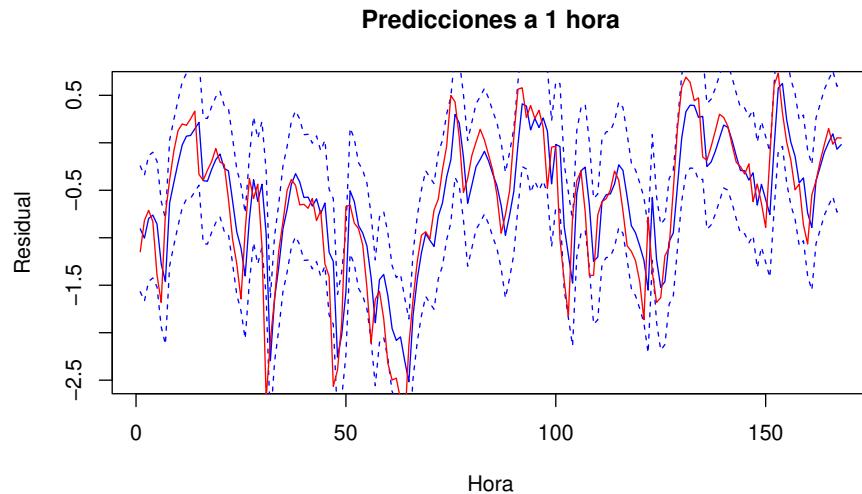


Figura 4.12: Intervalos de predicción a una hora para el modelo AR(24)

localización 73 para intentar encontrar un modelo con un buen equilibrio entre bondad en el ajuste de los datos de entrenamiento (en el sentido de la independencia y normalidad de los residuales) y una buena capacidad de generalización (buen ajuste en el sentido del RMSE en los datos de validación). En la Tabla 4.8 se muestran los RMSE de los distintos modelos entrenados.

Tabla 4.8: RMSE de los distintos modelos ARMA

Periodo de entrenamiento	RMSE 1h	RMSE 2h	RMSE 3h
2 años	0.51850	0.69985	0.79244
1 año	0.51993	0.70086	0.79373
6 meses	0.52239	0.70567	0.79907
3 meses	0.52185	0.70682	0.80038

De nuevo, debido a que en ning n de los casos se ha encontrado un modelo cuyos residuales puedan considerarse independientes (esto es m s notable en esta localizaci n que la 73, ya que para esta \'ltima pod amos encontrar un modelo que mejoraba el p-valor del test de Ljung-Box), vamos a seleccionar el modelo el modelo que mejor se ajusta a los datos de validaci n.

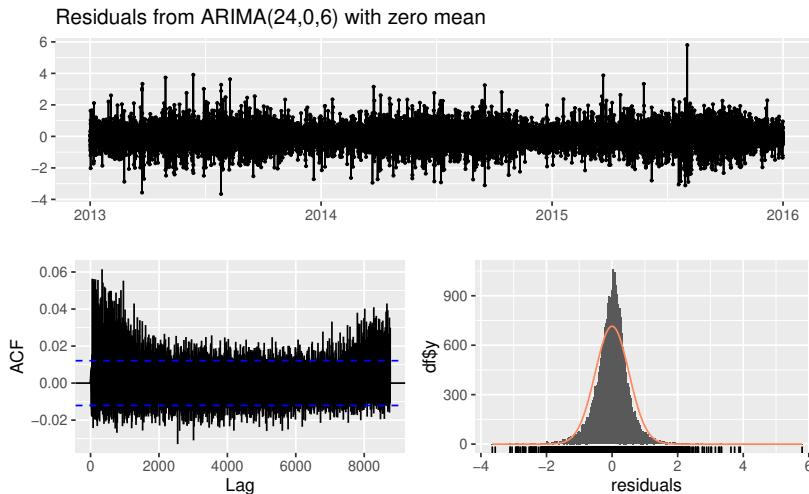


Figura 4.13: Gráficas de residuales del modelo ARMA(24,6) seleccionado para la localización 2137

#### 4.1.4. Modelo ARMA final para la localización 2173

Una vez analizada la influencia de la longitud de la serie de entrada en los residuales y la capacidad predictiva y observar que se obtienen modelos con mayor capacidad de generalización utilizando la mayor cantidad de observaciones posibles de la serie de entrada, procedemos a realizar la selección y entrenamiento de un modelo ARMA con las observaciones correspondientes a los años 2013, 2014 y 2015.

Al realizar la selección y entrenamiento con los tres primeros años de la serie de los residuales de la velocidad del tiempo  $\{Z_t\}$  obtenemos un modelo ARMA(24,6) que vuelve a tener un mal ajuste en el sentido de los residuales [Figura 4.9]. En lo que respecta a la capacidad predictiva, analizando los RMSE de las predicciones para el año 2016 [Tabla 4.9] observamos un comportamiento similar al de los diferentes modelos que hemos probado, con un ajuste ligeramente peor en lo que respecta a las predicciones a una hora, aunque con una mejor capacidad predictiva a dos y tres horas.

Tabla 4.9: RMSE de las predicciones del modelo ARMA(24,6) seleccionado para la localización 2137

nº horas	1	2	3
RMSE	0.5210933	0.6915848	0.7886777

En lo que respecta a las predicciones, si analizamos los intervalos de predicción a una hora [Figura 4.14] observamos que, si bien las predicciones exactas son mejorables, los valores observados si que están contenidos dentro de los

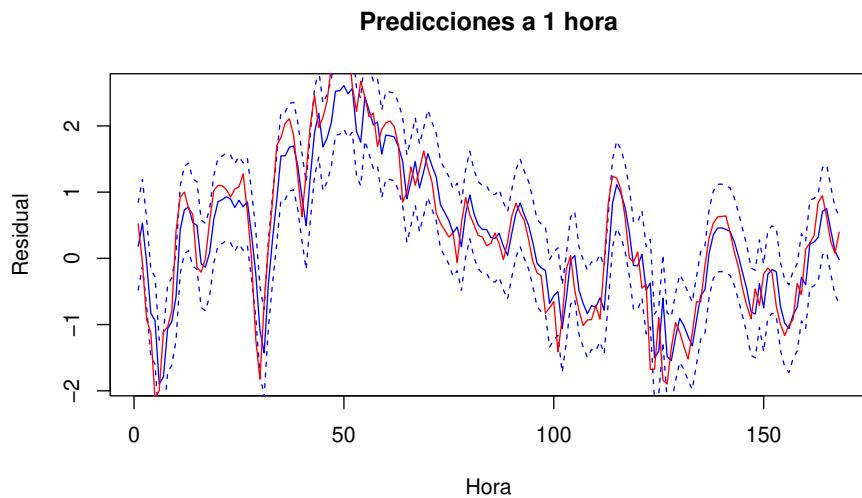


Figura 4.14: Intervalos de predicción a una hora para el modelo ARMA(24,6) seleccionado para la localización 2137

intervalos de predicción.

## 4.2. Predicción con Redes Neuronales

Para poder aplicar las metodologías de redes neuronales de la Sección 2.5 debemos adaptar las observaciones de la serie temporal a un formato que nos permita utilizarla como entrada para las distintas redes neuronales.

Nos centraremos en el RMSE en el conjunto test (utilizando, como en el caso de los modelos ARMA, los años 2013-2015 como conjunto de entrenamiento y el año 2016 como conjunto test) para seleccionar el mejor modelo, y una vez seleccionado construiremos y visualizaremos sus intervalos de predicción.

### 4.2.1. Transformación utilizando las frecuencias significativas

La aproximación que vamos a tomar para la primera transformación consiste en considerar las observaciones de las frecuencias significativas de un año, medio año, un día, doce horas y ocho horas obtenidas en el análisis de Fourier realizado por Huang et al. (2021), construyendo a partir de la serie de los residuales de la velocidad del viento  $\{Y_t\}$  una matriz que contenga las frecuencias significativas para cada observación a partir del instante  $py = 24 \cdot 365$  ( $py$  es el número de periodos de la serie original en un año), ya que para los instantes  $\{1, \dots, py\}$  no existe la observación de la serie correspondiente a un año antes. Siguiendo la notación descrita para las redes neuronales, tenemos un patrón de entrada

$$\mathbf{x}_n = (x_1, x_2, x_3, x_4, x_5) = (y_{n-py}, y_{n-phy}, y_{n-pd}, y_{n-12}, y_{n-8})$$

para cada valor objetivo

$$t_n = y_n, \quad n \in \{py + 1, py + 2, \dots, T\},$$

donde los valores  $phy$  y  $pd$  se corresponden al número de periodos de la serie original en medio año y un día, respectivamente, es decir,  $phy = py/2$  y  $pd = 24$ . Podemos representar las observaciones transformadas como matrices de atributos  $X$  y valores objetivo  $\mathbf{t}$  de la forma

$$X = \begin{bmatrix} y_1 & y_{1+phy} & y_{py+1-pd} & y_{py-11} & y_{py-7} \\ y_2 & y_{2+phy} & y_{py+2-pd} & y_{py-10} & y_{py-6} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_i & y_{i+phy} & y_{py+i-pd} & y_{py+i-12} & y_{py+i-8} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{3py} & y_{3py+phy} & y_{4py-pd} & y_{4py-12} & y_{4py+i-8} \end{bmatrix} \quad \mathbf{t} = \begin{bmatrix} y_{py+1} \\ y_{py+2} \\ \vdots \\ y_{py+i} \\ \vdots \\ y_{4py} \end{bmatrix}.$$

Vamos a utilizar validación cruzada con las primeras  $2py$  filas de la matriz de atributos y sus valores observados correspondientes para seleccionar los mejores modelos de una capa para ambas localizaciones utilizando las librerías **caret** y **nnet** de R.

### **Red de una capa oculta para la localización 73**

Realizamos la validación cruzada con la rejilla de parámetros  $\text{decay} \in \{0.5, 0.1, 0.01\}$  y  $\text{size} \in \{1, 2, \dots, 25\}$  cuyos parámetros corresponden al parámetro de regularización por caída del peso y al número de neuronas en la capa oculta, respectivamente.

El modelo seleccionado de entre todas las opciones posibles consiste en una red con 24 neuronas ocultas y un parámetro de regularización de 0.5 con un RMSE medio entre folds de 0.8795043. Dicho modelo es esperable que no tenga capacidad predictiva excesivamente buena. De todos modos, vamos a analizar su RMSE para las predicciones en test. Al realizar las predicciones obtenemos un RMSE para las predicciones de las observaciones del 2016 de 0.9352844, que es prácticamente igual de malo que el que obtendríamos prediciendo la media de la serie para todas las observaciones. Si analizamos los intervalos de predicción observamos que, además de que las predicciones son mucho más “suaves” que las observaciones, los intervalos calculados con una desviación igual a la de la serie no contienen las observaciones en su interior.

*Observación 1.* Al utilizar únicamente las frecuencias significativas, siendo la última la de 8 horas antes y disponiendo de las observaciones de la serie temporal hasta el instante  $t$ , podemos realizar la predicción del instante  $t+8$ , por lo que no necesitaremos utilizar valores predichos anteriores para realizar las predicciones a hasta 8 pasos. Es decir, los modelos construidos a partir de la transformación de frecuencias significativas tienen la ventaja de tener la misma precisión en las predicciones a uno, dos y tres periodos (horas).

*Observación 2.* Debido a los malos RMSE observados parece más razonable centrarse en entrenar otro tipo de modelos que en expandir el espacio de búsqueda del parámetro de regularización.

### **Red de una capa oculta para la localización 2137**

Del mismo modo que para la localización 73, hemos comenzado con la rejilla de parámetros  $\text{decay} \in \{0.5, 0.1, 0.01\}$  y  $\text{size} \in \{1, 2, \dots, 25\}$ , cuyo significado se ha explicado en la sección anterior.

El modelo seleccionado de entre todas las opciones posibles consiste en una red con 19 neuronas ocultas y un parámetro de regularización de 0.5 con un RMSE medio entre folds de 0.8811621. Al realizar las predicciones obtenemos un RMSE para las predicciones de las observaciones del 2016 de 0.9256865, que no es peor que el de predecir directamente con la media de la serie, pero está cerca (Recordemos que la serie de residuales de la velocidad del viento,  $\{Y_t\}$  tiene desviación 1). Como los resultados obtenidos con esta transformación de los datos no son buenos, vamos a entrenar modelos utilizando una transformación que contenga más información. Podemos observar un comportamiento de los intervalos de predicción muy similares a los de la localización anterior.

#### 4.2.2. Transformación añadiendo información a las frecuencias significativas

Otra aproximación a tener en cuenta consiste en tomar para cada observación de la serie temporal, además de los valores correspondientes a las frecuencias significativas, las observaciones desde la observación actual hasta la frecuencia significativa más cercana. Siguiendo la notación descrita para las redes neuronales, tenemos un patrón de entrada

$$\mathbf{x}_n = (y_{n-py}, y_{n-phy}, y_{n-pd}, y_{n-12}, y_{n-8}, y_{n-7}, y_{n-6}, y_{n-5}, y_{n-4}, y_{n-3}, y_{n-2}, y_{n-1})$$

para cada valor objetivo

$$t_n = y_n, \quad n \in \{py + 1, py + 2, \dots, T\}.$$

#### Red de una capa oculta para la localización 73

Comenzamos realizando la validación cruzada con el mismo espacio de búsqueda que para los modelos con la transformación anterior, es decir, con parámetros de regularización 0.5, 0.1, 0.01 y un número de neuronas entre 1 y 25.

El modelo seleccionado de entre todas las opciones posibles consiste en una red con 12 neuronas en la capa oculta y un parámetro de regularización de 0.5, obteniendo una precisión en el conjunto de entrenamiento de 0.4843027, sin embargo, si analizamos la capacidad predictiva del modelo (en el sentido de los RMSE de las predicciones a una, dos y tres horas para el año 2016 [Tabla 4.10]) observamos que la capacidad de generalización del modelo es bastante mejorable. Si analizamos los intervalos de predicción de las predicciones [Figura 4.15], observamos que, si bien se observa una mejora notable respecto al modelo entrenado únicamente con las frecuencias significativas, todavía existe un amplio margen de mejora.

Tabla 4.10: RMSE de las predicciones del modelo con una única capa oculta de 12 neuronas y parámetro de regularización 0.5 para la localización 73 usando datos de las 7h anteriores más frecuencias significativas.

nº horas	1	2	3
RMSE	0.7368629	0.853078	0.9221011

Como hemos obtenido el valor máximo del espacio de búsqueda, vamos a cambiar el espacio de búsqueda del parámetro decay a los valores 0.5, 0.6, 0.7, 0.8, 0.9 y 1.

Realizando la validación cruzada en el nuevo espacio de búsqueda se selecciona un modelo con 22 neuronas en la capa oculta y un parámetro de regularización de 0.8. El modelo con 22 neuronas y parámetro de regularización 0.8 tiene un RMSE medio entre folds de 0.4834414. Sin embargo, analizando los RMSE de las predicciones a una, dos y tres horas para el año 2016 [Tabla 4.11], observamos que sigue existiendo un amplio margen de mejora, y como el modelo

con menos neuronas tiene un mejor ajuste en entrenamiento, este será el que escojamos.

Tabla 4.11: RMSE de las predicciones del modelo con una única capa oculta de 22 neuronas y parámetro de regularización 0.8 para la localización 73 usando datos de las 7h anteriores más frecuencias significativas.

nº horas	1	2	3
RMSE	0.7414821	0.8632859	0.9329571

### Red de una capa oculta para la localización 2137

Comenzamos realizando la validación cruzada con el mismo espacio de búsqueda que para el modelo de la localización 73, es decir, con parámetros de regularización 0.5, 0.1, 0.01 y un número de neuronas entre 1 y 25.

El modelo seleccionado de entre todas las opciones posibles consiste en una red con 10 neuronas en la capa oculta y un parámetro de regularización de 0.5, obteniendo un RMSE en validación de 0.5086070. Sin embargo, si analizamos la capacidad predictiva del modelo (en el sentido de los RMSE de las predicciones a una, dos y tres horas para el año 2016 [Tabla 4.12]) observamos que la capacidad de generalización del modelo es bastante mejorable. Si analizamos los intervalos de predicción de las predicciones [Figura 4.16], observamos que, si bien se observa una mejora notable respecto al modelo entrenado únicamente con las frecuencias significativas, todavía existe un amplio margen de mejora.

Tabla 4.12: RMSE de las predicciones del modelo con una única capa oculta de 10 neuronas y parámetro de regularización 0.5 para la localización 2137 usando datos de las 7h anteriores más frecuencias significativas.

nº horas	1	2	3
RMSE	0.7046581	0.8102288	0.8662693

Como hemos obtenido el valor máximo del espacio de búsqueda, vamos a cambiar el espacio de búsqueda del parámetro decay a los valores 0.5, 0.6, 0.7, 0.8, 0.9 y 1.

Realizando la validación cruzada en el nuevo espacio de búsqueda se selecciona un modelo con 19 neuronas en la capa oculta y un parámetro de regularización de 1 (el máximo posible), por lo que puede que tenga sentido aumentar el espacio de búsqueda del parámetro de regularización. El modelo con 19 neuronas y parámetro de regularización 1 tiene un RMSE medio entre folds de 0.5079752. Si analizamos los RMSE de las predicciones a una, dos y tres horas para el año 2016 [Tabla 4.13], observamos que el modelo con menos neuronas tiene mayor capacidad de generalización (en el sentido del RMSE en el conjunto de entrenamiento). Sin embargo, como los RMSE de ambos modelos están relativamente cerca, vamos a expandir el espacio de búsqueda una vez más para intentar encontrar los mejores hiperparámetros.

Tabla 4.13: RMSE de las predicciones del modelo con una única capa oculta de 19 neuronas y parámetro de regularización 1 para la localización 2137 usando datos de las 7h anteriores más las frecuencias significativas.

nº horas	1	2	3
RMSE	0.7089287	0.8128849	0.8684515

Expandiendo el espacio de búsqueda del parámetro de regularización a el conjunto  $\{1, 1.1, 1.2\}$  volvemos a obtener el modelo con 19 neuronas ocultas y parámetro de regularización 1, que sabemos que tiene una peor capacidad de generalización que el modelo con 10 neuronas y parámetro de regularización 0.5.

### Perceptrón multicapa para la localización 73

Como hemos podido observar en las secciones anteriores, los modelos de redes neuronales de una capa tienen problemas a la hora de predecir nuevos valores, al menos con el número de neuronas permitido por la librería `nnet` de R. Por lo tanto, nuestro siguiente paso consistirá en utilizar redes neuronales con varias capas ocultas para comprobar si la capacidad predictiva se resiente debido a que el modelo no es lo suficiente complejo o debido a que no tiene una buena capacidad de generalización.

De manera similar al proceso realizado para las redes de una capa, seleccionaremos mediante validación cruzada modelos de redes neuronales de varias capas mediante la librería `SRNNS`, realizando la búsqueda en la rejilla de parámetros  $\text{layer1} \in \{3, 4, \dots, 10\}$ ,  $\text{layer2} \in \{1, 2, \dots, 7\}$ ,  $\text{layer3} \in \{1, 2, \dots, 10\}$ , que corresponden al número de neuronas en la primera, segunda y tercera capa oculta, respectivamente (esta selección viene motivada por los resultados obtenidos para la red multicapa de la localización 2137).

El modelo seleccionado por validación cruzada consta de 5, 6 y 9 neuronas en la primera, segunda y tercera capa oculta, respectivamente. Obteniendo unos RMSE peores que si predijéramos el valor constante 0 [Tabla 4.14] por lo que podemos concluir que el problema que teníamos se debía a la falta de capacidad de generalización más que a la falta de complejidad del modelo.

Tabla 4.14: RMSE de las predicciones del modelo con 5,6 y 9 neuronas en las capas ocultas

nº horas	1	2	3
RMSE	0.9403219	1.163371	1.23959

### Perceptrón multicapa para la localización 2137

Al igual que para la localización 73, vamos a probar si la falta de capacidad predictiva del modelo se debe a que el modelo no es lo suficiente complejo o a que no tiene una buena capacidad de generalización.

Seleccionamos mediante validación cruzada modelos de redes neuronales de varias capas mediante la librería **SRNNS**, realizando la búsqueda en la rejilla de parámetros  $\text{layer1} \in \{3, 4, \dots, 7\}$ ,  $\text{layer2} \in \{1, 2, \dots, 7\}$ ,  $\text{layer3} \in \{1, 2, \dots, 7\}$ , que corresponden al número de neuronas en la primera, segunda y tercera capa oculta, respectivamente.

El modelo seleccionado por validación cruzada consta de 7, 3 y 7 neuronas en la primera, segunda y tercera capa oculta, respectivamente. Los RMSE obtenidos se muestran en la Tabla Tabla 4.15. Como los parámetros **layer1** y **layer3** seleccionados están en el límite del espacio de búsqueda vamos a aumentar el espacio de búsqueda de estas variables a  $\text{layer1} \in \{3, 4, \dots, 10\}$ ,  $\text{layer3} \in \{1, 2, \dots, 10\}$ .

Tabla 4.15: RMSE de las predicciones del modelo con 7,3 y 7 neuronas en las capas ocultas

nº horas	1	2	3
RMSE	0.7036953	0.8175858	0.8793941

El modelo seleccionado por validación cruzada en el nuevo espacio de búsqueda tiene 7, 5 y 9 neuronas en la primera, segunda y tercera capa oculta, respectivamente. Los RMSE obtenidos se muestran en la tabla 4.16.

Tabla 4.16: RMSE de las predicciones del modelo con 7,5 y 9 neuronas en las capas ocultas

nº horas	1	2	3
RMSE	0.7158819	0.8342612	0.8977508

Aunque el modelo con 7,3 y 7 neuronas en las capas ocultas tiene un RMSE a una hora ligeramente mejor que la red neuronal con una capa oculta de 10 neuronas, consideraremos el segundo como mejor opción por tener mejores RMSE a 2 y 3 horas, además de ser un modelo mucho más simple.

#### 4.2.3. Intervalos de predicción del modelo final para la localización 73

Una vez analizados los RMSE de los distintos modelos seleccionamos un modelo con una única capa oculta de 12 neuronas y parámetro de regularización 0.5. Se observa un  $PICP \geq 0.95$  en el conjunto de entrenamiento con  $\sigma = 0.6$ . Si representamos los intervalos de predicción en el conjunto de test [Figura 4.15] observamos que su calidad es peor que para los del modelo ARMA, tenemos intervalos más amplios y que cubren una menor cantidad de los valores observados.

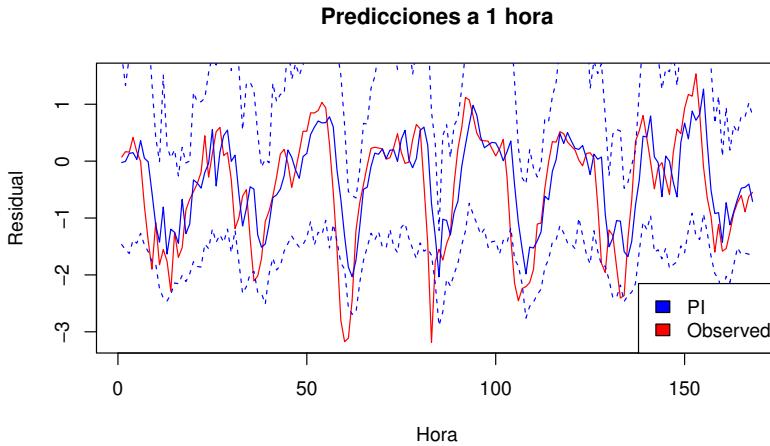


Figura 4.15: Intervalos de predicción para la red neuronal con una capa oculta de 12 neuronas y parámetro de regularización 0.5 para la localización 73 usando datos de las 7h anteriores más las frecuencias significativas.

#### 4.2.4. Intervalos de predicción del modelo final para la localización 2137

En el caso de la localización 2137 nos tomamos como modelo final la red neuronal con una única capa oculta de 10 neuronas y parámetro de regularización 0.5. En este caso observamos un  $PICP \geq 0.95$  en el conjunto de entrenamiento con  $\sigma = 0.7$ . Analizando la gráfica de los intervalos de predicción [Figura 4.16] observamos que su calidad es peor la del modelo ARMA, ya que, a pesar de que los intervalos de predicción cubren la mayoría de los valores observados, la amplitud de los intervalos para la red neuronal es mucho mayor que para el modelo ARMA.

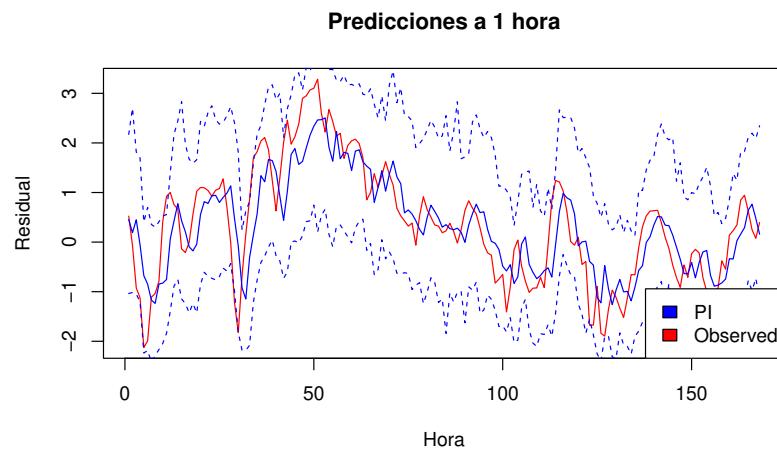


Figura 4.16: Intervalos de predicción para la red neuronal con una capa de 10 neuronas y parámetro de regularización 0.5 para la localización 2137 usando datos de las 7h anteriores más las frecuencias significativas.

# Capítulo 5

## Conclusiones

Antes de comenzar a describir las conclusiones obtenidas vamos a realizar un breve resumen de los temas tratados durante el trabajo.

Hemos comenzado introduciendo los conceptos básicos de las series temporales y los modelos ARMA, describiendo como aplicarlos con el software R. Posteriormente hemos realizado una introducción a los modelos de redes neuronales, junto con su correspondiente aplicación en R. También hemos descrito el proceso realizado por Huang et al. (2021) para obtener la serie de residuales de la velocidad del viento con la que hemos trabajado.

A continuación hemos entrenado distintos modelos ARMA, estudiando la influencia de la dimensionalidad de la serie de entrada en la capacidad predictiva del modelo y obteniendo modelos finales con resultados bastante satisfactorios en el sentido de los intervalos de predicción.

Finalmente hemos entrenado distintos modelos de redes neuronales, tanto de 2 capas (con una única capa oculta) como de hasta 4 capas (con 3 capas ocultas), obteniendo peores resultados que con los modelos ARMA.

### 5.1. Comparación de los modelos en las distintas localizaciones

Vamos a proceder a comparar las capacidades predictivas de los modelos ARMA y de redes neuronales seleccionados para cada una de las localizaciones, comparando los RMSE de las predicciones a una, dos y tres horas de cada uno de los modelos.

Analizando los RMSE de las predicciones [Tabla 5.1] observamos un comportamiento similar en ambas localizaciones. El modelo ARMA tiene mejor capacidad predictiva a una, dos y tres horas. Sin embargo, podemos observar que mientras que el modelo ARMA tiene una pérdida notable de capacidad entre las predicciones a una y dos horas, la caída para la red neuronal es menos notable.

A la vista de los resultados podemos concluir que los modelos ARMA resultan más útiles a la hora de predecir las series temporales de residuales del

Tabla 5.1: RMSE de las predicciones del modelo ARMA(24,5), la red neuronal de dos capas con 12 neuronas para la localización 73 y el modelo ARMA(24,5) y la red neuronal de dos capas con 19 neuronas para la localización 2137

Localización	Modelo	RMSE 1h	RMSE 2h	RMSE 3h
73	ARMA	0.5077213	0.7181873	0.8156235
73	NN	0.7368629	0.853078	0.9221011
2137	ARMA	0.5210933	0.6915848	0.7886777
2137	NN	0.7046581	0.8102288	0.8662693

viento que hemos tratado, al menos sin tener un comportamiento más profundo del comportamiento de la serie que nos permita seleccionar transformaciones que representen una mayor cantidad de información de la serie. Esto era de esperar en cierto modo, ya que los modelos ARMA se definen para series temporales, mientras que los modelos de redes neuronales clásicos que hemos tratado requieren de una adaptación de la serie para poder modelizarla.

## 5.2. Conclusiones sobre la implementación

Como toda el análisis se ha realizado en R vamos a hacer un pequeño resumen de los puntos fuertes y/o debilidades de cada uno de los modelos a la hora de implementarlos en dicho lenguaje.

En lo que respecta a los modelos ARMA, a pesar de que la experimentación se programa de manera relativamente sencilla, las funciones de selección automática son computacionalmente costosas. Esto, unido a que las opciones de paralelización incluidas tanto en el paquete `forecast` (la función `auto.arima` dispone de un argumento `parallel` que no funciona correctamente) como en `doParallel` no funcionan correctamente en este contexto hace que algunos de los modelos tarden varios días en entrenarse (se tardó tres días en obtener el modelo entrenado con 2 años para la localización 73).

El caso de las redes es similar en el sentido de que la implementación del entrenamiento es muy sencilla, gracias en gran medida al paquete `caret`. Además, las funciones de entrenamiento funcionan en tiempos muy razonables siempre que elijamos la librería correcta, ya que, por ejemplo al aplicar `train` con el método '`neuralnet`' forzaba a seleccionar un ratio de aprendizaje máximo de  $10^{-5}$  para poder ejecutarlo, con lo que la computación se vuelve inabordable para nuestro problema (hubo que detener su ejecución tras una semana para aprovechar la capacidad computacional en el entrenamiento con otros métodos).

El alto coste computacional de los modelos ARMA puede suponer que no sea viable implementarlos para realizar predicciones a corto plazo de series de alta frecuencia, ya que puede suceder que se requiera un tiempo mayor para ajustar el modelo que el tiempo del que se dispone entre observaciones. Esto puede resultar en que no podamos entrenar el modelo con cada nueva observación y termine quedando obsoleto. Las redes neuronales no cuentan con esta limitación, ya que

disponen de métodos en serie para actualizarse, de modo que podemos actualizar la red con cada nueva observación sin tener que realizar el entrenamiento con todos los datos. Por tanto, es muy posible que si se encuentra un modelo de redes neuronales adecuado para el problema su implementación resulte más práctica para realizar predicciones a corto plazo de series temporales de alta frecuencia.

Las pruebas mencionadas anteriormente se han realizado en un equipo con sistema operativo Windows 10, 8 GB de memoria RAM y un procesador Intel Core i5 de 2.4 GHz con 4 núcleos.

### 5.3. Líneas futuras

Una línea futura podría consistir en realizar un análisis más profundo de las frecuencias significativas de la serie de Fourier de la serie de residuales del viento de modo que podamos definir una transformación de los datos que nos proporcione un buen equilibrio entre el número de variables y la información representada en éstas.

Otra opción consiste en intentar adaptar los métodos propuestos para utilizar la información espacial, ya que es de esperar que el comportamiento de las series temporales sea similar en localizaciones cercanas.

Como se ha mencionado a la hora de comparar los resultados, hemos utilizado modelos de redes neuronales clásicos, que no están pensados para el análisis de series temporales. En nuestro caso la adaptación a las series temporales se ha realizado transformando los datos, obteniendo resultados mejorables. Por tanto, otra futura línea de investigación podría consistir en la utilización de redes neuronales con un planteamiento más cercano al de las series temporales y que se adecuen mejor al problema.

### Material Suplementario

El código R para replicar todos los análisis realizados en este trabajo se encuentra disponible en el siguiente repositorio de GitHub: [https://github.com/spatialstatisticsupna/TFM\\_HarkaitzGoyena](https://github.com/spatialstatisticsupna/TFM_HarkaitzGoyena)

### Agradecimientos

Este Trabajo Fin de Máster ha sido realizado bajo la financiación de las Ayudas de Iniciación a la Investigación de la Universidad Pública de Navarra en el ámbito de sus Institutos de Investigación durante el curso académico 2020/2021 (resolución nº 1133/2021).

# Bibliografía

- Almeida, V. and Gama, J. (2015). Prediction intervals for electric load forecast: Evaluation for different profiles. In *2015 18th International Conference on Intelligent System Application to Power Systems (ISAP)*, pages 1–6. IEEE.
- Bergmeir, C. and Benítez, J. M. (2012). Neural networks in R using the stuttgart neural network simulator: RSNNS. *Journal of Statistical Software*, 46(7):1–26.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.
- Brockwell, P. J., Brockwell, P. J., Davis, R. A., and Davis, R. A. (2016). *Introduction to time series and forecasting*. Springer.
- Brockwell, P. J. and Davis, R. A. (2009). *Time series: theory and methods*. Springer Science & Business Media.
- Dickey, D. A. and Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74(366a):427–431.
- Giani, P., Tagle, F., Genton, M. G., Castruccio, S., and Crippa, P. (2020). Closing the gap between wind energy targets and implementation for emerging countries. *Applied Energy*, 269:115085.
- Gneiting, T. (2002). Nonseparable, stationary covariance functions for space–time data. *Journal of the American Statistical Association*, 97(458):590–600.
- Huang, H., Castruccio, S., and Genton, M. G. (2021). Forecasting high-frequency spatio-temporal wind power with dimensionally reduced echo state networks. *arXiv preprint arXiv:2102.01141*. <https://arxiv.org/abs/2102.01141>.
- Hyndman, R., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., O’Hara-Wild, M., Petropoulos, F., Razbash, S., Wang, E., and Yasmeen, F. (2021). *forecast: Forecasting functions for time series and linear models*. R package version 8.15. <https://pkg.robjhyndman.com/forecast/>.
- IPCC (2014). *Part a: Global and sectoral aspects. In AR5 Climate Change 2014: Impacts, Adaptation, and Vulnerability*. Cambridge University Press.

- Kuhn, M. (2021). *caret: Classification and Regression Training*. R package version 6.0-88. <https://CRAN.R-project.org/package=caret>.
- Kwiatkowski, D., Phillips, P. C., Schmidt, P., and Shin, Y. (1992). 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):159–178.
- Ljung, G. M. and Box, G. E. (1978). On a measure of lack of fit in time series models. *Biometrika*, 65(2):297–303.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.
- Skamarock, W. C., Klemp, J. B., Dudhia, J., Gill, D. O., Barker, D. M., Duda, M. G., Huang, X.-Y., Wang, W., and Powers, J. G. (2008). A description of the advanced research WRF version 3, NCAR Technical Note. *National Center for Atmospheric Research: Boulder, CO, USA*.
- Trapletti, A. and Hornik, K. (2020). *tseries: Time Series Analysis and Computational Finance*. R package version 0.10-48. <https://CRAN.R-project.org/package=tseries>.
- Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. ISBN 0-387-95457-0.

# Índice de figuras

2.1.	Ejemplo de un perceptrón multicapa general . . . . .	15
2.2.	Representación de la función de error $E(\mathbf{w})$ como una superficie sobre el espacio de los pesos. Fuente: Bishop (2006) . . . . .	16
2.3.	Representación del cálculo de $\delta_j$ para las unidades ocultas $j$ por propagación hacia atrás de los $\delta$ de las unidades $k$ a las que la unidad $j$ envía conexiones. Fuente: Bishop (2006) . . . . .	20
3.1.	Posición de las localizaciones con cuyos datos vamos a trabajar (en gris el resto de localizaciones del conjunto de datos) . . . . .	26
3.2.	Velocidades residuales medias del viento (en azul) junto con sus desviaciones estándar (en rojo) en la localización 73 . . . . .	28
3.3.	Velocidades residuales medias del viento (en azul) junto con sus desviaciones estándar (en rojo) en la localización 2137 . . . . .	28
4.1.	ACF y PACF muestrales de la serie de entrenamiento . . . . .	31
4.2.	Distribución de los residuales del modelo AR(24) . . . . .	32
4.3.	Intervalos de predicción a una hora para el modelo AR(24) (Zoom a la primera semana) Las líneas azules representan la predicción junto con sus intervalos, mientras que la roja representa las observaciones . . . . .	32
4.4.	Intervalos de predicción a una hora para el modelo ARMA(24,13)	33
4.5.	Intervalos de predicción a una hora para el modelo ARMA(24,11)	34
4.6.	Gráficas de análisis de residuales del modelo ARMA(24,3) entrenado con 6 meses . . . . .	35
4.7.	Intervalos de predicción a una hora para el modelo ARMA(24,3) entrenado con 6 meses . . . . .	35
4.8.	Gráficas de análisis de residuales del modelo ARMA(24,3) entrenado con 3 meses . . . . .	36
4.9.	Gráficas de residuales del modelo ARMA(24,5) seleccionado para la localización 73 . . . . .	37
4.10.	Intervalos de predicción a una hora para el modelo ARMA(24,5) seleccionado para la localización 73 . . . . .	38
4.11.	ACF y PACF muestrales de la serie de entrenamiento de la localización 2137 . . . . .	39

4.12. Intervalos de predicción a una hora para el modelo AR(24) . . . . .	40
4.13. Gráficas de residuales del modelo ARMA(24,6) seleccionado para la localización 2137 . . . . .	41
4.14. Intervalos de predicción a una hora para el modelo ARMA(24,6) seleccionado para la localización 2137 . . . . .	42
4.15. Intervalos de predicción para la red neuronal con una capa oculta de 12 neuronas y parámetro de regularización 0.5 para la localización 73 usando datos de las 7h anteriores más las frecuencias significativas. . . . .	49
4.16. Intervalos de predicción para la red neuronal con una capa de 10 neuronas y parámetro de regularización 0.5 para la localización 2137 usando datos de las 7h anteriores más las frecuencias significativas. . . . .	50

# Índice de tablas

4.1.	RMSE de las predicciones del modelo AR(24) . . . . .	30
4.2.	RMSE de las predicciones del modelo ARMA(24,13) obtenido utilizando la serie de entrenamiento completa . . . . .	30
4.3.	RMSE de las predicciones del modelo ARMA(24,11) obtenido utilizando la mitad de la serie de entrenamiento correspondiente al año 2014 . . . . .	33
4.4.	RMSE de las predicciones del modelo ARMA(24,3) obtenido utilizando las observaciones de la segunda mitad de 2014 . . . . .	34
4.5.	RMSE de las predicciones del modelo ARMA(24,3) obtenido utilizando las observaciones de los últimos tres meses de 2014 . . . . .	34
4.6.	RMSE de las predicciones del modelo ARMA(24,5) seleccionado para la localización 73 . . . . .	37
4.7.	RMSE de las predicciones del modelo AR(24) . . . . .	40
4.8.	RMSE de los distintos modelos ARMA . . . . .	40
4.9.	RMSE de las predicciones del modelo ARMA(24,6) seleccionado para la localización 2137 . . . . .	41
4.10.	RMSE de las predicciones del modelo con una única capa oculta de 12 neuronas y parámetro de regularización 0.5 para la localización 73 usando datos de las 7h anteriores más frecuencias significativas. . . . .	45
4.11.	RMSE de las predicciones del modelo con una única capa oculta de 22 neuronas y parámetro de regularización 0.8 para la localización 73 usando datos de las 7h anteriores más frecuencias significativas. . . . .	46
4.12.	RMSE de las predicciones del modelo con una única capa oculta de 10 neuronas y parámetro de regularización 0.5 para la localización 2137 usando datos de las 7h anteriores más frecuencias significativas. . . . .	46
4.13.	RMSE de las predicciones del modelo con una única capa oculta de 19 neuronas y parámetro de regularización 1 para la localización 2137 usando datos de las 7h anteriores más las frecuencias significativas. . . . .	47
4.14.	RMSE de las predicciones del modelo con 5,6 y 9 neuronas en las capas ocultas . . . . .	47

4.15. RMSE de las predicciones del modelo con 7,3 y 7 neuronas en las capas ocultas . . . . .	48
4.16. RMSE de las predicciones del modelo con 7,5 y 9 neuronas en las capas ocultas . . . . .	48
5.1. RMSE de las predicciones del modelo ARMA(24,5), la red neuronal de dos capas con 12 neuronas para la localización 73 y el modelo ARMA(24,5) y la red neuronal de dos capas con 19 neuronas para la localización 2137 . . . . .	52