Escuela Técnica Superior de Ingeniería (ICAI)

Grado en Ingeniería Electromecánica

Especialidad Mecánica

Forecasting System based on Analytics of Interval-valued Databases

---

Author: Pablo Cebral de Juan-Roncero

Supervisor: Carlos Maté Jiménez

Madrid, Julio de 2018

Forecasting System based on Analytics of Interval-valued Databases

Author: Pablo Cebral de Juan-Roncero
Supervisor: Carlos Maté Jiménez
Collaborating Entity: ICAI - Universidad Pontificia Comillas

Abstract

Improvements in technology in the past two decades have led to an incredible growth in the amount of data that is recorded, treated and analyzed every day. Modern databases are now so huge and more importantly, heterogeneous, that harnessing relevant data has become more and more challenging. In this 'Big Data' scenario the ability to exploit data is key in almost any business. A smart use of the information that is available may be the key differentiator between failure and success in an environment as competitive as the current world. Strategic decisions are now data driven and data analysis is crucial in almost any field. Forecasting has long been present, way before the modern concept of technology was even established, the Roman civilization used what was known as divination, or as Cicero described it: 'The power to see, understand, and explain premonitory signs given to men by the gods', (Cicero, De Divinatione, 45 B.C.). Ever since, forecasters have constantly pursued to reduce the magical or godly component in forecasting, wishing to utopianly eliminate it by including statistics and mathematics to build more precise models that provided more accurate estimations.

The will of forecasters and analysts to produce the ultimate prediction has always been thwarted because of uncertainty. Uncertainty describes situations that have ambiguous or unknown information and it is inherent to any physical process. Thus, it cannot be eliminated, only reduced. Forecasters have mostly

focused on the refinement of the techniques that perform the prediction but may have overlooked the importance of the data they are operating with. Classic time series analysis relies on point-valued data, this is, for a given time period t a crisp value x_t is the representative of the whole series until a new observation x_{t+1} is collected. This approach has recently found to be limited in some fields of study, as a point representation disregards a lot of relevant information that explains certain phenomena. In this context, a new line of investigation has been recently opened that proposes the formulation of the classic methods for point-valued data using Symbolic Data Analysis (SDA).

In time series analysis, for instance, representation of non-monotonic phenomena using intervals is of much better use. Say an analysis of a highly volatile process needs to be carried out, by using points as representatives of the process, the information that is related to the recurrent fluctuations is disregarded and can easily acquainted for using ranges as representatives.

SDA traces back years ago but it till is a highly unexplored field, where major scientific advances have only been found in the context histograms and, especially, intervals with Interval Arithmetic (IA). IA has already provided researchers and mathematicians with an incredibly powerful tool to implement analysis techniques in fields such as descriptive statistics or data mining. In the context of forecasting and data analytics, the development of models using crisp data is widespread among the researching community, with a collection of well-established methods that have suffered little to none modifications since its first proposal. However, the use of IA in applied analytics is still only a growing field of investigation as proved by the number of publications of articles and citations related to the topic, that have raised exponentially over the past decade.

The purpose of this project is to continue this line of investigation, with an initial exploration of the state of the art in the context of classical time series, provided that most methods designed for crisp data are applicable to intervals. Linear methods, such as linear regression, exponential smoothing, autoregression and moving average

approaches will be discussed, as well as other nonlinear methods for classic time series analysis, such as k-nearest neighbors or artificial neural networks.

A special focus will be deployed in the latter, since there will be necessary to develop a thorough understanding of the aforementioned structures in order to introduce the later part of the project: the implementation of a nonlinear forecasting method based on the use of artificial neural network structures adapted to handle interval-valued data. The application that will be carried out has already been proposed in previous research and has proved to yield promising results in the predictive analysis field, namely forecasting.

However, prior to the documentation of the model implemented, it is necessary to clarify the specificities of interval time series as well as interval arithmetic, that will be crucial for the later implementation and evaluation of the proposed forecasting technique. For instance, establishing forecast accuracy measures will be shown to be a task far more difficult for ITS than it is for point-valued data.

Ultimately, the original implementation of the established model of an interval multilayer perceptron in the MatLab programming environment will be thoroughly explained, acquainting for the mathematical and computational details of the tool to serve as background knowledge that will facilitate the understanding of the case studies that have been carried out for the project. Finally, the model will be evaluated in the context of real financial ITS forecasting, specifically in the fields of foreign exchange and equity markets. The method will be evaluated against robust benchmark forecasting techniques in said fields that will acknowledge for the predictive capability while helping to shed some light over the weaknesses of the method. Inspiring ideas of future improvements as well as other fields of applications that may benefit from the model application, such as electricity markets and preventive medicine.

Sistemas de Predicción basados en Analítica de Bases de datos de Intervalos

Autor: Pablo Cebral de Juan-Roncero
Director: Carlos Maté Jiménez
Entidad Colaboradora: ICAI - Universidad Pontificia Comillas

Resumen

Las mejoras en la tecnología en las últimas décadas han llevado a un crecimiento increíble en la cantidad de datos que se registran, tratan y analizan todos los días. Las bases de datos modernas ahora son tan enormes y lo más importante, heterogéneas, que aprovechar los datos de manera pertinente se ha convertido en una tarea cada vez más desafiante. En este escenario 'Big Data', la capacidad de explotación de los datos es clave en casi cualquier sector empresarial. Un uso inteligente de la información puede ser el factor diferenciador entre el fracaso y el éxito en un entorno tan competitivo como el mundo actual. Las decisiones estratégicas son basadas en análisis de datos en casi cualquier campo. Los métodos de predicción han estado presentes mucho antes incluso de que fuera establecido el concepto moderno de tecnología, la civilización romana utilizaba lo que era conocido como adivinación o, como Cicero describió: ' el poder de ver, entender y explicar signos premonitorios, otorgado a los hombres por los dioses', (Cicerón, De Divinatione, 45 A.C.). Desde entonces, los analistas han buscado constantemente maneras de reducir el componente mágico o piadoso de la generación de pronósticos, deseando eliminarla de manera utópica mediante la inclusión de técinas estadísticas y matemáticas para la generación de modelos más robustos que proporcionen estimaciones más precisas.

La voluntad de los expertos y analistas de producir una estimación perfecta siempre se ha visto frustrada debido a la incertidumbre. La incertidumbre es inherente

a cualquier proceso del mundo real, y se magnifica en situaciones con información ambigua o desconocida. Así, es muy difícil eliminarla, pudiendo solamente reducirla. Los analistas se han centrado históricamente, sobre todo, en el refinamiento de las técnicas que de estimación, generando modelos de mayor complejidad cada vez, pero puede que hayan pasado por alto la importancia de los datos con los que están operando. En el contexto de análisis de series temporales clásicas se ha basado históricamente en la explotación datos puntuales, esto es, para un momento determinado en el tiempo, llamado período t, un punto x_t representa toda la serie hasta que una nueva observación, x_{t+1}, sea recogida. Este enfoque ha encontrado recientemente limitaciones importantes en algunos campos de estudio, debido a que la representación puntual de un fenómeno omite mucha información relevante que puede ser explicativa de las características que definen la naturaleza del mismo. En este contexto, una nueva línea de investigación se ha abierto recientemente, la cual propone la formulación de los métodos clásicos de análisis de datos establecidos para valores puntuales utilizando análisis de datos simbólicos (ADS).

En el marco de las series temporales, por ejemplo, la representación de fenómenos no monotónos utilizando datos de intervalares ha resultado en grandes mejoras. Supongamos el contexto de análisis de un proceso altamente volátil, el cual se ha planteado desde el paradigma clásico de uso de datos puntuales como representantes del proceso. La información que se relaciona con las fluctuaciones que ocurren de manera reiterada en el proceso difícilmente puede quedar reflejada en la aproximación puntual, es entonces cuando la formulación intervalar comienza a considerarse intersante.

Los inicios del análisis de datos simbólicos se remontan años atrás; no obstante, por el momento sigue siendo un campo altamente inexplorado, donde sólo se han encontrado avances científicos relevantes en el marco de los datos en forma de histogramas y, especialmente, en la representación intervalar con la pertinente aritmética de intervalos (AI). La AI ha proporcionado ya a investigadores y matemáticos con una herramienta increíblemente poderosa para implementar técnicas de análisis en campos como la estadística descriptiva o la minería de datos, entre

otros. En el contexto del análisis de datos y los métodos de predicción, el desarrollo de modelos utilizando datos puntuales está muy asentado entre la comunidad científica, con una colección de métodos bien establecidos que han sufrido poca o ninguna modificación desde su primera propuesta. Sin embargo, el uso de la AI en el marco de la analítica aplicada está sufriendo constantes avances y es ya un prometedor campo de investigación. Como demuestra el aumento exponencial en la última década del número de publicaciones de artículos y citas relacionadas con el tema.

El propósito de este proyecto es continuar esta línea de investigación, llevando a cabo una exploración inicial del estado del arte en el contexto clásico de las series temporales, debido a que la mayoría de los métodos establecidos para datos puntuales es también aplicable en el paradigma intervalar. Los métodos lineales, tales como la regresión lineal, el suavizado exponencial, los modelos autorregresivos y los métodos basados en el uso de promedios, así como otros métodos no lineales para el análisis de temporales clásicas, tales como k-más cercano a vecinos o el uso redes neuronales artificiales serán analizados.

Se prestará especial atención a los métodos que trabajan las redes neuronales artificiales, ya que será necesario desarrollar un conocimiento profundo de las estructuras mencionadas con el fin de poder introducir la segunda parte del proyecto: la implementación de un método generación de pronósticos no lineal basado en el uso de estructuras de redes neuronal artificiales adaptadas al manejo de datos de tipo intervalo, en concreto, la adaptación del consolidado perceptrón multicapa para datos de intervalo. La implementación que se llevará a cabo se ha propuesto ya en investigaciones anteriores y ha conseguido resultados prometedores en el campo del análisis predictivo, es decir, y concretamente en los métodos de predicción.

Sin embargo, antes de la documentación del modelo implementado, será necesario aclarar las particularidades de las series temporales bajo el marco del paradigma intervalar, así como de la aritmética de intervalos asociada. Ésto será crucial para el posterior entendimiento de la implementación y su consiguiente evaluación de la herramienta propuesta. Por ejemplo, establecer medidas de evaluación de la precisión

de pronóstico será una tarea significativamente más compleja para datos de intervalo que su correspondiente desarrollo en el contexto de datos de tipo de punto.

En última instancia, la implementación original del modelo descrito en el entorno de programación MatLab será documentada en detalle. Atendiento a las especificaciones matemáticas y computacionales asociadas al modelo que servirán como punto de partida para el entendimiento de la posterior evaluación de la herramienta con series temporales de intervalo reales. Concretamente, se analizará el rendimiento del perceptrón multicapa para intervalos en el marco de las series financieras, específicamente en el contexto de los mercados de divisas y de acciones. El método propuesto será evaluado tanto de manera individual atendiendo a errores de predicción en las series generadas, como de manera relativa utilizando como referencia modelos predictivos muy establecidos en el marco de los casos de estudio realizados. De esta manera podrá evaluarse la capacidad predictiva de la herramienta al mismo tiempo que se facilitará la identificación de los puntos débiles del modelo que servirá para orientar las posibles propuestas de mejora. Además, se sugerirán posibles campos de aplicación alternativos a los analizados en este proyecto, como la predicción de precios de electricidad, o la aplicación en contextos de medicina preventiva.

Contents

1. Single-valued Data Forecasting Methods **9**

1.1. Time Series 9

1.1.1. Definition 9

1.1.2. Decomposition: Trend, Seasonality and Residuals 9

1.2. Introduction to Forecasting 12

1.2.1. Simple Forecasting Methods 12

1.2.2. Error Measures 13

1.3. Linear Forecasting Methods 16

1.3.1. Linear Regression 16

1.3.2. Exponential Smoothing 17

1.4. Autoregressive Moving Average Methods 19

1.4.1. Background Concepts 19

1.4.2. ACF and PACF Models 19

1.4.3. Non-seasonal ARIMA models 21

1.4.4. Seasonal ARIMA 22

1.5. Non-linear Methods 26

1.5.1. Non-linear Regression 26

1.5.2. k-NN Method 27

1.5.3. Neural Networks 29

2. Artificial Neural Networks Application in Forecasting **31**

2.1. Definition 32

2.2. Components of an ANN 33

2.2.1. Connections . . . 34
2.2.2. The Propagation Function . . . 34
2.2.3. Network Input . . . 34
2.2.4. Threshold Value . . . 34
2.2.5. The Activation Function . . . 34
2.2.6. Output function . . . 36
2.3. Types of ANN . . . 36
2.3.1. Single-Layer Perceptron . . . 36
2.3.2. Multi Layer Perceptron . . . 37
2.3.3. Radial Basis Network (RBN) . . . 37
2.3.4. Recurrent Neural Network . . . 39
2.3.5. Hopfield Network . . . 39
2.4. Training of a ANN . . . 41
2.4.1. Supervised Learning . . . 41
2.4.2. Reinforcement Learning . . . 42
2.4.3. Unsupervised Learning . . . 43
2.4.4. Gradient Descent and Ascent . . . 45
2.4.5. Newton's Method . . . 47
2.4.6. Levenberg-Marquardt . . . 48
2.4.7. Delta Rule . . . 48
2.5. MLP training: Step by Step . . . 50
2.5.1. Backpropagation of Error . . . 51
2.6. Overfitting in ANN . . . 52
2.6.1. Definition . . . 52
2.6.2. How to Prevent Overfitting . . . 52
3. Forecasting with Interval-Valued Data **55**
3.1. Intervals and their Advantages in Statistics . . . 55
3.2. Properties of Intervals . . . 56
3.2.1. Definition and Notation . . . 56
3.2.2. Interval Arithmetic . . . 57

3.2.3. Interval Time Series 59
3.3. Standard Distance Measures 60
3.4. Intervals and Probability 63
3.4.1. Probability Distribution Function 63
3.4.2. Distance Measures between Probability Distributions 64
3.4.3. Properties of $\mathcal{L}^1$ and $\mathcal{L}^2$ 65
3.4.4. Application of Probability-based Metrics 66
3.4.4.1. Error Measures 66
3.5. Statistical Forecasting with ITS 67
3.5.1. Simple Methods 67
3.5.2. Exponential Smoothing 68
3.5.3. ARIMA Models for ITS 70
3.5.4. Linear Regression 71
3.6. Non-linear Forecasting with ITS 73
3.6.1. Non-linear Regression 73
3.6.2. k-NN Method 74
3.6.3. Interval Artificial Neural Networks 76
3.6.3.1. Interval Feed-forward Network 76
3.6.3.2. iMLP 78

4. interval MultiLayer Perceptron: a Matlab Approach **83**

4.1. Foundation 83
4.1.1. Objective 84
4.1.2. Target Users 84
4.2. Model Layout and Hyperparametrization 84
4.2.1. Neuron Evaluation 85
4.3. Learning Algorithm 85
4.3.1. SGDM 86
4.3.2. Training Advantages 87
4.4. Technical Requirements and Dependencies 88

4.4.1. Matlab . . . 88
4.4.2. Technical Requirements . . . 90
4.5. Functionality . . . 91
4.5.1. Model Fitting . . . 91
4.5.2. Forecasting . . . 94

5. Case Studies: Univariate Analysis with Historical ITS Data 97
5.1. Foreign Exchange Markets . . . 97
5.1.1. Introduction . . . 97
5.1.2. Forecasting the Forex markets . . . 99
5.1.2.1. Random Walk model . . . 99
5.1.3. Setup . . . 101
5.1.4. Results: AUD/USD ITS . . . 103
5.1.5. Results: EUR/USD ITS . . . 105
5.2. Stock Markets . . . 107
5.2.1. Introduction . . . 107
5.2.2. Setup . . . 112
5.2.3. Results: IBEX35 ITS . . . 113
5.2.4. Results: FORD Motor Company ITS . . . 114

6. Conclusions and Open Paths for the Future 117
6.1. Conclusions . . . 117
6.1.1. Learning Capability . . . 118
6.1.2. Forecast Performance . . . 120
6.2. Open Paths for the Future . . . 126
6.2.1. Tool Optimization . . . 127
6.2.2. Assesment of Predictive Power in other Frameworks . . . 128

Appendices 131

A. Project Management 131
A.1. Work Description . . . 131
A.1.1. Study . . . 131

A.1.2. Research 132
A.1.3. iMLP Matlab 132
A.1.4. Conclusion 132
A.2. Cost Analysis 133
A.2.1. Labor 133
A.2.2. Material 134
A.2.3. Total 135
B. Optimal Size for Hidden Layer: Number of Hidden Neurons 137
B.1. Structured trial and error method 137
B.2. Rule of Thumb Methods 138
Bibliography 138

List of Figures

1. Daily high values of S&P500 10
2. Daily closing values of S&P500 with underlying trend line 11
3. Daily change of the S&P500. Differencing removes trend but keeps seasonal components. 12
4. Comparison of statistical forecasting methods on Nasdaq Composite monthly data 14
5. IBEX 35 daily data. First plot shows original IBEX35 Daily data. Second plot shows series after first-order differentiation, removing downward trend and making the series stationary. 20
6. ACF and PACF plots of an AR(1) model. Notice how while the ACf is slowly decaying, the PACF drops after lag number 1 since its correlation is not accounted for on the following lags. 21
7. Apple Inc. monthly data. 24
8. ACF and PACF plots of the Log-Change of Apple Inc. monthly data. . . . 25
9. Forecast using the ARIMA(0,1,0) model on Apple Inc. 26
10. Histogram of the residuals of the forecasting process. 27
11. 1-NN and 99-NN feature spaces. 28
12. Similarity between ANN and human brain.**Source:**https://viblo.asia/p/overview-of-artificial-neural-networks-and-its-applications-ORNZqwQb50n. 33
13. Logistic function varying value of k. 35
14. Single-output SLP. 36
15. RBN schema to calculate wind-speed output. 38

16. Figure on the left (A) shows 3-D vector function. Figure on the right (B) shows 2-D representation of $-g$ values towards the minimum of the vector space. **Source:** www.researchgate.net 46
17. Multi-Layer Perceptron that will illustrate the training process. 50
18. Typical graph of the accuracy of both the training and the test sets of an **overfit** model. **Source:** https://deeplearning4j.org/earlystopping 53
19. Example of an ITS. Daily Range of the price of S&P500 index. 60
20. Three-dimensional representation of the extreme case of a non-convex parameter space. 88
21. Testing error comparison between optimizers SGD, Adam, Adam-Clip(1,∞) and Adam-Clip(0,1) from evaluation of DenseNet proposed in [41]. 89
22. FX market hours by time zone and global market. 99
23. Figure on the left shows the change (log) in the EUR/USD currency pair (actual) along with the model predictions. Figure on the right shows analogous results for the change in the USD/JPY. 101
24. Sample Partial Autocorrelation Function of the classic time series of the range of AUD/USD daily bid. Spike at lag 20 shows relevancy of information until that time lag. 102
25. iRMSE error metric against number of training epochs for AUD/USD daily bid price ITS. 104
26. Payoff and profit from an option buying or a long call option. 110
27. Dow Jones Industrial Average daily price. 111
28. Predicted ITS and Actual series for a 13-step-ahead forecast horizon of the AUD/USD Daily Bid price ITS. 126
29. Predicted ITS and Actual series for 90-step-ahead forecast horizon of the EUR/USD Daily Bid price ITS. 127
30. Predicted ITS and Actual series for 60-step-ahead forecast horizon of the IBEX 35 Daily price ITS. 128
31. Predicted ITS and Actual series for 70-step-ahead forecast horizon of FORD Motor Company Daily price ITS. 129

List of Tables

1. Comparison between learning paradigms in machine learning. 44
2. Layout of an ITS data file. 91
3. AUD/USD Daily Bid ITS. 105
4. EUR/USD Daily Bid ITS. 106
5. IBEX35 Daily Price ITS. 114
6. FORD Motor Company Daily ITS. 116
7. Validation-to-error percentage. 120
8. Stand-alone accuracy measures of the proposed method forecasts compared to the rival predictors. 122
9. Stand-alone accuracy measures of the proposed method forecasts compared to the rival predictors. 123
10. Summary of forecasting errors of the proposed method compared to benchmarks in terms of the U Statistic metric and the Average Relative Variance for interval time series. 124
11. Summary of forecasting errors of the proposed method compared to benchmarks in terms of the U Statistic metric and the Average Relative Variance for interval time series. 125
12. Horly breakdown of the project development. 133
13. Equipment-related costs of the project. 134
14. Total costs of the project development. 134

Chapter 1

Single-valued Data Forecasting Methods

1.1. Time Series

1.1.1. Definition

A time series is defined as a set of measurements of a single variable collected over equally-spaced time intervals. A key characteristic in time series is that ordering is very important. Since there is dependency within the data, changing it could change the whole meaning of the time series.

Time series are usually analyzed in order to describe important features of the data values such as correlations, past behavior or likelikehood to change when external causal factors interact. Finally, a model that describes the pattern of the time series can be determined and used for future-value forecasting or as a control standard for similar variables.

1.1.2. Decomposition: Trend, Seasonality and Residuals

As described in [WEI, 2006][86], time series have patterns hidden that can be identified by plotting the data. These patterns exhibit the particular behaviors of

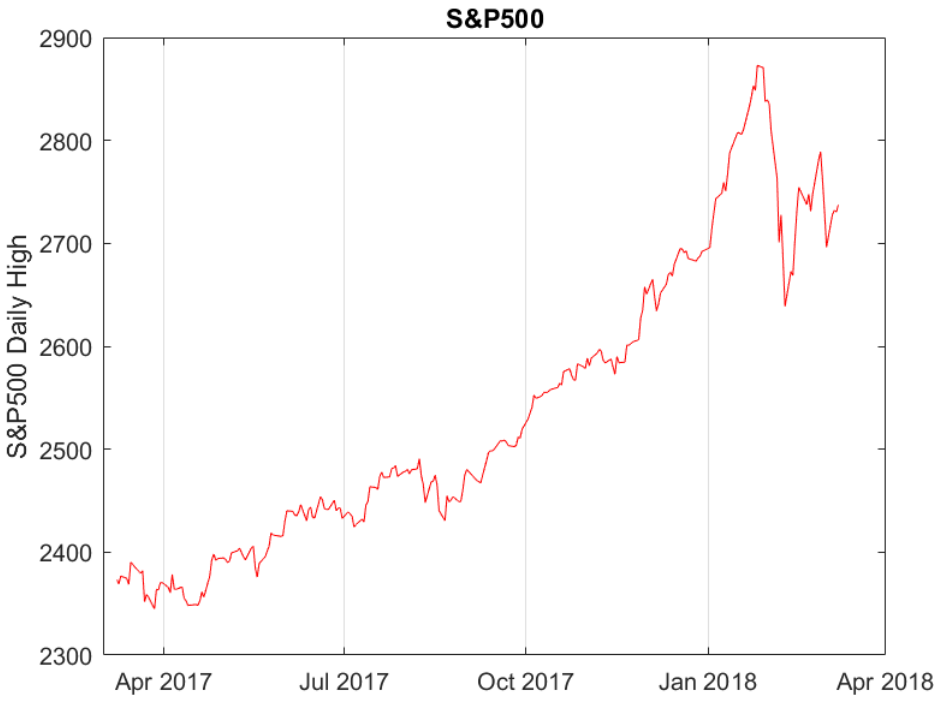


Figure 1. Daily high values of S&P500

the components of the time series. It is very useful to separate said components and to analyze them individually in order to understand the time series better.

Trend In a time series, a trend exists when there is a long-term increase or decrease in the data. This pattern can be both linear or non-linear and it might change its direction over time. The trend component tends to soften when the time period analyzed grows.

Seasonality The seasonal component is always associated to a fixed and known period of time within the series. Seasonality represents the influence of said period in the variable of interest.

Residuals Every real process contains an erratic component that is the main source of uncertainty in forecasting future behavior.

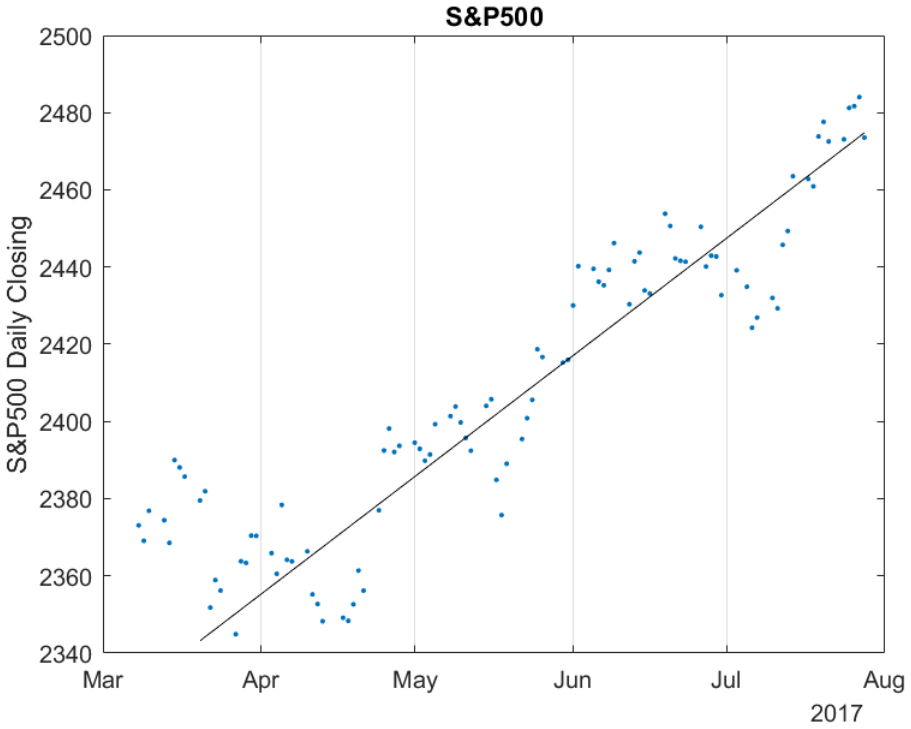


Figure 2. Daily closing values of S&P500 with underlying trend line

Altogether, a time series at period t, y_t, can be expressed as a combination of its underlying components:

$$y_t = f(T_t, S_t, R_t)$$

Where T_t represents the trend component and S_t and R_t the seasonal and residual, respectively.

If an additive model is adopted, then the time series can be modeled following:

$$y_t = T_t + S_t + R_t$$

however, other models are more appropriate if the time series shows that a component is stronger than the others. For instance, the multiplicative model $y_t = T_t \cdot S_t \cdot R_t$ is really useful for series in which the seasonal pattern varies with the level of the time series.

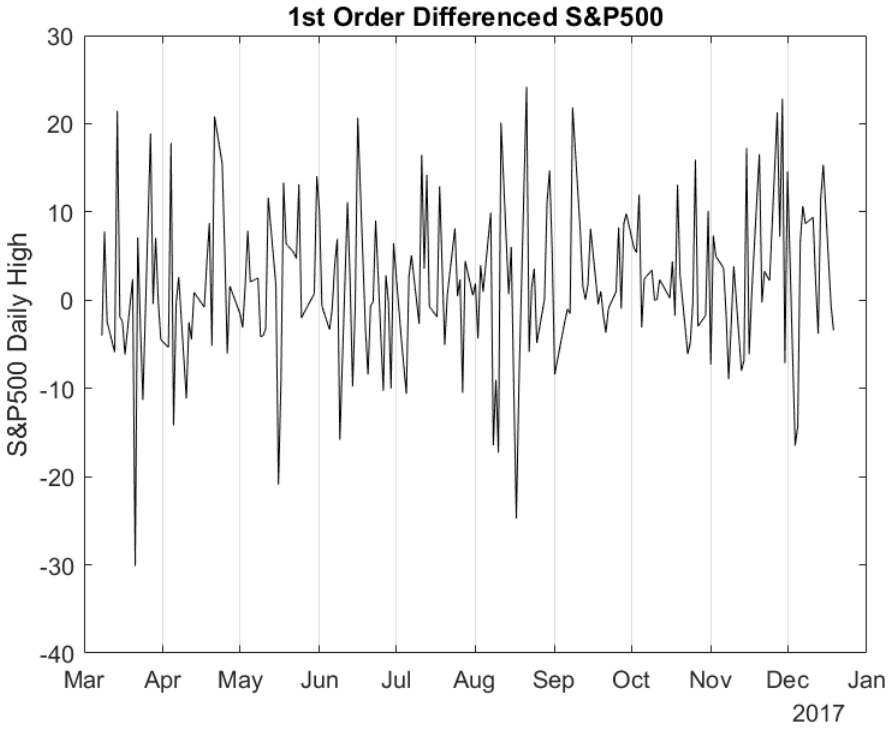


Figure 3. Daily change of the S&P500. Differencing removes trend but keeps seasonal components.

1.2. Introduction to Forecasting

1.2.1. Simple Forecasting Methods

Although forecasting using time series may sound complex, as illustrated in [WEI, 2013][87], there are some methods that are at the same time very simple and effective.

Average Method The average value of the available historical data is the reference for future forecasts. Let $y_1, ..., y_T$ be the available data set, every forecast will satisfy:

$$\hat{y}_{T+h|T} = \bar{y} = \frac{\sum_{i=1}^{n} y_i}{T} \tag{1.1}$$

This method is only appropriate for data showing no trend or seasonality since the average of a data set neglects their existence.

Naïve Method Any future forecast using the historical data set will be equal to the last observation in the set:

$$\hat{y}_{T+h|T} = y_T \tag{1.2}$$

This approach is particularly applicable for processes in which the forecast is most likely to be unrelated to the past data, as in a random walk process. Random walks will be covered in further detail in following sections.

Seasonal Naïve Method A similar approach is used when the historical data set shows a strong seasonal component. In this case, the forecast obtained will be equal to the last observation from the same season:

$$\hat{y}_{T+h|T} = y_{T+h-km} \tag{1.3}$$

Where m represents the seasonal period and $k = \left\lfloor \frac{h-1}{m} \right\rfloor + 1$. The notation $\lfloor argument \rfloor$ denotes the integer part of the argument.

Drift Method Another variation built on the naïve method adds the effect of the trend component. The forecasts will fall in a line between the first and the last observed values:

$$\hat{y}_{T+h|T} = y_T + h\frac{y_T - y_1}{T-1} \tag{1.4}$$

This method performs specially well in data sets showing strong linear trends since the extrapolation used is also linear.

1.2.2. Error Measures

No forecasting technique is capable of providing perfect predictions, so in order to analyze the accuracy of the forecast, an error measure has to be established.

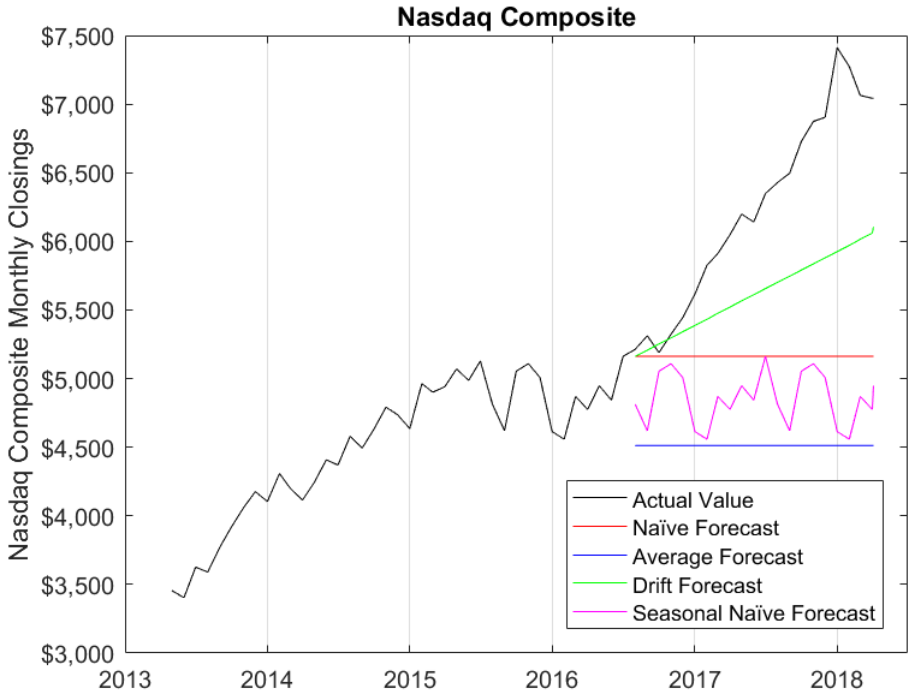


Figure 4. Comparison of statistical forecasting methods on Nasdaq Composite monthly data

Error measures are based on the concept of distance. For point-value data, the forecast error is simply the distance between the forecast and the actual value $e_i = y_i - \hat{y}_i$. The most common scale-dependent error measures as described in [MAKRIDAKIS, 1993][49] are:

Mean Absolute Error (MAE)

$$MAE = \frac{\sum_{i=1}^{T} |e_i|}{T} \tag{1.5}$$

Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{T} {e_i}^2}{T}} \tag{1.6}$$

This error measures, as said, depend on the scale of the data. Therefore, new measurement approaches will be introduced in order to perform comparisons between series of different scales and between different series.

Mean Absolute Percentage Error (MAPE) The percentage error is defined to solve scale-dependency, and it is calculated given by $p_i = \frac{e_i}{y_i}$. Based on this concept, the most used measure is the MAPE:

$$MAPE = \frac{\sum_{i=1}^{T} |p_i|}{T} \tag{1.7}$$

The usage of this approach is limited since it has three major drawbacks: data analyzed must have a meaningful zero, it yields extreme values when y_i approaches zero and , due to its formulation, it puts a heavier penalization on negative values of $\hat{y}_i$.

Symmetric Mean Absolute Percentage Error (sMAPE) In an attempt to solve the bias problem, [GOODWIN, LAWTON, 1999][29] proposed a variation of the MAPE. Its calculation follows:

$$sMAPE = \frac{1}{T} \sum_{i=1}^{T} \frac{2|y_i - \hat{y}_i|}{y_i + \hat{y}_i} \tag{1.8}$$

It still fails to correct the problem of extreme values as if y_i is close to zero, $\hat{y}_i$ is likely to also be close to zero.

An alternative to percentage errors when comparing data of series on different scales is the use of scaled errors. For time series data without a strong seasonal component, better results are obtained using naïve forecasts:

$$q_i = \frac{e_i}{\frac{1}{T-1} \cdot \sum_{j=2}^{T} |y_j - y_{j-1}|}$$

If the time series is seasonal, the scale error is better defined using seasonal naïve forecasts:

$$q_i = \frac{e_i}{\frac{1}{T-m} \cdot \sum_{j=m+1}^{T} |y_j - y_{j-m}|}$$

Mean Absolute Scaled Error (MASE) Based on this idea, the MASE is simply given by:

$$MASE = \frac{\sum_{i=1}^{T} |q_i|}{T} \tag{1.9}$$

1.3. Linear Forecasting Methods

1.3.1. Linear Regression

Simple Regression When analyzing time series data it is both common and accurate, as discussed in [BOWERMAN, O'CONNEL, 1990][14], to assume that the forecast and the predictor variables are related by a simple linear model such as $y = \beta_0 + \beta_1 x + \varepsilon$ which implies an underlying straight-line model, $\beta_0 + \beta_1 x_i$, combined with the deviation term ε_i.

Least Squares Estimation In order to determine coefficients β_k a fitting of the straight line through the data needs to be done. The procedure consists on the minimization of the sum of the squared error terms:

$$\sum_{i=1}^{n} {\varepsilon_i}^2 = \sum_{i=1}^{n} \Big(y_i - (\beta_0 + \beta_1 x_i) \Big)^2$$

The problem provides the following solutions for the coefficients:

$$\hat{\beta}_1 = \frac{\sigma(y,x)}{\sigma^2(x)} \quad , \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{1.10}$$

where $\bar{y}$ and $\bar{x}$ are the average values of the variables and σ and σ^2 denote standard deviation and variance.

Residuals and Fitting Accuracy Let $\hat{y}_i$ be the fitted value correspondent to observed value y_i, calculated by $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1$, then residual e_i will be given by:

$$e_i = y_i - \hat{y}_i$$

This new variable, by definition, should be random and uncorrelated to the predictor variable if the model under study is, in fact, linear. Plotting residuals against the predictor variable is actually a very useful way of checking for non-linear correlations.

The most common characteristics that a non-linear model may show are:

- **Heteroscedasticity**. Unequal variability of the forecast along the range of values of the predictor variable or non-constant variance of the residuals.
- **Non-linear trends**. The simple linear regression model, as stated before, assumes a linear underlying model and will not be accurate otherwise.

Coefficient of Determination A measure of how good is the performance of the model obtained can be handled by means of the coefficient of determination, R^2, defined in [OZER, D., 1985][63] as:

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{1.11}$$

The metric $R^2 \in [0, 1]$ and the model obtained will be more accurate the closer R^2 is to the unit.

1.3.2. Exponential Smoothing

Simple Exponential Smoothing Exponential smoothing methods use weighted averages to impute importance to observations based on their age as introduced in [HOLT, 1957][34] later revised by [GARDNER, 1985][27]. The models associate exponentially decaying weighted averages to observations, from newest to oldest, so more importance is given to the latest available data.

Level Component In SES models, as discussed in [BROWN, 1959][17] only the level component, ℓ_t, is considered when formulating forecasts. The smoothing equation that defines ℓ_t follows a weighted average between the latest observation y_t and its corresponding forecast ℓ_{t-1}:

$$\ell_t = \alpha y_t + (1 - \alpha) \cdot \ell_{t-1}$$

Hence, the forecast is given by:

$$\hat{y}_{T+1|T} = \ell_t \tag{1.12}$$

Error Correction Form Retrieving the definition of the error of a forecast used in previous sections. Considering that at time t, $\ell_{t-1} = \hat{y}_t$ and rearranging the weighted average form:

$$\hat{y}_{t+1|t} = \ell_t + \alpha \cdot e_{t-1} \tag{1.13}$$

The level of adjustment of the forecast to preceding errors will grow as the value of α approaches 1. The optimum value of α is such that error is minimum. This is chosen by iteration using measures like MAPE and MSE.

Holt's Linear Trend Method For series with relevant trend components, an extra smoothing equation is proposed so equation 1.12 is rewritten as follows:

$$\hat{y}_{T+1|T} = \ell_t + hb_t \tag{1.14}$$

Where

$$b_t = \beta^* \cdot (\ell_t - \ell_{t-1}) + (1 - \beta^*) \cdot b_{t-1}$$

The level component used in SES is still present, and b_t denotes an estimate of the trend of the series at the observed time. The coefficient h makes the forecast a linear function with the number of periods of the forecast.

Other methods to incorporate trend in the forecast are the Exponential and Damped trend methods, which will not be analyzed here in detail.

Seasonal Component In order to incorporate seasonality into the model, [WINTERS,1960][89] proposes a modification of Holt's previous method, adding a third smoothing equation to the forecast. Hence, a new smoothing parameter, s_t is defined as:

$$s_t = \gamma^* \cdot (y_t - \ell_t) + (1 - \gamma^*) \cdot s_{t-m}$$

Where m acquaints for the season periodicity. This third equation yields the new forecast equation

$$\hat{y}_{T+1|T} = \ell_t + hb_t + s_{t-m+h^+_m} \tag{1.15}$$

Where $h^+{}_m = \lfloor (h-1) \mod m \rfloor + 1$ ensures the usage of the final year's data to define the seasonal indices.

1.4. Autoregressive Moving Average Methods

1.4.1. Background Concepts

A Random Walk model A random walk is used to model stochastic processes in which every observation is random and non-dependent of the previous observations, this is, a variable with zero mean and unitary variance. It can be described following the first definition proposed in [WORKING, 1934][90]:

$$y_t = y_{t-1} + e_t \tag{1.16}$$

Where e_t is white noise[1].

Stationarity As introduced in [DICKEY, FULLER, 1976][23] for a series to be stationary, its observations must be independent of the time of the observation. Thus, stationarity implies difficult long-term predictions.

Differencing One way to make a time series stationary is to use differentiation, this is, creating a new series formed by the difference between consecutive observations. Note that if the resulting series is still not stationary (i.e. its variance increases with time) other transformations as taking logarithms of the series might be necessary.

1.4.2. ACF and PACF Models

As discussed in [BOX, PIERCE, 1968][16] In order to determine the order q of the adequate Moving Average model, graphic analysis of the Autocorrelation Function (ACF) and the Partial Autocorrelation Function (PACF) are very convenient.

[1]White noise in descrete time is any random signal whose samples are uncorrelated and that has zero mean and finite variance.

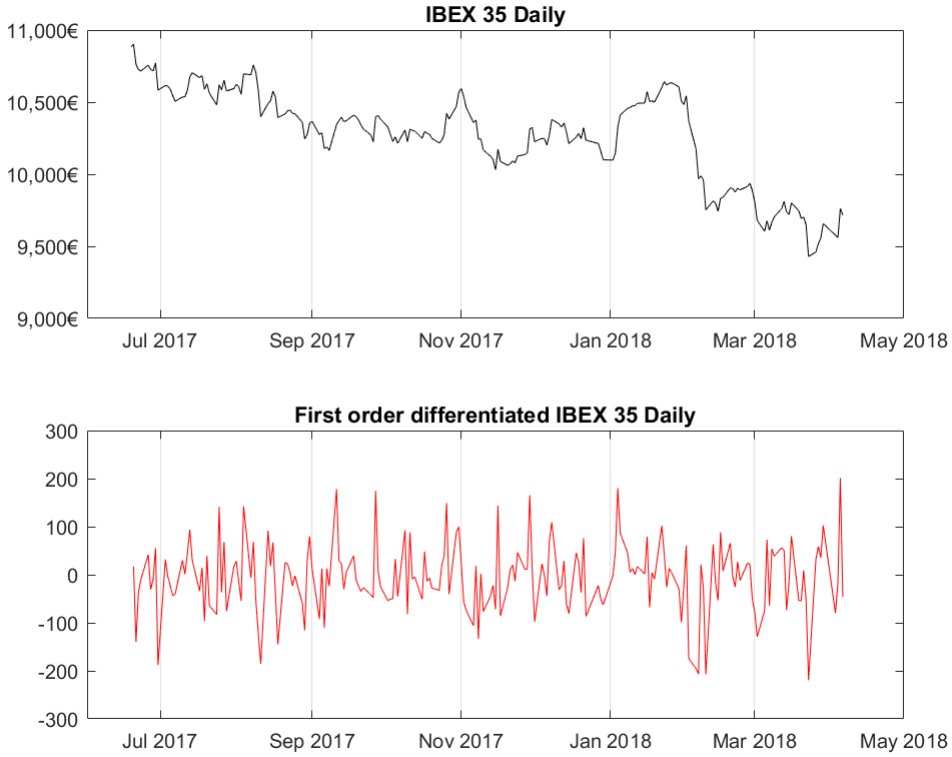


Figure 5. IBEX 35 daily data. First plot shows original IBEX35 Daily data. Second plot shows series after first-order differentiation, removing downward trend and making the series stationary.

Before diving into the details of said analysis, we shall understand both functions from a mathematical point of view.

$$ACF = \frac{\sum_{i=j+1}^{T} (y_i - \bar{y})(y_{i-j} - \bar{y})}{\sum_{i=1}^{T} (y_i - \bar{y})^2} \tag{1.17}$$

As stated in the equation, the ACF is a measure of the memory of the process analyzed at different time lags as noted in [ULLA, GILES, 2011][85]. But the ACF lacks information about the autocorrelation between non-consecutive observations, for instance, x_t and x_{t+h} if $h \geq 2$.

To fulfill this task, the PACF has been defined as the correlation between observations at times t and $t+h$ discounting the correlation between the intermediate time-steps.

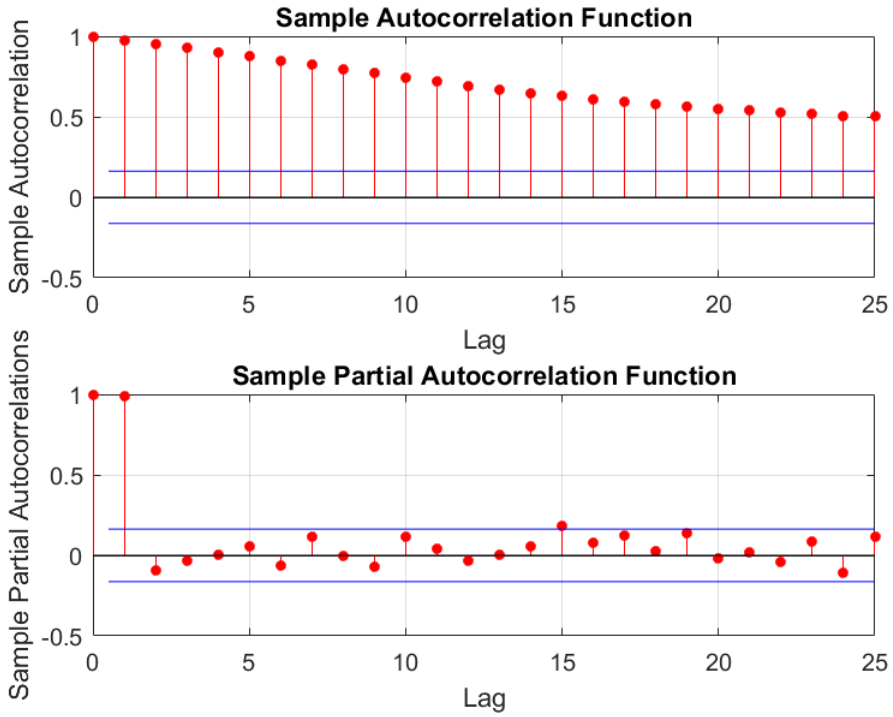


Figure 6. ACF and PACF plots of an AR(1) model. Notice how while the ACf is slowly decaying, the PACF drops after lag number 1 since its correlation is not accounted for on the following lags.

1.4.3. Non-seasonal ARIMA models

Some processes are better resembled by AutoRegressive models, others may need Moving Average calculations to reproduce the series and some stochastic series may require of Differentiation techniques for the adequate transformation of the series.

Actually, most real-life measures need the combination of the three to be able to resemble the behavior, this is an AutoRegressive Integrated Moving Average model

(ARIMA). In this context, "Integration" refers to the reversal of the Differentiation applied to the series.

The full model can be identified as $ARIMA(p, d, q)$ where p is the order of the underlying $AR(p)$ part, d is the order of the differencing the series has and q represents the order of the $MA(q)$.

Backshift Notation Since ARIMA models rely on time series lags, it is very useful to introduce the backshift operator B to obtain more compact equations. For instance, shifting back one period within a time series can be expressed following:

$$y_{t-1} = By_t$$

The operator can be applied n times to produce lag-shifts of n periods:

$$y_{t-n} = B^n \cdot y_t$$

The operator is also useful when denoting differentiation, namely, the d^{th} order differentiation of a series can be put as:

$$y_t^{(d)} = (1 - B)^d \cdot y_t$$

ARIMA Model Having defined backshift notation, it is clever to draw upon the B operator to identify ARIMA models, for example, the ARIMA(p, d, q) follows

$$y_t^{(d)} = y_t^{(d)} \sum_{i=1}^{p} \phi_i \cdot B^i + e_t \sum_{i=1}^{q} \theta_i \cdot B^i + e_t$$

and if the backshift-operator transformation to denote differentiation is applied, the equation yields:

$$e_t \cdot \left(1 - \sum_{i=1}^{q} \theta_i \cdot B^i\right) = (1 - B)^d \cdot y_t \cdot \left(1 - \sum_{i=1}^{p} \phi_i \cdot B^i\right) \tag{1.18}$$

1.4.4. Seasonal ARIMA

If a time series shows a strong seasonal pattern (i.e. electricity demand), a rearrangement of the aforesaid ARIMA model can be made to incorporate the seasonality and yield more accurate results when forecasting.

Let S be the time span between observation that are homologous within a seasonal pattern, the ARIMA(p,d,q)x$(P,D,Q)S$ model will predict $\hat{y}_t$ based on time lags that are multiples of S. The terms P, D, Q refer to the seasonal terms of the AR(P), MA(Q) and the seasonal order of differentiation D.

Example Now that the structure of seasonal and non-seasonal ARIMA models is clear, it is opportune to describe how the order of the terms is obtained and adjusted. For this purpose it is crucial to understand the ACF and PACF plots that have been introduced earlier in this chapter.

To better demonstrate the order identification procedure, a step-by-step example on the fitting of an ARIMA(p,d,q) model on the monthly opening prices of Apple Inc will be carried out:

1. Time Series Pre-processing

The starting point of the fitting procedure is to establish the order of differentiation to make the TS stationary. In this case, as Figure 7 shows, a first-order differentiation created a zero-mean time series with increasing variance (non-stationary). Thus, a logarithmic transformation was required in order to get the stationary series.

Normally, the amount of differentiating required matches the time-lag in which the PACF plot decays rapidly to zero.

2. Determination of p and q.

Parameter selection for the AR and the MA terms of the model can be done analytically (AIC and BIC tests) as described in [AKAIKE, 1998][3] and [SCHWARZ, 1978][76] or graphically, inspecting the ACF and PACF plots of the stationary series. The latter will be used in this example to fit the ARMA model to the data:

- data will follow an ARMA$(p,0)$ model if ACF plot is exponentially decaying or sinusoidal and there is a significant spike at lag p in PACF but none beyond it.
- data will follow an ARMA$(0,q)$ model if PACF plot is exponentially decaying or sinusoidal and there is a significant spike at lag q in ACF but none beyond it.

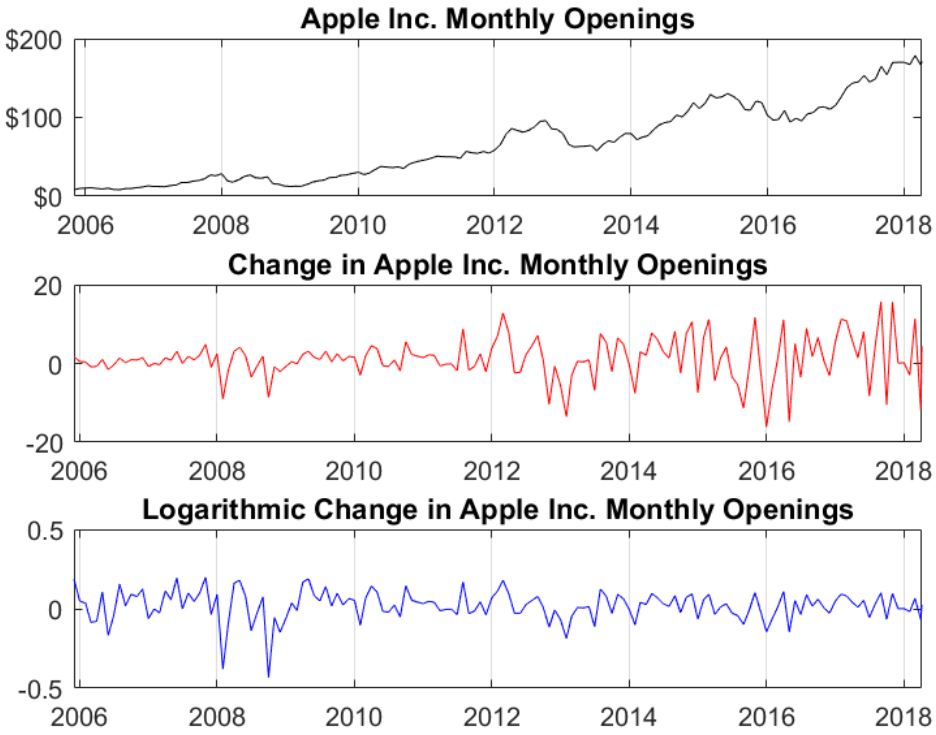


Figure 7. Apple Inc. monthly data.

Since both the ACF and PACF plots in Figure 8 show no significant correlations ($|ACF| > 0.5$), an ARMA$(0, 0)$ model will be selected, resulting on an ARIMA$(0, 1, 0)$, also knowing as a random walk with drift.

3. Parameter Estimation

Once the model structure is chosen, the parameter estimation can be easily done using automated `MATLAB` functions like `arima{}` that will create a forecasting-ready model structure. For this series, the model fit yielded the equation:

$$\hat{y}_t = y_{t-1} + 0.640357 + e_t$$

which implies an upward trend ($c > 0$) and an stochastic trend.

4. Forecasting

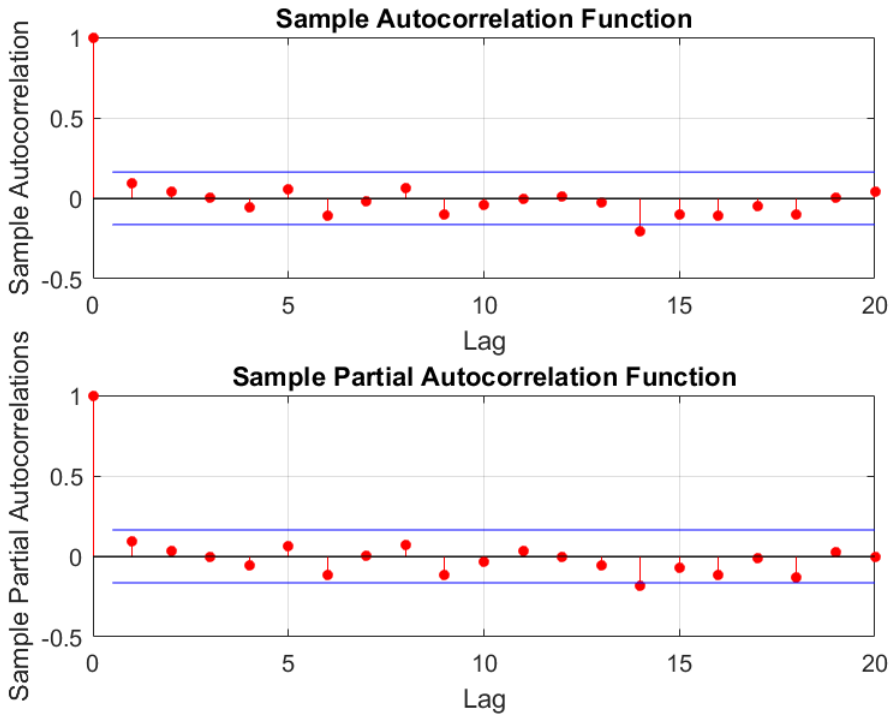


Figure 8. ACF and PACF plots of the Log-Change of Apple Inc. monthly data.

After completing the model-fitting procedure, forecasts for future time steps can be easily obtained using the `predict{}` function. Results shown in Figure 9 denote a success on the fitting of the series linear trend but fail to explain the stochastic underlying process.

5. Residual Analysis

After creating the new series made up of forecasts, an error computing is convenient to get conclusions from the results. The analysis of the distribution of the residual series $R_t = y_t - \hat{y}_t$ carried out on Figure 10 shows that the errors the model makes tend to be positively distributed. In conclusion, the model can and should be optimized in order to produce zero-mean residuals.

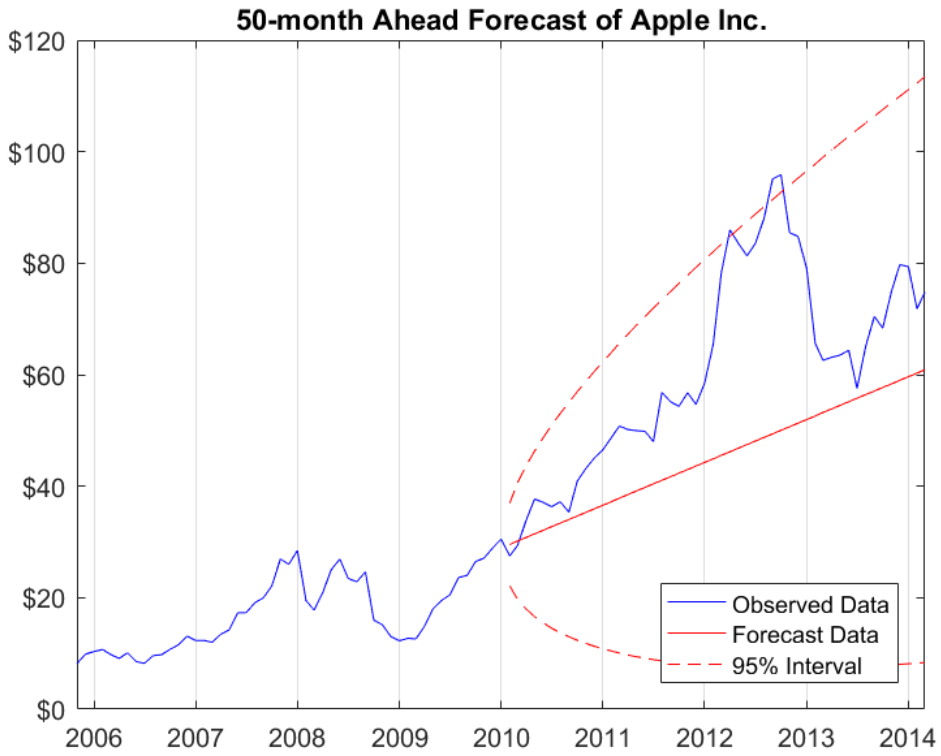


Figure 9. Forecast using the ARIMA(0,1,0) model on Apple Inc.

1.5. Non-linear Methods

1.5.1. Non-linear Regression

Considering the concepts explained in Section 1.3.1, if the process under study where to have non-linear relationships between its variables a Non-Linear Regression approach would be a better option to produce forecasts.

Let x_t be the predictor variable (will use only one for simplicity) of the process explained by y_t, then the regression equation will be

$$y_t = f(x_t) + e_t$$

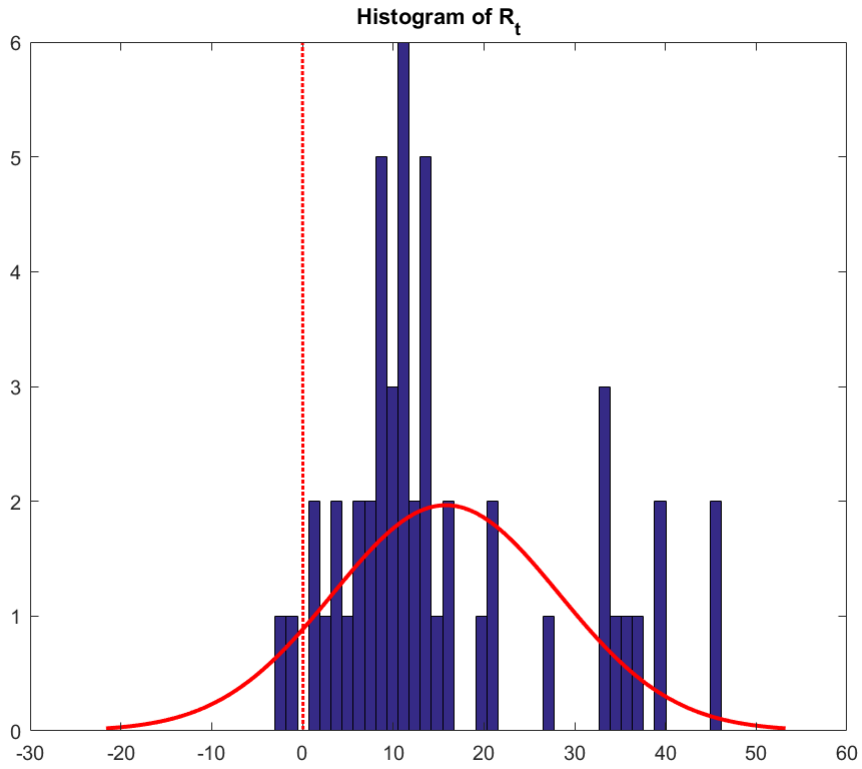


Figure 10. Histogram of the residuals of the forecasting process.

with f being a non-linear function (quatradic, exponential, logarithmic, etc.). As described in [HARTLEY, 1961][30] the problem may be solve, analogously to a linear regression problem, by meas of a fitting through Least Squares estimation.

1.5.2. k-NN Method

As introduced in [YAKOWITZ, 1987][91], k-Nearest Neighbors is a non-parametric algorithm that establishes a set of classes based on observed data and yields future values based on proximity to a number of k observations that are most similar to the input, the k-NN. Thus, it is a distance-based model that can be used both for classification and regression problems.

The algorithm follows a two-step procedure:

1. For a given input x it searches through the feature space of the training set, calculating the distance between each pair of observations, to find the k closest instances.

2. It yields a prediction based on a combination of said k neighbors.

Selection of k The selection of the number of neighbors that will be considered in the calculations is arbitrary, but may be optimized via *trial-and-error* techniques. A higher k will produce slower computation times and smoother predictions. However, overfitting is extremely frequent and it can produce a biased model as shown in Figure 11

Figure 11. 1-NN and 99-NN feature spaces.

Distance Measure In order to rank the observations during the search, a distance calculation is computed. The most common metric used for point data in k-NN is the Euclidean distance, given by:

$$dist(x, y_i) = \sqrt{\sum_{i=1}^{k} (x - y_i)^2} \tag{1.19}$$

Forecasting The forecast values are given by an combination of the selected k-neighbors using either the mean for regression problems or the mode of the observations for classification.

For more precise forecasts, [COST,SALZBERG, 1993][21] propose an assignment of a set of normalized weights w_i. Usually, the inverse of the distance is used to calculate the weights

$$w_i = \frac{1}{|dist(x, x_i)| + \epsilon} \quad for \quad 1 \leq i \leq k$$

where ϵ is a small constant that prevents extreme values of w_i when distance between observations approaches zero.

Thus, predicted values are yielded following:

$$\hat{y}_t = \frac{\sum_{i=1}^{k} w_i \cdot f(y_i)}{\sum_{i=1}^{k} w_i} \tag{1.20}$$

1.5.3. Neural Networks

Forecasting methods that rely on Artificial Neural Network structures will be explained deeply in Chapter 4.

Chapter 2

Artificial Neural Networks Application in Forecasting

Real world processes are built on numerous subtle factors that, due to their nature, are very difficult to replicate using a rigid algorithm. Moreover, the recent spread on Big Data structures, as discussed in [MANYIKA et al., 2011][50] has made the task far more demanding since the number of variables that are considered in predictive analysis has been multiplied hundred-folds. Thus, as adaptive models that allow dynamical learning are necessary to provide solutions to current modeling problems.

[HAYKIN, 1994][33] defined ANN as a massively parallel combination of simple processing units which can acquire knowledge from environment through a learning process and store the knowledge in its internal connections, whereas most researchers explain the behavior comparing it to that of the human brain as shown in [ARBIB, 1989][6], later restated in [HARVEY, 1994][31]. This latter definition will probably be more accurate in the context of *Deep Learning* algorithms but, in *Machine Learning*, the former is more realistic.

In the end, the so-called learning of a ANN is nothing more than the adjustment of the parameters that build the mathematical model with the objective to be able to provide accurate and general solutions to a certain paradigm as will be discussed later this chapter. For now, let's introduce the main *ML* problem archetypes.

Classification The task of producing a mapping function f that for a given input yields a discrete output value (usually a categorical value) in machine learning is called classification, it is a supervised learning technique widely used in pattern recognition problems like Natural Language Processing.

Clustering The grouping of input unlabeled data in clusters of similar values is the second widespread type of machine learning problems. This unsupervised learning procedure creates sets of observations that are similar between them and different to those of other clusters.

Prediction Producing future time-horizon values out of the adaption of a ANN model to training data is the last main field of machine learning. This document will analyze, specially, the benefits of the data-driven and self-adapting ANN structures in forecasting.

2.1. Definition

The first important fact about ANN is that there is not only "one" neural network but a broad range of different structures that share some common characteristics. These features have the peculiarity to be, in some way or another, attempts to replicate behaviors that biological species use in their learning processes.

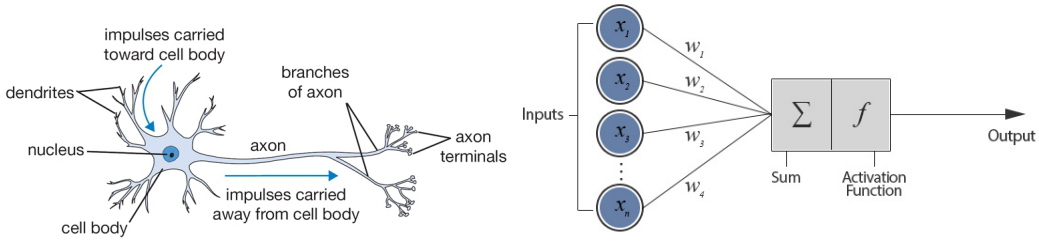
One could dwell on these learning procedures and the complete range similarities between artificial and biological learning units but we will stick to what are the most critical ones by means of distinguishing ANN from classical algorithms.

Learning Capability The first and most obvious required feature is the ability to learn. As opposed to classical algorithms that needed a rigid program to function, ANN are "intelligent" enough to teach themselves when a training input is fed to them, or even better, they are capable of adapting to rewards/punishments like little children do and assimilate new knowledge.

Generalization Capability The goal of a ANN is to be learn from a specific problem example and then be able to generate reasonable solutions to different challenges of the same *class*.

Fault Tolerance The same way a poor nutrition decreases the number of brain cells in a human but allows them, at least for a moderate period of time, to maintain the organism fully-functioning, ANN need to be *fault-tolerant* to external stimuli such as the input of noisy signals.

But these human-like capabilities are not the only resemblance of human nature in ANN. Their simplest *processing unit* are as well referred to as *neurons*. These units process information and transforms it ready to be passed on to the next unit. The *neurons* also have an *excitation state* as human neurons do.



(a) Human brain neuron **(b)** ANN Perceptron

Figure 12. Similarity between ANN and human brain.**Source:**https://viblo.asia/p/overview-of-artificial-neural-networks-and-its-applications-ORNZqwQb50n.

2.2. Components of an ANN

The following description of the components of an Artificial Neural Network is inspired by [KRIESEL, 2007][43] Let a neural network be constituted of two sets (N, V) and a function g, where N is the set of neurons that form the net, V is a sorted set of pairs of integers (i, j) that refer to connections (weights) between neurons. For simplicity, we will refer to the connection from neuron i to j as $w_{i,j}$.

2.2.1. Connections

The aforementioned weights between pairs of neurons. These numeric values either amplify ($w_{i,j} > 1$) or dim the input signal ($0 \leq w_{i,j} \leq 1$).

2.2.2. The Propagation Function

Let j be a neuron of the network and o_{i_n} the set of output signals from the n preceding connected neurons i_n. Then the propagation function of neuron j is defined following:

$$f_{prop} = net_j = \sum_{i=1}^{n} (o_i \cdot w_{i,j}) \tag{2.1}$$

2.2.3. Network Input

As stated before, network input of neuron j is the result of the transformation of the propagation function. Strictly, this is not the *input* of the network itself but that of the activation function.

2.2.4. Threshold Value

Picking back the human brain example, the neurons of the network must always be *excited* to a certain extent because of f_{prop}.

However, it is necessary to define an activation state such that the neuron will output different values according to a predefined *threshold value*, Θ_j. For instance if a threshold value is set to 0.5, the neuron will output 1 when $net_j \geq 0.5$ and 0 otherwise.

2.2.5. The Activation Function

The afore-mentioned *activation state* is established by the activation function, a_j or shortly ***activation***. Formally,

$$a_j(t) = f_{act}(net_j(t), a_j(t-1), \Theta_j) \tag{2.2}$$

where dependence of time is included because previous activation state $(a_j(t-1))$ is taken into account in the current activation.

Binary Also referred to as *Heaviside function*, the ***binary activation*** is simply

$$a_j(t) = \begin{cases} 1 & \text{if } net_j(t) \geq \Theta_j \\ 0 & \text{otherwise} \end{cases}$$

Continuous activation For some types of learning procedures, differentiability of activation function is required and thus the *binary function* is not useful.

Thus, other commonly used functions are the *hyperbolic tangent* $(tanh(x))$ and the *logistic/sigmoid function* :

$$a_j(t) = \frac{1}{1 + e^{-kx}} \tag{2.3}$$

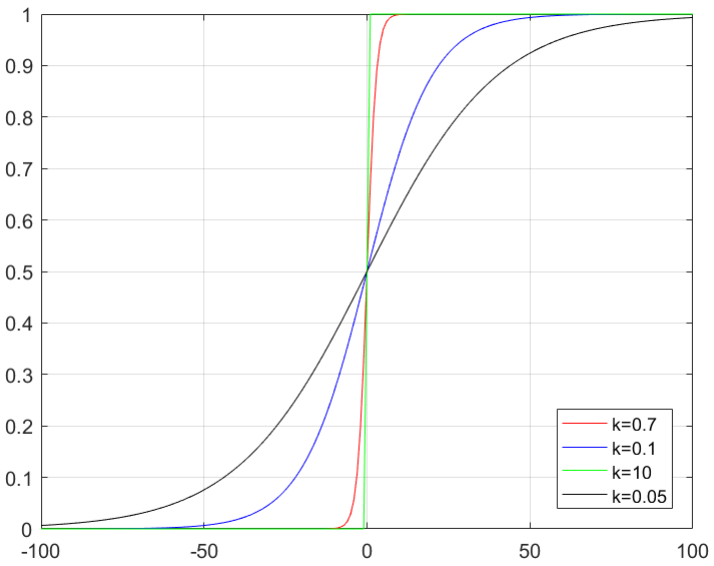


Figure 13. Logistic function varying value of k.

Where k is the rate of growth that adjusts the slope of the function.

2.2.6. Output function

Briefly introduced earlier this section, the *output function* of a neuron, f_{out} is the transformation that calculates the output signals o_j from the *activation state*. Usually the *identity* function is used as f_{out}, yielding:

$$a_j = o_j$$

2.3. Types of ANN

2.3.1. Single-Layer Perceptron

The most elemental feed-forward neural network is the generalized version of a single neuron as stated by [RAUDYS, 1998][68], the **SLP** is a perceptron with only one layer of trainable connections. This type of ANN is not capable of assimilating information in non-linearly separable problems, as was proved in [MINSKY, PAPERT, 1969][54].

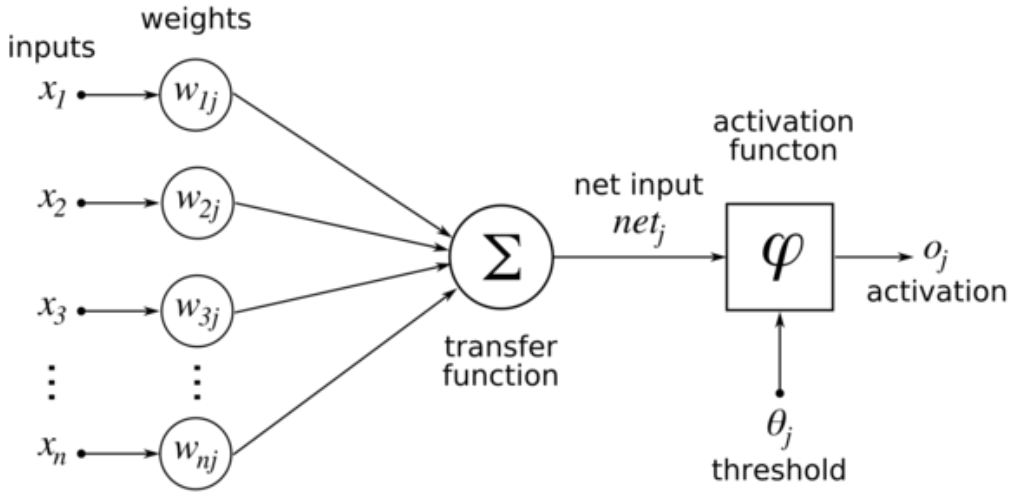


Figure 14. Single-output SLP.

The number of output neurons is arbitrary and should be chosen by trial-and-error iterative processes.

2.3.2. Multi Layer Perceptron

[RUCK, ROGERS, KABRISKY, OXLEY et al., 1990][75] described how, if two (or more) layers of weights are introduced in the network structure of the SLP, a new neural network archetype is created: the **MLP**. Then the *extra* layers, the ones that are not the input and output, are referred to as *hidden layers*.

Adding an extra layer provides the model with the capability of creating convex polygons to divide the input space, thus making the MLP an universal function approximator. Formally, a n-layer perceptron has n variable weight layers and $n+1$ neuron layers.

Algorithm The algorithm behind the learning and weight-adjustment of the *multi-layer perceptron* will be explained in detail later this chapter.

2.3.3. Radial Basis Network (RBN)

Another type of neural network schema is the one defined in [ADELI, HUNG, 1995][1] and later extended in [ADELI, KARIM, 2000][2],[KARIM, ADELI, 2003][40] as **Radial-Basis NN**. This time, instead of approximating functions by means of polygonal space-divisions, the network performs an aggregation of stretched/compressed *Gaussian bells*.

Developed appreciably later than perceptrons, this form of feed-forward network has exactly three layers of neurons, i.e. only a hidden stage of nodes which is, indeed, where RBN and MLP mostly differ. In fact, the task of the *output layer* is simply adding the values originated in the hidden nodes together and transferring the sum.

Algorithm Let a **RBN** of 5 layers, input $I = 1$, hidden $H = 3$ and output $O = 1$ be an example of the computation process.

Allow neuron $h = 1$ in the hidden-layer to be characterized by a two-dimensional *Gaussian bell* with center c_1, and width σ_1. The activation of said neuron is most sensible to the position of the input vector x, relative to its center, c_1, being more

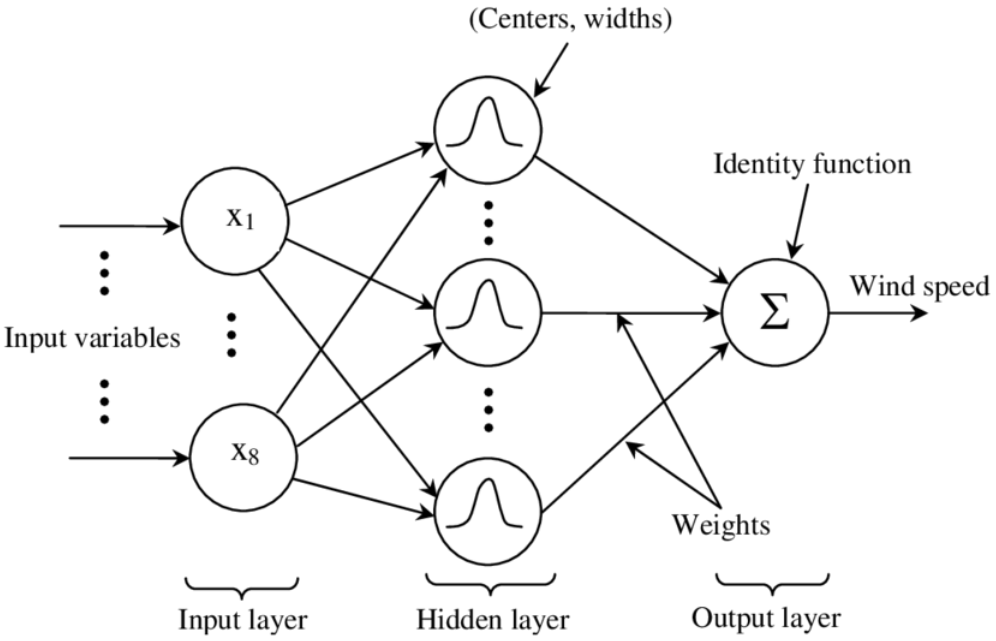


Figure 15. RBN schema to calculate wind-speed output.

excited the smaller the *norm* $r_1 = \| x - c_1 \|$ of x is. It is frequent to use the *Euclidean distance* as metric, yielding:

$$r_1 = \sqrt{\sum_{i=1}^{n} (x_i - c_1)^2}$$

Where n is the number of values in the input vector.

The activation function for the neuron processes the distance calculate by means of the *Gaussian bell*:

$$f_{act,1} = e^{\frac{-{r_1}^2}{2{\sigma_1}^2}} \tag{2.4}$$

Once the output of the f_{act} is calculated, it is passed through via a weighted connection $w_{i,o}$ to the output neuron o, where the inputs from the other two hidden nodes will be aggregated following:

$$y_o = \sum_{h \in H} w_{h,o} \cdot f_{act}(r_h) \tag{2.5}$$

Resulting in y_o, the output signal of the network.

2.3.4. Recurrent Neural Network

The structure of a **Recurrent Neural Network** (henceforth RNN) was first unveiled by John Hopfield in [HOPFIELD, 1982][35] and it is very similar to that of a *feed-forward* network such as the MLP, with the only distinction that RNN have special nodes that are able to store values to feed the network in successive time-steps. These characteristic neurons are referred to as *recurrents*.

2.3.5. Hopfield Network

With a name given by its developer John Hopfield in 1982, **Hopfield Networks** are another form of *unsupervised learning* in recurrent neural networks.

Its structure is formed of a set K of fully-linked neurons without direct recurrence, making for a unique architecture where there is no differentiation among *input, hidden* and *output* nodes. This schema can be seen as a square symmetrical matrix of the $K = I \cdot J$ weights of the connections:

$$W_{i,j} = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,k} \\ w_{2,1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ w_{k,1} & \dots & \dots & w_{k,k} \end{bmatrix}$$

Introducing initial conditions of no direct recurrences and full-linking between neurons results:

$$W_{i,j} = \begin{bmatrix} 0 & w_{1,2} & \dots & w_{1,k} \\ w_{1,2} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ w_{1,k} & \dots & \dots & 0 \end{bmatrix} \tag{2.6}$$

Algorithm For a given set of *training patterns* $p \in P$, there would be p training *stages* to complete the network's learning process. Each stage will then be comprised of $t = 1, 2, ..., k$ time-steps with $t = k$ being named as *time of convergence*. For the

given set, weight values are computed using the pair $[p_i, p_j]$, corresponding to the states of the directly-connected neurons by means of:

$$w_{i,j} = \sum_{p \in P} p_i \cdot p_j \tag{2.7}$$

Once the weight matrix, W is initialized, the training will consist of a *shuffling* process in which every neuron will be selected at random and its *change* in state is calculated will be calculated individually by means of the influenced of the other j neurons:

$$x_k(t) = f_{act}\left(\sum_{j \in K} w_{j,k} \cdot x_j(t-1)\right) \tag{2.8}$$

f_{act} is, in this case, the binary threshold function. The process will stop when *convergence* is met; every neuron is updated and no change in their state is required. Then output states of the training are stored as $y \in [-1, 1]^K$ and the network is prepare to process new data.

Let a feed-forward network with three hidden nodes ($H = 3$) and two output neurons ($O = 2$) be the focus of our example. To make it *recurrent*, there would be two different approaches:

- **Jordan Networks** To transform the original feed-forward network into what was envisioned in [JORDAN, 1997][39] as a *Jordan* network, a set of $K = 0 = 2$ *context* neurons should be inserted. These extra nodes take the network's output values at time t through weighted connections and feed them to the input layer at time $t + 1$ via *fully-linked* connections (non-weighted).

- **Elman Networks** Another similar approach is the one introduced in [ELMAN, 1990][25] by inserting a set of $K = O + H = 5$ *context* nodes, so every processing unit's output at time-step t will be back-fed into the same neuron at the next iteration, $t + 1$. Input neurons are excluded from the recurrence and only send the information forwards.

2.4. Training of a ANN

One of the most powerful applications of Artificial Neural Network structures is extracting generalized solutions from a set of unseen inputs. For these outputs to be precise, the architecture must be **trained** first as discussed in [ANDERSON, 1972][5].

Bearing in mind the structural components of a ANN explained earlier this chapter, the training process can be carried out by adjusting any of the following characteristics:

- Adding new connections.
- Updating weight of existing connections. If a weight is set to 0 it is equivalent to removing the connection.
- Changing the threshold value of neurons.
- Modifying neuron functions (activation, propagation or output).

Out of the four options the most extended, as proved in [ROSENBLATT, 1962][73] is the update of weights. This procedure also enables connection removal and change in threshold values (the latter only if a *bias neuron* is included in the network structure). So for the purpose of this chapter, the training processed will be focused on **weight updates**.

There are several training *paradigms* to ensemble an ANN. Let P be the dataset that will be used during the *training process*. Each paradigm involves specific approaches as to how to manipulate the dataset in order to complete the teaching of the network. The training paradigm are generally grouped under the following categories.

2.4.1. Supervised Learning

Supervised learning may easily be the most intuitive of the learning criteria. The set P includes, as well as the input patterns, their corresponding correct results

in the form of output signals. Therefore, for every new input data, a **SL** network is able to compute the predicted result and automatically compare it to the truth. Thus, *prediction error* is calculated directly and it is used to adjust the weights of the network.

For this training to be executed, the set must be completed, meaning that for every input the system needs the correct output in order to improve performance. This *paradigm* presents the downside of requiring a complete dataset which is not always feasible, but its execution is rather simple.

An example of this paradigm could be the teaching of a machine to play chess. The training set for this task would consist in a set of historical data of, for instance, a number of games played by a professional player. The set should contain for every possible sequence of moves of both the player and the opponent,an outcome in the form of *win/lose*. Thus, the system's ability to play chess would be limited to that of the player used for the training.

2.4.2. Reinforcement Learning

In *reinforcement learning*, as introduced in [SUTTON, BARTO, 1998][79], the procedure may seem similar to that of SL but the differences are quite remarkable. The core of the contrast relies on the idea of *intermediate states* and the *reward function*.

As opposed to the decisions the system makes in SL, which are independent from one another, every new input in **RL** creates a state for which the training set has a reference in the form of a *reward function* that allows the state to be *rewarded* or *punished*. The system, based on the outcome of said function, is self-adjusted to improve performance for the next state.

Reusing the previous example of the game of chess, in this case, for every move the player makes from the starting point the system would receive a reward/punishment and make the next decision seeking to maximize the cumulative reward. This procedure is useful when the task to perform is unknown but once it is completed it is easy to judge if it was done correctly.

2.4.3. Unsupervised Learning

The ultimate form of learning is *unsupervised learning*, which is essentially an autonomous learning paradigm. For the execution of the training the only required data is the input set. The system will need to be able to extract patterns from the input data so that it internalizes the underlying similarities in order to classify and produce outputs.

This learning procedure is almost exclusively used in clustering problems and requires a large set of data.

It is important to bear in mind that not every paradigm is applicable to every machine learning problem, and the selection among them may be crucial in order to obtain the desired results.

Table 1. Comparison between learning paradigms in machine learning.

	Supervised Learning	Reinforcemente Learning	Unsupervised Learning
Training Set	Input/Output Data	Terminal/Intermediate States and Reward Function	Input Data
Advantages	Exploitation ability[2]	Exploration ability[3]	Little "apriori" design
	Control over learning	Generalization	Generalization
	Training done once	Non-capped performance	
Disadvantages	Requirement of complete set	Design complexity	Uncontrolled classification
	Tendence to overfitting	Continuous learning	Costly training until good performance
	Computation time	(requires online training)	

[2]Exploitation is the ability to extract patterns from given information.
[3]Exploration is the capability of considering all the plausible scenarios given a certain situation.

Now that the main learning-paradigm groupings have been introduced, it is appropriate to go into deeper detail on how some of the most-extended learning algorithms operate.

2.4.4. Gradient Descent and Ascent

Most *supervised learning* machines rely on the *backpropagation of error* as a way to implement the learning into the network and perform weight-adjustment, as depicted in [PINEDA, 1987][65] and [WERBOS, 1988][88]. The mathematical basis under the procedure is no other that the **gradient descent**.

Gradient Prior to diving into the details of the method, let's define first the concept of the gradient of a function. Let f be a n-dimensional function and g be its local *gradient vector*. For clarity, a two-dimensional function will be used in this definition, but no dimensional restrictions are applied in the gradient calculation. The *nabla* operator is used to denote the gradient operation as in:

$$g(x, y) = \nabla f(x, y)$$

Mathematically, the *gradient* of any differentiable point within a vector space, locates the direction towards the steepest ascent. Imagine our two-dimensional function as a hill, the gradient of a given point would provide the quickest path to reach the top of the hill. It is important to remark that the norm of the gradient, $\alpha = \parallel g \parallel$, provides the step-size or learning rate, this is, the rate of increase of the function in the direction of the gradient. Sticking to our hill, it will provide the slope to the closest stationary point.

Gradient Descent Let $s = (x_0, y_0)$ be the starting point of the problem and $f(x, y)$ the function that defines the vector space. The *gradient descent* path from point s would be exactly $-g = -\nabla f(s)$ and would define the steepest direction towards the minimum of the function.

Gradient Ascent Analogous to the *gradient descent* but now the direction is that of the gradient g. It points to the fastest path to reach a maximum within function f.

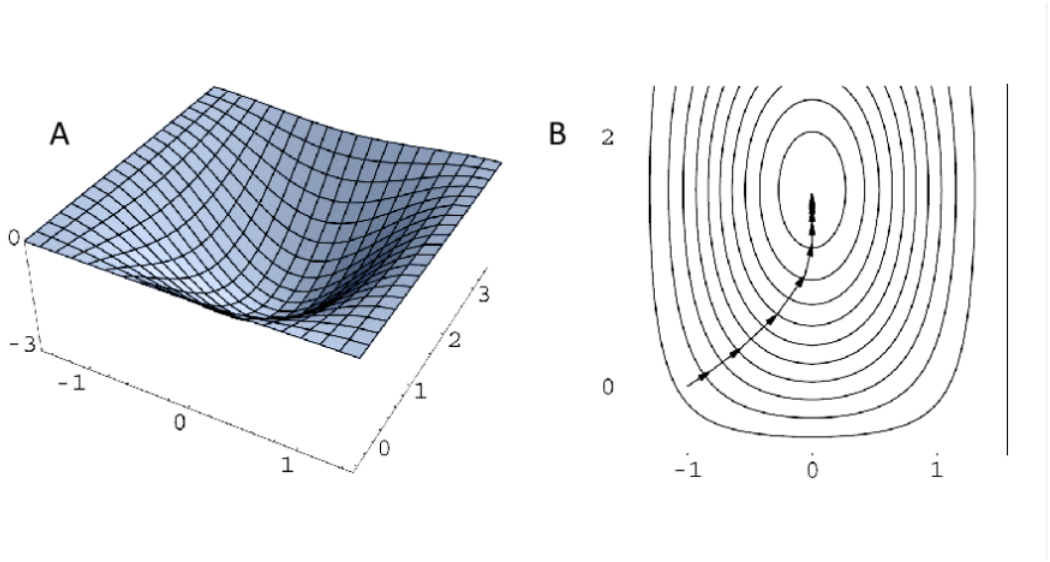


Figure 16. Figure on the left (A) shows 3-D vector function. Figure on the right (B) shows 2-D representation of $-g$ values towards the minimum of the vector space. **Source:** www.researchgate.net

Value-update Rule Let a problem be one of linear regression, with only two parameters: weight w and bias b. After the parameters are randomly initialized and an arbitrary learning rate, α is selected, each epoch during the training, parameter-update will be given by:

$$w_1 = w_0 - \alpha \nabla_w f(x, y) \tag{2.9}$$

$$b_1 = b_0 - \alpha \nabla_b f(x, y) \tag{2.10}$$

The parameter-adjustment will continue until the cost function finds a local minimum, or until the maximum number of training iterations are completed if convergence is not met.

2.4.5. Newton's Method

Another well-known but rarely used learning algorithm is *Newton's method*. The approach uses the *Hessian* matrix (H) to maximize a previously defined *likelihood function*. To clarify the explanation of this learning procedure, let's define a problem archetype.

Let our problem be one of binary classification, this is, given a linear combination of predictor variables (x) and function weights (θ) , the objective will be to obtain a dichotomous signal via the transformation of the input. Let the independent variable $y \in \{0, 1\}$ and the selected model to be a **logistic regression** represented by the sigmoid function $h(x) = \dfrac{1}{1 + e^{-\theta x}}$. If a probability mass function for the distribution is defined by:

$$P(y|x;\theta) = (h_\theta(x))^y(1 - h_\theta(x))^{1-y} \tag{2.11}$$

The right-hand side of the equation will then be the **likelihood function**. Expanding the function over the complete space of our n observations, the *cumulative likelihood* will then fulfill:

$$L(\theta) = \prod_{i=1}^{n} h_\theta(x_i)^{y_i}(1 - h_\theta(x_i)^{1-y_i}) \tag{2.12}$$

But this product could yield values approaching 0, so instead, the **log-likelihood function** will be used, providing a strictly concave environment with one global maximum that will boost convergence.

Value-update Rule So now the objective will be to find the *critical point*, θ, that will maximize the function. For this, a value-update rule can be used to improve performance at each iteration:

$$\theta_{n+1} = \theta_n + H^{-1}_{\ell(\hat{\theta})} \nabla \ell(\theta) \tag{2.13}$$

This second-order approach requires the calculation of the second derivatives of the log-likelihood function which computationally is very expensive.

2.4.6. Levenberg-Marquardt

A similar approach that seeks to solve the memory overuse of the *Newton's method* is the **LM least squares optimization**, introduced in [LEVENBERG, 1944][44] and extended in [MARQUADT, 1963][51]. This second-order algorithm prevents the calculation of the Hessian matrix by using an estimate value instead, boosting computation speed. Let a quadratic error function, e, the *Hessian estimate* for the residuals would be given by:

$$H = J^T \cdot J$$

And the local gradient:

$$g = J^T \cdot e$$

Where J represents the *Jacobian matrix* of first derivatives of the errors with respect to the network's parameters (weights and biases).

Value-update Rule Assuming the biases are set to zero for simplicity, the parameter-update rule of the algorithm will be given by:

$$w_1 = w_0 - (H + \mu I)^{-1} g \tag{2.14}$$

With μ being a scalar constant that controls the algorithm. As it approaches zero, the method turns into 2.13 only using the *Hessian estimate* and as μ values increase, the algorithm resembles *gradient descent*.

2.4.7. Delta Rule

Let a **SLP** with two input layers and a single output neuron be the case of study. Let the weights of the network, $w_{j,o}$ for $j = 1, 2$, to be randomly initialized and the *teaching input* be a set P of defined pairs in the form of (p, t), where p is a training pattern and t the corresponding reference for the output. An *error vector* for every teaching pattern, e_p, can be defined as:

$$e_p = (t_p - y_p)$$

Expanding this definition to the complete set of input patterns, a *global error function*, $E = f(W)$, can be defined as the normalized mapping of the *pattern-specific* error vectors in terms of the weight matrix of the network. Formally, for our single-output structure:

$$E(W) = \sum_{p \in P} \left(\sum_{o \in O} (t_p - y_p)^2 \right) = {}^4 \sum_{p \in P} (t_p - y_p)^2 \tag{2.15}$$

Now the goal is to minimize the global error function. Since the only degree of freedom within the structure are the weights to the output layer $w_{j,o}$ (henceforth we will omit the subscript o for simplicity), the iterative process will consist on, for every pattern p, applying small variations to the weights and analyzing whether $E_p(W)$ increases or decreases. Since the error function is proportional to the weight matrix,W, we can obtain:

$$\Delta w_j = -\eta \cdot \frac{\partial E_p(W)}{\partial w_j} \tag{2.16}$$

Using calculus' *chain rule* on equation 2.16 we get:

$$\Delta w_j = -\eta \cdot \frac{\partial E_p(W)}{\partial o_p} \cdot \frac{\partial o_p}{\partial w_j} \tag{2.17}$$

Taking a look at the first multiplicative term of the right-hand side of the equation, we notice that the change of the *pattern-specific* error function is exactly equal to the difference between the pattern's target and the corresponding output. Coincidentally, this is where the name **Delta** of the algorithm comes from, since this term is named δ_p.

Introducing it into equation 2.17:

$$\Delta w_j = -\eta \cdot (-\delta_p) \cdot \frac{\partial o_p}{\partial w_j} \tag{2.18}$$

Recalling the *activation function* for the output layer is the identity, $a_j = o_j$, and assuming linearity in the activation function of the input layer, we can state that the only change in the output signal when a weight w_j suffers a small variation comes from a change in the input of the network. Thus:

$$\frac{\partial o_p}{\partial w_j} = \frac{\partial \sum\limits_{j \in J} (o_p w_j)}{\partial w_j} = o_p \tag{2.19}$$

[4]Notice that this equality is only true because we are only considering one output neuron.

Bringing together 2.17 and 2.19 and expanding for the whole set of teaching patterns yield the weight update rule for the algorithm:

$$\Delta w_j = \eta \cdot \sum_{p \in P} o_p \delta_p \tag{2.20}$$

2.5. MLP training: Step by Step

Now that some of the main learning algorithms have been introduced, this section will try to illustrate the training process of a *supervised learning* structured like the **Multi-Layer Perceptron**, which will be the core of this document and the focus of analysis henceforth.

For the purpose of this section, a structure with one input neuron, $I = \{i\}$, a hidden layer formed by only two neurons $H = \{h_1, h_2\}$ and a single output, $O = \{o\}$ neuron will be used.

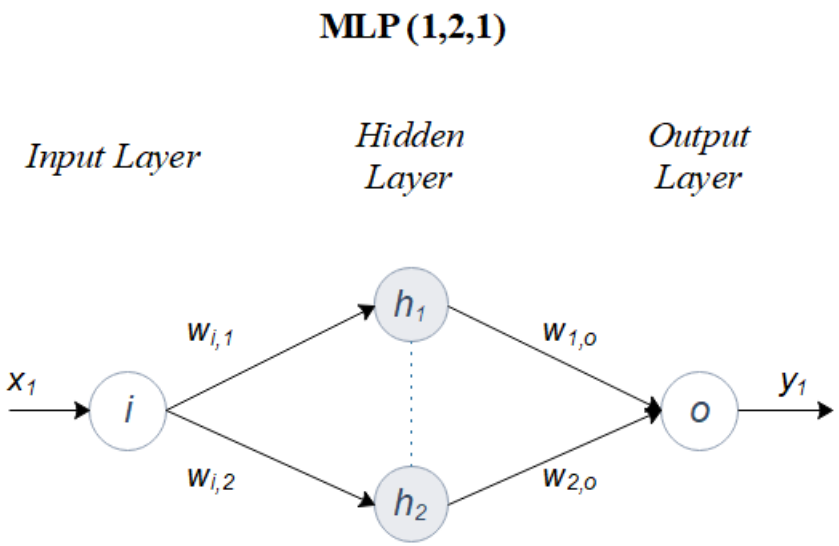


Figure 17. Multi-Layer Perceptron that will illustrate the training process.

The activation function that will be considered will be the semi-linear *hyperbolic tangent*, as introduced on a previous section. MLP architectures use the **backpropagation of error** to adjust the parameters, i.e. the weighs of the

connections. This iterative approach requires the activation function to be differentiable but not necessarily linear as in the Delta Rule. Actually, the case explained in the previous section is an specific of the general backpropagation of error algorithm used in MLP architectures.

2.5.1. Backpropagation of Error

The basis of this *gradient descent* algorithm are the same ones as the ones that allow the Delta Rule to be of any use. Actually, to generalize the Delta Rule is sufficient to expand the concept of δ to every *inner* neuron of the network. For our $(1, 2, 1)$ architecture, we will need to focus on the four weights of the intermediate links: $\{w_{i,1}, w_{i,2}\}$ and $\{w_{1,o}, w_{2,o}\}$.

Once again the objective of the training process will be to minimize the global error function $E(W)$[5], which again is supposed to be quadratic in the form of:

$$E(w) = \frac{1}{2} \sum_{o \in O} e^2(w) \tag{2.21}$$

For this, we have to analyze the effect of the change in weights of the network with respect to the output signal. Let h_1 be the selected neuron of analysis for an arbitrary iteration. If a small change is applied and assuming the same correlation between the weight matrix and the global error as in 2.16:

$$\Delta w_{1,o} = -\eta \frac{\partial E(W)}{\partial w_{1,o}} \tag{2.22}$$

Introducing neuron's input definition from the beginning of the chapter[6] and applying the *chain rule* we obtain:

$$\Delta w_{1,o} = -\eta \underbrace{\frac{\partial E(W)}{\partial net_{h_1}}}_{-\delta_{h_1}} \cdot \underbrace{\frac{\partial net_{h_1}}{\partial w_{1,o}}}_{o_i} \tag{2.23}$$

So finally the **weight-update rule** is as follows:

$$\Delta w_{1,o} = \eta \cdot \delta_{h_1} \cdot o_i \tag{2.24}$$

[5]We are omitting the specific error function for individual input patterns p and analyzing the compound of the whole set P.

[6]$net_{h_1} = o_i w_{i,1}$

Expanding the rule to the complete set of *actionable* weights for every iteration, the weight-update process will be completed.

Train-stopping Criteria To avoid infinite loops if convergence in the training process is not met, i.e. the global error does not decrease to the desired value, some criteria for *early-stopping* has to be pre-defined. The most-common criterion are:

- Maximum number of train iterations or train epochs
- Minimum error decrease during n consecutive iterations.

2.6. Overfitting in ANN

A very common issue that arises in *machine learning* models like artificial neural networks is **overfitting**, as discussed in [HAWKINS, 2004][32]. This is, the model digests successfully the sample data it has been trained with but fails to provide accurate results when the data set is substituted.

2.6.1. Definition

Formally, a model is *overfit* when it assimilates the noise and the details that appear within the *training data* in a way that balances bias and variance and is able to give precise solutions with unseen data.

How is overfitting of a model detected? A very-efficient way is to use an *out-of-sample* test set. An *OOS* set consists in data that shares the general specifics of the training set but has different noise signals and details. An overfit model will estimate the training data accurately but will provide poor results against the test set.

2.6.2. How to Prevent Overfitting

As discussed before, the usual way to detect this issue is with the use of a test set but there are some other equally-effective approaches.

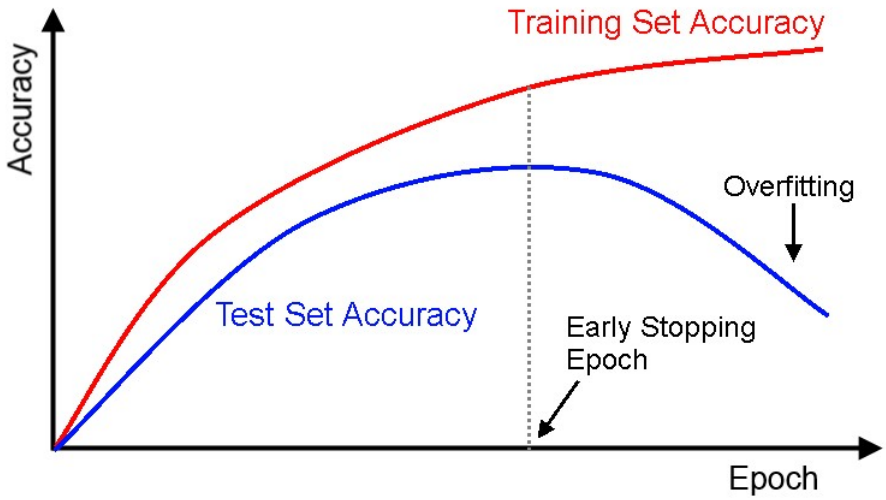


Figure 18. Typical graph of the accuracy of both the training and the test sets of an **overfit** model. **Source:** https://deeplearning4j.org/earlystopping

Cross Validation One way to make the model more effective on filtering the noise in the data is by tuning its hyper-parameters[7] according to not only the training but also the test set, this is known as **cross validation** as discussed in [RON, 1995][71].

For instance, if a model is designed to predict financial data and the available data is a historical record of the previous twenty years, one might want to use nineteen years for the training set and leave out the last one as the test set. Then, the choice of the training hyper-parameters will have to be such that maximizes both accuracy measures.

Regularization The idea of *regularization* presented in [POGGIO, GIROSI, JONES, 1995][66] focuses on constraining the *flexibility*[8] of the model so it does not assimilate the points that represent randomness and sticks to the ones that explain the underlying properties of the training data.

[7]The hyper-parameters of a model are the number of training rounds, the learning rate and other structural design components of the model such as the number of neuron.

[8]Flexibility in machine learning refers to the ability of the model to fit the complete set of training data points.

The procedure to implement **regularization** in the training of a machine learning algorithm is to include penalization terms in the cost function that will be the objective of the minimization problem during the training. This procedure is very common when there is a need to decrease the variance of a model in a regression problem without increasing the bias.

Lasso or L1 Regularization [TIBSHIRANI, 1996][81] Let a linear regression with p predictor variables to be the problem-archetype example for this explanation and the cost function to be the *RSS*. Then the goal will be to minimize:

$$RSS = \sum_{i=1}^{n} \left(y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j \cdot x_{ij} \right) \right)^2 \tag{2.25}$$

Then by adding the regularization term, λ, the new cost function yields:

$$RSS_{L1} = RSS + \lambda \cdot \sum_{j=1}^{p} \parallel \beta_j \parallel \tag{2.26}$$

The first-order regularization prevents the model to yield β coefficients that approach infinite to minimize the error function thus making the learning more efficient. It is important to be aware that meanwhile the coefficients of a *least-squares* regression are scale-independent, those of **Lasso** regression are dependent of the mean of the predictor values. This requires the standardization of the training data by means of *scaling* and *normalizing* the sample set so the mean equals zero.

Ridge or L2 Regularization [TIKHONOV, 1943, 1963][83][84] This procedure is analogous to that of *Lasso* regression but it uses the squares of the coefficients as the regularized argument, turning 2.26 into:

$$RSS_{L2} = RSS + \lambda \cdot \sum_{j=1}^{p} \beta_j^2 \tag{2.27}$$

Chapter 3

Forecasting with Interval-Valued Data

3.1. Intervals and their Advantages in Statistics

At times, a point-value representation of a phenomenon may be insufficient to accurately describe the reality of the process.

Interval Analysis was conceived as an endeavor to be a more authentic representation of the real world processes. Moore, with [MOORE, 1966][55] and [MOORE, 1979][56] established the basis for the development of interval analysis in the field of computational science as an approach to diminish the numerical errors that computers make when rounding and truncating numbers in their computations.

[MATE, 2012][52] introduces another application in engineering, where the measuring errors of some sensors are better acquainted for by treating the measurement as an interval, rather than an unique crisp value. For this matter, there is a real need to study the distance between measurements to obtain accurate error statistics, this is why interval arithmetic was designed in the field of *Numeric Analysis* and *Computational Mathematics*.

3.2. Properties of Intervals

Now that some of the advantages of handling interval data with respect to crisp values have been presented, we shall go into detail about the specifics of intervals.

3.2.1. Definition and Notation

Definition Formally, an interval is the convex set of real numbers that lie within two extreme values. There are several ways to represent an interval.

Standard Notation Probably the most extended way of representing an interval is by means of its two extreme values. The literature may be confusing at times when it comes to inclusion of said values.

In this paper, brackets will denote inclusion of the value in the set and parenthesis will designate exclusion. Let x_l and x_u be the lower and upper *boundaries* of a given interval, $[x]$, then:

- $[x_l, x_u] \equiv$ a closed interval, both values are included in the set.
- $[x_l, x_u) \equiv$ a left-closed, right-open interval, The lower value lies within the set but the upper is excluded.
- $(x_l, x_u] \equiv$ a right-closed, left-open interval. The upper value lies within the set but the lower is excluded.
- $(x_l, x_u) \equiv$ an open interval, both values are excluded from the set.

Spherical Notation As in coordinates, there is also a representation of intervals in terms of a center and a *radius*.

Since the previously defined convex set is one-dimensional the term *spherical* is rather relative and it will have to be thought of as the projection of a ball centered at x_c and of radius x_r onto the real line. Thus, interval $[x]$ can be very-well expressed as $\langle x_c, x_r \rangle$.

Notation Equivalence Both notations are equally valid and therefore interchangeable. To convert one form into the other a system of two linear equations is sufficient:

$$x_c = \frac{x_u + x_l}{2}$$
$$x_r = \frac{x_u - x_l}{2}$$

Yielding the standard-notation equivalence, $[x_c - x_r, x_c + x_r]$.

3.2.2. Interval Arithmetic

Interval Arithmetic was introduced in [SUNAGA, 1958][78] and later restated in [ALEFELD, MAYER, 2000][4] and holds that, as any set of numbers, intervals also have specific mathematical properties that define the operations and transformations.

Center Shift To accomplish the shifting or translation of the *center* of an interval it is sufficient to add a *degenerate interval*[9], a constant $[k]$:

$$\begin{aligned}\langle x_c, x_r\rangle + [k] &= [x_c - x_r, x_c + x_r] + [k, k] \\ &= [(x_c + k) - x_r, (x_c + k) + x_r] \\ &= \langle x_c + k, x_r\rangle\end{aligned}$$

Radius Growth Another way to perform an interval transformation is to alter the size of its *radius* through the addition of a symmetric interval centered around the origin, $[k] = \langle 0, k\rangle$ with $k > 0$[10].

$$\begin{aligned}\langle x_c, x_r\rangle + [k] &= [x_c - x_r, x_c + x_r] + [-k, k] \\ &= [x_c - (x_r + k), x_c + x_r + k] \\ &= \langle x_c, x_r + k\rangle\end{aligned}$$

[9] A degenerate interval is equivalent to a point values since the upper and lower boundaries are exactly equal.

[10] Note that if we allow k to be negative, the resultant radius might also be negative which is not mathematically plausible.

Radius Shrinkage The only way to perform a radius shrinkage is by means of the combination of an *homothetic transformation* and a translation. With $k \in [0, 1)$:

$$\begin{aligned} k \cdot \langle x_c, x_r \rangle - (k-1) \cdot [x_c] &= k \cdot [x_c - x_r, x_c + x_r] - (k-1) \cdot [x_c, x_c] \\ &= [x_c - kx_c, x_c + kx_r] \\ &= \langle x_c, kx_r \rangle \end{aligned}$$

Interval Operators [REDONDO, 2013][69] defines a generic operation denoted by the operator $\langle o \rangle$ between two intervals, $[x]$ and $[y]$, as:

$$\begin{aligned} [x]\langle o \rangle [y] &= [x_l, x_u]\langle o \rangle [y_l, y_u] \\ &= \Bigg[\min\Big(x_l\langle o \rangle y_l, x_l\langle o \rangle y_u, x_u\langle o \rangle y_l, x_u\langle o \rangle y_u \Big), \\ &\qquad \max\Big(x_l\langle o \rangle y_l, x_l\langle o \rangle y_u, x_u\langle o \rangle y_l, x_u\langle o \rangle y_u \Big) \Bigg] \end{aligned}$$

Applying the aforementioned to the well-known operations *addition*:

$$\begin{aligned} [x] + [y] &= [x_l, x_u] + [y_l, y_u] \\ &= [x_l + y_l, x_u + y_u] \end{aligned} \tag{3.1}$$

And *multiplication*:

$$\begin{aligned} [x] \cdot [y] &= [x_l, x_u] \cdot [x_l, y_u] \\ &= [x_l \cdot y_l, x_u \cdot y_u] \end{aligned} \tag{3.2}$$

Elementary Functions Let f be a one-variable function, monotonic in the interval $[x_l, x_u]$; then, by definition, the image of said function will be another interval expressed in terms of the images of the extreme values of interval $[x]$:

$$f([x]) = f([x_l, x_u]) = \Bigg[\min\Big(f(x_l), f(x_u) \Big), \max\Big(f(x_l), f(x_u) \Big) \Bigg] \tag{3.3}$$

This definition yields some basic functions:

- **Opposite:** $\equiv -[x_l, x_u] = [-x_u, -x_l]$

- **Inverse:** $\equiv \frac{1}{[x_l,x_u]} = \left[\frac{1}{x_u}, \frac{1}{x_l}\right]$, for $x_l > 0$ or $x_u < 0$
- **Exponential:** $\equiv e^{[x_l,x_u]} = [e^{x_l}, e^{x_u}]$
- **Logarithm:** $\equiv log[x_l, x_u] = [logx_l, logx_u]$, for $x_l > 0$

Set Operators Since intervals are sets, the arithmetic rules of set operators are also applicable.

- **Union:** $\equiv [x] \cup [y] = \Big[\min(x_l, y_l), \max(x_u, y_u)\Big]$
- **Intersection:** $\equiv [x] \cap [y] = \Big[\max(x_l, y_l), \min(x_u, y_u)\Big]$
- **Symmetrical Difference:** $\equiv [x] \triangle [y] = \frac{[x]\cup[y]}{[x]\cap[y]}$
- **Hull**[11]**:** $\equiv [x] \sqcup [y] = \Big[\min(x_l, y_l), \max(x_u, y_u)\Big]$

3.2.3. Interval Time Series

As described in subsection 1.1.1, any variable that is time-dependent can be considered a time series. In the case of intervals, if a phenomena can be describe in terms a non-stationary interval variable, then it can be referred to as an **Interval Time Series**, henceforth *ITS*, as introduced in [ARROYO, 2008][7].

Notation Let a set of T interval-value observations with a unitary periodicity to be an ITS and $[x_l, x_u]_t$ to be an arbitrary observation at time t, then the complete set will be given by:

$$[x_l, x_u]_t = [x]_t, \ for\ t \in [1, T] \equiv [x]_T$$

[11] Although it may seem like the given definition of the *hull* of an interval is exactly equal to that of the *union* operator, the latter requires the sets to be connected while the former still holds if the two intervals are disjoint sets.

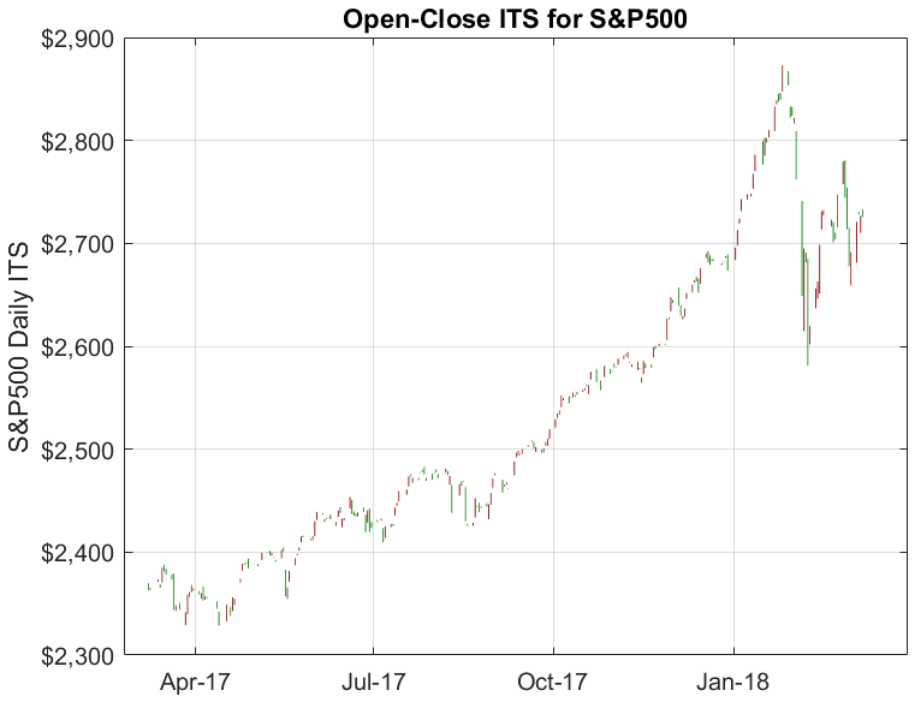


Figure 19. Example of an ITS. Daily Range of the price of S&P500 index.

3.3. Standard Distance Measures

[ARROYO, MATÉ, 2006][8] Analogous to crisp-value modeling, to evaluate accuracy it is necessary to establish an error measure. Since performance in forecasting is graded in terms of how far apart is the prediction from the actual value, for ITS forecasting we shall give a definition of how distances are measured.

Difference Following interval arithmetic discussed earlier this chapter, subtraction of two intervals is given by:

$$[y] - [x] = [y_l, y_u] - [x_l, x_u] = [y_l - x_u, y_u - x_l] \tag{3.4}$$

This technique is not very useful in terms of evaluating accuracy of predictions since for the case of perfect accuracy, $[\hat{y}] = [y]$, the interval difference yields:

$$\begin{aligned}[y] - [\hat{y}] &= [y_l - \hat{y_u}, y_u - \hat{y_l}] \\ &= [y_l - y_u, y_u - y_l]\end{aligned}$$

Which only equals [0] if both intervals are degenerate.

Euclidean Distance The interval form of the *euclidean distance* is very similar to that of point-values:

$$d_E([x],[y]) = \frac{1}{\sqrt{2}}\sqrt{(y_l - x_l)^2 + (y_u - x_u)^2} \tag{3.5}$$

Normalized Symmetric Distance Another approach based that is based on the previous definitions of set operators. If we consider the range of an interval, $w([x]) = x_u - x_l$, then:

$$d_{NSD}([x],[y]) = \frac{w([x] \triangle [y])}{w([x] \cup [y])} \tag{3.6}$$

However, this metric is very limited since it is only applicable if the *union* exists – if the intervals are non-disjoint. A modification has been defined in [MORELL] to provide a solution to this, given by:

$$d_{NSD*}([x],[y]) = 2 - \frac{w([x]) + w([y])}{w([x] \cup [y])} \tag{3.7}$$

The metric yields values in $[0, 2]$ following:

$$d_{NSD*}([x],[y]) = \begin{cases} 2 & \text{if } [x] \text{ and } [y] \text{ are non-equal degenerate intervals.} \\ (1,2) & \text{if } [x] \text{ and } [y] \text{ are non-degenerate disjoint intervals.} \\ 1 & \text{if the intervals are tangent to each other.} \\ (0,1) & \text{if the intersection is a non-empty set.} \\ 0 & \text{if both intervals are non-degenerate and equal.} \end{cases}$$

Hausdorff Distance The definition for the *Hausdorff* metric is analogous to that for crisp values:

$$\begin{aligned} d_H([x],[y]) &= \max\Big(|x_l - y_l|, |x_u - y_u|\Big) \\ &= |x_c - y_c| + |x_r - y_r| \end{aligned} \tag{3.8}$$

Ichino-Yaguchi Distance [ICHINO, YAGUCHI, 1994][37] Let γ be a control parameter, ranging $[0, 0.5]$, then the metric is given by:

$$\begin{aligned} d_{IY}([x],[y]) &= w([x] \cup [y]) - w([x] \cap [y]) \\ &+ \gamma\Big(2w([x] \cap [y]) - w([x]) - w([y])\Big) \end{aligned} \tag{3.9}$$

The author recommends a value of $\gamma = 2$ so (3.9) is simplified onto:

$$d_{IY}^{y=0.5}([x],[y]) = w([x] \cup [y]) - 0.5\Big(w([x]) - w([y])\Big) \tag{3.10}$$

De Carvalho Distance A modification of the previous metric is proposed in [DE CARVALHO, 1996][22] so that the metric is normalized to $[0, 1]$:

$$d_{DC}([x],[y]) = \frac{d_{IY}([x],[y])}{w([x] \cup [y])} \tag{3.11}$$

General $\mathcal{L}^2$ distance Proposed in [GONZALEZ, VELASCO, 2004][28], a reformulation of the euclidean distance with center and radii series follows:

$$d_{\mathcal{L}^2}([x],[y]) = \sqrt{(x_c - y_c)^2 + \theta(x_r - y_r)^2} \tag{3.12}$$

Muñoz San Roque Discrepancy Designed for measuring the accuracy of a multi-layer perceptron applied to ITS. [MUÑOZ, MATÉ, ARROYO, SARABIA, 2007][57] introduces parameter β in the linear combination of the series' centers and radii, turning Equation 3.12 into:

$$d_{MSR}([x],[y]) = \sqrt{\beta(x_c - y_c)^2 + (1-\beta)(x_r - y_r)^2} \tag{3.13}$$

3.4. Intervals and Probability

3.4.1. Probability Distribution Function

Intervals happen to be a specially useful when representing ranging phenomena, this is, non-static processes that have a high variability over short periods of time. For instance, on intraday trading, intervals are crucial to capture the variations of prices within a given time period.

Since ticking or directional data is highly related to probability, it is interesting to associate a probability distribution to an interval time series. Also, this association will be beneficial to compute the distance between to intervals using divergence measures.

Principle of Broader Distribution It is usual to assume the values within a given interval are distributed either as a Gaussian or a uniform distribution. However this is not always the case, so the identification of the probability distribution of an interval is not trivial.

Although inferring the standard distribution for a single interval may be very difficult, if the sample is extended and subsets of similar intervals are generated, the task is considerably eased.

According to [POLLARD, 2000][67], the disregard on information that implies associating an interval to a standard distribution is negligible compared to the associated statistical benefits.

Usual Distributions in ITS As introduced earlier, it is very common to associated either a normal or a uniform distribution to an ITS.

- **Uniform:** Typical for directional data where the changes occur at the extreme values of the interval but the direction is hardly ever changing.

- **Normal:** Usual for high-variability or ticking data where the changes in the direction of the data are frequent but constant. Short runs can occur at times, resulting in larger standard deviation of the distribution.

Parameter Identification The inferring of the representative parameters of the standard distributions is crucial.

In the case of a uniform distribution it is sufficient to have the lower and upper boundaries of the interval as representatives of the distribution function or, equivalently, the center and radius.

In the case of Gaussian distributions, the required paremeters are:

- **Mean:** It is simply associated to the center value of the interval.
- **Variance:** It can be calculated from the standard deviation and is linearly correlated[12] to the radius of the interval in the form $x_d = f(\sigma_d) = k \cdot \sigma_d$. Where k is a constant that is virtually static.

3.4.2. Distance Measures between Probability Distributions

Total Variation As defined in [CHA, 2007][18], the *total variation* or $\mathcal{L}^1$ distance, is a measure of the divergence between two probability distributions. It is obtained by adding the areas of the two non-overlapping regions of the density functions.

Let $P(x)$ and $Q(x)$ be two probability density functions, then the $\mathcal{L}^1$ distance is given by:

$$\delta(P, Q) = \frac{1}{2} \int_{-\infty}^{\infty} |P(x) - Q(x)| dx \tag{3.14}$$

[12]The correlation between the standard deviation and the radius is inferred from the application of the principle of broader distribution to an ITS and the analysis of the behavior of the parameters over several aggregated periods.

Hellinger Distance Another measurement of the divergence between two probability distributions is the *Hellinger distance* [NIKULIN, 1994][60], also referred to as $\mathcal{L}^2$:

$$H^2(P,Q) = \frac{1}{2}\int_{-\infty}^{\infty}\left(\sqrt{P(x)} - \sqrt{Q(x)}\right)^2 dx \tag{3.15}$$

3.4.3. Properties of $\mathcal{L}^1$ and $\mathcal{L}^2$

True Distance Definition For a metric to be considered a *true distance*, there are some required conditions that have to be met.

Thus, not all measures of the divergence between probability distributions are considered valid distance metrics unless they fulfill the following requirements:

- **Non-negativity:** $d([x],[y]) \geq 0$
- **Symmetry:** $d([x],[y]) = d([y],[x])$
- **Subadditivity:** $d([x],[z]) \leq d([x],[y]) + d([y],[z])$
- **Identity of indiscernibles:** $d([x],[y]) = 0 \Leftrightarrow [x] = [y]$

The demonstration of the validness of the metrics $\mathcal{L}^1$ and $\mathcal{L}^2$ as true distance measures will not be carried out in this paper so the reader will have to take a small leap of faith and believe it.

Scale Independence An advantage of the two metrics that have been introduced is that they are, by definition, not dependent of the scale of the data they are measuring.

The multiplicative factor $\frac{1}{2}$ makes the range of values of both metrics bounded to $0 \leq \delta(P,Q) \leq 1$. This will be beneficial when comparing errors of variables on different scales.

3.4.4. Application of Probability-based Metrics

3.4.4.1. Error Measures

Since we have made clear that $\mathcal{L}^1$ and $\mathcal{L}^2$ distances are normalized by definition, to obtain a valid and most-importantly an interpretable error measure of an ITS forecast, it is sufficient to aggregate said distances over the whole set of observations, as discussed in [ZELLNER, TOBIAS, 2000][92].

For instance, using one of the proposed following metrics:

iMAPE The *Mean Absolute Percentage Error* is defined in terms of the MAPE of the classic time series extracted from the lower and upper limits of the ITS and the center and radii time series:

$$mape_l = \frac{100}{T} \cdot \sum_{i=1}^{T} \frac{|({y^l}_t - \hat{y}_t^l)|}{{y^l}_t} \; ; \; mape_u = \frac{100}{T} \cdot \sum_{i=1}^{T} \frac{|({y^u}_t - \hat{y}_t^u)|}{{y^u}_t}$$
$$mape_c = \frac{100}{T} \cdot \sum_{i=1}^{T} \frac{|({y^c}_t - \hat{y}_t^c)|}{{y^c}_t} \; ; \; mape_r = \frac{100}{T} \cdot \sum_{i=1}^{T} \frac{|({y^r}_t - \hat{y}_t^r)|}{{y^r}_t}$$

And aggregating them as an arithmetic mean:

$$iMAPE = \frac{mape_l + mape_u + mape_c + mape_r}{4} \tag{3.16}$$

iRMSE The *Root Mean Squared Error* for ITS may then be defined following:

$$rmse_l = \sqrt{\frac{1}{T}\sum_{i=1}^{T}({y^l}_t - \hat{y}_t^l)^2} \; ; \; rmse_u = \sqrt{\frac{1}{T}\sum_{i=1}^{T}({y^u}_t - \hat{y}_t^u)^2}$$
$$rmse_c = \sqrt{\frac{1}{T}\sum_{i=1}^{T}({y^c}_t - \hat{y}_t^c)^2} \; ; \; rmse_r = \sqrt{\frac{1}{T}\sum_{i=1}^{T}({y^r}_t - \hat{y}_t^r)^2}$$

And aggregating them as an arithmetic mean:

$$iRMSE = \frac{rmse_l + rmse_u + rmse_c + rmse_r}{4} \tag{3.17}$$

iUTHEIL The *U statistic* proposed in [THEIL, 1966][80] adapted to intervals is defined in terms of the lower and upper boundaries of the data, as:

$$U_I = \sqrt{\frac{\sum_{t=2}^{T}(y^l{}_t - \hat{y}_t^l)^2 + \sum_{t=2}^{T}(y^u{}_t - \hat{y}_t^u)^2}{\sum_{t=2}^{T}(y^l{}_t - y^l{}_{t-1})^2 + \sum_{t=2}^{T}(y^u{}_t - y^u{}_{t-1})^2}} \tag{3.18}$$

iARV The *ARV* metric for evaluation of forecast accuracy with ITS is given by:

$$ARV_I = \frac{\sum_{t=1}^{T}(y^l{}_t - \hat{y}_t^l)^2 + \sum_{t=1}^{T}(y^u{}_t - \hat{y}_t^u)^2}{\sum_{t=1}^{T}(y^l{}_t - \bar{y}_t^l)^2 + \sum_{t=1}^{T}(y^u{}_t - \bar{y}^u{}_t)^2} \tag{3.19}$$

3.5. Statistical Forecasting with ITS

3.5.1. Simple Methods

A brief overview on the methods explained on section 1.2 will be carried out, pointing out the main differences of the techniques when applied to interval time series.

Average Method All forecasts are set to the mean of the past available observations. Let $[y]_1, ..., [y]_T$ be the available data set, every forecast will satisfy:

$$[\hat{y}]_{T+h|T} = [\bar{y}] = \frac{\sum_{i=1}^{n}[y]_i}{T} \tag{3.20}$$

This method is a great benchmark for ITS that show stationary centers and radii.

Naïve Method Any forecast will be equal to the last interval in the set:

$$[\hat{y}]_{T+h|T} = [y]_T \tag{3.21}$$

This approach is specially advantageous when the series of the centers resembles a random walk process.

Seasonal Naïve Method A similar approach is used when the historical data set shows a strong seasonal component. In this case, the forecast obtained will be equal to the last observation from the same season:

$$[\hat{y}]_{T+h|T} = [y]_{T+h-km} \tag{3.22}$$

Where m represents the seasonal period as defined in 1.2.1.

Drift Method To acquaint an underlying trend in a time series, the drift term is introduced. The forecasted interval's center will fall in a line between the first and the last observed intervals' centers:

$$[\hat{y}]_{T+h|T} = [y]_T + h\frac{y_{cT} - y_{c1}}{T-1} \tag{3.23}$$

This approach provides good results for series with linearly growing centers.

3.5.2. Exponential Smoothing

Simple Exponential Smoothing The implementation of SES forecasting for intervals is straightforward since the *weighted average* used in its formulation is the same as that of classic time series.

Following the definition proposed in [ARROYO, MUÑOZ, MATÉ, SARABIA, 2007][10], the *level component* of the SES method applied to ITS:

$$\begin{aligned} [\ell]_t &= \alpha \cdot [y]_t + (1-\alpha) \cdot [\ell]_{t-1} \\ [\hat{y}]_{t+h|t} &= [\ell]_t \end{aligned} \tag{3.24}$$

The *error correction form* will not be considered when working with ITS because it implies the subtraction of intervals and, as has been discussed earlier, this operation is problematic.

Trend Influence on ITS Before introducing the proposed formulation for the *Double Exponential Smoothing* for intervals, it is appropriate to discuss how trend influences the different components of an ITS.

For the purpose of this discussion, let the two components of an arbitrary ITS, $[x]_t = \langle x_c, x_r \rangle_t$, be two separate classic time series[13]: ${x^c}_t$ and ${x^r}_t$. If the series where to have an ascending long-period trend, the effect will be accounted for in the series of the centers. Whereas if the sampling period is set to daily observations and the series experiences a short-term run, the radii series will be affected.

However, these two behaviors are not independent. Actually, the standard deviation shows a strong linear correlation to the range of the interval.

Double Exponential Smoothing The DES model proposed for interval-valued time series is a combination of a classic trend smoothing of the centers and an interval smoothing of both the center and radii time series. Analogous to the formulation for classic time series, DES uses two smoothing parameters, $\alpha, \beta^* \in [0, 1]$.

Following the level component form introduced in [MAIA, CARVALHO, LUDERMIR, 2008][48]:

$$\begin{aligned} b_t &= \beta^* \cdot ({\ell^c}_t - {\ell^c}_{t-1}) + (1 - \beta^*) \cdot b_{t-1} \\ [\ell]_t &= \alpha \cdot [y]_t + (1 - \alpha) \cdot ([\ell]_{t-1} + b_{t-1}) \\ [\hat{y}]_{t+h|t} &= [\ell]_t + hb_t \end{aligned} \tag{3.25}$$

Seasonality Influence on ITS Contrary to the influence of shift patterns with respect to an ITS. The seasonality of a process is usually reflected in both components of the series, centers and radii.

Thus, the application of a *Triple Exponential Smoothing* approach has to be limited to one of the two and requires identification of the seasonal patterns prior to modeling.

Let the two smoothing parameters introduced in section 1.3.2, $\gamma^* \in [0, 1]$ and ${h^+}_m = \lfloor (h-1) \mod m \rfloor + 1$ then [NOGALES, 2011][62] introduces two component formulations.

[13]The superscript c or r denotes centers and radii classic time series, respectively. This notation will be used when a subscript is required for clarity.

Center Seasonal Smoothing If the series of the centers shows trend and seasonality but the radius does not, then the smoothing equations follow:

$$\begin{aligned} {s^c}_t &= \gamma^* \cdot ({y^c}_t - {\ell^c}_t) + (1-\gamma^*) \cdot {s^c}_{t-m} \\ b_t &= \beta^* \cdot ({\ell^c}_t - {\ell^c}_{t-1}) + (1-\beta^*) \cdot b_{t-1} \\ [\ell]_t &= \alpha \cdot ([y]_t - {s^c}_{t-m}) + (1-\alpha) \cdot ([\ell]_{t-1} + b_{t-1}) \\ [\hat{y}]_{t+h|t} &= [\ell]_t + hb_t + {s^c}_{t-m+h^+_m} \end{aligned} \tag{3.26}$$

Radius Seasonal Smoothing If the opposite scenario occurs, where radii are seasonal[14] but centers are not. The smoothing equations shift onto:

$$\begin{aligned} {s^r}_t &= \gamma^* \cdot \frac{{y^r}_t}{{\ell^r}_{t-1}} + (1-\gamma^*) \cdot {s^r}_{t-m} \\ [\ell]_t &= \alpha \cdot \frac{[y]_t + {y^c}_t({s^r}_{t-m} - 1)}{{s^r}_{t-m}} + (1-\alpha) \cdot \ell]_{t-1} \\ [\hat{y}]_{t+h|t} &= [\ell]_t {s^r}_{t-m+h^+_m} - {\ell^c}_t({s^r}_{t-m+h^+_m} - 1) \end{aligned} \tag{3.27}$$

3.5.3. ARIMA Models for ITS

AutoRegressive Integrated Moving Averages approaches to interval-valued time series have not been very much exploited by the scientific community, although some studies have been conducted as in [MAIA et al., 2006][47].

Differencing ITS Although the idea of *differentiation* has no direct equivalence in the context of interval-valued data because it requires the mathematical properties of the subtraction that have been recurrently proved to be overruled for intervals, [REDONDO, 2013][69] proposed the following definition for the differencing of an ITS:

$$[y]_t^{d=1} = \frac{d\langle {y^c}_t, {y^r}_t \rangle}{d(t-1,t]} = \left\langle {y^c}_t - {y^c}_{t-1}, \frac{{y^r}_t + {y^r}_{t-1}}{2} \right\rangle \tag{3.28}$$

That has the benefits of allowing *undifferencing* of the series and that has yielded promising results if the radii series represents a stationary process.

[14]Notice that this form of TES is rather a DES since trend in radii is neglected.

Decomposition Approach The first approach that proved to ensue considerable performance was the modeling of the ITS as two separate classic time series, center and radii series.

The handling of the series is exactly equal to that of classic time series, as discussed in section 1.4. With the forecast, in this case, being provided by:

$$[\hat{y}_{t+h|t}] = \langle \hat{y}^c_{t+h|t}, \hat{y}^r_{t+h|t} \rangle \tag{3.29}$$

With $\hat{y}^c_{t+h|t}$ and $\hat{y}^r_{t+h|t}$ being the predicted values of the classic *ARIMA* forecast for the center series and the radii series, respectively.

This method provides great results on the center series because of the linear nature of the component, but fails to assimilate the stochastic nature of the range of the ITS.

3.5.4. Linear Regression

Let a simple regression problem with one dependent interval-valued variable, $[y_i]$ and one interval-valued predictor $[x_i]$ be the simple regression problem under consideration for a set of n observations. Then it is appropriate to define the following linear model:

$$[y] = \beta[x] + B \tag{3.30}$$

That can be separated onto two different linear equations, one for each of the basic series that compose the ITS:

$$[y] = \beta[x] + B \equiv \begin{cases} y^c = \beta x^c + B^c \\ y^r = |\beta| x^r + B^r \end{cases} \tag{3.31}$$

With $\beta \in \mathbb{R}$ and $B = [B_l, B_u] \ / \ B_l, B_u \in \mathbb{R}$.

Then the objective of the problem will be the minimization of the distance between the predicted values and the actual observations of the dependent variable while fulfilling equation 3.31.

This can be achieved by the arrangement of the parameters, $\hat{\beta}$ and $\hat{B}$, via the solution of the *least squares* problem:

$$\frac{1}{n}\sum_{i=1}^{n} d^2\Big([y_i], \hat{\beta}[x_i] + \hat{B}\Big) = \min_{\beta\in\mathbb{R},} \frac{1}{n}\sum_{i=1}^{n} d_{\mathcal{L}^2}\Big([y_i], \beta[x_i] + B\Big) \tag{3.32}$$

The solution to this problem is proposed by various authors, as shown in [BILLARD, DIDAY, 2000][13] and [NETO, DE CARVALHO][58], but the approach defined in [SINOVA et al. 2012][77] will be considered due to its effectiveness.

Let $\mathbf{1}_{\mathbb{R}_+}$ be a function such that:

$$\mathbf{1}_{\mathbb{R}_+} = \begin{cases} 1 & \text{if } x \in [0, \infty) \\ 0 & \text{otherwise} \end{cases}$$

Let $\hat{\beta}_0$ be:

$$\hat{\beta}_0 = \begin{cases} \infty & \text{if } {x^r}_i = 0,\ \forall i \\ \min\limits_{i:{y^r}_i\neq 0} \frac{{y^r}_i}{{x^r}_i} & \text{otherwise} \end{cases}$$

Let $\hat{\sigma}^2_{[x]}$ be the in-sample θ-variance of $[x]$, defined as:

$$\hat{\sigma}^2_{[x]} = \sigma^2(x^c) + \theta\sigma^2(x^r)$$

with:

$$\sigma^2(x^j) = \frac{1}{n}\sum_{i=1}^{n}({x^j}_i - \bar{x}^j)^2$$
$$\bar{x}^j = \frac{1}{n}\sum_{i=1}^{n} x^j$$

Let $\hat{\sigma}_{[x],[y]}$ be the in-sample θ-covariance of $[x]$ and $[y]$, defined as:

$$\hat{\sigma}_{[x],[y]} = \sigma(x^c, y^c) + \theta\sigma(x^r, y^r)$$

With:

$$\sigma(x^j, y^j) = \frac{1}{n}\sum_{i=1}^{n}({x^j}_i - \bar{x}^j)({y^j}_i - \bar{y}^j)$$
$$\bar{y}^j = \frac{1}{n}\sum_{i=1}^{n} y^j$$

Then finally, the solution fulfills:

$$\begin{cases} \hat{\beta} = \left(2 \cdot \mathbf{1}_{\mathbb{R}_+}(\hat{\sigma}_{[x],[y]} - \hat{\sigma}_{-[x],[y]} - 1)\right) \cdot \min\left(\hat{\beta}_0, \max(0, \frac{\hat{\sigma}_{[x],[y]}}{\hat{\sigma}^2_{[x]}}, \frac{\hat{\sigma}_{-[x],[y]}}{\hat{\sigma}^2_{[x]}})\right) \\ \hat{B} = [\bar{y}] - \hat{\beta}[\bar{x}] \end{cases} \tag{3.33}$$

with:

$$[\bar{x}] = [\bar{x^c} - \bar{x^r}, \bar{x^c} + \bar{x^r}]$$
$$[\bar{y}] = [\bar{y^c} - \bar{y^r}, \bar{y^c} + \bar{y^r}]$$

Choosing the Value of θ Control parameter θ adjusts the influence of the variance of the radii series on the interval variance of the variable. Thus, the choice for the value shall not be arbitrary and shall be such that will provided a minimum of the distance[15] between the forecasts and the actual values.

3.6. Non-linear Forecasting with ITS

3.6.1. Non-linear Regression

An interval approach to *Non-linear Regression* is proposed in [NETO,DE CARVALHO, 2008][58]. The method handles the interval-valued data using classical non-linear regression modeling applied to the separate time series of the midpoints and the range.

Formulation Let $[y]$ be an interval-valued variable dependent of a set of P interval-valued predictors, $[x_p]$ for $p \in P$. In this book, we will assume $P = 1$ for simplicity, but the method is not limited on that regard. Then the relationship between the dependent variable and the predictors can be modeled following:

$$[y_i] = f([x_i], \theta) + [\epsilon_i] \equiv \begin{cases} {y^c}_i = f_c({x^c}_i, \theta^c) + {\epsilon^c}_i \\ {y^r}_i = f_r({x^r}_i, \theta^r) + {\epsilon^r}_i \end{cases} \tag{3.34}$$

[15]The distance used in the minimization algorithm may be any of the proposed in sections 3.3 or 3.4.2.

Where:

- ${x^c}_i, {x^r}_i, \ \forall i = 1, ..., n$ are the classical time series of the centers and radii for the predictor variable.
- ${y^c}_i, {y^r}_i, \ \forall i = 1, ..., n$ are the classical time series of the centers and radii for the dependent variable.
- f_c, f_r are non-linear functions. For instance, the logistic function may be used.
- ${\epsilon^c}_i, {\epsilon^r}_i$ are the classical time series of the centers and radii for the random error[16].
- θ^c, θ^r are unknown parameters.

Then the *Sum of Squares of the erros*, SS for the model may be defined by:

$$\begin{aligned} SS &= \sum_{i=1}^{n} {\epsilon^c}_i + \sum_{i=1}^{n} {\epsilon^r}_i \\ &= \sum_{i=1}^{n} \left({y^c}_i - f_c({x^c}_i, \theta^c) \right)^2 + \sum_{i=1}^{n} \left({y^r}_i - f_r({x^r}_i, \theta^r) \right)^2 \end{aligned} \tag{3.35}$$

Solution The objective of the problem will be to minimize expression 3.35 while fulfilling requirement 3.34. To do so, the estimation of parameters θ^c, θ^r is achieved by utilizing non-linear optimization algorithms such as *BFGS Gradient*, *Conjugate Gradient* or *Stochastic Gradient* which are modifications to the algorithms presented in 2.4.

3.6.2. k-NN Method

An interval approach to the method introduced in 1.5.2 is proposed in [ARROYO, MATÉ, 2009][9] as both an ITS forecasting technique and a pattern-recognition method for interval data. In this book we will focus on the forecasting approach.

[16]This series have to fulfill $\bar{\epsilon}^c = \bar{\epsilon}^r = 0$, have finite variance and be uncorrelated.

Formulation Let $[x]_t$ be an ITS with $t = 1, ..., n$ that can be rearranged into a d-dimensional interval vector following, with d being the time lags of the series:

$$[x]^d{}_t = \Big([x]_t, [x]_{t-1}, ..., [x]_{t-d+1}\Big) \tag{3.36}$$

With $t = d, ..., n$.

Then the distance between the last vector of the ITS and the remaining available observations is calculated by means of the *Mean Distance Error* of any of the metrics introduced in 3.3, thus:

$$d^q\Big([x]^d{}_n, [x]^d{}_t\Big) = \left(\frac{\sum_{i=1}^{d}\Big(d\Big([x]_{n-i+1}, [x]_{t-i+1}\Big)\Big)^q}{d}\right)^{\frac{1}{q}} \tag{3.37}$$

Then a (n-d)-dimensional vector can be defined with the computed values of the $n - d$ distances:

$$[x]^{n-d} = \Big([x]_1, ..., [x]_{n-d}\Big)$$

Forecasting Let k be the pre-defined[17] value of the observations that will be considered for computing in the algorithm.

Then another vector containing the k smaller distances from expression 3.6.2 may be defined as:

$$[x]^d{}_{T_p} = \Big([x]^d{}_{T_1}, ..., [x]^d{}_{T_k}\Big)$$

Then a forecast, $[\hat{x}]_{n+1}$, may be obtained via the weighted average given by:

$$[\hat{x}]_{n+1} = \sum_{p=1}^{k} w_p \cdot [x]^d{}_{T_p} \tag{3.38}$$

With $w_p \geq 0$ and $\sum_{p=1}^{k} w_p = 1$. The assignment of the weights may be equitable such that $w_p = \frac{1}{k}$, $\forall p = 1$ or it may be inversely proportional to the distance of the observed *neighbor*, following:

$$w_p = \frac{\psi_p}{\sum_{p=1}^{k} \psi_p}$$

Where $\psi_p = d^q\Big([x]^d{}_n, [x]^d{}_{T_p} + \xi\Big)^{-1}$ and ξ is a tiny constant that prevents values of ψ_p to approach infinity.

[17]This value, as discussed in 1.5.2 may be arbitrary but it is subject to optimization.

3.6.3. Interval Artificial Neural Networks

Another approach for modeling non-linear phenomena is using machine learning architectures with interval-valued data, henceforth **iANN**.

Definition For any neural network structure to be consider an iANN it must meet at least one of the following requirements:

- The input data must be interval-valued.
- The parameters of the network, i.e. the synaptic weights and bias must be interval-valued.

3.6.3.1. Interval Feed-forward Network

An example of an approach that fulfills the second requirement but not the first one is [ISHIBUCHI et al., 1993][38] proposal of a *feed-forward* network with interval weights and biases. The model maps crisp input data to an interval-valued output.

Architecture The iANN structure proposed has I input neurons, H hidden neurons and a single output neuron, O. The number of input neurons will match the number of input values ($I = n$) and the number of hidden neurons is arbitrary but subject to optimization.

Let X be a n-dimensional vector of input data, then the output of each of an arbitrary **input neuron**, j, will fulfill:

$$net_j = f_{act}(x_i) \tag{3.39}$$

$$o_j = f_{prop}(net_j) \tag{3.40}$$

Where:

- $f_{act} = \begin{cases} 1 & \text{if } x_i \in [0, \infty),\ \forall i \in I \\ 0 & \text{oterwise} \end{cases}$
- f_{prop} is the identity function.

Then for an arbitrary hidden unit, h, it is obtained:

$$[net]_h = \sum_{j \in I} \Big([w]_{h,j} \cdot o_j + [\Theta]_h \Big), \ \forall h \in H \tag{3.41}$$

$$[o]_h = f_{prop}\Big([net]_h\Big) = \Big[f_{prop}(net_{l,h}), f_{prop}(net_{u,h})\Big] \tag{3.42}$$

Where:

- $[w]_{j,h}$ is the interval-valued weight of the connection from input neuron j to hidden neuron h.
- $[\Theta]_h$ is the interval-valued bias of neuron h.
- $[net]_h$ is the interval network input of unit h.
- f_{prop} is the sigmoid function defined in 2.3. Since this function is monotonic for the defined domain, equation 3.42 always holds.

Then, finally, for the output neuron o:

$$[net]_o = \Big[net_{l,o}, net_{u,o}\Big] = \tag{3.43}$$

$$\begin{cases} net_{l,o} = \sum_{h \in H} \Big(\underset{w_{l,h} \geq 0}{w_{l,h} \cdot o_{l,h}} \Big) + \sum_{h \in H} \Big(\underset{w_{l,h} < 0}{w_{l,h} \cdot o_{u,h}} \Big) + \theta_l \\ net_{u,o} = \sum_{h \in H} \Big(\underset{w_{u,h} \geq 0}{w_{u,h} \cdot o_{u,h}} \Big) + \sum_{h \in H} \Big(\underset{w_{u,h} < 0}{w_{u,h} \cdot o_{l,h}} \Big) + \theta_u \end{cases}$$

$$[o] = f_{prop}\Big([net]_o\Big) \tag{3.44}$$

Loss Function Let a set p of training patterns of the form $\Big(x_p, [t]_p\Big)$ be the input for the learning of the neural network, where x_p is a crisp input set and $[t]_p$ the corresponding interval values that represent the desired output.

Then the cost function proposed for minimization during the training stage is given by:

$$E_p = E_{l,p} + E_{u,p} = \tag{3.45}$$

$$= \frac{1}{2}\Big(t_{l,p} - o_{l,p}\Big)^2 + \frac{1}{2}\Big(t_{u,p} - o_{u,p}\Big)^2 \tag{3.46}$$

Learning Rule The proposed algorithm for the adjustment of weights during the training stage is presented with an application on the inner weights that connect the hidden layer and the output. It is a *gradient descent* algorithm with a *momentum* term.

$$[w]_{h,o}(p+1) = [w]_{h,o}(p+1) + \Delta[w]_{h,o}(p+1) \tag{3.47}$$

Where the change in the weight for a new input pattern, $p+1$ is given by[18]:

$$\Delta[w]_h(p+1) = \begin{cases} \Delta w_{l,h}(p+1) = -\eta\left(\frac{\partial E_p}{\partial w_{l,h}}\right) + \gamma \Delta w_{l,h}(p) \\ \Delta w_{u,h}(p+1) = -\eta\left(\frac{\partial E_p}{\partial w_{u,h}}\right) + \gamma \Delta w_{u,h}(p) \end{cases} \tag{3.48}$$

Where:

- $\Delta w_{l,h}(p), \Delta w_{u,h}(p)$ are the lower and upper boundaries of the change in weight of the connection for the previous input pattern, p.
- η is a constant that represents the learning rate.
- γ is a constant that represents the momentum.

3.6.3.2. iMLP

A different approach that handles interval-valued data with crisp weights, i.e. fulfilling requirement one but not two of the above-mentioned, is the **iMLP** proposed in [MUÑOZ et al., 2007][57]. Although a functionality as an interval-valued function approximator was also defined, in this book we will focus on its applications in forecasting.

Architecture The suggested architecture for obtaining a forecast based on n observations of an ITS, $[x]_t$, is composed of $I = n$ input neurons, one hidden layer with H neurons and a single output neuron O.

[18]Subscript o indicating connection towards output neuron will be omitted for clarity.

Let $[x]_t = \langle {x^c}_t, {x^r}_t \rangle$ with $t = 1, ..., n$ be the set of historical interval-valued observations that will serve as input for the network, then:

$$[net]_t = f_{act}([x]_t) \tag{3.49}$$

$$[o]_t = f_{prop}([net]_j) \tag{3.50}$$

Where:

- $f_{act} = \begin{cases} 1 & \text{if } x_t \in [0, \infty),\ \forall t = 1, ..., n \\ 0 & \text{oterwise} \end{cases}$
- f_{prop} is the identity function.

Then the interval neuron input of the j-th hidden unit, $[net]_j$, may be obtained as the weighted linear combination of the n input variables, following:

$$[net]_j = \Theta_j + \sum_{t=1}^{n} w_{t,j} \cdot [o]_t = \\ = \left\langle \Theta_j + \sum_{t=1}^{n} w_{t,j} \cdot {o^c}_t, \sum_{t=1}^{n} |w_{t,j}| {o^r}_t \right\rangle \tag{3.51}$$

Where:

- Θ_j is the crisp bias of neuron j.
- $w_{t,j}$ is the crisp weight of the connection between input neuron t and hidden unit j.
- $[o]_t$ is the interval output obtained from neuron t, as in 3.50.

Then for a defined non-linear *activation function*, $g()$ monotonic for the input domain, the output of said neuron is obtained following:

$$[o]_j = g\Big([net]_j\Big)$$

[57] proposed the *hyperbolic tangent* as activation, yielding:

$$[o]_j = tanh\Big([net]_j\Big) = \Big[tanh({net^c}_j - {net^r}_j), tanh({net^c}_j + {net^r}_j)\Big] = \\ = \left\langle \frac{tanh({net^c}_j - {net^r}_j) + tanh({net^c}_j + {net^r}_j)}{2}, \right. \\ \left. \frac{tanh({net^c}_j + {net^r}_j) - tanh({net^c}_j - {net^r}_j)}{2} \right\rangle \tag{3.52}$$

Finally, another weighted linear combination is performed in order to produce the interval input for the output neuron, which will be equal to the output of the network, $[\hat{y}]_{t+1}$:

$$[\hat{y}]_{t+1|t} = \Theta_o + \sum_{j=1}^{h} w_{h,o} \cdot [o]_j = \left\langle \Theta_o + \sum_{j=1}^{h} w_{h,o} \cdot {o^c}_j, \sum_{j=1}^{h} |w_{j,o}| {o^r}_j \right\rangle \tag{3.53}$$

Loss Function Let a set p of training patterns of the form $\Big([x]_p, [t]_p\Big)$ be the input for the learning of the neural network, representing the input ITS and the corresponding desired interval output.

Then a supervised training shall be performed in order to minimize a proposed global cost function of the form:

$$E = \frac{1}{p} \sum_{i=1}^{p} d\Big([t]_p, [\hat{y}]_p\Big) \tag{3.54}$$

Learning Rule A low-memory Quasi-Newton method is proposed for the optimization of the parameters of the structure. Adopting the measure of discrepancy presented in 3.13, it is required to calculate the first-order derivatives of the loss function which will be used for updating the synaptic weights via *backpropagation.*

Then the general weight-update rule of an arbitrary connection is given by:

$$\frac{\partial E}{\partial w} = \frac{2}{p} \sum_{i=1}^{p} \left(\beta \Big| \hat{y}_p^c - {t^c}_p \Big| \cdot \frac{\partial \hat{y}_p^c}{\partial w} + (1-\beta) \cdot \Big| \hat{y}_p^r - {t^r}_p \Big| \cdot \frac{\partial \hat{y}_p^r}{\partial w} \right) \tag{3.55}$$

If $g = identity$ and $w = w_{h,o}$ we obtain the weight-update rule for the connections between the hidden and the output layer:

$$\frac{\partial \hat{y}_p^c}{\partial w_{h,o}} = \begin{cases} 1 & \text{if } h = 0 \\ {o^c}_{h,p} & \text{if } h > 0 \end{cases}$$

$$\frac{\partial \hat{y}_p^r}{\partial w_{h,o}} = \begin{cases} 0 & \text{if } h = 0 \\ sgn(w_{h,o}) \cdot {o^c}_{h,p} & \text{if } h > 0 \end{cases}$$

Setting $g = tanh$ and $w = w_{j,h}$ we obtain the weight-update rule for the connections between the input and the hidden layer:

$$\frac{\partial \hat{y}_p^c}{\partial w_{j,h}} = \frac{\partial \hat{y}_p^c}{\partial {o^c}_{h,p}} \cdot \frac{\partial {o^c}_{h,p}}{\partial w_{j,h}}$$
$$\frac{\partial \hat{y}_p^r}{\partial w_{j,h}} = \frac{\partial \hat{y}_p^r}{\partial {o^r}_{h,p}} \cdot \frac{\partial {o^r}_{h,p}}{\partial w_{j,h}}$$

Then realizing $g' = sech^2$, the partial derivatives may be evaluated separately, following:

$$\begin{aligned}
\frac{\partial \hat{y}_p^c}{\partial {o^c}_{h,p}} &= w_h \\
\frac{\partial {o^c}_{h,p}}{\partial w_{j,h}} &= \frac{1}{2} sech^2\Big({net^c}_{h,p} + {net^r}_{h,p}\Big)\Big({x^c}_{t,p} + sgn(w_{j,h}) \cdot {x^c}_{t,p}\Big) \\
&\quad + \frac{1}{2} sech^2\Big({net^c}_{h,p} - {net^r}_{h,p}\Big)\Big({x^c}_{t,p} - sgn(w_{j,h}) \cdot {x^c}_{t,p}\Big) \\
\frac{\partial \hat{y}_p^r}{\partial {o^r}_{h,p}} &= |w_h| \\
\frac{\partial {o^r}_{h,p}}{\partial w_{j,h}} &= \frac{1}{2} sech^2\Big({net^c}_{h,p} + {net^r}_{h,p}\Big)\Big({x^c}_{t,p} + sgn(w_{j,h}) \cdot {x^r}_{t,p}\Big) \\
&\quad - \frac{1}{2} sech^2\Big({net^c}_{h,p} - {net^r}_{h,p}\Big)\Big({x^c}_{t,p} - sgn(w_{j,h}) \cdot {x^r}_{t,p}\Big)
\end{aligned}$$

Chapter 4

interval MultiLayer Perceptron: a Matlab Approach

4.1. Foundation

In the context of predictive analysis and machine learning, it is increasingly common for mathematical and statistical software to bring the user the opportunity to model time series. Actually, developers and marketers are continuously tailoring new tools that allow *low-level* users to benefit from the advantages of predictive analysis and modeling. Thus, almost anybody with a computer and an internet connection is currently capable of implementing, for instance, a neural network model in a user-friendly environment that does not require thorough understanding of the theoretical background of the technology.

As defined in subsubsection 3.6.3.2, the interval MultiLayer Perceptron is the adaptation of the MLP but designed specifically to interpret and handle interval data. Since the basic features and characteristics have already been discussed, in this chapter we will focus on the modifications that have been introduced from the original architecture described in [MUÑOZ et al. 2007][57] and its implementation in the `Matlab` environment.

4.1.1. Objective

The tool has been designed as a forecasting instrument. As discussed in subsubsection 3.6.3.2, the interval MultiLayer Perceptron is an adaptation of the MLP designed specifically to interpret and handle interval data. In this case, the architecture has been outlined to work with interval time series of any kind.

Since the basic features and characteristics of the **iMLP** have already been discussed, in this chapter we will focus on the modifications that have been introduced from the original idea described in [MUÑOZ et al. 2007][57] along with its implementation in the Matlab programming environment.

4.1.2. Target Users

Although the tool has been designed to provide a forecasting experience with little effort, it does require basic programming skills since it has not yet been equipped with a graphical interface that would allow home users to generate forecasts effortlessly. This initial version is then thought out to be used by **analysts, data scientists** and **developers**.

4.2. Model Layout and Hyperparametrization

As mentioned, the model implementation is based on the original idea of a feed-forward network that will handle interval time series with crisp parameters, thus resulting on the interval MultiLayer Perceptron architecture. Although most of the features and specifics of the model are common, there are some modifications that have been introduced from the original code written in `C` language.

The structure of the `Matlab iMLP` is very similar to the one proposed in the original paper for ITS modeling. It is comprised of three neuron layers: input, hidden and output.

- **Input Layer:** the number of neurons of the input layer will be determined by the number of observations of the historical data that will be provided to

the network. For instance, fitting of $[\hat{y}]_{t+1} = f([x]_t, [x]_{t-1}, ..., [x]_{t-d})$, the input layer will consist of d neurons and $d + 1$ synaptic weights.

- **Hidden Layer:** the number of hidden neurons is arbitrary and the user's choice will be free in this matter. Although some recommendations are included in Appendix B.

- **Output Layer:** the number of neurons of the output layer is fixed to one since the model will produce one forecast for a given set of input patterns.

4.2.1. Neuron Evaluation

With regard to the activation function used to trigger the individual neuron signals, the original proposal from the iMLP paper has been preserved and the activation functions of each of the layers are the following:

- **Input Layer:** the function used as activation for the input layer is f_{act} : *identity* that yields an output equal to the input received.

- **Hidden Layer:** the activation of the hidden layer requires some mathematical properties in order to implement the *backpropagation* of error for the training of the model as described in subsection 2.2.5. The one chosen for model training was $f_{act} : tanh$.

- **Output Layer:** analogous to the input layer, the output layer (neuron) uses the *identity* function as activation, so the weighted sum of the hidden outputs is unchanged.

4.3. Learning Algorithm

As introduced in subsubsection 3.6.3.2, the original network learning is based on a form of error *backpropagation*, namely a low-memory quasi-Newton method. In this project, a slight modification has been introduced with regard to the training algorithm and the implementation of the network learning is based on a **Stochastic**

Gradient Descent with Momentum (SGDM) procedure. A thorough explanation of the algorithm will now be carried out.

4.3.1. SGDM

The stochastic gradient-based used in this implementation is based on the original procedure described in [ROBBINS, MONRO, 1951][70] of the stochastic optimizer Stochastic Gradient Descent. The algorithm is a first-order gradient method that requires low memory usage and is proved to provide efficient results for non-convex loss functions. Let our model fitting problem be defined considering only cost function analysis, then a general formulation could be:

$$\begin{gathered}\text{minimize } E(W) \\ \text{subject to } W \in \mathbb{R}^n\end{gathered}$$

Where W will be the set of synaptic weights of the iMLP network structure, $\mathbb{R}^n$ denotes the n-dimensional Euclidean space and $E : \mathbb{R}^n \to \mathbb{R}$ is a differentiable function over the input domain that represents the in-sample. Then for a given timestep (training epoch) t, the update rule for an arbitrary weight will be given by:

$$\Delta w_t = -\eta_t \cdot \upsilon_t \tag{4.1}$$

$$\upsilon_t = \upsilon_{t-1} + \nabla_w E(w_{t-1}) \tag{4.2}$$

Where:

- $\eta_t = \eta_{t-1} k^t$: denotes the decreasing learning rate[19] at timestep t.
- γ: denotes the momentum parameter.
- $\nabla_w E(w_{t-1})$: denotes the gradient estimate of the cost function with respect to the previous weight value w_{t-1}.

The form proposed in this book is the online supervised version of the original algorithm and works as shown in Algorithm 1.

[19]The decreasing learning rate condition is demonstrated in [NOCEDAL, WRIGHT, 2006][61].

4.3.2. Training Advantages

The proposed algorithm has some unique underlying mathematical and statistical properties that are beneficial for the user with respect to other machine learning techniques. With respect to the original SGD algorithm proposed by [KESKAR, SOCHER, 2017][41], the improved SGDM method provides faster convergence on high-dimensional parameter spaces such as the one created by the error function of a neural network while maintaining generalization accuracy and computational requirements.

Convergence As opposed to other *gradient-descent* algorithms, the introduction of stochastic noise in the error function by means of random-sampling the training data is helpful in terms of convergence achievement. As demonstrated in [BERTSEKAS, TSIKILIS, 2000][11], any stochastic differentiable function has a probability of one of converging.

This is easily understood with a visual example. Let f be an arbitrary non-convex objective function, subject to minimization, for a given input training pattern. The 3D representation of said function is shown in Figure 20. The process of finding a minimum that takes place in regular gradient descent methods is very likely to proliferate on local minima and thus miss out the global minimum that will result in optimum results.

Since the vector space is fixed for a given input pattern, if the latter is remained constant during the training process and the algorithm does fall in one of the aforementioned saddle points, convergence probability will rapidly decrease towards 0. As a solution to this issue, SGDM shuffling of the input data provides dynamic parameter spaces so the algorithm *jumps out* of this local minima on its path to the global minimum.

Generalization Accuracy The proposed learning method presents outstanding exploration capabilities. As discussed in [KESKAR, SOCHER, 2017][41], SGD algorithms clearly outperform classical gradient descent methods such as

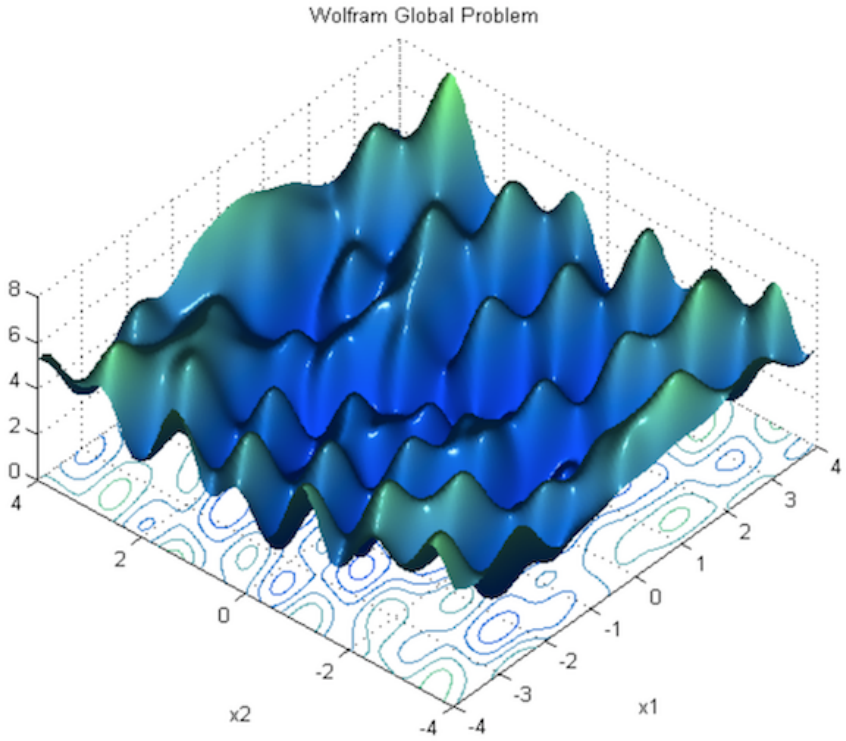


Figure 20. Three-dimensional representation of the extreme case of a non-convex parameter space.

Adam [KINGMA, BA, 2015][42], Adagrad [DUCHI et al., 2009][24] or RMSprop [TIELEMAN, HINTON, 2012][82] for a large number of training epochs. Although the original GD algorithms achieve massive decreases on in-sample error within few epochs of the training process, stochastic algorithms result on significant performance enhancement when the number of iterations is increased as shown in Figure 21.

4.4. Technical Requirements and Dependencies

4.4.1. Matlab

`Matlab`, abbreviation for MATrix LABoratory was created by Cleve Molder in 1984 as a mathematical software and is now a part of the corporation **MathWorks Inc.**, specialized on mathematical computing software that have also developed

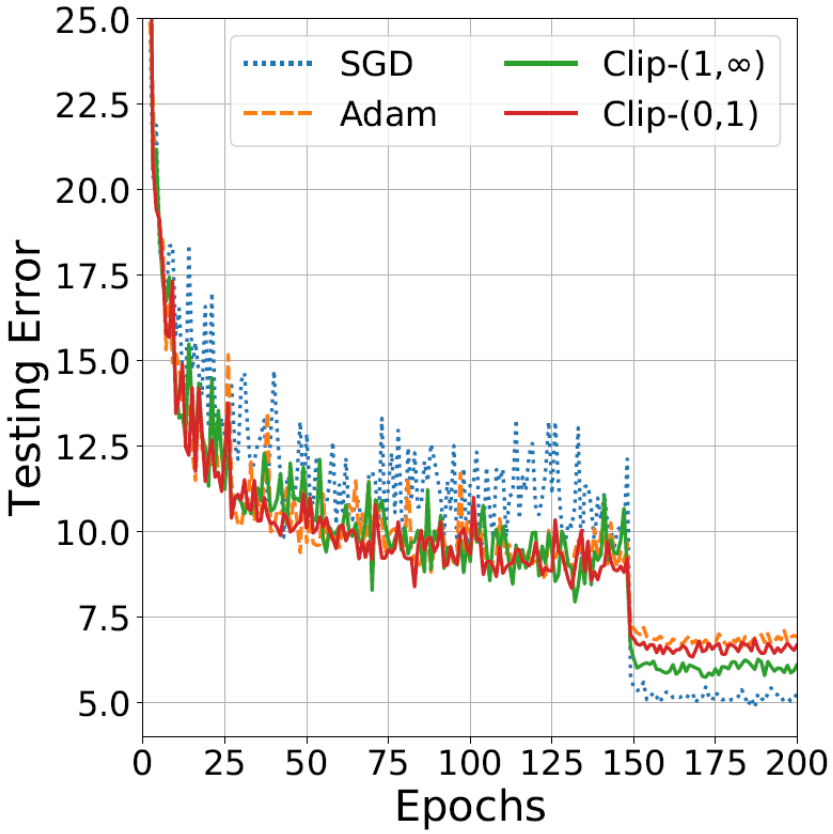


Figure 21. Testing error comparison between optimizers SGD, Adam, Adam-Clip(1,∞) and Adam-Clip(0,1) from evaluation of DenseNet proposed in [41].

`Simulink`. Originally written in `C` language and mostly used by control engineers, it is now widely used within the scientific community because of its powerful data manipulation capabilities, its graphical interface and its integrability with other programming languages.

Language `Matlab` is built on its own programming language. As opposed to other programming software that handle numbers separately and one at a time, `Matlab` operates on whole matrices and arrays for basic tasks such as creating variables, indexing data, arithmetic operating and data storing.

Mathematics The software provides wide numerical computation methods for data analysis, model creating and algorithm development, all oh which built on libraries that optimize processing and boost calculation velocity.

Libraries and Toolboxes `Matlab` offers a variety of built-in toolboxes that provide user-friendly environments to implement calculations and simulations on a range of different scientific fields such as:

- Control Systems
- Signal Processing and Wireless Communications
- Image Processing
- Application Deployment
- Data Analysis and Reporting

These libraries are rigorously tested as opposed to other open-source solutions and thoroughly documented.

Licensing Although `Matlab` is privately run and therefore not free, it does provide special access to students at an affordable price. For the purpose of this project, a `Matlab R2016b` license was used provided by Comillas ICAI as part of the tuition fee at no additional cost.

4.4.2. Technical Requirements

This tool was developed from scratch as a simple `Matlab` script that uses both original built-in packages and custom functions.

Hardware The instrument was designed to be fully functional within the `Matlab` environment, with low processing and computational usage that allow home users to use it smoothly without extremely powerful hardware. Actually, the main hardware that handled the creation and development is a personal computer with an Intel Core i5-6200U processing unit and a RAM of 8 Gb.

Software The tool was developed on 64-bit Windows operating system and `Matlab R2016b`, but it will run on other operating systems such as `Linux` or *Macintosh* just fine. Its integration with **Microsoft Office Excel** requires a license of this kind in order to provide custom data import as a functionality.

4.5. Functionality

4.5.1. Model Fitting

The tool provides both classical and interval time series model fits. The model archetype is fixed to a feed-forward MLP that can be applied in both classification and regression problems.

Data Import The user can import any time series of choice via the integration with **Microsoft Office Excel**. The imported data must have the following format in order for the tool to handle it correctly:

- **File type:** `.xlsx` and `.xls` files are supported.
- **Data structure:** time series data must be placed on the first page of the book. The first row will store variable names and time series will be arranged on separate columns, with the very first being the dates.

head_dates	headseries1_{low}	headseries1_{high}	...	headseriesn_{low}	headseriesn_{high}
date_{t}	series1_{low}(t)	series1_{high}(t)	...	seriesn_{low} (t)	seriesn_{high} (t)
date_{t+1}	series1_{low}(t+1)	series1_{high}(t+1)	...	seriesn_{low}(t+1)	seriesn_{high}(t+1)
...	...	...	...	...	...
date_{t+T}	series1_{low}(t+T)	series1_{high}(t+T)	...	seriesn_{low}(t+T)	seriesn_{high}(t+T)

Table 2. Layout of an ITS data file.

Data Pre-processing In order to prepare the data for model fitting, some transformations must be performed:

- **Center and Radii TS generation:** the tool gets ITS data in [low , high] format and creates center and radius time series.

- **Data split:** to improve model training, the input data will be split into three data sets (training, test and validation). The user will be able to select what percentage of the data will be used during training.

- **Normalization (optional):** standarization transformations can be done to the input data but are not required for the model fitting.

Model Hyperparameter Selection The user will have to provide the structure for the MLP model. For time series regression the number of input and output neurons are fixed so the only parameters to choose from will be:

- Number of neurons of the hidden layer.

- Activation function for the hidden neurons. To choose from the ones described in subsection 2.2.5.

Training Parameter Selection To prepare the tool for fitting the training data, a set of parameters have to be established. Some of them are required and others are optional:

- **Required**

  - `Learning rate:` η , controls weight adjustment with respect to the gradient of the loss function.

  - `Momentum constant:` γ , establishes the influence of the change in weight applied in the previous iteration of the process for the current iteration.

  - `Minibatch size:` sets the size of the random subgroups that will serve as input for every training iteration as part of the learning algorithm.

- **Optional**

- `Maximum epochs:` maximum number of training epochs[20].
- `Maximum error:` threshold value that establishes a minimum for the loss function that will stop the training process.
- `Minimum change in error:` threshold value that will set the minimum change in error for which the training will continue.
- `Number of non-improving iterations:` number of training epochs after which, if no change in error occurs, the training will stop automatically.

Error Measure Parameters In order to evaluate in-training performance, the tool will calculate some of the error measures introduced in section 3.3 and subsection 3.4.2 that will require parameter selection.

- **MSRD:** as described in Equation 3.13, this distance requires parameter β to be assigned. By default, $\beta = 0.5$.
- **IY:** the Ichino-Yaguchi metric requires parameter γ to be assigned. By default, $\gamma = 0.5$.
- **DC:** analogous to the IY metric, the De Carvalho distance requires parameter γ to be established. By default, $\gamma = 0.5$.
- $\mathcal{L}^2$**:** the general $\mathcal{L}^2$ requires the setting of θ. By default, $\theta = 0.7$.

Output After the training process is finished, the user will receive a summary of the fitting that will include the in-sample errors of the forecasts with respect to the target vectors of the test set as well as the number of iterations. The performance evaluation will include:

- **Normalized distances** between time series and model fit.

[20] An epoch in batch training is considered a measure of the number of times the algorithm uses the complete set of training patterns.

- **Benchmark comparison** if a benchmark is produced by the analysis, the model fit will be compared to in terms of total error and error reduction.
- **Error reduction plot** the individual errors of each epoch will be plotted for both training and validation.

Also, a model object will be created that will store the network parameters. This object will be required in order to obtain step-ahead forecasts.

4.5.2. Forecasting

Input In order to generate forecasts from a *trained* model. The user will have to provide three objects to the tool:

- **Model object:** the output object from the training process. This object will store the network parameters (weights) along with the structure definition.
- **Historical data:** the data that will serve as input to the forecast function will ideally be of the same type as the data used during training to obtain accurate results.

Output The output of the forecast will be the predicted series. If a validation set is available, performance evaluation will be carried out and a summary of the out-of-sample errors will be generated. The error metrics will be the ones used for training performance evaluation.

Plotting The predicted series will be plotted along with the validation targets to provide a graphic evaluation of results. The plots will then be exportable as `.jpg` or `.png` files.

Algorithm 1 SGDM learning algorithm

Require: η ▷ Learning rate.

Require: γ ▷ Momentum constant.

Require: p ▷ Batch size.

Require: $epoch_{max}$ ▷ Maximum number of training epochs.

Require: $error_{max}$ ▷ Maximum training error.

procedure SGDM($net, input, output, trp$) ▷ Optimize net weights subject to input, output and training patterns.
  initialize $W \leftarrow rand([-1, 1])$ ▷ Get set of random synaptic weights.
  while $epoch \leq epoch_{max}$ **do**
    while $convergence \leftarrow 0$ **do**
      for $t = 1 : n$ **do**
        initialize $[x]_t \leftarrow batch(trainingset)$ ▷ Shuffle training data.
        initialize $[y]_t \leftarrow batch(testset)$ ▷ Shuffle test data.
        $\eta_{epoch} \leftarrow \eta_{epoch-1} \cdot k^{epoch}$
        $[\hat{y}] \leftarrow net([x])$
        $err_t \leftarrow d([\hat{y}]_t, [y]_t)$
        $\Delta w_t \leftarrow -\eta_{epoch} \cdot (\nabla_{w_t}(E)) + \gamma \cdot (\Delta w_{t-1})$
        $w_t \leftarrow w_{t-1} + \Delta w_t$
      end for
    end while
    $E_{epoch} \leftarrow (1/n) \cdot sum(err_t)$ ▷ Calculate epoch error.
    if $E_{epoch} \geq error_{max}$ **then**
      $convergence \leftarrow 1$
    else
      $convergence \leftarrow 0$
    end if
  end while
  return W ▷ The optimum weights are returned.
end procedure

Chapter 5

Case Studies: Univariate Analysis with Historical ITS Data

5.1. Foreign Exchange Markets

5.1.1. Introduction

Also referred to as FX or Forex market, the foreign exchange market operates on *currency pairs* that are, essentially, the conversion ratio of a currency of a country on that of another country. Forex trading may be seen, in a way, as a reflection of the buyer/seller's impression on the strength of the economic indicators of a country with respect to another.

Trading Instruments The transactional elements of the FX markets are, as defined in [MATÉ, 2014][53]

- **FX Spot Rate** The currency exchange rate, at present time, at which a currency air can be bought or sold.

- **FX Forward Outrights** An outright is an agreement between two counterparts to exchange a currency on a future date at a fixed rate.

- **FX Swap** A FX swap is a contract to buy an amount of the base currency at an agreed rate, and simultaneously resell the same amount of the base currency for a later value date to the same counterpart, also at an agreed rate (or viceversa). Technically an FX swap is a combination of a spot deal and a reverse outright deal.

- **Currency Swap** A cross-currency basis swap agreement is a contract in which one party borrows one currency from another party and simultaneously lends the same value, at current spot rates, of a second currency to that party.

The foreign exchange market is highly striking because of the specific trading features that they offer for traders.

Liquidity The Forex market is very appealing for retail traders because of its huge liquidity. Approximately $4 trillion were traded daily in 2014 and now the market volume is close to $5 trillion.

Continuous Operation The foreign exchange market is functioning throughout the whole day on weekdays, meaning the trader may open/close operations at any time, as opposed to other common markets such as stocks or commodities. As shown in Figure 22, the market joins the banking hours of the five major global markets with an overlap between $12pm$ GMT+1 and $13pm$ GMT+1, resulting in trading volume peaks around that time.

Trading Conditions Due to the high liquidity, Forex brokers offer affordable conditions for small traders. These conditions are, essentially but not limited to:

- **Low Transaction Costs** The *spread*[21] that brokers apply to FX is significantly low due to the high trading volume.

[21]The spread is the difference between the buy and sell prices and it is the main form of revenue for brokers.

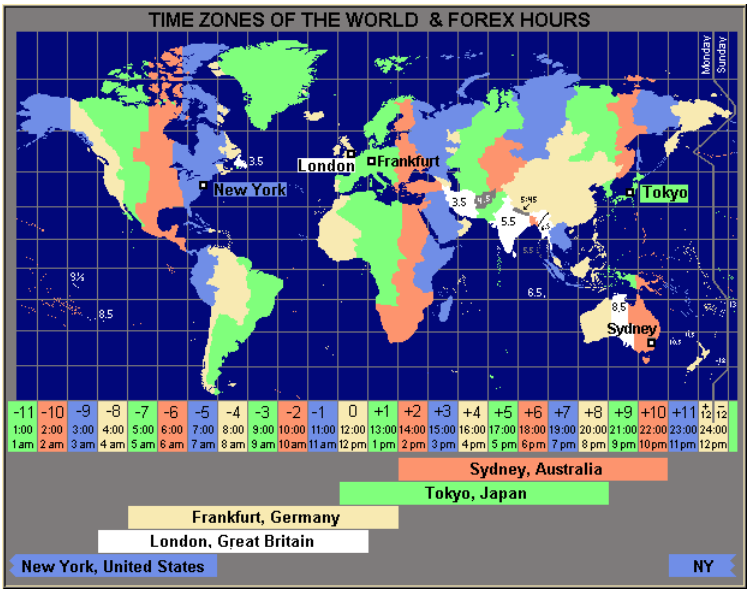


Figure 22. FX market hours by time zone and global market.

- **Leverage** FX markets have high-leveraged instruments, meaning that a small retail trader can take big positions requiring only a small price of the overall amount. This results in low-entry-level instruments.

5.1.2. Forecasting the Forex markets

5.1.2.1. Random Walk model

As introduced in section 1.2, a random walk time series is defined as one whose first-order differences are stationary. This definition implies some specifics that are relevant in time series modeling and forecasting.

Time Independence As discussed in [EVERETT, 1997][26], a random walk process is characterized by statistical features that are not dependent on the time of the observation. This property is highly determinant in forecasting because it entails that the series autocorrelation is only dependent on the time-lag between observations.

This, along with the fact presented in [HULL, 1995][36] that demonstrates that the proportional changes of a random walk time series, i.e. direction and size, are randomly chosen from a Gaussian distribution. Thus, when predicting this specific series, the best prediction is the naïve forecast that implies no change between a future value and the immediate previous available observation.

The Random Walk Hypothesis [PEIRSON et al., 1995][64] introduced a modification of the *Efficient Market Hypothesis* developed by [Fama 1970] that states that an asset price in an efficient market scenario reflects all the available information about the asset: the Random Walk Hypothesis. This subset of the EMH is based on the idea that the foreign exchange market is efficient and thus, the best model explanation will be obtained with a random walk.

Relative Forecasting Accuracy [ROSSI, 2013][74] Provided a thorough analysis of the *predictive power* of monetary models from a skeptic perspective. The author analyses the predictability of the forex markets through economic models based on a series of predictors:

- **Traditional Predictors** such as interest rate differentials, price and inflation differentials, productivity differentials and portfolio balance.
- **Taylor rule Fundamentals** that are based on the connection between relative interest rates, output gaps and inflation levels.
- **Extended Imbalance Measures** such as net foreign assets.
- **Commodity Prices** and other predictors such as Oil prices.

Concluding that the predictability of the FX instruments is very limited and highly sensitive to time horizons, sample period and forecast evaluation models.

[CHINN et al., 2011][20] sustained that, although monetary models explain most of the behavior in the context of foreign exchange rates, there is a relevant underlying stochastic component associated to currency pairs that may be accurately acquainted for using a random walk model.

To test the hypothesis, a practical study was carried out using inter-dealer order flow data on the EUR/USD and USD/JPY pairs and comparing the performance of a random walk model, a monetary model and a hybrid model. The results shown in Figure 23 reflect that, although for longer time horizons the monetary and hybrid models outperform the naïve predictions, on short-time horizons the RW performance is more stable.

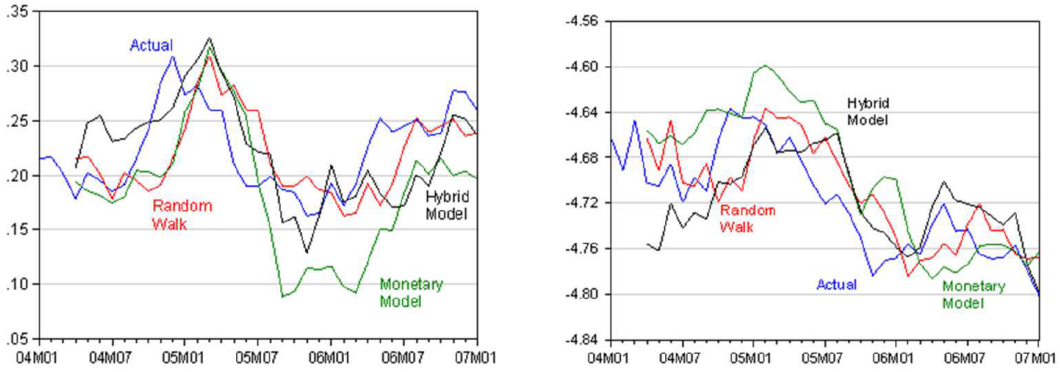


Figure 23. Figure on the left shows the change (log) in the EUR/USD currency pair (actual) along with the model predictions. Figure on the right shows analogous results for the change in the USD/JPY.

5.1.3. Setup

Objective The objective of this study is to asses forecasting performance of a neural network structure in the context of Foreign Exchange Market instruments with interval-valued time series as input data.

Model Definition The model used will be the one depicted in chapter 4 and the forecast generation will be based on lagged data of the form:

$$[\hat{y}]_{t+1} = f([x]_{t-1}, [x]_{t-2}, ..., [x]_{t-z})$$
$$\text{with} \quad z \in \mathbb{N}$$

The choice of the lagged data, parameter z has been carried out in order to feed the model with input that has the information necessary to identify autocorrelations

in the time series. In this case, since the series is observed daily, a time lag of 20 days has been established, matching the 20 trading days of a month. A visual explanation can be shown in Figure 24.

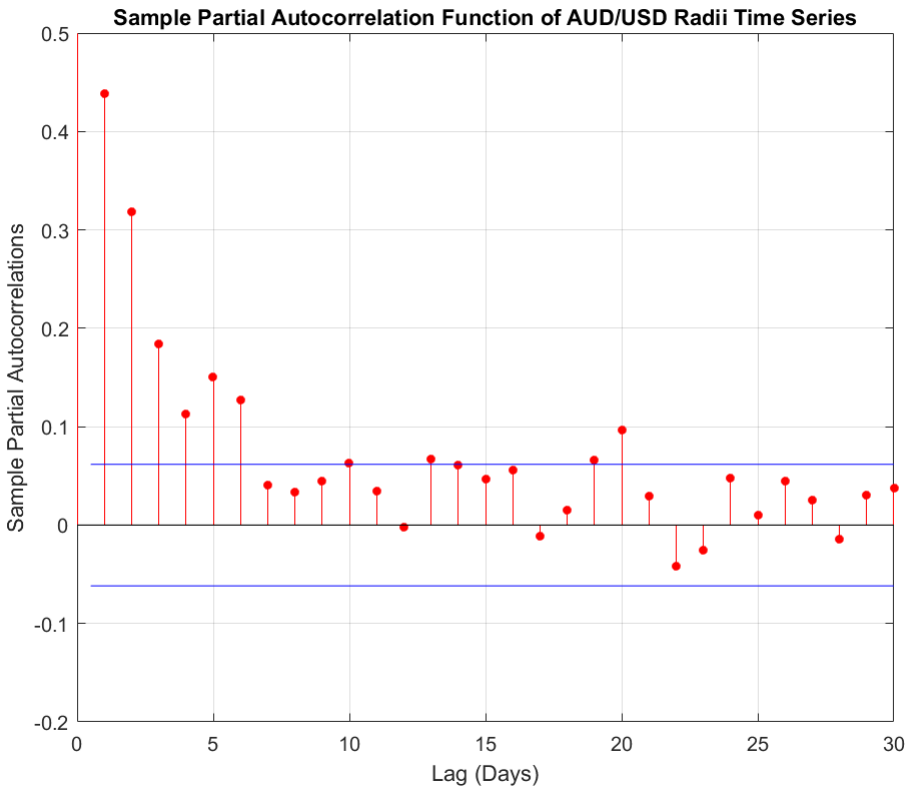


Figure 24. Sample Partial Autocorrelation Function of the classic time series of the range of AUD/USD daily bid. Spike at lag 20 shows relevancy of information until that time lag.

Data Structure The currency pairs used for this case study are the EUR/USD and AUD/USD[22] from January 2007 to December 2016 and have been structured as follows:

[22]Data extracted from Investing.com website.

- **EUR/USD:** daily interval data generated through aggregation of hourly observations. January 2007 to December 2015 data was used during training and 2016 was left as out-of-sample validation.
- **AUD/USD:** daily interval data generated through aggregation of hourly observations. January 2011 through December 2013 data was used during training and 2014 was left as out-of-sample validation.

5.1.4. Results: AUD/USD ITS

Neural Network Structure Choice The optimum structure for the data was found to be an iMLP(20,3,1). The number of input neurons is fixed due to the input data and the choice of the number of hidden networks has been carried out by (forward) structured trial-and-error, as discussed in Appendix B.

Error Parameter Choice The selected distance metrics required parameter choosing:

- **Muñoz San Roque Discrepancy:** $\beta = 0.5$;
- **Ichino-Yaguchi:** $\gamma = 0.5$;
- **De Carvalho:** $\gamma = 0.5$;
- **General L2:** $\theta = 1/3$;

Training Parameter Choice The set of required parameters used during the training of the iMLP were:

- **Learning Rate:** $\eta_0 = 0.01$;
- **Momentum Constant:** $\gamma = 0.9$;
- **Minibatch Size:** $p = 20$;
- **Maximum Number of Epochs:** $\text{epoch}_{max} = 800$;

- **Maximum Training Error:** $\text{iRMSE}_{max} = 0.01$;

Parameter tuning has been carried out manually since the parameter optimization procedure has not been developed at the moment for the proposed iMLP network. For instance, examining the plot of the loss error against the number of training epochs in Figure 25, it is clear that adding more than 1000 epochs to the training does not report any benefits.

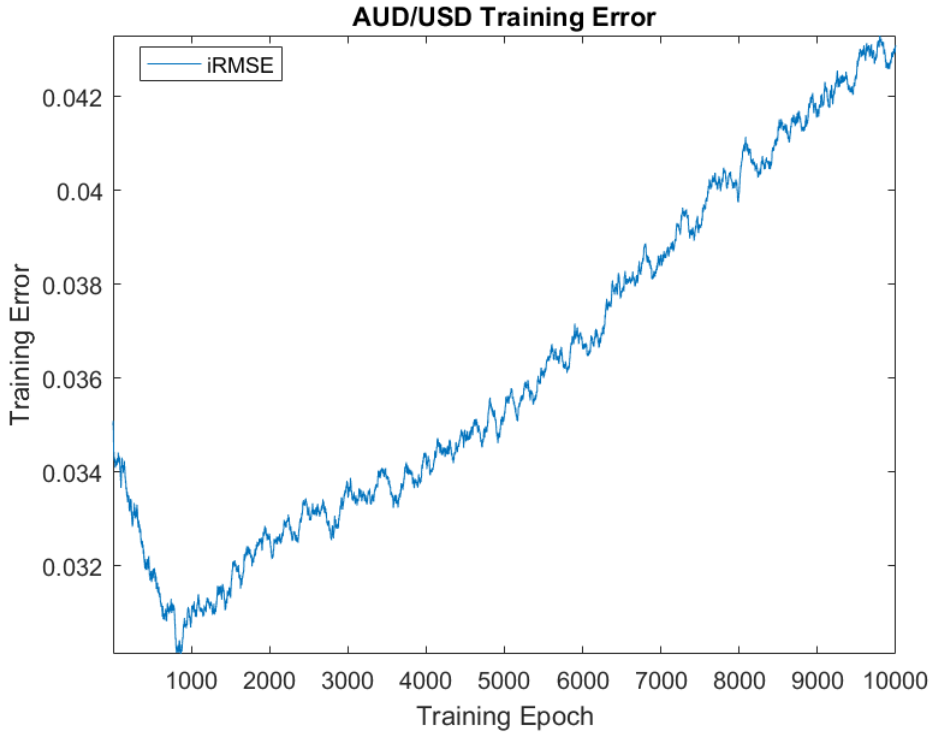


Figure 25. iRMSE error metric against number of training epochs for AUD/USD daily bid price ITS.

Benchmark Analysis The proposed forecasting techniques used as benchmarks for performance comparison are:

- Random Walk model without drift.
- Short Moving Average: MA(5).
- Long Moving Average: MA(20).

AUD/USD		iMAPE	iRMSE	iARV	iUTHEIL
Training	iMLP	90.0214%	0.1136	0.2557	1.0510
	RW	**51.7280%**	0.1039	0.2315	1
	MA(5)	**51.7280%**	**0.0700**	**0.0952**	**0.6412**
	MA(20)	146.3307%	0.1401	0.3906	1.2990
Validation	iMLP	2.9737%	0.1330	0.4000	1.0331
	RW	15.7276%	0.1189	0.3748	1
	MA(5)	15.7276%	**0.0820**	**0.1727**	**0.6788**
	MA(20)	**1.3529%**	0.1109	0.2930	0.8843

Table 3. AUD/USD Daily Bid ITS.

Results: Fit and Validation Table 3 shows the in-sample and out-of-sample performance of the aforementioned models in terms of Mean Absolute Percentage Error and Root Mean Squared Error of the Muñoz San Roque Distance metric, as defined in Equation 3.4.4.1 and Equation 3.4.4.1 as well as the U Statistic for ITS and the ARV metric applied to intervals, following Equation 3.4.4.1 and Equation 3.4.4.1. All of them compared to the benchmark forecasting metrics established earlier.

5.1.5. Results: EUR/USD ITS

Neural Network Structure Choice The optimum structure for the data was found to be an iMLP(3,3,1). The number of input neurons is fixed due to the input data and the choice of the number of hidden networks has been carried out by (forward) structured trial-and-error, as discussed in AppendixB.

Training Parameter Choice The set of required parameters used during the training of the iMLP were:

- **Learning Rate:** $\eta_0 = 0.01$;
- **Momentum Constant:** $\gamma = 0.9$;
- **Batch Size:** $p = 3$;

- **Maximum Number of Epochs:** $\text{epoch}_{max} = 200$;
- **Maximum Training Error:** $\text{MSRD}_{max} = 0.01$;

Error Parameter Choice The selected distance metrics required parameter choosing:

- **Muñoz San Roque Discrepancy:** $\beta = 0.5$;
- **Ichino-Yaguchi:** $\gamma = 0.5$;
- **De Carvalho:** $\gamma = 0.5$;
- **General L2:** $\theta = 1/3$;

Benchmark Analysis The proposed forecasting techniques used as benchmarks for performance comparison are:

- Random Walk model without drift.
- Short Moving Average: MA(5).
- Long Moving Average: MA(20).

EUR/USD		iMAPE	iRMSE	iARV	iUTHEIL
Training	iMLP	9.9804%	0.0956	0.2403	0.9283
	RW	**7.5497%**	0.0987	0.2788	1
	MA(5)	**7.6487%**	**0.0675**	**0.1224**	**0.6625**
	MA(20)	9.3571%	0.0908	0.2047	0.8569
Validation	iMLP	32.4360%	0.1059	0.3897	0.9700
	RW	36.7123%	0.1070	0.4142	1
	MA(5)	34.5103%	**0.0856**	**0.2504**	**0.7776**
	MA(20)	**13.9440%**	0.1197	0.4729	1.0685

Table 4. EUR/USD Daily Bid ITS.

Results: Fit and Validation Table 4 shows the in-sample and out-of-sample forecast accuracy of the aforementioned model applied to the EUR/USD daily bid price in terms of Mean Absolute Percentage Error and Root Mean Squared Error of the Muñoz San Roque Distance metric, as defined in Equation 3.4.4.1 and Equation 3.4.4.1 as well as the U Statistic for ITS and the ARV metric applied to intervals, following Equation 3.4.4.1 and Equation 3.4.4.1. All of them compared to the benchmark forecasting metrics established earlier.

5.2. Stock Markets

5.2.1. Introduction

The term *Stock Market* refers to the aggregation of markets and exchanges where equities of publicly held companies, bonds, and other securities are traded. Also referred to as the *Equity Market*, this collection of markets is one of the most crucial components of a free-market economy since it provides companies with the opportunity to access capital in exchange for giving different types of investors a portion of ownership.

Market Agents The main structure of this marketplace is comprised of two major sections:

- **Primary market** where new issues are first sold through IPOs[23]. Institutional investors are the principal agents in this context and provide the company that is being offered with money directly from the buying of its shares.
- **Secondary market** after the initial selling of the companies' shares is completed and the share price is established, individual investors come into play and the trading is continued. In this part the company does not receive funds per transaction.

[23]Initial public offerings.

Within these two divisions, several *players* adopt different roles. In general and without paying attention to small details the agents of the Stock Market may be grouped in the following categories:

- **Stockbrokers** act as intermediaries between investors and the stock exchange, taking care of the buyings and sellings as licensed professionals.
- **Stock analysts** are in charge of facilitating insights to investors on stocks. Based on thorough research, they give advise on whether to buy, sell or hold a position in the market.
- **Portfolio managers** handle client portfolios and set the investment strategies for the money they hold.
- **Investment bankers** represent companies in various capacities, such as private companies that want to go public via an IPO or companies that are involved in pending mergers and acquisitions.

Stock Exchanges The term comprises any organized and regulated financial market where different securities are traded. They serve as both primary and secondary markets and they are characteristic because they impose stringent rules, listing requirements, and statutory requirements that are binding on all listed and trading parties.

Almost all exchanges are what is referred to as *auction exchanges* where buyers and sellers enter competitive bids and orders, respectively, throughout the trading hours.

The first stock exchange that is documented was established in the early XVII century by the *Dutch East India Company* in Amsterdam. Later renamed as the **Amsterdam Bourse** it was the first marketplace to formally issue stocks and bonds. Currently there are many stock exchanges spread worldwide, the three largest exchanges based on market capitalization are the **New York Stock Exchange (NYSE)**, the **London Stock Exchange (LSE)** and the **Tokyo Stock Exchange (TSE)**.

Trading Products There are two major types of trading items that differ, essentially, on whether the product is listed on a stock exchange or traded directly between parties through a dealer.

The former are referred to as **listed securities** and they are identified because they must meet reporting regulations of the SEC[24] as well as those of other exchanges they are traded at. The latter are designated as **Over-the-counter** securities (OTCs) and, since they are not compulsory meeting reporting requirements, they are more opaque in the sense that there is harder to find credible information about them, as it happens with information on private companies.

As an example of **listed securities**, hereafter are listed the instruments offered as listed securities on the NYSE:

- **Equities/Stocks** comprise the set of shares of small, medium and large-cap companies. The acquisition of an equity represents an ownership interest of the investor relative to the asset.

- **Options** provide investors with the right to operate an asset at an agreed-upon price known as *strike price* as shown in Figure 26. They are characterized by having an expiration date and, although in American markets the right to exercise (buy or sell) the product is valid during the complete time between the acquisition by the option buyer and the expiration date, in Europe the exercise is restricted to the expiration date.

- **Bonds** give stockholders easy access to the debt market offering fixed income investment opportunities. A bond contract defines a money loan from the investor to the issuer over a fixed period of time at a fixed interest rate. The most common bond types are corporate bonds, municipal bonds and treasury bonds.

- **Exchange-Traded Products** are somewhat more complex derivative investment products. ETPs are traded intra-day and their price is derived from other

[24]The Securities and Exchange Comission is the regulatory body in charge of overseeing the United States stock markets.

instruments such as commodities, interest rates, share prices, etc. They include exchange-traded funds (ETFs), exchange-traded vehicles (ETVs), exchange-traded notes (ETNs) and certificates. This type of security is becoming more and more appealing to investors since it blends the simplicity of stocks with the diversified risk of mutual funds[25].

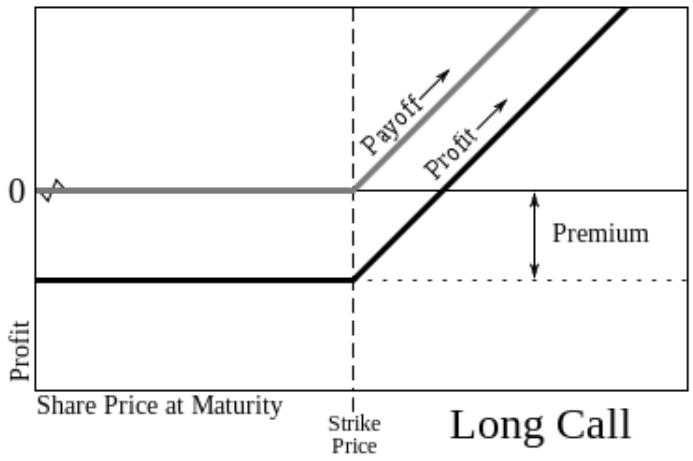


Figure 26. Payoff and profit from an option buying or a long call option.

Stock Market Performance Indicators In order to serve as a general overview of the performance of a certain market, the concept of **indexes** as stock market indicators was first unveiled in 1896 by finance journalist Charles H.Dow. He proposed the first version of a stock index, currently known as the **Dow Jones Industrial Average (DJIA)**, as the arithmetic mean of the top 12 stocks in the market at the time. Later in 1928, the form of calculation had to be changed because of the multiple company mergers and stock splits that were taking place and the consequent number warping. Right now, the DJIA is formed by 30 companies that are not necessarily related to the industrial sector.

Tracing back to 1860, Henry Varnum Poor published the *History of the Railroads and Canals of the United States*, the first financial history of American railroad

[25]Mutual funds are investment vehicles made up of a pool of money from different investors and are managed by professional managers.

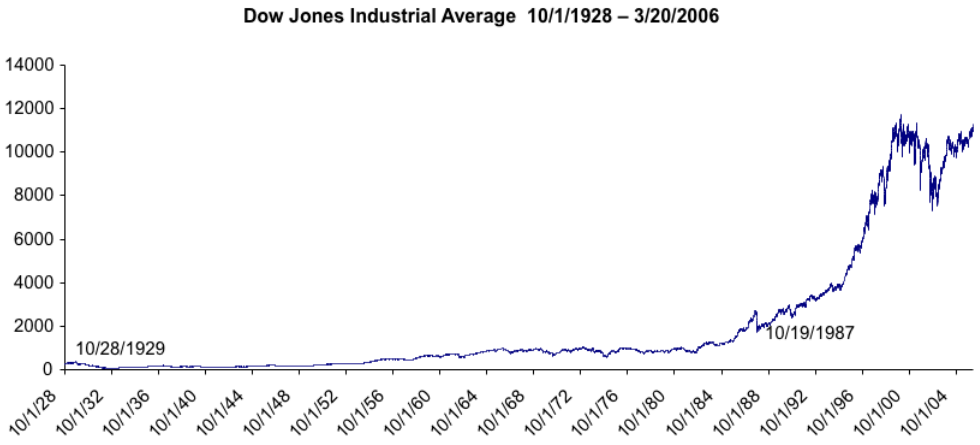


Figure 27. Dow Jones Industrial Average daily price.

companies. The publication was a huge success and it encouraged Poor to continue providing investors with trading information on said companies.

Years later, in the mid XX century, the rating company Standard Statistics started keeping track of an index made up of 223 stocks of companies from varied sectors in the market but was forced to reduce the list to 90 stocks because there was no machine capable of computing the index update with such large number. The index differed from the aforementioned DJIA mainly in the aggregation of stocks. Standard Statistics proposed a market-weighted average[26] formula as opposed to the price-based approach of the Dow.

After the financial crash occurred in 1929 Standard Statistics absorbed Poor's company, that had gone bankrupt, thus forming the **Standard & Poor's 90 Index.** Twenty years later, S&P acquired an IBM computer that boosted the computational capabilities of the company, allowing the index to expand to the current set of 500 companies.

Currently there are countless market indexes that provide passive investors with tradable instruments in almost any stock exchange or OTC dealer. Naming some of the most popular indexes worldwide besides the two previously introduced:

[26]The market-weighted average was based on the stock capitalization and prevented the index from big fluctuations due to the biggest companies.

- NASDAQ 100: the index is formed as a subset of the NASDAQ Composite that tracks all the listed stocks on the NASDAQ stock market, this index is heavily weighted towards information technology companies.
- EURO STOXX 50: this index is comprised of the top 50 biggest companies of the Eurozone. It is also weighted through market capitalization.
- NIKKEI 225: this pool of the 225 most liquid companies from the TSE dates back to 1949 and is calculated as a price-adjusted arithmetic mean, similar but not equal to the Dow Jones.
- IBEX 35: the Spanish index is comprised of the top 35 companies that are traded within the *Spanish Interconnected Stock System (SISS)*, pooling stocks from the four main Spanish stock exchanges of Madrid, Barcelona, Bilbao and Valencia. Its calculation is analogous to that of the S&P500.

5.2.2. Setup

Objective The objective of this case study is to asses the predictive power of the implementation of the iMLP in the context of Stock Markets, handling interval time series.

Model Definition The model used will be the one depicted in chapter 4 and the forecast generation will be based on lagged data of the form:

$$[\hat{y}]_{t+1} = f([x]_{t-1}, [x]_{t-2}, ..., [x]_{t-z})$$
$$\text{with} \quad z \in \mathbb{N}$$

Data Structure The time series that will be analyzed during this study will be the stock performance indicator introduced earlier, S&P500 and the equity of FORD Motor Company[27].

[27]Data extracted from Yahoo Finance

- **IBEX35:** daily interval data extracted directly from historical database. January 2014 to December 2015 data was used during training and 2016 was left as out-of-sample validation.
- **FORD Motor Company** daily interval data extracted directly from historical database. January 2016 through February 2018 data was used during training and March 2018 through June 2018 was left as out-of-sample dataset.

5.2.3. Results: IBEX35 ITS

Neural Network Structure Choice The optimum structure for the data was found to be an iMLP(5,4,1). The number of input neurons is fixed due to the input data and the choice of the number of hidden networks has been carried out by (forward) structured trial-and-error, as discussed in AppendixB.

Error Parameter Choice The selected distance metrics required parameter choosing:

- **Muñoz San Roque Discrepancy:** $\beta = 0.5$;
- **Ichino-Yaguchi:** $\gamma = 0.5$;
- **De Carvalho:** $\gamma = 0.5$;
- **General L2:** $\theta = 1/3$;

Training Parameter Choice The set of required parameters used during the training of the iMLP were:

- **Learning Rate:** $\eta_0 = 0.01$;
- **Momentum Constant:** $\gamma = 0.9$;
- **Minibatch Size:** $p = 5$;
- **Maximum Number of Epochs:** $\text{epoch}_{max} = 30$;
- **Maximum Training Error:** $\text{iRMSE}_{max} = 0.005$;

Benchmark Analysis The proposed forecasting techniques used as benchmarks for performance comparison are:

- Random Walk model without drift.
- Short Moving Average: MA(5).
- Long Moving Average: MA(20).

IBEX35		iMAPE	iRMSE	iARV	iUTHEIL
Training	iMLP	**12.5800%**	0.1362	0.4313	0.9703
	RW	14.4662%	0.1381	0.4581	1
	MA(5)	13.4967%	**0.1079**	**0.2673**	**0.7639**
	MA(20)	21.2370%	0.1595	0.5494	1.0952
Validation	iMLP	**29.9008%**	0.1082	**0.3602**	0.9421
	RW	33.3261%	0.1123	0.4058	1
	MA(5)	32.8271%	**0.0957**	0.3793	**0.8295**
	MA(20)	41.6230%	0.1503	0.6973	1.3108

Table 5. IBEX35 Daily Price ITS.

Results: Fit and Validation Table 5 shows the in-sample and out-of-sample performance of the aforementioned models in terms of Mean Absolute Percentage Error and Root Mean Squared Error of the Muñoz San Roque Distance metric, as defined in Equation 3.4.4.1 and Equation 3.4.4.1 as well as the U Statistic for ITS and the ARV metric applied to intervals, following Equation 3.4.4.1 and Equation 3.4.4.1. All of them compared to the benchmark forecasting metrics established earlier.

5.2.4. Results: FORD Motor Company ITS

Neural Network Structure Choice The optimum structure for the data was found to be an iMLP(4,4,1). The number of input neurons is fixed due to the input data and the choice of the number of hidden networks has been carried out by (forward) structured trial-and-error, as discussed in AppendixB.

Training Parameter Choice The set of required parameters used during the training of the iMLP were:

- **Learning Rate:** $\eta_0 = 0.01$;
- **Momentum Constant:** $\gamma = 0.9$;
- **Batch Size:** $p = 4$;
- **Maximum Number of Epochs:** $\text{epoch}_{max} = 55$;
- **Maximum Training Error:** $\text{iRMSE}_{max} = 0.005$;

Error Parameter Choice The selected distance metrics required parameter choosing:

- **Muñoz San Roque Discrepancy:** $\beta = 0.5$;
- **Ichino-Yaguchi:** $\gamma = 0.5$;
- **De Carvalho:** $\gamma = 0.5$;
- **General L2:** $\theta = 1/3$;

Benchmark Analysis The proposed forecasting techniques used as benchmarks for performance comparison are:

- Random Walk model without drift.
- Short Moving Average: MA(5).
- Long Moving Average: MA(20).

Results: Fit and Validation Table 6 shows the in-sample and out-of-sample forecast accuracy of the aforementioned model applied to the FORD Motor Company daily stock price in terms of Mean Absolute Percentage Error and Root Mean Squared Error of the Muñoz San Roque Distance metric, as defined in Equation 3.4.4.1 and

FORD M. C.		iMAPE	iRMSE	iARV	iUTHEIL
Training	iMLP	**1.6537%**	0.0987	0.2837	0.8855
	RW	15.7534%	0.1077	0.3618	1
	MA(5)	15.5831%	**0.0756**	**0.1670**	**0.7794**
	MA(20)	14.7996%	0.1050	0.2996	0.9099
Validation	iMLP	**27.2911%**	0.1219	**0.3924**	0.8958
	RW	28.4010%	0.1337	0.6136	1
	MA(5)	28.2290%	**0.0970**	0.4075	**0.7079**
	MA(20)	29.0418%	0.1299	0.5224	0.9227

Table 6. FORD Motor Company Daily ITS.

Equation 3.4.4.1 as well as the U Statistic for ITS and the ARV metric applied to intervals, following Equation 3.4.4.1 and Equation 3.4.4.1. All of them compared to the benchmark forecasting metrics established earlier.

Chapter 6

Conclusions and Open Paths for the Future

6.1. Conclusions

Along the discussion of the case studies that have been focus of analysis during this chapter, the main objective has been to asses the predictive power of the machine learning model that has been implemented in this project.

In order to extract relevant conclusions on the learning capabilities of the neural network structure proposed and thoroughly described in chapter 4, some forecasts have been produced using interval time series as presented throughout this chapter.

To account for the performance of the model in the context of financial data such as forex exchange rates and equity prices, two differentiated analysis will be carried out hereafter:

- **Learning Capability** of the proposed model. This will be examined in terms of **convergence** and **generalization accuracy**, as introduced in subsection 4.3.2.

- **Forecast Performance** of the iMLP structure compared to proven benchmark forecasting methods in the context of financial time series analysis. In order to do so, two different evaluating approaches have been drawn upon. Firstly,

stand-alone accuracy will be addressed with the analysis of MAPE and RMSE adapted to acquaint for interval time series. Secondly, relative forecast accuracy will be approached using Theil's inequality coefficient U adapted to ITS as well as the interval adaptation of the Average Relative Variance.

6.1.1. Learning Capability

Convergence Convergence of machine learning nonlinear algorithms, namely neural network approaches, is dependent mostly of four variables:

- Loss function convexity.
- Learning algorithm choice.
- Processing hardware.
- Dataset size.

Considering the proposed model architecture in this project has, at least, one hidden neuron in the hidden layer, the loss function of the training process is compulsory **non-convex**[28]. Given this, convergence meeting should be harder the more complex the network architecture is, e.g. convergence time is proportional to the number of hidden neurons.

The selected learning algorithm, **SGDM**, guarantees convergence in the weight optimization problem in exchange of higher computational cost compared to other learning methods, as introduced in subsection 4.3.2. However, the model shows favorable data-fitting results at relatively small computational costs: in-sample error minimization is achieved rather quickly in terms of both training epochs and computation times.

The hardware used to carry out this project runs on a Graphic Processing Unit that boosts computational times. On top of that, both train and validation sets are

[28]Proof that for a neural network with >0 hidden neurons the loss function is non-convex is extended in the literature, namely [AUER, HERBSTER, MANFRED, 1996][12]

of little size compared to the common data used in machine learning problems, talk about Big Data problems.

Generalization Accuracy In the context of machine learning and, specifically when applying supervised learning algorithms as the one proposed in this project, the greater reason of poor performance is lack of **generalization accuracy**. Reminding that, in the end, a supervised learning **training** process consists on the approximation of a target function given a set of sample data, it is easy for models to tend to err on the side of replicating the training data, thus resulting in weak performance during **validation** with out-of-sample data.

For our specific case of time series forecasting, given a set of historical data as training input, the model will try to to find the optimum function approximation that maps the relevant attributes of the data. For a training to be carried out effectively, the algorithm must have in mind the bias/variance trade-off that exists in supervised learning methods.

- **Bias error** is associated to poor assimilation of the set features and will result in large in-sample or training errors.
- **Variance error** is attributed to excessive sensitivity to small fluctuations of the training data set and will produce poor out-of-sample performance or large validation error.

In order to account for the aforementioned phenomena, the proposed learning Stochastic Gradient Descent with Momentum shuffles the input data to expand the explored vector space during training, as described in subsection 4.3.2 For the case studies analyzed in this project, Table 7 shows a validation-to-train error percentage calculated from the RMSE of the forecasts obtained from the model for the different datasets as:

$$E_{rate} = 100 \cdot \frac{validation_{error} - train_{error}}{train_{error}}$$

It yields negative percentage values if generalization accuracy of the model is favorable and positive values if, on the contrary, the model is **overfit**.

	AUD/USD	EUR/USD	IBEX35	FORD
E_ratio	17.0775%	10.7741%	-20.5580%	7.5056%

Table 7. Validation-to-error percentage.

The results imply **favorable induction** of the model to the training data for most of the analyzed cases, specifically for the stock price series. On the forex market forecasts, the model overfits a little, but it is not significant.

6.1.2. Forecast Performance

In order to asses the predictive power of the proposed model in the context of financial markets, some real life data sets comprised of financial interval time series in daily range have been gathered. This is, for each trading day the ITS is formed by the lowest price observed during trading hours and the highest price of the day. The series may be extracted directly from a given source as it was done with stock prices ITS or generated through aggregation of hourly observations in order to determine the lower and upper bounds of the trading day. The forecast analysis has not been performed using the interval time series as is, but manually transforming the series in order to obtain the spherical notation, following:

$$[X]_t = [x_l, x_u]_t = \langle x_c, x_r \rangle_t$$

Once the center and radius classic series are computed, the data set is divided onto train and validation sets in order to asses the inductive capabilities of the proposed MLP neural network model adapted to intervals.

Contrary to some ITS forecasting approaches that have been proposed by the scientific community that handle the fitting of the input data as two separate classic time series (centers and radii) for merging the interval series back together after the predicted time series is generated the interval Multilayer Perceptron approach (iMLP) applied in this project is designed to manage ITS data directly as input, and produce the corresponding output also as an interval time series, as depicted in the original model proposition in [57].

The ITS input data is pre-processed before being fed into the neural net, since it is proven that scaling the series helps boosting convergence in the context of feed-forward neural networks. The scaling procedure is done in terms of the base and the range of the time series, following:

After the forecasts are generated, the data is re-scaled back using the original parameters (base and range). Once the predicted series is collected, the **rival forecasts** are computed from the original series. For the purpose of this project, the following rival forecasting models have been chosen as they are thought to be very simple yet significantly robust in the context of financial univariate time series analysis:

- **Random Walk without Drift** specially in order to asses the robustness of the proposed iMLP against rather arbitrary behaviors, the first order naïve forecast is set as the first rival predictor.
- **Short Moving Average of period 5** a classic time series forecasting approach, widely used in the financial forecasting context, it is a very hard-to-beat forecast.
- **Long Moving Average of period 20** analogous to the previously defined rival predictor, it acquaints for the trend of the series since the rolling window is wider. It provides very accurate predictions in trending contexts.

The post-sample accuracy will then be evaluated following the evaluation strategy proposed in [CHEN,YANG,2004][19].

Stand-Alone Accuracy Analysis Stand-alone accuracy metrics are commonly related to the computation of a certain cost function, for instance an accuracy measure based on absolute or quadratic loss functions. Historically, to evaluate post-sample stand-alone accuracy, MSE-criterion-based measures have been the go-to choice by researchers and forecasters.

As its very name indicates, **stand-alone** measures do not require rival forecasts to be calculated, therefore they analyze the time series separately.

For the purpose of this study, stand-alone metrics Root Mean Squared Error and Mean Absolute Percentage Error have been evaluated adapted to interval time series and then evaluated considering each ITS to be theoretically identical to a classic time series. Table 8 and Table 9 shows the post-sample accuracy in terms of the aforementioned metrics for the predicted ITS as well as the rival forecasts evaluated independently.

CASE 1 : FOREX MARKETS			
AUD/USD Daily Bid ITS (2014)			
Type	Forecasting Method	iMAPE	iRMSE
Benchmark	Short Moving Average (5 periods)	15.7276%	0.0820
Benchmark	Random Walk without Drift	15.7276%	0.1189
Proposed Method	**interval MultiLayer Perceptron**	**2.9737%**	**0.1330**
Benchmark	Long Moving Average (20 periods)	1.3529%	0.1109
EUR/USD Daily Bid ITS (2016)			
Type	Forecasting Method	iMAPE	iRMSE
Benchmark	Short Moving Average (5 periods)	34.5103%	0.0856
Benchmark	Random Walk without Drift	36.7123%	0.1070
Proposed Method	**interval MultiLayer Perceptron**	**37.4360%**	**0.1259**
Benchmark	Long Moving Average (20 periods)	13.9440%	0.1197

Table 8. Stand-alone accuracy measures of the proposed method forecasts compared to the rival predictors.

Table 9 shows the proposed model stand-alone accuracy as well as the metrics evaluated for the rival predictors.

It is relevant to note that in the context of Forex Exchange rates, rival predictors outperform the iMLP in terms of stand-alone accuracy. This may be mainly attributed to the little non-linearity of the series, a context in which simple forecasts based on averages are very reliable as the iMAPE metric shows. The long moving average MA(20) outperforms any other forecast method in terms of Mean Absolute Percentage Error.

CASE 2 : STOCK MARKETS			
IBEX 35 ITS (2016)			
Type	Forecasting Method	iMAPE	iRMSE
Benchmark	Short Moving Average (5 periods)	32.8271%	0.0957
Benchmark	Random Walk without Drift	33.3261%	0.1123
Proposed Method	**interval MultiLayer Perceptron**	**29.9008%**	**0.1082**
Benchmark	Long Moving Average (20 periods)	41.6230%	0.1503
FORD M. C. Daily ITS (2018)			
Type	Forecasting Method	iMAPE	iRMSE
Benchmark	Short Moving Average (5 periods)	28.2290%	0.0970
Benchmark	Random Walk without Drift	28.4010%	0.1337
Proposed Method	**interval MultiLayer Perceptron**	**27.2911%**	**0.1219**
Benchmark	Long Moving Average (20 periods)	29.0418%	0.1299

Table 9. Stand-alone accuracy measures of the proposed method forecasts compared to the rival predictors.

However, when forecasting equity prices, the performance of the proposed iMLP model is improved, with significantly low values for both MAPE and RMSE and outperforming in both cases the Random Walk and the MA(20). The reason behind this may be the underlying non-linearities of the stock prices that are better acquainted for by a neural network like the iMLP.

Relative Accuracy Analysis In volatile frameworks, stand-alone measures may yield very large numbers because of an unpredictable anomaly, exogenous to the time series under analysis. As a result, relative accuracy comes very handy when assessing post-sample performance because of both the simplicity and robustness of the benchmark forecasts. In the context of improving predictive power of a model, an outlier may throw out the stand-alone based loss function and thus attribute excessive importance to the anomalous period with respect to the rest of the series.

Relative accuracy may reduce the bias introduced by potential trends or seasonal components, provided that the benchmark forecast handles these issues appropriately. Nonetheless, the choosing of an adequate relative forecast measure is not an easy task. For the purpose of this project, the relative measures chosen to evaluate the model post-sample performance have been Theil's U adapted to intervals and the Average Relative Variance applied to ITS.

CASE 1 : FOREX MARKETS			
AUD/USD Daily Bid ITS (2014)			
Type	Forecasting Method	iUTHEIL	iARV
Benchmark	Short Moving Average (5 periods)	0.6412	0.1727
Proposed Method	**interval MultiLayer Perceptron**	**1.0510**	**0.4000**
Benchmark	Random Walk without Drift	1	0.3748
Benchmark	Long Moving Average (20 periods)	1.2990	0.2930
EUR/USD Daily Bid ITS (2016)			
Type	Forecasting Method	iUTHEIL	iARV
Benchmark	Short Moving Average (5 periods)	0.7776	0.2504
Proposed Method	**interval MultiLayer Perceptron**	**0.9700**	**0.3897**
Benchmark	Random Walk without Drift	1	0.4142
Benchmark	Long Moving Average (20 periods)	1.0685	0.4729

Table 10. Summary of forecasting errors of the proposed method compared to benchmarks in terms of the U Statistic metric and the Average Relative Variance for interval time series.

Table 10 shows the proposed model relative accuracy as well as the metrics evaluated for the rival predictors.

Table 11 and Table 10 show the proposed iMLP model relative accuracy as well as the metrics evaluated for the rival predictors. The iMLP's relative performance is significant, improving the Random Walk model in 75% of the cases and yielding better predictions than the average of the observations 100% of the times. However,

CASE 2 : STOCK MARKETS			
IBEX 35 Daily ITS (2016)			
Type	Forecasting Method	iUTHEIL	iARV
Benchmark	Short Moving Average (5 periods)	0.8295	0.3793
Proposed Method	**interval MultiLayer Perceptron**	**0.9421**	**0.3602**
Benchmark	Random Walk without Drift	1	0.4058
Benchmark	Long Moving Average (20 periods)	1.3108	0.6973
FORD Motors Comp. Daily ITS (2018)			
Type	Forecasting Method	iUTHEIL	iARV
Benchmark	Short Moving Average (5 periods)	0.7079	0.4075
Proposed Method	**interval MultiLayer Perceptron**	**0.8958**	**0.3924**
Benchmark	Random Walk without Drift	1	0.4075
Benchmark	Long Moving Average (20 periods)	0.9227	0.5224

Table 11. Summary of forecasting errors of the proposed method compared to benchmarks in terms of the U Statistic metric and the Average Relative Variance for interval time series.

it is shown that for Foreign Exchange rates, the rival predictors outperform the iMLP.

In conclusion, no major assumptions can be extracted from the results of these case studies other than the iMLP provides promising forecast performance in the field of financial interval time series prediction. It is still to confirm its predictive power against other classic forecasting methods that have provided excellent results in financial ITS frameworks, such as exponential smoothing methods for ITS, ARIMA approaches or kNN distance-based algorithms.

Predicted ITS Plots Hereafter are presented some of the predicted ITS generated from the iMLP for different forecast horizons. Plots have been generated using an original plotting strategy proposed in [MACEDA, 2018][46].

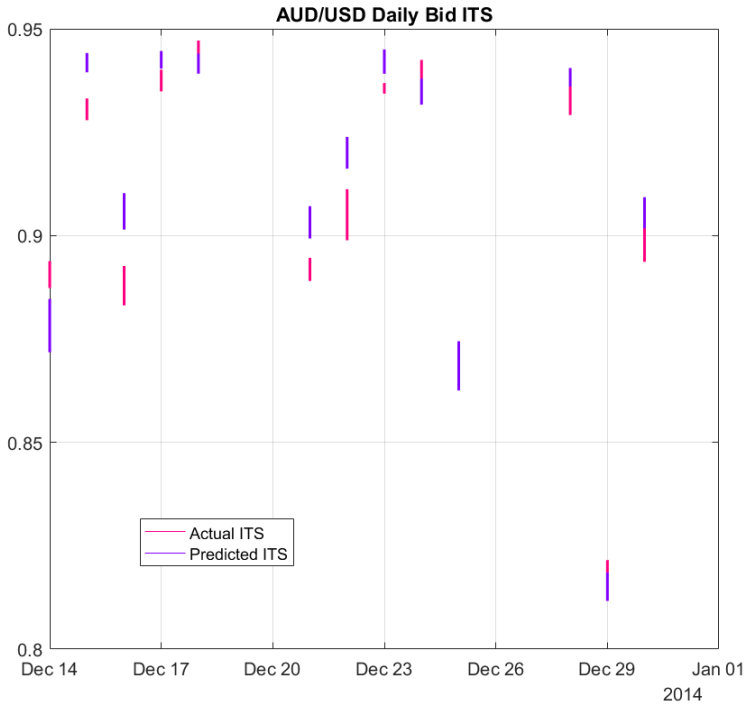


Figure 28. Predicted ITS and Actual series for a 13-step-ahead forecast horizon of the AUD/USD Daily Bid price ITS.

It is remarkable that the predictive power is not a function of the forecast horizon since the forecasts are generated as individual one-step-ahead forecasts from a rolling time window.

6.2. Open Paths for the Future

After completion of the project, improvement possibilities from where this project is finished at are introduced hereafter. They are not limited, and any researcher or data analyst that may feel inspired by the study that has been carried out is thus invited to contribute on this line of investigation.

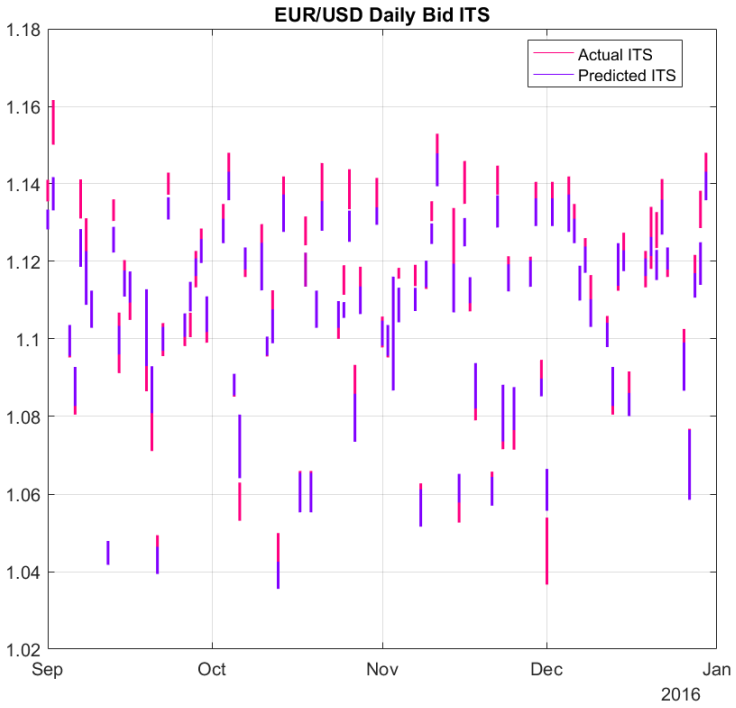


Figure 29. Predicted ITS and Actual series for 90-step-ahead forecast horizon of the EUR/USD Daily Bid price ITS.

6.2.1. Tool Optimization

The implementation of the iMLP in MatLab, despite being thoroughly designed, it is subject to major improvements.

Parameter Tuning System The tuning of both hyperparameters of the net and the training parameters has been handled manually since the implementation of a line search has not been developed at the moment. Its realization will result in major improvements in forecast efficiency as well as usability for lower-level users.

Graphical Interface At the moment the tool is run on a MatLab script, this is, usability is very limited and the effort-to-result ratio is subject to substantial reduction in this matter. A toolbox approach is thus recommended for the iMLP implementation in MatLab.

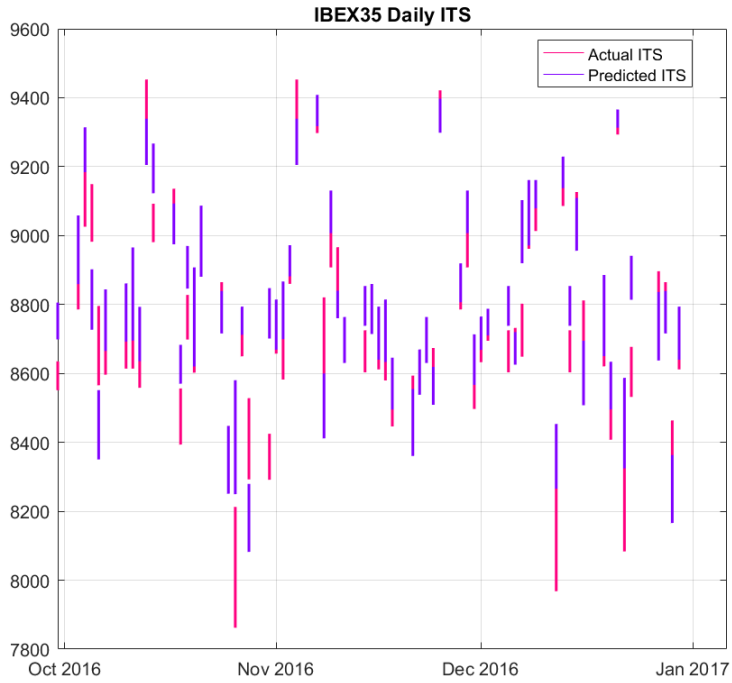


Figure 30. Predicted ITS and Actual series for 60-step-ahead forecast horizon of the IBEX 35 Daily price ITS.

Adding of a Second Hidden Layer An improvement towards a Deep Learning structure will be of high relevance, since it will provide with the structure to learn on much more complex problems.

6.2.2. Assesment of Predictive Power in other Frameworks

As mentioned earlier this chapter, conclusions on the predictive power of the model are for now limited to the specific case studies that have been presented in this project. It is yet to be determined the prediction accuracy in order fields of study. As plausible suggestions, forecasting electricity demand prices may be of interest to the researching community.

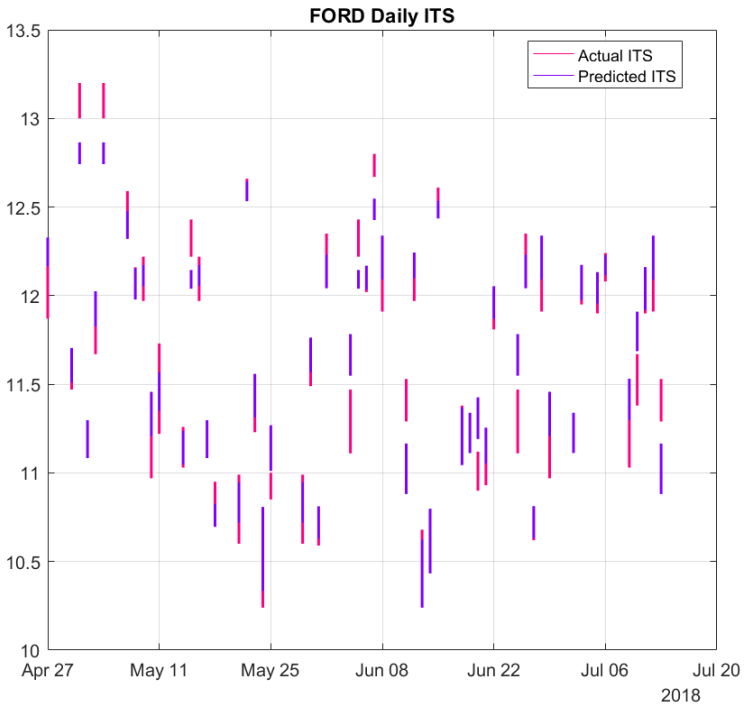


Figure 31. Predicted ITS and Actual series for 70-step-ahead forecast horizon of FORD Motor Company Daily price ITS.

Appendix A

Project Management

A.1. Work Description

The categorized tasks that have been carried out during the course of the project may be grouped under the following:

A.1.1. Study

Since the foundations developed in this area over the four years of electromechanical engineering have been rather weak, a thorough study of the state of the art had to be carried out in the initial stages of the project, such as:

- **Crisp time series** properties and analysis techniques, with concepts such as error and benchmarking, as well as linear analysis methods, namely exponential smoothing and ARIMA forecasting techniques.
- **Interval time series**, comprising the understading of the mathematical properties of interval-valued data, as well as its implications in terms of time series analysis and forecasting.
- **Nonlinear analytical methods**, namely k-nearest neighbors and specially artificial neural networks, that have been crucial in order to enable the implementation of the `iMLP Matlab`.

- **Matlab** programming environment. Although used in some classes along the university years, specific syntax had to be thoroughly understood.

A.1.2. Research

Building on the knowledge acquired in the initial studies, it was possible to investigate:

- New forecasting methods that have been recently proposed to apply intervals in the context of time series analysis. Namely symbolic data forecasting techniques such as de kNN method for intervals or the fuzzy-logic based algorithms.
- Implementations of neural networks in the context of interval time series, namely recurrent networks with fuzzy weights and the iMLP that resulted in the implementation of the tool of this project.

A.1.3. iMLP Matlab

With the goal of implementing a neural network model designed in `C` in a new programming language, `Matlab`:

- Understanding of the theoretical background around the model and data.
- Thorough study of the original method proposed in [57].
- Developing the code from scratch, designing the complete set of functions and subrutines to implement the model.

A.1.4. Conclusion

The work from this thesis had to be sorted and compiled, including:

- **Case studies** that were thought out during the research phase of the project but had to be modified and improved along the full project development.
- **Memoir** as the physical proof of the work behind the project. It introduces the project, presents its implications, limitations and discusses its conclusions.

Task	Description	Hours
Study	Crisp time series	20
	Interval time series	40
	Nonlinear analytical methods	30
	Matlab	40
Research	Forecasting techniques with ITS	30
	Neural network models in the context of ITS	50
iMLP Matlab	Design	50
	Programming	160
	Testing	30
Conclusion	Case Studies	50
	Memoir	190
	Total	**640**

Table 12. Horly breakdown of the project development.

Hourly breakdown Table 12 illustrates the hours dedicated to each specific task during this project.

A.2. Cost Analysis

A.2.1. Labor

Although throughout the course of the project, a six-month part-time internship was carried out, in terms of working hours, the project could have reported roughly 500 hours of available working hours.

Estimating an internship salary based on personal experience and benchmark analysis and estimating 500€ per month or roughly 6.25€ per hour. Considering the total amount of hours deployed to the project development, the labor cost associated can be estimated at **3125€**.

Description	Cost[€]
High-end Laptop	100
Matlab 2016b License	800
Wireless Mouse and Keyboard	5
Total	**1830€**

Table 13. Equipment-related costs of the project.

A.2.2. Material

Table 13 shows the equipment-related costs that have been assumed during the project. For the Matlab license, although included in the tuition fee, the cost has been accounted for as the public price of the package, since the university costs associated have not been considered for the cost analysis.

The equipment has been used for one year and it can be amortized by up to 25% yearly.

Additionally, considering the number of hours spent on the project, the equipment costs have been estimated to be roughly 10% of the total cost of the materials (laptop and accesories).

Accounting for the printing costs that had to be faced, a total of 80€ has been deployed to printing of drafts and final document.

Description	Cost [€]
Labor	3125
Material	905
Printing services	80
Total	**4110€**

Table 14. Total costs of the project development.

A.2.3. Total

Table 14 summarizes the total spending during the project.

Appendix B

Optimal Size for Hidden Layer: Number of Hidden Neurons

B.1. Structured trial and error method

The characteristics of this method is to do repeated attempts until results are optimum or until the last attempt of agent.

Forward Approach This method works as a from-bottom-to-top approach. Initially we take a small number of hidden neurons to train and validate the net. Then the number of hidden neurons is increased step by step while computing the training and validation. The process is repeated until in-sample and out-of-sample performance is optimized.

Backward Approach Analogous to the first method. It is now the opposite way of working: from a large number of hidden neurons until train and validation errors stop decreasing.

B.2. Rule of Thumb Methods

The number of neurons to be used in hidden layer is determined by this method as described below:

- The number of hidden neurons should not be neither greater in size than the input layer nor smaller than the output layer. Suppose the size of the input layer is 8 and the size of the output layer is 4, then the number hidden neurons must be within 4 and 8.
- The number of hidden neurons should be the sum of $\frac{2}{3}$ of the input layer size plus the size of the output layer. Say we have 9 input neurons and 2 output neurons then the number of hidden neurons should be 8.
- The number of hidden neurons should not be more than double the input layer size.

Bibliography

[1] **ADELI, H., HUNG, S.L. (1995)**. Machine Learning-Neural Networks, Genetic Algorithms, and Fuzzy System. *Circuits and Systems*, 1995, 7, 11.

[2] **ADELI, H., KARIM, A. (2000)**. Fuzzy-Wavelet RBFNN Model for freeway incident detection. *Journal of Transportation Engineering*, 2000, 126, 6.

[3] **AKAIKE, H.(1998)**. Information Theory and an Extension of the Maximum Likelihood Principle. *Kitagawa G. (eds) Selected Papers of Hirotugu Akaike. Springer Series in Statistics (Perspectives in Statistics). Springer, New York, NY.*

[4] **ALEFELD, G., MAYER, G. (2000)**. Interval analysis: theory and applications. *Journal of Computational and Applied Mathematics*, 2000, 121, 421-464.

[5] **ANDERSON, J.A. (1972)**. Simple neural network generating an interactive memory. *Mathematical Biosciences*, 1972, 14, 197-220.

[6] **ARBIB, M.A. (1989)**. The Metaphorical Brain 2: Neural Networks and Beyond 2nd Ed. *John Wiley & Sons, Inc. New York, NY, USA 1989.*

[7] **ARROYO, J. (2008)**. Metodos de prediccion para series temporales de intervalos e histogramas. *PhD thesis. Universidad Ponti*

cia Comillas, 2008.

[8] **ARROYO, J., MATÉ, C. (2006)**. Introducing interval time series: accuracy measures. *Compstat. 2006.*

[9] **ARROYO, J., MATÉ, C. (2009)**. Forecasting histogram time series with k-nearest neighbours methods. *International Journal of Forecasting*, 2009, 25, 192-207.

[10] **ARROYO, J., MUÑOZ, A., MATÉ, C., SARABIA, A. (2007)**. Exponential smoothing methods for interval time series. *Proceedings of Symposium, Espoo (Finland)*, 2007, 231-240.

[11] **BERTSEKAS, D.P., TSIKILIS, J.N. (2000)**. Gradient convergence in gradient methods with errors. *Society for Industrial and Applied Mathematics*, 2000, 10, 627-642.

[12] **AUER, P., HERBSTER, M., WARMUTH, M.K. (1996)**. Exponentially many local minima for single neurons. *Proceedings: Neural Information Processing Conference*, 1996, 316-317.

[13] **BILLARD, L., DIDAY, E. (2000)**. Regression Analysis for Interval-Valued Data. *Studies in Classification, Data Analysis, and Knowledge Organization*, 2000, 369-374.

[14] **BOWERMAN, B., O'CONNEL, R. (1990)**. Linear statistical models: an applied approach. 2nd Ed. *PWS-Kent Pub. Co., 1990*

[15] **BOX, G.E.P., JENKINS, G.M. (1970)**. Time series analysis: Forecasting and control. *San Francisco: Holden Day., 1970*

[16] **BOX, G.E.P., PIERCE, D.A. (1968)**. Distribution of Residual Autocorrelations in Integrated Autoregressive-Moving Average Time Series Models. *April, Madison: Department of Statistics, University of Wisconsin. Technical Report no.154.*

[17] **BROWN, R.G. (1959)**. Statistical forecasting for inventory control. *McGraw Hill,NY, 1959.*

[18] **CHA, S.-H. (2007)**. Comprehensive survey on distance and similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 2007, 1, 300-307.

[19] **CHEN, Z., YANG, Y. (2004)**. Assessing forecast accuracy measures. *Iowa State University*, 2004.

[20] **CHINN, M.D. (2011)**. Order flow and the monetary model of exchange rates: Evidence from a novel data set. *Journal of Money, Credit and Banking*, 2011, 43, 1599-1624.

[21] **COST, S., SALZBERG, S. (1993)**. A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning*, 1993, 10, 57-78.

[22] **DE CARVALHO, F.A.T. (1996)**. Histogrammes et indices de proximite en analyse de donnees symboliques. *Actes de l'ecole d'ete sur l'analyse des donnees symboliques. Universite de Paris IX - Dauphine, Paris, 1996.*

[23] **DICKEY, D.A., FULLER, W.A. (1979)**. Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *Journal of the American Statistical Association*, 1979, 74.

[24] **DUCHI et al. (2009)**. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 2011, 12, 2121?2159.

[25] **ELMAN, J.L. (1990)**. Finding Structure in Time. *Cognitive Science*, 1990, 14, 179-211.

[26] **EVERETT, J.E. (1997)**. Simulation to reduce variability in iron ore stockpiles. *Department of Information Management and Marketing, The University of Western Australia. Perth.*

[27] **GARDNER, E.S. (1985)**. Exponential smoothing: The state of the art. *Journal of Forecasting*, 1985, 4, 1-28.

[28] **GONZÁLEZ, L., VELASCO, F. (2004)**. Sobre núcleos, distancias y similitud entre intervalos. *Revista Iberoamericana de Inteligencia Articial*, 2004, 8, 111-117.

[29] **GOODWIN, P., LAWTON, R. (1999)**. On the asymmetry of the symmetric MAPE. *International Journal of Forecasting*, 1999, 15, 405-408.

[30] **HARTLEY, H.O. (1961)**. The Modified Gauss-Newton Method for the Fitting of Non-Linear Regression Functions by Least Squares. *Technometris*, 1961, 3.

[31] **HARVEY, R.L. (1994)**. Neural Network Principles. *Prentice Hall, NY, USA, 1994.*

[32] **HAWKINS, D.M. (2004)**. The Problem of Overfitting. *Journal of Chemical Information and Modeling*, 2004, 44, 1-12.

[33] **HAYKIN, S. (1994)**. Neural Networks: A Comprehensive Foundation. 1st Ed. *Prentice Hall, NJ, USA, 1994.*

[34] **HOLT, C. (1957)**. Forecasting Trends and Seasonal by Exponentially Weighted Averages. *International Journal of Forecasting*, 2004, 20, 513.

[35] **HOPFIELD, J.J. (1982)**. Neural networks and physical systems with emergent collective computational abilities. *National Academy of Sciences*, 1982, 79, 2554-2558.

[36] **HULL, J. (1995)**. Introduction to Futures and Options Markets. *Prentice Hall, New Jersey, 1995.*

[37] **ICHINO, M., YAGUCHI, H. (1994)**. Generalized Minkowski metrics for mixed feature-type data analysis. *IEEE Trans. on Systems, Man. and Cybernetics*, 1994, 24, 698-708.

[38] **ISHIBUCHI, H. et al. (1993)**. Fuzzy neural networks with fuzzy weights and fuzzy biases. *Proceedings of the I.C.N.N.*, 1993, 93, 1650-1655.

[39] **JORDAN, M. (1997)**. Serial Order: A Parallel Distributed Processing Approach. *Advances in Psychology. Neural-Network Models of Cognition*, 1997, 121, 471-495.

[40] **KARIM, A., ADELI, H. (2003)**. Radial Basis Function Neural Network for Work Zone Capacity and Queue Estimation. *Journal of Transportation Engineering*, 2000, 129, 5.

[41] **KESKAR, N.S., SOCHER, R. (2017)**. Improving Generalization Performance by Switching from Adam to SGD. *ResearchGate*, 2017.

[42] **KINGMA, D., BA, J. (2015)**. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR 2015)*, 2015.

[43] **KRIESEL, D. (2007)**. A Brief Introduction to Neural Networks. *Available at http://www.dkriesel.com*

[44] **LEVENBERG, K. (1944)**. A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quarterly of Applied Mathematics*, 1944, 2, 164-168.

[45] **OETIKER, T., PARTL, H., HYNA, I., SCHLEGL, E.**, *The Not So Short Introduction to LaTeX 2ε*,2009.

[46] **MACEDA, P. (2018)**. Basic Analytic System for Interval-Valued Data *Master Thesis Industrial Engineering. Pontifical Comillas University. Madrid*, 2018.

[47] **MAIA, S.A.L., CARVALHO, F., LUDERMIR, T.B. (2006)**. Symbolic interval time series forecasting using a hybrid model. *Symposium on Neural Networks*, 2006.

[48] **MAIA, S.A.L., CARVALHO, F., LUDERMIR, T.B. (2006)**. Forecasting models for interval-valued time series. *Neurocomputing*, 2008, 71, 3344-3352.

[49] **MAKRIDAKIS, S. (1993)**. Accuracy measures: theoretical and practical concerns. *International Journal of Forecasting*, 1993, 9, 527-529.

[50] **MANYIKA, J., CHUI, M., BROWN, B. et al. (2011)**. Big data: The next frontier for innovation, competition, and productivity. *McKinsey Global Institute. May 2011.*

[51] **MARQUADT, D. (1963)**. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *IAM Journal on Applied Mathematics*, 1963, 11, 431-441.

[52] **MATÉ, C. (2012)**. El análisis de intervalos. Aplicaciones en la ingeniería. *Anales de mecánica y electricidad*, 2012, 89, 3, 20-17.

[53] **MATÉ, C. (2014)**. Forecasting Exchange Rates: Past, Present and Future. *Exploratory Workshop on Business Forecasting. CUNEF.*, 2014.

[54] **MINSKY, M., PAPERT, S. (1969)**. Perceptrons: an introduction to computational geometry. *MIT Press, MA, USA, 1969.*

[55] **MOORE, R.E. (1966)**. Interval Analysis. *Prentice-Hall, New York, 1966.*

[56] **MOORE, R.E. (1979)**. Methods and Applications of Interval Analysis. *SIAM, PA, 1966.*

[57] **MUÑOZ, A., MATÉ, C., ARROYO, J., SARABIA, A. (2007)**. iMLP: applying multilayer perceptrons to interval-valued data *Neural Processing Letters*, 2007, 25, 157-169.

[58] **NETO, E.A.L., DE CARVALHO, F.A.T. (2008)**. Centre and Range method for fitting a linear regression model to symbolic interval data. *Computational Statistics & Data Analysis*, 2008, 52, 1500-1515.

[59] **NETO, E.A.L., DE CARVALHO, F.A.T. (2017)**. Nonlinear regression applied to interval-valued data. *Journal Pattern Analysis & Applications*, 2017, 20, 809-824.

[60] **NIKULIN, M.S. (1994)**. Hellinger distance. *Encyclopedia of Mathematics*, Springer Science+Business Media B.V. / Kluwer Academic Publishers 1994.

[61] **NOCEDAL, J., WRIGHT, S. (2006)**. Numerical optimization. *Springer Science & Business Media, 2006.*

[62] **NOGALES, L.M. (2011)**. Sistema de predicción del ciclo de negocio basado en datos de intervalo. *Master's thesis. Universidad Ponti*

cia Comillas, 2011.

[63] **OZER, D. (1985)**. Correlation and the coefficient of determination. *Psychological Bulletin*, 1985, 97, 307-315.

[64] **PEIRSON, G. et al. (1995)**. Business Finance. *McGraw-Hill, Sydney, 1995.*

[65] **PINEDA, F.J. (1987)**. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters,*, 1987, 59, 2229-2232.

[66] **POGGIO, T., GIROSI, F., JONES, M. (1995)**. Regularization theory and neural networks architectures. *MIT Press, MA, USA, 1995.*

[67] **POLLARD, D. (2000)**. Asymptopia. *Unpublished book.*

[68] **RAUDYS, S. (1998)**. Evolution and generalization of a single neurone: I. Single-layer perceptron as seven statistical classifiers. *Neural Networks*, 1998, 11, 283-296.

[69] **REDONDO, J. (2013)**. Interval Time Series Analysis and Forecasting. *Universidad Pontificia de Comillas*, 2013.

[70] **ROBBINS, H., MONRO, S. (1951)**. A stochastic approximation method. *The annals of mathematical statistics*, 1951, 400-407.

[71] **RON, K. (1995)**. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1995, 2, 1137-1143.

[72] **ROSENBLATT, F. (1958)**. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958, 65, 386-408.

[73] **ROSENBLATT, F. (1962)**. Principles of Neurodynamics. *Spartan, New York, 1962.*

[74] **ROSSI, B. (2013)**. Exchange rate predictability. *Journal of Economic Literature*, 2013, 51, 1063-1119.

[75] **RUCK, D.W., ROGERS, S.K., KABRISKY, M., OXLEY, M.E. et al. (1990)**. The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1990, 1, 4.

[76] **SCHWARZ, T. (1978)**. Information criteria for discriminating among alternative regression models. *Econometrica*, 1979, 74.

[77] **SINOVA, B., COLUBI, A., GIL, M.A., GONZÁLEZ-RODRÍGUEZ, G. (2012)**. Interval arithmetic-based simple linear regression between interval data: discussion and sensitivity analysis on the choice of the metric. *Information Sciences*, 2012, 199, 109-124.

[78] **SUNAGA, T. (1958)**. Theory of an interval algebra and its application to numerical analysis. *Japan Journal of Industrial and Applied Mathematics*, 1958, 26, 125-143.

[79] **SUTTON, R.S., BARTO, A.G. (1998)**. Reinforcement Learning: An Introduction. *MIT Press, Cambridge, MA, 1998.*

[80] **THEIL, H. (1966)**. Applied Economic Forecasts. *Amsterdam, North Holland, 1966.*

[81] **TIBSHIRANI, R. (1996)**. Regression Shrinkage and Selection via the lasso. *Journal of the Royal Statistical Society*, 1996, 58, 267-288.

[82] **TIELEMAN, T., HINTON, G. (2012)**. Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012, 4.

[83] **TIKHONOV, A.N. (1943)**. On the stability of inverse problems. *Doklady Akademii Nauk SSSR.*, 1943, 39, 195-198.

[84] **TIKHONOV, A.N. (1963)**. Solution of incorrectly formulated problems and the regularization method. *Doklady Akademii Nauk SSSR.*, 1963, 4, 1035-1038.

[85] **ULLA, A., GILES, D. (2011)**. Handbook of empirical economics and finance. 1st Ed. *CRC Press, 2011.*

[86] **WEI, W. (2006)**. Time Series Analysis?Univariate and Multivariate Methods,2nd Ed. *Boston, MA: Pearson Addison-Wesley, 2006.*

[87] **WEI, W. (2013)**. The Oxford Handbook of Quantitative Methods in Psychology, Vol. 2: Statistical Analysis. *OUP USA, 2013.*

[88] **WERBOS, P.J. (1988)**. Backpropagation: Past and future. *Proceedings ICNN-88, San Diego*, 1988, 343-353.

[89] **WINTERS, P.R. (1960)**. Forecasting sales by exponentially weighted moving averages. *Management Science*, 1960, 6, 324-342.

[90] **WORKING, H. (1934)**. A Random-Difference Series for Use in the Analysis of Time Series. *Journal of the American Statistical Association.*, 1934.

[91] **YAKOWITZ, S. (1987)**. Nearest-neighbour methods for time series analysis. *Journal of Time Series Analysis.*, 1987, 8, 235-247.

[92] **ZELLNER, A., TOBIAS, J. (2000)**. A note on aggregation, disaggregation and forecasting performance. *Journal of Forecasting*, 2000, 19, 457-265.

[93] **Universidad Pontificia Comillas**, *Página web de Proyectos Fin de Carrera.* http://www.iit.upcomillas.es/pfc