Università
Ca'Foscari
Venezia

Corso di Laurea magistrale (*ordinamento ex D.M. 270/2004*)
in Amministrazione, Finanza e Controllo

Tesi di Laurea

Ca' Foscari
Dorsoduro 3246
30123 Venezia

# Forecasting Energy Market:
## an Artificial Neural Network Approach

**Relatore**
Prof.ssa Diana Barro

**Co-relatore**
Prof. Pianca Paolo

**Laureando**
Ugo Dario
Matricola 835313

**Anno Accademico**
**2013 / 2014**

# Forecasting energy market: an artificial neural network approach

Ugo Dario

February 12, 2015

# Contents

# Chapter 1

# Neural Network

## 1.1 Introduction

Artificial Neural Networks (ANNs) could be considered as a class of mathematical algorithms that can reach a solution to a number of specific problems. The name "Neural Network" is given because these "class of algorithms" try to emulate the information processing capabilities of the human brain.

The brain is composed of a large number of computing elements (approximately $10^{11}$) called neurons. The neurons are highly connected to (approximately $10^4$ connections per neuron) each other forming a network. The neuron has three principal components: the dendrites, the cell body and the axon. The dendrites are the receptive parts which carry electrical signals into the cell body. The cell body sums the incoming signals and produces an output impulse. The axon carries this impulse to other neurons, and the contact between an axon of one cell and a dendrite of another cell is called a synapse [34].

The structure of artificial neural network reproduce the architecture of the human brain. It has input vectors (dendrites) that are summarized into a transfer function (cell body) which produces an output (axon). As in the human brain, where the strengths of the individual synapses base the function of the neural network, in ANN the input vectors are multiplied by *weights*. These *weights* correspond to the synapses' strength.

ANN are used for many different scopes, that can be divided in four main classes of application :

1. *Clustering*: the neural network divides input data by similarity. For example: market segmentation done by grouping people according to their buying patterns.

2. *Fitting*: the neural network map a data set of numeric inputs with a set of numeric targets. For example: estimating engine emission levels based on measurements of fuel consumption and speed; or predicting a patient's body-fat level based on body measurements.

3. *Pattern Recognition*: the neural network classifies input data into a set of target categories. It could be used for example in a tumor classification (malignant or benign) based on the information of the cell size, clump thickness, mitosis ecc.

4. *Simulation*: past values (as input) are used to predict future values (as output). For example: predict the future value of a stock market; or simulating a dynamic model based on an input data.

Even if some processing tasks could be achieved by traditional techniques (as regression analysis), the major advantage of the neural networks are:

- their ability to ***generalise*** any kind of function;
- their ***flexibility*** in mapping different input data with the target.

Generalization and flexibility make neural networks a "universal approximator" that can well approximate any arbitrary function. And even if ANNs could be used in linear problems (without particular advantages), they are superior of other techniques in nonlinear modelling.

There are different neural network architectural designs, depending on the singular problem, but in general it is possible to divide them into three main categories: Supervised Learning, Reinforcement Learning, Unsupervised Learning.

**Supervised Learning:** the output targets are known; during the training process the weights of the inputs data are adjusted in order to reduce the

error between output and target. The most used example is a *multilayer perceptron* architecture.

**Reinforcement Learning:** the targets are not known during the training process and only a series of *"measures"* are used in the feedback to lead the network performance. An example is the driven robot designs.

**Unsupervised Learning:** there is total lack of feedback. The network self-discover some classes and organize the input pattern into categories. It is used to identify groups of data, and the most common examples of architecture is the *Kohonen Self-Organising Maps*.

The purpose of this paper will specifically treat *Supervised Learning* architectures; and multilayer perceptron will be used for the forecasting in the financial market.

## 1.2 Neuron Model

Neural network structure is formed from input data, a transfer function and an output. Figure 1.1 shows a simple-input neuron: $p$ is a scalar input, $w$

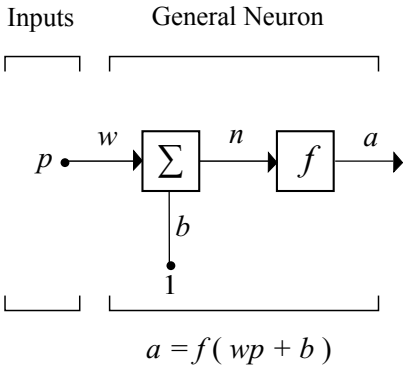



Figure 1.1: Simple-Input Neuron

the weight, $f$ the transfer function and $a$ the output; $n$ is the element of the function. Scalar input $p$ is multiplied by the scalar weight $w$ to form $wp$. This is the argument of the transfer function $f$, which produces the scalar output $a$. Typically the neuron has a bias $b$, an *adjustable* parameter which permits the network to be more flexible[1]. The *bias* can be considered as a weight $w$ with a constant input of 1. It is added to the product $wp$ as an argument of the transfer function.

We can consider the neuron as a linear equation $Y = \beta_0 + \beta_1 X$ where our input data ($p$) is the $X$ that is multiplied by the parameter $\beta_1$ (the *weight*) and adjusted by parameter $\beta_0$ (the *bias*) which gives us an output $Y$. We can call $n$ the input argument of the function; in Figure1.1 we can consider $n$ as: $n = wp + b$. The *transfer function* is chosen by the designer of the net according to the specific goal: there are different transfer functions for different purposes.

Figure 1.2 shows a neuron with $R - elements$ input vector: $\mathbf{p}$ is the vector of the $R - elements$; $\mathbf{W}$ is the matrix (single row); $n$ is the net input formed by the weighted input and a *bias*, and it is a scalar. The neuron output $a$ can be

[1] It shifts the function $f$ to the left by an amount $b$.Neuron without a *bias* will always have a net input $n$ of zero when the network inputs $p$ are zero. This may not be desirable and can be avoided by the use of a bias. (Hagan,1996)

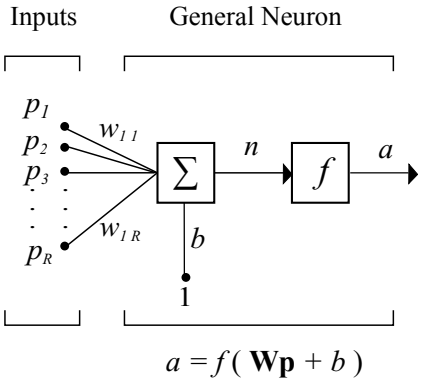



Figure 1.2: Muliple-Input Neuron

written as:

$$a = f(\mathbf{W}\mathbf{p} + b)$$

Figure 1.2 is what we can consider a neuron. But in many problems just one neuron is not sufficient. For this reason it is common to create a *"layer of neurons"* which operate in parallel. This architecture is shown in Figure 1.3.

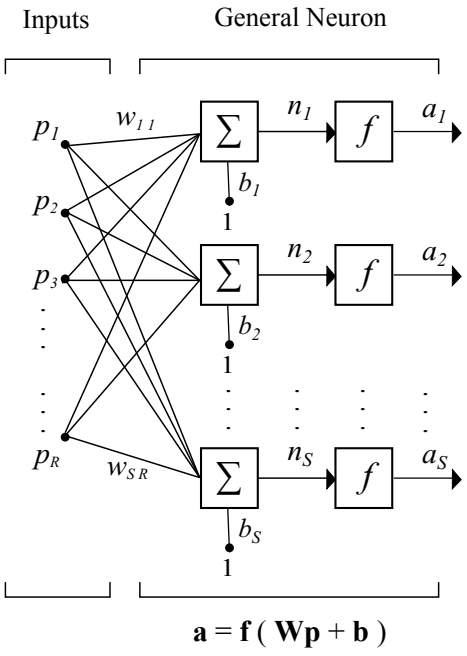



Figure 1.3: Leyer of Neurons

There are $R$ input elements connected to each neuron input through the weight

matrix **W**. There are $S$ neurons and each $i$th neuron sums the weighted input elements with its bias. All the $n(i)$ scalar elements form a net input vector **n** of $S - elements$ and **a** is the column vector of the S neuron layer outputs. It is common to have $R \neq S$ : the number of input elements is usually different from the number of neurons.

We can write the output **a** as:

$$\mathbf{a} = f(\mathbf{W}\mathbf{p} + \mathbf{b})$$

where **W** is the matrix:

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,R} \\ w_{2,1} & w_{2,2} & \dots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \dots & w_{S,R} \end{bmatrix}$$

The neuron's transfer function does not have to be the same for all the neurons. We can have different transfer functions for different neurons' layers, working in parallel.

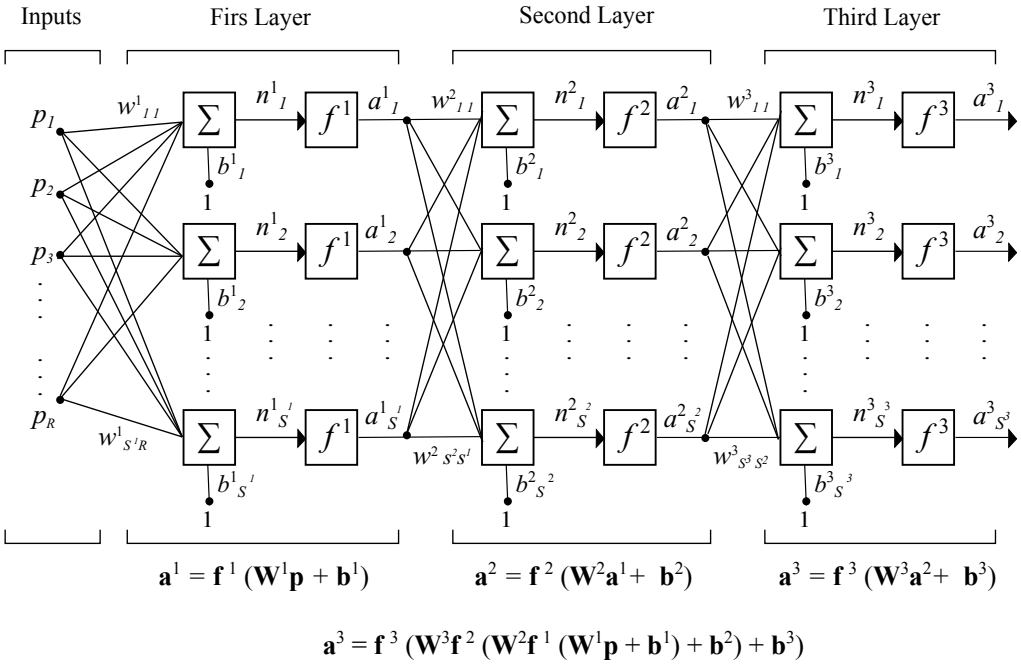
Now we can combine different neurons' layers (Figure 1.3) together to create Multiple Layers of Neurons. Each layer has: its own weight matrix **W**, bias vector **b**, net input **n**, output vector **a**. (We do not consider the input as a layer). We call *output layer* the layer that has its output as the network output. All other layers are called *hidden layers*.

Figure 1.4 shows a multiple layers network with one output layer and two hidden layers. Different layers can have different numbers of neurons.

We use multilayer networks because it is more powerful than single-layer networks.

## 1.3 Learning Rule (Training Algorithm)

As previously said there are different neural architectures: supervised learning, unsupervised learning and reinforcement learning; determined by problem specifications. In order to perform its proper task all problems need to determine the weights and biases values so as to minimize the difference between the output network and its task. To reach this goal, the network has to




$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3)$$

Figure 1.4: Multiple Leyers of Neurons

use an algorithm which modify the weights and biases after each interaction; and different algorithms exist for different purposes. The process that is going to modify the weights and biases values is known as *learning rule* or *training algorithm*.

In supervised learning, targets are given with a set of examples:

$$(p_1, t_1), (p_2, t_2), \ldots, (p_Q, t_Q)$$

where $p_q$ is an input value and $t_q$ is the corresponding target. The learning rule is used here to adjust weights and biases values in order to reduce the error between targets and network outputs:

$$e = (t - a)$$

In Reinforcement learning, we do not have the correct target value, so the learning rule gives only a grade (*measure*) that is used as parameter of network performance.

In Unsupervised learning, where we do not have a target, the learning rule is quite different: most of these algorithms work as a clustering operation; weights and biases are modified in answer only to its inputs.

The most common learning rule use in the supervised learning multilayer network is the backpropagation algorithm.

## 1.4 Backpropagation

The standard backpropagation (BP) algorithm, also known as *Error backpropagation algorithm* or *Generalizes Delta Learning Rule*, is a gradient descent method that is nothing other that a generalization of the least mean square (LMS) algorithm of Bernard Windrow and Marcian Hoff. It is used to reduce the error between the target value $t$ and the network output $a$. To do this, backpropagation algorithm uses, as in LMS algorithm, the mean square error (MSE) as performance index:

$$E_k = \frac{1}{2}\sum_{i=1}^{S}(t_{ki} - a_{ki})^2 = \frac{1}{2}\|\mathbf{t}_k - \mathbf{a}_k\|^2 \tag{1.1}$$

where $t_i$ is the desired output and $a_i$ its related $i$th network output, at the iteration $k$. [2]

To reduce the error function $E$ the BP adjusts the value of the weights, and biases, along the negative of the gradient of the performance function $E$. The problem can be viewed as an optimal control problem, where the weight vector at each stage is the control vector for that stage [11]. Gradient descent algorithm (also knows as steepest descent algorithm) can be written:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \mathbf{g}_k \tag{1.2}$$

where $\mathbf{x}_k$ is the vector of current weight and bias, $\mathbf{g}_k$ is the current gradient and $\eta_k$ is the learning rate parameter which control algorithm convergence and has small values between $0 < \eta < 1$. In neural networks we can write the eq. (1.2) in two components that define the new values of weights and biases at the iteration $(k+1)$ as :

$$\mathbf{W}(k+1) = \mathbf{W}(k) - \eta_k \nabla E_w \tag{1.3}$$

and

$$\mathbf{b}(k+1) = \mathbf{b}(k) - \eta_k \nabla E_b \tag{1.4}$$

[2] Note that $k$ refers to a specific pattern that is at the input and produces the output error. There are two mode for update the weights and biases: iterative mode and batch mode. We will see them in Chapter 3.

where $\nabla E_w$ is the vector of partial derivative of the function $E$, to minimize, with respect to weights:

$$\nabla E_w = \frac{\partial E}{\partial \mathbf{W}} \tag{1.5}$$

and $\nabla E_b$ is the partial derivative of the function $E$ with respect to biases:

$$\nabla E_b = \frac{\partial E}{\partial \mathbf{b}} \tag{1.6}$$

At each iteration the error $E_{k+1}$ has to be compared with $E(max)$: the maximum acceptable value error chosen. For $E_{k+1} \leq E(max)$ the training session is completed and the values of weights and bias will be $\mathbf{W}(k+1)$ and $\mathbf{b}(k+1)$; for $E_{k+1} > E(max)$ a new training cycle will be initialized.
In a single layer of neurons (see Figure 1.3) we know that the output of the $i$th neuron is:

$$\mathbf{a} = a_i = f(n_i) \tag{1.7}$$

where:

$$n_i = \sum_{j=1}^{R} w_{ij} p_j + b_i \tag{1.8}$$

for $j = 1, 2, ..., R$ and $i = 1, 2, ..., S$.

In order to reduce function error in eq.(1.1), we need to calculate the partial derivative of the error with respect to both weights and biases. Because the error is an indirect function of the weight, the chain rule will be used. If we have a function $f$ that is explicit function of $g$, which is itself a function of a variable $x$, then $f$ is also a function of $x$. Formally chain rule is written as:

$$\frac{df(g(x))}{d(x)} = \frac{df(g)}{dg} \cdot \frac{dg(x)}{dx}$$

If we apply the chain rule in the partial derivatives of eq. (1.5) and eq. (1.6) we have:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial n_i} \cdot \frac{\partial n_i}{\partial w_{ij}} \tag{1.9}$$

$$\frac{\partial E}{\partial b_i} = \frac{\partial E}{n_i} \cdot \frac{\partial n_i}{\partial b_i} \tag{1.10}$$

The second term in (1.9) is:

$$\frac{\partial n_i}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} (\sum_{j=1}^{R} w_{ij} p_j + b_i) = p_j \tag{1.11}$$

Regarding the first term, we note that $E$ is a composite function of $n$, so that we can write function $E$ as:

$$E_i = E(a_i) = E(f(n_i)) \tag{1.12}$$

applying the chain rule again, the first term of eq.(1.9) becomes:

$$\frac{\partial E}{\partial n_i} = \frac{\partial E}{\partial a_i} \cdot \frac{\partial a_i}{\partial n_i} \tag{1.13}$$

where the second term is the derivatives of the function with respect to its variable:

$$\frac{\partial a_i}{\partial n_i} = \frac{\partial f(n_i)}{\partial n_i} = f'(n_i) \tag{1.14}$$

and the first term is equal to:

$$\frac{\partial E}{\partial a_i} = -(t_i - a_i) \tag{1.15}$$

We define the partial derivative $\partial E/\partial n_i$ in eq.(1.13) the *error signal term* $\delta$:

$$\delta_{ai} = \frac{\partial E}{\partial n_i} = -(t_i - a_i) f'(n_i) \tag{1.16}$$

The eq.(1.9) can be rewritten as:

$$\frac{\partial E}{\partial w_{ij}} = -(t_i - a_i) f'(n_i) p_j = \delta_{ai} p_j \tag{1.17}$$

Using the same method eq. (1.10) can be written as:

$$\frac{\partial E}{\partial b_i} = -(t_i - a_i) 1 = \delta_{ai} \tag{1.18}$$

In a single layer of neurons we can rewrite eq.(1.3) and eq.(1.4), to update the weights and biases values at iteration $(k+1)$, as:

$$\mathbf{W}(k+1) = \mathbf{W}(k) - \eta \boldsymbol{\delta}_a \mathbf{p} \tag{1.19}$$

and

$$\mathbf{b}(k+1) = \mathbf{b}(k) - \eta \boldsymbol{\delta}_a \tag{1.20}$$

These are the final formulas for weights and biases adjustment of a network with a single layer of neurons; where $\eta$ is the learning rate and $\boldsymbol{\delta}_a$ is defined as:

$$\boldsymbol{\delta}_a = \begin{bmatrix} \delta_{a1} \\ \delta_{a2} \\ \vdots \\ \delta_{aS} \end{bmatrix}$$

But in a multilayer network, with $M$ layers, the outputs of the first layer become the input of the second layer; the output of the second layer become the input of the third layer, and so on until the $M$th layer. If we use a *superscript* to identify the layers, we can write eq.(1.7) as:

$$\mathbf{a}^m = f(\mathbf{n}^m) = f(\mathbf{W}^m\mathbf{a}^{m-1} + \mathbf{b}^m) \qquad \text{for} \quad m = 1, 2, ..., M$$

where

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{ij}^m a_j^{m-1} + b_i^m$$

with $i$ as the number of the neurons output of layer $m$ and $j$ as the number of the neurons input of layer $m$ (or number of output of layer neurons $m-1$). Thus, we can write:

$$\mathbf{a}^0 = \mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{bmatrix} \quad ; \quad \mathbf{a} = \mathbf{a}^M$$

where $\mathbf{p}$ are the external inputs and $\mathbf{a}$ are the network outputs. In order to minimize the function error $E$ we will use partial derivative as in eq.(1.9) and eq.(1.10), which in a multilayer network becomes:

$$\frac{\partial E}{\partial w_{ij}^m} = \frac{\partial E}{\partial n_i^m} \cdot \frac{\partial n_i^m}{\partial w_{ij}^m} \tag{1.21}$$

and

$$\frac{\partial E}{\partial b_i^m} = \frac{\partial E}{\partial n_i^m} \cdot \frac{\partial n_i^m}{\partial b_i^m} \tag{1.22}$$

The second term of the eq.(1.21) is:

$$\frac{\partial n_i^m}{\partial w_{ij}^m} = \frac{\partial}{\partial w_{ij}^m}\left(\sum_{j=1}^{S^{m-1}} w_{ij}^m a_j^{m-1} + b_i^m\right) = a_j^{m-1} \tag{1.23}$$

The first term is the *error signal term* $\delta$ we defined in (1.16) that, in multilayer networks, can be written as:

$$\delta_{ai^m} = \frac{\partial E}{\partial n_i^m} = \frac{\partial E}{\partial a_i^m} \cdot \frac{\partial a_i^m}{\partial n_i^m} \tag{1.24}$$

Where:

$$\frac{\partial a_i^m}{\partial n_i^m} = \frac{\partial f(n_i^m)}{\partial n_i^m} = f'(n_i^m) \tag{1.25}$$

and with the first terms as:

$$\frac{\partial E}{\partial a_i^m} = \frac{\partial}{\partial a_i^m} \quad \left[\frac{1}{2}\sum_{i=1}^{S^M}(t_i - a_i)^2\right] = -\sum_{i=1}^{S^M}(t_i - a_i)\cdot\frac{\partial f(\mathbf{n}^M)}{\partial a_i^m} \tag{1.26}$$

It now remain to compute the partial derivative of $\mathbf{a}= \mathbf{a}^M = f(\mathbf{n}^M)$, and it is this procedure that gives the term of *backpropagation*.In fact the *error signal term* $\delta_a$ we defined in (1.24) has a recurrence relationship in which the $\delta_{a^m}$ at layer $m$ is computed from the $\delta_{a^{m-1}}$ at layer $m-1$. In other words we have to start from the $\delta_{a^M}$ at the last layer through the $\delta_a$ of the fist layer. We can write the network output as:

$$\mathbf{a}^M = f(\mathbf{n}^M) = f(\mathbf{W}^M\mathbf{a}^{M-1} + \mathbf{b}^M)$$

Thus, if we start from the last layer, the second term of eq. (1.26) is:

$$\frac{\partial f(\mathbf{n}^M)}{\partial \mathbf{a}^{M-1}} = \frac{\partial f\left(\sum_{i=1}^{S^M} w_{ij}^M a_j^{M-1} + b_i^M\right)}{\partial a_j^{M-1}} = f'(n_i^M)\cdot w_{ij}^M \tag{1.27}$$

The eq. (1.26) for the final layer can now be written as:

$$\frac{\partial E}{\partial a_j^{M-1}} = \frac{\partial E}{\partial a_i^M}\cdot\frac{\partial a_i^M}{\partial n_i^M}\cdot\frac{\partial n_i^M}{\partial a_j^{M-1}} = -(t_i - a_i)f'(n_i^M)\cdot w_{ij}^M \tag{1.28}$$

where $i$ as the network output and $j$ as the input of the last layer. Recall the *error signal term* at the $M$-th layer defined in eq.(1.16), we can write the error signal term $\delta$ at the neurons layer $M-1$ as:

$$\delta_{a^{M-1}} = \frac{\partial E}{\partial n_i^{M-1}} = -(t_i - a_i)f'(n_i^M)w_{ij}^M\cdot f'(n_j^{M-1}) = \delta_{a^M}\cdot w_{ij}^M\cdot f'(n_j^{M-1}) \tag{1.29}$$

Using vector notation we can generalize the above expression as:

$$\delta_{\mathbf{a}}^M = -(\mathbf{t} - \mathbf{a})\mathbf{F}'(\mathbf{n}^M) \tag{1.30}$$

and

$$\delta_{\mathbf{a}}^m = \mathbf{F}'(\mathbf{n}^m)\cdot(\mathbf{W}^{m+1})^T\cdot\delta_{\mathbf{a}}^{m+1} \tag{1.31}$$

for $m = (M-1),(M-2),...,3,2,1$. Where $\mathbf{F'}(\mathbf{n}^m)$ is the diagonal matrix of derivative of the functions at the neurons layer $m$, define as:

$$\mathbf{F}'(\mathbf{n}^m) = \begin{bmatrix} f'(n_1^m) & 0 & \dots & 0 \\ 0 & f'(n_2^m) & 0\dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & f'(n_{S^m}^m) \end{bmatrix} \tag{1.32}$$

and $\mathbf{W}^{m+1}$ is the matrix of weights $w_{ij}$, with $j$ as input layer and $i$ as output layer, at the neurons layer $m+1$. Finally, using eq. (1.23) and eq. (1.31), we can rewrite eq. (1.3) as:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \eta \delta^m_{\mathbf{a}} \cdot (\mathbf{a}^{m-1})^T \tag{1.33}$$

This is the final formula for update weights in a multiple layers network using the approximate steepest descent algorithm.

For updating the biases the process is the same; change only the second term of the eq. (1.22): $\partial n^m_i / \partial b^m_i$ that is equal to 1. The final formula for updating biases becomes:

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \eta \delta^m_{\mathbf{a}} \tag{1.34}$$

# Chapter 2

# Energy Commodities Analysis

## 2.1 An overview

Continuous population growing and general expansion of economies are increasing global energy demand. But at the same time the global warming problem on one hand and world oil shortage on the other hand are leading to different energy resources. Energy market in this term is becoming increasingly important. Even if oil still remains the world's leading fuel with 32,9% of global energy consumption in 2013, it continued to lose market share in place of other energy sources for the fourteenth consecutive year [12]. According with *BP Statistical Review of World Energy* in the 2013 the *"consumption and production increase for all fuel, reaching record level for every fuel type except nuclear power"*. After the accident in Japan on 11 March 2011, in fact, the opportunities for new development of nuclear power have been weakened and many countries have pushed to bring forward the shutdown. This has allowed an increase in other energy sources, as renewables and natural gas. The latter is especially taking an increasingly important role. Cleaner than oil and coal, natural gas accounted for 23,7% of primary global energy consumption in 2013. In the same year gas consumption grew globally by 1,4% and gas production rose by 1,1% [12]. *LNG* (Liquefied Natural Gas) and *shale gas* are going to play significant roles in the energy market. Because of its abundance, clean burning properties and efficiency as a fuel can be used for multiple purposes. LNG is

expected to play an increasingly important role in natural gas industries and energy markets in the next several years [49]. Kumar et al. (2011) provided an overview on demands and supplies of natural gas LNG and it concluded that by the year 2030 demand of LNG will be three times the actual one.
LNG supply is projected to grow 4,4% p.a. to 2030 and the demand is driven by Europe (5,2% p.a.) and non-OECD Asia countries (8.2% p.a.) [87]. The LNG imports in Asia rose five times in the last decades and natural gas accounts for 10% of the primary energy consumption, compared to the 24% rise in the world [49]. Most interesting is the US energy situation and its reflection for future forecasting. The US has continued research in new technology and in the optimization process on the non-traditional or unconventional gas reservoirs, as shale gas, tight gas and coal bed methane (CBM). Accounting for only one per cent of the US gas output in 2000, now shale gas represents 20% and a potential of 50% by 2035 is forecast [15].
Development on shale gas in the US is due in part to the substantial and inexpensive drilling capacity and in part to the need respond to the decline of conventional gas production [14].
Also the development of tight gas has been very successful and entire US conventional and unconventional gas productions can still provide for nearly 90% of the total US gas demand. The remainder is covered by Canadian and Mexican pipelines.

Furthermore, shale gas represents a huge opportunity in several different countries and it could be an answer to future energy supply. EIA (U.S. Energy Information Administration) has estimated in 2013, shale gas reserves in Europe for a total of 639 trillion cubic feet, compared with 862 trillion cubic feet of America [81]. Figure 2.1 shows estimated reserves areas in the world in 2013 by EIA.

But different reasons make shale gas controversial. In Europe diverse geological characteristics, high population density and more expensive costs of extraction activity (likely to be three and a half times more than the one in the US [60]) have dampened forward development. Alert on drinking water contaminated by chemicals used on the fracturing process and the uncertainty about potential augmentation on earthquakes on the fractured area are some

of the principal reasons that have led to interrupt the use and the development on shale gas in different countries. Today only the US, Canada and China produce shale gas in commercial quantities [82].

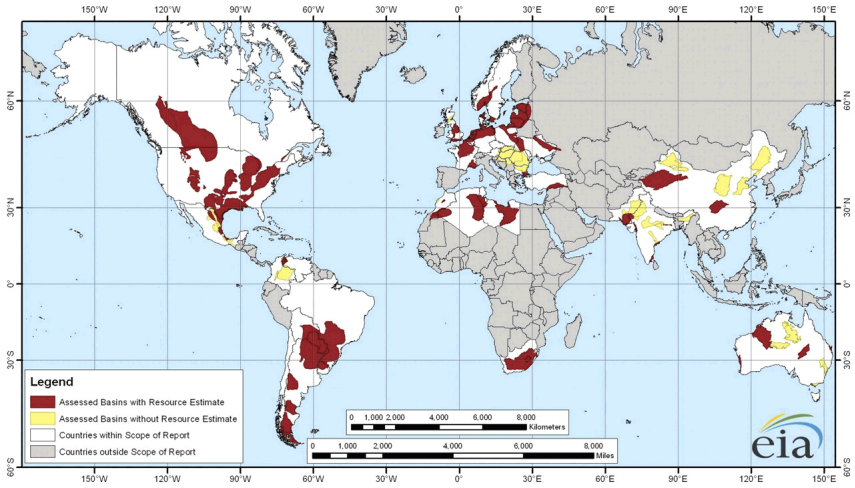



Figure 2.1: Shale Gas Map Reserves (Source:[84])

Gas competes mainly with coal in the current energy market: weaker gas prices usually lead to lower demand for coal [69] [83]. From *BP Statistical Review of World Energy* in the 2013, coal consumption grew by 3% in 2013 and it is still the fastest-growing fossil fuel. Its share in global energy primary consumption reached 30,1%, the highest since 1970. However price declined globally for a second consecutive year (Figure 2.2) [12].

This quick overview on the actual energy market permit us to better understand the complex and diverse dynamics that exist on the energy market. Obviously, it does not give an exhaustive view on the whole energy market situation in the different world regions but this is beyond the scope of this paper. However, it can helps in better understanding commodities' price dynamics in the energy market. The energy market is not only characterised by the natural reserves availability and technological developments, but also by both national and international political decisions taken. The global commitment on the $CO_2$ reduction, the political decision on nuclear power-use in the place of renewable energy, or to develop a particular technology on shale gas rather than another one, gives us an idea of how many different variables can

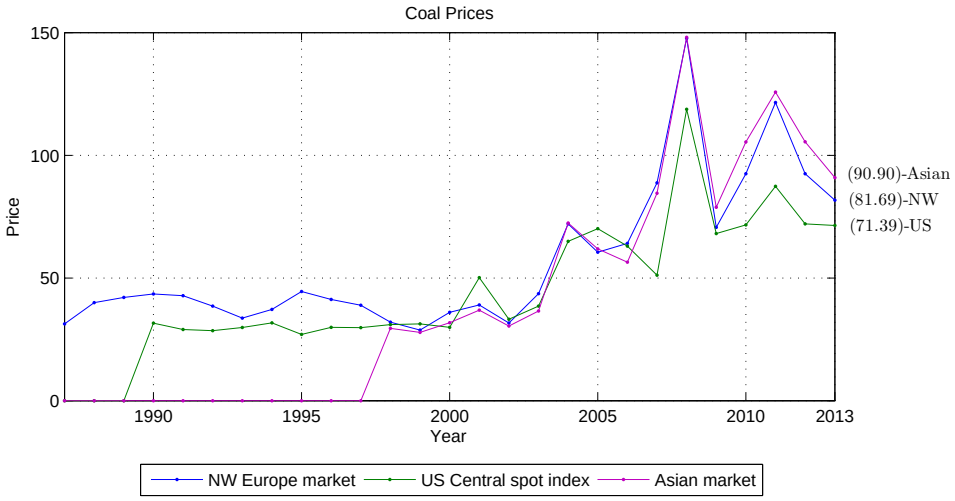



Figure 2.2: Coal Prices

play significant roles in this market.

Moreover, it has to be considered that some countries are energy-dependent from others, at least for the time-being. In fact new power resources (as it is the case for the shale gas in US) or new efficient ways on commodity transport (as it is LNG for Europe) can further change the actual behaviour on this market. And all these aspects will be reflected in the prices.

## 2.2 Price Analysis

In this paper, crude oil and natural gas prices will be analysed in order to do forecasting.

Oil still remains the greatest important commodity since it is still the most significant production factor of many economies [84]. Crude oil has different qualities: it can change in density and consistency, it can be very thin and lightweight or thick and heavy, or semi-solid. The major international benchmarks are Brent (from North Sea) and West Texas Intermediate (WTI). It also due to their high quality compared with OPEC Basket oils [1]. Figure 2.2 shows the

[1]Organization of the Petroleum Exporting Countries (OPEC), formed in 1960, was created in order to better negotiate with oil companies about production, price and license. The objective is price stability for petroleum producers. The member Countries are: Iran, Iraq, Kuwait, Saudi

major trade crude oil movements in the world in 2013 [12]

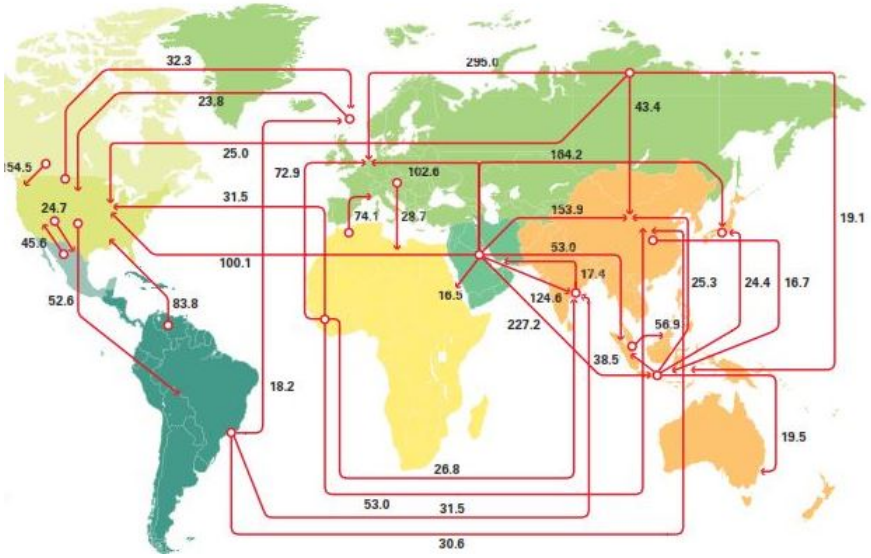



Figure 2.3: Map of crude oil trade in the world. (Source[81])

Usually Brent and WTI oil coupled in prices: until 2009 the average spread was around ±$3, but since the autumn of 2010 Brent has started to be higher. Figure 2.2 shows how the spread between WTI and Brent prices decoupled in the last 3 years. It reaches the maximum divergence in August 2012 of $23 and since 2011 the average discount of WTI to Brent was of $14.81. Only in 2013 the annual average price for Brent declined for the first time since 2009, and the discount on WTI fell to $10.67 on average [12]. US Energy Information Administration attributes the price spread to crude oil oversupply from North America.

The crude oil trends affect financial markets, industrial markets and government decisions through its impact on macroeconomic variables [32]. Its importance has grown in the last several years due to its high price variation, e.g. Brent crude oil price, after it had reached a historical record with a monthly average of 134 US dollars per barrel in July 2008, dropped down to 43 US

Arabia, Venezuela (that were the Founder Members of the Organization), later joined by Qatar, Indonesia, Libya, the United Arab Emirates, Algeria, Nigeria, Ecuador, Gabon and Angola. Today the Organization has 12 Members, after which Gabon (in 1995) and Indonesia (in 2009) suspended their membership [63]

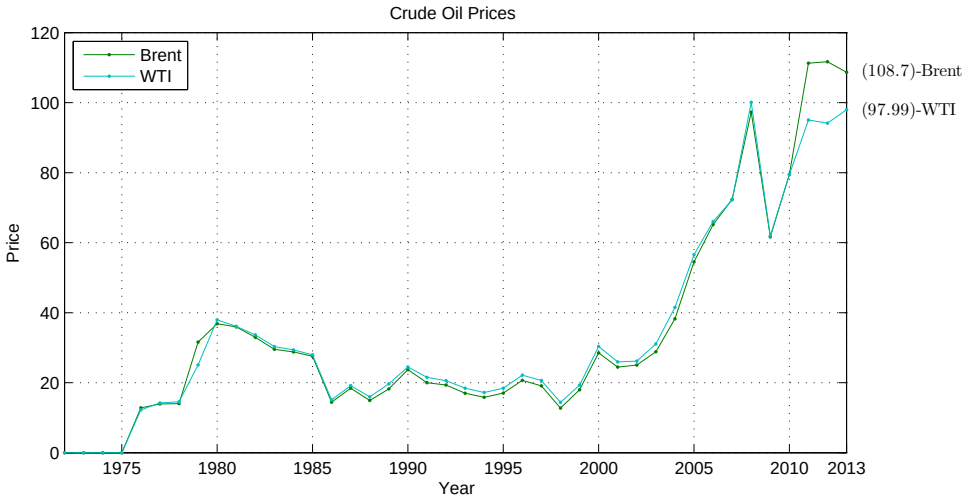



Figure 2.4: Crude Oil Prices. (Source: Bloomberg)

dollars per barrel in February 2009 [2]. But large oil price movements create uncertainty in the markets which is then reflected in different economics environments. Pindyck (1999) showed that large oil price variability causes delays in business investments and different studies showed the impact of crude oil volatility on macroeconomy (Lee et all (1995); Chen (2007); Huang et al. (2005)).

Pindyck (1999) and Regnier (2007) compared the volatility of different commodities and both concluded that crude oil has the highest one. Huang et al. (1996) found that changes on oil prices are reflected on both oil companies stocks prices and the wide stock market. A similar study was conducted by Sadorsky (1999, 2003) that arrived at the same conclusions.

There is a vast research on the influence that oil price has in GDP (Hamilton, 1996), stock returns (Sadorsky, 1999; Huang et al., 1996) and interest rates (Papapetru,2001; Ferderer,1996). And upon these elements are built a wide quantity of financial instruments, i.e. futures, swap and option contracts and others indexed securities.

An important study on the Energy market was conducted by Oberndorfer (2009) whi analysed the stock returns of energy corporations from the Eurozone. The study shows that oil price changes and oil price volatility affect oil

[2] Bloomberg,time series CO1 Comdty

and gas stocks. Price changes impacts positively on stock return , instead price volatility impacts negatively. Energy stock returns do not seem to be related to gas market development, and this is in contrast to the Canadian experience where there is a stock return sensitivity to a variation in gas prices (Boyer and Fillon, 2007).

A reason for the different result in Europe could be found in the presence of long-term contracts of gas selling where the price of gas is linked to oil prices in order to prevent any switching fuel incentives. In Europe price changes in coal affect stock returns of utility companies but less than oil effect. According to Oberndorfer's (2009) analysis the oil price is the main indicator used in stock market for the energy price developments as a whole [62].

One main difference between oil market and gas market is that the first one is globally connected while the second one is segmented. Three major gas region markets exist, with different characteristics: Europe, North America and Pacific Asia. Figure 2.5 shows the different trends that gas price has had in the principal regional natural gas markets.

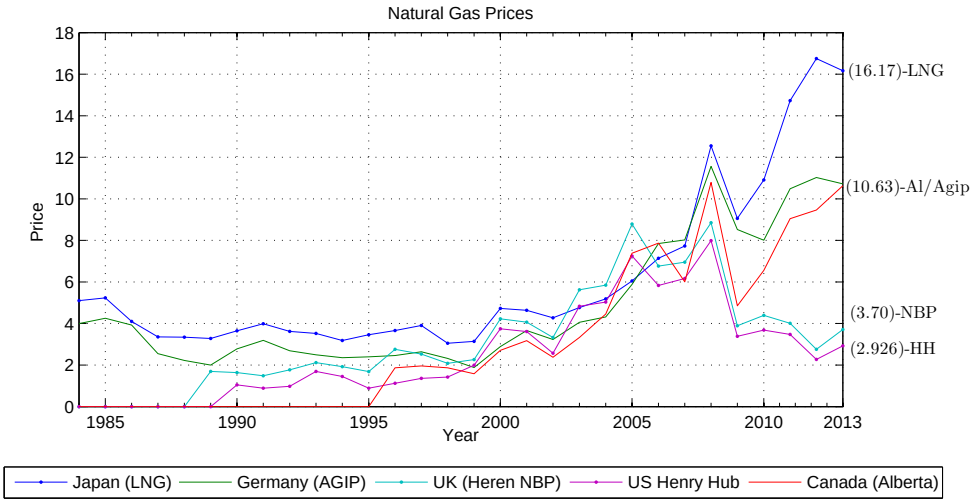



Figure 2.5: Natural Gas Prices. [82]

European market is the largest one and its principal gas suppliers are Russia, Norway, Netherlands and Algeria [12]. The trade is done principally by pipeline and only a small part as LNG (it accounts the 12% on the total pipeline). As mentioned above the continental natural gas is sold by long-term contracts

that link gas prices to oil prices. These contracts usually include clauses *"take-or-pay"* where the buyer agrees to buy a certain quantity of gas volume or must pay a penalty. European trading hubs are well connected and provide an efficient way to dissolve disequilibrium between national markets. The result of this gas-to-gas competition, that characterise European market, has led to equal price in almost all European countries [28]. Noteworthy was the opening of the Interconnector pipeline between Bacton (UK) and Zeebrugge in October 1998 that has permitted UK to supply natural gas to the Continental markets, where reserves and gas production are much lower. But at the same time it became a way to arbitrage activities on gas prices for UK traders: buy from Continental when the price is lower than in UK and sell when it is higher. But this activity linked, indirectly, UK gas prices to oil prices (since Continental gas price is linked with oil) and has led the two markets (Continental and UK) to a price convergence [65]. Anyway Panagiotidis (2007) and Asche et al. (2006) provided evidence that UK gas was cointegrated with Brent oil prices before the opening of the Interconnector and that gas and oil prices are moving together in the long-run. Hermsen and Jepma (2011) in a recent study proved the integration between UK and Continental market.

The Asia-Pacific region is characterised by its large use of LNG if compared with other regions. The total imports of LNG in 2013 was 238,1 billion cubic metres, contrast to 51,5 of Europe and 11,6 to North America [12]. Major suppliers countries are Indonesia, Malaysia, Australia and Middle East and principal importers are China, Japan and South Korea. The pipeline trade cover the 22% of the total import. Commonly LNG long-term contracts supply are linked to oil price by using JCC [3] as reference price index. This explains why today in the Asia-Pacific region the LNG prices are integrated with crude oil prices. However Siliverstovs et al (2005) show that this integration is with Brent oil and even with European gas prices, but not with US gas prices. Substantially, we can see that in both European and Asia-Pacific regions gas prices

[3] JCC is the acronym of Japanese Custom Cleared, or better knows as Japanese Crude Cocktail. It consists in a average price of the top twenty most traded crude oils import by volume into Japan. JCC is published every month by the Japanese government. Created in Japan in the early 1970s, today it is commonly uses also by Taiwan and South Korea as a reference index in their own long-term LNG Contracts.

are coupled with oil prices because for its (crude oil) stability price [28]. But this condition may stop to subsist. In long-term contract both side have interest to be linked to oil price, as long as oil price becomes too hight. In that case who previously sold gas indexed might have to relinquish it. With a netback-base LNG trading along with the tremendous reduction on LNG transportation and liquefaction cost [31], gas price could be decoupled from oil price and might contribute to an oil-independent gas market [28] in both European and Asia-Pacific regions.

North-American region has different characteristics. The exportations are inter-regional and almost totally traded by pipeline. The extra-regional imports are traded as LNG and weight only the 8.5% of total imports. Canada is the major pipeline supplier (78,9 billion cubic metres in 2013) while LNG imports are mainly from Trinidad & Tobago (2.6 bcm), Peru (2.5 bcm) and Qatar (2.6 bcm) [12]. Oil and gas prices have not an explicit link as in Europe and Asia-Pacific, but after opening network access in 1985 [4] a strong convergence in the North American gas markets have been seen. The price benchmark for oil is the West Texas Intermediate (WTI) and Henry Hub (HH) for natural gas. Even if there are periods where oil and gas prices decoupled in the short run, there is a rather stable relationship between the two commodities [86]. Serletis and Herbert (1999) showed that US natural gas prices and US fuel oil prices were cointegrated. And Barcella (1999) concluded that crude oil was hight correlated with gas prices and that their were cointegrated cause the relevant inter-fuel competition in the US electric power sector. But a recent study conducted by Péter Erdős (2012) shows that US oil and gas prices co-moved in short-term and were integrated in the long term during the period 1997-2008 but, since 2009, they have decoupled each others. The oversupply in US market of shale gas brought down US gas prices and led US prices lower than Europe. Erdős concluded that *"intercontinental arbitrage is a necessary condition*

[4]The Federal Energy Regulatory Commission (FERC) is an independent American agency that has the mission to regulate the interstate sales and the transportation of electricity and natural gas in the USA since 1935. In the 1985, FERC Order N.436 provided incentives for interstate pipeline companies to offer transportation service to producers and end users [85]. This permitted consumer to negotiate prices directly with producers and contract separately for transportation. After deregulation, production and usage of natural gas increased.

*of the long-term US gas oil price integration"*. In fact even if US oil and gas markets are integrated [4], Siliversstovs et al. (2005) find that US gas prices are not integrated with Brent crude oil prices.

Figure 2.6 shows the major trade movements in the 2013. It permits to see the differences among Europe, Asia-Pacific and North American regions in term of pipeline and LNG gas uses.

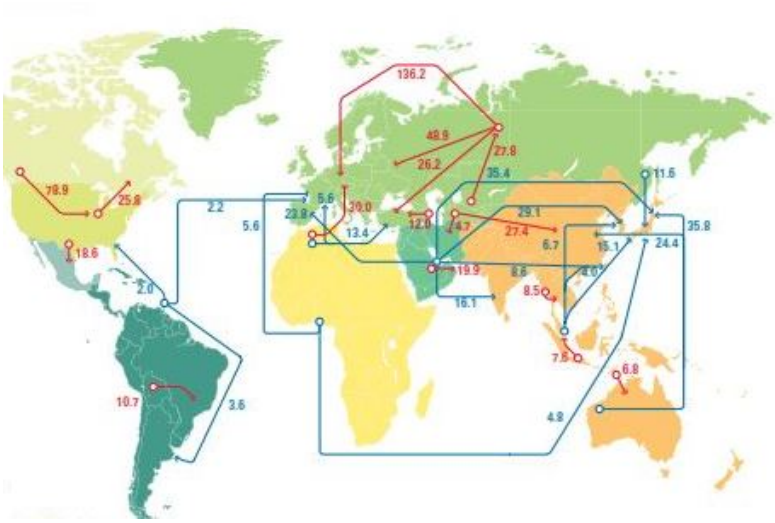



Figure 2.6: Major trade movement of natural gas in the worls. (Source[81])

## 2.3 Data Analysis

The empirical analysis is made on the Brent and West Texas Intermediate (WTI) spot prices for the crude oil and Henry Hub (HH) and National Balancing Point (NBP) spot prices for natural gas. Brent is used as international benchmark for crude oil price and WTI is the main benchmark in the North American crude oil market. HH is the largest of the 39 trading natural gas hubs in the US, and NBP is the most liquid gas trading point in Europe. NBP is similar in concept to the HH but differs in that it is a virtual trading location. The datasets consist of daily average prices over the period from September, 2001 to September, 2014.

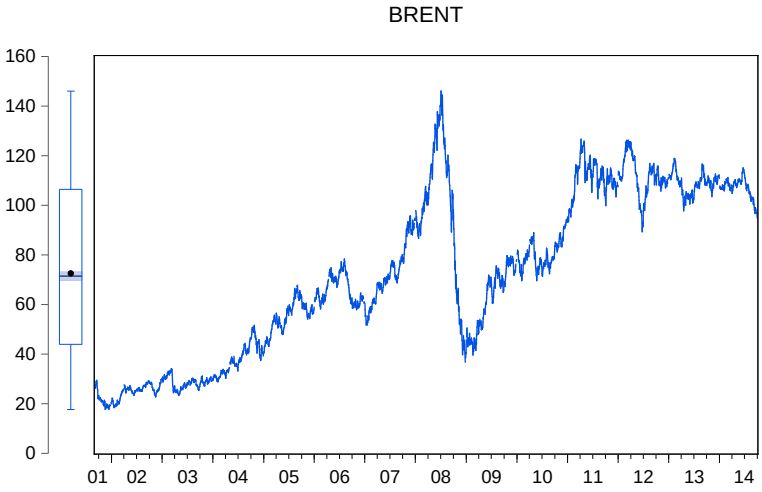
**Crude Oil** The total observations are 3353 for Brent and 3283 for WTI. The prices are in US dollars and cents per barrel [5]. Figure 2.7 shows the dynamics of the two prices considered. We can see how, after financial crisis in the 2008, oil prices have been characterized of higher volatility.

We have proceeded to log transform prices for then analyse the price series and log series. Figure 2.9 presents the basic descriptive statistics of the two series on Brent and WTI prices. Figure 2.8(a) and Figure 2.8(b) represent the price series, while Figure 2.9(a) and Figure 2.9(b) represent value of the log-transform series. It is clear that both (prices in level and log-transform prices) have not a normal distribution. The skewness is negative in all cases; it means that the lower tail is thicker than upper tail. The kurtosis are substantially lower than 3 and Jarque-Bera test have high values, greater than the critical value of the chi-square distribution (5.99 at the 5% level of significance)[6].

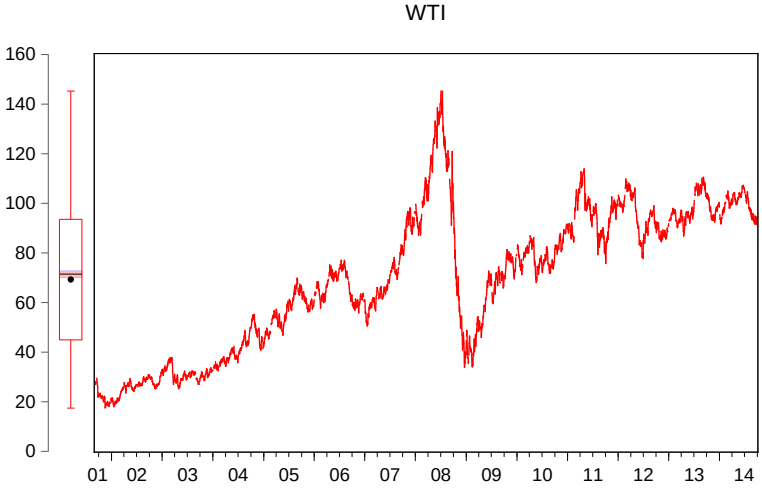
[5] A barrel of crude oil is equal to 42 U.S. gallons or 158.9873 litres.

[6] In the descriptive data series it useful see the value of *skewness*, *kurtosis* and *Jarque-Bera statistic* to evaluate if the data series approximates a particular probability distribution.

- *Skewness* gives informations about the symmetry of probability distribution around its mean. The skewness of a symmetric distribution, such as the normal distribution, is 0. It is computed as: $S = (1/N)\sum_{i=1}^{N}(y_i - \overline{y}/\widehat{\sigma})^3$, where $\widehat{\sigma}$ is an estimator for the standard deviation that is based on the biases estimator for the variance ($\widehat{\sigma} = s\sqrt{(N-1)/N}$); $N$ is the number of observation and $\overline{y}$ is the mean of the series. Skewness is positive when right tail of distribution is thicker than left tail, and negative when left tail is thicker.
- *Kurtosis* provides a measure of the peakedness or flatness of the distribution. In the normal distribution is value is equal to 3. It is computed as: $K = 1/N\sum_{i=1}^{N}((y_i - \overline{y})/\widehat{\sigma})^4$

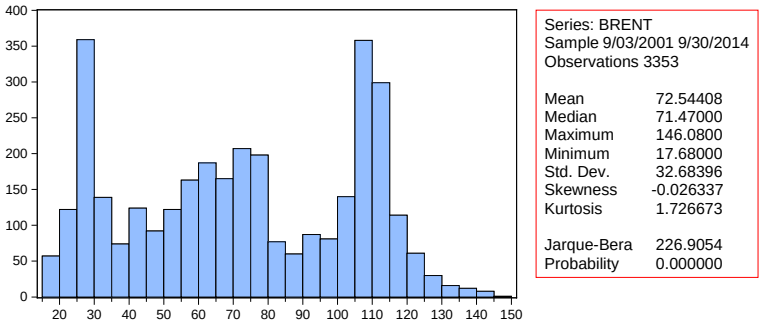



(a) Brent 2001-2014

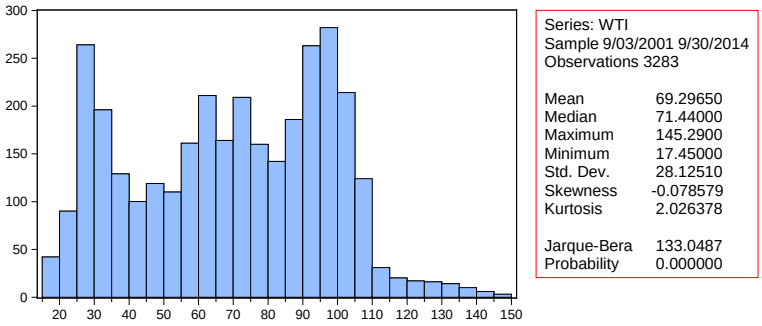



(b) Wti 2001-2014

Figure 2.7: Brent and Wti prices dynamics during period 2001-2014. To the left of each figure is reported the respective box-plot diagram.




(a) Brent prices

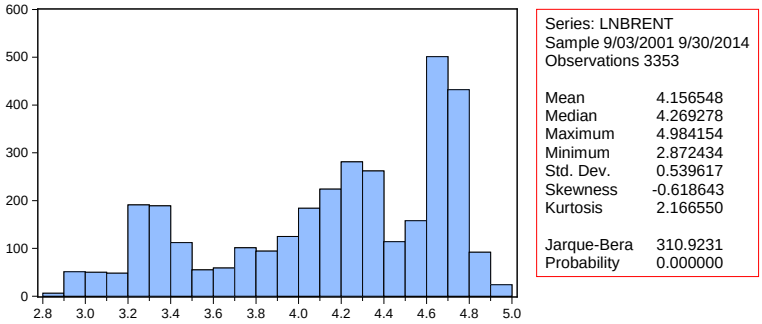



(b) Wti prices

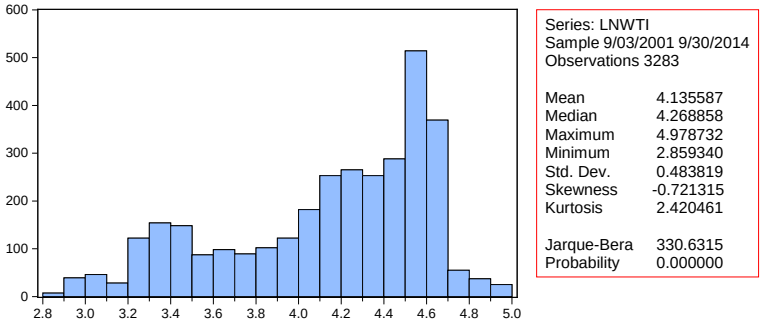
Figure 2.8: Histogram and descriptive statistic data on Brent and Wti prices

In order to assume stationarity in the series we have proceeded with unit-root test. We have applied three different tests: the Augmented Dickey-Fuller test (ADF) and the Phillips-Perron test (PP), which assume the existence of unit root as the null hypothesis $H_0$ and stationarity as the alternative hypothesis $H_1$. Then we have compared these with the Kwiatkowski-Phillips-Schmidt-Shin test (KPSS) which assumes stationarity as the null hypothesis $H_0$ and the existence of unit root as the alternative $H_1$. Table 2.2 reports the results of the tests. The tests were used on each of series, no-transform and log-transform prices. We can see that all tests reject the hypothesis of stationary at 10%, 5% and 1% level for all series considered. We have proceeded the tests on the dif-

- *Jarque-Bera test* is a test statistic that measure the difference of the skewness and kurtosis of the series with those of the normal distribution. The JB statistic is distributed as $\chi^2$ with 2 degree of freedom. If the JB test value is greater than the critical value of the $\chi^2$, null hypothesis of normal distribution is rejected. It computes as: $JB = N/6(S^2 + (K-3)^2/4)$.




(a) Log-Brent prices




(b) Log-Wti prices

Figure 2.9: Histogram and descriptive statistic data on Brent and Wti log-transformed prices.

ferenced series in order to see if the series have multiple unit-root. As already know in the literature, we can see also in our results that the oil prices are integrated of first-order and that ADF and PP tests reject the null hypothesis of the existence of unit root at I(1). Alike KPSS test fails to reject the null hypothesis of stationarity in all differenced series considered.

| Unit Root Test | | | | | | |
|---|---|---|---|---|---|---|
| | **Price in levels** | | | | | |
| | With Intercept and Trend | | | Without Trend | | |
| | ADF | PP | KPSS | ADF | PP | KPSS |
| Brent | (-2.44) [0.3596] | (-2.40) [0.3807] | (0.25) | (-1.35) [0.6080] | (-1.55) [0.5086] | (6.19) |
| Log-Brent | (-2.27) [0.4517] | (-1.91) [0.6493] | (0.83) | (-1.81) [0.3740] | (-1.30) [0.6338] | (6.24) |
| WTI | (-2.19) [1.0000] | (-2.35) [0.4047] | (0.42) | (1.54) [0.9994] | (-1.77) [0.3986] | (5.66) |
| Log-WTI | (-1.12) [0.9245] | (-2.19) [0.4942] | (0.92) | (-0.11) [0.9467] | (-1.63) [0.4671] | (5.83) |
| | **Price in 1st differences** | | | | | |
| | With Intercept and Trend | | | Without Trend | | |
| | ADF | PP | KPSS | ADF | PP | KPSS |
| Brent | (-21.64) [0.0000] | (-60.33) [0.0000] | (0.05) | (-21.64) [0.0000] | (-60.33) [0.0001] | (0.08) |
| Log-Brent | (-8.98) [0.0000] | (-60.68) [0.0000] | (0.05) | (-8.98) [0.0000] | (-60.69) [0.0001] | (0.08) |
| WTI | (-4.77) [0.0005] | (-58.58) [0.0000] | (0.04) | (-4.80) [0.0001] | (-58.58) [0.0001] | (0.08) |
| Log-WTI | (-7.34) [0.0000] | (-57.55) [0.0000] | (0.04) | (-7.34) [0.0000] | (-57.54) [0.0001] | (0.09) |
| Test Critical Value | | | | | | |
| ADF/PP | 1%(-3.96), 5%(-3.41), 10%(-3.12). | | | 1%(-3.43), 5%(-2.86), 10%(-2.57). | | |
| KPSS | 1%(0.22), 5%(0.15), 10%(0.12). | | | 1%(0.74), 5%(0.46), 10%(0.35). | | |

Note: The table reports the ADF and PP unit root tests and the KPSS stationary test. The tests analyse both trend-stationarity and no-trend stationarity. The Akaike Info Criterion (AIC) is used for the selection of lag in ADF test, whereas PP and KPSS tests used the Newey-West automatic bandwidth selection with Bartlett kernel estimator. The parentheses report the t-statistic value for ADF and PP tests, instead the LM-statistic for KPSS test. The square brackets show the respectively p-value. On the top of the table there are the results in the levels. On the bottom the results on the first-differences. The last two rows report the critical values of the respective tests at 1%, 5% and 10% levels.

Table 2.1: Unit Root Test on crude oil prices.

**Natural Gas** The total observations for HH and NBP are 3273 and 3345 respectively. The prices are in US dollars and cents per Million Btu[7]. Figure 2.10

[7] British thermal unit (Btu) is the quantity of heat required to raise the temperature of 1 pound of liquid water by 1 degree Fahrenheit at the temperature at which water has its greatest density

shows the dynamics of prices during the period September 2001- September 2014. We can note a short and high volatility occurred in brief periods. In the February 2003, from the 23th to the 25th, a severe cold in the middle regions of the US rose prices from $ 6.4 to $ 18.4 for then return at $ 10.4 the day after. In the 2005, during December, two hurricanes damaged Gulf of Mexico production, processing and transport infrastructure: the prices peaked at $15.40 and trading was suspended twice. In 2008 the "turbulence" of market drove price from $ 6/8 in average of the prior year to $ 13 of July 2, for close in December at $ 5.63.

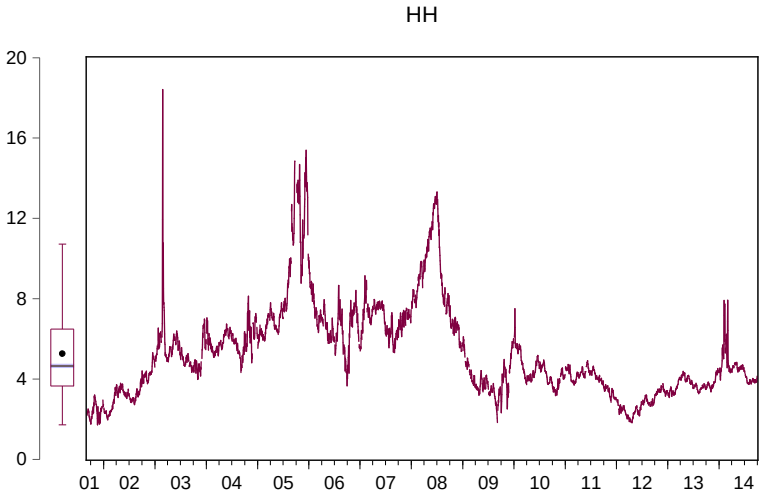
Figure(2.11) reports the main statistical data of HH and NBP price, instead in the Figure(2.12) is reported the statistical data on the log series. We can see how the distribution on prices are characterized of a positive skewness, with the right tail longer than the left tail due, as we said above, to the short periods with high volatility. Median is lower than mean and Kurtosis is 5.48 for HH and 2.15 for NBP , far respect the normal value of 3. JB statistic is 1817.301, and 131.27 for HH and NBP respectively, substantially greater than critical value 5.99 at 5% level of significance. For HH the prices looks like a log-normal distribution, but it is not. In fact we can see from Figure(2.12) that also the log-price distributions is not normal. JB test, while having a significant lower value of 16.17, is however larger than the critical value of 9.21 at 1% level of significance. We can see how the NBP price is significantly higher than HH price: the mean value for NBP is 41.8 compare to the 5.26 of HH. The root unit test is reported in Table2.2. We can see how ADF and PP tests reject the null hypothesis of unit root in the level with trend and intercept. On the other hand KPSS reject the null hypothesis of stationarity on the levels without the tren, for both price and log-price series. Under this considerations we consider both HH and NBP series non-stationarity on the levels.

(approximately 39 degrees Fahrenheit).

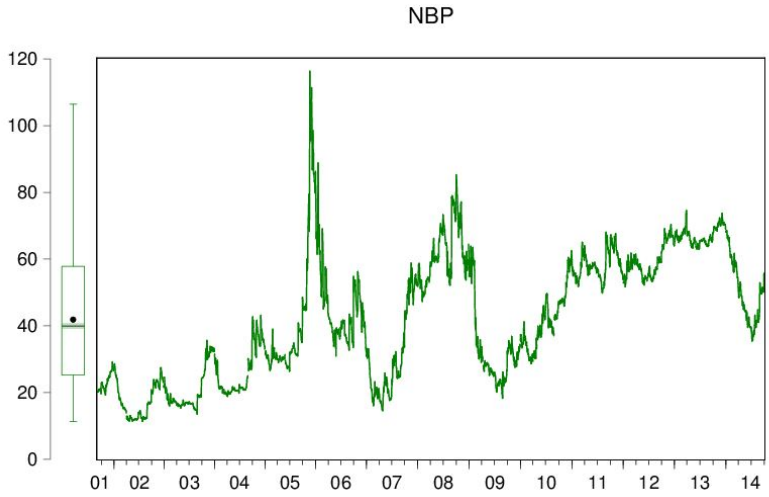
| Unit Root Test | | | | | | |
|---|---|---|---|---|---|---|
| | Price in levels | | | | | |
| | With Intercept and Trend | | | Without Trend | | |
| | ADF | PP | KPSS | ADF | PP | KPSS |
| HH | (0.21) [0.9981] | (-3.55) [0.0347] | (0.96) | (0.29) [0.9775] | (-3.33) [0.0138] | (1.67) |
| Log-HH | (-0.39) [0.9878] | (-3.11) [0.1050] | (1.04) | (-0.29) [0.9237] | (-2.83) [0.0549]* | (1.71) |
| NBP | (-9.471770) [0.0000] | (-3.384468) [0.0536] | (0.16) | (-3.096104) [0.0270] | (-2.756760) [0.0648] | (3.91) |
| Log-NBP | (-3.645871) [0.0262] | (-3.068153) [0.1142] | (0.26) | (-2.450422) [0.1281] | (-2.352854) [0.1556] | (4.27) |
| | Price in 1st differences | | | | | |
| | With Intercept and Trend | | | Without Trend | | |
| | ADF | PP | KPSS | ADF | PP | KPSS |
| HH | (-10.11) [0.0000] | (-57.18) [0.0000] | (0.03) | (-10.11) [0.0000] | (-57.18) [0.0001] | (0.04) |
| Log-HH | (-7.96) [0.0000] | (-52.86) [0.0000] | (0.05) | (-7.96) [0.0000] | (-52.86) [0.0001] | (0.12) |
| NBP | (-9.471770) [0.0000] | (-55.52183) [0.0000] | (0.02) | (-9.470264) [0.0000] | (-51.53042) [0.001] | (0.02) |
| Log-NBP | (-10.23185) [0.0000] | (-54.93185) [0.0000] | (0.02) | (-10.23323) [0.0000] | (-54.93971) [0.0001] | (0.03) |
| Test Critical Value | | | | | | |
| ADF/PP | 1%(-3.96), 5%(-3.41), 10%(-3.12). | | | 1%(-3.43), 5%(-2.86), 10%(-2.57). | | |
| KPSS | 1%(0.22), 5%(0.15), 10%(0.12). | | | 1%(0.74), 5%(0.46), 10%(0.35). | | |

Note: Note: The table report the ADF and PP unit root tests and the KPSS stationary test. The Akaike Info Criterion (AIC) is used for the selection of lag in ADF test, whereas PP and KPSS test used the Newey-West automatic bandwidth selection with Bartlett kernel estimator.

Table 2.2: Unit Root Test on natural gas prices.

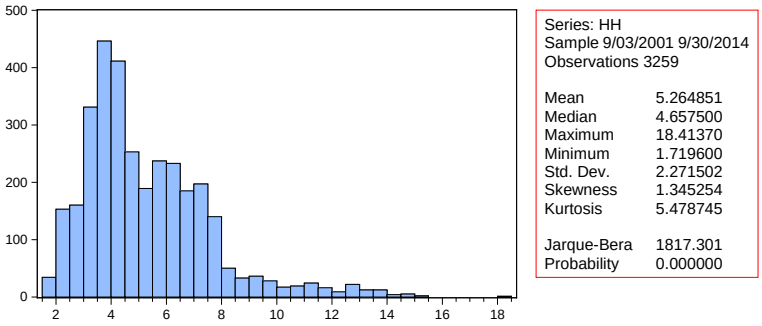



(a) Henry Hub Natural Gas Prices

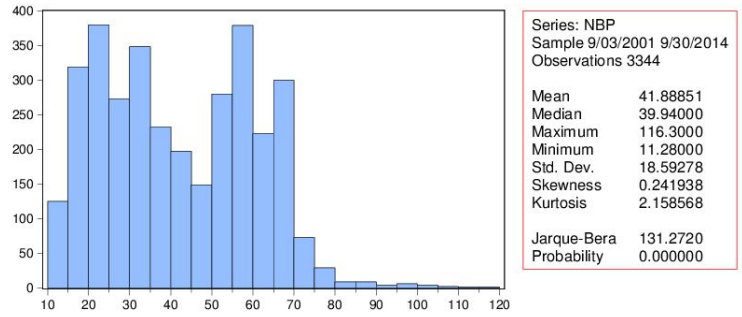



(b) NBP Natural Gas Prices

Figure 2.10: Natural gas price dynamics in Henry Hub and NBP markets

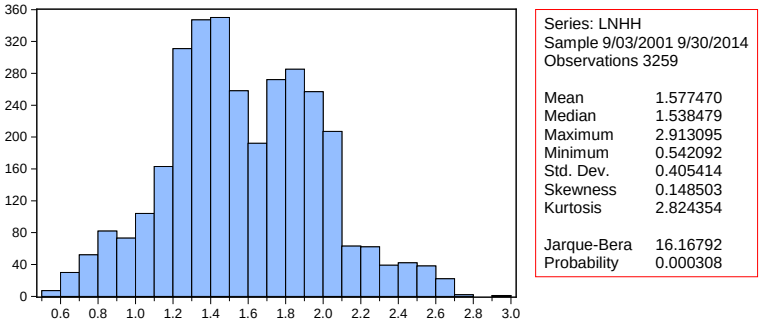



(a) Henry Hub prices distribution

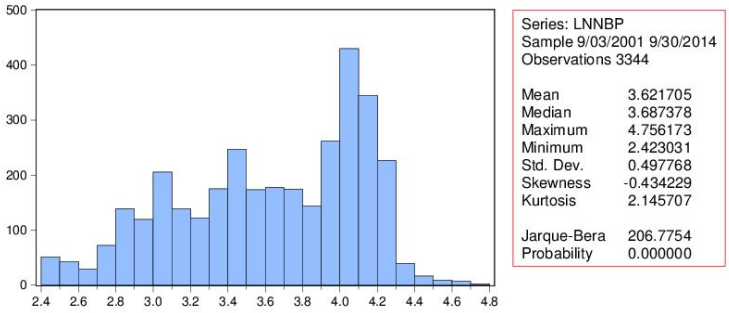



(b) NBP price distribution

Figure 2.11: Histogram and descriptive statistic data on HH and NBP prices.




(a) Henry Hub log-prices distribution




(b) NBP log-prices distribution

Figure 2.12: Histogram and descriptive statistic data on HH and NBP log-transform prices.

# Chapter 3

# Forecasting with ANN

Different models exist to forecast financial time series. In this paper we will focus on non-linear models and in particular we will see if the artificial neural networks (ANNs) can be a good tool in the forecast of energy commodities prices. As we have seen in the first chapter, there are different neural network architectures, depending on the specific problem. The most popular and the most widely used in forecasting is the multi-layer perceptrons (MLP) thanks to their capability of input-output mapping [6]. As we have seen in the second chapter, commodities prices (and financial markets in general) are characterized by non-stationarity environment and several non linear time series models have been developed, such as the autoregressive conditional heteroscedastic (ARCH), Bilinear Model, threshold autoregressive (TAR),generalized autoregressive conditional heteroscedastic (GARCH) or the autoregressive integrated moving average (ARIMA). However, most of these models try to explain the time series process through a linear relationship between dependent and independent variables and they work well under the hypothesis of stationary data sets. This is a very difficult task since the series extracted from the real world are not generally linear and, even if they were, there are too many patterns that can be linked to a model and that could be part of that. ANNs, on the contrary, does not need any a priori hypothesis about relationship between input and output. They are a completely data-drive self-adaptive system and a ANN with adequate complexity is capable of approximating any function to a rea-

sonable accuracy [17]. Furthermore Tan (2001) concluded in his research that, of all artificial intelligence, ANNs are the ones giving the best results in noisy data environments and dealing best with uncertainty. There have been a number of reported studies that have used ANNs in financial time series forecasting (Amano, Marchesi, & Murru,2005; Castiglione, 2001; Chang, Yang, Dong, & Abraham, 2005; Refenes, Burgess, & Bentz, 1997; Yudong, Lenan, 2008). An interesting survey conducted by Atsalakis G. S. and Valavanis K.P. (2009) resumed 100 published articles focused on neural network and neuro-fuzzy techniques; and all these studies were applied to forecast different stock markets. Whilst the financial market, and financial time series in general, have been covered by a number of studies on ANNs implementation, there is not large literatures about using ANNs system in commodities prices and in particular on energy prices. Kohzadi N., et al. (1996) compared ARIMA model with a feed-forward neural network in forecast cattle and wheat prices, and achieved a better performance with NN model than ARIMA. On the other hand Panella M., et al. (2012) analysed different neural networks techniques to forecast crude oil, natural gas and electricity prices. The result of the study demonstrated that all different neural model tested provide a good performance in forecasting energy commodity prices, in particular the Adaptive Neurofuzzy Inference System (ANFIS) was the network that reached the best performance. In our study we want to analyse how artificial neural networks perform in the energy market environment and if it can be used in forecasting prices, compared to standard linear model.

## 3.1 Model elements

We choose to use a feed-forward multilayer perceptron network (MLP). The network is typically composed of one input vector, one or more hidden layers, one output vector. The first element to choose is the number of hidden layers.

**Hidden Layers** Different studies were conducted in this field. Even if one hidden layer can approximate any complex non-linear function [24], two hidden layers are usually preferred because they ”can provide more benefit for some type of problem” [9]. Furthermore it is demonstrated that more than

two hidden layer do not give any improved result [20]. According to Zhang et al. (1998) one single hidden layer in a feed-forward neural network is enough in dealing with forecasting problems. In our model we have proceed using a single hidden layer and a single output layer.

**Hidden Nodes** A crucial problem in the network structure definition is the choice of hidden nodes. There is no theoretical basis for selecting this parameter. In general fewest neurons are preferred to have better generalization, but too few may not give enough power to model and learn data [93]. The more the number of hidden neurons, the more capacity the network has to memorize rather than generalize [36]. Some researchers have found, thanks to empirical studies, some rules to avoid the overfitting. If we define $n$ as the number of input nodes, Lippmann(1987) and Hecht-Nielsen (1990) determinates the maximum number of hidden nodes as $(2n+1)$, Wong (1991) as $(2n)$, Tang and Fishwick (1993) as $(n)$ and Kang (1991) as $(n/2)$. Other solutions exist (see Kim et al. (2003), Jaruszewicz and Mandziuk (2004), Kim and Lee (2004),Tan (2001) for a survey) and it is clear that this task is still under experimentation [3]. Trial snd error is the most common way. The basic rule can start with few neurons and then increase them until performance is maximized [43]. On the contrary, neurons are removed when weight values associated with those neurons remain constant for a large number of training epochs [56]. Anyway, in forecasting problem Tang and Fishwick concluded that the effect of hidden neuron does not have any relevant impact on performance (that is not true for instance in case of patter recognition problems). In our model, according with Zhang et al. (1998), we started with a number of hidden nodes equal to the number of input nodes. We added sequentially nodes until performance increased significantly.

**Input Nodes** The number of input neurons is one of the most important parameter and plays a significant role in MLP networks with forecasting purpose. When we use a network to forecast future, the number of inputs nodes corresponds to the number of variable in the input vector. In general we can define,

in an ANN, the functional relationship between output and inputs as:

$$y = f(x_1, x_2, ....x_p) \tag{3.1}$$

where $y$, the output, can be considered as the regressand, and $x_1, x_2, ..., x_p$, the inputs, can be considered as $p$ regressors. In this term, the neural network is functionally equivalent to a non-linear regression model [93]. In the specific case of time series forecasting we can rewrite eq. 3.1 as:

$$y_{t+1} = f(y_t, y_{t-1}, ..., y_{t-p}) \tag{3.2}$$

where $y_t$ is the observation at time $t$ (our output), and the past values $(y_t, y_{t-1}, ..., y_{t-p})$ is the input vector. The problem consists in determining the number of lag observations to use at each interaction. In fact the size of lagged observation are used in the training set as a pattern for identify the weights values. For better understanding let us consider a time series with $N$ observations in the training set. Our network gives us one-step-ahead prediction, as we have seen in eq. 3.2. Suppose we have a lag number of $p$ observations. So we have $N - p$ training patterns. The first training pattern will be composed of $(y_1, y_2, ..., y_p)$. It is the input vector at the interaction one and returns $y_{p+1}$ as a output. The second training pattern will have $(y_2, y_3, ..., y_{p+1})$ as input and $y_{p+2}$ as output. And so on until the last pattern that will be composed of $(y_{N-p}, y_{N-p+1}, ...y_{N-1})$ as input and will have $y_N$ as output. Many authors have tried to find some theoretical or empirical rules to identify the right number of temporal-lags to use in the input vector. Tang and Fishwick (1993) simply use the number of autoregressive (AR) terms of a univariate time series. Sharda and Patil (1992) suggest using twelve inputs for monthly data or four for quarterly. Cheung at al. (1996) propose to use maximum entropy principles. In contrast Zhang et al. (1998) say that AR can't be a solution to identify the right input number nodes since moving average (MA) processes have AR term equal to zero. Furthermore through the literature we can find contrasting opinions on the use of a fixed lag related on temporal frequency (monthly rather than quarterly): Tang et al. (1991) report benefit using more input nodes, instead Lachtemacher and Fuller (1995) find just the opposite. Zhang et al. (1998) propose to use theoretical research developed in non-linear time series analysis as: Lagrange multiplier tests, likelihood ratio-based tests, bispectrum tests and others. Recent

studies have introduced the use of genetic algorithms (GA) to determinate the number of input neurons and hidden neurons [1] or in other cases by using fuzzy inference methods. These kind of hybrid approaches not only try to find the best solution in terms of input nodes numbers, but they are focused in finding the best ANN architecture in terms of input nodes, hidden nodes and hidden layers. As we can see determining input number nodes is still a crucial task, furthermore none of these approaches guarantee the optimal solution [45].
Based on all these aspects, the main idea in literature is to achieve the proper number of input and hidden neurons via trial and error and heuristic approach [36]. The neuron's number depends on the problem and the procedure is to try different neurons networks varying input and hidden nodes. This usual approach is the one we have used in this paper.

**Output Nodes** The number of output nodes is usually determined by the choices made for the input and hidden layers. It depends on the particular problem and it is usually easy to identify. Anyway, it is not free of controversial opinions in the literature. For a time series forecasting problem there are two types of forecasting: one-step ahead and multi-step ahead. In the first one there is just one output node that gives the $\hat{y}_{t+1}$ predicted value. On the other hand, in multi-step ahead there are two different ways: the iterative mode and the direct mode. Iterative forecast is the same used in the Box-Jenkins model, in which the predicted value is used as input for the next forecast. This method requests only one output node and the sequence can be represented as:

$$\hat{y}_{t+1} = f(y_t, y_{t-1}, ..., y_{t-n})$$

$$\hat{y}_{t+2} = f(\hat{y}_{t+1}, y_t, ..., y_{t-n+1})$$

$$. \quad . \quad .$$

$$\hat{y}_{t+k} = f(\hat{y}_{t+k-1}, \hat{y}_{t+k-2}, ..., y_t, ..., y_{t-n+k-1})$$

where the values $\hat{y}$ represents the estimated values, $y$ are the observed data and $f$ are the functions of the ANN. As in the Box-Jenkins models, this prediction

[1] Genetic algorithms (GAs) are adaptive heuristic procedures that mimic the process of natural selection and biological evolution. GAs are used in optimization problems to reach the best solution.

works well for short-term forecasting, and longer the period to forecast, less accurate is the prediction. On the other hand, in the direct mode the network uses as many outputs as the length of the forecast horizon. The ANN uses all the past values as inputs to predict directly the $k$ step-ahead values. The network has $k$ output nodes that can be represented as:

$$\hat{y}_{t+1} = f_1(y_t, y_{t-1}, ..., y_{t-n})$$

$$\hat{y}_{t+2} = f_2(y_t, y_{t-1}, ..., y_{t-n})$$

$$. \quad . \quad .$$

$$\hat{y}_{t+k} = f_k(y_t, y_{t-1}, ..., y_{t-n})$$

where the $f_1, f_2, ..., f_k$ are the functions of the ANN. The weakness of this method is that it utilizes past data to forecast future. Zhang et al. (1998) explained the reason why the direct method is better in a multi-period neural network. They affirm that direct method has the benefits of the iterative Box-Jenkins methods but without the restriction of linearity correlation among the lagged data. However, Hill et al. (1994) and Weigend et al. (1992) had better results in their studies with iterative method instead of direct ones. From the literature we can assume that both iterative and direct methods can be used in a multi-step ahead prediction [94][36][93]. Generally for both one-step ahead and multi-step ahead prediction one output node is used [94].

## 3.2 Model Parameters

Once the architecture of the network is specified, there are some parameters that have to be chosen.

**Performance Function** As we have seen in Chapter 1, network can be considered as a "function" that has to minimize the differences between the targets and the outputs. In order to see how the network, or the different networks work we need to identify a performance measurement. The typical performance function used is the mean of square error:

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{n=1}^{N}(t_i - a_i)^2$$

where $(t_i - a_i)$ is the difference between the $i$-th target and the $i$-th output. However this measure is not universally accepted. Some authors criticized the limitation associated with this measure [22][2]. Noteworthy is the assumption, from the statistics point of view, for which the MSE, as defined above, does not consider the number of parameters that the model has to estimate. In the ANNs the estimated parameters are the arc weights. So, the correct sum of squared errors in the training section has to be divided by the degree of freedom as:

$$MSE(AW) = \frac{\sum_{n=1}^{N}(e_i)^2}{N - AW}$$

where $AW$ is the number of arc weights and node biases [93]. A number of others performance functions for forecasting measurement exist, all with advantages and limitations. The most commonly used, after MSE, are: the mean absolute error (MAE), the sum squared error (SSE) and the root mean square error (RMSE).

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|e_i| \quad ; \quad SSE = \sum_{n=1}^{N}(e_i)^2 \quad ; \quad RMSE = \sqrt{MSE}$$

**Transfer Function** The transfer function (also called *activation function*) determinates the relationship between inputs and outputs. In general, all differentiable functions can be used as transfer functions in a network, but in practice, differentiable, bounded and monotonically increasing functions are preferred. The most common transfer function used in literature are the sigmoid (or logistic) function and the hyperbolic tangent function:

$$Sig(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$

$$Tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{3.4}$$

Transfer functions introduce a degree of non-linearity, useful in most neural network applications. Even if non theoretical rules exist to choice one particular transfer function rather another [37], there are some heuristic rules that can help in that choice. Klimasauskas (1991) proposes sigmoid function for classification problems, hyperbolic tangent function in forecasting. But in general a network has different layers, with different nodes. So, different activation functions are used in a network and the choice depends on the particular problem.

For example, if the targets of a problem are binary, logic activation function (as *hard limit* with values 0 and 1) is suitable in the output layer. Commonly in forecasting problem a *linear* function is used in output layer in order to assure no bounded values. But linear output involves a limitation in time series with trend [23]. It has been shown that a two layers network with hyperbolic/sigmoid functions in hidden layer and a linear output layer is capable of approximating any function of interest (giving an appropriate number of hidden neurons) [34][40]. Multiple layers of neurons with non-linear transfer functions allow the network to learn non-linear and linear relationship between inputs and outputs. The linear output layer allows the network produce value outside the range $[-1,1]$. For our purpose, according to the main studies and practices on forecasting [93][36][25], that kind of function sequence will be used.

**Training Algorithm** When a *Performance Function* is defined, during the training period weights and biases are adjusted in order to minimize the "cost function". This minimization is done using different optimization algorithms. The "goal" would be to find the global minimum of the cost function but, at the moment, there are not algorithms that can guarantee optimal solution in a "reasonable" amount of time [93]. For this reason the problem is limited to find that algorithm that can guarantee the best local optima solution when the global minimum is not reachable. A number of optimization methods exist, the most popular is the backpropagation algorithm that we have seen in section 1.4. The basic backpropagation algorithm is a gradient descent method. Recalling eq.(1.2) is defines as:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \mathbf{g}_k$$

where $\mathbf{x}_k$ is the vector of current weights and biases, $\mathbf{g}_k$ is the current gradient and $\eta_k$ is the learning rate. We can rewrite the equation above for the weight as:

$$\triangle\, w_{i,j}(k+1) = \triangle\, w_{i,j}(k) - \eta \frac{\partial E}{\partial w_{i,j}} \tag{3.5}$$

The partial derivatives $\frac{\partial E}{\partial w_{i,j}}$ is computed in two passes (or steps): the fist is called *forward pass* and the second *backward pass*. In the forward pass an input

vector of the training set is applied to the network and is propagated, layer by layer, through it producing the output. After that the output is compared with the target and the error is propagated backward though the network (backward pass). In this last pass the weight and biases are adjusted in order to reduce the error. Actually there are two different ways to adjust the weight and biases.
The first is called *incremental mode* in which weights and biases are updated after each input is applied to the network. This means that, for instance, if we have an input vector with three input nodes, the weights and biases will be updated three times during one pass (epoch) through the training set. It is commonly used with dynamic networks as adaptive filters. The second is called *batch mode*: here the weights and the biases are update only after all inputs have been presented. This means, for instance, that with the same input vector with three input nodes, the weights and the biases are update just once at each epoch. One we have decided the way in which the weights and biases have to be updated, the crucial element to be specify, in the eq. (3.5), is the learning rate $\eta$. It determinates the changes of the weights and the biases: the larger the learning rate, the bigger the step. However, a small value of $\eta$ makes the learning process slower, while too large values may cause network to oscillate in the weight space from one side to the other. The main problems of this algorithm are the slow capacity to convergence, inefficiency and lack of robustness [70]. To overcome this instability a momentum term $\phi$ has been added to the original gradient method as follow:

$$\triangle w_{i,j}(k+1) = \phi \triangle w_{i,j}(k) - \eta \frac{\partial E}{\partial w_{i,j}}$$

The momentum term leads the algorithm to a faster convergence and minimizes the oscillations in the weight space. It adds a fraction $\phi$ of the previous weight update to the current one. This makes the new weight change to pointing at the same direction as the previous one. Momentum $\phi$ and learning rate $\eta$ are chosen in the interval $[0, 1]$. However the best combination of these parameters has to be chosen by experimentation [93][2]. Even if the standard backpropagation algorithm is adopted by most researches, it is often too slow

[2]Different studies were conducted to find the best learning parameters combinations, but the conclusion are inconsistent. See Chakraborty et al. (1992); Sharda and Patil (1992) and Tang and Fishwick (1993) for surveys.

for practical problems. Different variations or modifications have been made on this algorithm. The "new" algorithms can converge from ten to one hundred times faster than gradient descent or gradient descent with momentum [25]. These faster algorithms can be divided in two categories: the first uses heuristic techniques and the second uses standard numerical optimization. In the first category there are for example: the *Variable Learning Backpropagation* that changes the learning rate during the training process, as the algorithm moves across the performance surface; or the *Resilient Backpropagation* that use only the sign of the derivative, instead of the magnitude, to determinate the direction of the weight. Examples of standard numerical optimization are: the *Conjugate Gradient Algorithms* where the algorithm is performed along conjugate direction instead of steepest descent direction (this assures faster convergence)[3]; the *Quasi-Newton Algorithms* that use an approximation of Hessian matrix of the performance index at the current values of the weights and biases to update the gradient [4]; and the *Levenberg-Marquardt* that approximate the Hessian matrix and gradient with the computation of Jacobian matrix and the network errors [5]. These and other optimization algorithms can be found in

[3]**Conjugate gradient**: All the conjugate gradient algorithms start out by searching in the steepest descent direction on the first iteration. Recalling eq.1.2 we define $p_0 = -g_0$, where $\mathbf{g}$ is the gradient. The eq. 1.2 is now defined as: $\mathbf{x}_{k+1} = \mathbf{x}_k + \eta_k \mathbf{p}_k$. This is the first line search that define the optimal distance to move along. The next search direction is determined so that it is conjugate to previous search direction: $\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}$. The various versions of the conjugate gradient algorithm change by the manner in which $\beta_k$ is computed. For a discussion of the different conjugate gradient algorithms see [29] and [34]

[4]**Quasi-Newton Algorithm**: Using Newton's method eq.1.2 can be written as: $\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k$, where $\mathbf{A}_k^{-1}$ is the Hessian matrix (the second derivatives) of the performance index at the current values of the weights and biases. Since compute the Hessian matrix is expansive (in terms of time and memory), the quasi-Newton methods is a class of algorithms that approximate the Hessian matrix and don't require the second derivatives. The most known algorithm of this class is the Broyden, Fletcher, Goldfard, and Shanno (BFGS) algorithm. See [26] for a discussion of the BFGS algorithm.

[5]**Levenberg-Marquardt**: The Levenberg-Marquardt can be considered as a quasi-Newton method algorithm with the difference that for the computation of the Hessian matrix it uses the Jacobian matrix. When the performance has the form of a sum of square, the Hessian matrix can be approximated as: $\mathbf{H} = \mathbf{J}^T\mathbf{J}$; and the gradient can be approximate like: $\mathbf{g} = \mathbf{J}^T\mathbf{e}$, where $\mathbf{J}$ is the Jacobian matrix and $\mathbf{e}$ is the vector of the network errors. The Jacobian matrix contains the first derivatives of the network errors with respect to the weight and biases. In this terms the eq.1.2 can be written as: $\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}]^{-1}\mathbf{J}^T\mathbf{e}$, where $\mathbf{I}$ is the identity matrix and $\mu$ is a parameter that

literature and in most of neural network software package. For a more exhaustive explanation on the backpropagation variation algorithms see Chapter 12 "Variations on Backpropagation" of [34].

In this paper the first step was compare the different training algorithms, after that we have proceeded with the one with the best performance. In general, Levenberg-Marquardt is the faster algorithm for training but it requires, compare to others, more memory (in term of software computation).

**Pre-processing and Post-processing.** Before the input data are used in the training section, pre-processing techniques are usually done in order to give a more "suitable" data for the network. For instance it could happen that larger number override smaller ones, or cause saturation in hidden nodes, impeding process to learn. There are different pre-processing data techniques applied before training section, and different techniques exist for different problems. For example in a classification problem the pre-processing data will be different than in a fitting problem. The pre-processing techniques depend also on the size that we use in a network. If we have a small size data set available, the network will have small data to train and it could cause inability for the network to learn well the process. In this case it is recommended to get new data, if is possible, or interject random noise in the available data in order to generate a new one [10]. Vice versa, if we have a huge quantity of data different authors suggest to reduce the noise and transform data to permit at the network to better learn the main variable in the process [27] [76]. A main study in this field was conducted by Yu et al. (2007) in where they divided *data preparation* from *data pre-processing*. Most authors consider data pre-processing equivalent to data preparation. Actually, data pre-processing is a phase of data preparation. They divide data preparation in three phases: 1) pre-analysis data: in which data of interest are identified and collected; 2) pre-processing data: in which data are examined and analysed, and data post-analysis in which soma data are validated and adjusted. In their study many solutions were given to

decreases at each successful step (reduction of cost function) and increases otherwise. With $\mu = 0$ we have just the Newton's method, while with a large value of $\mu$ the algorithm is like a gradient descent with s small learning rate. For a discussion of this algorithm see [57][33].

face the recurring phase problems. For instance genetic algorithm is proposed as a solution of the data variable selection in the first phase, or decreasing (or increasing) data set in presence of overfitting (or underfitting) in the post-analysis phase. As far as pre-processing process is concerned, they individuate some recurrent issues that we can summarise in the table bellow:

| **Main Pre-Processing Issues** | **Solutions** |
|---|---|
| • Too much data | • Dara sampling |
| • Too little data | • Data re-gathering |
| • Missing data | • Data repairing |
| • Noisy data and outliers | • Data denoising |
| • Multi-scale data | • Data normalization |
| • Trending and Seasonal deta | • Detrending |
| • Nonstationary data | • Difference |

In financial time series data the present of nonstationarity process, trending and seasonality often occurs. And it was shown that presenting a trend in the time series data do not allow the ANN to perform very well, since non-linear transfer functions in the first layer constrain the model to the input range value [23] [92]. Similar studies demonstrated that seasonal elements have a significant impact on ANNs prediction [88]. For this reason data detrending and deseasonalization are requested by some authors as: Weigend et al. (1994), Tseng et al. (2002) and Moody (1995). A simple way that is widely used is by differencing or log-differencing the data. On the other hand some authors consider differencing as unnecessary step in order to forecasting with ANNs (see [5], [7]), and proposed as alternative to log transform the series [6]. However, at the moment there are no successful methods that can determinate which technique is the most suitable in a ANN [95] in order to detrend the series and, although its importance is known on tha ANNs application, this has yet to be fully investigated [6]. In order to eliminate the multi-scale issues it is a common practice in ANNs to normalize the series before it will be used in the network. Then the non linear activation functions used in the network have the ”squashing” role in restricting the network output, for instance into the boundary [0,1] or [-1,1]. So, in general data normalization helps to improve

the performance of neural network [55]. In term of implementation, this helps also not to exceed computer capacity when the range of the data is too large. Different techniques have been proposed depending on the specific problem (see Azoff (1994), Weigend et al. (1992) and Yu et al. (2007) for a survey). The choice strictly depends on the activation functions in the layers [6] The most used in literature and in practice is the *line scaling* method defined as:

$$I = a + \frac{(b-a)(x-x_{min})}{x_{max}-x_{min}} \tag{3.6}$$

This formula allows the data value $x$ to be transformed in the input data $I$ into the range $[a,b]$, where $x_{min}$ is the minimum value of the data series and $x_{max}$ is the maximum value of the same data series. $[a,b]$ is the minimal and the maximal values of the range. As the input, also the target are usually normalized.

In our study all the prices have been transformed into the range [-1,1] by the eq. 3.6, with $a = -1$ and $b = 1$, before being used in the network. The network output was then rescaled to the original scale [7]

## 3.3 Analysis and Result

Now we are going to see how the ANNs work with the data series saw in section 2.3. We have proceeded using MATLAB R2013b and in particular its implemented Neural Network Toolbox. The purposes of our implementation is focused: firstly on understanding how the ANNs work with commodity time series data. ANNs are used in different fields and among the most used there are: pattern classification, fitting data and cluster self organization (see [10] and [96]). For this reason we would like to test if ANNs could be used in the "commodities environment". Secondly on seeing how it works in forecasting non stationary data series. Different network architectures exists of ANNs, and different models are presented in literatures for forecasting use. In the practice, with temporal data, the generalized feed-forward backpropagation model

[6] Commonly used with logistic function is: $I = (x - x_{min})/(x_{max} - x_{min})$, where $I$ is a value into the range $[0,1]$, and $x_{max}$ and $x_{min}$ are the maximal and minimal value of the original data series.

[7] In many problems, as in the time series forecasting problem, the choice of the range $[a,b]$ is closely related to the activation functions used in the output nodes. Some authors scale the data to the range $[0.1, 0.9]$ or $[0.2, 0.8]$ based on the fact that the logistic functions have asymptotic limits. This limits may be cause "lie" value [3].

(or MLP) is not the most commonly used. Different variations on the general structure have been made and more are under studies. We are now proceeding with illustrating the structure of the static feed-forward backpropagation network and then we will see the more powerful dynamic and recurrent networks as a comparison.

**Feed-forward Network** The static feed-forward network is composed, as we saw in 1.2, by a input vector, a single or multiple layers of hidden neurons and a output layer. The input vectors have the corresponding target vectors, and the network tries to better approximate the function during the training. There are non recurrent connection in the layers, no directly connection from output to input, and no filters between the layers. The absence of these elements characterizes the so called *static network*. The input elements are directly multiply by the weights, summed and utilized by the transfer function $f$. The estimated outputs are calculated directly from the inputs nodes through feed-forward connections. An example of a two-layer feed-forward network is shown in Figure(3.1).

### 3.3.1 Dynamic Network

The category of dynamic networks, well known in signal theories such as *Layered Digital Dynamic Network* (LDDN), is characterized by the presence of feedback elements in the network (from a layer to another layer, or from a layer to the same layer) or for the presence of *filters*. The filters are a kind of "memory" of the neural system that allow the network to estimate the output not only by the current values, but also by the previous network values. The previous values can come from any layers of the network, depending on the specific purpose of the problem. When a network presents this kind of filters (also know as *buffers*) referring to a temporal data, as in our case, we are speaking about a *Time-Delay Neural Network* (TDNN).
When the network presents a feedback connection, it is also called *recurrent* network (RNN). In this kind of network weights have two different effects on the network output: one direct and one indirect. The direct effect is the same we have seen in the static feed-forward network: a change in the weights at

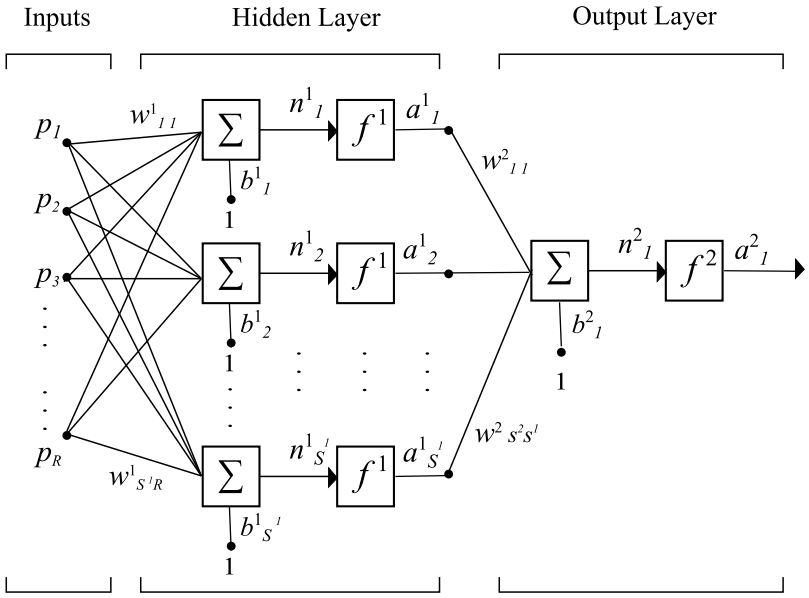



Figure 3.1: A simple Feed-Forward Neural Network, where $p_i$ is the input elements, $w$ represents the weight associate at each input, $f$ is the neuron transfer function, $n$ is the output of the sum operator and $a$ represents the network output.

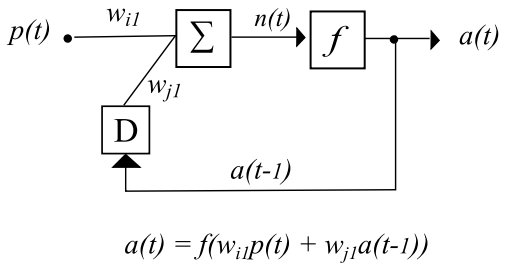



Figure 3.2: Recurrent Neural Network.

the current time step causes an immediate change in the output at the current time step. The indirect effect is due to the previous weight changed that are, in the recurrent network, also a function of the weight. Figure(3.2) can help to better understand this process.

In this figure we can represent the network output as:

$$a(t) = w_{i1}p(t) + w_{j1}a(t-1)$$

where $a(t)$ is the expected values at the time $t$, and it is influence from the current input $p(t)$ and the previous output $a(t-1)$. But $a(t-1) = w_{11}p(t-1)$, so the weights at the current time step are, indirectly, influenced by the weights at the previous time. This is also why, in the signal theory, a feed-forward dynamic networks are called *finite impulse response* (FIR), while a recurrent dynamic networks are called *infinite impulse response* (IIR): the response to an impulse does not become exactly zero past a certain point, but continues indefinitely.

**Focus Time-Delay Neural Network.** A common dynamic model used in temporal data is the *Focused Time-Delay Neural Network* (FTDNN) that adds in a feed-forward network a filter called *tapped delay line* (TDL). The filter is a $N$ dimensional vector composed by the input signal at the present time $t$ and $(N-1)$ delayed signal of the previous $(N-1)$ time. Figure(3.3) shows a TDL filter.

This filter is add in a FTDNN after the input vector, so all input signals go into the network with the $(N-1)$ unit delays and each unit is then weighted and summed. In this manner the network has a shot-term memory of the previous period and can analyse the data in a more efficient way [8]. A diagram of the

[8]Dynamic networks differ from static networks because they have a *memory*, so they are able to

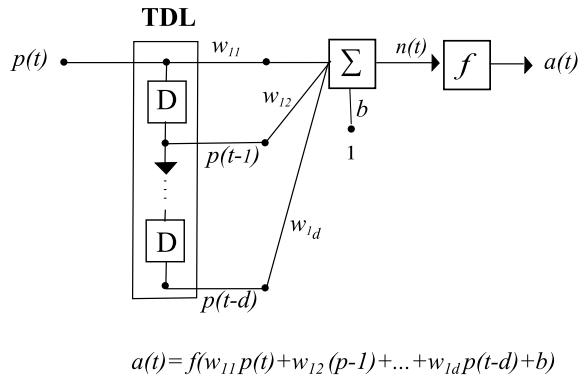



Figure 3.3: Tapped Delay Line Filter

FTDLL is shown in the Figure(3.4).

The network has one hidden layer and one output layer. The hidden layer commonly has logistic functions or hyperbolic tangent functions, to enable non-linearity of the network, while the output layer uses *Linear* transfer function in order to allow for the unknown output range of the series.

**Some Recurrent Network** Some example of recurrent network commonly used for temporal data is the *Elman neural network*. It belongs to the class of RNN models: it presents two-layer network with a feedback from the first layer output to the first layer input. This feedback connection permits the network to detect and generate time-varying patterns. The original Elman network has a feedback units with one-step delay. When the training starts the weights of the first layers will be initialized and the recurrent unit will be memorize those for re-input in the network in the next training step. Figure(3.5) shows a Elman network architecture.

Formally we can write Elman network mathematically as:

$$a(t-1) = f_h\Big[b_h + \sum_{h=1}^{S_2} w_{hi} f_i\Big(b_i + \sum_{j=1}^{R} w_{ij} p_j + \sum_{i=1}^{S^1} w_{ii} a_i(t-1)\Big)\Big]$$

where $p_j$ are the input elements, with $j = 1, 2, ..., R$; $w_{ij}, w_{ii}, w_{hi}$ are the weights of the connections between the input and hidden neurons, between the feed-

"learn" sequential or time-varying pattern [25]. Thanks to this characteristic, this kind of network are used in many different areas such as: communication system, power system, speech recognition and others. See [58]

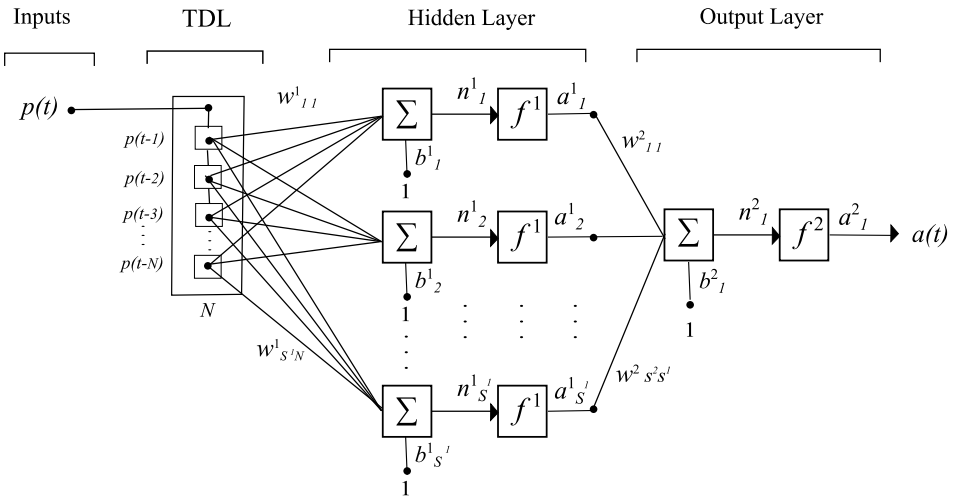



Figure 3.4: Focus Time-Delay Neural Network with a TDL filter with N-delays for the input vector. The input element $p(t)$ is calculated by the previous input values $[p(t-1), p(t-2), ..., p(t-N)]$ by the TDL filter. The network gives the output $a$ at time $(t)$.

back and the hidden neurons, and between the hidden layer and the output layer, respectively, with $i = 1, 2, ..., S^1$ represent the first layer neurons number and $h = 1, 2, ....S^2$ represent the second layer neurons number. $a_i(t-1)$ is the feedback from the prior period; $b_i$ and $b_h$ are the biases of the hidden layer and output layer, and $f_h()$ and $f_i()$ are the hidden and output neurons transfer functions. The original Elman network has *Tanh* transfer function in the first layer neurons, and *Linear* transfer function in the second hidden layer neurons. It is also possible to modify the numbers of delays in the recurrent connection and the transfer functions in the layers. The generalized model is known in the practice as *layer-recurrent network* (LRN).

### 3.3.2 Experiment Results

In our study we have chosen to work with the dynamic FTDNN model with a single TDL filter at the input to the first layer. We have applied a log transformation on the price as suggested by Balkin (2000) and then we have proceeded to determinate the number of input nodes and hidden nodes by trial and error. This testing is made with three principal training algorithms: the gradient de-

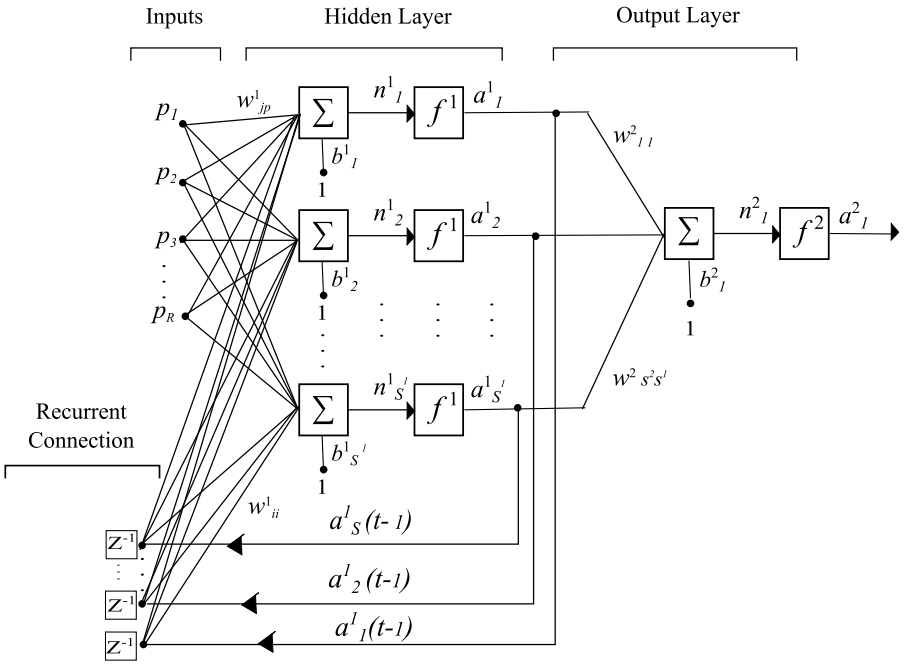



Figure 3.5: The architecture of Elman recurrent neural network. The inputs $\mathbf{p}$ at time $(t-1)$ are weighted and summed. The results $\mathbf{n}$ of the sum operators go into the input functions $f$. The $n$ outputs of the functions return beck as inputs of the first layer and are summed with the new inputs $\mathbf{p}$ at the time $(t)$. The $z^{-1}$ indicates the delay of the "signal" input $a(t-1)$. The output of the network is $a(t)$ and is calculated as $a(t) = f(\mathbf{w}_2 f(\mathbf{w_1 p} + \mathbf{w}_1 \mathbf{a}(t-1) + \mathbf{b}_1) + b_2)$, where $\mathbf{w}_1$ and $\mathbf{b}_1$ are the weights and biases of the hidden layer and $\mathbf{w}_2$ and $b_2$ are the weights and bias of the output layer.

scent with momentum, the conjugate gradient and the Levenberg-Marquardt. The data are divided in 2 parts: one for training, validation and testing, and another one for prediction analysis. The prediction is made on the last 360 days of each time series. The rest of the data is divided in blocks as: 70% for training the network, 15% for validation and 15% for testing. All data is transformed, by the line scaling method (see eq.3.6), into the range $[-1,1]$ before being used by the network, then the network output is transformed back to the original scale. The performance function of the training is the MSE. The training sections are repeated 30 times for any data set, and each training section has 150 epochs, but the system stops automatically the training when the validation set error increases for 6 times in a row. In this way overfitting problem is reduced (but not eliminated). At each time the weights and biases are initialized by the Nguyen-Windrow function. The network with the best performance is chosen to be used in the prediction analysis. The Table 3.1 reports the result of the testing sections for all four series. The table shows the mean of the performance of the thirty testing loops for different input and hidden nodes number.

We can see how the best performance for gas prices is reached with 5 input nodes, instead WTI in the tests with 2 input nodes. Brent presents good performance with both 2 and 10 nodes. Different testing was also made using more than 10 inputs node, but not significant result was reached. By trial and error we have also tried to find the best number of hidden nodes. Not significant difference occurred since 40 neurons. With more than 40 neurons the network tends to overfitting. In the Table 3.1 there are also reported the results given by the three different training algorithm used. We can see that the Levenberg-Marquardt algorithm is the one that gives the best performance in all four series. The conjugate gradient instead gives good result for WTI and NBP time-series, but did not perform as well on the Brent and HH prices. Less indicated for this problem is the standard gradient descent algorithm with momentum, that gives a performance that is significantly distant from the other two training algorithms. The Figure(3.6) shows the regression plots of the testing sections for the four series, with the best network selected. In this figures the network outputs are plotted versus the respectively targets values. The

Table 3.1: Training Performance Results

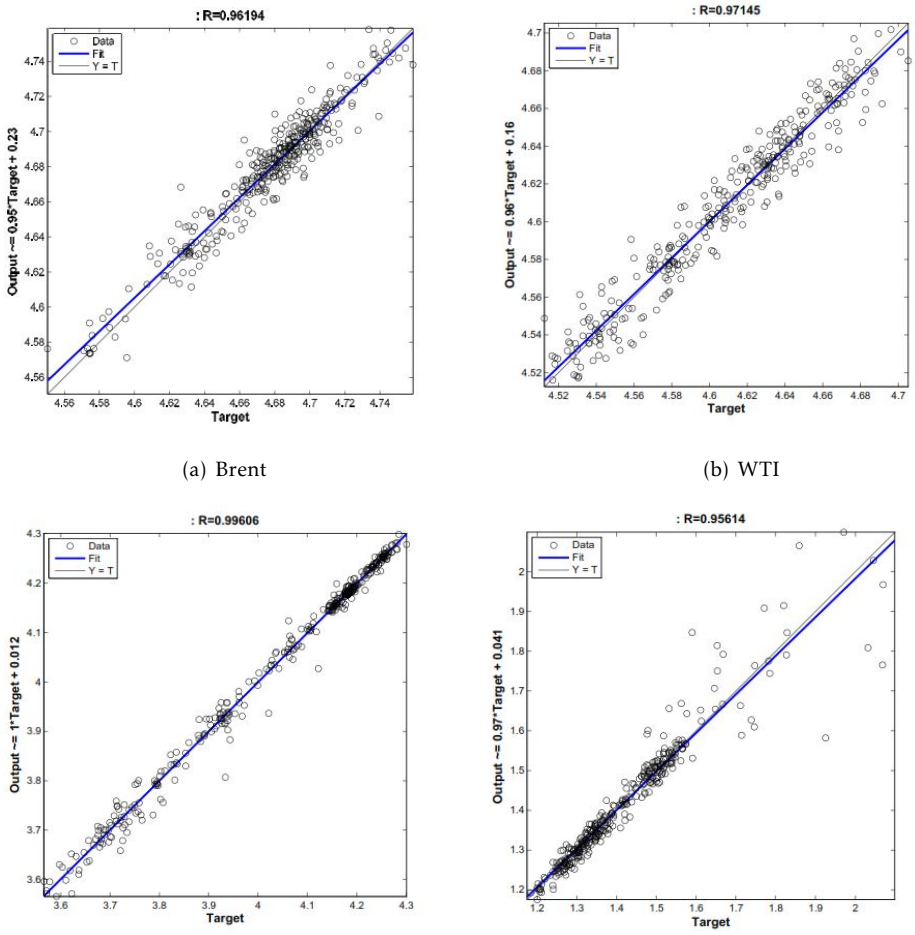
| IN | HN | Brent | WTI | NBP | HH |
|---|---|---|---|---|---|
| **Gradient Descent Algorithm with Momentum** | | | | | |
| 1:2 | 5 | 0.0057 | 0.0634 | 0.0230 | 0.6652 |
| 1:2 | 10 | 1.1470 | 0.1737 | 0.1727 | 0.2515 |
| 1:2 | 20 | 0.0261 | 0.5770 | 0.3782 | 0.0897 |
| 1:5 | 5 | 0.0014 | 0.2613 | 0.0308 | 0.0479 |
| 1:5 | 10 | 0.0057 | 0.044 | 0.0040 | 0.0254 |
| 1:5 | 20 | 0.6091 | 0.0671 | 0.0295 | 0.1686 |
| 1:10 | 5 | 0.0637 | 0.2507 | 0.0032 | 0.0041 |
| 1:10 | 10 | 0.0368 | 0.0470 | 0.0021 | 0.0324 |
| 1:10 | 20 | 0.0123 | 0.0238 | 0.0054 | 0.0161 |
| **Conjugate Gradient Algoritm** | | | | | |
| 1:2 | 5 | 0.0018 | 5.3196e-04 | 7.2333e-04 | 0.0345 |
| 1:2 | 10 | 9.5972e-04 | 5.2878e-04 | 4.3961e-04 | 0.0025 |
| 1:2 | 20 | 0.0177 | 4.6008e-04 | 3.9301e-04 | 0.0395 |
| 1:5 | 5 | 0.0015 | 3.6290e-04 | 7.8017e-04 | 0.0033 |
| 1:5 | 10 | 0.0045 | 5.1413e-04 | 6.2422e-04 | 0.0016 |
| 1:5 | 20 | 0.0072 | 3.1560e-04 | 3.5001e-04 | 0.0026 |
| 1:10 | 5 | 0.0015 | 6.0924e-04 | 7.1346e-04 | 0.0011 |
| 1:10 | 10 | 0.0019 | 4.4511e-04 | 3.6349e-04 | 0.0014 |
| 1:10 | 20 | 0.0013 | 5.3832e-04 | 3.9048e-04 | 0.0023 |
| **Levenberg-Marquardt Algoritm** | | | | | |
| 1:2 | a 5 | 4.5027e-04 | 2.6656e-04 | 3.5858e-04 | 9.4426e-04 |
| 1:2 | 10 | 4.1663e-04 | 2.6714e-04 | 3.4884e-04 | 9.6686e-04 |
| 1:2 | 20 | 4.0767e-04 | 2.6739e-04 | 3.5617e-04 | 9.3086e-04 |
| 1:5 | 5 | 4.1313e-04 | 2.6980e-04 | 3.3946e-04 | 9.2012e-04 |
| 1:5 | 10 | 5.2715e-04 | 2.7976e-04 | 3.3933e-04 | 8.8957e-04 |
| 1:5 | 20 | 4.0539e-04 | 2.7161e-04 | 3.5691e-04 | 8.4833e-04 |
| 1:10 | 5 | 4.1437e-04 | 2.7439e-04 | 3.4536e-04 | 9.1778e-04 |
| 1:10 | 10 | 4.1802e-04 | 2.7352e-04 | 3.4266e-04 | 8.4327e-04 |
| 1:10 | 20 | 4.0881e-04 | 2.7418e-04 | 3.5348e-04 | 9.6596e-04 |

Note:The performance is calculated by MSE.

IN represents the number of input nodes.

The HN is the numbers of hidden nodes

small black solid line represents the ideal fit, when output is equal to target ($Y = T$). Instead the blue line is the best linear fit between network outputs and targets values. The plot gives also the correlation coefficient $R^2$ between the outputs and the targets [9]. We can see how the outputs values of the four networks track the targets reasonably well, and also all the R-values are around 0.9.




(a) Brent (b) WTI

(c) NBP (d) HH

Figure 3.6: Training Section Regression Plots

For the prediction test we have chosen the following network:

[9] The correlation coefficient $R^2$ is used in linear regression model for indicate how well a variation in the $Y$ (regressand) is explained by the $X_i$ regressors. With $R^2 = 1$ the model can "explain" perfectly the regressand $Y$ variation. When $R^2 = 0$ the model can not explain $Y$ variation.

- **Brent** : $(1:5,20)$ $\longrightarrow$ 5 input nodes, 20 hidden nodes.
- **WTI** : $(1:2,5)$ $\longrightarrow$ 2 input nodes, 5 hidden nodes.
- **NBP** : $(1:5,5)$ $\longrightarrow$ 5 input nodes, 5 hidden nodes.
- **HH** : $(1:5,20)$ $\longrightarrow$ 5 input nodes, 20 hidden nodes.

Now we will proceed to analyse how these networks works with new simple series. Until now we haveused a so called "open-loop" network. All the training section, validation and testing included, are done in open-loop (also know as *series-parallel architecture*). In the open-loop, the inputs of the network come directly from outset: in this case the price at the time $t$, for instance, is an external information that is putted into the network as new element. When the FDTNN network in in open-loop mode, it performs one-step ahead prediction: as we can see in Figure(3.4) the price value at the time $t$ is predicting from the past prices $[(t-1),(t-2),...,(t-N)]$. When we close the loop, the output of the network are used as input into the system. At the same time, just closing the loop is, in general, insufficient because the close-network feeds back the output with errors instead of the target. Using the target to complete the design of close-network may be necessary to reduce the fed back error. Just for better understand how the prediction works, we report the Matlab code here:

```
% Close-loop for Multi-step Prediction
[netc,xi]=closeloop(net,Xf);
[Y2,xf]=netc(X2,xi);
```

In the first raw the command `closeloop=[net,Xf]`, closes the open-network `net` from the final input `Xf`, that is our final price of the testing section. This command returns `[netc,xi]`, where `netc` is the close-network and `xi` is the input delay vector used in the TDL. We can represent `xi` as:

$$xi = p(t-1), p(t-2), ..., p(t-N)$$

In the second row the program uses the close-network on the series `X2`, that is our second series we use for prediction. `X2` is necessary to reduce the fed back errors of the network. The close-network uses its own values as input,

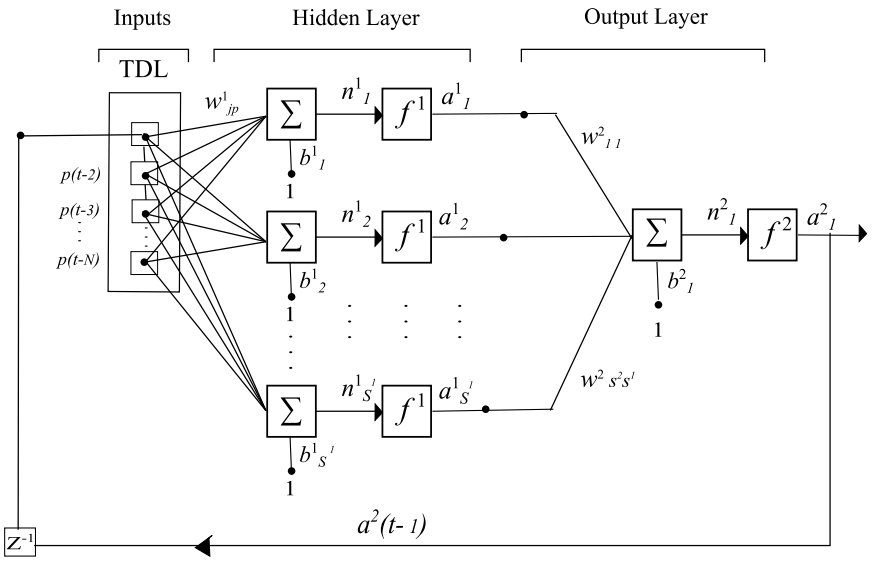



Figure 3.7: A Closed FTDLL Neural Network

but correct its past output values with the real past values. Figure(3.7) shows how the close-loop works.

| **Prediction Performance** | | |
|---|---|---|
| | **MSE** | **RMSE** |
| **Brent** | 1.0008e-04 | 0.0100 |
| **WTI** | 1.2605e-04 | 0.0112 |
| **NBP** | 3.3711e-04 | 0.0183 |
| **HH** | 0.0026 | 0.0481 |

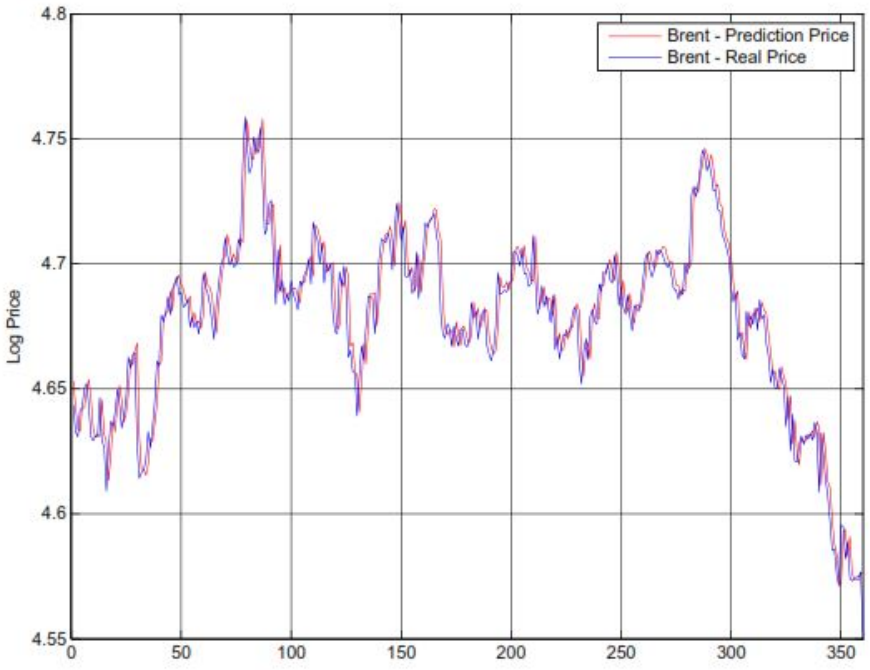
Table 3.2: My caption

How we can see, when we closed the loop, the network became a recurrent neural network with a TDL filter. It looks seems at the Elman model we saw in paragraph 3.3.1, but with the difference that the recursive element is not from the hidden layer but form the output layer. The prediction results are shown bellow. For the Brent, WTI and NBP the performances are good as was in the test, even better for Brent and WTI series. Worse than the prior performance test is the results for HH series. However the predicted prices are so close to the real prices. We can see from the Figure(3.8) and Figure(3.9) show how the neural network forecasting. We can see how the curve follows so well the real

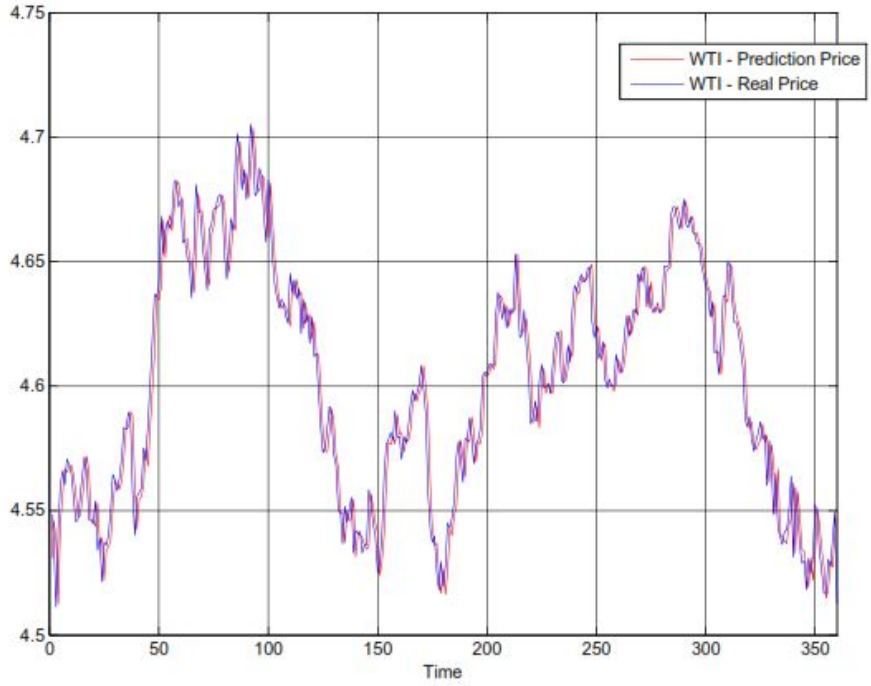
curve prices. The Table**??** reports the MSE and the *Root Mean Squared Error* (RMSE) values of the prediction test.

## 3.4 Conclusion

Artificial Neural Networks are used in different fields of application: from biological use, to telecommunication application, to financial tools. In this paper a *Focused Time Delay Neural Network* was used to forecasting commodities prices. In particular were analysed Brent and WTI prices for crude oil and HH and NBP prices for natural gas. We have proceeded with a description on how the ANNs work and how their structures are composed. Then we have made an overview on the crude oil and natural gas commodities market. We saw the different elements that can influence these commodities price and the reflection that their price fluctuations have on financial and economy markets. Furthermore the time series are characterized by non-stationarity, therefore they are difficult to be represented and forecast by traditional linear models. The use of an ANN is indicated in this kind of environment. The literature on the use of ANNs for forecasting is vast, and different approaches are proposed for the same problem. We have summarized most of the main elements that need to be chosen to create an ANN and the way to select them. Different architecture models exist for time series prediction. The choice of the FTDNN model seems to be correct. The TDL filter gives a short memory to the network and allows it to better recognise the fluctuation by the ”training” of the past data. At the same time, the past values, are also the limit of this system. In fact, we have seen that if new fluctuations occur, out of the ”range” of the past values, the network is not able to respond well to the new target, because the weights values are limited on the past data, and new training is needed. Even this limit, the results given permit to conclude that a FTDNN can be used in forecasting crude oil and natural gas. But, thanks to its flexibility on the parameter, the same model can be extended to the commodities market.

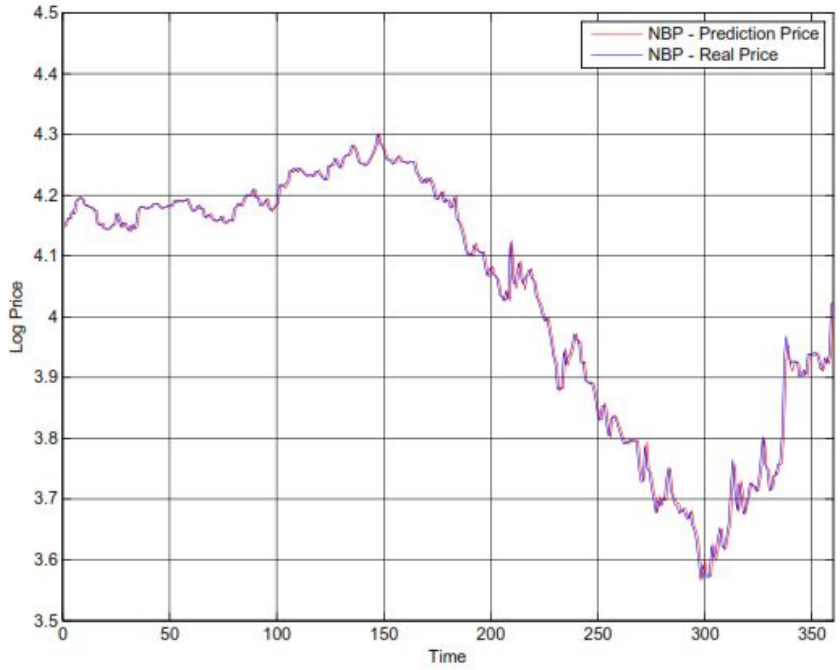



(a) Brent-Forecasting

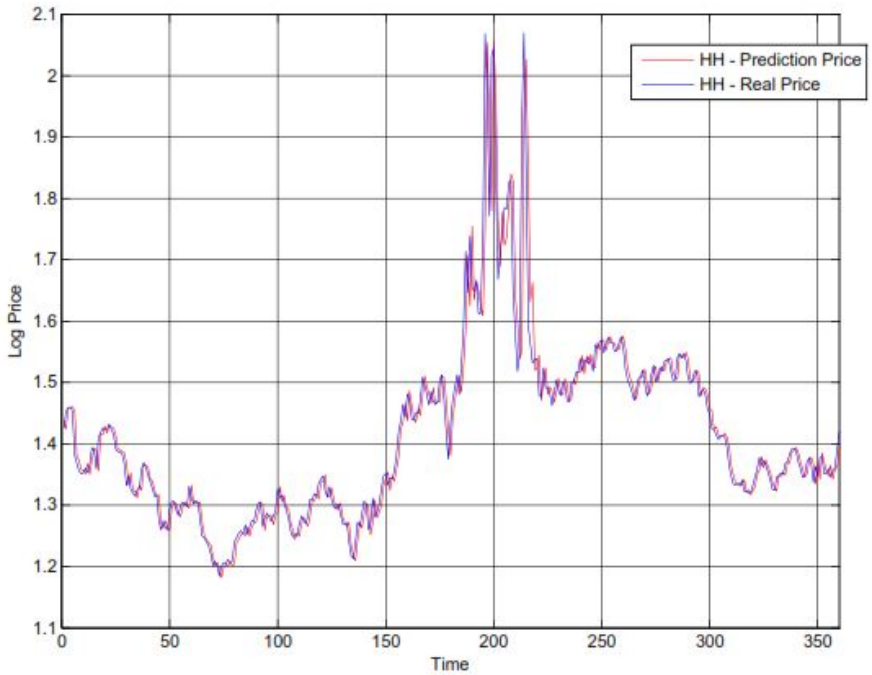



(b) WTI-Forecasting

Figure 3.8: Crude Oil Forecasting with FTDNN




(a) NBP-Forecasting




(b) HH-Forecasting

Figure 3.9: Natrual Gas Forecasting with FTDNN

# Bibliography

[1] Armano G., Marchesi M., Murru A., 2005. A hybrid genetic-neural architecture for stock indexes forecasting. Information Science, 170, 3-33.

[2] Armstrong J. S., Fildes R., 1995. Correspondence: On the selection of error measures for comparisons among forecasting methods. Journal of Forecasting 14, 67-71.

[3] Azoff M. E., 1994. Neural network time series forecasting of financial markets. Chichester: Wiley.

[4] L.J., Griffin J.M., 2006. Testing for market integration: crude oil, coal, and natural gas. Energy Journal 27,55-71.

[5] Balkin S. D., 1999. Stationarity Concern When Forecasting using Neural Networks. INFORMS Presentation, Philadelphia, PA.

[6] Balkin S. D., Ord J. K., 2000. Automatic neural network modelling for univariate time series. International Journal of Forecasting 16,509-515.

[7] Balkin S. D., 2000. A Statistical Implementation of Feedforward Neural Networks for Univariate Time Series Forecasting. The Pennsylvania State University, Department of Management Science & Information System, Unpublished Ph.D. dissertation.

[8] Barcella M.L., May 1999. The pricing of gas. Oxford Institute for Energy Studies, Oxford.

[9] Barron A. R., 1994. A comment on "Neural networks: A review from a statistical perspective". Statistical Science 9 (1), 33-35.

[10] Basheer I. A., Hajmeer M., 2000. Artificial neural networks: fundamentals, computing, design and application. Journal of Microbiological Methods 43, 3-31.

[11] Bertsekas D., Nonlinear programming, pag.156-157.

[12] BP Statistical Review of World Energy, June 2013, $63^{rd}$ edition. www.bp.com/statisticalreview.

[13] Castiglione F., 2001. Forecasting price increments using an artificial neural network. Advance in Complex System 4(1), 45-46.

[14] Cera, 2009. Gas from shale: potential outside North America?. Cambridge Energy Research Associate Report.

[15] Cera, 2010. Fueling North America's Energy Future. Cambridge Energy Research Associate Report.

[16] Chakraborty K., Mehrotra K., Mohan C. K., Ranka S., 1992. Forecasting the behavior of multivariate time series using neural networks. Neural Networks 5, 961-970.

[17] Charaniya N.A., Dudul S.V., 2013. Design of Neural Network Models for Daily Rainfall Prediction. International Journal of Computer Application, 61,Jan-2013.

[18] Chen S.S., Chen H.C., 2007. Oil prices and real exchange rates. Energy Economics, 29, 390-404.

[19] Chen Y. H., Yang B., Dong J., & Abraham A. (2005). Time-series forecasting using flexible neural tree model. Information Science 174, 219-235.

[20] Chester D. L., 1990. Why two hidden layers are better than one? In: Proceeding of the International Joint Conference on Neural Networks, 1265-1268.

[21] Cheung K.H., Szeto K. Y., Tam K. Y., 1996. Maximum-entropy approach to identify time-series lag structure for developing intelligent forecasting system. International Journal of Computational Intelligence and Organization 1 (2), 94-106.

[22] Clements M. P., Hendry D. F., 1993. On the limitations of comparing mean square forecast errors. Journal of Forecasting 12, 615-637.

[23] Cottrell M., Girard B., Girard Y., Mangeas M., Muller C., 1995. Neural modelling for time series: a statistical stepwise method for weight elimination. IEEE Transactions on Neural Networks 6 (6), 1355-1364.

[24] Cybenko G., 1989. Approximation by superpositions of a sigmoidal function. Mathematical Control Signals System 2, 303-314.

[25] Demuth H., Beale M., Hagan M., 2009. Neural Network Toolbox User Guide. The MathWorks, Inc., Chapter 5.

[26] Dennis J. E., & Schnabel R. B., 1983. Numerical Methods for Unconstrained Optimization and Nonlinear Equation. Englewood Cliffs, NJ: Prentice-Hall.

[27] Dowla F. U., Rogers L. L., 1995. Solving Problems in Environmental Engineering and Geoscience With Artificial Neural Networks. MIT Press, Cambridge, MA.

[28] Erdős P., 2012. Have oil and gas prices got separated?. Energy Policy, 49, 707-718.

[29] Fletcher R., & Reeves C. M., 1964. Function minimization by conjugate gradients. Computer Journal 7, 149-154.

[30] Foster E. A., 2002. Commodity futures price prediction, an artificial intelligence approach. Final paper in Master of Science at the University of Georgia.

[31] Gandolphe S.C., 2005. LNG cost reductions and flexibility in LNG trade add to security of gas supply. Energy Price & Taxes, 1st Quarter 2005-xxix.

[32] Gilava T., 2010. Econometric of crude oil markets.

[33] Hagan M. T., Menhai M., 1994. Training feed-forward network with the Marquardt algorithm. IEEE Transaction on Neural Networks 5 (6), 989-993.

[34] Hagan M. T., Demuth H. B., Beale M.,1996. Neural Network Design. Boston, MA: PWS Publishing, Chapter 11.

[35] Harmsen R., Jepman C., 27 January 2011.. North West European gas market: integrated already. European Energy Review.

[36] HamzaCebi C., Akay D., Kutay F., 2009. Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. Expert System with Application 36, 3839-3844.

[37] Hassoun M. H., 1995. Fundamentals of Artificial Neural Network. MIT Press, Cambridge, MA.

[38] Hecht-Nielsen R., 1990. Neurocomputing. Addison-Wesley, Menlo Park, CA.

[39] Hill T., Marquez L., O'Connor M., Remus W., 1994. Artificial neural networks for forecasting and decision making. International Journal of Forecasting 10, 5-15.

[40] Hornik K. M., Stinchcombe M., White H., 1989. Multilayer feedforward networks are universal approximators. Neural Network 2 (5), 359-366.

[41] Huang B.N., Hwang M.J., Peng H.P., 2005. The asymmetry of the impact of oil price shocks on economic activities: an application of the multivariate threshold model. Energy Economics, 27, 455476.

[42] Jaruszewicz M., & Mandziuk J., 2004. One day prediction of NIKKEI Index considering information from other stock markets. In International conference on artificial intelligence and soft computing ICAISC 2004.

[43] Kaastra I., & Boyd M., 1996. Designing a neural network for forecasting financial and econometric time series. Neurocomputing 10, 215-236.

[44] Kang S., 1991. An Investigation of the Use of Feedforward Neural Networks for Forecasting. Ph.D. Thesis, Kent State University.

[45] Khashei M, Bijari M., 2010. An artificial neural network *(p,d,q)* model for timeseries forecasting. Expert System with Application 37, 479-489.

[46] Kim J. H., et al., 2003. Stock price prediction using backpropagation neural networks in KOSPI. In International conference on artificial intelligence IC-AI'03, pp. 200-203.

[47] Kim K. J., & Lee W. B., 2004. Stock market prediction using artificial neural networks with optimal feature transformation. Neural Computing and Applications 13(3), 255-260.

[48] Klimasauskas C. C., 1991. Applying neural network. Part 3: Training a neural network, PC-AI, May/June, 20-24.

[49] Kumar S., Know H.T., Choi K.H., Cho J.H., Lim W., Moon I., 2011.Current status and future projections of LNG demand and supplies: A global prospective. Energy Policy,39,4097-4104. Elsevier.

[50] Kohzadi N., Boyd M. S., Kermanshahi B., Kaastra I., 1996. A comparison of artificial neural network and time series model for forecasting commodity price. Neurocomputing 10, 169-181.

[51] Lachtermacher G., Fuller J. D., 1995. Backpropagation in time-series forecasting. Journal of Forecasting 14, 381-393.

[52] Lee K., Ni S., Ratti R.A., 1995. Oil shocks and the macroeconomy: the role of price variability. The Energy Journal, 16, 3956.

[53] Lin X., Yang Z., Song Y., 2009. Short-term stock price prediction based on echo state networks. Expert System with Application 36, 7313'7317.

[54] Lippmann R. P., 1987. An introduction to computing with neural nets. IEEE ASSP Magazine, April,4-22.

[55] Lou M., 1993. Preprocessing data for neural networks. Technical Analysis of Stocks & Commodities Magazine.

[56] Marin D., Varo A., & Guerrero J. E., 2007. Non-linear regression methods in NIRS quantitative analysis. Talanta 72, 28-42.

[57] Marquardt D. W., 1963. An algorithm for the least-square estimation of nonlinear parameters. Journal of the Society for Industrial and Applied Mathematics 11 (2), 431-441.

[58] Medsker L. R., Jain L. C., 2000. Recurrent neural networks: design and application. Boca Raton, FL: CRC Press.

[59] Moody J., 1995. Economic forecasting: challenges and neural network solution.Proceeding of International Symposium on Artificial Neural Networks.

[60] Morton, Michael Quentin. Unlocking the Earth: A Short History of Hydraulic Fracturing. GeoExpro 10 (6). Retrieved 27 February 2014.

[61] Neumann A., 2009. Linking natural gas markets-Is LNG doing its job? Energy Journal 30, 187-200.

[62] Oberndorfer U., 2009. Energy prices, volatility, and the stock market: Evidence from the Eurozone. Energy Policy, 37, 5787-5795.

[63] OPEC official website. http://www.opec.org/opec_web/en/about_us/25.htm

[64] Panella M., Barcellona F., Santucci V., D'Ecclesia R. L., 2012. Neural Network to Model Energy Commodity Price Dynamics. Conference paper at the Proceedings of 30th USAEE/IAEE North American Conference.

[65] Panagiotidis T., Rutledge E., 2006. Oil and gas markets in the UK: evidence form a cointegrating approach. Energy Economics, 29, 329-347.

[66] Pindyck R.S., 1999. The long-run evolution of energy prices. The Energy Journal, 20(2), 1-28.

[67] Refenes A. P. N., Burgess A. N., & Bentz Y., 1997. Neural networks in financial engineering: A study in methodology. IEEE Transactions on Neural Networks 8(6), 1222-1267.

[68] Regnier E.,2007. Oil and energy price volatility. Energy Economics, 29, 405-427.

[69] Reuters Agency, 27 Aug. 2014. EURO COAL- European prices edge lower on weaker gas market. http://af.reuters.com/article/energyOilNews/idAFL5N0QX4CP20140827.

[70] Rumelhart D., & McClelland J., 1986. Parallel distribution processing. Cambridge, MA: MIT Press.

[71] Sadorsky P., 1999. Oil price shocks and stock market activity. Energy Economics, 21, 449-469.

[72] Sadorsky P., 2003. The macroeconomic determinants of technology stock price volatility. Review of Financial Economics, 12, 191205.

[73] Serletis A., Herbert J., 1999. The message in North American energy prices. Energy Economics 21, 471-483.

[74] Siliverstovs B., L'Hégaret G., Neumann A., von Hirschhausen C., 2005. International market integration for natural gas? A cointegration analysis of price in Europe, North America and Japan. Energy Economics, 27, 603-615.

[75] Sharda R., Patil R. B., 1992. Connectionist approach to time series prediction: An empirical test. Journal of Intelligent Manufacturing 3m 317-323.

[76] Swingler K., 1996. Applying neural networks: a practical guide. Academic Press, New York.

[77] Tan C. N. W., 2001. Arificial neural networks: Applications in financial distress prediction and foreign exchange trading.

[78] Tang Z., Almeida C., Fishwick P. A., 1991. Time series forecasting using neural networks vs Box-Jenkins methodology. Simulation 57 (5), 303-310.

[79] Tang Z., Fishwick P. A., 1993. Feedforward neural nets as models for time series forecasting. ORSA Journal on Computing 5(4), 374-385.

[80] Tseng F. M., Yu H. C., Tzeng G. H., 2002. Combining neural network model with seasonal time series ARIMA model. Technological, Forecasting and Social Change 69, 71-87.

[81] US Energy Information Administration, 2014. Technically Recoverable Shale Oil and Shale Gas Resources: An Assessment of 137 Shale Formations in 41 Countries Outside the United States; http://www.eia.gov/analysis/studies/worldshalegas/.

[82] US Energy Information Administration,23 Oct 2013. North America leads the world in production of shale gas. http://www.eia.gov/todayinenergy/detail.cfm?id=13491.

[83] US Energy Information Administration,25 Nov 2014. Electricity Monthly Update. http://www.eia.gov/electricity/monthly/update/resource_use.cfm#tabs_con-3.

[84] US Energy Information Administration, Sep. 2014. International Energy Outlook 2014. http://www.eia.gov/forecasts/ieo/.

[85] FERC Order 436: Open Access Blueprint (1985). http://www.eia.gov/oil_gas/natural_gas/analysis_publications/ngmajorleg/ferc436.html.

[86] Villar J.A., Joutz F.L., 2006. The relationship between crude oil and natural gas prices. Energy Information Agency, Office of Oil and Gas.

[87] Weijermars R.,2011. Strategy implications of world gas market dynamics. Energy Strategy Reviews 1,66-70. Elsevier.

[88] Weigend A. S., Huberman B. A., Rumelhart D. E., 1992. Predicting sunspots and exchange rates with connectionist networks. In: Casdagli M., Eubank S., (Eds.), Nonlinear Modeling and Forecasting. Addison-Wesley, Redwood City, CA, pp. 395-432.

[89] Weigend A. S., Gershenfeld N. A., 1994. Time Series Prediction: Forecasting the Future and Understanding the Past. Addison-Wesley.

[90] Wong F. S., 1991. Time series forecasting using backpropagation neural networks. Neurocomputing 2, 147-159.

[91] Yudong Z., Lenan W., 2009. Stock market prediction of S&P 500 via combination of improved BCO approach and BP neural network. Expert Systems with Applications 36, 8849-8854.

[92] Yu L., Wang S., Lai K. K., 2007. Foreign-exchange-rate forecasting with artificial neural networks. Springer Science + Business Media.

[93] Zhang G., Patuwo E., Hu M. Y., 1998. Forecasting with artificial neural networks: The state of art. International Journal of Forecasting 14, 35-62.

[94] Zhang G. P., Patuwo B. E., Hu M. Y., 2001. A simulation study of artificial neural networks for nonlinear time-series forecasting. Computers & Operations Research 28, 381-896.

[95] Zhang G. P., & Qi M., 2005. Neural network forecasting for seasonal and trend time series. European Journal of Operational Research 160 (2), 501-514.

[96] Zurada J. M., 1992. Introduction to Artificial Neural Systems. West Publishing Company.