

T.C.
MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES IN
PURE AND APPLIED SCIENCES

COMPARISON OF ARTIFICIAL NEURAL NETWORKS AND
TRADITIONAL METHODS FOR ELECTRICITY
CONSUMPTION FORECASTING

Tuğba YILDIRIM

THESIS
FOR THE DEGREE OF MASTER OF SCIENCE
IN
ENGINEERING MANAGEMENT

SUPERVISOR
Yrd. Doç. Dr. Bahar SENNAROĞLU

İSTANBUL 2010

T.C.
MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES IN
PURE AND APPLIED SCIENCES

COMPARISON OF ARTIFICIAL NEURAL NETWORKS AND
TRADITIONAL METHODS FOR ELECTRICITY
CONSUMPTION FORECASTING

Tuğba YILDIRIM
(141102420050034)

THESIS
FOR THE DEGREE OF MASTER OF SCIENCE
IN
ENGINEERING MANAGEMENT

SUPERVISOR
Yrd. Doç. Dr. Bahar SENNAROĞLU

İSTANBUL 2010

ACKNOWLEDGEMENT

I would like to thank to my advisor, Asst.Prof.Dr.Bahar Sennaroğlu for her guidance and encouragement.

Needless to mention, all members of my family deserve the biggest appreciation for their support and understanding throughout my education life.

Tuğba Yıldırım

CONTENTS

	<u>PAGE NUMBER</u>
ACKNOWLEDGEMENT.....	i
CONTENTS.....	ii
ABSTRACT.....	vi-vii
ABBREVIATIONS.....	viii
FIGURES.....	x
TABLES.....	xiii
CHAPTER I ENERGY RESOURCES AND ELECTRICITY	
I.1 GENERAL BACKGROUND.....	1
I.2. ENERGY RESOURCES OF TURKEY.....	1
I.3. NON-RENEWABLE ENERGY AND ELECTRIC.....	2
I.3.1. Coal.....	2
I.3.2. Oil.....	3
I.3.2. Natural gas.....	3
I.3.3. Nuclear Power.....	3
I.4. RENEWABLE ENERGY AND ELECTRIC.....	4
I.4.1. Hydropower.....	4
I.4.2. Wind Power.....	5
I.4.3. Solar Energy	6
I.4.4. Geothermal.....	6
I.4.5. Biogas.....	7
I.5. HISTORICAL DEVELOPMENT OF ELECTRICITY IN TURKEY..	7
I.5.1. Development of electricity consumption in turkey.....	9
I.6. ELECTRICAL ENERGY CONSUMPTION AND PRODUCTION OF TURKEY VERSUS WORLD.....	9
CHAPTER II. ARTIFICIAL NEURAL NETWORK.....	10
II.1. HUMAN BRAIN.....	11
II.2. MODELS OF NEURON.....	12

II.3. A FRAMEWORK FOR DISTRIBUTED REPRESENTATION.....	16
II.3.1. Processing Units.....	17
II.3.2. Connections Between Units.....	17
II.3.3. Activation and Output Rules.....	18
II.4. TYPES OF ACTIVATION FUNCTION.....	19
II.5. NETWORKS WITH LINEAR ACTIVATION FUNCTIONS: THE DELTA RULE.....	21
II.6. THE GENERALIZED DELTA RULE.....	22
II.7. NETWORK TOPOLOGIES.....	22
II.8. TRAINING OF ARTIFICIAL NEURAL NETWORKS.....	23
II.8.1. Paradigms Of Learning.....	23
II.8.2. Modifying Patterns Of Connectivity.....	23
II.9. NETWORK SIZING.....	24
II.10. WEIGHTS AND LEARNING PARAMETERS.....	25
II.11. LEARNING PROCESS.....	25
II.11.1. Hebbian Learning.....	25
II.11.2. Competitive Learning.....	27
II.11.3. Boltzmann Learning.....	28
II.11.4. Learning With a Teacher.....	28
II.11.5. Learning Without a Teacher.....	30
II.11.6. Learning Tasks.....	33
II.12. NETWORK ARCHITECTURES.....	34
II.12.1. Perceptron Learning Rule And Convergence Theorem..	36
II.13. FEEDFORWARD NEURAL NETWORK.....	36
II.13.1. Single-Layer Perceptron.....	38
II.13.2. Multilayer Perceptrons.....	39
II.13.3 The Adaptive Linear Element (ADALINE).....	43
II.13.4 MADALINE.....	46
II.13.5 Backpropagation.....	47
II.13.6 Exclusive-Or Problem.....	47
II.14. RECURRENT NETWORKS.....	48

II.14.1. The Elman Network.....	49
II.14.2. The Jordan Network.....	51
II.14.3. The Hopfield Network.....	51
II.15. KOHONEN NETWORK.....	52
II.16. PRINCIPAL COMPONENT NETWORKS.....	52
II.17. THE NONLINEAR AUTOREGRESSIVE NETWORK WITH EXOGENOUS INPUTS (NARX).....	53
CHAPTER III FORECASTING METHODS.....	57
III.1. DECOMPOSITION METHODS FOR TIME-SERIES FORECASTING.....	57
III.2. EXPLANATORY VERSUS TIME SERIES FORECASTING.....	58
III.3. TYPES OF MODELS.....	59
III.3.1. Types Of Patterns In Time-Series Data.....	61
III.4. SIMPLE TIME-SERIES SMOOTHING METHODS.....	62
III.5. LEAST SQUARES ESTIMATES.....	62
III.6. SMOOTHING METHODS.....	64
III.6.1. Averaging Methods.....	65
III.6.1.1. The Mean.....	65
III.6.1.2. Moving Averages.....	66
III.6.2. Exponential Smoothing.....	69
III.6.3. Other Smoothing Methods.....	72
III.7. REGRESSION METHODS.....	73
III.7.1. Simple Regression.....	74
III.7.2. Multiple Regression.....	76
III.8. FUNDAMENTALS OF TIME-SERIES ANALYSIS.....	76
III.9. ADVANCED TIME-SERIES METHODS.....	77
III.10. THE BOX-JENKINS METHOD.....	78
III.11. MULTIVARIATE TIME-SERIES ANALYSIS.....	79
III.12. STATIONARITY.....	80
III.12.1. Removing Non-Stationarity In a Time Series.....	80
III.12.2. A Random Walk Model.....	80
CHAPTER IV FORECASTING METHODS.....	82
IV.1. ARTIFICIAL NEURAL NETWORK MODELS.....	82
IV.1.1. Multilayer Perceptron Models (MLP).....	83

IV.1.1.1. First MLP Model (MLP-14).....	84
IV.1.1.2. Second MLP Model(MLP-21).....	87
IV.1.2. ELMAN Network Models.....	89
IV.1.2.1. First Elman model (ELMAN-1).....	90
IV.1.2.2. Second Elman model (ELMAN-34).....	93
IV.1.3. Radial Basis Function Model (RBF).....	96
IV.1.4. NARX Models.....	100
IV.1.4.1. First NARX Model (NARX-2).....	100
IV.1.4.2. Second NARX Model (NARX-9).....	104
IV.2 FORECASTING WITH REGRESSION MODEL.....	107
IV.3 FORECASTING WITH ARIMA MODEL.....	112
CHAPTER V CONCLUSION.....	118
REFERENCES.....	120
APPENDIX.....	123
CURRICULUM VITAE	

ÖZET

ELEKTRİK TÜKETİMİ TAHMİNİ İÇİN YAPAY SİNİR AĞLARI VE GELENEKSEL METODLARIN KARŞILAŞTIRILMASI

Elektrik hayat kalitemizin temel öğelerinden biridir. Yaşamımızı devam ettirebilmemiz için ve kesintisiz enerji tedarikine ihtiyaç duyarız. Elektrik modern ekonomilerde tüm sektörler için anahtar maddedir.

Artan elektrik enerjisi talebi problemlerinin çözümlerinde çeşitli metotların kullanılmasına ihtiyaç duyulur. Enerji kaynakları üretimi ve tüketimi tahminleri Türkiye'ye uygun politikalar izlemeye yardımcı olur.

Bu çalışmada, uzun dönemli elektrik tüketimi tahminleri yapılırken yapay sinir ağları ve bazı zaman serileri modellerinin teorik ve uygulamalı olarak incelenmesi amaçlanmıştır. Çalışma dahilinde yapay sinir ağları metodolojisi ve zaman serileri analizinin nasıl yapılacağı teorik olarak açıklanmış, ardından uygulamaya yer verilmiştir. Elde edilen sonuçlar, klasik modellerden elde edilen sonuçlar ile karşılaştırılmış ve her modelin öngörü performansı değerlendirilmiştir. Sonuç olarak yapay sinir ağları tahmin modellerinin, geleneksel modellere güçlü alternatif olabileceği görülmüştür.

Tuğba Yıldırım

ABSTRACT

COMPARISION OF ARTIFICIAL NEURAL NETWORKS AND TRADITIONAL METHODS FOR ELECTRICITY CONSUMPTION

Electricity is fundamental to the quality of our lives. We are totally dependent on abundant and uninterrupted supply of electricity for living and working. It is a key ingredient in all sectors of modern economies.

Turkey needs to solve increased energy demand problem with varied methods. Thus, Turkey pursues policies according to energy resources consumption and production forecasts.

In this study, it is aimed to examine both theoretical and applied analysis with artificial neural network and time series analysis models for long term electricity consumption forecasting. Within study, time series analysis and artificial neural Networks are explained, and an application is included. Also, used results of classical models and predictions of each model used, are compared. The comparison shows artificial neural networks forecasting methods could be a strong alternative to the traditional methods.

Tuğba Yıldırım

ABBREVIATIONS

Mtoe	: Milion tons of oil equivalent
MW	: Megawat
GW	: Gigawat
GWh	: Gigawat-hour
TW	: Tetrawat
TWh	: Tetrawat-hour
kW	: Kilowat
LNG	: Liquefied natural gas
GNP	: Gross National Product
SES	: Simple Exponential Smoothing Method
ME	: Mean Error
MAE	: Mean Absolute Error
MSE	: Mean Squared Error
MA	: Moving Average
AR	: Autoregression
ARMA	: Autoregressive Moving Average
ARIMA	: Autoregressive Integrated Moving Average
LMS	: Least Mean Square
ANN	: Artificial Neural Network
MLP	: Multilayer Perceptron
ADALINE	: Adaptive Linear Neuron
ALC	: Adaptive Linear Combiner
BPN	: Back Propagation Network
GDR	: Generalized Delta Rule
CPU	: Central Processing Unit
OOP	: Object-Oriented Program
CAM	: Content-Addressable Memory
TEIAS	: Turkish Electricity Transmission Corporation
TUIK	: Turkish Statistical Institute
RBF	: Radial Basis Function

MLP	: Multilayer Perceptron
NARX	: Nonlinear Autoregressive Network With Exogenous Inputs
GDP	: Gross Domestic Product
Log-sig	: Logistic-sigmoid
Tan-sig	: Tangent-sigmoid
R	: Correlation Coefficient
PCA	: Principal Component Analysis
PCS	: Principal Component Scores

FIGURES

	<u>PAGE NUMBER</u>
Figure II.1 Block diagram representation of nervous system.....	11
Figure II.2 Nonlinear model of a neuron.....	13
Figure II.3 Affine transformation produced by the presence of a bias; note that $v_k = b_k \text{ at } u_k = 0.$	14
Figure II.4 Another nonlinear model of neuron.....	15
Figure II.5 The basic components of an artificial neural network. The propagation rule used here is the ‘standard’ weighted summation.....	16
Figure II.6 Various activation functions for a unit.....	19
Figure II.7 A simple competitive learning network. Each of the four outputs 0 is connected to all inputs i	28
Figure II.8 Block diagram of learning with a teacher.	29
Figure II.9 Block diagram of reinforcement learning.....	31
Figure II.10 Block diagram of unsupervised learning.....	32
Figure II.11 Feedforward or acyclic network with a single layer of neurons.....	35
Figure II.12 A feedforward network with one hidden layer and one output.....	37
Figure II.13 A perceptron network with two hidden layers and four total layers....	40
Figure II.14 Signal-flow graph of the perceptron.....	41
Figure II.15 The Adaline.....	45
Figure II.16 A typical feedforward neural network.....	48
Figure II.17 Recurrent neural network architecture.....	49
Figure II.18 An Elman network architecture.....	50
Figure II.19 NARX network structure.....	54
Figure II.20 NARX network parallel and series-parallel mode architecture.....	55
Figure III.1 Explanatory or causal relationship.....	58
Figure III.2 Time-Series Relationship.....	59
Figure III.3 Time-Series relationship with random noise.....	63
Figure III.4 A strategy for appraising any of the smoothing methods of forecast...	64

Figure IV.1 MLP-14 model architecture.....	84
Figure IV.2 MLP-14 model training, validation, target correlation coefficient slope.....	85
Figure IV.3 MLP-14 model training output correlation coefficient slope.....	85
Figure IV.4 Comparison with real consumption and ANN consumption.....	86
Figure IV.5 Training, test and validation forecasts.....	86
Figure IV.6 MLP-21 model architecture.....	87
Figure IV.7 MLP-21 model training, validation, target correlation coefficient slope.....	87
Figure IV.8 MLP-21 model training output correlation coefficient slope.....	88
Figure IV.9 Comparison with real consumption and ANN consumption.....	88
Figure IV.10 Training, test and validation forecast.....	89
Figure IV.11 ELMAN-1 model architecture.....	90
Figure IV.12 ELMAN-1 model training, validation, target correlation coefficient slope.....	91
Figure IV.13 ELMAN-1 model training output correlation coefficient slope.....	91
Figure IV.14 Comparison with real consumption and ANN consumption.....	92
Figure IV.15 Training, test and validation forecasts.....	93
Figure IV.16 ELMAN-34 model architecture.....	93
Figure IV.17 ELMAN-34 model training, validation, target correlation coefficient slope.....	94
Figure IV.18 ELMAN-34 model training output correlation coefficient slope.....	94
Figure IV.19 Comparison with real consumption and ANN consumption.....	95
Figure IV.20 Training, test and validation forecasts.....	96
Figure IV.21 RBF model architecture.....	97
Figure IV.22 RBF model training, validation, target regression slope.....	97
Figure IV.23 RBF model training output regression slope.....	98
Figure IV.24 Comparison with real consumption and ANN consumption.....	99
Figure IV.25 Training, test and validation forecasts.....	99
Figure IV.26 NARX-2 model architecture.....	100
Figure IV.27 NARX-2 error slope.....	101
Figure IV.28 NARX-2 correlation coefficient slope	102
Figure IV.29 NARX-2 forecast between 1970-2005.....	102
Figure IV.30 NARX-2 correlation coefficient slope.....	103

Figure IV.31 NARX-2 forecast between 2008-2015.....	103
Figure IV.32 NARX-9 model architecture.....	104
Figure IV.33 NARX-9 error slope.....	104
Figure IV.34 NARX-9 correlation coefficient slope.....	105
Figure IV.35 NARX-9 forecast between 1970-2005.....	106
Figure IV.36 NARX-9 correlation coefficient slope.....	106
Figure IV.37 NARX-9 forecast between 2008-2015.....	107
Figure IV.38 Matrix plot of year, P,IP,GNP, EEI, EEE, D, EEP.....	108
Figure IV.39 Residual analysis.....	111
Figure IV.40 Autocorrelation function for RESI1.....	111
Figure IV.41 Time series plot of electricity consumption.....	112
Figure IV.42 Autocorrelation function for $\ln(C)$	113
Figure IV.43 Partial Autosorrelation function for $\ln(C)$	114
Figure IV.44 Residual analysis.....	116
Figure IV.45 Time Series plot for $\ln(C)$	116

TABLES

	<u>PAGE NUMBER</u>
Table IV.1 Principal Component Analysis.....	109
Table IV.2 Regression Analysis.....	110
Table IV.3 ARIMA model $\ln(C)$	115
Table V.1 Forecast and Values of Error Measures.....	118

CHAPTER I

INTRODUCTION

ENERGY RESOURCES

I.1. GENERAL BACKGROUND

Energy is fundamental to the quality of our lives. Nowadays, we are totally dependent on an abundant and uninterrupted supply of energy for living and working. It is a key ingredient in all sectors of modern economies.

We use energy daily to provide heat and electricity:

- in our homes for lighting, domestic appliances, televisions, computers, etc.;
- in factories to power the manufacture of the products we use everyday; and
- in cars, trucks, ships and aeroplanes to transport people and goods

Energy issues are directly related to the development of a country and the living standards of its people.

Energy is an indispensable factor for the social and economic development of societies. The usage level of electricity is an indication of the economic prosperity of nations.

I.2. ENERGY RESOURCES OF TURKEY

Turkey is located between Europe and Asia, bordering the Mediterranean, Aegean and Black Seas. Turkey is an important candidate to be an ‘energy bridge’ between Middle East, Caspian Region and Russia which have rich oil and natural gas sources.

Energy is one of Turkey’s most important development priorities. Rapid increase in domestic energy demand has forced Turkey to increase its dependence on foreign energy supplies) [9].

Turkey has very limited indigenous energy resources and has to import around 65% of primary energy to meet her needs. It is a large importer of primary energy despite having ample renewable energy sources [10].

Due to a young and growing population, low per capita electricity consumption, rapid urbanization and—until recently—strong economic growth,

increasing consumer oriented attitudes, Turkey's primary energy demand is rising rapidly. Turkey's electricity consumption had been growing much faster than its production.

Turkey's natural energy resources are quite diversified; hard coal, lignite, asphaltite, oil, natural gas, hydro, geothermal, wood, animal and plant wastes, solar and secondary energy resources such as coke and briquettes are produced and consumed. Although Turkey's oil and natural gas reserves are limited, coal reserves are quite abundant.

I.3. NON-RENEWABLE ENERGY AND ELECTRIC

I.3.1. Coal

Coal is the fastest growing fossil fuel and its large reserves would make it a popular candidate to meet the energy demand of the global community, short of global warming concerns and other pollutions.

Coal is a fossil fuel extracted from the ground by deep mining, coal mining (open-pit mining or strip mining). It is a readily combustible black or brownish-black sedimentary rock. It is composed primarily of carbon and hydrocarbons, along with assorted other elements, including sulfur. Often associated with the Industrial Revolution, coal remains an enormously important fuel and is the most common source of electricity world-wide.

Turkey has both hard coal and lignite deposits. Coal is a highly efficient and cheap resource of energy for Turkey, also plentiful coal resources exist in Europe, but the quality of the lignite is not high and is suitable for use in the power plants.

Almost 80% of lignite production is used in power plants. In contrast, less than 9% of hard coal supply is used for power generation. Turkish government expects total coal supply to rise from 20.1 Mtoe in 1999 to 118.4 Mtoe in 2020; more than five times higher than current figures. Behind this increase lies the expectation of tremendous growth in hard coal and lignite fired power generation, and in industrial use of coal [11].

The hard coal is found in the western part of the country, especially in the Zonguldak Basin, which has more than 700 million metric tons of workable reserves, about 80% of which can be coked. Lignite deposits are widespread and plentiful; reserves are estimated at more than 8 billion metric tons [9].

Turkish Hard Coal Cooperation (TTK) has a de facto monopoly in hard coal production, processing and distribution.

Coal and Electricity

Fossil fuels are excellent sources of energy for our transportation needs; however they are also the primary source of electrical energy in the world today. Coal power plants account for at least 60% of our national energy and 52% of the world's demand. We, as a world, burn approximately 1.9 billion tons of coal a year to generate electricity [29].

I.3.2. Oil

As Turkey's economy has expanded in recent years, the consumption of oil has increased.

Oil provides nearly half of Turkey's total energy requirement but gradually it is leaving its place to natural gas. Around 90% of Turkey's oil supplies are imported mainly from the Middle East and Russia. The total oil reserve of Turkey is 931 million tons [12]. The share of oil in electricity generation was about 30% in 1970 and decreased to about 8% in 2001 [11].

I.3.2. Natural gas

Natural gas is the fastest growing primary energy source in the world.

The 0.57 billion cubic meter of natural gas that is produced in Turkey in 2000, which meets only 3.8% of domestic consumption [9].

Presently, the largest share of Turkey's imported natural gas comes from Russia.

I.3.3. Nuclear Power

Nuclear power is generated using Uranium, which is a metal mined in various parts of the world. Nuclear power produces around 11% of the world's energy needs, and produces huge amounts of energy from small amounts of fuel, without the pollution that you'd get from burning fossil fuels [30].

Nuclear power stations work in pretty much the same way as fossil fuel-burning stations, except that a "chain reaction" inside a nuclear reactor makes the heat instead. The reactor uses Uranium rods as fuel, and the heat is generated by

nuclear fission. Neutrons smash into the nucleus of the uranium atoms, which split roughly in half and release energy in the form of heat. Carbon dioxide gas is pumped through the reactor to take the heat away, and the hot gas then heats water to make steam. The steam drives turbines which drive generators. Modern nuclear power stations use the same type of turbines and generators as conventional power stations.

The Turkish government has just declared that nuclear power plant with a total capacity of 1300MW would be built at Akkuyu in 2012 [31]. In fact, most European countries lean towards becoming increasingly nonnuclear in the future, basic reason being the large quantities of highly radioactive waste these plants generate. A major part of Turkey's land is in the seismic zone. However, Turkish government is still considering to establish the nuclear power plant to overcome the possible high electricity demand in the future [11].

I.4. RENEWABLE ENERGY AND ELECTRIC

Achieving sustainable development is a target that is now widely seen as important to worldwide public opinion.

Renewable, except hydraulic, have only very minor shares in power generation in Turkey: geothermal accounts for 0.1% or 89.6 GWh, wind for about 0.09 or 62.4GWh and combustible renewable for about 0.2% or 49.5 GWh. Economically-feasible renewable energy potential in Turkey is estimated at a total of capacity 495.4 TWh/yr with the potential for 196.7 TWh/yr of biomass energy, 124 TWh/yr of hydropower, 102.3 TWh/yr of solar energy, 50 TWh/yr of wind energy, and 22.4 TWh/yr of geothermal energy [11].

I.4.1. Hydropower

Hydroelectricity is the only renewable today making a large contribution to world energy production. Hydroelectric power now supplies about 715,000 MWe or 19% of world electricity [32].

Turkey's second largest energy source after coal is hydro. Turkey has about 1% of the total world hydroelectric potential. There are many rivers in Turkey and five separate watersheds. As of the end of 2001, Turkey had 125 hydroelectric power plants in operation [9].

The three largest dams in Turkey are the 2400MW Ataturk dam, the sixth-largest rock fall dam in the world, on the main trunk of the Euphrates; the 1800MW

Karakaya dam; and the 1200MW Ilisu dam, the largest hydro project on the Tigris River [11].

Most of them are owned and operated by DSI; independent companies who own hydroelectric projects in Turkey include Birecik AS, which owns a 672MW power plant on the Euphrates River, and CEAS which presently has more than 1000MW generating capacity. Hydroelectric power plants in Turkey currently account for about 40% of Turkey's electricity demand [13].

In 1999, there were 114 hydroelectric power plants in operation in Turkey. In 2001, Total hydroelectric power plant capacity in Turkey was 11,672.9MW, with estimated mean annual generation of 24 TWh. In 2005, the 1440MW Afsin- Elbistan B Thermal Power Plant will open in southern Anatolia, southwest Turkey and contribute an estimated 7.5% of the country's predicted electricity generation needs [33].

Hydroelectric power plants in Turkey currently account for about 40% of Turkey's electricity demand [9].

I.4.2. Wind Power

Wind power is one of the most cost competitive renewables today.

Wind energy is the fastest growing energy source in the world and wind power is one of the most widely used alternative sources of energy today. It is a clean and renewable source of electricity [11].

Wind power, the technology of using the wind to generate electricity, is the fastest-growing new source of electricity worldwide.

Turkey has a considerable potential for electricity generation from wind. Electricity generation through wind energy for general use was first realized at Cesme Altinyunus Resort Hotel (The Golden Dolphin Hotel) in Izmir, in Turkey in 1986 with a 55kW nominal wind power capacity [14]. Between 1986 and 1998, there were some attempts to generate electricity from wind, but they were never successful. The first wind power facility with a total installed capacity of 1.5MW, excepting to the small wind system with a capacity of 55 kW, commissioned in February 1998 is located near the city of Izmir at Cesme Germiyan in western Turkey. In November 1998, ARES (Alacati Wind Energy Plant) which has 12 wind turbines for a total capacity of 7.2MW was commissioned [15].

The biggest available wind energy power plant in Turkey is the BORES (Bozcaada Wind Energy Plant) with 10.2MW power, is to be sold after repaying its cost. It was constructed in January 2000 and uses 17 wind turbines. Each turbine has 600kW power. This facility generates 35 million kWh electric energy per year [16]. In 2001, the electricity generation from the wind was reached to 62.4GWh. In addition to above operating power plants, many individuals apply to the Ministry of Energy and Natural Sources to generate wind energy [11].

I.4.3. Solar Energy

Solar energy used during 2005 was approximately 93.4 GW; however, the available resources are 120,000 TW [25].

Only a small fraction of available resources are sufficient to entirely replace fossil fuels and nuclear power as an energy source [25].

The consumption of solar hot water and solar space heating was estimated at 88 GWt (gigawatts of thermal power) in 2004 [34].

Turkey lies in a sun belt between 36 and 42°N latitudes. The yearly average solar radiation is 1303kWh/m² year and the total yearly radiation period is approximately 2623 h [11].

Total installed power is nearly 300 kW [11].

I.4.4. Geothermal

Turkey is the seventh richest country in the world in geothermal potential for its direct use and for electricity generation [11].

Turkey has significant potential for geothermal power production, possessing one-eighth of the world's total geothermal potential. At the end of 1999, Turkey's total installed capacity for direct heating was 820MWh, of which about 390MWh provided heating for 51,600 residences, about 100MWh provided heating for about 45 hectares of greenhouses, and about 330MWh was used to provide heated water for about 200 spas [13].

The only operating geothermal power plant of Turkey is the Denizli-Kizildere Geothermal Power Plant (DKGPP), located near the province of Denizli in Western Anatolia with an installed capacity of 20.4MW [17]. The DKGPP was put into operation in 1984 and has been operated since then. It produced an electrical energy

of 89,597MWh in 2001, representing an electric power of 10.6MW in the same year [11].

I.4.5. Biogas

Biogas production potential has been estimated at 1.5–2 Mtoe but neither the government nor the private sector has yet expressed any interest in using this resource. There are two BOT power plant projects using biogas (Ankara and Adana waste-to-power plants), but not much progress has been made and the government fears the projects will lapse. There is one autoproducer waste-to-energy power plant in operation in Izmit. This plant was commissioned in 1998 and has an installed capacity of 5.4MW. A contract has been signed for another autoproducer plant of 5.4MW in Istanbul, to be constructed by the municipality of Istanbul [11].

I.5. HISTORICAL DEVELOPMENT OF ELECTRICITY IN TURKEY

The electricity industry in Turkey dates back to 1902, when a 2kW hydropower system was connected to a water mill in Tarsus. The first larger-scale power plant was built in Istanbul in 1913. The whole installed capacity was 29,664kW when Turkish Republic was established in 1923 and the production was 45 GWh/year in those years and the electricity was only available in three cities, namely Istanbul, Adapazari and Tarsus. The first Turkish electricity company, Kayseri ve Civarı Elektrik Turk Elektrik Inc., was established in 1926. At the end of the 1920s, there were 48 power plants (74.8MW) producing 106.3GWh electricity and per capita electrical consumption was 6.2kWh [11].

At the beginning of the 1950s, installed capacity was about 408MW generating 789.5GWh electricity, only the 23% of the population used electricity, while per capita consumption increased to 32kWh [11].

The Ministry of Energy and Natural Resources of Turkey (MENR) was established in December 1963, and was responsible for Turkey's energy policy. This was followed by passing the law creating the Turkish Electricity Administration Commission (TEK), which would create a monopoly in the Turkish electricity sector at all stages. All generation assets were passed to TEK except the ones that belong to Cukurova Elektrik T.A.S. and Kepez ve Antalya Havalisi Elektrik Santralleri T.A.S. The transmission and distribution business, which was managed by the municipals, were left to the local governments [18].

By 1970, installed capacity had increased to about 2234.9MW, and both growing power consumption and the government's electrification plans required more coherent organization of the power industry. At that time, only 7% of all villages were electrified. As a consequence, the government established the Turkish Electricity Authority (Turkiye Elektrik Kurumu, TEK) as a fully state-owned and state run entity that year. Installed hydroelectric power plants provided approximately 32% of the electrical power during 1970 [11].

Up to 1971, the power plants were operated by some governmental organizations and institutions established with a special law. According to a law issued in 1971, Turkish Electricity Authority (TEK) was responsible as a whole for the generation and distribution of electricity throughout the country [19]. In 1982, electrification had reached about 60% of all villages, and installed capacity had grown to 6639MW. All plants and networks owned by municipalities and unions were transferred to TEK that year [11].

Variation of electricity generation in Turkey shows a sudden and unpredictable change.

In the years following the first oil crisis, economic conditions deteriorated, with high unemployment, a nearly fivefold increase in the balance of payments deficit between 1973 and 1979, large external debt and annual inflation rates exceeding 100% in 1980. It forced Turkey to use the national and renewable energy source, hydraulic. After 1982, the effect of oil crisis was recovered and again Turkey started to use thermal power [11].

Although the wind of the privatization studies has blown since 1984, the private sector was not so much active on this matter until 1985 because the details of the privatization were not examined in the first law issued [19]. In 1987, construction of Karakaya dam (1800MW) and in 1992 construction of Ataturk dam (2400MW) were finished and they started producing electricity. Therefore, the share of hydroelectric generation increased. After 1987, Turkey started importing natural gas and most of it used for electricity generation. During 1986–1993, the installation of hydroelectric system increased steadily. After 1993, there is not recognizable change for the hydroelectric installation [11].

I.5.1. Development of electricity consumption in turkey

Turkey's electricity demand grew quickly in the past and is still growing rapidly. Turkey had about 198418 GWh of installed generating capacity in 2008.

Turkey's Electricity Consumption increased from 7307 GWh in 1970 to 198085 GWh in 2008.

Installed electricity capacity in Turkey reached about 41417 MW in 2008

I.6. ELECTRICAL ENERGY CONSUMPTION AND PRODUCTION OF TURKEY VERSUS WORLD

Energy inputs for Turkey include both primary resources and minor contributions from other sources. The power plants, refineries, and coke production facilities of integrated steel factories can be analyzed together in the conversion sector. Although activities in this sector are not end uses, they involve significant losses. The diverse end users are dealt with by first aggregating them into sub-sectors and then into transportation, industrial, and residential and commercial sectors [13].

CHAPTER II

ARTIFICIAL NEURAL NETWORK

Work on artificial neural Networks, commonly referred to as ‘neural networks’, has been motivated right from its inception by the recognition that the human brain computes in an entirely different way from the conventional digital computer. The brain is a highly complex, nonlinear, and parallel computer (information processing system). It has the capability to organize its structural constituents, known as neurons, so as to perform certain computations (e.g., pattern recognition, perception, and motor control) many times faster than the fastest digital computer in existence today.

A neural network is a machine that is designed to model the way in which the brain performs a particular task or function of interest; the network is usually implemented by using electronic components or is simulated in software on a digital computer [1].

A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

Knowledge is acquired by the network from its environment through a learning process.

Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

The procedure used to perform the learning process is called a learning algorithm, the function of which is to modify the synaptic weights of the network in an orderly fashion to attain a desired design objective.

A first wave of interest in neural Networks emerged after the introduction of simplified neurons by McCulloch and Pitts in 1943. These neurons were presented as models of biological neurons and as conceptual components for circuits that could perform computational tasks [2].

When Minsky and Papert published their book *Perceptrons* in 1969 in which they showed the deficiencies of perceptron models, most neural network funding was redirected and researchers left the field. Only a few researchers continued their

efforts, most notably Teuvo Kohonen, Stephen Grossberg, James Anderson, and Kunihiko Fukushima [2].

The interest in neural Networks re-emerged only after some important theoretical results were attained in the early eighties (most notably the discovery of error back-propagation), and new hardware developments increased the processing capacities. This renewed interest is reflected in the number of scientists, the amounts of funding, the number of large conferences, and the number of journals associated with neural networks.

Artificial neural networks can be most adequately characterised as ‘computational models’ with particular properties such as the ability to adapt or learn, to generalise, or to cluster or organise data, and which operation is based on parallel processing. However, many of the abovementioned properties can be attributed to existing (non-neural) models; the intriguing question is to which extent the neural approach proves to be better suited for certain applications than existing models. To date an equivocal answer to this question is not found.

II.1. HUMAN BRAIN

The human nervous system may be viewed as a three-stage system, as depicted block diagram of Figure II.1. Central to the system is the brain, represented by the neural (nevre) net, which continually receives information, perceives it, and makes appropriate decisions. Two sets of arrows are shown in the figure. Those pointing from left to right indicate the forward transmission of information-bearing signals through the system. The arrows pointing from right to left signify the presence of feed-back in the system. The receptors convert stimuli from the human body or the external environment into electrical impulses that convey information to the neural net (brain). The effectors convert electrical impulses generated by the neural net into discernible responses as system outputs [1].

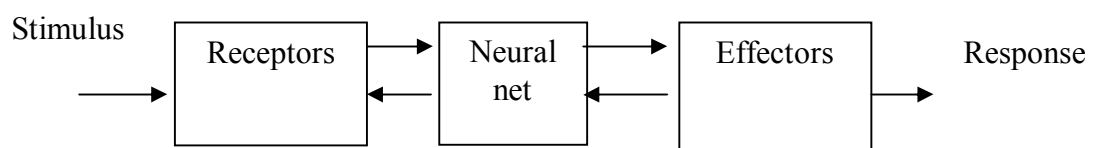


Figure II.1 Block diagram representation of nervous system [1].

The artificial neurons we use to build our neural Networks truly primitive in comparison to those found in the brain. The neural networks we are presently able to design are just as primitive compared to the local circuits and the interregional circuits in the brain. With neurobiological analogy as the source of inspiration, and the wealth of theoretical and technological tools that we are bringing together, it is certain that in another decade our understanding of artificial neural networks will be much more sophisticated than it is today.

II.2. MODELS OF NEURON

A neuron is an information-processing unit that is fundamental to the operation of a neural network. The block diagram of Fig.II.2 shows the model of a neuron, which forms the basis for designing (artificial) neural networks. Here we identify three basic elements of the neuronal model:

1. A set of synapses or connecting links, each of which is characterized by a weight or strength of its own. Specifically, a signal x_j at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj} . It is important to make a note of the manner in which the subscripts of the synaptic weight w_{kj} are written. The first subscript refers to the neuron in question and the second subscript refers to the input end of the synapse to which the weight refers. Unlike a synapse in the brain, the synaptic weight of an artificial neuron may lie in a range that includes negative as well as positive values.

2. An adder for summing the input signals, weighted by the respective synapses of the neuron; the operations described here constitute a linear combiner.

3. An activation function for limiting the amplitude of the output of a neuron. The activation function is also referred to as squashing function in that it squashes (limits) the permissible amplitude range of the output signal to some finite value [1].

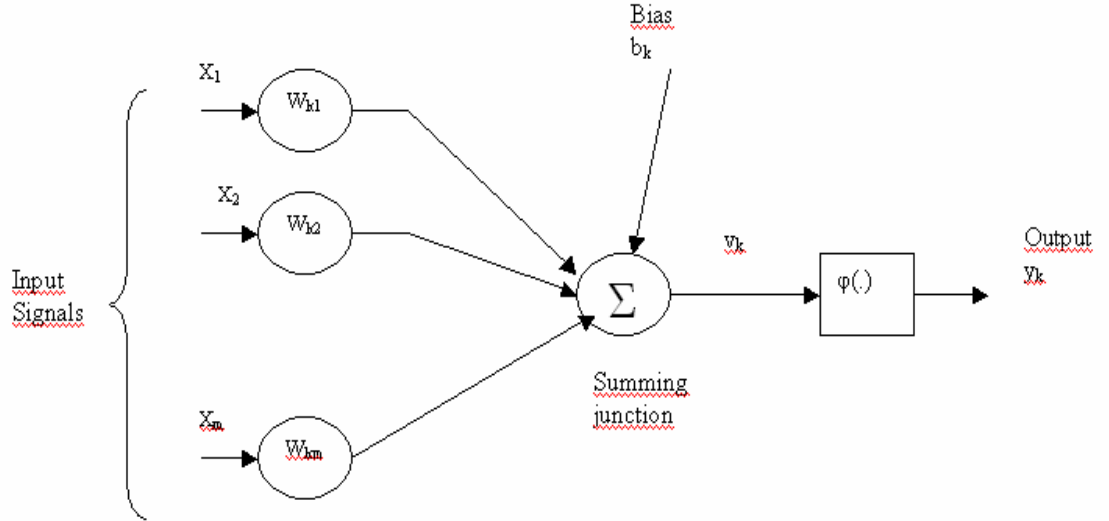


Figure II.2 Nonlinear model of a neuron [1].

Typically, the normalized amplitude range of the output of a neuron is written as the closed unit interval $[0,1]$ or alternatively $[-1,1]$.

The neuronal model of Figure II.2 also includes an externally applied bias, denoted by b_k . The bias b_k has the effect of increasing or lowering the net input of the activation function, depending on whether it is positive or negative, respectively.

The mathematical terms, we may describe a neuron k by writing the following pair of equations:

$$u_k = \sum_{j=1}^m w_{kj} x_j \quad (\text{II.1})$$

and

$$y_k = \varphi(u_k b_k) \quad (\text{II.2})$$

where $x_1, x_2, \dots, x_m$ are the input signals; $w_{k1}, w_{k2}, \dots, w_{km}$ are the synaptic weights of neuron k ; u_k is the linear combiner output due to the input signals; b_k is the bias; $\varphi(.)$ is the activation function; and y_k is the output signal of the neuron. The use of bias b_k has the effect of applying an affine transformation to the output u_k of the linear combiner in the model of Fig. II.2, as shown by

$$v_k = u_k + b_k \quad (\text{II.3})$$

In particular, depending on whether the bias b_k is positive and negative, the relationship between the induced local field or activation potential v_k of neuron k and the linear combiner output u_k is modified in the manner illustrated in Fig. II.3; hereafter the term ‘induced local field’ is used. Note that as a result of this affine transformation, the graph of v_k versus u_k no longer passes through the origin [1].

The bias b_k is an external parameter of artificial neuron k. We may account for its presence as in Eq. (II.2), equivalently, we may formulate the combination of Eqs.(II.1) to (II.3) as follows:

$$v_k = \sum_{j=0}^m w_{kj} x_j \quad (\text{II.4})$$

and

$$y_k = \varphi v_k \quad (\text{II.5})$$

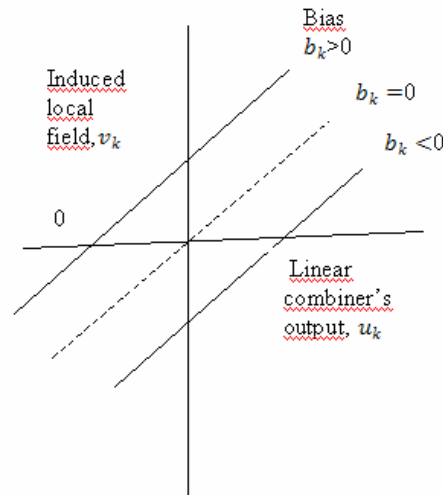


Figure II.3 Affine transformation produced by the presence of a bias; note that $v_k = b_k$ at $u_k = 0$ [1].

In Eq.(II.4) we have added a new synapse. Its input is

$$x_0 = +1 \quad (\text{II.6})$$

And its weight is

$$w_{k0} = b_k \quad (\text{II.7})$$

We may therefore reformulate the model of neuron k as Fig.II.3 In this figure, the effect of the bias is accounted for by doing two things: (1) adding a new input signal fixed at +1, and (2) adding a new synaptic weight equal to the bias b_k . Although the models of Figs. II.2 and II.3 are different in appearance, they are mathematically equivalent [1].

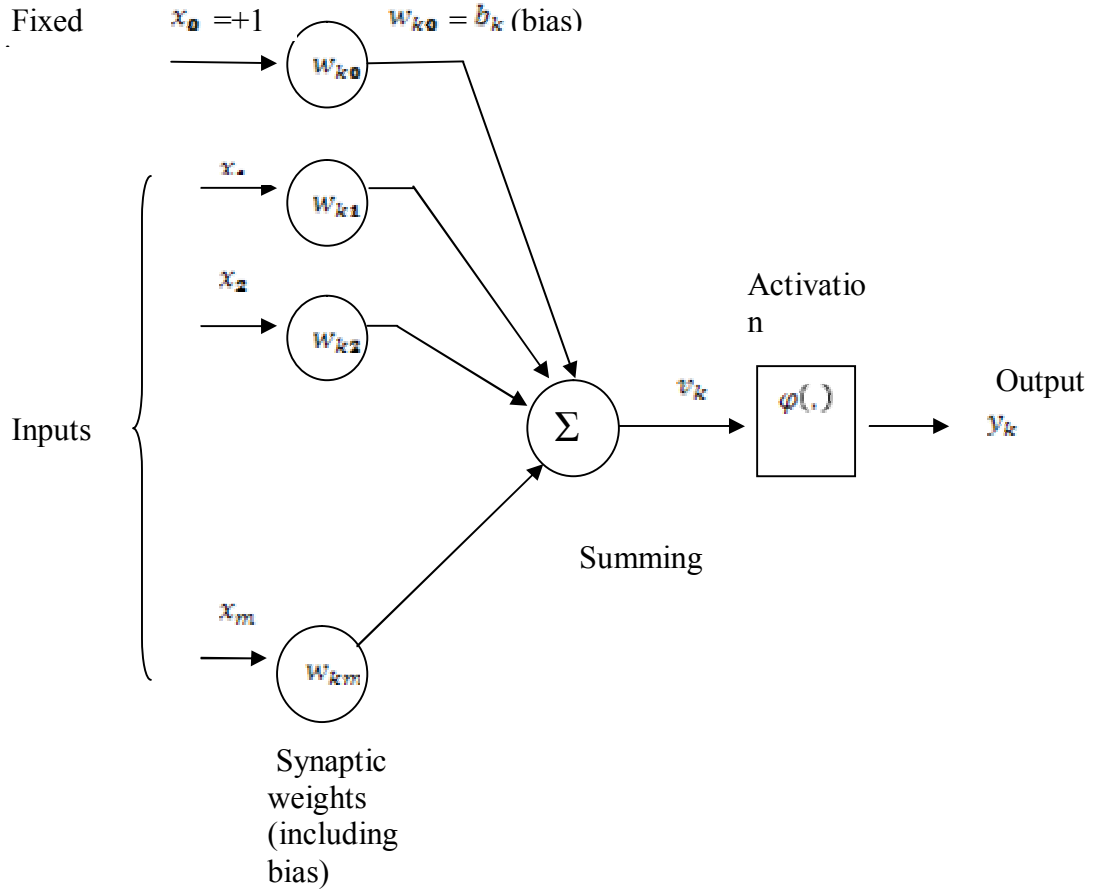


Figure II.4 Another nonlinear model of neuron [1].

II.3. A FRAMEWORK FOR DISTRIBUTED REPRESENTATION

An artificial network consists of a pool of simple processing units which communicate by sending signals to each other over a large number of weighted connections [2].

A set of major aspects of a parallel distributed model can be distinguished (cf. Rumelhart and McClelland, 1986

- a set of processing units ('neurons', 'cells');
- a state of activation y_k for every unit, which equivalent to the output of the unit;
- connections between the units. Generally each connection is defined by a weight w_{jk} which determines the effect which the signal of unit j has on unit k ;
- a propagation rule, which determines the effective input s_k of a unit from its external inputs;
- an activation function F_k , which determines the new level of activation based on the effective input $s_k(t)$ and the current activation $y_k(t)$ (i.e., the update);
- an external input (aka bias, offset) θ_k for each unit;
- a method for information gathering (the learning rule);
- an environment within which the system must operate, providing input signals and – if necessary – error signals [2].

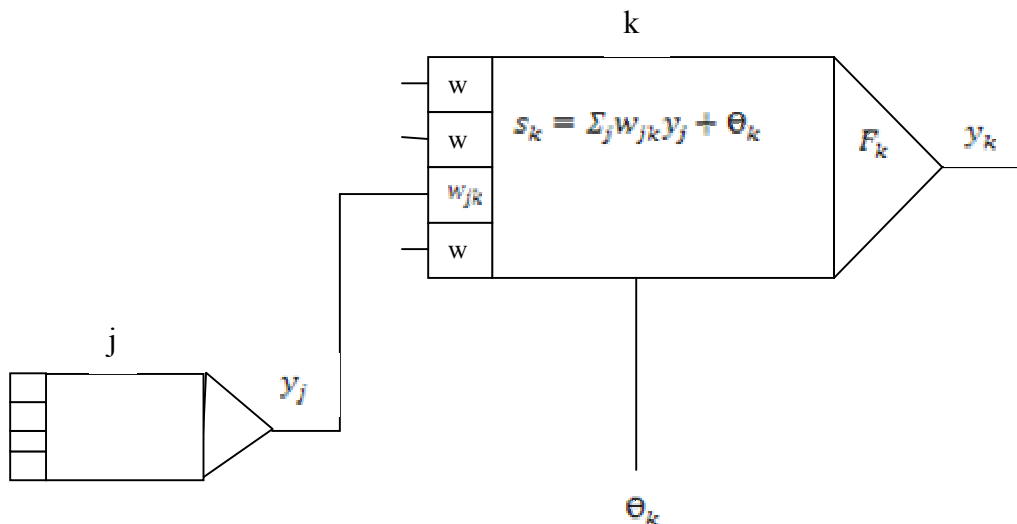


Figure II.5 The basic components of an artificial neural network. The propagation rule used here is the 'standard' weighted summation [2].

II.3.1. Processing Units

Each unit performs a relatively simple job: receive input from neighbours or external sources and use this to compute an output signal which is propagated to other units. Apart from this processing, a second task is the adjustment of the weights. The system is inherently parallel in the sense that many units can carry out their computations at the same time.

Within neural systems it is useful to distinguish three types of units: input units (indicated by an index i) which receive data from outside the neural network, output units (indicated by an index o) which send data out of the neural network and hidden units (indicated by an index h) which input and output signals remain within the neural network.

During operation units can be updated either synchronously or asynchronously. With synchronous updating, all units update their activation simultaneously; with asynchronous updating, each unit has a (usually fixed) probability of updating its activation at a time t , and usually only one unit will be able to do this at a time. In some cases the latter model has some advantages.

II.3.2. Connections Between Units

In most cases we assume that each unit provides an additive contribution to the input of the unit with which it is connected. The total input to unit k is simply the weighted sum of the separate outputs from each of the connected units plus a bias or offset term θ_k :

$$S_k(t) = \sum_j w_{jk}(t) + \theta_k(t) \quad (\text{II.8})$$

The contribution for positive w_{jk} is considered as an excitation and for negative w_{jk} as inhibition. In some cases more complex rules for combining inputs are used, in which a distinction is made between excitatory and inhibitory inputs. We call units with a propagation rule sigma units.

A different propagation rule, introduced by Feldman and Ballard, is known as the propagation rule for the sigma-pi unit:

$$s_k(t) = \sum_j w_{jk}(t) \prod_m y_j(t) + \theta_k(t) \quad (\text{II.9})$$

Often, the y_{jm} are weighted before multiplication. Although these units are not frequently used, they have their value for gating of input [2].

II.3.3. Activation and Output Rules

We also need a rule which gives the effect of the total input on the activation of the unit. We need a function F_k which takes the total input $s_k(t)$ and the current activation $y_k(t)$ and produces a new value of the activation of the unit k [2]:

$$y_k(t+1) = F_k(y_k(t), s_k(t)) \quad (\text{II.10})$$

Often, the activation function is a nondecreasing function of the total input of the unit:

$$y_k(t+1) = F_k(s_k(t)) = F_k\left(\sum_j w_{jk}(t)y_j(t) + \theta_k(t)\right) \quad (\text{II.11})$$

Although activation functions are not restricted to nondecreasing functions. Generally, some sort of threshold function is used: a hard limiting threshold function (a sgn function), or a linear or semi-linear function, or a smoothly limiting threshold. For this smoothly limiting function often a sigmoid (S-shaped) function like

$$y_k = \frac{1}{(1 + e^{-s_k})} \quad (\text{II.12})$$

is used. In some applications a hyperbolic tangent is used, yielding output values in the range $[-1, +1]$ [2].

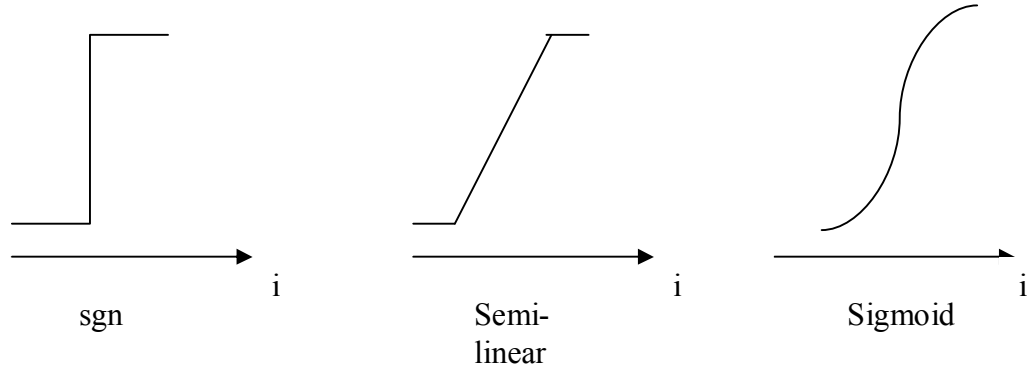


Figure II.6 Various activation functions for a unit

In some cases, the output of a unit can be a stochastic function of the total input of the unit. In that case the activation is not deterministically determined by the neuron input, but the neuron input determines the probability p that a neuron get a high activation value:

$$P(y_{k1}) = \frac{1}{1 + e^{-\frac{ik}{T}}} \quad (\text{II.13})$$

in which T (cf. Temperature) is a parameter which determines the slope of the probability function.

II.4. TYPES OF ACTIVATION FUNCTION

The activation function, denoted by $\phi(v)$, defines the output of a neuron in terms of the induced local field v . Here we identify three basic types of activation functions:

1. Threshold Function.
2. Piecewise-Linear Function.
3. Sigmoid Function

Networks With Threshold Activation Functions

A single layer feed-forward network consists of one or more output neurons o , each of which is connected with a weighting factor w_{io} to all of the inputs i . In the simplest case the network has only two inputs and a single output. The input of the

neuron is weighted sum of the inputs plus the bias term. The output of the network is formed by the activation of the output neuron, which is some function of the input:

$$y = F\left(\sum_{i=1}^2 w_i x_i + \theta\right) \quad (\text{II.14})$$

The activation function F can be linear so that we have a linear network, or nonlinear. In this section we consider the threshold function:

$$F(s) = \begin{cases} 1 & \text{if } s > 0 \\ -1 & \text{otherwise} \end{cases} \quad (\text{II.15})$$

The output of the network thus is either +1 or -1, depending on the input. The network can now be used for a classification task: it can decide whether an input pattern belongs to one of two classes. If the total input is positive, the pattern will be assigned to class +1, if the total input is negative, the sample will be assigned to class -1. The separation between the two classes in this case is a straight line, given by the equation [2]:

$$w_1 x_1 + w_2 x_2 + \theta = 0 \quad (\text{II.16})$$

The single layer network represents a linear discriminant function.

A geometrical representation of the linear threshold neural network can be written as

$$x_2 = -\left(\frac{w_1}{w_2}\right)x_1 - \left(\frac{\theta}{w_2}\right) \quad (\text{II.17})$$

and we see that the weights determine the slope of the line and the bias determines the ‘offset’ i.e. how far the line is from the origin.

We will describe two learning methods for these types of networks: the ‘perceptron’ learning rule and the ‘delta’ or ‘LMS’ rule. Both methods are iterative procedures that adjust the weights. A learning sample is presented to the network. For each weight the new value is computed by adding a correction to the old value. The threshold is updated in a same way [2]:

$$w_i(t+1) = w_i(t) + \Delta w_i(t) \quad (\text{II.18})$$

$$\theta(t+1) = \theta(t) + \Delta \theta(t) \quad (\text{II.19})$$

II.5. NETWORKS WITH LINEAR ACTIVATION FUNCTIONS: THE DELTA RULE

For a single layer network with an output unit with a linear activation function the output is simply given by

$$y = \sum_j w_j x_j + \theta \quad (\text{II.20})$$

Such a simple network is able to represent a linear relationship between the value of the output unit and the value of the input units. By thresholding the output value, a classifier can be constructed (such as Widrow’s Adaline), but here we focus on the linear relationship and use the network for a function approximation task. In high dimensional input spaces the network represents a (hyper) plane and it will be clear that also multiple output units may be defined [2].

Suppose we want to train the network such that a hyperplane is fitted as well as possible d^p . For every given input sample, the output of the network differs from target value d^p by $(d^p - y^p)$, where y^p is the actual output for this pattern. The delta-rule now uses a cost – or error function based on these differences to adjust the weights [2].

The error function, as indicated by the name least mean square, is the summed squared error. That is, the total error E is defined to be

$$E = \sum_P E^P = \frac{1}{2} \sum_P (d^P - y^P)^2 \quad (\text{II.21})$$

where the index p ranges over the set of input patterns and E^P represents the error on pattern p . The LMS procedure finds the values of all the weights that minimise the error function by a method called gradient descent. The idea is to make a change in the weight proportional to the negative of the derivative of the error as measured on the current pattern with respect to each weight [2]:

$$\Delta P_{wj} = -\gamma \left(\frac{\partial E^P}{\partial w^j} \right) \quad (\text{II.22})$$

where γ is a constant of proportionality.

The delta rule modifies weight appropriately for target and actual outputs of either polarity and for both continuous and binary input and output units. These characteristics have opened up a wealth of new applications.

II.6. THE GENERALIZED DELTA RULE

We shall present a detailed derivation of the generalized delta rule (GDR), which is the learning algorithm for the network.

The backpropagation network is a layered, feedforward network that is fully interconnected by layers. Thus, there are no feedback connections and no connections that bypass one layer to go directly to a later layer. Although only three layers are used in discussion, more than one hidden layer is permissible.

A neural network is called a mapping network if it is able to compute some functional relationship between its input and its output [3].

II.7. NETWORK TOPOLOGIES

- Feed-forward networks, where the data flow from input to output units is strictly feed-forward. The data processing can extend over multiple (layers of) units, but no feedback connections are present, that is, connections extending from outputs of units to inputs of units in the same layer or previous layers.
- Recurrent networks that do contain feedback connections. Contrary to feed-forward networks, the dynamical properties of the network are important. In some cases, the activation values of the units undergo a relaxation process such that the

network will evolve to a stable state in which these activations do not change anymore. In other applications, the change of the activation values of the output neurons are significant, such that the dynamical behaviour constitutes the output of the network.

II.8. TRAINING OF ARTIFICIAL NEURAL NETWORKS

A neural network has to be configured such that the application of a set of units produces (either 'direct' or via a relaxation process) the desired set of outputs. Various methods to set the strengths of the connection exist. One way is to set the weights explicitly, using a priori knowledge. Another way is to 'train' the neural network by feeding it teaching patterns and letting it change its weights according to some learning rule.

II.8.1. Paradigms Of Learning

We can categorise the learning the situations in two distinct sorts. These are:

- *Supervised learning or Associative learning* in which the network is trained by providing it with input and matching output patterns. These input-output pairs can be provided by an external teacher, or by the system which contains the network (self-supervised).
- *Unsupervised learning or Self-organisation* in which an (output) unit is trained to respond to clusters of pattern within the input. In this paradigm the system is supposed to discover statistically salient features of the input population. Unlike the supervised learning paradigm, there is no a priori set of categories into which the patterns are to be classified; rather the system must develop its own representation of the input stimuli.

II.8.2. Modifying Patterns Of Connectivity

Both learning paradigms discussed above result in an adjustment of the weights of the connections between units, according to some modification rule. Virtually all learning rules for models of this type can be considered as a variant of the Hebbian learning rule suggested by Hebb in his classic book *Organization of Behaviour* (1949) [2]. The basic idea is that if two units j and k are active simultaneously, their interconnection must be strengthened. If j receives input from k , the simplest version of Hebbian learning prescribes to modify the weight w_{jk} with

$$\Delta w_{jk} = \gamma y_j y_k \quad (\text{II.23})$$

where γ is a positive constant of proportionality representing the learning rate. Another common rule uses not the actual activation of unit k but the difference between the actual and desired activation for adjusting the weights:

$$\Delta w_{jk} = \gamma y_j (d_k - y_k) \quad (\text{II.24})$$

in which d_k is the desired activation provided by a teacher. This is often called the Widrow-Hoff rule or the delta rule [2].

II.9. NETWORK SIZING

The size of the input layer is usually dictated by the nature of the application. You can often determine the number of output nodes by deciding whether you want analog values or binary values on the output units.

Determining the number of units to use in the hidden layer is not usually as straightforward as it is for the input and output layers. The main idea is to use as few hidden-layer units as possible, because each unit adds to the load on the Central Processing Unit (CPU) during simulation. Of course in a system that is fully implemented in hardware (one processor per processing element), additional CPU loading is not as much of a consideration. We hesitate to offer specific guidelines except to say that, in our experience, for Networks of reasonable size (hundreds and thousands of inputs), the size of the hidden layer needs to be only a relatively small fraction of that of the input layer. If the network fails to converge to a solution it may be that more hidden nodes are required. If it does converge, you might try fewer hidden nodes and settle on a size on the basis of overall system performance [3].

It is also possible to remove hidden units that are superfluous. If you examine the weight values on the hidden nodes periodically as the network trains, you will see that weights on certain nodes change very little from their starting values. These nodes may not be participating in the learning process, and fewer hidden units may

suffice. There is also an automatic method, developed by Rumelhart, for pruning unneeded nodes from the network [3].

II.10. WEIGHTS AND LEARNING PARAMETERS

Weights should be initialized to small, random values –say between ± 0.5 - as should the bias terms, that appear in the equations for the net input to a unit. It is common practice to treat this bias value as another weight, which is connected to a fictitious unit that always has an output of 1 [3].

II.11. LEARNING PROCESS

The property that is of primary significance for a neural network is the ability of the network to learn from its environment, and to improve its performance through learning.

We define learning in the context of neural Networks as:

Learning is a process by which the free parameters of a neural network are adapted through a process of simulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place [1].

This definition of the learning process implies the following sequence of events:

1. The neural network is stimulated by an environment.
2. The neural network undergoes changes in its free parameters as a result of this stimulation.
3. The neural network responds in a new way to the environment because of the changes that have occurred in its internal structure [1].

A prescribed set of well-defined rules for the solution of a learning problem is called a learning algorithm. As one would expect, there is no unique learning algorithm for the design of neural Networks.

II.11.1. Hebbian Learning

When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased.

Hebb proposed this change as a basis of associative learning which would result in an enduring modification in the activity pattern of a spatially distributed ‘assembly of nerve cells’.

This statement is made in a neurobiological context. We may expand and rephrase it as a two part rule:

1. If two neurons on either side of a synapse (connection) are activated simultaneously, then the strength of that synapse is selectively increased.
2. If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated [1].

Such synapse is called a Hebbian synapse. More precisely, we define a Hebbian synapse that uses time dependent, highly local, and strongly interactive mechanism to increase synaptic efficiency as a function of the correlation between the presynaptic and postsynaptic activities. From this definition we may deduce the following four key mechanisms that characterize a Hebbian synapse:

1. Time-dependent mechanism. This mechanism refers to the fact that the modifications in a Hebbian synapse depend on the exact time of occurrence of the presynaptic and postsynaptic signals.
2. Local mechanism. By its very nature, a synapse is the transmission site where information-bearing signals are in spatiotemporal contiguity. This locally available information is used by a Hebbian synapse to produce a local synaptic modification that is input specific.
3. Interactive mechanism. The occurrence of a change in a Hebbian synapse depends on signals on both sides of the synapse. That is, a Hebbian form of learning depends on a ‘true interaction’ between presynaptic and postsynaptic signals in the sense that we cannot make a prediction from either one of these two activities by itself. Note also that this dependence or interaction may be deterministic or statistical in nature.
4. Conjunctional or correlational mechanism. One interpretation of Hebb’s postulate of learning is that the condition for a change in synaptic efficiency is the conjunction of presynaptic and postsynaptic signals. Thus, according to this interpretation, the co-occurrence of presynaptic and postsynaptic signals is sufficient to produce the synaptic modification. It is for this reason that a Hebbian synapse is sometimes referred to as a conjunctional synapse. For another interpretation of Hebb’s postulate of learning, we may think of the interactive mechanism

characterizing a Hebbian synapse in statistical terms. In particular, the correlation over time between presynaptic and postsynaptic signals is viewed as being responsible for a synaptic change. Accordingly, a Hebbian synapse is also referred to as a correlational synapse. Correlation is indeed the basis of learning [1].

II.11.2. Competitive Learning

In competitive learning, the output neurons of a neural network compete among themselves to become active. Whereas in a neural network based on Hebbian learning several output neurons may be active simultaneously, in competitive learning only a single output neuron is active at any one time. It is this feature that makes competitive learning highly suited to discover statistically salient features that may be used to classify a set of input patterns.

There are three basic elements to a competitive learning rule:

- A set of neurons that are all the same expect for some randomly distributed synaptic weights, and which therefore respond differently to a given set of input patterns.
- A limit imposed on the 'strength' of each neuron.
- A mechanism that permits the neurons to compete for the right to respond to a given subset of inputs, such that only one output neuron, or only one neuron per group, is active at a time. The neuron that wins the competition is called winner-takes-all-neuron.

Accordingly the individual neurons of the network learn to specialize on ensembles of similar patterns; in so doing they become feature detectors for different classes of input patterns.

In the simplest form of competitive learning, the neural network has a single layer of output neurons, each of which is fully connected to the input nodes. The network may include feedback connections among the neurons. In the network architecture described herein, the feedback connections perform lateral inhibition, with each neuron tending to inhibit the neuron to which it is laterally connected [1].

Clustering

Competitive learning is a learning procedure that divides a set of input patterns in clusters that are inherent to the input data. A competitive learning network is

provided only with input vectors x and thus implements an unsupervised learning procedure. We will show its equivalence to a class of ‘traditional’ clustering algorithms shortly. Another important use of these networks is vector quantisation.

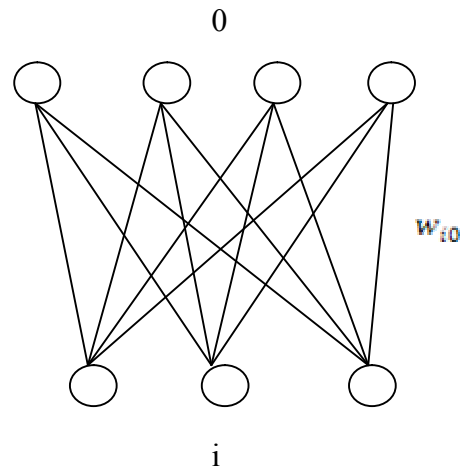


Figure II.7 A simple competitive learning network. Each of the four outputs 0 is connected to all inputs i [2].

An example of a competitive learning network is shown in figure II.7. All output units o are connected to all input units i with weights w_{io} . When an input pattern x is presented, only a single output unit of the network (the winner) will be activated. In a correctly trained network, all x in one cluster will have the same winner

II.11.3. Boltzmann Learning

The Boltzmann learning rule, named in honor of Ludwig Boltzmann, is a stochastic learning algorithm derived from ideas rooted in statistical mechanism. A neural network designed on the basis of the Boltzmann learning rule is called a Boltzmann machine [1].

In a Boltzmann machine the neurons constitute a recurrent structure, and they operate in a binary manner since, for example, they are either in an ‘on’ state denoted by $+1$ or in an ‘off’ state denoted by -1 [1].

II.11.4. Learning With a Teacher

We begin by considering learning with a teacher, which is also referred to as supervised learning. Figure II.6 shows a block diagram that illustrates this form of

learning. In conceptual terms we may think of the teacher as having knowledge of the environment, with that knowledge being represented by a set of input-output examples. The environment is, however, unknown to the neural network of interest. Suppose now that the teacher and the neural network are both exposed to a training vector drawn from the environment. By virtue of built-in knowledge, the teacher is able to provide the neural network with a desired response for that training vector. Indeed, the desired response represents the optimum action to be performed by the neural network. The network parameters are adjusted under the combined influence of the training vector and the error signal. The error signal is defined as the difference between the desired response and actual response of the network. This adjustment is carried out iteratively in a step-by-step fashion with the aim of eventually making the neural network emulate the teacher; the emulation is presumed to be optimum in some statistical sense. In this way knowledge of the environment available to the teacher is transferred to the neural network through training as fully as possible. When this condition is reached, we may then dispense with the teacher and let the neural network deal with the environment completely by itself [1].

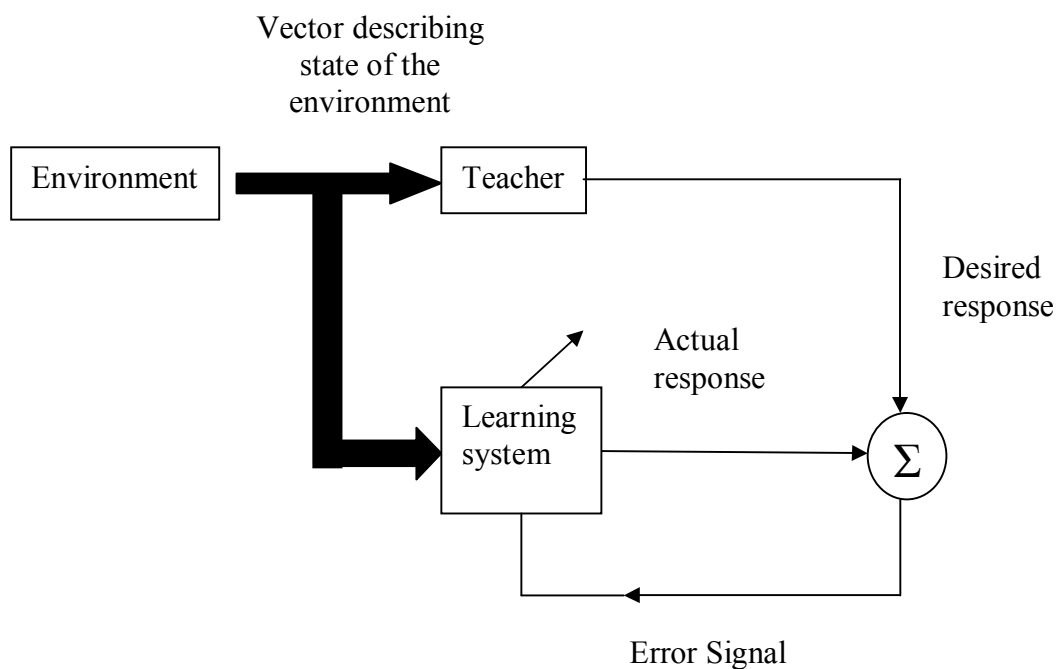


Figure II.8 Block diagram of learning with a teacher [1].

As a performance measure for a closed-loop feedback system we may think in terms of the mean-square error or the sum of squared errors over the training sample, defined as a function of the free parameters of the system. This function may be visualized as a multidimensional error-performance surface or simply error surface, with the free parameters as coordinates. The true error surface is averaged over all possible input-output examples. Any given operation of the system under the teacher's supervision is represented as a point on the error surface. For the system to improve performance over time and therefore learn from the teacher, the operating point has to move down successively toward a minimum point of the error surface; the minimum point may be a local minimum or a global minimum. A supervised learning system is able to do this with the useful information it has about the gradient of the error surface corresponding to the current behavior of the system. The gradient of an error surface at any point is a vector that points in the direction of steepest descent. In fact, in the case of supervised learning from examples, the system may use an instantaneous estimate of the gradient vector, with the example indices presumed to be those of time. The use of such an estimate results in a motion of the operating point on the error surface that is typically in the form of a 'random walk'. Nevertheless, given an algorithm designed to minimize the cost function, an adequate set of input-output examples, and enough time permitted to do the training, a supervised learning system is usually able to perform such tasks as pattern classification and function approximation [1].

II.11.5. Learning Without a Teacher

In supervised learning, the learning process takes place under the tutelage of a teacher. However, in the paradigm known as learning without a teacher, as the name implies, there is no teacher to oversee the learning process. That is to say, there are no labeled examples of the function to be learned by the network. Under this second paradigm, two subdivisions are identified:

1. Reinforcement Learning/Neurodynamic Programming

In reinforcement learning, the learning of an input-output mapping is performed through continued interaction with the environment in order to minimize

a scalar index of performance. Figure II.9 shows the block diagram of one form of a reinforcement learning system built around a critic that converts a primary reinforcement signal received from the environment into a higher quality reinforcement signal called the heuristic reinforcement signal, both of which are scalar inputs. The system is designed to learn under delayed reinforcement, which means that the system observes a temporal sequence of stimuli (i.e., state vectors) also received from the environment, which eventually result in the generation of the heuristic reinforcement signal. The goal of learning is to minimize a cost-to-go function, defined as the expectation of the cumulative cost of actions taken over a sequence of steps instead of simply the immediate cost. It may turn out that certain actions taken earlier in that sequence of time steps are in fact the best determinants of overall system behavior. The function discover these actions and to feed them back to the environment [1].

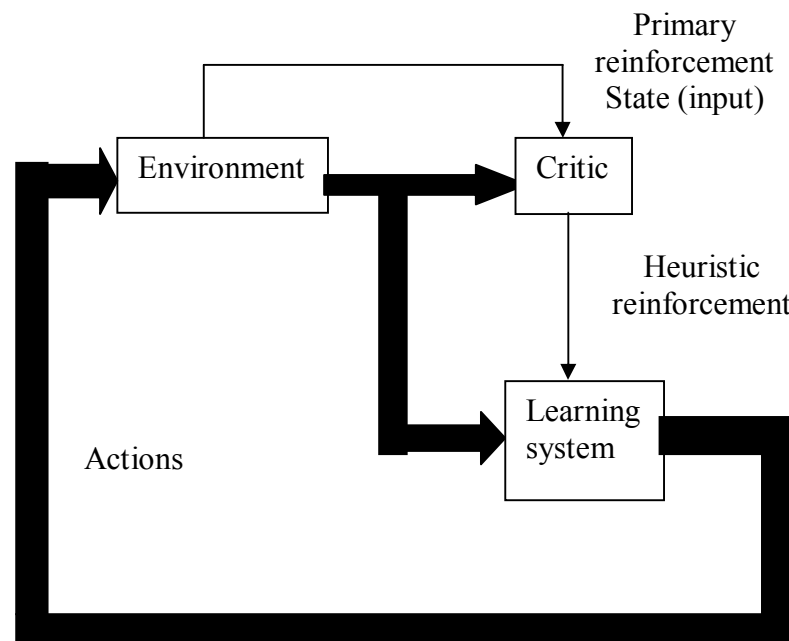


Figure II.9 Block diagram of reinforcement learning [1].

Delayed-reinforcement learning is difficult to perform for two basic reasons:

- There is no teacher to provide a desired response at each step of the learning process.

- The delay incurred in the generation of the primary reinforcement signal implies that the learning machine must solve a temporal credit assignment problem. By this we mean that the learning machine must be able to assign credit and blame individually to each action in the sequence of time steps that led to the final outcome, while the primary reinforcement may only evaluate the outcome. Notwithstanding these difficulties, delayed-reinforcement learning is very appealing. It provides the basis for the system to interact with its environment, thereby developing the ability to learn to perform a prescribed task solely on the basis of the outcomes of its experience that result from the interaction.

Reinforcement learning is closely related to dynamic programming, which was developed by Bellman in the context of optimal control theory. Dynamic programming provides the mathematical formalism for sequential decision making. By casting reinforcement learning within the framework of dynamic programming, the subject matter becomes all the richer for it, as demonstrated in Bertsekas and Tsitsiklis [1].

2. Unsupervised Learning

In unsupervised or self-organized learning there is no external teacher or critic to oversee the learning process, as indicated in Figure II.10. Rather, provision is made for a task-independent measure of the quality of representation that the network is required to learn, and the free parameters of the network are optimized with respect to that measure. Once the network has become tuned to the statistical regularities of the input data, it develops the ability to form internal representations for encoding features of the input and thereby to create new classes automatically [1].

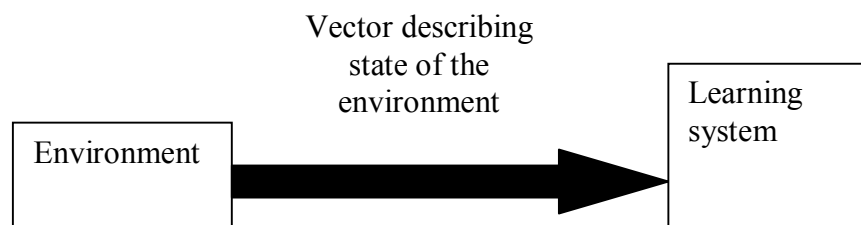


Figure II.10 Block diagram of unsupervised learning [1].

To perform unsupervised learning we may use competitive learning rule. For example, we may use a neural network that consists of two layers- an input layer and a competitive layer. The input layer receives the available data. The competitive layer consists of neurons that compete with each other (in accordance with a learning rule) for the 'opportunity' to respond to features contained in the input data. In its simplest form, the network operates in accordance with a 'winner-takes-all' strategy.

II.11.6. Learning Tasks

Pattern Association

An associative memory is a brainlike distributed memory that learns by association. Association has been known to be a prominent feature of human memory since Aristotle, and all models of cognition use association in one form or another as the basic operation.

Pattern Recognition

Pattern recognition is formally defined as the process whereby a received pattern/signal is assigned to one of a prescribed number of classes. A neural network performs pattern recognition by first undergoing a training session, during which the network is repeatedly presented a set of input patterns along with the category to which each particular pattern belongs. Later, a new pattern is presented to the network that has not been seen before, but which belongs to the same population of patterns used to train the network. The network is able to identify the class of that particular pattern because of the information it has extracted from the training data.

Control

The control of a plant is another learning task that can be done by a neural network; by a 'plant' we mean a process or critical part of a system that is to be maintained in a controlled condition. The relevance of learning to control should not be surprising because, after all, the human brain is a computer, the outputs of which as a whole system are actions.

- Indirect learning. Using actual input-output measurements on the plant, a neural network model is first constructed to produce a copy of it.

- Direct learning.

Filtering

The term filter often refers to a device or algorithm used to extract information about a prescribed quantity of interest from a set of noisy data. The noise may arise from a variety of sources.

We may use a filter to perform three basic information processing tasks:

1. Filtering
2. Smoothing
3. Prediction

Beamforming

Beamforming is a spatial form of filtering and is used to distinguish between the spatial properties of a target signal and background noise. The device used to do the beamforming is called a beamformer.

Memory

Memory refers to the relatively enduring neural alterations induced by the interaction of an organism with its environment. Memory and learning are intricately connected. When a particular activity pattern is learned, it is stored in the brain where it can be recalled later when required. Memory may be divided into 'short-term' and 'long-term' memory, depending on the retention time. Short-term memory refers to a compilation of knowledge representing the 'current' state of the environment. Any discrepancies between knowledge stored in short-term memory and a 'new' state are used to update the short-term memory. Long-term memory, on the other hand, refers to knowledge stored for a long time or permanently.

II.12. NETWORK ARCHITECTURES

We may identify three fundamentally different classes of network architectures:

1. Single-Layer Feedforward Networks

In a layered neural network the neurons are organized in the form of layers. In the simplest form of a layered network, we have an input layer of source nodes that

projects onto an output layer of neurons (computation nodes). In other words, this network is strictly a feedforward or acyclic type. It is illustrated in Fig. II.11 for the case of four nodes in both the input and output layers. Such a network is called a single-layer network, with the designation ‘single-layer’ referring to the output layer of computation nodes (neurons). We do not count the input layer of source nodes because no computation is performed there.

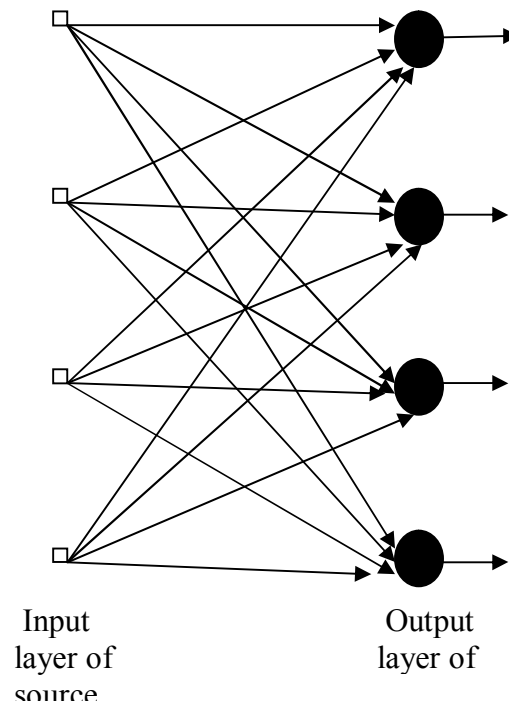


Figure II.11 Feedforward or acyclic network with a single layer of neurons [1]

2. Multilayer Feedforward Networks

The second class of a feedforward neural network distinguishes itself by the presence of one or more hidden layers, whose computation nodes are correspondingly called hidden neurons or hidden units. The function of hidden neurons is to intervene between the external input and the network output in some useful manner. By adding one or more hidden layers, the network is enabled to extract higher-order statistics.

3. Recurrent Networks

A recurrent network distinguishes itself from a feedforward neural network in that it has at least one feedback loop.

II.12.1. Perceptron Learning Rule And Convergence Theorem

Suppose we have a set of learning samples consisting of an input vector x and a desired output $d(x)$. For a classification task the $d(x)$ is usually +1 or -1. The perceptron learning rule is very simple and can be stated as follows:

- 1) Start with random weights for the connections.
- 2) Select an input vector x from the set of training samples;
- 3) If $y \neq d(x)$ (the perceptron gives an incorrect response), modify all connections w_i according to: $\Delta w_i = d(x)x_i$
- 4) Go back to 2 [2].

Note the procedure is very similar to the Hebb rule; the only difference is that, when the network responds correctly, no connection weights are modified. Besides modifying the weights, we must also modify the threshold θ . This θ is considered as a connection w_0 between the output neuron and a ‘dummy’ predicate unit which is always on: $x_0 = 1$. Given the perceptron learning rule as stated above, this threshold is modified according to [2]:

$$\Delta\theta = \begin{cases} 0 & \text{if perceptron responds correctly} \\ d(x) & \text{otherwise} \end{cases} \quad (\text{II.25})$$

II.13. FEEDFORWARD NEURAL NETWORK

A feedforward neural network is an artificial neural network where connections between the units do not form a directed cycle. This is different from recurrent neural networks.

The feedforward neural network was the first and arguably simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

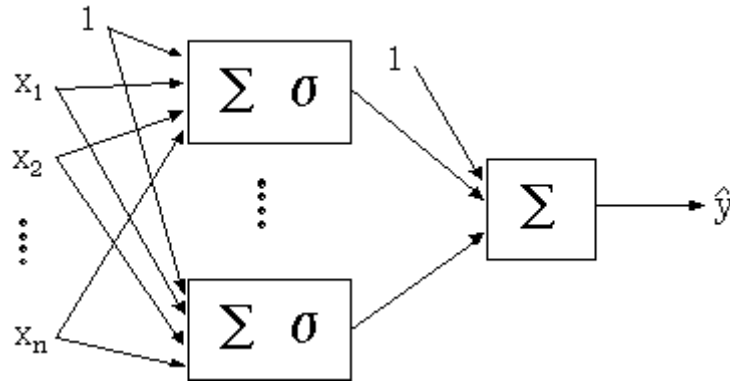


Figure II.12 A feedforward network with one hidden layer and one output [38]

Figure II.12 illustrates a one-hidden-layer FF network with inputs $x_1 \dots x_N$ and output $\hat{y}$. Each arrow in the figure symbolizes a parameter in the network. The network is divided into layers. The input layer consists of just the inputs to the network. Then follows a hidden layer, which consists of any number of neurons, or hidden units placed in parallel. Each neuron performs a weighted summation of the inputs, which then passes a nonlinear activation function, also called the neuron function.

Mathematically the functionality of a hidden neuron is described by ;

$$\sigma \left[\sum_{j=1}^n w_j x_j + b_j \right] \quad (\text{II.26})$$

where the weights $\{w_j, x_j\}$ are symbolized with the arrows feeding into the neuron.

The network output is formed by another weighted summation of the outputs of the neurons in the hidden layer. This summation on the output is called the output layer. In Figure II.12 there is only one output in the output layer since it is a single-output problem. Generally, the number of output neurons equals the number of outputs of the approximation problem.

The size of the input and output layers are defined by the number of inputs and outputs of the network and, therefore, only the number of hidden neurons has to be specified when the network is defined. The network in Figure II.12 is sometimes referred to as a three-layer network, counting input, hidden, and output layers. However, since no processing takes place in the input layer, it is also sometimes called a two-layer network. To avoid confusion this network is called a one-hidden-layer FF network throughout this documentation.

II.13.1. Single-Layer Perceptron

In the formative years of neural networks (1943-1958), several researchers stand out for their pioneering contributions:

- McCulloch and Pitts (1943) for introducing the idea of neural networks as computing machines.
- Hebb (1949) for postulating the first rule for self-organized learning.
- Rosenblatt (1958) for proposing the perceptron as the first model for learning with a teacher [1].

The perceptron is the simplest form of a neural network used for the classification of patterns said to be linearly separable. Basically, it consists of a single neuron with adjustable synaptic weights and bias. The algorithm used to adjust the free parameters of this neural network first appeared in a learning procedure developed by Rosenblatt (1958,1962) for his perceptron brain model. Indeed, Rosenblatt proved that if the patterns (vectors) used to train the perceptron are drawn from two linearly separable classes, then the perceptron algorithm converges and positions the decision surface in the form of a hyperplane between the two classes. The proof of convergence of the algorithm is known as the perceptron convergence theorem. The perceptron built around a single neuron is limited to performing pattern classification with only two classes. By expanding the output (computation) layer of the perceptron to include more than one neuron, we may correspondingly form classification with more than two classes. However, the classes have to be linearly separable for the perceptron to work properly. The important point is that insofar as the basic theory of the perceptron as a pattern classifier is concerned, we need consider only the case of a single neuron. The extension of the theory to the case of more than one neuron is trivial [1].

The single neuron also forms the basis of an adaptive filter, a functional block that is basic to the ever-expanding subject of signal processing. The development of adaptive filtering owes much to the classic paper of Widrow and Hoff (1960) for pioneering the so-called least-mean-square (LMS) algorithm, also known as the delta rule. The LMS algorithm is simple to implement yet highly effective in application. Indeed, it is the workhorse of linear adaptive filtering, linear in the sense

that the neuron operates in its linear mode. Adaptive filters have been successfully applied in such diverse fields as antennas, communication systems, radar, sonar, seismology, and biomedical engineering [1].

The LMS algorithm and the perceptron are naturally related.

A perceptron can be created using any values for the activated and deactivated states as long as the threshold value lies between the two. Most perceptrons have outputs of 1 or -1 with a threshold of 0 and there is some evidence that such networks can be trained more quickly than networks created from nodes with different activation and deactivation values [35].

Perceptrons can be trained by a simple learning algorithm that is usually called the delta rule. It calculates the errors between calculated output and sample output data, and uses this to create an adjustment to the weights, thus implementing a form of gradient descent.

Single-unit perceptrons are only capable of learning linearly separable patterns; in 1969 in a famous monograph entitled *Perceptrons* Marvin Minsky and Seymour Papert showed that it was impossible for a single-layer perceptron network to learn an XOR function. They conjectured (incorrectly) that a similar result would hold for a multi-layer perceptron network. Although a single threshold unit is quite limited in its computational power, it has been shown that networks of parallel threshold units can approximate any continuous function from a compact interval of the real numbers into the interval [-1,1]. This very recent result can be found in [20].

A multi-layer neural network can compute a continuous output instead of a step function. A common choice is the so-called logistic function:

$$y = \frac{1}{1 + e^{-x}} \quad (\text{II.27})$$

With this choice, the single-layer network is identical to the logistic regression model, widely used in statistical modeling. The logistic function is also known as the sigmoid function. It has a continuous derivative, which allows it to be used in backpropagation.

II.13.2. Multilayer Perceptrons

Multilayer feedforward network is an important class of neural networks. Typically, the network consists of a set of sensory units (source nodes) that

constitute the input layer, one or more hidden layers of computation nodes, and an output layer of computation nodes. The input signal propagates through the network in a forward direction, on a layer-by-layer basis. These neural networks are commonly referred to as multilayer perceptrons (MLPs).

A multilayer perceptron is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate output. It is a modification of the standard linear perceptron in that it uses three or more layers of neurons (nodes) with nonlinear activation functions, and is more powerful than the perceptron in that it can distinguish data that is not linearly separable [21].

Multilayer Perceptrons Architecture

The network diagram shown above is a full-connected, three layer, feed-forward, perceptron neural network. “Fully connected” means that the output from each input and hidden neuron is distributed to all of the neurons in the following layer. “Feed forward” means that the values only move from input to hidden to output layers.

All neural networks have an input layer and an output layer, but the number of hidden layers may vary. Here is a diagram of a perceptron network with two hidden layers and four total layers:

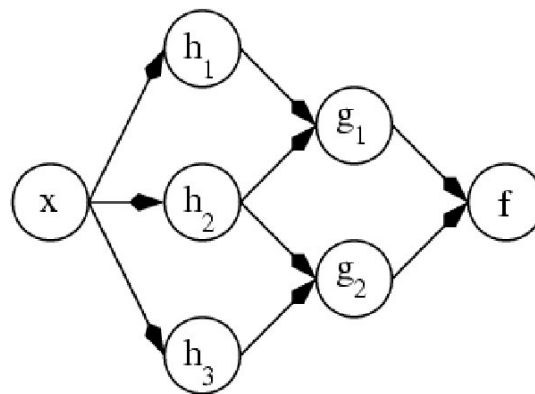


Figure II.13 A perceptron network with two hidden layers and four total layers [37]

When there is more than one hidden layer, the output from one hidden layer is fed into the next hidden layer and separate weights are applied to the sum going into each layer

An MLP is a network of simple *neurons* called *perceptrons*. The basic concept of a single perceptron was introduced by Rosenblatt in 1958. The perceptron computes a single *output* from multiple real-valued *inputs* by forming a linear combination according to its input *weights* and then possibly putting the output through some nonlinear activation function. Mathematically this can be written as

$$y = \varphi \left(\sum_{i=1}^n w_i x_i + b \right) = \varphi(w^T x + b) \quad (\text{II.28})$$

where w denotes the vector of weights, x is the vector of inputs, b is the bias and φ is the activation function. A signal-flow graph of this operation is shown in Figure II.14 [1] [22].

The original Rosenblatt's perceptron used a Heaviside step function as the activation function φ . Nowadays, and especially in multilayer networks, the activation function is often chosen to be the logistic sigmoid $\frac{1}{1 + e^{-x}}$ or the hyperbolic tangent $\tanh(x)$. They are related by $\frac{\tanh(x) + 1}{2} = \frac{1}{1 + e^{-2x}}$. These functions are used because they are mathematically convenient and are close to linear near origin while saturating rather quickly when getting away from the origin. This allows MLP networks to model well both strongly and mildly nonlinear mappings.

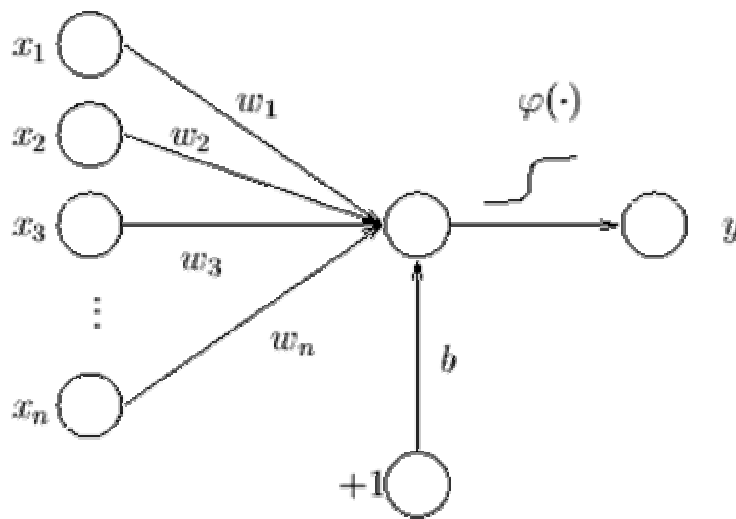


Figure II.14 Signal-flow graph of the perceptron [39].

Training Multilayer Perceptrons Networks

The goal of the training process is to find the set of weight values that will cause the output from the neural network to match the actual target values as closely as possible. There are several issues involved in designing and training a multilayer perceptron network:

- Selecting how many hidden layers to use in the network.
- Deciding how many neurons to use in each hidden layer.
- Finding a globally optimal solution that avoids local minima.
- Converging to an optimal solution in a reasonable period of time.
- Validating the neural network to test for overfitting.

Once the number of layers, and number of units in each layer, has been selected, the network's weights and thresholds must be set so as to minimize the prediction error made by the network. This is the role of the training algorithms. The historical cases that you have gathered are used to automatically adjust the weights and thresholds in order to minimize this error. This process is equivalent to fitting the model represented by the network to the training data available. The error of a particular configuration of the network can be determined by running all the training cases through the network, comparing the actual output generated with the desired or target outputs. The differences are combined together by an error function to give the network error. The most common error functions are the sum squared error (used for regression problems), where the individual errors of output units on each case are squared and summed together, and the cross entropy functions (used for maximum likelihood classification).

Multilayer perceptron have been applied successfully to solve some difficult and diverse problems by training them in a supervised manner with a highly popular algorithm known as the error back-propagation algorithm. This algorithm is based on the error-correction learning rule. As such, it may be viewed as a generalization of an equally popular adaptive filtering algorithm.

Basically, error back-propagation learning consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, an activity pattern (input vector) is applied to the sensory nodes of the network, and its effect propagates through the network layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights of the networks are all fixed. During the backward pass, on the

other hand, the synaptic weights are all adjusted in accordance with an error-correction rule. Specifically, the actual response of the network is subtracted from a desired (target) response to produce an error signal. This error signal is then propagated backward through the network, against the direction of synaptic connections- hence the name ‘error back-propagation’. The synaptic weights are adjusted to make the actual response of the network move closer to the desired response in a statistical sense. The error back-propagation algorithm is also referred to in the literature as the back-propagation algorithm, or simply back-prop. Henceforth we will refer to it as the back-propagation algorithm. The learning process performed with the algorithm is called back-propagation learning.

Multilayer perceptrons using a backpropagation algorithm are the standard algorithm for any supervised-learning pattern recognition process and the subject of ongoing research in computational neuroscience and parallel distributed processing. They are useful in research in terms of their ability to solve problems stochastically, which often allows one to get approximate solutions for extremely complex problems like fitness approximation.

Currently, they are most commonly seen in speech recognition, image recognition, and machine translation software, but they have also seen applications in other fields such as cyber security. In general, their most important use has been in the growing field of artificial intelligence, although the multilayer perceptron does not have connections with biological neural networks as initial neural based networks have [23].

II.13.3 The Adaptive Linear Element (ADALINE)

The Adaline is a device consisting of a single processing element; as such, it is not technically a neural network.

The term Adaline is an acronym; however, its meaning has changed somewhat over the years. Initially called the ADaptive LInear NEuron, it became the ADaptive LInear Element, when neural Networks fell out of favor in that late 1960’s. There are two basic modifications required to make the general PE structure into an adaline. The first modification is the addition of a connection with weight, w_0 , which we refer to as the bias term. This term is a weight on a connection that has its input value always equal to 1. The inclusion of such a term is largely a matter of experience [3].

The second modification is the addition of a bipolar condition on the output. This Adaline called adaptive linear combiner (ALC). If the output of the ALC is positive, the Adaline output is +1. If the ALC output is negative, the Adaline output is -1. Because much of the interesting processing takes place in the ALC portion of the Adaline, we shall concentrate on the ALC. Later, we shall add back the binary output condition [3].

The ALC performs a sum-of-products calculation using the input and weight vectors, and applies an output function to get a single output value.

$$y = w_0 + \sum_{j=1}^n w_j x_j \quad (\text{II.29})$$

where w_0 is the bias weight. If we make the identification, $x_0 = 1$, we can rewrite the preceding equation as

$$y = \sum_{j=0}^n w_j x_j \quad (\text{II.30})$$

or, in vector notation,

$$y = \mathbf{w}^t \mathbf{x} \quad (\text{II.31})$$

The output function in this case is the identity function, as is the activation function. The use of the identity function as both output and activation functions means that the output is the same as the activation, which is the same as the net input to the unit. [3].

The Adaline is ADaptive in the sense that there exists a well-defined procedure for modifying the weights in order to allow the device to give the correct output value for the given input [3].

An important generalisation of the perceptron training algorithm was presented by Widrow and Hoff as the ‘least mean square’ (LMS) learning procedure, also known as the delta rule. The main functional difference with the perceptron training rule is the way the output of the system is used in the learning rule. The perceptron learning rule uses the output of the threshold function (either -1 or +1) for learning.

The delta-rule uses the net output without further mapping into output values -1 or +1 [2].

The learning rule was applied to the ‘adaptive linear element’ also named Adaline, developed by Widrow and Hoff. In a simple physical implementation (Figure II.15) this device consists of a set of controllable resistors connected to a circuit which can sum up currents caused by the input voltage signals. Usually the central block, the summer, is also followed by a quantiser which outputs either +1 of -1, depending on the polarity of the sum [2].

Although the adaptive process is here exemplified in a case when there is only one output, it may be clear that a system with many parallel outputs is directly implementable by multiple units of the above kind.

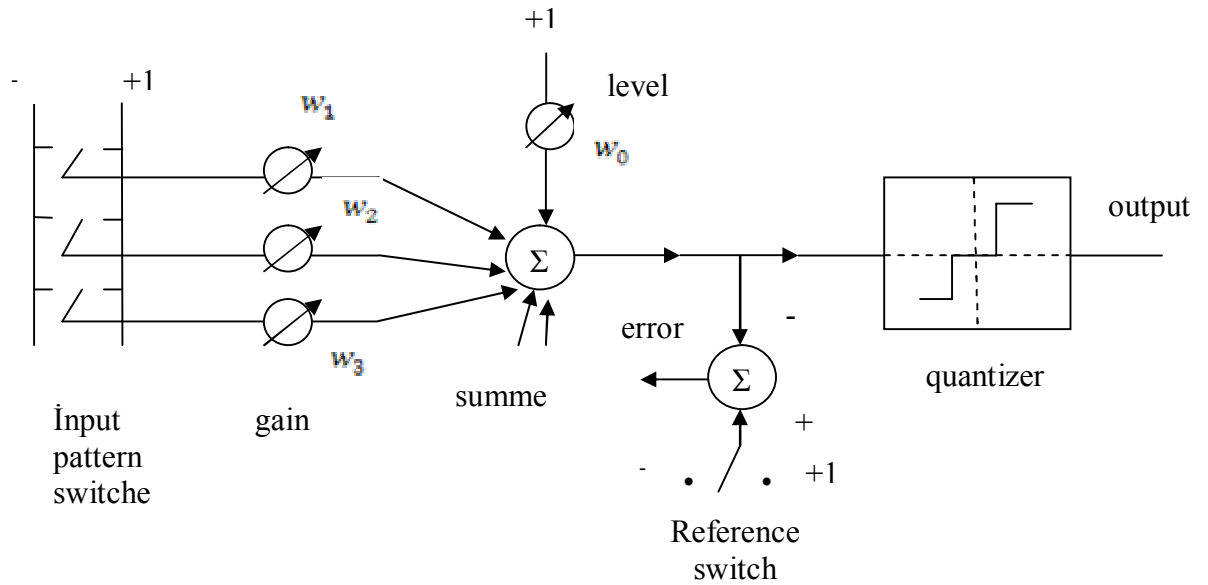


Figure II.15 The Adaline [2]

If the input conductances are denoted by w_i , $i = 0, 1, \dots, n$, and the input and output signals by x_i and y , respectively, then the output of the central block is defined to be

$$y = \sum_{i=1}^n w_i x_i + \theta \quad (\text{II.32})$$

where $\theta \equiv w_0$. The purpose of this device is to yield a given value $y = d^p$ at its output when the set of values x_i^p , $i = 1, 2, \dots, n$, is applied at the inputs. The problem is to determine the coefficients w_i , $i = 0, 1, \dots, n$, in such a way that the input-output response is correct for a large number of arbitrarily chosen signal sets. If an exact mapping is not possible, the average error must be minimised, for instance, in the sense of least squares. An adaptive operation means that there exists a mechanism by which the w_i can be adjusted, usually iteratively, to attain the correct values. For the Adaline, Widrow introduced the delta rule to adjust the weights [2].

II.13.4 MADALINE

The adaline resembles the perceptron closely; it also has some of the same limitations as the perceptron. For example, a two-input Adaline cannot compute the XOR function. Combining adalines in a layered structure can overcome this difficulty.

Madaline Architecture

Madaline is the acronym for Many Adalines. The madaline could be presented with a large-dimensional input vector – say, pixel values from a raster scan. With suitable training, the network could be taught to respond with a binary +1 on one of several output nodes, each of which corresponds to a different category of input image. Examples of each categorization are {cat, dog, armadillo, javelina} and {Flogger, Tom Cat, Eagle, Fulcrum}. In such a network, each of four nodes in the output layer corresponds to a single class. For a given input pattern, a node would have a +1 output if the input pattern corresponded to the class represented by that particular node. The other three nodes would have a -1 output. If the input pattern were not a member of any known class, the results from the network could be ambiguous [3].

To train such a network, we might be tempted to begin with the LMS algorithm at the output layer. Since the network is presumably trained with previously identified input patterns, the desired output vector is known. What we do not know is the desired output for a given node on one of the hidden layers. Furthermore, the LMS algorithm would operate on the analog outputs of the ALC, not on the bipolar

output values of the Adaline. For these reasons, a different training strategy has been developed for the Madaline.

II.13.5 Backpropagation

There are many potential computer applications that are difficult to implement because there are many problems unsuited to solution by a sequential process. Applications that must perform some complex data translation, yet have no predefined mapping function to describe the translation process, or those that must provide a ‘best guess’ as output when presented with noisy input data are but two examples of problems of this type.

An ANS that we have found to be useful in addressing problems requiring recognition of complex patterns and performing nontrivial mapping functions is the backpropagation network (BPN), formalized first by Werbos, and later by Parker and by Rummelhart and McClelland [2].

The Backpropagation Network

To illustrate some problems that often arise when we are attempting to automate complex pattern-recognition applications, let us consider the design of a computer program that must translate a 5x7 matrix of binary numbers representing the bit-mapped pixel image of an alphanumeric character to its equivalent eight-bit ASCII code [3].

II.13.6 Exclusive-Or Problem

One of Minsky and Papert’s most discouraging results shows that a single layer perceptron cannot represent a simple exclusive-or function [2].

In a simple network with two inputs and one output, the net input is equal to:

$$s = w_1x_1 + w_2x_2 + \theta \quad (\text{II.33})$$

The output of the perceptron is zero when s is negative and equal to one when s is positive.

For the specific XOR problem we geometrically show that by introducing hidden units, thereby extending the network to a multi-layer perceptron, the problem can be solved.

II.14. RECURRENT NETWORKS

A recurrent neural network (RNN) is a class of neural network where connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior.

Another tool that has been used for temporal classification problems is recurrent neural networks. A neural network is made of individual units termed neurons. Each neuron has a weight associated with each input. A function of the weights and inputs (typically, a squashing function applied to the sum of the weight-input products) is then generated as an output.

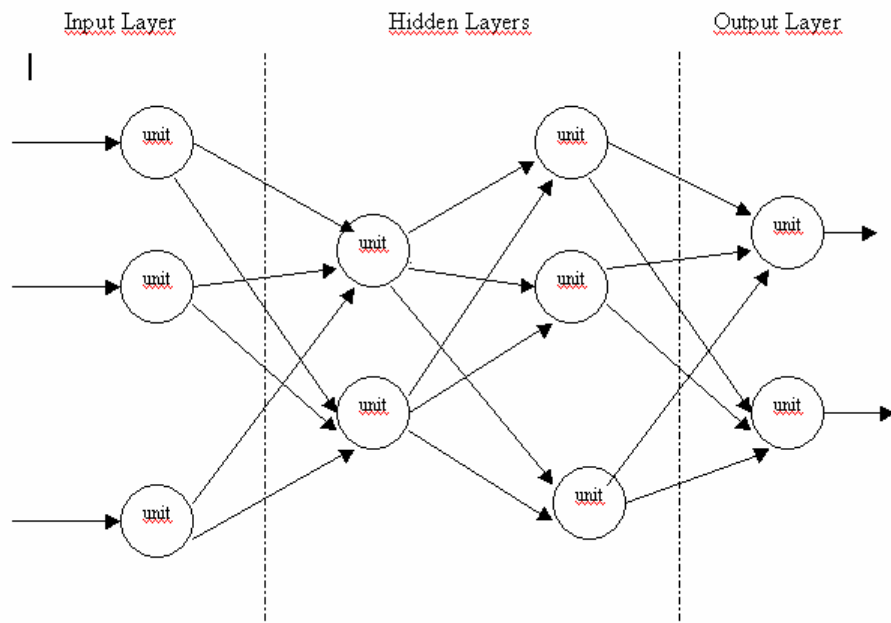


Figure II.16 A typical feedforward neural network [36]

These individual units are connected together as shown in Figure II.16, with an input layer, an output layer and usually one or more hidden layers. Typically, the input layer consists of one unit per attribute, and the output layer of one unit per class. The number of units and topology within the hidden layer is defined by the user. Through algorithms such as backpropagation, the weights of the neural net can be adjusted so as to produce an output on the appropriate unit when a particular pattern at the input is observed. The backpropagation algorithm works by running the training instance through the neural network, and calculating the difference

between the desired and actual outputs. These differences are then “propagated back” from the output layer to the hidden and input layers in the form of modifications to the weights of each of the component neurons. This modification is done in a manner proportional to the contribution to the difference in output, so that the weights most responsible for the difference are modified the most [36].

A recurrent neural network (RNN) is a modification to this architecture to allow for temporal classification, as shown in Figure II.17. In this case, a “context” layer is added to the structure, which retains information between observations. At each timestep, new inputs are fed into the RNN. The previous contents of the hidden layer are passed into the context layer. These then feed back into the hidden layer in the next time step.

To do classification, postprocessing of the outputs from the RNN is performed; so, for example, when a threshold on the output from one of the nodes is observed, we register that a particular class has been observed.

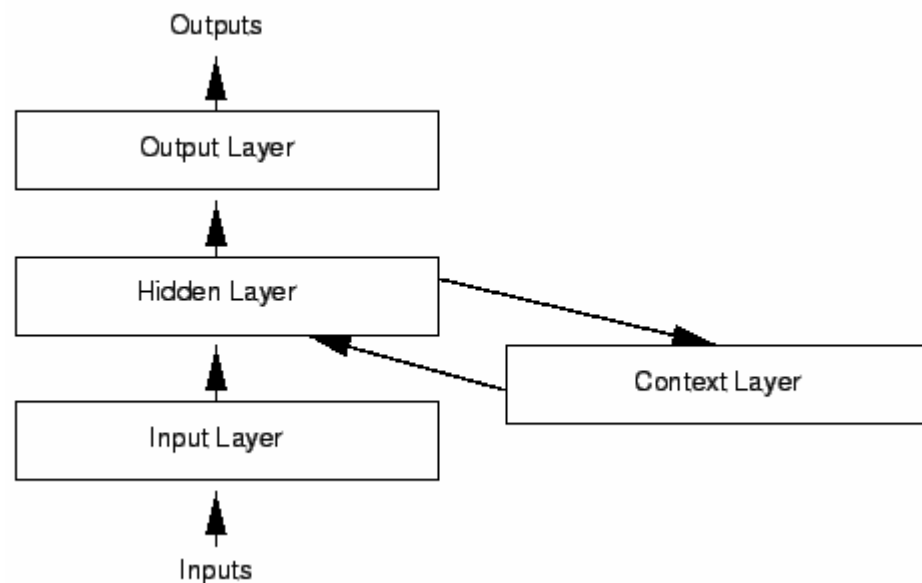


Figure II.17 Recurrent neural network architecture [36].

II.14.1. The Elman Network

The Elman network was introduced by Elman in 1990. In this network, a set of context units are introduced, which are extra input units whose activation values are fed back from the hidden units. Thus, the network is very similar to the Jordan

network, except that (1) the hidden units instead of the output units are fed back; and (2) the extra input units have no self-connections [2].

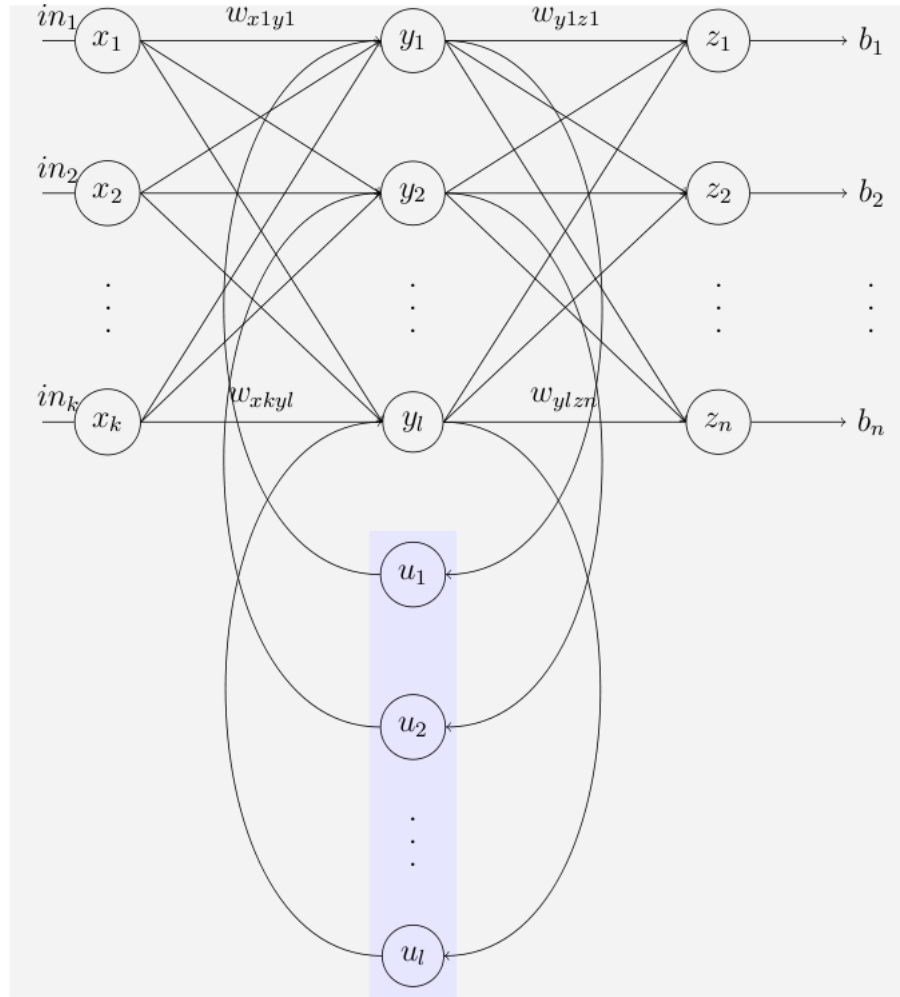


Figure II.18 An Elman network architecture [26].

A three-layer network is used, with the addition of a set of "context units" in the input layer. There are connections from the middle (hidden) layer to these context units fixed with a weight of one [24]. At each time step, the input is propagated in a standard feed-forward fashion, and then a learning rule is applied. The fixed back connections result in the context units always maintaining a copy of the previous values of the hidden units (since they propagate over the connections before the learning rule is applied). Thus the network can maintain a sort of state, allowing it to perform such tasks as sequence-prediction that are beyond the power of a standard multilayer perceptron [26].

Again the hidden units are connected to the context units with a fixed weight of value +1. learning is done as follows:

1. the context units are set to 0; $t = 1$;
2. pattern x^t is clamped, the forward calculations are performed once;
3. the back-propagation learning rule is applied;
4. $t \leftarrow t+1$; go to 2.

The context units at step t thus always have the activation value of the hidden units at step $t-1$ [2].

II.14.2. The Jordan Network

This network architecture is similar to the Elman network. The context units are however fed from the output layer instead of the hidden layer.

One of the earliest recurrent neural network was the Jordan network. In Jordan network, the activation values of the output units are fed back into the input layer through a set of extra input units called the state units. There are as many state units as there are output units in the network. The connections between the output and state units have a fixed weight of +1; learning takes place only in the connections between input and hidden and output units. Thus all the learning rules derived for the multi-layer perceptron can be used to train this network.

II.14.3. The Hopfield Network

The Hopfield network is a recurrent neural network in which all connections are symmetric. Invented by John Hopfield in 1982, this network guarantees that its dynamics will converge. If the connections are trained using Hebbian learning then the Hopfield network can perform as robust content-addressable memory, resistant to connection alteration [26].

One of the earliest recurrent neural networks reported in literature was the auto-associator independently described by Anderson and Kohonen in 1977. It consists of a pool of neurons with connections between each unit i and j , $i \neq j$. All connections are weighted [2].

In 1982, Hopfield brings together several earlier ideas concerning these networks and presents a complete mathematical analysis based on Ising spin models.

It is therefore that this network, which we will describe in this chapter, is generally referred to as the Hopfield network [2].

The Hopfield network consists of a set of N interconnected neurons which update their activation values asynchronously and independently of other neurons. All neurons are both input and output neurons. The activation values are binary. Originally, Hopfield chose activation values of 1 and 0, but using values +1 and -1 presents some advantages discussed below. We will therefore adhere to the latter convention.

The state of the system is given by the activation values $y = (y_k)$. The net input $s_k(t+1)$ of a neuron k at cycle $t+1$ is a weighted sum

$$s_k(t+1) = \sum y_j(t)w_{jk} + \theta_k \quad (\text{II.34})$$

II.15. KOHONEN NETWORK

The Kohonen network can be seen as an extension to the competitive learning network, although this is chronologically incorrect. Also, Kohonen network has a different set of applications.

In the Kohonen network, the output units in S are ordered in some fashion, often in a two-dimensional grid or array, although this is application-dependent. The ordering, which is chosen by user, determines which output neurons are neighbours.

Now, when learning patterns are presented to the network, the weights to the output units are thus adapted such that the order present in the input space R^N is preserved in the output, i.e., the neurons in S . This means that learning patterns which are near to each other in the input space (where ‘near’ is determined by the distance measure used in finding the winning unit) must be mapped on output units which are also near to each other, i.e., the same or neighbouring units. Thus, if inputs are uniformly distributed in R^N and the order must be preserved, the dimensionality of S must be at least

II.16. PRINCIPAL COMPONENT NETWORKS

The Networks presented in the previous sections can be seen as (nonlinear) vector transformations which map an input vector to a number of binary output element or neurons. The weights are adjusted in such a way that they could be

considered as prototype vectors (vectorial means) for the input patterns for which the competing neuron wins.

The self-organising transform described in this section rotates the input space in such a way that the values of the output neurons are as uncorrelated as possible and the energy or variances of the patterns is mainly concentrated in a few output neurons.

II.17. THE NONLINEAR AUTOREGRESSIVE NETWORK WITH EXOGENOUS INPUTS (NARX)

The NARX network is a dynamical neural architecture commonly used for input output modeling of nonlinear dynamical systems. All the specific dynamic networks discussed so far have either been focused networks, with the dynamics only at the input layer, or feedforward networks. The nonlinear autoregressive network with exogenous inputs (NARX) is a recurrent dynamic network, with feedback connections enclosing several layers of the network. The NARX model is based on the linear ARX model, which is commonly used in time-series modeling.

In time series modeling, a nonlinear autoregressive exogenous model (NARX) is a nonlinear autoregressive model which has exogenous inputs. This means that the model relates the present value of the time series to both.

- past values of the time series; and
- present and past values of the driving (exogenous) series.

In addition model contains:

- an “error” or “residual” term

which relates to the fact that knowledge of the other terms will not enable the present value of the time series to be predicted exactly. The defining equation for the NARX model is

$$y(t) = f(y(t-1), y(t-2), \dots, y(t-n_y), u(t-1), u(t-2), \dots, u(t-n_u)) \quad (\text{II.35})$$

or in a compact form;

$$y(t) = f[y(t-1); u(t-1)] \quad (\text{II.36})$$

where $u(t-1) \in \mathbb{R}$ and $y(t-1) \in \mathbb{R}$ denote, the input and output of the model at discrete time $t-1$, while $d_u \geq 1$ and $d_y \geq 1$, $d_u \leq d_y$ are the input-memory and output-memory orders. The vectors $y(t-1)$ and $u(t-1)$ denote the output and input regressors. The function f is (generally unknown) nonlinear function which should be approximated. The next value of the dependent output signal $y(t)$ is regressed on previous values of the output signal and previous values of an independent (exogenous) input signal. You can implement the NARX model by using a feedforward neural network to approximate the function f . A diagram of the resulting network is shown below, where a two-layer feedforward network is used for the approximation. This implementation also allows for a vector ARX model, where the input and output can be multidimensional.

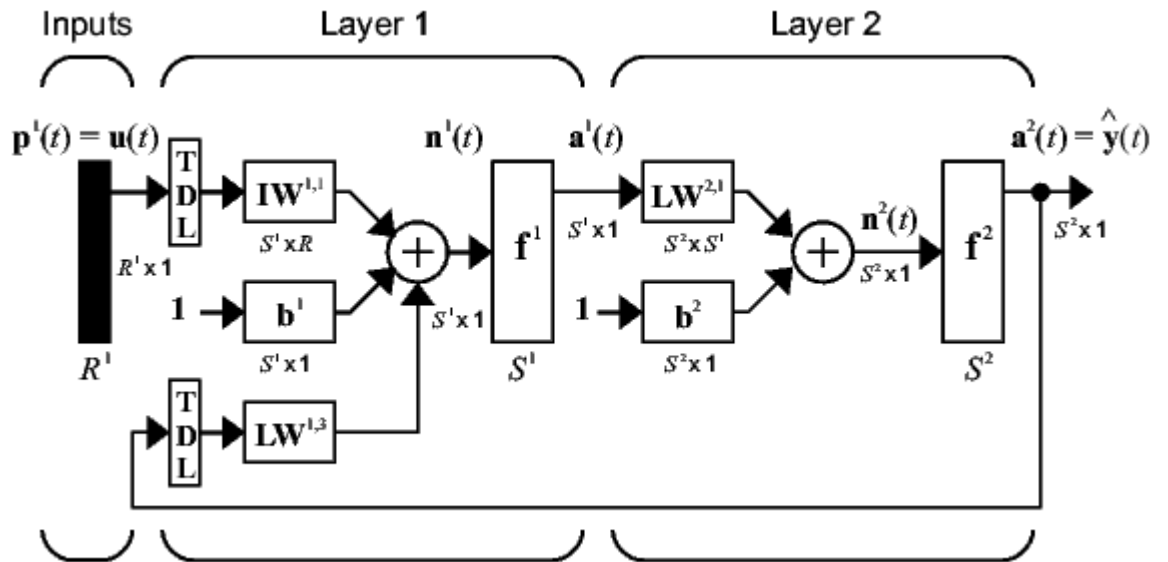


Figure II.19 NARX network structure [40].

There are many applications for the NARX network. It can be used as a predictor, to predict the next value of the input signal. It can also be used for nonlinear filtering, in which the target output is a noise-free version of the input signal. The use of the NARX network is demonstrated in another important application, the modeling of nonlinear dynamic systems.

Before demonstrating the training of the NARX network, an important configuration that is useful in training needs explanation. You can consider the output of the NARX network to be an estimate of the output of some nonlinear

dynamic system that you are trying to model. The output is fed back to the input of the feedforward neural network as part of the standard NARX architecture, as shown in the left figure below. Because the true output is available during the training of the network, you could create a series-parallel architecture, in which the true output is used instead of feeding back the estimated output, as shown in the right figure below. This has two advantages. The first is that the input to the feedforward network is more accurate. The second is that the resulting network has a purely feedforward architecture, and static backpropagation can be used for training.

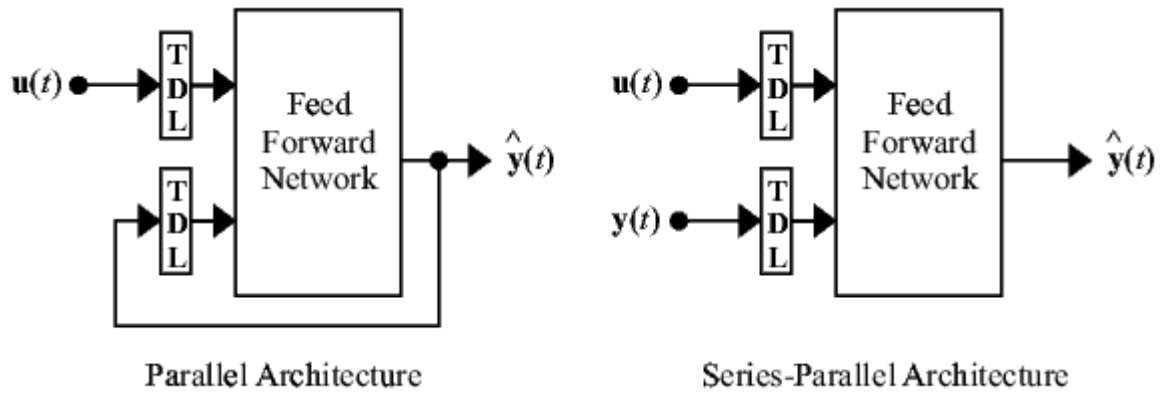


Figure II.20 NARX network parallel and series-parallel mode architecture [40]

In what concern training the NARX network, it can be carried out in one out of two modes:

_ Series-Parallel (SP) Mode - The following demonstrates the use of the series-parallel architecture for training an NARX network to model a dynamic system. In this case, the output's regressor is formed only by actual values of the system's output:

$$\begin{aligned}\hat{y}(t) &= \hat{f}[y_{sp}(t-1); u(t-1)] \\ &= \hat{f}[y(t-1), \dots, y(t-d_y)];\end{aligned}$$

$$u(t-1), u(t-2), \dots, u(t-d_u) \quad (II.37)$$

In order for the parallel response to be accurate, it is important that the network be trained so that the errors in the series-parallel configuration are very small.

_ Parallel (P) Mode - In this case, estimated outputs are fed back and included in the output's regressor:

$$\begin{aligned}\hat{y}(t) &= \hat{f}[y_p(t-1); u(t-1)] \\ &= \hat{f}[y(t-1), \dots, y(t-d_y)];\end{aligned}$$

$$u(t-1), u(t-2), \dots, u(t-d_u) \quad (\text{II.38})$$

NARX network eliminates a considerable portion of its representational capabilities as a dynamic network; that is, all the dynamic information that could be learned from the past memories of the output (feedback) path is discarded.

As a tool for nonlinear system identification, the NARX network has been successfully applied to a number of real-world input-output modelling problems, such as heat exchangers, waste water treatment plants, catalytic reforming systems in a petroleum refinery and nonlinear time series prediction .

CHAPTER III

FORECASTING METHODS

III.1. DECOMPOSITION METHODS FOR TIME-SERIES FORECASTING

Smoothing is to eliminate randomness so the pattern can be projected into the future and used as the forecast. The pattern can be broken down (decomposed) into subpatterns that identify each component of time series separately. Such a breakdown can frequently facilitate improved accuracy in forecasting and help in better understanding the behavior of the series.

Decomposition methods usually try to identify three separate components of the basic underlying pattern that tend to characterize economic and business series. These are the trend, the cycle, and the seasonal factors. The trend represents the long-run behavior of the data and can be increasing, decreasing, or unchanged. Often, the trend can be approximated by a straight line, but an exponential, S-curve, or some other long-term pattern may exist in certain situations. The cyclical factor represents the ups and downs of the economy or of a specific industry. It is common to such series as the gross national product (GNP), index of industrial production, demand for housing, sales of industrial goods such as automobiles, stock prices, bond rates, money supply, and interest rates. The cycle often follows the pattern of a wave, passing from a large to a small value and back again to a large value. The seasonal factor relates to periodic fluctuations of constant length that are caused by such things as temperature, rainfall, month of the year, timing of holidays, and corporate policies. The distinction between seasonality and cyclicity is that seasonality repeats itself at fixed intervals such as a year, a month, or a week, while cyclical factors have a longer duration that varies from cycle to cycle[5].

Decomposition assumes that the data consist of

$$\text{Data} = \text{pattern} + \text{error} = f(\text{trend, cycle, seasonality}) + \text{error}.$$

Thus in addition to the components of the pattern, an element of error or randomness is also assumed to be present. This error is assumed to be the difference

between the combined effect of the three subpatterns of the series and the actual data.

Decomposition methods are among the oldest forecasting approaches. They were used in the beginning of this century by economists attempting to identify and control the business cycle. The basis of current decomposition methods was established in the 1920s when the concept of ratio to trend was introduced. Since that time, decomposition approaches have been used widely by both economists and managers [5].

The general mathematical representation of the decomposition approach is:

$$Y_t = f(S_t, T_t, E_t) \quad (\text{III.1})$$

Where Y_t is time series value (actual data) at period t ,

S_t is the seasonal component at period t ,

T_t is the trend-cycle component at period t , and

E_t is the irregular component at period t [6].

III.2. EXPLANATORY VERSUS TIME SERIES FORECASTING

Explanatory forecasting assumes a cause and effect relationship between the inputs to the system and its output, as shown in Figure III.1.

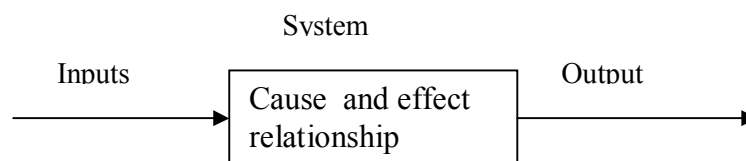


Figure III.1 Explanatory or causal relationship

According to explanatory forecasting, any change in inputs will affect the output of the system in a predictable way, assuming the cause and effect relationship is

constant. The first task of forecasting is to find the cause and effect relationship by observing the output of the system and relating that to the corresponding inputs.

Unlike explanatory forecasting, time series forecasting treats the system as a black box and makes no attempt to discover the factors affecting its behavior. As shown in Figure III.2, the system is simply viewed as an unknown generating process.

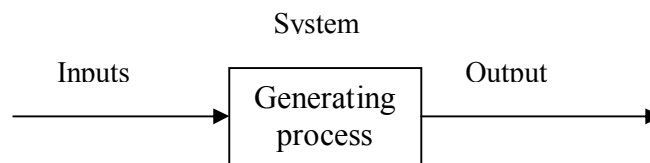


Figure III.2 Time-Series Relationship

There two main reasons for wanting to treat a system as a black box. First, the system may not be understood, and even if it were understood it may be extremely difficult to measure the relationships assumed to govern its behavior. Second, the main concern may be only to predict what will happen and not to know why it happens [7].

Quite often it is possible to forecast by using either causal or time series approaches.

III.3. TYPES OF MODELS

A forecasting model consists of the procedures used by that technique in developing a forecast. Clearly, a wide range of models could be used, but for quantitative methods they fall into two fairly well-defined categories. By understanding the properties of each of these models, the manager can get a better grasp of the assumptions that underlie individual forecasting techniques and the pros and cons of using them in specific situations [5].

The first type of quantetive forecasting model, and perhaps the most common, is the time-series model. Two factors are important in a time-series model: the data series to be forecast and the period of time to be used. A time-series model always assumes that some pattern or combination of patterns is recurring over time. Thus,

by identifying and extrapolating that pattern, forecasts for subsequent time periods can be developed [5].

In addition to the importance of the sequence of the periods as a variable in a time-series model, such a model assumes explicitly that the underlying pattern can be identified solely on the basis of historical data from that series. This means that it is not particularly helpful to the manager in predicting the impact of certain decisions he or she may make. A time-series model may be appropriate for forecasting environmental factors such as the general economy and level of employment or for forecasting activity levels where there are substantial inertia and little impact from individual decisions, but inappropriate for forecasting weekly sales resulting from changes in pricing and advertising [5].

One advantage of time-series models is that the basic rules of accounting are oriented toward sequential time periods. This means that in most firms data are readily available on the basis of these time periods and can be used in the application of a time-series forecasting method [5].

Time-series forecasting treats the system as a black box and makes no attempt to discover the factors affecting its behavior [5].

There are three main reasons for wanting to treat a system as a black box. First, the system may not be understood, and even if it is understood, it may be extremely difficult to measure the relationships assumed to govern its behavior. Second, the main concern may be only to predict what will happen, not to know why it happens. During the eighteenth, nineteenth, and twentieth centuries, for example, several scientists were concerned with the magnitude of sunspots. Little was known at that time about the reasons for the sunspots or the sources of energy of the sun. This lack of knowledge, however, did not hinder investigators who collected and analysed data about the frequency of sunspots. Schuster found a regular pattern in the magnitude of sunspots, and he and several others were able to predict their continuation through time-series analysis. Third, while it may be of value to know why something happens as well as predicting what will happen, the cost of doing the former may be extremely high, while the cost of the latter- using time-series methods for example- may be relatively low [5].

The second type of quantitative forecasting method is causal, or explanatory. Under such methods any change in inputs will affect the output of the system in a predictable way, assuming the cause and effect relationship is constant. The first task

of forecasting is to find the cause and effect relationship by observing the output of the system and relating that to the corresponding outputs [5].

Basically, a causal, or explanatory, method assumes that the value of a certain variable (the output) is a function of one or more other variables (the inputs). In very narrow sense a time-series model could be called an explanatory model, since the actual values are assumed to be a function of the time period alone. The descriptor, causal or explanatory model, however, is generally reserved for models with variables other than time [5].

The real strength of an explanatory model as a forecasting method is that a manager can develop a range of forecasts corresponding to a range of values for the different variables. However the difficulty of these methods is that they require information on several variables in addition to the variable that is being forecast. As a result, their data requirements are much larger than the data requirements of a time-series model. In addition, since explanatory models generally relate several factors, they usually take longer to develop and are more sensitive to changes in the underlying relationships than would be a time-series model. Furthermore, they require an estimation of future values of the input factors before the output variable can be forecast [5].

Quite often it is possible to forecast by using either causal or time-series methods.

III.3.1. Types Of Patterns In Time-Series Data

A horizontal pattern exists when there is no trend in the data. (statistically, this is referred to as stationarity.) When such a pattern exists, the series generally is referred to as a stationary: that is, it does not tend to increase or decrease over time in any systematic way. Thus it is just as likely that the next value of the series will be below it [5].

A seasonal pattern exists when series fluctuates according to some seasonal factor. The seasons may be the months or the four seasons of the year, but they could also be the hours of the day, the days of the week, or the days in a month. Seasonal patterns exist for a number of different reasons, varying from the way in which a firm has chosen to handle certain operations (internally caused seasons) to external factors such as the weather [5].

A cyclical pattern is similar to a seasonal pattern, but the length of a single cycle is generally longer than one year. Many series, such as the number of housing starts, the price of metals, gross national product (GNP), and the sales of many companies, contain a cyclical pattern. The cyclical pattern is a difficult one to predict, because it does not repeat itself at constant intervals of time, and its duration is not uniform [5].

A trend pattern exists when there is a general increase or decrease in the value of the variable over time. The sales of many companies, the gross national product, prices, and many other business and economic indicators follow a trend pattern [5].

III.4. SIMPLE TIME-SERIES SMOOTHING METHODS

In such situations managers frequently use a class of forecasting methods referred to as smoothing methods. With all methods of this type, historical data are used to obtain a 'smoothed' value for the series. That smoothed value then becomes the forecast for the future value of the series [5].

Two subclasses of smoothing methods will be described. The first, called averaging methods, conforms to the conventional definition of an average – namely, equal weighting (or smoothing) of the number of values included in the average. The second, called exponential smoothing methods, applies an unequal set of weights to past data. These weights decay in an exponential manner from the most recent data value to the most distant value [5].

The basic notion inherent in exponential smoothing and other forms of smoothing techniques is that there is some underlying pattern in the values of the variables to be forecast and that the historical observations of each variable represent the underlying pattern as well as random fluctuations. The goal of these forecasting methods is to distinguish between the random fluctuations and the basic underlying pattern by 'smoothing' the historical values. This amounts to eliminating the extreme values found in the historical sequence and basing a forecast on some smoothed intermediate values [5].

III.5. LEAST SQUARES ESTIMATES

$$\text{Data} = \text{pattern} + \text{error.} \quad (\text{III.2})$$

The critical task in forecasting is to separate the pattern from the error component so that the former can be used for forecasting.

The general procedure for estimating the pattern of a relationship, whether causal or time series, is through fitting some functional form in such a way as to minimize the error component of equation (III.2). One form of this estimation is least squares.

The name least squares is based on the fact that this estimation procedure seeks to minimize the sum of the squared errors in equation (III.2).

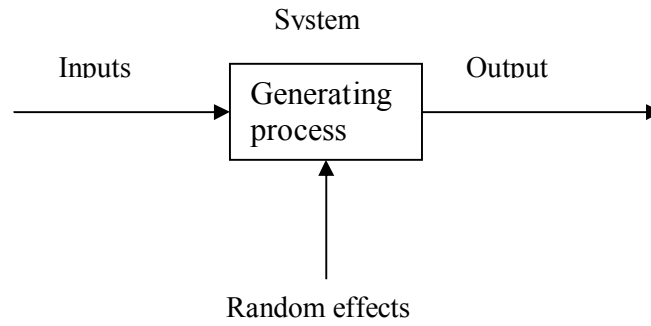


Figure III.3 Time-Series relationship with random noise

$$\text{Error} = \text{data} - \text{pattern.} \quad (\text{III.3})$$

For convenience, the error will be denoted by e , the data by X and the pattern . In addition, subscript i ($i = 1, 2, 3, \dots, 12$) will be added to denote the i th customer [7].

Using this notation, equation (III.3) becomes:

$$e_i = X_i - \bar{X} \quad (\text{III.4})$$

To examine the squared error, both sides of (III.4) must be squared, giving:

$$e_i^2 = (X_i - \bar{X})^2 \quad (\text{III.5})$$

III.6. SMOOTHING METHODS

If the mean is used a forecasting tool, then, as with all forecasting methods optimal use requires a knowledge of the conditions that determine its appropriateness. For the mean, the condition is that the data must be stationary, a term meaning that the process generating the data is in equilibrium around a constant value and that the variance around the mean remains constant over time.

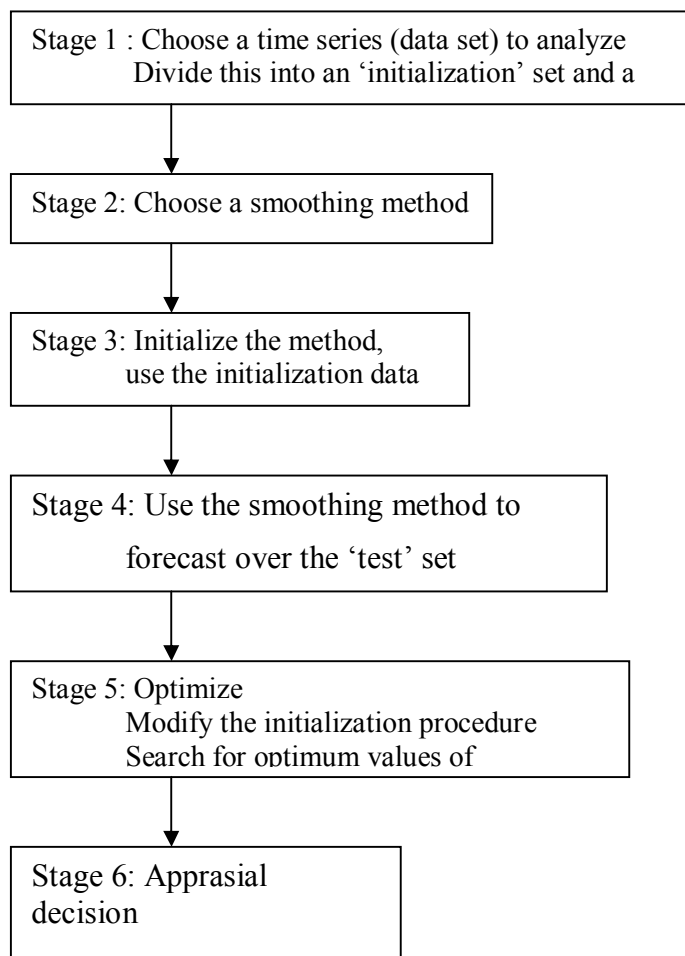


Figure III.4 A strategy for appraising any of the smoothing methods of forecast [7].

Thus if a time series is generated by a constant process subject to random error (or noise), then the mean is a useful statistic and can be used as a forecast for the next period(s). In terms of data requirements only two numbers need to be stored-

the latest mean and the number of data periods used to compute the mean. However, if the time series involves a trend (in an upward or downward direction), or a seasonal effect (strong sales of paper cups for hot drinks in winter months, for example), or both a trend and a seasonal effect, then the simple average is no longer able to capture the data pattern.

Figure III.4 describes a strategy for appraising any forecasting methodology. In stage 1, the time series of interest is identified. This may be a real data series (e.g., sales of a product) or, for learning purposes, it may be an artificially generated time series. The data set is then divided into two parts – an ‘initialization set’ and a ‘test set’ – so that an appraisal of a forecasting method can be conducted. In stage 2, a forecasting method is chosen from the list of smoothing methods. Stage 3 makes use of the initialization data set to get the forecasting method started. Estimates of any trend components, seasonal components, and parameter values are made at this stage. In stage 4 the method is applied to the test set to see how well it does. After each forecast, the forecasting error is determined, and over the complete test set certain measures of forecasting success will be determined. Stage 5 is really an iterative phase. Since there is no guarantee that initialization process and/or searching for optimum values of parameters in the model. Finally, in stage 6 the forecasting method is appraised as to its suitability for various kinds of data patterns and application potential is thereby made clearer [7].

III.6.1. Averaging Methods

III.6.1.1. The Mean

Given a data set covering the last N time periods:

$$X_1 \ X_2 \ X_3 \ \dots \qquad X_{N-1} \ X_N$$

and a decision to use the first T data points as ‘the initialization set’ and the rest as a ‘test set’

$X_1 \ X_2 \ \dots$	X_T	INITIALIZATION SET
$X_{T+1} \ \dots$	X_N	TEST SET

the method of simple averages is to take the average of all data in the initialization set

$$\bar{X} = \frac{\sum_{i=1}^T X_i}{T} = F_{t+1} \quad (\text{III.6})$$

As the forecast for period (T+1)'s data becomes available it is possible to compute an error :

$$e_{T+1} = X_{T+1} - F_{T+1} \quad (\text{III.7})$$

For period (T+2) the situation is

X_1	$X_2 \dots$	X_T	X_{T+1}	INITIALIZATION SET
$X_{T+2} \dots$	X_N			TEST SET

The average of all observed data as the forecast [6].

$$F_{t+1} = \frac{1}{t} \sum_{i=1}^t Y_i \quad (\text{III.8})$$

III.6.1.2. Moving Averages

The trend cycle can be estimated by smoothing the series to reduce the random variation. A range of smoothers is available, but we will begin with the simplest and oldest smoother, a moving average.

Simple Moving Average

The idea behind moving averages is that observations which are nearby in time are also likely to be close in value. So taking an average of the points near an observation will provide a reasonable estimate of the trend-cycle at that observation.

The term 'moving average' is used to describe this procedure because each average is computed by dropping the oldest observation and including the next observation.

One way to modify the influence of past data on the mean-as-a-forecast is to specify at the outset just how many past observations will be included in a mean. The term moving average is used to describe this procedure because as each new

observation becomes available, a new average can be computed by dropping the oldest observation and including the newest one. This moving average will then be the forecast for the next period.

$X_1 X_2 \dots X_T$

INITIALIZATION SET

$X_{T+1} \dots X_N$

TEST SET

Time

Moving Average

Forecast

$$T \quad \bar{X} = \frac{X_1 + X_2 + \dots + X_T}{T} \quad F_{t+1} = \bar{X} = \sum_{i=1}^T \frac{X_i}{T}$$

Compared with the simple mean (of all pat data) the moving average of order T has the following characteristics:

- it deals only with the latest T periods of known data,
- the number of data points in each average does not change as time goes on.

But it also has the following disadvantages:

- it requires more storage because all of the T latest observations must be stored, not just the average,
- it cannot handle trend or seasonality very well, although it can do better than the total mean [7].

If substantial randomness is contained in the series, a naive approach will produce forecasts that will vary considerably. To eliminate this randomness we might consider the use of some kind of average of recent observed values. The method of moving averages does this by taking a set of observed values, finding their average, and then using that average as a forecast for the coming period. The actual number of observations included in the average is specified by the manager and remains constant. The term moving average is used because as each new

observation becomes available, a new average can be computed and used as a forecast [5].

The method of moving averages could be used to forecast two or three months in advance as well as one month in advance. However, the likelihood of error would be much greater because the same three actual values used in a three-month moving average for one month ahead would be used in a three-month moving average for three months ahead. Thus in most cases the method of simple moving averages is applied for only one period in advance.

For a better understanding of the assumptions underlying the technique of moving averages and some of its advantages and limitations, it is necessary to look briefly at the mathematical representation of this method. In simple terms the technique of forecasting with moving averages can be represented as follows [5] :

$$F_{t+1} = S_t = \frac{X_t + X_{t-1} + \dots + X_{t-N+1}}{N} = \frac{1}{N} \sum_{i=t-N+1}^t X_i \quad (\text{III.9})$$

Where F_{t+1} is the forecast for time $t+1$,

S_t is the smoothed value at time t ,

X_i is the actual value at time i ,

i is the time period,

N is the number of values included in the average [5].

The moving average forecast for period t is given by

$$F_{t+1} = \frac{X_t}{N} - \frac{X_{t-N}}{N} + F_t \quad (\text{III.10})$$

Double Moving Averages

The centered moving averages are an example of how a moving average can itself be smoothed by another moving average. Together the smoother is known as a double moving average.

Double, triple, and other moving average combinations, on their own, imply a weighting of past data, where the largest weights are given to the values of the set of past data. As such they are useful for smoothing (as opposed to forecasting) data series and will most often be used as centered moving averages. However, when

used in the forecasting context the implied system places more emphasis on the most recent data.

III.6.2. Exponential Smoothing

At least two major limitations to the use of moving averages have prompted most forecasters to apply the method of exponential smoothing in its place. First, to compute a moving average forecast it is necessary to store the last N observed values, which takes up considerable space and in many computer systems is costly. Second, the method of moving averages gives equal weight to each of the last N observations and no weight at all to observations before period $(t-N)$; that is, the weight given to each of the last N observations is $1/N$ and 0 for any previous observations. A strong argument can be made that since the most recent observations contain the most information about what will happen in the future, they should be given relatively more weight than the older observations. What we would like is a weighting scheme that would apply the most weight to the most recent observed values and decreasing weights to the older values. Exponential smoothing satisfies this requirement and eliminates the need for storing the historical values of the variable [5].

In principle, exponential smoothing operates in a manner analogous to that of moving averages by ‘smoothing’ historical observations to eliminate randomness. The mathematical procedure for performing this smoothing, however, is somewhat different from that used in moving averages.

A reasonable estimate would be the forecast value from the preceding period. Thus equation (III.11) could be modified as:

$$F_{t+1} = \frac{1}{N}X_t + \left(1 - \frac{1}{N}\right)F_t \quad (\text{III.11})$$

What we now have is a forecast that weights the most recent observation with a weight of value $1/N$ and the most recent forecast with a weight of value $(1-1/N)$. If we substitute a symbol called alpha in place of $1/N$, we have

$$F_{t+1} = \alpha X_t + (1 - \alpha)F_t \quad (\text{III.12})$$

This equation is the general form used in computing a forecast by the method of exponential smoothing. Note that it immediately eliminates one of the problems associated with moving averages in that extensive historical data no longer need to be stored. Rather, only the most recent observation, the most recent forecast, and a value for α are required to prepare a new forecast [5].

Exponentially decreasing weights as the observations get older. Thus they are called exponential smoothing procedures. As in the case of moving averages there are single, double, and more complicated exponential smoothing methods. They all have in common the property that recent values are given relatively more weight in forecasting than older observations.

In the case of moving averages, the weights assigned to observations are a by-product of the particular MA system adopted. In exponential smoothing, however, there are one or more smoothing parameters to be determined explicitly, and these choices determine the weights assigned to observations, as will be indicated below.

Exponential smoothing methods all have in common the property that recent values are given relatively more weight in forecasting than the older observations.

In exponential smoothing, however, there are one or more smoothing parameters to be determined explicitly.

Single Exponential Smoothing

$$F_{t+1} = F_t + \left(\frac{X_t}{N} - \frac{X_{t-N}}{N} \right) \quad (\text{III.13})$$

Suppose we wish to forecast the next value of our time series Y_t which is yet to be observed. Our forecast is denoted by F_t . When the observation Y_t becomes available, the forecast error is found to be $Y_t - F_t$. The method of exponential forecasting takes the forecast for the previous period and adjusts it using the forecast error. That is, the forecast for the next period is

$$F_{t+1} = F_t + \alpha(Y_t - F_t) \quad (\text{III.14})$$

where α is a constant between 0 and 1 [6].

Double Exponential Smoothing: BROWN'S ONE-PARAMETER LINEAR METHOD

The underlying rationale of Brown's linear exponential smoothing is similar to that of linear moving averages: since both the single and double smoothed values lag the actual data when a trend exists, the difference between the single and double smoothed values can be added to the single smoothed values and adjusted for trend.

Triple Exponential Smoothing: BROWN'S ONE-PARAMETER QUADRATIC METHOD

Just as linear exponential smoothing can be used to predict data with a basic trend pattern, higher forms of smoothing can be used when the basic underlying pattern of the data is quadratic, cubic, or higher order. To go from linear quadratic smoothing, the basic approach is to incorporate an additional level of smoothing and to deal with a quadratic forecasting equation.

Linear Exponential Smoothing

Single exponential smoothing is appropriate when the data series contains a horizontal pattern (that is, it does not contain a trend). If single exponential smoothing is used with a data series that contains a trend, the forecasts will trail behind (lag) that trend. The method of linear exponential smoothing avoids this problem by explicitly recognizing the presence of trend.

Linear exponential smoothing recognizes this fact and uses it through equation to prepare a smoothed estimate of the trend in the data series:

$$T_t = \beta(S_t - S_{t-1}) + (1 - \beta)T_{t-1} \quad (\text{III.15})$$

where S_t is the equivalent of the single exponential smoothed value,

β is a smoothing coefficient, analogous to α , and

T_t is the smoothed trend in the data series [7].

Exponential Smoothing: PEGELS' CLASSIFICATION

An important consideration in dealing with exponential smoothing methods that deal with separate trend and seasonal aspects is whether or not the model should

be additive (linear) or multiplicative (nonlinear). Pegels has provided a simple but useful framework for discussing these matters.

$$S_t = \alpha P + (1 - \alpha)Q \quad (\text{III.16})$$

III.6.3. Other Smoothing Methods

- Chow's Adaptive Control Method
- Brown's One-Parameter Adaptive Method
- Box-Jenkins Three-Parameter Smoothing
- Harrison's Harmonic Smoothing Method

Holt's Linear Method

Holt (1957) extended single exponential smoothing to linear exponential smoothing to allow forecasting of data with trends. The forecast for Holt's linear exponential smoothing is found using two smoothing constants, α and β (with values between 0 and 1), and three equations:

$$L_t = \alpha Y_t + (1 - \alpha)(L_{t-1} + b_{t-1}) \quad (\text{III.17})$$

$$b_t = \beta(L_t - L_{t-1}) + (1 - \beta)b_{t-1} \quad (\text{III.18})$$

$$F_{t+m} = L_t + b_t m \quad (\text{III.19})$$

L_t denotes an estimate of the level of the series at time t and b_t denotes an estimate of the slope of the series at time t [6].

Winters' Linear and Seasonal Exponential Smoothing

Another useful form of smoothing deserves discussion. It was developed by Winters in the early sixties. This method produces results similar to those of linear exponential smoothing as we have discussed it, but it has the extra advantage of being capable of dealing with seasonal data as well as data that have a trend. It can, therefore, be used to deal with data series that contain a trend and seasonal pattern in addition to cycle/randomness.

Winters' linear and seasonal exponential smoothing is based on three equations, each of which smooths a factor associated with one of the three components of the pattern-randomness/cycle, linear, and seasonal. In this respect it is similar to linear exponential smoothing which smooths for cycle/randomness and adjust for trend. However, Winters' method includes an additional parameter to deal with seasonality. There are three basic smoothing equations involved in Winters' method. They are as follows:

$$S_t = \alpha \frac{X_t}{I_{t-L}} + (1 - \alpha)(S_{t-1} + T_{t-1}) \quad (\text{III.20})$$

$$T_t = \beta(S_t - S_{t-1}) + (1 - \beta)T_{t-1} \quad (\text{III.21})$$

$$I_t = \gamma \frac{X_t}{S_t} + (1 - \gamma)I_{t-1} \quad (\text{III.22})$$

where S is a smoothed value of the deseasonalized series,

T is a smoothed value of the trend,

I is a smoothed value of the seasonal factor, and

L is the length of seasonality (e.g., number of months or quarters in a year) [5].

The forecast based on Winters' method is computed as

$$F_{t+m} = (S_t + mT_t)I_{t-L+m} \quad (\text{III.23})$$

III.7. REGRESSION METHODS

The exponential smoothing methods were suggested to be appropriate for immediate or short-term forecasting when large numbers of forecasts are needed.

The decomposition methods were found to require more computations.

The decomposition approach to forecasting requires more time and is therefore restricted to forecasting fewer items than the simpler smoothing methods.

Single time-series, it is quite another to come up with other variables that relate to the data series of interest and to develop a model that expresses the functional relationship among the variables.

Sometimes the forecaster will wish to predict one variable Y (e.g., sales) and have available one explanatory variable X (e.g., advertising, expenditure). The objective is to develop an explanatory model relating Y and X . This is known as simple regression. In other situations, there will be one variable to forecast (Y) and several explanatory variables ($X_1, X_2, \dots, X_k$) and the objective will be to find a functional that relates Y to all of the explanatory variables. This is multiple regression Y on X_1 through X_k [6].

If the data are measured over time, then it will be called time series regression. If the data measurements are all taken at the same time, it will be referred to as cross-sectional regression.

III.7.1. Simple Regression

‘Simple regression’ will refer to any regression of a single Y variable (the forecast and dependent variable) on a single X variable (the explanatory or independent variable). The general situation will involve a set of n paired observations to be denoted:

$$(X_i, Y_i) \text{ for } i = 1, 2, 3, \dots, n.$$

The exponential smoothing methods were suggested to be appropriate for immediate or short-term forecasting when large numbers of forecasts are needed, such as at the operating level of a company.

Sometimes the forecaster will be dealing with one dependent measure and one independent measure. The objective is to develop an explanatory model relating these two measures.

The reason for making this distinction between linear and nonlinear in the coefficients is computational interest. It is relatively easy to solve for the coefficients of a regression equation when it is linear in the coefficients. It is much more difficult if the equation is not linear in the coefficients. Whether it is linear or nonlinear in the variables is of less importance.

The forecaster, then, must decide on how many variables to deal with, which one(s) will be dependent and which will be independent, and which functional form

will be chosen. If the data are measured over time, then it will be called time-series regression. If there is no time index it will be referred to as cross-sectional regression. Eventhough cross-sectional regression does not deal with time explicitly, many important decisions affecting the future are made on the basis of such studies. In that sense it is proper to consider cross-sectional regression in a book on forecasting.

Single equation models are more restrictive than multiple equation models; they are also simpler to deal with and may be sufficient for many applications. In the hands of skilled forecaster, there are many variations on the conventional regression technique that can be used to good effect. Some of the independent variables can be dummy variables to indicate the presence or absence of some significant event or to denote seasonality.

In attempting to model the relationships within one set of variables it might seem wise to remove another set of variables that interfere with the measures of interest. This process is called ‘partialling out a set of variables’, and really means trying to remove the influence of these variables on the underlying relationships being explored.

Least Squares Estimation

The line of the best fit is chosen to be the one which yields the smallest value for this sum of the squared errors. This is known as ordinary least squares (OLS) estimation or just LS (least squares) for short.

The Correlation Coefficient

It often occurs that two variables are related to each other.

The coefficient of correlation, r , is a relative measure of the linear association between two numerical variables. It can vary from 0 (which indicates no correlation) to ± 1 (which indicates perfect correlation). When the correlation coefficient is greater than 0, the two variables are said to be positively correlated (when one is large, the other is large), and when it is less than 0, they are said to be negatively correlated (when one is large, the other is small) [6].

The correlation coefficient can range from an extreme value of -1 (perfect negative correlation) through zero to an extreme value of $+1$ (perfect positive correlation) [6].

Intuitively, the correlation tell us two things.

- 1- The sign of the correlation coefficient (+ or -) indicates the direction of the relationship between the two variables. If it is positive, they tend to increase and decrease together; if it is negative, one increases while the other decreases; if it is close to zero, they move their separate ways.
- 2- The magnitude of the correlation coefficient is a measure of the strength of the association- meaning that as the absolute value of the correlation moves away from zero, the two variables are more strongly associated [6].

III.7.2. Multiple Regression

In multiple regression there is one dependent variable to be predicted, but there are two or more independent variables. The general form of multiple regression is

$$Y = b_0 + b_1X_1 + b_2X_2 + \dots + b_kX_k + e \quad (\text{III.24})$$

Thus if sales were the variable to be forecast, several factors such as GNP, advertising, prices, competition, R&D budget, and time could be tested for their influence on sales by using regression. If it is found that these variables do influence the level of sales, they can be used to predict future values of sales [7].

III.8. FUNDAMENTALS OF TIME-SERIES ANALYSIS

Application of a general class of forecasting methods involves two basic tasks: analysis of the data series and selection of the forecasting model that best fits the data series. Thus in using a smoothing method, analysis of the data series for seasonality aids in selection of a specific smoothing method that can handle the seasonality (or its absence). A similiar sequence of analysis and selection is used in working with decomposition methods and regression methods.

Smoothing methods base their forecasts on the principle of averaging (smoothing) past errors by adding a percentage of the error to a percentage of the previous forecast. Mathematically, single smoothing methods are of the form:

$$F_{t+1} = F_t + \alpha(X_t - F_t) = F_t + \alpha(e_t) \quad (\text{III.25})$$

Time-series decomposition methods are based on the principle of ‘breaking down’ a time series into each of its components of seasonality, trend, cycle, and randomness and then forecasting by predicting each component separately and recombining those predictions. Both smoothing and decomposition methods Express their forecasts as a function of time only.

In general, regression methods attempt to forecast variations in some variable of interest, the dependent variable, on the basis of variations in a number of other factors, the independent variables. In multiple regression, for example, the causal or explanatory model is of the form :

$$Y = b_0 + b_1X_1 + b_2X_2 + \dots + b_kX_k + e \quad (\text{III.26})$$

Autoregressive (AR) models can be effectively coupled with moving average (MA) models to form a very general and useful class of time series models called autoregressive moving average (ARMA) schemes or processes.

III.9. ADVANCED TIME-SERIES METHODS

Autocorrelation

The autocorrelation among successive values of a time series is a key tool in identifying the basic pattern and determining an appropriate model corresponding to a data series. It is the association (mutual correspondence) between two variables and describes what tends to happen to one of the variables if there is a change in the other.

The degree of this correlation is measured by correlation coefficient, which varies between +1 and -1. A value close to +1 implies a strong positive linear relationship between the two variables. This means that when the value of one variable increases, the value of the other tends to increase also. Similarly, a correlation coefficient close to -1 indicates the opposite- that increases in one variable will be associated with decreases in the other. A coefficient of 0 indicates that the two variables are statistically unrelated- that no matter what happens to one variable, nothing can be said about the value of the other [5]

An autocorrelation coefficient is similar to correlation coefficient except that it describes the association (mutual dependence) among values of the same variable but at different time periods. To see what is meant by this, suppose we construct a number of artificial variables from a single variable by changing the time origin in the data.

III.10. THE BOX-JENKINS METHOD

Autoregressive Integrated Moving Average (ARIMA) models have been studied extensively by George Box and Gwilym Jenkins (1976), and their names have frequently been used synonymously with general ARIMA processes applied to time-series analysis, forecasting, and control. Autoregressive (AR) models were first introduced by Yule (1926) and later generalized by Walker (1931), while moving average (MA) models were first used by Slutsky (1937). It was the work of Wold (1938), however, that provided the theoretical foundations of combined ARMA processes. Building on Wold's work, ARMA models have developed in three directions- efficient identification and estimation procedures (for AR, MA, and mixed ARMA processes), extension of the results to include seasonal time series, and the simple extension to include nonstationary processes (ARIMA) [7].

Box and Jenkins (1976) have effectively put together in a comprehensive manner the relevant information required to understand and use univariate time series ARIMA models. The theoretical underpinnings described in Box and Jenkins (1976) are quite sophisticated, but it is possible for the nonspecialist to get a clear understanding of the essence of ARIMA methodology [7].

The first thing to note is that most time series are nonstationary, and the AR and MA aspects of an ARIMA model refer only to a stationary time series. Therefore, it is necessary to have a notational distinction between the original nonstationary time series and its stationary counterpart, after differencing.

The Box-Jenkins Approach

The Box-Jenkins approach to forecasting time series is one of the most popular forecasting methods among academicians. Its popularity is due partly to its generality, partly to its strong theoretical foundations, and partly to its success in empirical comparisons, which have found it to be as accurate as many complex econometric models.

Box and Jenkins' contributions have been their theoretical work and their practical experience in making the best use of the method. Their work also has helped to popularize the field of forecasting and to illustrate how accurate forecasts can be obtained through time series methods.

The Box-Jenkins Methodology For ARIMA Models The Partial Autocorrelation Coefficient

In regression analysis, if the forecast variable Y is regressed on explanatory variables X_1 and X_2 , then it might be of interest to ask how much explanatory power does X_1 have if the effects of X_2 are somehow partialled out first. Typically, this means regression Y on X_2 , getting the residual error from this analysis, and finding the correlation of the residuals with X_1 . In time series analysis there is a similar concept.

Partial autocorrelation are used to measure the degree of association between Y_t and Y_{t-k} , when the effects of other time lags- $1, 2, 3, \dots, k-1$ – are removed.

The partial autocorrelation coefficient of order k is denoted by α_k and can be calculated by regressing Y_t against $Y_{t-1}, \dots, Y_{t-k}$ [6]:

$$Y_t = b_0 + b_1 Y_{t-1} + b_2 Y_{t-2} + \dots + b_k Y_{t-k} \quad (\text{III.27})$$

this is unusual regression because the explanatory variables on the right hand side are previous values of the forecast variable Y_t . These are simply time-lagged values of the forecast variable, and therefore the name autoregression (AR) is used to describe equations of the form of (III.27) [6].

The partial autocorrelation, α_k , is the estimated coefficient b_k from this multiple regression. The first partial autocorrelation is always equal to the first autocorrelation. Varying the number of terms on the right hand side of (III.27) will give the partial autocorrelations for different values of k [6].

III.11. MULTIVARIATE TIME-SERIES ANALYSIS

The increased complexity of the methodology, we will focus on the bivariate case and will only make passing reference to more ambitious cases. The field is

relatively new, but there is considerable interest at the present time and as the computer software for these approaches becomes more available, it is likely that applications will be on the increase.

In 1975, Box and Tiao wrote an important paper on intervention analysis, where they applied the transfer function methodology to economic problems in which the input variable could also be a dichotomous variable [7].

Since multivariate (transfer function) models combine some of the characteristics of the univariate ARIMA models and some of the characteristics of multiple regression analysis, what we are taking about is a method that blends the time-series approach with causal approach.

III.12. STATIONARITY

Stationarity means that there is no growth or decline in the data. The data must be roughly horizontal along the time axis. In other words the data fluctuate around a constant mean, independent of time, and the variance of the fluctuation remains essentially constant over time.

III.12.1. Removing Non-Stationarity In a Time Series

Trends or other non-stationary patterns in the level of a series, result in positive autocorrelations that dominate the autocorrelation diagram. Therefore it is important to remove the non-stationarity, so other correlation structure can be seen before proceeding with time series model building. One way of removing non-stationarity is through the method of differencing. We define the differenced series as the change between each observation in the original series:

$$Y_t^1 = Y_t - Y_{t-1}$$

The differenced series will have only n-1 values since it is not possible to calculate a difference Y_1 for the first observation [6].

III.12.2. A Random Walk Model

$$Y_t = Y_{t-1} + e_t \tag{III.28}$$

This model is very widely used for non-stationary data and is known as a ‘random walk’ model. Random walks typically have long periods of apparent trends up or down which can suddenly change direction unpredictably [6].

CHAPTER IV

FORECASTING METHODS

The aim of this study is to constitute the equations for forecasting electricity consumption by using an artificial neural networks and classical time series analysis techniques for determining the future energy level of Turkey and comparing the methods.

Appendix-1 presents a list of all variables used in this study. These are the variables for the period between 1970 and 2008 were obtained from Turkish Electricity Transmission Corporation (TEIAS), Turkish Statistical Institute (TUIK).

The variables are consisting of population, installed capacity, electricity consumption, gross national product, electricity imports, electricity exports, mid-annual exchange rates, electricity generation, industry sector electricity consumption, average electricity sale price, total electricity sale and service price, total electricity sale, gross domestic product, consumer price index, real gross domestic product growth..

IV.1. ARTIFICIAL NEURAL NETWORK MODELS

Appendix-1 presents a list of all variables used in this study. These are the variables for the period between 1970 and 2008 were obtained from TEIAS (Turkish Electricity Transmission Corporation), TUIK (Turkish Statistical Institute). Matlab was used for all neural network simulations.

Electricity consumption is modeled as a function of economic indicators such as population, installed capacity, electricity imports, electricity exports, mid-annual exchange rates, electricity generation, industry sector electricity consumption, average electricity sale prices, total electricity sale and service price, total electricity sale, gross domestic product, consumer price index, real GDP growth and electricity consumption.

In order to obtain electricity consumption forecast, four different artificial neural network (ANN) models are developed Multilayer Perceptron (MLP), Elman Network, Radial Basis Function (RBF) and Nonlinear Autoregressive with Exogenous Inputs models were used in the neural network.

While related data series for 1970-2005 period are used for training of the network and the validation purpose and data for 2005-2008 period are used for the testing in MLP, Elman network and RBF models.

Population, installed capacity, electricity imports, electricity exports, mid-annual exchange rates, electricity generation, industry sector electricity consumption, average electricity sale prices, total electricity sale and service price, total electricity sale, gross domestic product, consumer price index, real GDP growth are used in the input layer of the network. The electricity consumption is in the output layer. This input-output network models are a result of relationships that exist among electricity consumption and several other variables.

In NARX models electricity consumption data are used as input data .

We followed these methodology in ANN :

- To collect the data
- To determine the network structure (how many inputs, outputs, etc.)
- To determine the learning parameters (activation function, etc.)
- To generate initial weight values. (initially they are random)
- To choose samples in the learning set and load them to the network.
- To make estimations in the learning.
- To compare the realized outputs with expected outputs.
- To change the weights [4].

IV.1.1. Multilayer Perceptron Models (MLP)

Multilayer perceptron (MLP) training algorithm is used as the neural network topology. Tangent-sigmoid (tansig) and logistic-sigmoid (logsig) transfer functions are selected in the hidden and output layer processing elements. Performance comparison among different models is made based on sum and mean square error.

The algortihm is as follows:

- 1) Determination of input(s) and output variables of the model.
- 2) Divide data set into three subsets: training, test and validation data.
- 3) Develop the conventional regression model fort he training data set. Then, forecast electricity consumption fort he test period.
- 4) Use the ANN method to estimate the relation between input(s) and output(s).

There are several issues involved in designing and training a multilayer perceptron network:

- selecting how many hidden layers to use in the network
- deciding how many neurons to use in each hidden layer
- finding a globally optimal solution that avoids local minima
- converging to an optimal solution in a reasonable period of time
- validating the neural network to test for overfitting.

First data set is divided into three into three subsets: 10% of validation data set, 30% of training data set, 60% of test data set.

For improved performance all variables are normalized into a smaller range through a linear transform. Each input is multiplied by a connection weight. Products and biases are summed, then transformed through a transfer function to generate a result, and finally the output obtained.

40 multilayer perceptron networks are developed with changing the layer numbers, the hidden layer numbers, the activation functions and the performance criteria parameters. The results are shown in Appendix-2 Training correlation coefficient (R), validation R, test R, real mean square error (MSE) and real R (correlation coefficient) values of 40 MLP networks are given in to the table. Among 40 MLP networks two of them (MLP-14 and MLP-21) are chosen which were resulted in close values to real values.

IV.1.1.1. First MLP Model (MLP-14)

The selected multilayer perceptron (MLP) ANN structure is shown in Figure IV.1 MLP-14 network is consisting of an input layer, 15 process elements in hidden layer and a single output layer with MSE (mean squared error) performance criteria and logsig (logistic-sigmoid) activation function. The initial values are generated randomly.

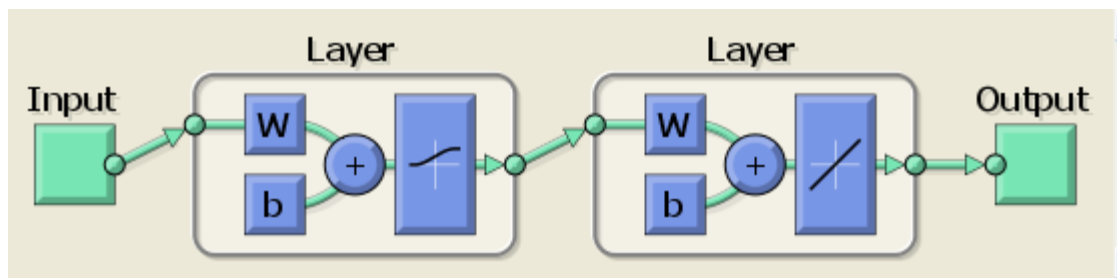


Figure IV.1 MLP-14 model architecture

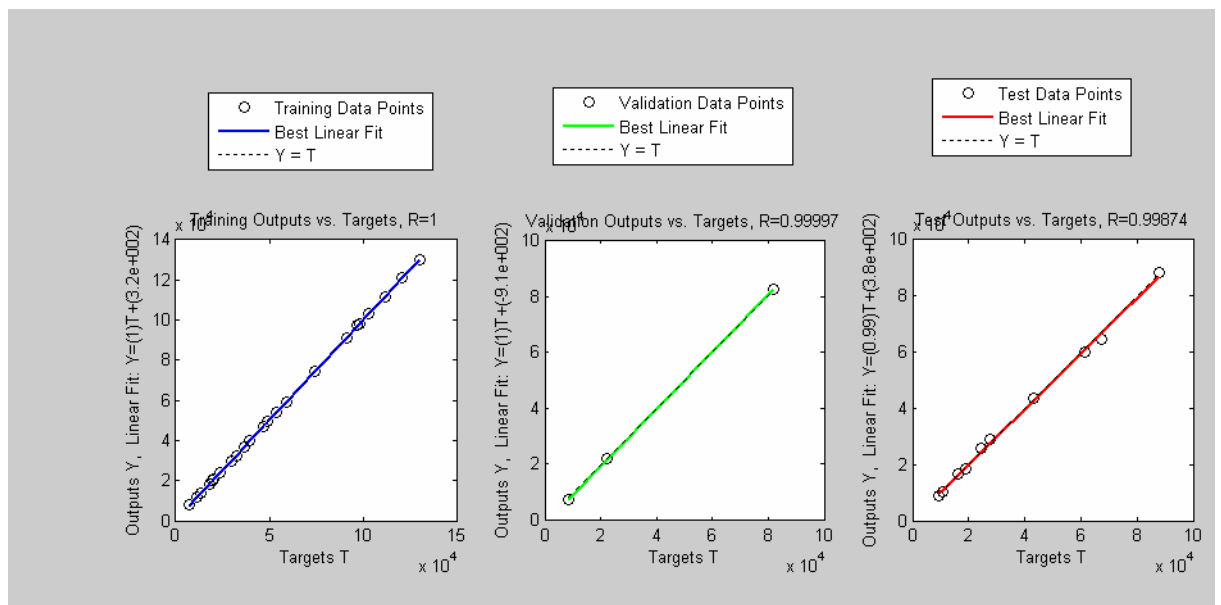


Figure IV.2 MLP-14 model training, validation, target correlation coefficient slope

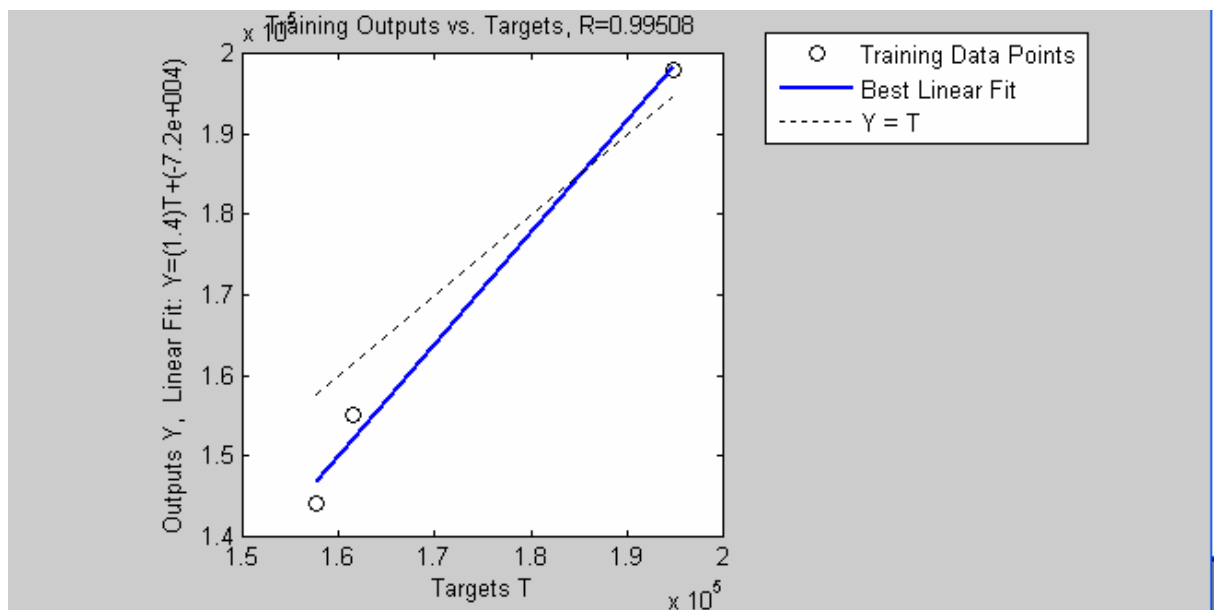


Figure IV.3 MLP-14 model training output correlation coefficient slope

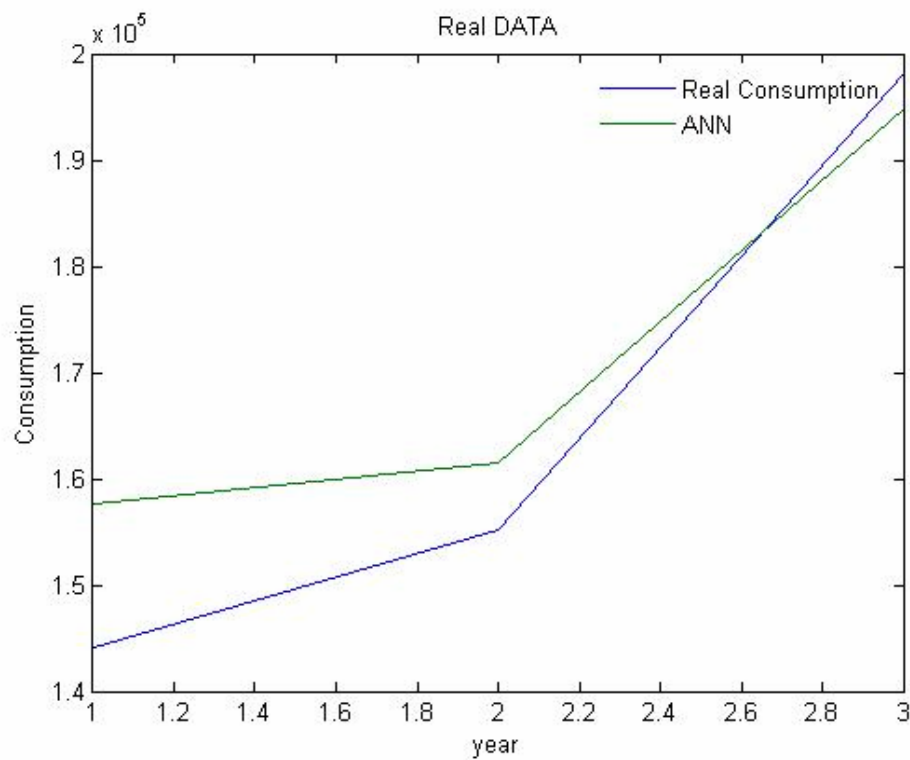


Figure IV.4 Comparison with real consumption and ANN consumption

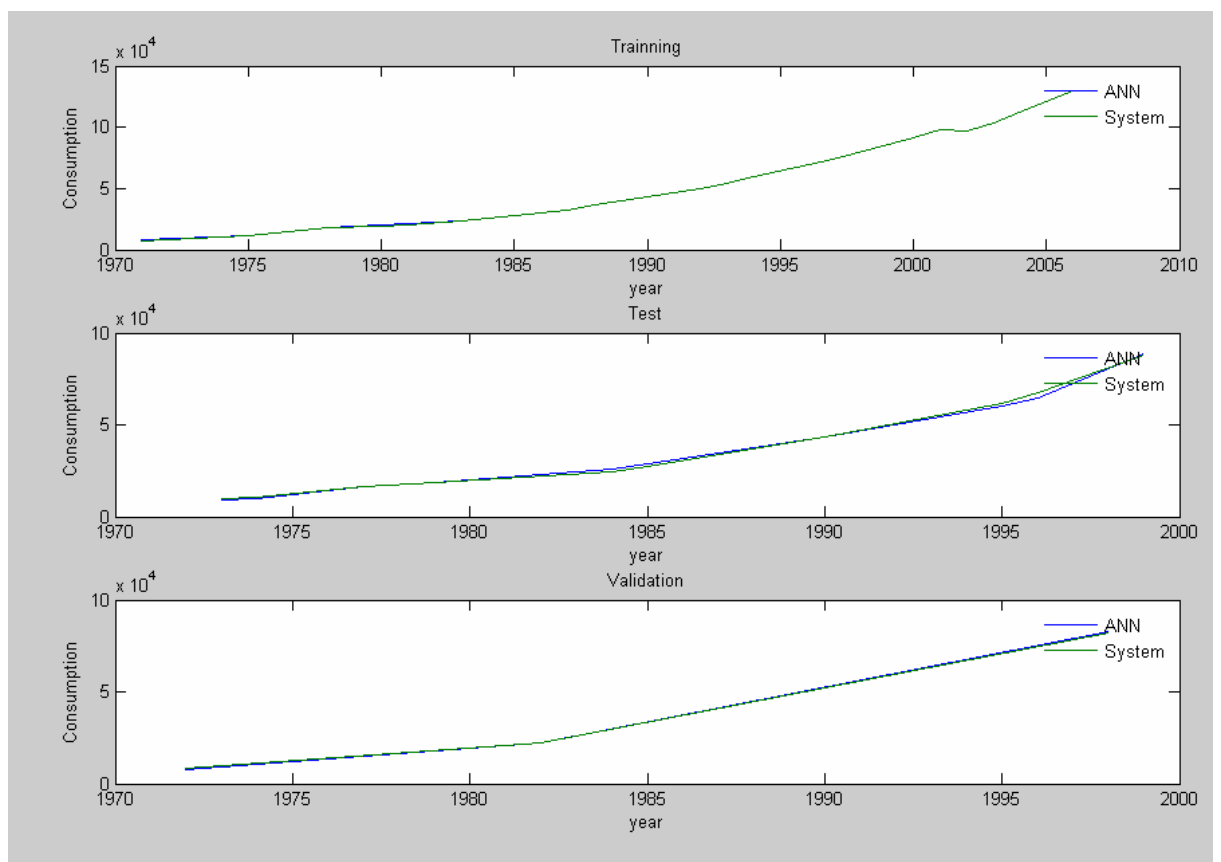


Figure IV.5 Training, test and validation forecasts

As we seen in figures real MSE value is equal to 0,107082 and real correlation coefficient (R) value is equal to 0,995079. Linear slope equation for Figure IV.3 is $Y=(1.4)T+(-7.2e+004)$. This is the optimum result of single layer MLP ANN in 40 MLP models. The Figure IV.4 is shown the comparison between the real consumption and the consumption which developed from MLP model. The result is optimum result of single layer MLP model. Figure IV.5 shows the MLP-14 model's training, test, and validation forecasts.

IV.1.1.2. Second MLP Model(MLP-21)

The selected multilayer perceptron (MLP) ANN structure is shown in Figure IV.6 (and is shown in Appendix-7). MLP-21 network is consisting an input layer, 3 proces elements in each of two hidden layers and an output layer with MSE (mean squared error) performance criteria and tansig (Tangent-sigmoid) activation function. The initial values are generated randomly.

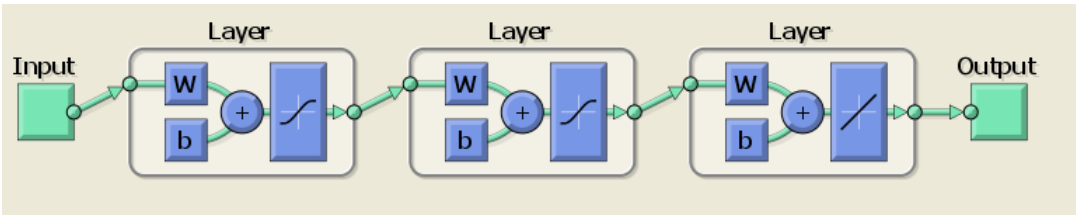


Figure IV.6 MLP-21 model architecture

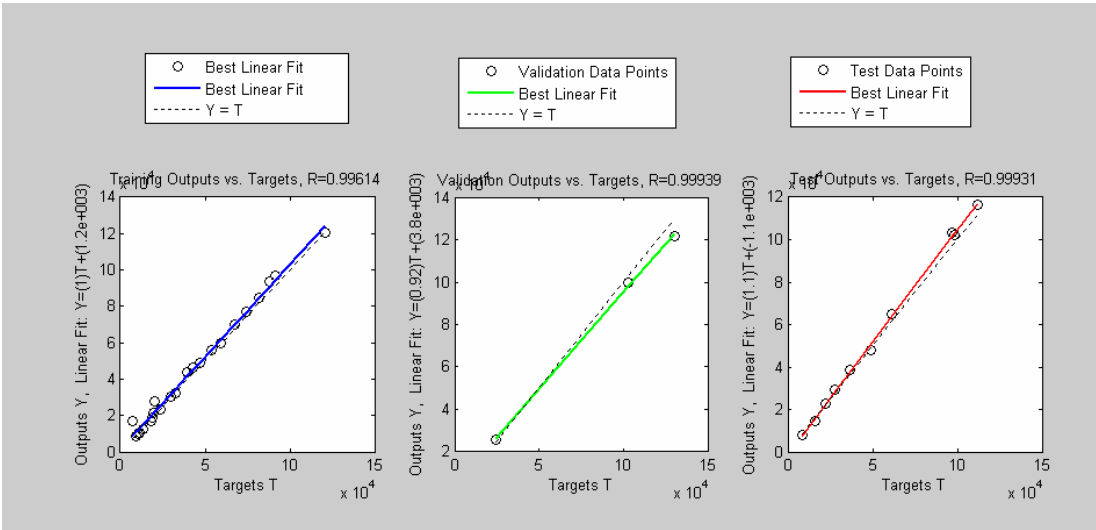


Figure IV.7 MLP-21 model training, validation, target correlation coefficient slope

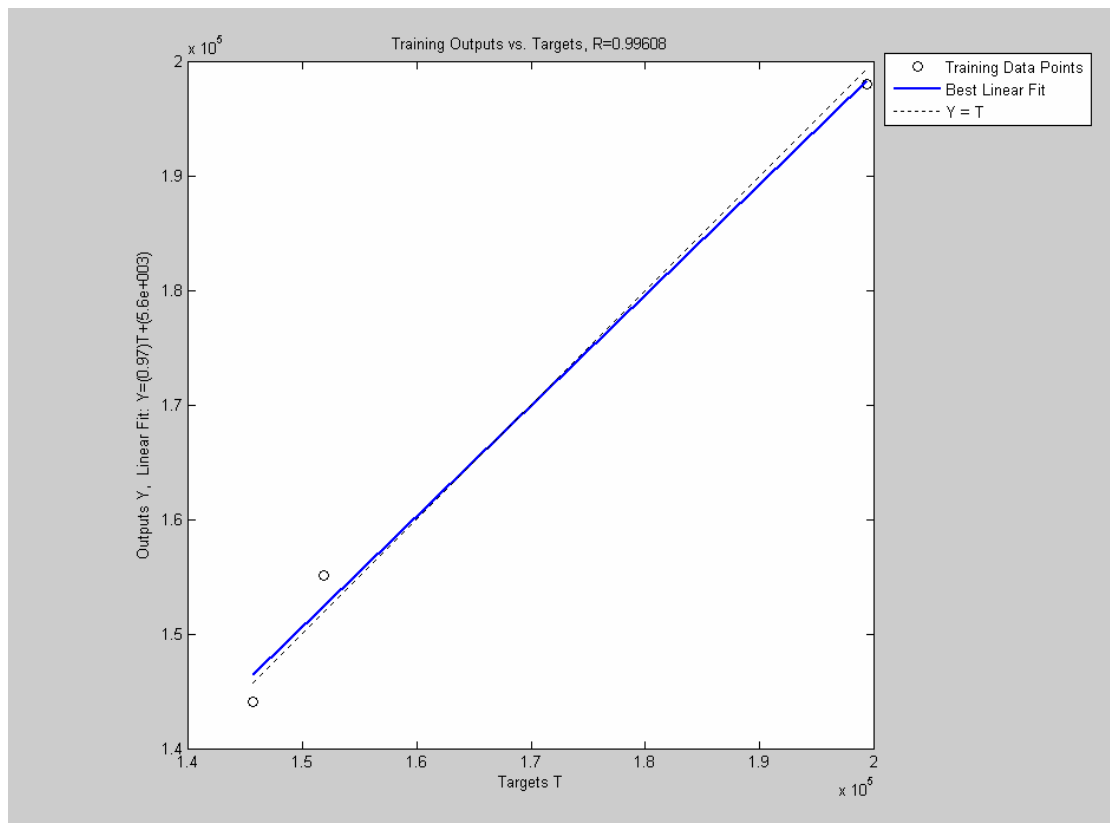


Figure IV.8 MLP-21 model training output correlation coefficient slope

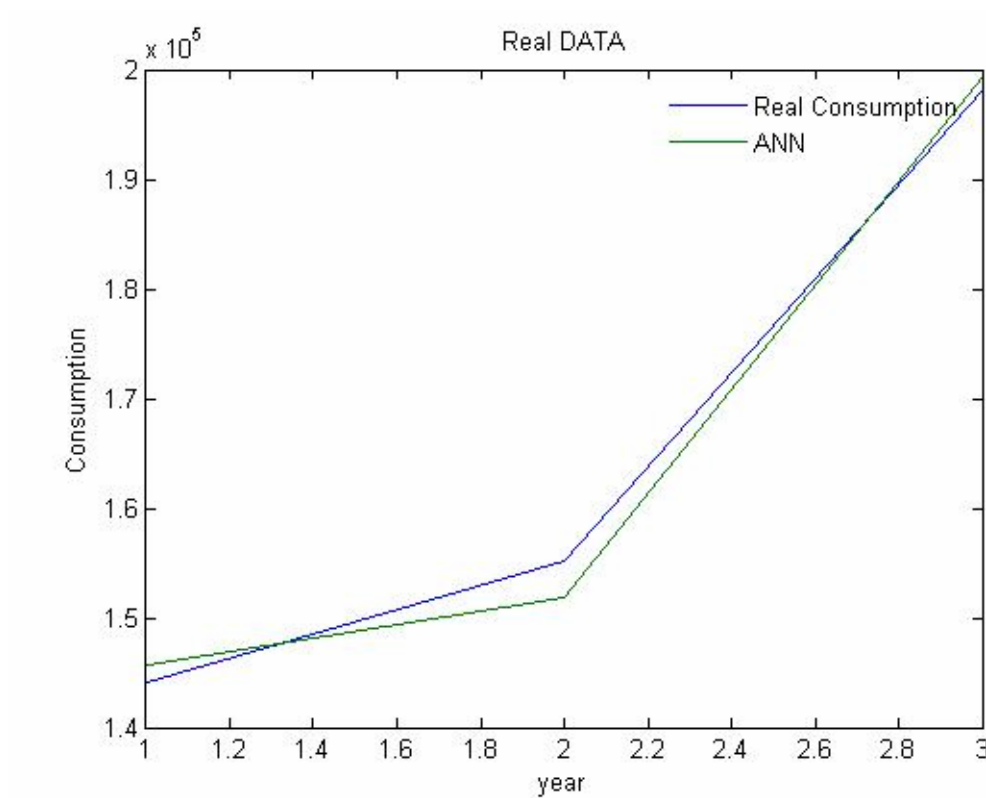


Figure IV.9 Comparison with real consumption and ANN consumption

As we seen in figures real MSE value is equal to 0,006727 and real correlation coefficient (R) value is equal to 0,996076. Linear slope equation for Figure IV.8 is $Y=(0.97)T+(5.6e+003)$. This is the optimum result of two layered MLP ANN in 40 MLP models. The Figure IV.9 is shown the comparison between the real consumption and the consumption which developed from MLP model. The result is optimum result of single layer MLP model. Figure IV.10 shows the MLP-21 model's training, test, and validation forecasts.

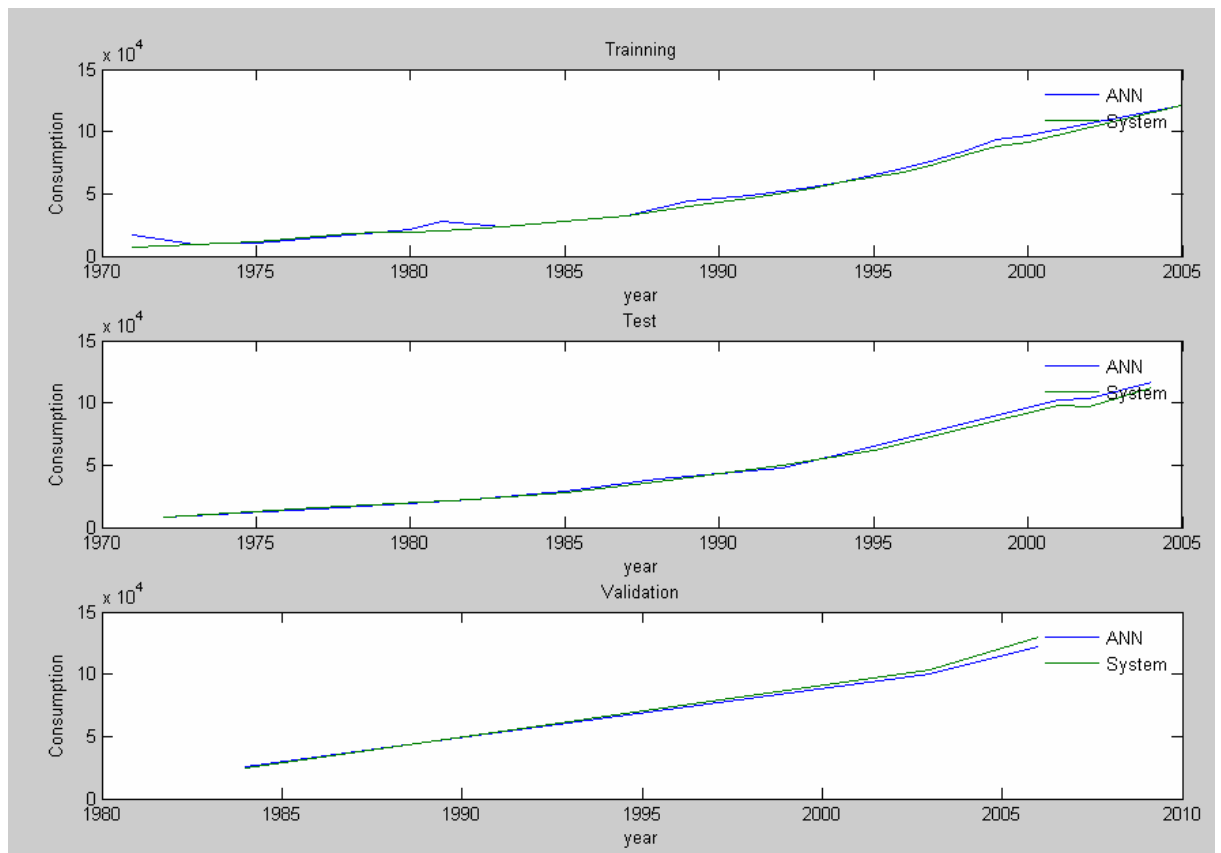


Figure IV.10 Training, test and validation forecast

IV.1.2. ELMAN Network Models

The algorithm is as follows:

- Determination of input(s) and output variables of the model.
- Divide data set into three subsets: training, test and validation data.
- Develop the conventional regression model for the training data set. Then, forecast electricity consumption for the test period.

- Use the ANN method to estimate the relation between input(s) and output(s).

First data set is divided into three into three subsets: 10% of validation data set, 30% of training data set, 60% of test data set.

For improved performance all variables are normalized into a smaller range through a linear transform. Each input is multiplied by a connection weight. Products and biases are summed, then transformed through a transfer function to generate a result, and finally the output obtained.

40 multilayer perceptron networks are developed with changing the layer numbers, the hidden layer numbers, the activation functions and the performance criteria parameters. The results are shown in Appendix-3. Training correlation coefficient (R), validation R, test R, real mean square error (MSE) and real R (correlation coefficient) values of 40 Elman networks are given in to the table. Among 40 Elman networks two of them (ELMAN-1 and ELMAN-34) are chosen which were resulted in close values to real values.

IV.1.2.1. First Elman model (ELMAN-1)

The selected Elman network structure is shown in Figure IV.11 Elman-1 network is consisting of an input layer, 3 process elements in hidden layer and an output layer with MSE (mean squared error) performance criteria and tansig (Tangent-sigmoid activation function. The initial values are generated randomly.

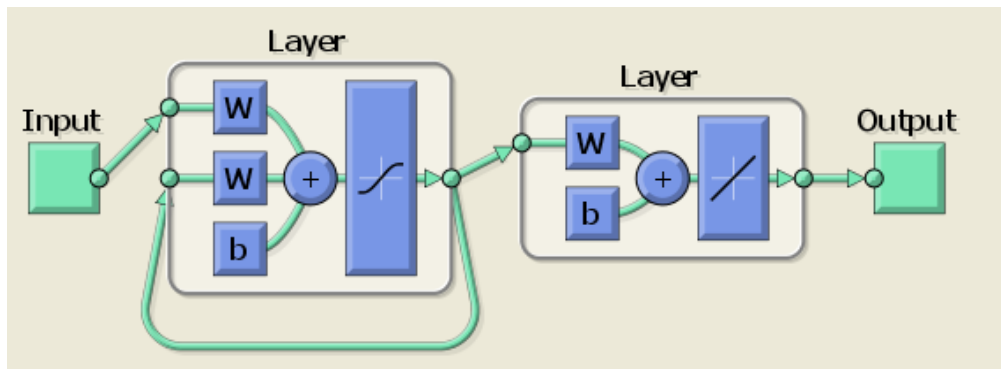


Figure IV.11 ELMAN-1 model architecture

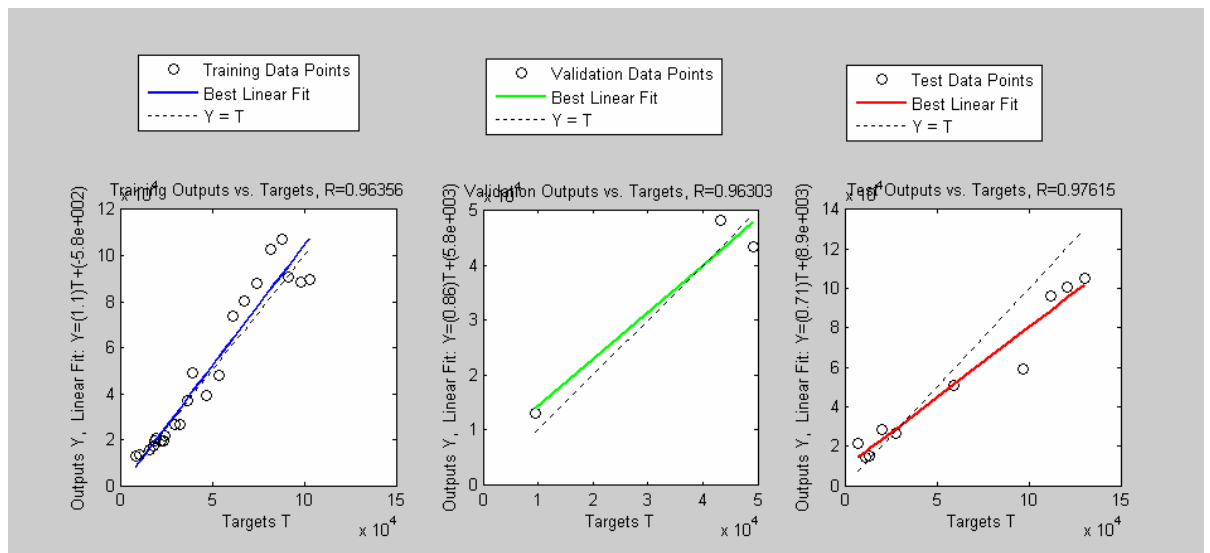


Figure IV.12 ELMAN-1 model training, validation, target correlation coefficient slope

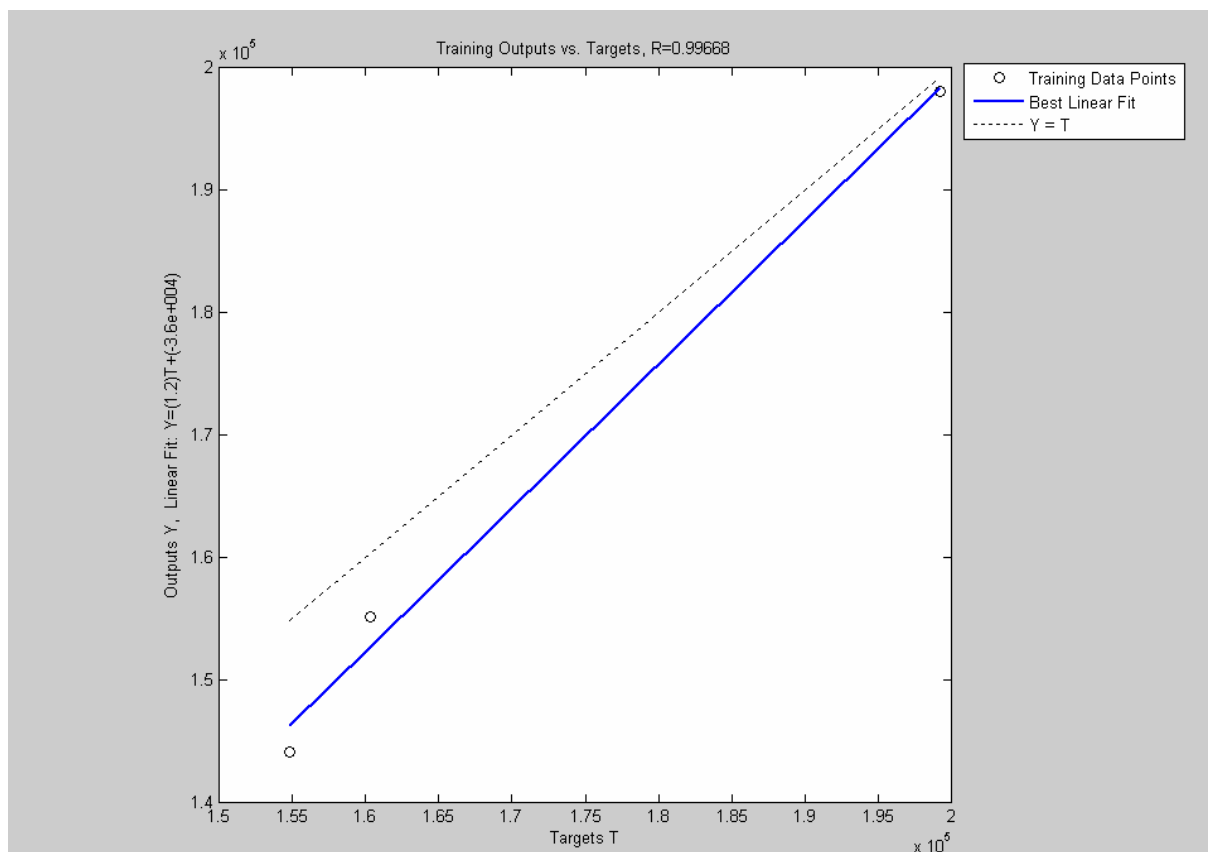


Figure IV.13 ELMAN-1 model training output correlation coefficient slope

As we seen in figures real MSE value is equal to 0,06598 and real correlation coefficient (R) value is equal to 0,996676. Linear slope equation for Figure IV.13 is $Y=(1.2)T+(-3.6e+004)$. This is the optimum result of single layered Elman model in 40 Elman models. The Figure IV.14 is shown the comparison between the real consumption and the consumption which developed from Elman model. The result is optimum result of single layer Elman model.

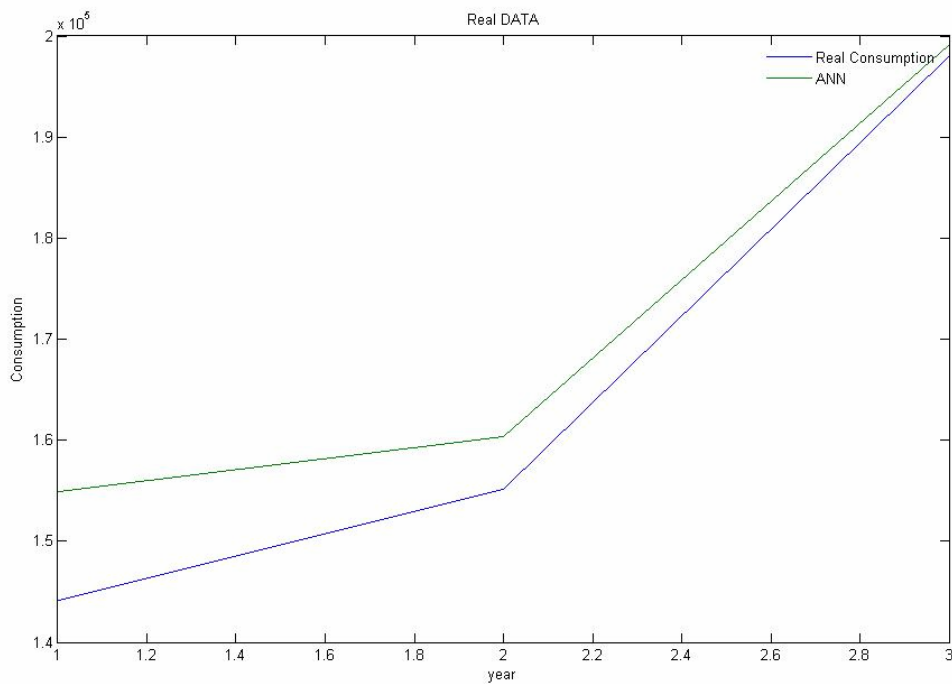


Figure IV.14 Comparison with real consumption and ANN consumption

Figure IV.15 shows the ELMAN-1 model's training, test, and validation forecasts.

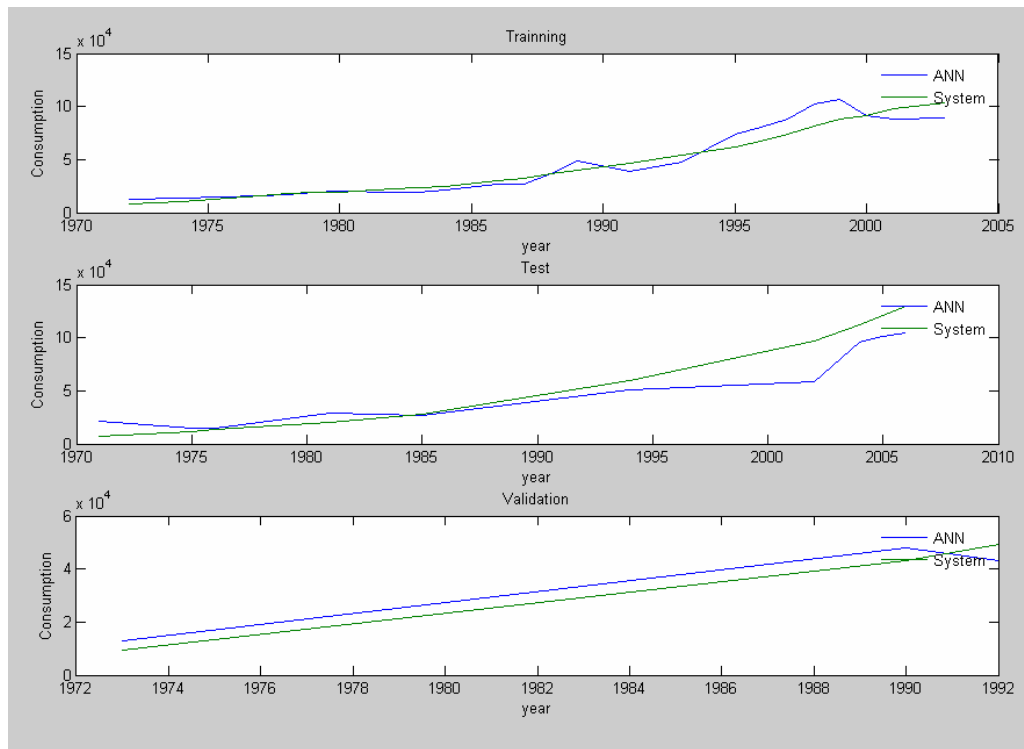


Figure IV.15 Training, test and validation forecasts

IV.1.2.2. Second Elman model (ELMAN-34)

The selected Elman network structure is shown in Figure IV.16 Elman-34 network is consisting an input layer, 15 process elements in each of two hidden layers and an output layer with MSE (mean squared error) performance criteria and logsig (logistic-sigmoid) activation function. The initial values are generated randomly.

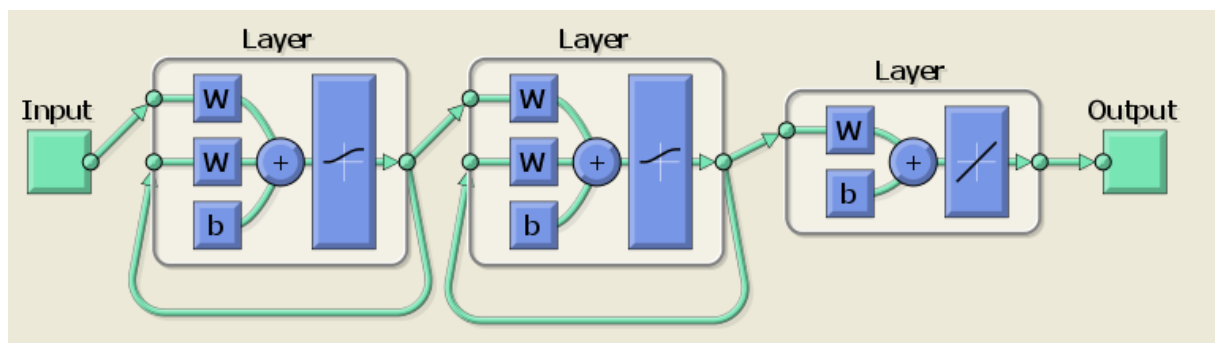


Figure IV.16 ELMAN-34 model architecture

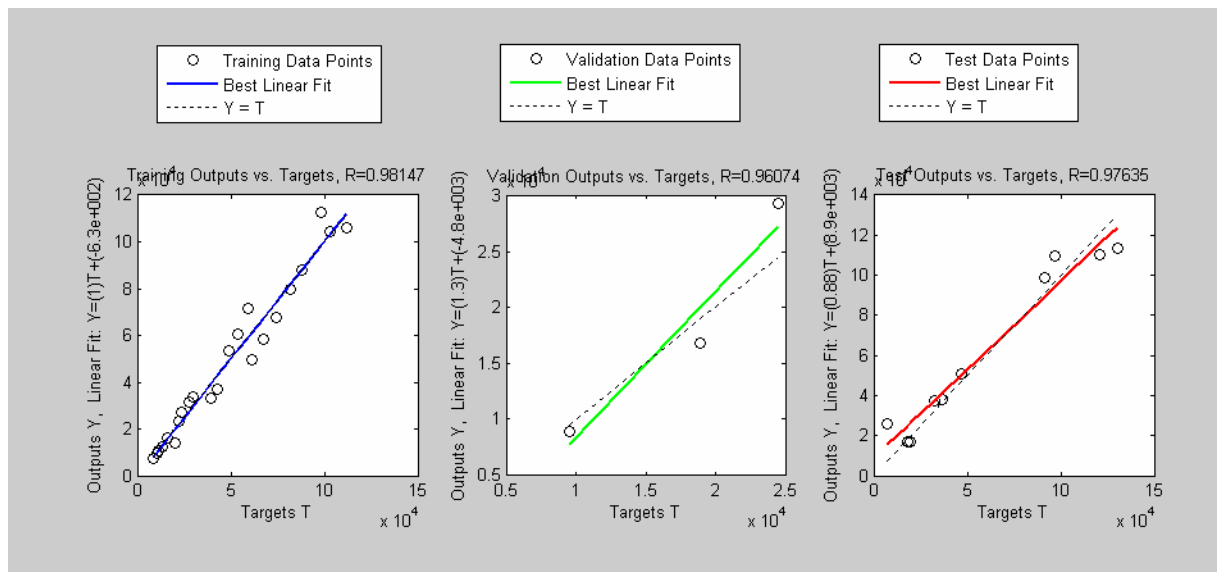


Figure IV.17 ELMAN-34 model training, validation, target correlation coefficient slope

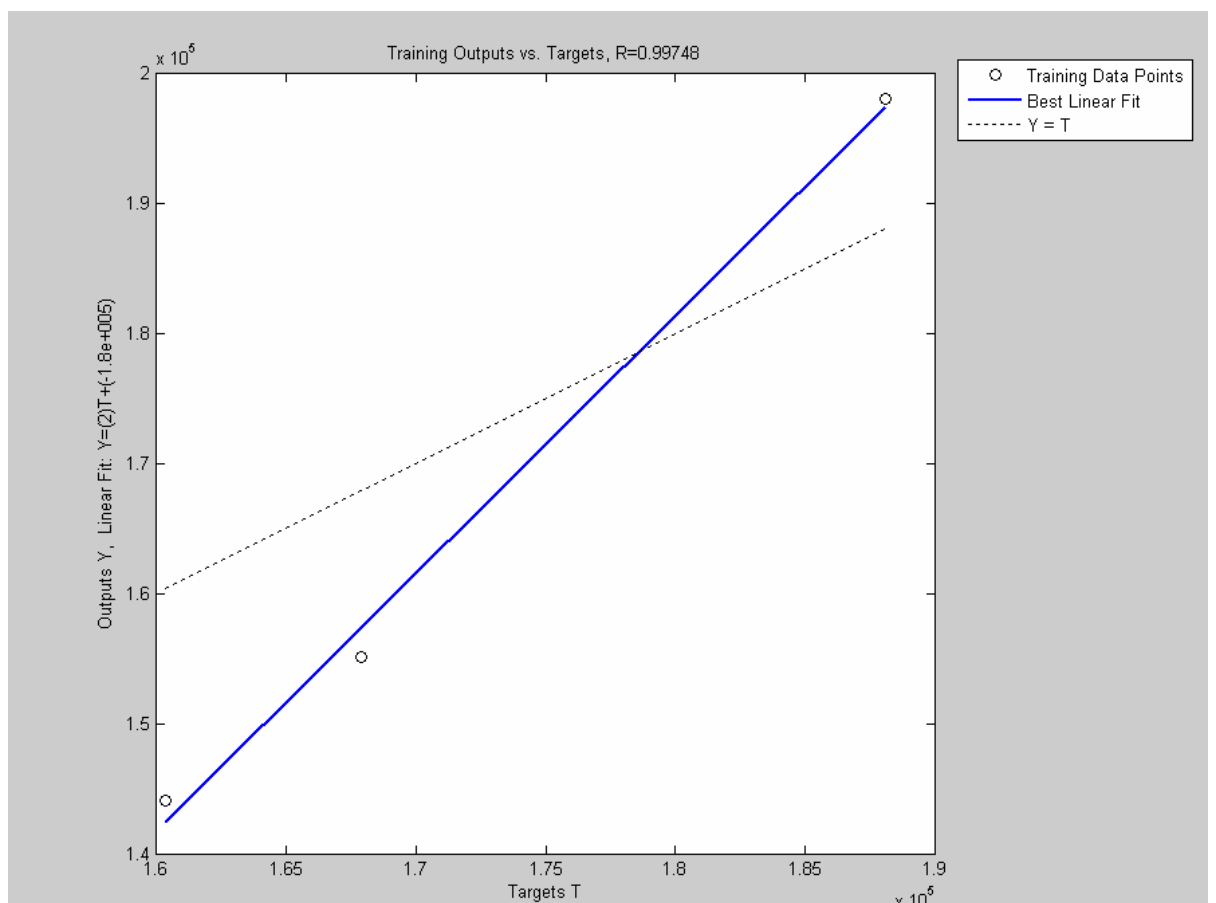


Figure IV.18 ELMAN-34 model training output correlation coefficient slope

As we seen in figures real MSE value is equal to 0,241451 and real correlation coefficient (R) value is equal to 0,997484. Linear slope equation for Figure IV.18 is $Y=(2T+(-1.8e+005))$. This is the optimum result of two layered Elman model in 40 Elman models. The Figure IV.19 is shown the comparison between the real consumption and the consumption which developed from Elman model. The result is optimum result of single layer Elman model.

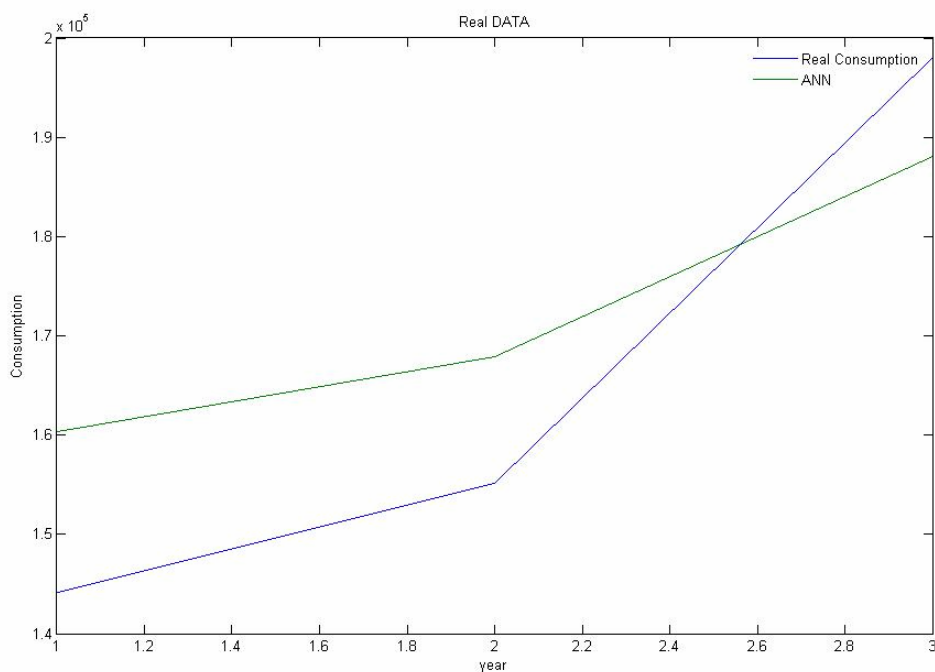


Figure IV.19 Comparison with real consumption and ANN consumption

Figure IV.20 shows the ELMAN-34 model's training, test, and validation forecasts

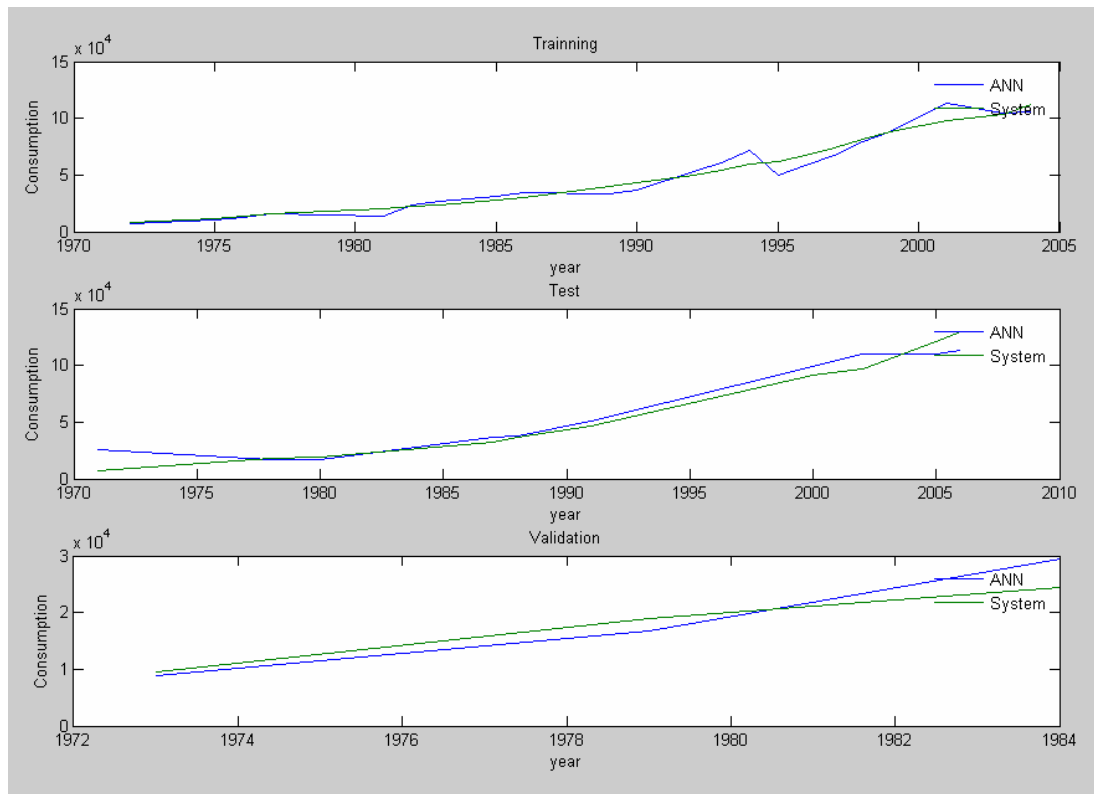


Figure IV.20 Training, test and validation forecasts

IV.1.3. Radial Basis Function Model (RBF)

First data set is divided into three subsets: 10% of validation data set, 30% of training data set, 60% of test data set.

For improved performance all variables are normalized into a smaller range through a linear transform. Each input is multiplied by a connection weight. Products and biases are summed, then transformed through a transfer function to generate a result, and finally the output obtained.

The Radial Basis Function (RBF) network structure is shown in Figure IV.21 RBF network is consisting of an input layer, a hidden layer and an output layer with MSE performance criteria. The initial values are generated randomly.

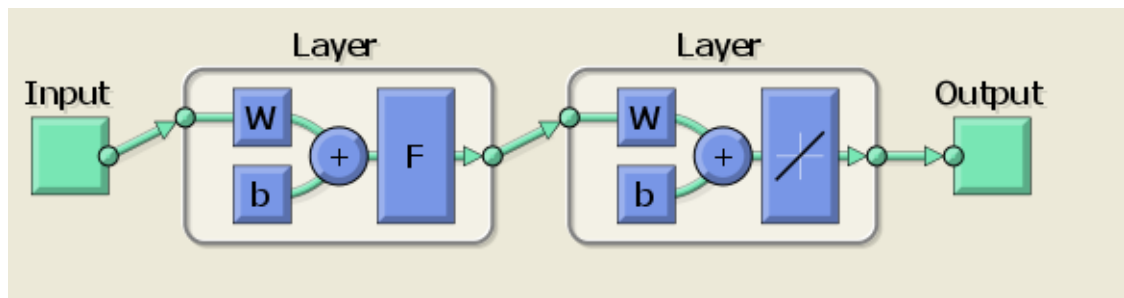


Figure IV.21 RBF model architecture

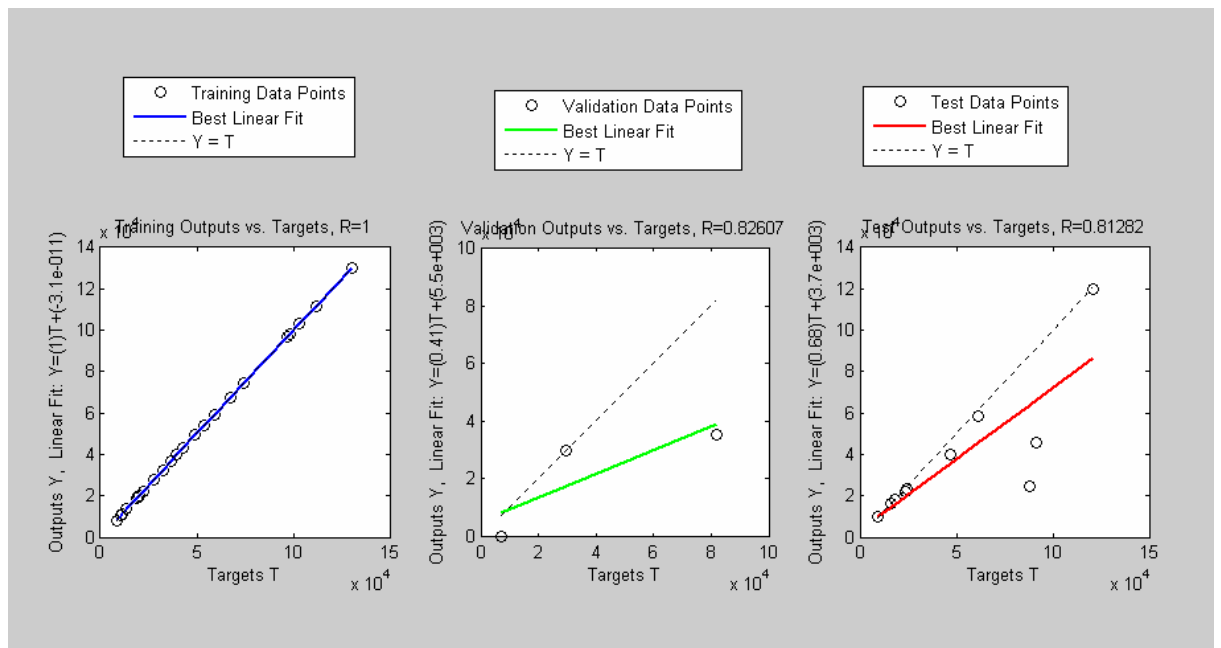


Figure IV.22 RBF model training, validation, target regression slope

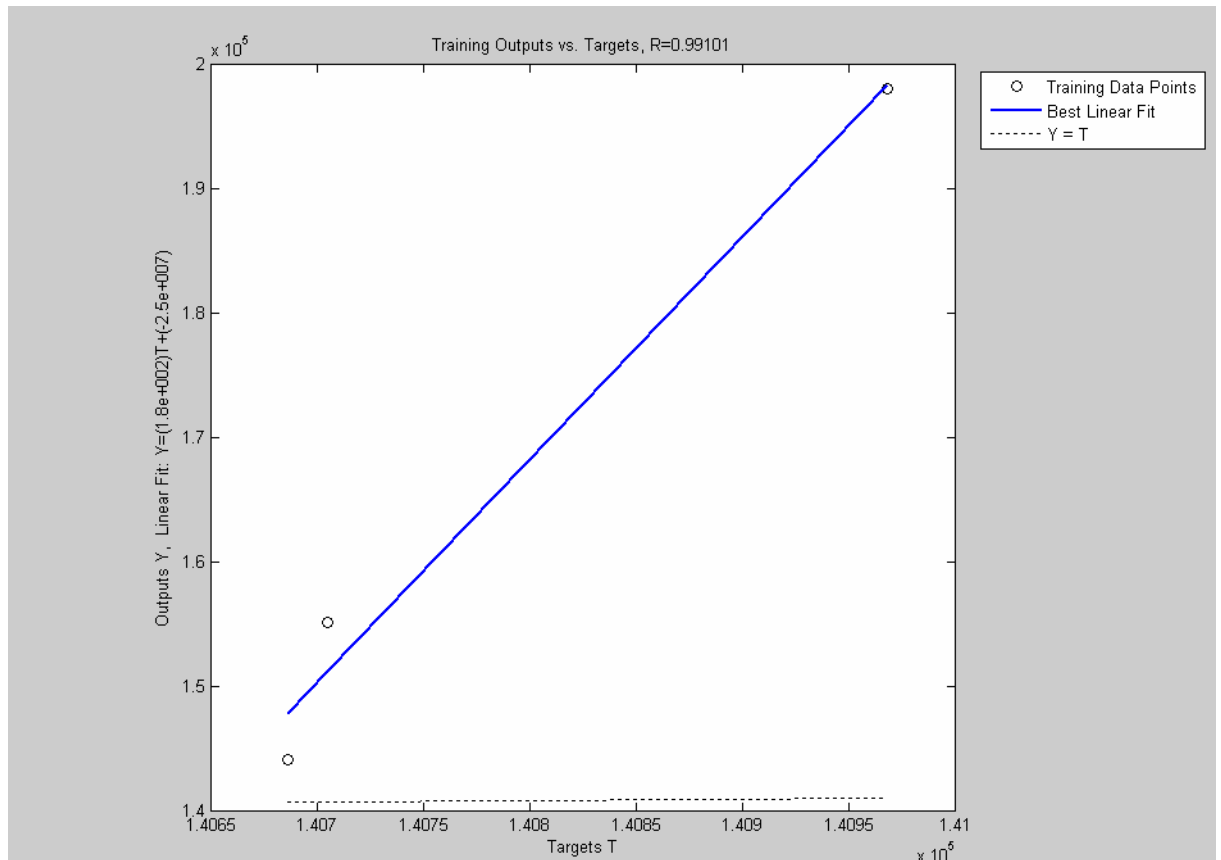


Figure IV.23 RBF model training output regression slope

As we seen in figures real MSE value is equal to 1,592593 and real correlation coefficient (R) value is equal to 0,991012. Linear slope equation for Figure IV.23 is $Y=(1.8e+002)T+(-2.5e+007)$. This is the optimum result of RBF model. The Figure IV.24 is shown the comparison between the real consumption and the consumption which developed from RBF model. Figure IV.25 shows the RBF model's training, test, and validation forecasts

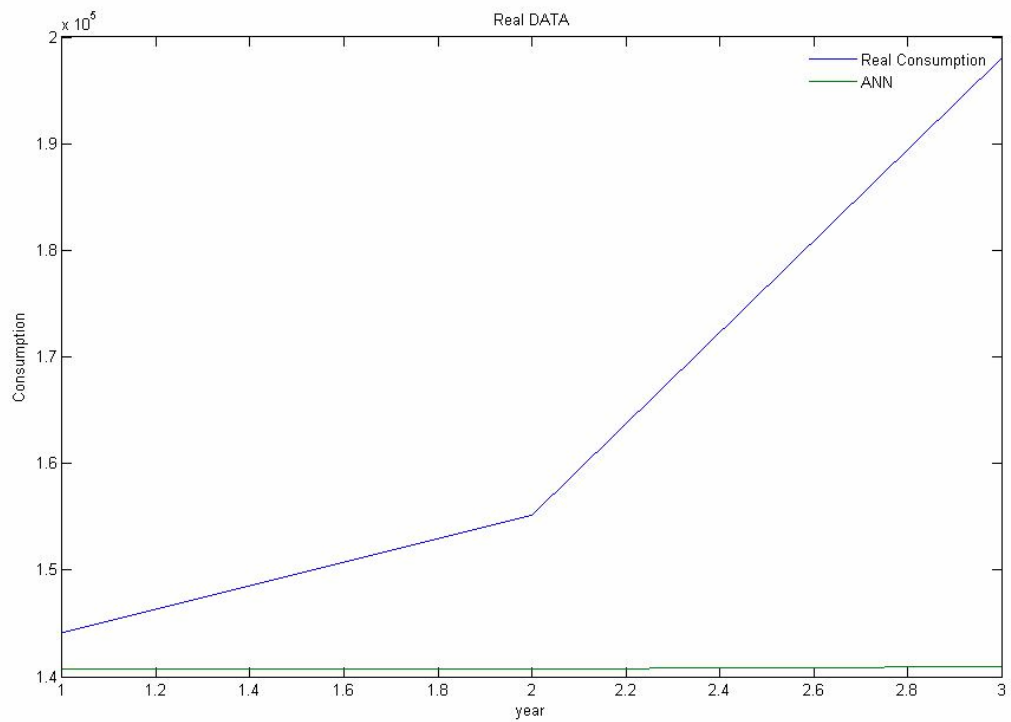


Figure IV.24 Comparison with real consumption and ANN consumption

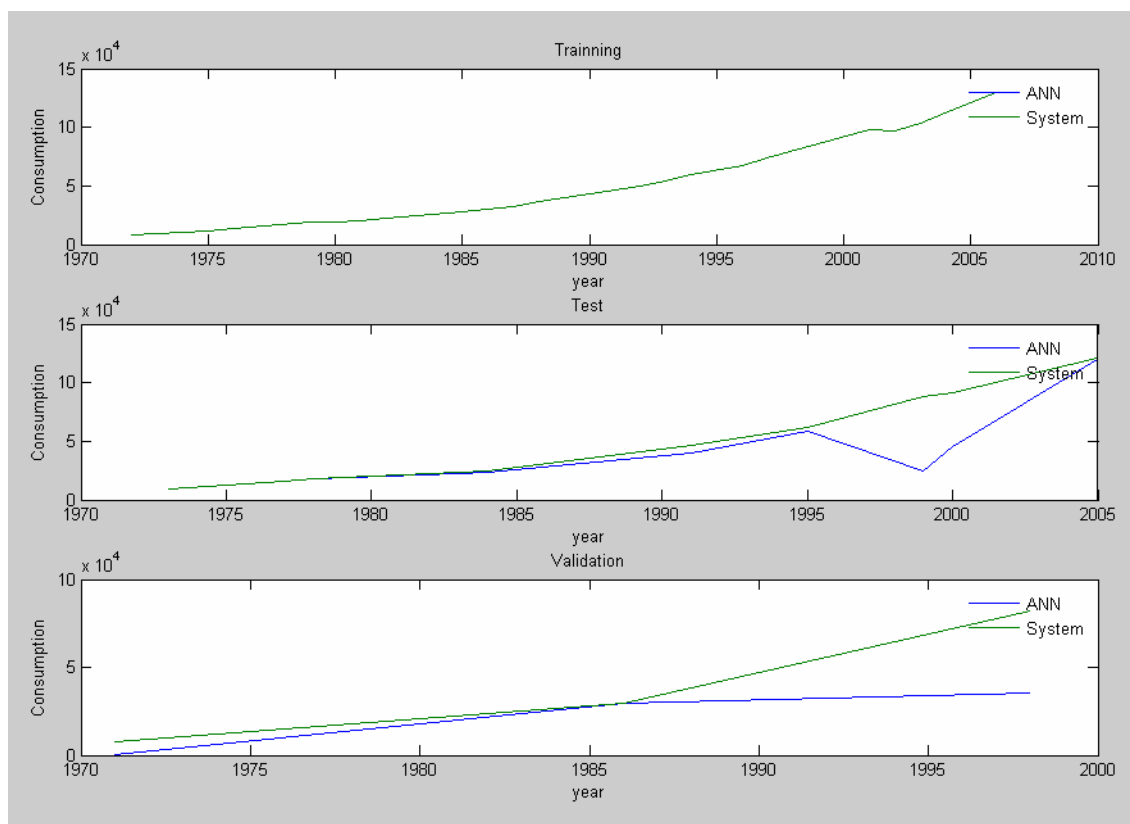


Figure IV.25 Training, test and validation forecasts

IV.1.4. NARX Models

The aim of NARX (nonlinear autoregressive network with exogenous inputs) models are to evaluate, in qualitative and quantitative terms, the predictive ability of the NARX networks electricity consumption data set.

The goal in the experiments is to evaluate if the output regressor in the input layer of the NARX network improves its predictive performance.

Once given networks have been trained, they are required to provide estimates of the future sample values of a given time series for a certain prediction horizon.

While related data series for 1970-2005 period are used for training of the models, and data for 1999-2008 are used for testing of the models. Data for 2006-2015 are estimated by NARX models.

For improved performance all variables are normalized into a smaller range through a linear transform.

10 NARX models are developed with changing the hidden layer numbers. The results shown in Appendix-5. Among 10 NARX networks two of them (NARX-2 and NARX-9) are chosen which were resulted in close values to real values.

IV.1.4.1. First NARX Model (NARX-2)

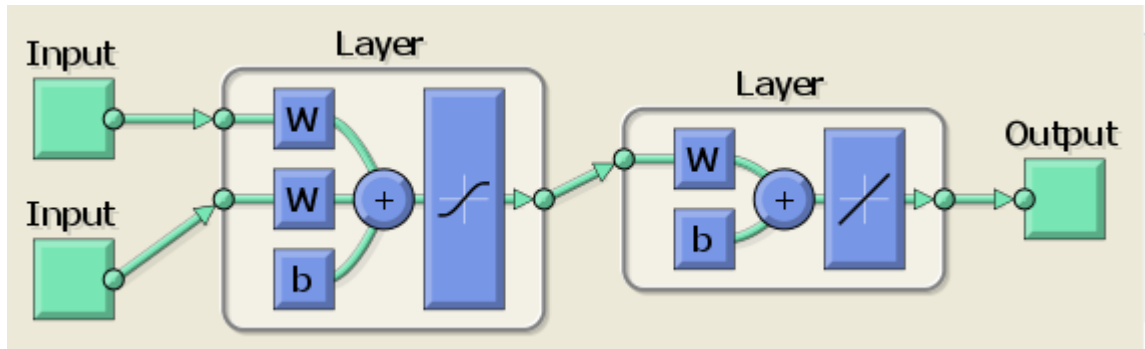


Figure IV.26 NARX-2 model architecture

The selected NARX network structure is shown in Figure IV.26. To facilitate the performance comparison, network has an input layer, 5 process elements in hidden layers and an output layer with sse (sum squared error) performance criteria. All neurons in hidden layer use tansig (tangent sigmoid) activation function and all

neurons in output layer use purelin activation function. The initial values are generated randomly.

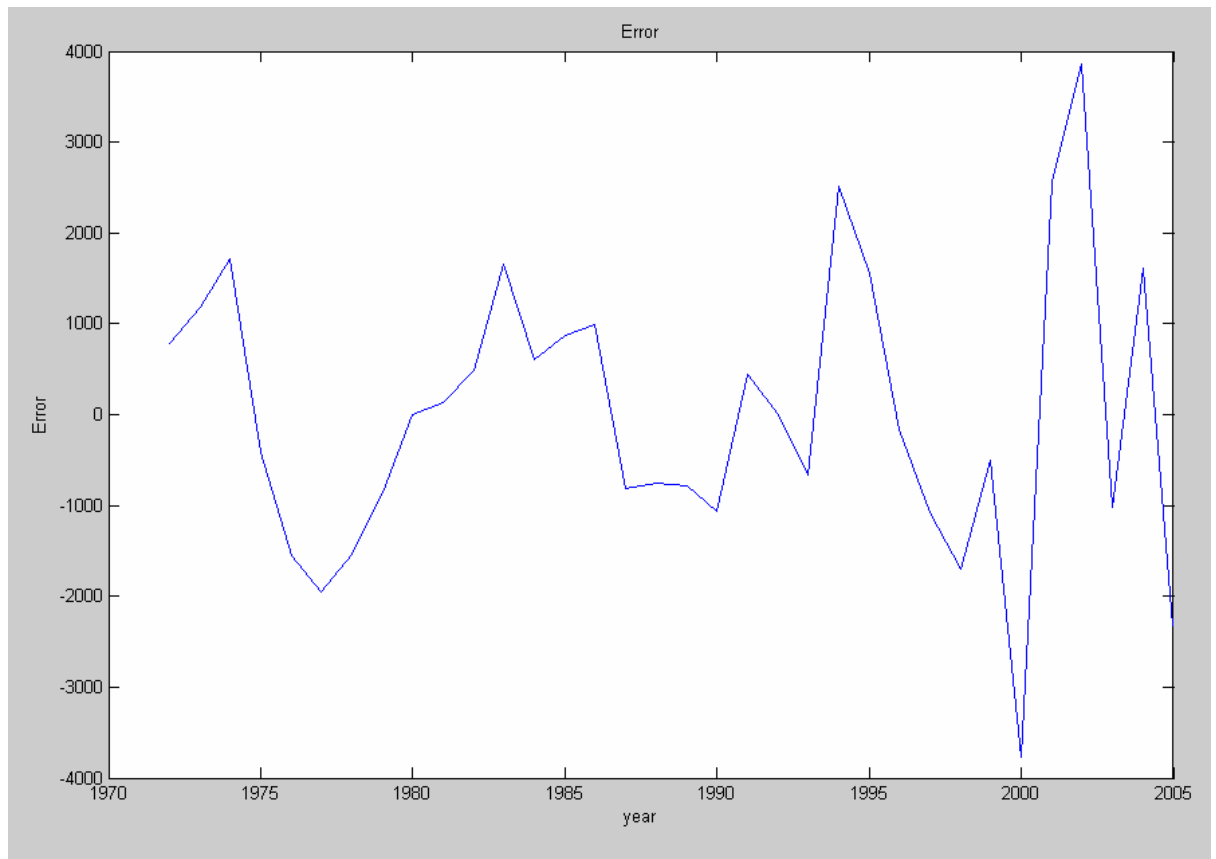


Figure IV.27 NARX-2 error slope

As we seen in Figure IV.28 and Figure IV.30 train correlation coefficient value is equal to 0,99905 and narx correlation coefficient value is equal to 0,98921. This is the optimum result of NARX model in 10 NARX models. The Figure IV.29 is shown the NARX prediction slope from 1970 to 2005 . Figure IV.31 is shown the predicton of assumption from 2008 to 2015.

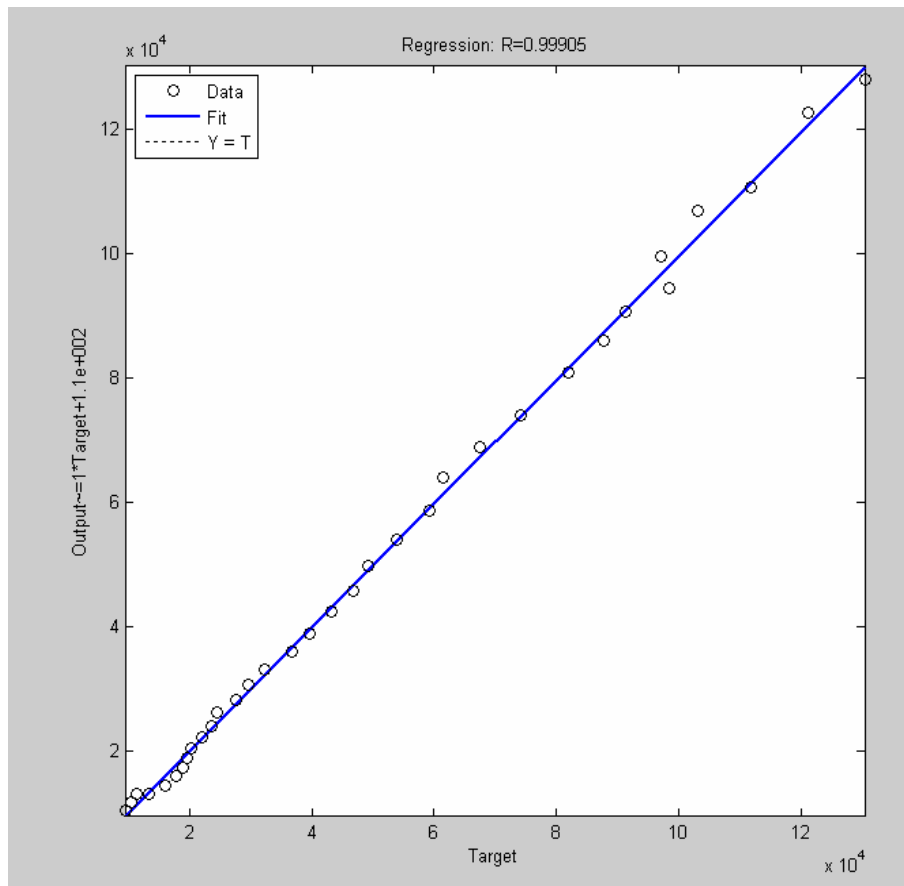


Figure IV.28 NARX-2 correlation coefficient slope

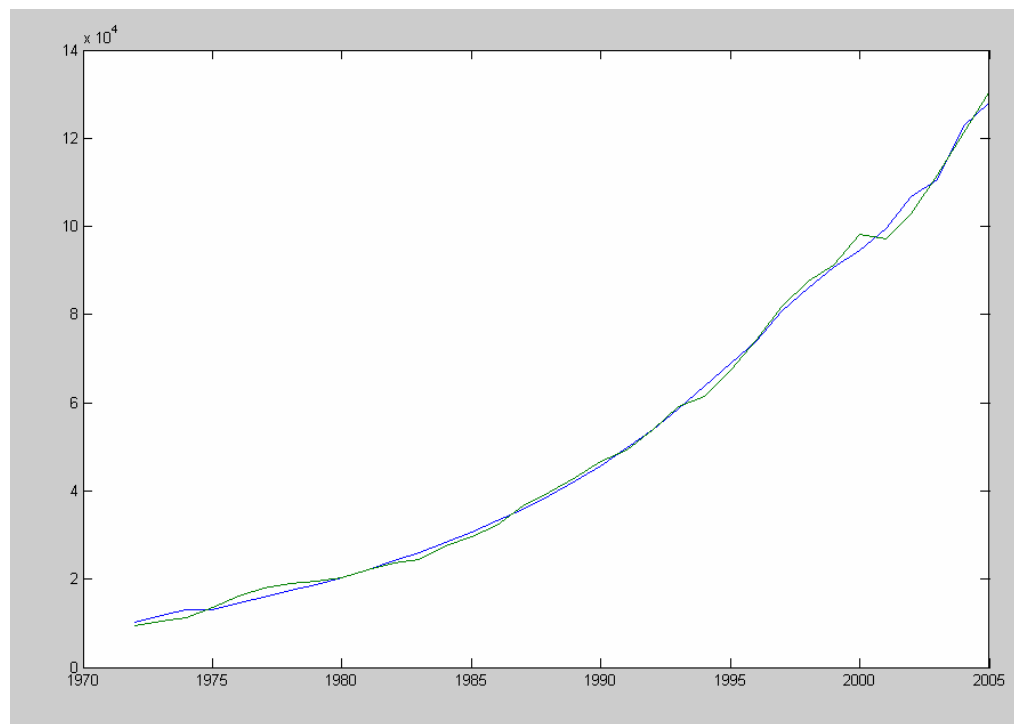


Figure IV.29 NARX-2 forecast between 1970-2005

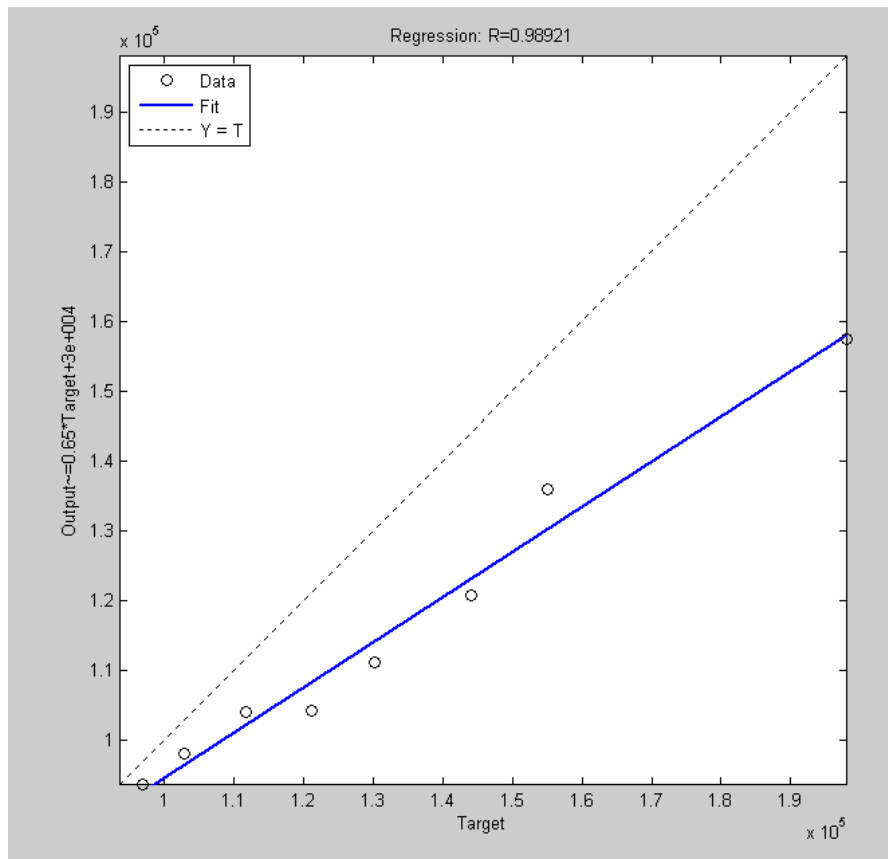


Figure IV.30 NARX-2 correlation coefficient slope

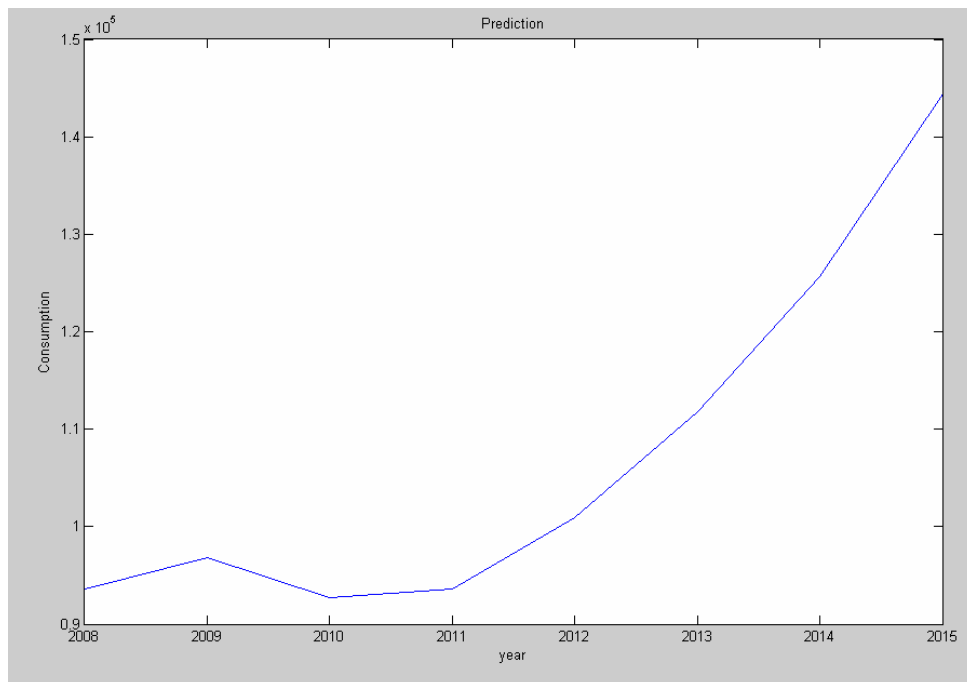


Figure IV.31 NARX-2 forecast between 2008-2015

IV.1.4.2. Second NARX Model (NARX-9)

The selected NARX network structure is shown in Figure IV.32 To facilitate the performance comparison, network has an input layer, 50 process elements in hidden layer and an output layer with sse (sum squared error) performance criteria. All neurons in hidden layer use tansig (tangent sigmoid) activation function and all neurons in output layer use purelin activation function. The initial values are generated randomly

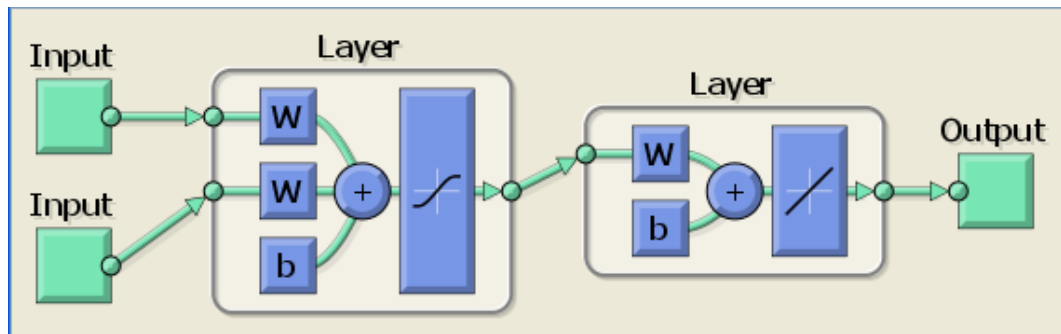


Figure IV.32 NARX-9 model architecture

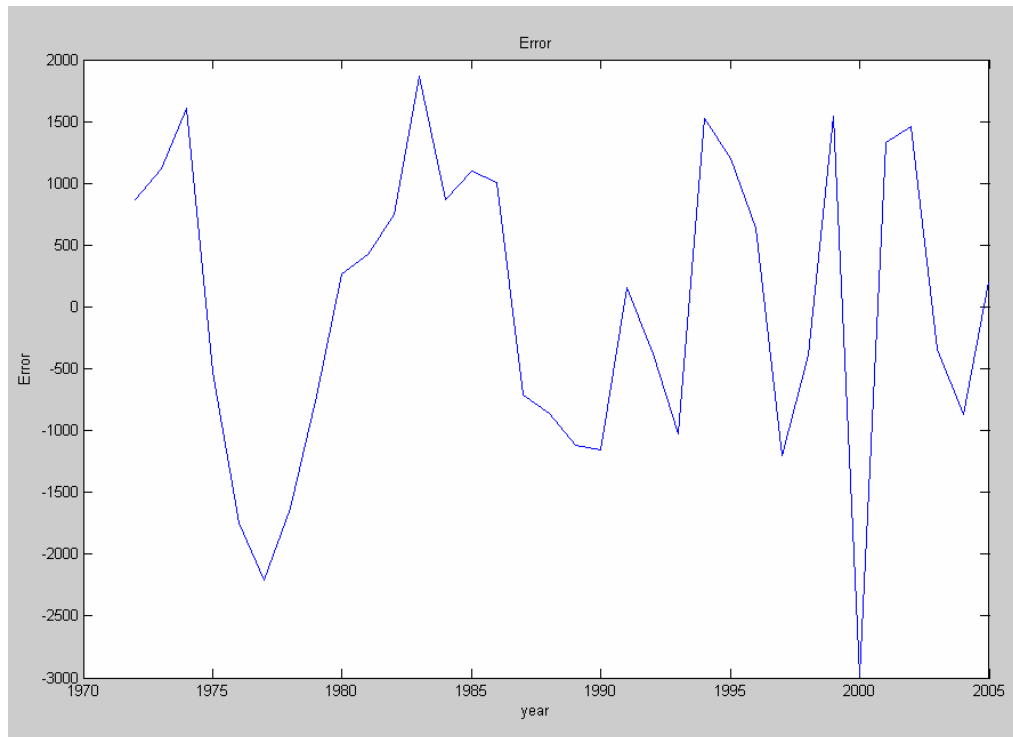


Figure IV.33 NARX-9 error slope

As we seen in Figure IV.34 and Figure IV.36 train correlation coefficient value is equal to 0,99941 and narx correlation coefficient value is equal to 0,98826 This is the optimum result of NARX model in 10 NARX models. Figure IV.35 is shown the prediction slope from 1970 to 2005. Figure IV.37 shows the NARX forecasts between 2008-2015.

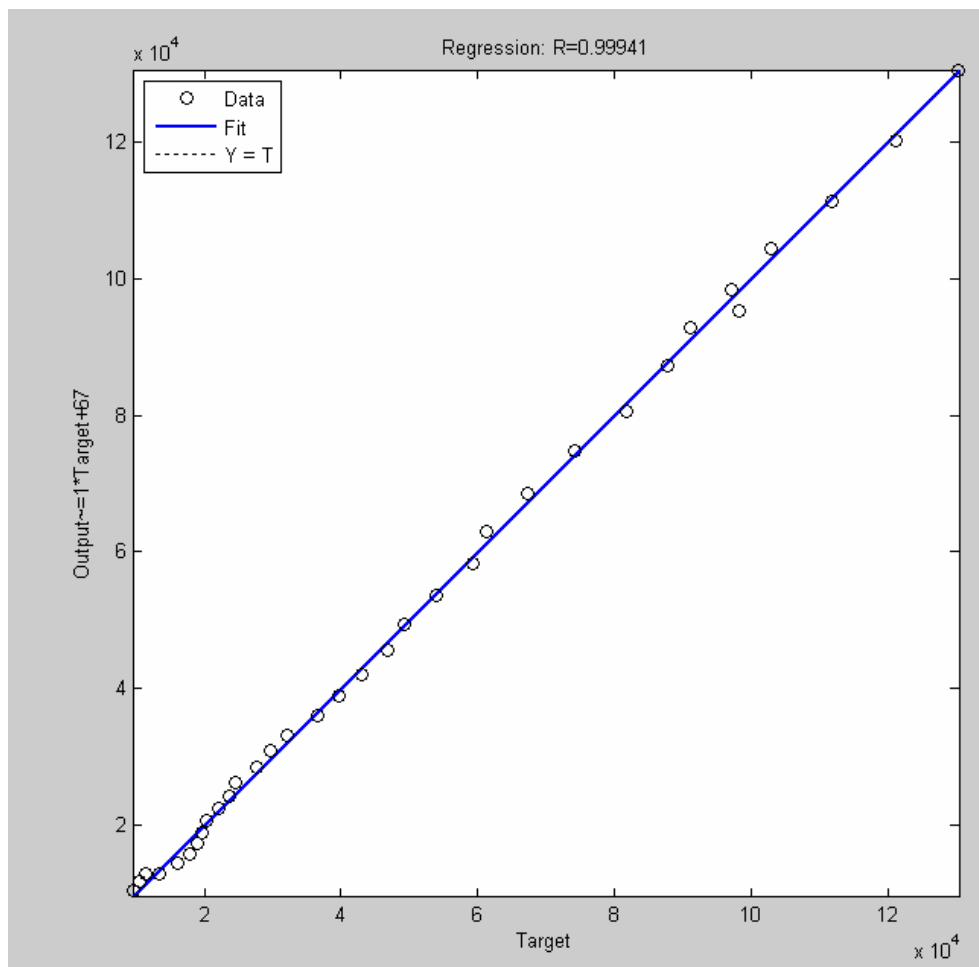


Figure IV.34 NARX-9 train correlation coefficient slope

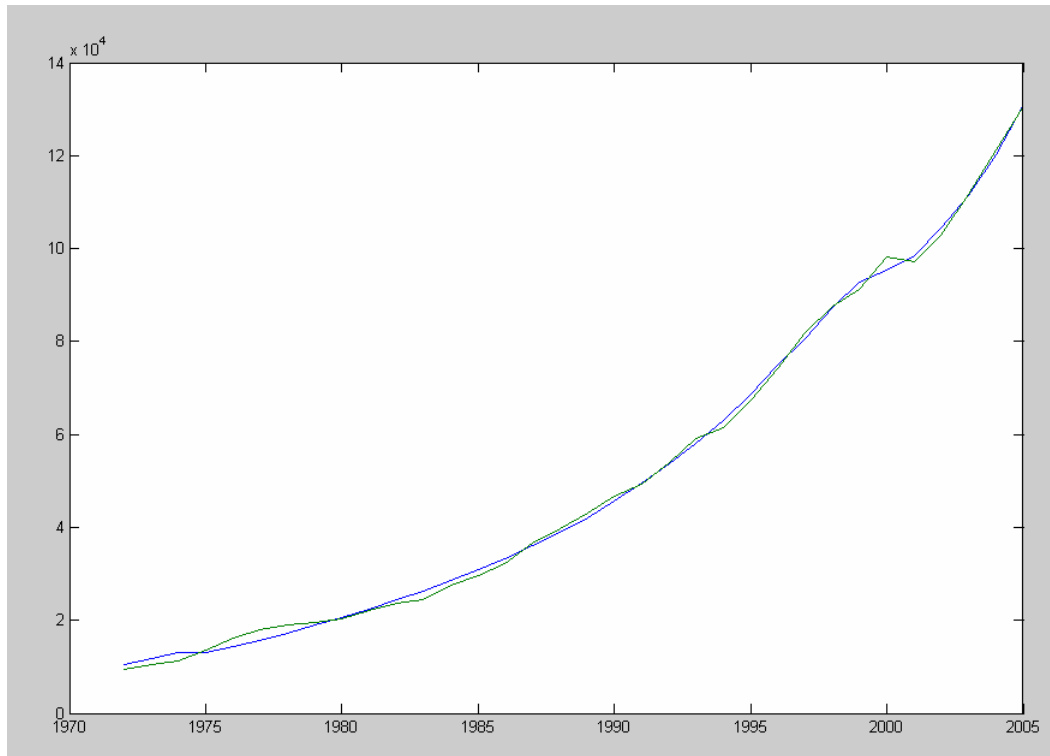


Figure IV.35 NARX-9 forecast between 1970-2005

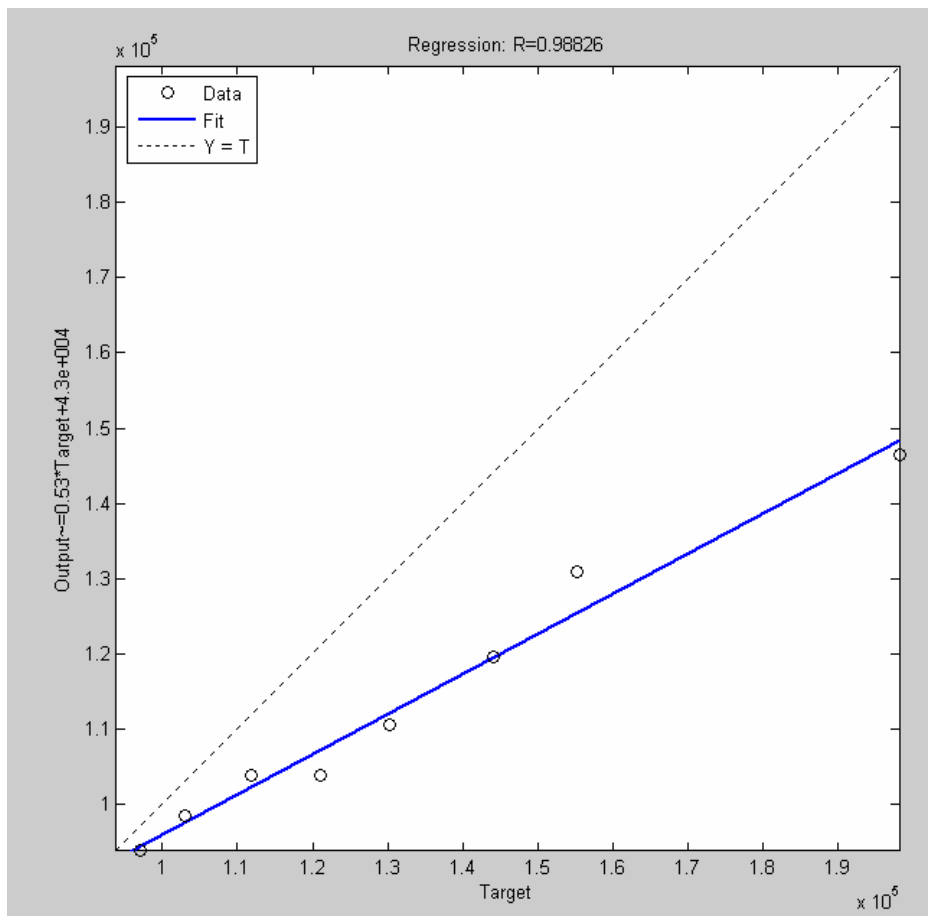


Figure IV.36 NARX-9 correlation coefficient slope

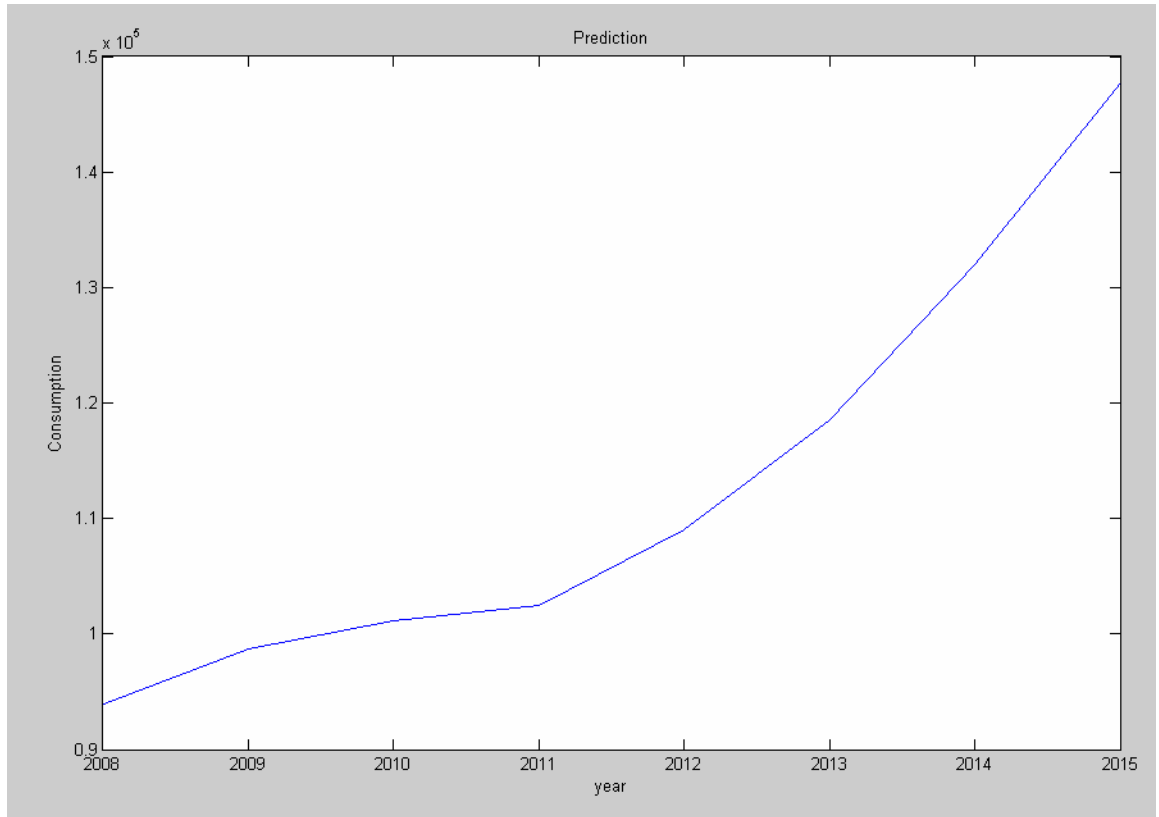


Figure IV.37 NARX-9 forecast between 2008-2015

IV.2 FORECASTING WITH REGRESSION MODEL

Population (P), installed electricity capacity (IP), gross national product (GNP), electricity energy imports (EEI), electricity energy exports (EEE), mid-annual exchange rates (D), electricity energy production (EEP), industry electricity energy consumption (IEEC), total electricity energy price (TEEP), gross domestic product (GDP), consumer price index (CPI), average electricity sale prices (AP) are used in the input datas. The electricity consumption is in the output datas. This input-output network models are a result of relationships that exist among electricity consumption and several other variables.

While related series data for 1970-2005 period were used for training of the model. The data between 2006 to 2008 are left as test set. To compare model performances mean absolute error (MAE), mean absolute percentage error (MAPE) ,and mean squared error (MSE) were used.

Variables are highly correlated (Appendix-6). Therefore we use principal component analysis (PCA). Principal component analysis consists of finding an orthogonal transformation of the original variables to a new set of uncorrelated variables, called principal components, which are derived in decreasing order of importance. These components are linear combinations of the original variables. The analyst often hopes that the first few components will account for most of the variation in the original data so that the effective dimensionality of the data can be reduced [8].

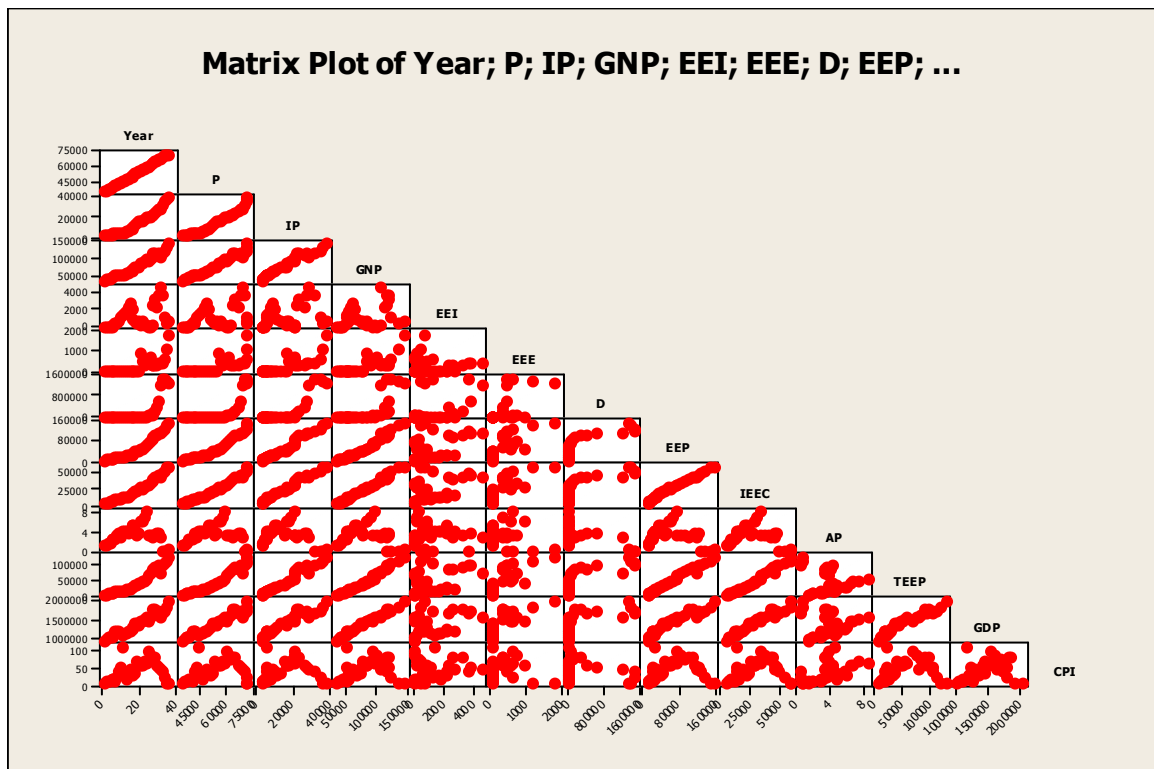


Figure IV.38 Matrix plot of variables

Since 77.6 % of the variation is explained by the first principal component. It is decided to use this component as an independent variable in the regression analysis. The other variable considered in this analysis is AP (average electricity sale prices). Principal component scores (PCS) are found by;

$$\begin{aligned} \text{PCS} = & (0.322 \times Z_{\text{Year}}) + (0.322 \times Z_P) + (0.325 \times Z_{\text{GNP}}) + (0.325 \times Z_{\text{GNP}}) + \\ & (0.142 \times Z_{\text{EEI}}) + (0.259 \times Z_{\text{EEE}}) + (0.258 \times Z_D) + (0.326 \times Z_{\text{EEP}}) + (0.327 \times \\ & Z_{\text{IEEC}}) + (0.324 \times Z_{\text{TEEP}}) + (0.319 \times Z_{\text{GDP}}) + (0.091 \times Z_{\text{CPI}}) \end{aligned}$$

In this equation the standardization is done by ,

$$Z = \frac{X - \bar{X}}{s} \quad (IV.1)$$

where ,

X is the value of the variable

$\bar{X}$ is the average of the observation for that variable

s is the standard deviation of the for that variable

Table IV.1 Principal Component Analysis

Principal Component Analysis: Year; P; IP; GNP; EEI; EEE; D; EEP; IEEC; TEEP; G								
Eigenanalysis of the Correlation Matrix								
Eigenvalue	9,3097	1,2675	0,9938	0,1830	0,1661	0,0525	0,0177	0,0054
Proportion	0,776	0,106	0,083	0,015	0,014	0,004	0,001	0,000
Cumulative	0,776	0,881	0,964	0,980	0,993	0,998	0,999	1,000
Eigenvalue	0,0024	0,0010	0,0005	0,0004				
Proportion	0,000	0,000	0,000	0,000				
Cumulative	1,000	1,000	1,000	1,000				
Variable	PC1	PC2						
Year	0,322	0,118						
P	0,322	0,119						
IP	0,325	-0,066						
GNP	0,325	0,032						
EEI	0,142	0,300						
EEE	0,259	-0,299						
D	0,258	-0,387						
EEP	0,326	-0,053						
IEEC	0,327	0,003						
TEEP	0,324	-0,018						
GDP	0,319	0,060						
CPI	0,091	0,794						

Table IV.2 Regression Analysis

Regression Analysis: C versus PCS; AP

The regression equation is

$$C = 54361 + 11685 \text{ PCS} - 1469 \text{ AP}$$

Predictor	Coef	SE Coef	T	P
Constant	54360.8	869.0	62.55	0.000
PCS	11685.2	133.9	87.25	0.000
AP	-1468.7	215.4	-6.82	0.000

S = 2383.27 R-Sq = 99.6% R-Sq(adj) = 99.6%

Analysis of Variance

Source	DF	SS	MS	F	P
Regression	2	45929524833	22964762416	4043.10	0.000
Residual Error	33	187439647	5679989		
Total	35	46116964479			

Source	DF	Seq SS
PCS	1	45665475929
AP	1	264048904

The assumptions of regression analysis that the errors are uncorrelated, normally distributed random variables with mean zero and constant variance are all satisfied. The related graphs are given in Figure IV.39 and Figure IV.40.

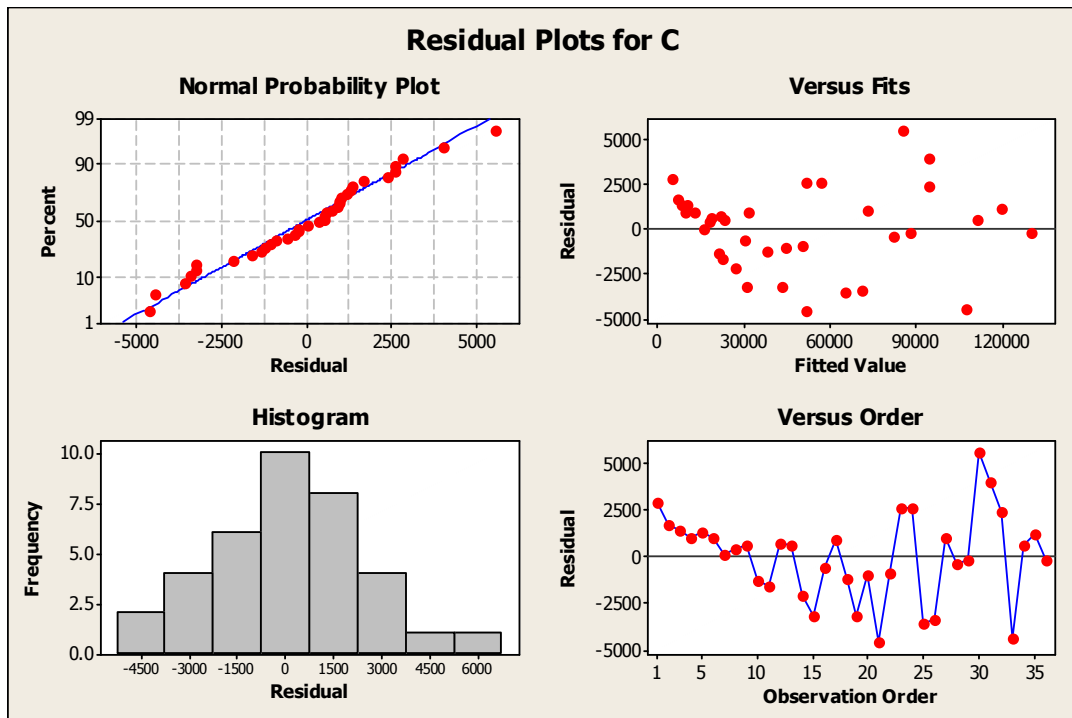


Figure IV.39 Residual analysis

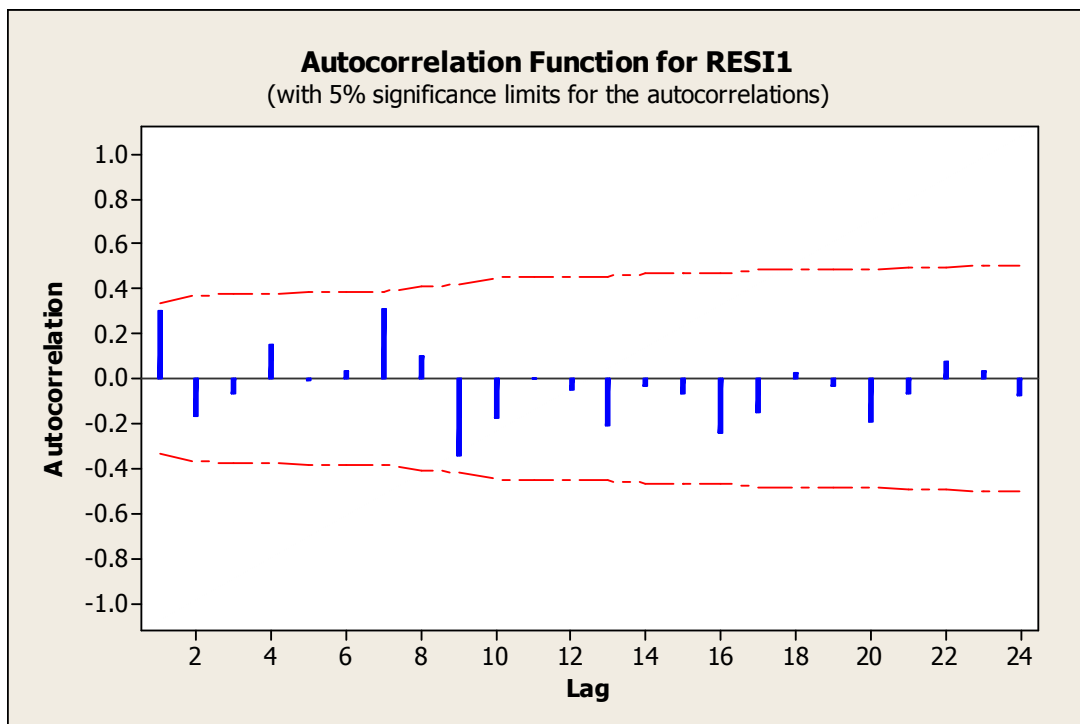


Figure IV.40 Autocorrelation function for RESI1

IV.3 FORECASTING WITH ARIMA MODEL

In order to use an ARIMA model we can usually assess stationarity using a time series plot:

- 1) If a time series is plotted and there is no evidence of a change in the mean over time, then we say the series is stationary in the mean.
- 2) If the plotted series shows no obvious change in the variance over time, then we say the series is stationary in the variance

Electricity consumption data is non-stationary in both mean and variance (Figure IV.41).

Achieving stationarity in variance is through a logarithmic transformation of the data. The logged data are clearly non-stationary and require differencing.

There is trend present in the time series data. In order to make data stationary in variance $\ln$ transformation is used. In order to make data stationary in its mean the second order differences are taken.

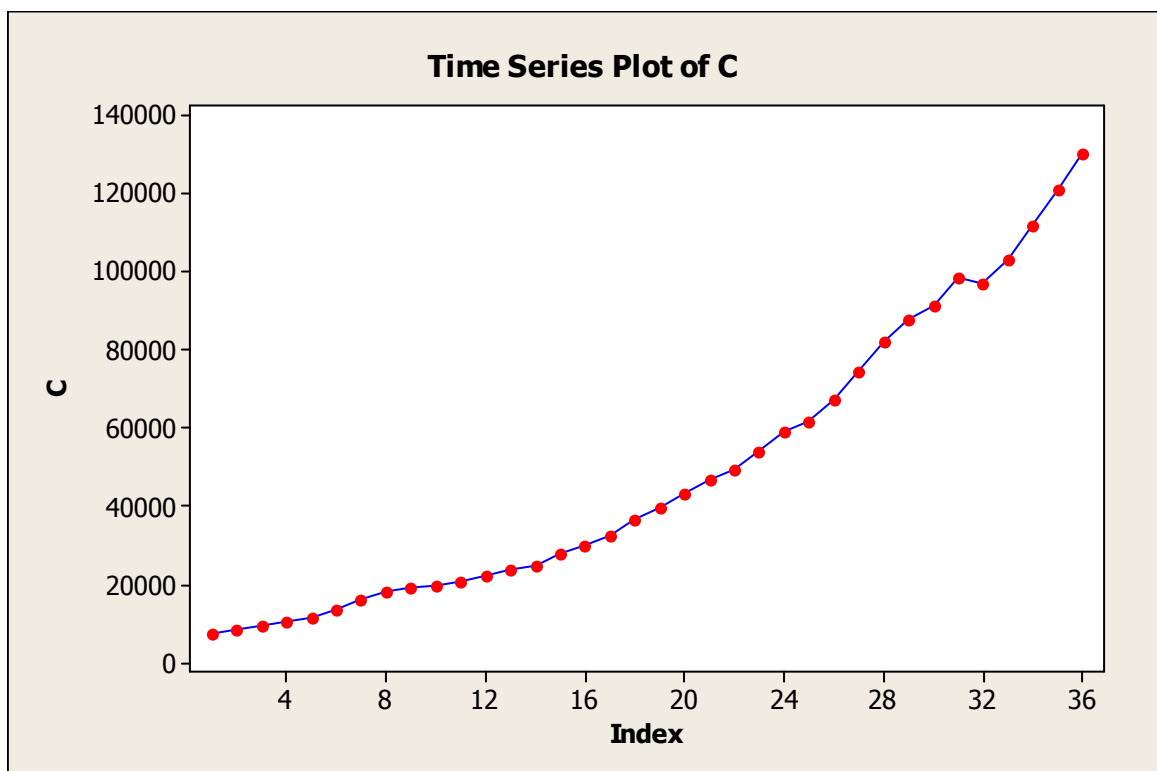


Figure IV.41 Time series plot of electricity consumption

The autocorrelation function shows an exponential decay (Figure IV.42), there is a single spike in the partial autocorrelation function (Figure IV.43). Therefore, it is considered that AR(1) model will be appropriate to use.

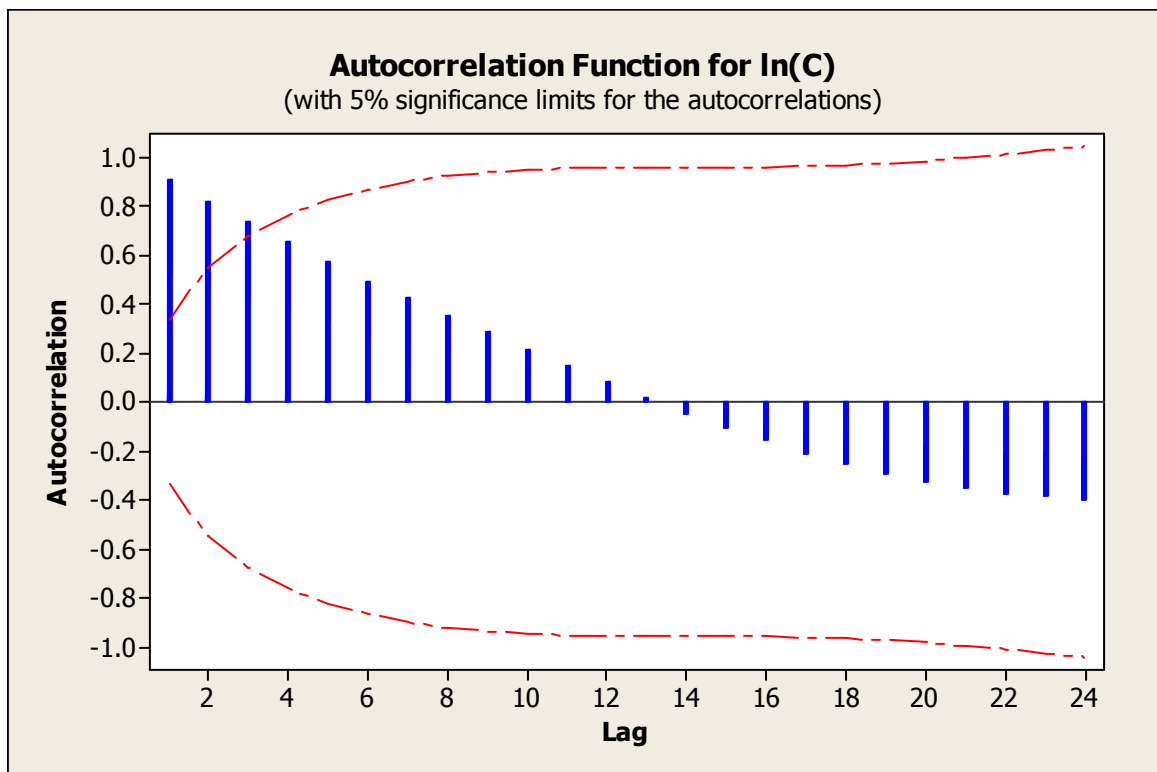


Figure IV.42. Autocorrelation function for $\ln(C)$

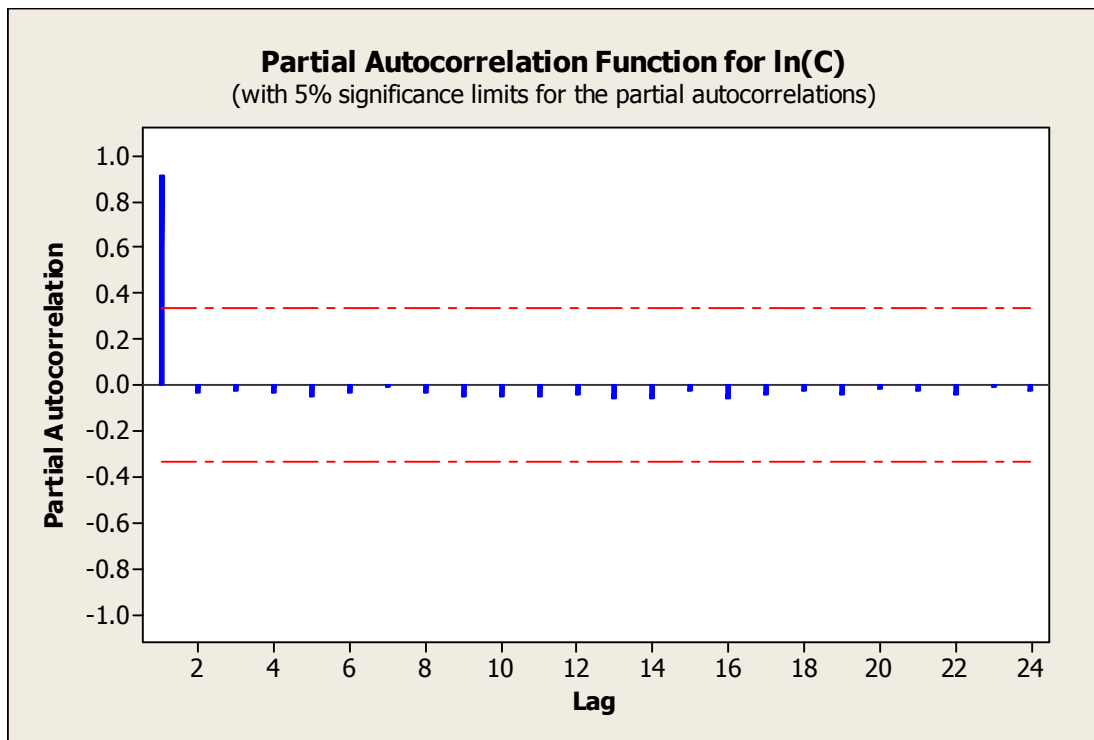


Figure IV.43 Partial Autocorrelation function for $\ln(C)$

The ARIMA (1,2,0) model fitted to the data is as follows:

$$Y_t = c + \theta_1 Y_{t-1} + e_t$$

$$(Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = c + \theta_1 [(Y_{t-1} - Y_{t-2}) - (Y_{t-2} - Y_{t-3})] + e_t$$

$$Y_t - 2Y_{t-1} + Y_{t-2} = c + \theta_1 (Y_{t-1} - 2Y_{t-2} + Y_{t-3}) + e_t$$

$$Y_t = c + (\theta_1 + 2)Y_{t-1} - (2\theta_1 + 1)Y_{t-2} + \theta_1 Y_{t-3} + e_t$$

$$Y_t = c + (\theta_1 + 2)Y_{t-1} - (2\theta_1 + 1)Y_{t-2} + \theta_1 Y_{t-3} + e_t$$

where

$$Y_t = \ln C$$

The results of the analysis is given in Table IV.3 and the parameter estimates of the model is as follows:

$$c = -0,002138$$

Table IV.3 ARIMA model $\ln(C)$

ARIMA Model: $\ln(C)$

Estimates at each iteration

Iteration	SSE	Parameters	
0	0.343978	0.100	0.089
1	0.102125	-0.050	0.033
2	0.060691	-0.200	0.008
3	0.056421	-0.303	-0.002
4	0.056413	-0.311	-0.002
5	0.056413	-0.311	-0.002
6	0.056413	-0.311	-0.002

Relative change in each estimate less than 0.0010

Final Estimates of Parameters

Type	Coef	SE Coef	T	P
AR 1	-0.3112	0.1680	-1.85	0.073
Constant	-0.002138	0.007200	-0.30	0.768

Differencing: 2 regular differences

Number of observations: Original series 36, after differencing 34

Residuals: SS = 0.0563942 (backforecasts excluded)
MS = 0.0017623 DF = 32

Modified Box-Pierce (Ljung-Box) Chi-Square statistic

Lag	12	24	36	48
Chi-Square	8.8	15.5	*	*
DF	10	22	*	*
P-Value	0.547	0.839	*	*

Forecasts from period 36

95% Limits				
Period	Forecast	Lower	Upper	Actual
37	11.8502	11.7679	11.9325	
38	11.9209	11.7594	12.0824	
39	11.9902	11.7303	12.2501	

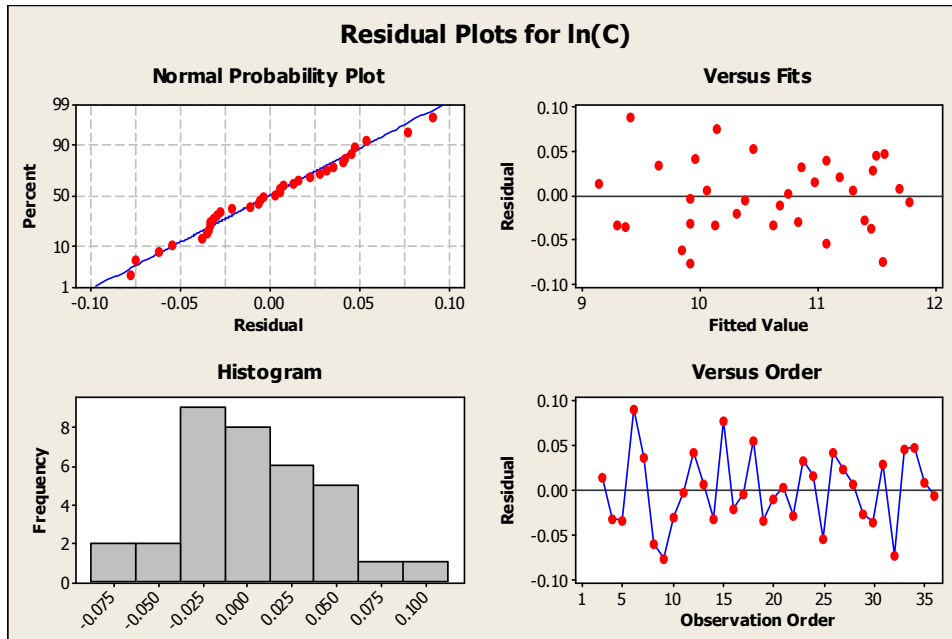


Figure IV.44 Residual analysis

There isn't any pattern in the residual versus fits graph therefore, it's concluded that constant variance assumption is valid. There isn't any pattern in the residual versus order graph therefore, it's concluded that residuals are independent (uncorrelated). Normal probability plot shows that residuals are normally distributed with mean zero (Figure IV.44).

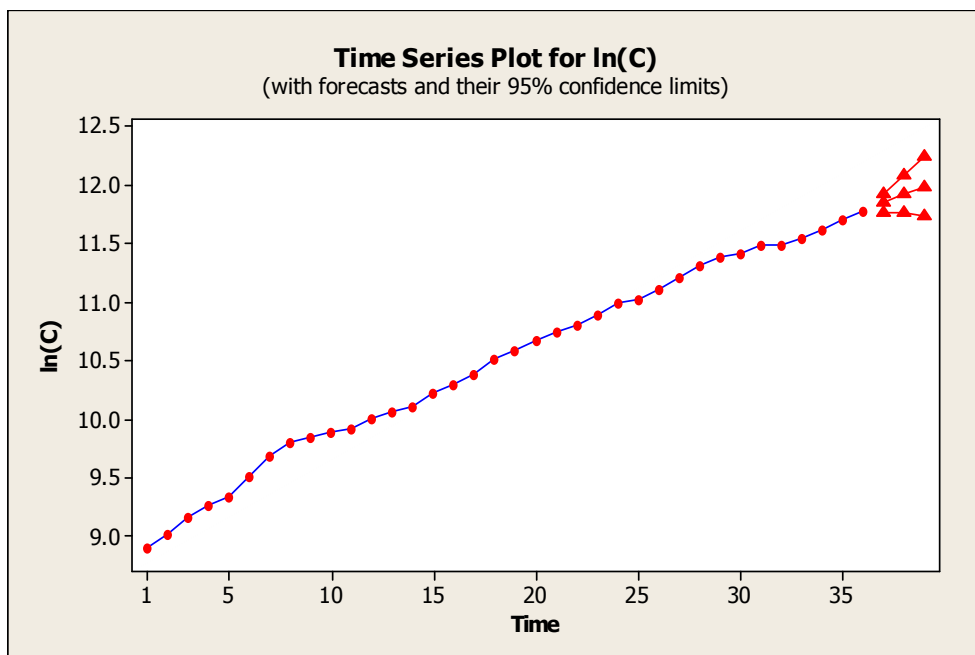


Figure IV.45 Time Series plot for $\ln(C)$

The forecast are obtained for $Y_t = \ln C$ (Figure IV.45) as 11.8502, 11.9209, and 11.9902 respectively for years 2006, 2007, and 2008. These forecast values are transformed to original units by formula $C = e^{Y_t}$ and the original forecasts are found as 140118, 150382, and 161164.

CHAPTER V

CONCLUSION

In this study, electricity consumption was forecasted using multilayer perceptron (MLP) model, ELMAN network model, radial basis function (RBF) model, nonlinear autoregressive with exogenous inputs (NARX) model, an ARIMA model, and a regression model.

A more realistic way of assessing accuracy of a model is to separate the end of series as a test set and not use this set in the model fitting. Then the models are compared according to their ability to forecast the data which have been withheld.

Table V.1 Forecasts and Values of Error Measures

Year	Electricity Consumption	Forecasts from Models			
		ARIMA (1,2,0)	Regression	MLP-14	MLP-21
2006	144,091.40	140,118.31	141,720.73	157,615.64	145,682.76
2007	155,135.20	150,381.51	148,052.27	161,520.99	151,914.72
2008	198,085.00	161,164.26	142,428.71	194,852.52	199,428.35
Error Measures					
MSE		467,174,632	1,051,136,765	78,044,045	4,902,841
MAE		15,215.84	21,703.30	7,714.17	2,051.73
MAPE		8.15	11.44	5.04	1.29

Year	Electricity Consumption	Forecasts from Models				
		ELMAN-1	ELMAN-34	RBF	NARX-2	NARX-9
2006	144,091.40	154,861.67	160,369.59	140,685.97	120,760.44	119,653.76
2007	155,135.20	160,328.75	167,896.56	140,705.08	135,919.91	130,946.88
2008	198,085.00	199,221.98	188,080.18	140,968.02	157,461.55	146,468.91
Error Measures						
MSE		48,088,145	175,976,121	1,160,724,973	854,608,655	1,282,164,730.84
MAE		5,700.27	13,014.79	24,984.18	27,723.23	33,414.02
MAPE		3.80	8.19	13.50	16.36	19.54

The data at the end of the three periods were separated before the models were fitted. This test set consisted of years from 2006 to 2008. Then the models were

compared on the basis of how well they forecasted these periods. As comparison measures mean squared error (MSE), mean absolute error (MAE), and mean absolute percentage error (MAPE) were used. The results are given in Table V.1.

According to error measures the best performing model is MLP-21. The second in the list is ELMAN-1. It is observed that MLP models and ELMAN models are predicting more successfully than both ARIMA (1,2,0) model and regression model. RBF, NARX2 and NARX-9 models perform worse than all other models in this comparison.

It is concluded that neural network forecasting methods can perform better than explicit model-based methods especially when required assumptions for these methods cannot be met. In this study, since input variables were highly correlated in order to avoid multicollinearity in regression, regression analysis were performed after principal components analysis which resulted in losing a part of variability explaining the forecast variable. Whereas, neural network forecasting methods are flexible in the inclusion of input variables which provides a great advantage. Therefore, neural network forecasting methods could be a strong alternative to the traditional methods.

The future study can be carried out considering more input variables that are thought to have some effects on the forecast variable. In addition, other techniques such as genetic programming can be applied.

REFERENCES

- [1] Haykin,S.: “ Neural Networks: A Comprehensive Foundation”, 2 nd Edition.; Prentice Hall, McMaster University, Ontario, Canada, **(1999)**.
- [2] Kröse,B.; Smagt, P.: “An Introduction to Neural Networks”, 8 th Edition; The University of Amsterdam, **(1996)**.
- [3] Freeman, J.A.; Skapura, D:M.; “Neural Networks Algorithms, Applications and Programming Techniques”,Addison-Wesley Publishing Company,**(1991)**.
- [4] Öztemel, E.; “Yapay Sinir Ağları”; Papatya Yayıncılık Eğitim, İstanbul, Türkiye, **(2006)**.
- [5] Wheelright, S.C.; Makridakis, S.: “Forecasting Methods for Management”, 4 th Edition; John Wiley & Sons, New York, **(1985)**.
- [6] Markidakis,S.; Wheelwright, S.C.; Hyndman, R.J.; “Forecasting Methods and Applications”, 3 rd Edition; John Wiley & Sons, America, **(1998)**.
- [7] Markidakis,S.; Wheelwright, S.C.; Mcgee, V.E.; “Forecasting Methods and Applications”, 2 nd Edition; **(1977)**.
- [8] Chatfield,C.; Collins, A.J.; “Introduction to Multivariate Analysis”, Chapman&Hall, London, UK, **(1992)**
- [9] Tunc, M.; Camdali, U.; Parmaksizoglu, C.: “Comparison of Turkey’s electrical energy consumption and production with some European countries and optimization of future electrical power supply investments in Turkey”, Energy Policy, Vol 34, Issue 1,Januarry **(2006)**, 50-59
- [10] Ogulata,R.T.: “Energy Sector and wind energy potential in Turkey”, Renewable and Sustainable Energy Reviews, Vol 7, Issue 6, December **(2003)**, 469-484
- [11] Ozturk, H.K; Yilanci, A.; Atalay, O.: “Past, present and future status of electricity in Turkey and the share of energy sources”, Renewable and Sustainable Energy Reviews, Vol 11, No:2, **2007**,183-209
- [12] Topcu,Y.I.; Ulengin, F.: “Energy for the future: an integrated decision aid for the case of Turkey”, Energy, Vol 29, Issue 1, January **(2004)** , 137-54.

- [13] Tunc,M.; Camdali, U.; Liman, T.; Deger, A.: “Electrical energy consumption and Production of Turkey versus world”, Energy Policy,Vol 34, Issue 17, November **(2006)**, 3284-3292
- [14] Hepbasli, A.; Ozdamar, A.; Ozalp, N.: “Present status and potential of enewable energy sources in Turkey”, Energy Sources, Vol 23, Issue 7, August **(2001)** ,631-648.
- [15] Evrendilek, F.; Ertekin, C.: “Assessing the potential of renewable energy sources in Turkey”, Renewable Energy,Vol 28, Issue 15, December **(2003)**, 2303-2315
- [16] Aras, H.: “Wind energy status and its assessment in Turkey”, Renewable Energy, Vol 28, Issue 14, November (2003),2213-2220
- [17] Gokcen, G.; Ozturk, H.K.; Hepbasli,A.: “Overview of Kizildere geothermal power plant in Turkey”, Energy Conversion and Management, Vol 45, Issue 1, January **(2004)** ,83-98
- [18] Hepbasli, A.: “Development and restructuring of Turkey’s electricity sector: a review”, Renewable Sustainable Energy Reviews,Vol 9, Issue 4, August **(2005)** , 311-43
- [19] Hepbasli, A.; Özalp, H.: “Co-generation studies in Turkey: an application of a seramic factory in İzmir, Turkey”, Applied Thermal Engineering, Vol 22, Issue 6, April **(2002)**,679-691
- [20] Peter, A.; Harald, B.; Wolfgang, M.: “A learning rule for very simple universal approximators consisting of a single layer of perceptrons”, Neural Networks, Vol 21, Issue 5, June **(2008)**, Pages 786-795
- [21] Cybenko,G.: “Approximation by superpositions of a sigmoidal function”, Mathematics of Control,Signal and Systems, Vol 2,**1989**, 303-314
- [22] Christopher, B.: “Neural Networks for Pattern Recognition”, Oxford University Press, Oxford, **(1995)**
- [23] Wasserman, P.D.; Schwartz, T.; „Neural networks. II. What are they and why is everybody so interested in them now?”, IEEE Expert, Volume 3, Issue 1, **(1988)**, 10-15
- [24] Holk, C.: “Neural Networks as Cybertenic Systems, 2nd and revised edition; Bielefeld University, Universitaetsstraße 25, 33615 Bielefeld, Germany
- [25] <http://www.juliantrubin.com/encyclopedia/electricity/energy-resources.html> (Accessed: December 2009)

- [26] http://en.wikipedia.org/wiki/Recurrent_neural_network (Accessed: December 2009)
- [27] <http://www.teias.gov.tr/istatistik2008/index.htm> (Accessed: August 2009)
- [28] <http://www.tuik.gov.tr> (Accessed: August 2009)
- [29] www.umich.edu/~gs265/society/fossilfuels.htm (Accessed: December 2009)
- [30] www.darvill.clara.net/eltenerg/nuclear/htm. (Accessed: December 2009)
- [31] <http://www.eia.doe.gov/emeu/cabs/turkey.html;2003> (Accessed: December 2009)
- [32] <http://en.wikipedia.org/wiki/hydropower> (Accessed: December 2009)
- [33] <http://www.worldpumps.com> (Accessed: December 2009)
- [34] http://en.wikipedia.org/wiki/world_energy_and_consumption (Accessed: December 2009)
- [35] http://en.wikipedia.org/wiki/Feedforward_neural_network (Accessed: December 2009)
- [36] <http://www.cse.unsw.edu.au/~waleed/phd/html/node37.html> (Accessed: December 2009)
- [37] <http://www.dtrek.com/MLFNfourLayer.jpg&imgrefurl> (Accessed: December 2009)
- [38] <http://documents.wolfram.com/applications/neuralnetworks> (Accessed: December 2009)
- [39] <http://www.cis.hut.fi> (Accessed: December 2009)
- [40] <http://www.mathworks.com/access/helpdesk/help/toolbox/nnet> (Accessed: December 2009)

APPENDIX

APPENDIX-1 : Data series [27] [28]

Year	Population (1000 Person)	Installed Capacity Mwatt	Electricity Consumption (GWh)	Gross National Product(with 1987 Prices -1000 TL	Electricity Imports (GWh)	Electricity Exports (GWh)
1970	35321	2234,90	7307,80	34469	0	0
1971	36215	2577,90	8289,30	36897	0	0
1972	37133	2711,30	9527,30	40279	0	0
1973	38073	3192,50	10530,10	42255	0	0
1974	39037	3732,10	11358,70	43633	0	0
1975	40026	4186,60	13491,70	46275	96	0
1976	40916	4364,20	16078,90	50438	332	0
1977	41769	4727,20	17968,80	51944	492	0
1978	42641	4868,70	18933,80	52582	621	0
1979	43531	5118,70	19663,10	52324	1044	0
1980	44439	5118,70	20398,20	50870	1341	0
1981	45540	5537,60	22030,00	53317	1616	0
1982	46688	6638,60	23586,80	54963	1773	0
1983	47864	6935,10	24465,10	57279	2221	0
1984	49070	8461,60	27635,20	61350	2653	0
1985	50307	9121,60	29708,60	63989	2142	0
1986	51433	10115,20	32209,70	68315	777	0
1987	52561	12495,10	36697,30	75019	572	0
1988	53715	14520,60	39721,50	76108	381	0
1989	54894	15808,20	43120,00	77347	559	0
1990	56098	16317,60	46820,00	84592	176	907
1991	57193	17209,10	49282,90	84887	759	506
1992	58248	18716,10	53984,70	90323	189	314
1993	59323	20337,60	59237,00	97677	213	589
1994	60417	20859,80	61400,90	91733	31	570
1995	61532	20954,30	67393,90	99028	0	696
1996	62667	21249,40	74156,60	106080	270	343
1997	63823	21891,90	81885,00	114874	2492	271
1998	65001	23354,00	87704,60	119303	3299	298
1999	66200	26119,30	91201,90	112044	2330	285
2000	67421	27264,10	98295,70	119144	3791	437
2001	68529	28332,40	97070,00	107783	4579	433
2002	69626	31845,80	102948,00	116338	3588	435
2003	70712	35587,00	111766,00	123165	1158	588
2004	70712	36824,00	121141,90	135308	464	1144
2005	70712	38843,50	130262,90	145651	636	1798
2006	70712	40564,80	144091,40	148504	573	2236
2007	70712	40835,70	155135,20	152188	864	2422
2008	71517	41417,20	198085,00	155873	789	1122

APPENDIX-1 : Data series [27] [28]

Year	Mid-Annual Exchange Rates (US \$)	Electricity Generation GWh	Industry Sector Electricity Consumption GWh	Average Electricity Sale Price (old TL/kWh)	Average Electricity Sale Price (Cent/kWh)	Total Electricity Sale and Service Price (Million TL)
1970	11	8175	4615	0,15	1,36	844,01
1971	15	9273	5260	0,20	1,33	1387,27
1972	14	10657	6092	0,24	1,71	1997,24
1973	14	11814	6920	0,26	1,86	2442,39
1974	14	12846	7404	0,39	2,79	4033,39
1975	14	15031	8559	0,40	2,86	4882,48
1976	16	17459	10305	0,45	2,81	6637,22
1977	18	19555	11761	0,62	3,44	10017,04
1978	24	20500	12171	0,98	4,08	15515,55
1979	38	21182	12280	1,26	3,32	22557,21
1980	76	21882	12687	3,32	4,37	61829,33
1981	110	23345	13919	4,91	4,46	98494,98
1982	161	25131	14806	7,31	4,54	164610,97
1983	224	25667	15141	8,69	3,88	201181,10
1984	365	28723	17553	15,45	4,23	408187,83
1985	518	31912	19008	29,34	5,66	809405,37
1986	669	36880	20242	38,46	5,75	1155642,95
1987	856	41745	23026	39,52	4,62	1340405,48
1988	1421	45649	24355	50,82	3,58	1869529,76
1989	2121	48809	26653	86,98	4,10	3469092,50
1990	2608	54232	28062	165,47	6,34	7383533,98
1991	4170	56591	27056	270,08	6,48	12675618,60
1992	6869	63105	30015	501,58	7,30	25387053,00
1993	10986	69864	32589	927,83	8,45	51419722,54
1994	29704	73783	32468	1083,90	3,65	71435772,00
1995	45705	81859	36337	1571,85	3,44	114071989,00
1996	81137	90084	38963	2730,04	3,36	217660145,00
1997	151428	98246	41515	4850,08	3,20	427855540,00
1998	260040	105499	44020	8747,62	3,36	819752399,00
1999	417581	110702	44623	16487,04	3,95	1504590687,52
2000	623749	118698	46686	25333,90	4,06	2569719098,00
2001	1222921	116252	45364	36081,16	3,11	3491342505,00
2002	1504598	123727	48642	5847,60	0,39	598518103,10
2003	1495307	135248	54081	6143,49	0,41	672673332,00
2004	1422341	145066	58042	6245,47	0,44	731638300,00
2005	1340800	155469	58721	6200,00	0,47	787164872,00
2006	1431100	169543	67172	6400,00	0,45	906271767,04
2007	1301500	183340	69450	5400,00	0,41	885283367,50
2008	1292900	198418	71188	5200,00	0,40	897333000,00

APPENDIX-1 : Data series [27] [28]

Year	Total Electricity Sale and Service Price(1000 \$)	Total Electricity Sale (GWh)	Gross Domestic Product (With 1987 Prices -YTL)	Consumer Price Index	Real GDP Growth (Percentage Change)
1970	76728	5606	955951	8,1	
1971	92485	7012	984252	16,5	5,6
1972	142660	8464	1031230	13,7	7,4
1973	174457	9344	1038579	16,0	3,3
1974	288099	10239	1069599	18,6	5,6
1975	348749	12183	1116530	19,8	7,2
1976	414826	14640	1208102	16,4	10,5
1977	556502	16213	1223744	28,0	3,4
1978	646481	15897	1216735	47,2	1,5
1979	593611	17956	1184419	56,8	-0,6
1980	813544	18624	1131824	115,6	-2,4
1981	895409	20080	1158074	33,9	4,9
1982	1022428	22516	1169850	21,9	3,6
1983	898130	23154	1197831	31,4	5,0
1984	1118323	26425	1246814	48,4	6,7
1985	1562559	27584	1267764	45,0	4,2
1986	1727418	30051	1326932	34,6	7,0
1987	1565894	33917	1421623	38,9	9,5
1988	1315644	36787	1420577	73,7	2,1
1989	1635593	39884	1393590	63,3	0,3
1990	2831110	44621	1487082	60,3	9,3
1991	3039717	46934	1471998	63,8	0,9
1992	3695888	50614	1530808	72,3	6,0
1993	4680477	55420	1623613	66,1	8,0
1994	2404921	65906	1507540	106,3	-5,5
1995	2495832	72572	1587953	93,6	7,2
1996	2682625	79728	1670657	80,3	7,0
1997	2825472	88216	1802677	84,5	7,5
1998	3152409	93712	1829754	86,7	3,1
1999	3603111	91259	1719567	64,8	-4,7
2000	4121978	101434	1760856	56,4	7,4
2001	3095444	74976	1601407	53,5	-7,5
2002	397793	102353	1703562	47,2	7,9
2003	449856	109494	1774594	25,6	5,8
2004	514390	117147	1904088	10,7	8,9
2005	586648	126139	2036782	8,2	7,4
2006	633269	142063	2134082	9,6	6,1
2007	680202	164553	2166808	7,1	4,6
2008	690469	172635	2199534	10,4	3,3

APPENDIX-2 : Multilayer Perceptron performance criteria and performances

Performance Criteria					Performance			
Number	Number of Hidden Layers	Hidden Layer Neuron Number *	Performance Criteria	Activation Function	Train mse	Test mse	Valid mse	Train R
1	1	3	mse	Tansig	3,29E-04	5,93E-03	0,0008494	0,9997273
2	1	5	mse	Tansig	8,39E-03	1,75E-02	0,0055291	0,9894824
3	1	10	mse	Tansig	1,54E-05	1,65E-02	3,36E-05	0,9999886
4	1	15	mse	Tansig	1,28E-03	2,25E-03	0,0032879	0,9992534
5	1	20	mse	Tansig	6,13E-04	1,19E-01	0,0467849	0,9999057
6	1	3	sse	Tansig	2,84E-06	1,52E-03	0,0001893	0,9999979
7	1	5	sse	Tansig	1,75E-01	6,73E-02	0,076332	0,9541373
8	1	10	sse	Tansig	2,27E-05	3,05E-02	0,0021218	0,999991
9	1	15	sse	Tansig	3,73E-07	7,86E-04	0,0002127	0,9999996
10	1	20	sse	Tansig	1,38E-03	3,69E-03	0,004411	0,9992283
11	1	3	mse	Logsig	1,60E-05	7,70E-03	0,0068858	0,9999785
12	1	5	mse	Logsig	0,0015465	1,33E-03	0,003627	0,9991543
13	1	10	mse	Logsig	1,80E-04	1,80E-03	0,0006265	0,9999719
14	1	15	mse	Logsig	1,27E-05	4,63E-04	0,0001498	0,9999965
15	1	20	mse	Logsig	2,31E-08	4,27E-02	0,0002714	1
16	1	3	sse	Logsig	3,79E-04	5,63E-02	0,0045678	0,9993287
17	1	5	sse	Logsig	1,26E-03	7,08E-03	0,0003916	0,9984566
18	1	10	sse	Logsig	8,28E-08	1,79E-03	0,0138768	1
19	1	15	sse	Logsig	3,78E-06	9,46E-03	0,001749	0,9999941
20	1	20	sse	Logsig	2,88E-07	2,26E-03	0,0054397	0,9999998
21	2	3	mse	Tansig	0,0031675	2,55E-03	0,0073381	0,9961369
22	2	5	mse	Tansig	6,03E-03	1,19E-02	0,0821338	0,9970999
23	2	10	mse	Tansig	0,0069409	1,40E-02	0,0028944	0,9959824
24	2	15	mse	Tansig	1,73E-06	2,15E-03	0,0018572	0,9999975
25	2	20	mse	Tansig	0,0011987	2,59E-02	0,0012569	0,9987741
26	2	3	sse	Tansig	1,62E-03	4,02E-03	0,0054223	0,9986765
27	2	5	sse	Tansig	0,0026954	9,37E-03	0,0140138	0,9970781
28	2	10	sse	Tansig	2,05E-05	1,87E-02	0,275978	0,999991
29	2	15	sse	Tansig	4,84E-03	7,55E-02	0,010057	0,998495
30	2	20	sse	Tansig	6,19E-08	2,18E-02	0,0001087	0,9999999
31	2	3	mse	Logsig	1,43E-05	6,85E-01	2,93E-05	0,9999894
32	2	5	mse	Logsig	5,93E-02	5,46E-02	0,0777995	0,9222213
33	2	10	mse	Logsig	4,32E-07	7,13E-04	0,0062181	0,9999996
34	2	15	mse	Logsig	1,43E-05	2,95E-02	0,0481596	0,9999903
35	2	20	mse	Logsig	1,74E-03	1,28E-02	0,0018255	0,9992006
36	2	3	sse	Logsig	2,81E-05	2,97E-02	0,0227706	0,9999594
37	2	5	sse	Logsig	1,07E-02	1,33E-02	4,72E-03	0,9888524
38	2	10	sse	Logsig	1,06E-02	3,98E-02	6,12E-02	0,9972488
39	2	15	sse	Logsig	6,46E-09	4,80E-04	0,0055489	1
40	2	20	sse	Logsig	1,33E-06	3,26E-03	0,002139	0,9999984

* Equal number of neurons were used in hidden layers.

APPENDIX-2 : Multilayer Perceptron performance criteria and performances

Performance Criteria					Performance			
Number	Number of Hidden Layers	Hidden Layer Neuron Number *	Performance Criteria	Activation Function	Validation R	Test R	real mse	real R
1	1	3	mse	Tansig	0,999995523	0,997042	0,2413101	0,8742164
2	1	5	mse	Tansig	0,990364041	0,9772934	0,07011	0,9741617
3	1	10	mse	Tansig	0,999999297	0,9748001	1,7113405	0,8820529
4	1	15	mse	Tansig	0,999130027	0,9941951	0,3123447	0,7749143
5	1	20	mse	Tansig	0,972977717	0,9364412	0,9612356	0,576962
6	1	3	sse	Tansig	0,999992011	0,9969067	0,2211387	0,853117
7	1	5	sse	Tansig	0,551849852	0,9962956	0,3986476	0,9849424
8	1	10	sse	Tansig	0,999917717	0,9808172	1,3777138	0,7965164
9	1	15	sse	Tansig	0,999988149	0,9989447	0,0969018	0,9414418
10	1	20	sse	Tansig	0,9940037	0,9936205	2,6438442	0,2299306
11	1	3	mse	Logsig	0,993413195	0,9894394	2,0480956	-0,9995046
12	1	5	mse	Logsig	0,999663931	0,9986423	0,4023156	0,9493262
13	1	10	mse	Logsig	0,999904504	0,9993078	0,0833872	0,9834712
14	1	15	mse	Logsig	0,999973476	0,9987394	0,1070817	0,9950787
15	1	20	mse	Logsig	0,999145032	0,9454619	0,0265374	0,9856804
16	1	3	sse	Logsig	0,995353376	0,9482475	0,947642	0,634008
17	1	5	sse	Logsig	0,99997498	0,9893434	1,437079	-0,1077105
18	1	10	sse	Logsig	0,990443386	0,9990287	0,4031129	0,7797858
19	1	15	sse	Logsig	0,999987121	0,98659	0,2902695	0,875885
20	1	20	sse	Logsig	0,980953778	0,9949404	0,6805355	0,6711375
21	2	3	mse	Tansig	0,999391622	0,9993062	0,006727	0,9960762
22	2	5	mse	Tansig	0,855188302	0,9777852	1,3146273	-0,4608961
23	2	10	mse	Tansig	0,999759581	0,9788326	0,3117415	0,9999776
24	2	15	mse	Tansig	0,999999999	0,9985943	1,3953782	0,7603696
25	2	20	mse	Tansig	0,999117489	0,9810638	2,1012039	0,5606354
26	2	3	sse	Tansig	0,999821268	0,9928656	0,2987026	0,9722901
27	2	5	sse	Tansig	0,965810595	0,9885611	0,1855792	0,985731
28	2	10	sse	Tansig	0,880998552	0,9775756	1,4780364	-0,9434759
29	2	15	sse	Tansig	0,999957604	0,9861126	3,9910222	-0,7528514
30	2	20	sse	Tansig	0,99995843	0,9786329	0,8510709	0,0037558
31	2	3	mse	Logsig	0,998813147	0,3536075	4,6271817	-0,7893813
32	2	5	mse	Logsig	0,970504068	0,5667222	1,0017466	0,8049325
33	2	10	mse	Logsig	0,998720238	0,9978603	0,1414947	0,9464703
34	2	15	mse	Logsig	0,941392847	0,982584	0,9086548	0,3698867
35	2	20	mse	Logsig	0,995971383	0,9821366	0,3887159	0,9799562
36	2	3	sse	Logsig	0,898910245	0,8367804	0,8412058	0,6107063
37	2	5	sse	Logsig	0,994013774	0,9889643	0,2832594	0,8019513
38	2	10	sse	Logsig	0,817186025	0,9753845	0,079667	0,9999169
39	2	15	sse	Logsig	0,999894802	0,9972158	0,377046	0,7815027
40	2	20	sse	Logsig	0,999010785	0,9956991	0,0837599	0,9553301

* Equal number of neurons were used in hidden layers.

APPENDIX-3 : Elman models performance criteria and performances

Performance Criteria					Performance		
Number	Number of Hidden Layers	Hidden Layer Neuron Number *	Performance Criteria	Activation Function	Train mse	Test mse	Train R
1	1	3	mse	Tansig	2,16E-02	8,22E-02	0,9635625
2	1	5	mse	Tansig	2,23E-02	2,02E-02	0,9799653
3	1	10	mse	Tansig	1,10E-01	6,21E-02	0,8528672
4	1	15	mse	Tansig	3,51E-02	1,65E-01	0,9564365
5	1	20	mse	Tansig	3,22E-02	1,25E-01	0,9647717
6	1	3	sse	Tansig	1,61E-02	9,74E-03	0,979729
7	1	5	sse	Tansig	3,77E-02	4,28E+00	0,95594
8	1	10	sse	Tansig	5,93E-02	8,15E-02	0,9221766
9	1	15	sse	Tansig	6,80E-02	8,64E-02	0,9396718
10	1	20	sse	Tansig	7,70E-02	4,18E-02	0,9262641
11	1	3	mse	Logsig	0,0165029	1,65E-02	0,9704092
12	1	5	mse	Logsig	0,0192539	1,99E-02	0,9650533
13	1	10	mse	Logsig	1,22E+00	1,82E+00	-0,8124912
14	1	15	mse	Logsig	3,32E-03	4,77E-03	0,994942
15	1	20	mse	Logsig	0,0057697	1,28E-02	0,9904609
16	1	3	sse	Logsig	5,61E-02	3,51E-02	0,9123506
17	1	5	sse	Logsig	3,96E-01	1,17E-01	0,8616217
18	1	10	sse	Logsig	0,0196891	3,66E-02	0,9613926
19	1	15	sse	Logsig	6,26E-03	1,50E-02	0,9885959
20	1	20	sse	Logsig	0,0152941	5,75E-03	0,9792451
21	2	3	mse	Tansig	0,4150624	3,20E-01	0,6970621
22	2	5	mse	Tansig	2,94E-02	3,63E-02	0,9648408
23	2	10	mse	Tansig	0,0289882	6,96E-02	0,9798604
24	2	15	mse	Tansig	1,93E-02	3,19E-02	0,9736675
25	2	20	mse	Tansig	0,1490659	2,70E-01	0,7516705
26	2	3	sse	Tansig	4,82E-02	1,01E-01	0,9323196
27	2	5	sse	Tansig	0,6722247	4,84E-01	-0,4615072
28	2	10	sse	Tansig	0,0271754	1,17E-02	0,9862021
29	2	15	sse	Tansig	2,38E-01	5,10E-01	0,518627
30	2	20	sse	Tansig	0,9129583	6,44E-01	0,4259848
31	2	3	mse	Logsig	0,5354022	4,84E-01	-0,7636387
32	2	5	mse	Logsig	4,43E-01	2,00E-01	0,113976
33	2	10	mse	Logsig	2,24E-01	3,28E-01	0,937018
34	2	15	mse	Logsig	0,0103324	2,67E-02	0,9814665
35	2	20	mse	Logsig	4,10E-03	1,41E-02	0,9939005
36	2	3	sse	Logsig	0,4622654	5,44E-01	0,1652355
37	2	5	sse	Logsig	4,72E-01	6,42E-01	0,7279242
38	2	10	sse	Logsig	2,13E-02	1,04E-02	0,9726492
39	2	15	sse	Logsig	0,0453193	7,07E-02	0,9612236
40	2	20	sse	Logsig	1,67E-01	2,71E-01	0,7124466

* Equal number of neurons were used in hidden layers.

APPENDIX-3 : Elman models performance criteria and performances

Performance Criteria					Performance			
Number	Number of Hidden Layers	Hidden Layer Neuron Number*	Performance Criteria	Activation Function	Validation R	Test R	real mse	real R
1	1	3	mse	Tansig	0,96302975	0,976154	0,06598	0,996676
2	1	5	mse	Tansig	0,9841746	0,958703	0,434259	0,676139
3	1	10	mse	Tansig	0,95040885	0,849727	0,072684	0,995264
4	1	15	mse	Tansig	0,74147134	0,840454	0,965989	0,765272
5	1	20	mse	Tansig	0,95652908	0,845613	0,987903	0,999982
6	1	3	sse	Tansig	0,99870724	0,987228	0,35918	0,962672
7	1	5	sse	Tansig	0,99501259	-0,04966	7,044617	0,716149
8	1	10	sse	Tansig	0,99619036	0,921465	0,389532	0,929423
9	1	15	sse	Tansig	0,98937922	0,944076	1,593072	-0,90642
10	1	20	sse	Tansig	0,99951453	0,968085	0,925115	-0,0718
11	1	3	mse	Logsig	0,99928027	0,978023	0,233612	0,997891
12	1	5	mse	Logsig	0,93164446	0,975118	0,217256	0,929617
13	1	10	mse	Logsig	-0,22762213	-0,74699	2,91974	-0,9966
14	1	15	mse	Logsig	0,92759715	0,993152	0,706616	0,77188
15	1	20	mse	Logsig	0,99999903	0,979383	0,256421	0,887658
16	1	3	sse	Logsig	0,99998344	0,952396	2,20734	-0,45993
17	1	5	sse	Logsig	0,95999516	0,935946	0,397897	0,986799
18	1	10	sse	Logsig	0,98108981	0,956339	0,470173	0,819587
19	1	15	sse	Logsig	1	0,983898	1,147519	0,149097
20	1	20	sse	Logsig	0,99840168	0,991661	0,504633	0,995972
21	2	3	mse	Tansig	0,52390451	0,652375	0,562799	0,785076
22	2	5	mse	Tansig	0,98568705	0,912315	0,202071	0,962862
23	2	10	mse	Tansig	0,99679945	0,86107	2,615457	-0,99642
24	2	15	mse	Tansig	0,98003776	0,958726	0,351913	0,841547
25	2	20	mse	Tansig	0,99379117	0,695846	1,651657	-0,2775
26	2	3	sse	Tansig	0,95359182	0,812278	0,321299	0,769092
27	2	5	sse	Tansig	-0,99448346	-0,73409	1,090102	-0,99097
28	2	10	sse	Tansig	0,99882074	0,990251	0,443671	0,991433
29	2	15	sse	Tansig	0,99099251	0,274368	1,906336	-0,43284
30	2	20	sse	Tansig	0,27218582	0,417699	0,61044	0,666177
31	2	3	mse	Logsig	-0,99928015	-0,98147	1,013641	-0,7604
32	2	5	mse	Logsig	-0,93600942	0,592787	0,7815	-0,82291
33	2	10	mse	Logsig	0,93967867	0,958672	0,625597	0,634197
34	2	15	mse	Logsig	0,96074191	0,976354	0,241451	0,997484
35	2	20	mse	Logsig	0,99994955	0,982966	0,715346	0,824194
36	2	3	sse	Logsig	-0,94308041	-0,26505	0,862821	-0,91513
37	2	5	sse	Logsig	-0,99461639	0,741336	1,113066	-0,54906
38	2	10	sse	Logsig	0,99930963	0,983585	0,357979	0,891222
39	2	15	sse	Logsig	0,96515024	0,941572	0,755646	0,502974
40	2	20	sse	Logsig	0,9044666	0,818416	0,327524	0,809412

* Equal number of neurons were used in hidden layers.

APPENDIX-4 : RBF models performance criteria and performances

Performance Criteria			Performance					
Number	Number of Hidden Layers	Performance Criteria	Train mse	Test mse	Train R	Validation R	real mse	real R
1	1	mse	1,79E-31	0,1608297	1	0,812822928	1,5925928	0,991012

APPENDIX-5 : NARX models performance criteria and performances

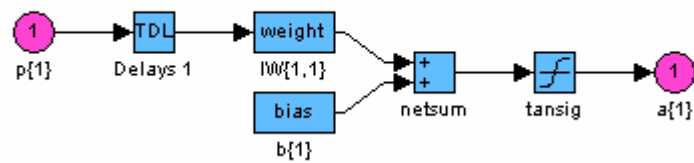
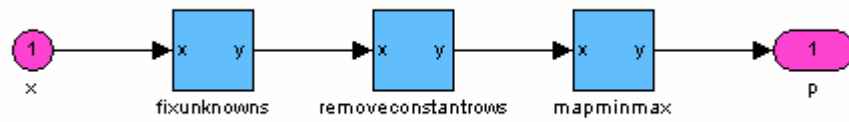
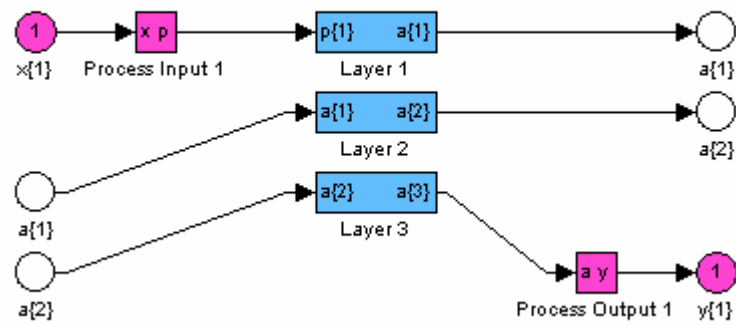
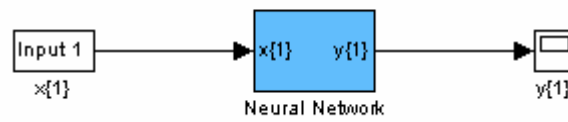
Number	Number of Hidden Layers	Hidden Layer Neuron Number*	Performance Criteria	Activation Function	Train R	Narx error	narxR
1	1	3	sse	tansig	0,99905	2382539,8	0,98921
2	1	5	sse	tansig	0,99905	2,38E+06	0,98921
3	1	10	sse	tansig	0,99899	2511677,3	0,98662
4	1	15	sse	tansig	0,999	2510874	0,98661
5	1	20	sse	tansig	0,999	2508677,6	0,98661
6	1	25	sse	tansig	0,999	2509864,5	0,98659
7	1	30	sse	tansig	0,999	2509559,2	0,98659
8	1	40	sse	tansig	0,999	2509166	0,98659
9	1	50	sse	tansig	0,99941	1,48E+06	0,98826
10	1	60	sse	tansig	0,999	2508725	0,98658

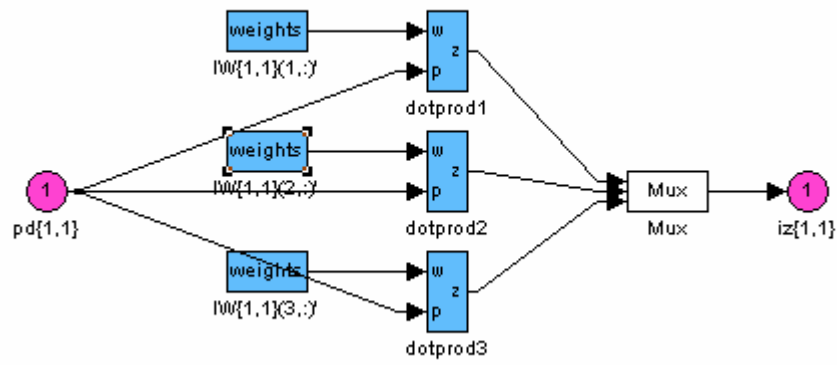
* Equal number of neurons were used in hidden layers.

APPENDIX-6: Correlations among variables

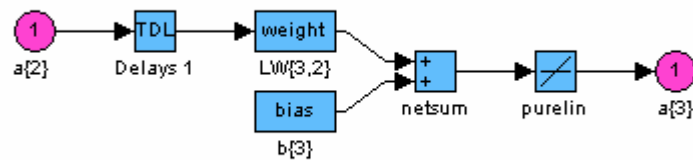
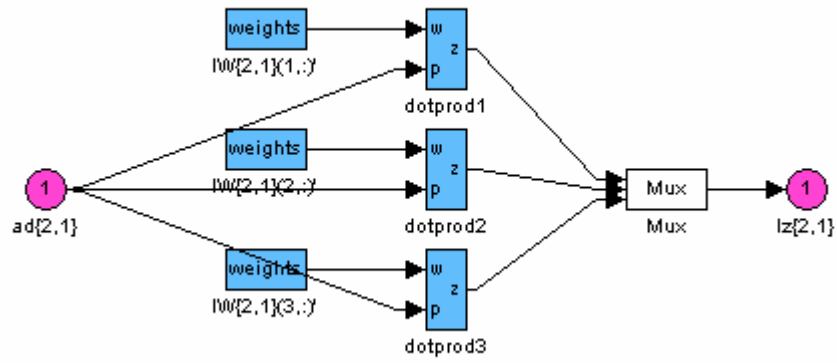
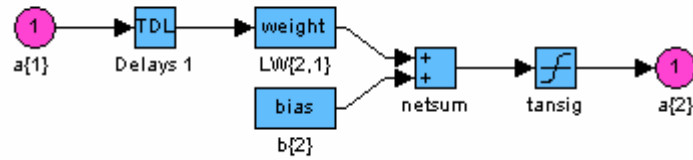
Correlations: Year; P; IP; GNP; EEI; EEE; D; EEP; IEEC; AP; TEEP; GDP; CPI										
	Year	P	IP	GNP	EEI	EEE	D	EEP	IEEC	
P	0,999 0,000									
IP	0,968 0,000	0,970 0,000								
GNP	0,980 0,000	0,980 0,000	0,980 0,000							
EEI	0,455 0,005	0,459 0,005	0,361 0,031	0,387 0,020						
EEE	0,724 0,000	0,715 0,000	0,803 0,000	0,783 0,000	-0,007 0,967					
D	0,700 0,000	0,703 0,000	0,813 0,000	0,720 0,000	0,424 0,010	0,642 0,000				
EEP	0,959 0,000	0,962 0,000	0,989 0,000	0,983 0,000	0,424 0,010	0,781 0,000	0,820 0,000			
IEEC	0,983 0,000	0,984 0,000	0,991 0,000	0,994 0,000	0,416 0,012	0,770 0,000	0,776 0,000	0,993 0,000		
AP	-0,024 0,887	-0,033 0,847	-0,203 0,236	-0,113 0,510	-0,028 0,869	-0,177 0,301	-0,566 0,000	-0,251 0,139	-0,168 0,327	
TEEP	0,957 0,000	0,959 0,000	0,979 0,000	0,986 0,000	0,406 0,014	0,773 0,000	0,776 0,000	0,993 0,000	0,989 0,000	
GDP	0,967 0,000	0,965 0,000	0,956 0,000	0,993 0,000	0,357 0,032	0,768 0,000	0,664 0,000	0,960 0,000	0,977 0,000	
CPI	0,382 0,021	0,382 0,021	0,223 0,190	0,309 0,066	0,192 0,262	0,044 0,797	-0,201 0,241	0,218 0,201	0,276 0,103	
AP TEEP GDP										
TEEP	-0,244 0,152									
GDP	-0,063 0,715	0,968 0,000								
CPI	0,452 0,006	0,255 0,134	0,332 0,048							
Cell Contents: Pearson correlation P-Value										

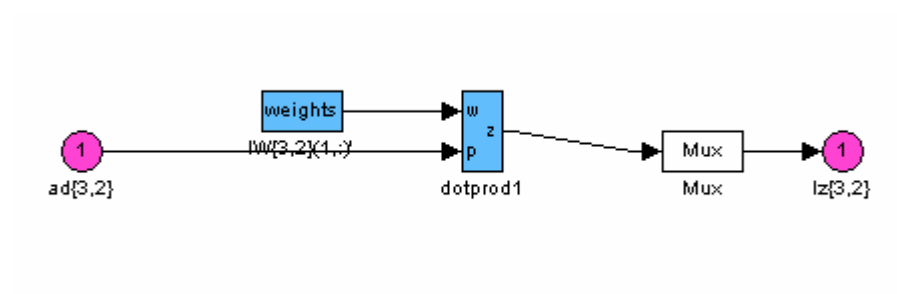
APPENDIX-7: MLP-21 model structure





APPENDIX-7: MLP-21 model sructure





CURRICULUM VITAE

Tuğba YILDIRIM was born in Ordu in 1982. After completing her primary education in Ordu at Merkez Primary School, she attended to Ordu Anatolian High School.

She graduated from the Electrical Engineering Department of Yıldız Technical University in 2004. She started her graduate studies in the Engineering Management Department of Marmara University in 2005. She has been working at İSKİ as electrical engineer for four years. Her research interests are forecasting techniques and energy systems.