Artificial Neural Networks Model For Air Quality in The Region of İzmir

By
Savaş BİRGİLİ

A Dissertation Submitted to the
Graduate School in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

Department: Environmental Engineering
Major : Environmental Engineering
(Environmental Pollution and Control)

İzmir Institute of Technology
İzmir, Turkey

We approve the thesis of **Savaş BİRGİLİ**

Date of Signature

.....

Assoc. Prof. Dr. Gökmen TAYFUR — **23.09.2002**
Supervisor
Department of Civil Engineering

.....

Assist. Prof. Dr. Aysun SOFUOĞLU — **23.09.2002**
Co-Supervisor
Department of Chemical Engineering

.....

Assist. Prof. Dr. Fikret İNAL — **23.09.2002**
Department of Chemical Engineering

.....

Assist. Prof. Dr. Sedat AKKURT — **23.09.2002**
Department of Mechanical Engineering

.....

Assist. Prof. Dr. Serhan ÖZDEMİR — **23.09.2002**
Department of Mechanical Engineering

.....

Assist. Prof. Dr Selahattin YILMAZ — **23.09.2002**
Head of Interdisciplinary
Environmental Engineering (Environmental Pollution and Control)

ACKNOWLEDGEMENT

I would like to express my sincere thanks to Assoc. Prof. Dr. Gökmen Tayfur for his guidance and valuable helps throughout the research. I also would like to thank Asst. Prof Dr. Aysun Sofuoğlu and Asst. Prof Dr. Fikret İnal for their helps and advises.

Special thanks go to Sibel Üren for her valuable helps.

I would like to express my grateful thanks to my colleagues from Environmental Engineering Department and especially my friends Kaya Hünler, and O. Emin Yüreklitürk.

Finally I want to thank to my all family members for their unlimited supports in all my life.

ABSTRACT

Artificial neural networks are nonlinear mapping structures are shown to be universal and highly flexible function approximators for the cases, especially where the underlying data relationships are unknown. Feed-forward artificial neural networks that are trained with the back-propagation algorithm are a useful tool for modeling environmental systems. Back-propagation networks employ a modeling philosophy that unknown model parameters (i.e. connection weights) are optimized in order to obtain the best match between a historical set of model inputs and corresponding outputs.

In this study, a systematic approach to the development of the artificial neural networks based forecasting model is presented. SO_2, and dust values are predicted with different topologies, inputs and transfer functions. Temperature and wind speed values are used as input parameters for the models. The back-propagation learning algorithm is used to train the networks. R^2 (correlation coefficient), and daily average errors are employed to investigate the accuracy of the networks. MATLAB 6 neural network toolbox is used for this study. The study results indicate that the neural networks are able to make accurate predictions even with the limited number of parameters. Results also show that increasing the topology of the network and number of the inputs, increases the accuracy of the network. Best results for the SO_2 forecasting are obtained with the network with two hidden layers, hyperbolic tangent function as transfer function and three input variables (R^2 was found as 0,94 and daily average error was found as 3,6 $\mu g/m^3$). The most accurate results for the dust forecasting are also obtained with the network with two hidden layer, hyperbolic tangent function as transfer function and three input variables (R^2 was found as 0,92 and daily average error was found as 3,64 $\mu g/m^3$).

SO_2 and dust predictions using their last seven days values as an input are also studied, and R^2 is calculated as 0,94 and daily average error is calculated as 4,03 $\mu g/m^3$ for SO_2 prediction and R^2 is calculated as 0,93 and daily average error is calculated as 4,32 $\mu g/m^3$ for dust prediction and these results show that the neural network can make accurate predictions.

ÖZ

Yapay sinir ağları özellikle veriler arasındaki ilişkilerin tamamıyla bilinemediği durumlarda her türlü verilerle uygulanabilme özelliklerinden dolayı çok kullanışlı olan doğrusal olmayan modelleme sistemleridir. Geri yayılma algoritması ile eğitilen ileri beslemeli yapay sinir ağları, çevresel sistemlerin modellenmesinde sık kullanılan ve iyi sonuçlar verebilen bir yöntemdir. Geriye yayılma algoritması, bilinmeyen model değişkenlerinin (bağlantı ağırlıkları gibi), sistem girdileriyle çıktıları arasındaki en iyi uyumu sağlamak üzere ayarlanması prensibinden hareketle geliştirilmiştir.

Bu çalışma, yapay sinir ağlarının kullanılması ile SO_2 ve toz değerlerinin farklı ağ topolojileri, değişik sayıda girdi setleri ve farklı tasvir fonksiyonlarının kullanılması ile öngörülmesi üzerinedir. Günlük sıcaklık ve rüzgar hızı değerleri de modellerde girdi olarak kullanılmıştır. Yapay sinir ağlarının eğitilmesinde geri yayılma algoritması kullanılmıştır. Öngörülerin doğruluklarının incelenmesi amacıyla R^2 (korelasyon katsayısı) ve ortalama günlük hata değerleri kullanılmıştır. Çalışmadan elde edilen sonuçlara göre, yapay sinir ağları kısıtlı verilerin kullanılması durumunda bile başarılı öngörüler yapabilmektedir. Ağ topolojilerinin geliştirilmesi ve girdi sayılarının arttırılması ile öngörülerin kesinliklerinin de arttığı gözlenmiştir. SO_2 öngörüsüne yönelik çalışmalarda en iyi sonuçlar iki gizli katmanın bulunduğu ağlarda, hiperbolik tanjant fonksiyonunun tasvir fonksiyonu olarak seçildiği ve toz, sıcaklık ve rüzgar hızı değerlerinin girdi olarak kullanıldığı sistemde elde edilmiştir (bu çalışmanın sonuçlarında R^2 0,94 ve ortalama günlük hata ise 3,6 $\mu g/m^3$ olarak bulunmuştur). Toz öngörüsüne ait en iyi sonuçlar da iki gizli katmanlı ağlarda, tasvir fonksiyonu olarak hiperbolik tanjant fonksiyonunun kullanıldığı ve SO_2, sıcaklık ve rüzgar hızı değerlerinin girdi olarak kullanıldığı sistemde elde edilmiştir (bu çalışmanın sonuçlarında R^2 0,92 ve ortalama günlük hata ise 3,64 $\mu g/m^3$ olarak bulunmuştur).

SO_2 ve toz değerlerinin öngörüsüne yönelik bir diğer çalışmada ise öngörüsü yapılacak değişkenlerin son yedi günlük değerleri girdi olarak kullanılmış ve sekizinci güne ait değerin öngörülmesine çalışılmıştır. SO_2 değerlerinin öngörüsü için R^2 0,94 ve ortalama günlük hata ise 4,03 $\mu g/m^3$ olarak bulunmuştur. Toz değerlerinin öngörü çalışmasında ise R^2 0,93 ve ortalama günlük hata ise 4,32 $\mu g/m^3$ olarak hesaplanmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Urban air pollution has emerged as the most acute problem in recent years because of its detrimental effects on health and living conditions. To prevent any further decline in air quality, scientific planning of analytical methods and pollution control is required. Within this framework it is necessary to analyze and specify all pollution sources and their contribution to air quality, to study the various factors which cause the pollution phenomenon, and to develop tools for reducing pollution by introducing alternatives to existing practices.

In most of the studies conducted on atmospheric dispersion of pollutants, wide-ranging prediction techniques, including Gaussian models and numerical models are generally used. The primary inputs to dispersion models include emission inventory, meteorological data and receptor locations. The major output from these models is the predicted ground-level concentration at specified receptor locations. The models are mainly based on the mathematical formulation of the physics and chemistry of the atmosphere, which govern the dispersion of pollutants.

Mathematical models are based on a fundamental mathematical description of atmospheric processes in which the effects are generated by causes (Zannetti 1983, 1994). Such models aim to resolve the underlying chemical and physical equations that control pollutant concentrations and therefore require detailed emission data and meteorological conditions for the region of interest. There are generally severe limitations in accuracy of the data. In addition, some input data are not easily acquired by environmental protection agencies or local industries. This means that if these inputs are unknown, then the application of the mathematical models is problematic. Therefore, in such cases, it is much more practical to rely on statistical models.

Statistical models are based on semi-empirical statistical relations among available data and measurements. They do not necessarily reveal any relation between cause and effect. They attempt to determine the underlying relationship between sets of input data and targets. Examples of statistical models are correlation analysis (Abdul-Wahab et al., 1996) and time series analysis (Hsu, 1992). However, the complex and sometimes non-linear relationships of multiple variables can make statistical models awkward and complicated (Comrie, 1997). Therefore, it is expected that they will underperform when used to model the relationship between environmental and the other variables that are extremely non-linear.

Artificial neural networks provide an attractive alternative tool for both forecasting researchers and practitioners. Several distinguishing features of artificial neural networks make them valuable and attractive for a forecasting task. As opposed to the traditional model-based methods, artificial neural networks are data-driven self-adaptive methods in that there are few a priori assumptions about the models for problems under study. They learn from examples and capture subtle functional relationships among the data even if the underlying relationships are unknown or hard to describe. Thus artificial neural networks are well suited for problems whose solutions require knowledge that is difficult to specify but for which there are enough data or observations. This modeling approach with the ability to learn from experience is very useful for many practical problems since it is often easier to have data than to have good theoretical guesses about the underlying laws governing the systems from which data are generated.

The purpose of this research is to investigate the modeling of the SO_2 and dust parameters with artificial neural networks for the city center of İzmir. For this purpose meteorological data (temperature and wind speed) are selected as input. These data sets are divided into two groups; training and testing set. The training set is used to train the system and back-propagation algorithm is selected as a learning algorithm.

Testing set is presented to the system to produce the output which is referred as predicted outputs and these outputs are compared with the actual (desired) values.

In Chapter 2, fundamental principles of artificial neural networks are described. Historical development of the neural networks and learning rules are also given in this chapter.

In Chapter 3, back-propagation algorithm is given and mathematical basis of the back-propagation algorithm is examined. Training procedure is also described in this chapter.

In Chapter 4, artificial neural networks are studied as a forecasting tool. Model development processes for forecasting and the forecasting procedure are given in this chapter. Literature review of the forecasting the environmental variables is also given in this chapter.

In Chapter 5, sulphur dioxide and dust parameters are explained; their sources, health effects, and hazards on human beings and environment are given. In this chapter, the training and testing sets (which are used for modeling), and the modeling procedure are also given.

In Chapter 6, experimental results of SO_2 and dust predictions are given. According to the results, neural networks are able to make acceptable and accurate predictions. These results are discussed in this section.

Chapter 2

ARTIFICIAL NEURAL NETWORKS

Artificial neural networks are one of the artificial intelligence related technologies. The term "artificial intelligence", in its broadest sense, encompasses a number of technologies that include, but is not limited to, expert systems, neural networks, genetic algorithms, fuzzy logic systems, cellular automata, chaotic systems, and anticipatory systems. Interestingly, most of these technologies have their origins in biological or behavioral phenomena related to human or animal. Consequently many of these technologies are simple analogs of human and/or animal systems.

In 1956 the term "artificial intelligence" came into common use at a conference at Dartmouth College, and it is defined as "Computer process that attempts to emulate the human thought process which is associated with activities that require the use of intelligence".

2.1. Historical Development of Neural Networks

It is possible to say that first studies of neural networks have been started at the beginning of the 1940s. McCulloch and Pits (1943) designed what are generally regarded as the first neural networks. These researchers recognized that combining many simple neurons into neural system was the source of increased computational power. The flow of information through the net assumes a unit time step for a signal to travel from one neuron to the next. This time delay allows the net to model some physiological processes, such as the perception of hot and cold.

At the and of 1940's Hebb who is a psychologist at McGill University designed the first learning law for artificial neural networks.

His premise was that if two neurons were active simultaneously, then the strength of the connection between them should be increased. Refinements were subsequently made to this rather general statement to allow computer simulations (Rochester et al., 1956).

Together with several other researchers Block (1962), Minsky and Papert (1969), Rosenblatt (1958, 1959, 1962) introduced and developed a large class of artificial neural networks called "perceptrons". The perceptron learning rule uses an iterative weight adjustment that is more powerful than the Hebb rule.

Widrow and Hoff (1960), developed a learning rule that is closely related to the perceptron learning rule. The delta rule adjusts the weights to reduce the difference between the net input to the output unit and the desired output. This results in the smallest mean squared error. The Widrow-Hoff learning rule for a single-layer network is a precursor of the backpropagation rule for multi-layer networks.

During the 1970's the early work of Kohonen (1972), dealt with associative memory neural networks. His more recent work (Kohonen, 1982) has been the development of self-organizing feature maps that use a topological structure for the cluster units. These networks have been applied to speech recognition (Kohonen et al., 1987).

Two of the reasons for the quite years of the 1970's were the failure of single-layer perceptrons to be able to solve such simple problems (mappings) as the X_{OR} function and the lack of a general method of training a multi-layer net. A method for propagating information about errors at the output units back to the hidden units had been discovered in the previous decade (Werbos, 1974) but had not gained wide publicity. This method discovered by Parker (1985) and by LeCun (1986) before it became widely known. Parker's work came to attention of the Parallel Distributed Processing Group led by psychologist David Rumelhart of the University of the California and James McClelland of Carneigie-Mellon University who refined and publicized it.

A Nobel prize winner John Hopfield has developed a number of neural networks based on fixed weights and adaptive activations (Hopfield, 1982, 1984; Hopfield and Tank, 1985, 1986, 1987). These networks can serve as associative memory nets and can be used to solve constraint satisfaction problems such as the "Traveling Salesman Problem".

2.2. Fundamentals of Neural Networks

Artificial neural networks are computer programs that emulate biological neural networks. In order to process vague, noisy, or incomplete information, researchers are turning to biological neural systems as a model for a new computing paradigm.

Artificial neural systems are unlike traditional programming methods (Wasserman, 1993). Generally, traditional methods use deductive reasoning to apply known rules to situations to produce outputs. Each new situation may require that another rule be implemented. The programs can become quite large and complicated in an attempt to address all possible situations. Artificial neural systems, however, automatically construct associations based upon the results of known situations. For each new situation, the neural system automatically adjusts itself and eventually generalizes these new situations.

Neural networks are a form of artificial intelligence related techniques. They are massively parallel systems that rely on dense arrangements of interconnections and simple processors. Artificial neural networks take their name from the networks of nerve cells in the brain (Nelson et al., 1994). They are motivated by the neural architecture and operation of the brain. Although a great deal of biological detail is eliminated in these computing models, the artificial neural networks retain enough of the structure observed in the brain to provide insight into how biological neural processing may work.

2.2.1. Biological Basis of Neural Networks

The human brain is a very complex system capable of thinking, remembering, and problem solving. The fundamental cellular unit of the brain's nervous system is a "neuron". It is a simple processing element that receives and combines signals from other neurons through input paths called "dendrites". If the combined input signal is strong enough, it activates the neuron to produce an output signal along the axon that connects to the dendrites of many other neurons. Figure 2.1 is a sketch of a neuron showing the various components.

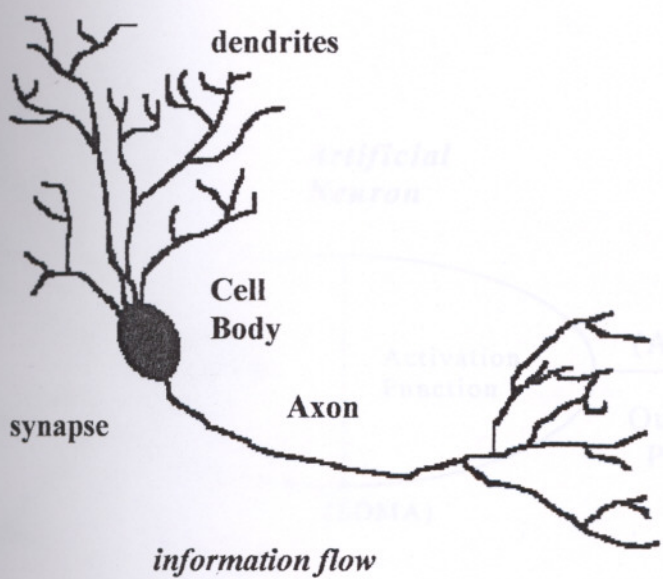


Figure 2.1 A biological neuron and components (Soucek, 1989)

2.2.2. Artificial Neurons

An artificial neuron is a model whose components have direct analogs to components of an actual neuron. Figure 2.2 shows the schematic representation of an artificial neuron. The input signals are represented by x_0, x_1,....., x_n. These signals are continuous variables, not the discrete electrical pulses that occur in the brain. Each of these input is modified by weight (w_0, w_1 ....., w_n). These weights can be either positive or negative, corresponding to acceleration or inhibition of

the flow of electrical signals. This processing element consist of two parts (Tsoukalas and Uhrig, 1997). The first part simply aggregates (sums) the weighted inputs resulting in a quantity I; the second part is effectively a nonlinear filter, usually called the "activation function" through which the combined signal flows.

$$I_j = \sum_{i=1}^{n} w_{ij} x_i \qquad \text{Sum of weighted input} \qquad (2.1)$$

$$y_j = \varphi(I_j) \qquad \text{Activation Function} \qquad (2.2)$$

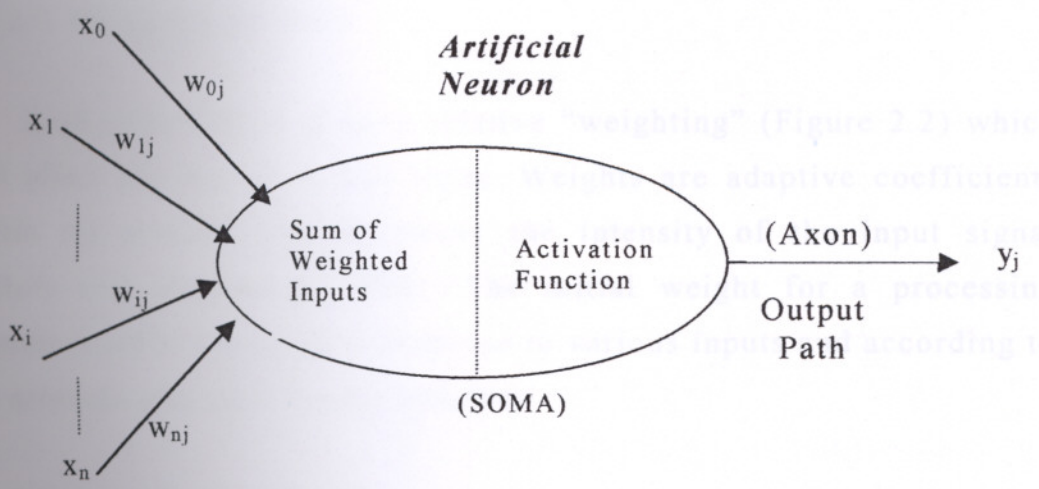


Figure 2.2 Schematic representation of an artificial neuron (Tsoukalas and Uhrig, 1997)

2.2.3. The Basic Components of Artificial Neurons

An artificial neuron (Figure 2.2) can also be called as a processing element, and the processing element handles several basic functions. A processing element can evaluate the input signals, and after determining the strength of the each input signal, it can compare the threshold level with the summed input. Finally it can determine the output according to this comparison.

2.2.3.1. Inputs and Outputs

Just as there are many inputs to a biological neuron, there should be many input signals to the processing element and all of them should come simultaneously. Depending on the threshold level processing element could be activated and then it may produce or not produce an output.

In addition, just as real neurons are affected by things other than inputs, some networks provide a mechanism for other influences. This extra input is called as a "bias term" or a "forcing term". (Lek and Guegan, 1999).

2.2.3.2. Weighting Factors

Each input will be given a relative "weighting" (Figure 2.2) which will affect the impact of that input. Weights are adaptive coefficients within the network that determine the intensity of the input signal (Nelson and Illingworth, 1994). The initial weight for a processing element could be modified in response to various inputs and according to the networks own rules for modification.

2.2.3.3. Transfer Functions

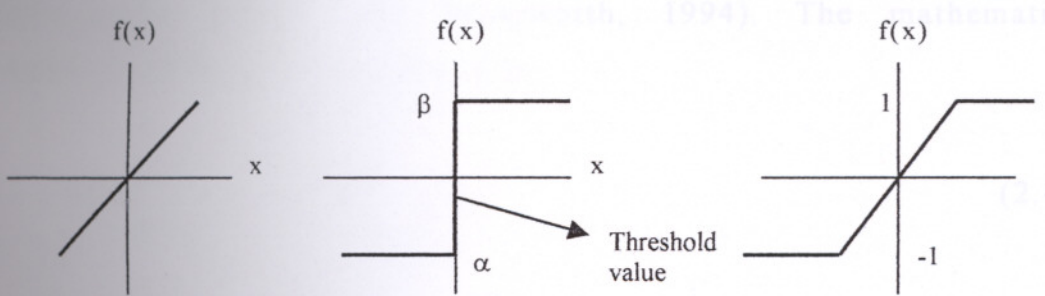
The relationship between inputs and outputs at any instant is specified by the transfer or the activation function. The sum of the weighted inputs becomes the input to the transfer function, which specifies the output from the particular processing element. A number of transfer functions are commonly used. Typically they are non-linear and Figure 2.3 reviews the transfer functions which are generally used in neural network studies.

• **Linear Function:** The linear transfer function (Figure 2.3a) calculates the neuron's output by simple equation (where α is a scalar) :

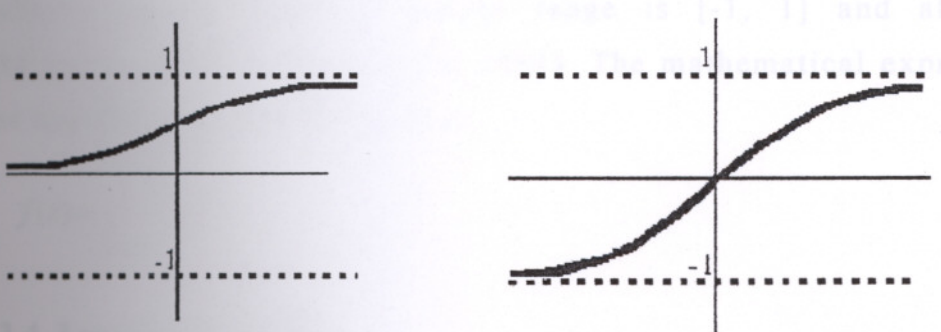
$$f(x) = \alpha.x. \quad (2.3)$$

This neuron can be trained to find a linear approximation to a nonlinear function. A linear network cannot be made to perform a nonlinear computation (Zurada, 1992).

• **Step (Hard Limiter) Function:** The hard limit transfer function forces a neuron to output a β if its net input reaches a threshold, otherwise it outputs α (Figure 2.3b). This allows a neuron to make a decision or classification (Tsoukalas and Uhrig, 1997). It can say yes or no. This kind of neuron is often trained with the perceptron learning rule, and generally parameters are chosen as $\beta = 1$ and $\alpha = 0$ or 1 in the literature.



a-) linear function **b-) step (hard limiter function)** **c-) ramping function**



d-) sigmoid function **e-) hyperbolic tangent function**

Figure 2.3 Sample Transfer Functions (Bose and Liang, 1996)

• **Ramping Function:** For inputs less than -1 ramping function (Figure 2.3c) produces -1. For inputs in the range -1 to +1 it simply returns to the linear function. For inputs greater than +1 it produces +1, but this function is not a continuous function at the intersection points (Tsoukalas and Uhrig, 1997). This network can be tested with one or more input vectors which are presented as initial conditions to the network. After the initial conditions are given, the network produces an output which is then fed back to become the input. This process is repeated over and over until the output stabilizes.

• **Sigmoid Function:** The sigmoid transfer function (Figure 2.3d) takes the input, which may have any value between plus and minus infinity, and squashes the output into the range 0 to 1. This transfer function is commonly used in backpropagation networks, in part because it is differentiable (Nelson and Illingworth, 1994). The mathematical expression of the sigmoid function is:

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.4}$$

• **Hyperbolic Tangent Function:** Alternatively, multi-layer networks may use the hyperbolic tangent transfer function (Figure 2.3e). Hyperbolic tangent functions output range is [-1, 1] and also its derivative is continuous (LiMin Fu, 1994). The mathematical expression of the hyperbolic tangent function is :

$$f(x) = \frac{2}{1+e^{-2x}} - 1 \tag{2.5}$$

2.2.3.4. Learning Functions

If a local memory is attached to the processing element, the processing element could restore the previous computations results and modify the weights according to these results. The ability to change the weights is called as "learning" (Engel, 1994).

2.3. Artificial Neural Networks

An artificial neural network can be defined as "a data processing system consisting of a large number of simple, and highly interconnected processing elements in an architecture inspired by the structure of the human brain" (Tsoukalas and Uhrig, 1997).

Once the dynamics of an individual processing element are decided the next step in specifying a neural network is to determine the combination of the individual processing elements. These processing elements are usually organized into a sequence of layers with connections between the layers.

The number of elements needed for the input and output layers will depend on the number of inputs and outputs for the system. Figure 2.4 shows the simple neural network architecture.

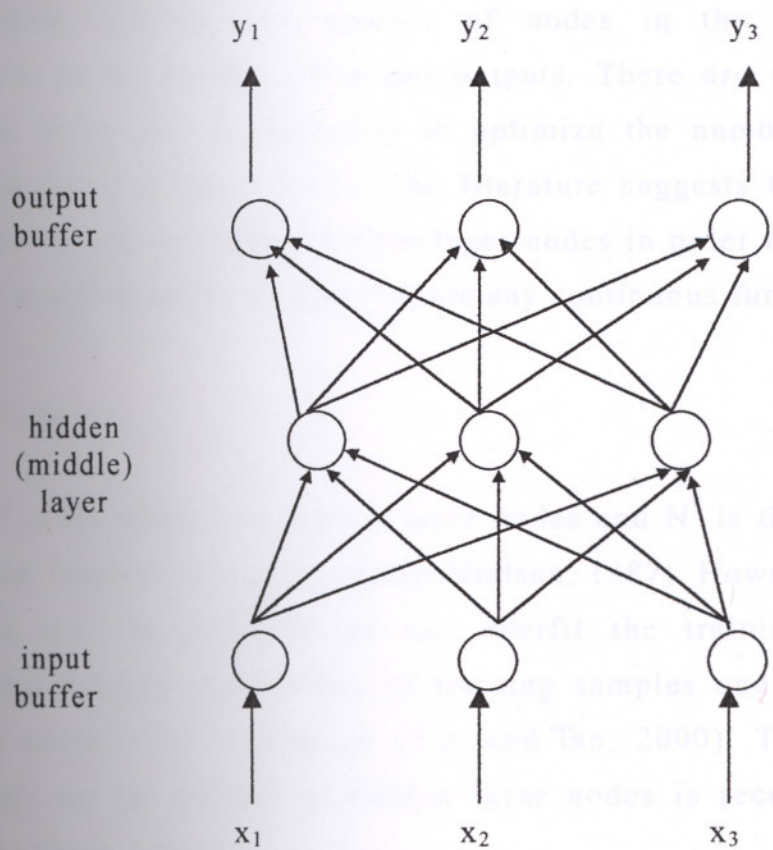


Figure 2.4 Example of an Neural Architecture (Rafiq et. al., 2001)

2.3.1. Architecture (Topology) of Neural Networks

Network geometry is generally defined by the number of hidden layer nodes and the number of nodes in each of these layers. It determines the number of model parameters that need to be estimated. If there is an insufficient number of parameters, it may be difficult to obtain convergence during training, as the network may be unable to obtain an adequate fit to the training data. On the other hand, if too many parameters are used in relation to the number of available training samples, the network may lose its ability to generalize. In addition, keeping the number of parameters to a minimum reduces the computational time needed for training. It has been shown that artificial neural networks with one hidden layer can approximate any continuous function, given sufficient degrees of freedom (Maier and Dandy, 2001).

The number of nodes in the input layer is equal to the number of model inputs, whereas the number of nodes in the output layer corresponds to the number of model outputs. There are some general guidelines which may be followed to optimize the number of hidden layers and nodes in these layers. The literature suggests the following upper limit for the number of hidden layer nodes in order to ensure that artificial neural networks to approximate any continuous function;

$$N^H \leq 2N^I + 1 \qquad (2.6)$$

where N^H is the number of hidden layer nodes and N^I is the number of input nodes (number of inputs) (Hecht-Nielsen, 1987). However, in order to ensure that the networks do not overfit the training data, the relationship between the number of training samples and the network size also needs to be considered (Yao and Tan, 2000). The following upper limit for the number of hidden layer nodes is recommended to satisfy the above criteria;

$$N^H \leq \frac{N^{TR}}{N^I + 1} \qquad (2.7)$$

where N^{TR} is the number of training samples. Consequently, the upper limit for the number of hidden layer nodes may be taken as the smaller of the values for N^H obtained using (3) and (4). However, in many instances, good performance can be obtained with fewer nodes.

The optimum number of hidden layer nodes, on the other hand, generally has been found by using a trial and error approach.

The network architecture is also characterized by the interconnections of nodes in layers. The connections between nodes in a network fundamentally determine the behavior of the network. For most forecasting as well as other applications, the networks are fully connected in that all nodes in one layer are only fully connected to all nodes in the next higher layer except for the output layer.

Neural networks are often classified as single-layer or multi-layer.

- *Single-Layer Neural Networks:* A single-layer neural network has one layer of connection weights. Often, the units can be distinguished as input units, which receive signals from the outside world, and output units, from which the response of the net can be read (Fausett, 1994).

- *Multi-Layer Neural Networks:* A multi-layer neural network is a net with one or more layers of nodes between the input units and the output units. Multi-layer neural networks can solve more complicated problems than single-layer neural networks , but training may be more difficult. Figure 2.4 is an example of an multi-layer neural network architecture (Tsoukalas and Uhrig, 1997).

2.3.2. Learning and Recall

Neural networks perform two major functions; "learning" and "recall". Learning is the process of adapting the connection weights in

an artificial neural network to produce the desired output in response to data presented to the input buffer. Recall is the process of accepting an input stimulus and producing an output response in accordance with the network weight structure (Corchado and Fyfe, 1999).

There are two types of learning; "supervised" and "unsupervised".

• **Supervised learning:** Supervised learning assumes the availability of a teacher or supervisor who classifies the training samples. In this mode the actual output of a neural network is compared to the desired output. Weights, which are generally randomly set to begin with, are then adjusted by the network so that the next iteration, or cycle, will produce a closer match. With supervised learning, it is necessary to train the neural network. Training consist of presenting input and output data to network. This data is generally referred to as the "training set" (Zhang et al., 1997).

• **Unsupervised learning:** Unsupervised learning also called as self-supervised learning. Here, networks use no external influences to adjust their weights. Instead there is an internal monitoring or performance. The network looks for regularities or trends in the input signals, and makes adaptations according to the function on the network (Zhang et al., 1997).

2.3.3. Learning Laws

2.3.3.1 Hebbian Learning without a Teacher

The first learning rule was introduced by Hebb (1949) as :

$$\Delta w_{ij} = \eta a_i o_j \qquad (2.8)$$

where η is the constant of proportionally representing the learning rate; o_i is output from unit j, and is connected to the input of unit i through

the weight w_{ij}; a_i is the state of activation and the output o_j is a function of the activation state . According to this rule, where unit i and j are simultaneously excited, the strength of the connection between them increases in proportion to the product of their activations.

2.3.3.2. The Delta Rule (Widrow-Hoff Rule) with a Teacher

This rule is based on the simple idea of continuously modifying the strengths of the connections to reduce the difference (the delta) between the desired output and the current output. This learning rule is also referred as least mean square (LMS) learning rule because it minimizes the mean squared error (Spellman, 1999).

$$\Delta w_{ij} = \eta[t_j - y_j]x_i \qquad (2.9)$$

where η is the learning rate, x as training input, t is the target output for the input x.

2.3.3.3. The Kohonen Learning Rule without a Teacher

This rule was inspired by learning in biological systems. In this procedure, the processing elements compete for the opportunity of learning. The processing element with the largest output is declared the winner and has the capability of inhibiting its competitors as well as exciting its neighbors; for this reason, sometimes this rule is also referred as the competitive learning rule (Bose and Liang, 1996).

$$w_{new} = w_{old} + \eta(x - w_{old}) \qquad (2.10)$$

where x is the input vector, w_{new} is the new weight factor and η is the learning rate.

2.3.3.4 The Hopfield Minimum-Energy Rule

Hoppfield's study concentrates on the units that are symmetrically connected. The units are always in one of two states: +1 or –1. The

global energy of the system is defined as:

$$E = -\sum_{i<j} w_{ij} s_i s_j + \sum \theta_i s_i \qquad (2.11)$$

$$\Delta E_k = \sum_i w_{ki} s_i - \theta_k \qquad (2.12)$$

where s_i is the state of the ith unit (-1 or 1), θ_i is the threshold, and ΔE_k is the difference between the energy of the whole system with the kth hypothesis false and its energy with the kth hypothesis true (Bose and Liang, 1996).

2.3.3.5. The Boltzmann Learning Rule

The Boltzmann learning algorithm is designed for a machine with symmetrical connections. The binary threshold in a perceptron is deterministic, but in a Boltzmann machine it is probabilistic:

$$p_i = \frac{1}{1+e^{-\Delta E_i/T}} \qquad (2.13)$$

where p_i is the probability for the ith unit to be in state 1, P(x) is a sigmoidal probability function, T is a parameter analogous to temperature and measures the noise introduced into the decision; and $\Delta E_i = \sum w_{ij} s_j$ is the total input to the unit. This rule has been designed to solve a class of optimization problems in vision. The learning is supervised and the energy gradient with respect to w_{ij} depends on the behavior of only the ith and jth units (Reich et al., 1999).

2.3.3.6. The Back-Propagation Learning

The back propagation of errors technique is the most commonly used generalization of the Delta Rule. This procedure involves two phases. The first phase, called the "forward phase", occurs when the input is presented and propagated forward through the network to compute an output value for each processing element. For each

processing element, all current outputs are compared with the desired output, and the difference, or error is computed (Bose and Liang, 1996).

In the second phase, called the "backward phase", the recurrent difference computation (from the first phase) is performed in a backward direction. Only when these two phases are complete can new inputs be presented.

Chapter 3

BACK-PROPAGATION ALGORITHM

Back-propagation is a systematic method for training multiple (three or more) layer artificial neural systems. The elucidation of this training algorithm in 1986 by Rumelhart et al., was the key step in making neural networks practical in many real-world situations. However it was developed independently by Parker in 1982. This method is simply a gradient descent method to minimize the total squared error of the output computed by the net.

Back-error propagation is the most widely used of the neural network paradigms and has been applied successfully in applications in a broad range of areas.

Back-propagation network is usually layered, with each layer fully connected to the layers below and above. When the network is given an input, the updating of activation values propagates forward from the input layer of processing units, through each internal layer, to the output layer of processing units. The output units then provide the network's response. When the networks corrects its internal parameters, the correction mechanism starts with the output units and back-propagates backward through each internal layer to the input layer. Hence, it is named as "back-error propagation", or "back-propagation".

The power of back-propagation lies in its ability to train hidden layers. The network has middle or hidden layers of processing units. Each hidden layer acts as a layer of "feature detectors" which are units that respond to specific features in the input pattern. These feature detectors organize as learning takes place, and are developed in such a way that they accomplish the specific task presented to the network.

3.1. Network Structure

Back-propagation employs three or more layers of processing units. Figure 3.1 shows the topology for a typical three-layer back-propagation network. The bottom layer of units is the input layer (the only units in the network that receive external input). The layer above is the hidden layer, in which the processing units are interconnected to layers and below. The top layer is the output layer.

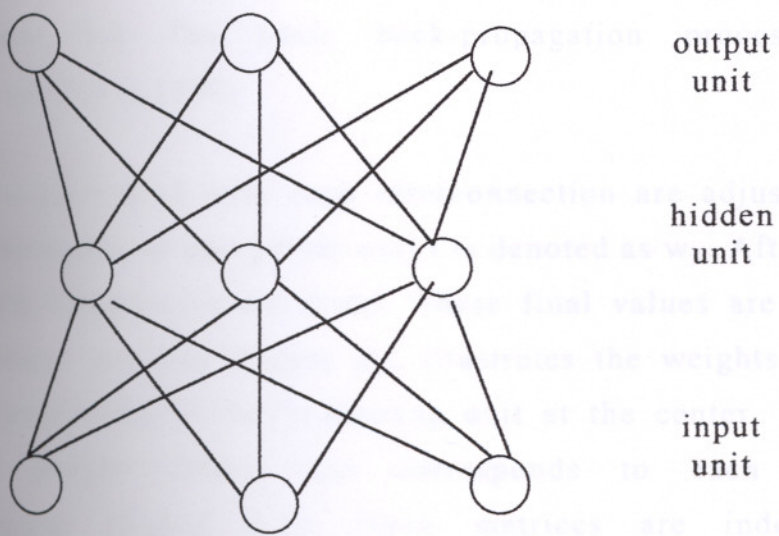


Figure 3.1 A three-layered back-propagation network, fully interconnected (Tsoukalas and Uhrig, 1997)

The layers in Figure 3.1 are fully interconnected (each processing unit is connected to every unit in the layer above and in the layer below). Units are not connected to other units in the same layer.

Back-propagation networks do not have to be fully interconnected, although most applications have been done with fully interconnected layers.

In Figure 3.2, a basic back-propagation processing unit is shown. Inputs are at the left, and outputs are at the right. The processing unit has a weighted sum of inputs (S_j), an output value (a_j), and an associated error value (δ_j) that is used during weight adjustments.

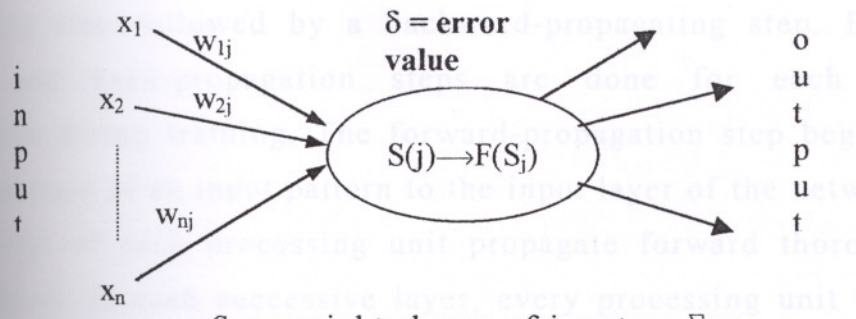


Figure 3.2 The basic back-propagation processing unit (Tsoukalas and Uhrig,1997)

Weights associated with each interconnection are adjusted during learning. The weight to unit j from unit i is denoted as w_{ji}. After learning is completed, the weights are fixed. These final values are then used during "testing" sessions. Figure 3.2 illustrates the weights along the incoming connections to the processing unit at the center. There is a matrix of weight values that corresponds to each layer of interconnections (Figure 3.3); these matrices are indexed with superscripts to distinguish in different layers.

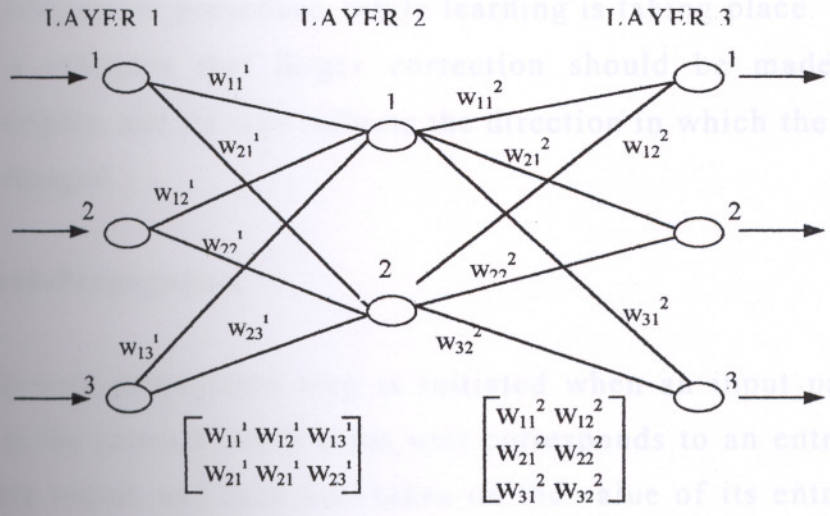


Figure 3.3 Weight matrices of a three-layered back-propagation system (Zurada, 1992)

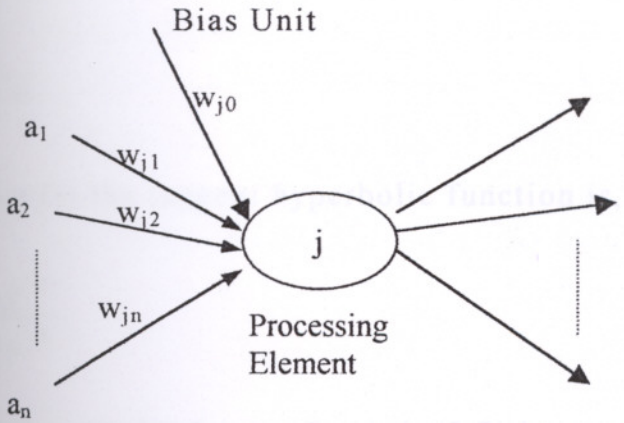
The back-propagation learning algorithm involves a forward-propagating step followed by a backward-propagating step. Both the forward and back-propagation steps are done for each pattern presentation during training. The forward-propagation step begins with the presentation of an input pattern to the input layer of the network, the output value of each processing unit propagate forward thorough the hidden layers. In each successive layer, every processing unit sums its inputs and then applies a transfer function (sigmoid or tangent hyperbolic function in this study) to compute its output. The output layer of units then produces the output of the network.

The backward-propagation step begins with the comparison of the network's output pattern to the target data, when the difference, or "error" is calculated. The backward-propagation step then calculates error values for hidden units and changes incoming weights, starting with the output layer and moving backward through the successive hidden layers. In this back-propagating step, the network corrects its weights in such a way as to decrease the observed error.

The error value (δ) associated with each processing unit reflects the amount of error associated with that unit. This parameter is used during the weight-connection procedure, while learning is taking place. A larger value for δ indicates that larger correction should be made to the incoming weights, and its sign reflects the direction in which the weights should be changed.

3.2. Forward-Propagation

The forward-propagation step is initiated when an input pattern is presented to the network. Each input unit corresponds to an entry in the input pattern vector, and each unit takes on the value of its entry. After the activation level for the first layer of units is set, the remaining layers perform a forward-propagation step, which determines the activation levels of the other layers of units.



$$S_j = \sum a_j w_{ji}$$

$$\text{output} = f(S_j)$$

Figure 3.4 The forward-propagation step (Soucek, 1989)

Figure 3.4 illustrates the specifics of the forward-propagation step. Incoming connections to unit j are at the left and originate at units in the layer below. Output values from these units arrive at unit j and are summed by;

$$S_j = \sum_i a_i w_{ji} \qquad (3.1)$$

where;

a_i = the activation level of unit i,

w_{ji} = the weight from unit i to unit j (unit i is one layer below unit j)

After the incoming sum S_j is computed, a function f is used to compute $f(S_j)$. The function f, sigmoid or tangent hyperbolic functions has already been discussed and illustrated in Figure 2.3d and 2.3e. f(x) approaches 1 asymptotically as x gets larger, and f(x) approaches 0 asymptotically as x becomes a greater negative value for sigmoid function; and f(x) approaches 1 asymptotically as x gets larger, and f(x) approaches -1 asymptotically as x becomes a greater negative value for tangent hyperbolic function.

The equation for the sigmoid function is,

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.2}$$

and the equation for the tangent hyperbolic function is,

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

Since here x is the weighted sum of unit j, (3.2) becomes;

$$f(S_j) = \frac{1}{1+e^{-S_j}} = \frac{1}{1+e^{-\sum a_i w_{ji}}} \tag{3.4}$$

and (3.3) becomes;

$$f(S_j) = \frac{e^{S_j} - e^{-S_j}}{e^{S_j} + e^{-S_j}} = \frac{e^{\sum a_i w_{ji}} - e^{-\sum a_i w_{ji}}}{e^{\sum a_i w_{ji}} + e^{-\sum a_i w_{ji}}} \tag{15}$$

After the transfer functions computed (Sj), the resulting value becomes the activation level of unit j. This value, the output of unit j, is sent along the output interconnections (on the right of Figure 3.4).

The input layer units constitute a special case. These units do not perform the weighted sum on their inputs because each input unit simply assumes the corresponding value taken from the input vector. The input layer is considered to be a layer of the network, although it does not perform the weighted sum and sigmoid calculations.

Some back-propagation networks employ a bias unit (Figure 3.4) as part of every layer except the output layer. Each bias unit is connected to all units in the next higher layer, and its weights to them are adjusted during the back-error propagation. The bias units provide a constant term in the weighted sum of units in the next layer. The result is sometimes an improvement on the convergence properties of the network.

The bias unit also provides a "threshold" effect on each unit it targets. It contributes a constant term in the summation S_j, which is the operand in the transfer functions (3.4) and (3.5). This is equivalent to translating the function's curves to the left or to the right.

For example, suppose the bias unit (a_0) has an output value of 1.0 and a weight,

$$c = w_{j0}$$

then let

$$z = \sum_{i=1} a_i w_{ji} \qquad (3.6)$$

Then z is the incoming sum from all of the units other than the bias unit. If the bias unit contributes the constant c to the incoming sum of unit j, then this sum becomes, "z + c". Constant c translates the graph by the amount c, thus moving the threshold of the sigmoid or tangent hyperbolic curve from 0 to –c. In this way, the bias units provide an adjustable threshold for each target unit. The threshold for unit j then comes from the value of w_{j0}, which is the weight of the interconnection from the bias unit.

3.3. Backward Propagation

Figure 3.5 illustrates the back propagation network. The error values are calculated for all processing units and weight changes are calculated for all interconnections. The calculations begin at the output layer and progress backward through the network to the input layer. The calculated errors propagate backward through the network, where they are used to adjust the weights.

The error-correction step takes place after a pattern is presented at the input layer and the forward-propagation step is completed. Each processing unit in the output layer produces a single real number for its

output, which is compared to the target output specified in the training set. Based on this difference, an error value is calculated for each unit in the output layer.

Next, the weights are adjusted for all of the interconnections that go into the output layer. Then an error value is calculated for all of the units in the hidden layer that is just below the output layer. Then, the weights are adjusted for all interconnections that go into the hidden layer. The process is continued until the last layer of weights has been adjusted.

The problem of the back-propagation algorithm is that the middle-layer neurons have no target values. Hence, the training is more complicated, because the error must be propagated back through the network, including the non-linear functions, layer by layer.

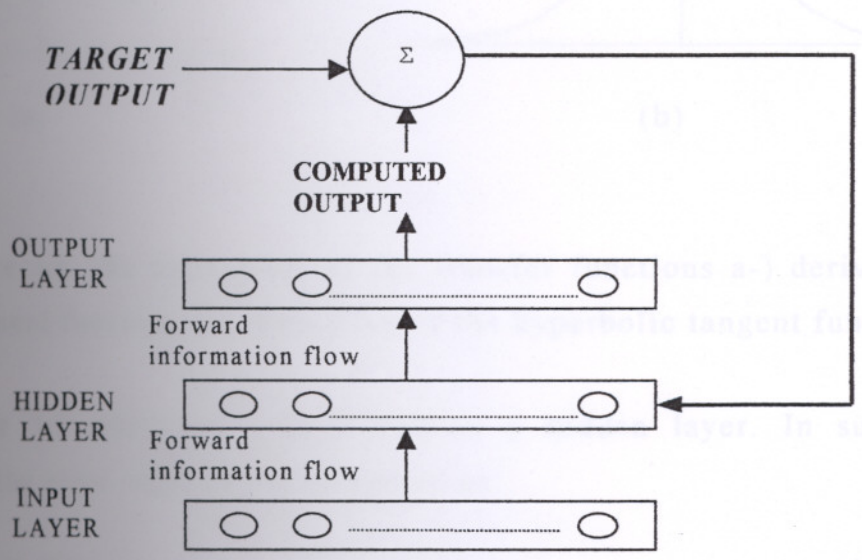


Figure 3.5 The Back-propagation network (Fausett, 1992)

The error value, denoted by the variable δ, is simple to compute for the output layer and somewhat more complicated for the hidden layers. If unit j is in the output layer, then its error value (δ_j) is:

$$\delta_j = (t_j - a_j) f'(S_j) \qquad (3.7)$$

where;

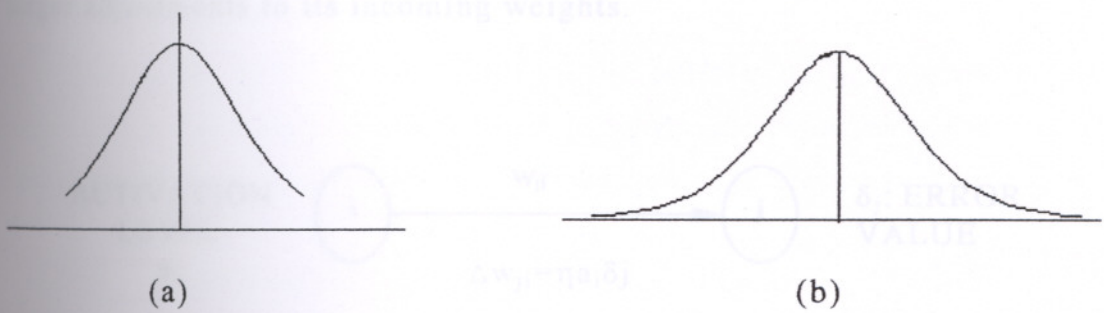
t_j = the target value for unit j

a_j = the output value for unit j

f'(x) = the derivative of the transfer function f

Sj = weighted sum of inputs to j

The quantity (t_j-a_j) reflects the amount of error. The f', scales the error to force a stronger correction when the sum Sj is near the rapid rise in the curves. Figure 3.6 illustrates the form of the f' functions.



(a) (b)

Figure 3.6 The derivatives of the transfer functions a-) derivative of the sigmoid function, b-) derivative of the hyperbolic tangent function

Figure 3.2 illustrates j as a unit in a hidden layer. In such a situation, the error value of j is computed as:

$$\delta_j = f'(S_j)\sum_k \delta_k w_{kj} \qquad (3.8)$$

In this case, a weighted sum is taken of the δ values of all units that receive output from unit j. The f' again serves to scale this output by emphasizing the region of rapid rise of the sigmoid function.

The adjustment of the connection weights is done using the δ values of the processing unit. Each interconnection weight is adjusted by taking

into account the δ value of the unit that receives input from that interconnection. The connection weight adjustment is done as follows:

$$\Delta w_{ji} = \eta \delta_j a_i \qquad (3.9)$$

Figure 3.7 diagrams the adjustment of weight w_{ji}, which goes to unit j from unit i. The amount adjusted depends on three factors: δ_j, a_i, and η. This weight adjustment equation is known as the generalized-delta rule.

The size of the weight adjustment is proportional to δ_j, the error value of the target unit. Thus a larger error value for unit j results in larger adjustments to its incoming weights.

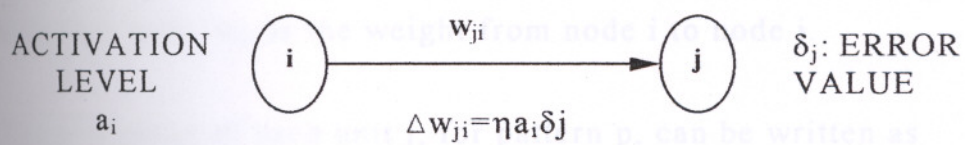


Figure 3.7 Process of Weight Updating (Fu, 1994)

The weight adjustment is also proportional to a_i, the output value for the originating unit. If this output value is small, then the weight adjustment is small. If this output value is large, then the weight adjustment is large. Thus a higher activation value for incoming unit i results in a larger adjustment to its outgoing weight.

The variable η in the weight-adjustment equation (3.9) is the learning rate. Its value is chosen by the neural network user, and usually reflects the rate of learning of the network. Values that are very large can lead to instability in the network, and unsatisfactory learning. Values that are too small can lead to excessively slow learning. Sometimes the learning rate is varied in an attempt to produce more efficient learning of the network. For example, allowing the value of η to begin at a high value and to decrease during the learning session can

sometimes produce better learning performance.

3.4. Back-Propagation Learning Algorithm (Generalized Delta Rule)

Upon performing neural network analysis, the accuracy of the results can be determined through error analysis.

The error function is defined to be proportional to the square of the difference between the actual and desired output, for all the patterns to be learned.

$$E_p = \frac{1}{2}\sum_j (t_{pj} - a_{pj})^2 \qquad (3.10)$$

Where, Ep is the error function for pattern p, t_{pj} represents the target output for pattern p on node j, while a_{pj} represents the actual output at that node. w_{ji} is the weight from node i to node j.

The activation of each unit j, for pattern p, can be written as

$$S_{pj} = \sum_i w_{ji} a_{pi} \qquad (3.11)$$

i.e. S_{pj} is simply the weighted sum.

The output from each unit j is the threshold function fj acting on the weighted sum. In the perceptron, this was the step function; in the back-propagation (multi-layer perceptron), it is generally the sigmoid function, but hyperbolic tangent function is also used in this study.

$$a_{pj} = f_j(S_{pj}) \qquad (3.12)$$

The following relationship

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial S_{pj}} \frac{\partial S_{pj}}{\partial w_{ji}} \qquad (3,13)$$

can be written by the chain rule.

The derivative of S_{pj} with respect to w_{ji} can be written as follows:

$$\frac{\partial S_{pj}}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_k w_{jk} a_{pk}$$

$$= \sum_k \frac{\partial w_{jk}}{\partial w_{ji}} a_{pk}$$

$$= a_{pi} \qquad (3.14)$$

since $\frac{\partial w_{jk}}{\partial w_{ji}} = 0$, except when k = i it equals to 1.

The change in error can be defined as a function of the change in the net inputs to a unit.

$$-\frac{\partial E_p}{\partial S_{pj}} = \delta_{pj} \qquad (3.15)$$

and (23) becomes

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} a_{pi} \qquad (3.16)$$

Decrease in the value of the error function, Ep, implies that the weight changes are proportional to δpj api, i.e.

$$\Delta_p w_{ji} = \eta \delta_{pj} a_{pi} \qquad (3.17)$$

It is necessary to determine δpj for each of the unit. If known, then Ep can be decreased. Using (3.16) and the chain rule:

$$\delta_{pj} = -\frac{\partial E_p}{\partial S_{pj}} = -\frac{\partial E_p}{\partial a_{pj}} \frac{\partial a_{pj}}{\partial S_{pj}} \tag{3.18}$$

Consider the second term in (3.13), and from (3.12),

$$\frac{\partial a_{pj}}{\partial S_{pj}} = f_j'(S_{pj}) \tag{3.19}$$

Consider now the first term in (3.18). From (3.10), E_p can be differentiated with respect to a_{pj}, giving

$$\frac{\partial E_p}{\partial a_{pj}} = -(t_{pj} - a_{pj}) \tag{3.20}$$

Thus,

$$\delta_{pj} = f_j'(S_{pj})(t_{pj} - a_{pj}) \tag{3.21}$$

This is useful for the output units, since the target and output are both available, but not for the hidden units, since their targets are not known.

So, if units j is not an output unit, it can be written, by the chain rule such that

$$\frac{\partial E_p}{\partial a_{pj}} = \sum_k \frac{\partial E_p}{\partial S_{pk}} \frac{\partial S_{pk}}{\partial a_{pj}}$$

$$\frac{\partial E_p}{\partial a_{pj}} = \sum_k \frac{\partial E_p}{\partial S_{pk}} \frac{\partial}{\partial a_{pj}} \sum_i w_{ki} a_{pi} \tag{3.22}$$

$$\frac{\partial E_p}{\partial a_{pj}} = -\sum_k \delta_{pk} w_{kj} \tag{3.23}$$

using (3.11) and (3.15), and noticing that the sum drops out since the partial differential is non-zero for only one value, as shown in (3.14).

Substituting (3.23) in (3.18), the following is obtained:

$$\delta_{pj} = f_j'(S_{pj})\sum_k \delta_{pk} w_{kj} \qquad (3.24)$$

This equation represents the change in the error function, with respect to the weights in the network. This provides a method for changing the error function and reducing it. The function is proportional to the errors δ_{pk} in subsequent units, so the error has to be calculated in the output units first (given by (3.21)) and then passed back through the net to the earlier units to allow them to alter their connection weights. It is the back passing of this error value that leads to the network being referred to as "back-propagation networks". Equation (3.21) and (3.24) together define how multi-layer networks are trained.

One advantage of using the sigmoid function as the non-linear threshold function is that it is quite like the step function, and so should demonstrate behavior of a similar nature. The sigmoid function is defined as,

$$f(S_{pj}) = \frac{1}{1+e^{-S_{pj}}} \qquad (3.25)$$

and has the range $0 < f(S_{pj}) < 1$.

The major reason for its use is that it has a simple derivative, and this makes the implementation of the back-propagation system much easier. Given that the output of a unit, apj is given by

$$a_{pj} = f(S_{pj}) = \frac{1}{1+e^{-S_{pj}}} \qquad (3.26)$$

The derivative with respect to that unit, $f'(S_{pj})$ is given by

$$f'(S_{pj}) = \frac{e^{-S_{pj}}}{(1+e^{-S_{pj}})^2}$$

$$f'(S_{pj}) = f(S_{pj})(1 - f(S_{pj}))$$

$$f'(S_{pj}) = a_{pj}(1 - a_{pj}). \qquad (3.27)$$

One advantage of using the hyperbolic tangent function as the non-linear threshold function is that it has the range $-1 < f(S_{pj}) < 1$.

The hyperbolic tangent function is defined as

$$f(S_{pj}) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad (3.28)$$

Given that the output of a unit, apj is given by

$$a_{pj} = f(S_{pj}) = \frac{e^{S_{pj}} - e^{-S_{pj}}}{e^{S_{pj}} + e^{-S_{pj}}} \qquad (3.29)$$

The derivative with respect to that unit, $f'(S_{pj})$ is given by

$$f'(S_{pj}) = \left[1 + \frac{e^{S_{pj}} - e^{-S_{pj}}}{e^{S_{pj}} + e^{-S_{pj}}}\right] * \left[1 - \frac{e^{S_{pj}} - e^{-S_{pj}}}{e^{S_{pj}} + e^{-S_{pj}}}\right]$$

$$f'(S_{pj}) = \left[1 + f(S_{pj})\right] * \left[1 - f(S_{pj})\right] \qquad (3.30)$$

The derivative is a simple function of the outputs.

3.5. Network Training

Back-propagation networks are trained by supervised learning technique, whereby the network is presented with a series of pattern pairs (each pair consisting of an input pattern and a target output

pattern). Each pattern is a matrix or vector format of real numbers. The target output pattern is the desired response to the input pattern and is used to determine the error values in the network when the weights are adjusted.

The target output pattern is sometimes designed to represent a classification for the input pattern. In this way, the network may be presented with a series of input patterns together with the classification for each input pattern. In other applications, the target output is simply a desired pattern response to the input pattern, and the network is trained to be a pattern-mapping system.

The patterns in the training set are presented to the network repeatedly. Each training iteration consists of presenting each input/output pattern pair once. When all patterns in the training set have been presented, the training iteration is completed, and the next training iteration is begun. A typical back-propagation example can entail hundreds or thousands of training iterations.

The training procedure is as follows:

1. Randomizing the weights to small values (both positive and negative) to ensure that the network is not saturated by large values of weights. (If all weights start at equal values, and the desired performance requires unequal weights, the network would not be trained at all.)

2. applying the training matrix to the network after selecting the training set

3. calculating the error, or the difference between desired and calculated outputs

4. adjusting the weights of the network again in a way to minimize the error

5. repeating these steps for each pair of input-output values in the training set until the error for the entire system is acceptably low.

3.6. Convergence Criteria

When a network is trained successfully, the error decreases as the training session progresses. It is important, then, to have a quantitative measure of learning. The average error is usually calculated to reflect the degree to which learning has taken place in the network. This measure reflects how close the network is getting to the correct answers. As the network learns, its average error decreases.

Convergence is a process whereby the error value for the network gets closer and closer to 0. Convergence is not always easy to achieve because the process may take an exceedingly long time and sometimes the network gets stuck in a local minimum and stops learning altogether.

Ideally, it is desired to find a global minimum. This corresponds to the lowest error value possible. Unfortunately, it is possible to encounter a local minimum. Nevertheless, a local minimum is surrounded by higher ground, and the network usually does not leave a local minimum by the standard back-propagation algorithm described. Special techniques should be used to get out of a local minimum.

The appearance of a local minimum is not always a significant problem. Back-propagation networks typically converge to a good error value when the training examples are clearly distinguishable. When a local minimum is encountered, the network may be able to avoid entering that local minimum by a number of techniques, for example, changing the learning parameter or the number of hidden units. These techniques tend to change the scenario involved with moving on a different path in error plane and may cause the network to avoid the local minimum.

Adding small random values to the weights allows the network to escape from a local minimum once it is encountered by moving the position of the network from a local minimum to a random point some distance away. If the new position is sufficiently removed from the local minimum, then convergence may proceed in a new direction without

getting stuck in the same local minimum again.

3.7. Strengths and Limitations of Back-Propagation Algorithm

The principal strength of back-error propagation is its relatively general pattern-mapping capability; it can learn a tremendous variety of pattern-mapping relationships. It does not require any a priori knowledge of a mathematical function that maps the input patterns to the output patterns; back-propagation merely needs examples of the mapping to be learned. The flexibility of the paradigm is enhanced by many design choices available (choices for the number of layers, interconnections, processing units, the learning constant, and data representations), As a result, back-error propagation might be able to address a broad spectrum of applications.

The largest drawback with back-propagation algorithm appears to be its convergence time. Training sessions can require hundreds or thousands of iterations for relatively simple problems. Realistic applications may have thousands of examples in a training set, and it may take days of computing time (or more) to complete training. Usually this length needs to be done only during the development of network, because most applications require a trained network and do not need on-line retraining of the net.

A variety of special techniques have been developed in a attempt to decrease convergence time and to avoid local minima. A "momentum" term is sometimes used to speed convergence procedures. Improvements in convergence have also been found by varying the learning parameter η by starting with a larger value for η and progressing to smaller values. Techniques for avoiding local minima include changing the network or the training set, and adding small random values to the weights.

Chapter 4

FORECASTING WITH ARTIFICIAL NEURAL NETWORKS

Recent research activities have shown that artificial neural networks have powerful pattern classification and pattern recognition capabilities. Currently, artificial neural networks are being used for a wide variety of tasks in many different fields of business, industry and science (Widrow et al., 1994).

One major application area of artificial neural networks is forecasting (Kolehmainen et al., 2000). Artificial neural networks provide an attractive alternative tool for both forecasting researchers and practitioners. Several distinguishing features of artificial neural networks make them valuable and attractive for a forecasting task.

First, as opposed to the traditional model-based methods, artificial neural networks are data-driven self-adaptive methods in that there are few a priori assumptions about the models for problems under study. They learn from examples and capture suitable functional relationships among the data even if the underlying relationships are unknown or hard to describe. Thus artificial neural networks are well suited for problems whose solutions require knowledge that is difficult to specify but for which there are enough data or observations. This modeling approach with the ability to learn from experience is very useful for many practical problems since it is often easier to have data than to have good theoretical guesses about the underlying laws governing the systems from which data are generated.

Second, artificial neural networks can generalize. After learning the data presented to them (a sample), artificial neural networks can often correctly infer the unseen part of a population even if the

sample data contain noisy information. As forecasting is performed via prediction of future behavior (the unseen part) from examples of past behavior, it is an ideal application area for neural networks, at least in principle.

Third, artificial neural networks are universal functional approximators. It has been shown that a network can approximate any continuous function to any desired accuracy (Irie and Miyake, 1988; Hornik et al., 1989; Cybenko, 1989; Funahashi, 1989; Hornik, 1991, 1993). Artificial neural networks have more general and flexible functional forms than the traditional statistical methods can effectively deal with. Any forecasting model assumes that there exists an underlying (known or unknown) relationship between the inputs (the past values of the time series and/or other relevant variables) and the outputs (the future values). Frequently, traditional statistical forecasting models have limitations in estimating this underlying function due to the complexity of the real system. Artificial neural networks can be a good alternative method to identify such a function.

Finally, artificial neural networks are nonlinear. Forecasting has long been the domain of linear statistics. The traditional approaches to time series prediction assume that the time series under study are generated from linear processes. Linear models have advantages in that they can be understood and analyzed in great detail, and they are easy to explain and implement. However, they may be totally inappropriate if the underlying mechanism is nonlinear. It is unreasonable to assume a priori that a particular realization of a given time series is generated by a linear process. In fact, real world systems are often nonlinear (Tasadduq et al., 2002).

The idea of using artificial neural networks for forecasting is not new. The first application dates back to 1964. Hu (1964), in his thesis, used the Widrow's adaptive linear network to weather forecasting. Due to the lack of a training algorithm for general multi-layer networks at the

time, the research was quite limited. It is not until 1986 when the back-propagation algorithm was introduced (Rumelhart et al., 1986) that there has been much development in the use of artificial neural networks for forecasting.

4.1. Forecasting the Environmental Variables with Artificial Neural Networks

In recent years, artificial neural networks have become a popular and useful tool for modeling environmental systems. They have already been successfully used to simulate the export of nutrients from river basins (Clair et al., 1996), to forecast salinity (DeSilets et al., 1992) and ozone levels (Comrie, 1997), to predict air pollution (Boznar et al., 1993) and the functional characteristics of ecosystems (Paruelo et al., 1997), and to model algal growth, and transport in rivers (Whitehead et al., 1997). Many environmental modelers are "experimenting" with artificial neural networks on datasets for which the use of more conventional techniques (e.g., regression) has been unsuccessful.

4.1.1. Model Development Process

4.1.1.1. Data Normalization

Nonlinear activation functions have the squashing role in restricting or squashing the possible output from a node to, typically, [0,1] for sigmoid function or [-1,1] for tangent hyperbolic function. Data normalization is often performed before the training process begins. As mentioned earlier, when nonlinear transfer functions are used at the output nodes, the desired output values must be transformed to the range of the actual outputs of the network. Even if a linear output transfer function is used, it may still be advantageous to standardize the outputs as well as the inputs to avoid computational problems, to meet algorithm requirement, and to facilitate network learning (Mok and Tam, 1998).

Here are the generally used methods for normalization:

- Linear transformation to [0,1]: $x_n = \frac{x_o - x_{min}}{x_{max} - x_{min}}$ (Zurada, 1992)
- Linear transformation to [-1,1]: $x_n = 2*\left(\frac{x_o - x_{min}}{x_{max} - x_{min}}\right) - 1$ (Zurada, 1992)
- Statistical normalization: $x_n = \frac{x_o - \bar{x}}{s}$ (Soucek, 1992)
- Simple normalization: $x_n = \frac{x_o}{x_{max}}$ (Bose and Liang, 1996)

Normalization of the output values (targets) is usually dependent of the normalization of the inputs. For time series forecasting problems, the normalization of targets is typically performed together with the inputs. The choice of range to which inputs and targets are normalized depends largely on the activation function of output nodes, with typically [0, 1] for logistic function and [-1, 1] for hyperbolic tangent function. Several researchers scale the data only to the range of [0.1, 0.9] (Srinivasan et al., 1994) or [0.2, 0.8] (Tang et al., 1993) based on the fact that the nonlinear activation functions usually have asymptotic limits (they reach the limits only for infinite input values) and the guess that possible outputs may lie, for example, only in [0.1, 0.9], or even [0.2, 0.8] for a logistic function (Bose and Liang, 1996).

In this study, linear normalization methods have been used for sigmoid function and hyperbolic tangent function and their ranges were [0,1] for sigmoid function and [-1,1] for hyperbolic tangent function.

It should be noted that, as a result of normalizing the target values, the observed output of the network will correspond to the normalized range. Thus, to interpret the results obtained from the network, the outputs must be rescaled to the original range. From the user's point of view, the accuracy obtained by the artificial neural networks should be based on the rescaled data set.

4.1.1.2. Training and Testing Sets

As it is mentioned earlier, training and testing sample are typically required for building an artificial neural network forecaster. The training sample is used for artificial neural networks model development and the test sample is adopted for evaluating the forecasting ability of the model. Sometimes a third one called the validation sample is also utilized to avoid the overfitting problem or to determine the stopping point of the training process (Tsoukalas and Uhrig, 1997). It is common to use one test set for both validation and testing purposes particularly with small data sets.

The first issue here is the division of the data into the training and test sets. Although there is no general solution to this problem, several factors such as the problem characteristics, the data type and the size of the available data should be considered in making the decision. It is critical to have both the training and test sets representative of the population or underlying mechanism. This has particular importance for time series forecasting problems. Inappropriate separation of the training and test sets will affect the selection of optimal artificial neural network structure and the evaluation of artificial neural network forecasting performance.

The literature offers little guidance in selecting the training and the test samples. Most authors select them based on the rule of 90% vs. 10%, 80% vs. 20% or 70% vs. 30%, etc. Some choose them based on their particular problems.

In this study, 273 daily average values of dust, SO_2, temperature and wind speed were used. Of 273, first 197 were used as a training set and 76 were used as testing set.

Another closely related factor is the sample size. No definite rule exists for the requirement of the sample size for a given problem. The amount of data for the network training depends on the network

structure, the training method, and the complexity of the particular problem or the amount of noise in the data on hand. In general, as in any statistical approach, the sample size is closely related to the required accuracy of the problem. The larger the sample size, the more accurate the results will be. Perez et al. (2000) test the effect of different training sample size and find that as the training sample size increases, the artificial neural network forecaster performs better.

4.1.1.3. Performance Measures

Although there can be many performance measures for an artificial neural network forecaster like the modeling time and training time, the ultimate and the most important measure of performance is the prediction accuracy it can achieve beyond the training data. However, a suitable measure of accuracy for a given problem is not universally accepted by the forecasting academicians and practitioners. An accuracy measure is often defined in terms of the forecasting error which is the difference between the actual (desired) and the predicted value. There are number of measures of accuracy in the forecasting literature and each has advantages and limitations (Bose and Liang, 1996). The most frequently used measures are;

- the mean absolute deviation (MAD) = $\frac{\sum|e_t|}{N}$
- the sum of squared error (SSE) = $\sum(e_t)^2$
- the mean squared error (MSE) = $\frac{\sum(e_t)^2}{N}$
- the root mean squared error (RMSE) = $\sqrt{MSE}$
- the mean absolute percentage error (MAPE) = $\frac{1}{N}\sum\left|\frac{e_t}{e_y}\right|\times 100$

where e_t is the individual forecast error; y_t is the actual value; and N is the number of error terms. For this study, the mean squared error is used to measure the accuracy of the models.

4.2. Literature Review of the Modeling of the Environmental Variables with Artificial Neural Networks

In most of the studies conducted on environmental parameters, wide-ranging prediction techniques, including gaussian models and numerical models are generally used. These kind of models are mainly based on the mathematical formulation of the physics and chemistry of the atmosphere, which govern the dispersion of pollutants. Gaussian models can be applied to any situation where the distribution pattern of the pollutants in that direction can be represented by a gaussian or normal distribution over the selected averaged time. Even though these models have some physical basis, detailed information about the sources of pollutants and other parameters is not generally known. To overcome this limitation, statistical models are also used to facilitate the prediction of pollutant concentrations (Ziomass et al, 1995; Finzi et al., 1982). These models assume that the relationship between the variables is statistical in nature. However, such models require information about the distribution of the data which is generally not known a priori (Comrie, et al., 1997). Recently, neural network based models have also been applied to predict pollutant concentrations. These models provide a better alternative to statistical models because of their computational efficiency and generalization ability. They can handle data having high dimensionality.

Chelani et al. (2002), studied on a sulphur dioxide concentration prediction in New Delhi by using a three-layer neural network model with a one hidden layer and they used the Levenberg-Marquardt algorithm to train the network. They compared the results of artificial neural networks with multivariate regression model. The input parameters to the model included wind speed (in $km.h^{-1}$), temperature (in °C), relative humidity (%) and the wind direction. In their study, Chelani et al., concluded that the neural network model provides a better alternative than regression models.

Gardner et al., (1999) used the multi-layer perceptron neural networks to the prediction of the hourly NO_x and NO_2 concentration in London. Their multi-layer perceptron have two hidden layer each of which contained 20 nodes. Hourly NO_x and NO_2 concentration and hourly meteorological data were used for their model. The meteorological data included wind speed ($m.s^{-1}$), vapor pressure (mbar), dry bulb temperature(°C), visibility (presented to the system in the UKMO synoptic code) and base of lowest cloud(presented to the system in the UKMO synoptic code). Their work showed that the multi-layer perceptron neural networks can accurately model the relationship between local meteorological data and NO_x and NO_2 concentrations.

Sucar et. al., (1997) proposed an algorithm for structure learning in predictive expert system based on a probabilistic network representation. They applied this method for ozone prediction in Mexico City and their network was a three-layer Bayesian network. They used eight parameters for their model and these are; wind direction and velocity, temperature, relative humidity, sulphur dioxide, carbon monoxide, nitrogen dioxide and ozone. They represented random 400 samples to the system as a training set and tested their system with 100 random samples to estimate the ozone. They observed that their estimations are acceptable and also their work also showed that wind velocity and the wind direction are the major parameters effecting the ozone level.

Maier et al., (2001) worked on two case studies; the forecasting of salinity in the River Murray, South Australia, and the forecasting of incidences of a species group of cyanobacterium *Anabaena* spp. in the River Murray. The data used for the salinity case study included daily salinities and the daily flow. The data used for the blue-green algea case study included weekly values of blue–green algea concentrations, turbidity, color, temperature, total phosphorus, soluble phosphorus, oxidized nitrogen, and total iron. Both the case studies have been done with three-layer feed-forward neural networks which are trained by back-propagation algorithm. They observed better results for salinity

forecasting then the blue-green algea case study but prediction of the algea concentrations was also acceptable.

Murtagh et al., (2000) studied the prediction of the oceanic upwelling off the Mauretanian coast. The data available to them consisted of daily satellite SST data for 1982 (these were sunshine data and only available if the sky is not cloudy), daily wind and surface heat flux data, and the topology of the region. The multi-layer perceptron architecture was used with one hidden layer. They used 299 values as training set and 40 values as testing set, but their studies did not make them satisfied, because their model's mean squared error was more then 0,25.

Perez et al., (2001) studied the prediction of NO and NO_2 concentrations near a street with heavy traffic in Santiago, Chile. They also used meteorological data in their study and this meteorological data included temperature, wind velocity and relative humidity. Perez et al., developed a three-layer feed-forward neural network and used back-propagation algorithm to train their model. They also compared their predictions with linear regression model and obtained that neural network model had more accurate results.

Chapter 5

FORECASTING THE SO_2 AND DUST WITH ARTIFICIAL NEURAL NETWORKS FOR IZMIR

5.1. Sulphur Dioxide (SO_2)

Sulphur dioxide (SO_2) is a potent respiratory irritant when inhaled. Asthmatics are particularly susceptible. SO_2 acts directly on the upper airways (nose, throat, trachea and major bronchi), producing rapid responses within minutes. It achieves maximum effect in 10 to 15 minutes, particularly in individuals with significant airway reactivity, such as asthmatics and those suffering similar bronchospastic conditions.

Epidemiological studies have shown significant associations between daily average SO_2 levels and mortality from respiratory and cardiovascular causes. Increases in hospital admissions and emergency room visits for asthma, and respiratory disease have also been associated with ambient SO_2 levels (Jones, 1999). These associations were observed with up to a two-day lag period. Long-term exposure to SO_2 and fine particle sulphates ($SO4^{2-}$) has been associated with an increase in mortality from lung cancer and development of asthma and cardio-pulmonary obstructive disease (Jones, 1999). Increases in respiratory symptoms have also been associated with SO_2 levels (Cogliani, 2001).

The guideline values for sulphur dioxide are 350 $\mu g/m^3$ (1-hour average) and 120 $\mu g/m^3$ (24-hour average) (WHO, 1999). These values are set to provide protection of lung function and prevent other respiratory symptoms of vulnerable sub-groups in the population, including asthmatics and those with chronic obstructive lung disease. They are in line with current international guideline values and standards. The short-term guideline value has been removed, as it is not appropriate for managing air quality in large air sheds, however, shorter-

term criteria for sulphur dioxide may be appropriate for assessing industrial discharges.

5.2. Dust

Dust is a particulate contaminant suspended in the atmosphere that may consists of solids or liquid droplets and vary in diameter from 0.001 μm to 100 μm. It can be directly emitted or can be formed in the atmosphere when gaseous pollutants such as SO_2 and NO_X react to form fine particles (Bockreis and Jager, 1999).

Like CO_2 dust limits the amount of energy arriving and leaving the surface of the planet, with consequent changes in weather patterns. Industrial, transport and domestic equipment all release particles of dust into the atmosphere. The exposure of man to dust can lead to a wide variety of respiratory diseases, including pulmonary fibrosis, obstructive lung disease, allergy and lung cancer (Mussio et al., 2001). Particles can cause irritation to the eyes, nose and throat. Some of the larger particles (>10 μm diameter) reaching the nose or throat will be filtered out by the body's natural defence system. However, very tiny particles that reach deep into the lung may be absorbed into the blood stream or cause lung or other health problems. Such particles are those less than 10 μm in diameter (Mussio et al., 2001).

The particulate pollutants may be classified according to their optical properties, nature and size as follows:

- Smoke particulate consists of solid and liquid particles ranging from 0.05 to 1.0 μm which are formed during incomplete combustion of carbonaceous materials. It includes smoke of gaseous pollutants like oxides of sulphur, oxides of nitrogen, CO and hydrocarbons etc.

- Dust particles are composed of fine solid particles ranging from 1-100 μm.
- Mist or liquid particles are formed by condensation of a vapour, having a size less than 10 μm.

• Spray It constitutes from liquid particles obtained from the parent liquid by the process of mechanical disintegration like atomization.

• Fumes are generally obtained by the condensation of vapours by sublimation, distillation, boiling, calcination and by several other chemical reactions. Generally organic solvents, metals and metallic oxides form fume particles having a size less than 1 μm. Carbon particles, ash, asbestos, oil, grease and acids form particulates which are widely distributed in air.

Sources of the dust are;

- Energy (Coal burning, fossil fuel burning stoves, power plants, dung burning smelters)
- Industry
  - Construction industry (cement factory, mining and quarrying)
  - Chemical industry (carbon powder manufacturers pharmaceuticals, industrial additives, pesticides and other agricultural chemicals)
  - Metal processing industry (metal flakes, dust and fumes)
  - Light industries (pre-treatment, drying operations, paint pigments)
- Transport
- Natural Sources (volcano eruption, wind erosion, sea salt, forest fire)

Major concerns for human health from exposure to PM-10 include: effects on breathing and respiratory systems, damage to lung tissue, cancer, and premature death. The elderly, children, and people with chronic lung disease, influenza, or asthma, are especially sensitive to the effects of particulate matter. Table 5.1 shows the limit values for Dust matters.

Table 5.1 Air quality guidelines on Dust (WHO, 1994)

Reference period	Limit value
one year (median of daily values)	80 $\mu g/m^3$
winter (median of daily values)	130 $\mu g/m^3$
year peak (98 percentile of daily values)	250 $\mu g/m^3$

5.3. Sulphur Dioxide and Dust Pollution in İzmir

The İZSU Air Pollution Controlling Department has been monitoring the sulphur dioxide and dust concentration in the air of İzmir at four locations since 1997. Since the main SO_2 and dust sources for İzmir are industrial processes, traffic, commercial and residential activities, the stations are located in Bornova, Konak, Karşıyaka and Alsancak. These districts are chosen according to their local properties. All this districts have high commercial and residential activities, and especially Karşıyaka is affected by the industrial activities.

Daily average values of SO_2 and dust values from 01.01.2001 to 30.09.2001 have been used in this study.

Since the air quality is directly related with the meteorological parameters, in this study daily average values of temperature and wind speed are used, and these data have been collected by the Meteorological Weather Forecasting Department.

These data are separated into two groups; training (first 197 daily values) and testing set (last 76 daily values), and this extraction has been done with neuro-shell extraction program (NeuroShell2, Ward-Systems Group Inc.). These sets are illustrated in Figure 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, and 5.8.

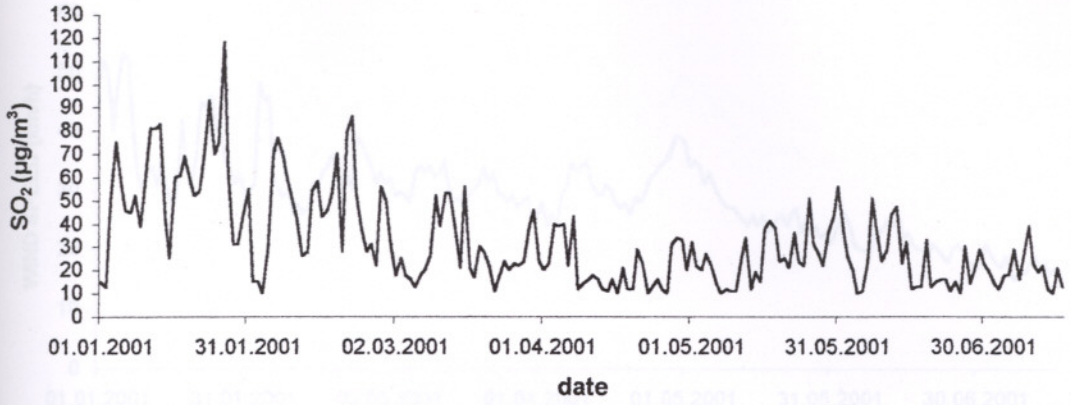


Figure 5.1 Daily SO_2 values for training set

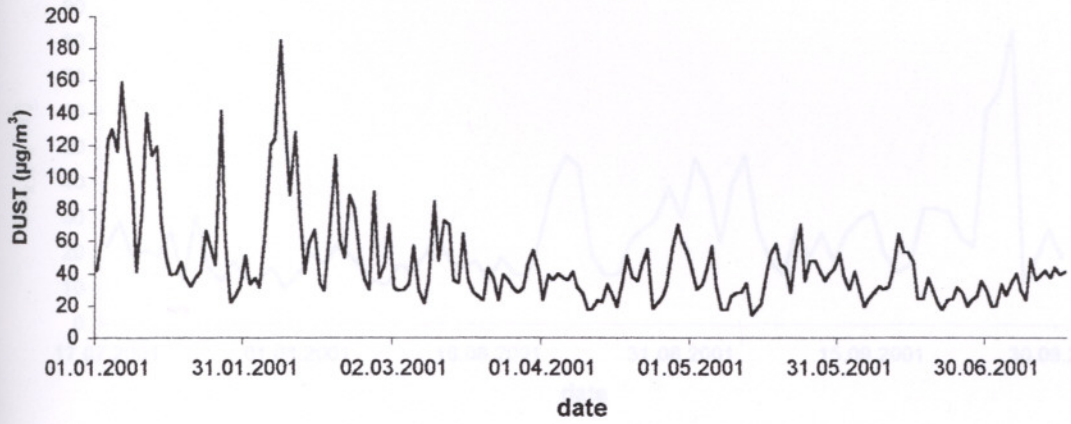


Figure 5.2 Daily dust values for training set

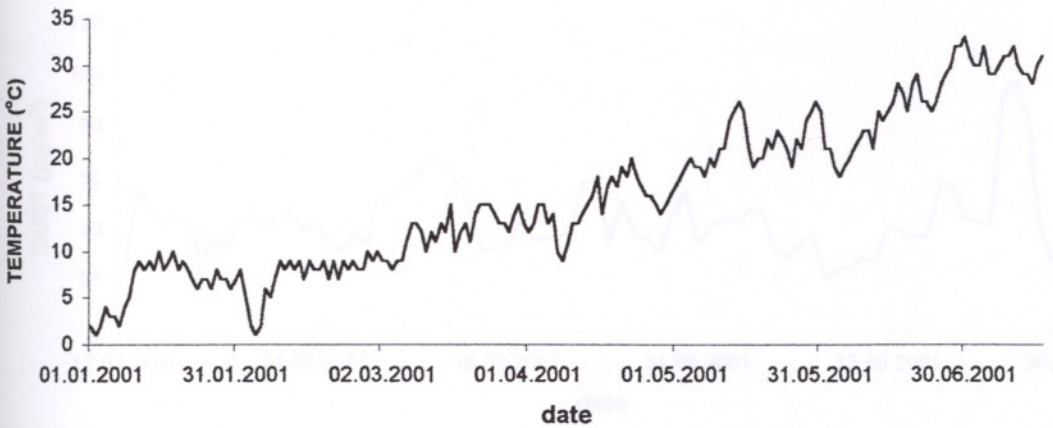


Figure 5.3 Daily temperature values for training set

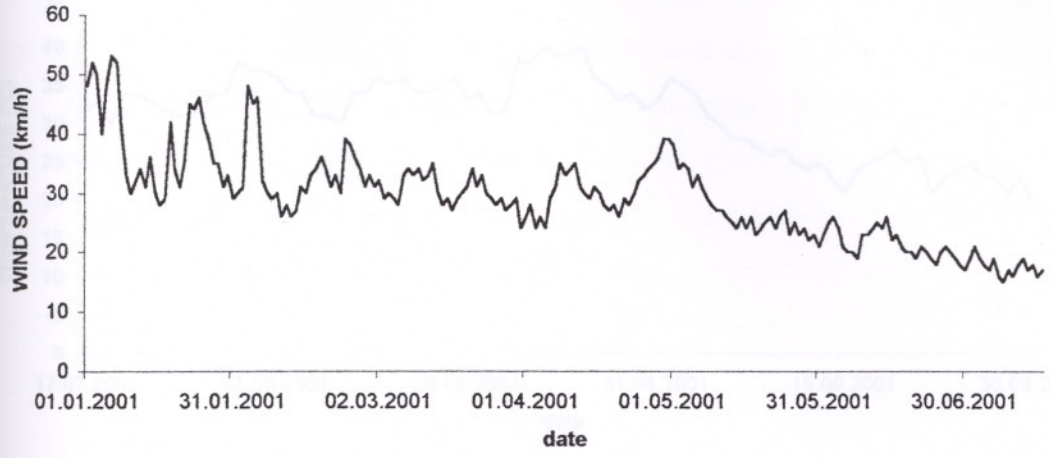


Figure 5.4 Daily wind speeds for training set

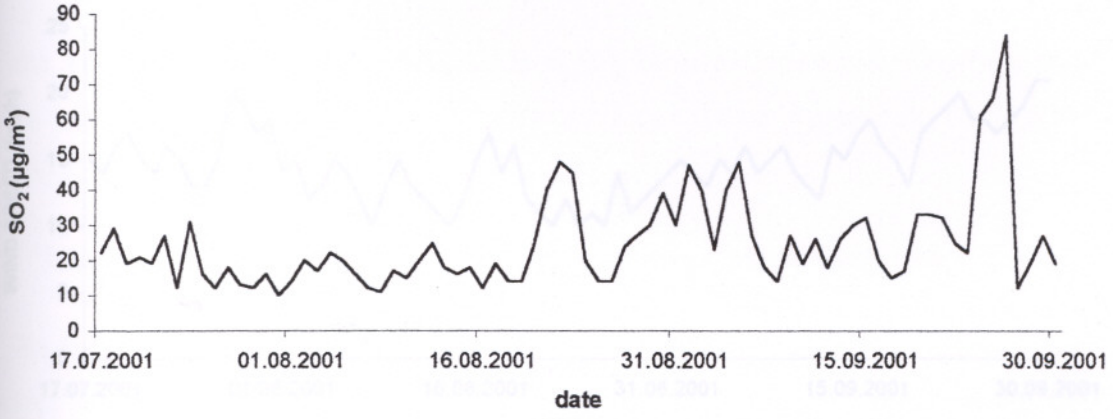


Figure 5.5 Daily SO_2 values for testing set

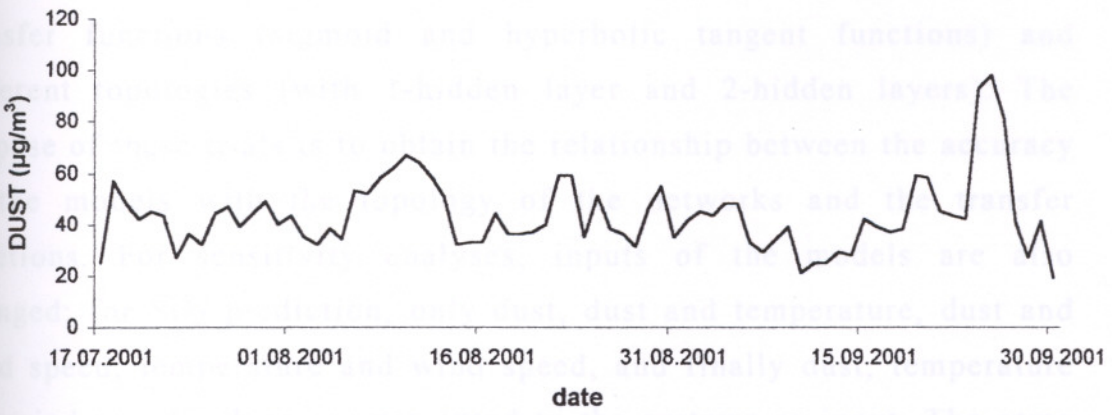


Figure 5.6 Daily dust values for testing set

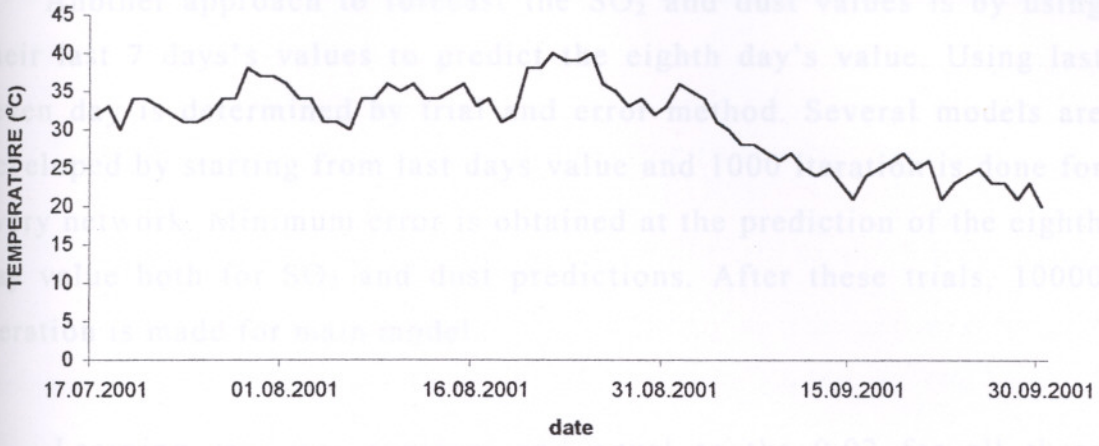


Figure 5.7 Daily temperature values for testing set

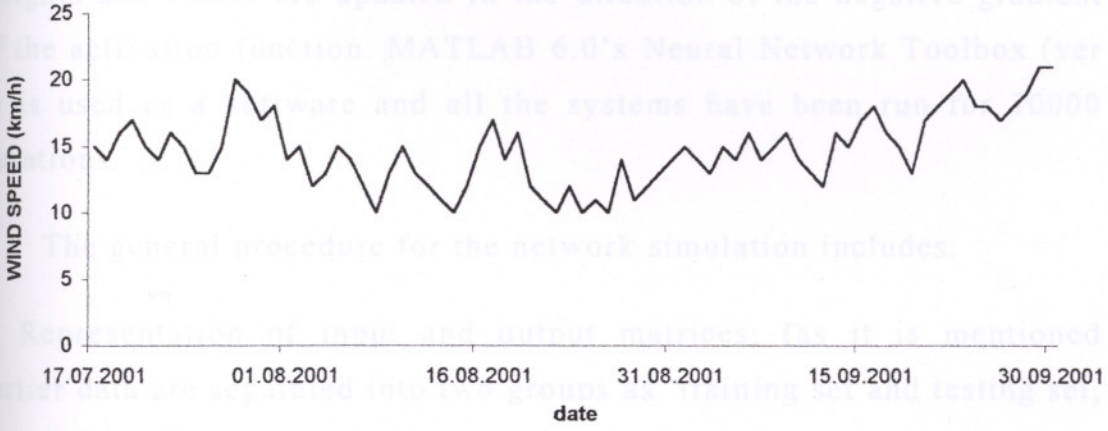


Figure 5.8 Daily wind speeds for testing set

Several trials have been done with these sets with two different transfer functions (sigmoid and hyperbolic tangent functions) and different topologies (with 1-hidden layer and 2-hidden layers). The purpose of these trials is to obtain the relationship between the accuracy of the models with the topology of the networks and the transfer functions. For sensitivity analyses, inputs of the models are also changed; for SO_2 prediction, only dust, dust and temperature, dust and wind speed, temperature and wind speed, and finally dust, temperature and wind speed values are presented to the systems as input. The same procedure is followed for dust prediction, input sets are changed with the same method. Aim of these trials is to determine the efficiencies of the input parameters and deciding their relationships.

Another approach to forecast the SO_2 and dust values is by using their last 7 days's values to predict the eighth day's value. Using last seven day is determined by trial and error method. Several models are developed by starting from last days value and 1000 iteration is done for every network. Minimum error is obtained at the prediction of the eighth day value both for SO_2 and dust predictions. After these trials, 10000 iteration is made for main model.

Learning rate was constant and equal to the 0.02 for all these studies and back-propagation algorithm is used to train the systems. Batch Gradient Descent Rule is used as a training method whereas the weights and biases are updated in the direction of the negative gradient of the activation function. MATLAB 6.0's Neural Network Toolbox (ver 4) is used as a software and all the systems have been run for 10000 iterations.

The general procedure for the network simulation includes:

1. Representation of input and output matrices; (as it is mentioned earlier data are separated into two groups as training set and testing set, Figure 5.1-5.8)

2. Representation of the transfer functions; (sigmoid and hyperbolic tangent functions are used)

3. Selection of the network structure; (two different network topologies have been developed; with one hidden layer and two hidden layers and hidden layers with three nodes)

4. Assigning of the random weights; (initial random weights are assigned)

5. Selection of the learning procedure (Gradient Descent Rule is applied);

6. Presentation of the test pattern and prediction or validation set of data for generalization (training of the network completed after 10000

iterations, than testing set is represented to the system but validation set is not used in this study)

The learning of weights is done using the following procedure:

1. Selection of random numbers for all weights;

2. Calculation of output vectors (referred also as the network output) and comparison with the target output (referred also as the desired output);

3. If the network output is approximately equal to the desired output, then continue with step 1, and if not, weights are corrected according to the correction rule and then continue with step 1.

Chapter 6

RESULTS AND DISCUSSION

6.1. Prediction of Sulphur Dioxide

The network topologies for the SO_2 prediction is shown in Figure 6.1. For this trial input parameters are dust, temperature and wind speed. The network of the first trial has one hidden layer with three nodes and the network of the second trial has two hidden layers with three nodes in each hidden layers, and system is trained with 10000 iterations.

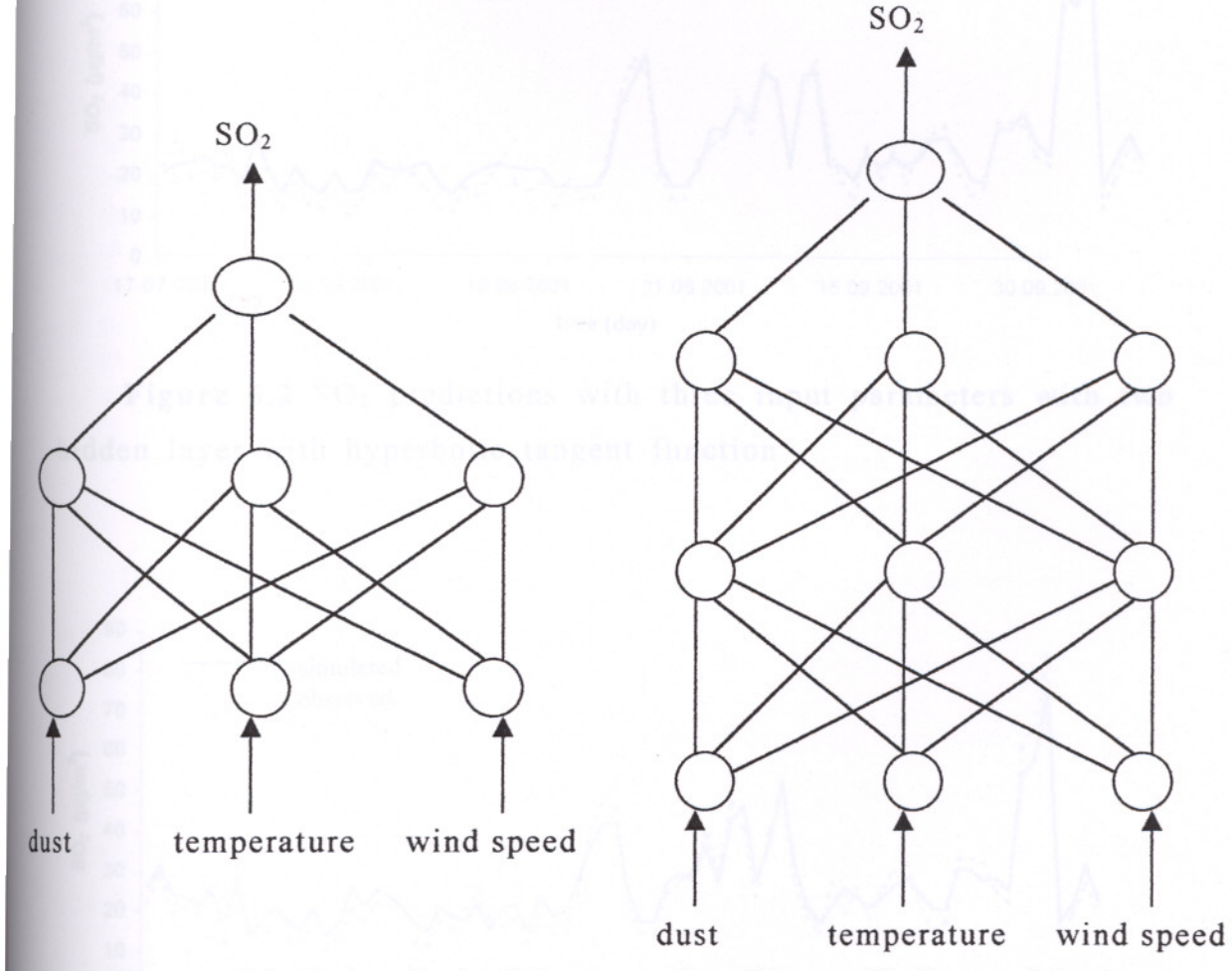


Figure 6.1 Network Architectures for SO_2 predictions with three input parameters

Graphics in the Figure 6.2, 6.3, 6.4 and 6.5 shows the results of the neural network models. It is obtained that neural networks are able

to learn from the training set and make the predictions accurately. R^2 (correlation coefficient between desired and predicted outputs) and daily average error values are used to control the accuracy of the systems. These values are shown in Table 6.1. According to the R^2 values and daily average errors, increasing the topology of the network gives more accurate results than the simple neural networks topologies. It is also obtained that using the hyperbolic tangent function makes more accurate predictions then the sigmoid function.

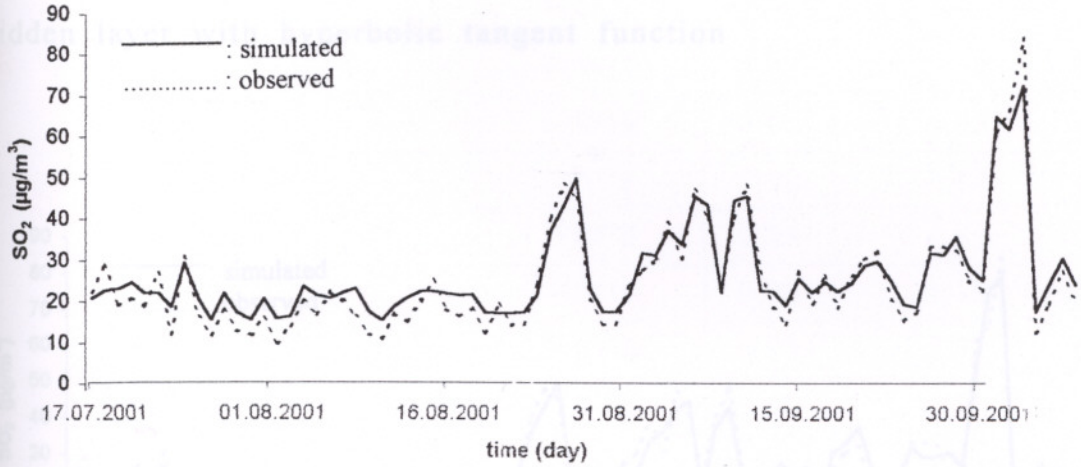


Figure 6.2 SO_2 predictions with three input parameters with two hidden layer with hyperbolic tangent function

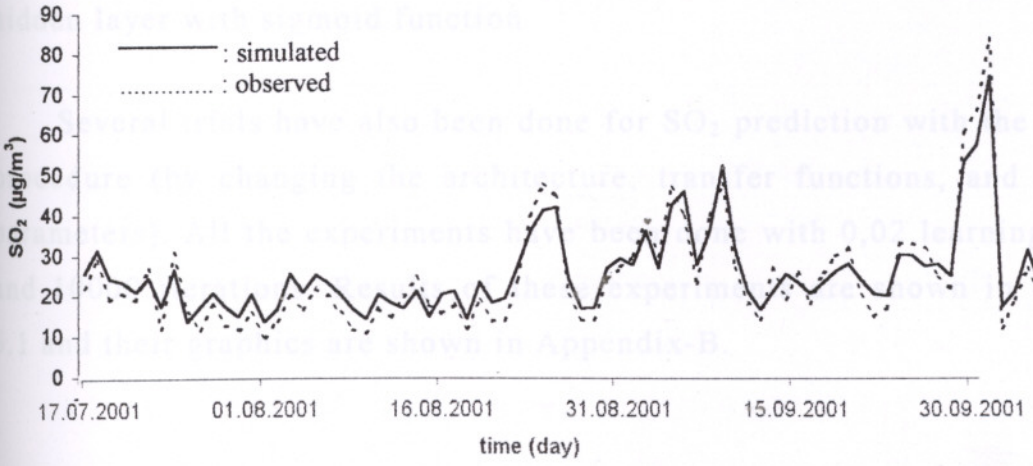


Figure 6.3 SO_2 predictions with three input parameters with two hidden layer with sigmoid function

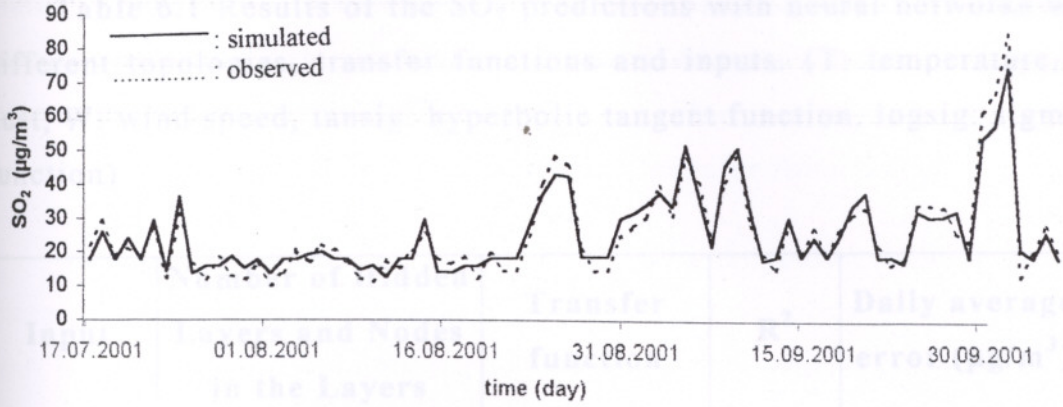


Figure 6.4 SO_2 predictions with three input parameters with one hidden layer with hyperbolic tangent function

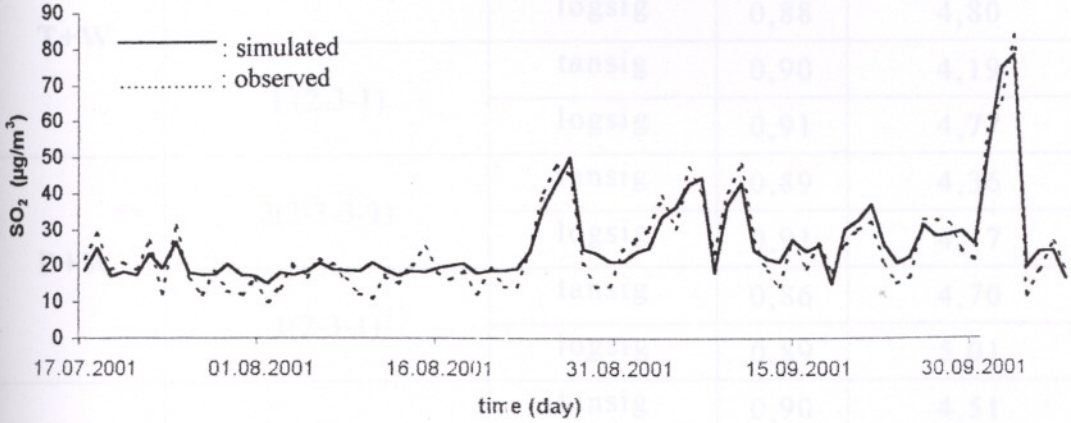


Figure 6.5 SO_2 predictions with three input parameters with one hidden layer with sigmoid function

Several trials have also been done for SO_2 prediction with the same procedure (by changing the architecture, transfer functions, and input parameters). All the experiments have been done with 0,02 learning rate and 10000 iterations. Results of these experiments are shown in Table 6.1 and their graphics are shown in Appendix-B.

Table 6.1 Results of the SO_2 predictions with neural networks with different topologies, transfer functions and inputs. (T: temperature, D: dust, W: wind speed, tansig: hyperbolic tangent function, logsig: sigmoid function)

Input	Number of Hidden Layers and Nodes in the Layers	Transfer function	R^2	Daily average error ($\mu g/m^3$)
D+T+W	2 (3-3-3-1)	tansig	0,94	3,60
		logsig	0,93	3,98
	1 (3-3-1)	tansig	0,92	3,34
		logsig	0,90	4,08
T+W	2 (2-3-3-1)	tansig	0,92	3,84
		logsig	0,88	4,80
	1 (2-3-1)	tansig	0,90	4,19
		logsig	0,91	4,77
D+W	2(2-3-3-1)	tansig	0,89	4,36
		logsig	0,91	4,17
	1(2-3-1)	tansig	0,86	4,70
		logsig	0,89	5,01
D+T	2(2-3-3-1)	tansig	0,90	4,51
		logsig	0,87	4,75
	1(2-3-1)	tansig	0,89	4,79
		logsig	0,88	4,99
D	2 (1-3-3-1)	tansig	0,80	5,21
		logsig	0,83	4,77
	1 (1-3-1)	tansig	0,76	5,57
		logsig	0,82	5,26

According to these results, increasing the topology of the networks generally gives better and accurate results. However, sometimes increasing the number of hidden layers could make the system unstable. For example, when the temperature and the wind speed are input to the

network with sigmoid function, network with one hidden layer gave more accurate results than network with two hidden layers. Similar case is also occurred when the dust and the temperature are given to the system as inputs with sigmoid function being the transfer function.

Using more variables as input leads to the better results. This is because system can generalize data better and possible errors or mistakes can be distributed through the system and they can be minimized and cannot effect the results too much. These results are also consistent with the literature.

Another study on the prediction of SO_2 is done by using the last seven day's sulphur dioxide values as inputs. Topology of the network is shown in the Figure 6.6. This experiment is also done with 0.02 learning rate and 10000 iterations.

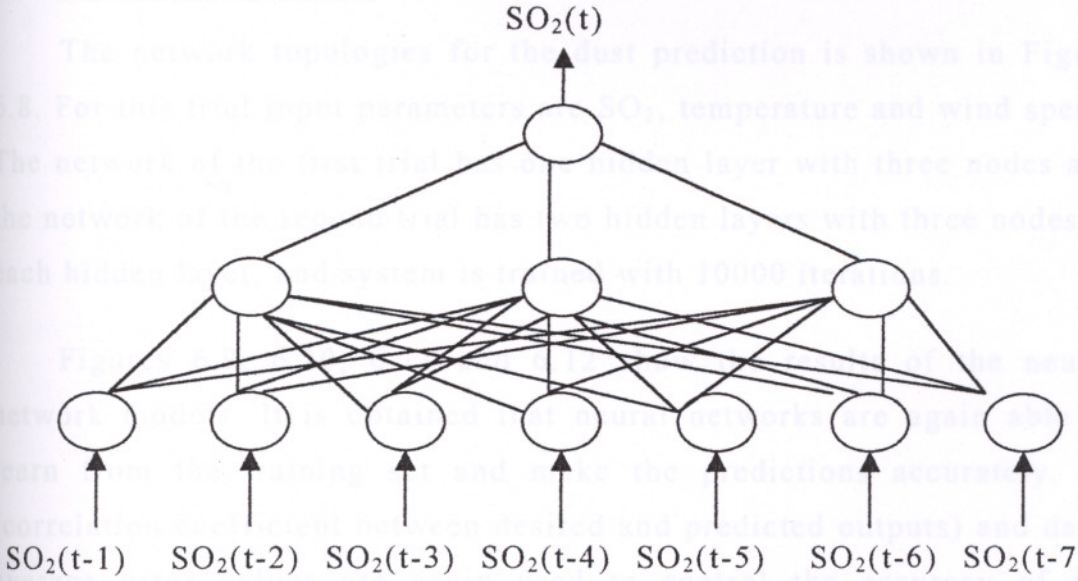


Figure 6.6 Network Architecture for SO_2 predictions with last seven day's values.

Sigmoid function is used as transfer function and R^2 is calculated as 0,94 and daily average error is calculated as 4,03 $\mu g/m^3$ and these results are showing that neural network gives accurate results. Figure 6.7 illustrates the comparison between simulated and observed outputs.

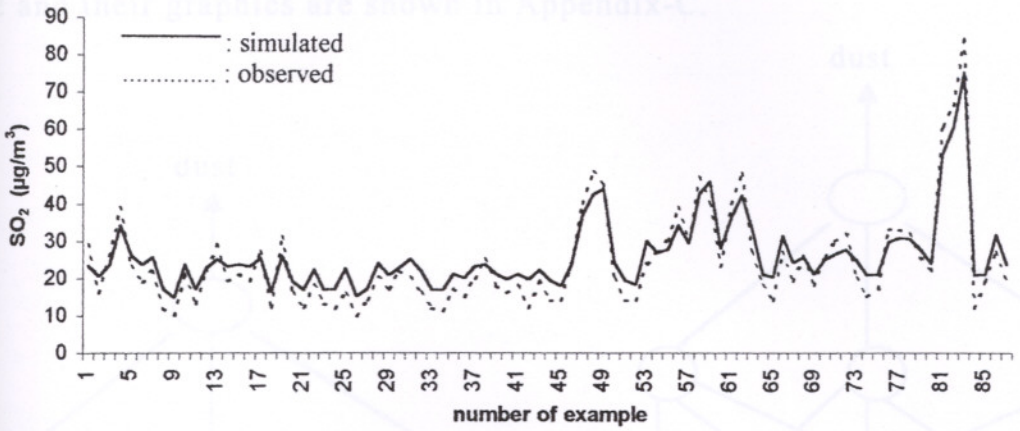


Figure 6.7 SO_2 predictions with last seven days sulphur dioxide values

6.2 Prediction of Dust

The network topologies for the dust prediction is shown in Figure 6.8. For this trial input parameters are SO_2, temperature and wind speed. The network of the first trial has one hidden layer with three nodes and the network of the second trial has two hidden layers with three nodes in each hidden layer, and system is trained with 10000 iterations.

Figures 6.9, 6.10, 6.11 and 6.12 show the results of the neural network models. It is obtained that neural networks are again able to learn from the training set and make the predictions accurately. R^2 (correlation coefficient between desired and predicted outputs) and daily average error values are again used to control the accuracy of the systems. These values are shown in Table 6.2. According to the R^2 values and daily average errors, increasing the topology of the network gives more accurate results than the simple neural networks topology. It is also obtained that using the sigmoid function makes more accurate predictions than the hyperbolic tangent function.

Several trials have also been done for dust prediction with the same procedure (by changing the architecture, transfer functions, and input parameters). All the experiments have been done with 0,02 learning rate

and 10000 iterations. Results of these experiments are shown in Table 6.2 and their graphics are shown in Appendix-C.

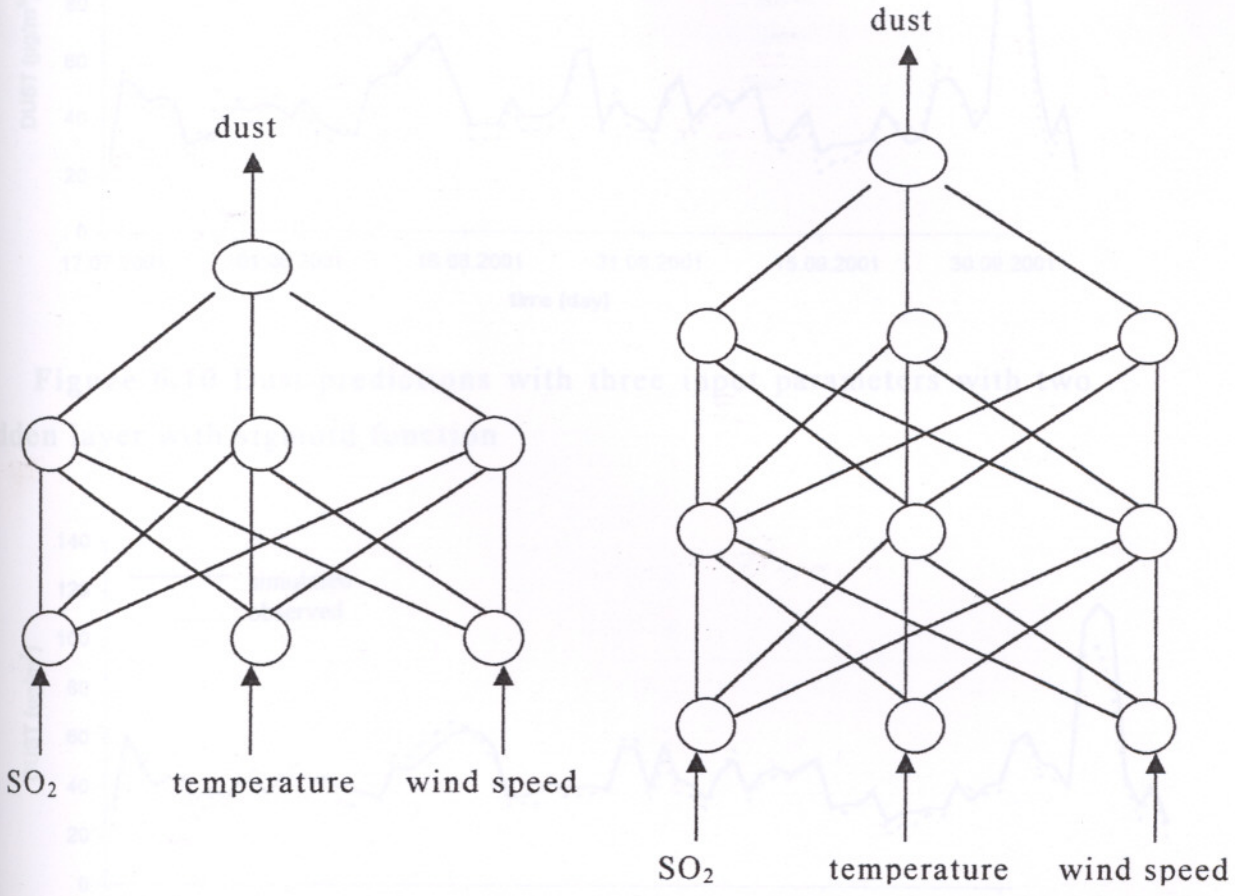


Figure 6.8 Network Architectures for dust predictions with three input parameters

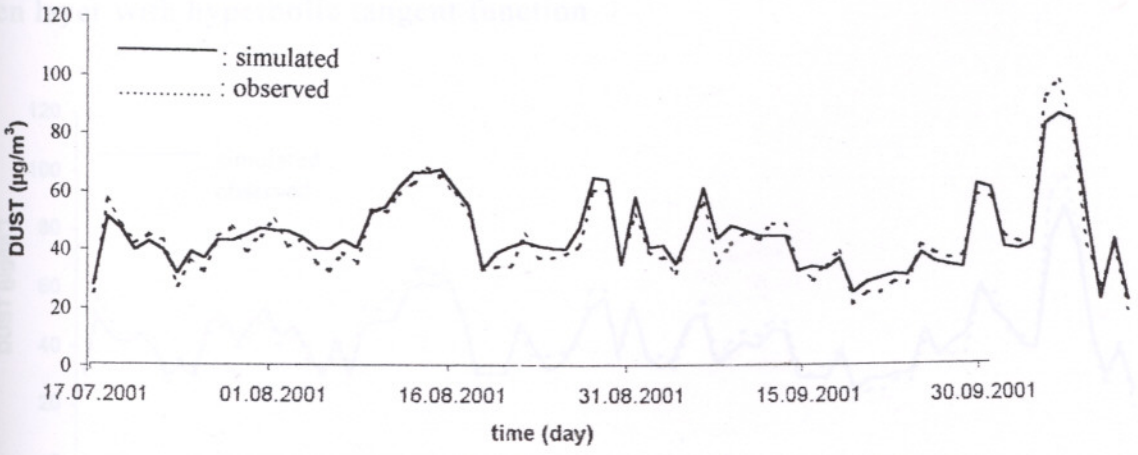


Figure 6.9 Dust predictions with three input parameters with two hidden layer with hyperbolic tangent function

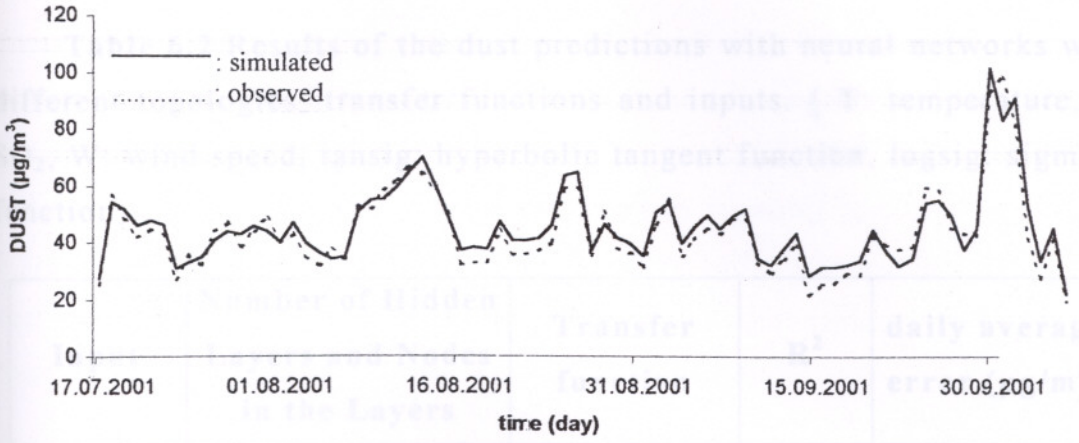


Figure 6.10 Dust predictions with three input parameters with two hidden layer with sigmoid function

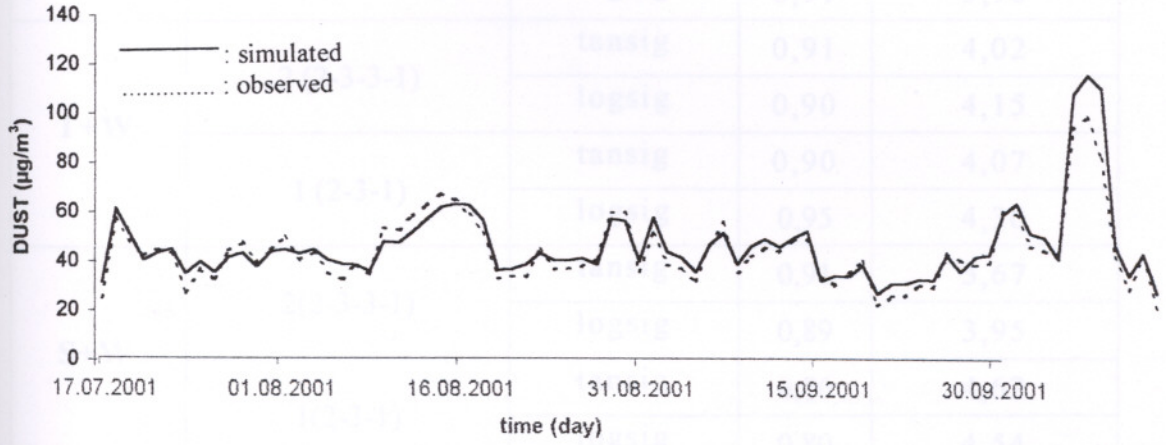


Figure 6.11 Dust predictions with three input parameters with one hidden layer with hyperbolic tangent function

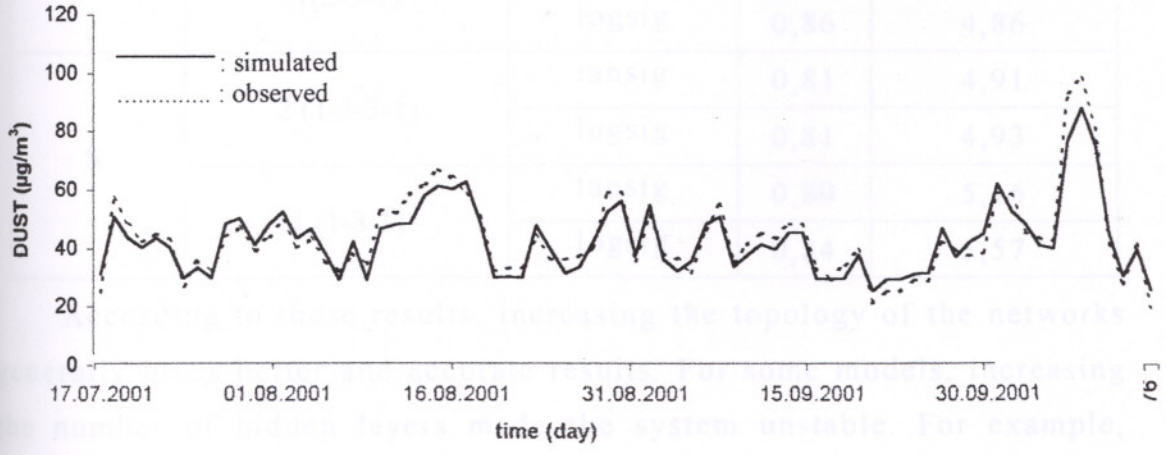


Figure 6.12 Dust predictions with three input parameters with one hidden layer with sigmoid function

Table 6.2 Results of the dust predictions with neural networks with different topologies, transfer functions and inputs. (T: temperature, S: SO_2, W: wind speed, tansig: hyperbolic tangent function, logsig: sigmoid function)

Input	Number of Hidden Layers and Nodes in the Layers	Transfer function	R^2	daily average error ($\mu g/m^3$)
S+T+W	2 (3-3-3-1)	tansig	0,92	3,64
		logsig	0,91	4,08
	1 (3-3-1)	tansig	0,90	4,09
		logsig	0,91	3,90
T+W	2 (2-3-3-1)	tansig	0,91	4,02
		logsig	0,90	4,15
	1 (2-3-1)	tansig	0,90	4,07
		logsig	0,95	4,30
S+W	2(2-3-3-1)	tansig	0,91	3,67
		logsig	0,89	3,95
	1(2-3-1)	tansig	0,86	4,62
		logsig	0,80	4,54
S+T	2(2-3-3-1)	tansig	0,92	3,70
		logsig	0,92	3,51
	1(2-3-1)	tansig	0,87	4,42
		logsig	0,86	4,86
S	2 (1-3-3-1)	tansig	0,81	4,91
		logsig	0,81	4,93
	1 (1-3-1)	tansig	0,80	5,16
		logsig	0,84	5,57

According to these results, increasing the topology of the networks generally gives better and accurate results. For some models, increasing the number of hidden layers made the system unstable. For example, when the temperature and the wind speed are input to the network with sigmoid function, network with one hidden layer gave more accurate

results than network with two hidden layer. The similar case is also occurred when the SO_2 and the temperature are inputs with hyperbolic tangent function. It is also obtained that changing the topology of the network does not make any differences when sigmoid function is used as a transfer function when the SO_2 is the only input parameter. Increasing the number of the parameters also increased the system performance, best results are occurred with three input parameters.

Another study on the prediction of dust is done by using the last seven days dust values as inputs. The topology of the network is shown in the Figure 6.13. This experiment is also done with 0.02 learning rate with 10000 iterations.

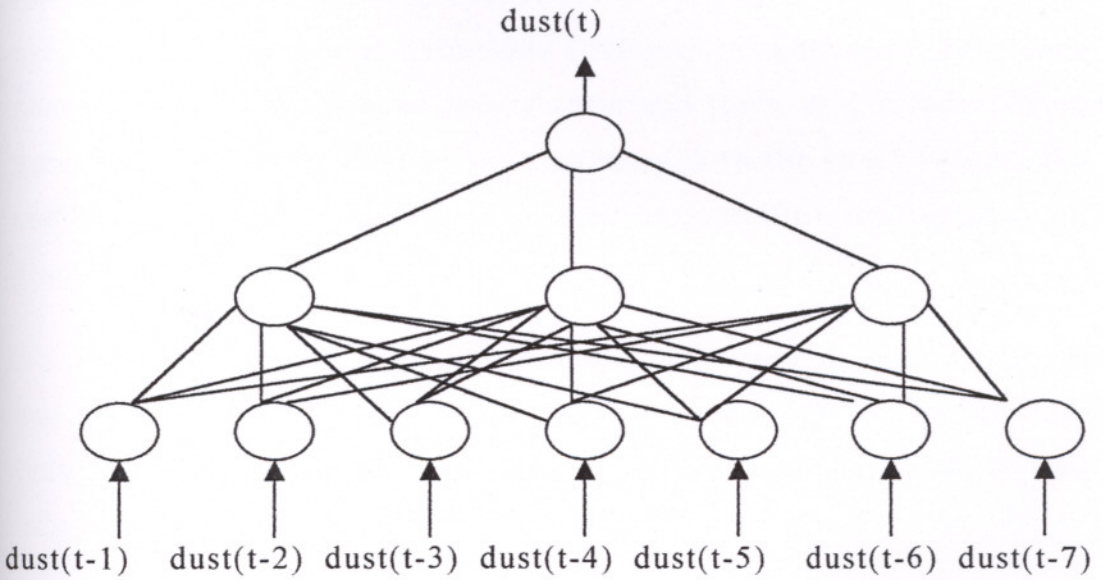


Figure 6.13 Network Architecture for dust predictions with last seven days values.

Sigmoid function is used as a transfer function and R^2 is calculated as 0,93 and daily average error is calculated as 4,32 $\mu g/m^3$ and these results are showing that neural network gives accurate results. Figure 6.14 illustrates the comparison between predicted and actual outputs.

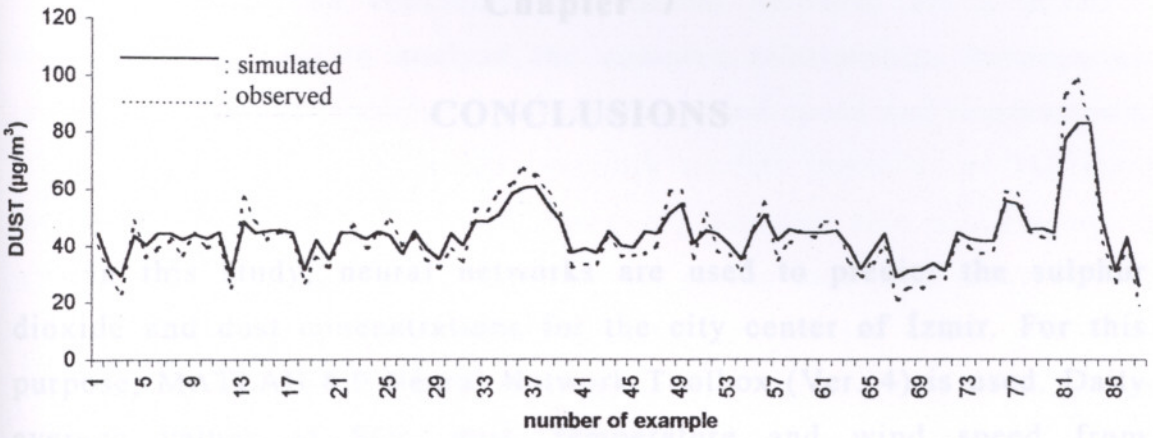


Figure 6.14 Dust predictions with last seven days dust values

Chapter 7

CONCLUSIONS

In this study, neural networks are used to predict the sulphur dioxide and dust concentrations for the city center of İzmir. For this purpose, MATLAB 6.0 Neural Network Toolbox (Ver. 4) is used. Daily average values of SO_2, dust, temperature and wind speed from 01.01.2001 to 30.09.2001 have been used in this study. These data are separated into two groups; training set (first 197 daily values) and testing set (last 97 daily values). Back-propagation learning algorithm is used to train the neural networks. Different networks are developed by changing the topologies of the systems and types of the inputs. Results generated from the networks are compared with the exact values, and R^2 and daily average errors are calculated to determine the accuracy of the models.

The satisfactory prediction of the actual measured data by the model has shown that neural networks can accurately model the relationship between local meteorological data and SO_2-dust concentrations in an urban environment.

In addition, several artificial neural networks models are constructed in this study, and results have shown that improving the architecture of the networks gives better and more accurate results. For SO_2 forecasting, best results are found by using three input parameters and a network with two hidden layers. For the dust prediction studies, the best and the most accurate results are found by using three input parameters with two hidden layers. The reason of this conclusion is that improvement in the system (increasing the number of the hidden layers) increases the system's fault tolerance because errors can be distributed throughout the system which decreases the differences between the desired and actual outputs.

It may be concluded that neural networks are a powerful computational tool to analyse the complex relationships between air pollutants and meteorological parameters (wind speed and temperature). The performance of the neural network models improves as the more number of inputs are provided; however, they can also make accurate predictions with limited data. Although giving only one parameter as input is decreasing the performance of the artificial neural networks models, it can be said that results are still acceptable both for SO_2 and dust predictions.

As future work, when further meteorological parameters (e.g. relative humidity, NO_2, CO, visibility) become available, the performance of the neural network can be improved. With larger number of data, better sensitivity analysis can be attained for air pollutants. The better way to study air pollution and to make better predictions of air pollutants is to establish a gauging station and collect the appropriate data as much as possible.

REFERENCES

N. K. Base and P. Liang, *Neural Network Fundamentals with Graphs, Algorithms and Applications*, (McGraw-Hill, Singapore, 1996).

A. Bockreis, J. Jager, "Odour monitoring by the combination of sensors and neural networks," *Environmental Modeling & Software* 14, (1999), 421–426.

M. Boznar, M. Lesjak, P. Mlakar, "A neural network based method for short-term predictions of ambient SO_2 concentrations in highly polluted industrial areas of complex terrain," *Atmospheric Environment* 27B (2), (1993), 221-230.

A. B. Chelani, C.V. C. Rao, K.M. Phadke, M.Z. Hasan "Prediction of sulphur dioxide concentration using artificial neural networks," *Environmental Modeling & Software* 17, (2002), 161–168.

. A. Clair, J. M. Ehrman, "Variations in discharge and dissolved organic carbon and nitrogen export from terrestrial basins with changes in climate: A neural network approach," *Limnol. Oceanogr.* 41 (1996), 921-927.

E. Cogliani, "Air pollution forecast in cities by an air pollution index highly correlated with meteorological variables," *Atmospheric Environment* 35, (2001), 2871-2877.

A. C. Comrie, "Comparing neural networks and regression models for ozone forecasting", *Journal of Air Waste Management Association* 47, (1997), 653-663.

J.M. Corchado, C. Fyfe, "Unsupervised neural method for temperature forecasting," *Artificial Intelligence in Engineering* 13, (1999), 351–357.

L. Desilets, B. Golden, Q. Wang, R. Kumar, "Predicting salinity in the Chesapeake Bay using back-propagation," *Computer. Operation Research* 19, (1992), 227-285,.

A. Engel, "Complexity of learning in artificial neural networks," *Theoretical Computer Science* 265, (2001), 285 –306.

L. Fausett, *Fundamentals of Neural Networks*, (Prentice-Hall Inc., USA, 1994).

L. Fu, *Neural Networks in Computer Intelligence*, (McGraw-Hill, 1994).

M.W. Gardner, S.R. Dorling, "Neural network modeling and prediction of hourly NO_x and NO_2 concentrations in urban air in London," *Atmospheric Environment* 33, (1999), 709-719.

M.W. Gardner and S. R. Dorling, "Artificial neural networks (The multi-layer perceptron)-a review of applications in the atmospheric sciences," *Atmospheric Environment* 32, (1998), 2627-2636.

A.P. Jones, "Indoor air quality and health," *Atmospheric Environment* 33, (1999), 4535-4564.

M. Kikuchi, K. Fukushima, "Invariant pattern recognition with eye movement: A neural network model," *Neurocomputing* 38, (2001), 1359-1365.

M. Kolehmainen, H. Martikainen, T. Hiltunen and J. Ruuskanen, "Forecasting air quality parameters using hybrid neural network modeling," *Environmental Monitoring and Assessment* 65, (2000), 277–286.

M. Kolehmainen, H. Martikainen, J. Ruuskanen "Neural networks and periodic components used in air quality forecasting," *Atmospheric Environment* 35, (2001), 815-825.

S. Lek and F. Gue´gan, "Artificial neural networks as a tool in ecological modeling, an introduction," *Ecological Modeling* 120, (1999), 65–73.

H. R. Maier, G. C. Dandy, "Understanding the behavior and optimizing the performance of back-propagation neural networks: an empirical study," *Environmental Modeling & Software* 13, (1998), 179–191.

H. R. Maier and G. C. Dandy, "Neural Network Based Modeling of Environmental Variables: A Systematic Approach," *Mathematical and Computer Modeling* 33, (2001), 669-682.

K. M. Mok and S. C. Tam, "Short-term prediction of SO_2 prediction in Macau with artificial neural networks," *Energy and Buildings* 28, (1998), 279-286.

F. Murtagh, G. Zheng, J.G. Campbell, A. Aussem, "Neural network modeling for environmental prediction," *Neurocomputing* 30, (2000), 65-70.

P.Mussio, A. W. Gnyp, P. F. Henshaw, "A fluctuating plume dispersion model for the prediction of odour-impact frequencies from continuous stationary sources," *Atmospheric Environment* 35, (2001), 2955-2962.

M. M. Nelson and W. T. Illingworth, *A Practical Guide to Neural Nets*, (Addison Wesley, USA, 1994).

J.M. Paruelo, F. Tomasel, "Prediction of functional characteristics of ecosystem-A comparison of artificial neural networks and regression models," *Ecological Modeling* 98 (2/3), (1997), 173-186.

A. Pelliccioni, U. Poli, "Use of neural net models to forecast atmospheric pollution," *Environmental Monitoring and Assessment* 65, (2000), 297-304.

P. Perez "Prediction of sulfur dioxide concentrations at a site near downtown Santiago, Chile," *Atmospheric Environment* 35, (2001), 4929 –4935.

P. Perez, A. Trier, "Prediction of NO and NO_2 concentrations near a street with heavy traffic in Santiago, Chile," *Atmospheric Environment* 35, (2001), 1783-1789.

P. Perez, A. Trier, J. Reyes "Prediction of $PM_{2.5}$ concentrations several hours in advance using neural networks in Santiago, Chile," *Atmospheric Environment* 34, (2000), 1189-1196.

M. Y. Rafiq, G. Bugmann, D. J. Easterbrook, " Neural network design for engineering applications," *Computers and Structures* 79, (2001), 1541-1552.

S.L. Reich, D.R. Gomez, L.E. Dawidowski, "Artificial neural network for the identification of unknown air pollution sources," *Atmospheric Environment* 33, (1999), 3045-3052.

B. Soucek, *Neural and Concurrent Real Time Systems*, (John Wiley, USA, 1989).

G. Spellman, "An application of artificial neural networks to the prediction of surface ozone concentrations in the United Kingdom," *Applied Geography* 19, (1999), 123–136.

L. E. Sucar and J.P. Brito, "Learning structure from data and ts application to ozone prediction," *Applied Intelligence* 7, (1997), 327–338.

I. Tasadduq, S. Rehman, K. Bubshait, "Application of neural networks for the prediction of hourly mean surface temperatures in Saudi Arabia," *Renewable Energy* 25, (2002), 545–554.

P.G. Whitehead, A. Howard, C. Arulmani, "Modeling algal growth and transport in rivers-A comparison of time series analysis, dynamic mass balance and neural network techniques," *Hydrobiologia* 349, (1997), 39-46.

J. Yao, C. L. Tan, "A case study on using neural networks to perform technical forecasting of forex," *Neurocomputing* 34, (2000), 79-98

L. H. Tsoukalas and R. E. Uhrig, *Fuzzy and Neural Approaches in Engineering*, (John Wiley,USA, 1997).

G. Zhang, B. E. Patuwo, M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting* 14, (1998), 35–62.

J. M. Zurada, *Introduction to Artificial Neural Systems*, (West Publishing, 1992).

APPENDIX – A

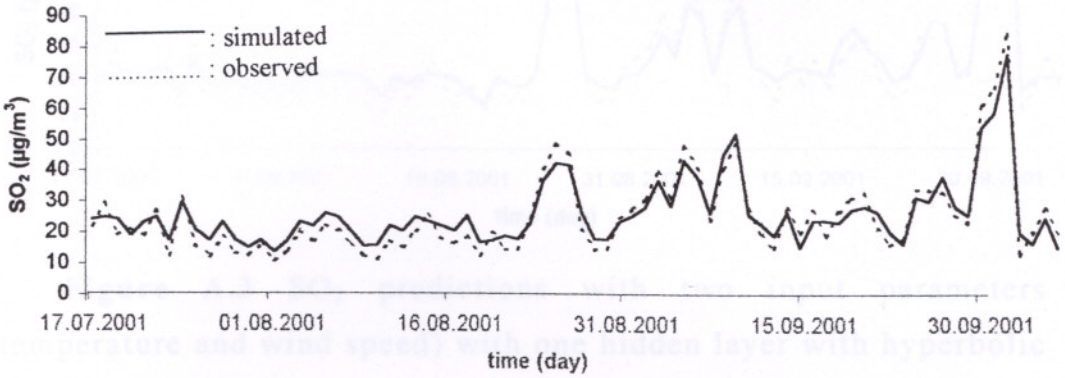


Figure A.1 SO_2 predictions with two input parameters (temperature and wind speed) with two hidden layer with hyperbolic tangent function

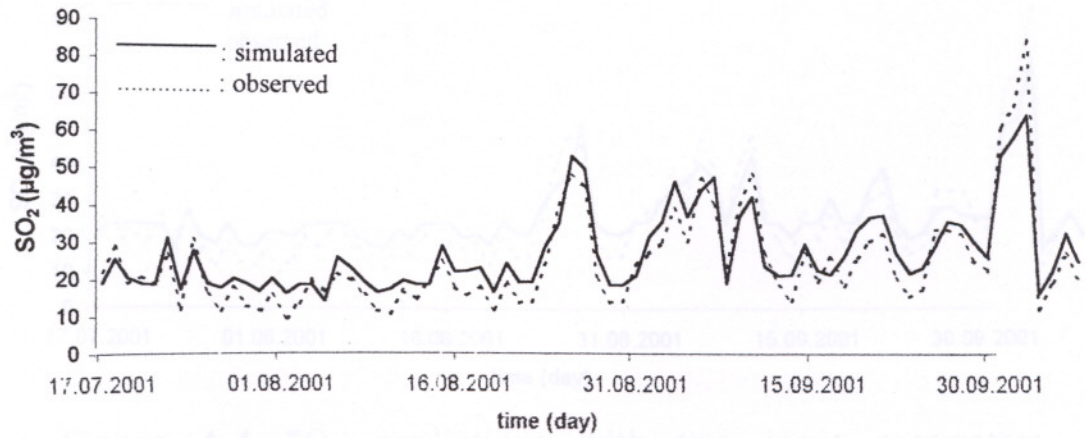


Figure A.2 SO_2 predictions with two input parameters (temperature and wind speed) with two hidden layer with sigmoid function

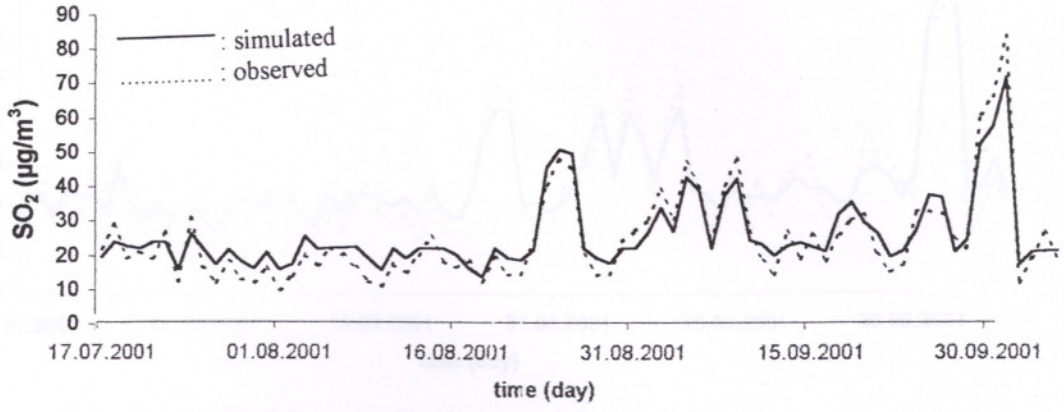


Figure A.3 SO_2 predictions with two input parameters (temperature and wind speed) with one hidden layer with hyperbolic tangent function

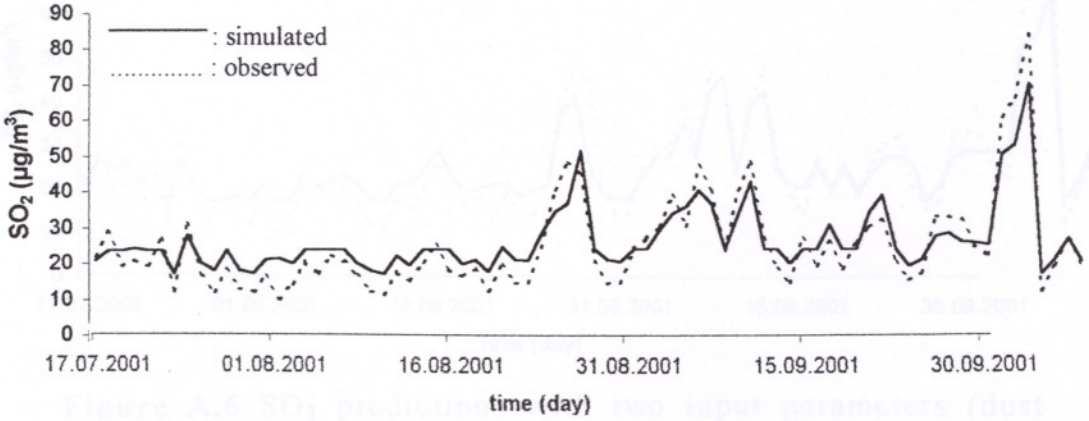


Figure A.4 SO_2 predictions with two input parameters (temperature and wind speed) with one hidden layer with sigmoid function

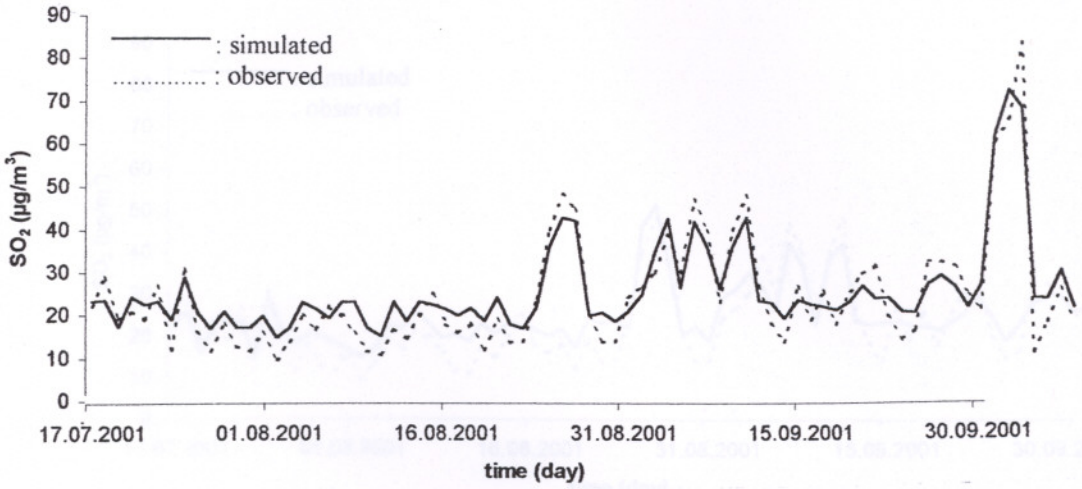


Figure A.5 SO_2 predictions with two input parameters (dust and wind speed) with two hidden layer with hyperbolic tangent function

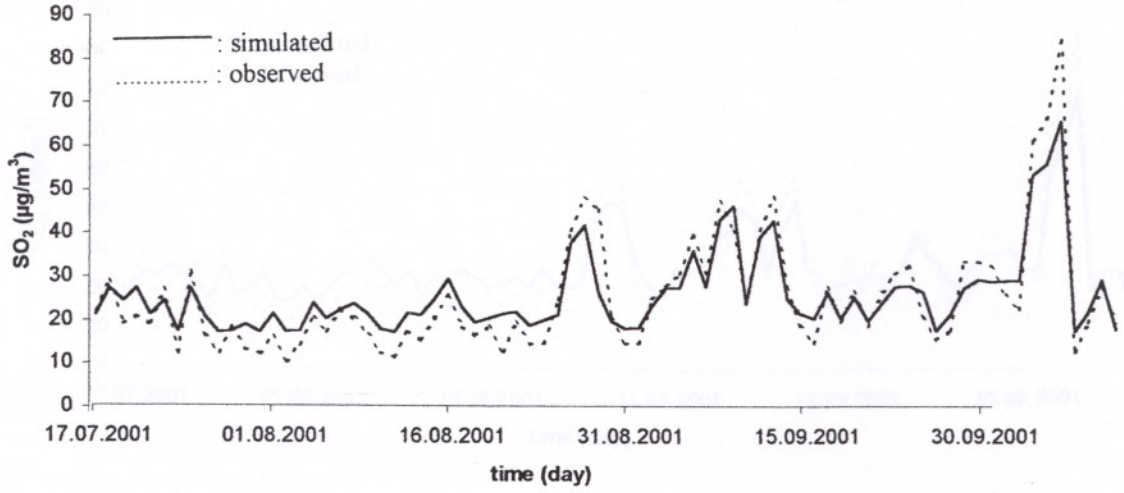


Figure A.6 SO_2 predictions with two input parameters (dust and wind speed) with two hidden layer with sigmoid function

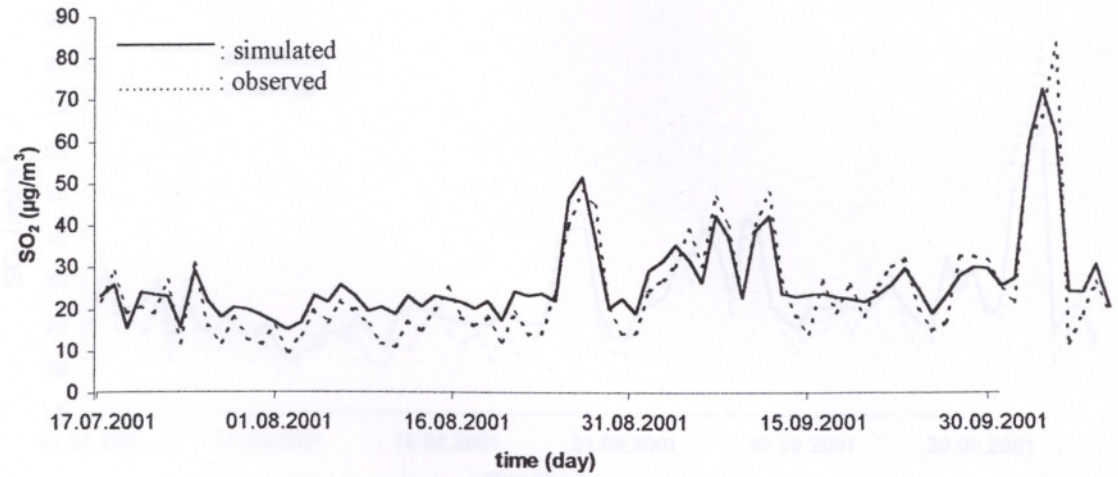


Figure A.7 SO_2 predictions with two input parameters (dust and wind speed) with one hidden layer with hyperbolic tangent function

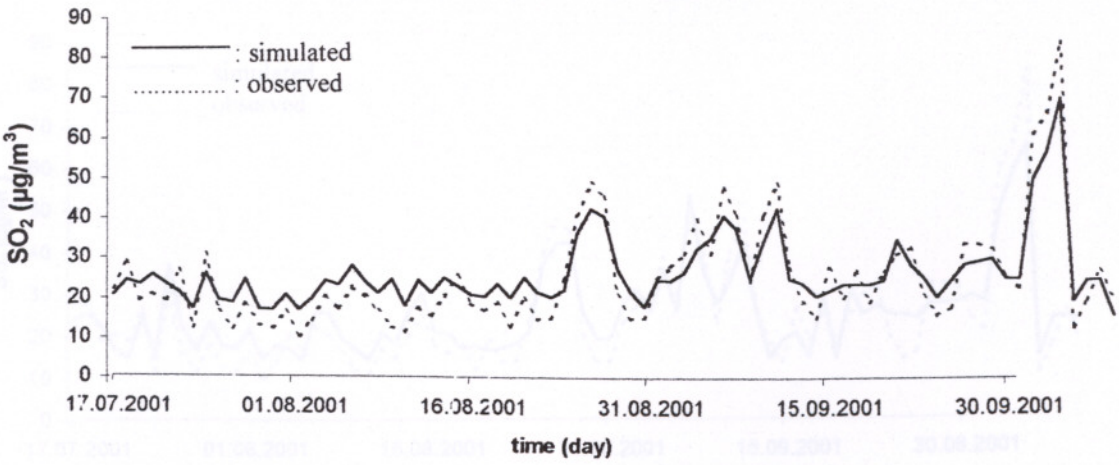


Figure A.8 SO_2 predictions with two input parameters (dust and wind speed) with one hidden layer with sigmoid function

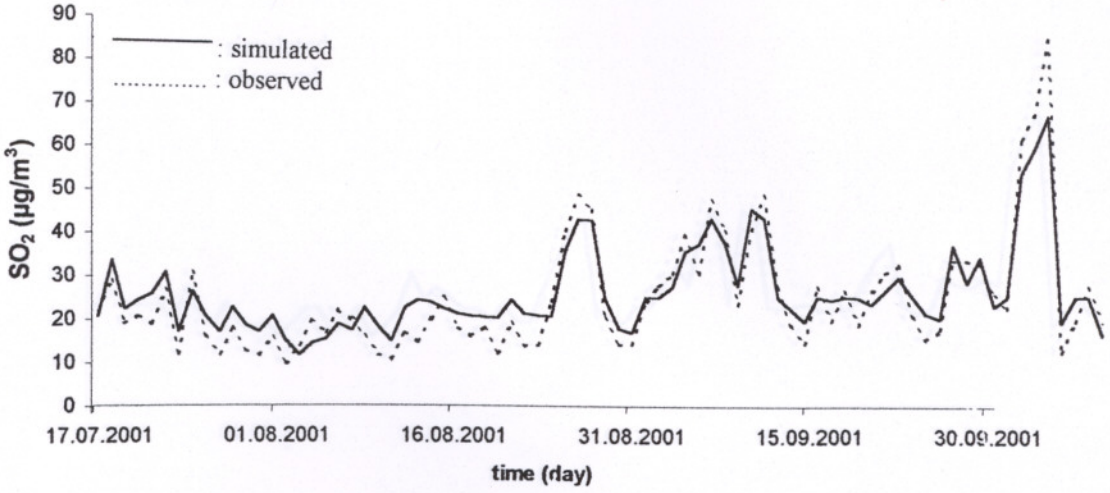


Figure A.9 SO_2 predictions with two input parameters (dust and temperature) with two hidden layer with hyperbolic tangent function

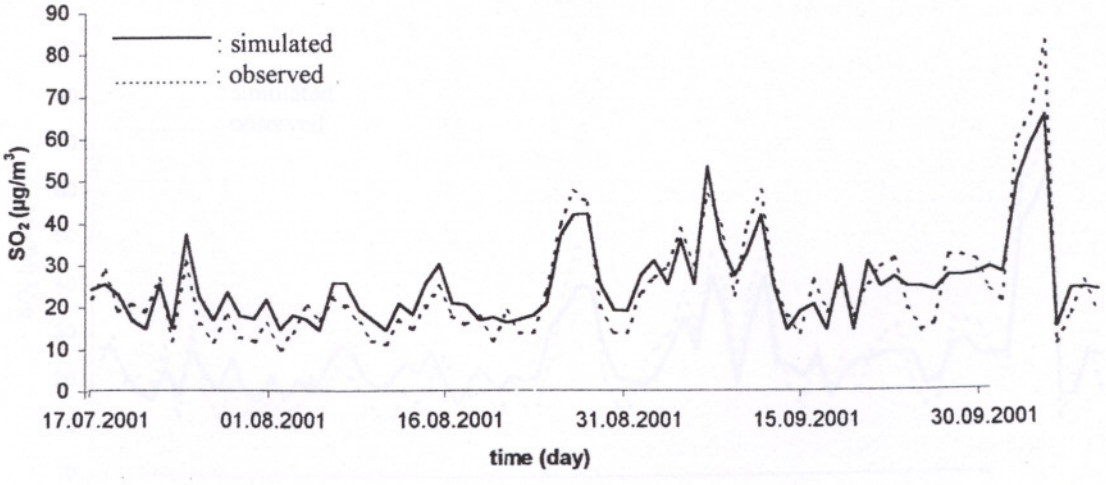


Figure A.10 SO_2 predictions with two input parameters (dust and temperature) with two hidden layer with sigmoid function

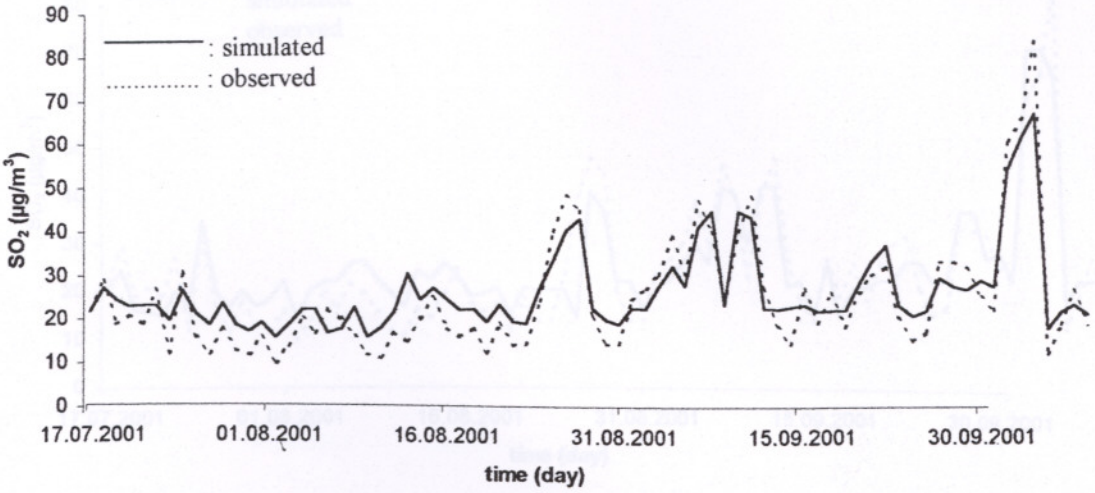


Figure A.11 SO_2 predictions with two input parameters (dust and temperature) with one hidden layer with hyperbolic tangent function

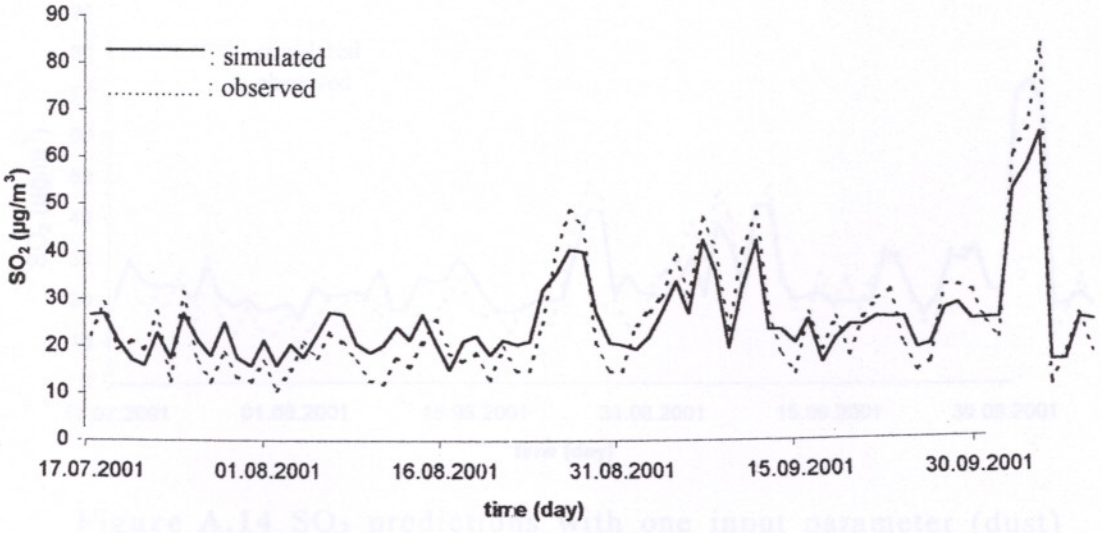


Figure A.12 SO_2 predictions with two input parameters (dust and temperature) with one hidden layer with sigmoid function

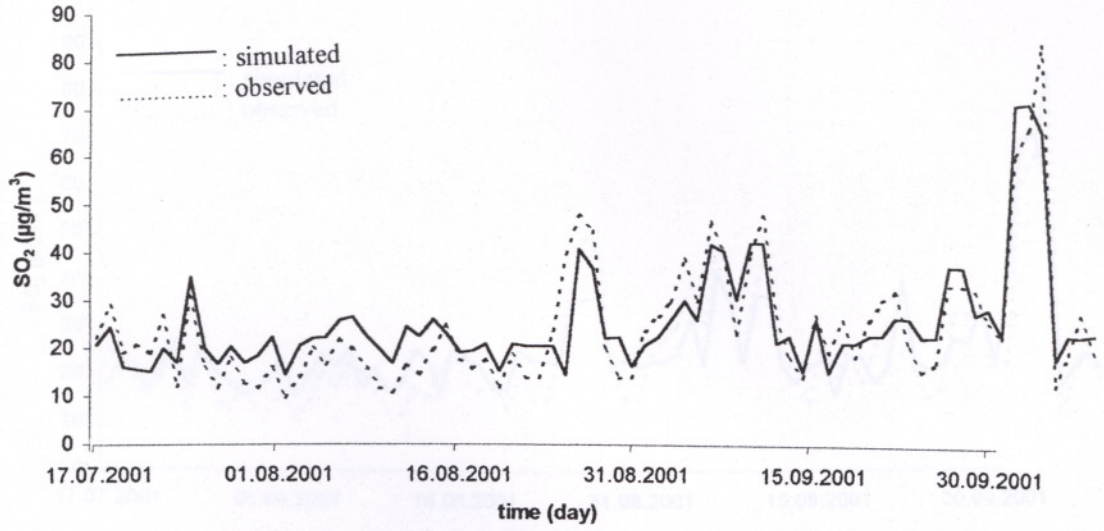


Figure A.13 SO_2 predictions with one input parameter (dust) with two hidden layer with hyperbolic tangent function

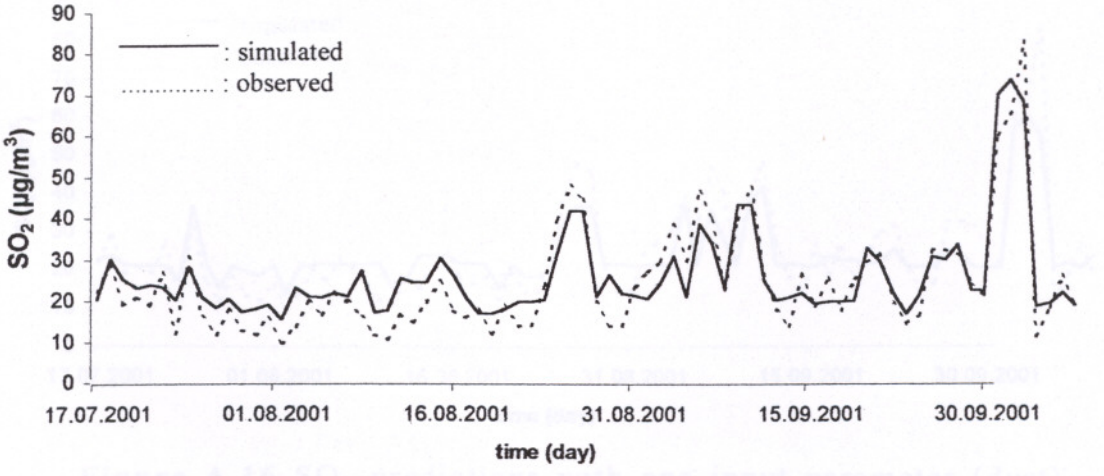


Figure A.14 SO_2 predictions with one input parameter (dust) with two hidden layer with sigmoid function

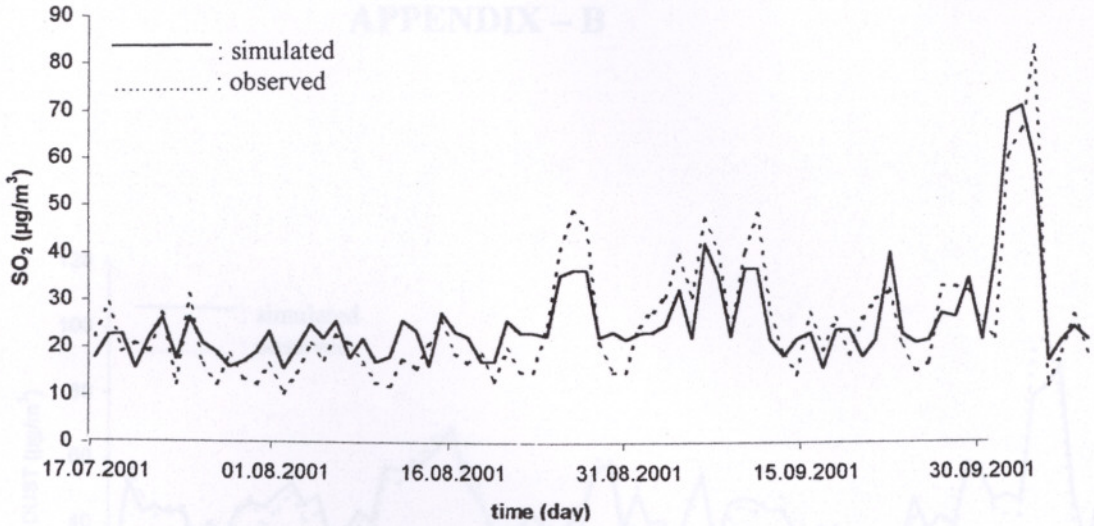


Figure A.15 SO_2 predictions with one input parameter (dust) with one hidden layer with hyperbolic tangent function

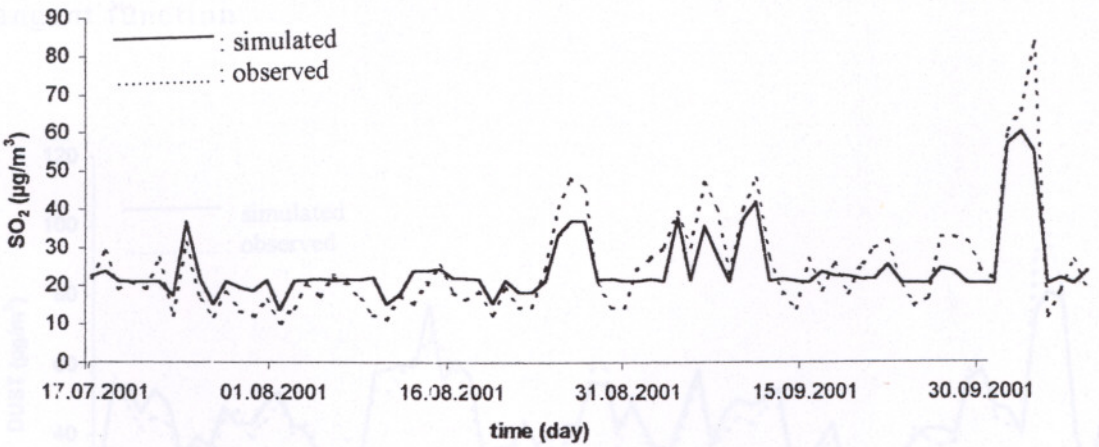


Figure A.16 SO_2 predictions with one input parameter (dust) with one hidden layer with sigmoid function

APPENDIX – B

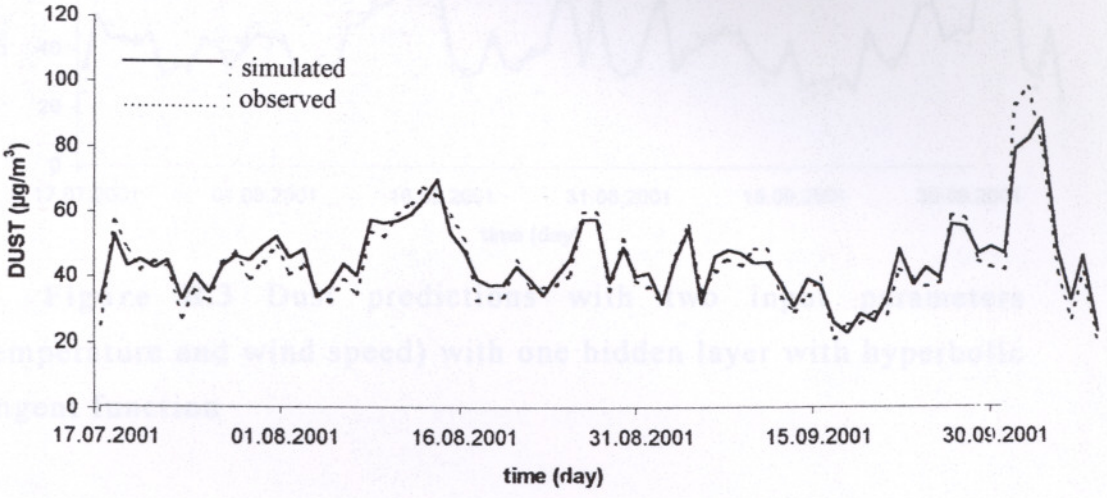


Figure B.1 Dust predictions with two input parameters (temperature and wind speed) with two hidden layer with hyperbolic tangent function

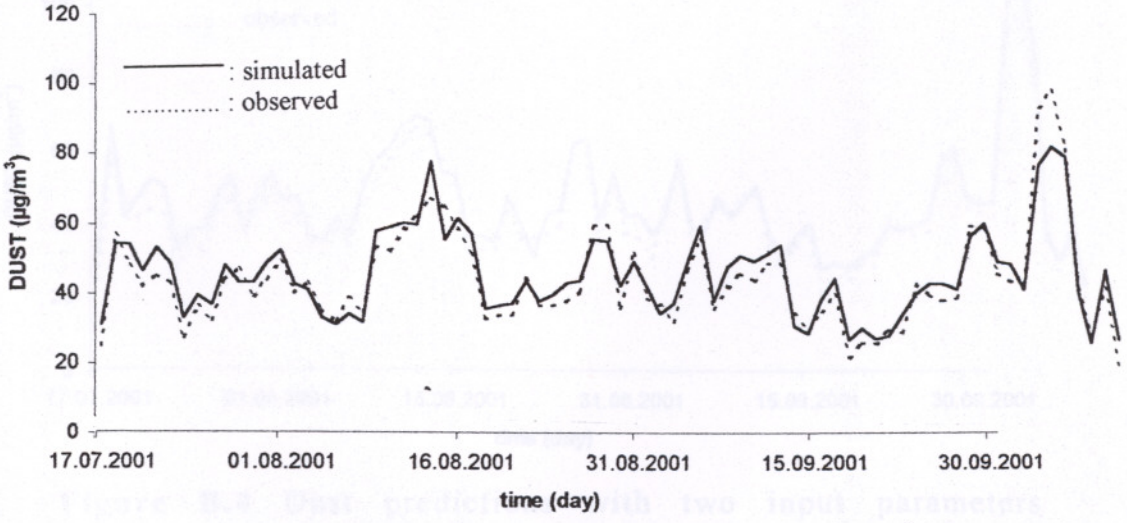


Figure B.2 Dust predictions with two input parameters (temperature and wind speed) with two hidden layer with sigmoid function

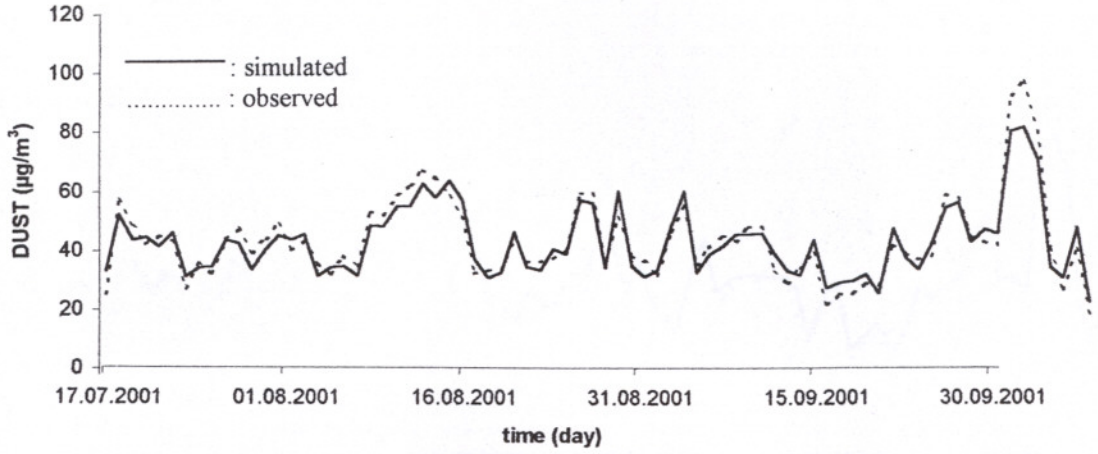


Figure B.3 Dust predictions with two input parameters (temperature and wind speed) with one hidden layer with hyperbolic tangent function

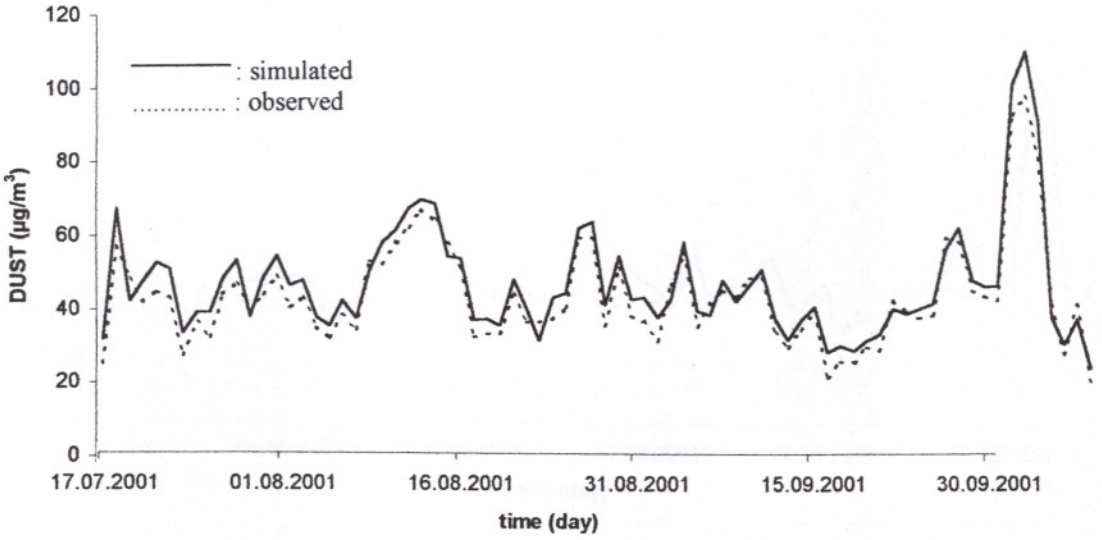


Figure B.4 Dust predictions with two input parameters (temperature and wind speed) with one hidden layer with sigmoid function

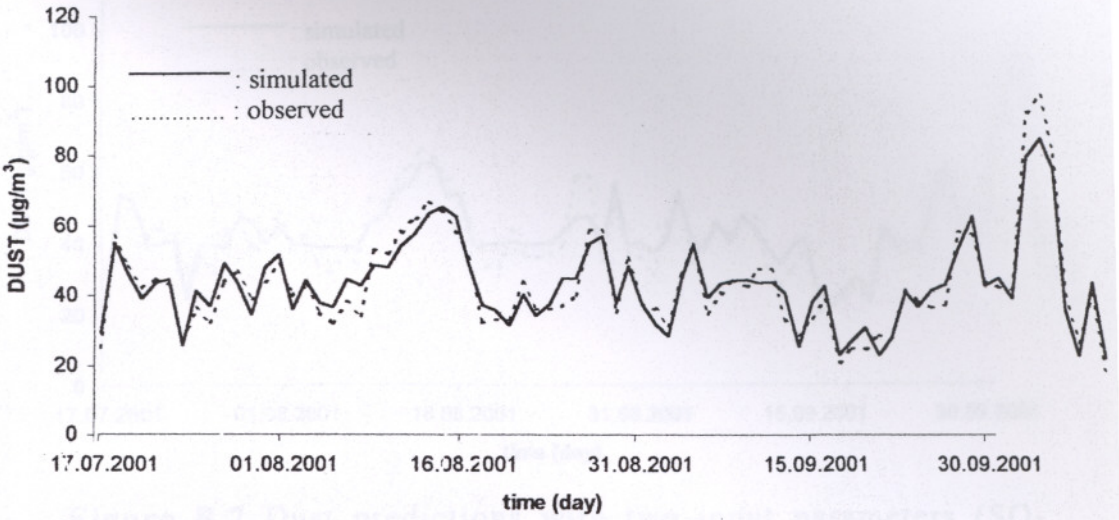


Figure B.5 Dust predictions with two input parameters (SO_2 and wind speed) with two hidden layer with hyperbolic tangent function

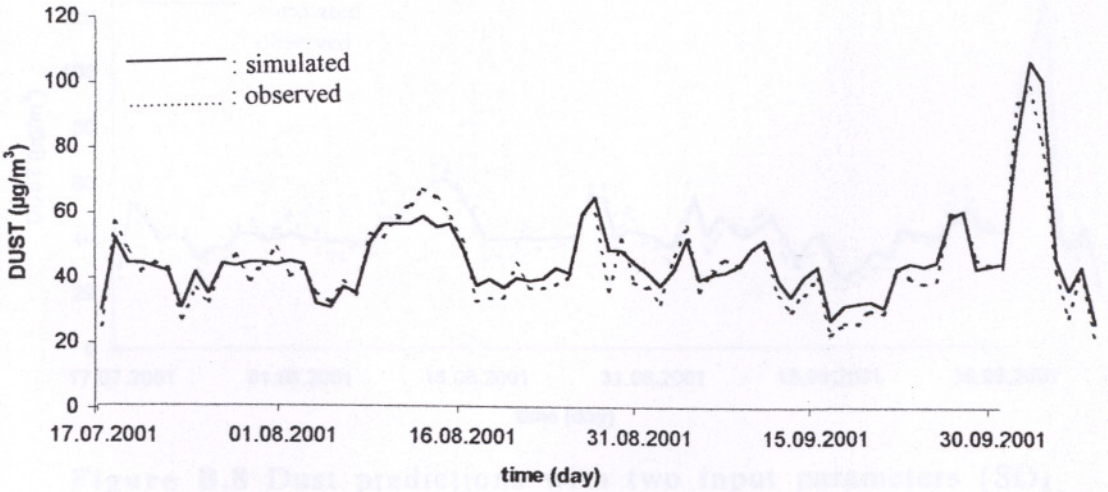


Figure B.6 Dust predictions with two input parameters (SO_2 and wind speed) with two hidden layer with sigmoid function

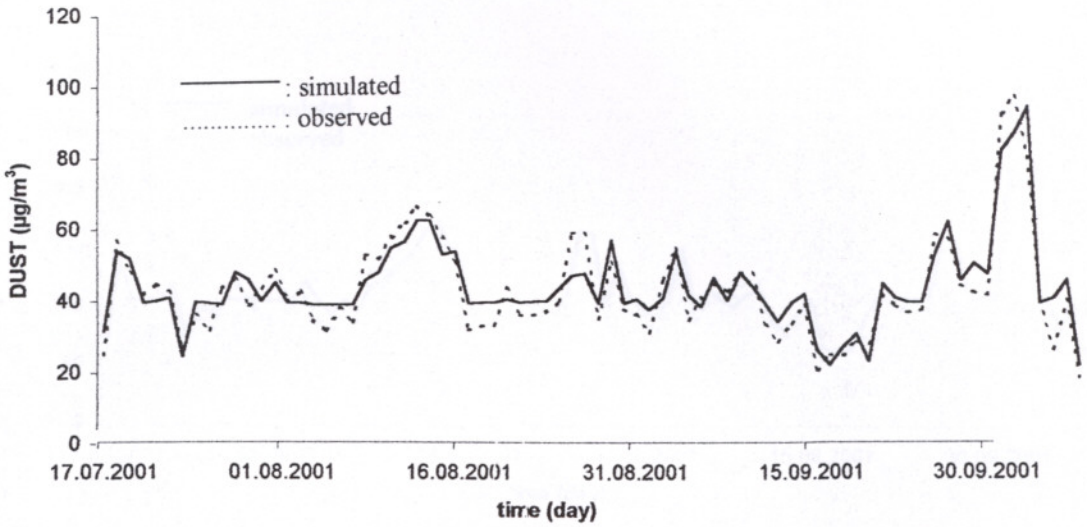


Figure B.7 Dust predictions with two input parameters (SO_2 and wind speed) with one hidden layer with hyperbolic tangent function

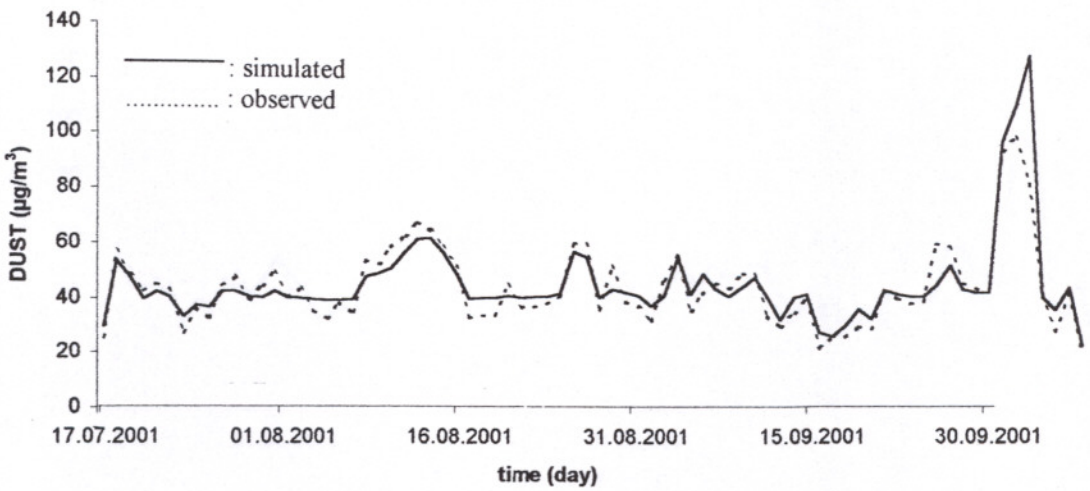


Figure B.8 Dust predictions with two input parameters (SO_2 and wind speed) with one hidden layer with sigmoid function

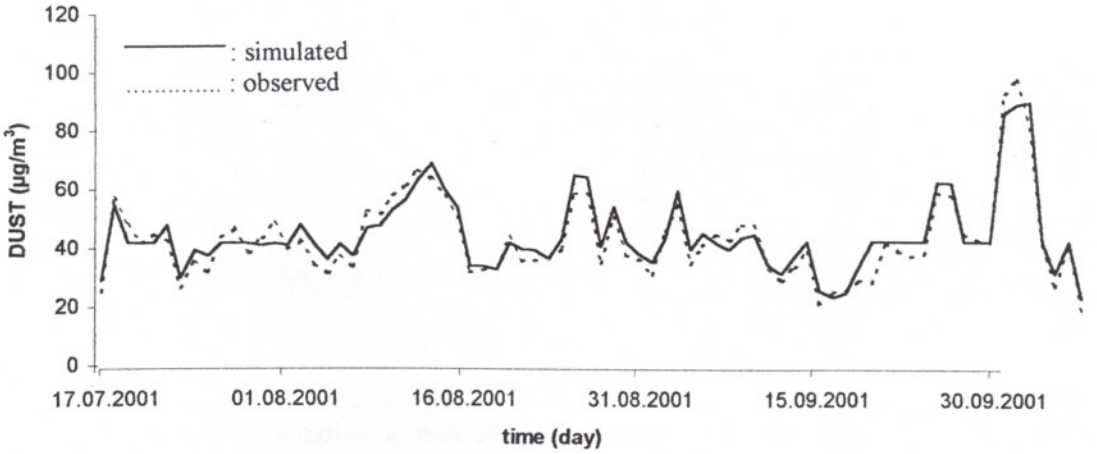


Figure B.9 Dust predictions with two input parameters (SO_2 and temperature) with two hidden layer with hyperbolic tangent function (—:predicted, ---: actual output)

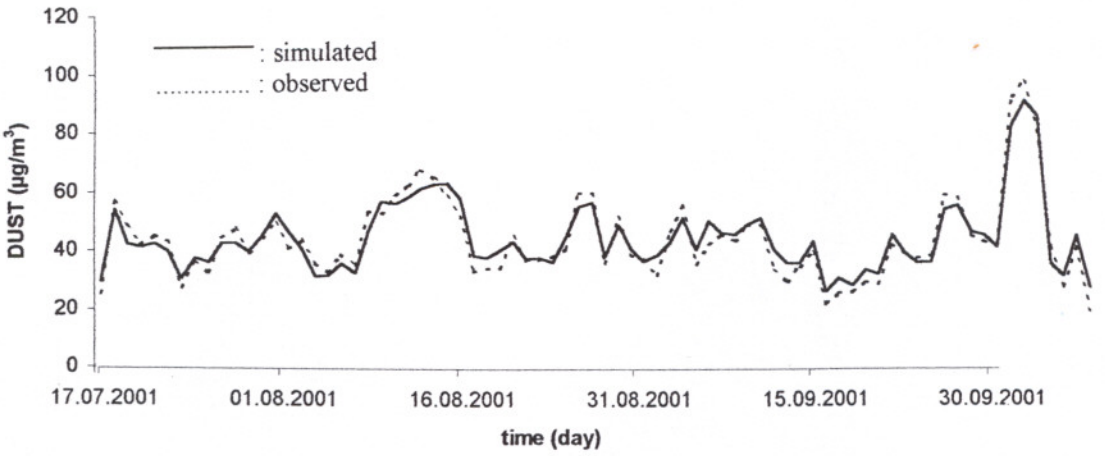


Figure B.10 Dust predictions with two input parameters (SO_2 and temperature) with two hidden layer with sigmoid function

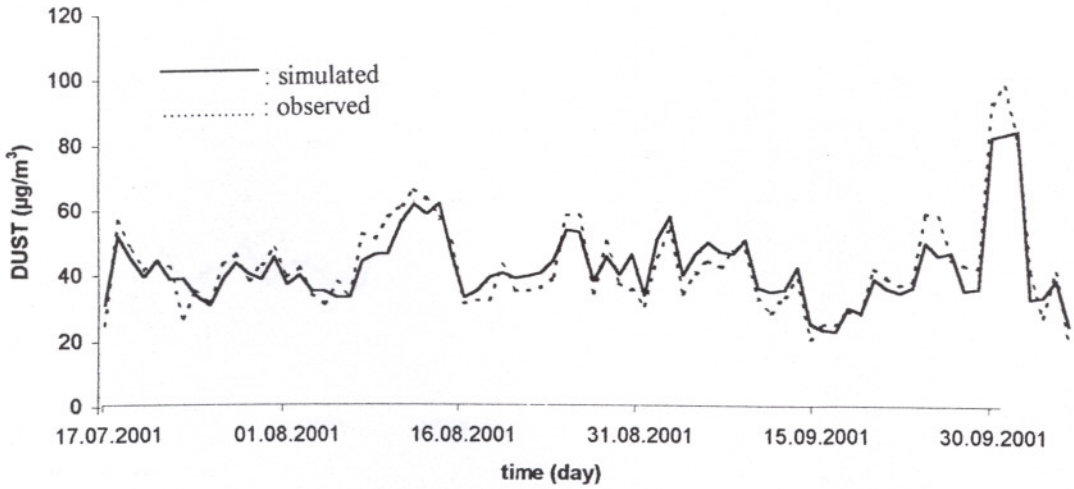


Figure B.11 Dust predictions with two input parameters (SO_2 and temperature) with one hidden layer with hyperbolic tangent function

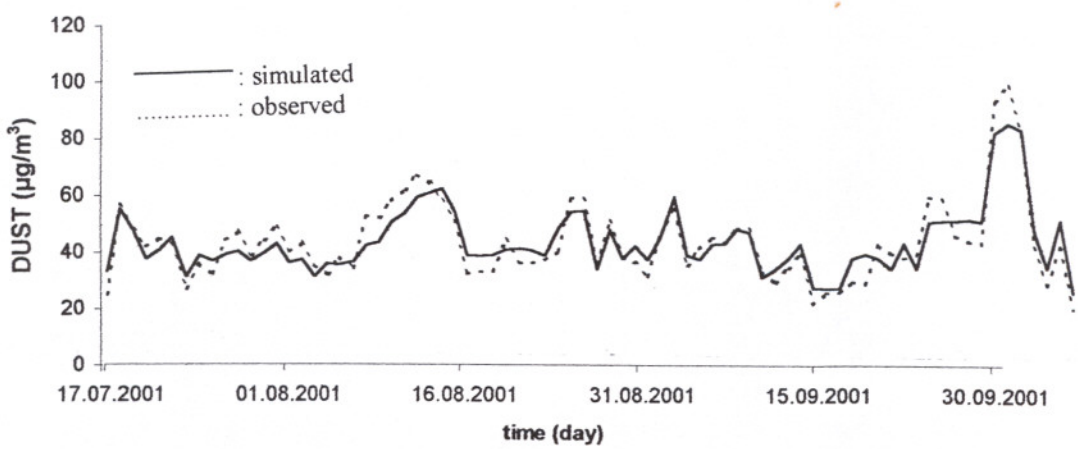


Figure B.12 Dust predictions with two input parameters (SO_2 and temperature) with one hidden layer with sigmoid function

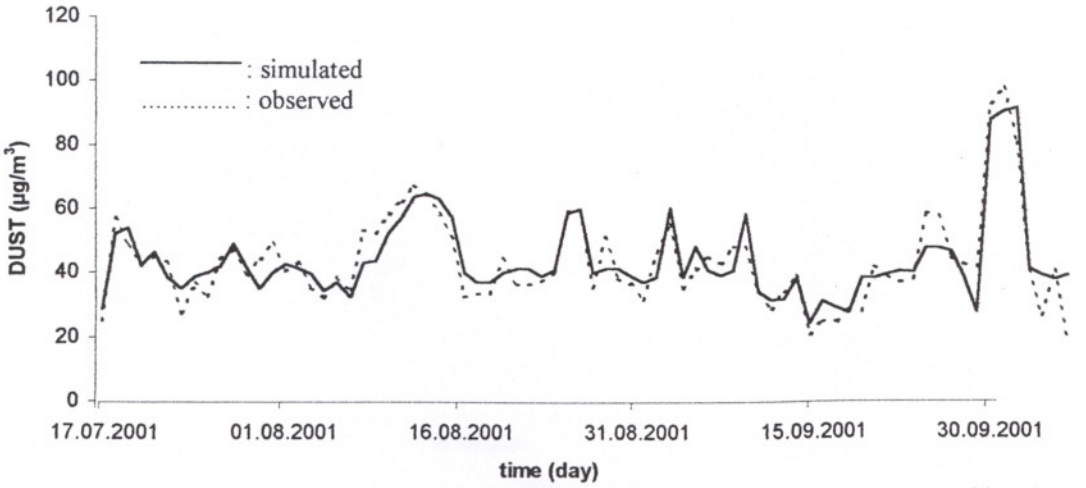


Figure B.13 Dust predictions with one input parameter (SO_2) with two hidden layer with hyperbolic tangent function

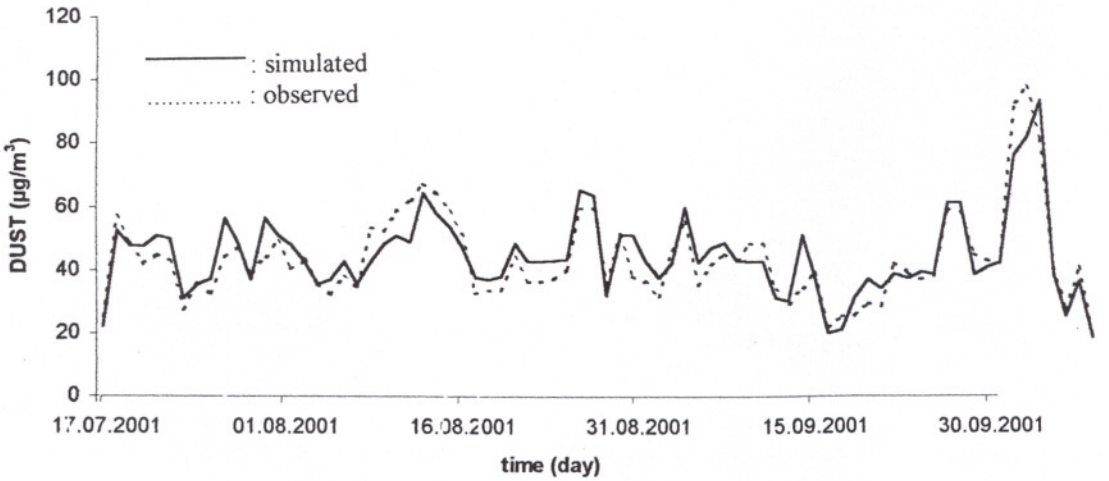


Figure B.14 Dust predictions with one input parameter (SO_2) with two hidden layer with sigmoid function

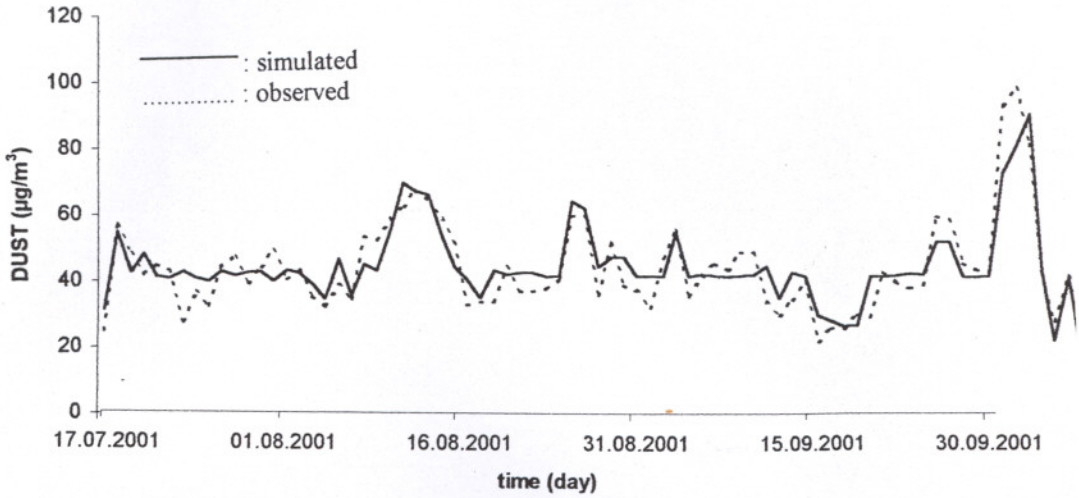


Figure B.15 Dust predictions with one input parameter (SO_2) with one hidden layer with hyperbolic tangent function

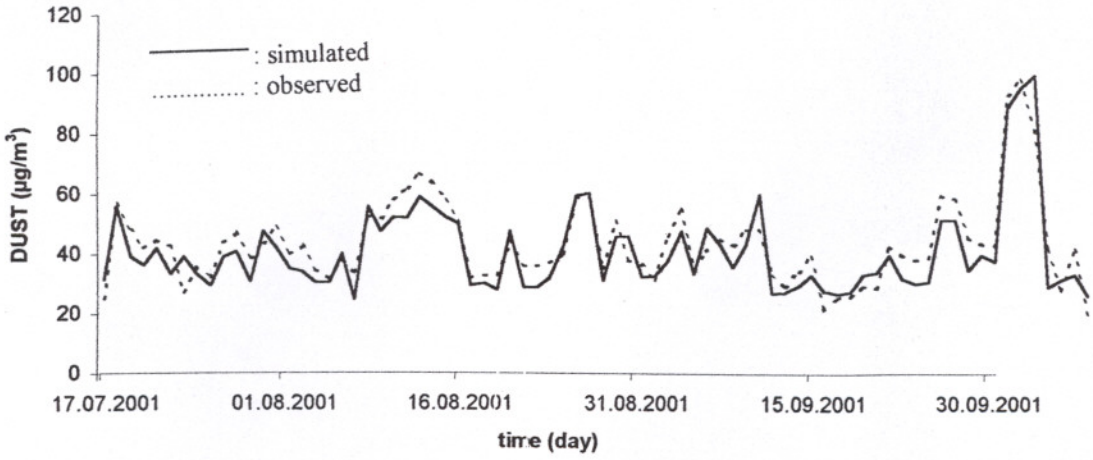


Figure B.16 Dust predictions with one input parameter (SO_2) with one hidden layer with sigmoid function